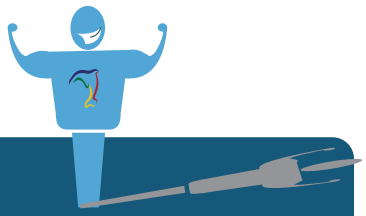
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

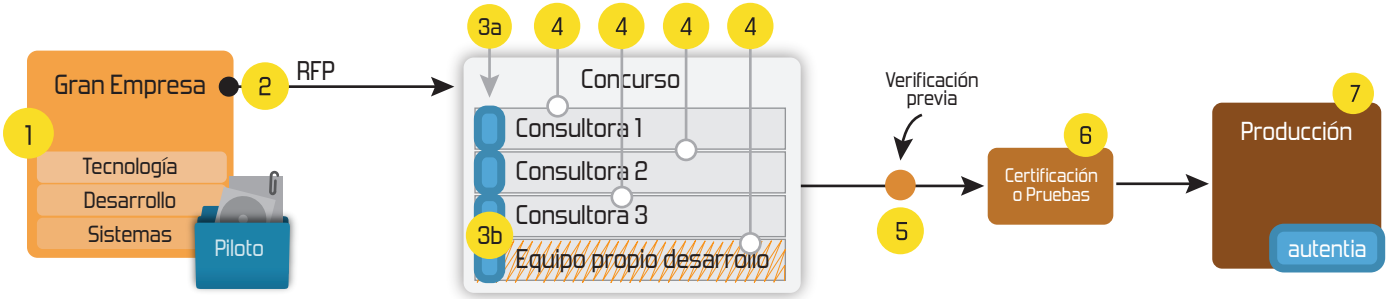
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

5.3 Trabajo habitual con el repositorio local
5.3.1 Añadir un fichero al repositorio.
5.3.2 Conocer el estado de los ficheros.
5.3.3 Ver el historial de cambios
5.3.4 Conocer las diferencias entre ficheros.
5.3.5 Subir los cambios al repositorio local.
5.4 Propagando los cambios a otro repositorio.
5.4.1 Sincronización con otro repositorio.
5.4.2 Subir los cambios a otro repositorio.
6. Conclusiones.

1. Introducción

Mercurial es un sistema de control de versiones distribuido libre, gratuito y similar a Git o Bazaar. Todos ellos rompen con el modelo tradicional de trabajo cliente/servidor de los repositorios como CVS o Subversión.

En los repositorios tradicionales existe una clara diferencia entre los conceptos servidor y cliente: el servidor es el que mantiene y controla el versionado de los ficheros, y el cliente se descarga copias del servidor para que sean modificadas y subidas posteriormente con el fin de que sean vistas por el resto de usuarios. En cambio, en los repositorios distribuidos, los clientes ya no se descargan una copia del repositorio, sino que lo clonan, comportándose como cliente y servidor al mismo tiempo, es decir, que la copia podría comportarse a su vez como servidor ante otros clientes (construyendo así una estructura distribuida).

A pesar de que la estructura y gestión es algo más compleja que la de un SCM tradicional, existen situaciones en las que los repositorios distribuidos son muy útiles, por ejemplo, en proyectos donde los desarrolladores se encuentren continuamente incomunicados y necesitan realizar acciones tan habituales como commit (locales), branch, etc ...

Además de los comandos tradicionales que se pueden ejecutar sobre un sistema de control de versiones, los repositorios distribuidos añaden nuevos como: push y pull. Estos dos comandos permiten a un usuario subir los cambios locales a un servidor remoto y actualizarse con los cambios realizados por otros usuarios respectivamente.

Lo que pretendemos con este tutorial es que tengáis una primera toma de contacto con Mercurial, que sepáis cómo instalarlo y que repasemos los comandos básicos para empezar a trabajar con él.

2. Entorno

Entorno utilizado para escribir este tutorial:

- **Hardware**: Mac Book Pro (Core 2 Duo 2,8 Ghz, 4 GB RAM, 500 GB)
- **Sistema Operativo**: Snow Leopard
- **Mercurial**: 1.6.3
- **Python**: 2.6.1

tipo cómic de Autentia

Histórico de NOTICIAS

Últimos Tutoriales

Instalación de Ubuntu Server 8.04 LTS 32bits en una máquina virtual con VMWare Workstation

Manual de Javascript

Instalación de Ubuntu Server 10.04 LTS 32bits en una máquina virtual con VMWare Workstation

Sniping en eBay (o cómo ganar las subastas)

Instalación de Ubuntu Desktop 8.04 LTS 32bits en una máquina virtual con VMWare Workstation

Últimos Tutoriales del Autor

MediaWiki - NamespacePermissions

Facebook Social Plugins

Desplegando en Tomcat un proyecto web con Hudson.

DBUnit y aplicaciones JDBC

3. Instalación

Aunque existe la posibilidad de descargarnos el código fuente para generar una versión de Mercurial por nosotros mismos, en este tutorial vamos a instalar una versión ya compilada.

Para ello, accedemos a la página oficial de Mercurial (http://mercurial.selenic.com/). En la parte superior derecha pulsamos sobre "Download Now" y nos descargamos el instalador. Un apunte muy importante es que para que Mercurial se instale correctamente, tendremos que tener instalado en nuestro sistema la versión 2.4 o superior de python.

Intercomunicación entre portlets.

Síguenos a través de:

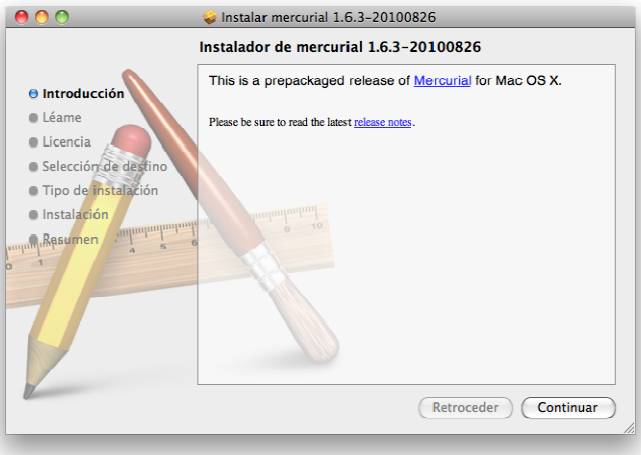
Requirements

- Python 2.4 (get python)

Another OS?

Get mercurial for:

- Mac OS X
- Windows
- other

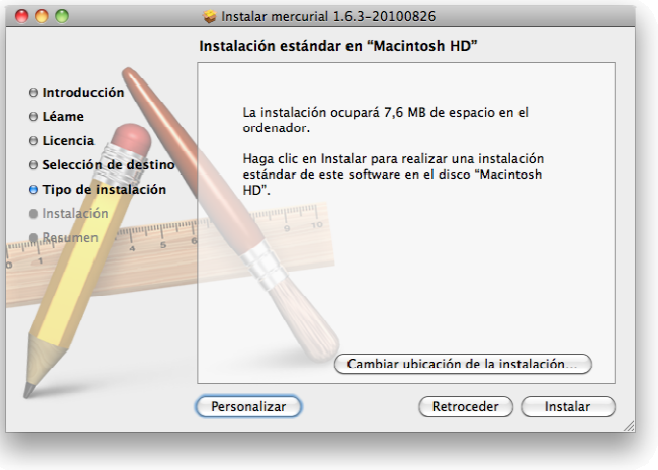


Últimas ofertas de empleo

2010-08-30
Otras - Electricidad - BARCELONA.

2010-08-24
Otras Sin catalogar - LUGO.

2010-06-25
T. Información - Analista / Programador - BARCELONA.



Finalizada la instalación de Mercurial debemos comprobar que se ha hecho correctamente. Abrimos un terminal y ejecutamos el siguiente comando: "hg -version". Si todo ha ido bien deberíamos ver algo como esto.

```
01. Mercurial Distributed SCM (version 1.6.3+20100826)
02.
03. Copyright (C) 2005-
    2010 Matt Mackall <mpm@selenic.com> and others
04. This is free software; see the source for copying conditi
05. warranty; not even for MERCHANTABILITY or FITNESS FOR A F
```

4. Configuración de Mercurial

Lo primero que debemos hacer despues de instalar Mercurial es configurarlo. Este SCM permite tener varias rutas donde depositar el fichero de configuración. Su orden de lectura será siempre el mismo, lo que permite poder sobreescribir cierta configuración definida en los ficheros. A continuación os muestro el orden de lectura y sus paths de cada fichero de configuración:

/.hg/hgrc
$HOME/.hgrc
/etc/mercurial/hgrc
/etc/mercurial/hgrc.d/*.rc
/etc/mercurial/hgrc
/etc/mercurial/hgrc.d/*.rc

Esta información se encuentra disponible ejecutando "hg help config" desde la línea de comandos.

Para una mayor claridad en su configuración, Mercurial ha agrupado las propiedades en secciones. Cada sección permite configurar una parte específica. En nuestro caso sólo añadiremos la configuración de nuestro usuario para que no tengamos que añadirlo cada vez que hagamos commit de nuestro ficheros.

Creamos un fichero .hgrc en nuestro home de usuario con la siguiente información:

```
01. [ui]
02. username=Angel Garcia <angel.garcia@autentia.com>
```

Si alguien quiere conocer más sobre la configuración de Mercurial, tenéis todas las secciones y propiedades aquí.

5 Empezar a trabajar con Mercurial

5.1 Ayuda integrada

Mercurial dispone de información de ayuda muy útil en aquellas situaciones en las que no recordemos en detalle el uso de un comando. Es muy sencilla de utilizar:

```
hg help add
hg help commit
hg help config
hg help init
```

5.2 Trabajando con repositorios Mercurial

5.2.1 Creación de un repositorio Mercurial

La creación de un repositorio Mercurial es realmente sencilla, nos situamos en el directorio donde deseamos crear el repositorio y ejecutamos el siguiente comando:

```
hg init
```

!!Enhorabuena!!! has creado tu primer repositorio Mercurial. Como decimos, es realmente sencillo. Si echamos un vistazo al directorio, veremos que en los ficheros ocultos se ha creado un directorio llamado .hg. Este directorio es utilizado por Mercurial para gestionar el repositorio.

5.2.2 Clonación de un repositorio Mercurial

En el caso de que ya exista un repositorio y queramos trabajar con él, tendremos que descargarnoslo. En Mercurial esta operación se denomina "clonar", ya que lo que realmente se esta haciendo es realizar una copia exacta del repositorio remoto. Para ello ejecutamos:

```
hg clone http://hg.mozilla.org/mobile-browser/
```

Mercurial permite utilizar varios protocolos para acceder a los repositorios: http, https, ssh, file y local, pero los más utilizados son https y ssh.

5.3 Trabajo habitual con el repositorio local.

Como en cualquier otro repositorio de control de versiones podemos realizar las operaciones más habituales como: añadir un fichero, realizar commits, ver diferencias entre versiones etc... En este apartado veremos los más usados para trabajar con el repositorio local que hayamos clonado.

5.3.1 Añadir un fichero al repositorio

Si hemos añadido un nuevo fichero al repositorio, para que áste se encuentre bajo el control del repositorio se deberá ejecutar el comando:

```
hg add file1 file2 file2
```

Si no se establece ningún fichero, Mercurial incorporará al repositorio todos los ficheros nuevos añadidos hasta el momento.

5.3.2 Conocer el estado de los ficheros

Otra comando muy habitual es el que permite conocer el estado de los ficheros en el repositorio: que ficheros son nuevos, cuales se han modificado, etc... En este caso tendremos que hacer:

```
hg status
```

Si existe alguna alteración del repositorio local, mostraría un salida como esta:

```
M xx.txt
? pp.txt
```

5.3.3 Ver el historial de cambios

Para poder echar un vistazo al historio de cambios en el repositorio tendremos que hacer:

```
hg log
```

Por defecto este comando imprime un párrafo por cada cambio que haya sido grabado. Cada uno de estos eventos es llamado changeset y puede contener un registro de cambios a varios ficheros. Cada changeset tendrá: un identificador, el usuario que lo ha realizado, la fecha y un comentario que en su momento introdujo. A continuación tenéis un ejemplo:

```
changeset:   2:1be7b831a0a2
tag:         tip
user:        agarcia
date:        Thu Sep 09 10:46:04 2010 +0200
summary:     dfafssd

changeset:   1:a8646fd88d99
user:        agarcia
date:        Thu Sep 09 10:41:30 2010 +0200
summary:     dfsisdfsa

changeset:   0:d1bcf36d2c6a
user:        agarcia
date:        Tue Sep 07 23:28:17 2010 +0200
summary:     test hudson with mercurial
```

5.3.4 Conocer las diferencias entre ficheros

Para conocer los cambios que se han realizado sobre un fichero debemos utilizar el comando "diff". Si no se le indica ningún fichero, nos mostrará las diferencias de todos los ficheros modificados del repositorio.

```
hg diff
```

Como podéis ver, Mercurial nos mostraría algo como:

Por defecto este comando imprime un párrafo por cada cambio que haya sido grabado. Cada uno de estos eventos es llamado changeset y puede contener un registro de cambios a varios ficheros. Cada changeset tendrá: un identificador, el usuario que lo ha realizado, la fecha y un comentario que en su momento introdujo. A continuación tenéis un ejemplo:

```
diff -r 1be7b831a0a2 pp.txt
--- /dev/null   Thu Jan 01 00:00:00 1970 +0000
+++ b/pp.txt    Wed Sep 15 12:57:55 2010 +0200
@@ -0,0 +1,1 @@
+fdjasdfksjds dfs
diff -r 1be7b831a0a2 xx.txt
--- a/xx.txt    Thu Sep 09 10:46:04 2010 +0200
+++ b/xx.txt    Wed Sep 15 12:57:55 2010 +0200
@@ -0,0 +1,1 @@
+sdffsff
```

5.3.5 Subir los cambios al repositorio local

Una vez que se hayan añadido o modificado los ficheros, si queremos hacer commit para subirlos al repositorio local haremos:

```
hg commit
```

5.4 Propagando los cambios a otro repositorio

La filosofía de trabajo de los repositorios distribuidos cambia respecto a los repositorios tradicionales, no tanto cuando se trabaja con nuestro repositorio local clonado (disponemos de los mismas operaciones), sino a la hora de propagar cambios a otros repositorios. Las posibilidades de trabajo y diferentes configuraciones son casi infinitas.

Al tener cada desarrollador un repositorio clonado que se comporta como un cliente y servidor, los desarrolladores pueden subir y actualizarse con los cambios de otros repositorios que no sean el repositorio original desde el que se realizó la clonación.

Por ejemplo nos permite situaciones como la de que dos desarrolladores estuvieran trabajando en una funcionalidad de una aplicación y pudieran intercambiarse sus modificaciones sin necesidad de subirlos al repositorio original. Sólo cuando la funcionalidad estuviera finalizada se subirían al repositorio maestro.

Para permitir este modo de trabajo, los repositorios distribuidos incorporan nuevos comandos exclusivos para trabajar entre repositorios. En los siguientes apartados veremos algunos de ellos.

5.4.1 Sincronización con otro repositorio

Cuando queramos incorporar algún cambio realizado en otro repositorio al nuestro, debernos hacerlo así:

```
hg pull
```

El repositorio por defecto que utilizará para actualizar los ficheros, será el que utilizamos con el comando clone. Si lo que queremos es actualizarnos con los cambios de otro repositorio deberemos añadir la url.

```
hg pull https://site.com/examplehg
```

También podemos saber que ficheros van a ser actualizados por el comando pull. Para ello tenemos que hacer:

```
hg incoming
```

De igual forma que con el comando pull, si queremos hacer incoming de otro repositorio diferente al que utilizamos para clonar deberíamos añadir la url.

```
hg incoming https://site.com/examplehg
```

En este punto debemos comentaros una cosa que hemos pasado por alto: la relación entre un repositorio y su directorio de trabajo. "hg pull" sólo se encarga de actualizar los cambios en el repositorio, pero si echamos un vistazo no hay ningún rastro de ese cambio en nuestro directorio de trabajo. Eso es debido a que el comando pull no modifica el directorio de trabajo por defecto. Para poder actualizar nuestro directorio debemos ejecutar;

```
hg update
```

5.4.2 Subir los cambios a otro repositorio

Para subir los cambios que hayamos realizados sobre nuestro repositorio a otro, debemos hacer lo siguiente:

```
hg push
```

Si lo que queremos es revisar que cambios van a ser subidos ejecutaremos esto:

```
hg outgoing
```

Si os dais cuenta son los comando complementarios de pull e incoming.

6 Conclusión

En este tutorial hemos visto un pequeña parte del modo de trabajo con los repositorios distribuidos y en concreto con Mercurial. Os recomiendo que accedáis a la wiki de Mercurial y echáis un vistazo para conocer en detalle muchas más características del producto, o esperar a próximos tutoriales en los que veremos muchas más cosas.

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS

Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2010 © All Rights Reserved | Texto Legal y condiciones