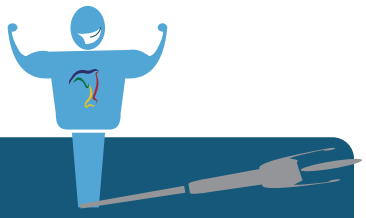
autentia

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...

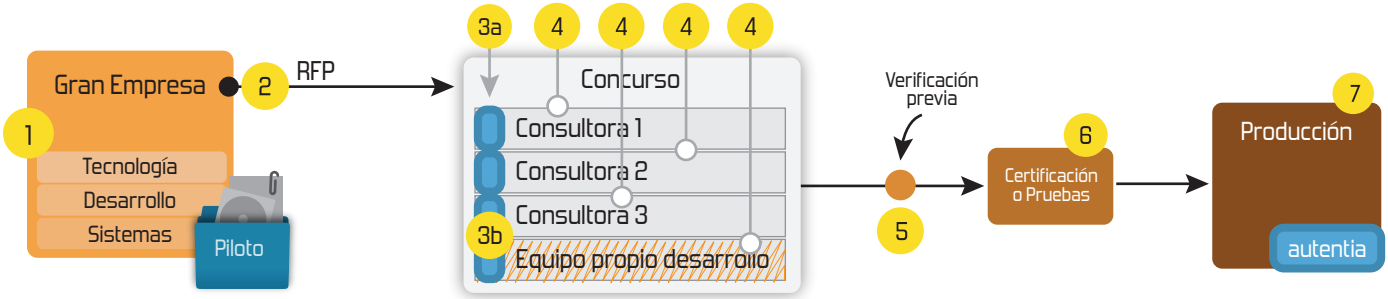
1. Desarrollo de componentes y proyectos a medida



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

Ultimas Noticias

» Disponible la primera versión de los plugins para integrar Maven y Bugzilla.
» Comentando el libro: Guía de Estilo, Protocolo y Etiqueta en la empresa.
» Comentando el libro: "El viaje a la felicidad. Nuevas claves científicas" de Eduardo Punset
» Lanzamiento del nuevo Web de Autentia
» Si se pregunta ¿Qué ofrece este Web?
» Historia de la Informática. Capítulo 73. 1996 (2ª Parte)
» Autentia cumple 6 años
» Autentia colabora en la difusión de las metodologías ágiles en español.
» Alimarket.es: Primera aplicación pública del framework wuija by Autentia

+Noticias Destacadas

» Lanzamiento del nuevo Web de Autentia
» Contratos ágiles: Vendiendo Scrum a tus clientes.
» Quinta charla Autentia + Proyectalis + Agile Spain: Contratos ágiles: Vendiendo Scrum a tus clientes
» Lo mejor de esta semana: Curso de Scrum con Ángel Medinilla

+Comentarios Cómic

+Enlaces

Tutorial desarrollado por

Borja Lázaro de Rafael

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Catálogo de servicios de Autentia

Descargar (6,2 MB)

Descargar en versión comic (17 MB)

AdictosAlTrabajo.com es el Web de difusión de conocimiento de Autentia.

autentia real business solutions

Catálogo de cursos

Descargar este documento en formato PDF: releaseBugzilla.pdf

Fecha de creación del tutorial: 2009-09-11

Automatizar acciones en el Bugzilla al generar versiones con Maven

Índice de contenido

Introducción.

Cómo ya hemos visto varias veces, la creación de plugins de maven es bastante sencilla. Como recordatorio os remito a este tutorial "Desarrollo de Plugins para Maven" como referencia a la creación de plugins de Maven

Aprovechando la facilidad de crear nuestros propios plugins, en este tutorial vamos a mostrar como automatizar un conjunto de acciones que hay que hacer siempre en los sistemas de gestión de incidencias, tales como dar de alta una nueva versión del producto, cerrar las incidencias que soluciona la nueva versión, etc.

En este caso vamos a ver automatizar estas acciones en Bugzilla.

Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portatil Samsung R70 (Intel(R) Core(TM)2 Duo 2,5Ghz, 2046 MB RAM, 232 Gb HD)
- Sistema Operativo: Windows Vista Home Premium
- Máquina Virtual Java: JDK 1.5.0_14 de Sun Microsystems (http://java.sun.com/javase/downloads/index_jdk5.jsp
- IDE Eclipse 3.3 (Europa) (http://www.eclipse.org/downloads/)

Creación del plugin complementario.

Como hemos explicado en la introducción, la solución que hemos adoptado es crear un plugin de Maven para que se conecte a nuestro Bugzilla y realice todas las acciones de forma automática.

En primer lugar identificamos los parámeteros que necesita nuestro plugin para conectarse al Bugzilla y generar el fichero changes.xml:

- productName: Nombre del producto en el Bugzilla del que obtenemos la lista de bugs.
- bugzillaUser: Usuario para el login en el Bugzilla.
- bugzillaPassword: Contraseña del usuario para el login en el Bugzilla.
- loginPage: Página de login en el Bugzilla.
- loginRequired: Parámetro para indicar si es necesario hacer login en el Bugzilla para recuperar la lista de bugs. (por defecto true)
- parentOnly: Al tener los proyectos de Maven herencia, con este parámetros le indicamos que sólo genere la información de cambios en el proyecto padre.
- versionSuffix: El nombre de la versión es común que tenga un sufijo mientras se está desarrollando, por lo que al consultar en el Bugzilla se debe eliminar. (Por defecto "-SNAPSHOT").
- bugsToClose: Identifica el "status" de los bugs que deben ser cerrados al generar la versión. (Por defecto "RESOLVED, CLOSED").
- bugsToMove: Identifica el "status" de los bugs que deben ser movidos al nuevo "milestone". (Por defecto "UNCONFIRMED, NEW, ASSIGNED, REOPENED").

Además de estos parámetros hay que tener en cuenta al proyecto de Maven que también debe ser accesible desde nuestro plugin, por lo que en nuestro código debemos añadir:

Catálogo de servicios Autentia (PDF 6,2MB)

En formato comic...

Google

Web
www.adictosaltrabajo.com
Buscar

Últimos tutoriales

2009-09-11
Release Bugzilla Maven Plugin

2009-09-11
Enlazar Bugzilla con MavenChangesPlugin

2009-09-08
Sobre las reglas de codificación o... ¿de dónde salen esos caracteres "raros"?.

2009-08-28
Cómo hacer deploy del site de Maven en SourceForge

2009-08-26
Ordenación por cantidades en informe cruzado

2009-08-20
Selenium IDE-Incorporando while en los test

2009-08-14
Blender y JMonkeyEngine. Exportación de archivos Blender y uso de los mismos en JMonkeyEngine

2009-08-14
5º tutorial TNT Concept Versión 0.16.1 Gestión de informes, vacaciones y utilidades

2009-08-14
Joomla 1.5. Instalación y configuración

2009-08-13
Introducción a los diagramas EPC (Event-Driven Process Chain)

2009-08-10
Blender. Animaciones avanzadas y renderización

2009-08-10
Gestión de Calidad, tablón y seguimiento en TNT Concept Versión 0.16.1

2009-08-10
Cómo hacer una página web

2009-08-06
Tips And Tricks JUnit Spring

2009-08-03
Instalación de VirtualBox PUEL

2009-08-03
Gestión de contactos y pedidos en TNT Concept versión 0.16.1

2009-08-03
Comentando el libro: La estrategia del océano azul

2009-07-30
Funciones esenciales para crear un juego.

2009-07-30
2º tutorial TNT Concept versión 1.16.1

2009-07-29
Hibernate Search, Bridges, Analizadores y más

2009-07-24
Migración de EJB3 a JPA y Spring.

2009-07-20

view plain print ?

```
01.  /**
02.   * The Maven Project.
03.   *
04.   * @parameter expression="${project}"
05.   * @required
06.   * @readonly
07.   */
08.  private MavenProject project;
09.
10.  /**
11.   * Bugzilla product name.
12.   *
13.   * @parameter expression="${changesMavenPlugin.productName}"
14.   * @required
15.   */
16.  private String productName;
17.
18.  /**
19.   * Bugzilla user.
20.   *
21.   * @parameter expression="${changesMavenPlugin.bugzillaUser}"
22.   * @required
23.   */
24.  private String bugzillaUser;
25.
26.  /**
27.   * Bugzilla password for the user.
28.   *
29.   * @parameter expression="${changesMavenPlugin.bugzillaPassword}"
30.   * @required
31.   */
32.  private String bugzillaPassword;
33.
34.  /**
35.   * Bugzilla login page.
36.   *
37.   * @parameter expression="${changesMavenPlugin.loginPage}" default-value="index.cgi"
38.   */
39.  private String loginPage = "index.cgi";
40.
41.  /**
42.   * Bugzilla login required.
43.   *
44.   * @parameter expression="${changesMavenPlugin.loginRequired}" default-value="true"
45.   */
46.  private boolean loginRequired = true;
47.
48.  /**
49.   * Version name suffix that is removed from the Bugzilla Http request.
50.   *
51.   * @parameter expression="${changesMavenPlugin.versionSuffix}" default-value="-SNAPSHOT"
52.   */
53.  private String versionSuffix = "-SNAPSHOT";
54.
55.  /**
56.   * Release in Bugzilla only in parent project.
57.   *
58.   * @parameter expression="${changesMavenPlugin.parentOnly}" default-value="true"
59.   */
60.  private boolean parentOnly = true;
61.
62.  /**
63.   * Bugzila bug status values that the bug will be closed.
64.   *
65.   * @parameter expression="${changesMavenPlugin.bugsToClose}" default-value="RESOLVED, CLOSED"
66.   */
67.  private String bugsToClose = "RESOLVED, CLOSED";
68.
69.  /**
70.   * Bugzila bug status values the the bug will change the target milestone.
71.   *
72.   * @parameter expression="${changesMavenPlugin.bugsToMove}" default-value="UNCONFIRMED, NEW, ASSIGNED, REOPENED"
73.   */
74.  private String bugsToMove = "UNCONFIRMED, NEW, ASSIGNED, REOPENED';
75.
```

El proceso principal de nuestro plugin se realiza en el método "execute()" que necesariamente debe implementar al heredar de "org.apache.maven.plugin.AbstractMojo". Los pasos que sigue son:

- Preprocesamiento de parámetros: Para controlar si seguir con la ejecución y preparar los datos necesarios para el resto del proceso.
- Login en el Bugzilla.
- Crear la nueva versión en el Bugzilla.
- Cerrar los bugs que soluciona la nueva versión.
- Crear un nuevo "milestone" para la próxima versión.
- Cambiar el milestone de los bugs que no están solucionado al nuevo milestone creado.

El código del método "execute()" es:

Directorio de ejemplos de jMonkey Engine

2009-07-19
JSR-179 Location API para J2ME: Posicionamiento geográfico en nuestras aplicaciones.

2009-07-16
Gestión de Usuarios en TNT Concept versión 0.16.1

2009-07-16
Continuación del Tutorial: JMonkeyEngine, Creación de nuestro primer juego.

2009-07-16
Como implementar el Scene Monitor para analizar las escenas en jMonkeyEngine

2009-02-26
Transformaciones de escena en JMonkeyEngine

2009-07-15
Detalles del juego de la moto en jMonekyEngine.

2009-07-14
JMonkeyEngine, Creación de nuestro primer juego.

2009-07-13
Ajax tests con Selenium: prototype.js e ICEfaces.

2009-07-08
AOP con AspectJ y Maven

2009-07-07
Instalación y configuración de Eclipse Galileo

2009-07-07
Iniciarse en el manejo de JME, Creación de un Cloth.

2009-07-06
Primeros pasos con Blender: Pintando nuestra mascota en 3D

2009-07-06
DBUnit-Exportar e Importar BBDD

2009-07-05
JMeter, Pruebas de stress sobre aplicaciones web: Grabando y reproduciendo navegaciones

2009-07-02
Axis2: Invocación de Servicios Web usando distintos MEP

2009-07-02
Instalación OpenOffice

2009-07-02
Juegos 3D en Java: Blender y JMonkeyEngine

2009-06-20
StAX (Xml Pull Parser): Streaming API para XML

2009-06-15
Configuración de la desconexión de usuarios con ICEFaces

2009-06-10
LWUIT: Una librería gráfica tipo AWT o Swing para J2ME

2009-06-10
Mapas mentales con XMind

2009-02-26
Redimensionar Imagenes en Windows Vista

2009-06-08
UploadFile con Icefaces + Hibernate + Anotaciones

2009-06-05
Habilitar exportación en Liferay

2009-06-01
Registrar Liferay en Eclipse

2009-05-29
Liferay Social Office

2009-05-28
Broadcast con Ustream

2009-05-25
Tabla datos accesible con ordenación y

```
view plain print ?
01. /*
02.  * (non-Javadoc)
03.  * @see org.apache.maven.plugin.AbstractMojo#execute()
04.  */
05. public void execute() throws MojoExecutionException {
06.     getLog().debug("Entering...");
07. 
08.     if (productName == null) {
09.         // use project name if is null
10.         productName = project.getName();
11.     }
12. 
13.     // check if the project is a parent project
14.     if (parentOnly && (project.getParent() != null)) {
15.         return;
16.     }
17. 
18.     // gets version name from project
19.     versionName = project.getVersion();
20.     final int index = versionName.indexOf(versionSuffix);
21.     if (index != -1) {
22.         // removing version suffix
23.         versionName = versionName.substring(0, index);
24.     }
25.     // evaluate next milestone(version)
26.     newMilestone = getNextMilestone();
27.     // ask for version names
28.     try {
29.         versionName = prompter.prompt("What is the product release Bugzilla version name? , versionName);
30.     } catch (final PrompterException e) {
31.         throw new MojoExecutionException("Could not get new version name.', e);
32.     }
33.     try {
34.         newMilestone = prompter.prompt("What is the product new development Bugzilla milestone? , newMilestone);
35.     } catch (final PrompterException e) {
36.         throw new MojoExecutionException("Could not get milestone name.', e);
37.     }
38. 
39.     // Bugzilla url from issueManagement
40.     bugzillaURL = project.getIssueManagement().getUrl();
41. 
42.     final WebConversation wc = new WebConversation();
43. 
44.     // perform Login
45.     boolean loginSuccess = true;
46.     if (loginRequired) {
47.         loginSuccess = login(wc);
48.     }
49. 
50.     if (loginSuccess) {
51.         // create the new version in Bugzilla
52.         createVersion(wc);
53.         // close resolved bugs
54.         closeBugs(wc);
55.         // create the new milestone
56.         createMilestone(wc);
57.         // change milestone for unresolved bugs
58.         changeMilestone(wc);
59. 
60.     } else {
61.         throw new MojoExecutionException("The username or password you entered is not valid.");
62.     }
63.     getLog().debug("Exiting...");
64. }
65. 
```

Como se puede ver en el código, después de un procesamiento previo de los parámetros se empieza con las peticiones al Bugzilla; hacemos el login y simulamos la navegación por las páginas del Bugzilla para ir ejecutando las acciones que estamos automatizando.

Esta navegación la realizamos con la librería "httpunit"; por lo que añadimos su dependencia en el fichero pom.xml:

```
view plain print ?
01. <dependency>
02.     <groupId>httpunit</groupId>
03.     <artifactId>httpunit</artifactId>
04.     <version>1.6.2</version>
05. </dependency>
06. 
```

A modo de ejemplo, ponemos parte del código en el que se muestra como simulamos la navegación utilizando httpunit. Primero hay que crear una conversación, que viene a ser como una sesión con el servidor:

```
view plain print ?
01. .....
02. final WebConversation wc = new WebConversation();
03. .....
04. 
```

y después realizamos la serie de peticiones ... (en este caso el login):

```
view plain print ?
01. .....
02. final WebRequest req = new GetMethodWebRequest(bugzillaURL + loginPage);
03. WebResponse resp;
04. resp = wc.getResponse(req);
05. final WebForm webForm = resp.getFormWithName("login");
06. .....
07. webForm.setParameter("Bugzilla_login", bugzillaUser);
08. webForm.setParameter("Bugzilla_password", bugzillaPassword);
09. webForm.submit();
10. resp = wc.getCurrentPage();
11. .....
12. 
```

De esta forma, siempre y cuando utilicemos la misma conversación, podemos seguir haciendo peticiones al servidor bajo la misma sesión. Así pues, el resto de acciones en el Bugzilla lo realizaremos con la misma conversación (sesión). Se irán solicitando formularios y enviándose, de forma que se llevarán a cabo todas las acciones previstas.

Ahora sólo queda hacer que nuestro plugin pueda ser utilizado desde otros proyectos instalándolo en nuestro repositorio local ejecutando:

```
mvn clean install
```

Utilizando nuestro plugin.

Éste plugin lo deberemos ejecutar cuando generemos una "release" con Maven. Por lo que el plugin debe estar incluido en los proyectos en su fichero "pom.xml" de la siguiente forma.:

```
view plain  print  ?
01. <project ...>
02.   .....
03.   <!-- Lo utiliza el plugin para obtener la URL del Bugzilla--
04.   <issueManagement>
05.     <system>Bugzilla</system>
06.     <url>https://host/cgi-bin/bugzilla3/</url>
07.   </issueManagement>
08.   .....
09.   <build>
10.         <plugins>
11.       .....
12.             <plugin>
13.               <!-- nuestro plugin de generación del changes.xml --
14.                 <groupId>com.autentia.mvn.plugir</groupId>
15.                 <artifactId>releaseBugzilla</artifactId>
16.                 <version>1.0-SNAPSHOT</version>
17.                 <configuration>
18.                     <productName> *** nombre del producto en el bugzilla ***</productName>
19.                     <bugzillaUser>${bugzillaUser}</bugzillaUser>
20.                     <bugzillaPassword>${bugzillaPassword}</bugzillaPassword>
21.                 </configuration>
22.             </plugin>
23.             .....
24.         </plugins>
25.   </build>
26.   .....
27.   <!-- Repositorio para descargar el plugin --
28.   <pluginRepositories>
29.     <pluginRepository>
30.         <id>AutentiaBugzillaMaven-release</id>
31.         <name>Local Maven repository of release</name>
32.         <url>http://bugzillachanges.sourceforge.net/maven-repositor</url>
33.         <snapshots>
34.             <enabled>false</enabled>
35.         </snapshots>
36.         <releases>
37.             <enabled>true</enabled>
38.         </releases>
39.       </pluginRepository>
40.   </pluginRepositories>
41. </project>
42.
```

De los parámetros de configuración de nuestro plugin, se pueden considerar un poco distintos los parámetros "bugzillaUser" y "bugzillaPassword", ya que es algo particular de cada usuario. Al ser el "pom.xml" un fichero que se suele compartir, no es aconsejable que aparezcan estos datos; por lo que nos valemos de las ventajas de Maven para recuperarlos del perfil activo en el fichero "settings.xml"; de esta forma en nuestro fichero "setting.xml" deberá aparecer:

```
view plain  print  ?
01. <settings>
02.     <profiles>
03.       .....
04.       <profile>
05.          <id>identificador del perfil activ</id>
06.          .....
07.          <properties>
08.               <bugzillaUser>user</bugzillaUser>
09.               <bugzillaPassword>password</bugzillaPassword>
10.           </properties>
11.       </profile>
12.       .....
13.     </profiles>
14.     .....
15. </settings>
16.
```

Finalmente sólo queda que nuestro plugin se ejecute. Para eso añadimo el "goal" específico a la línea de comandos para que el contexto de Maven llame a la ejecución de nuestro plugin:

```
mvn com.autentia.mvn.plugin:releaseBugzilla:release
```

Línea de comandos

Como no todo es Maven en esta vida, ni Java, y podemos tener en Bugzilla la gestión de otros proyectos, hemos añadido la posibilidad de ejecutar este plugin desde la línea de comandos pasándole los parámetros necesarios.

```
java -jar releaseBugzilla-XX.jar -help

Nota:El nombre del fichero jar debe ser el correspondiente a la versión del plugin.

Uso: java -jar releaseBugzila-XX.jar args
        args:
                -productname: Nombre del producto en el bugzilla. Este parámetro es obligatorio.
                -url: URL del bugzilla, debe terminar con \"/\". (eg. https://host/cgi-bin/bugzilla3/). Este parámetro es obligatorio.
                -version: Nombre de la versión actual.
                -suffix: Sufijo del nombre de la versión que será eliminado. Por defecto \"-SNAPSHOT\"".
                -user: Usuario del Bugzilla.
                -password: Contraseña del usuario en Bugzilla.
                -loginpage: Página de login en el Bugzilla. (eg. index.cgi). Por defecto \"index.cgi\".
                -loginrequired: true o false, indica si es obligatorio el login en el Bugzilla. Por defecto \"true\".
                -bugs2close: Estados de los bugs que van a ser cerrados. Para más de un estado separarlos por espacios. Por defcto \"RESOLVED VERIFIED\'
                -bugs2move: Estado de los bugs en los que se va a cambiar el "target milestone"al nuevo "milestone". Para más de un estado separarlos p
```

Conclusiones

Como hemos visto varias veces, el desarrollo de plugins de Maven nos sirve para completar nuestras necesidades. En este caso, al tener la generación de versiones con Maven, parece lógico que aquellas acciones que podamos ir automatizando las hagamos en un plugin propio de Maven, que se ejecutará al generar una nueva versión.

Si queréis todo el código fuente de este plugin lo podéis conseguir en sourceforge.

Documentación extra la podéis encontrar aquí.

Un saludo.
Borja Lázaro de Rafael.

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.

Nombre: E-Mail:

Comentario:

Enviar comentario **Texto Legal y condiciones de uso**

- Puedes inscribirte en nuestro servicio de notificaciones haciendo clic aquí.
- Puedes firmar en nuestro libro de visitas haciendo clic aquí.
- Puedes asociarte al grupo AdictosAlTrabajo en XING haciendo clic aquí.
- Añadir a favoritos Technorati.

Recuerda

Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

Gestión de contenidos

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Enlazar Bugzilla con MavenChangesPlugin	En este tutorial veremos como enlazar Bugzilla con MavenChangesPlugin	2009-09-11	2	-	-	
Sobre las reglas de codificación o... ¿de dónde salen esos caracteres "raros"?.	En este tutorial vamos a tratar de dar algo de luz a los errores que tenemos habitualmente con la codificación de caracteres en aplicaciones en las que se ven implicados varios sistemas que intercambian o almacenan información.	2009-09-08	211	Muy bueno	3	
Cómo hacer deploy del site de Maven en SourceForge	Este tutorial nos enseña un poco mas sobre Maven	2009-08-28	303	Muy bueno	1	
Ordenación por cantidades en informe cruzado	Nico nos explica en ese tutorial cómo lograr ordenar por cantidades en informes cruzados usando JasperReports e iReport	2009-08-26	372	Muy bueno	4	
AOP con AspectJ y Maven	Programacion orientada a aspectos con AspectJ y Maven	2009-07-08	1005	Bueno	2	
DBUnit-Exportar e Importar BBDD	DBUnit como complemento de los test unitarios con carga a una base de datos	2009-07-06	1261	Muy bueno	6	
JMeter, Pruebas de stress sobre aplicaciones web: Grabando y reproduciendo navegaciones	En este tutorial Carlos García nos enseñará a grabar y reproducir navegaciones con JMeter, para poder realizar pruebas de carga o stress sobre aplicaciones Web	2009-07-05	2216	Bueno	10	
Configuración de la desconexión de usuarios con ICEFaces	Este tutorial muestra la manera de configurar y traducir la ventana de desconexión o pérdida de sesión del usuario en ICEFaces.	2009-06-15	2045	Muy bueno	11	
Tabla datos accesible con ordenación y paginación	En este tutorial vamos a ver como podemos hacer una tabla de datos con ordenación y paginación aplicando los criterios de accesibilidad y los conceptos de mejora progresiva y Javascript no obstrusivo.	2009-05-25	3343	Bueno	15	
Plugin Hibernate3 para Maven	En este tutorial veremos las posibilidades que nos ofrece el plugin de Hibernate3 para Maven, como por ejemplo, la generación del esquema de base de datos desde clases con anotaciones.	2009-05-02	1507	Bueno	8	

Nota: