Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
CuttingEdge.Conditions Namespace
Send Feedback
## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternativeExceptionCondition</td>
<td>An instance of this type is returned from the <code>WithExceptionOnFailure&lt;</code> method overloads and allow you to specify the exception type that should be thrown on failure.</td>
</tr>
<tr>
<td>Condition</td>
<td>Entry point methods to start validating pre- and postconditions. Adam</td>
</tr>
<tr>
<td>ConditionValidator&lt;<code>&lt;T&gt;</code>&gt;</td>
<td>Enables validation of pre- and postconditions. This class isn't used directly by developers. Instead the class should be created by the <code>Requires</code> and <code>Ensures</code> extension methods.</td>
</tr>
<tr>
<td>PostconditionException</td>
<td>The exception that is thrown when a method's postcondition is not valid.</td>
</tr>
<tr>
<td>ValidatorExtensions</td>
<td>Extension methods for <code>ConditionValidator&lt;</code> method.</td>
</tr>
</tbody>
</table>
### Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConstraintViolationType</td>
<td>This enumeration is used to determine the type of exception the validator should throw.</td>
</tr>
</tbody>
</table>

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
An instance of this type is returned from the `WithExceptionOnFailure<(Of <(TException)>))()` method overloads and allow you to specify the exception type that should be thrown on failure.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)
Public MustInherit Class AlternativeExceptionCondition

C#

public abstract class AlternativeExceptionCondition

Visual C++

public ref class AlternativeExceptionCondition abstract

JavaScript

CuttingEdge.Conditions.AlternativeExceptionCondition = function();
Type.createClass(
    'CuttingEdge.Conditions.AlternativeExceptionCondition');
Inheritance Hierarchy

System::Object
CuttingEdge.Conditions::AlternativeExceptionCondition
See Also

AlternativeExceptionCondition Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
AlternativeExceptionCondition Constructor

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Protected Sub New

**C#**

protected AlternativeExceptionCondition()

**Visual C++**

protected:
AlternativeExceptionCondition()

**JavaScript**

CuttingEdge.Conditions.AlternativeExceptionCondition = function();
See Also

AlternativeExceptionCondition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `AlternativeExceptionCondition` type exposes the following members.
### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternativeExceptionCondition</td>
<td></td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified System.Object is equal to the current System.Object. (Overrids <strong>Object..::.Equals(Object).</strong>)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code of the current instance. (Overrids <strong>Object..::.GetHashCode()().</strong>)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Requires</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <strong>String</strong> that represents the <strong>AlternativeExceptionCondition</strong>. (Overrids <strong>Object..::.ToString()()</strong>.)</td>
</tr>
</tbody>
</table>
See Also

AlternativeExceptionCondition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The **AlternativeExceptionCondition** type exposes the following members.
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified System.Object is equal to the current System.Object. (Overrides <strong>Object::{:}.Equals(Object).</strong>)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code of the current instance. (Overrides <strong>Object::{:}.GetHashCode().</strong>)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Requires</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <strong>String</strong> that represents the <strong>AlternativeExceptionCondition</strong>. (Overides <strong>Object::{:}.ToString().</strong>).</td>
</tr>
</tbody>
</table>
See Also

AlternativeExceptionCondition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Determines whether the specified System.Object is equal to the current System.Object.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
<ObsoleteAttribute("This method is not part of the Conditions framework. Please use the Requires().IsEqualTo method.", True)>
Public Overrides Function Equals ( _
    obj As Object _
) As Boolean
```

#### C#

```csharp
[ObsoleteAttribute("This method is not part of the Conditions framework. Please use the Requires().IsEqualTo method.", true)]
public override bool Equals(
    Object obj
)
```

#### Visual C++

```cpp
[ObsoleteAttribute(L"This method is not part of the Conditions framework.", true)]
public: bool Equals(
    Object^ obj
) override
```

#### JavaScript

```javascript
function equals(obj);
```

### Parameters

- **obj**
  - Type: `System::Object`
  - The System.Object to compare with the current System.Object.

### Return Value
true if the specified System.Object is equal to the current System.Object; otherwise, false.
See Also

AlternativeExceptionCondition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Returns the hash code of the current instance.

**Namespace:**  [CuttingEdge.Conditions](CuttingEdge.Conditions)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Overrides Function GetHashCode As Integer

C#

public override int GetHashCode()

Visual C++

public:
virtual int GetHashCode() override

JavaScript

function getHashCode();

Return Value

The hash code of the current instance.
See Also

AlternativeExceptionCondition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Gets the `Type` of the current instance.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Function GetType As Type

C#

public Type GetType()

Visual C++

public:
Type^ GetType()

JavaScript

function getType();

Return Value

The Type instance that represents the exact runtime type of the current instance.
See Also

AlternativeExceptionCondition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

AlternativeExceptionCondition:::Requires Method

AlternativeExceptionCondition Class  See Also  Send Feedback
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Requires&lt;(Of &lt;T&gt;)(T)</code></td>
<td>Returns a new <a href="#">ConditionValidator</a> that allows you to validate the preconditions of the given argument, given it a default ArgumentName of 'value'.</td>
</tr>
<tr>
<td><code>Requires&lt;(Of &lt;T&gt;)(T, String)</code></td>
<td>Returns a new <a href="#">ConditionValidator</a> that allows you to validate the preconditions of the given argument.</td>
</tr>
</tbody>
</table>
See Also

AlternativeExceptionCondition Class
AlternativeExceptionCondition Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Returns a new ConditionValidator that allows you to validate the preconditions of the given argument, given it a default ArgumentName of 'value'.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public MustOverride Function Requires(Of T) ( _
    value As T _
) As ConditionValidator(Of T)

C#

public abstract ConditionValidator<T> Requires<T>(
    T value
)

Visual C++

public:
    generic<typename T>
    virtual ConditionValidator<T>^ Requires(
        T value
    ) abstract

JavaScript

JavaScript does not support generic types or methods.

Parameters

value
    Type: T
    The value of the argument to validate.
Type Parameters

T
The type of the argument to validate.

Return Value

A new ConditionValidator containing the value and "value" as argument name.
The following example shows how to use the Requires extension method.

```csharp
using CuttingEdge.Conditions;

public class Person
{
    private int age;

    public int Age
    {
        get { return this.age; }
        set
        {
            // Throws an InvalidOperationException when value is less than 0
            Condition.WithExceptionOnFailure<InvalidOperationException>()
                .Requires(value).IsGreaterOrEqual(0);

            this.age = value;
        }
    }
}
```

See the `ConditionValidator<Of <(T)>>` class for more code examples.
See Also

AlternativeExceptionCondition Class
Requires Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Returns a new `ConditionValidator` that allows you to validate the preconditions of the given argument.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public MustOverride Function Requires(Of T) ( _
    value As T, _
    argumentName As String _
) As ConditionValidator(Of T)

C#

public abstract ConditionValidator<T> Requires<T>(
    T value,
    string argumentName
)

Visual C++

public:
    generic<typename T>
    virtual ConditionValidator<T>^ Requires(
        T value,
        String^ argumentName
    ) abstract

JavaScript

JavaScript does not support generic types or methods.

Parameters

value
    Type: T
    The value of the argument to validate.

argumentName
    Type: System::String
    The name of the argument to validate.
Type Parameters

T
   The type of the argument to validate.

Return Value

A new ConditionValidator containing the value and argumentName.
Examples

The following example shows how to use the **Requires** extension method.

```csharp
using CuttingEdge.Conditions;

public class Point
{
    private readonly int x;
    private readonly int y;

    public Point(int x, int y)
    {
        // Throws an InvalidOperationException when x is less than 0
        Condition.WithExceptionOnFailure<InvalidOperationException>()
            .Requires(x, "x").IsGreaterOrEqual(0);

        // Throws an InvalidOperationException when y is less than 0
        Condition.WithExceptionOnFailure<InvalidOperationException>()
            .Requires(y, "y").IsGreaterOrEqual(0);

        this.x = x;
        this.y = y;
    }

    public int X { get { return this.x; } }
    public int Y { get { return this.y; } }
}
```

See the **ConditionValidator<(Of <(T)>)>** class for more code examples.
See Also

AlternativeExceptionCondition Class
Requires Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Returns a `String` that represents the `AlternativeExceptionCondition`.

**Namespace:**  `CuttingEdge.Conditions`  
**Assembly:**  `CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)`
Syntax

**Visual Basic (Declaration)**

Public Overrides Function ToString As String

**C#**

public override string ToString()

**Visual C++**

public:
virtual String^ ToString() override

**JavaScript**

function toString();

**Return Value**

A String that represents the AlternativeExceptionCondition.
See Also

AlternativeExceptionCondition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

Condition Class

**Entry point methods to start validating pre- and postconditions.**

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

Public NotInheritable Class Condition

**C#**

public static class Condition

**Visual C++**

public ref class Condition abstract sealed

**JavaScript**

CuttingEdge.Conditions.Condition = function();

Type.createClass(
    'CuttingEdge.Conditions.Condition');
Inheritance Hierarchy

System::Object
CuttingEdge.Conditions::Condition
See Also

Condition Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `Condition` type exposes the following members.
**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensures</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Requires</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>WithExceptionOnFailure&lt;Of TException&gt;</td>
<td>Returns a new <a href="#">AlternativeExceptionCondition</a> that allows you to specify the exception type that has to be thrown in case a validation fails.</td>
</tr>
</tbody>
</table>
See Also

Condition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `Condition` type exposes the following members.
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensures</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Requires</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>WithExceptionOnFailure&lt;Of&lt;TException&gt;&gt;</td>
<td>Returns a new <a href="#">AlternativeExceptionCondition</a> that allows you to specify the exception type that has to be thrown in case a validation fails.</td>
</tr>
</tbody>
</table>
See Also

Condition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Condition Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensures&lt;Of&lt;(T)&gt;)&lt;T&gt;</td>
<td>Returns a new <a href="#">ConditionValidator</a> that allows you to validate the given argument, given it a default ArgumentName of 'value'.</td>
</tr>
<tr>
<td>Ensures&lt;Of&lt;(T)&gt;)&lt;T, String&gt;</td>
<td>Returns a new <a href="#">ConditionValidator</a> that allows you to validate the postconditions of the given object.</td>
</tr>
</tbody>
</table>
See Also

Condition Class
Condition Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Returns a new `ConditionValidator` that allows you to validate the given argument, given it a default ArgumentName of 'value'.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function Ensures(Of T) ( _
  value As T _) As ConditionValidator(Of T)
)

**C#**

public static ConditionValidator<T> Ensures<T>(
  T value
)

**Visual C++**

public:
  generic<typename T>
  static ConditionValidator<T>^ Ensures(
    T value
  )

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

value
  Type: T
  The value of the argument to validate.
Type Parameters

T

The type of the argument to validate.

Return Value

A new `ConditionValidator` containing the value and "value" as argument name.
Examples

For an example of the usage of **Ensures** see the `Ensures<(Of <(T)>)>(T, String)` overload.
See Also

Condition Class
Ensures Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Returns a new `ConditionValidator` that allows you to validate the postconditions of the given object.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

**Visual Basic (Declaration)**

Public Shared Function Ensures(Of T) ( _
    value As T, _
    argumentName As String _
) As ConditionValidator(Of T)

**C#**

public static ConditionValidator&lt;T&gt; Ensures&lt;T&gt;(  
    T value,
    string argumentName
)

**Visual C++**

public:
    generic&lt;typename T&gt;
    static ConditionValidator&lt;T&gt;^ Ensures(  
        T value,
        String^ argumentName
    )

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

**value**
- Type: T
  - The object to validate.

**argumentName**
- Type: System:::String
  - The name of the argument to validate
Type Parameters

T
    The type of the object to validate.

Return Value

A new ConditionValidator containing the value and argumentName.
Examples

The following example shows a way to use the Ensures method. Shown is an IObjectBuilder interface which contract states that the BuildObject method should never return null. That contract, however, is not enforced by the compiler or the runtime. To allow this contract to be validated, the ObjectBuilderValidator class is a decorator for objects implementing the IObjectBuilder interface and it ensures that the given contract is fulfilled, by checking the return value of the called BuildObject of the wrapped IObjectBuilder.

```csharp
using CuttingEdge.Conditions;

public interface IObjectBuilder
{
    /// <summary>Builds an object.</summary>
    /// <returns>Returns a newly built object. Will not return null.</returns>
    object BuildObject();
}

public class ObjectBuilderValidator : IObjectBuilder
{
    public object BuildObject()
    {
        object obj = wrappedObjectBuilder.BuildObject();

        // When obj == null, a PostconditionException is thrown, with the following message:
        // "Postcondition 'the value returned by IObjectBuilder.BuildObject() should not be null' failed."
        Conditions.Ensures(obj, "the value returned by IObjectBuilder.BuildObject()")
            .IsNotNull();

        return obj;
    }
```
private readonly IObjectBuilder wrappedObjectBuilder;

/// <summary>
/// Initializes a new instance of the <see cref="ObjectBuilderValidator"/> class.
/// </summary>
/// <param name="objectBuilder">The object builder.</param>
/// <exception cref="ArgumentNullException">
/// Thrown when <paramref name="objectBuilder"/> is a null reference.
/// </exception>
public ObjectBuilderWrapper(IObjectBuilder objectBuilder)
{
    // Throws a ArgumentNullException when objectBuilder == null.
    Condition.Requires(objectBuilder, "objectBuilder").IsNotNull();

    this.wrappedObjectBuilder = objectBuilder;
}

See the <see cref="ConditionValidator(Of Of(T)>)"/> class for more code examples.
See Also

Condition Class
Ensures Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Condition...:::Requires Method

Condition Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires&lt;(Of &lt;T&gt;)(T)</td>
<td>Returns a new <a href="#">ConditionValidator</a> that allows you to validate the preconditions of the given argument, given it a default ArgumentName of 'value'.</td>
</tr>
<tr>
<td>Requires&lt;(Of &lt;T&gt;)(T, String)&gt;</td>
<td>Returns a new <a href="#">ConditionValidator</a> that allows you to validate the preconditions of the given argument.</td>
</tr>
</tbody>
</table>
See Also

Condition Class
Condition Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function Requires(Of T) ( _
    value As T _
) As ConditionValidator(Of T)
```

**C#**

```csharp
public static ConditionValidator<T> Requires<T>(
    T value
)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static ConditionValidator<T>^ Requires(
        T value
    )
```

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

- **value**
  - Type: T
  - The value of the argument to validate.
**Type Parameters**

T

The type of the argument to validate.

**Return Value**

A new `ConditionValidator` containing the value and "value" as argument name.
Examples

The following example shows how to use the **Requires** method.

```csharp
using CuttingEdge.Conditions;

public class Person
{
    private int age;

    public int Age
    {
        get { return this.age; }
        set
        {
            // Throws an ArgumentOutOfRangeException when value is less than 0
            Condition.Requires(value).IsGreaterOrEqual(0);
            this.age = value;
        }
    }
}
```

See the **ConditionValidator(Of(T))** class for more code examples.
See Also

Condition Class
Requires Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library
Condition...:::Requires<(Of <(T)>)> Method (T, String)

Returns a new ConditionValidator that allows you to validate the preconditions of the given argument.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function Requires(Of T) ( _
    value As T, _
    argumentName As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> Requires<T>(
    T value,
    string argumentName
)

Visual C++

public:
    generic<typename T>
    static ConditionValidator<T>^ Requires(
        T value,
        String^ argumentName
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

value
    Type: T
    The value of the argument to validate.

destination
    Type: System::String
    The destination of the argument to validate.
Type Parameters

T
The type of the argument to validate.

Return Value

A new ConditionValidator containing the value and argumentName.
Examples

The following example shows how to use the `Requires` method.

```csharp
using CuttingEdge.Conditions;

public class Point
{
    private readonly int x;
    private readonly int y;

    public Point(int x, int y)
    {
        // Throws an ArgumentOutOfRangeException when x is less than 0
        Condition.Requires(x, "x").IsGreaterOrEqual(0);

        // Throws an ArgumentOutOfRangeException when y is less than 0
        Condition.Requires(y, "y").IsGreaterOrEqual(0);

        this.x = x;
        this.y = y;
    }

    public int X { get { return this.x; } }
    public int Y { get { return this.y; } }
}

See the `ConditionValidator<Of<T>>` class for more code examples.
See Also

Condition Class
Requires Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

Condition...:::WithExceptionOnFailure<(Of <(TException)>)> Method

Returns a new AlternativeExceptionCondition that allows you to specify the exception type that has to be thrown in case a a validation fails.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function WithExceptionOnFailure(Of TException As Exception)

C#

public static AlternativeExceptionCondition WithExceptionOnFailure<TException>()
where TException : Exception

Visual C++

public:
    generic<typename TException>
    where TException : Exception
    static AlternativeExceptionCondition^ WithExceptionOnFailure()

JavaScript

JavaScript does not support generic types or methods.
### Type Parameters

TException

The type of the exception to throw.

### Return Value

A new [AlternativeExceptionCondition](https://example.com).
Examples

The following example shows how to use the `WithExceptionOnFailure` method.

```csharp
using CuttingEdge.Conditions;

public class Point
{
    private readonly int x;
    private readonly int y;

    public Point(int x, int y)
    {
        // Throws an InvalidOperationException when x is less than 0
        Condition.WithExceptionOnFailure<InvalidOperationException>(
            () => Requires(x, "x")
                .IsGreaterOrEqual(0)
                .IsLessThan(100);

        this.x = x;
        this.y = y;
    }

    public int X { get { return this.x; } }
    public int Y { get { return this.y; } }
}
```

See the `ConditionValidator<Of(T)>` class for more code examples.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the supplied <code>TException</code> is abstract or does not contain a public constructor with a single parameter of type <code>String</code>.</td>
</tr>
</tbody>
</table>
See Also

Condition Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Enables validation of pre- and postconditions. This class isn't used directly by developers. Instead the class should be created by the Requires and Ensures extension methods.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public MustInherit Class ConditionValidator(Of T)

C#

public abstract class ConditionValidator<T>

Visual C++

generic<typename T>
public ref class ConditionValidator abstract

JavaScript

JavaScript does not support generic types or methods.
Type Parameters

T
   The type of the argument to be validated
Examples

The following example shows how to use CuttingEdge.Conditions.

```csharp
using System.Collections;
using CuttingEdge.Conditions;

public class ExampleClass
{
    private enum StateType { Uninitialized = 0, Initialized };

    private StateType currentState;

    public ICollection GetData(int? id, string xml, IEnumerable col)
    {
        // Check all preconditions:
        Condition.Requires(id, "id")
            .IsNotNull() // throws ArgumentNullException on failure
            .IsInRange(1, 999) // ArgumentException on failure
            .IsNotEqualTo(128); // throws ArgumentException on failure

        Condition.Requires(xml, "xml")
            .StartsWith("<data>") // throws ArgumentException on failure
            .EndsWith("</data>"); // throws ArgumentException on failure

        Condition.Requires(col, "col")
            .IsNotNull() // throws ArgumentNullException on failure
            .IsEmpty(); // throws ArgumentException on failure

        // Do some work

        // Example: Call a method that should return a not null ICollection
        object result = BuildResults(xml, col);

        // Check all postconditions:
    }
```
// A PostconditionException will be thrown at failure.
Condition.Ensure(result, "result")
  .IsNull(
  .IsOfType(typeof(ICollection));

return result as ICollection;
}
}

The following code examples shows how to extend the library with your own 'Invariant' entry point method. The first example shows a class with an Add method that validates the class state (the class invariants) before adding the Person object to the internal array and that code should throw an InvalidOperationException.

```csharp
using CuttingEdge.Conditions;

public class Person { }

public class PersonCollection
{
  public PersonCollection(int capicity)
  {
    this.Capacity = capicity;
  }

  public void Add(Person person)
  {
    // Throws a ArgumentNullException when person == null
    Condition.Requires(person, "person").IsNotNull();

    // Throws an InvalidOperationException on failure
    Invariants.Invariant(this.Count, "Count").IsLessOrEqual(this.Capacity);

    this.AddInternal(person);
  }

  public int Count { get; private set; }
```
```csharp
  public int Capacity { get; private set; }

  private void AddInternal(Person person)
  {
    // some logic here
  }

  public bool Contains(Person person)
  {
    // some logic here
    return false;
  }
}

The following code example will show the implementation of the Invariants class.

```
public InvariantValidator(string argumentName, T value) : base(argumentName, value)
{
}

protected override void ThrowExceptionCore(string condition, string additionalMessage, ConstraintViolationType type)
{
    string exceptionMessage = string.Format("Invariant '{0}' failed.", condition);

    if (!String.IsNullOrEmpty(additionalMessage))
    {
        exceptionMessage += " "+ additionalMessage;
    }

    // Optionally, the 'type' parameter can be used, but never throw an
    // exception when the value of 'type' is unknown or unvalid.
    throw new InvalidOperationException(exceptionMessage);
}
}
Inheritance Hierarchy

System..:::Object
  CuttingEdge.Conditions..:::ConditionValidator<Of <(T)>>>>
See Also

ConditionValidator(Of T) Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Initializes a new instance of the **ConditionValidator(Of <T>*) class.

**Namespace:**  CuttingEdge.Conditions  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Protected Sub New ( _
    argumentName As String, _
    value As T _
)

C#

protected ConditionValidator(
    string argumentName,
    T value
)

Visual C++

protected:
ConditionValidator(
    String^ argumentName,
    T value
)

JavaScript

CuttingEdge.Conditions.ConditionValidator = function(argumentName, \n
Parameters

argumentName
    Type: System::String
    The name of the argument to be validated

value
    Type: T
    The value of the argument to be validated
See Also

ConditionValidator(Of T) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `ConditionValidator<Of <(T)>>` type exposes the following members.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Gets the value of the argument.</td>
</tr>
</tbody>
</table>
Gets the value of the argument.

**Namespace:** [CuttingEdge.Conditions](CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public ReadOnly Value As T

C#

public readonly T Value

Visual C++

public:
    initonly T Value

JavaScript

value
See Also

ConditionValidator(Of (Of T)>) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The **ConditionValidator**<**(Of** `<T>`**)) type exposes the following members.
### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ConditionValidator(Of&lt;T&gt;&gt;&lt;(T)&gt;)</code></td>
<td>Initializes a new instance of the <code>ConditionValidator(Of&lt;T&gt;&gt;&lt;(T)&gt;)</code> class.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified System.Object is equal to the current System.Object. (Overrides <strong>Object.::Equals(Object).</strong>)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code of the current instance. (Overrides <strong>Object.::GetHashCode()</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>ThrowException</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>ThrowExceptionCore</strong></td>
<td>Throws an exception.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <strong>String</strong> that represents the <strong>ConditionValidator&lt;(Of &lt;(T)&gt;)&gt;</strong>. (Overrides <strong>Object.::ToString()</strong>.)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Value</td>
<td>Gets the value of the argument.</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ArgumentName</strong></td>
<td>Gets the name of the argument.</td>
</tr>
</tbody>
</table>
See Also

ConditionValidator(Of (T)>) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `ConditionValidator<Of<T>>` type exposes the following members.
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified System.Object is equal to the current System.Object. (Overrides <code>Object::Equals(Object)</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code of the current instance. (Overrides <code>Object::GetHashCode()</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>ThrowException</strong></td>
<td>Overloaded.</td>
</tr>
<tr>
<td><strong>ThrowExceptionCore</strong></td>
<td>Throws an exception.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <code>String</code> that represents the <code>ConditionValidator&lt;Of &lt;(T)&gt;&gt;</code>. (Overrides <code>Object::ToString()</code>.)</td>
</tr>
</tbody>
</table>
See Also

ConditionValidator(Of (T)) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Determines whether the specified System.Object is equal to the current System.Object.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

```vbnet
<ObsoleteAttribute("This method is not part of the conditions framework.
True")>
Public Overrides Function Equals ( _
    obj As Object _
) As Boolean
```

C#

```csharp
[ObsoleteAttribute("This method is not part of the conditions framework.
true)]
public override bool Equals(
    Object obj
)
```

Visual C++

```cpp
[ObsoleteAttribute(L"This method is not part of the conditions framework.
true)]
public:
    virtual bool Equals(
        Object^ obj
    ) override
```

JavaScript

```javascript
function equals(obj);
```

Parameters

obj

Type: System::Object

The System.Object to compare with the current System.Object.

Return Value
true if the specified System.Object is equal to the current System.Object; otherwise, false.
See Also

[ConditionValidator(Of(Of(T)>)> Class](Overview)

[CuttingEdge.Conditions Namespace](Overview)

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Returns the hash code of the current instance.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Overrides Function GetHashCode As Integer

**C#**

public override int GetHashCode()

**Visual C++**

public:
virtual int GetHashCode() override

**JavaScript**

function getHashCode();

**Return Value**

The hash code of the current instance.
See Also

ConditionValidator(Of (T)) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ConditionValidator(Of (Of T)>).::GetType Method

ConditionValidator(Of (Of T)> Class  See Also  Send Feedback

Gets the Type of the current instance.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Function GetType As Type

**C#**

public Type GetType()

**Visual C++**

public:  
Type^ GetType()

**JavaScript**

function getType();

### Return Value

The Type instance that represents the exact runtime type of the current instance.
See Also

ConditionValidator(Of (T)) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ConditionValidator<Of <(T)>>).::.ThrowException Method

ConditionValidator<Of <(T)>> Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThrowException(String)</td>
<td>Throws an exception.</td>
</tr>
<tr>
<td>ThrowException(String, String, ConstraintViolationType)</td>
<td>Throws an exception.</td>
</tr>
</tbody>
</table>
See Also

ConditionValidator(Of T) Class
ConditionValidator(Of T) Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ConditionValidator<Of <(T)>>.:::ThrowException Method (String)

Throws an exception.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Sub ThrowException ( 
    condition As String _
)
```

**C#**

```csharp
public void ThrowException(
    string condition
)
```

**Visual C++**

```cpp
public:
void ThrowException(
    String^ condition
)
```

**JavaScript**

```javascript
function throwException(condition);
```

### Parameters

**condition**

Type: `System::String`

Describes the condition that doesn't hold, e.g., "Value should not be null".
See Also

ConditionValidator(Of (T)>) Class
ThrowException Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ConditionValidator(Of <T>)::.::ThrowException Method (String, String, ConstraintViolationType)

Throws an exception.

**Namespace:** [CuttingEdge.Conditions](https://example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Sub ThrowException (_
    condition As String, _
    additionalMessage As String, _
    type As ConstraintViolationType _
)

C#

public void ThrowException(
    string condition,
    string additionalMessage,
    ConstraintViolationType type
)

Visual C++

public:
void ThrowException(
    String^ condition,
    String^ additionalMessage,
    ConstraintViolationType type
)

JavaScript

function throwException(condition, additionalMessage, type);

Parameters

condition
    Type: System::String
    Describes the condition that doesn't hold, e.g., "Value should not be null".

additionalMessage
    Type: System::String
An additional message that will be appended to the exception message, e.g. "The actual value is 3.". This value may be null or empty.

type
Type: **CuttingEdge.Conditions..:::ConstraintViolationType**
Gives extra information on the exception type that must be build. The actual implementation of the validator may ignore some or all values.
See Also

ConditionValidator(Of (T)) Class
ThrownException Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Throws an exception.

**Namespace:** [CuttingEdge.Conditions](https://www.example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Protected MustOverride Sub ThrowExceptionCore ( _
    condition As String, _
    additionalMessage As String, _
    type As ConstraintViolationType _
)

C#

protected abstract void ThrowExceptionCore(  
    string condition,  
    string additionalMessage,  
    ConstraintViolationType type
)

Visual C++

protected:
virtual void ThrowExceptionCore(  
    String^ condition,  
    String^ additionalMessage,  
    ConstraintViolationType type  
) abstract

JavaScript

function throwExceptionCore(condition, additionalMessage, type);
An additional message that will be appended to the exception message, e.g. "The actual value is 3.". This value may be null or empty.

type

Type: CuttingEdge.Conditions.ConstraintViolationType
Gives extra information on the exception type that must be build. The actual implementation of the validator may ignore some or all values.
Remarks

Implement this method when deriving from `ConditionValidator<Of <(T)>).` The implementation should at least build the exception message from the condition and optional `additionalMessage`. Usage of the type is completely optional, but the implementation should at least be flexible and be able to handle unknown `ConstraintViolationType` values. Values may be added in future releases.
Examples

For an example see the documentation for `ConditionValidator<(Of <(T)>)`. 
See Also

ConditionValidator<Of<(T)>>) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Returns a **String** that represents the **ConditionValidator(Of (T)>)**.

**Namespace:**  **CuttingEdge.Conditions**

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)
Public Overrides Function ToString As String

C#
public override string ToString()

Visual C++
public:
virtual String^ ToString() override

JavaScript
function toString();

Return Value
A String that represents the ConditionValidator(Of T).
See Also

ConditionValidator(Of (T)) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `ConditionValidator<
(Of <(T)>)>` type exposes the following members.
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentName</td>
<td>Gets the name of the argument.</td>
</tr>
</tbody>
</table>
See Also

ConditionValidator(Of (T)) Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ConditionValidator(Of <T>)(Of <T>).::ArgumentName Property

Gets the name of the argument.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public ReadOnly Property ArgumentName As String

C#

public string ArgumentName { get; }

Visual C++

public:
property String^ ArgumentName {
    String^ get();
}

JavaScript

function get_argumentName();
See Also

`ConditionValidator<(Of <(T)>)> Class`  
`CuttingEdge.Conditions Namespace`

Send [feedback](#) on this topic to Microsoft.
This enumeration is used to determine the type of exception the validator should throw.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Enumeration ConstraintViolationType

C#

public enum ConstraintViolationType

Visual C++

public enum class ConstraintViolationType

JavaScript

CuttingEdge.Conditions.ConstraintViolationType = function();
CuttingEdge.Conditions.ConstraintViolationType.createEnum('CuttingEdc
## Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Lets the Validator to throw the default exception for that instance.</td>
</tr>
<tr>
<td>OutOfRangeViolation</td>
<td>Lets the Validator optionally throw an exception type appropriate for values that are out of range.</td>
</tr>
<tr>
<td>InvalidEnumViolation</td>
<td>Lets the Validator optionally throw an InvalidEnumArgumentException.</td>
</tr>
</tbody>
</table>

[InvalidEnumArgumentException](https://docs.microsoft.com/en-us/dotnet/api/system.invalidenumargumentexception)
See Also

CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The exception that is thrown when a method's postcondition is not valid.

**Namespace:**  [CuttingEdge.Conditions](https://github.com/CuttingEdge/Conditions)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

```vbnet
<SerializableAttribute> _
Public NotInheritable Class PostconditionException _
   Inherits Exception
```

C#

```csharp
[SerializableAttribute]
public sealed class PostconditionException : Exception
```

Visual C++

```cpp
[SerializableAttribute]
public ref class PostconditionException sealed : public Exception
```

JavaScript

```javascript
CuttingEdge.Conditions.PostconditionException = function();
Type.createClass(
   'CuttingEdge.Conditions.PostconditionException',
   Exception);
```
Inheritance Hierarchy

System..:::Object
System..:::Exception
CuttingEdge.Conditions..:::.PostconditionException
See Also

PostconditionException Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

PostconditionException Constructor

PostconditionException Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PostconditionException()</code></td>
<td>Initializes a new instance of the <code>PostconditionException</code> class.</td>
</tr>
<tr>
<td><code>PostconditionException(String)</code></td>
<td>Initializes a new instance of the <code>PostconditionException</code> class with a specified error message.</td>
</tr>
<tr>
<td><code>PostconditionException(String, Exception)</code></td>
<td>Initializes a new instance of the <code>PostconditionException</code> class.</td>
</tr>
</tbody>
</table>
See Also

PostconditionException Class
PostconditionException Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Initializes a new instance of the PostconditionException class.

**Namespace:**  CuttingEdge.Conditions  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Sub New

C#

public PostconditionException()

Visual C++

public:
PostconditionException()

JavaScript

CuttingEdge.Conditions.PostconditionException = function();
See Also

PostconditionException Class
PostconditionException Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Initializes a new instance of the **PostconditionException** class with a specified error message.

**Namespace:**  CuttingEdge.Conditions  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Sub New (_
    message As String _
)

C#

public PostconditionException(
    string message
)

Visual C++

public:
PostconditionException(
    String^ message
)

JavaScript

CuttingEdge.Conditions.PostconditionException = function(message);

Parameters

message
    Type: System::String
    The message that describes the error.
See Also

PostconditionException Class
PostconditionException Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
PostconditionException Constructor (String, Exception)

Initializes a new instance of the `PostconditionException` class.

**Namespace:** [CuttingEdge.Conditions](https://example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Sub New ( _
    message As String, _
    inner As Exception _
)
```

**C#**

```csharp
public PostconditionException(
    string message,
    Exception inner
)
```

**Visual C++**

```cpp
public:
PostconditionException(
    String^ message,
    Exception^ inner
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.PostconditionException = function(message, ir
```

**Parameters**

**message**

Type: `System::::String`

The message that describes the error.

**inner**

Type: `System::::Exception`

The exception that is the cause of the current exception, or a null reference (Nothing in Visual Basic) if no inner exception is specified.
See Also

PostconditionException Class
PostconditionException Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `PostconditionException` type exposes the following members.
## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostconditionException</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Equals**          | Determines whether the specified [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object) is equal to the current [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object).
|                     | (Inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object).)                                                                                                                      |
| **Finalize**        | Allows a [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object) to attempt to free resources and perform other cleanup operations before the [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object) is reclaimed by garbage collection.
|                     | (Inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object).)                                                                                                                      |
| **GetBaseException**| When overridden in a derived class, returns the [Exception](https://docs.microsoft.com/en-us/dotnet/api/system.exception) that is the root cause of one or more subsequent exceptions.
|                     | (Inherited from [Exception](https://docs.microsoft.com/en-us/dotnet/api/system.exception).)                                                                                                                  |
| **GetHashCode**     | Serves as a hash function for a particular type. [GetHashCode()](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) is suitable for use in hashing algorithms and data structures like a hash table.
|                     | (Inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object).)                                                                                                                      |
| **GetObjectData**   | When overridden in a derived class, sets the [SerializationInfo](https://docs.microsoft.com/en-us/dotnet/api/system.serialization.serializationinfo) with information about the exception.
|                     | (Inherited from [Exception](https://docs.microsoft.com/en-us/dotnet/api/system.exception).)                                                                                                                  |
| **GetType**         | Gets the runtime type of the current instance. (Inherited from [Exception](https://docs.microsoft.com/en-us/dotnet/api/system.exception).)                                                                 |
| **MemberwiseClone** | Creates a shallow copy of the current [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object). (Inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object).)                              |
| **ToString**        | Creates and returns a string representation of the current exception. (Inherited from [Exception](https://docs.microsoft.com/en-us/dotnet/api/system.exception).)                                               |
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>Gets a collection of key/value pairs that provide additional, user-defined information about the exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>HelpLink</strong></td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>HResult</strong></td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>Gets the Exception instance that caused the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>Gets a message that describes the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>StackTrace</strong></td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>TargetSite</strong></td>
<td>Gets the method that throws the current exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>
See Also

PostconditionException Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The **PostconditionException** type exposes the following members.
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a> is equal to the current <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a>. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a> is reclaimed by garbage collection. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>When overridden in a derived class, returns the <a href="https://learn.microsoft.com/de-de/dotnet/api/system.exception">Exception</a> that is the root cause of one or more subsequent exceptions. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.exception">Exception</a>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object.gethashcode">GetHashCode()</a> is suitable for use in hashing algorithms and data structures like a hash table. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>GetObjectData</strong></td>
<td>When overridden in a derived class, sets the <a href="https://learn.microsoft.com/de-de/dotnet/api/system.io.serialization.serializationinfo">SerializationInfo</a> with information about the exception. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.exception">Exception</a>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the runtime type of the current instance. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.exception">Exception</a>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a>. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.object">Object</a>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Creates and returns a string representation of the current exception. (Inherited from <a href="https://learn.microsoft.com/de-de/dotnet/api/system.exception">Exception</a>.)</td>
</tr>
</tbody>
</table>
See Also

PostconditionException Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The PostconditionException type exposes the following members.
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>Gets a collection of key/value pairs that provide additional, user-defined information about the exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>HelpLink</strong></td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>HResult</strong></td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>Gets the Exception instance that caused the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>Gets a message that describes the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>StackTrace</strong></td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>TargetSite</strong></td>
<td>Gets the method that throws the current exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>
See Also

PostconditionException Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Extension methods for `ConditionValidator<Of <(T)>>`.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)
Public NotInheritable Class ValidatorExtensions

C#
public static class ValidatorExtensions

Visual C++
public ref class ValidatorExtensions abstract sealed

JavaScript
CuttingEdge.Conditions.ValidatorExtensions = function();
Type.createClass(
    'CuttingEdge.Conditions.ValidatorExtensions');
Inheritance Hierarchy

System::Object
CuttingEdge.Conditions::ValidatorExtensions
See Also

ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions Members

ValidatorExtensions Class  Methods  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>ContainsAll</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>ContainsAny</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotContain</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotContainAll</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotContainAny</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotEndWith</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotHaveLength</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotStartWith</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>EndsWith</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Evaluate</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>HasLength</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsEqualTo</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsFalse</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotGreaterOrEqual</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotGreaterThan</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotInRange</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotInfinity</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotLessOrEqual</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotLessThan</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotLongerOrEqual</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotLongerThan</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNaN</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNegativeInfinity</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotEmpty</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotEqualTo</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotGreaterOrEqual</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
 IsNotGreaterThan  Overloaded.
 IsNotInfinity    Overloaded.
 IsNotInRange     Overloaded.
 IsNotLessOrEqual Overloaded.
 IsNotLessThan    Overloaded.
 IsNotLongerOrEqual Overloaded.
 IsNotLongerThan  Overloaded.
 IsNotNaN         Overloaded.
 IsNotNegativeInfinity Overloaded.
 IsNotNull        Overloaded.
 IsNotNullOrEmpty Overloaded.
 IsNotNullOrWhiteSpace Overloaded.
 IsNotOfType      Overloaded.
 IsNotPositiveInfinity Overloaded.
 IsNotShorterOrEqual Overloaded.
 IsNotShorterThan  Overloaded.
 IsNull            Overloaded.
 IsnullOrEmpty     Overloaded.
 IsNullOrWhiteSpace Overloaded.
 IsOfType          Overloaded.
 IsPositiveInfinity Overloaded.
 IsShorterOrEqual  Overloaded.
 IsShorterThan     Overloaded.
 IsTrue            Overloaded.
 StartsWith        Overloaded.
See Also

ValidatorExtensions Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions Methods

[ValidatorExtensions Class]  [See Also]  [Send Feedback]
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>ContainsAll</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>ContainsAny</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotContain</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotContainAll</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotContainAny</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotEndWith</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotHaveLength</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>DoesNotStartWith</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>EndsWith</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Evaluate</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>HasLength</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsEqualTo</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsFalse</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsGreaterOrEqual</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsGreaterThan</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsInfinity</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsInRange</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsLessOrEqual</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsLessThan</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsLongerOrEqual</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsLongerThan</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNaN</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNegativeInfinity</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotEmpty</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotEqualTo</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>IsNotGreaterOrEqual</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
- IsNotGreaterThan Overloaded.
- IsNotInfinity Overloaded.
- IsNotInRange Overloaded.
- IsNotLessOrEqual Overloaded.
- IsNotLessThan Overloaded.
- IsNot LongerOrEqual Overloaded.
- IsNot LongerThan Overloaded.
- IsNotNaN Overloaded.
- IsNotNegativeInfinity Overloaded.
- IsNull Overloaded.
- IsNullOrEmpty Overloaded.
- IsNullOrWhiteSpace Overloaded.
- IsNotOfType Overloaded.
- IsNotPositiveInfinity Overloaded.
- IsNotShorterOrEqual Overloaded.
- IsNotShorterThan Overloaded.
- IsNull Overloaded.
- IsNullOrEmpty Overloaded.
- IsNullOrWhiteSpace Overloaded.
- IsOfType Overloaded.
- IsPositiveInfinity Overloaded.
- IsShorterOrEqual Overloaded.
- IsShorterThan Overloaded.
- IsTrue Overloaded.
- StartsWith Overloaded.
See Also

ValidatorExtensions Class
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...::Contains Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains(ConditionValidator&lt;(Of (String)&gt;), String)</td>
<td>Checks whether the given value contains the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Contains&lt;(Of &lt;(TCollection)&gt;)&gt;)(ConditionValidator&lt;(Of &lt;(TCollection)&gt;&gt;, Object)</td>
<td>Checks whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.</td>
</tr>
<tr>
<td>Contains&lt;(Of &lt;(TCollection, TElement)&gt;)(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), TElement)</td>
<td>Checks whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.</td>
</tr>
<tr>
<td>Contains(ConditionValidator&lt;(Of (String)&gt;), String, String)</td>
<td>Checks whether the given value contains the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Contains&lt;(Of &lt;(TCollection)&gt;)&gt;)(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), Object, String)</td>
<td>Checks whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.</td>
</tr>
</tbody>
</table>
Contains<(Of <(TCollection, TElement)>)>(ConditionValidator<(Of <(TCollection)>)>, TElement, String)

Checks whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value contains the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://example.com)
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function Contains ( _
    validator As ConditionValidator(Of String), _
    value As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> Contains(
    ConditionValidator<string> validator,
    string value
)

Visual C++

public:
static ConditionValidator<String^>^ Contains(
    ConditionValidator<String^>^ validator,
    String^ value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.contains = function(validator,

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(String)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
    Type: System...:::String
    The value to compare.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value contains no null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
Contains Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::Contains Method (ConditionValidator<Of <(String)>>, String, String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value contains the specified value. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function Contains ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator(Of String) Contains(ConditionValidator(Of String) validator,
    string value,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<String>^ Contains(ConditionValidator<String>^ validator,
    String^ value,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.contains = function(validator

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator(Of String>)
The ConditionValidator(Of T) that holds the value that has to be checked.

value
Type: `System.String`
The value to compare.

conditionDescription
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value contains no null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
Contains Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function Contains(Of TCollection As IEnumerable) ( _
validator As ConditionValidator(Of TCollection), _
element As Object _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> Contains<TCollection>(
    ConditionValidator<TCollection> validator,
    Object element
)
where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ Contains(
        ConditionValidator<TCollection>^ validator,
        Object^ element
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of
    <(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
element

Type: System..::.Object

The element that should contain the given value.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentException</code></td>
<td>Thrown when the Value of the specified validator does not contain element, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>ArgumentNullException</code></td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>PostconditionException</code></td>
<td>Thrown when the Value of the specified validator does not contain element, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
Contains Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function Contains(Of TCollection As IEnumerable) ( _
    validator As ConditionValidator(Of TCollection), _
    element As Object, _
    conditionDescription As String _
) As ConditionValidator(Of TCollection) _

**C#**

public static ConditionValidator<TCollection> Contains<TCollection>(
    ConditionValidator<TCollection> validator,
    Object element,
    string conditionDescription
)

where TCollection : IEnumerable

**Visual C++**

public:
    generic<typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ Contains(
    ConditionValidator<TCollection>^ validator,
    Object^ element,
    String^ conditionDescription
)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of
    <(TCollection)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**element**
- **Type:** `System::Object`
- The element that should contain the given value.

**conditionDescription**
- **Type:** `System::String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Type Parameters

TCollection
    The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator does not contain element, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator does not contain element, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**Contains Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions..:::Contains(Of (TCollection, TElement)>)) Method
(ConditionValidator(Of <TCollection>)), TElement)

Checks whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

Public Shared Function Contains(Of TCollection As {Object, IEnumerable} validator As ConditionValidator(Of TCollection), _
  element As TElement ) As ConditionValidator(Of TCollection)

**C#**

public static ConditionValidator<TCollection> Contains<TCollection, ConditionValidator<TCollection, TElement>> validator,
  TElement element
)
where TCollection : Object, IEnumerable<TElement>

**Visual C++**

generic<typename TCollection, typename TElement>
where TCollection : Object, IEnumerable<TElement>
static ConditionValidator<TCollection>^ Contains( ConditionValidator<TCollection>^ validator,
  TElement element
)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
element
  Type: TElement
  The element that should contain the given value.
Type Parameters

TCollection
   The type of the value to check.
TElement
   The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain element, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain element, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[Contains Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
עלה Condions reference library

ValidatorExtensions..::..Contains<(Of <(TCollection, TElement)>)> Method
(ConditionValidator<(Of <(TCollection)>), TElement, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value contains the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function Contains(Of TCollection As IEnumerable(Of TE] validator As ConditionValidator(Of TCollection), _) element As TElement, _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

**C#**

public static ConditionValidator<TCollection> Contains<TCollection, ConditionValidator<TCollection> validator, TElement element, string conditionDescription 
)

where TCollection : IEnumerable<TElement>

**Visual C++**

public:

generic<typename TCollection, typename TElement>

where TCollection : IEnumerable<TElement>
static ConditionValidator<TCollection>^ Contains( ConditionValidator<TCollection>^ validator, TElement element,
    String^ conditionDescription
)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions.:::ConditionValidator<Of 
<(TCollection)>)

The ConditionValidator<Of <(T)>>(conditionDescription that holds the value that has to be
checked.

element
   Type: TElement
   The element that should contain the given value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
   The type of the value to check.
TElement
   The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator does not contain element, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator does not contain element, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
Contains Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::ContainsAll Method

ValidatorExtensions Class   See Also   Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainsAll&lt;(Of &lt;(TCollection, TElement)&gt;))(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), IEnumerable&lt;(Of &lt;(TElement)&gt;))</td>
<td>Checks whether the given value contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements. Checks whether the given value contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.</td>
</tr>
</tbody>
</table>
ContainsAll<(Of <(TCollection, TElement)>)>(ConditionValidator<(Of <(TCollection)>)>, IEnumerable<(Of <(TElement)>)>, String)

Checks whether the given value contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.
See Also

[ValidatorExtensions Class]
[ValidatorExtensions Members]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
ValidatorExtensions..::.ContainsAll(Of <(TCollection)>)(ConditionValidator(Of <(TCollection)>), IEnumerable)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function ContainsAll(Of TCollection As IEnumerable) (validator As ConditionValidator(Of TCollection), _) elements As IEnumerable _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> ContainsAll<TCollection>(ConditionValidator<TCollection> validator,
IEnumerable elements
)
where TCollection : IEnumerable

Visual C++

public:
  generic<typename TCollection>
  where TCollection : IEnumerable
  static ConditionValidator<TCollection>^ ContainsAll(ConditionValidator<TCollection>^ validator,
IEnumerable^ elements
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator(Of (Of (TCollection))>
The ConditionValidator(Of (Of (T>)>) that holds the value that has to be checked.
elements
  Type: System.Collections.IEnumerable
  The list of elements.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ContainsAll Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function ContainsAll(Of TCollection As IEnumerable) (validator As ConditionValidator(Of TCollection), _
  elements As IEnumerable, _
  conditionDescription As String _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> ContainsAll<TCollection>(ConditionValidator<TCollection> validator,  
IEnumerable elements,  
string conditionDescription)

where TCollection : IEnumerable

Visual C++

public:
  generic<typename TCollection>
  where TCollection : IEnumerable

static ConditionValidator<TCollection>^ ContainsAll(  
ConditionValidator<TCollection>^ validator,  
IEnumerable^ elements,  
String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Constraints,:::ConditionValidator<(Of  
  (Of <(TCollection)>))  
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**elements**
Type: System.Collections::IEnumerable
The list of elements.

**conditionDescription**
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ContainsAll Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

**Namespace:** [CuttingEdge.Conditions](http://www.cuttingedge.com/Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function ContainsAll(Of TCollection As IEnumerable(Of TElement), validator As ConditionValidator(Of TCollection), elements As IEnumerable(Of TElement)) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> ContainsAll<TCollection, TElement>(ConditionValidator<TCollection> validator, IEnumerable<TElement> elements)

where TCollection : IEnumerable<TElement>

Visual C++

public:
generic<typename TCollection, typename TElement>
where TCollection : IEnumerable<TElement>
static ConditionValidator<TCollection>^ ContainsAll(ConditionValidator<TCollection>^ validator, IEnumerable<TElement>^ elements)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
elements
Type: System.Collections.Generic.IEnumerable<TElement>
The list of elements.
**Type Parameters**

**TCollection**
- The type of the value to check.

**TElement**
- The type that can be considered an element of the TCollection.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...:::ArgumentException</td>
<td>Thrown when the Value of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System...:::ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions...:::PostconditionException</td>
<td>Thrown when the Value of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- ContainsAll Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ContainsAll<
(OF
(TCollection,
TElement>)>
) Method
(ConditionValidator<
(OF
(TCollection>)>
), IEnumerable<
(OF
<TElement>)>
), String)

ValidatorExtensions Class See Also Send Feedback

Checks whether the given value contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function ContainsAll(Of TCollection As IEnumerable(Of TElement), _
    elements As ConditionValidator(Of TCollection), _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> ContainsAll<TCollection, TElement>(
    ConditionValidator<TCollection> validator,
    IEnumerable<TElement> elements,
    string conditionDescription
)
where TCollection : IEnumerable<TElement>

Visual C++

generic<typename TCollection, typename TElement>
where TCollection : IEnumerable<TElement>
static ConditionValidator<TCollection>^ ContainsAll(TCollection>^ validator,
    IEnumerable<TElement>^ elements,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions,,::,ConditionValidator<Of 
<(TCollection)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

elements
    Type: System.Collections.Generic.IEnumerable<TElement>
The list of elements.

conditionDescription
    Type: System.String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
The type of the value to check.

TElement
The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator does not contain all of the elements of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ContainsAll Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::ContainsAny Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainsAny&lt;(Of &lt;(TCollection, TElement)&gt;),(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), IEnumerable&lt;(Of &lt;(TElement)&gt;)))&gt;</td>
<td>Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.</td>
</tr>
<tr>
<td>ContainsAny&lt;(Of &lt;(TCollection)&gt;),(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), IEnumerable)&gt;</td>
<td>Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.</td>
</tr>
<tr>
<td>ContainsAny&lt;(Of &lt;(TCollection, TElement)&gt;),(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), IEnumerable&lt;(Of &lt;(TElement)&gt;), String)&gt;</td>
<td>Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.</td>
</tr>
</tbody>
</table>
ContainsAny(Of (TCollection)>)
(ConditionValidator(Of TCollection), IEnumerable, String)

empty the collection is considered to not contain any element.
Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.:::ContainsAny<(Of <(TCollection<>)>)> Method
(ConditionValidator<(Of <(TCollection<>)>), IEnumerable)

Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function ContainsAny(Of TCollection As IEnumerable) (validator As ConditionValidator(Of TCollection), _ elements As IEnumerable _) As ConditionValidator(Of TCollection)

**C#**

public static ConditionValidator<TCollection> ContainsAny<TCollection>(ConditionValidator<TCollection> validator, IEnumerable elements)

where TCollection : IEnumerable

**Visual C++**

generic<typename TCollection>

where TCollection : IEnumerable

static ConditionValidator<TCollection>^ ContainsAny(ConditionValidator<TCollection>^ validator, IEnumerable^ elements)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<Of <(TCollection)>)

The ConditionValidator<Of <(T>)> that holds the value that has to be checked.
elements
Type: System.Collections:::IEnumerable
The list of elements.
Type Parameters

TCollection
  The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator does not contain any element of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator does not contain any element of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ContainsAny Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ContainsAny Method

ValidatorExtensions.ContainsAny<(Of <(TCollection)>)> Method
(ConditionValidator<(Of <(TCollection)>)>, IEnumerable, String)

Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function ContainsAny(Of TCollection As IEnumerable) (validator As ConditionValidator(Of TCollection), _
    elements As IEnumerable, _
    conditionDescription As String _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> ContainsAny<TCollection>(
    ConditionValidator<TCollection> validator,
    IEnumerable elements,
    string conditionDescription
)

where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ ContainsAny(
    ConditionValidator<TCollection>^ validator,
    IEnumerable^ elements,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of 
    <(TCollection)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

elements
   Type: System.Collections.Generic.IEnumerable
   The list of elements.

conditionDescription
   Type: System.String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the Value of the specified validator does not contain any element of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the Value of the specified validator does not contain any element of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**ContainsAny Overload**
**CuttingEdge.Conditions Namespace**

Send feedback on this topic to Microsoft.
Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

**Namespace:** [CuttingEdge.Conditions](https://example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

Public Shared Function ContainsAny(Of TCollection As(IEnumerable(Of TElement)), validator As ConditionValidator(Of TCollection), elements As IEnumerable(Of TElement)) As ConditionValidator(Of TCollection)

#### C#

public static ConditionValidator<TCollection> ContainsAny<TCollection, TElement>(ConditionValidator<TCollection> validator, IEnumerable<TElement> elements)

where TCollection : IEnumerable<TElement>

#### Visual C++

public:

generic<typename TCollection, typename TElement>

where TCollection : IEnumerable<TElement>

static ConditionValidator<TCollection>^ ContainsAny(ConditionValidator<TCollection>^ validator, IEnumerable<TElement>^ elements)

#### JavaScript

JavaScript does not support generic types or methods.

### Parameters

**validator**

Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(TCollection)>>

The ConditionValidator<Of <(T)> > that holds the value that has to be checked.
elements
Type: `System.Collections.Generic.IEnumerable<TElement>`
The list of elements.
**Type Parameters**

TCollection
   The type of the value to check.
TElement
   The type that can be considered an element of the TCollection.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th></th>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the Value of the specified validator does not contain any element of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td></td>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td></td>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the Value of the specified validator does not contain any element of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ContainsAny Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function ContainsAny(Of TCollection As IEnumerable(Of TElement), _
validator As ConditionValidator(Of TCollection), _
elements As IEnumerable(Of TElement), _
conditionDescription As String _) As ConditionValidator(Of TCollection)

**C#**

```csharp
public static ConditionValidator<TCollection> ContainsAny<TCollection, TElement>(
    ConditionValidator<TCollection> validator,
    IEnumerable<TElement> elements,
    string conditionDescription
)
```

where TCollection : IEnumerable<TElement>

**Visual C++**

```cpp
public:
    template<typename TCollection, typename TElement>
    static ConditionValidator<TCollection>^ ContainsAny(
        ConditionValidator<TCollection>^ validator,
        IEnumerable<TElement>^ elements,
        String^ conditionDescription
    )
```

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

**validator**

Type: `CuttingEdge.Conditions..:::ConditionValidator<Of (Of <TCollection>)>)`  
The `ConditionValidator<Of <T>)>` that holds the value that has to be
checked.

**elements**
Type: `System.Collections.Generic::IEnumerable<TElement>`
The list of elements.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
  The type of the value to check.
TElement
  The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain any element of the given elements list, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain any element of the given elements list, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[ContainsAny Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
Include Protected Members  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::DoesNotContain Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DoesNotContain(ConditionValidator&lt;(Of &lt;(String)&gt;), String)</code></td>
<td>Checks whether the given value does not contain the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotContain&lt;(Of &lt;(TCollection)&gt;)&gt;(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), Object)</code></td>
<td>Checks whether the given value does not contain the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.</td>
</tr>
<tr>
<td><code>DoesNotContain&lt;(Of &lt;(TCollection, TElement)&gt;)&gt;(ConditionValidator&lt;(Of &lt;(TCollection)&gt;)&gt;, TElement)</code></td>
<td>Checks whether the given value does not contain the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.</td>
</tr>
<tr>
<td><code>DoesNotContain(ConditionValidator&lt;(Of &lt;(String)&gt;), String, String)</code></td>
<td>Checks whether the given value does not contain the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotContain&lt;(Of &lt;(TCollection)&gt;)&gt;</code></td>
<td>Checks whether the given value does not contain the specified element. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
(ConditionValidator(Of TCollection, Object, String)
otherwise. When the value is a null reference it is considered empty and therefore won't contain element.
Checks whether the given value does not contain the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.

DoesNotContain(Of TCollection, TElement>(ConditionValidator(Of TCollection, TElement, String))
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::DoesNotContain Method (ConditionValidator<Of <(String)>), String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value does not contain the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContain ( _
    validator As ConditionValidator(Of String), _
    value As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> DoesNotContain(
    ConditionValidator<string> validator,
    string value
)

Visual C++

public: ConditionValidator<string>^ DoesNotContain(
    ConditionValidator<string>^ validator,
    String^ value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.doesNotContain = function

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(String)>)> The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
    Type: System..:::String The value to compare.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator contains value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator and value are both null references, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator contains value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContain Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..:::DoesNotContain Method (ConditionValidator<Of <(String)>>, String, String)

Checks whether the given value does not contain the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function DoesNotContain ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    conditionDescription As String _
) As ConditionValidator(Of String)

**C#**

public static ConditionValidator<string> DoesNotContain(ConditionValidator<string> validator,
    string value,
    string conditionDescription)

**Visual C++**

public:
static ConditionValidator<String^>^ DoesNotContain(ConditionValidator<String^>^ validator,
    String^ value,
    String^ conditionDescription)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.doesNotContain = function

**Parameters**

**validator**

Type: **CuttingEdge.Conditions..::.ConditionValidator<(Of <(String)>)>**

The **ConditionValidator(Of (T)>)** that holds the value that has to be checked.

**value**
Type: `System::String`
The value to compare.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator and value are both null references, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContain Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::DoesNotContain<(Of <(TCollection)>)> Method
(ConditionValidator<(Of <(TCollection)>)>), Object)

`ValidatorExtensions Class`  `See Also`  `Send Feedback`

Checks whether the given value does not contain the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.

**Namespace:**  `CuttingEdge.Conditions`
**Assembly:**  `CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)`
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContain(Of TCollection As IEnumerable, 
    validator As ConditionValidator(Of TCollection), 
    element As Object) 
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContain<TCollection>(
    ConditionValidator<TCollection> validator,
    Object element
)
where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ DoesNotContain( 
        ConditionValidator<TCollection>^ validator,
        Object^ element
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions::ConditionValidator(Of TCollection)

The ConditionValidator(Of T <T>) that holds the value that has to be checked.
element
  Type: System..::.Object
  The element that should contain the given value.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator does contain element, while the specified validator is created using the <strong>Requires</strong> extension method. Thrown when the <strong>Value</strong> of the specified validator does contain element, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td></td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**

**DoesNotContain Overload**

**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.DoesNotContain<(Of <(TCollection<>)>)) Method
(ConditionValidator<(Of <(TCollection<>)>)), Object, String)

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value does not contain the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContain(Of TCollection As IEnumerable, validator As ConditionValidator(Of TCollection), _
         element As Object, _
         conditionDescription As String) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesnNotContain<TCollection>(ConditionValidator<TCollection> validator, Object element,
                                 string conditionDescription)

where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ DoesnNotContain(ConditionValidator<TCollection>^ validator,
                                                             Object^ element,
                                                             String^ conditionDescription)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Contrains::::ConditionValidator<(Of TCollection)> (Of TCollection>)
    The ConditionValidator<(Of TCollection)> that holds the value that has to be
checked.

element
Type: System::Object
The element that should contain the given value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection

The type of the value to check.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator does contain element, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator does contain element, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContain Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::DoesNotContain<(Of <(TCollection, TElement)>)> Method (ConditionValidator<(Of <(TCollection)>), TElement)

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value does not contain the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContain(Of TCollection As IEnumerable(validator As ConditionValidator(Of TCollection), _
element As TElement)) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContain<TCollection>(validator, TElement element)

where TCollection : IEnumerable<TElement>

Visual C++

public:

generic<typename TCollection, typename TElement>

where TCollection : IEnumerable<TElement>

static ConditionValidator<TCollection>^ DoesNotContain(ConditionValidator<TCollection>^ validator, TElement element)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<Of (Of <TCollection>)>)

The ConditionValidator<Of <(T)> > that holds the value that has to be checked.
element
  Type: TElement
  The element that should contain the given value.
Type Parameters

TCollection
   The type of the value to check.
TElement
   The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SystemArgumentNullException</code></td>
<td>Thrown when the Value of the specified validator does contain element, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.PostconditionException</code></td>
<td>Thrown when the Value of the specified validator does contain element, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContain Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::DoesNotContain<(Of <(TCollection, TElement)>)> Method (ConditionValidator<(Of <(TCollection)>), TElement, String)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value does not contain the specified element. An exception is thrown otherwise. When the value is a null reference it is considered empty and therefore won't contain element.
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContain(Of TCollection As IEnumerable(validator As ConditionValidator(Of TCollection), _
element As TElement, _
conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContain<TCollection>(validator, TElement element, string conditionDescription

Visual C++

public:

generic<typename TCollection, typename TElement>
where TCollection : IEnumerable<TElement>
static ConditionValidator<TCollection>^ DoesNotContain(ConditionValidator<TCollection>^ validator, TElement element, String^ conditionDescription

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of
<(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

element
   Type: TElement
   The element that should contain the given value.

conditionDescription
   Type: System.String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
   The type of the value to check.

TElement
   The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator does contain element, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator does contain element, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**DoesNotContain Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::DoesNotContainAll Method
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DoesNotContainAll</strong>&lt;sup&gt;1&lt;/sup&gt;(Of &lt;sup&gt;1&lt;/sup&gt;(TCollection, TElement)&gt;(ConditionValidator&lt;sup&gt;1&lt;/sup&gt;(Of &lt;sup&gt;1&lt;/sup&gt;(TCollection)&gt;, IEnumerable&lt;sup&gt;1&lt;/sup&gt;(Of &lt;sup&gt;1&lt;/sup&gt;(TElement)&gt;)&lt;sup&gt;2&lt;/sup&gt;)&lt;/sup&gt;)</td>
<td>Checks whether the given value does not contain all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.</td>
</tr>
</tbody>
</table>

<sup>1</sup>Conditions: **TCollection** : Type parameter; **TElement** : Type parameter.
DoesNotContainAll(Of (TCollection, TElement)>(ConditionValidator(Of (TCollection), IEnumerable(Of (TElement)>, String))

Checks whether the given value does not contain all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

Checks whether the given value does not contain all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value does not contain all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContainAll(Of TCollection As IEnumerat
    validator As ConditionValidator(Of TCollection), _
    elements As IEnumerable _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContainAll<TCol] ConditionValidator<TCollection> validator,
    IEnumerable elements
}
where TCollection : IEnumerable

Visual C++

public:
generic<typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ DoesNotContainAll( ConditionValidator<TCollection>^ validator,
    IEnumerable^ elements
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Controns..:::ConditionValidator(Of TCollection>
The ConditionValidator(Of TCol> that holds the value that has to be checked.
elements
  Type: System.Collections::*::IEnumerable
  The list of elements.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>System.</em>::&lt;code&gt;ArgumentException&lt;/code&gt;</td>
<td>Thrown when the <code>Value</code> of the specified validator does contain all of the elements of the given elements list, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><em>System.</em>::&lt;code&gt;ArgumentNullException&lt;/code&gt;</td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the specified elements list is null or empty, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><em>CuttingEdge.Conditions.</em>::&lt;code&gt;PostconditionException&lt;/code&gt;</td>
<td>Thrown when the <code>Value</code> of the specified validator does contain all of the elements of the given elements list, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContainAll Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value does not contain all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

**Namespace:** [CuttingEdge.Conditions](http://www.example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContainAll(Of TCollection As IEnumerable validator As ConditionValidator(Of TCollection), _
   elements As IEnumerable, _
   conditionDescription As String _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContainAll<TCol]
   ConditionValidator<TCollection> validator,
   IEnumerable elements,
   string conditionDescription
)
where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ DoesNotContainAll(ConditionValidator<TCollection>^ validator,
    IEnumerable^ elements,
    String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of (TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**elements**
Type: System.Collections.Enumerable
The list of elements.

**conditionDescription**
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator does contain all of the elements of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System::ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and the specified elements list is null or empty, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator does contain all of the elements of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContainAll Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
validatorExtensions.DoesNotContainAll(Of TCollection, TElement)
Method (ConditionValidator(Of TCollection), IEnumerable(Of TElement))

`validatorExtensions Class`  `See Also`  `Send Feedback`

Checks whether the given value does not contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.

**Namespace:** [CuttingEdge.Conditions](CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContainAll(Of TCollection As IEnumerable(Of TElement), _
  validator As ConditionValidator(Of TCollection), _
  elements As IEnumerable(Of TElement)) _
  As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContainAll<TCol] ConditionValidator<TCollection> validator, 
  IEnumerable<TElement> elements
}
where TCollection : IEnumerable<TElement>

Visual C++

public:
  generic<typename TCollection, typename TElement>
  where TCollection : IEnumerable<TElement>
  static ConditionValidator<TCollection>^ DoesNotContainAll( 
    ConditionValidator<TCollection>^ validator, 
    IEnumerable<TElement>^ elements
  )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator(Of 
  <(TCollection)>)
  The ConditionValidator(Of (T)>) that holds the value that has to be checked.
elements
    Type: System.Collections.Generic.IEnumerable<TElement>
    The list of elements.
**Type Parameters**

TCollection
   The type of the value to check.
TElement
   The type that can be considered an element of the TCollection.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...:::ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator does contain all of the elements of the specified elements list, while the specified validator is created using the <code>Requires</code> extension method. Thrown when the <code>Value</code> of the specified validator is a null reference and the specified elements list is null or empty, while the specified validator is created using the <code>Requires</code> extension method. Thrown when the <code>Value</code> of the specified validator does contain all of the elements of the specified elements list, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
<tr>
<td>System...:::ArgumentNullException</td>
<td></td>
</tr>
<tr>
<td>CuttingEdge.Conditions...:::PostconditionException</td>
<td></td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContainAll Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions:::DoesNotContainAll<(Of <(TCollection, TElement)>)>(ConditionValidator<(Of <(TCollection)>)>, IEnumerable<(Of <(TElement)>)>, String)

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value does not contains all of the specified elements. An exception is thrown otherwise. When the elements collection is a null reference or an empty list, the collection is considered to contain all of the specified (even if the value itself is empty). When the given value is empty and the given elements list isn't, the collection is considered to not contain all of the specified elements.
### Syntax

**Visual Basic (Declaration)**

Public Shared Function DoesNotContainAll(Of TCollection As IEnumerable, validator As ConditionValidator(Of TCollection), _
elements As IEnumerable(Of TElement), _
conditionDescription As String _) As ConditionValidator(Of TCollection)

**C#**

```csharp
public static ConditionValidator<TCollection> DoesNotContainAll<TCollection,
ConditionValidator<TCollection> validator, 
IEnumerable<TElement> elements,
string conditionDescription
)
where TCollection : IEnumerable<TElement>
```

**Visual C++**

```cpp
public:
generic< typename TCollection, typename TElement>
where TCollection : IEnumerable<TElement>
static ConditionValidator<TCollection>^ DoesNotContainAll( 
ConditionValidator<TCollection>^ validator, 
IEnumerable<TElement>^ elements, 
String^ conditionDescription
)
```

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

**validator**

Type: `CuttingEdge.Conditions::ConditionValidator<Of <(TCollection)>>(Of <(TCollection)>)`

The `ConditionValidator<Of <(T)>>()` that holds the value that has to be
checked.

elements
  Type: System.Collections.Generic.IEnumerable<TElement>
  The list of elements.

conditionDescription
  Type: System.String
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
  The type of the value to check.
TElement
  The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator does contain all of the elements of the specified elements list, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and the specified elements list is null or empty, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator does contain all of the elements of the specified elements list, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContainAll Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::DoesNotContainAny Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DoesNotContainAny</strong>&lt;Of *(Of *(TCollection, TElement&gt;)), IValidator&gt;(Of *(Of *(TCollection&gt;), IEnumerable&lt;Of *(Of *(TElement&gt;))))&gt;</td>
<td>Checks whether the given value does not contain any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.</td>
</tr>
<tr>
<td><strong>DoesNotContainAny</strong>&lt;Of *(Of *(TCollection&gt;), IValidator&gt;(Of *(Of *(TCollection&gt;), IEnumerable&lt;Of *(Of *(TElement&gt;))]</td>
<td>Checks whether the given value does not contain any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.</td>
</tr>
<tr>
<td><strong>DoesNotContainAny</strong>&lt;Of *(Of *(TCollection, TElement&gt;)), IValidator&gt;(Of *(Of *(TCollection&gt;), IEnumerable&lt;Of *(Of *(TElement&gt;)), String)&gt;</td>
<td>Checks whether the given value does not contain any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or</td>
</tr>
</tbody>
</table>
DoesNotContainAny(Of (TCollection<>))(ConditionValidator(Of (TCollection<>)), IEnumerable, String)

Checks whether the given value does not contain any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value does not contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContainAny(Of TCollection As IEnumberal 
    validator As ConditionValidator(Of TCollection), _ 
    elements As IEnumberal _) 
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContainAny<TCol] 
    ConditionValidator<TCollection> validator, 
    IEnumberal elements 
)
where TCollection : IEnumberal

Visual C++

public: 
    generic<typename TCollection> 
    where TCollection : IEnumberal 
    static ConditionValidator<TCollection>^ DoesNotContainAny( 
        ConditionValidator<TCollection>^ validator, 
        IEnumberal^ elements 
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator 
    Type: CuttingEdge.Conditions..:::ConditionValidator<Of 
        <(TCollection)>)
    The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.
elements
Type: System.Collections::IEnumerable
The list of elements.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>System.</em>::* ArgumentException</td>
<td>Thrown when the Value of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><em>CuttingEdge.Conditions.</em>::* PostconditionException</td>
<td>Thrown when the Value of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [DoesNotContainAny Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidateExtensions..::.DoesNotContainAny(Of (Of (TCollection)>)>) Method
(ConditionValidator(Of (Of (TCollection)>)>), IEnumerable, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value does not contain any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContainAny(Of TCollection As IEnumerable) 
validator As ConditionValidator(Of TCollection), _
elements As IEnumerable, _
conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContainAny<TCol> 
ConditionValidator<TCollection> validator, 
IEnumerable elements, 
string conditionDescription

where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ DoesNotContainAny( 
        ConditionValidator<TCollection>^ validator, 
        IEnumerable^ elements, 
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

elements
   Type: System.Collections.IEnumerable
   The list of elements.

conditionDescription
   Type: System.String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [DoesNotContainAny Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::DoesNotContainAny<Of <(TCollection, TElement)>>)
Method (ConditionValidator<Of <(TCollection)>), IEnumerable<Of <(TElement)>))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value does not contains any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
`Syntax`

**Visual Basic (Declaration)**

Public Shared Function DoesNotContainAny(Of TCollection As `IEnumerable` validator As `ConditionValidator(Of TCollection)`, elements As `IEnumerable(Of TElement)`) As `ConditionValidator(Of TCollection)`

**C#**

public static `ConditionValidator<TCollection>` DoesNotContainAny<TCollection, TElement>(`ConditionValidator<TCollection>` validator, `IEnumerable<TElement>` elements)

where TCollection : `IEnumerable<TElement>`

**Visual C++**

public:

generic<typename TCollection, typename TElement>
where TCollection : `IEnumerable<TElement>`
static `ConditionValidator<TCollection>`^ DoesNotContainAny(`ConditionValidator<TCollection>`^ validator, `IEnumerable<TElement>`^ elements)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator
Type: `CuttingEdge.Conditions::ConditionValidator<Of <(TCollection)>)`
The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.
elements
Type: System.Collections.Generic.IEnumerable<TElement>
The list of elements.
Type Parameters

TCollection
   The type of the value to check.
TElement
   The type that can be considered an element of the TCollection.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotContainAny Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::DoesNotContainAny<(Of <(TCollection, TElement)>)> Method (ConditionValidator<(Of <(TCollection)>), IEnumerable<(Of <(TElement)>)), String)

Checks whether the given value does not contain any of the specified elements. An exception is thrown otherwise. When the value is a null reference or an empty list it won't contain any elements. When the elements list is null or empty the collection is considered to not contain any element.

**Namespace:** [CuttingEdge.Conditions](CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotContainAny(Of TCollection As IEnumerable validator As ConditionValidator(Of TCollection), _
elements As IEnumerable(Of TElement), _
conditionDescription As String _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> DoesNotContainAny<TCol>
ConditionValidator<TCollection> validator,
IEnumerable<TElement> elements,
string conditionDescription

where TCollection : IEnumerable<TElement>

Visual C++

public:

generic<typename TCollection, typename TElement>
where TCollection : IEnumerable<TElement>
static ConditionValidator<TCollection>^ DoesNotContainAny(ConditionValidator<TCollection>^ validator,
IEnumerable<TElement>^ elements,
String^ conditionDescription

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions.::.ConditionValidator<(Of 
<(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

elements
   Type: System.Collections.Generic.IEnumerable<TElement>
   The list of elements.

conditionDescription
   Type: System.String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
The type of the value to check.

TElement
The type that can be considered an element of the TCollection.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain one or more elements of the given elements list, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidateExtensions Class](#)
[DoesNotContainAny Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::DoesNotEndWith Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DoesNotEndWith(ConditionValidator(Of &lt;(String)&gt;), String)</code></td>
<td>Checks whether the end of the given value does not match the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotEndWith(ConditionValidator(Of &lt;(String)&gt;), String, String)</code></td>
<td>Checks whether the end of the given value does not match the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotEndWith(ConditionValidator(Of &lt;(String)&gt;), String, StringComparison)</code></td>
<td>Checks whether the end of the given value does not match the specified value using the specified comparison option. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotEndWith(ConditionValidator(Of &lt;(String)&gt;), String, StringComparison, String)</code></td>
<td>Checks whether the end of the given value does not match the specified value using the specified comparison option. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
DoesNotEndWith Method (ConditionValidator<(Of <(String)>), String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the end of the given value does not match the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function DoesNotEndWith ( _
    validator As ConditionValidator(Of String), _
    value As String) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> DoesNotEndWith(
    ConditionValidator<string> validator,
    string value)
```

**Visual C++**

```cpp
public: ConditionValidator<String^>^ DoesNotEndWith(
    ConditionValidator<String^>^ validator,
    String^ value)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.doesNotEndWith = function
```

**Parameters**

**validator**
Type: `CuttingEdge.Conditions::ConditionValidator<(Of <(String)>)>`
The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**value**
Type: `System::String`
The value to compare.
Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...:..ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator ends with value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>System...:..ArgumentNullException</td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions...:..PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator ends with value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [DoesNotEndWith Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions...:::DoesNotEndWith Method (ConditionValidator<Of <(String)>), String, String)

ValidatorExtensions Class   See Also   Send Feedback

Checks whether the end of the given value does not match the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function DoesNotEndWith (_
    validator As ConditionValidator(Of String), _
    value As String, _
    conditionDescription As String _
) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> DoesNotEndWith(
    ConditionValidator<string> validator,
    string value,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
    static ConditionValidator<String^>^ DoesNotEndWith(
        ConditionValidator<String^>^ validator,
        String^ value,
        String^ conditionDescription
    )
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.doesNotEndWith = function
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions..::.ConditionValidator<Of <(String)>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**value**
Type: `System::String`
The value to compare.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator ends with value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions...::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator ends with value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- DoesNotEndWith Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
DoesNotEndWith Method (ConditionValidator<Of<(String)>), String, StringComparison)

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the end of the given value does not match the specified value using the specified comparison option. An exception is thrown otherwise.
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function DoesNotEndWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison _
) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> DoesNotEndWith(
    ConditionValidator<string> validator,
    string value,
    StringComparison comparisonType
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<String^>^ DoesNotEndWith(
    ConditionValidator<String^>^ validator,
    String^ value,
    StringComparison comparisonType
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.doesNotEndWith = function
```

**Parameters**

- **validator**
  - Type: `CuttingEdge.Conditions.:::ConditionValidator<(Of <(String)>)>`
  - The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

- **value**
Type: `System.String`
The value to compare.

comparisonType
Type: `System.StringComparison`
One of the `StringComparison` values that determines how this string and value are compared

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator ends with value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator ends with value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotEndWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::DoesNotEndWith Method (ConditionValidator<Of <(String)>), String, StringComparison, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the end of the given value does not match the specified value using the specified comparison option. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function DoesNotEndWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison, _
    conditionDescription As String _
) As ConditionValidator(Of String)
```

#### C#

```csharp
public static ConditionValidator&lt;string&gt; DoesNotEndWith(
    ConditionValidator&lt;string&gt; validator,
    string value,
    StringComparison comparisonType,
    string conditionDescription
)
```

#### Visual C++

```cpp
public:
static ConditionValidator^(^ string^) DoesNotEndWith(
    ConditionValidator^(^ string^) validator,
    string^ value,
    StringComparison^ comparisonType,
    string^ conditionDescription
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.doesNotEndWith = function
```

### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions::<T,__::ConditionValidator<(Of <(String>>)>)`
  - The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.
value
   Type: System.String
   The value to compare.

comparisonType
   Type: System.StringComparison
   One of the StringComparison values that determines how this string and value are compared

conditionDescription
   Type: System.String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator ends with value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator ends with value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotEndWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
□  Include Protected Members
□  Include Inherited Members

CuttingEdge.Conditions reference library

ValidatorExtensions...:::DoesNotHaveLength Method

ValidatorExtensions Class  See Also  Send Feedback
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DoesNotHaveLength(ConditionValidator&lt;Of &lt;(String)&gt;, Int32)</code></td>
<td>Checks whether the given value is unequal in length to length. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><code>DoesNotHaveLength&lt;Of &lt;(TCollection)&gt;)&gt;(ConditionValidator&lt;Of &lt;(TCollection)&gt;, Int32)</code></td>
<td>Checks whether the number of elements in the given value, is different from the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td><code>DoesNotHaveLength(ConditionValidator&lt;Of &lt;(String)&gt;, Int32, String)</code></td>
<td>Checks whether the given value is unequal in length to length. An exception is thrown otherwise. A null reference is considered to have a length of 0. Checks whether the number of elements in the given value, is different from the specified numberOfElements</td>
</tr>
</tbody>
</table>
| `DoesNotHaveLength<Of <(TCollection)>)>(ConditionValidator<Of <(TCollection)>,)` | (ConditionValidator<Of <(TCollection)>,)
argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.
See Also

[ValidatorExtensions Class]
[ValidatorExtensions Members]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript

CuttingEdge.Conditions reference library

`ValidatorExtensions..::.DoesNotHaveLength Method (ConditionValidator<Of (String)>, Int32)`

[ValidatorExtensions Class](#)  [See Also](#)  [Send Feedback](#)

Checks whether the given value is unequal in length to `length`. An exception is thrown otherwise. A null reference is considered to have a length of 0.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function DoesNotHaveLength ( _
    validator As ConditionValidator(Of String), _
    length As Integer _) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> DoesNotHaveLength(ConditionValidator<string> validator, int length)
```

**Visual C++**

```cpp
public:
static ConditionValidator<String^>^ DoesNotHaveLength(ConditionValidator<String^>^ validator, int length)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.doesNotHaveLength = funct
```

### Parameters

**validator**

- Type: `CuttingEdge.Conditions..:::ConditionValidator<(Of <(String)>)>`
- The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**length**

- Type: `System..:::Int32`
- The invalid length.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the length of <code>Value</code> of the specified validator equals length, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and length equals 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the length of <code>Value</code> of the specified validator equals length, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**

**DoesNotHaveLength Overload**

**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::.DoesNotHaveLength Method (ConditionValidator<(Of <(String)>), Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is unequal in length to length. An exception is thrown otherwise. A null reference is considered to have a length of 0.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function DoesNotHaveLength ( _
    validator As ConditionValidator(Of String), _
    length As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of String)

**C#**

public static ConditionValidator<string> DoesNotHaveLength(
    ConditionValidator<string> validator,
    int length,
    string conditionDescription
)

**Visual C++**

public:
static ConditionValidator<String^>^ DoesNotHaveLength(
    ConditionValidator<String^>^ validator,
    int length,
    String^ conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.doesNotHaveLength = funct

### Parameters

**validator**

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(String)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

**length**
Type: `System::Int32`
The invalid length.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the length of Value of the specified validator equals length, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the Value of the specified validator is a null reference and length un equals 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the length of Value of the specified validator equals length, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[DoesNotHaveLength Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Checks whether the number of elements in the given value, is different from the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function DoesNotHaveLength(Of TCollection As IEnumerable) validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer _
) As ConditionValidator(Of TCollection)
```

**C#**

```csharp
public static ConditionValidator<TCollection> DoesNotHaveLength<TCollection>(ConditionValidator<TCollection> validator, int numberOfElements)
```

where TCollection : IEnumerable

**Visual C++**

```cpp
public:
    template<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ DoesNotHaveLength(ConditionValidator<TCollection>^ validator, int numberOfElements)
```

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

**validator**

Type: `CuttingEdge.Conditions..::.ConditionValidator(Of <TCollection>)`<br>

The `ConditionValidator(Of <TCollection>)` that holds the value that has to be checked.
numberOfElements

Type: System::Int32

The number of elements the collection should not contain.
**Type Parameters**

TCollection  
  The type of the value to check.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain the number of elements as specified with the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> equals 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain the number of elements as specified with the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotHaveLength Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

**ValidatorExtensions...:::DoesNotHaveLength(Of (TCollection)>>) Method**
(ConditionValidator(Of (TCollection)>), Int32, String)

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the number of elements in the given value, is different from the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function DoesnNotHaveLength(Of TCollection As IEnumerat
validator As ConditionValidator(Of TCollection), _
numberOfElements As Integer, _
conditionDescription As String _
) As ConditionValidator(Of TCollection)

**C#**

public static ConditionValidator<TCollection> DoesnNotHaveLength<TCol
ConditionValidator<TCollection> validator,
int numberOfElements,
string conditionDescription
}
where TCollection : IEnumeratable

**Visual C++**

public:

generic<typename TCollection>
where TCollection : IEnumeratable
static ConditionValidator<TCollection>^ DoesnNotHaveLength( ConditionValidator<TCollection>^ validator,
int numberOfElements,
String^ conditionDescription
}

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of
<(TCollection)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**numberOfElements**
Type: `System::Int32`
The number of elements the collection should not contain.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain the number of elements as specified with the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> equals 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does contain the number of elements as specified with the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotHaveLength Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DoesNotStartWith(ConditionValidator&lt;OfString, String&gt;)</code></td>
<td>Checks whether the given value does not start with the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotStartWith(ConditionValidator&lt;OfString, String, String&gt;)</code></td>
<td>Checks whether the given value does not start with the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotStartWith(ConditionValidator&lt;OfString, String, StringComparison&gt;)</code></td>
<td>Checks whether the given value does not start with the specified value using the specified comparison option. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>DoesNotStartWith(ConditionValidator&lt;OfString, String, StringComparison, String&gt;)</code></td>
<td>Checks whether the given value does not start with the specified value using the specified comparison option. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

 ValidatorExtensions Class
 ValidatorExtensions Members
 CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions...:::DoesNotStartWith Method (ConditionValidator<Of <(String>)>, String)

Checks whether the given value does not start with the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotStartWith ( _
    validator As ConditionValidator(Of String), _
    value As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<String> DoesNotStartWith(
    ConditionValidator<String> validator,
    string value
)

Visual C++

public:
static ConditionValidator<String^>^ DoesNotStartWith(
    ConditionValidator<String^>^ validator,
    String^ value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.doesNotStartWith = functi;

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator(Of <(String)>)
    The ConditionValidator(Of <(T)>>) that holds the value that has to be checked.

value
    Type: System..:::String
    The value to compare.
Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator start with value, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator start with value, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotStartWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.DoesNotStartWith Method (ConditionValidator<Of <(String>>), String, String)

Validatorextensions Class  See Also  Send Feedback

Checks whether the given value does not start with the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotStartWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> DoesNotStartWith(
    ConditionValidator<string> validator, 
    string value, 
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<String^> DoesNotStartWith(
        ConditionValidator<String^> validator, 
        String^ value, 
        String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.doesNotStartWith = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(String)>)>)
The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.

value
Type: `System::String`
The value to compare.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator start with value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator start with value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- DoesNotStartWith Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value does not start with the specified value using the specified comparison option. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotStartWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> DoesNotStartWith(
    ConditionValidator<string> validator,
    string value,
    StringComparison comparisonType
)

Visual C++

public:
static ConditionValidator<String^>^ DoesNotStartWith(
    ConditionValidator<String^>^ validator,
    String^ value,
    StringComparison comparisonType
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.doesNotStartWith = function
Type: `System.String` 
The value to compare.

comparisonType  
Type: `System.StringComparison`  
One of the `StringComparison` values that determines how this string and value are compared

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator start with value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator start with value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotStartWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions:::DoesNotStartWith Method (ConditionValidator<Of <(String)>>, String, StringComparison, String)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function DoesNotStartWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison, _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> DoesNotStartWith(ConditionValidator<string> validator,
    string value,
    StringComparison comparisonType,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<String^>^ DoesNotStartWith(ConditionValidator<String^>^ validator,
    String^ value,
    StringComparison comparisonType,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.doesNotStartWith = functi

Parameters

validator

Type: CuttingEdge.Conditions::ConditionValidator(Of String>)
The ConditionValidator(Of (T)> that holds the value that has to be checked.
value
  Type: System.String
  The value to compare.

comparisonType
  Type: System.StringComparison
  One of the StringComparison values that determines how this string and value are compared

conditionDescription
  Type: System.String
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator start with value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator start with value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
DoesNotStartWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions:::EndsWith Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EndsWith(ConditionValidator(Of &lt;(String)&gt;), String)</code></td>
<td>Checks whether the end of the given value matches the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>EndsWith(ConditionValidator(Of &lt;(String)&gt;), String, String)</code></td>
<td>Checks whether the end of the given value matches the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>EndsWith(ConditionValidator(Of &lt;(String)&gt;), String, StringComparison)</code></td>
<td>Checks whether the end of the given value matches the specified value using the specified comparison option. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>EndsWith(ConditionValidator(Of &lt;(String)&gt;), String, StringComparison, String)</code></td>
<td>Checks whether the end of the given value matches the specified value using the specified comparison option. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.:::EndsWith Method (ConditionValidator<Of (String)>, String)

Checks whether the end of the given value matches the specified value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function EndsWith (_
    validator As ConditionValidator(Of String), _
    value As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> EndsWith(
    ConditionValidator<string> validator,
    string value
)

Visual C++

public:
static ConditionValidator<String^>^ EndsWith(
    ConditionValidator<String^>^ validator,
    String^ value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.endsWith = function(validator,

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(String)>>)
The ConditionValidator<(Of <(T)>>) that holds the value that has to be checked.

value
Type: System..:::String
The value to compare.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not end with value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions...::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not end with value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- **ValidatorExtensions Class**
- **EndsWith Overload**
- **CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::.EndsWith Method (ConditionValidator<Of <(String)>, String, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the end of the given value matches the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function EndsWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    conditionDescription As String _)
    As ConditionValidator(Of String)

C#

public static ConditionValidator<string> EndsWith(
    ConditionValidator<string> validator,
    string value,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<string>^ EndsWith(
        ConditionValidator<string>^ validator,
        String^ value,
        String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.endsWith = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions::ConditionValidator<(Of <(String)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: `System.String`
The value to compare.

`conditionDescription`
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not end with value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not end with value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
EndsWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the end of the given value matches the specified value using the specified comparison option. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function EndsWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison _
) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> EndsWith(
    ConditionValidator<string> validator,
    string value,
    StringComparison comparisonType
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<String^>^ EndsWith(
    ConditionValidator<String^>^ validator,
    String^ value,
    StringComparison comparisonType
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.endsWith = function(validator,
```

**Parameters**

- **validator**
  - Type: `CuttingEdge.Conditions..::.ConditionValidator(Of <(String)>)`
  - The `ConditionValidator(Of <(T)>)` that holds the value that has to be checked.

- **value**
Type: `System::String`
The value to compare.

`comparisonType`
Type: `System::StringComparison`
One of the `StringComparison` values that determines how this string and value are compared

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.....ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not end with value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>System.....ArgumentNullException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.....PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not end with value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
EndsWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::.EndsWith Method (ConditionValidator<Of <(String)>), String, StringComparison, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the end of the given value matches the specified value using the specified comparison option. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function EndsWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison, _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> EndsWith(
    ConditionValidator<string> validator,
    string value,
    StringComparison comparisonType,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<String^>^ EndsWith(
        ConditionValidator<String^>^ validator,
        String^ value,
        StringComparison comparisonType,
        String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.endsWith = function(validator,

Parameters

validator

Type: CuttingEdge.Conditions::ConditionValidator<Of <(String)>)>
The ConditionValidator<Of <(T)>)> that holds the value that has to be checked.
value
Type: System.String
The value to compare.

comparisonType
Type: System.StringComparison
One of the StringComparison values that determines how this string and value are compared.

conditionDescription
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not end with value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not end with value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
EndsWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions.Evaluate Method
ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluate&lt;(Of &lt;<em>&gt;)&gt; (ConditionValidator&lt;(Of &lt;</em>&gt;), Boolean)</td>
<td>Checks whether the specified condition equals <code>true</code>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Evaluate&lt;(Of &lt;<em>&gt;)&gt; (ConditionValidator&lt;(Of &lt;</em>&gt;), Expression&lt;(Of &lt;(Func&lt;(Of &lt;<em>&gt;), Boolean&gt;)</em>&gt;)*&gt;)</td>
<td>Checks whether the specified expression evaluates <code>true</code> on the given value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Evaluate&lt;(Of &lt;<em>&gt;)&gt; (ConditionValidator&lt;(Of &lt;</em>&gt;), Boolean, String)</td>
<td>Checks whether the specified condition equals <code>true</code>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Evaluate&lt;(Of &lt;<em>&gt;)&gt; (ConditionValidator&lt;(Of &lt;</em>&gt;), Expression&lt;(Of &lt;(Func&lt;(Of &lt;<em>&gt;), Boolean&gt;)</em>&gt;)*&gt;, String)</td>
<td>Checks whether the specified expression evaluates <code>true</code> on the given value. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[ValidatorExtensions Members](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Checks whether the specified condition equals **true**. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://www.cuttingedge-conditions.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function Evaluate(Of T) ( _
    validator As ConditionValidator(Of T), _
    condition As Boolean _
) As ConditionValidator(Of T)

### C#

public static ConditionValidator<> Evaluate<T>(
    ConditionValidator<> validator,
    bool condition
)

### Visual C++

public:
    generic<typename T>
    static ConditionValidator<>^ Evaluate(
        ConditionValidator<>^ validator,
        bool condition
)

### JavaScript

JavaScript does not support generic types or methods.

## Parameters

**validator**

Type: CuttingEdge.Conditions:::ConditionValidator<>(Of <(T)>)
The ConditionValidator<>(Of <(T)>) that holds the value that has to be checked.

**condition**

Type: System:::Boolean
true to prevent an Exception from being thrown; otherwise, false.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the condition equals false, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the condition equals false and the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel:::InvalidEnumArgumentException</code></td>
<td>Thrown when the condition equals false and the <code>Value</code> of the specified validator is an <code>Enum</code> type, while the specified</td>
</tr>
</tbody>
</table>
validator is created using the \texttt{Requires} extension method. Thrown when the condition equals false, while the specified validator is created using the \texttt{Ensures} extension method.

\texttt{CuttingEdge.Conditions}...; \texttt{PostconditionException}
See Also

ValidatorExtensions Class
Evaluate Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the specified condition equals true. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function Evaluate(Of T) ( _
    validator As ConditionValidator(Of T), _
    condition As Boolean, _
    conditionDescription As String _
) As ConditionValidator(Of T)
```

**C#**

```csharp
public static ConditionValidator<T> Evaluate<T>(
    ConditionValidator<T> validator,
    bool condition,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
    generic<typename T>
    static ConditionValidator<T>^ Evaluate( 
        ConditionValidator<T>^ validator,
        bool condition,
        String^ conditionDescription
    )
```

**JavaScript**

JavaScript does not support generic types or methods.

#### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions..:::ConditionValidator(Of <T>)(Of <T>=>)`
  - The `ConditionValidator(Of <T>)(Of <T>)` that holds the value that has to be checked.

- **condition**
Type: `System::Boolean`  
**true** to prevent an `Exception` from being thrown; otherwise, false.

**conditionDescription**

Type: `System::String`  
Describes the condition that should hold. i.e.: 'value should be valid'. When the description contains a `{0}` marker, that marker will be replaced with the actual name of the parameter. The description will be used in the message of the thrown exception.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the condition equals false, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the condition equals false and the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel..::.InvalidEnumArgumentException</td>
<td>Thrown when the condition equals false and the Value of the specified validator is an Enum type, while the specified</td>
</tr>
</tbody>
</table>
validator is created using the Requires extension method. Thrown when the condition equals false, while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
Evaluate Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions:::Evaluate(Of <(T)> ) Method (ConditionValidator(Of <(T)>), Expression(Of <(Func(Of <(T, Boolean)>)>)>))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the specified expression evaluates true on the given value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function Evaluate(Of T) ( _
    validator As CuttingEdge.Conditions.:::ConditionValidator(Of T), _
    expression As Expression(Of Func(Of T, Boolean))) _
) As CuttingEdge.Conditions.:::ConditionValidator(Of T)
```

#### C#

```csharp
public static ConditionValidator<T> Evaluate<T>(
    ConditionValidator<T> validator,
    Expression<Func<T, bool>> expression
)
```

#### Visual C++

```cpp
public:
    generic<typename T>
    static ConditionValidator<T>^ Evaluate(
        ConditionValidator<T>^ validator,
        Expression<Func<T, bool>>^ expression
    )
```

#### JavaScript

JavaScript does not support generic types or methods.

### Parameters

**validator**

- **Type:** `CuttingEdge.Conditions.:::ConditionValidator<(Of <(T)>)>`
- The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**expression**

- **Type:** `System.Linq.Expressions.:::Expression<(Of <(Func<(Of <(T, Boolean)>)>))>}`
The `Expression<Of <(TDelegate)>>()` that will be compiled to an `Func<Of <(T, TResult)>>()` and executed. When the expression is a null reference (Nothing in VB) it is considered to evaluate `false`. 
Type Parameters

T
The type of the Value of the specified validator.

ReturnValue
The specified validator instance.
Remarks

This method will display a string representation of the specified expression. Although it can therefore give a lot of useful information in the exception message, the expression has to be compiled on each call. Try using the other Evaluate<(Of <(T)>)(ConditionValidator<(Of <(T)>), Boolean)> overload in performance sensitive parts of your program.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the expression evaluated false or is a null reference, while the specified validator is created using the Requires extension method. Thrown when the expression evaluated false or is a null reference and the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the expression evaluated false or is a null reference and the Value of the specified validator is a null reference, while the specified validator is created using the Requires extension method. Thrown when the expression evaluated false or is a null reference and the Value of the</td>
</tr>
</tbody>
</table>
System.ComponentModel::InvalidEnumArgumentException specified validator is an Enum type, while the specified validator is created using the Requires extension method. Thrown when the expression evaluated false or is a null reference, while the specified validator is created using the Ensures extension method.

CuttingEdge.Conditions::PostconditionException
See Also

ValidatorExtensions Class
Evaluate Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the specified expression evaluates **true** on the given value. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function Evaluate(Of T) ( _
    validator As ConditionValidator(Of T), _
    expression As Expression(Of Func(Of T, Boolean)), _
    conditionDescription As String _) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> Evaluate<T>(
    ConditionValidator<T> validator,
    Expression<Func<T, bool>> expression,
    string conditionDescription
)

Visual C++

public:
    generic<typename T>
    static ConditionValidator<T>^ Evaluate(
        ConditionValidator<T>^ validator,
        Expression<Func<T, bool>>^ expression,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions::ConditionValidator(Of (Of<T>))
The ConditionValidator(Of (Of<T>)) that holds the value that has to be checked.

equation
Type: `System.Linq.Expressions::Expression<Of ^(Func<Of ^(T, Boolean^)>)>^>

The `Expression<Of ^(TDelegate^)>^>` that will be compiled to an `Func<Of ^(T, TResult^)>^>` and executed. When the expression is a null reference (Nothing in VB) it is considered to evaluate `false`.

`conditionDescription`

Type: `System::String`

Describes the condition that should hold. i.e.: 'value should be valid'. When the description contains a `{0}` marker, that marker will be replaced with the actual name of the parameter. The description will be used in the message of the thrown exception.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
Remarks

This method will display a string representation of the specified expression. Although it can therefore give a lot of useful information in the exception message, it the expression has to be compiled on each call. Try using the other Evaluate<(Of <T>))(ConditionValidator<(Of <T>)), Boolean) overload in performance sensitive parts of your program.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the expression evaluated false or is a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the expression evaluated false or is a null reference and the <strong>Value</strong> of the specified validator is a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
</tbody>
</table>
System.ComponentModel::InvalidEnumArgumentException specified validator is an Enum type, while the specified validator is created using the Requires extension method. Thrown when the expression evaluated false or is a null reference, while the specified validator is created using the Ensures extension method.

CuttingEdge.Conditions:::PostconditionException
See Also

- ValidatorExtensions Class
- Evaluate Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...::HasLength Method
ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HasLength(ConditionValidator&lt;(Of &lt;(String&gt;)&gt;, Int32)</strong></td>
<td>Checks whether the given value is equal in length to length. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><strong>HasLength&lt;(Of &lt;(TCollection&gt;)&gt;)(ConditionValidator&lt;(Of &lt;(TCollection&gt;)&gt;, Int32)</strong></td>
<td>Checks whether the given value has the number of elements as specified by numberOfElements. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td><strong>HasLength(ConditionValidator&lt;(Of &lt;(String&gt;)&gt;, Int32, String)</strong></td>
<td>Checks whether the given value is equal in length to length. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><strong>HasLength&lt;(Of &lt;(TCollection&gt;)&gt;)(ConditionValidator&lt;(Of &lt;(TCollection&gt;)&gt;, Int32, String)</strong></td>
<td>Checks whether the given value has the number of elements as specified by numberOfElements. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is equal in length to length. An exception is thrown otherwise. A null reference is considered to have a length of 0.

**Namespace:** [CuttingEdge.Conditions](http://CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function HasLength ( _
   validator As ConditionValidator(Of String), _
   length As Integer _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> HasLength(
   ConditionValidator<string> validator,
   int length
)

Visual C++

public:
static ConditionValidator<String^>^ HasLength(
   ConditionValidator<String^>^ validator,
   int length
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.hasLength = function(vali

Parameters

validator
   Type: CuttingEdge.Conditions..::.ConditionValidator(Of (String>)
   The ConditionValidator(Of (T>) that holds the value that has to be checked.

length
   Type: System..::.Int32
   The valid length.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the length of Value of the specified validator un equals length, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and length un equals 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the length of Value of the specified validator un equals length, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [HasLength Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions.:::HasLength Method (ConditionValidator<((String>), Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is equal in length to length. An exception is thrown otherwise. A null reference is considered to have a length of 0.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function HasLength (_
    validator As ConditionValidator(Of String), _
    length As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of String)
```

#### C#

```csharp
public static ConditionValidator<string> HasLength(
    ConditionValidator<string> validator,
    int length,
    string conditionDescription
)
```

#### Visual C++

```cpp
public:
static ConditionValidator<String^>^ HasLength(
    ConditionValidator<String^>^ validator,
    int length,
    String^ conditionDescription
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.hasLength = function(vali
```

### Parameters

**validator**
- **Type:** `CuttingEdge.Conditions..::.ConditionValidator<(Of <(String)>)>`
- The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**length**
Type: `System..::..Int32`
The valid length.

conditionDescription
Type: `System..::..String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the length of <code>Value</code> of the specified validator un equals length, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and length un equals 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the length of <code>Value</code> of the specified validator un equals length, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
HasLength Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

**ValidatorExtensions..:::HasLength(Of (TCollection>) Method**
(ConditionValidator(Of (TCollection>)>, Int32)

**ValidatorExtensions Class**  See Also  Send Feedback

Checks whether the given value has the number of elements as specified by numberOfElements. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function HasLength(Of TCollection As IEnumerable) ( _
  validator As ConditionValidator(Of TCollection), _
  numberOfElements As Integer _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> HasLength<TCollection>(
  ConditionValidator<TCollection> validator,
  int numberOfElements
)
where TCollection : IEnumerable

Visual C++

public:
  generic<typename TCollection>
  where TCollection : IEnumerable
  static ConditionValidator<TCollection>*^ HasLength(
    ConditionValidator<TCollection>*^ validator,
    int numberOfElements
  )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator<(Of
  <(TCollection)>)
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
numberOfElements

Type: `System..::.Int32`

The number of elements the collection should contain.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain the number of elements as specified with the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while <code>numberOfElements</code> is bigger than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator does not contain the number of elements as specified with the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[HasLength Overload]
[CuttingEdge.Contraints Namespace]

Send [feedback] on this topic to Microsoft.
Checks whether the given value has the number of elements as specified by `numberOfElements`. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function HasLength(Of TCollection As IEnumerable) ( _
    validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> HasLength<TCollection>(
    ConditionValidator<TCollection> validator,
    int numberOfElements,
    string conditionDescription
)

where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ HasLength(
        ConditionValidator<TCollection>^ validator,
        int numberOfElements,
        String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::;ConditionValidator<(Of
<(TCollection)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**numberOfElements**
- Type: `System::Int32`
- The number of elements the collection should contain.

**conditionDescription**
- Type: `System::String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator does not contain the number of elements as specified with the <em>numberOfElements</em> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator is a null reference, while <em>numberOfElements</em> is bigger than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator does not contain the number of elements as specified with the <em>numberOfElements</em> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
HasLength Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.:::IsEmpty Method
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsEmpty(ConditionValidator&lt;Of &lt;(String&gt;)&gt;)</code></td>
<td>Checks whether the given value is an Empty()() string. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEmpty&lt;Of &lt;(TCollection)&gt;)&gt;(ConditionValidator&lt;Of &lt;(TCollection)&gt;)&gt;)</code></td>
<td>Checks whether the given value contains no elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.</td>
</tr>
<tr>
<td><code>IsEmpty&lt;Of &lt;(String)&gt;)&gt;, String&gt;</code></td>
<td>Checks whether the given value is an Empty()() string. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEmpty&lt;Of &lt;(TCollection)&gt;)&gt;(ConditionValidator&lt;Of &lt;(TCollection)&gt;)&gt;)</code></td>
<td>Checks whether the given value contains no elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..:::IsEmpty Method (ConditionValidator<Of <(String)>>)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is an Empty()() string. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsEmpty ( _
    validator As ConditionValidator(Of String) _
) As ConditionValidator(Of String)

### C#

public static ConditionValidator<string> IsEmpty(
    ConditionValidator<string> validator
)

### Visual C++

public:
static ConditionValidator<String^>^ IsEmpty(
    ConditionValidator<String^>^ validator
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEmpty = function(

### Parameters

**validator**

Type: CuttingEdge.Conditions::<::ConditionValidator<(Of <(String)>)>>

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

### Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not empty, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not empty, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsEmpty Method (ConditionValidator<Of <(String)>>, String)

[ValidatorExtensions Class]  [See Also]  [Send Feedback]

Checks whether the given value is an Empty()() string. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEmpty ( _
    validator As ConditionValidator(Of String), _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsEmpty(
    ConditionValidator<string> validator,
    string conditionDescription
)

Visual C++

public: ConditionValidator<String^>^ IsEmpty(
    ConditionValidator<String^>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEmpty = function(validator, conditionDescription) {

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(String)>)> The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
    Type: System..::: String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is not empty, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is not empty, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsEmpty(Of (Of (TCollection>))>) Method
(ConditionValidator(Of (Of (TCollection>))>)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value contains no elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEmpty(Of TCollection As IEnumerable)( _
    validator As ConditionValidator(Of TCollection) _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsEmpty<TCollection>(
    ConditionValidator<TCollection> validator
)
where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ IsEmpty(
    ConditionValidator<TCollection>^ validator
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::..ConditionValidator<Of TCollection>

The ConditionValidator<Of T>(T) that holds the value that has to be checked.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator is not empty, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge:::Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is not empty, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsEmpty Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
Checks whether the given value contains no elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.

**Namespace:** [CuttingEdge.Conditions](https://www.cuttingedge-conditions.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEmpty(Of TCollection As IEnumerable) ( _
    validator As ConditionValidator(Of TCollection), _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

class IsEmpty<TCollection>
{
    public static ConditionValidator<TCollection> IsEmpty<TCollection>(
        ConditionValidator<TCollection> validator,
        string conditionDescription
    )
    where TCollection : IEnumerable

Visual C++

public:
    template<typename TCollection>
    static ConditionValidator<TCollection>^ IsEmpty(
        ConditionValidator<TCollection>^ validator,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator(Of (Of (Of TCollection)>)

The ConditionValidator(Of (Of (T)>)> that holds the value that has to be checked.
conditionDescription
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '0' for the ArgumentName.
## Type Parameters

**TCollection**

The type of the value to check.

### Return Value

The specified validator instance.
## Exceptions

**Exception**

| System:::ArgumentException |

Thrown when the **Value** of the specified validator is not empty, while the specified validator is created using the **Requires** extension method.

| CuttingEdge.Conditions:::PostconditionException |

Thrown when the **Value** of the specified validator is not empty, while the specified validator is created using the **Ensures** extension method.
See Also

ValidatorExtensions Class
IsEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
Include Protected Members  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsEqualTo Method

[ValidatorExtensions Class]  [See Also]  [Send Feedback]
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsEqualTo(ConditionValidator&lt;Of &lt;(Byte)&gt;&gt;, Byte)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsEqualTo(ConditionValidator&lt;Of &lt;(DateTime)&gt;&gt;, DateTime)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsEqualTo(ConditionValidator&lt;Of &lt;(Decimal)&gt;&gt;, Decimal)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsEqualTo(ConditionValidator&lt;Of &lt;(Double)&gt;&gt;, Double)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsEqualTo(ConditionValidator&lt;Of &lt;(Int16)&gt;&gt;, Int16)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsEqualTo(ConditionValidator&lt;Of &lt;(Int32)&gt;&gt;, Int32)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsEqualTo(ConditionValidator&lt;Of &lt;(Int64)&gt;&gt;, Int64)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsEqualTo&lt;Of &lt;(T)&gt;&gt; (ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;&gt;)&gt;&gt;, Nullable&lt;Of &lt;(T)&gt;&gt;)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Single()&gt;), Single)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;()</code> (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;()&gt;&gt;, T)</td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsEqualTo(ConditionValidator&lt;Of (Int64)&gt;, Int64, String)</code></td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;(ConditionValidator&lt;Of (Nullable&lt;Of &lt;(T)&gt;&gt;)&gt;, Nullable&lt;Of &lt;(T)&gt;&gt;, String)</code></td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;(ConditionValidator&lt;Of (Nullable&lt;Of &lt;(T)&gt;&gt;)&gt;, T, String)</code></td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo(ConditionValidator&lt;Of (Single)&gt;, Single, String)</code></td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsEqualTo&lt;Of &lt;(T)&gt;&gt;(ConditionValidator&lt;Of &lt;(T)&gt;&gt;, T, String)</code></td>
<td>Checks whether the given value is equal to the specified value. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Byte), _
    value As Byte )
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsEqualTo( 
    ConditionValidator<byte> validator, 
    byte value 
)

Visual C++

public: ConditionValidator<unsigned char>^ IsEqualTo( 
    ConditionValidator<unsigned char>^ validator, 
    unsigned char value 
)

JavaScript

CuttingEdge.Conditions.BuilderExtensions.isEqualTo = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<Of '<Byte'>>)
The ConditionValidator<Of '<T>'>) that holds the value that has to be checked.

value
    Type: System..:::Byte
    The valid value to compare with.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not equal to value, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not equal to value, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
 IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::..EqualTo Method (ConditionValidator<Of <(Byte)>), Byte, String)

[ValidatorExtensions Class]  [See Also]  [Send Feedback]

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Byte), _
    value As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsEqualTo(ConditionValidator<byte> validator, byte value, string conditionDescription)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsEqualTo(ConditionValidator<unsigned char>^ validator, unsigned char value, String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Byte)>)>
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

type

value
Type: `System::Byte`
The valid value to compare with.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsEqualTo Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](https://feedback.microsoft.com) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.EqualTo Method (ConditionValidator<Of <(DateTime)>, DateTime)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of DateTime), _
    value As DateTime _) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsEqualTo(
    ConditionValidator<DateTime> validator,
    DateTime value
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsEqualTo(
    ConditionValidator<DateTime>^ validator,
    DateTime value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(DateTime)>)
    The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

value
    Type: System..::.DateTime
    The valid value to compare with.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..:::.EqualTo Method (ConditionValidator<
(DateTime; DateTime), DateTime, String)

Checks whether the given value is equal to the specified value. An exception is
thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of DateTime), _
    value As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsEqualTo(
    ConditionValidator<DateTime> validator,
    DateTime value,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsEqualTo(
    ConditionValidator<DateTime>^ validator,
    DateTime^ value,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(DateTime)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: **System..:::DateTime**
The valid value to compare with.

*conditionDescription*
Type: **System..:::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td></td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
The `ValidatorExtensions::::IsEqualTo` method in the `ConditionValidator<Of <(Decimal)>, Decimal>` class checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Decimal), _
    value As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsEqualTo(ConditionValidator<decimal> validator, decimal value)

Visual C++

public: ConditionValidator<Decimal>^ IsEqualTo(ConditionValidator<Decimal>^ validator, Decimal value)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Decimal)=>))
The ConditionValidator<(Of <(T)=>)) that holds the value that has to be checked.

value
    Type: System..::.Decimal
    The valid value to compare with.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsEqualTo Method (ConditionValidator<Of <(Decimal)>, Decimal, String)

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Decimal), _
    value As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsEqualTo(
    ConditionValidator<decimal> validator,
    decimal value,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<Decimal>^ IsEqualTo(
        ConditionValidator<Decimal>^ validator,
        Decimal value,
        String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(validator,

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Decimal)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: `System::Decimal`
The valid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsEqualTo Method (ConditionValidator<Of <(Double)>>, Double)

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsEqualTo ( 
    validator As ConditionValidator(Of Double), _
    value As Double _
) As ConditionValidator(Of Double)

**C#**

public static ConditionValidator<
double> IsEqualTo(
    ConditionValidator<
double> validator,
    double value
)

**Visual C++**

public: ConditionValidator<double>^ IsEqualTo(
    ConditionValidator<double>^ validator,
    double value
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali;

**Parameters**

validator

Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Double)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value

Type: System..::: Double

The valid value to compare with.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsEqualTo Method (ConditionValidator(Of[Double]>), Double, String)

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _  
    validator As ConditionValidator(Of Double), _  
    value As Double, _  
    conditionDescription As String _  
) As ConditionValidator(Of Double)

#### C#

public static ConditionValidator<double> IsEqualTo(  
    ConditionValidator<double> validator,  
    double value,  
    string conditionDescription
)

#### Visual C++

public:  
static ConditionValidator<double>^ IsEqualTo(  
    ConditionValidator<double>^ validator,  
    double^ value,  
    String^ conditionDescription
)

#### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

### Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Double)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: `System::Double`
The valid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is not equal to value, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is not equal to value, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsEqualTo Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Short), _
    value As Short _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsEqualTo(
    ConditionValidator<short> validator,
    short value
)

Visual C++

public:
static ConditionValidator<short>^ IsEqualTo(
    ConditionValidator<short>^ validator,
    short value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(validator, value) {

Parameters

validator
    Type: CuttingEdge.Conditions.::: ConditionValidator(Of (Int16))
    The ConditionValidator(Of (T)) that holds the value that has to be checked.

value
    Type: System.::: Int16
    The valid value to compare with.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://www.example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Short), _
    value As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsEqualTo(
    ConditionValidator<short> validator,
    short value,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<short>* IsEqualTo(
    ConditionValidator<short>* validator,
    short value,
    String* conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions.::.ConditionValidator<(Of <(Int16)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: `System.Int16`
The valid value to compare with.

`conditionDescription`
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Integer), _
    value As Integer )
    As ConditionValidator(Of Integer)
```

### C#

```csharp
public static ConditionValidator<int> IsEqualTo(
    ConditionValidator<int> validator,
    int value)
```

### Visual C++

```cpp
public:
static ConditionValidator<int>^ IsEqualTo(
    ConditionValidator<int>^ validator,
    int value)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(validator, value) {
```

## Parameters

**validator**

Type: `CuttingEdge.Conditions...:::ConditionValidator<(Of <(Int32)>)>

The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**value**

Type: `System...:::Int32`

The valid value to compare with.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator is not equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <em>Value</em> of the specified validator is not equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Integer), _
    value As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

#### C#

public static ConditionValidator<int> IsEqualTo(
    ConditionValidator<int> validator,
    int value,
    string conditionDescription
)

#### Visual C++

public:
static ConditionValidator<int>^ IsEqualTo(
    ConditionValidator<int>^ validator,
    int^ value,
    String^ conditionDescription
)

#### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(validator

### Parameters

**validator**

Type: `CuttingEdge.Conditions...::ConditionValidator<(Of <(Int32)>)>`

The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**value**
Type: **System::Int32**
The valid value to compare with.

conditionDescription
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Long), _
    value As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsEqualTo(
    ConditionValidator<long> validator,
    long value
)

Visual C++

public:
static ConditionValidator<long long>^ IsEqualTo(
    ConditionValidator<long long>^ validator,
    long long value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<Of <(Int64)>>
The ConditionValidator<Of <(T)> > that holds the value that has to be checked.

value
Type: System..:::Int64
The valid value to compare with.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsEqualTo Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsEqualTo Method (ConditionValidator<Of <(Int64)>), Int64, String)

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Long), _
    value As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)
```

### C#

```csharp
public static ConditionValidator<long> IsEqualTo(
    ConditionValidator<long> validator,
    long value,
    string conditionDescription
)
```

### Visual C++

```cpp
public:
static ConditionValidator<long long>^ IsEqualTo(
    ConditionValidator<long long>^ validator,
    long long value,
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali
```
Type: `System::Int64`
The valid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsEqualTo Method (ConditionValidator<Of <(Single)>), Single)

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsEqualTo ( 
    validator As ConditionValidator(Of Single), _
    value As Single _
) As ConditionValidator(Of Single)

**C#**

public static ConditionValidator<float> IsEqualTo(
    ConditionValidator<float> validator,
    float value
)

**Visual C++**

public: static ConditionValidator<float>^ IsEqualTo(
    ConditionValidator<float>^ validator,
    float value
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(val;

**Parameters**

**validator**

Type: CuttingEdge.Conditions::ConditionValidator(Of (Of Single))->
The ConditionValidator(Of (Of T))-> that holds the value that has to be checked.

**value**

Type: System::Single
The valid value to compare with.
**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsEqualTo Method (ConditionValidator<Of <(Single)>), Single, String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo ( _
    validator As ConditionValidator(Of Single), _
    value As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsEqualTo(
    ConditionValidator<float> validator,
    float value,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsEqualTo(
    ConditionValidator<float>^ validator,
    float^ value,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isEqualTo = function(vali

Parameters

validator

Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Single)>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: `System::Single`
The valid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsEqualTo Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::EqualTo<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>), Nullable<(Of <(T)>)>))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsEqualTo(Of T As {Structure, New}) ( _  
  validator As ConditionValidator(Of Nullable(Of T)), _  
  value As Nullable(Of T) _  ) As ConditionValidator(Of Nullable(Of T))

**C#**

public static ConditionValidator<Nullable<T>> IsEqualTo<T>( 
  ConditionValidator<Nullable<T>> validator, 
  Nullable<T> value 
)

where T : struct, new()

**Visual C++**

public:
  generic<typename T>
  where T : value class, gcnew()
  static ConditionValidator<Nullable<T>>^ IsEqualTo( 
    ConditionValidator<Nullable<T>>^ validator, 
    Nullable<T> value 
)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of <(T))->))>)

The ConditionValidator<(Of <(T)->)> that holds the value that has to be checked.
value

Type: System::Nullable<Of <(T)>>

The valid value to compare with.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.EqualTo(Of (Of T)) Method
(ConditionValidator(Of Nullable(Of Nullable(Of (Of T))), Nullable(Of Nullable(Of (Of T)))), String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    value As Nullable(Of T), _
    conditionDescription As String
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsEqualTo<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> value,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsEqualTo(
        ConditionValidator<Nullable<T>>^ validator,
        Nullable<T>^ value,
        string^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

value
Type: System::Nullable<(Of <(T)>)
The valid value to compare with.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsEqualTo Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::EqualTo<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>>)>), T)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    value As T _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsEqualTo<T>(
    ConditionValidator<Nullable<T>> validator,
    T value
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsEqualTo(
        ConditionValidator<Nullable<T>>^ validator,
        T value
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of Nullable<(Of
    <(T)>))>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.
value
    Type: T
    The valid value to compare with.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- IsEqualTo Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    value As T, _
    conditionDescription As String
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsEqualTo<T>(
    ConditionValidator<Nullable<T>> validator,
    T value,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsEqualTo(
        ConditionValidator<Nullable<T>>^ validator,
        T value,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions::.ConditionValidator<(Of (Nullable<(Of
        <(T)>)))>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

value
Type: T
The valid value to compare with.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System...:::ArgumentException</td>
<td>Thrown when the <em>Value</em> of the specified validator is not equal to value, while the specified validator is created using the <em>Requires</em> extension method.</td>
</tr>
<tr>
<td>System...:::ArgumentNullException</td>
<td>Thrown when the <em>Value</em> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <em>Requires</em> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions...:::PostconditionException</td>
<td>Thrown when the <em>Value</em> of the specified validator is not equal to value, while the specified validator is created using the <em>Ensures</em> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](ValidatorExtensions_Class)
[IsEqualTo Overload](IsEqualTo_Overload)
[CuttingEdge.Conditions Namespace](CuttingEdge.Conditions_Namespace)

Send [feedback](feedback) on this topic to Microsoft.
Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    value As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsEqualTo<T>(
    ConditionValidator<T> validator,
    T value
)
where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
static ConditionValidator<T>^ IsEqualTo(
    ConditionValidator<T>^ validator,
    T value
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(T)>)>.
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: T
The valid value to compare with.
Type Parameters

T
   The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System.ComponentModel..::.InvalidEnumArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is an <strong>Enum</strong> type and not equal to value, while the</td>
</tr>
</tbody>
</table>
CuttingEdge.Conditions::PostconditionException

specified validator is created using the Requires extension method.
Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.EqualTo<Of <(T)>) Method
(ConditionValidator<Of <(T)>>, T, String)

Checks whether the given value is equal to the specified value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsEqualTo(Of T As IComparable)( _
    validator As ConditionValidator(Of T), _
    value As T, _
    conditionDescription As String _) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsEqualTo<T>(
    ConditionValidator<T> validator,
    T value,
    string conditionDescription
)

where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsEqualTo(
        ConditionValidator<T>^ validator,
        T value,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator(Of <T>)(Of<T>)
The ConditionValidator(Of <T>)(Of<T>) that holds the value that has to be checked.
value
   Type: T
   The valid value to compare with.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the
   placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the Value of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>System.ComponentModel..::.InvalidEnumArgumentException</strong></td>
<td>Thrown when the Value of the specified validator is an Enum type and not equal to value, while the</td>
</tr>
</tbody>
</table>
CuttingEdge.Conditions::PostconditionException

specified validator is created using the Requires extension method. Thrown when the Value of the specified validator is not equal to value, while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
IsEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
□  Include Protected Members
□  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:IsFalse Method

GraduatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsFalse(ConditionValidator&lt;(Of &lt;(Boolean)&gt;))</td>
<td>Checks whether the given value is <strong>false</strong>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsFalse(ConditionValidator&lt;(Of &lt;(Nullable&lt;(Of &lt;(Boolean)&gt;)&gt;)&gt;))</td>
<td>Checks whether the given value is <strong>false</strong>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsFalse(ConditionValidator&lt;(Of &lt;(Boolean)&gt;), String)</td>
<td>Checks whether the given value is <strong>false</strong>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsFalse(ConditionValidator&lt;(Of &lt;(Nullable&lt;(Of &lt;(Boolean)&gt;)&gt;)&gt;), String)</td>
<td>Checks whether the given value is <strong>false</strong>. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is **false**. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsFalse ( _
        validator As ConditionValidator(Of Boolean) _) As ConditionValidator(Of Boolean)

C#

public static ConditionValidator<bool> IsFalse(
        ConditionValidator<bool> validator
)

Visual C++

public: ConditionValidator<bool>^ IsFalse(
        ConditionValidator<bool>^ validator
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isFalse = function(validat

Parameters

validator
        Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Boolean)>)>)
        The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

Return Value

The specified validator instance.
Exceptions

Exception

System:::ArgumentException

Thrown when the Value of the specified validator is true, while the specified validator is created using the Requires extension method.

CuttingEdge.Conditions:::PostconditionException

Thrown when the Value of the specified validator is true, while the specified validator is created using the Ensures extension method.
See Also

- ValidatorExtensions Class
- IsFalse Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the given value is **false**. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsFalse ( _
    validator As ConditionValidator(Of Boolean), _
    conditionDescription As String _
) As ConditionValidator(Of Boolean)

C#

public static ConditionValidator<bool> IsFalse( 
    ConditionValidator<bool> validator,
    string conditionDescription
)

Visual C++

public: ConditionValidator<bool>^ IsFalse( 
    ConditionValidator<bool>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isFalse = function(validator,

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator(Of <(Boolean)>)
The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.

conditionDescription
    Type: System..::: String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is <strong>true</strong>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is <strong>true</strong>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsFalse Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is false. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsFalse ( _
    validator As ConditionValidator(Of Nullable(Of Boolean)) _
) As ConditionValidator(Of Nullable(Of Boolean))
```

**C#**

```csharp
public static ConditionValidator<Nullable<bool>> IsFalse(
    ConditionValidator<Nullable<bool>> validator
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<Nullable<bool>>^ IsFalse(
    ConditionValidator<Nullable<bool>>^ validator
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isFalse = function(validator)

Parameters

*validator*

Type: `CuttingEdge.Conditions.::;ConditionValidator<(Of < Nullable<(Of <(Boolean)>{})>)>)`

The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator is true or null, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is true or null, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsFalse Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is **false**. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](http://CuttingEdge.Conditions)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsFalse ( _
    validator As ConditionValidator(Of Nullable(Of Boolean)), _
    conditionDescription As String _
) As ConditionValidator(Of Nullable(Of Boolean))

C#

public static ConditionValidator<Nullable<bool>> IsFalse( 
    ConditionValidator<Nullable<bool>> validator,
    string conditionDescription
)

Visual C++

public: ConditionValidator<Nullable<bool>>^ IsFalse( 
    ConditionValidator<Nullable<bool>>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isFalse = function(validator, conditionDescription)

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of <(Boolean>)>)>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
Type: System..:::String
The description of the condition that should hold. The string may hold the
placeholder '{0}' for the {0}.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is <code>true</code> or null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is <code>true</code> or null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsFalse Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
□  Include Protected Members
□  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...::.IsGreaterOrEqual Method
ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsGreaterOrEqual(ConditionValidator(Of Byte), Byte)</td>
<td>Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsGreaterOrEqual(ConditionValidator(Of DateTime), DateTime)</td>
<td>Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsGreaterOrEqual(ConditionValidator(Of Decimal), Decimal)</td>
<td>Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsGreaterOrEqual(ConditionValidator(Of Double), Double)</td>
<td>Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsGreaterOrEqual(ConditionValidator(Of Int16), Int16)</td>
<td>Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsGreaterOrEqual(ConditionValidator(Of Int32), Int32)</td>
<td>Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
IsGreaterOrEqual<(Of <(T)>)>
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>, Nullable<(Of <(T)>)>)
IsGreaterOrEqual<(Of <(T)>)>
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>), T)
IsGreaterOrEqual(ConditionValidator<(Of <(Single)>)>, Single)
IsGreaterOrEqual<(Of <(T)>)>
(ConditionValidator<(Of <(T)>)>, T)
IsGreaterOrEqual(ConditionValidator<(Of <(Byte)>)>, Byte, String)
IsGreaterOrEqual(ConditionValidator<(Of <(DateTime)>)>, DateTime, String)
IsGreaterOrEqual(ConditionValidator<(Of <(Decimal)>)>, Decimal, String)
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

IsGreaterOrEqual(ConditionValidator<
(Of <(Double)?>), Double, String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

IsGreaterOrEqual(ConditionValidator<
(Of <(Int16)?>), Int16, String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

IsGreaterOrEqual(ConditionValidator<
(Of <(Int32)?>), Int32, String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

IsGreaterOrEqual(ConditionValidator<
(Of <(Int64)?>), Int64, String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

IsGreaterOrEqual<(Of <(T)?>)
(ConditionValidator<(Of <(Nullable<(Of
<(T)?)>)>), Nullable<(Of <(T)?)>),
String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

IsGreaterOrEqual<(Of <(T)?>)
(ConditionValidator<(Of <(Nullable<(Of
<(T)?)>)>), T, String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

IsGreaterOrEqual(ConditionValidator<
(Of <(Single?)>), Single, String)
IsNull
(IsGreaterOrEqual(Of (T)>)
(ConditionValidator(Of (Of (T)>)>, T, String)
value is greater or equal to the specified minValue. An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte ) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsGreaterOrEqual(
    ConditionValidator<byte> validator,
    byte minValue
)

Visual C++

public:
    static ConditionValidator<unsigned char>^ IsGreaterOrEqual(
        ConditionValidator<unsigned char>^ validator,
        unsigned char minValue
    )

JavaScript

CuttingEdge.Conditions.IdentifierExtensions.isGreaterOrEqual = function (validator,
    minValue)

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Byte)>)
The ConditionValidator(Of <(T)> that holds the value that has to be checked.

minValue
    Type: System..::.Byte
    The lowest valid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual (validator As ConditionValidator(Of Byte), minValue As Byte, conditionDescription As String) As ConditionValidator(Of Byte)

### C#

public static ConditionValidator<byte> IsGreaterOrEqual(ConditionValidator<byte> validator, byte minValue, string conditionDescription)

### Visual C++

public: static ConditionValidator<unsigned char>* IsGreaterOrEqual(ConditionValidator<unsigned char>* validator, unsigned char* minValue, String* conditionDescription)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function(

### Parameters

**validator**

- Type: `CuttingEdge.Conditions..:::ConditionValidator<(Of <(Byte)>)>`
- The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**minValue**
Type: `System::Byte`
The lowest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual (_
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsGreaterOrEqual(
    ConditionValidator<DateTime> validator,
    DateTime minValue
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsGreaterOrEqual(
    ConditionValidator<DateTime>^ validator,
    DateTime^ minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(DateTime)>)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
    Type: System..::: DateTime
    The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsGreaterOrEqual Method (ConditionValidator<Of <(DateTime)>, DateTime, String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsGreaterOrEqual (_
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)
```

**C#**

```csharp
public static ConditionValidator<DateTime> IsGreaterOrEqual(
    ConditionValidator<DateTime> validator,
    DateTime minValue,
    string conditionDescription
)
```

**Visual C++**

```csharp
public:
static ConditionValidator<DateTime>^ IsGreaterOrEqual( 
    ConditionValidator<DateTime>^ validator,
    DateTime^ minValue, 
    String^ conditionDescription 
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function ...
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions...:::ConditionValidator<Of <(DateTime)>>`

The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minValue**
Type: `System::DateTime`
The lowest valid value.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...::IsGreaterOrEqual Method (ConditionValidator<Of <(Decimal)>, Decimal)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal )
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsGreaterOrEqual(
    ConditionValidator<decimal> validator,
    decimal minValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsGreaterOrEqual(
    ConditionValidator<Decimal>^ validator,
    Decimal minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Decimal)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
    Type: System..:::Decimal
    The lowest valid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsGreaterOrEqual Method (ConditionValidator<Of 
<(Decimal)>, Decimal, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater or equal to the specified minValue. An
exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsGreaterOrEqual(
    ConditionValidator<decimal> validator,
    decimal minValue,
    string conditionDescription
)

Visual C++

public: ConditionValidator<decimal>^ IsGreaterOrEqual( 
    ConditionValidator<decimal>^ validator, 
    Decimal^ minValue, 
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = functi

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Decimal)>))
The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: **System::Decimal**
The lowest valid value.

**conditionDescription**
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::{:}ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::{:}PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic
C#
Visual C++
JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsGreaterOrEqual Method (ConditionValidator<Of<Double>>, Double)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsGreaterOrEqual(
    ConditionValidator<double> validator,
    double minValue
)

Visual C++

public:
static ConditionValidator<double>^ IsGreaterOrEqual(
    ConditionValidator<double>^ validator,
    double minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Double)>)
    The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
    Type: System..::.Double
    The lowest valid value.
**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsGreaterOrEqual Method (ConditionValidator<Of <(Double)>), Double, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsGreaterOrEqual(
    ConditionValidator<double> validator,
    double minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<double>^ IsGreaterOrEqual(
    ConditionValidator<double>^ validator,
    double^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = functi

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Double)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: `System::Double`  
The lowest valid value.

conditionDescription
Type: `System::String`  
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short _) As ConditionValidator(Of Short)
```

### C#

```csharp
public static ConditionValidator<short> IsGreaterOrEqual(ConditionValidator<short> validator, short minValue)
```

### Visual C++

```cpp
public: ConditionValidator<short>^ IsGreaterOrEqual(ConditionValidator<short>^ validator, short minValue)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function;
```

## Parameters

**validator**
- Type: `CuttingEdge.Conditions::ConditionValidator<Of (Int16)>`
- The `ConditionValidator<Of <T>>` that holds the value that has to be checked.

**minValue**
- Type: `System::Int16`
- The lowest valid value.
**Return Value**

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

* [ValidatorExtensions Class](#)
* [IsGreaterOrEqual Overload](#)
* [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions.:::IsGreaterOrEqual Method (ConditionValidator<Of <(Int16)>>, Int16, String)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual (_
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    conditionDescription As String _) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsGreaterOrEqual(
    ConditionValidator<short> validator,
    short minValue,
    string conditionDescription)

Visual C++

public:
static ConditionValidator<short>^ IsGreaterOrEqual(
    ConditionValidator<short>^ validator,
    short^ minValue,
    String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function
Type: `System::Int16`
The lowest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual (_
    validator As ConditionValidator(Of Integer), _
    minValue As Integer) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsGreaterOrEqual(
    ConditionValidator<int> validator,
    int minValue
)

Visual C++

public:
static ConditionValidator<int>^ IsGreaterOrEqual(
    ConditionValidator<int>^ validator,
    int minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(Int32)>>
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

minValue
    Type: System..::.Int32
    The lowest valid value.
**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::.::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Constraints::.::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

`ValidatorExtensions Class`
`IsGreaterOrEqual Overload`
`CuttingEdge.Conditions Namespace`

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsGreaterOrEqual( ConditionValidator<int> validator,
    int minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsGreaterOrEqual( ConditionValidator<int>^ validator,
    int^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int32)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: `System::Int32`
The lowest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue,</td>
</tr>
<tr>
<td></td>
<td>while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the Value of the specified validator is smaller than minValue,</td>
</tr>
<tr>
<td></td>
<td>while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsGreaterOrEqual Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Long), _
    minValue As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsGreaterOrEqual(ConditionValidator<long> validator, long minValue)

Visual C++

public: ConditionValidator<long long>^ IsGreaterOrEqual(ConditionValidator<long long>^ validator, long long minValue)

JavaScript

CuttingEdge.Conditions.NavigatorExtensions.isGreaterOrEqual = function

Parameters

validator
Type: CuttingEdge.Conditions:::ConditionValidator(Of (Int64))
The ConditionValidator(Of (T)) that holds the value that has to be checked.

minValue
Type: System:::Int64
The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic □ C#
□ Visual C++
□ JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsGreaterOrEqual Method (ConditionValidator<Of <(Int64 )>, Int64 , String))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsGreaterOrEqual (_
    validator As ConditionValidator(Of Long), _
    minValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)
```

### C#

```csharp
public static ConditionValidator<long> IsGreaterOrEqual(
    ConditionValidator<long> validator,
    long minValue,
    string conditionDescription
)
```

### Visual C++

```cpp
public:
static ConditionValidator<long long>^ IsGreaterOrEqual(
    ConditionValidator<long long>^ validator,
    long_long minValue,
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function
```

## Parameters

**validator**

Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Int64)>>)`

The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minValue**
Type: `System::Int64`
The lowest valid value.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>System::ArgumentOutOfRangeException</em></td>
<td>Thrown when the <em>Value</em> of the specified validator is smaller than <em>minValue</em>, while the specified validator is created using the <em>Requires</em> extension method.</td>
</tr>
<tr>
<td><em>CuttingEdge.Conditions::PostconditionException</em></td>
<td>Thrown when the <em>Value</em> of the specified validator is smaller than <em>minValue</em>, while the specified validator is created using the <em>Ensures</em> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single _
) As ConditionValidator(Of Single)

**C#**

public static ConditionValidator<float> IsGreaterOrEqual(
    ConditionValidator<float> validator,
    float minValue
)

**Visual C++**

public:
static ConditionValidator<float>^ IsGreaterOrEqual(
    ConditionValidator<float>^ validator,
    float minValue
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function

**Parameters**

**validator**
Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Single)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

**minValue**
Type: System...:::Single
The lowest valid value.
**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsGreaterOrEqual Method (ConditionValidator<Of <(Single)>>, Single, String)

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsGreaterOrEqual(ConditionValidator<float> validator, float minValue, string conditionDescription)

Visual C++

public:
static ConditionValidator<float>^ IsGreaterOrEqual(ConditionValidator<float>^ validator, float minValue, String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterOrEqual = function(...

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Single)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: `System::Single`
The lowest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _ minValue As Nullable(Of T) _) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterOrEqual<T>(ConditionValidator<Nullable<T>> validator, Nullable<T> minValue)

where T : struct, new()

Visual C++

public:

generic<typename T>

where T : value class, gcnew()

static ConditionValidator<Nullable<T>>^ IsGreaterOrEqual(ConditionValidator<Nullable<T>>^ validator, Nullable<T> minValue)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<Of (Nullable<Of <(T)>>)>

The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.
minValue
Type: System.Nullable<T>
The lowest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

Anchor:ValidatorExtensions Class
Anchor:IsGreaterOrEqual Overload
Anchor:CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsGreaterOrEqual(Of <(T)> ) Method
(ConditionValidator(Of Nullable(Of Nullable(Of (T)> ) ) ) , Nullable(Of Nullable(Of (T)> ) ) , String)

See Also  Send Feedback

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
minValue As Nullable(Of T), _
conditionDescription As String)
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterOrEqual<T>(ConditionValidator<Nullable<T>> validator,
Nullable<T> minValue,
string conditionDescription)

where T : struct, new()

Visual C++

public:
generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>::^ IsGreaterOrEqual(ConditionValidator<Nullable<T>>::^ validator,
Nullable<T>::^ minValue,
String^ conditionDescription)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<Of <(Nullable<Of <(T)>>)>)
The ConditionValidator<Of <(T)>> that holds the value that has to be
checked.

minValue
Type: System::Nullable<T>
The lowest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class  
IsGreaterOrEqual Overload  
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..:::IsGreaterOrEqual<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>))>, T)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
  minValue As T _) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterOrEqual<T>(
  ConditionValidator<Nullable<T>> validator,
  T minValue
)

where T : struct, new()

Visual C++

public:
  generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsGreaterOrEqual(
  ConditionValidator<Nullable<T>>^ validator,
  T minValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of (Nullable<(Of <(T)>))>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
minValue
  Type: T
  The lowest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _ minValue As T, _ conditionDescription As String) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterOrEqual<T>(ConditionValidator<Nullable<T>> validator, T minValue, string conditionDescription)

where T : struct, new()

Visual C++

public:
generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsGreaterOrEqual(ConditionValidator<Nullable<T>>^ validator, T minValue, String^ conditionDescription)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
   Type: CuttingEdge.Conditions..:::ConditionValidator<(Of Nullable<(Of T())))
The ConditionValidator<(Of (T())) that holds the value that has to be
checked.

minValue
  Type: T
  The lowest valid value.

conditionDescription
  Type: `System.String`
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**  
**IsGreaterOrEqual Overload**  
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:) on this topic to Microsoft.
ValidatorExtensions.IsGreaterOrEqual<(Of <(T)>)> Method
(ConditionValidator<(Of <(T)>)>, T)

Checks whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsGreaterOrEqual(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T _) As ConditionValidator(Of T)
```

**C#**

```csharp
public static ConditionValidator<T> IsGreaterOrEqual<T>(
    ConditionValidator<T> validator,
    T minValue
)
```

where T : IComparable

**Visual C++**

```cpp
public:
    generic<typename T>
where T : IComparable
static ConditionValidator<T>^ IsGreaterOrEqual(
    ConditionValidator<T>^ validator,
    T minValue
)
```

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

**validator**

Type: `CuttingEdge.Conditions..:::ConditionValidator<(Of <(T)>)>`

The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**minValue**
Type: T
The lowest valid value.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel:::InvalidEnumArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code> and is an <code>Enum</code> type, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
</tbody>
</table>
validator is created using the **Ensures** extension method.
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the given value is greater or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsGreaterOrEqual(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)
```

**C#**

```csharp
public static ConditionValidator<T> IsGreaterOrEqual<T>(
    ConditionValidator<T> validator,
    T minValue,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
    template<typename T>
    static ConditionValidator<T>^ IsGreaterOrEqual(
        ConditionValidator<T>^ validator,
        T minValue,
        String^ conditionDescription
    )
```

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

**validator**

Type: **CuttingEdge.Conditions::<:: ConditionValidator<(Of <(T)>)>**

The **ConditionValidator<(Of <(T)>)>** that holds the value that has to be checked.
minValue
Type: T
The lowest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Requires</strong> extension method. Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue and is an <strong>Enum</strong> type, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System.ComponentModel..::.InvalidEnumArgumentException</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td></td>
</tr>
</tbody>
</table>
validator is created using the \texttt{Ensures} extension method.
See Also

ValidatorExtensions Class
IsGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsGreaterThan Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsGreaterThan(ConditionValidator&lt;Of &lt;(Byte)&gt;, Byte)</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsGreaterThan(ConditionValidator&lt;Of &lt;(DateTime)&gt;, DateTime)</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsGreaterThan(ConditionValidator&lt;Of &lt;(Decimal)&gt;, Decimal)</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsGreaterThan(ConditionValidator&lt;Of &lt;(Double)&gt;, Double)</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsGreaterThan(ConditionValidator&lt;Of &lt;(Int16)&gt;, Int16)</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsGreaterThan(ConditionValidator&lt;Of &lt;(Int32)&gt;, Int32)</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsGreaterThan(ConditionValidator&lt;Of &lt;(Int64)&gt;, Int64)</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsGreaterThan&lt;Of &lt;(T)&gt;, (ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;, Nullable&lt;Of &lt;(T)&gt;)&gt;))&gt;</code></td>
<td>Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
IsGreaterThan(Of <(T)>)>
(ConditionValidator(Of (Nullable(Of <(T)>)>), T)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(ConditionValidator(Of <(Single)<>), Single)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(Of <(T)>)>
(ConditionValidator(Of <(T)<>), T)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(ConditionValidator(Of <(Byte)<>), Byte, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(ConditionValidator(Of <(DateTime)<>), DateTime, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(ConditionValidator(Of <(Decimal)<>), Decimal, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(Of <(Double)<>), Double, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(ConditionValidator(Of <(Int16)<>), Int16, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(ConditionValidator(Of <(Int32)<>), Int32, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.
IsGreaterThan(ConditionValidator<(Of <(Int64)>), Int64, String) is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan<(Of <(T)>)> (ConditionValidator<(Of <(Nullable<(Of <(T)>)>), Nullable<(Of <(T)>), String)) Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan<(Of <(T)>)> (ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>), T, String) Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan(ConditionValidator<(Of <(Single)>), Single, String) Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

IsGreaterThan<(Of <(T)>)> (ConditionValidator<(Of <(T)>), T, String) Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conitions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsGreaterThan Method (ConditionValidator<Of (Byte)>, Byte)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte _) As ConditionValidator(Of Byte)

C#

public static ConditionValidator< byte> IsGreaterThan(ConditionValidator< byte> validator, byte minValue)

Visual C++

public: ConditionValidator<unsigned char>^ IsGreaterThan(ConditionValidator<unsigned char>^ validator, unsigned char minValue)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Byte)>)
The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

minValue
    Type: System..::.Byte
    The highest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions....:..IsGreaterThan Method (ConditionValidator(Of Byte), Byte, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

C#

```}

```csharp
public static ConditionValidator<Byte> IsGreaterThan(
    ConditionValidator<Byte> validator,
    byte minValue,
    string conditionDescription
)

Visual C++

```cpp
public:
static ConditionValidator<unsigned char>^ IsGreaterThan(
    ConditionValidator<unsigned char>^ validator,
    unsigned char minValue,
    String^ conditionDescription
)

JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

**validator**

Type: `CuttingEdge.Conditions.::.ConditionValidator<Of <(Byte)>)`
The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minValue**
Type: `System::Byte`
The highest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentException`.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.argumentoutofrangeexception">Requires extension method</a>.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <a href="https://docs.microsoft.com/en-us/dotnet/api/cuttingedge.conditions.postconditionexception">Ensures extension method</a>.</td>
</tr>
</tbody>
</table>
See Also

- `ValidatorExtensions Class`
- `IsGreaterThan Overload`
- `CuttingEdge.Conditions Namespace`

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..:::IsGreaterThan Method (ConditionValidator<(Of <(DateTime)>), DateTime)

**ValidatorExtensions Class**  See Also  Send Feedback

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsGreaterThan ( _  
    validator As ConditionValidator(Of DateTime), _  
    minValue As DateTime _  
) As ConditionValidator(Of DateTime)

**C#**

public static ConditionValidator<DateTime> IsGreaterThan(  
    ConditionValidator<DateTime> validator,  
    DateTime minValue  
)

**Visual C++**

public: ConditionValidator<DateTime>^ IsGreaterThan(  
    ConditionValidator<DateTime>^ validator,  
    DateTime^ minValue  
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

**Parameters**

**validator**  
Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(DateTime)>>)  
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

**minValue**  
Type: System..::: DateTime  
The highest invalid value.
**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- `ValidatorExtensions Class`
- `IsGreaterThan Overload`
- `CuttingEdge.Conditions Namespace`

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..:::IsGreaterThan Method (ConditionValidator<(Of <(DateTime)>), DateTime, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsGreaterThan(
    ConditionValidator<DateTime> validator,
    DateTime minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsGreaterThanOrEqual(
    ConditionValidator<DateTime>^ validator,
    DateTime^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(DateTime)>)>)
The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

minValue
Type: `System::DateTime`
The highest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsGreaterThan(
    ConditionValidator<decimal> validator,
    decimal minValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsGreaterThan(
    ConditionValidator<Decimal>^ validator,
    Decimal minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Decimal)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
    Type: System...:::Decimal
    The highest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to</td>
</tr>
<tr>
<td></td>
<td><code>minValue</code>, while the specified validator is created using the <code>Requires</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to</td>
</tr>
<tr>
<td></td>
<td><code>minValue</code>, while the specified validator is created using the <code>Ensures</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsGreaterThan Method (ConditionValidator<Of <(Decimal)>, Decimal, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsGreaterThan(
    ConditionValidator<decimal> validator,
    decimal minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsGreaterThan(
    ConditionValidator<Decimal>^ validator,
    Decimal^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function("}

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Decimal)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: `System.Decimal`
The highest invalid value.

**conditionDescription**
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**
The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- `ValidatorExtensions Class`
- `IsGreaterThan Overload`
- `CuttingEdge.Conditions Namespace`

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsGreaterThan Method (ConditionValidator<Of (Double)>, Double)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan( _
    validator As ConditionValidator(Of Double), _
    minValue As Double _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsGreaterThan(  
    ConditionValidator<double> validator,  
    double minValue
)

Visual C++

public:  
static ConditionValidator<double>^ IsGreaterThan(  
    ConditionValidator<double>^ validator,  
    double minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Double)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
    Type: System..::.Double
    The highest invalid value.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [IsGreaterThan Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](mailto:) on this topic to Microsoft.
ValidatorExtensions.::.IsGreaterThan Method (ConditionValidator<Of <(Double)>>, Double, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsGreaterThan( 
    ConditionValidator<double> validator, 
    double minValue, 
    string conditionDescription 
)

Visual C++

public: static ConditionValidator<double>^ IsGreaterThan( 
    ConditionValidator<double>^ validator, 
    double^ minValue, 
    String^ conditionDescription 
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Double)>))
    The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: `System::Double`
The highest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsGreaterThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions.IsGreaterThan Method (ConditionValidator<Of (Int16)>, Int16)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsGreaterThan(
    ConditionValidator<short> validator,
    short minValue
)

Visual C++

public:
    static ConditionValidator<short>^ IsGreaterThan(
        ConditionValidator<short>^ validator,
        short minValue
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions.:::ConditionValidator<Of <(Int16)>>)
The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.

minValue
    Type: System.:::Int16
    The highest invalid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsGreaterThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions.IsGreaterThan Method (ConditionValidator<Of <(Int16)>, Int16, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsGreaterThan(
    ConditionValidator<short> validator,
    short minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<short>^ IsGreaterThan(
    ConditionValidator<short>^ validator,
    short^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int16)>))
    The ConditionValidator(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: `System.Int16`
The highest invalid value.

`conditionDescription`
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::..IsGreaterThan Method (ConditionValidator<Of
<Int32>), Int32)

Checks whether the given value is greater than the specified minValue. An
exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer )
) As ConditionValidator(Of Integer)
```

### C#

```csharp
public static ConditionValidator<int> IsGreaterThan(
    ConditionValidator<int> validator,
    int minValue
)
```

### Visual C++

```cpp
public:
static ConditionValidator<int>^ IsGreaterThan(
    ConditionValidator<int>^ validator,
    int minValue
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(
```

### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Int32)>>`
  - The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

- **minValue**
  - Type: `System..:::Int32`
  - The highest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsGreaterThan(
    ConditionValidator<int> validator,
    int minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsGreaterThanOrEqual(        
    ConditionValidator<int>^ validator,
    int^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int32)>)>  
The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

minValue
Type: \texttt{System::Int32}
The highest invalid value.

\textbf{conditionDescription}
Type: \texttt{System::String}
The description of the condition that should hold. The string may hold the placeholder '{0}' for the \texttt{ArgumentName}.

\textbf{Return Value}

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::GreaterThan Method (ConditionValidator<Of (Int64)>, Int64)

ValidatorExtensions Class, See Also, Send Feedback

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Long), _
    minValue As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsGreaterThan(
    ConditionValidator<long> validator,
    long minValue
)

Visual C++

public:
static ConditionValidator<long long>^ IsGreaterThanOrEqualTo(
    ConditionValidator<long long>^ validator,
    long long minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<Of <(Int64)>>
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

minValue
    Type: System..::: Int64
    The highest invalid value.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsGreaterThan Method (ConditionValidator<Of (Int64)>, Int64, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Long), _
    minValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)
```

**C#**

```csharp
public static ConditionValidator<long> IsGreaterThan(
    ConditionValidator<long> validator,
    long minValue,
    string conditionDescription
)
```

**Visual C++**

```c++
public:
static ConditionValidator<long long>^ IsGreaterThan(
    ConditionValidator<long long>^ validator,
    long long minValue,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

**Parameters**

**validator**

Type: `CuttingEdge.Conditions...::ConditionValidator<(Of <(Int64)>))`

The `ConditionValidator<(Of <(T)>)` that holds the value that has to be checked.

**minValue**
Type: System::Int64
The highest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**  
**IsGreaterThan Overload**  
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions:::IsGreaterThan Method (ConditionValidator<Of <(Single)>>, Single)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsGreaterThan ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single _
) As ConditionValidator(Of Single)
```

### C#

```csharp
public static ConditionValidator<float> IsGreaterThan(
    ConditionValidator<float> validator,
    float minValue
)
```

### Visual C++

```cpp
public:
static ConditionValidator<float>^ IsGreaterThan(
    ConditionValidator<float>^ validator,
    float minValue
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(
```

## Parameters

- **validator**
  - Type: `CuttingEdge.Conditions..:::ConditionValidator(Of <(Single)>)`
  - The `ConditionValidator(Of <(T)>)` that holds the value that has to be checked.

- **minValue**
  - Type: `System..:::Single`
  - The highest invalid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to <strong>minValue</strong>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to <strong>minValue</strong>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsGreaterThan Method (ConditionValidator<Of <(Single)>>, Single, String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan ( _
   validator As ConditionValidator(Of Single), _
   minValue As Single, _
   conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsGreaterThan(ConditionValidator<float> validator,
   float minValue,
   string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsGreaterThan(ConditionValidator<float>^ validator,
   float^ minValue,
   String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isGreaterThan = function(

Parameters

validator
   Type: CuttingEdge.Conditions...:::ConditionValidator<Of <(Single)>>)
The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.

minValue
Type:  \texttt{System::Single}
The highest invalid value.

\texttt{conditionDescription}
Type:  \texttt{System::String}
The description of the condition that should hold. The string may hold the placeholder '{0}' for the \texttt{ArgumentName}.

\textbf{Return Value}

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As Nullable(Of T) _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> minValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>> IsGreaterThan(
        ConditionValidator<Nullable<T>> validator,
        Nullable<T> minValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<Of <>Nullable<Of <(T)>>)>

The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.
minValue
Type: `System::Nullable<T>`
The highest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to</td>
</tr>
<tr>
<td></td>
<td><code>minValue</code>, while the specified validator is created using the <code>Requires</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td><code>minValue</code>, while the specified validator is created using the <code>Ensures</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::.IsGreaterThan<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>), Nullable<(Of <(T)>)>), String)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** cuttingEdge. Conditions

**Assembly:** CuttingEdge. Conditions (in CuttingEdge. Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan(Of T As {Structure, New}) ( _
  validator As ConditionValidator(Of Nullable(Of T)), _
  minValue As Nullable(Of T), _
  conditionDescription As String) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterThan<T>(
  ConditionValidator<Nullable<T>> validator,
  Nullable<T> minValue,
  string conditionDescription
)
where T : struct, new()

Visual C++

public:
  generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsGreaterThan(
  ConditionValidator<Nullable<T>>^ validator,
  Nullable<T>^ minValue,
  String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of
<(T)>)>)>)>
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

minValue
Type: System::Nullable<T>
The highest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..:::.IsGreaterThan<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>), T)

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T _) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterThan<T>(
    ConditionValidator<Nullable<T>> validator,
    T minValue)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static conditionValidator<Nullable<T>>^ IsGreaterThan(  
        conditionValidator<Nullable<T>>^ validator,
        T minValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of (Nullable<(Of (T))>)>)

The ConditionValidator<(Of (T)>) that holds the value that has to be checked.
minValue
  Type: T
  The highest invalid value.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions::IsGreaterThan(Of (T)) Method (ConditionValidator(Of Nullable(Of (T))), T, String)

See Also: ValidatorExtensions Class

Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsGreaterThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> minValue,
    string conditionDescription
)
where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>> IsGreaterThan(
        ConditionValidator<Nullable<T>> validator,
        Nullable<T> minValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions::.ConditionValidator<(Of Nullable<(Of T))>
    The ConditionValidator<(Of (T))> that holds the value that has to be
checked.

minValue
   Type: T
   The highest invalid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Type Parameters**

T

The type of the Value of the specified validator.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsGreaterThan Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
ValidatorExtensions.IsGreaterThan(Of T) Method
(ConditionValidator(Of T), T)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan(Of T As IComparable)( _
  validator As ConditionValidator(Of T), _
  minValue As T _) _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsGreaterThan<T>(
  ConditionValidator<T> validator,
  T minValue
)

where T : IComparable

Visual C++

public:
  generic<typename T>
  where T : IComparable
  static ConditionValidator<T>^ IsGreaterThan(
    ConditionValidator<T>^ validator,
    T minValue
  )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(T)>)>
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: T
The highest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel..::.InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue and is an Enum type, while the specified validator is created using the Requires extension method.</td>
</tr>
</tbody>
</table>

Thrown when the Value of the specified validator is smaller or equal to minValue,
while the specified validator is created using the `Ensures` extension method.
See Also

[ValidatorExtensions Class](#)
[IsGreaterThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Checks whether the given value is greater than the specified minValue. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsGreaterThan(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsGreaterThan<T>(
    ConditionValidator<T> validator, 
    T minValue, 
    string conditionDescription
)
where T : IComparable

Visual C++

public:
    generic<typename T>
where T : IComparable
static ConditionValidator<T>^ IsGreaterThan( 
    ConditionValidator<T>^ validator, 
    T minValue, 
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator(Of <T>))
The ConditionValidator(Of <T>)) that holds the value that has to be checked.
minValue
Type: T
The highest invalid value.

conditionDescription
Type: System..:::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue and is an <strong>Enum</strong> type, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue,</td>
</tr>
</tbody>
</table>
while the specified validator is created using the \texttt{Ensures} extension method.
See Also

ValidatorExtensions Class
IsGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...::IsInfinity Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsInfinity(ConditionValidator&lt;(Of Double&gt;)&gt;)</td>
<td>Checks whether the given value is infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsInfinity(ConditionValidator&lt;(Of Single&gt;)&gt;)</td>
<td>Checks whether the given value is infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsInfinity(ConditionValidator&lt;(Of Double), String&gt;)</td>
<td>Checks whether the given value is infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsInfinity(ConditionValidator&lt;(Of Single), String&gt;)</td>
<td>Checks whether the given value is infinity. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[ValidatorExtensions Members]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Checks whether the given value is infinity. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInfinity ( _
  validator As ConditionValidator(Of Double) _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsInfinity(
  ConditionValidator<double> validator
)

Visual C++

public:
  static ConditionValidator<double>^ IsInfinity(
    ConditionValidator<double>^ validator
  )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInfinity = function(val

Parameters

validator
  Type: CuttingEdge.Conditions::ConditionValidator(Of Double)
  The ConditionValidator(Of Double) that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsInfinity (_
    validator As ConditionValidator(Of Double), _
    conditionDescription As String _
) As ConditionValidator(Of Double)

**C#**

```csharp
public static ConditionValidator<
double> IsInfinity(
    ConditionValidator<
double> validator,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<
double>^ IsInfinity(
    ConditionValidator<
double>^ validator,
    String^ conditionDescription
)
```

**JavaScript**

```
CuttingEdge.Conditions.ValidatorExtensions.isInfinity = function(val)
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions:::ConditionValidator<Of <(Double)>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**conditionDescription**
- Type: `System:::String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Image" /></td>
<td>Thrown when the <code>Value</code> of the specified validator is not infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentException</code></td>
<td></td>
</tr>
<tr>
<td><img src="image.png" alt="Image" /></td>
<td>Thrown when the <code>Value</code> of the specified validator is not infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is infinity. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInfinity ( _
    validator As ConditionValidator(Of Single) _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsInfinity(
    ConditionValidator<float> validator
)

Visual C++

public:
static ConditionValidator<float>^ IsInfinity(
    ConditionValidator<float>^ validator
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInfinity = function(val

Parameters

validator
Type: CuttingEdge.Conditions::<::ConditionValidator<Of <(Single)>>)
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not infinity, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not infinity, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is infinity. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInfinity ( _
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsInfinity(  
    ConditionValidator<float> validator,  
    string conditionDescription
)

Visual C++

public: ConditionValidator<float>^ IsInfinity(  
    ConditionValidator<float>^ validator,  
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInfinity = function(va

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of (Single)>)
    The ConditionValidator(Of (T)>) that holds the value that has to be checked.

conditionDescription
    Type: System..::.String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is not infinity, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is not infinity, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript  Include Protected Members  Include Inherited Members  CuttingEdge.Conditions reference library  ValidatorExtensions:::InRange Method  ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsInRange(ConditionValidator&lt;(Of &lt;(Byte)?&gt;), Byte, Byte)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;(Of &lt;(DateTime)?&gt;), DateTime, DateTime)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;(Of &lt;(Decimal)?&gt;), Decimal, Decimal)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;(Of &lt;(Double)?&gt;), Double, Double)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;(Of &lt;(Int16)?&gt;), Int16, Int16)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;(Of &lt;(Int32)?&gt;), Int32, Int32)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;(Of &lt;(Int64)?&gt;), Int64, Int64)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange&lt;(Of &lt;(T)&gt;), (ConditionValidator&lt;(Of &lt;(Nullable&lt;(Of &lt;(T)&gt;)&gt;))&gt;, Nullable&lt;(Of &lt;(T)&gt;))&gt;</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Nullable&lt;Of &lt;(T)&gt;&gt;)</td>
<td>exception is thrown otherwise. Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsInRange&lt;Of &lt;(T)&gt;&gt;) (ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;&gt;)&gt;), T, T)</td>
<td>IsInRange(ConditionValidator&lt;Of &lt;(Single)&gt;), Single, Single)</td>
</tr>
<tr>
<td>IsInRange&lt;Of &lt;(T)&gt;&gt;) (ConditionValidator&lt;Of &lt;(T)&gt;), T, T)</td>
<td>IsInRange&lt;Of &lt;(Byte)&gt;), Byte, Byte, String)</td>
</tr>
<tr>
<td>IsInRange(ConditionValidator&lt;Of &lt;(DateTime)&gt;), DateTime, DateTime, String)</td>
<td>IsInRange(ConditionValidator&lt;Of &lt;(Decimal)&gt;), Decimal, Decimal, String)</td>
</tr>
<tr>
<td>IsInRange(ConditionValidator&lt;Of &lt;(Double)&gt;), Double, Double, String)</td>
<td>IsInRange(ConditionValidator&lt;Of &lt;(Int16)&gt;), Int16, Int16, String)</td>
</tr>
<tr>
<td>IsInRange(ConditionValidator&lt;Of &lt;(Int32)&gt;), Int32, Int32, String)</td>
<td>IsInRange&lt;Of &lt;(T)&gt;&gt;) (ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;&gt;)&gt;), T, T)</td>
</tr>
<tr>
<td>Signature</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;Of ((Int64)&gt;)&gt;, Int64, Int64, String)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange&lt;Of &lt;(T)&gt;&gt;() (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;&gt;()&gt;, Nullable&lt;Of &lt;(T)&gt;&gt;, Nullable&lt;Of &lt;(T)&gt;&gt;, String)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange&lt;Of &lt;(T)&gt;&gt;() (ConditionValidator&lt;Of Nullable&lt;Of &lt;(T)&gt;&gt;()&gt;, T, T, String)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange(ConditionValidator&lt;Of &lt;(Single)&gt;&gt;&gt;&gt;, Single, Single, String)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsInRange&lt;Of &lt;(T)&gt;&gt;() (ConditionValidator&lt;Of &lt;(T)&gt;&gt;, T, T, String)</code></td>
<td>Checks whether the given value is between <code>minValue</code> and <code>maxValue</code> (including those values). An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)  
**Assembly:**  [CuttingEdge.Conditions](#) (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsInRange (validator As ConditionValidator(Of Byte), minValue As Byte, maxValue As Byte) As ConditionValidator(Of Byte)
```

**C#**

```csharp
public static ConditionValidator<byte> IsInRange(ConditionValidator<byte> validator, byte minValue, byte maxValue)
```

**Visual C++**

```cpp
public:
static ConditionValidator<unsigned char>^ IsInRange(ConditionValidator<unsigned char>^ validator, unsigned char minValue, unsigned char maxValue)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(validator, minValue, maxValue)
```

**Parameters**

- **validator**
  - Type: `ConditionValidator<Of (Of Byte)>`
  - The `ConditionValidator<Of (Of T)>` that holds the value that has to be checked.

- **minValue**
Type: \texttt{System::Byte}
The lowest valid value.

\texttt{maxValue}
Type: \texttt{System::Byte}
The highest valid value.

\textbf{Return Value}

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...::InRange Method (ConditionValidator<Of <(Byte)>>, Byte, Byte, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte, _
    maxValue As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsInRange(
    ConditionValidator<byte> validator,
    byte minValue,
    byte maxValue,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<unsigned char>* IsInRange( 
        ConditionValidator<unsigned char>* validator, 
        unsigned char minValue, 
        unsigned char maxValue, 
        String* conditionDescription 
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(validat
minValue
   Type: System::.Byte
   The lowest valid value.

maxValue
   Type: System::.Byte
   The highest valid value.

conditionDescription
   Type: System::.String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
</tr>
</tbody>
</table>

### Condition
Thrown when the `Value` of the specified validator is not in the specified range, while the specified validator is created using the `Requires` extension method. Thrown when the `Value` of the specified validator is not in the specified range, while the specified validator is created using the `Ensures` extension method.
See Also

[ValidatorExtensions Class]
[IsInRange Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
ValidatorExtensions..::..InRange Method (ConditionValidator<Of <(DateTime)>, DateTime, DateTime)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime, _
    maxValue As DateTime _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsInRange(
    ConditionValidator<DateTime> validator,
    DateTime minValue,
    DateTime maxValue
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsInRange(
    ConditionValidator<DateTime>^ validator,
    DateTime^ minValue,
    DateTime^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(validator, minValue, maxValue)

Parameters

validator
    Type: CuttingEdge.Conditions...::ConditionValidator<(Of <(DateTime)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: System::DateTime
The lowest valid value.

maxValue
Type: System::DateTime
The highest valid value.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::InRange Method (ConditionValidator<Of <(DateTime)>, DateTime, DateTime, String)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
validator As ConditionValidator(Of DateTime), _
minValue As DateTime, _
maxValue As DateTime, _
conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsInRange( 
ConditionValidator<DateTime> validator, 
DateTime minValue, 
DateTime maxValue, 
string conditionDescription 
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsInRange( 
ConditionValidator<DateTime>^ validator, 
DateTime^ minValue, 
DateTime^ maxValue, 
String^ conditionDescription 
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali

Parameters

validator

Type: CuttingEdge.Conditions.:::ConditionValidator<(Of ((DateTime))>)
The ConditionValidator<(Of (T)>) that holds the value that has to be checked.
minValue
  Type: System::DateTime
  The lowest valid value.

maxValue
  Type: System::DateTime
  The highest valid value.

conditionDescription
  Type: System::String
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::..ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::..PostconditionException</td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#  □  Visual C++  □  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::..InRange Method (ConditionValidator<Of
<Decimal>>, Decimal, Decimal)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    maxValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsInRange(
    ConditionValidator<decimal> validator,
    decimal minValue,
    decimal maxValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsInRange(
    ConditionValidator<Decimal>^ validator,
    Decimal minValue,
    Decimal maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions...::ConditionValidator<(Of <(Decimal)>))
    The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: `System::Decimal`
The lowest valid value.

`maxValue`
Type: `System::Decimal`
The highest valid value.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <em>Value</em> of the specified validator is not in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <em>Value</em> of the specified validator is not in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::InRange Method (ConditionValidator<Of<br>(Decimal)>, Decimal, Decimal, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    maxValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsInRange(
    ConditionValidator<decimal> validator,
    decimal minValue,
    decimal maxValue,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<Decimal>^ IsInRange(
        ConditionValidator<Decimal>^ validator,
        Decimal^ minValue,
        Decimal^ maxValue,
        String^ conditionDescription
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(validator, minValue,...

Parameters

validator
    Type: CuttingEdge.Conditions.:::ConditionValidator<(Of <(Decimal)>)>
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
minValue
   Type: System::Decimal
   The lowest valid value.

maxValue
   Type: System::Decimal
   The highest valid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::InRange Method (ConditionValidator<Of <(Double)>), Double, Double)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    maxValue As Double)
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsInRange (ConditionValidator<double> validator,
    double minValue,
    double maxValue
)

Visual C++

public:
static ConditionValidator<double>^ IsInRange (ConditionValidator<double>^ validator,
    double^ minValue,
    double^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Double)>)
The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

minValue
Type: `System::Double`
The lowest valid value.

`maxValue`
Type: `System::Double`
The highest valid value.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    maxValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsInRange(
    ConditionValidator<double> validator,
    double minValue,
    double maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<double>^ IsInRange(
    ConditionValidator<double>^ validator,
    double minValue,
    double maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali

Parameters

validator

Type: CuttingEdge.Conditions.::ConditionValidator(Of (Double))
The ConditionValidator(Of (T)) that holds the value that has to be checked.
minValue
   Type: System::Double
   The lowest valid value.

maxValue
   Type: System::Double
   The highest valid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System::ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions::PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::..InRange Method (ConditionValidator<
(> > ), Int16, Int16)

Checks whether the given value is between minValue and maxValue (including
those values). An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    maxValue As Short _
) As ConditionValidator(Of Short)
```

**C#**

```csharp
public static ConditionValidator<short> IsInRange(
    ConditionValidator<short> validator,
    short minValue,
    short maxValue
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<short>^ IsInRange( 
    ConditionValidator<short>^ validator,
    short minValue,
    short maxValue
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali
```

**Parameters**

**validator**

Type: `CuttingEdge.Conditions..::.ConditionValidator<Of <(Int16)>>)`

The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minValue**
Type: `System::::Int16`
The lowest valid value.

maxValue
Type: `System::::Int16`
The highest valid value.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::InRange Method (ConditionValidator<Of <(Int16)>>, Int16, Int16, String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsInRange (_
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    maxValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)
```

**C#**

```csharp
public static ConditionValidator<short> IsInRange(
    ConditionValidator<short> validator, _
    short minValue, _
    short maxValue, _
    string conditionDescription
)
```

**Visual C++**

```c++
public:
static ConditionValidator<short>^ IsInRange(
    ConditionValidator<short>^ validator, _
    short minValue, _
    short maxValue, _
    String^ conditionDescription
)
```

**JavaScript**

```
CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali;
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions.::{:;ConditionValidator<(Of <(Int16)>)}>

The `ConditionValidator<(Of <(T)>)` that holds the value that has to be checked.
minValue
   Type: System::Int16
   The lowest valid value.

maxValue
   Type: System::Int16
   The highest valid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions:::InRange Method (ConditionValidator<(Of <(Int32)>), Int32, Int32)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    maxValue As Integer _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsInRange(
    ConditionValidator<int> validator,
    int minValue,
    int maxValue
)

Visual C++

public:
static ConditionValidator<int>^ IsInRange(
    ConditionValidator<int>^ validator,
    int^ minValue,
    int^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(Int32)>))
    The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.

minValue
Type: `System::Int32`
The lowest valid value.

maxValue
Type: `System::Int32`
The highest valid value.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.InRange Method (ConditionValidator<Of Int32>, Int32, Int32, String)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    maxValue As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsInRange(ConditionValidator<int> validator, int minValue, int maxValue, string conditionDescription)

Visual C++

public:
static ConditionValidator<int>^ IsInRange(ConditionValidator<int>^ validator, int minValue, int maxValue, String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(validator, minValue, maxValue, conditionDescription) {
    return new ConditionValidator<number>(validator, minValue, maxValue, conditionDescription);
}

Parameters

validator
Type: CuttingEdge.Conditions.:::ConditionValidator(Of <(Int32)>)
The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.
minValue
   Type: System::Int32
   The lowest valid value.

maxValue
   Type: System::Int32
   The highest valid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is not in the specified range, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is not in the specified range, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

- **ValidatorExtensions Class**
- **IsInRange Overload**
- **CuttingEdge.Conditions Namespace**

Send feedback on this topic to Microsoft.
Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange (_
    validator As ConditionValidator(Of Long), _
    minValue As Long, _
    maxValue As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsInRange(
    ConditionValidator<long> validator,
    long minValue,
    long maxValue
)

Visual C++

public:
static ConditionValidator<long long>^ IsInRange(
    ConditionValidator<long long>^ validator,
    long long minValue,
    long long maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int64)>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: System::Int64
The lowest valid value.

maxValue
Type: System::Int64
The highest valid value.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...::InRange Method (ConditionValidator<Of Петр>(Int64), Int64, Int64, String)

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsInRange ( 
    validator As ConditionValidator(Of Long), _
    minValue As Long, _
    maxValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)
```

### C#

```cs
public static ConditionValidator<long> IsInRange( 
    ConditionValidator<long> validator,
    long minValue,
    long maxValue,
    string conditionDescription
)
```

### Visual C++

```cpp
public: 
static ConditionValidator<long long>^ IsInRange( 
    ConditionValidator<long long>^ validator,
    long long minValue, 
    long long maxValue, 
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali
```

## Parameters

- **validator**
  - Type: `CuttingEdge.Conditions.::ConditionValidator<(Of (Int64)>)`
  - The `ConditionValidator<(Of (T)>)` that holds the value that has to be checked.
minValue
   Type: System::Int64
   The lowest valid value.

maxValue
   Type: System::Int64
   The highest valid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**  
**IsInRange Overload**  
**CuttingEdge.Conditions Namespace**

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.InRange Method (ConditionValidator<Of <(Single)>>, Single, Single)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single, _
    maxValue As Single _
) As ConditionValidator(Of Single)
```

### C#

```csharp
public static ConditionValidator<float> IsInRange( 
    ConditionValidator<float> validator, 
    float minValue, 
    float maxValue
)
```

### Visual C++

```cpp
public: 
static ConditionValidator<float>^ IsInRange( 
    ConditionValidator<float>^ validator, 
    float minValue, 
    float maxValue
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali
```
Type: **System::Single**
The lowest valid value.

```
maxValue
```
Type: **System::Single**
The highest valid value.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::.::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::.::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..:::.IsInRange Method (ConditionValidator<Of <(Single)>), Single, Single, String)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single, _
    maxValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsInRange(
    ConditionValidator<float> validator,
    float minValue,
    float maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsInRange(  
    ConditionValidator<float>^ validator,
    float minValue,
    float maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isInRange = function(vali...

Parameters

validator
    Type: CuttingEdge.Conditions::ConditionValidator<Of <(Single)>>
    The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.
**minValue**
   Type: `System..::Single`
   The lowest valid value.

**maxValue**
   Type: `System..::Single`
   The highest valid value.

**conditionDescription**
   Type: `System..::String`
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

_ValidatorExtensions Class_
_IsInRange Overload_
_CuttingEdge.Conditions Namespace_

Send feedback on this topic to Microsoft.
Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsInRange(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As Nullable(Of T), _
    maxValue As Nullable(Of T)) As ConditionValidator(Of Nullable(Of T))
```

#### C#

```csharp
public static ConditionValidator<Nullable<T>> IsInRange<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> minValue,
    Nullable<T> maxValue
)
```

where T : struct, new()

#### Visual C++

```cpp
public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsInRange(
        ConditionValidator<Nullable<T>>^ validator,
        Nullable<T> minValue,
        Nullable<T> maxValue
    )
```

#### JavaScript

JavaScript does not support generic types or methods.

### Parameters

**validator**

Type: `CuttingEdge.Conditions..::.ConditionValidator<(Of Nullable<(Of T)>)>`

The `ConditionValidator<(Of T)>` that holds the value that has to be
checked.

```csharp
minValue
  Type: System::Nullable<Of <(T)>>
  The lowest valid value.
```

```csharp
maxValue
  Type: System::Nullable<Of <(T)>>
  The highest valid value.
```
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::InRange(Of <(T)>) Method
(ConditionValidator(Of <(Nullable<(Of <(T)>))>), Nullable<(Of <(T)>)), Nullable<(Of <(T)>)), String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

Public Shared Function IsInRange(Of T As {Structure, New})( _
  validator As ConditionValidator(Of Nullable(Of T)), _
  minValue As Nullable(Of T), _
  maxValue As Nullable(Of T), _
  conditionDescription As String _
) As ConditionValidator(Of Nullable(Of T))

**C#**

public static ConditionValidator<Nullable<T>> IsInRange<T>(
  ConditionValidator<Nullable<T>> validator,
  Nullable<T> minValue,
  Nullable<T> maxValue,
  string conditionDescription
)

where T : struct, new()

**Visual C++**

public:
  generic<typename T>
  where T : value class, gcnew()
  static ConditionValidator<Nullable<T>>^ IsInRange(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T>^ minValue,
    Nullable<T>^ maxValue,
    String^ conditionDescription
  )

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

*validator*

Type: `CuttingEdge.Conditions...::ConditionValidator<Nullable<Nullable<T>>>`
The `ConditionValidator<Of `(T)>`) that holds the value that has to be checked.

**minValue**
Type: `System..:::Nullable<Of `(T)>`)  
The lowest valid value.

**maxValue**
Type: `System..:::Nullable<Of `(T)>`)  
The highest valid value.

**conditionDescription**
Type: `System..:::String`  
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
// Visual Basic  C#
// Visual C++  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions.IsInRange(Of T) Method
(ConditionValidator(Of Nullable(Of T)), T, T)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T, _
    maxValue As T _) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsInRange<T>(
    ConditionValidator<Nullable<T>> validator,
    T minValue,
    T maxValue
)
where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsInRange(
        ConditionValidator<Nullable<T>>^ validator,
        T minValue,
        T maxValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions,:::ConditionValidator<(Of Nullable<(Of
    <(T)>))>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

minValue
  Type: T
  The lowest valid value.

maxValue
  Type: T
  The highest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::..InRange(Of (T)> ) Method
(ConditionValidator(Of Nullable(Of (T)> ) ), T, T, String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsInRange(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T, _
    maxValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of Nullable(Of T))
```

#### C#

```csharp
public static ConditionValidator<Nullable<T>> IsInRange<T>(
    ConditionValidator<Nullable<T>> validator,
    T minValue,
    T maxValue,
    string conditionDescription
)
```

where T : struct, new()

#### Visual C++

```cpp
public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsInRange(
        ConditionValidator<Nullable<T>>^ validator,
        T minValue,
        T maxValue,
        String^ conditionDescription
    )
```

#### JavaScript

JavaScript does not support generic types or methods.

### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions...::ConditionValidator<Of Nullable<Of`
The `ConditionValidator<(Of <T>)>` that holds the value that has to be checked.

**minValue**
- Type: `T`
- The lowest valid value.

**maxValue**
- Type: `T`
- The highest valid value.

**conditionDescription**
- Type: `System::::String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](https://www.example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T, _
    maxValue As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsInRange<T>(
    ConditionValidator<T> validator,
    T minValue,
    T maxValue
)

where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsInRange(ConditionValidator<T>^ validator,
        T minValue,
        T maxValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator(Of <(T)>)
The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.
minValue
    Type: T
    The lowest valid value.

maxValue
    Type: T
    The highest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel..::.InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is not in the specified range and is an Enum type, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified</td>
</tr>
</tbody>
</table>
validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
InRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.InRange<(Of <(T)>)> Method
(ConditionValidator<(Of <(T)>)>, T, T, String)

Checks whether the given value is between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsInRange(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T, _
    maxValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsInRange<T>(
    ConditionValidator<T> validator,
    T minValue,
    T maxValue,
    string conditionDescription
)
where T : IComparable

Visual C++

public:
generic< typename T>
where T : IComparable
static ConditionValidator<T>^ IsInRange(
    ConditionValidator<T>^ validator,
    T minValue,
    T maxValue,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator(Of (T)>)
The **ConditionValidator<*(Of *(T)>)* that holds the value that has to be checked.

**minValue**
- Type: T
- The lowest valid value.

**maxValue**
- Type: T
- The highest valid value.

**conditionDescription**
- Type: System...::String
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel::InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is not in the specified range and is an Enum type, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is not in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
</tbody>
</table>
validator is created using the \texttt{Ensures} extension method.
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript  Include Protected Members  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...::IsLessOrEqual Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsLessOrEqual(ConditionValidator(Of Byte), Byte)</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessOrEqual(ConditionValidator(Of DateTime), DateTime)</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessOrEqual(ConditionValidator(Of Decimal), Decimal)</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessOrEqual(ConditionValidator(Of Double), Double)</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessOrEqual(ConditionValidator(Of Int16), Int16)</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessOrEqual(ConditionValidator(Of Int32), Int32)</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessOrEqual(ConditionValidator(Of Int64), Int64)</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessOrEqual(Of搪 (T)), (ConditionValidator(Of Nullable(Of (T)))),</td>
<td>Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
Nullable<(Of <(T)>))

IsLessOrEqual<(Of <(T)>)
(ConditionValidator<(Of
<(Nullable<(Of <(T)>)>)>), T)

IsLessOrEqual(ConditionValidator<(Of
<(Single)>), Single)

IsLessOrEqual<(Of <(T)>)
(ConditionValidator<(Of <(T)>), T)

IsLessOrEqual(ConditionValidator<(Of
<(Byte)>), Byte, String)

IsLessOrEqual(ConditionValidator<(Of
<(DateTime)>), DateTime, String)

IsLessOrEqual(ConditionValidator<(Of
<(Decimal)>), Decimal, String)

IsLessOrEqual(ConditionValidator<(Of
<(Double)>), Double, String)

IsLessOrEqual(ConditionValidator<(Of
<(Int16)>), Int16, String)

IsLessOrEqual(ConditionValidator<(Of
<(Int32)>), Int32, String)

exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.
IsLessOrEqual(ConditionValidator<(Of 
<Int64>)), Int64, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

IsLessOrEqual<(Of <(T)>)>
(ConditionValidator<(Of 
<Nullable<(Of <(T)>)>>>(), Nullable<(Of <(T)>)>, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

IsLessOrEqual<(Of <(T)>)>
(ConditionValidator<(Of 
<Nullable<(Of <(T)>)>>>(), T, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

IsLessOrEqual(ConditionValidator<(Of 
<(Single)>>>>, Single, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

IsLessOrEqual<(Of <(T)>)>
(ConditionValidator<(Of <(T)>>>>, T, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.
See Also

[ValidatorExtensions Class](#)
[ValidatorExtensions Members](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsLessOrEqual Method (ConditionValidator<Of <(Byte)>>, Byte)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Byte), _
    maxValue As Byte )
) As ConditionValidator(Of Byte)
```

**C#**

```csharp
public static ConditionValidator<byte> IsLessOrEqual(
    ConditionValidator<byte> validator,
    byte maxValue
)
```

**Visual C++**

```c++
public: ConditionValidator<unsigned char>^ IsLessOrEqual(
    ConditionValidator<unsigned char>^ validator,
    unsigned char maxValue
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions..::.ConditionValidator(Of <(Byte)>)`
- The `ConditionValidator(Of <(T)>)` that holds the value that has to be checked.

**maxValue**
- Type: `System..::.Byte`
- The highest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsLessOrEqual Method (Condition Validator<Of <(Byte) >>, Byte, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual (_
    validator As ConditionValidator(Of Byte), _
    maxValue As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsLessOrEqual(
    ConditionValidator<byte> validator,
    byte maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsLessOrEqual(
    ConditionValidator<unsigned char>^ validator,
    unsigned char maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<Of (Of (Type))>
    The ConditionValidator<Of (Of (Type))> that holds the value that has to be checked.

maxValue
The highest valid value.

conditionDescription
  Type: System..::.Byte
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsLessOrEqual Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsLessOrEqual(
    ConditionValidator<DateTime> validator,
    DateTime maxValue
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsLessOrEqual(
    ConditionValidator<DateTime>^ validator,
    DateTime^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <DateTime>)(Of <DateTime>)>
    The ConditionValidator(Of <T>)(Of <T>) that holds the value that has to be checked.

maxValue
    Type: System..::.DateTime
    The highest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsLessOrEqual Method (ConditionValidator(Of DateTime), DateTime, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsLessOrEqual(ConditionValidator<DateTime> validator, DateTime maxValue, String conditionDescription)

Visual C++

public:
static ConditionValidator<DateTime>^ IsLessOrEqual(ConditionValidator<DateTime>^ validator, DateTime^ maxValue, String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Condsitions...:::ConditionValidator<(Of <(DateTime)>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: **System::DateTime**
The highest valid value.

conditionDescription
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [IsLessOrEqual Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions...::.IsLessOrEqual Method (ConditionValidator<Of <(Decimal)>>, Decimal)

**ValidatorExtensions Class**  ** See Also  ** Send Feedback

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Decimal), _
    maxValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsLessOrEqual(
    ConditionValidator<decimal> validator,
    decimal maxValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsLessOrEqual( 
    ConditionValidator<Decimal>^ validator, 
    Decimal ^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Decimal)>)> 
    The ConditionValidator<(Of <*>>) that holds the value that has to be checked.

maxValue
    Type: System...:::Decimal 
    The highest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- IsLessOrEqual Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...::IsLessOrEqual Method (ConditionValidator<Of <(Decimal)>, Decimal, String)

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Decimal), _
    maxValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsLessOrEqual(
    ConditionValidator<decimal> validator,
    decimal maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsLessOrEqual(
    ConditionValidator<Decimal>^ validator,
    Decimal^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Decimal)>))
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: System::Decimal
The highest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than maxValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than maxValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsLessOrEqual Method (ConditionValidator<Of<Double> >, Double)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Double), _
    maxValue As Double _
) As ConditionValidator(Of Double)

C#

class IsLessOrEqual (ConditionValidator<double>, double maxValue

Visual C++

public static ConditionValidator<double> IsLessOrEqual (ConditionValidator<double> validator, double maxValue

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions.::: ConditionValidator(Of (Double>)
    The ConditionValidator(Of (T>) that holds the value that has to be checked.

maxValue
    Type: System.::: Double
    The highest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsLessOrEqual Method (ConditionValidator<
(Double)>, Double, String)

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
  validator As ConditionValidator(Of Double), _
  maxValue As Double, _
  conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<
double> IsLessOrEqual( 
  ConditionValidator<
double> validator, 
  double maxValue, 
  string conditionDescription
)

Visual C++

public: 
static ConditionValidator<
double>^ IsLessOrEqual( 
  ConditionValidator<
double>^ validator, 
  double^ maxValue, 
  String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Double)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: **System::Double**
The highest valid value.

**conditionDescription**
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
CuttingEdge.Conditions reference library  
ValidatorExtensions..::.IsLessOrEqual Method (ConditionValidator<Of <(Int16)>, Int16)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Short), _
    maxValue As Short _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsLessOrEqual(ConditionValidator<short> validator, short maxValue)

Visual C++

public:
static ConditionValidator<short>^ IsLessOrEqual(ConditionValidator<short>^ validator, short maxValue)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
Type: CuttingEdge.Conditions..::: ConditionValidator<Of <(Int16)>)
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

maxValue
Type: System..::: Int16
The highest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsLessOrEqual Method (ConditionValidator<Of (Int16)>, Int16, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsLessOrEqual ( _
  validator As ConditionValidator(Of Short), _
  maxValue As Short, _
  conditionDescription As String _
) As ConditionValidator(Of Short)

**C#**

public static ConditionValidator<short> IsLessOrEqual(
  ConditionValidator<short> validator,
  short maxValue,
  string conditionDescription
)

**Visual C++**

public:

static ConditionValidator<short>^ IsLessOrEqual(
  ConditionValidator<short>^ validator,
  short^ maxValue,
  String^ conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

**Parameters**

- **validator**
  Type: CuttingEdge.Conditions... ConditionValidator(Of (Int16>))
  The ConditionValidator(Of (<T>)) that holds the value that has to be checked.

- **maxValue**
Type: System::Int16
The highest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsLessOrEqual Method (ConditionValidator<Of <(Int32)>), Int32)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( 
    validator As ConditionValidator(Of Integer), 
    maxValue As Integer 
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsLessOrEqual( 
    ConditionValidator<int> validator, 
    int maxValue 
)

Visual C++

public: ConditionValidator<int>^ IsLessOrEqual( 
    ConditionValidator<int>^ validator, 
    int maxValue 
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(Int32)>)> 
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
    Type: System:::Int32 
    The highest valid value.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#  □  Visual C++  □  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..:::IsLessOrEqual Method (ConditionValidator<Of <(Int32)>, Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
  validator As ConditionValidator(Of Integer), _
  maxValue As Integer, _
  conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsLessOrEqual(
  ConditionValidator<int> validator,
  int maxValue,
  string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsLessOrEqual(
  ConditionValidator<int>^ validator,
  int^ maxValue,
  String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int32)>))
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: `System::Int32`
The highest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsLessOrEqual Method (ConditionValidator<Of <(Int64)>, Int64)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Long), _
    maxValue As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsLessOrEqual(
    ConditionValidator<long> validator,
    long maxValue
)

Visual C++

public:
    static ConditionValidator<long long>^ IsLessOrEqual(
        ConditionValidator<long long>^ validator,
        long long maxValue
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions::ConditionValidator<(Of <(Int64)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
    Type: System::Int64
    The highest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Long), _
    maxValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsLessOrEqual(
    ConditionValidator<long> validator,
    long maxValue,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<long long>^ IsLessOrEqual(
        ConditionValidator<long long>^ validator,
        long long maxValue,
        String^ conditionDescription
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int64)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: **System::Int64**
The highest valid value.

**conditionDescription**
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://www.cuttingedge-conditions.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual (_
    validator As ConditionValidator(Of Single), _
    maxValue As Single _) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsLessOrEqual(ConditionValidator<float> validator, float maxValue)

Visual C++

public:
static ConditionValidator<float>^ IsLessOrEqual(ConditionValidator<float>^ validator, float maxValue)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<Of <(Single)>>
    The ConditionValidator<Of <(T)>>() that holds the value that has to be checked.

maxValue
    Type: System..:::Single
    The highest valid value.
Return Value

The specified validator instance.
Exceptions

---

**Exception**

**System..::.ArgumentOutOfRangeException**

**Condition**

Thrown when the **Value** of the specified validator is greater than **maxValue**, while the specified validator is created using the **Requires** extension method.

**CuttingEdge.Conditions..::.PostconditionException**

Thrown when the **Value** of the specified validator is greater than **maxValue**, while the specified validator is created using the **Ensures** extension method.
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsLessOrEqual Method (ConditionValidator<Of <(Single)>>, Single, String)

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual ( _
    validator As ConditionValidator(Of Single), _
    maxValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsLessOrEqual( 
    ConditionValidator<float> validator, 
    float maxValue,
    string conditionDescription 
)

Visual C++

public: 
static ConditionValidator<float>^ IsLessOrEqual( 
    ConditionValidator<float>^ validator, 
    float^ maxValue, 
    String^ conditionDescription 
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessOrEqual = function(

Parameters

validator 
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of (<Single>))>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: `System::Single`
The highest valid value.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the \texttt{Value} of the specified validator is greater than \texttt{maxValue}, while the specified validator is created using the \texttt{Requires} extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the \texttt{Value} of the specified validator is greater than \texttt{maxValue}, while the specified validator is created using the \texttt{Ensures} extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsLessOrEqual Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As Nullable(Of T) _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsLessOrEqual<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> maxValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsLessOrEqual(ConditionValidator<Nullable<T>>^ validator,
    Nullable<T>^ maxValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of (Nullable<(Of<T>)>)>)

The ConditionValidator<(Of (T)>) that holds the value that has to be checked.
maxValue

Type: `System::Nullable<(Of <(T)>)>`

The highest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsLessOrEqual Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
CuttingEdge.Conditions reference library

**ValidatorExtensions..::.IsLessOrEqual(Of (Of T)> ) Method**

(ConditionValidator(Of Nullable(Of Nullable(T)> ) ), Nullable(Of Nullable(T)> ), String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As Nullable(Of T), _
    conditionDescription As String
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsLessOrEqual<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> maxValue,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsLessOrEqual(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T>^ maxValue,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of
    <(T)>))>)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

maxValue
  Type: System::Nullable<Of <(T)>>
  The highest valid value.

conditionDescription
  Type: System::String
  The description of the condition that should hold. The string may hold the
  placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- **ValidatorExtensions Class**
- **IsLessOrEqual Overload**
- **CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Checks whether the given value is smaller or equal to the specified `maxValue`. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual(Of T As {Structure, New}) ( _
   validator As ConditionValidator(Of Nullable(Of T)), _
   maxValue As T _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsLessOrEqual<T>(
   ConditionValidator<Nullable<T>> validator,
   T maxValue
)
where T : struct, new()

Visual C++

public:
   generic<typename T>
   where T : value class, gcnew()
   static ConditionValidator<Nullable<T>>^ IsLessOrEqual( 
      ConditionValidator<Nullable<T>>^ validator,
      T maxValue
   )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
   Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)>
   The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
maxValue
  Type: T
  The highest valid value.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions::IsLessOrEqual<Of<T>> Method
(ConditionValidator<Of<Nullable<Of<T>>>>, T, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsLessOrEqual(Of T As {Structure, New})( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As T, _
    conditionDescription As String) As ConditionValidator(Of Nullable(Of T))
```

### C#

```csharp
public static ConditionValidator<Nullable<T>> IsLessOrEqual<T>(
    ConditionValidator<Nullable<T>> validator, T maxValue,
    string conditionDescription
)
```

where T : struct, new()

### Visual C++

```cpp
public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsLessOrEqual( 
        ConditionValidator<Nullable<T>>^ validator, T maxValue, 
        String^ conditionDescription
    )
```

### JavaScript

JavaScript does not support generic types or methods.

## Parameters

**validator**
- Type: `CuttingEdge.Conditions.:::ConditionValidator<Of (Nullable<Of (T)>)>`
- The `ConditionValidator<Of <(T)>)>` that holds the value that has to be
checked.

maxValue
Type: T
The highest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the \textit{Value} of the specified validator is greater than \textit{maxValue}, while the specified validator is created using the \texttt{Requires} extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the \textit{Value} of the specified validator is greater than \textit{maxValue}, while the specified validator is created using the \texttt{Ensures} extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsLessOrEqual<T>(
    ConditionValidator<T> validator,
    T maxValue
)
where T : IComparable

Visual C++

public:
    generic<typtename T>
where T : IComparable
static ConditionValidator<T>^ IsLessOrEqual(ConditionValidator<T>^ validator,
    T maxValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(T)>))
    The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

maxValue
Type: T
The highest valid value.
Type Parameters

T
  The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than maxValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel:::InvalidEnumArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is an <code>Enum</code> type and is greater than maxValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than maxValue, while the specified validator is</td>
</tr>
</tbody>
</table>
created using the 
**Ensures**
extension method.
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions.::.IsLessOrEqual<(Of <(T)>)> Method
(ConditionValidator<(Of <(T)>)), T, String)

Checks whether the given value is smaller or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessOrEqual(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsLessOrEqual<T>(
    ConditionValidator<T> validator,
    T maxValue,
    string conditionDescription
)

where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsLessOrEqual(  
        ConditionValidator<T>^ validator,
        T maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::..ConditionValidator<(Of <(T)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
maxValue
   Type: T
   The highest valid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Type Parameters**

T

The type of the `Value` of the specified validator.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel..::.InvalidEnumArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is an <code>Enum</code> type and is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is</td>
</tr>
</tbody>
</table>
created using the `Ensure` extension method.
See Also

ValidatorExtensions Class
IsLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic □ C#
□ Visual C++
□ JavaScript
□ Include Protected Members
□ Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsLessThan Method
ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(Byte)&gt;), Byte</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(DateTime)&gt;), DateTime</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(Decimal)&gt;), Decimal</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(Double)&gt;), Double</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(Int16)&gt;), Int16</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(Int32)&gt;), Int32</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(Int64)&gt;), Int64</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan&lt;(Of &lt;(T)&gt;)&gt; (ConditionValidator&lt;Of &lt;(Nullable&lt;(Of &lt;(T)&gt;)&gt;), Nullable&lt;(Of &lt;(T)&gt;)&gt;&gt;)</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan&lt;(Of &lt;(T)&gt;)&gt; (ConditionValidator&lt;Of &lt;(Nullable&lt;(Of &lt;(T)&gt;)&gt;)&gt;), T)</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsLessThan(ConditionValidator&lt;Of &lt;(Single)&gt;), Single</td>
<td>Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
IsLessThan<(Of <(T)>)>(ConditionValidator<(Of <(T)>)>, T)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(Byte)>)>, Byte, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(DateTime)>)>, DateTime, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(Decimal)>)>, Decimal, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(Double)>)>, Double, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(Int16)>)>, Int16, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(Int32)>)>, Int32, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(Int64)>)>, Int64, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan<(Of <(T)>)>(ConditionValidator<(Of <(Nullable<Of <(T)>>)>), Nullable<Of <(T)>>>, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan<(Of <(T)>)>(ConditionValidator<(Of <(Nullable<Of <(T)>>)>), Nullable<Of <(T)>>>, T, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan(ConditionValidator<(Of <(Single)>)>, Single, String)

Checks whether the given value is less than the specified maxValue.
An exception is thrown otherwise.

IsLessThan<(Of <(T)>)>
(ConditionValidator<(Of <(T)>), T, String) less than the specified maxValue. An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Byte), _
    maxValue As Byte
) As ConditionValidator(Of Byte)
```

#### C#

```csharp
public static ConditionValidator< byte> IsLessThan(
    ConditionValidator< byte> validator,
    byte maxValue
)
```

#### Visual C++

```cpp
public:
static ConditionValidator< unsigned char>^ IsLessThan(
    ConditionValidator< unsigned char>^ validator,
    unsigned char maxValue
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val
```

### Parameters

**validator**
- Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Byte)>)
- The ConditionValidator(Of <(T)>), that holds the value that has to be checked.

**maxValue**
- Type: System..::.Byte
- The lowest invalid value.
Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
C#
Visual Basic  
Visual C++  
JavaScript

**CuttingEdge.Conditions reference library**

**ValidatorExtensions.:::IsLessThan Method** *(ConditionValidator<(Of (Byte)>), Byte, String)*

**ValidatorExtensions Class**  
**See Also**  
**Send Feedback**

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:**  
**Assembly:**  
**CuttingEdge.Conditions**  
CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Byte), _
    maxValue As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsLessThan(
    ConditionValidator<byte> validator,
    byte maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsLessThan(
    ConditionValidator<unsigned char>^ validator,
    unsigned char maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val}
Type: `System::Byte`
The lowest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

* [ValidatorExtensions Class](#)
  * [IsLessThan Overload](#)
  * [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions:::IsLessThan Method (ConditionValidator<(DateTime)>, DateTime)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsLessThan(
    ConditionValidator<DateTime> validator,
    DateTime maxValue
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsLessThan(
    ConditionValidator<DateTime>^ validator,
    DateTime maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <<DateTime>>)> The ConditionValidator<(Of <<T>>)> that holds the value that has to be checked.

maxValue
    Type: System..::: DateTime
    The lowest invalid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic
C#
Visual C++
JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsLessThan Method (ConditionValidator<Of <(DateTime)>), DateTime, String)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.
## Syntax

### Visual Basic (Declaration)

```
Public Shared Function IsLessThan ( 
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)
```

### C#

```
public static ConditionValidator<DateTime> IsLessThan(
    ConditionValidator<DateTime> validator,
    DateTime maxValue,
    string conditionDescription
)
```

### Visual C++

```
public: 
static ConditionValidator<DateTime>^ IsLessThan( 
    ConditionValidator<DateTime>^ validator, 
    DateTime^ maxValue, 
    String^ conditionDescription 
)
```

### JavaScript

```
CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val
```

### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions...:::ConditionValidator<Of <(DateTime)>>`
  - The `ConditionValidator<Of <(T)>>>` that holds the value that has to be checked.

- **maxValue**
Type: `System.DateTime`
The lowest invalid value.

conditionDescription
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsLessThan Method (ConditionValidator<Of <(Decimal)>), Decimal)

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Decimal), _
    maxValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsLessThan(
    ConditionValidator<decimal> validator,
    decimal maxValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsLessThan(
    ConditionValidator<Decimal>^ validator,
    Decimal^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

Parameters

validator
Type: CuttingEdge.Conditions.:::ConditionValidator<(Of <(Decimal)>)>
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: System.:::Decimal
The lowest invalid value.
**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsLessThan Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Decimal), _
    maxValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsLessThan( ConditionValidator<decimal> validator,
    decimal maxValue,
    string conditionDescription
)

Visual C++

public: static ConditionValidator<Decimal>^ IsLessThan( ConditionValidator<Decimal>^ validator,
    Decimal^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Decimal)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: System::Decimal
The lowest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System::ArgumentOutOfRangeException</strong></td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions::PostconditionException</strong></td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsLessThan Method (ConditionValidator<Of<Double>>, Double)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    _
    validator As ConditionValidator(Of Double), _
    maxValue As Double _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsLessThan( ConditionValidator<double> validator, double maxValue
}

Visual C++

public:
static ConditionValidator<double>^ IsLessThan( ConditionValidator<double>^ validator, double maxValue
}

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Double)>))
The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

maxValue
    Type: System..::: Double
    The lowest invalid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsLessThan Method (ConditionValidator<Of[Double]>), Double, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Double), _
    maxValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

**C#**

public static ConditionValidator<
double> IsLessThan( 
    ConditionValidator<
double> validator, 
    double maxValue, 
    string conditionDescription 
)

**Visual C++**

public:
static ConditionValidator<
double>^ IsLessThan( 
    ConditionValidator<
double>^ validator, 
    double^ maxValue, 
    String^ conditionDescription 
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

**Parameters**

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Double)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: `System::*::Double`
The lowest invalid value.

conditionDescription
Type: `System::*::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::..ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::..PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsLessThan Method (ConditionValidator<Of (Int16)>, Int16)

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Short), _
    maxValue As Short _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsLessThan(
    ConditionValidator<short> validator,
    short maxValue
)

Visual C++

public:
static ConditionValidator<short>^ IsLessThan(
    ConditionValidator<short>^ validator,
    short maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Int16)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
    Type: System..::: Int16
    The lowest invalid value.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsLessThan Method (ConditionValidator<Of (Int16)>, Int16, String)

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Short), _
    maxValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsLessThan(
    ConditionValidator<short> validator,
    short maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<short>^ IsLessThan(
    ConditionValidator<short>^ validator,
    short^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val}
Type: System::Int16
The lowest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

**ValidatorExtensions...::IsLessThan Method (ConditionValidator<(Of (Int32)>, Int32)**

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
   validator As ConditionValidator(Of Integer), _
   maxValue As Integer _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsLessThan(ConditionValidator<int> validator,
   int maxValue
)

Visual C++

public: ConditionValidator<int>^ IsLessThan(ConditionValidator<int>^ validator,
   int maxValue
)

JavaScript

CuttingEdge.Conditions.VeiatorExtensions.isLessThan = function(val

Parameters

validator
   Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Int32)>>)
   The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
   Type: System..::: Int32
   The lowest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsLessThan Method (ConditionValidator<Of <Int32>>, Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace**: CuttingEdge.Conditions
**Assembly**: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Integer), _
    maxValue As Integer, _
    conditionDescription As String _) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsLessThan( 
    ConditionValidator<int> validator,
    int maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsLessThan( 
    ConditionValidator<int>^ validator,
    int^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

Parameters

validator
    Type: CuttingEdge.Conditions::ConditionValidator<Of <(Int32)>>
    The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

maxValue
Type: `System::Int32`
The lowest invalid value.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::.::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::.::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [IsLessThan Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Checking whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Long), _
    maxValue As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsLessThan(
    ConditionValidator<long> validator,
    long maxValue
)

Visual C++

public:
static ConditionValidator<long long>^ IsLessThan(
    ConditionValidator<long long>^ validator,
    long long maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val]  

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int64)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
    Type: System..::.Int64
    The lowest invalid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsLessThan Method (ConditionValidator<Of <(Int64)>, Int64, String)

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Long), _
    maxValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsLessThan(ConditionValidator<long> validator,
    long maxValue,
    string conditionDescription)

Visual C++

public:
static ConditionValidator<long long>* IsLessThan(ConditionValidator<long long>* validator,
    long long maxValue,
    String* conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val]

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Int64)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: `System::Int64`
The lowest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsLessThan Method (ConditionValidator<Of <(Single)>), Single)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan ( _
    validator As ConditionValidator(Of Single), _
    maxValue As Single _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsLessThan(
    ConditionValidator<float> validator,
    float maxValue
)

Visual C++

public: ConditionValidator<float>^ IsLessThan(
    ConditionValidator<float>^ validator,
    float maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator<Of <(Single)>>
    The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

maxValue
    Type: System:::Single
    The lowest invalid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.:::.IsLessThan Method (ConditionValidator<Of <(Single)>>, Single, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsLessThan (  
    validator As ConditionValidator(Of Single), _
    maxValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

### C#

public static ConditionValidator<float> IsLessThan(  
    ConditionValidator<float> validator,
    float maxValue,
    string conditionDescription
)

### Visual C++

public:
static ConditionValidator<float>^ IsLessThan(  
    ConditionValidator<float>^ validator,
    float^ maxValue,
    String^ conditionDescription
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLessThan = function(val}
Type: \texttt{System:::Single}
The lowest invalid value.

conditionDescription
Type: \texttt{System:::String}
The description of the condition that should hold. The string may hold the placeholder '{0}' for the \texttt{ArgumentName}.

\textbf{Return Value}

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- **ValidatorExtensions Class**
- **IsLessThan Overload**
- **CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions..::.IsLessThan(Of <(T)>)) Method
(ConditionValidator(Of <(Nullable<(Of <(T)>)>)>), Nullable<(Of <(T)>)>))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan(Of T As {Structure, New}) ( _
validator As ConditionValidator(Of Nullable(Of T)), _
maxValue As Nullable(Of T) _) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsLessThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> maxValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsLessThan(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T>^ maxValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Nullable<(Of <(T)>))>)
The ConditionValidator(Of <(T)>)) that holds the value that has to be checked.
maxValue

    Type: System::Nullable<(Of <(T)>))
    The lowest invalid value.
### Type Parameters

**T**

The type of the *Value* of the specified validator.

### Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions:::IsLessThan(Of <T>)) Method
(ConditionValidator(Of Nullable(Of <T>)), Nullable(Of <T>), String)

**ValidatorExtensions Class**  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As Nullable(Of T), _
    conditionDescription As String
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsLessThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> maxValue,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsLessThan(
        ConditionValidator<Nullable<T>>^ validator,
        Nullable<T>^ maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions.::.ConditionValidator<(Of Nullable<(Of T)>))>
The ConditionValidator<(Of (T)>)) that holds the value that has to be
checked.

maxValue
Type: `System::Nullable<(Of <(T)>)>`
The lowest invalid value.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions.:::IsLessThan<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>), T)

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As T _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsLessThan<T>(
    ConditionValidator<Nullable<T>> validator,
    T maxValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsLessThan(
        ConditionValidator<Nullable<T>>^ validator,
        T maxValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of <(T)>)>))>

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
maxValue
    Type: T
    The lowest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As T, _
    conditionDescription As String
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsLessThan<T>(
    ConditionValidator<Nullable<T>> validator,
    T maxValue,
    string conditionDescription
)
where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsLessThan(
        ConditionValidator<Nullable<T>>^ validator,
        T maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<Of (Of Nullable<(Of <(T)><>)>)
The ConditionValidator<(Of <(T)><>)) that holds the value that has to be
checked.

maxValue
Type: T
The lowest invalid value.

conditionDescription
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::.ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class  
IsLessThan Overload  
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsLessThan<T>(
    ConditionValidator<T> validator,
    T maxValue
}
where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsLessThan(
        ConditionValidator<T>^ validator,
        T maxValu
    }

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(T)>)
    The ConditionValidator<(Of <(T)> that holds the value that has to be checked.

maxValue
Type: T
The lowest invalid value.
**Type Parameters**

T

The type of the Value of the specified validator.

**Return Value**

The specified validator instance.
## Exceptions

| Exception |
|-----------
| **System..::.ArgumentOutOfRangeException** |

Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.

| Exception |
|-----------
| **System.ComponentModel..::.InvalidEnumArgumentException** |

Thrown when the Value of the specified validator is an Enum type and is greater or equal to maxValue, while the specified validator is created using the Requires extension method.

Thrown when the Value of the specified validator is greater or equal to maxValue,
CuttingEdge.Conditions...:::PostconditionException

while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsLessThan<(Of <T>)> Method
(ConditionValidator<(Of <T>)>, T, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is less than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLessThan(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsLessThan<T>(
    ConditionValidator<T> validator,
    T maxValue,
    string conditionDescription
)

where T : IComparable

Visual C++

public:
    generic< typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsLessThan(
        ConditionValidator<T>^ validator,
        T maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(T)>)
    The ConditionValidator<(Of <(T)> )> that holds the value that has to be checked.
maxValue
   Type: T
   The lowest invalid value.

conditionDescription
   Type: System :: String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
- **Type Parameters**

  T

  The type of the *Value* of the specified validator.

- **Return Value**

  The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel..::.InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is an Enum type and is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
</tbody>
</table>

Thrown when the Value of the specified validator is greater or equal to maxValue,
while the specified validator is created using the `Ensures` extension method.
See Also

ValidatorExtensions Class
IsLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.Is LongerOrEqual Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IsLongerOrEqual(ConditionValidator&lt;(Of &lt;(String)&gt;), Int32)</strong></td>
<td>Checks whether the given value is longer or equal in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><strong>IsLongerOrEqual&lt;(Of &lt;(TCollection)&gt;)&gt; (ConditionValidator&lt;(Of &lt;(TCollection)&gt;), Int32)</strong></td>
<td>Checks whether the number of elements in the given value, is more than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td><strong>IsLongerOrEqual(ConditionValidator&lt;(Of &lt;(String)&gt;), Int32, String)</strong></td>
<td>Checks whether the given value is longer or equal in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><strong>IsLongerOrEqual&lt;(Of &lt;(TCollection)&gt;)&gt; (ConditionValidator&lt;(Of &lt;(TCollection)&gt;), Int32, String)</strong></td>
<td>Checks whether the number of elements in the given value, is more than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
reference, it is considered to have 0 elements.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsLongerOrEqual Method (ConditionValidator<Of <(String)>>, Int32)

Checks whether the given value is longer or equal in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLongerOrEqual ( _
    validator As ConditionValidator(Of String), _
    minLength As Integer _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<String> IsLongerOrEqual(
    ConditionValidator<String> validator,
    int minLength
)

Visual C++

public: ConditionValidator<String>*^ IsLongerOrEqual(
    ConditionValidator<String>*^ validator,
    int minLength
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLongerOrEqual = funcion

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(String)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minLength
    Type: System..::: Int32
    The smallest valid length.
Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the length of the <code>Value</code> of the specified validator is smaller than <code>minLength</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and <code>minLength</code> is greater than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the length of the <code>Value</code> of the specified validator is smaller than <code>minLength</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLongerOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is longer or equal in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdgeCONDitions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsLongerOrEqual ( _
    validator As ConditionValidator(Of String), _
    minLength As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of String)
```

### C#

```csharp
public static ConditionValidator<string> IsLongerOrEqual(
    ConditionValidator<string> validator,
    int minLength,
    string conditionDescription
)
```

### Visual C++

```cpp
public:
static ConditionValidator<String^>^ IsLongerOrEqual(
    ConditionValidator<String^>^ validator,
    int minLength,
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isLongerOrEqual = function
```

## Parameters

**validator**
- Type: `CuttingEdge.Conditions..::.ConditionValidator<Of <(String)>>`
- The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minLength**
Type: **System..::..Int32**
The smallest valid length.

**conditionDescription**
Type: **System..::..String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the length of the Value of the specified validator is smaller than minLength, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the Value of the specified validator is a null reference and minLength is greater than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the length of the Value of the specified validator is smaller than minLength, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLongerOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

**ValidatorExtensions..:::IsLongerOrEqual(Of (TCollection)>)(Of (TCollection)>) Method**

(ConditionValidator(Of (TCollection)>, Int32)

Checks whether the number of elements in the given value, is more than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLongerOrEqual(Of TCollection As IEnumerable
validator As ConditionValidator(Of TCollection), _
numberOfElements As Integer _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsLongerOrEqual<TCollection>
ConditionValidator<TCollection> validator,
int numberOfElements

where TCollection : IEnumerable

Visual C++

public:

generic< typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ IsLongerOrEqual(
    ConditionValidator<TCollection>^ validator,
    int numberOfElements
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<Of
<((TCollection)>)
The ConditionValidator<Of <(T)>>() that holds the value that has to be
checked.
numberOfElements
   Type: System.Int32
   The collection must contain the same amount or more elements than this value.
**Type Parameters**

TCollection
   The type of the value to check.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator contains less than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is greater than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator contains less than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLongerOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the number of elements in the given value, is more than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLongerOrEqual(Of TCollection As IEnumerable,
 validator As ConditionValidator(Of TCollection), _
 numberOfElements As Integer, _
 conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsLongerOrEqual<TCollection>(
    ConditionValidator<TCollection> validator,
    int numberOfElements,
    string conditionDescription
)

where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsLongerOrEqual(
        ConditionValidator<TCollection>^ validator,
        int numberOfElements,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of
<(<TCollection>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**numberOfElements**
Type: **System:::Int32**
The collection must contain the same amount or more elements than this value.

**conditionDescription**
Type: **System:::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the Value of the specified validator contains less than specified by the numberOfElements argument, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and the numberOfElements is greater than 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>PostconditionException</td>
<td>Thrown when the Value of the specified validator contains less than specified by the numberOfElements argument, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLongerOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsLongerThan Method

ValidatorExtensions Class  See Also  Send Feedback
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![IsLongerThan](ConditionValidator(Of (String&gt;), Int32))</td>
<td>Checks whether the given value is longer in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><img src="Of" alt="IsLongerThan" title="TCollection&gt;" />&gt;(ConditionValidator(Of (TCollection&gt;), Int32))</td>
<td>Checks whether the number of elements in the given value, is more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td>![IsLongerThan](ConditionValidator(Of (String&gt;), Int32, String))</td>
<td>Checks whether the given value is longer in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><img src="Of" alt="IsLongerThan" title="TCollection&gt;" />&gt;(ConditionValidator(Of (TCollection&gt;), Int32, String))</td>
<td>Checks whether the number of elements in the given value, is more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
reference, it is considered to have 0 elements.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is longer in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLongerThan ( _
    validator As ConditionValidator(Of String), _
    minLength As Integer _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsLongerThan(ConditionValidator<string> validator,
    int minLength
)

Visual C++

public:
static ConditionValidator<String>^ IsLongerThan(ConditionValidator<String>^ validator,
    int minLength
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLongerThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(String)>)
The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.

minLength
    Type: System..::.Int32
    The biggest invalid length.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the length of the <code>Value</code> of the specified validator is smaller or equal to <code>minLength</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and <code>minLength</code> is greater or equal to 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the length of the <code>Value</code> of the specified validator is smaller or equal to <code>minLength</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLongerThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsLongerThan Method (ConditionValidator<Of <(String)>>, Int32, String)

Checks whether the given value is longer in length than minLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.

**Namespace:**  [CuttingEdge.Conditions](CuttingEdge.Conditions)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsLongerThan ( _
    validator As ConditionValidator(Of String), _
    minLength As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsLongerThan(ConditionValidator<string> validator,
    int minLength,
    string conditionDescription)

Visual C++

public:
static ConditionValidator<String^>^ IsLongerThan(ConditionValidator<String^>^ validator,
    int minLength,
    String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isLongerThan = function(

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(String>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minLength
Type: System::Int32
The biggest invalid length.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the length of the Value of the specified validator is smaller or equal to minLength, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and minLength is greater or equal to 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the length of the Value of the specified validator is smaller or equal to minLength, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsLongerThan Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions.IsLongerThan Method
(ConditionValidator(Of TCollection), Int32)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the number of elements in the given value, is more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.
Syntax

Visual Basic (Declaration)

Public Shared Function IsLongerThan(Of TCollection As IEnumerable) (validator As ConditionValidator(Of TCollection), _
numberOfElements As Integer _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsLongerThan<TCollection>(
    ConditionValidator<TCollection> validator,
    int numberOfElements
)
where TCollection : IEnumerable

Visual C++

public:
    template<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsLongerThan(ConditionValidator<TCollection>^ validator,
    int numberOfElements
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator(Of TCollection)
    The ConditionValidator(Of <T>) that holds the value that has to be checked.
numberOfElements
   Type: System..::..Int32
   The collection must contain the same amount or less elements than this value.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
Exceptions

**Exception**

**Condition**

**System..::.ArgumentException**

Thrown when the Value of the specified validator contains less or the same amount of elements as specified by the numberOfElements argument, while the specified validator is created using the Requires extension method.

**System..::.ArgumentNullException**

Thrown when the Value of the specified validator is a null reference and the numberOfElements is greater or equal to 0, while the specified validator is created using the Requires extension method.

**CuttingEdge.Conditions..::.PostconditionException**

Thrown when the Value of the specified validator contains less or the same amount of elements as specified by the numberOfElements argument, while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
IsLongerThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsLongerThan<
(TCollection>
)> Method
(ConditionValidator<
(TCollection>
>, Int32, String)

Checks whether the number of elements in the given value, is more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsLongerThan(Of TCollection As IEnumerable) (validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer, _
    conditionDescription As String _) As ConditionValidator(Of TCollection)

**C#**

public static ConditionValidator<TCollection> IsLongerThan<TCollection>(
    ConditionValidator<TCollection> validator,
    int numberOfElements,
    string conditionDescription
)

where TCollection : IEnumerable

**Visual C++**

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsLongerThan(
        ConditionValidator<TCollection>^ validator,
        int numberOfElements,
        String^ conditionDescription
    )

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<Of (Of (TCollection)>)

The ConditionValidator<Of (T<T>)> that holds the value that has to be
checked.

numberOfElements
Type: System::Int32
The collection must contain the same amount or less elements than this value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
  The type of the value to check.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator contains less or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is greater or equal to 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator contains less or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsLongerThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsNaN Method
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNaN(ConditionValidator&lt;OF &lt;(Double)&gt;))</code></td>
<td>Checks whether the given value is a valid number. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNaN(ConditionValidator&lt;OF &lt;(Single)&gt;))</code></td>
<td>Checks whether the given value is a valid number. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNaN(ConditionValidator&lt;OF &lt;(Double)&gt;, String)</code></td>
<td>Checks whether the given value is a valid number. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNaN(ConditionValidator&lt;OF &lt;(Single)&gt;, String)</code></td>
<td>Checks whether the given value is a valid number. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
**ValidatorExtensions.IsNaN Method (ConditionValidator<Of <(Double)>))**

**ValidatorExtensions Class**  See Also  Send Feedback

Checks whether the given value is a valid number. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsNaN ( _
    validator As ConditionValidator(Of Double) _
) As ConditionValidator(Of Double)

### C#

public static ConditionValidator<double> IsNaN(
    ConditionValidator<double> validator
)

### Visual C++

public:
static ConditionValidator<double>*^ IsNaN(
    ConditionValidator<double>*^ validator
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNaN = function(validator

---

### Parameters

- **validator**
  - Type: CuttingEdge.Conditions:::ConditionValidator(Of (Double))
  - The ConditionValidator(Of (T)) that holds the value that has to be checked.

### Return Value

- The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::..IsNaN Method (ConditionValidator<Of <(Double)>), String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is a valid number. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNaN (validator As ConditionValidator(Of Double), conditionDescription As String) As ConditionValidator(Of Double)
```

**C#**

```csharp
public static ConditionValidator<double> IsNaN(ConditionValidator<double> validator, string conditionDescription)
```

**Visual C++**

```cpp
public: ConditionValidator<double>^ IsNaN(ConditionValidator<double>^ validator, String^ conditionDescription)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNaN = function(validator, conditionDescription)
```

**Parameters**

**validator**
Type: `CuttingEdge.Conditions::ConditionValidator(Of (Double)>)`
The `ConditionValidator(Of (T)>)` that holds the value that has to be checked.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 


Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsNaN Method (ConditionValidator<Of <(Single)>))

ValidatorExtensions Class    See Also    Send Feedback

Checks whether the given value is a valid number. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNaN ( _
    validator As ConditionValidator(Of Single) _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNaN(
    ConditionValidator<float> validator
)

Visual C++

public:
    static ConditionValidator<float>^ IsNaN(
        ConditionValidator<float>^ validator
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNaN = function(validator:

Parameters

validator
    Type: CuttingEdge.Conditions,:::ConditionValidator<(Of <(Single)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is a valid number. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](https://example.com)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNaN ( _
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)

**C#**

public static ConditionValidator<float> IsNaN(
    ConditionValidator<float> validator,
    string conditionDescription
)

**Visual C++**

public:
static ConditionValidator<float>^ IsNaN( 
    ConditionValidator<float>^ validator,
    String^ conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNaN = function(validator,

**Parameters**

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Single)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
Type: System..:::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not a valid number, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
□  Include Protected Members
□  Include Inherited Members

CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsNegativeInfinity Method

ValidatorExtensions Class   See Also   Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNegativeInfinity(ConditionValidator&lt;Of (Double)&gt;)</code></td>
<td>Checks whether the given value is negative infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNegativeInfinity(ConditionValidator&lt;Of (Single)&gt;)</code></td>
<td>Checks whether the given value is negative infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNegativeInfinity(ConditionValidator&lt;Of (Double)&gt;, String)</code></td>
<td>Checks whether the given value is negative infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNegativeInfinity(ConditionValidator&lt;Of (Single)&gt;, String)</code></td>
<td>Checks whether the given value is negative infinity. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is negative infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNegativeInfinity (_
    validator As ConditionValidator(Of Double) _) 
    As ConditionValidator(Of Double)
```

#### C#

```csharp
public static ConditionValidator<double> IsNegativeInfinity(
    ConditionValidator<double> validator)
```

#### Visual C++

```cpp
public:
static ConditionValidator<double>^ IsNegativeInfinity(
    ConditionValidator<double>^ validator)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNegativeInfinity = func
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions::<:,:;:::ConditionValidator(Of <(Double>)>)`

The `ConditionValidator(Of <(T)>)` that holds the value that has to be checked.

### Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNegativeInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsNegativeInfinity Method (ConditionValidator<Of[Double]>, String)

Checks whether the given value is negative infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```
Public Shared Function IsNegativeInfinity ( _
    validator As ConditionValidator(Of Double), _
    conditionDescription As String _
) As ConditionValidator(Of Double)
```

#### C#

```
public static ConditionValidator<double> IsNegativeInfinity(  
    ConditionValidator<double> validator,  
    string conditionDescription
)
```

#### Visual C++

```
public:  
static ConditionValidator<double>^ IsNegativeInfinity(  
    ConditionValidator<double>^ validator,  
    String^ conditionDescription
)
```

#### JavaScript

```
CuttingEdge.Conditions.ValidatorExtensions.isNegativeInfinity = func
```

### Parameters

**validator**

- Type: `CuttingEdge.Conditions::ConditionValidator<Of <Double>>`
- The `ConditionValidator<Of <T>>` that holds the value that has to be checked.

**conditionDescription**

- Type: `System::String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNegativeInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is negative infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNegativeInfinity ( _
    validator As ConditionValidator(Of Single) _) As ConditionValidator(Of Single)
```

### C#

```csharp
public static ConditionValidator<float> IsNegativeInfinity(
    ConditionValidator<float> validator
)
```

### Visual C++

```cpp
public:
static ConditionValidator<float>^ IsNegativeInfinity(
    ConditionValidator<float>^ validator
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNegativeInfinity = func
```

## Parameters

validator

Type: `CuttingEdge.Conditions::ConditionValidator<Of <(Single)>)`

The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

## Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNegativeInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is negative infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNegativeInfinity (_
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)
```

### C#

```csharp
public static ConditionValidator<float> IsNegativeInfinity(
    ConditionValidator<float> validator,
    string conditionDescription
)
```

### Visual C++

```cpp
public:
static ConditionValidator<float>^ IsNegativeInfinity(
    ConditionValidator<float>^ validator,
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNegativeInfinity = func
```

## Parameters

**validator**
- Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Single)>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**conditionDescription**
- Type: `System..:::String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions...PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not negative infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNegativeInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic

C#

Visual C++

JavaScript

Include Protected Members

Include Inherited Members

CuttingEdge.Conditions reference library

ValidatorExtensions.IsNotEmpty Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotEmpty(ConditionValidator(Of String))</td>
<td>Checks whether the given value is not an Empty() string. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEmpty(Of TCollection) (ConditionValidator(Of TCollection))</td>
<td>Checks whether the given value does contain elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.</td>
</tr>
<tr>
<td>IsNotEmpty(ConditionValidator(Of String), String)</td>
<td>Checks whether the given value is not an Empty() string. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEmpty(Of TCollection) (ConditionValidator(Of TCollection), String)</td>
<td>Checks whether the given value does contain elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.NotEmpty Method (ConditionValidator<Of <(String)>)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is not an Empty()() string. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsNotEmpty ( _
    validator As ConditionValidator(Of String) _
) As ConditionValidator(Of String)

### C#

public static ConditionValidator<string> IsNotEmpty( ConditionValidator<string> validator )

### Visual C++

public: ConditionValidator<String^>^ IsNotEmpty( ConditionValidator<String^>^ validator )

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEmpty = function(val

## Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(String)>>

The ConditionValidator<Of <(T)>>() that holds the value that has to be checked.

## Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator is empty, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is empty, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the given value is not an Empty\(())\) string. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotEmpty ( 
    validator As ConditionValidator(Of String), 
    conditionDescription As String _
) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> IsNotEmpty( 
    ConditionValidator<string> validator, 
    string conditionDescription
)
```

**Visual C++**

```cpp
public: ConditionValidator<String^>^ IsNotEmpty( 
    ConditionValidator<String^>^ validator, 
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.MatcherExtensions.isNotEmpty = function(val)
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(String)>)`

The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**conditionDescription**

Type: `System:::String`

The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is empty, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is empty, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEmpty Overload
CuttingEdge.Constraints Namespace

Send feedback on this topic to Microsoft.
C#  

CuttingEdge.Conditions reference library  

ValidatorExtensions..:::NotEmpty<(Of <(TCollection)>)> Method  
(ConditionValidator<(Of <(TCollection)>)>))  

ValidatorExtensions Class  
See Also  
Send Feedback

Checks whether the given value does contain elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.

Namespace: CuttingEdge.Conditions  
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEmpty(Of TCollection As IEnumerable) ( _
    validator As ConditionValidator(Of TCollection) _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsNotEmpty<TCollection>
    ConditionValidator<TCollection> validator
)
where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsNotEmpty(
        ConditionValidator<TCollection>^ validator
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<Of
        <(TCollection)>)
    The ConditionValidator<Of <(T)>>) that holds the value that has to be
    checked.
Type Parameters

TCollection
The type of the value to check.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is empty, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is empty, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value does contain elements. An exception is thrown otherwise. When the value is a null reference it is considered empty.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEmpty(Of TCollection As IEnumerable) ( _
    validator As ConditionValidator(Of TCollection), _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsNotEmpty<TCollection>
    (ConditionValidator<TCollection> validator, 
     string conditionDescription
)
where TCollection : IEnumerable

Visual C++

public: 
    generic<typename TCollection>
    where TCollection : IEnumerable 
    static ConditionValidator<TCollection>^ IsNotEmpty(
        ConditionValidator<TCollection>^ validator, 
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of 
        (TCollection)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
conditionDescription

Type: `System.String`

The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.
Type Parameters

TCollection
  The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is empty, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is empty, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEmpty Overload
CuttingEdge.Conditions Namespace

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
Include Protected Members  Include Inherited Members

CuttingEdge.Conditions reference library
ValidatorExtensions...:::!.IsNotEqualTo Method

ValidatorExtensions Class  See Also  Send Feedback
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotEqualTo(ConditionValidator&lt;Of &lt;(Byte)&gt;), Byte</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo(ConditionValidator&lt;Of &lt;(DateTime)&gt;), DateTime</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo(ConditionValidator&lt;Of &lt;(Decimal)&gt;), Decimal</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo(ConditionValidator&lt;Of &lt;(Double)&gt;), Double</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo(ConditionValidator&lt;Of &lt;(Int16)&gt;), Int16</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo(ConditionValidator&lt;Of &lt;(Int32)&gt;), Int32</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo(ConditionValidator&lt;Of &lt;(Int64)&gt;), Int64</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;Of &lt;(T)&gt;&gt;(ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;)&gt;), Nullable&lt;Of &lt;(T)&gt;&gt;)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;Of &lt;(T)&gt;&gt;&gt; (ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;&gt;&gt;)&gt;), T)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;ConditionValidator&lt;Of &lt;(Single)&gt;&gt;&gt;, Single)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;Of &lt;(T)&gt;&gt; (ConditionValidator&lt;Of &lt;(T)&gt;&gt;), T)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;ConditionValidator&lt;Of &lt;(Byte)&gt;&gt;&gt;, Byte, String)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;ConditionValidator&lt;Of &lt;(DateTime)&gt;&gt;&gt;, DateTime, String)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;ConditionValidator&lt;Of &lt;(Decimal)&gt;&gt;&gt;, Decimal, String)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;ConditionValidator&lt;Of &lt;(Double)&gt;&gt;&gt;, Double, String)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;ConditionValidator&lt;Of &lt;(Int16)&gt;&gt;&gt;, Int16, String)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotEqualTo&lt;ConditionValidator&lt;Of &lt;(Int32)&gt;&gt;&gt;, Int32, String)</td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsNotEqualTo(ConditionValidator&lt;Of Int64&gt;, Int64, String)</code></td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotEqualTo(Of &lt;T&gt;())</code></td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsValidTo&lt;null(Nullable&lt;Of &lt;T&gt;()&gt;)&gt;, Nullable&lt;Of &lt;T&gt;()&gt;, String)</code></td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotEqualTo&lt;null(Nullable&lt;Of (Single)&gt;)&gt;, Single, String)</code></td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotEqualTo&lt;null(Nullable&lt;Of &lt;T&gt;()&gt;)&gt;, T, String)</code></td>
<td>Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Byte), _
    value As Byte _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator< byte> IsNotEqualTo(  
    ConditionValidator< byte> validator,  
    byte value
)

Visual C++

public:  
static ConditionValidator<unsigned char>^ IsNotEqualTo(  
    ConditionValidator< unsigned char>^ validator,  
    unsigned char value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<Of <(Byte)>>)
The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.

value
    Type: System..:::Byte
    The invalid value to compare with.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsNotEqualTo Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotEqualTo Method (ConditionValidator<Of.Byte>, Byte, String)

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Byte), _
    value As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)
```

**C#**

```csharp
public static ConditionValidator<byte> IsNotEqualTo(
    ConditionValidator<byte> validator,
    byte value,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<unsigned char>* IsNotEqualTo(
    ConditionValidator<unsigned char>* validator,
    unsigned char value,
    String* conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(
```

### Parameters

- **validator**
  Type: `ConditionValidator(Of <(T)>)`
  The `ConditionValidator(Of <(T)>)` that holds the value that has to be checked.

- **value**
Type: **System::Byte**
The invalid value to compare with.

**conditionDescription**
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**

The specified validator instance.
Exceptions

- **System..::.ArgumentException**

  Thrown when the `Value` of the specified validator is equal to value, while the specified validator is created using the `Requires` extension method.

- **CuttingEdge.Conditions..::.PostconditionException**

  Thrown when the `Value` of the specified validator is equal to value, while the specified validator is created using the `Ensures` extension method.
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://www.cuttingedge-conditions.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of DateTime), _
    value As DateTime _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsNotEqualTo(ConditionValidator<DateTime> validator, DateTime value)

Visual C++

public:
static ConditionValidator<DateTime>^ IsNotEqualTo(ConditionValidator<DateTime>^ validator, DateTime^ value)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(DateTime)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
    Type: System..::.DateTime
    The invalid value to compare with.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotEqualTo Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::: IsNotEqualTo Method (ConditionValidator(Of <(DateTime)>), DateTime, String)

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of DateTime), _
    value As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

**C#**

public static ConditionValidator<DateTime> IsNotEqualTo(ConditionValidator<DateTime> validator, DateTime value, string conditionDescription)

**Visual C++**

public:
static ConditionValidator<DateTime>^ IsNotEqualTo(ConditionValidator<DateTime>^ validator, DateTime value, String^ conditionDescription)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

**Parameters**

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(DateTime)>)>  
The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

value
Type: `System::DateTime`
The invalid value to compare with.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
 CutterEdge.Conditions reference library

ValidatorExtensions..:::.IsNotEqualTo Method (ConditionValidator<(Of <(Decimal)>), Decimal)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:**  CutterEdge.Conditions  
**Assembly:**  CutterEdge.Conditions (in CutterEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Decimal), _
    value As Decimal _
) As ConditionValidator(Of Decimal)
```

### C#

```csharp
public static ConditionValidator<decimal> IsNotEqualTo(
    ConditionValidator<decimal> validator,
    decimal value
)
```

### Visual C++

```csharp
public: ConditionValidator<decimal>^ IsNotEqualTo(
    ConditionValidator<decimal>^ validator,
    Decimal value
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions..::.ConditionValidator<Of <(Decimal)>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**value**
- Type: `System..::.Decimal`
- The invalid value to compare with.
**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator is equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
**Visual Basic**  **C#**  **Visual C++**  **JavaScript**

CuttingEdge.Conditions reference library

**ValidatorExtensions..:::IsNotEqualTo Method** (ConditionValidator<Of (Decimal>), Decimal, String)

*ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:**  **CuttingEdge.Conditions**  **Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Decimal), _
    value As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)
```

**C#**

```csharp
public static ConditionValidator<decimal> IsNotEqualTo(
    ConditionValidator<decimal> validator,
    decimal value,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<Decimal>^ IsNotEqualTo(
    ConditionValidator<Decimal>^ validator,
    Decimal^ value,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(
```

**Parameters**

**validator**

Type: `CuttingEdge.Conditions...:::ConditionValidator<Of <(Decimal)>>)`

The `ConditionValidator<Of <(T)>>)` that holds the value that has to be checked.

**value**
Type: `System::Decimal`
The invalid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Double), _
    value As Double ) As ConditionValidator(Of Double)
```

#### C#

```csharp
public static ConditionValidator<double> IsNotEqualTo(
    ConditionValidator<double> validator,
    double value
)
```

#### Visual C++

```cpp
public:
static ConditionValidator<double>^ IsNotEqualTo(
    ConditionValidator<double>^ validator,
    double value
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions:::ConditionValidator<Of <(Double)>>`

The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**value**

Type: `System:::Double`

The invalid value to compare with.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentException</td>
<td>Thrown when the Value of the specified validator is equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Double), _
    value As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsNotEqualTo(
    ConditionValidator<double> validator,
    double value,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<double>^ IsNotEqualTo(
    ConditionValidator<double>^ validator,
    double^ value,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(\n
Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Double)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type:  System:::Double
The invalid value to compare with.

conditionDescription
Type:  System:::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is equal to value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is equal to value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsNotEqualTo Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Short), _
    value As Short _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsNotEqualTo(  
    ConditionValidator<short> validator,  
    short value
)

Visual C++

public:  
    static ConditionValidator<short>^ IsNotEqualTo(  
        ConditionValidator<short>^ validator,  
        short value
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(Int16)>>
    The ConditionValidator<Of <(T)>>() that holds the value that has to be checked.

value
    Type: System..::.Int16
    The invalid value to compare with.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.EventArgs</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotEqualTo Method (ConditionValidator< OF <(Int16)>, Int16, String)

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Short), _
    value As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsNotEqualTo(
    ConditionValidator<short> validator,
    short value,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<short>^ IsNotEqualTo(
    ConditionValidator<short>^ validator,
    short value,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(Int16)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: `System::Int16`
The invalid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Requires</code> extension method. Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td></td>
</tr>
</tbody>
</table>

---
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions:::IsNotEqualTo Method (ConditionValidator<Of <(Int32)>, Int32)

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Integer), _
    value As Integer _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotEqualTo(
    ConditionValidator<int> validator,
    int value
)

Visual C++

public:
static ConditionValidator<int>*^ IsNotEqualTo(
    ConditionValidator<int>*^ validator,
    int value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

Parameters

- **validator**
  Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Int32)>)
  The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.

- **value**
  Type: System..::.Int32
  The invalid value to compare with.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

* [ValidatorExtensions Class](#)
* [IsNotEqualTo Overload](#)
* [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotEqualTo Method (ConditionValidator<(Of <(Int32))). Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Integer), _
    value As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotEqualTo(
    ConditionValidator<int> validator,
    int value,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsNotEqualTo(
    ConditionValidator<int>^ validator,
    int^ value,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(Int32)>>
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

value
Type: `System::Int32`
The invalid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [IsNotEqualTo Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions.IsNotEqualTo Method (ConditionValidator<Of <(Int64)>, Int64)

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Long), _
    value As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsNotEqualTo(
    ConditionValidator<long> validator,
    long value
)

Visual C++

public:
static ConditionValidator<long long>^ IsNotEqualTo(
    ConditionValidator<long long>^ validator,
    long_long value
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

Parameters

validator
    Type: CuttingEdge.Conditions.:::ConditionValidator(Of <(Int64)>)
    The ConditionValidator(Of <(T)>)) that holds the value that has to be checked.

value
    Type: System.:::Int64
    The invalid value to compare with.
**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Long), _
    value As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)

### C#

public static ConditionValidator<long> IsNotEqualTo( 
    ConditionValidator<long> validator,
    long value,
    string conditionDescription
)

### Visual C++

public: 
static ConditionValidator<long long>^ IsNotEqualTo( 
    ConditionValidator<long long>^ validator,
    long long value,
    String^ conditionDescription
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(

### Parameters

#### validator

Type: CuttingEdge.Conditions...:::ConditionValidator(Of <(Int64)>)

The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

#### value
Type: `System::Int64`
The invalid value to compare with.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotEqualTo Method (ConditionValidator<Of (Single>, Single)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotEqualTo ( _
  validator As ConditionValidator(Of Single), _
  value As Single _
) As ConditionValidator(Of Single)
```

#### C#

```csharp
public static ConditionValidator<float> IsNotEqualTo(
  ConditionValidator<float> validator,
  float value
)
```

#### Visual C++

```cpp
public: ConditionValidator<float>^ IsNotEqualTo(
  ConditionValidator<float>^ validator,
  float value
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions...:::ConditionValidator<(Of <(Single)>)>`

The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**value**

Type: `System...:::Single`

The invalid value to compare with.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotEqualTo Method (ConditionValidator<Of <(Single)>, Single, String)

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

```
Public Shared Function IsNotEqualTo ( _
    validator As ConditionValidator(Of Single), _
    value As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)
```

**C#**

```csharp
public static ConditionValidator<float> IsNotEqualTo(
    ConditionValidator<float> validator,
    float value,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<float>^ IsNotEqualTo(
    ConditionValidator<float>^ validator,
    float value,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotEqualTo = function(
```

**Parameters**

**validator**

Type: `CuttingEdge.Conditions..::.ConditionValidator<Of (Single)>`

The `ConditionValidator<Of (<T>)>` that holds the value that has to be checked.

**value**
Type: System::Single
The invalid value to compare with.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotEqualTo<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>))>, Nullable<(Of <(T)>))>

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    value As Nullable(Of T) _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotEqualTo<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> value
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotEqualTo(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T> value
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of (Nullable<(Of (T)>)>)>)
    The ConditionValidator<(Of (T)>) that holds the value that has to be checked.
value
    Type: System::Nullable<(Of <(T)>))
The invalid value to compare with.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to <code>value</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator and the <code>value</code> are both null references, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Constraints..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to <code>value</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Validates whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    value As Nullable(Of T), _
    conditionDescription As String _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotEqualTo<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> value,
    string conditionDescription
}

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotEqualTo(T>(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T>^ value,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions::$_::ConditionValidator<(Of Nullable<(Of T))>)
The ConditionValidator<(Of T)> that holds the value that has to be
checked.

value
   Type: System::Nullable<T>
   The invalid value to compare with.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to <code>value</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator and the value are both null references, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to <code>value</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo(Of T As {Structure, New}) ( _
   validator As ConditionValidator(Of Nullable(Of T)), _
   value As T _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotEqualTo<T>(
   ConditionValidator<Nullable<T>> validator,
   T value
)
where T : struct, new()

Visual C++

public:
   generic<typename T>
   where T: value class, gcnew()
   static ConditionValidator<Nullable<T>>^ IsNotEqualTo(ConditionValidator<Nullable<T>>^ validator,
   T value
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
   Type: CuttingEdge.Conditions..::.ConditionValidator<(Of ((Nullable<(Of <(T)>)>))>
   The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
value
  Type: T
  The invalid value to compare with.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**  
**IsNotEqualTo Overload**  
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:) on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo(Of T As {Structure, New}) ( _ 
    validator As ConditionValidator(Of Nullable(Of T)), _ 
    value As T, _ 
    conditionDescription As String 
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotEqualTo<T>( 
    ConditionValidator<Nullable<T>> validator, 
    T value, 
    string conditionDescription 
) 
where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotEqualTo( 
        ConditionValidator<Nullable<T>>^ validator, 
        T value, 
        String^ conditionDescription 
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions::$_::ConditionValidator<(Of <(Nullable<(Of <(T)>)>))>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

value
Type: T
The invalid value to compare with.

conditionDescription
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.:::IsNotEqualTo<(Of <(T)>)> Method
(ConditionValidator<(Of <(T)>)>), T)

Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    value As T _) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotEqualTo<T>(
    ConditionValidator<T> validator,
    T value
) where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsNotEqualTo(%
        ConditionValidator<T>^ validator,
        T value
    )%

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(T)>)>
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

value
Type: T
The invalid value to compare with.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator and the value are both null references, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel:::InvalidEnumArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is equal to value and are of type <code>Enum</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
</tbody>
</table>
validator is created using the \texttt{Requires} extension method.

Thrown when the \texttt{Value} of the specified validator is equal to value, while the specified validator is created using the \texttt{Ensures} extension method.
See Also

[ValidatorExtensions Class](#)
[IsNotEqualTo Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Checks whether the given value is unequal to the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotEqualTo(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    value As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotEqualTo<T>(
    ConditionValidator<T> validator,
    T value,
    string conditionDescription
)
where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsNotEqualTo(
        ConditionValidator<T>^ validator,
        T value,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions,::,ConditionValidator<(Of <(T)>)
The ConditionValidator<(Of <(T)> )> that holds the value that has to be checked.
value
   Type: T
   The invalid value to compare with.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the
   placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator and the value are both null references, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System.ComponentModel..::.InvalidEnumArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is equal to value and are of type <strong>Enum</strong>, while the specified</td>
</tr>
</tbody>
</table>
**PostconditionException**

Thrown when the Value of the specified validator is equal to value, while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
IsNotEqualTo Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions.........IsNotGreaterOrEqual Method
ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotGreaterOrEqual(ConditionValidator&lt;Of[Byte&gt;(), Byte)</td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterOrEqual(ConditionValidator&lt;Of[DateTime&gt;(), DateTime)</td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterOrEqual(ConditionValidator&lt;Of[Decimal()], Decimal)</td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterOrEqual(ConditionValidator&lt;Of[Double&gt;(), Double)</td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterOrEqual(ConditionValidator&lt;Of[Int16()], Int16)</td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
**IsNotGreaterOrEqual**(ConditionValidator<Of<Int32>>, Int32) greater or equal to the specified maxValue. An exception is thrown otherwise. Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**IsNotGreaterOrEqual**(ConditionValidator<Of<Int64>>, Int64) Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**IsNotGreaterOrEqual**<Of<(T)>>(ConditionValidator<Of<(Nullable<Of<(T)>)>), Nullable<Of<(T)>)>), Nullable<Of<(T)>>) Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**IsNotGreaterOrEqual**<Of<(T)>>(ConditionValidator<Of<(Nullable<Of<(T)>)>), T>) Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**IsNotGreaterOrEqual**(ConditionValidator<Of<(Single)>), Single) Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**IsNotGreaterOrEqual**<Of<(T)>>(ConditionValidator<Of<(T)>), T) Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.
<table>
<thead>
<tr>
<th><code>(Byte)</code>, <code>Byte</code>, <code>String</code></th>
<th>exception is thrown otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNotGreaterOrEqual(ConditionValidator&lt;(Of </code>&lt;DateTime&gt;<code>), DateTime, String)</code></td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotGreaterOrEqual(ConditionValidator&lt;(Of </code>&lt;Decimal&gt;<code>), Decimal, String)</code></td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotGreaterOrEqual(ConditionValidator&lt;(Of </code>&lt;Double&gt;<code>), Double, String)</code></td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotGreaterOrEqual(ConditionValidator&lt;(Of </code>&lt;Int16&gt;<code>), Int16, String)</code></td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotGreaterOrEqual(ConditionValidator&lt;(Of </code>&lt;Int32&gt;<code>), Int32, String)</code></td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotGreaterOrEqual(ConditionValidator&lt;(Of </code>&lt;Int64&gt;<code>), Int64, String)</code></td>
<td>Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
IsNotGreaterOrEqual(Of <(T)>) (ConditionValidator(Of <(Nullable(Of <(T)>)>)), Nullable(Of <(T)>)>, String) exception is thrown otherwise.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

IsNotGreaterOrEqual(Of <(T)>) (ConditionValidator(Of <(Nullable(Of <(T)>)>)), T, String) exception is thrown otherwise.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

IsNotGreaterOrEqual(Of <(Single)>) (ConditionValidator(Of <(Single)>)>, Single, String) exception is thrown otherwise.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

IsNotGreaterOrEqual(Of <(T)>) (ConditionValidator(Of <(T)>)>, T, String) exception is thrown otherwise.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsNotGreaterOrEqual Method (ConditionValidator<Of (Byte)>, Byte)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Byte), _
    maxValue As Byte)
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsNotGreaterOrEqual( 
    ConditionValidator<byte> validator, 
    byte maxValue
)

Visual C++

public: ConditionValidator<unsigned char>^ IsNotGreaterOrEqual( 
    ConditionValidator<unsigned char>^ validator, 
    unsigned char maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator(Of (Byte))
  The ConditionValidator(Of (T)) that holds the value that has to be checked.

maxValue
  Type: System..::.Byte
  The lowest invalid value.
Return Value

The specified validator instance.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions...::: IsNotGreaterOrEqual Method (ConditionValidator<Of <(Byte)>), Byte, String)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
  validator As ConditionValidator(Of Byte), _
  maxValue As Byte, _
  conditionDescription As String _) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsNotGreaterOrEqual(
  ConditionValidator<byte> validator,
  byte maxValue,
  string conditionDescription
)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsNotGreaterOrEqual( 
  ConditionValidator<unsigned char>^ validator,
  unsigned char maxValue,
  String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator
  Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Byte)>)>)
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: `System::Byte`
The lowest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotGreaterOrEqual Method (ConditionValidator<Of <(DateTime)>), DateTime)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime _
) As ConditionValidator(Of DateTime)
```

**C#**

```csharp
public static ConditionValidator<DateTime> IsNotGreaterOrEqual(
    ConditionValidator<DateTime> validator,
    DateTime maxValue
)
```

**Visual C++**

```cpp
public: ConditionValidator<DateTime>^ IsNotGreaterOrEqual(
    ConditionValidator<DateTime>^ validator,
    DateTime^ maxValue
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur
```

**Parameters**

- **validator**
  - Type: `CuttingEdge.Conditions::ConditionValidator<Of <DateTime>>`
  - The `ConditionValidator<Of <T>>` that holds the value that has to be checked.

- **maxValue**
  - Type: `System::DateTime`
  - The lowest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..:::IsNotGreaterOrEqual Method (ConditionValidator<Of <(DateTime)>, DateTime, String)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsNotGreaterOrEqual(ConditionValidator<DateTime> validator, DateTime maxValue, String conditionDescription)

Visual C++

public: static ConditionValidator<DateTime>^ IsNotGreaterOrEqual(ConditionValidator<DateTime>^ validator, DateTime^ maxValue, String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of (DateTime))>
The ConditionValidator(Of (T)) that holds the value that has to be checked.

maxValue
Type: `System::DateTime`
The lowest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions...::: IsNotGreaterOrEqual Method (ConditionValidator<Of <(Decimal)>, Decimal)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
validator As ConditionValidator(Of Decimal), _
maxValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsNotGreaterOrEqual(
ConditionValidator<decimal> validator,
decimal maxValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsNotGreaterOrEqual(
ConditionValidator<Decimal>^ validator,
Decimal^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator

Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Decimal)>)>
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue

Type: System..::: Decimal
The lowest invalid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Decimal), _
    maxValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

### C#

public static ConditionValidator<decimal> IsNotGreaterOrEqual(ConditionValidator<decimal> validator,
    decimal maxValue,
    string conditionDescription
)

### Visual C++

public:
static ConditionValidator<Decimal>^ IsNotGreaterOrEqual(ConditionValidator<Decimal>^ validator,
    Decimal^ maxValue,
    String^ conditionDescription
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

## Parameters

- **validator**
  - Type: `CuttingEdge.Conditions::.ConditionValidator<(Of <(Decimal)>)>`
  - The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

- **maxValue**

- **conditionDescription**
  - String
Type: `System::Decimal`
The lowest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotGreaterOrEqual Method (ConditionValidator<Of <(Double)>), Double)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Double), _
    maxValue As Double _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsNotGreaterOrEqual( ConditionValidator<double> validator, double maxValue
)

Visual C++

public: ConditionValidator<double>^ IsNotGreaterOrEqual( ConditionValidator<double>^ validator, double maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator
    Type: CuttingEdge.Conditions::: ConditionValidator(Of &lt;(Double)&gt;)
    The ConditionValidator(Of &lt;(T)&gt;) that holds the value that has to be checked.

maxValue
    Type: System:::Double
    The lowest invalid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to</td>
</tr>
<tr>
<td></td>
<td><code>maxValue</code>, while the specified validator is created using the <code>Requires</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td><code>maxValue</code>, while the specified validator is created using the <code>Ensures</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotGreaterOrEqual Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions.::.IsNotGreaterOrEqual Method (ConditionValidator<((Of <(Double)>)), Double, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Double), _
    maxValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

**C#**

public static ConditionValidator<
double> IsNotGreaterOrEqual(
    ConditionValidator<
double> validator,
    double maxValue,
    string conditionDescription
)

**Visual C++**

public:
static ConditionValidator<
double>*^ IsNotGreaterOrEqual(
    ConditionValidator<
double>*^ validator,
    double*^ maxValue,
    String* conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

**Parameters**

validator

Type: CuttingEdge.Conditions::$_::ConditionValidator<(Of<(Double)>))

The ConditionValidator<(Of<(T)>>) that holds the value that has to be checked.

maxValue
Type: System::Double
The lowest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotGreaterOrEqual Method (ConditionValidator<Of<(Int16)>), Int16)

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Short), _
    maxValue As Short _
) As ConditionValidator(Of Short)
```

#### C#

```csharp
public static ConditionValidator<short> IsNotGreaterOrEqual(
    ConditionValidator<short> validator,
    short maxValue
)
```

#### Visual C++

```cpp
public: ConditionValidator<short>^ IsNotGreaterOrEqual(
    ConditionValidator<short>^ validator,
    short maxValue
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions..:::ConditionValidator<Of Int16)>

The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**maxValue**

Type: `System..:::Int16`

The lowest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::: IsNotGreaterOrEqual Method (ConditionValidator<Of <(Int16)>>, Int16, String)

**ValidatorExtensions Class**  See Also  Send Feedback

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Short), _
    maxValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)
```

#### C#

```csharp
public static ConditionValidator<short> IsNotGreaterOrEqual(
    ConditionValidator<short> validator,
    short maxValue,
    string conditionDescription
)
```

#### Visual C++

```cpp
public:
static ConditionValidator<short>^ IsNotGreaterOrEqual(  
    ConditionValidator<short>^ validator,
    short^ maxValue,  
    String^ conditionDescription
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur
```

### Parameters

- **validator**
  - Type: `ConditionValidator<Of <(Int16)>)`
  - The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

- **maxValue**
Type: `System::Int16`
The lowest invalid value.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[Link to ValidatorExtensions Class]
[Link to IsNotGreaterOrEqual Overload]
[Link to CuttingEdge.Conditions Namespace]

Send [Feedback] on this topic to Microsoft.
ValidatorExtensions..::..IsNotGreaterOrEqual Method (ConditionValidator<Of <(Int32)>, Int32)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Integer), _
    maxValue As Integer _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotGreaterOrEqual(
    ConditionValidator<int> validator,
    int maxValue
)

Visual C++

public:
static ConditionValidator<int>^ IsNotGreaterOrEqual(
    ConditionValidator<int>^ validator,
    int maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Int32)>)
The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.

maxValue
Type: System..::.Int32
The lowest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <em>Value</em> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <em>Value</em> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsNotGreaterOrEqual Method (ConditionValidator<,(Int32), Int32, String)

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Integer), _
    maxValue As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotGreaterOrEqual(
    ConditionValidator<int> validator,
    int maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsNotGreaterOrEqual(  
    ConditionValidator<int>^ validator,
    int^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = function

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int32)>))
The ConditionValidator<(Of <(T)>>) that holds the value that has to be checked.

maxValue
Type: \texttt{System::Int32}
The lowest invalid value.

conditionDescription
Type: \texttt{System::String}
The description of the condition that should hold. The string may hold the placeholder '{0}' for the \texttt{ArgumentName}.

\textbf{Return Value}

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
NotGreaterOrEqual Method (ConditionValidator<
(Int64>)>, Int64)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater or equal to the specified
maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Long), _
    maxValue As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsNotGreaterOrEqual(
    ConditionValidator<long> validator,
    long maxValue
)

Visual C++

public:
static ConditionValidator<long long>^ IsNotGreaterOrEqual(
    ConditionValidator<long long>^ validator,
    long long maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator
   Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Int64)>)
   The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

maxValue
   Type: System..::.Int64
   The lowest invalid value.
**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Long), _
    maxValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsNotGreaterOrEqual(
    ConditionValidator<long> validator,
    long maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<long long>^ IsNotGreaterOrEqual(
    ConditionValidator<long long>^ validator,
    long long maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int64)>))

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: System::Int64
The lowest invalid value.

columnDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...::: IsNotGreaterOrEqual Method (ConditionValidator<Of <(Single)>), Single)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Single), _
    maxValue As Single _
) As ConditionValidator(Of Single)
```

**C#**

```csharp
public static ConditionValidator<float> IsNotGreaterOrEqual(
    ConditionValidator<float> validator,
    float maxValue
)
```

**Visual C++**

```cpp
public: ConditionValidator<float>^ IsNotGreaterOrEqual(
    ConditionValidator<float>^ validator,
    float maxValue
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur
```

**Parameters**

**validator**
Type: `CuttingEdge.Conditions.:::ConditionValidator<Of <(Single)>>`
The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**maxValue**
Type: `System.:::Single`
The lowest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">System..::.ArgumentOutOfRangeException</a></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><a href="#">CuttingEdge.Conditions..::.PostconditionException</a></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...::: IsNotGreaterOrEqual Method (ConditionValidator<((Of <(Single)>)>, Single, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotGreaterOrEqual ( _
    validator As ConditionValidator(Of Single), _
    maxValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)
```

### C#

```csharp
public static ConditionValidator<float> IsNotGreaterOrEqual(
    ConditionValidator<float> validator,
    float maxValue,
    string conditionDescription
)
```

### Visual C++

```cpp
public:
static ConditionValidator<float>^ IsNotGreaterOrEqual(
    ConditionValidator<float>^ validator,
    float^ maxValue,
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterOrEqual = fur
```

## Parameters

**validator**
- Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Single)>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**maxValue**
Type: System.Single
The lowest invalid value.

conditionDescription
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual(Of T As {Structure, New}
  validator As ConditionValidator(Of Nullable(Of T)), _
  maxValue As Nullable(Of T) _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotGreaterOrEqual<T>
  ConditionValidator<Nullable<T>> validator,
  Nullable<T> maxValue
)
where T : struct, new()

Visual C++

public:
  generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotGreaterOrEqual(ConditionValidator<Nullable<T>>^ validator,
  Nullable<T>^ maxValue
)"

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator<(Of (Nullable<(Of
  <T)>)>)

The ConditionValidator<(Of <T>)> that holds the value that has to be checked.
maxValue
Type: System::Nullable<(Of <(T)>))
The lowest invalid value.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual(Of T As {Structure, New};
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As Nullable(Of T), _
    conditionDescription As String
) As ConditionValidator(Of Nullable(Of T))

C#

d

public static ConditionValidator<Nullable<T>> IsNotGreaterOrEqual<T;
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> maxValue,
    string conditionDescription
)
where T : struct, new()

Visual C++

d

generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotGreaterOrEqual(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T>^ maxValue,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of
<T>())>))>

The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

maxValue
Type: System::Nullable<(Of <(T)>)
The lowest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T
   The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to maxValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual(Of T As {Structure, New}',
validator As ConditionValidator(Of Nullable(Of T)), _
maxValue As T _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotGreaterOrEqual<T>: ConditionValidator<Nullable<T>> validator,
T maxValue

where T : struct, new()

Visual C++

public:
  generic<typeparamname T>
  where T : value class, gcnew()
  static ConditionValidator<Nullable<T>>^ IsNotGreaterOrEqual(
    ConditionValidator<Nullable<T>>^ validator,
    T maxValue
  )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions...,:ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
maxValue
    Type: T
    The lowest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual(Of T As {Structure, New})
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As T, _
    conditionDescription As String
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotGreaterOrEqual<T>:
    ConditionValidator<Nullable<T>> validator,
    T maxValue,
    string conditionDescription
)
where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotGreaterOrEqual(ConditionValidator<Nullable<T>>^ validator,
    T maxValue,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of Nullable<(Of T)>>
    The ConditionValidator<(Of T)> that holds the value that has to be
maxValue
   Type: T
   The lowest invalid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to</td>
</tr>
<tr>
<td></td>
<td><code>maxValue</code>, while the specified validator is created using the <code>Requires</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..:::PostconditionException</code></td>
<td><code>maxValue</code>, while the specified validator is created using the <code>Ensures</code></td>
</tr>
<tr>
<td></td>
<td>extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater or equal to the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotGreaterOrEqual<T>(
    ConditionValidator<T> validator,
    T maxValue
)

where T : IComparable

Visual C++

public:
    generic<typename T>
where T : IComparable
static ConditionValidator<T>^ IsNotGreaterOrEqual(
    ConditionValidator<T>^ validator,
    T maxValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions:::ConditionValidator(Of (Of (T)>))
The ConditionValidator(Of (Of (T)>)) that holds the value that has to be checked.

maxValue
Type: $T$
The lowest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is greater or equal to maxValue and is an Enum type, while the specified validator is created using the Requires extension method.</td>
</tr>
</tbody>
</table>

```csharp
System..::.ArgumentOutOfRangeException

System.ComponentModel..::.InvalidEnumArgumentException
```
CuttingEdge.Conditions...;

PostconditionException while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
IsNotGreaterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...::: IsNotGreaterOrEqual<Of <(T)>>) Method
(ConditionValidator<Of <(T)>>, T, String)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterOrEqual(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotGreaterOrEqual<T>(
    ConditionValidator<T> validator,
    T maxValue,
    string conditionDescription
)

where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsNotGreaterOrEqual(
        ConditionValidator<T>^ validator,
        T maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::..ConditionValidator(Of <(T)>)

The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.
maxValue
Type: T
The lowest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
### Type Parameters

$T$

The type of the `Value` of the specified validator.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel..::.InvalidEnumArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater or equal to <code>maxValue</code> and is an <code>Enum</code> type, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
</tbody>
</table>
CuttingEdge.Conditions...\PostconditionException

while the specified validator is created using the \Ensure extension method.
See Also

[ValidatorExtensions Class]
[IsNotGreaterOrEqual Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript  Include Protected Members  Include Inherited Members

CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsNotGreaterThan Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotGreaterThan(ConditionValidator&lt;Of &lt;(Byte)&gt;, Byte)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator&lt;Of &lt;(DateTime)&gt;, DateTime)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator&lt;Of &lt;(Decimal)&gt;, Decimal)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator&lt;Of &lt;(Double)&gt;, Double)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator&lt;Of &lt;(Int16)&gt;, Int16)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator&lt;Of &lt;(Int32)&gt;, Int32)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator&lt;Of &lt;(Of)&gt;), Byte)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsNotGreaterThan(Int64, Int64)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(Of (T)&gt;, ConditionValidator(Of Nullable(Of (T)&gt;)&gt;), Nullable(Of (T)&gt;)&gt;), T</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(Of (T)&gt;, ConditionValidator(Of (Single)&gt;)&gt;, Single)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(Of (T)&gt;, ConditionValidator(Of (T)&gt;)&gt;, T)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator(Of (Byte)&gt;)&gt;, Byte, String)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator(Of (DateTime)&gt;)&gt;, DateTime, String)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotGreaterThan(ConditionValidator(Of (Decimal)&gt;)&gt;, Decimal, String)</td>
<td>Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
IsNotGreaterThan(ConditionValidator<(Of <(Double>)>), Double, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

IsNotGreaterThan(ConditionValidator<(Of <(Int16>)>), Int16, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

IsNotGreaterThan(ConditionValidator<(Of <(Int32>)>), Int32, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

IsNotGreaterThan(ConditionValidator<(Of <(Int64>)>), Int64, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

IsNotGreaterThan<(Of <(T)>)>
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>), Nullable<(Of <(T)>)>>, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

IsNotGreaterThan<(Of <(T)>)>
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>), T, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

IsNotGreaterThan(ConditionValidator<(Of <(Single)>)>, Single, String)

Checks whether the given
IsNotGreaterThan<(Of <(T)>)
(ConditionValidator<(Of <(T)>), T, String)

value is not greater than the specified maxValue. An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Byte), _
    maxValue As Byte)
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsNotGreaterThan(
    ConditionValidator<byte> validator,
    byte maxValue
)

Visual C++

public: ConditionValidator<unsigned char>* IsNotGreaterThan(
    ConditionValidator<unsigned char>* validator,
    unsigned char maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Byte)>)
    The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

maxValue
    Type: System..::.Byte
    The lowest valid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotGreaterThan Method (ConditionValidator<Of (Byte)>, Byte, String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Byte), _
    maxValue As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsNotGreaterThan(
    ConditionValidator<byte> validator,
    byte maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsNotGreaterThan(
    ConditionValidator<unsigned char>^ validator,
    unsigned char maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Byte)>)>)
    The ConditionValidator(Of <(T)>)) that holds the value that has to be checked.

maxValue
Type: `System::Byte`
The lowest valid value.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotGreaterThan (_
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime _
) As ConditionValidator(Of DateTime)
```

**C#**

```csharp
public static ConditionValidator<DateTime> IsNotGreaterThan(
    ConditionValidator<DateTime> validator,
    DateTime maxValue
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<DateTime>^ IsNotGreaterThan(
    ConditionValidator<DateTime>^ validator,
    DateTime^ maxValue
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function
```

### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions::ConditionValidator<Of `<DateTime>`>`
  - The `ConditionValidator<Of `<T>`>` that holds the value that has to be checked.

- **maxValue**
  - Type: `System::DateTime`
  - The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotGreaterThan Method (ConditionValidator<Of 
(DateTime)>, DateTime, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of DateTime), _
    maxValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsNotGreaterThan(
    ConditionValidator<DateTime> validator,
    DateTime maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsNotGreaterThanOrEqual( 
    ConditionValidator<DateTime>^ validator,
    DateTime^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = functi

Parameters

validator
    Type: CuttingEdge.Conditions...::ConditionValidator<(Of <(DateTime)>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: `System::DateTime`
The lowest valid value.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
**ValidatorExtensions...::IsNotGreaterThan Method** (ConditionValidator<Of <(Decimal)>, Decimal)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Decimal), _
    maxValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsNotGreaterThan(ConditionValidator<decimal> validator, decimal maxValue)

Visual C++

public:
    static ConditionValidator<decimal>^ IsNotGreaterThan(ConditionValidator<decimal>^ validator, Decimal maxValue)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Decimal)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
    Type: System..:::Decimal
    The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions..::: IsNotGreaterThan Method (ConditionValidator(Of Decimal), Decimal, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Decimal), _
    maxValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsNotGreaterThan(
    ConditionValidator<decimal> validator,
    decimal maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<Decimal>* IsNotGreaterThan(
    ConditionValidator<Decimal>* validator,
    Decimal* maxValue,
    String* conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function (

Parameters

validator
    Type: CuttingEdge.Conditions.::.ConditionValidator(Of <(Decimal)>)

The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

maxValue
Type: `System::::Decimal`
The lowest valid value.

`conditionDescription`
Type: `System::::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsNotGreaterThan Method (ConditionValidator<Of <(Double)>, Double)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Double), _
    maxValue As Double _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<
double> IsNotGreaterThan(
    ConditionValidator<
double> validator,
    double maxValue
)

Visual C++

public:
    static ConditionValidator<
double>^ IsNotGreaterThan(
        ConditionValidator<
double>^ validator,
        double maxValue
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Double)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
    Type: System..::: Double
    The lowest valid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than maxValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is greater than maxValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsNotGreaterThan Method (ConditionValidator<Of <(Double)>), Double, String)

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Double), _
    maxValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)
```

### C#

```csharp
public static ConditionValidator<double> IsNotGreaterThan(
    ConditionValidator<double> validator,
    double maxValue,
    string conditionDescription
)
```

### Visual C++

```cpp
public:
static ConditionValidator< double >^ IsNotGreaterThan(
    ConditionValidator< double >^ validator,
    double ^ maxValue,
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions.:::ConditionValidator<Of <(Double)>>`
  - The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**maxValue**
Type: `System::Double`
The lowest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic □ C#
□ Visual C++
□ JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::: IsNotGreaterThan Method (ConditionValidator<Of <(Int16)>, Int16)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Short), _
    maxValue As Short)
) As ConditionValidator(Of Short)
```

**C#**

```csharp
public static ConditionValidator<short> IsNotGreaterThan(ConditionValidator<short> validator,
    short maxValue
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<short>^ IsNotGreaterThan(ConditionValidator<short>^ validator,
    short maxValue
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function
```

**Parameters**

**validator**

Type: `CuttingEdge.Conditions:::ConditionValidator<Of <(Int16)>)`
The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**maxValue**

Type: `System:::Int16`
The lowest valid value.
Return Value

The specified validator instance.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>System,::,.ArgumentException</em></td>
<td>Thrown when the <em>Value</em> of the specified validator is greater than maxValue, while the specified validator is created using the <em>Requires</em> extension method.</td>
</tr>
<tr>
<td><em>CuttingEdge.Conditions,::,.PostconditionException</em></td>
<td>Thrown when the <em>Value</em> of the specified validator is greater than maxValue, while the specified validator is created using the <em>Ensures</em> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..:::.IsNotGreaterThan Method (ConditionValidator<(Of <(Int16)>), Int16, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Short), _
    maxValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)

**C#**

public static ConditionValidator<short> IsNotGreaterThan(
    ConditionValidator<short> validator,
    short maxValue,
    string conditionDescription
)

**Visual C++**

public:
static ConditionValidator<short>^ IsNotGreaterThan(
    ConditionValidator<short>^ validator,
    short^ maxValue,
    String^ conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = functj

**Parameters**

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<Of <(Int16)>>)
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

maxValue
Type: **System::Int16**
The lowest valid value.

**conditionDescription**
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**
The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsNotGreaterThan Method (ConditionValidator<Of <(Int32)>, Int32)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Integer), _
    maxValue As Integer )
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotGreaterThan(
    ConditionValidator<int> validator,
    int maxValue
)

Visual C++

public:
static ConditionValidator<int>^ IsNotGreaterThan(
    ConditionValidator<int>^ validator,
    int maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of Int32)>)
The ConditionValidator<(Of <T>)> that holds the value that has to be checked.

maxValue
    Type: System..::.Int32
The lowest valid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsNotGreaterThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Checking whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Integer), _
    maxValue As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotGreaterThan(
    ConditionValidator<int> validator,
    int maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsNotGreaterThan(
    ConditionValidator<int>^ validator,
    int^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Int32)>))
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: System::Int32
The lowest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsNotGreaterThan (  
    validator As ConditionValidator(Of Long), _  
    maxValue As Long   _  
) As ConditionValidator(Of Long)

**C#**

```csharp
public static ConditionValidator<long> IsNotGreaterThan(
    ConditionValidator<long> validator,
    long maxValue
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<long_long>^ IsNotGreaterThan(
    ConditionValidator<long_long>^ validator,
    long_long maxValue
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function
```

### Parameters

**validator**  
Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Int64)>>`  
The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**maxValue**  
Type: `System..:::Int64`  
The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions..::.IsNotGreaterThan Method (ConditionValidator<Of <(Int64)>, Int64, String)

**ValidatorExtensions Class**  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Long), _
    maxValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsNotGreaterThan( 
    ConditionValidator<long> validator, 
    long maxValue, 
    string conditionDescription 
)

Visual C++

public: 
static ConditionValidator<long long>^ IsNotGreaterThan( 
    ConditionValidator<long long>^ validator, 
    long long maxValue, 
    String^ conditionDescription 
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Int64)>)}
    The ConditionValidator(Of <(T)>)} that holds the value that has to be checked.

maxValue
Type: `System::Int64`
The lowest valid value.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsNotGreaterThan Method (ConditionValidator<Of <(Single)>), Single)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Single), _
    maxValue As Single _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotGreaterThan( 
    ConditionValidator<float> validator, 
    float maxValue
)

Visual C++

public: ConditionValidator<float>^ IsNotGreaterThan( 
    ConditionValidator<float>^ validator, 
    float maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = the

Parameters

validator
Type: CuttingEdge.Conditions::< ConditionValidator<Of <<Single>>>
The ConditionValidator<Of <<T>>>() that holds the value that has to be checked.

maxValue
Type: System::<Single
The lowest valid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsNotGreaterThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library
ValidatorExtensions..:::.IsNotGreaterThan Method (ConditionValidator<Of <(Single)>), Single, String)

ValidatorExtensions Class   See Also   Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsNotGreaterThan ( _
    validator As ConditionValidator(Of Single), _
    maxValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

**C#**

public static ConditionValidator<float> IsNotGreaterThan(
    ConditionValidator<float> validator,
    float maxValue,
    string conditionDescription
)

**Visual C++**

public:
static ConditionValidator<float>^ IsNotGreaterThan(
    ConditionValidator<float>^ validator,
    float^ maxValue,
    String^ conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNotGreaterThan = function

### Parameters

**validator**

- **Type:** CuttingEdge.Conditions...:::ConditionValidator<(Of <(Single)>>)
- The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

**maxValue**
Type: **System:: Single**
The lowest valid value.

**conditionDescription**
Type: **System:: String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- IsNotGreaterThan Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic □ C#
□ Visual C++
□ JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::: IsNotGreaterThan<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Nullable<(Of <(T)>)>)>)>), Nullable<(Of <(T)>)>)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan(Of T As {Structure, New})(
    validator As ConditionValidator(Of Nullable(Of T)), _
    maxValue As Nullable(Of T)
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotGreaterThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> maxValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>:: IsNotGreaterThan(ConditionValidator<Nullable<T>>:: validator,
    Nullable<T> maxValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
maxValue
Type: System.Nullable<T>
The lowest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
validatorExtensions::IsNotGreaterThan<
(of <(T)>)> method
(ConditionValidator<(Of <(Nullable<
(of <(T)>))>)>, Nullable<
(of <(T)>)),
String)

validatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not greater than the specified maxValue. An
exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
maxValue As Nullable(Of T), _
conditionDescription As String) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotGreaterThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> maxValue,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotGreaterThan(
        ConditionValidator<Nullable<T>>^ validator,
        Nullable<T>^ maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of <(T)>))>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**maxValue**
Type: `System.Nullable<T>`
The lowest valid value.

**conditionDescription**
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotGreaterThan(Of (T)) Method
(ConditionValidator(Of (Nullable(Of (T))))>, T)

ValidatorExtensions Class   See Also   Send Feedback

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
maxValue As T _) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotGreaterThan<T>(
    ConditionValidator<Nullable<T>> validator, T maxValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotGreaterThan(
        ConditionValidator<Nullable<T>>^ validator,
        T maxValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
maxValue
  Type: T
  The lowest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotGreaterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsNotGreaterThan<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<((Of <(T)>)>)>)>, T, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _ maxValue As T, _ conditionDescription As String) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotGreaterThan<T>(ConditionValidator<Nullable<T>> validator, T maxValue, string conditionDescription)

where T : struct, new()

Visual C++

public:
    generic< typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotGreaterThan(ConditionValidator<Nullable<T>>^ validator,
        T maxValue,
        String^ conditionDescription)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions::::ConditionValidator<(Of Nullable<(Of T)>)>)
The ConditionValidator<(Of T)> that holds the value that has to be
checked.

maxValue
  Type: T
  The lowest valid value.

conditionDescription
  Type: System::String
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

ReturnValue

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is greater than <code>maxValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsNotGreaterThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions...:::.IsNotGreaterThan(Of <(T)>) Method
(ConditionValidator(Of <(T)>)>, T)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T _
) As ConditionValidator(Of T)

C#

class ConditionValidator<T> {
    public static ConditionValidator<T> IsNotGreaterThan<T>(
        ConditionValidator<T> validator,
        T maxValue
    )
}

where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsNotGreaterThan(ConditionValidator<T>^ validator,
                                                T maxValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(T)>)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxValue
Type: T
The lowest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel::::InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue and is an Enum type, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::::PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified</td>
</tr>
</tbody>
</table>
validator is created using the
Ensures extension method.
See Also

**ValidatorExtensions Class**  
**IsNotGreaterThan Overload**  
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions:::IsNotGreaterThan<Of<(T)>>() Method
(ConditionValidator<Of<(T)>()), T, String)

Checks whether the given value is not greater than the specified maxValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotGreaterThan(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    maxValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotGreaterThan<T>(
    ConditionValidator<T> validator,
    T maxValue,
    string conditionDescription
)

where T : IComparable

Visual C++

public:
    generic< typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsNotGreaterThan(
        ConditionValidator<T>^ validator,
        T maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(T)>)> The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
maxValue
Type: T
The lowest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel::InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue and is an Enum type, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is greater than maxValue, while the specified</td>
</tr>
</tbody>
</table>
validator is created using the `Ensures` extension method.
See Also

*ValidatorExtensions Class*
*IsNotGreaterThan Overload*
*CuttingEdge.Conditions Namespace*

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...::IsNotInfinity Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| IsNotInfinity(ConditionValidator<<Of
  <(Double)>>>)) | Checks whether the given value is not infinity. An exception is thrown otherwise. |
| IsNotInfinity(ConditionValidator<<Of
  <(Single)>>>)) | Checks whether the given value is not infinity. An exception is thrown otherwise. |
| IsNotInfinity(ConditionValidator<<Of
  <(Double)>>, String) | Checks whether the given value is not infinity. An exception is thrown otherwise. |
| IsNotInfinity(ConditionValidator<<Of
  <(Single)>>, String) | Checks whether the given value is not infinity. An exception is thrown otherwise. |
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsnNotInfinity Method (ConditionValidator<(Of <(Double)>=))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not infinity. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInfinity ( _
    validator As ConditionValidator(Of Double) _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsNotInfinity(
    ConditionValidator<double> validator
)

Visual C++

public:
static ConditionValidator<double>^ IsNotInfinity( 
    ConditionValidator<double>^ validator
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInfinity = function(

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator<Of (Of (Double)>)
    The ConditionValidator<Of (T)> that holds the value that has to be checked.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotInfinity Method (ConditionValidator<Of <(Double)>>, String)

Checks whether the given value is not infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsNotInfinity ( _
    validator As ConditionValidator(Of Double), _
    conditionDescription As String _
) As ConditionValidator(Of Double)

**C#**

```csharp
public static ConditionValidator<double> IsNotInfinity(
    ConditionValidator<double> validator,
    string conditionDescription
)
```

**Visual C++**

```csharp
public: ConditionValidator<double>^ IsNotInfinity(
    ConditionValidator<double>^ validator,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotInfinity = function(
```

### Parameters

**validator**

Type: `CuttingEdge.Conditions::ConditionValidator<Of <(Double)>)`

The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**conditionDescription**

Type: `System::String`

The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator is infinity, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is infinity, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInfinity ( _
    validator As ConditionValidator(Of Single) _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotInfinity(
    ConditionValidator<float> validator
)

Visual C++

public: 
static ConditionValidator<float>^ IsNotInfinity(
    ConditionValidator<float>^ validator
)

JavaScript

CuttingEdge.Conditions.SerializerExtensions.isNotInfinity = function(

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator(Of (Single)>)
The ConditionValidator(Of (T)> that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is infinity, while the specified validator is created using the <a href="#">Requires</a> extension method. Thrown when the <strong>Value</strong> of the specified validator is infinity, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotInfinity Method (ConditionValidator<Of <(Single)>, String)

Checks whether the given value is not infinity. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInfinity ( _
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotInfinity(
    ConditionValidator<float> validator,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsNotInfinity(
    ConditionValidator<float>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions-validatorExtensions.isNotInfinity = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Single)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be
cHECKED.

conditionDescription
    Type: System..::: String
    The description of the condition that should hold. The string may hold the
placeholder '{0}' for the ArgumentName.
**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.Exception</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.Exception</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
□  Include Protected Members
□  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions....:IsInRange Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotInRange(ConditionValidator(Of &lt;(Byte)&gt;), Byte, Byte)</td>
<td>Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotInRange(ConditionValidator(Of &lt;(DateTime)&gt;), DateTime, DateTime)</td>
<td>Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotInRange(ConditionValidator(Of &lt;(Decimal)&gt;), Decimal, Decimal)</td>
<td>Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotInRange(ConditionValidator(Of &lt;(Double)&gt;), Double, Double)</td>
<td>Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotInRange(ConditionValidator(Of &lt;(Int16)&gt;), Int16, Int16)</td>
<td>Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotInRange(ConditionValidator(Of &lt;(Int32)&gt;), Int32, Int32)</td>
<td>Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
maxValue (including those values). An exception is thrown otherwise.

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.
IsNotInRange(ConditionValidator(Of (Double), Double, Double, String))

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

IsNotInRange(ConditionValidator(Of (Int16), Int16, Int16, String))

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

IsNotInRange(ConditionValidator(Of (Int32), Int32, Int32, String))

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

IsNotInRange(ConditionValidator(Of (Int64), Int64, Int64, String))

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

IsNotInRange(Of (Nullable(Of (T))), Nullable(Of (Nullable(Of (T)))), Nullable(Of (Nullable(Of (T))), String))

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

IsNotInRange(Of (Nullable(Of (T))), Nullable(Of (Nullable(Of (T))), T, T, String))

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

IsNotInRange(ConditionValidator(Of (Single), Single, Single, String))

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.
IsNotInRange(Of T)> (ConditionValidator(Of T>, T, T, String)) is not between minValue and maxValue (including those values). An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotInRange Method (ConditionValidator<Of <(Byte)>), Byte, Byte)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte, _
    maxValue As Byte _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsNotInRange(  
    ConditionValidator<byte> validator,  
    byte minValue,  
    byte maxValue  
)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsNotInRange(  
    ConditionValidator<unsigned char>^ validator,  
    unsigned char minValue,  
    unsigned char maxValue  
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(\n
Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Byte)>)>  
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: System::Byte
The lowest invalid value.

maxValue
Type: System::Byte
The highest invalid value.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotInRange Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions..::..IsNotInRange Method (ConditionValidator<Of Byte>, Byte, Byte, String)

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( validator As ConditionValidator(Of Byte), _
    minValue As Byte, _
    maxValue As Byte, _
    conditionDescription As String _) As ConditionValidator(Of Byte)

C#

public static ConditionValidator< byte > IsNotInRange( ConditionValidator< byte > validator,
    byte minValue,
    byte maxValue,
    string conditionDescription)

Visual C++

public:
static ConditionValidator< unsigned char >^ IsNotInRange( ConditionValidator< unsigned char >^ validator,
    unsigned char minValue,
    unsigned char maxValue,
    String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator

Type: CuttingEdge.Conditions.:: ConditionValidator<Of <(Byte)>)
The ConditionValidator<Of <(T)>) that holds the value that has to be checked.
minValue
   Type: System::Byte
   The lowest invalid value.

maxValue
   Type: System::Byte
   The highest invalid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..:::.IsNotInRange Method (ConditionValidator<Of (DateTime)>, DateTime, DateTime)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange (_
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime, _
    maxValue As DateTime _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsNotInRange(
    ConditionValidator<DateTime> validator,
    DateTime minValue,
    DateTime maxValue
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsNotInRange(
    ConditionValidator<DateTime>^ validator,
    DateTime^ minValue,
    DateTime^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(DateTime)>))
The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: `System::DateTime`  
The lowest invalid value.

maxValue  
Type: `System::DateTime`  
The highest invalid value.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotInRange Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions..::.IsNotInRange Method (ConditionValidator<Of <(DateTime)>>, DateTime, DateTime, String)

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime, _
    maxValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsNotInRange(
    ConditionValidator<DateTime> validator,
    DateTime minValue,
    DateTime maxValue,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<DateTime>^ IsNotInRange(
    ConditionValidator<DateTime>^ validator,
    DateTime^ minValue,
    DateTime^ maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator

Type: CuttingEdge.Conditions.:::ConditionValidator<(Of (DateTime))>
The ConditionValidator<(Of (T)>) that holds the value that has to be checked.
minValue
   Type: System..:::DateTime
   The lowest invalid value.

maxValue
   Type: System..:::DateTime
   The highest invalid value.

conditionDescription
   Type: System..:::String
   The description of the condition that should hold. The string may hold the
   placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    maxValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsNotInRange(
    ConditionValidator<decimal> validator,
    decimal minValue,
    decimal maxValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsNotInRange(
    ConditionValidator<Decimal>^ validator,
    Decimal minValue,
    Decimal maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Decimal)>))
The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: **System::Decimal**
The lowest invalid value.

**maxValue**
Type: **System::Decimal**
The highest invalid value.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is in the specified range, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is in the specified range, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..:::IsNotInRange Method (ConditionValidator<Of
<(Decimal)>), Decimal, Decimal, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    maxValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsNotInRange(  
ConditionValidator<decimal> validator,
    decimal minValue,
    decimal maxValue,
    String conditionDescription
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsNotInRange( 
    ConditionValidator<Decimal>^ validator,
    Decimal minValue,
    Decimal maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator

Type: CuttingEdge.Conditions.:::ConditionValidator<(Of cheap (Decimal)>)
The ConditionValidator<(Of (T) )> that holds the value that has to be checked.
minValue
   Type: System::Decimal
   The lowest invalid value.

maxValue
   Type: System::Decimal
   The highest invalid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method. Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#  □  Visual C++  □  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..:::..IsNotInRange Method (ConditionValidator<Of[Double]>), Double, Double)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    maxValue As Double _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsNotInRange(
    ConditionValidator<double> validator,
    double minValue,
    double maxValue
)

Visual C++

public:
static ConditionValidator<double>^ IsNotInRange( 
    ConditionValidator<double>^ validator,
    double^ minValue,
    double^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Double)>))
    The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: **System::Double**
The lowest invalid value.

maxValue
Type: **System::Double**
The highest invalid value.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotInRange Method (ConditionValidator<Of Double>, Double, Double, String)

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    maxValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<
    double> IsNotInRange(ConditionValidator<
    double> validator,
    double minValue,
    double maxValue,
    string conditionDescription
)

Visual C++

public:
    static ConditionValidator<double>^ IsNotInRange(ConditionValidator<double>^ validator,
    double minValue,
    double maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions::ConditionValidator<(Of <(Double)>))
The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.
minValue
  Type: `System::Double`
  The lowest invalid value.

maxValue
  Type: `System::Double`
  The highest invalid value.

conditionDescription
  Type: `System::String`
  The description of the condition that should hold. The string may hold the
  placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange (_
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    maxValue As Short _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsNotInRange(
    ConditionValidator<short> validator,
    short minValue,
    short maxValue
)

Visual C++

public:
static ConditionValidator<short>^ IsNotInRange(
    ConditionValidator<short>^ validator,
    short minValue,
    short maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Int16)>))
The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: `System::::Int16`
The lowest invalid value.

`maxValue`
Type: `System::::Int16`
The highest invalid value.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    maxValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)
```

### C#

```csharp
public static ConditionValidator<short> IsNotInRange(  
    ConditionValidator<short> validator,  
    short minValue,  
    short maxValue,  
    string conditionDescription
)
```

### Visual C++

```cpp
public:  
    static ConditionValidator<short>^ IsNotInRange(  
        ConditionValidator<short>^ validator,  
        short minValue,  
        short maxValue,  
        String^ conditionDescription
    )
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(
```

## Parameters

- **validator**
  
  Type: CuttingEdge.Conditions.:::ConditionValidator(Of (Int16)>)
  
  The ConditionValidator(Of (T)>) that holds the value that has to be checked.
minValue
   Type: System::Int16
   The lowest invalid value.

maxValue
   Type: System::Int16
   The highest invalid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.Exception.ArgumentException</code></td>
<td>Thrown when the Value of the specified validator is in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is in the specified range, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotInRange Method (ConditionValidator(Of <(Int32)>), Int32, Int32)

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    maxValue As Integer _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotInRange(
    ConditionValidator<int> validator,
    int minValue,
    int maxValue
)

Visual C++

public:
static ConditionValidator<int>^ IsNotInRange(
    ConditionValidator<int>^ validator,
    int^ minValue,
    int^ maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions...::ConditionValidator<(Of <(Int32)>))
The ConditionValidator(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: **System::Int32**
The lowest invalid value.

maxValue
Type: **System::Int32**
The highest invalid value.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsNotInRange Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..:::InRange Method (ConditionValidator<Of <(Int32)>>, Int32, Int32, String)

See Also  Send Feedback

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    maxValue As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotInRange( 
    ConditionValidator<int> validator,
    int minValue,
    int maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsNotInRange( 
    ConditionValidator<int>^ validator,
    int minValue,
    int maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
Type: CuttingEdge.Conditions.:::ConditionValidator(Of Int32)>)
The ConditionValidator(Of (T)> that holds the value that has to be checked.
minValue
Type: System::Int32
The lowest invalid value.

maxValue
Type: System::Int32
The highest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
 IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Long), _
    minValue As Long, _
    maxValue As Long _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsNotInRange(
    ConditionValidator<long> validator,
    long minValue,
    long maxValue
)

Visual C++

public:
static ConditionValidator<long long>^ IsNotInRange(
    ConditionValidator<long long>^ validator,
    long long minValue,
    long long maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Int64)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: `System::Int64`
The lowest invalid value.

maxValue
Type: `System::Int64`
The highest invalid value.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Long), _
    minValue As Long, _
    maxValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)
```

**C#**

```csharp
public static ConditionValidator<long> IsNotInRange(
    ConditionValidator<long> validator,
    long minValue,
    long maxValue,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<long long>^ IsNotInRange(
    ConditionValidator<long long>^ validator,
    long long minValue,
    long long maxValue,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(
```

**Parameters**

- **validator**
  - Type: `CuttingEdge.Conditions::ConditionValidator<Of (Int64)>)`
  - The `ConditionValidator<Of (T)>)` that holds the value that has to be checked.
minValue
   Type: System::Int64
   The lowest invalid value.

maxValue
   Type: System::Int64
   The highest invalid value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsInRange Method (ConditionValidator<<Of<Single>>, Single, Single)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single, _
    maxValue As Single _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotInRange(
    ConditionValidator<float> validator,
    float minValue,
    float maxValue
)

Visual C++

public:
    static ConditionValidator<float>^ IsNotInRange(
    ConditionValidator<float>^ validator,
    float minValue,
    float maxValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Single)>))
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: System:::Single
The lowest invalid value.

maxValue
Type: System:::Single
The highest invalid value.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [IsNotInRange Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions..::..IsNotInRange Method (ConditionValidator<Of <(Single)>>, Single, Single, String)

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single, _
    maxValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotInRange(
    ConditionValidator<float> validator,
    float minValue,
    float maxValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsNotInRange(
    ConditionValidator<float>^ validator,
    float minValue,
    float maxValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotInRange = function(

Parameters

validator
Type: CuttingEdge.Conditions.:::ConditionValidator<Of <(Single)>>)
The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.
**minValue**
Type: *System::Single*
The lowest invalid value.

**maxValue**
Type: *System::Single*
The highest invalid value.

**conditionDescription**
Type: *System::String*
The description of the condition that should hold. The string may hold the placeholder '{0}' for the *ArgumentName*.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.dll::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.dll::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..:::IsNotInRange<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>), Nullable<(Of <(T)>)>), Nullable<(Of <(T)>)>))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange(Of T As {Structure, New}) ( _
  validator As ConditionValidator(Of Nullable(Of T)), _
  minValue As Nullable(Of T), _
  maxValue As Nullable(Of T) _
) As ConditionValidator(Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotInRange<T>(
  ConditionValidator<Nullable<T>> validator,
  Nullable<T> minValue,
  Nullable<T> maxValue
)

where T : struct, new()

Visual C++

public:
  generic<typename T>
  where T : value class, gcnew()
  static ConditionValidator<Nullable<T>>^ IsNotInRange(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T> minValue,
    Nullable<T> maxValue
  )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of <(T)>))>))>
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**minValue**
Type: `System.Nullable<Of <(T)>>`  
The lowest invalid value.

**maxValue**
Type: `System.Nullable<Of <(T)>>`  
The highest invalid value.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the Value of the specified validator is in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is in the specified range and a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is in the specified range and an Enum type, while the</td>
</tr>
</tbody>
</table>
specified validator is created using the Requires extension method. Thrown when the Value of the specified validator is in the specified range, while the specified validator is created using the Ensures extension method.
See Also

**ValidatorExtensions Class**
**IsNotInRange Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As Nullable(Of T), _
    maxValue As Nullable(Of T), _
    conditionDescription As String _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotInRange<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> minValue,
    Nullable<T> maxValue,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotInRange(?
        ConditionValidator<Nullable<T>>^ validator,
        Nullable<T>^ minValue,
        Nullable<T>^ maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator(Of Nullable(Of
The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minValue**
Type: `System...::Nullable<Of <(T)>>`
The lowest invalid value.

**maxValue**
Type: `System...::Nullable<Of <(T)>>`
The highest invalid value.

**conditionDescription**
Type: `System...::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentException</td>
<td>Thrown when the Value of the specified validator is in the specified range, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System::ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is in the specified range and a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System.ComponentModel::InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is in the specified range and an Enum type, while the</td>
</tr>
</tbody>
</table>
specifying validator is created using the Requires extension method.
Thrown when the Value of the specified validator is in the specified range, while the specified validator is created using the Ensures extension method.

CuttingEdge.Conditions...::: PostconditionException
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T, _
    maxValue As T _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotInRange<T>(
    ConditionValidator<Nullable<T>> validator,
    T minValue,
    T maxValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotInRange( 
    ConditionValidator<Nullable<T>>^ validator,
    T minValue,
    T maxValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)>

The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

minValue
  Type: T
  The lowest invalid value.

maxValue
  Type: T
  The highest invalid value.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range and a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::: IsNotInRange<Of< T >> Method
(ConditionValidator<Of< Nullable<Of< T >> >>), T, T, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T, _
    maxValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotInRange<T>(
    ConditionValidator<Nullable<T>> validator,
    T minValue,
    T maxValue,
    string conditionDescription
)
where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotInRange(
        ConditionValidator<Nullable<T>>^ validator,
        T minValue,
        T maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(Nullable<Of
The `ConditionValidator(Of {T})` that holds the value that has to be checked.

**minValue**
- Type: {T}
  - The lowest invalid value.

**maxValue**
- Type: {T}
  - The highest invalid value.

**conditionDescription**
- Type: `System::String`
  - The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range and a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T, _
    maxValue As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotInRange<T>(
    ConditionValidator<T> validator,
    T minValue,
    T maxValue
)
where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsNotInRange(=
        ConditionValidator<T>^ validator,
        T minValue,
        T maxValue
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator(Of <(T)>)

The ConditionValidator(Of <(T)> ) that holds the value that has to be checked.
minValue
   Type: T
   The lowest invalid value.

maxValue
   Type: T
   The highest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range and a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is in the specified range, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions:::IsNotInRange<(Of <(T)>)> Method
(ConditionValidator<(Of <(T)>)>), T, T, String)

[ValidatorExtensions Class]  See Also  Send Feedback

Checks whether the given value is not between minValue and maxValue (including those values). An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotInRange(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T, _
    maxValue As T, _
    conditionDescription As String _) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotInRange<T>(
    ConditionValidator<T> validator,
    T minValue,
    T maxValue,
    string conditionDescription
)
where T : IComparable

Visual C++

public:
    generic<typename T>
    where T : IComparable
    static ConditionValidator<T>^ IsNotInRange( 
        ConditionValidator<T>^ validator,
        T minValue,
        T maxValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions...:::ConditionValidator(Of (T)->)
The `ConditionValidator<T>` that holds the value that has to be checked.

**minValue**

Type: T

The lowest invalid value.

**maxValue**

Type: T

The highest invalid value.

**conditionDescription**

Type: `System::String`

The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
## Type Parameters

T

The type of the `Value` of the specified validator.

### Return Value

The specified validator instance.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System...::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System...::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range and a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions...::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is in the specified range, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotInRange Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions....IsNotLessOrEqual Method
ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNotLessOrEqual(ConditionValidator&lt;Of &lt;(Byte)&gt;, Byte)</code></td>
<td>Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotLessOrEqual(ConditionValidator&lt;Of &lt;(DateTime)&gt;, DateTime)</code></td>
<td>Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotLessOrEqual(ConditionValidator&lt;Of &lt;(Decimal)&gt;, Decimal)</code></td>
<td>Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotLessOrEqual(ConditionValidator&lt;Of &lt;(Double)&gt;, Double)</code></td>
<td>Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotLessOrEqual(ConditionValidator&lt;Of &lt;(Int16)&gt;, Int16)</code></td>
<td>Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotLessOrEqual(ConditionValidator&lt;Of &lt;(Int32)&gt;, Int32)</code></td>
<td>Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotLessOrEqual(ConditionValidator&lt;Of &lt;(String)&gt;, String)</code></td>
<td>Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
Isn'tLessOrEqual<(Int64)>, Int64)

IsNotLessOrEqual<(Of <(T)>)>
(ConditionValidator<(Of <(Nullable<(Of
<(T)>)>)>), Nullable<(Of <(T)>)>)

IsNotLessOrEqual<(Of <(T)>)>
(ConditionValidator<(Of <(Nullable<(Of
<(T)>)>)>), T)

IsNotLessOrEqual(ConditionValidator<(Of
<(Single)>)>, Single)

IsNotLessOrEqual<(Of <(T)>)>
(ConditionValidator<(Of <(T)>)>, T)

IsNotLessOrEqual(ConditionValidator<(Of
<(Byte)>)>, Byte, String)

IsNotLessOrEqual(ConditionValidator<(Of
<(DateTime)>)>, DateTime, String)

IsNotLessOrEqual(ConditionValidator<(Of
<(Decimal)>)>, Decimal, String)

equal to the specified
minValue. An exception is
thrown otherwise.
Checks whether the given
value is not smaller or
equal to the specified
minValue. An exception is
thrown otherwise.
Checks whether the given
value is not smaller or
equal to the specified
minValue. An exception is
thrown otherwise.
Checks whether the given
value is not smaller or
equal to the specified
minValue. An exception is
thrown otherwise.
Checks whether the given
value is not smaller or
equal to the specified
minValue. An exception is
thrown otherwise.
Checks whether the given
value is not smaller or
equal to the specified
minValue. An exception is
thrown otherwise.
Checks whether the given
value is not smaller or
equal to the specified
minValue. An exception is
thrown otherwise.
Checks whether the given
value is not smaller or
equal to the specified
minValue. An exception is
thrown otherwise.
**Isn’tLessOrEqual(ConditionValidator<(Of (Double)>, Double, String)**

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Isn’tLessOrEqual(ConditionValidator<(Of (Int16)>, Int16, String)**

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Isn’tLessOrEqual(ConditionValidator<(Of (Int32)>, Int32, String)**

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Isn’tLessOrEqual(ConditionValidator<(Of (Int64)>, Int64, String)**

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Isn’tLessOrEqual<Of <(T)>) (ConditionValidator<Of <(Nullable<Of <(T)>)>)>, Nullable<Of <(T)>)>, String)**

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Isn’tLessOrEqual<Of <(T)>) (ConditionValidator<Of <(Nullable<Of <(T)>)>)>, T, String)**

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Isn’tLessOrEqual(ConditionValidator<Of <(Single)_), Single, String)**

Checks whether the given
IsNotLessOrEqual<(Of<(T)>),(ConditionValidator<(Of<(T)>),T,String)>

value is not smaller or equal to the specified minValue. An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsNotLessOrEqual Method (ConditionValidator<Of <(Byte)>>, Byte)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsNotLessOrEqual(
    ConditionValidator<byte> validator,
    byte minValue
)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsNotLessOrEqual(
    ConditionValidator<unsigned char>^ validator,
    unsigned char minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator(Of Of (Byte)>)
The ConditionValidator(Of Of (T)>) that holds the value that has to be checked.

minValue
Type: System..::.Byte
The highest invalid value.
**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotLessOrEqual Method (ConditionValidator<Of <(Byte)>), Byte, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<Byte> IsNotLessOrEqual(
    ConditionValidator<Byte> validator,
    byte minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<unsigned char>^ IsNotLessOrEqual( 
    ConditionValidator<unsigned char>^ validator,
    unsigned char minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of (Byte)>)
    The ConditionValidator(Of (T)> that holds the value that has to be checked.

minValue
Type: `System::Byte`
The highest invalid value.

conditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsNotLessOrEqual(ConditionValidator<DateTime> validator, DateTime minValue)

Visual C++

public: ConditionValidator<DateTime>^ IsNotLessOrEqual(ConditionValidator<DateTime>^ validator, DateTime^ minValue)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = function(

Parameters

validator
  Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(DateTime)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
  Type: System..::: DateTime
  The highest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotLessOrEqual Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic ❌  C# ❌  Visual C++ ❌  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...::: IsNotLessOrEqual Method (ConditionValidator<(DateTime>), DateTime, String)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

**C#**

public static ConditionValidator<DateTime> IsNotLessOrEqual(
    ConditionValidator<DateTime> validator,
    DateTime minValue,
    string conditionDescription
)

**Visual C++**

public: static ConditionValidator<DateTime>^ IsNotLessOrEqual(
    ConditionValidator<DateTime>^ validator,
    DateTime^ minValue,
    String^ conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = function

### Parameters

**validator**

Type: `CuttingEdge.Conditions::ConditionValidator<Of <(DateTime)>>`

The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minValue**
Type: \texttt{System::DateTime}
The highest invalid value.

\textbf{conditionDescription}

Type: \texttt{System::String}
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

\textbf{Return Value}

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentsOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

`ValidatorExtensions.:::IsNotLessOrEqual Method (ConditionValidator<Of <(Decimal)>), Decimal)`

`ValidatorExtensions Class`  `See Also`  `Send Feedback`

Checks whether the given value is not smaller or equal to the specified `minValue`. An exception is thrown otherwise.

**Namespace:**  `CuttingEdge.Conditions`  
**Assembly:**  `CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)`
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual (_
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsNotLessOrEqual(
    ConditionValidator<decimal> validator,
    decimal minValue
)

Visual C++

public:
static ConditionValidator<Decimal>^ IsNotLessOrEqual(
    ConditionValidator<Decimal>^ validator,
    Decimal minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of (Decimal))>
The ConditionValidator(Of (T)> that holds the value that has to be checked.

minValue
    Type: System..::.Decimal
    The highest invalid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotLessOrEqual Method (ConditionValidator<(Of <(Decimal)>)>, Decimal, String)

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)
```

**C#**

```csharp
public static ConditionValidator<decimal> IsNotLessOrEqual(
    ConditionValidator<decimal> validator,
    decimal minValue,
    string conditionDescription
)
```

**Visual C++**

```cpp
public: static ConditionValidator<Decimal>^ IsNotLessOrEqual(
    ConditionValidator<Decimal>^ validator,
    Decimal^ minValue,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi
```

### Parameters

- **validator**
  Type: `CuttingEdge.Conditions..::.ConditionValidator<(Of <(Decimal)>)>`
  The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

- **minValue**
Type: `System::Decimal`
The highest invalid value.

**conditionDescription**

Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double _
) As ConditionValidator(Of Double)
```

### C#

```csharp
public static ConditionValidator<double> IsNotLessOrEqual(
    ConditionValidator<double> validator,
    double minValue
)
```

### Visual C++

```cpp
public:
static ConditionValidator<double>^ IsNotLessOrEqual(
    ConditionValidator<double>^ validator,
    double minValue
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = function
```

## Parameters

- **validator**
  - Type: `CuttingEdge.Conditions:::ConditionValidator<Of <(Double)>>`
  - The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

- **minValue**
  - Type: `System:::Double`
  - The highest invalid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Constraints..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotLessOrEqual Method (ConditionValidator<Of (Double)>, Double, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<
    double> IsNotLessOrEqual( 
    ConditionValidator<
        double> validator,
    double minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<
    double>^ IsNotLessOrEqual( 
    ConditionValidator<
        double>^ validator,
    double^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi

Parameters

validator
Type: CuttingEdge.Conditions.:::ConditionValidator<
    Of <(Double)>>

The ConditionValidator<
    Of <(T)>> that holds the value that has to be checked.

minValue
Type: `System::Double`
The highest invalid value.

ConditionDescription
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotLessOrEqual Overload**
**CuttingEdge.Conditions Namespace**

Send feedback on this topic to Microsoft.
ValidatorExtensions.IsNotLessOrEqual Method (ConditionValidator<Int16>, Int16)

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
  validator As ConditionValidator(Of Short), _
  minValue As Short _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsNotLessOrEqual(
  ConditionValidator<short> validator,
  short minValue
)

Visual C++

public:
static ConditionValidator<short>^ IsNotLessOrEqual(
  ConditionValidator<short>^ validator,
  short^ minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(Int16)>>
  The ConditionValidator<Of <(T)>> that holds the value that has to be checked.

minValue
  Type: System..::.Int16
  The highest invalid value.
**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsNotLessOrEqual Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

**ValidatorExtensions..::.IsNotLessOrEqual Method (ConditionValidator(Of Int16), Int16, String)**

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)

#### C#

```csharp
public static ConditionValidator<short> IsNotLessOrEqual(
    ConditionValidator<short> validator,
    short minValue,
    String conditionDescription
)
```

#### Visual C++

```csharp
public:
static ConditionValidator<short>^ IsNotLessOrEqual(
    ConditionValidator<short>^ validator,
    short^ minValue,
    String^ conditionDescription
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = func;
```

### Parameters

- **validator**
  Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Int16)>)`
  The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

- **minValue**
Type: `System::Int16`
The highest invalid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::::ArgumentOutOfRangeException</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
</tbody>
</table>

| CuttingEdge.Conditions::::PostconditionException |

<table>
<thead>
<tr>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer)
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotLessOrEqual(
    ConditionValidator<int> validator,
    int minValue
)

Visual C++

public:
static ConditionValidator<int>^ IsNotLessOrEqual(
    ConditionValidator<int>^ validator,
    int minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(Int32)>)
The ConditionValidator(Of <(T)>),

minValue
Type: System..::.Int32
The highest invalid value.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsNotLessOrEqual Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual (_
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    conditionDescription As String _) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotLessOrEqual(
    ConditionValidator<int> validator,
    int minValue,
    string conditionDescription)

Visual C++

public:
static ConditionValidator<int>^ IsNotLessOrEqual(
    ConditionValidator<int>^ validator,
    int^ minValue,
    String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = function
Type: `System:::Int32`
The highest invalid value.

**conditionDescription**
Type: `System:::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Long), _
    minValue As Long ) As ConditionValidator(Of Long)
```

### C#

```csharp
public static ConditionValidator<long> IsNotLessOrEqual(
    ConditionValidator<long> validator,
    long minValue
)
```

### Visual C++

```cpp
public: ConditionValidator<long long>^ IsNotLessOrEqual(
    ConditionValidator<long long>^ validator,
    long long minValue
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = function
```

## Parameters

**validator**
- Type: `CuttingEdge.Conditions..:::ConditionValidator(Of <(Int64)>)`
- The `ConditionValidator(Of <(T)>)` that holds the value that has to be checked.

**minValue**
- Type: `System..:::Int64`
- The highest invalid value.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Constraints..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...::: IsNotLessOrEqual Method (ConditionValidator<Of <(Int64)>, Int64, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Long), _
    minValue As Long, _
    conditionDescription As String _
) As ConditionValidator(Of Long)

C#

public static ConditionValidator<long> IsNotLessOrEqual(
    ConditionValidator<long> validator,
    long minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<long long>^ IsNotLessOrEqual(
    ConditionValidator<long long>^ validator,
    long long minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = function

Parameters

validator
Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(Int64)>))
The ConditionValidator(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: System::Int64
The highest invalid value.

conditionDescription
  Type: System::String
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.Exception.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.Exception.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

* [ValidatorExtensions Class](ValidatorExtensions)
  * [IsNotLessOrEqual Overload](IsNotLessOrEqual)
* [CuttingEdge.Conditions Namespace](CuttingEdge)

Send [feedback](feedback) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions.:::IsNotLessOrEqual Method (ConditionValidator<Of <(Single)>>, Single)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:**  **CuttingEdge.Conditions**

**Assembly:**  CuttingEdge.Condsitions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single _
) As ConditionValidator(Of Single)
```

**C#**

```csharp
public static ConditionValidator<float> IsNotLessOrEqual(ConditionValidator<float> validator, float minValue)
```

**Visual C++**

```cpp
public: ConditionValidator<float>^ IsNotLessOrEqual(ConditionValidator<float>^ validator, float minValue)
```

**JavaScript**

```
CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = function
```

### Parameters

**validator**

- **Type:** `CuttingEdge.Conditions...::ConditionValidator<Of <(Single)>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**minValue**

- **Type:** `System::::Single`
- The highest invalid value.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotLessOrEqual Method (ConditionValidator(Of Single), Single, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single, _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotLessOrEqual(
    ConditionValidator<float> validator,
    float minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>* IsNotLessOrEqual(
    ConditionValidator<float>* validator,
    float minValue,
    String* conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessOrEqual = functi

Parameters

validator
Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Single)>)> The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: **System..:: Single**
The highest invalid value.

**conditionDescription**
Type: **System..:: String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**
The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.CutConditions..::.PostconditionException</strong></td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::Isn'tLessOrEqual<(Of <(T)>)> Method
(ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)>, Nullable<(Of <(T)>)>)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
  minValue As Nullable(Of T)) _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotLessOrEqual<T>(
  ConditionValidator<Nullable<T>> validator,
  Nullable<T> minValue
)
where T : struct, new()

Visual C++

public:
  generic<typename T>
  where T : value class, gcnew()
  static ConditionValidator<Nullable<T>>^ IsNotLessOrEqual(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T>^ minValue
  )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: `CuttingEdge.Conditions::ConditionValidator<Nullable<Nullable<T>>>` where T : struct

The `ConditionValidator<Nullable<T>>` that holds the value that has to be checked.
minValue
    Type: System.Nullable<T>
    The highest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdgeCONDITIONS.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
minValue As Nullable(Of T), _) conditionDescription As String) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotLessOrEqual<T>(ConditionValidator<Nullable<T>> validator,
Nullable<T> minValue,
string conditionDescription)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotLessOrEqual(ConditionValidator<Nullable<T>>^ validator,
Nullable<T>^ minValue,
string^ conditionDescription)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions::ConditionValidator<(Of Nullable(Of (Of (T))))>

The ConditionValidator<(Of (T))> that holds the value that has to be
checked.

minValue
Type: System::Nullable(Of (T))
The highest invalid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotLessOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
minValue As T _) As ConditionValidator(Of Nullable(Of T))
```

**C#**

```csharp
public static ConditionValidator<Nullable<T>> IsNotLessOrEqual<T>(ConditionValidator<Nullable<T>> validator, T minValue)
```

where T : struct, new()

**Visual C++**

```cpp
public:
generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotLessOrEqual(ConditionValidator<Nullable<T>>^ validator, T minValue)
```

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

**validator**

Type: `CuttingEdge.Conditions..:::ConditionValidator<Of (Nullable<Of (Nullable<Of <(T)>)>)>)`

The `ConditionValidator<Of <(T)>)>` that holds the value that has to be checked.
minValue
  Type: T
  The highest invalid value.
Type Parameters

T
   The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](https://www.cuttingedge-conditions.com)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual(Of T As {Structure, New}) (validator As ConditionValidator(Of Nullable(Of T)), _
   minValue As T, _
   conditionDescription As String ) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotLessOrEqual<T>(
   ConditionValidator<Nullable<T>> validator,
   T minValue,
   string conditionDescription
)

where T : struct, new()

Visual C++

public:
   generic<typename T>
   where T : value class, gcnew()
   static ConditionValidator<Nullable<T>>^ IsNotLessOrEqual(
      ConditionValidator<Nullable<T>>^ validator,
      T minValue,
      String^ conditionDescription
   )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
   Type: CuttingEdge.Conditions.:::ConditionValidator<(Of <(Nullable<(Of <(T)>))>)>)
   The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

minValue
Type: T
The highest invalid value.

conditionDescription
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://example.com)
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual(Of T As IComparable) ( _
validator As ConditionValidator(Of T), _
minValue As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotLessOrEqual<T>(
ConditionValidator<T> validator,
T minValue
)
where T : IComparable

Visual C++

public:
generic<typename T>
where T : IComparable
static ConditionValidator<T>^ IsNotLessOrEqual(  
ConditionValidator<T>^ validator,
T minValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(T)>))
The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

minValue
Type: T
The highest invalid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel..::.InvalidEnumArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is an <strong>Enum</strong> type and is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller or equal to <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
</tbody>
</table>
while the specified validator is created using the Ensures extension method.
See Also

- ValidatorExtensions Class
- IsNotLessOrEqual Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotLessOrEqual(Of <(T)>) Method
(ConditionValidator(Of <(T)>) , T , String)

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not smaller or equal to the specified minValue. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessOrEqual(Of T As IComparable) ( _
   validator As ConditionValidator(Of T), _
   minValue As T, _
   conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotLessOrEqual<T>(
   ConditionValidator<T> validator, 
   T minValue, 
   string conditionDescription
)
where T : IComparable

Visual C++

public:
   generic<typename T>
   where T : IComparable
   static ConditionValidator<T>* IsNotLessOrEqual(
      ConditionValidator<T>* validator, 
      T minValue, 
      String* conditionDescription
   )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions::ConditionValidator<Of <(T)>>
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.
minValue
  Type: T
  The highest invalid value.

conditionDescription
  Type: System::String
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
### Type Parameters

**T**

The type of the `Value` of the specified validator.

### Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller or equal to minValue, while the specified validator is created using the Requires extension method. Thrown when the Value of the specified validator is smaller or equal to minValue,</td>
</tr>
<tr>
<td>InvalidEnumArgumentException</td>
<td>Thrown when the Value of the specified validator is an Enum type and is smaller or equal to minValue, while the specified validator is created using the Requires extension method. Thrown when the Value of the specified validator is smaller or equal to minValue,</td>
</tr>
</tbody>
</table>
while the specified validator is created using the `Ensures` extension method.
See Also

ValidatorExtensions Class
IsNotLessOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript  Include Protected Members  Include Inherited Members

CuttingEdge.Conditions reference library

ValidatorExtensions...::IsNotLessThan Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| IsNotLessThan(ConditionValidator<
(Nullable<
(Of
(Byte)>)>), Byte) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
| IsNotLessThan(ConditionValidator<
(Of
(DateTime)>)>, DateTime) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
| IsNotLessThan(ConditionValidator<
(Of
(Decimal)>)>, Decimal) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
| IsNotLessThan(ConditionValidator<
(Of
(Double)>)>, Double) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
| IsNotLessThan(ConditionValidator<
(Of
(Int16)>)>, Int16) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
| IsNotLessThan(ConditionValidator<
(Of
(Int32)>)>, Int32) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
| IsNotLessThan(ConditionValidator<
(Of
(Int64)>)>, Int64) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
| IsNotLessThan<
(Of
(ConditionValidator<
(Nullable<
(Of
(T)>)>))>, T) | Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise. |
Nullable<Of <T>>)

IsNotLessThan<Of <T>>
(ConditionValidator<Of
<(Nullable<Of <T>>)>), T)

IsNotLessThan(ConditionValidator<Of
<(Single)>, Single)

IsNotLessThan<Of <T>>
(ConditionValidator<Of <T>>, T)

IsNotLessThan(ConditionValidator<Of
<(Byte)>, Byte, String)

IsNotLessThan(ConditionValidator<Of
<(DateTime)>, DateTime, String)

IsNotLessThan(ConditionValidator<Of
<(Decimal)>, Decimal, String)

IsNotLessThan(ConditionValidator<Of
<(Double)>, Double, String)

IsNotLessThan(ConditionValidator<Of
<(Int16)>, Int16, String)

IsNotLessThan(ConditionValidator<Of
<(Int32)>, Int32, String)

exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.
Isn'tLessThan(ConditionValidator<(Of Int64), Int64, String)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Isn'tLessThan<(Of <(T)>)>
(ConditionValidator<(Of Nullable<(Of <(T)>)>), Nullable<(Of <(T)>)>, String)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Isn'tLessThan<(Of <(T)>)>
(ConditionValidator<(Of Nullable<(Of <(T)>)>), T, String)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Isn'tLessThan<(Of <(T)>)>
(ConditionValidator<(Of <(Single)>)>, Single, String)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan (_
    validator As ConditionValidator(Of Byte), _
    minValue As Byte _
) As ConditionValidator(Of Byte)

C#

public static ConditionValidator<byte> IsNotLessThan(
    ConditionValidator<byte> validator,
    byte minValue
)

Visual C++

public:
static ConditionValidator<unsigned char>* IsNotLessThan(
    ConditionValidator<unsigned char>* validator,
    unsigned char minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator(Of Byte>)
    The ConditionValidator(Of <T>) that holds the value that has to be checked.

minValue
    Type: System:::Byte
    The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotLessThan Method (ConditionValidator<Of <(Byte)>), Byte, String)

**ValidatorExtensions Class**  [See Also]  [Send Feedback]

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions]  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Byte), _
    minValue As Byte, _
    conditionDescription As String _
) As ConditionValidator(Of Byte)

### C#

public static ConditionValidator<byte> IsNotLessThan(
    ConditionValidator<byte> validator,
    byte minValue,
    string conditionDescription
)

### Visual C++

public:
static ConditionValidator<unsigned char>^ IsNotLessThan(
    ConditionValidator<unsigned char>^ validator,
    unsigned char minValue,
    String^ conditionDescription
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

### Parameters

**validator**
- Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Byte)>))
- The **ConditionValidator<(Of <(T)>)>** that holds the value that has to be checked.

**minValue**
Type: **System::Byte**
The lowest valid value.

**conditionDescription**
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.

**Return Value**
The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime _
) As ConditionValidator(Of DateTime)
```

#### C#

```csharp
public static ConditionValidator<DateTime> IsNotLessThan(ConditionValidator<DateTime> validator, DateTime minValue)
```

#### Visual C++

```cpp
public:
static ConditionValidator<DateTime>^ IsNotLessThan(ConditionValidator<DateTime>^ validator, DateTime minValue)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions:::ConditionValidator<Of <(DateTime)>>`
- The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.

**minValue**
- Type: `System:::DateTime`
- The lowest valid value.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsNotLessThan Method (ConditionValidator<
(DateTime>>), DateTime, String)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of DateTime), _
    minValue As DateTime, _
    conditionDescription As String _
) As ConditionValidator(Of DateTime)

C#

public static ConditionValidator<DateTime> IsNotLessThan(
    ConditionValidator<DateTime> validator,
    DateTime minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<DateTime>^ IsNotLessThan( 
    ConditionValidator<DateTime>^ validator, 
    DateTime^ minValue, 
    String^ conditionDescription 
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions.:::ConditionValidator<(Of <(DateTime)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: `System..::.DateTime`
   The lowest valid value.

`conditionDescription`
   Type: `System..::.String`
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsNotLessThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript
CuttingEdge.Conditions reference library
ValidatorExtensions.IsNotLessThan Method (ConditionValidator<Of <(Decimal)>), Decimal)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal _
) As ConditionValidator(Of Decimal)

C#

public static ConditionValidator<decimal> IsNotLessThan(
    ConditionValidator<decimal> validator,
    decimal minValue
)

Visual C++

public:
    static ConditionValidator<Decimal>^ IsNotLessThan(
        ConditionValidator<Decimal>^ validator,
        Decimal^ minValue
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Decimal)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
    Type: System..::: Decimal
    The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsNotLessThan Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Decimal), _
    minValue As Decimal, _
    conditionDescription As String _
) As ConditionValidator(Of Decimal)
```

**C#**

```csharp
public static ConditionValidator<decimal> IsNotLessThan(
    ConditionValidator<decimal> validator,
    decimal minValue,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<Decimal>^ IsNotLessThan(
    ConditionValidator<Decimal>^ validator,
    Decimal^ minValue,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(
```

**Parameters**

- **validator**
  - Type: `CuttingEdge.Conditions..::.ConditionValidator<Of <(Decimal)>)`
  - The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

- **minValue**
Type: `System::Decimal`
The lowest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://example.com/CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double _
) As ConditionValidator(Of Double)
```

### C#

```csharp
public static ConditionValidator<
double> IsNotLessThan(
    ConditionValidator<
double> validator,
    double minValue
)
```

### Visual C++

```cpp
public:
static ConditionValidator<double>^ IsNotLessThan(
    ConditionValidator<double>^ validator,
    double minValue
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(
```

## Parameters

**validator**

Type: `CuttingEdge.Conditions:::ConditionValidator<Of <(Double)>)`
The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**minValue**

Type: `System:::Double`
The lowest valid value.
Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.ArgOutOfRangeException</code></td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsNotLessThan Method (ConditionValidator(Of Double), Double, String)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Double), _
    minValue As Double, _
    conditionDescription As String _
) As ConditionValidator(Of Double)
```

### C#

```csharp
public static ConditionValidator<double> IsNotLessThan(ConditionValidator<double> validator, double minValue, string conditionDescription)
```

### Visual C++

```cpp
public:
static ConditionValidator<double>^ IsNotLessThan(ConditionValidator<double>^ validator, double^ minValue, String^ conditionDescription)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(
```

#### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Double)>)`
  - The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

- **minValue**
Type: `System::Double`
The lowest valid value.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::{ArgumentOutOfRangeException}</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than minValue, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::{PostconditionException}</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than minValue, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::..IsNotLessThan Method (ConditionValidator<Of <(Int16)>, Int16)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short _) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsNotLessThan(
    ConditionValidator<short> validator,
    short minValue
)

Visual C++

public:
static ConditionValidator<short>^ IsNotLessThan(
    ConditionValidator<short>^ validator,
    short minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Int16)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
    Type: System..::: Int16
    The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

**ValidatorExtensions.::: IsNotLessThan Method (ConditionValidator<Of <(Int16)>, Int16, String)**

**ValidatorExtensions Class**  See Also  Send Feedback

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Short), _
    minValue As Short, _
    conditionDescription As String _
) As ConditionValidator(Of Short)

C#

public static ConditionValidator<short> IsNotLessThan(
    ConditionValidator<short> validator,
    short minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<short>^ IsNotLessThan(
    ConditionValidator<short>^ validator,
    short^ minValue,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator

Type: CuttingEdge.Conditions...:::ConditionValidator<(Of <(Int16)>)> 
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue
Type: System::Int16
The lowest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

Return Value
The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.:::IsNotLessThan Method (ConditionValidator<Of <(Int32)>), Int32)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotLessThan(
    ConditionValidator<int> validator,
    int minValue
)

Visual C++

public:
    static ConditionValidator<int>^ IsNotLessThan(
        ConditionValidator<int>^ validator,
        int minValue
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(Int32)>>)
The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.

minValue
  Type: System..::.Int32
  The lowest valid value.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentOutOfRangeException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..:::IsNotLessThan Method (ConditionValidator<Of <(Int32)>), Int32, String)

**ValidatorExtensions Class  See Also  Send Feedback**

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:  CuttingEdge.Conditions  
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Integer), _
    minValue As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of Integer)

C#

public static ConditionValidator<int> IsNotLessThan(
    ConditionValidator<int> validator,  
    int minValue,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<int>^ IsNotLessThan(  
    ConditionValidator<int>^ validator,  
    int^ minValue,  
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(int32)>))
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

minValue


Type: `System:::Int32`
The lowest valid value.

`conditionDescription`
Type: `System:::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System···.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions···.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic □ C#
□ Visual C++
□ JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions... IsNotLessThan Method (ConditionValidator<Of <(Int64)>), Int64)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
  validator As ConditionValidator(Of Long), _
  minValue As Long _
) As ConditionValidator(Of Long)

### C#

public static ConditionValidator<long> IsNotLessThan(
  ConditionValidator<long> validator,
  long minValue
)

### Visual C++

public:
static ConditionValidator<long long>^ IsNotLessThan(
  ConditionValidator<long long>^ validator,
  long long minValue
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

### Parameters

**validator**
   Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Int64)>)> The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

**minValue**
   Type: System..:::Int64 The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotLessThan Method (ConditionValidator<Of<(Int64)>>, Int64, String)

Validates whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotLessThan ( _
   validator As ConditionValidator(Of Long), _
   minValue As Long, _
   conditionDescription As String _
) As ConditionValidator(Of Long)

**C#**

```csharp
public static ConditionValidator<long> IsNotLessThan(
   ConditionValidator<long> validator,
   long minValue,
   string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<long_long>^ IsNotLessThan(
   ConditionValidator<long_long>^ validator,
   long_long minValue,
   String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(
```

**Parameters**

- **validator**
  
  Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Int64)>)>  
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

- **minValue**
Type: `System.Int64`
The lowest valid value.

conditionDescription
Type: `System.String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsNotLessThan Method (ConditionValidator<Of (Single)>, Single)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
    validator As ConditionValidator(Of Single), _
    minValue As Single _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotLessThan(
    ConditionValidator<float> validator,
    float minValue
)

Visual C++

public:
static ConditionValidator<float>^ IsNotLessThan(
    ConditionValidator<float>^ validator,
    float minValue
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<Of <(Single)>)
    The ConditionValidator<Of <(T)> ) that holds the value that has to be checked.

minValue
    Type: System..::: Single
    The lowest valid value.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentOutOfRangeException</code></td>
<td>Thrown when the <em>Value</em> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <em>Value</em> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::..IsNotLessThan Method (ConditionValidator&lt;(Of &lt;(Single)&gt;), Single, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan ( _
   validator As ConditionValidator(Of Single), _
   minValue As Single, _
   conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotLessThan(
   ConditionValidator<float> validator,
   float minValue,
   string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsNotLessThan(  
   ConditionValidator<float>^ validator,
   float^ minValue,
   String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotLessThan = function(

Parameters

validator
   Type: CuttingEdge.Conditions..:::ConditionValidator<(Of (Single)>)
   The ConditionValidator<(Of (T)>) that holds the value that has to be checked.

minValue
Type: **System.Text.RegularExpressions.Single**
The lowest valid value.

**conditionDescription**
Type: **System.Text.RegularExpressions.String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentOutOfRangeException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is smaller than minValue, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](http://github.com/CuttingEdge/Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As Nullable(Of T) _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotLessThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> minValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotLessThan(
    ConditionValidator<Nullable<T>>^ validator,
    Nullable<T> minValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions::ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
minValue
Type: System::*:Nullable<Of <(T)>>
The lowest valid value.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.Exception.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.Exception.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As Nullable(Of T), _
    conditionDescription As String)

) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotLessThan<T>(
    ConditionValidator<Nullable<T>> validator,
    Nullable<T> minValue,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotLessThan(
        ConditionValidator<Nullable<T>>^ validator,
        Nullable<T>^ minValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of Nullable<(Of T)>))
    The ConditionValidator<(Of T)> that holds the value that has to be
checked.

minValue
Type: System::Nullable<(Of <(T)>)
The lowest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is smaller than minValue, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T _) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator Nullable<T> IsNotLessThan<T>(
    ConditionValidator Nullable<T> validator,
    T minValue
)

where T : struct, new()

Visual C++

public:
    generic<typename T>
where T : value class, gcnew()
static ConditionValidator Nullable<T>^ IsNotLessThan(
    ConditionValidator Nullable<T>^ validator,
    T minValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions...::ConditionValidator<Of Nullable<Of
<T>>>)

The ConditionValidator<Of <(T)>> that holds the value that has to be checked.
minValue
  Type: T
  The lowest valid value.
## Type Parameters

**T**

The type of the `Value` of the specified validator.

### Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRange.Exception</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::: IsNotLessThan< (Of < (T) >) > Method
(ConditionValidator< (Of < (Nullable< (Of < (T) >) >) > ), T, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan(Of T As {Structure, New})( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    minValue As T, _
    conditionDescription As String)
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotLessThan<T>(
    ConditionValidator<Nullable<T>> validator,
    T minValue,
    string conditionDescription
)
where T : struct, new()

Visual C++

public:
    generic<typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotLessThan(
        ConditionValidator<Nullable<T>>^ validator,
        T minValue,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions::<::<ConditionValidator<(Of <(Nullable<(Of <(T>)>)>)>)>
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

minValue
Type: T
The lowest valid value.

conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Type Parameters**

T

The type of the `Value` of the specified validator.

**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotLessThan(Of<(T)>) Method
(ConditionValidator(Of<(T)>)(), T)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLessThan(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotLessThan<T>(
    ConditionValidator<T> validator,
    T minValue
)

where T : IComparable

Visual C++

public:

generic<typename T>

where T : IComparable

static ConditionValidator<T>^ IsNotLessThan(
    ConditionValidator<T>^ validator,
    T minValue
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator(Of <(T)>)

The ConditionValidator(Of <(T)>), that holds the value that has to be checked.

minValue
Type: \( T \)
The lowest valid value.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentOutOfRangeException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>System.ComponentModel..::.InvalidEnumArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is an <code>Enum</code> type and smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is</td>
</tr>
</tbody>
</table>
created using the Ensures extension method.
See Also

**ValidatorExtensions Class**
**IsNotLessThan Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:Send%20feedback%20on%20this%20topic%20to%20Microsoft.) on this topic to Microsoft.
ValidatorExtensions...::: IsNotLessThan<Of <(T)>> Method
(ConditionValidator<Of <(T)>>, T, String)

Checks whether the given value is not less than the specified minValue. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

Public Shared Function IsNotLessThan(Of T As IComparable) ( _
    validator As ConditionValidator(Of T), _
    minValue As T, _
    conditionDescription As String _
) As ConditionValidator(Of T)

**C#**

public static ConditionValidator<T> IsNotLessThan<T>(
    ConditionValidator<T> validator,
    T minValue,
    string conditionDescription
)

where T : IComparable

**Visual C++**

public:

generic<typename T>

where T : IComparable

static ConditionValidator<T^> IsNotLessThan( 
    ConditionValidator<T^> validator,
    T minValue,
    String^ conditionDescription
)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

**validator**

Type: CuttingEdge.Conditions::<;::ConditionValidator(Of <(T)>)>

The ConditionValidator(Of <(T)>) that holds the value that has to be checked.
minValue
  Type: T
  The lowest valid value.

conditionDescription
  Type: System::String
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T
  The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.:::.ArgumentOutOfRangeException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System.ComponentModel.:::.InvalidEnumArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is an <code>Enum</code> type and smaller than <code>minValue</code>, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions.:::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is smaller than <code>minValue</code>, while the specified validator is</td>
</tr>
</tbody>
</table>
created using the `Ensures` extension method.
See Also

ValidatorExtensions Class
IsNotLessThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#  □  Visual C++  □  JavaScript  □  Include Protected Members  □  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::.IsNotLongerOrEqual Method

ValidatorExtensions Class  See Also  Send Feedback
# Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotLongerOrEqual&lt;Of (TCollection)&gt;)(ConditionValidator&lt;Of (TCollection)&gt;, Int32)</td>
<td>Checks whether the number of elements in the given value, is not more than and not equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td>IsNotLongerOrEqual&lt;Of (TCollection)&gt;)(ConditionValidator&lt;Of (TCollection)&gt;, Int32, String)</td>
<td>Checks whether the number of elements in the given value, is not more than and not equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the number of elements in the given value, is not more than and not equal to the specified number of elements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** [CuttingEdge.Conditions](CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLongerOrEqual(Of TCollection As IEnumerable(
    validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsNotLongerOrEqual<TCollection>(
    ConditionValidator<TCollection> validator, _
    int numberOfElements
) where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsNotLongerOrEqual(
        ConditionValidator<TCollection>^ validator, _
        int numberOfElements
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<Of
    <(TCollection)>=

The ConditionValidator<Of <(T)>=> that holds the value that has to be checked.
numberOfElements
   Type: System..::.Int32
   The collection must contain less elements than this value.
Type Parameters

TCollection
The type of the value to check.

Return Value
The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is smaller or equal to 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNot Longer OrEqual Overload**
**CuttingEdge.Conditions Namespace**

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions..:::Isn'tLongerOrEqual<Of <(TCollection)>>() Method
(ConditionValidator<Of <(TCollection)>>(), Int32, String)

Checks whether the number of elements in the given value, is not more than and not equal to the specified numberofElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotLongerOrEqual(Of TCollection As IEnumerable:
    validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

**C#**

public static ConditionValidator<TCollection> IsNotLongerOrEqual<TC:
    ConditionValidator<TCollection> validator,
    int numberOfElements,
    string conditionDescription
}
where TCollection : IEnumerable

**Visual C++**

public:
    generic<typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ IsNotLongerOrEqual(ConditionValidator<TCollection>^ validator,
    int numberOfElements,
    String^ conditionDescription
}

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions,:::ConditionValidator<(Of _
    <(TCollection)>)

The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

numberOfElements
   Type: System::Int32
   The collection must contain less elements than this value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is smaller or equal to 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsNot LongerOrEqual Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Visual Basic
C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions.IsNotLongerThan Method
ValidatorExtensions Class
See Also
Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotLongerThan&lt;Of TCollection&gt; (ConditionValidator&lt;Of TCollection&gt;, Int32)</td>
<td>Checks whether the number of elements in the given value, is not more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotLongerThan&lt;Of TCollection&gt; (ConditionValidator&lt;Of TCollection&gt;, Int32, String)</td>
<td>Checks whether the number of elements in the given value, is not more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNot LongerThan<Of <(TCollection)>>) Method (ConditionValidator<Of <(TCollection)>>, Int32)

Checks whether the number of elements in the given value, is not more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsNotLongerThan(Of TCollection As IEnumerable,
    validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer _) As ConditionValidator(Of TCollection)
```

### C#

```csharp
public static ConditionValidator<TCollection> IsNotLongerThan<TCollie,
    ConditionValidator<TCollection> validator, int numberOfElements
}
```

where TCollection : IEnumerable

### Visual C++

```cpp
public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsNotLongerThan(ConditionValidator<TCollection>^ validator, int numberOfElements
}
```

### JavaScript

JavaScript does not support generic types or methods.

### Parameters

 validator 
Type: **CuttingEdge.Conditions..:::ConditionValidator<Of <(TCollection)>>**
The **ConditionValidator<Of <(T)>**) that holds the value that has to be checked.
numberOfElements
  Type: System::Int32
  The collection must contain the same amount or less elements than this value.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the Value of the specified validator contains more elements than specified by the numberOfElements argument, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the Value of the specified validator is a null reference and the numberOfElements is smaller than 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the Value of the specified validator contains more elements than specified by the numberOfElements argument, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[IsNotLongerThan Overload]
[CuttingEdge.Conditions Namespace]

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...::..IsNotLongerThan<(Of <(TCollection)>)> Method
(ConditionValidator<(Of <(TCollection)>)>, Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the number of elements in the given value, is not more than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotLongerThan(Of TCollection As IEnumerableViewer As ConditionValidator(Of TCollection), _
    numberOfElements As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsNotLongerThan<TCollectioviewer As ConditionValidator<TCollection> validator, int numberOfElements, string conditionDescription
)

where TCollection : IEnumerable

Visual C++

public:
generic< typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ IsNotLongerThan(ConditionValidator<TCollection>^ validator, int numberOfElements, String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::,ConditionValidator(Of <(TCollection)>)
The ConditionValidator(Of <(T)> ) that holds the value that has to be
checked.

**numberOfElements**
Type: `System::Int32`
The collection must contain the same amount or less elements than this value.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more elements than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is smaller than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more elements than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- IsNotLongerThan Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
Include Protected Members  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions..:..IsNotNaN Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotNaN(ConditionValidator(Of &lt;(Double)&gt;))</td>
<td>Checks whether the given value is a not valid number. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNaN(ConditionValidator(Of &lt;(Single)&gt;))</td>
<td>Checks whether the given value is a not valid number. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNaN(ConditionValidator(Of &lt;(Double)&gt;, String))</td>
<td>Checks whether the given value is a not valid number. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNaN(ConditionValidator(Of &lt;(Single)&gt;, String))</td>
<td>Checks whether the given value is a not valid number. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is a not valid number. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNotNaN ( _
    validator As ConditionValidator(Of Double) _
) As ConditionValidator(Of Double)
```

**C#**

```csharp
public static ConditionValidator<double> IsNotNaN(
    ConditionValidator<double> validator
)
```

**Visual C++**

```cpp
public: ConditionValidator<double>^ IsNotNaN(
    ConditionValidator<double>^ validator
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isNotNaN = function(validator
```

#### Parameters

- **validator**
  - **Type:** `CuttingEdge.Conditions.:::ConditionValidator<Of <(Double)>)`
  - The `ConditionValidator<Of <(T)>>>` that holds the value that has to be checked.

#### Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a valid number, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a valid number, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotNaN Method (ConditionValidator<Of <(Double)>), String)

See Also
Send Feedback

Checks whether the given value is a not valid number. An exception is thrown otherwise.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNaN ( _
    validator As ConditionValidator(Of Double), _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<
double> IsNotNaN(
    ConditionValidator<
double> validator,
    string conditionDescription
)

Visual C++

public: ConditionValidator<
double>^ IsNotNaN(
    ConditionValidator<
double>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotNaN = function(validator,

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Double)>))
    The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.

conditionDescription
    Type: System..::: String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a valid number, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a valid number, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotNaN Method (ConditionValidator<Of <(Single)>))

Declared in the `ValidatorExtensions` class.

This method checks whether the given value is a non-numeric value. If the value is not a valid number, an exception is thrown otherwise.

**Namespace:** CuttingEdge.Conidtions

**Assembly:** CuttingEdge.Conidtions (in CuttingEdge.Conidtions.dll)
Parameters

validator

Type: `CuttingEdge.Conditions.:::ConditionValidator<Of <(Single)>)`

The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>
| System.


ArgumentException | Thrown when the Value of the specified validator is a valid number, while the specified validator is created using the Requires extension method. |
| CuttingEdge.


Conditions.


PostconditionException | Thrown when the Value of the specified validator is a valid number, while the specified validator is created using the Ensures extension method. |
See Also

ValidatorExtensions Class
IsNotNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsNotNaN Method (ConditionValidator(Of (Of Single>), String)

Checks whether the given value is a not valid number. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNaN ( _
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotNaN(
    ConditionValidator<float> validator,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsNotNaN(
    ConditionValidator<float>^ validator,
    string^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotNaN = function(vali

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(Single)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
    Type: System..::: String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SystemARGV::ArgumentException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is a valid number, while the specified validator is created using the <a href="#">Requires</a> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdgeAVV::PostconditionException</code></td>
<td>Thrown when the <a href="#">Value</a> of the specified validator is a valid number, while the specified validator is created using the <a href="#">Ensures</a> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNaN Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::IsNotNegativeInfinity Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotNegativeInfinity(ConditionValidator&lt;Of&lt;Double&gt;&gt;)</td>
<td>Checks whether the given value is not negative infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNegativeInfinity(ConditionValidator&lt;Of&lt;Single&gt;&gt;)</td>
<td>Checks whether the given value is not negative infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNegativeInfinity(ConditionValidator&lt;Of&lt;Double&gt;, String&gt;)</td>
<td>Checks whether the given value is not negative infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNegativeInfinity(ConditionValidator&lt;Of&lt;Single&gt;, String&gt;)</td>
<td>Checks whether the given value is not negative infinity. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions..::.IsNotNegativeInfinity Method (ConditionValidator<Of <(Double)>))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not negative infinity. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNegativeInfinity ( _
  validator As ConditionValidator(Of Double) _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsNotNegativeInfinity(ConditionValidator<double> validator)

Visual C++

public:
static ConditionValidator<double>^ IsNotNegativeInfinity(ConditionValidator<double>^ validator)

JavaScript

CuttingEdge.Conditions_validatorExtensions.isNotNegativeInfinity = 1

Parameters

validator
  Type: CuttingEdge.Conditions::ConditionValidator<Of (Of Double)>)
  The ConditionValidator<Of (Of T)> that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is negative infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is negative infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNegativeInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not negative infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNegativeInfinity ( _
    validator As ConditionValidator(Of Double), _
    conditionDescription As String _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsNotNegativeInfinity(
    ConditionValidator<double> validator,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<double>^ IsNotNegativeInfinity(
    ConditionValidator<double>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotNegativeInfinity = 1

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Double)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
    Type: System..::.String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is negative infinity, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is negative infinity, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNegativeInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsNotNegativeInfinity Method (ConditionValidator<Of <(Single)>>)

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)

Checks whether the given value is not negative infinity. An exception is thrown otherwise.
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNegativeInfinity ( _
    validator As ConditionValidator(Of Single) _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotNegativeInfinity(
    ConditionValidator<float> validator
)

Visual C++

public: ConditionValidator<float>^ IsNotNegativeInfinity(
    ConditionValidator<float>^ validator
)

JavaScript

CuttingEdge.Conitions.ValidatorExtensions.isNotNegativeInfinity = 1

Parameters

validator
    Type: CuttingEdge.Conitions.:::ConditionValidator<(Of <(Single)>)> The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is negative infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is negative infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNegativeInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not negative infinity. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://www.cuttingedge-conditions.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNegativeInfinity ( _
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotNegativeInfinity(
    ConditionValidator<float> validator,
    string conditionDescription
)

Visual C++

public: ConditionValidator<float>^ IsNotNegativeInfinity(
    ConditionValidator<float>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotNegativeInfinity = 1

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of (Single))
    The ConditionValidator(Of (T)) that holds the value that has to be checked.

conditionDescription
    Type: System..::.String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Return Value**

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is negative infinity, while the specified validator is created using the <code>Requires</code> extension method. Thrown when the <code>Value</code> of the specified validator is negative infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [IsNotNegativeInfinity Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions.....NotNull Method

ValidatorExtensions Class  See Also  Send Feedback
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotNull&lt;Of &lt;(T)&gt;&gt;() (ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;)&gt;&gt;)</td>
<td>Checks whether the given value is not null. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNull&lt;Of &lt;(T)&gt;&gt;() (ConditionValidator&lt;Of &lt;(T)&gt;&gt;)</td>
<td>Checks whether the given value is not null. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNull&lt;Of &lt;(T)&gt;&gt;() (ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(T)&gt;)&gt;&gt;, String)</td>
<td>Checks whether the given value is not null. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsNotNull&lt;Of &lt;(T)&gt;&gt;() (ConditionValidator&lt;Of &lt;(T)&gt;&gt;, String)</td>
<td>Checks whether the given value is not null. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](ValidatorExtensions)
[ValidatorExtensions Members](ValidatorExtensions)
[CuttingEdge.Conditions Namespace](CuttingEdge)

Send [feedback](feedback) on this topic to Microsoft.
Checks whether the given value is not null. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```visualbasic
Public Shared Function IsNotNull(Of T As {Structure, New}) ( _
validator As ConditionValidator(Of Nullable(Of T)) _) As ConditionValidator(Of Nullable(Of T))
```

**C#**

```csharp
public static ConditionValidator<Nullable<T>> IsNotNull<T>(
    ConditionValidator<Nullable<T>> validator
) where T : struct, new()
```

**Visual C++**

```cpp
public:
generic<typename T>
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNotNull(
    ConditionValidator<Nullable<T>>^ validator
)
```

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: `CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of <(T)>)>))>

The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System:::ArgumentNullException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is null, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions:::PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is null, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNull Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not null. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNull(Of T As {Structure, New}) ( _
    validator As ConditionValidator(Of Nullable(Of T)), _
    conditionDescription As String)
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNotNull<T>(
    ConditionValidator<Nullable<T>> validator,
    string conditionDescription
)

where T : struct, new()

Visual C++

public:
    generic< typename T>
    where T : value class, gcnew()
    static ConditionValidator<Nullable<T>>^ IsNotNull(
        ConditionValidator<Nullable<T>>^ validator,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Nullable<(Of
<(T)>))>)>

The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

ReturnValue

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotNull Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](#) on this topic to Microsoft.
Checks whether the given value is not null. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotNull(Of T As Class) ( _
    validator As ConditionValidator(Of T) _
) As ConditionValidator(Of T)

**C#**

public static ConditionValidator<T> IsNotNull<T>(
    ConditionValidator<T> validator
)
where T : class

**Visual C++**

public:
    generic<typename T>
    where T : ref class
static ConditionValidator<T>^ IsNotNull(
    ConditionValidator<T>^ validator
)

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions:::ConditionValidator<Of <(T)>>
The ConditionValidator<Of <(T)>> that holds the value that has to be checked.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the <code>Value</code> of the specified validator is null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <code>Value</code> of the specified validator is null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- ValidatorExtensions Class
- IsNotNull Overload
- CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not null. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNull(Of T As Class) ( _
    validator As ConditionValidator(Of T), _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotNull<T>(
    ConditionValidator<T> validator,
    string conditionDescription
)

where T : class

Visual C++

public:
    generic<typename T>
    where T : ref class
    static ConditionValidator<T>^ IsNotNull(
        ConditionValidator<T>^ validator,
        String^ conditionDescription
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions::::ConditionValidator<(Of <(T)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.
Type Parameters

*T*

The type of the **Value** of the specified validator.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNull Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNotNullOrEmpty(ConditionValidator&lt;Of &lt;(String)&gt;)&gt;)</code></td>
<td>Checks whether the given value is not null and not an Empty()() string. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotNullOrEmpty(ConditionValidator&lt;Of &lt;(String)&gt;&gt;, String)</code></td>
<td>Checks whether the given value is not null and not an Empty()() string. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions.IsNotNullOrEmpty Method (ConditionValidator<Of <(String)>>)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not null and not an Empty()() string. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNullOrEmpty ( _
    validator As ConditionValidator(Of String) _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsNotNullOrEmpty(ConditionValidator<string> validator)

Visual C++

public:
static ConditionValidator<String^>^ IsNotNullOrEmpty(ConditionValidator<String^>^ validator)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotNullOrEmpty = function

Parameters

validator
  Type: CuttingEdge.Conditions.:::ConditionValidator(Of (Of String))
  The ConditionValidator(Of (Of T)) that holds the value that has to be checked.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is Empty(), while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null or empty, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null or empty, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNullOrEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..:::IsNotNullOrEmpty Method (ConditionValidator<(Of <(String)>)), String)

**ValidatorExtensions Class**  **See Also**  **Send Feedback**

Checks whether the given value is not null and not an Empty()() string. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

**Visual Basic (Declaration)**

Public Shared Function IsNotNullOrEmpty ( _
    validator As ConditionValidator(Of String), _
    conditionDescription As String _
) As ConditionValidator(Of String)

**C#**

public static ConditionValidator<string> IsNotNullOrEmpty(ConditionValidator<string> validator, string conditionDescription)

**Visual C++**

public: ConditionValidator<String>^ IsNotNullOrEmpty(ConditionValidator<String>^ validator, String^ conditionDescription)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNotNullOrEmpty = function

**Parameters**

validator
Type: CuttingEdge.Conditions..::: ConditionValidator<(Of <(String)>)> The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
Type: System..::: String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is Empty(), while the</td>
</tr>
<tr>
<td></td>
<td>specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null or empty, while</td>
</tr>
<tr>
<td></td>
<td>the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is null or empty, while</td>
</tr>
<tr>
<td></td>
<td>the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNullOrEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
Include Protected Members  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions::IsNotNullOrWhiteSpace Method
ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotNullOrWhiteSpace(ConditionValidator&lt;Of (String&gt;))</td>
<td>Checks whether the given value is not <strong>null</strong> (Nothing in Visual Basic), not empty, and does not consists only of white-space characters.</td>
</tr>
<tr>
<td>IsNotNullOrWhiteSpace(ConditionValidator&lt;Of (String&gt;), String)</td>
<td>Checks whether the given value is not <strong>null</strong> (Nothing in Visual Basic), not empty, and does not consists only of white-space characters.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [ValidatorExtensions Members](#)
- [CuttingEdge.Conditions Namespace](#)

Send feedback on this topic to Microsoft.
Checks whether the given value is not **null** (Nothing in Visual Basic), not empty, and does not consist only of white-space characters.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNullOrWhiteSpace (validator As ConditionValidator(Of String)) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsNotNullOrWhiteSpace(ConditionValidator<string> validator)

Visual C++

public:
static ConditionValidator<String^>^ IsNotNullOrWhiteSpace(ConditionValidator<String^>^ validator)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotNullOrWhiteSpace = 1

Parameters

validator
   Type: CuttingEdge.Conditions.:::ConditionValidator(Of (String)>)
The ConditionValidator(Of (T)> that holds the value that has to be checked.

Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is empty or consists only of white-space characters, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the length of the <code>Value</code> of the specified validator is <code>null</code>, empty or consists only of white-space characters, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNullOrWhiteSpace Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
 CUTTINGEDGE.CONDITIONS reference library

ValidatorExtensions...:::IsNotNullOrWhiteSpace Method
(ConditionValidator<Of <(String)>), String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is not **null** (Nothing in Visual Basic), not empty, and does not consists only of white-space characters.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotNullOrWhiteSpace ( _
    validator As ConditionValidator(Of String), _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsNotNullOrWhiteSpace(ConditionValidator<string> validator,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<String^>^ IsNotNullOrWhiteSpace(ConditionValidator<String^>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotNullOrWhiteSpace = 1

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator(Of String)<(Of String>)>
The ConditionValidator(Of T>) that holds the value that has to be checked.

conditionDescription
    Type: System..::.String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is empty or consists only of white-space characters, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the length of the <code>Value</code> of the specified validator is null, empty or consists only of white-space characters, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotNullOrWhiteSpace Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
Include Protected Members  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsNotOfType Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotOfType&lt;Of &lt;(T)&lt;&gt;)&gt; (ConditionValidator&lt;Of &lt;(T)&lt;&gt;), Type)</td>
<td>Checks whether the Type of the given value is not of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the <strong>IsNotNull</strong> method to check for null references).</td>
</tr>
<tr>
<td>IsNotOfType&lt;Of &lt;(T)&lt;&gt;)&gt; (ConditionValidator&lt;Of &lt;(T)&lt;&gt;), Type, String)</td>
<td>Checks whether the Type of the given value is not of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the <strong>IsNotNull</strong> method to check for null references).</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[ValidatorExtensions Members](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions..:::IsNotOfType(Of <(T)>) Method
(ConditionValidator(Of <(T)>), Type)

Checks whether the Type of the given value is not of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the IsNotnull method to check for null references).

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotOfType(Of T As Class) ( _
    validator As ConditionValidator(Of T), _
    type As Type _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotOfType<T>(
    ConditionValidator<T> validator,
    Type type
)
where T : class

Visual C++

public:
    generic<typename T>
    where T : ref class
    static ConditionValidator<T>^ IsNotOfType(
        ConditionValidator<T>^ validator,
        Type^ type
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(T)>)>
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

type
Type: System:::Type
The Type that will be used to perform the check.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator is of the specified type, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is of the specified type, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotOfType Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the `Type` of the given value is not of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the `IsNotNull` method to check for null references).

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotOfType(Of T As Class) ( _
    validator As ConditionValidator(Of T), _
    type As Type, _
    conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNotOfType<T>(
    ConditionValidator<T> validator,
    Type type,
    string conditionDescription
)

where T : class

Visual C++

public:
    generic<typename T>
where T : ref class
static ConditionValidator<T>^ IsNotOfType(
    ConditionValidator<T>^ validator,
    Type^ type,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <T>)>
    The ConditionValidator<(Of <T>)> that holds the value that has to be checked.
type
  Type: **System..:::Type**
  The *Type* that will be used to perform the check.

conditionDescription
  Type: **System..:::String**
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Type Parameters**

T

The type of the Value of the specified validator.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SystemARGV.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is of the specified type, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.ConditionsARGV.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is of the specified type, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotOfType Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  C#
Visual C++  JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::IsNotPositiveInfinity Method
ValidatorExtensions Class  See Also  Send Feedback
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNotPositiveInfinity(ConditionValidator&lt;Of&lt;Double&gt;&gt;)</code></td>
<td>Checks whether the given value is not positive infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotPositiveInfinity(ConditionValidator&lt;Of&lt;Single&gt;&gt;)</code></td>
<td>Checks whether the given value is not positive infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotPositiveInfinity(ConditionValidator&lt;Of&lt;Double&gt;, String&gt;)</code></td>
<td>Checks whether the given value is not positive infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNotPositiveInfinity(ConditionValidator&lt;Of&lt;Single&gt;, String&gt;)</code></td>
<td>Checks whether the given value is not positive infinity. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](ValidatorExtensions-Class)
[ValidatorExtensions Members](ValidatorExtensions-Members)
[CuttingEdge.Conditions Namespace](CuttingEdge.Conditions-Namespace)

Send [feedback](feedback) on this topic to Microsoft.
Checks whether the given value is not positive infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsNotPositiveInfinity ( _
    validator As ConditionValidator(Of Double) _
) As ConditionValidator(Of Double)

### C#

public static ConditionValidator<double> IsNotPositiveInfinity(ConditionValidator<double> validator)

### Visual C++

public:
static ConditionValidator<double>^ IsNotPositiveInfinity(ConditionValidator<double>^ validator)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotPositiveInfinity = 1

## Parameters

validator

Type: CuttingEdge.Conditions.:::ConditionValidator<(Of <(Double)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

## Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is positive infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is positive infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- `ValidatorExtensions Class`
- `IsNotPositiveInfinity Overload`
- `CuttingEdge.Conditions Namespace`

Send feedback on this topic to Microsoft.
Checks whether the given value is not positive infinity. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://www.cuttingedge-conditions.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotPositiveInfinity ( _
    validator As ConditionValidator(Of Double), _
    conditionDescription As String _) As ConditionValidator(Of Double)

**C#**

public static ConditionValidator<double> IsNotPositiveInfinity(
    ConditionValidator<double> validator,
    string conditionDescription
)

**Visual C++**

public: ConditionValidator<double>^ IsNotPositiveInfinity(
    ConditionValidator<double>^ validator,
    String^ conditionDescription
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isNotPositiveInfinity = 1

**Parameters**

**validator**
Type: CuttingEdge.Conditions..::.ConditionValidator<Of <(Double)>>
The ConditionValidator<Of <(T)>>() that holds the value that has to be checked.

**conditionDescription**
Type: System..::.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is positive infinity, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is positive infinity, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotPositiveInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is not positive infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotPositiveInfinity ( _
   validator As ConditionValidator(Of Single) _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotPositiveInfinity(
   ConditionValidator<float> validator
)

Visual C++

public:
static ConditionValidator<float>^ IsNotPositiveInfinity(
   ConditionValidator<float>^ validator
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotPositiveInfinity = 1

Parameters

validator
   Type: CuttingEdge.Conditions.:::ConditionValidator<(Of <(Single)>)>)
   The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is positive infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is positive infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotPositiveInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsNotPositiveInfinity Method (ConditionValidator<Of <(Single)>, String)

Checks whether the given value is not positive infinity. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotPositiveInfinity ( _
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsNotPositiveInfinity(ConditionValidator<float> validator,
    string conditionDescription)

Visual C++

public:
    static ConditionValidator<float>^ IsNotPositiveInfinity(ConditionValidator<float>^ validator,
        String^ conditionDescription)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNotPositiveInfinity = 1

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator(Of (Of (Single)>))
The ConditionValidator(Of (Of (T)>)) that holds the value that has to be checked.

conditionDescription
    Type: System:::String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is positive infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is positive infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotPositiveInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...::IsNotShorterOrEqual Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotShorterOrEqual&lt;(Of &lt;(TCollection)&gt;)(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), Int32))</td>
<td>Checks whether the number of elements in the given value, is not less than and not equals to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td>IsNotShorterOrEqual&lt;(Of &lt;(TCollection)&gt;)(ConditionValidator&lt;(Of &lt;(TCollection)&gt;), Int32, String))</td>
<td>Checks whether the number of elements in the given value, is not less than and not equals to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions..::..IsNotShorterOrEqual<(Of <(TCollection)>)> Method
(ConditionValidator<(Of <(TCollection)>)>, Int32)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the number of elements in the given value, is not less than and
not equals to the specified numberOfElements argument. An exception is thrown
otherwise. When the value is a null reference, it is considered to have 0
elements.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotShorterOrEqual(Of TCollection As IEnumerable validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsNotShorterOrEqual<TC (ConditionValidator<TCollection> validator,_
    int numberOfElements
)
where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsNotShorterOrEqual(ConditionValidator<TCollection>^ validator,
        int numberOfElements
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<Of
<(TCollection)>)

The ConditionValidator<Of <(T)>>) that holds the value that has to be checked.
numberOfElements

Type: `System::Int32`

The collection must contain more elements than this value.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains less or the same amount of elements as specified by the <strong>numberOfElements</strong> argument, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and the <strong>numberOfElements</strong> is greater or equal to 0, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains less or the same amount of elements as specified by the <strong>numberOfElements</strong> argument, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotShorterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the number of elements in the given value, is not less than and not equals to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** [CuttingEdge.Conditions](https://example.com/CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotShorterOrEqual(Of TCollection As IEnumerable validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer, _
    conditionDescription As String _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsNotShorterOrEqual<TCollection>
    (ConditionValidator<TCollection> validator, _
    int numberOfElements, _
    string conditionDescription)

where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ IsNotShorterOrEqual(ConditionValidator<TCollection>^ validator, _
    int numberOfElements, _
    String^ conditionDescription)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of _
    (TCollection)>)
    The ConditionValidator<(Of <(T)>>) that holds the value that has to be
checked.

numberOfElements  Type: System::Int32
The collection must contain more elements than this value.

conditionDescription  Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator contains less or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is greater or equal to 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator contains less or the same amount of elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNotShorterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript  Include Protected Members  Include Inherited Members

CuttingEdge.Conditions reference library

ValidatorExtensions...: IsNotShorterThan Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNotShorterThan&lt;(Of &lt;(TCollection)&gt;)&gt;</td>
<td>Checks whether the number of elements in the given value, is not less than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements. Checks whether the number of elements in the given value, is not less than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td>(ConditionValidator&lt;(Of &lt;(TCollection)&gt;&gt;), Int32)</td>
<td></td>
</tr>
<tr>
<td>IsNotShorterThan&lt;(Of &lt;(TCollection)&gt;)&gt;</td>
<td></td>
</tr>
<tr>
<td>(ConditionValidator&lt;(Of &lt;(TCollection)&gt;&gt;), Int32, String)</td>
<td></td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [ValidatorExtensions Members](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Checks whether the number of elements in the given value, is not less than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsNotShorterThan(Of TCollection As IEnumerable) validator As ConditionValidator(Of TCollection), _
   numberOfElements As Integer _
) As ConditionValidator(Of TCollection)

**C#**

public static ConditionValidator<TCollection> IsNotShorterThan<TCol](
   ConditionValidator<TCollection> validator,
   int numberOfElements
)

where TCollection : IEnumerable

**Visual C++**

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsNotShorterThan(
        ConditionValidator<TCollection>^ validator,
        int numberOfElements
    )

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

validator

Type: CuttingEdge.Conditions..::.ConditionValidator(Of
    <(TCollection)>)

The ConditionValidator(Of <(T)>)) that holds the value that has to be checked.
numberOfElements
   Type: System..::..Int32
   The collection must contain the same amount or more elements than this value.
**Type Parameters**

TCollection
   The type of the value to check.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains less elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and the <code>numberOfElements</code> is greater or equal to 0, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains less elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[IsNotShorterThan Overload](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:.IsNotShorterThan<(Of <(TCollection)>)> Method
(ConditionValidator<(Of <(TCollection)>)>), Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the number of elements in the given value, is not less than the specified numberofElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNotShorterThan(Of TCollection As IEnumerable) validator As ConditionValidator(Of TCollection), _
numberOfElements As Integer, _
conditionDescription As String _) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsNotShorterThan<TCollection>(
    ConditionValidator<TCollection> validator, 
    int numberOfElements, 
    string conditionDescription 
)

where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsNotShorterThan(
        ConditionValidator<TCollection>^ validator, 
        int numberOfElements, 
        String^ conditionDescription 
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::.ConditionValidator<(Of 
<(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**numberOfElements**
- **Type:** `System::Int32`
- The collection must contain the same amount or more elements than this value.

**conditionDescription**
- **Type:** `System::String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains less elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>System..::.ArgumentNullException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and the <code>numberOfElements</code> is greater or equal to 0, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator contains less elements as specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsNotShorterThan Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](#) on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::isNull Method
ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNull&lt;(Of &lt;T&gt;)&gt;</code> (ConditionValidator&lt;(Of Nullable&lt;(Of &lt;T&gt;)&gt;)&gt;))</td>
<td>Checks whether the given value is null. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNull&lt;(Of &lt;T&gt;)&gt;</code> (ConditionValidator&lt;(Of &lt;T&gt;)&gt;))</td>
<td>Checks whether the given value is null. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNull&lt;(Of &lt;T&gt;)&gt;</code> (ConditionValidator&lt;(Of Nullable&lt;(Of &lt;T&gt;)&gt;), String)&gt;</td>
<td>Checks whether the given value is null. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>IsNull&lt;(Of &lt;T&gt;)&gt;</code> (ConditionValidator&lt;(Of &lt;T&gt;), String)&gt;</td>
<td>Checks whether the given value is null. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is null. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](https://www.example.com)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNull(Of T As {Structure, New}) ( _
  validator As ConditionValidator(Of Nullable(Of T)) _
) As ConditionValidator(Of Nullable(Of T))

C#

public static ConditionValidator<Nullable<T>> IsNull<T>(
  ConditionValidator<Nullable<T>> validator
)
where T : struct, new()

Visual C++

public:
  generic<typename T>
  where T : value class, gcnew()
  static ConditionValidator<Nullable<T>>^ IsNull(^
    ConditionValidator<Nullable<T>>^ validator
  )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
  Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of <(T)>)>)>)>
  The ConditionValidator<(Of (T)>) that holds the value that has to be checked.
Type Parameters

T
The type of the Value of the specified validator.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- **ValidatorExtensions Class**
- **IsNull Overload**
- **CuttingEdge.Conditions Namespace**

Send [feedback](#) on this topic to Microsoft.
Checks whether the given value is null. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://docs.microsoft.com/en-us/dotnet/api/CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNull(Of T As {Structure, New}) ( _
  validator As ConditionValidator(Of Nullable(Of T)), _
  conditionDescription As String _
) As ConditionValidator(Of Nullable(Of T))

C#

class IsNull
{
public static ConditionValidator<Nullable<T>> IsNull<T>(
  ConditionValidator<Nullable<T>> validator,
  string conditionDescription
}
where T : struct, new()

Visual C++

public:

generic<
type> T
where T : value class, gcnew()
static ConditionValidator<Nullable<T>>^ IsNull(^
  ConditionValidator<Nullable<T>>^ validator,
  String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(Nullable<(Of <(T)>))))>

The ConditionValidator<(Of <(T)>)) that holds the value that has to be checked.
conditionDescription
Type: System::String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNull Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is null. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsNull(Of T As Class) ( _
    validator As ConditionValidator(Of T) _
) As ConditionValidator(Of T)
```

**C#**

```csharp
public static ConditionValidator<T> IsNull<T>(
    ConditionValidator<T> validator
)
```

where `T : class`

**Visual C++**

```cpp
public:
    generic<typename T>
    where T : ref class
    static ConditionValidator<T>^ IsNull(
        ConditionValidator<T>^ validator
    )
```

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

- **validator**
  - Type: `CuttingEdge.Conditions::ConditionValidator(Of <T>)`
  - The `ConditionValidator(Of <T>)` that holds the value that has to be checked.
Type Parameters

T
  The type of the `Value` of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td></td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNull Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Nullable type check

**Verb:**

Checks whether the given value is null. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNull(Of T As Class) ( _
   validator As ConditionValidator(Of T), _
   conditionDescription As String _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsNull<T>(
   ConditionValidator<T> validator,
   string conditionDescription
)
where T : class

Visual C++

public:
   generic<typename T>
   where T : ref class
   static ConditionValidator<T>^ IsNull(
      ConditionValidator<T>^ validator,
      String^ conditionDescription
   )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator
   Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(T)>)>
   The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
Type: **System::String**
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not null, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not null, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNull Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □ C#
□ Visual C++
□ JavaScript
□ Include Protected Members
□ Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:::.IsNullOrEmpty Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>IsNullOrEmpty(ConditionValidator&lt;Of (Of String&gt;&gt;))</code></td>
<td>Checks whether the given value is null or an Empty()() string. An exception is thrown otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>IsNullOrEmpty(ConditionValidator&lt;Of (Of String&gt;&gt;), String)</code></td>
<td>Checks whether the given value is null or an Empty()() string. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class](#)
[ValidatorExtensions Members](#)
[CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions..::..IsNullOrEmpty Method (ConditionValidator<Of <(String)>>)

Checks whether the given value is null or an Empty()() string. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNullOrEmpty ( _
        validator As ConditionValidator(Of String) _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsNullOrEmpty(ConditionValidator<string> validator)

Visual C++

public:
    static ConditionValidator<String^>^ IsNullOrEmpty(ConditionValidator<String^>^ validator)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNullOrEmpty = function(

Parameters

validator
    Type: CuttingEdge.Conditions.:::ConditionValidator<(Of String>)
    The ConditionValidator<(Of (T)>) that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the Value of the specified validator is not null or empty, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the Value of the specified validator is not null or empty, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNullOrEmpty Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is null or an Empty()() string. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNullOrEmpty ( _
    validator As ConditionValidator(Of String), _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsNullOrEmpty(  
    ConditionValidator<string> validator,  
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<String^>^ IsNullOrEmpty(  
    ConditionValidator<String^>^ validator,  
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNullOrEmpty = function(

Parameters

validator
  Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(String)>)>  
  The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
  Type: System..::.String  
  The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null or empty, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not null or empty, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- **ValidatorExtensions Class**
- **IsNullOrEmpty Overload**
- **CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions:::IsNullOrWhiteSpace Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNullOrWhiteSpace(ConditionValidator(Of &lt;(String&gt;))</code>)</td>
<td>Checks whether the given value is <strong>null</strong> (Nothing in Visual Basic), empty, or consists only of white-space characters.</td>
</tr>
<tr>
<td><code>IsNullOrWhiteSpace(ConditionValidator(Of &lt;(String&gt;), String))</code></td>
<td>Checks whether the given value is <strong>null</strong> (Nothing in Visual Basic), empty, or consists only of white-space characters.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is **null** (Nothing in Visual Basic), empty, or consists only of white-space characters.

**Namespace:** [CuttingEdge.Conditions](https://www.cuttingedge-development.com/conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNullOrWhiteSpace ( _
    validator As ConditionValidator(Of String) _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsNullOrWhiteSpace(ConditionValidator<string> validator)

Visual C++

public:
static ConditionValidator<String^>^ IsNullOrWhiteSpace(ConditionValidator<String^>^ validator)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNullOrWhiteSpace = func

Parameters

validator
Type: CuttingEdge.Conditions::$_::ConditionValidator<Of <(String)>>

The ConditionValidator<Of <(T)> > that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not <code>null</code>, not empty and does not consist of white-space characters, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not <code>null</code>, not empty and does not consist of white-space characters, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNullOrWhiteSpace Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#
□  Visual C++
□  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsNullOrWhiteSpace Method (ConditionValidator<Of <(String)>), String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is **null** (Nothing in Visual Basic), empty, or consists only of white-space characters.

**Namespace:**  CuttingEdge.Conditions
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsNullOrWhiteSpace ( _
    validator As ConditionValidator(Of String), _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsNullOrWhiteSpace( 
    ConditionValidator<string> validator, 
    string conditionDescription 
)

Visual C++

public: ConditionValidator<String^>^ IsNullOrWhiteSpace( 
    ConditionValidator<String^>^ validator, 
    String^ conditionDescription 
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isNullOrWhiteSpace = func

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(String)>)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
    Type: System..::.String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not <code>null</code>, not empty and does not consist only of white-space characters, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not <code>null</code>, not empty and does not consist only of white-space characters, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsNullOrWhiteSpace Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsOfTyple&lt;(Of &lt;(T)&gt;)&gt; (ConditionValidator&lt;(Of &lt;(T)&gt;)&gt;, Type)</td>
<td>Checks whether the Type of the given value is of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the IsNotNull method to check for null references).</td>
</tr>
<tr>
<td>IsOfTyple&lt;(Of &lt;(T)&gt;)&gt; (ConditionValidator&lt;(Of &lt;(T)&gt;)&gt;, Type, String)</td>
<td>Checks whether the Type of the given value is of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the IsNotNull method to check for null references).</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the `Type` of the given value is of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the `IsNotNull` method to check for null references.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsOfType(Of T As Class) ( _
    validator As ConditionValidator(Of T), _
    type As Type _
) As ConditionValidator(Of T)

C#

public static ConditionValidator<T> IsOfT<T>(
    ConditionValidator<T> validator,
    Type type
)
where T : class

Visual C++

public:
    generic<typename T>
    where T : ref class
    static ConditionValidator<T>^ IsOfT(
        ConditionValidator<T>^ validator,
        Type^ type
    )

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:..ConditionValidator<(Of <(T)>)

The ConditionValidator<(Of <(T)>)_ that holds the value that has to be checked.

type
Type: **System:::Type**

The **Type** that will be used to perform the check.
Type Parameters

T

The type of the Value of the specified validator.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not of the specified type, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not of the specified type, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsOfType Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checking whether the type of the given value is of type. An exception is thrown otherwise. When the given value is a null reference, the check will always pass, regardless of the specified type. Please use the `IsNotNull` method to check for null references.

**Namespace:** `CuttingEdge.Conditions`  
**Assembly:** `CuttingEdge.Conditions` (in `CuttingEdge.Conditions.dll`)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function IsOfType(Of T As Class)( _
    validator As ConditionValidator(Of T), _
    type As Type, _
    conditionDescription As String _
) As ConditionValidator(Of T)

**C#**

```csharp
public static ConditionValidator<T> IsOfType<T>(
    ConditionValidator<T> validator,
    Type type,
    string conditionDescription
)
```

where T : class

**Visual C++**

```cpp
public:
    template<typename T>
    static ConditionValidator<T>^ IsOfType(
        ConditionValidator<T>^ validator,
        Type^ type,
        String^ conditionDescription
    )
```

**JavaScript**

JavaScript does not support generic types or methods.

### Parameters

validator

Type: `CuttingEdge.Conditions::ConditionValidator<Of <(T)>(T)>`

The `ConditionValidator<Of <(T)>(T)>` that holds the value that has to be checked.
type
   Type: System::Type
   The Type that will be used to perform the check.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Type Parameters

T
The type of the Value of the specified validator.

Return Value
The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not of the specified type, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is not of the specified type, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsOfTypes Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsPositiveInfinity Method

ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsPositiveInfinity(ConditionValidator(Of &lt;(Double)&gt;))</td>
<td>Checks whether the given value is positive infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsPositiveInfinity(ConditionValidator(Of &lt;(Single)&gt;))</td>
<td>Checks whether the given value is positive infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsPositiveInfinity(ConditionValidator(Of &lt;(Double)&gt;, String))</td>
<td>Checks whether the given value is positive infinity. An exception is thrown otherwise.</td>
</tr>
<tr>
<td>IsPositiveInfinity(ConditionValidator(Of &lt;(Single)&gt;, String))</td>
<td>Checks whether the given value is positive infinity. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is positive infinity. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](https://www.example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsPositiveInfinity ( _
    validator As ConditionValidator(Of Double) _
) As ConditionValidator(Of Double)

C#

public static ConditionValidator<double> IsPositiveInfinity(ConditionValidator<double> validator)

Visual C++

public:
static ConditionValidator<double>* IsPositiveInfinity(ConditionValidator<double>* validator)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isPositiveInfinity = func

Parameters

validator
    Type: CuttingEdge.Conditions::ConditionValidator(Of Double)
    The ConditionValidator(Of T) that holds the value that has to be checked.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not positive infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not positive infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsPositiveInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value is positive infinity. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function IsPositiveInfinity ( _
    validator As ConditionValidator(Of Double), _
    conditionDescription As String _
) As ConditionValidator(Of Double)
```

#### C#

```csharp
public static ConditionValidator<double> IsPositiveInfinity(
    ConditionValidator<double> validator,
    string conditionDescription
)
```

#### Visual C++

```cpp
public:
static ConditionValidator<double>^ IsPositiveInfinity( 
    ConditionValidator<double>^ validator,
    String^ conditionDescription
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isPositiveInfinity = func
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions..::.ConditionValidator<(Of <(Double)>)>`
- The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**conditionDescription**
- Type: `System..::.String`
- The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`. 

Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator is not positive infinity, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator is not positive infinity, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsPositiveInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsPositiveInfinity Method (ConditionValidator<Of <(Single)>))

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is positive infinity. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsPositiveInfinity ( _
    validator As ConditionValidator(Of Single) _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsPositiveInfinity(ConditionValidator<float> validator)

Visual C++

public:
static ConditionValidator<float>^ IsPositiveInfinity(ConditionValidator<float>^ validator)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isPositiveInfinity = func

Parameters

validator
    Type: CuttingEdge.Conditions..::.ConditionValidator<(Of <(Single)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System..::.ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not positive infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions..::.PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not positive infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

- **ValidatorExtensions Class**
- **IsPositiveInfinity Overload**
- **CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Checks whether the given value is positive infinity. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsPositiveInfinity ( _
    validator As ConditionValidator(Of Single), _
    conditionDescription As String _
) As ConditionValidator(Of Single)

C#

public static ConditionValidator<float> IsPositiveInfinity(
    ConditionValidator<float> validator,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<float>^ IsPositiveInfinity(
    ConditionValidator<float>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isPositiveInfinity = func

Parameters

validator
    Type: CuttingEdge.Conditions::ConditionValidator<Of <(Single)>>
    The ConditionValidator<Of <(T)>>() that holds the value that has to be checked.

conditionDescription
    Type: System::String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not positive infinity, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is not positive infinity, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsPositiveInfinity Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#
Visual C++
JavaScript
Include Protected Members
Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...:..IsShorterOrEqual Method
ValidatorExtensions Class  See Also  Send Feedback
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsShorterOrEqual(ConditionValidator&lt;Of &lt;(String)&gt;&gt;, Int32)</code></td>
<td>Checks whether the given value is shorter or equal in length than maxLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><code>IsShorterOrEqual&lt;Of &lt;(TCollection)&gt;&gt;(ConditionValidator&lt;Of &lt;(TCollection)&gt;&gt;, Int32)</code></td>
<td>Checks whether the number of elements in the given value, is less than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td><code>IsShorterOrEqual(ConditionValidator&lt;Of &lt;(String)&gt;&gt;, Int32, String)</code></td>
<td>Checks whether the given value is shorter or equal in length than maxLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td><code>IsShorterOrEqual&lt;Of &lt;(TCollection)&gt;&gt;(ConditionValidator&lt;Of &lt;(TCollection)&gt;&gt;, Int32, String)</code></td>
<td>Checks whether the number of elements in the given value, is less than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
reference, it is considered to have 0 elements.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

`ValidatorExtensions..:::.IsShorterOrEqual Method (ConditionValidator<(Of <(String>)>), Int32)`

See Also
Send Feedback

Checks whether the given value is shorter or equal in length than `maxLength`. An exception is thrown otherwise. A null reference is considered to have a length of 0.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsShorterOrEqual ( _
    validator As ConditionValidator(Of String), _
    maxLength As Integer _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> IsShorterOrEqual(
    ConditionValidator<string> validator,
    int maxLength
)

Visual C++

public:
static ConditionValidator<String>^ IsShorterOrEqual(
    ConditionValidator<String>^ validator,
    int maxLength
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isShorterOrEqual = function;

Parameters

validator
    Type: CuttingEdge.Conditions..::: ConditionValidator(Of String)>
    The ConditionValidator(Of String) that holds the value that has to be checked.

maxLength
    Type: System..::: Int32
    The biggest valid length.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>System..::.ArgumentException</em></td>
<td>Thrown when the length of the Value of the specified validator is smaller than maxLength, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><em>System..::.ArgumentNullException</em></td>
<td>Thrown when the Value of the specified validator is a null reference and maxLength is smaller than 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td><em>CuttingEdge.Conditions..::.PostconditionException</em></td>
<td>Thrown when the length of the Value of the specified validator is smaller than maxLength, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsShorterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions:::IsShorterOrEqual Method (ConditionValidator<Of (String>), Int32, String)

Checks whether the given value is shorter or equal in length than maxLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.

**Namespace:** CuttingEdge.Conditions
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

Public Shared Function IsShorterOrEqual ( _
    validator As ConditionValidator(Of String), _
    maxLength As Integer, _
    conditionDescription As String )
) As ConditionValidator(Of String)

### C#

public static ConditionValidator<String> IsShorterOrEqual(    
    ConditionValidator<String> validator,
    int maxLength,
    string conditionDescription
)

### Visual C++

public:
static ConditionValidator<String>^ IsShorterOrEqual(    
    ConditionValidator<String>^ validator,
    int maxLength,
    String^ conditionDescription
)

### JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isShorterOrEqual = function

### Parameters

**validator**

Type: CuttingEdge.Conditions.:::ConditionValidator(Of String>)

The ConditionValidator(Of (T)> that holds the value that has to be checked.

**maxLength**
Type: `System::Int32`
The biggest valid length.

**conditionDescription**
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**
The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the length of the Value of the specified validator is smaller than maxLength, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and maxLength is smaller than 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the length of the Value of the specified validator is smaller than maxLength, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsShorterOrEqual Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the number of elements in the given value, is less than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsShorterOrEqual(Of TCollection As IEnumerable) validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsShorterOrEqual<TCollection>(ConditionValidator<TCollection> validator,
    int numberOfElements
)
where TCollection : IEnumerable

Visual C++

public:
    generic<typename TCollection>
where TCollection : IEnumerable
static ConditionValidator<TCollection>^ IsShorterOrEqual(ConditionValidator<TCollection>^ validator,
    int numberOfElements
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.
numberOfElements
Type: System::Int32
The collection must contain the same amount or less elements than this value.
Type Parameters

TCollection
   The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System:::ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more elements than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>System:::ArgumentNullException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is less than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions:::PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more elements than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**  
**IsShorterOrEqual Overload**  
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
ValidatorExtensions.IsShorterOrEqual<TCollection<T>, Of <(TCollection)>>(TCollection<T>, Int32, String)

Checks whether the number of elements in the given value, is less than or equal to the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsShorterOrEqual(Of TCollection As IEnumerable) validator As ConditionValidator(Of TCollection), _
    numberOfElements As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsShorterOrEqual<TCol](
    ConditionValidator<TCollection> validator,
    int numberOfElements,
    string conditionDescription
)

where TCollection : IEnumerable

Visual C++

generic<typename TCollection>

where TCollection : IEnumerable
static ConditionValidator<TCollection>^ IsShorterOrEqual(ConditionValidator<TCollection>^ validator,
    int numberOfElements,
    String^ conditionDescription
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(TCollection)>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

numberOfElements
   Type: System::Int32
   The collection must contain the same amount or less elements than this value.

conditionDescription
   Type: System::String
   The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
**Type Parameters**

TCollection
   The type of the value to check.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System:::ArgumentException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more elements than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>System:::ArgumentNullException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator is a null reference and the <code>numberOfElements</code> is less than 0, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><code>CuttingEdge.Conditions:::PostconditionException</code></td>
<td>Thrown when the <code>Value</code> of the specified validator contains more elements than specified by the <code>numberOfElements</code> argument, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsShorterOrEqual Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript
Include Protected Members  Include Inherited Members
CuttingEdge.Conditions reference library
ValidatorExtensions...::IsShorterThan Method
ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsShorterThan(ConditionValidator(Of (String)&gt;, Int32)</td>
<td>Checks whether the given value is shorter in length than maxLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td>IsShorterThan(Of (TCollection)&gt;) (ConditionValidator(Of (TCollection)&gt;, Int32)</td>
<td>Checks whether the number of elements in the given value, is less than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
<tr>
<td>IsShorterThan(ConditionValidator(Of (String)&gt;, Int32, String)</td>
<td>Checks whether the given value is shorter in length than maxLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.</td>
</tr>
<tr>
<td>IsShorterThan(Of (TCollection)&gt;) (ConditionValidator(Of (TCollection)&gt;, Int32, String)</td>
<td>Checks whether the number of elements in the given value, is less than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.</td>
</tr>
</tbody>
</table>
reference, it is considered to have 0 elements.
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □  C#  □  Visual C++  □  JavaScript
CuttingEdge.Conditions reference library

ValidatorExtensions...:::.IsShorterThan Method (ConditionValidator<Of <(String)>>, Int32)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is shorter in length than maxLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.

Namespace: CuttingEdge.Conditions
Assembly: CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

Public Shared Function IsShorterThan ( _
    validator As ConditionValidator(Of String), _
    maxLength As Integer _
) As ConditionValidator(Of String)

**C#**

public static ConditionValidator<string> IsShorterThan(ConditionValidator<string> validator,
    int maxLength
)

**Visual C++**

public:
    static ConditionValidator<String>^ IsShorterThan(ConditionValidator<String>^ validator,
    int maxLength
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.isShorterThan = function(

**Parameters**

validator
    Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(String)>)>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

maxLength
    Type: System..:::Int32
    The smallest invalid length.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..:::.ArgumentException</td>
<td>Thrown when the length of the Value of the specified validator is greater or equal to maxLength, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..:::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and maxLength is smaller or equal to 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..:::.PostconditionException</td>
<td>Thrown when the length of the Value of the specified validator is greater or equal to maxLength to, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

**ValidatorExtensions Class**
**IsShorterThan Overload**
**CuttingEdge.Conditions Namespace**

Send [feedback](mailto:feedback@microsoft.com) on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions.IsShorterThan Method (ConditionValidator<br>(Of<br>(String)), Int32, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is shorter in length than maxLength. An exception is thrown otherwise. A null reference is considered to have a length of 0.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsShorterThan ( _
    validator As ConditionValidator(Of String), _
    maxLength As Integer, _
    conditionDescription As String _
) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> IsShorterThan(
    ConditionValidator<string> validator,
    int maxLength,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<String^>^ IsShorterThan(
    ConditionValidator<String^>^ validator,
    int maxLength,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isShorterThan = function(
```

### Parameters

**validator**
- Type: `CuttingEdge.Conditions..::.ConditionValidator<Of <(String)>>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**maxLength**
Type: `System::Int32`
The smallest invalid length.

`conditionDescription`
Type: `System::String`
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the length of the Value of the specified validator is greater or equal to maxLength, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and maxLength is smaller or equal to 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the length of the Value of the specified validator is greater or equal to maxLength, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

- [ValidatorExtensions Class](#)
- [IsShorterThan Overload](#)
- [CuttingEdge.Conditions Namespace](#)

Send [feedback](#) on this topic to Microsoft.
Checks whether the number of elements in the given value, is less than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** [CuttingEdge.Conditions](#)  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsShorterThan(Of TCollection As IEnumerable) 
    validator As ConditionValidator(Of TCollection), _ 
    numberOfElements As Integer 
) As ConditionValidator(Of TCollection)
```

**C#**

```csharp
public static ConditionValidator<TCollection> IsShorterThan<TCollection>(
    ConditionValidator<TCollection> validator, 
    int numberOfElements
)
```

where TCollection : IEnumerable

**Visual C++**

```cpp
public:
    generic<typename TCollection>
    where TCollection : IEnumerable
    static ConditionValidator<TCollection>^ IsShorterThan(
        ConditionValidator<TCollection>^ validator, 
        int numberOfElements
    )
```

**JavaScript**

JavaScript does not support generic types or methods.

**Parameters**

**validator**

Type: `CuttingEdge.Conditions..::.ConditionValidator<Of <(TCollection)>>`

The `ConditionValidator<Of <(T)>>` that holds the value that has to be checked.
numberOfElements

Type: System::Int32

The collection must contain less elements than this value.
Type Parameters

TCollection
    The type of the value to check.

Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>Thrown when the Value of the specified validator contains more or the same amount of elements as specified by the numberOfElements argument, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator is a null reference and the numberOfElements is smaller or equal to 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System..::.ArgumentNullException</td>
<td>Thrown when the Value of the specified validator contains more or the same amount of elements as specified by the numberOfElements argument, while the specified validator is created using the Ensures extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator contains more or the same amount of elements as specified by the numberOfElements argument, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsShorterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the number of elements in the given value, is less than the specified numberOfElements argument. An exception is thrown otherwise. When the value is a null reference, it is considered to have 0 elements.

**Namespace:** [CuttingEdge.Conditions](http://CuttingEdge.Conditions)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsShorterThan(Of TCollection As IEnumerable) validator As ConditionValidator(Of TCollection), _
   numberOfElements As Integer, _
   conditionDescription As String _
) As ConditionValidator(Of TCollection)

C#

public static ConditionValidator<TCollection> IsShorterThan<TCollection>(ConditionValidator<TCollection> validator, _
   int numberOfElements, _
   string conditionDescription _
) _
where TCollection : IEnumerable

Visual C++

public:

generic<typename TCollection>
where TCollection : IEnumerable

static ConditionValidator<TCollection>^ IsShorterThan(ConditionValidator<TCollection>^ validator, _
   int numberOfElements, _
   String^ conditionDescription _
)

JavaScript

JavaScript does not support generic types or methods.

Parameters

validator

Type: CuttingEdge.Conditions..::..ConditionValidator<(Of _
   <(TCollection)>>)
The ConditionValidator<(Of <(T)>)> that holds the value that has to be
checked.

**numberOfElements**
Type: *System::Int32*
The collection must contain less elements than this value.

**conditionDescription**
Type: *System::String*
The description of the condition that should hold. The string may hold the placeholder '{0}' for the **ArgumentName**.
Type Parameters

TCollection
  The type of the value to check.

Return Value

The specified validator instance.
### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator contains more or the same amount of elements as specified by the numberOfElements argument, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and the numberOfElements is smaller or equal to 0, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator contains more or the same amount of elements as specified by the numberOfElements argument, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsShorterThan Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions.IsTrue Method
### Overload List

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="image" /></td>
<td><code>IsTrue(ConditionValidator&lt;Of &lt;(Boolean)&gt;&gt;)</code>)</td>
<td>Checks whether the given value is <strong>true</strong>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><img src="image.png" alt="image" /></td>
<td><code>IsTrue(ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(Boolean)&gt;&gt;)&gt;&gt;))</code></td>
<td>Checks whether the given value is <strong>true</strong>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><img src="image.png" alt="image" /></td>
<td><code>IsTrue(ConditionValidator&lt;Of &lt;(Boolean)&gt;&gt;, String)</code></td>
<td>Checks whether the given value is <strong>true</strong>. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><img src="image.png" alt="image" /></td>
<td><code>IsTrue(ConditionValidator&lt;Of &lt;(Nullable&lt;Of &lt;(Boolean)&gt;&gt;)&gt;&gt;), String)</code></td>
<td>Checks whether the given value is <strong>true</strong>. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions..::.IsTrue Method (ConditionValidator<Of <(Boolean)>))

Checks whether the given value is **true**. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsTrue ( _
validator As ConditionValidator(Of Boolean) _
) As ConditionValidator(Of Boolean)

C#

public static ConditionValidator<bool> IsTrue(
    ConditionValidator<bool> validator
)

Visual C++

public:
    static ConditionValidator<bool>^ IsTrue(
        ConditionValidator<bool>^ validator
    )

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isTrue = function(validat

Parameters

validator
    Type: CuttingEdge.Conditions.:::ConditionValidator<(Of (Boolean)>)
The ConditionValidator<(Of (T)>) that holds the value that has to be checked.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System::ArgumentException</td>
<td>Thrown when the Value of the specified validator is false, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions::PostconditionException</td>
<td>Thrown when the Value of the specified validator is false, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsTrue Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  C#  Visual C++  JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::IsTrue Method (ConditionValidator<Of <(Boolean)>>, String)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value is **true**. An exception is thrown otherwise.

**Namespace:**  CuttingEdge.Conditions

**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function IsTrue ( _
    validator As ConditionValidator(Of Boolean), _
    conditionDescription As String _
) As ConditionValidator(Of Boolean)

C#

public static ConditionValidator<bool> IsTrue(
    ConditionValidator<bool> validator,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<bool>^ IsTrue(
    ConditionValidator<bool>^ validator,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.isTrue = function(validat

Parameters

validator
    Type: CuttingEdge.Conditions:::ConditionValidator<(Of <(Boolean)>>)
    The ConditionValidator<(Of <(T)>)> that holds the value that has to be checked.

conditionDescription
    Type: System:::String
    The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.
Return Value

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
<td>Thrown when the Value of the specified validator is false, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
<td>Thrown when the Value of the specified validator is false, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsTrue Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions...:::.IsTrue Method (ConditionValidator<Of <(Nullable<Of <(Boolean)>)>>>)

Checks whether the given value is **true**. An exception is thrown otherwise.

**Namespace:**  [CuttingEdge.Conditions](#)  
**Assembly:**  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function IsTrue ( _
    validator As ConditionValidator(Of Nullable(Of Boolean)) _) As ConditionValidator(Of Nullable(Of Boolean))
```

**C#**

```csharp
public static ConditionValidator<Nullable<bool>> IsTrue(ConditionValidator<Nullable<bool>> validator)
```

**Visual C++**

```cpp
public:
static ConditionValidator<Nullable<bool>>^ IsTrue(ConditionValidator<Nullable<bool>>^ validator)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isTrue = function(validator
```

**Parameters**

- **validator**
  - Type: `CuttingEdge.Conditions.;;.ConditionValidator<(Of <(Nullable<(Of <(Boolean)<(Of
  - The `ConditionValidator<(Of <(T)>)>` that holds the value that has to be checked.

**Return Value**

- The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is <code>false</code> or null, while the specified validator is created using the <code>Requires</code> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <code>Value</code> of the specified validator is <code>false</code> or null, while the specified validator is created using the <code>Ensures</code> extension method.</td>
</tr>
</tbody>
</table>
See Also

*ValidatorExtensions Class*
*IsTrue Overload*
*CutingEdge.Conditions Namespace*

Send [feedback](#) on this topic to Microsoft.
ValidatorExtensions..:::..:::..:::..:::..:::.IsTrue Method (ConditionValidator<Of <(Nullable<Of <(Boolean)>)>), String)

Checks whether the given value is **true**. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](#)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
## Syntax

### Visual Basic (Declaration)

```vbnet
Public Shared Function IsTrue ( _
    validator As ConditionValidator(Of Nullable(Of Boolean)), _
    conditionDescription As String _
) As ConditionValidator(Of Nullable(Of Boolean))
```

### C#

```csharp
public static ConditionValidator<Nullable<bool>> IsTrue(
    ConditionValidator<Nullable<bool>> validator,
    string conditionDescription
)
```

### Visual C++

```cpp
public:
static ConditionValidator<Nullable<bool>>^ IsTrue(
    ConditionValidator<Nullable<bool>>^ validator,
    String^ conditionDescription
)
```

### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.isTrue = function(validator,
```

## Parameters

**validator**

Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(Nullable<Of <(Boolean)>)>)>`

The `ConditionValidator<Of <(T)>)>` that holds the value that has to be checked.

**conditionDescription**

Type: `System..:::String`

The description of the condition that should hold. The string may hold the
placeholder '{0}' for the ArgumentName.

Return Value

The specified validator instance.
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is <strong>false</strong> or null, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is <strong>false</strong> or null, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
IsTrue Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
CuttingEdge.Conditions reference library

ValidatorExtensions...:..:StartsWith Method

ValidatorExtensions Class  See Also  Send Feedback
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StartsWith(ConditionValidator&lt;(&lt;String&gt;), String)</code></td>
<td>Checks whether the given value starts with the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>StartsWith(ConditionValidator&lt;(&lt;String&gt;), String, String)</code></td>
<td>Checks whether the given value starts with the specified value. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>StartsWith(ConditionValidator&lt;(&lt;String&gt;), String, StringComparison)</code></td>
<td>Checks whether the given value starts with the specified value using the specified comparisonType. An exception is thrown otherwise.</td>
</tr>
<tr>
<td><code>StartsWith(ConditionValidator&lt;(&lt;String&gt;), String, StringComparison, String)</code></td>
<td>Checks whether the given value starts with the specified value using the specified comparisonType. An exception is thrown otherwise.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
ValidatorExtensions Members
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Checks whether the given value starts with the specified value. An exception is thrown otherwise.

**Namespace:** [CuttingEdge.Conditions](http://example.com)

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

#### Visual Basic (Declaration)

```vbnet
Public Shared Function StartsWith (_
    validator As ConditionValidator(Of String), _
    value As String _
) As ConditionValidator(Of String)
```

#### C#

```csharp
public static ConditionValidator<string> StartsWith(
    ConditionValidator<string> validator,
    string value
)
```

#### Visual C++

```cpp
public:
static ConditionValidator<String^>^ StartsWith(
    ConditionValidator<String^>^ validator,
    String^ value
)
```

#### JavaScript

```javascript
CuttingEdge.Conditions.ValidatorExtensions.startsWith = function(val
```

### Parameters

**validator**

- Type: `CuttingEdge.Conditions..:::ConditionValidator<Of <(String)>)`
- The `ConditionValidator<Of <(T)>)` that holds the value that has to be checked.

**value**

- Type: `System..:::String`
- The value to compare.
Return Value

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentException</td>
</tr>
</tbody>
</table>

Thrown when the Value of the specified validator does not start with `value`, while the specified validator is created using the Requires extension method.

<table>
<thead>
<tr>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>System..::.ArgumentNullException</td>
</tr>
</tbody>
</table>

Thrown when the Value of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the Requires extension method.

<table>
<thead>
<tr>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>CuttingEdge.Conditions..::.PostconditionException</td>
</tr>
</tbody>
</table>

Thrown when the Value of the specified validator does not start with `value`, while the specified validator is created using the Ensures extension method.
See Also

ValidatorExtensions Class
StartsWith Overload
CuttingEdge.Constraints Namespace

Send feedback on this topic to Microsoft.
ValidatorExtensions.:::StartsWith Method (ConditionValidator<Of (String>), String, String)

Checks whether the given value starts with the specified value. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions

**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
**Syntax**

**Visual Basic (Declaration)**

```vbnet
Public Shared Function StartsWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    conditionDescription As String _
) As ConditionValidator(Of String)
```

**C#**

```csharp
public static ConditionValidator<string> StartsWith(
    ConditionValidator<string> validator,
    string value,
    string conditionDescription
)
```

**Visual C++**

```cpp
public:
static ConditionValidator<String^>^ StartsWith(
    ConditionValidator<String^>^ validator,
    String^ value,
    String^ conditionDescription
)
```

**JavaScript**

```javascript
CuttingEdge.Conditions.ValidatorExtensions.startsWith = function(val
```

**Parameters**

- **validator**
  - Type: `CuttingEdge.Conditions...:::ConditionValidator<Of <(String)>>)`
  - The `ConditionValidator<Of <(T)>>)` that holds the value that has to be checked.

- **value**
Type: `System.String`  
The value to compare.

`conditionDescription`  
Type: `System.String`  
The description of the condition that should hold. The string may hold the placeholder '{0}' for the `ArgumentName`.

**Return Value**

The specified validator instance.
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not start with value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not start with value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
StartsWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.
Visual Basic  □ C#
□ Visual C++
□ JavaScript

CuttingEdge.Conditions reference library

ValidatorExtensions...:::StartsWith Method (ConditionValidator<Of (String>), String, StringComparison)

ValidatorExtensions Class  See Also  Send Feedback

Checks whether the given value starts with the specified value using the specified comparisonType. An exception is thrown otherwise.

Namespace:  CuttingEdge.Conditions
Assembly:  CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
### Syntax

**Visual Basic (Declaration)**

Public Shared Function StartsWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison _
) As ConditionValidator(Of String)

**C#**

public static ConditionValidator<string> StartsWith(
    ConditionValidator<string> validator,
    string value,
    StringComparison comparisonType
)

**Visual C++**

public:
static ConditionValidator<String^>^ StartsWith(
    ConditionValidator<String^>^ validator,
    String^ value,
    StringComparison comparisonType
)

**JavaScript**

CuttingEdge.Conditions.ValidatorExtensions.startsWith = function(val

### Parameters

**validator**

Type: CuttingEdge.Conditions..:::ConditionValidator<(Of <(String)>>)
The ConditionValidator(Of <(T)>) that holds the value that has to be checked.

**value**
Type: `System.String`
The value to compare.

`comparisonType`
Type: `System:StringComparison`
One of the `StringComparison` values that determines how this string and value are compared

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System..::.ArgumentException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not start with value, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>System..::.ArgumentNullException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the <strong>Requires</strong> extension method.</td>
</tr>
<tr>
<td><strong>CuttingEdge.Conditions..::.PostconditionException</strong></td>
<td>Thrown when the <strong>Value</strong> of the specified validator does not start with value, while the specified validator is created using the <strong>Ensures</strong> extension method.</td>
</tr>
</tbody>
</table>
See Also

[ValidatorExtensions Class]
[StartsWith Overload]
[CuttingEdge.Conditions Namespace]

Send [feedback] on this topic to Microsoft.
Checks whether the given value starts with the specified value using the specified comparison type. An exception is thrown otherwise.

**Namespace:** CuttingEdge.Conditions  
**Assembly:** CuttingEdge.Conditions (in CuttingEdge.Conditions.dll)
Syntax

Visual Basic (Declaration)

Public Shared Function StartsWith ( _
    validator As ConditionValidator(Of String), _
    value As String, _
    comparisonType As StringComparison, _
    conditionDescription As String _
) As ConditionValidator(Of String)

C#

public static ConditionValidator<string> StartsWith(
    ConditionValidator<string> validator,
    string value,
    StringComparison comparisonType,
    string conditionDescription
)

Visual C++

public:
static ConditionValidator<String>^ StartsWith(
    ConditionValidator<String>^ validator,
    String^ value,
    StringComparison comparisonType,
    String^ conditionDescription
)

JavaScript

CuttingEdge.Conditions.ValidatorExtensions.startsWith = function(val}
value
Type: System.String
The value to compare.

comparisonType
Type: System.StringComparison
One of the StringComparison values that determines how this string and value are compared

conditionDescription
Type: System.String
The description of the condition that should hold. The string may hold the placeholder '{0}' for the ArgumentName.

**Return Value**

The specified validator instance.
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:::ArgumentException</td>
<td>Thrown when the Value of the specified validator does not start with value, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>System:::ArgumentNullException</td>
<td>Thrown when the Value of the specified validator is a null reference and value is not a null reference, while the specified validator is created using the Requires extension method.</td>
</tr>
<tr>
<td>CuttingEdge.Conditions:::PostconditionException</td>
<td>Thrown when the Value of the specified validator does not start with value, while the specified validator is created using the Ensures extension method.</td>
</tr>
</tbody>
</table>
See Also

ValidatorExtensions Class
StartsWith Overload
CuttingEdge.Conditions Namespace

Send feedback on this topic to Microsoft.