

¿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

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

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

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)




Hosting Patrocinado por
enREDados.com


[Home](#) | [Quienes Somos](#) | [Empleo](#) | [Tutoriales](#) | [Contacte](#)



CoNcept

Lanzado TNTConcept versión 0.6 (12/07/2007)

Desde [Autentia](#) ponemos a vuestra disposición el software que hemos construido (100% gratuito y sin restricciones funcionales) para nuestra gestión interna, llamado TNTConcept (auTENTia).

Construida con las últimas tecnologías de desarrollo Java/J2EE (Spring, JSF, Acegi, Hibernate, Maven, Subversion, etc.) y disponible en licencia GPL, seguro que a muchos profesionales independientes y PYMES os ayudará a organizar mejor vuestra operativa.

Las cosas grandes empiezan siendo algo pequeño Saber más en: <http://tntconcept.sourceforge.net/>

 Autor: Cristóbal González Almiron 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. Contacte con Cristóbal González criskerberos-tutoriales@yahoo.com	NUEVO CATÁLOGO DE SERVICIOS DE AUTENTIA (PDF 6,2MB)  www.adictosaltrabajo.com es el Web de difusión de conocimiento de www.autentia.com  autentia real business solutions Catálogo de cursos
---	---

Descargar este documento en formato PDF [coregest3.pdf](#)

[Firma en nuestro libro de Visitas](#) <-----> [Asociarme al grupo AdictosAlTrabajo en eConozco](#)

hostinet.com

Dominios .es desde 4'95€



Fecha de creación del tutorial: 2007-09-10

Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: pruebas con Jetty y Tomcat

[Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: pruebas con Jetty y Tomcat](#)

[Introducción](#)

[Probando la aplicación con el plugin jetty](#)

[Probando la aplicación con el Tomcat 5.5 y Eclipse](#)

[Añadiendo la ventana de servidores](#)

[Añadiendo un servidor Tomcat local](#)

[Añadiendo el módulo de aplicación web a mano](#)

[Viendo las carpetas temporales del Tomcat](#)

[Probando la aplicación web en el navegador](#)

[Compilando código de Java 5.0](#)

[Preguntas frecuentes](#)

[Qué necesito para realizar los ejemplos y de dónde lo descargo](#)

[Modificando el faces-config.xml](#)

[Conclusión](#)

Introducción

En este tutorial seguimos la serie de nuestros tutoriales para hacer una aplicación web con Java, Maven, Eclipse e Hibernate. Vamos a realizar las pruebas de la aplicación sobre dos servidores web diferentes: el servidor Jetty, integrado en Maven, y el servidor Tomcat, que lo integraremos con Eclipse. La ventaja del servidor Jetty es que es muy fácil de configurar y de ejecutar, pero no permite la depuración de la aplicación. El Tomcat se ejecuta desde el Eclipse, y permite la depuración mediante puntos de ruptura en las clases java.

Probando la aplicación con el plugin jetty

El servidor Jetty es un servidor web escrito en Java y que se integra de manera sencilla en otras aplicaciones. Está integrado a través de un plugin en Maven, de manera que su uso es muy sencillo desde Maven, lo que nos permitirá probar y ejecutar nuestra aplicación web en segundos. Además el plugin de MyFaces configura automáticamente nuestra aplicación para poder usarla con Jetty.

Normalmente las aplicaciones web se pueden probar desde Maven 2 con el plugin de jetty. Para ello basta añadir al POM del proyecto las siguientes líneas:

```
pom.xml
<project>
...

    <build>
        <plugins>
            <plugin>
                <groupId>org.mortbay.jetty</groupId>
                <artifactId>maven-jetty-plugin</artifactId>
                <configuration>
                    <scanIntervalSeconds>10</scanIntervalSeconds>
                </configuration>
            </plugin>
        </plugins>
    </build>...
</project>
```

Como nuestra aplicación contiene el framework JSF, el plugin de jetty debe también incluir una serie de bibliotecas jars del Tomcat. Para ello cuando se crea el POM del proyecto, el plugin de myfaces le añade un perfil al POM que le permitirá ejecutar el plugin de jetty sin problemas. En nuestro caso nos queda la configuración del plugin como sigue:

```
coregest\pom.xml
<project>
...

    <!-- Profile to run jetty, so the tomcat jars are included in the bundle. They are not
    included by default -->
    <profiles>
        <profile>
            <id>jettyConfig</id>
            <build>
                <plugins>
                    <plugin>
                        <groupId>org.mortbay.jetty</groupId>
                        <artifactId>maven-jetty-plugin</artifactId>
                        <configuration>
                            <scanIntervalSeconds>10</scanIntervalSeconds>
                        </configuration>
                    </plugin>
                </plugins>
            </build>

            <dependencies>
                <dependency>
                    <groupId>net.sf.coregest</groupId>
                    <artifactId>coregest-core</artifactId>
                    <version>1.0-SNAPSHOT</version>
                    <scope>compile</scope>
                </dependency>
                <dependency>
                    <groupId>javax.servlet</groupId>
                    <artifactId>jsp-api</artifactId>
                    <version>2.0</version>
                    <scope>compile</scope>
                </dependency>
                <dependency>
                    <groupId>tomcat</groupId>
                    <artifactId>jasper-compiler</artifactId>
                    <version>5.5.9</version>
                    <scope>compile</scope>
                </dependency>
            </dependencies>
        </profile>
    </profiles>

```

```

        <groupId>tomcat</groupId>
        <artifactId>jasper-runtime</artifactId>
        <version>5.5.9</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>tomcat</groupId>
        <artifactId>jasper-compiler-jdt</artifactId>
        <version>5.5.9</version>
        <scope>compile</scope>
    </dependency>
</dependencies>

<pluginRepositories>
    <!-- Repository to get the jetty plugin -->
    <pluginRepository>
        <id>mortbay-repo</id>
        <name>mortbay-repo</name>
        <url>http://www.mortbay.org/maven2/snapshot</url>
    </pluginRepository>
</pluginRepositories>

</profile>
</profiles></project>

```

También debemos incluir la dependencia de nuestro módulo coregest-core, que he añadido al principio.

Este perfil se activa con la opción `-P nombreDelPerfil` (en nuestro caso `jettyConfig`) del maven, por lo que para llamar al plugin haremos `mvn -P jettyConfig jetty:run`



Al finalizar la ejecución del comando se lanza el servidor Web jetty



Ahora podemos probar la aplicación, probando con nuestro navegador en la dirección local <http://localhost:8080/coregest/index.jsp>, que es la página de inicio de la aplicación.



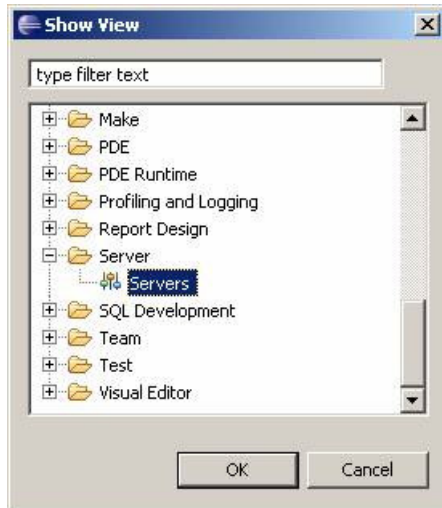
Como vemos, tal y como está construida la aplicación, inmediatamente nos redirige al servlet de jsf.

Probando la aplicación con el Tomcat 5.5 y Eclipse

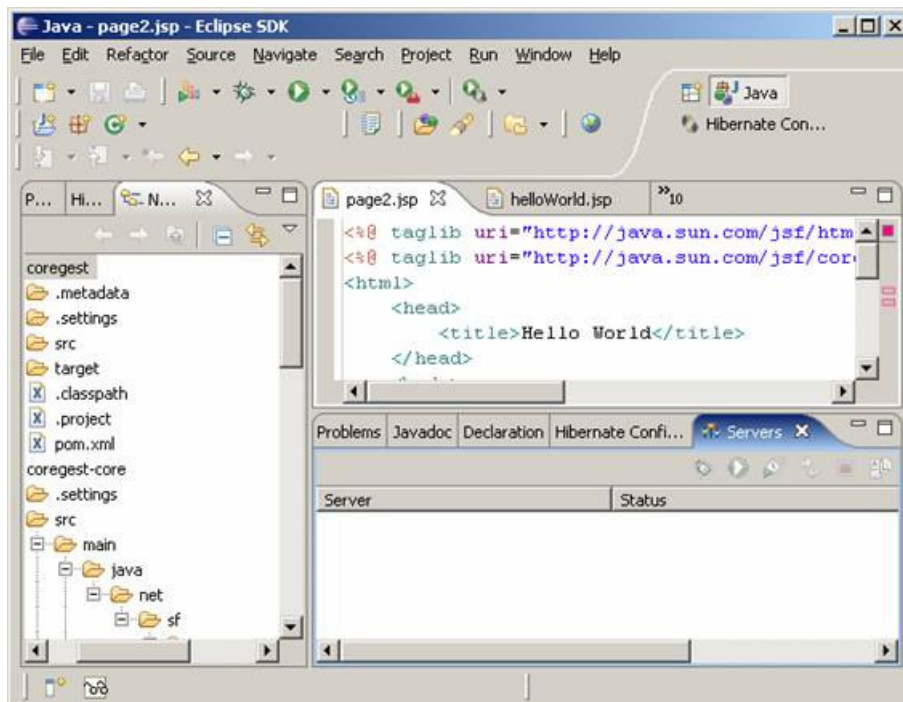
Para probar la aplicación con Tomcat 5.5 debemos tener instalado un Tomcat en local. Yo suelo instalarlo en D:\apps\apache-tomcat-5.5.17 a partir del zip correspondiente, que nos bajamos de <http://tomcat.apache.org>, por lo que se debe hacer esto primero. Basta simplemente con descomprimirlo en esta ruta para nuestros propósitos.

Añadiendo la ventana de servidores

Desde Eclipse 3.2, abrimos la ventana de Servers con la opción del menú Windows>ShowView\Others...,

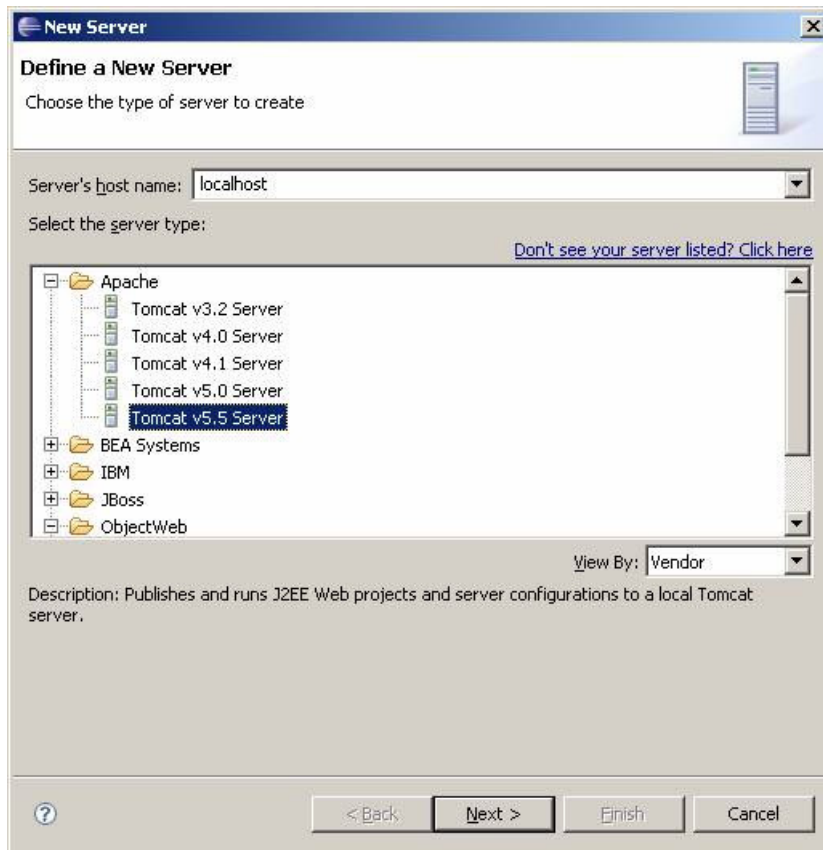


En este cuadro de diálogo seleccionamos Servers\Servers y pulsamos OK

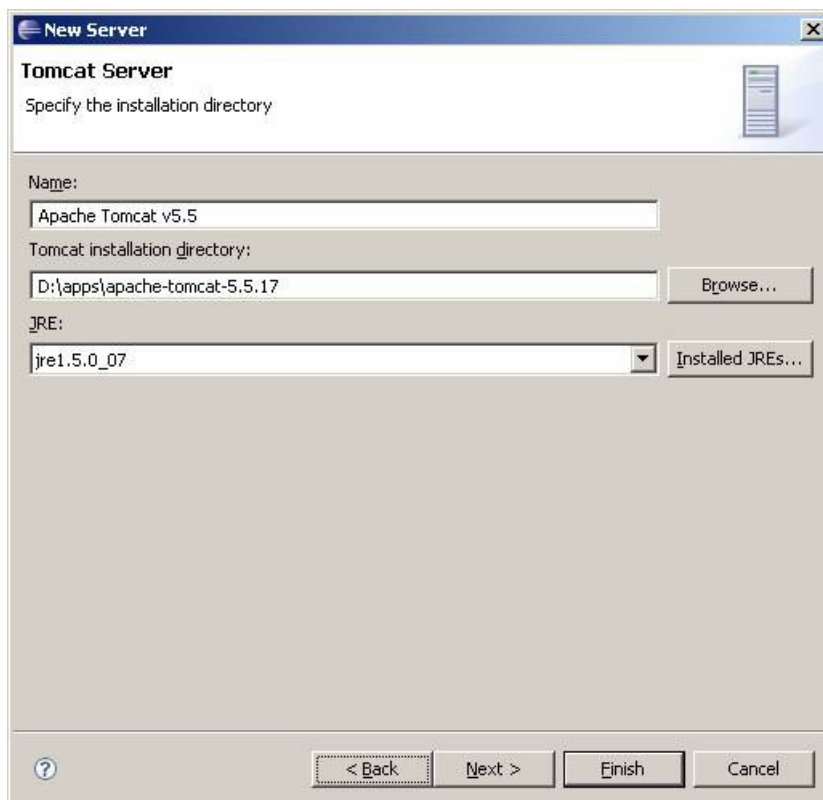


Añadiendo un servidor Tomcat local

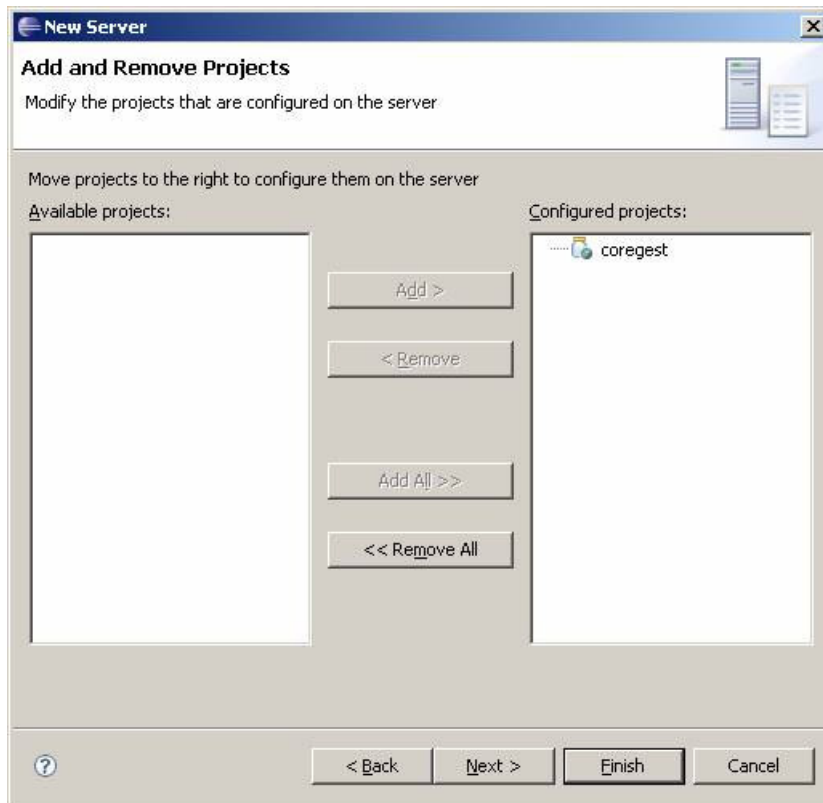
El Eclipse nos muestra ahora el panel de servidores. Con el botón secundario en el interior del panel añadimos un nuevo servidor Tomcat 5.5.



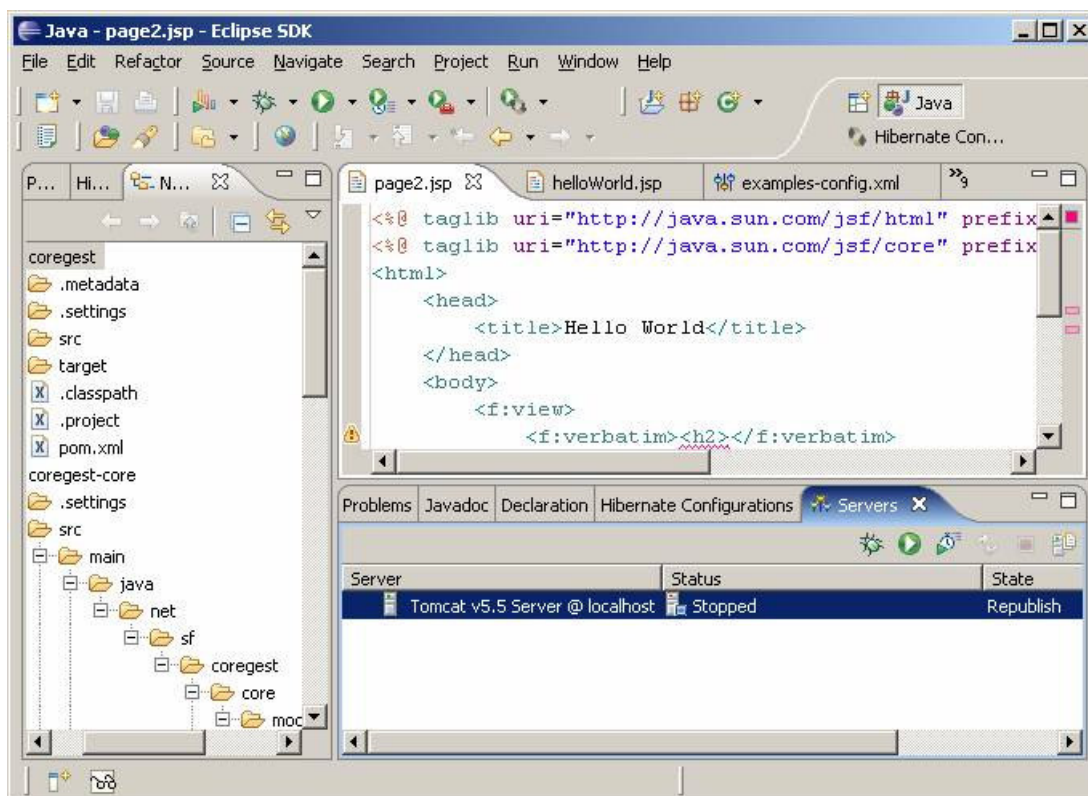
Pulsamos "Next"



Hay que poner nuestra ruta al tomcat y el runtime Java para el tomcat. Yo suelo poner siempre el que estoy usando, para que el no haya problemas si tengo varios.

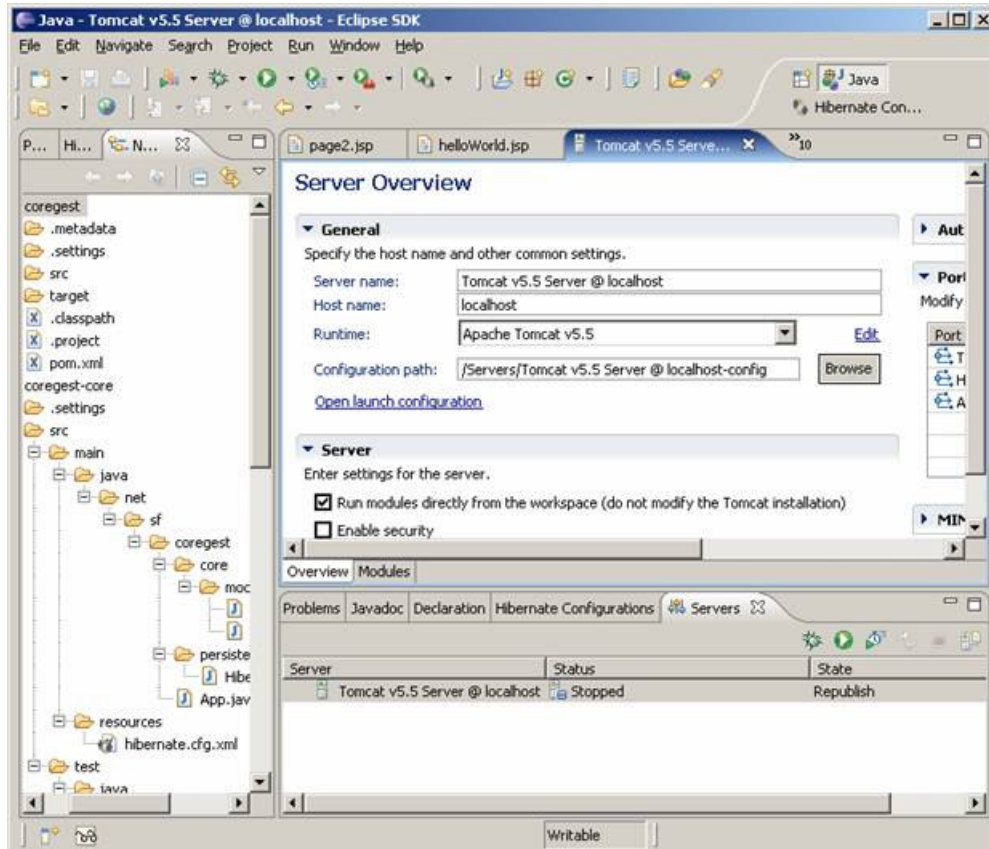


Añadimos el módulo que contiene el webapp al tomcat y finalizamos

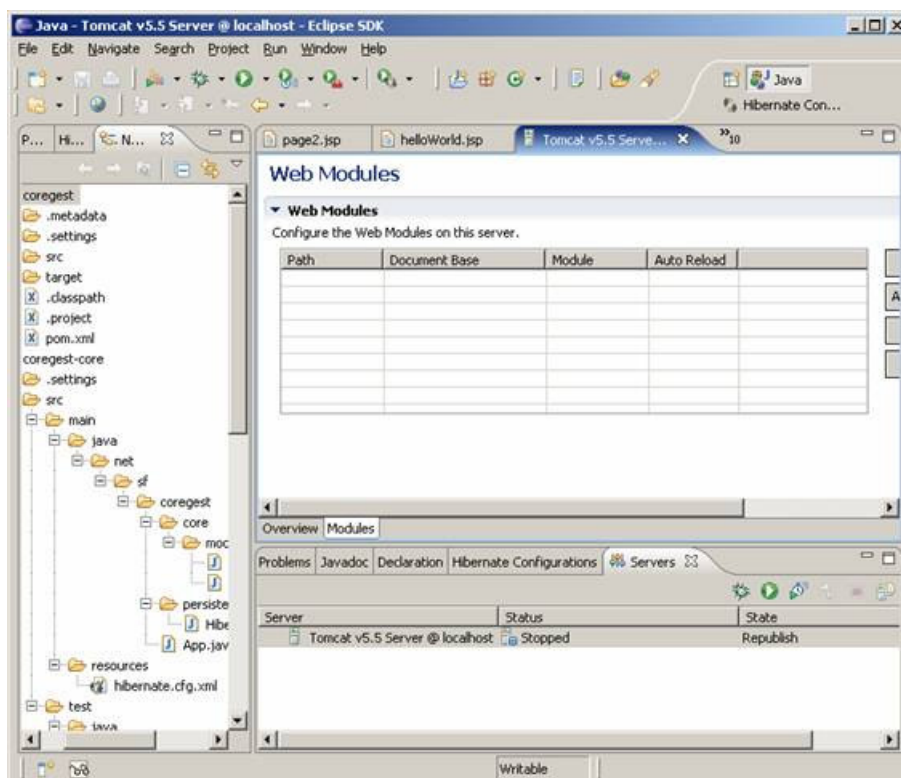


Añadiendo el módulo de aplicación web a mano

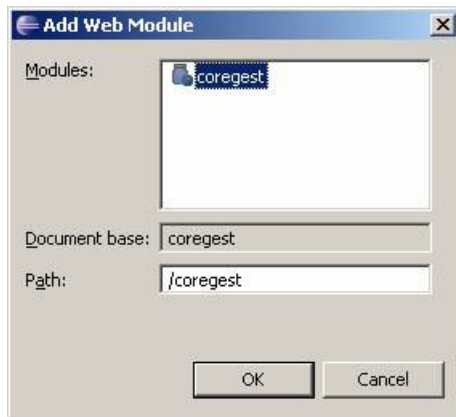
Ahora aparece la ventana de Servers con el nuevo Tomcat. Haciendo doble click en el servidor tomcat del panel Servers podemos ver su configuración



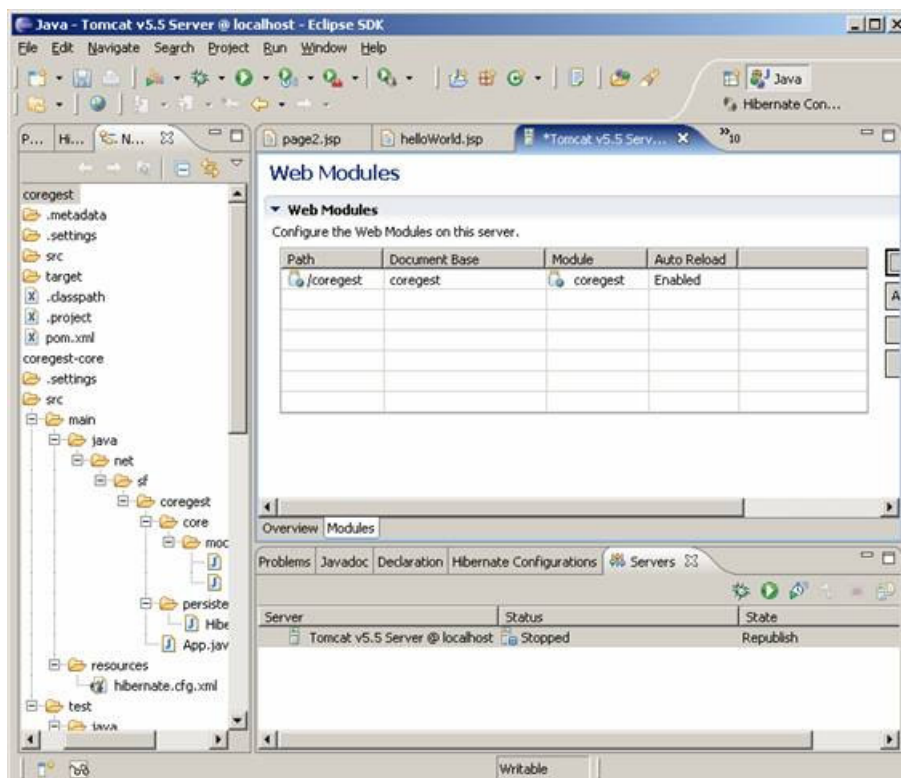
Hay que asegurarse de que en la pestaña “Modules” aparezca nuestro módulo coregest. En otro caso habremos tenido problemas... Pincho en la pestaña y ya veo que los tengo.



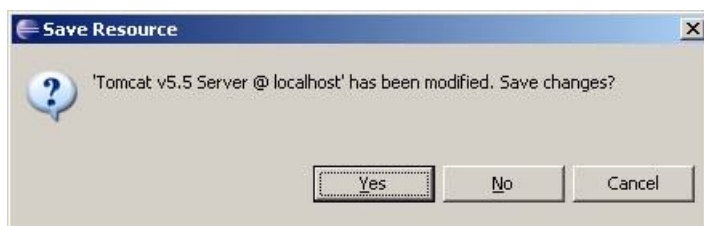
No hay problema, lo añado de nuevo. Pulso en “Add web module” y me sale el cuadro para elegir un módulo que sea un proyecto web:



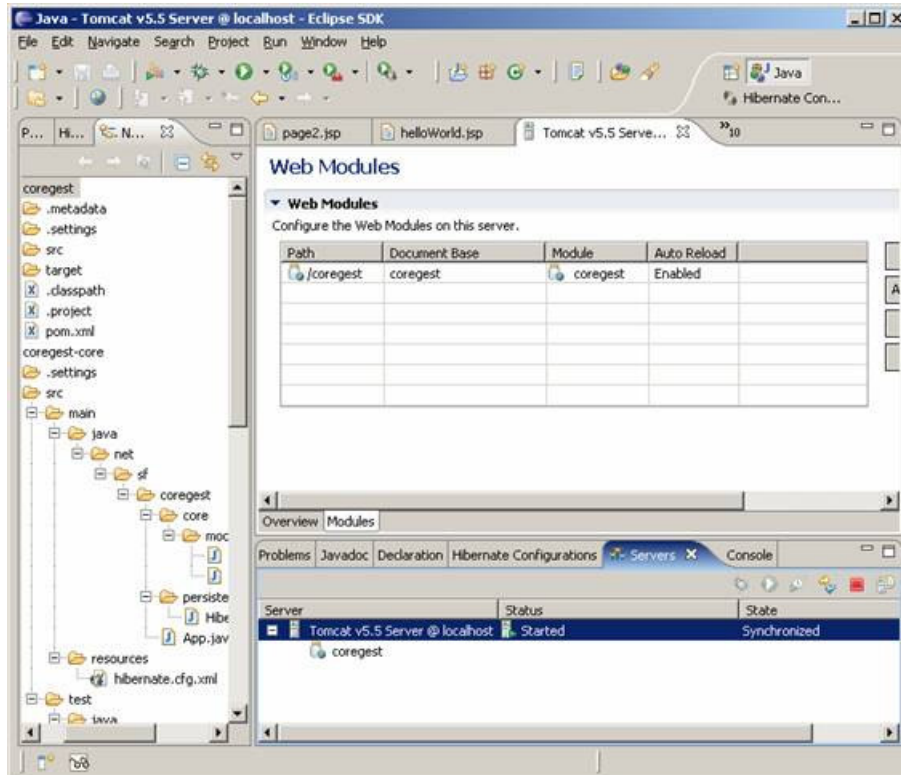
Eligo mi módulo coregest y pulso OK



Esto es otra cosa. Como vemos el Eclipse 3.2 tiene pequeños fallos por ahí sueltos. Ahora al pulsar el botón de ejecutar Eclipse nos pide que guardemos la configuración nueva del Tomcat:

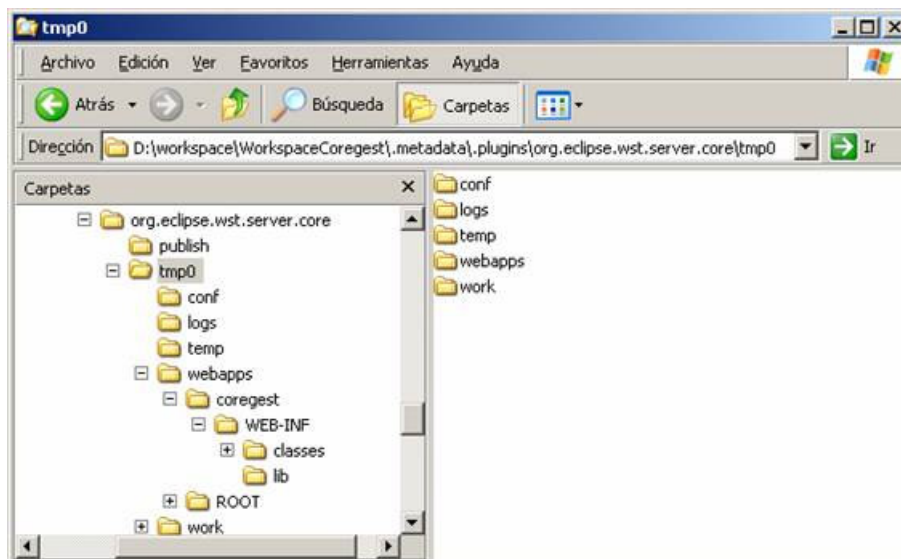


y se arranca el tomcat del Eclipse



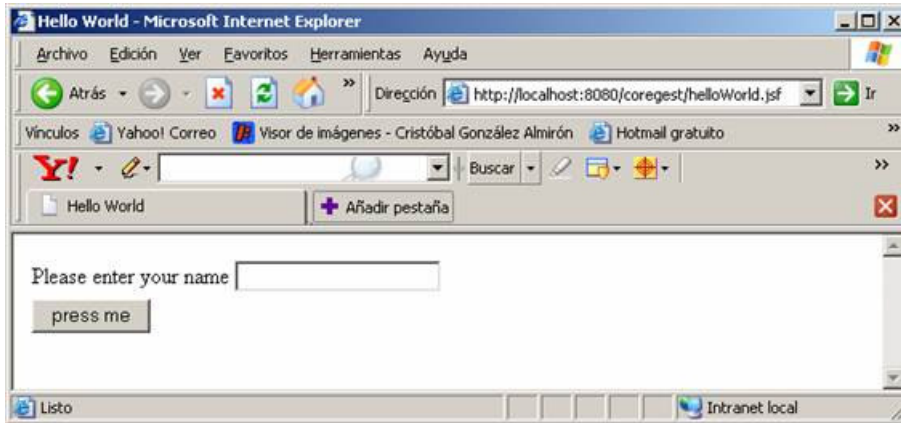
Viendo las carpetas temporales del Tomcat

Si queremos ver dónde están los ficheros de nuestro tomcat abrimos en el explorador la siguiente ruta: {workspace}\.metadata\plugins\org.eclipse.wst.server.core\tmp0. Dentro de esta carpeta están los archivos de configuración del tomcat y el webapp. También está la carpeta work (esa que deberemos borrar de vez en cuando para que Tomcat cargue las clases que hemos desarrollado y no las carga...). Mi consejo: crearos un acceso directo a esa carpeta porque lo vais a usar mucho con Eclipse 3.2 y Tomcat 5.5.



Probando la aplicación web en el navegador

Ahora sólo nos falta abrir el navegador en <http://localhost:8080/coregest/index.jsp> para probar si funciona todo.



Ya tenemos el Tomcat funcionando y listo para depurar nuestra aplicación.

Importante: estos pasos se aplican a Eclipse 3.2 y Tomcat 5.5. En Eclipse 3.3 el plugin ha cambiado y el de Eclipse 3.1 también era diferente. Aunque los pasos son similares hay que tener en cuenta los cambios.

Compilando código de Java 5.0

Si nuestro proyecto tiene módulos cuyo código fuente sea Java 5.0 (por ejemplo si usamos genéricos de Java como en List<MiClase>), debemos habilitar la compilación de dicho código fuente. Para ello debemos modificar el plugin de compilación, en el POM del proyecto

```
coregest\pom.xml
<project>

  <build>
    <finalName>coregest</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
```

Preguntas frecuentes

Qué necesito para realizar los ejemplos y de dónde lo descargo

- JDK 1.5. Hay que descargarlo de la página de Sun. <http://java.sun.com>, y buscar en download el JDK de la edición estándar (JDK 1.5 SE) . Normalmente tendremos vínculos a las descargas más populares en la página principal.
- Eclipse. Se descarga de www.eclipse.org. Hay que descargarse el Eclipse correspondiente a nuestra plataforma

Modificando el faces-config.xml

Cuando creamos el proyecto con el plugin de archetype de MyFaces, el fichero de configuración por defecto se llama example-config.xml. Si lo renombramos simplemente a faces-config.xml se podrá abrir correctamente el plugin de JSF de Eclipse.

Conclusión

En este tutorial hemos visto cómo probar nuestra aplicación web de dos modos diferentes, uno muy sencillo con Jetty y otro más avanzado y potente con Eclipse y Tomcat. Cada uno tiene sus ventajas y sus desventajas, pero siempre podremos probar rápidamente nuestra aplicación web.



This work is licensed under a [Creative Commons Attribution-Noncommercial-No Derivative Works 2.5 License](https://creativecommons.org/licenses/by-nc-nd/2.5/).
[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 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

[Configuración de una Aplicación Web en Tomcat con codificación UTF-8](#)

[Seguridad en Tomcat](#)

[SSL en Tomcat](#)

[Pruebas de integración con Maven](#)

[Crear un repositorio remoto y como hacer una 'release' con varios proyectos en Maven y Eclipse](#)

[Activar soporte SSL en Tomcat](#)

[Como generar con Maven un histórico de cambios del proyecto](#)

[Apache con Tomcat](#)

[Proyecto con JSF Myfaces, Maven y Eclipse](#)

[Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: aplicación multimódulo](#)

Descripción

Este tutorial nos cuenta como solucionar el problema que podemos encontrarnos al configurar nuestras aplicaciones web en Tomcat con UTF8

Os mostramos como proteger de un modo básico el acceso a recursos dentro de vuestro servidor de componentes Tomcat

En este tutorial se describe cómo configurar Tomcat para acceder a las aplicaciones web empleando HTTP sobre SSL

Este tutorial nos muestra un ejemplo para lanzar las pruebas de integración "engañando" a Maven para que no se lancen en la fase de test teniendo únicamente un módulo para ambas

En este tutorial vamos a explicar como podemos trabajar teniendo varios proyectos relacionados en Maven y en Eclipse

Os mostramos como activar el acceso SSL en Tomcat, utilizando certificados generados por Keygen (java)

Como generar con Maven un histórico de cambios del proyecto.

En este tutorial vamos a poder ver y comprobar como Apache Web Server y Tomcat pueden convivir perfectamente uno con el otro para así poder aumentar el rendimiento total de nuestra aplicación

En este tutorial vamos a aprender a construir una aplicación básica JSF (Java Server Pages) utilizando el Maven 2.0 y las bibliotecas de MyFaces. Lo mejor de todo es que para crear el ejemplo no vamos a programar ni una línea.

En este artículo se va a abordar el desarrollo de una aplicación Myfaces JSF con Maven que sea multimódulo.

Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores.

En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo.

Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.

[Patrocinados por enredados.com](http://www.enredados.com) [Hosting en Castellano con soporte Java/J2EE](http://www.enredados.com)



www.AdictosAlTrabajo.com Optimizado 800X600