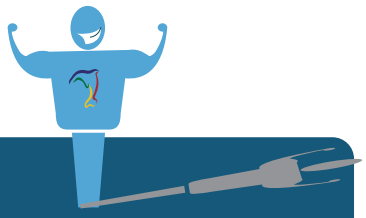
autentia

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...

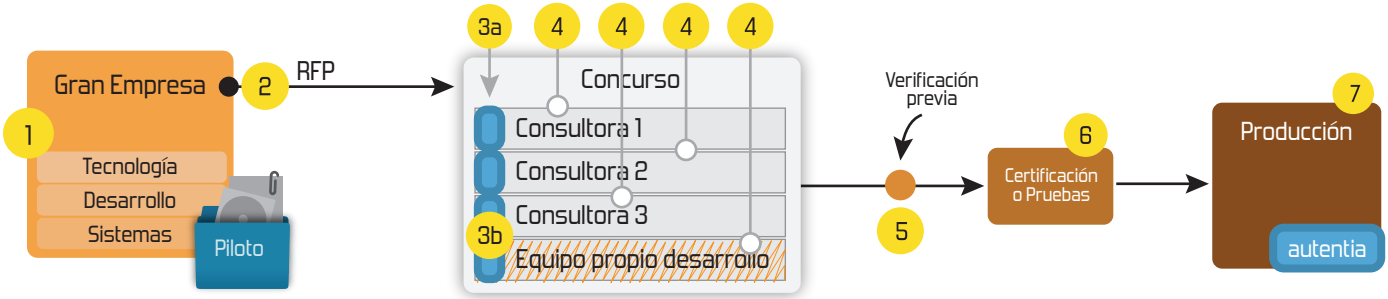
1. Desarrollo de componentes y proyectos a medida



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

Contacte con Cristóbal González
criskerberos-tutoriales@yahoo.com

Autor: Cristóbal González Almirón es consultor de desarrollo de proyectos informáticos.

Su experiencia profesional se ha desarrollado en empresas como Compaq, HP, Mapfre, Endesa, Repsol, Universidad Autónoma de Madrid, en las áreas de Desarrollo de Software (Orientado a Objetos), tecnologías de Internet, Técnica de Sistemas de alta disponibilidad y formación a usuarios.

Descargar este documento en formato PDF pruebasjwebunit.pdf

Fecha de creación del tutorial: 2006-11-17

Pruebas unitarias con jwebunit

Introducción

En el tutorial anterior sobre pruebas unitarias web, vimos varios frameworks (marcos de trabajo) de pruebas unitarias de tipo Open Source para el mundo Java. En este tutorial nos vamos a aproximar al framework jWebUnit, que es un proyecto muy interesante para realizar rápidamente una buena batería de pruebas unitarias para nuestra aplicación web.

Vamos a ver cómo podemos realizar pruebas unitarias con jwebunits. Este framework es una extensión de los frameworks junit y htmlunit, especialmente indicado para probar la interfaz de la aplicación. Con este framework, nuestras pruebas podrán interactuar con la aplicación web, simulando las diferentes acciones del usuario, y revisando

la interfaz que genera la aplicación en cada momento.

Instalación de jwebunit

Nos vamos a la página web del proyecto http://jwebunit.sourceforge.net y descargamos el zip que contiene el paquete jwebunits. Yo me he descargado la versión jwebunit-1.3.zip

Como en todos los proyectos de Sourceforge.net, en la sección de descargas encontraremos los ficheros. Una vez descargado el paquete lo descomprimimos y veremos lo siguiente:

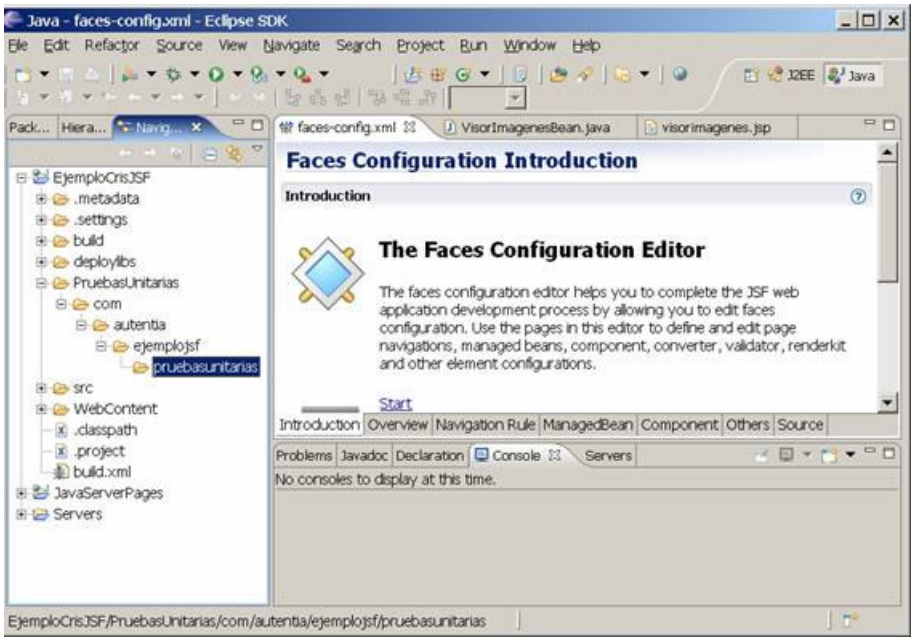
- Dos archivos jar, el que contiene jwebunit (jwebunit-core-1.3.jar) y el que contiene el plugin de htmlunit (jwebunit-htmlunit-plugin-1.3.jar)
- Una carpeta lib con todos los jar de los que depende.
- Los jar con los fuentes en la carpeta src
- El javadoc de jwebunit en la carpeta javadoc
- Los ficheros con las licencias

Normalmente jwebunit lo usaremos desde una herramienta de compilación automática, como puede ser ANT, Maven o similares, o desde un entorno de desarrollo, como puede ser Eclipse. En el caso de usar Maven, que es el entono de compilación usado por los desarrolladores de jwebunit, en la página del proyecto nos dan las sencillas instrucciones para incluir jwebunit en nuestros proyectos.

Para añadirlo manualmente a nuestros proyectos, incluiremos los jar en el classpath de nuestra herramienta de compilación o de nuestro IDE y crearemos los test unitarios de un modo similar a como se hace con junit.

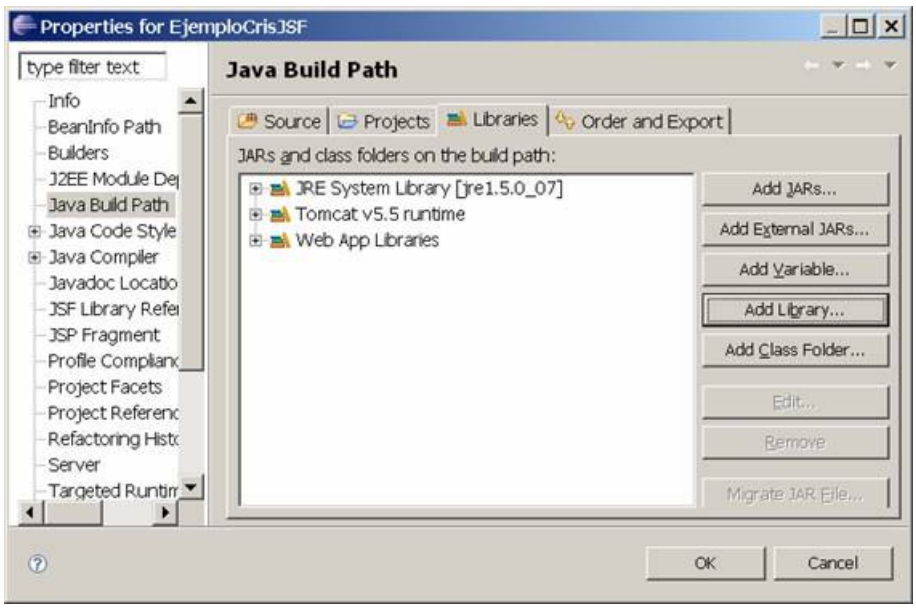
Ejemplo en Eclipse sobre una aplicación JSF

Creamos una carpeta para las pruebas unitarias, por ejemplo “PruebasUnitarias” y dentro de ella creamos un paquete, en nuestro caso com.autentia.ejemplojsf.pruebasunitarias:



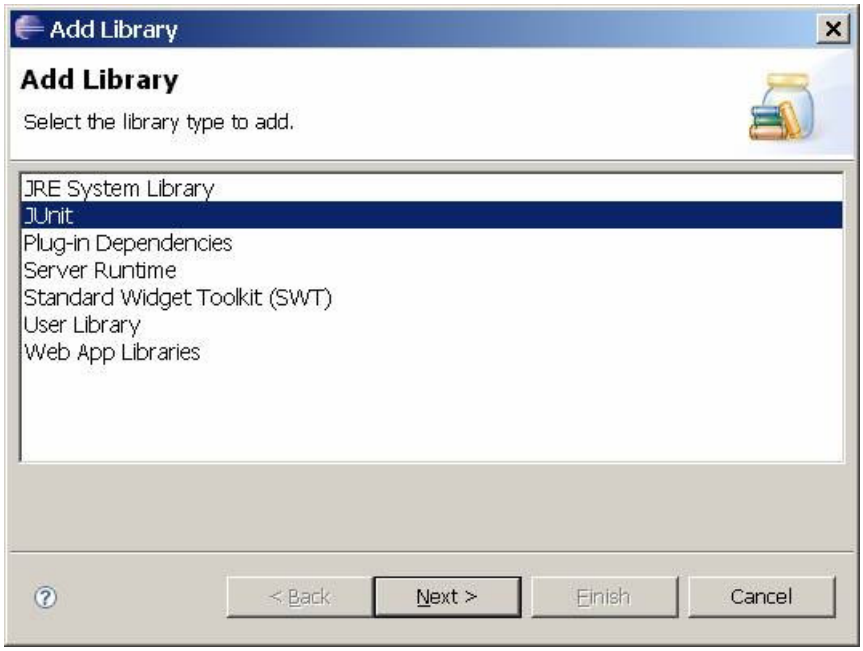
Ahora añadimos una prueba (test case, en terminología junit)

Para que podamos ejecutar los test hay que añadir al Java Build Path (el classpath del proyecto) las bibliotecas jar de jwebunit.

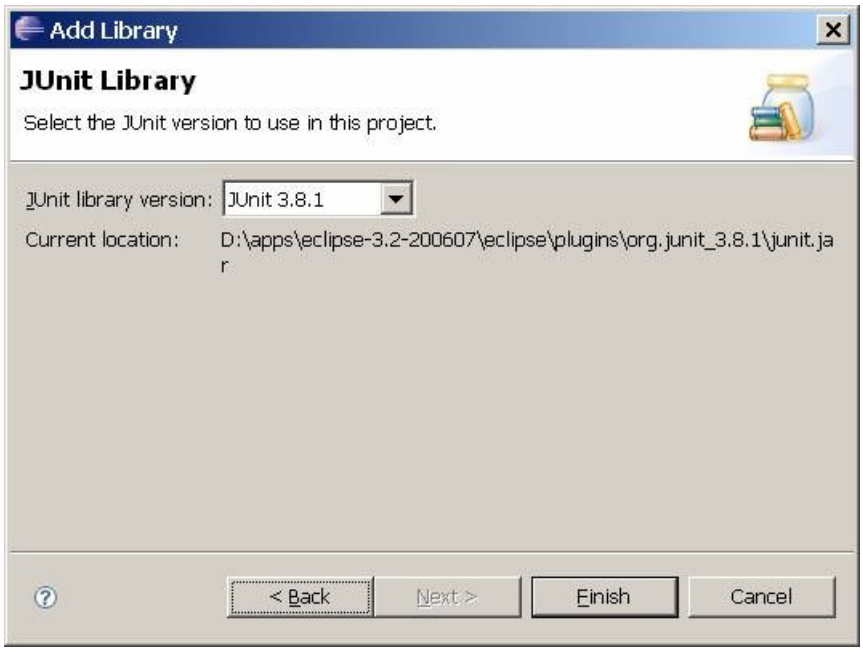


Por defecto nuestro proyecto no dispone del entorno junit ni de jwebunit

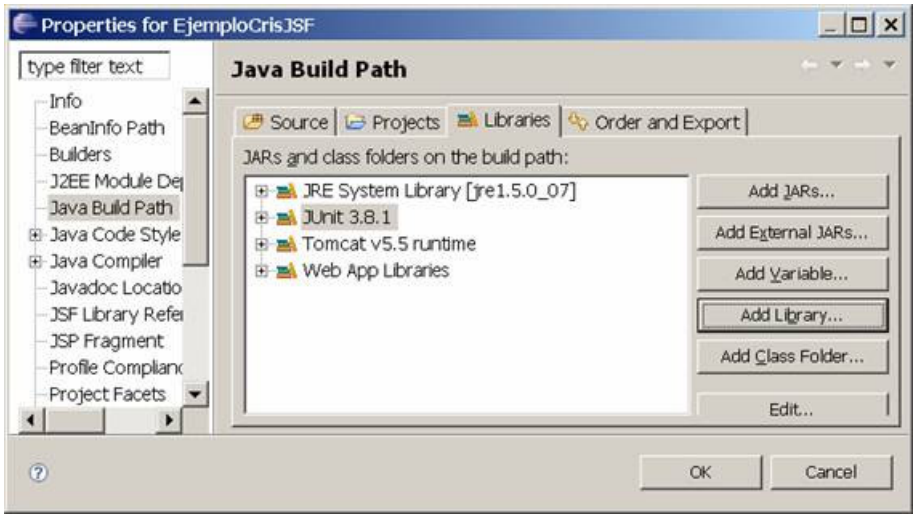
Ahora añadimos el entorno junit. Para ello pulsamos en “Add Libraries”



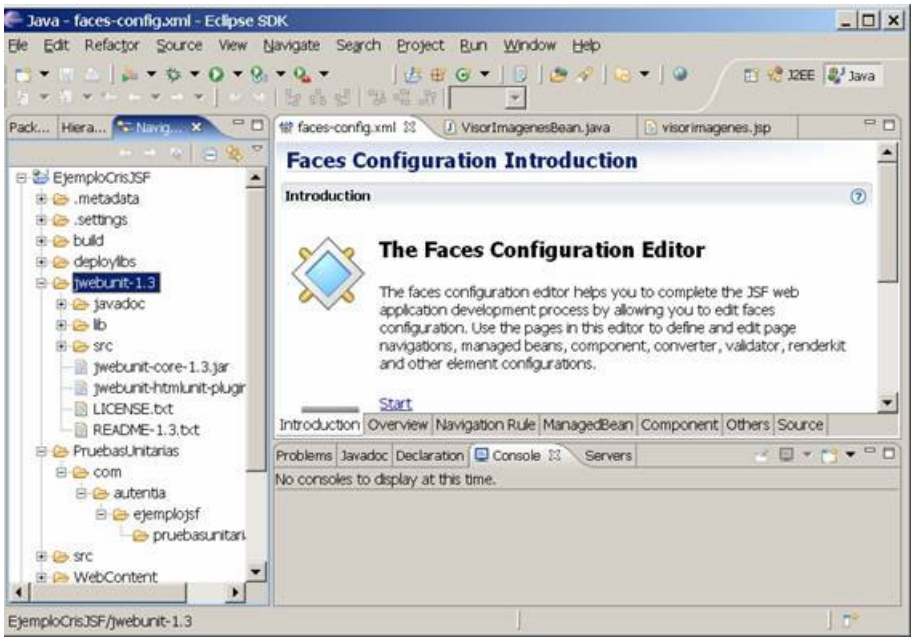
Seleccionamos “JUnit” y pulsamos “Next”



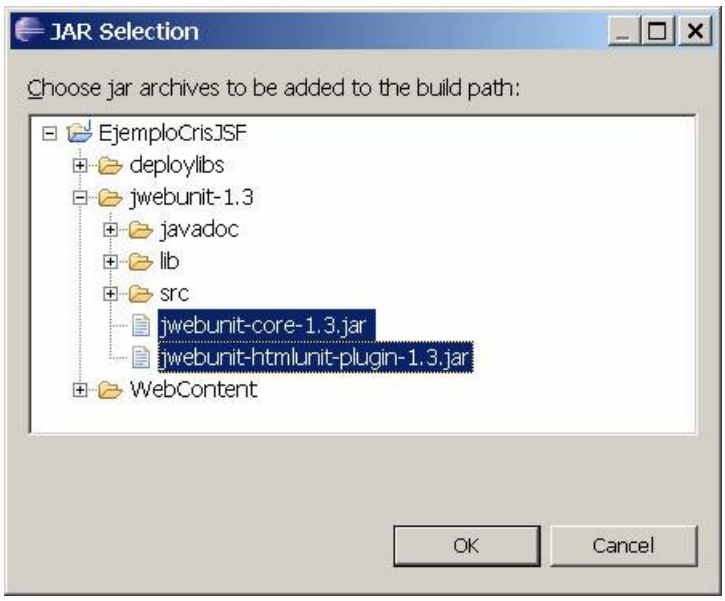
Seleccionamos la versión ofrecida por defecto y pulsamos “Finish”.



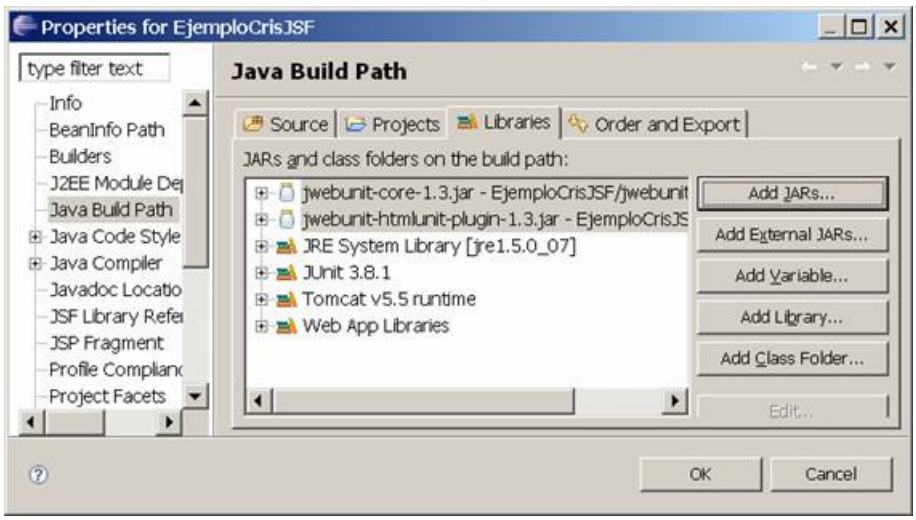
Ahora añadimos las bibliotecas de jWebUnit. Para ello copiamos todos los ficheros de jwebunit a una carpeta del proyecto. Si queremos borramos los que no nos hagan falta.



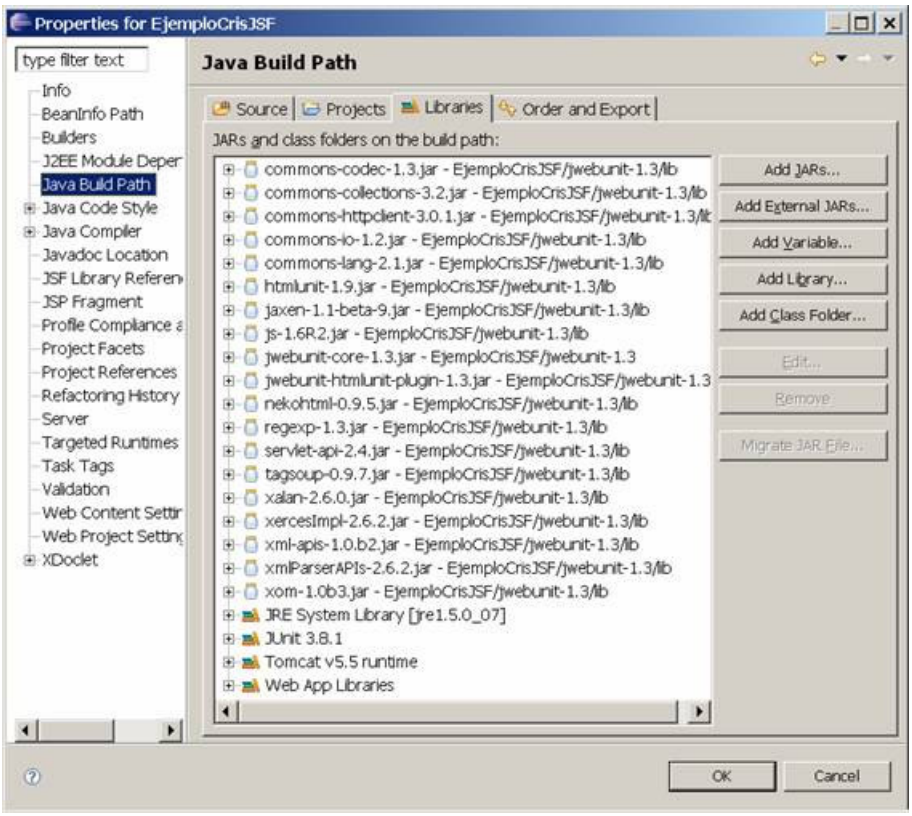
Volvemos a abrir el diálogo de “Java Build Path” como hemos hecho antes y pulsamos “Add Jars…”



Elegimos los jar de jWebUnit y pulsamos OK.



Ahora nos faltan las dependencias de jwebunit, que son aquellas que necesitemos de la carpeta lib del jwebunit, por ejemplo regexp-1.3.jar



Creación de un test unitario web con jwebunit

Vamos a probar una aplicación de prueba JSF que realiza lo siguiente:

- Al abrir la página de inicio.faces se muestra la página de inicio de la aplicación

- Tras introducir el usuario y la contraseña, se pulsa en el botón “Iniciar”, si la contraseña es correcta, nos lleva a la página de inicio principal. Si es incorrecta nos lleva a una página de error. Hemos puesto de contraseña “sesamo”. Si la contraseña no es correcta, nos muestra de nuevo el formulario y en el texto de usuario escribe “Reintentar”

El código de la página de inicio es:

Inicio.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Insert title here</title>
</head>
<body>
<f:view>
<h:form id="inicio">
   Usuario:
      <h:inputText id="username" value="#{userLoginBean.username}" />
      <br/>
      Contraseña:
      <h:inputSecret id="password" value="#{userLoginBean.password}" />
      <br/>
      <h:inputText value="#{userLoginBean.dominio}" disabled="#
{userLoginBean.verDominioDeshabilitado}" />
      <h:commandButton id="iniciar" actionListener="#
{userLoginBean.habilitarDominio}"
                  immediate="true" value="usar dominio..."/>
            <br>
      <h:commandButton id="submit" action="#
{userLoginBean.validarUsuario}" value="Iniciar" />

</h:form>
</f:view>

</body>
</html>
```

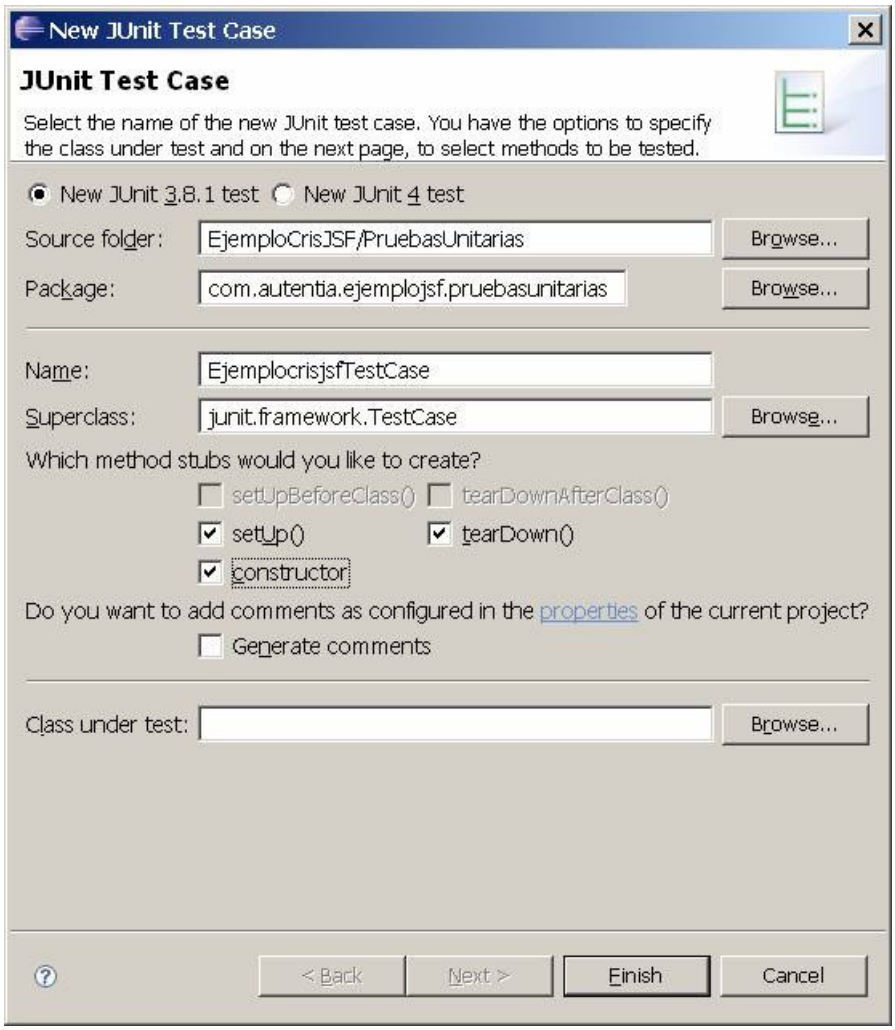
El código de la página principal es:

Principal.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Página principal</title>
</head>
<body>
<h2>Aplicación de ejemplo JSF </h2><br>
<h3>Cristóbal González Almirón</h3>
Bienvenido, <c:out value="${userLoginBean.username}" />
<f:view>
<h:form>
      <h:commandButton action="buscar"
value="Buscar"></h:commandButton>
      <h:commandButton action="mostrar"
value="Mostrar"></h:commandButton>

      <br>
      Categoria:
      <h:selectOneMenu value="#{formularioBusquedaBean.categoria}">
            <f:selectItems value="#
{formularioBusquedaBean.categorias}" />
      </h:selectOneMenu>
      <br>
      Este es el texto del formulario de búsqueda

</h:form>
</f:view>

</body>
</html>
```

Como vemos es un código muy simple, y nos basta para ver el jwebunit en funcionamiento.

Ahora creamos con el asistente de Eclipse una prueba unitaria web



El asistente del Eclipse nos genera automáticamente el siguiente código:

EjemplojcrissfTestCase.java

```
package com.autentia.ejemplojsf.pruebasunitarias;

import junit.framework.TestCase;

public class EjemplocrisjsfTestCase extends TestCase {

	public EjemplocrisjsfTestCase(String name) {
		super(name);
	}

	protected void setUp() throws Exception {
		super.setUp();
	}

	protected void tearDown() throws Exception {
		super.tearDown();
	}

}
```

Ahora ya sólo nos queda ir añadiendo las funciones de pruebas unitarias.

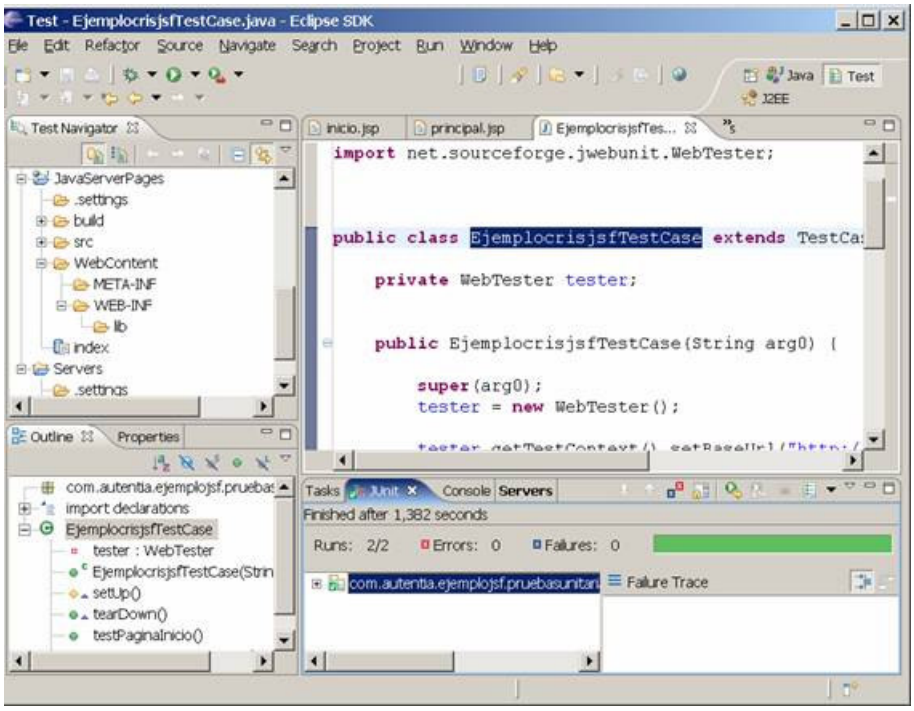
EjemplocrisjsfTestCase.java

```
package com.autentia.ejemplojsf.pruebasunitarias;

import junit.framework.TestCase;
import net.sourceforge.jwebunit.WebTester;
```

```
public class EjemplocrisjsfTestCase extends TestCase {

	private WebTester tester;


	public EjemplocrisjsfTestCase(String arg0) {

		super(arg0);
		tester = new WebTester();

		tester.getTestContext().setBaseUrl
("http://localhost:8080/EjemploCrisJSF");

	}

	protected void setUp() throws Exception {
		super.setUp();
	}

	public void tearDown() throws Exception {
		super.tearDown();
	}

	public void testPaginaInicio() {
		tester.beginAt("/inicio.faces");
		tester.assertButtonPresent("inicio:submit");
		tester.assertFormElementPresent("inicio:username");
		tester.setTextField("inicio:username", "usuario");
		tester.assertFormElementPresent("inicio:password");
		tester.setTextField("inicio:password", "sesamo");
		tester.clickButton("inicio:submit");
		//navegamos a la página principal
		tester.assertTextPresent("Bienvenido, usuario");
	}
	public void testPaginaInicioPasswordErronea() {
		tester.beginAt("/inicio.faces");
		tester.assertButtonPresent("inicio:submit");
		tester.assertFormElementPresent("inicio:username");
		tester.setTextField("inicio:username", "usuario");
		tester.assertFormElementPresent("inicio:password");
		tester.setTextField("inicio:password", "kk");
		tester.clickButton("inicio:submit");
		//volvemos a la página de inicio
		tester.assertFormPresent("inicio");
		tester.assertTextFieldEquals
("inicio:username","Reintentar");
	}
}
```

Ahora ejecutamos las pruebas unitarias. Para ello:

1. Arrancamos el servidor donde vamos a probar la aplicación. Si no sabes cómo, consulta mi tutorial sobre JSF en la página web de www.adictosaltrabajo.com
2. Prueba las páginas inicio.jsp y principal en el servidor, con el “Run as\Run on server”, para no encontrarnos errores “extraños” luego.
3. Pásate a la perspectiva “Test” en Eclipse
4. Pulsa con el botón derecho sobre el fichero EjemplocrisjsfTestCase.java y selecciona “Run as\Junit test”



Si todo va bien, obtendrás una pantalla como la que muestro. La raya verde indica que las dos pruebas unitarias ejecutadas han sido satisfactorias.

Resumen de funciones

Los objetos principales que utilizaremos para hacer las pruebas unitarias serán:

- TestContext. Establece un contexto para realizar las pruebas (locale, url base, cookies, autentificación).
- WebTester. Es el objeto que realizará las pruebas. Algunos de los métodos que puede utilizar son:
  - Métodos assert, del tipo assertButtonPresent("idBoton"). Algunos ejemplos (con el assert delante) serían las familias button* (present, notpresent, etc.), check*, element*, cookie*, form*, frame*, key*, link*, radio*, selectoption*, selectedoption*, submitButton*, table*, text*, title^*, windows*, donde el * indica que hay varios métodos disponibles, adaptados a cada tipo de elemento.
  - Métodos clic* para simular el pulsado sobre un elemento de la ventana
  - dumpHtml, que muestra en la salida estándar el código html que se está viendo en cada momento, muy útil para depurar nuestros test.
  - Métodos get, como getDialog, que permiten obtener varios elementos de la página o del contexto
  - Métodos goto, como gotoFrame, gotoWindow, gotoRootWindow, gotoPage, que nos permitirán navegar por la página.
  - Métodos submit, para que los formularios se envíen
  - Métodos set, para fijar el valor de los elementos de la página.

Puntos débiles

Los puntos débiles de Jwebunit vienen de los que muestren las bibliotecas de bajo nivel que utilicen. Por ejemplo, si usamos htmlunit, en su versión 1.9 tenemos por ejemplo los siguientes problemas:

- Si la página tiene un código javascript bastante complejo, nos puede dar problemas. Esto por ejemplo sucede si usamos determinados componentes JSF, que para almacenar su estado interno se apoyan fuertemente en código javascript.
- Upload de ficheros. Es un tema que no dispone de documentación.
- Alerts de javascript. Al no disponer de títulos o elementos html, no se pueden gestionar correctamente.
- La navegación por páginas con frames, que requiere un código algo complejo.

La nueva versión JWebUnit 2.0

Se está preparando (Noviembre 2006) la nueva versión 2.0 del framework, que ya soporta cuatro plugins: htmlunit, httpunit, Selenium y Jwebfit. Parece que este entorno se configura como el más prometedor para pruebas web automáticas desde java.
Estaremos atentos a la evolución de este proyecto, pues de su desarrollo dependerá la potencia de las pruebas unitarias web de nueva generación.

Conclusiones

Con Jwebunti podremos incluir de manera sencilla pruebas unitarias web en nuestros proyectos. Es un buen marco de trabajo (framework) tanto para realizar pruebas funcionales como para ver el comportamiento de componentes concretos de nuestra aplicación.
Si nuestro entorno de pruebas lo completamos con una plataforma de compilación automática, por ejemplo un sistema de versionado, compilación nocturna y de despliegue automático en un servidor web de pruebas, podremos

automatizar todo el proceso de compilación, empaquetado y pruebas de la aplicación. Pero esto lo dejamos para otro tutorial…

Puedes opinar sobre este tutorial aquí

Recuerda

que el personal de Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales)

¿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?

info@autentia.com

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos .....
Autentia = Soporte a Desarrollo & Formación

autentia
Real Business Solutions

Autentia S.L. Somos expertos en:
J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
y muchas otras cosas

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Soporte de Asserts en Java 1.4.x	Os mostramos como utilizar los asserts en Java (disponibles a partir de la versión 1.4)
Upload de ficheros en Java	Os mostramos como enviar ficheros a un servidor Web y manipularlos en un servlet en el servidor, gracias a APIs de apache
Pruebas unitarias Web para aplicaciones JSF	En este tutorial se puede encontrar una introducción y un análisis de los diferentes frameworks disponibles para realizar pruebas unitarias web de aplicaciones JSF
JUnit 4. Pruebas de Software Java	Tutorial que describe como utilizar la herramienta JUnit 4 para realizar pruebas de integridad y errores sobre Java.
Test con JUnit	Cuando se hacen desarrollo profesionales, no basta con hacer los programas, hay que asegurarse de que van a funcionar. Una de las técnicas más seguras es crear aplicaciones que incluyan el código para autoprobarse. Os mostramos como usar JUnit