Indice de Documentacion de Bazaar

Las ultimas versiones de estos documentos se encuentran disponibles en la pagina de Bazaar, http://doc.bazaar-vcs.org/en/.

Documentacion Principal

- Bazaar en cinco minutos
- Referencia Rapida
- Guia de Usuarios de Bazaar

Otra Documentacion

Mudandose a Bazaar (enlances web):

- Guias de Migracion - para equipos que mudan el historial de otros SCV
- Guias de Plugins

Otros Documentos (enlances web):

- Glosario
- Preguntas Frecuentes
- Referencia del API bzrlib

Especificaciones tecnicas, planes a futuro y varias notas tecnicas varias en Ingles:

- Catalogo de Documentos para Desarrolladores

Bazaar en cinco minutos

Introducción

Bazaar es un sistema de control de versiones distribuido que facilita que varias personas puedan trabajar de forma conjunta en proyectos de software.

A lo largo de los próximos cinco minutos, aprenderá cómo poner sus archivos bajo control de versiones, como registrar cambios en ellos, examinar su trabajo, publicarlo y enviar su trabajo para que sea integrado en el trunk de un proyecto.

Si prefiere una introducción más detallada, eche un vistazo a Aprendiendo Más.

Instalación

Esta guía no describe cómo instalar Bazaar pero normalmente es muy sencillo. Puede encontrar intrucciones de instalación en:

- **GNU/Linux:** Bazaar, probablemente, ya esté en su distribución GNU/Linux.
- **Windows:** instrucciones de instalación para Windows.
- **Mac OS X:** instrucciones de instalación para Mac OS X.

Para otras plataformas y para instalar desde el código fuente, vea las páginas de Descarga e Instalación.

Preséntese

Antes de empezar a trabajar, es conveniente que le diga a Bazaar quién es usted. De ese modo su trabajo será identificando correctamente en los logs de revisión.

Utilice su nombre y dirección de email en lugar de John Doe, teclee:

```
$ bzr whoami "John Doe <john.doe@gmail.com>"
```

Bazaar creará o modificará ahora un archivo de configuración, incluyendo su nombre y dirección de email.

Ahora compruebe que su nombre y dirección de email se han registrado correctamente:

```
$ bzr whoami
John Doe <john.doe@gmail.com>
```

Ponga archivos bajo control de versiones

Vamos a crear un directorio y algunos archivos para utilizar con Bazaar:

```
$ mkdir miproyecto
$ cd miproyecto
$ mkdir subdirectorio
$ touch test1.txt test2.txt test3.txt subdirectorio/test4.txt
```

Nota para usuarios de Windows: utilice Windows Explorer para crear sus directorios, luego haga click derecho en dichos directorios y seleccione `Nuevo archivo` para crear sus archivos.

Ahora vamos a hacer que Bazaar se inicialize en el directorio de su proyecto:

```
$ bzr init
```

Si parece que no ha ocurrido nada no se preocupe. Bazaar ha creado un branch dónde guardará sus archivos y su histórico de revisiones.

El siguiente paso es decirle a Bazaar a que archivos desea seguirles la pista. Ejecutando `bzr add` agregará recursivamente todos los elementos dentro del proyecto:

```
$ bzr add
added subdirectorio
added test1.txt
added test2.txt
added test3.txt
added subdirectorio/test4.txt
```

A continuación tome una instantánea de sus archivos agregándolos a su branch. Agregue un mensaje para explicar por qué hace el

commit:

```
$ bzr commit -m "Importación inicial"
```

Como Bazaar es un sistema de control de versiones distribuido, no necesita conectar con un servidor central para hacer el commit. Bazaar guarda su branch y todos sus commits dentro del directorio con el que está trabajando, busque el subdirectorio `.bzr`.

Haciendo cambios en sus archivos

Vamos a cambiar un archivo e introduzcamos ese cambio en su branch.

Edite `test1.txt` en su editor favorito y luego compruebe qué ha hecho:

```
$ bzr diff
=== modified file 'test1.txt'
--- test1.txt   2007-10-08 17:56:14 +0000
+++ test1.txt   2007-10-08 17:46:22 +0000
@@ -0,0 +1,1 @@
+test test test
```

Añada su trabajo al branch de Bazaar:

```
$ bzr commit -m "Añadida la primera línea de texto"
Committed revision 2.
```

Viendo el log de revisiones

Puede ver el histórico de su branch navegando su log:

```
$ bzr log
------------------------------------------------------------
revno: 2
committer: John Doe <john.doe@gmail.com>
branch nick: miproyecto
timestamp: Mon 2007-10-08 17:56:14 +0000
message:
  Añadida la primera línea de texto
------------------------------------------------------------
revno: 1
committer: John Doe <john.doe@gmail.com>
branch nick: miproyecto
timestamp: Mon 2006-10-08 17:46:22 +0000
message:
  Importación inicial
```

Publicando su branch con sftp

Hay un par de maneras para publicar su branch. Si ya tiene un servidor SFTP o se siente cómodo configurando uno, puede publicar su branch con el.

Sino salte a la siguiente sección para publicar con Launchpad, un servicio de hosting gratuito para Bazaar.

Vamos a suponer que desea publicar su branch en `www.example.com/miproyecto`:

```
$ bzr push --create-prefix sftp://su.nombre@example.com/~/publi
2 revision(s) pushed.
```

Bazaar creará un directorio `miproyecto` en el servidor remoto e introducirá su branch en él.

Ahora cualquiera podrá crear su propia copia de su branch tecleando:

```
$ bzr branch http://www.example.com/miproyecto
```

Nota: para utilizar sftp deberá instalar `paramiko` y `pyCrypto`. Vea http://bazaar-vcs.org/InstallationFaq para más información.

Publicando su branch con Launchpad

Launchpad es una suite de herramientas de desarrollo y hosting para proyectos de software libre. Puede utilizarlo para publicar su branch.

Si no dispone de una cuenta de Launchpad, siga la guia de registro de cuentas y registre una clave SSH en su nueva cuenta de Launchpad.

Cambie `john.doe` por su nombre de usuario de Launchpad, teclee:

```
$ bzr push bzr+ssh://john.doe@bazaar.launchpad.net/~john.doe/+j
```

Nota: `+junk` significa que este branch no está asociado con ningún proyecto concreto en Launchpad.

Ahora cualquiera podrá crear su propia copia de su branch tecleando:

```
$ bzr branch http://bazaar.launchpad.net/~john.doe/+junk/miproy
```

También puede ver información sobre su branch, histórico de revisiones incluido, en https://code.launchpad.net/people/+me/+junk/miproyecto

Creando su propia copia de otro branch

Para trabajar con el código de otra persona, tendrá que hacer su propia copia de su branch. Vamos a coger un ejemplo real, la interfaz GTK de Bazaar:

```
$ bzr branch http://bazaar.launchpad.net/~bzr/bzr-gtk/trunk bzr
Branched 292 revision(s).
```

Bazaar descargará todos los archivos y el histórico de revisiones completo del trunk branch del proyecto bzr-gtk y creará una copia llamada bzr-gtk.john.

Ahora dispone de su propia copia del branch y puede enviar cambios con o sin una conexión de red. Puede compartir su branch en cualquier momento publicándola y, si el equipo de bzr-gtk desea utilizar su trabajo, Bazaar les facilita integrar su branch dentro de su trunk branch.

Actualizando su branch desde el branch principal

Mientras envía cambios a su branch, es probable que otras personas también sigan enviando código al branch principal.

Para asegurarse de que su branch está al dia debería integrar los cambios desde el principal dentro de su branch personal:

```
$ bzr merge
Merging from saved parent location: http://bazaar.launchpad.net
All changes applied successfully.
```

Compruebe qué ha cambiado:

```
$ bzr diff
```

Si está contento con los cambios puede añadirlos en su branch personal:

```
$ bzr commit -m "Integración desde el branch principal"
Committed revision 295.
```

Integrando su trabajo en el branch principal

Después de haber trabajado en su branch personal de bzr-gtk puede que quiera enviar sus cambios de vuelta al proyecto. La manera más fácil es utilizando una instrucción merge.

Una instrucción merge es una petición de lectura mecánica para llevar a cabo una integración concreta. Por lo general contiene un parche de vista previa de la integración y, o bien contiene las revisiones necesarias, o proporciona un branch donde pueden encontrarse.

Sustituyendo `mycode.patch`, cree su instrucción merge:

```
$ bzr send -o mycode.patch
Using saved parent location: http://bazaar.launchpad.net/~bzr/b
```

Ahora puede enviar por email la instruccion merge al proyecto bzr-gtk quien, si así lo quieren, pueden utilizarla para integrar su trabajo dentro del branch principal.

Aprendiendo más

Puede encontrar más sobre Bazaar en la Guía de Usuario de Bazaar.

Para aprender sobre Bazaar por línea de comandos:

```
$ bzr help
```

Para aprender sobre comandos de Bazaar:

```
$ bzr help commands
```

Para aprender acerca del tema o comando ‘’foo’‘:

```
$ bzr help foo
```

Referencia Rapida

- PDF format
- PNG format
- SVG format

Guia de Usuarios de Bazaar

Introducción

Empezando

Control de Versionamiento Personal

Compartiendo con tus pares

Colaboración en equipo, modo centralizado

Colaboracion en equipo, modo distribuido

Un tour breve de los plugins mas populares

Integrando Bazaar en tu entorno

Temas varios

Usando `bzr version-info`

Repaso General

Este documento describe las formas de usar `bzr version-info` como parte del proceso de embeber la informacion de vesion a un proyecto.

Projecto Python

TODO: Figure out how to attach into `setup.py`

Si usa un archivo Makefile para construir su proyecto, puede generar un archivo on la informacion de version tan simple como:

```
library/_version.py:
      bzr version-info --format=python > library/_version.py
```

Eso genera un archivo que contiene 3 diccionarios:

- *version_info*: Un diccionario conteniendo informacion basica sobre el estado actual
- *revisions*: Un diccionario listando todas las revisiones en el historial del tree, junto con los tiempos y los mensajes de los commits. Esto por defecto esta en blanco salvi que use `--all` o *–include-history*` es provisto. Esto es util si quiere seguir que bugs arregla el lanzamiento de esa version. Para muchos proyectos es mas informacion de la que se va a necesitar.
- *file_revisions*: Un diccionario listando la revision que modifico por ultima vez todos los archivos del proyecto.

Esto puede ser usado similarmente a como se usan las palabras claves `$Id$` en los archivos controlados en CVS. La ultima fecha de modificacion puede ser determinada mirando en el mapa de `revisions`. Esto tambien esta vacio por defecto, y habilitado solo por `--all` o `--include-file-revisions`.

Check Clean

La mayoria de la informacion sobre el contenido del proyecto puede ser determinada a muy bajo costo con solo leer las entradas de revisiones. Sin embargo, puede ser util si el working tree fue actualizado completamente cuando fue empaquetado, o si hubo alguna modificacion local. Al proveer `--all` o `--check-clean`, `bzr` va a inspeccionar el working tree, y definir el `clean` flag en `version_info`, al igual que definir entradas en `file_revisions` como `modified` donde es apropiado.

Apéndice