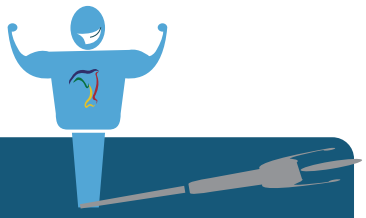
autentia

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...

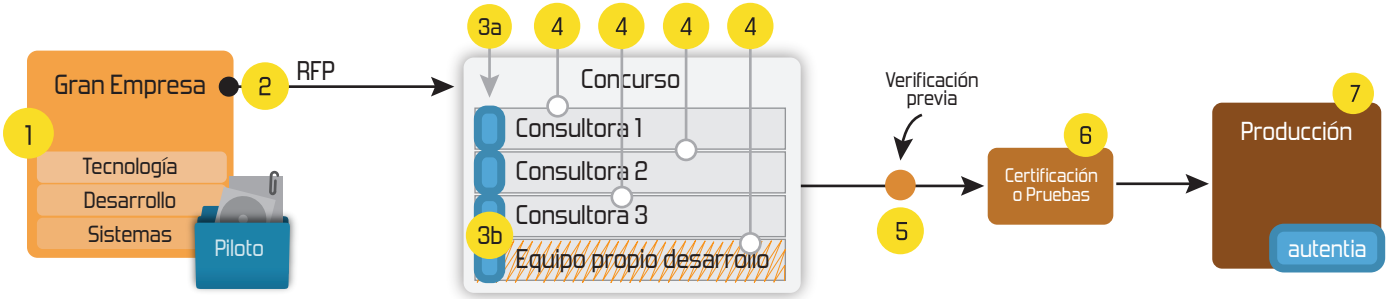
1. Desarrollo de componentes y proyectos a medida



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

adictos altrabajo Powered by autentia
Hosting Patrocinado por enREDados.com enREDados

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 (au**T**e**NT**ia).

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/



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.

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 coregest.pdf

Firma en nuestro libro de Visitas **<-----> Asociarme al grupo AdictosAlTrabajo en eConozco**

Eclipse Plugin
Apollo for Eclipse provides Java 5 & UML 2.1 Roundtrip
www.Gentleware.com

Centro Oficial Sun JAVA
Master , Prep. Exa Cert. , Cursos Java SE, Java EE, J2ME, JSF AJAX
www.programia.es

AltioLive
Real-time data visualisation, interaction and integration.
www.altio.com

Anuncios Google

Fecha de creación del tutorial: 2007-07-11

Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: aplicación multimódulo

Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: aplicación multimódulo

Introducción

Requisitos

Actualización del plugin de myfaces maven

Actualizando el plugin con maven

Separación de la lógica de negocio y presentación en dos módulos

Creación del proyecto base: el núcleo de la aplicación

Creación de módulo de la aplicación Web de interfaz de usuario

Uniendo los dos proyectos

Creando un proyecto maven que agrupe los módulos

Preparando los módulos para Eclipse

Añadiendo la variable del repositorio Maven local

Apñadiendo los módulos al Eclipse

Adadiendo el Facet de JSF

Conclusión

Introducción

Estoy preparando una nueva aplicación, llamada coregest, y ya que estamos, voy a compartir con vosotros cómo lo estoy haciendo.

Este tutorial es el primero de una serie de artículos dedicados al desarrollo de aplicaciones JSF con Maven. Lo voy a hacer aprovechando que estoy desarrollando una nueva aplicación en esta tecnología.

En este artículo vamos a abordar el desarrollo de una aplicación Myfaces JSF con Maven que sea multimódulo. Los puntos importantes que vamos a ver son:

- Aprender a buscar y actualizar plugins de Maven, sobre todo el de MyFaces (los que hayan leído mi tutorial anterior de JSF y Maven y lo hayan intentado lo entenderán…)
- Separar en módulos las aplicaciones con Maven
- También aprenderemos a integrar en Eclipse la biblioteca MyFaces para que podamos editar el fichero de configuración de JSF desde el Eclipse.

Requisitos

- Maven 2.0
- Java 5.0 (lo utilizaremos para hibernate)
- Eclipse 3.2
- Hibernate tools para eclipse 3.2 instaladas
- Artículos publicados en AdictosAlTrabajo:
    - JSF y Maven
    - Hibernate Tools
    - Crear el sitio web de documentación del proyecto con Maven Site

Actualización del plugin de MyFaces Maven

Al intentar crear el arquetipo de MyFaces Maven me ha dado problemas, por lo que tengo que actualizar el plugin.

De entrada encontré que la dirección de subversion para el plugin ha cambiado. Estos chicos…
Me sitúo en la carpeta del plugin (ver el artículo de jsf y maven)

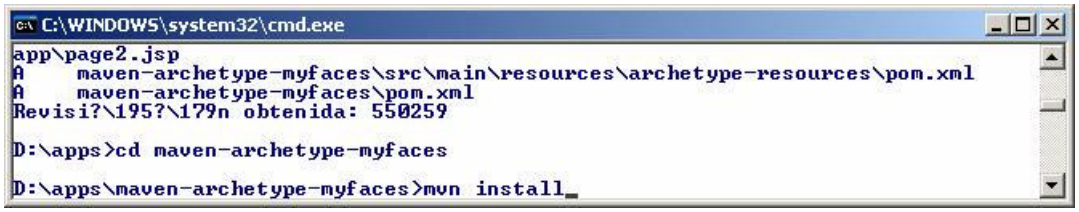


Borro el contenido y lo actualizo con

```
svn co http://svn.apache.org/repos/asf/myfaces/maven/trunk/archetypes/maven-archetype-myfaces/
```



Ahora lo instalo con mvn install



Se pone a trabajar hasta que finaliza la compilación e instalación:

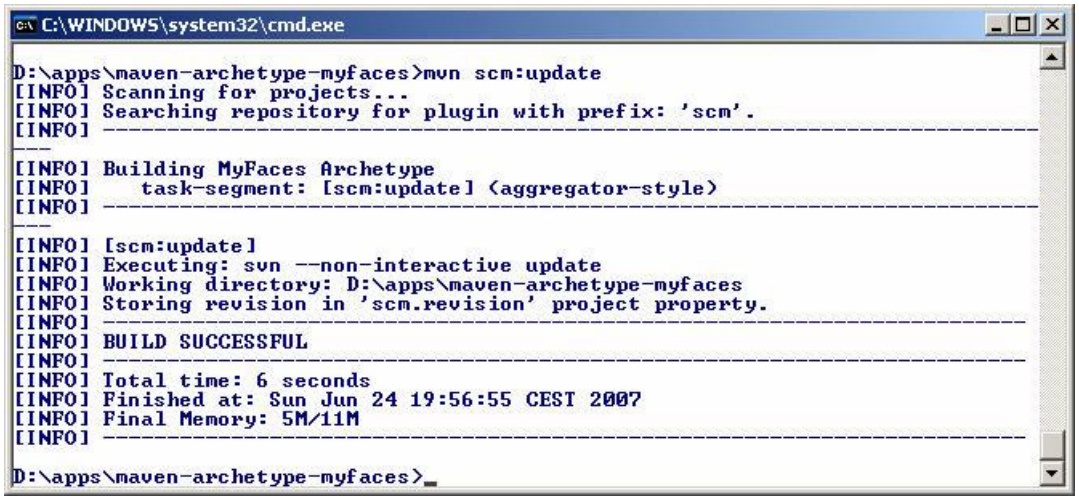


También habríamos podido resolver esto forzando a que el plugin de myfaces se convirtiese en local en una versión RELEASE, con la instrucción, según se indica en la página de plugins de maven (buscar el plugin install)

Svn install –DupdateReleaseInfo=true

Actualizando el plugin con maven

El método más sencillo para actualizarlo es hacer un mvn scm:update. Este comando asume que la información para el plugin scm de maven está actualizada en el pom del plugin. Al haber actualizado manualmente el plugin (debido al cambio de localización del respositorio svn dentro de apache) ahora me actualiza el plugin sin problemas.



Luego haríamos el mvn install y ya estaría.

Separación de la lógica de negocio y presentación en dos módulos

A medida que la aplicación crece debemos separar las partes de la misma en módulos. Ya que vamos a hacer una aplicación web, lo primero que debemos separar es la lógica de negocio de la de presentación y sesión. Para ello vamos a crear dos módulos: coregest-core y coregest (si lo llamo coregest-webapp luego hay que poner esto último en las url de pruebas y se hace pesado, pero sería el nombre correcto). Tendremos varias ventajas al hacerlo así:

- Las clases java van en un paquete jar. Al hacer pruebas de presentación sin modificar la lógica de negocio (suele ser habitual), la aplicación web se compila y empaqueta más rápido.
- Idealmente la lógica de negocio debería estar orientada a servicios (SOA), de manera que se pudiesen desarrollar y probar en un contenedor aislado, fuera de toda aplicación web. Esto facilita el desarrollo de pruebas unitarias y la migración a arquitecturas más avanzadas, así como la modularidad y reutilización.
- La inicialización de la lógica de negocio se hará desde el exterior, usando el patrón de inversión de control (IoC), al estilo de Spring, lo que nos permitirá probar la lógica en diferentes entornos sin modificar su empaquetado final.

Creación del proyecto base: el núcleo de la aplicación

D:\workspace\WorkspaceCoregest>mvn archetype:create -DgroupId=net.sf.coregest -DartifactId=coregest-core



Si nos sale el "BUILD SUCCESSFULLY" es que todo ha ido bien y se habrá creado la carpeta del módulo discarchiver-core en nuestro espacio de trabajo.

Creación de módulo de la aplicación Web de interfaz de usuario

Para crear el módulo web vamos a usar la siguiente línea (lo he puesto aquí para facilitar el copiado de la instrucción):

```
D:\workspace\WorkspaceCoregest>mvn archetype:create -DarchetypeGroupId=org.apach
e.myfaces.maven -DarchetypeArtifactId=maven-archetype-myfaces -DarchetypeVersion=
1.0-SNAPSHOT -DgroupId=net.sf.coregest -DartifactId=coregest
```



Si nos aparece el build successfully, todo ha ido bien. Se habrá creado la carpeta del proyecto correcta.

Uniendo los dos proyectos

Editamos el fichero pom.xml del proyecto coregest y le añadimos la dependencia de su módulo core:

D:\workspace\WorkspaceCoregest\pom.xml

```
...

<dependencies>
          <dependency>
                <groupId>net.sf.coregest</groupId>
                <artifactId>coregest-core</artifactId>
                <version>1.0-SNAPSHOT</version>
                <scope>compile</scope>

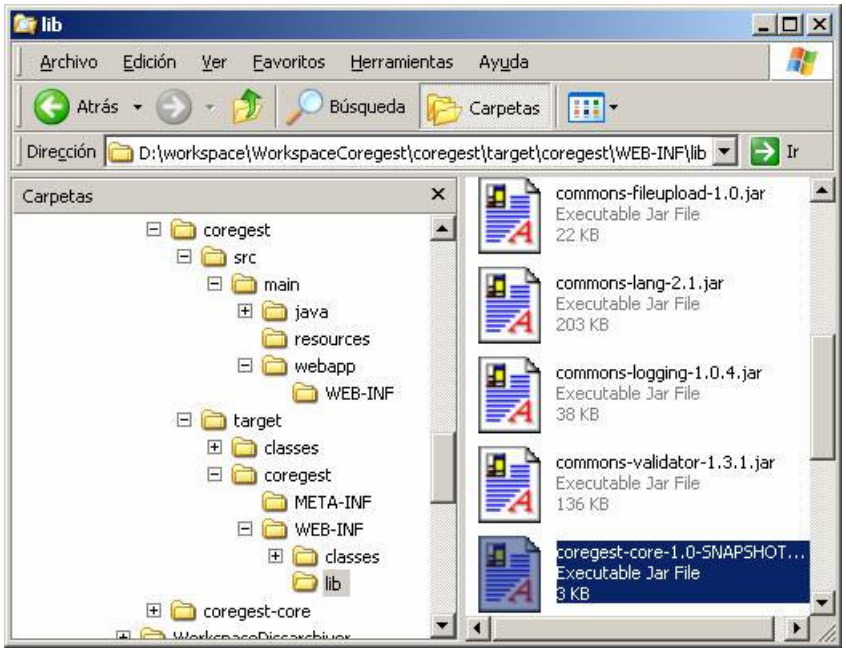
         </dependency>

            <dependency>
               <groupId>javax.servlet</groupId>
               <artif
...
```

Y probamos a crear el paquete con mvn package en el proyecto coregest:

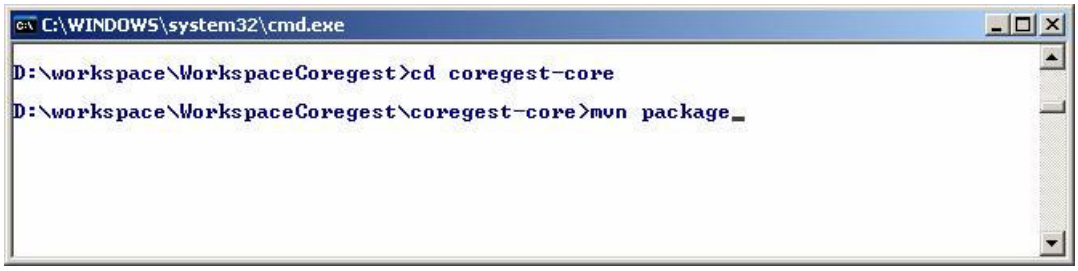


y el resultado es como esperamos...

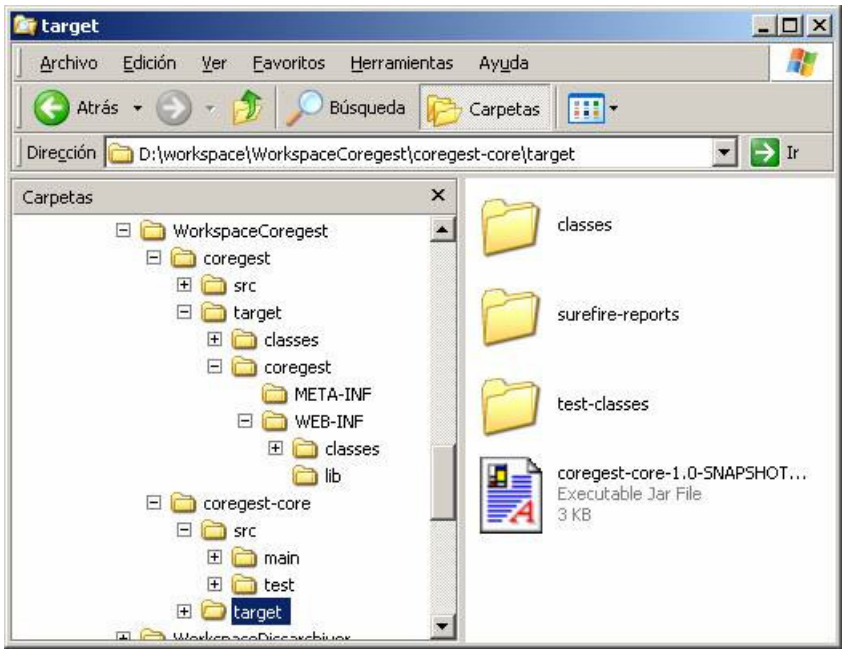




salvo que no se ha compilado el proyecto coregest-core. Hay que hacer un par de cosas: compilarlo par probarlo, rehaciendo de nuevo el coregest y crear un pom general que agrupe a ambos proyectos.
Compilamos el coregest-core como siempre



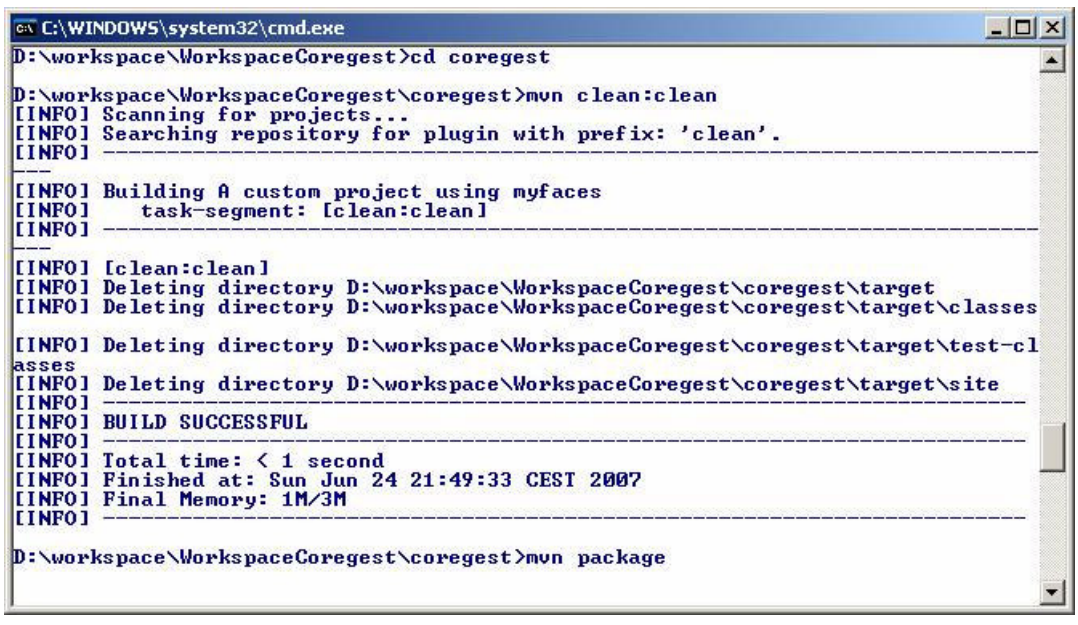
y nos crea el jar del módulo



Ya tenemos compilado el módulo coregest-core. Para que el módulo coregest lo pueda usar tiene que estar en el repositorio local. Esto lo logramos con mvn install:



ahora si reconstruimos el paquete coregest nos añadirá el coregest-core-10.SNAPSHOT.jar a la carpeta web-inf/lib de la aplicación web coregest.



Para reconstruir el paquete hay que hacer primero un mvn clean:clean, que limpiará la carpeta target del proyecto y luego el mvn package

Ahora ya podemos llamar a las clases de la lógica de negocio desde el módulo de presentación, ya que las clases ahora están en el classpath de la aplicación web, por haber sido metidas en la carpeta lib.

Creando un proyecto maven que agrupe los módulos

El siguiente paso es crear un proyecto que nos permita compilar y empaquetar todos los módulos con una sola operación. Para ello vamos a crear el siguiente pom en la raíz del espacio de trabajo:

Coregest.pom

```
<project>
    <name>Coregest</name>
          <url>http://coregest.sourceforge.net/</url>
    <modelVersion>4.0.0</modelVersion>

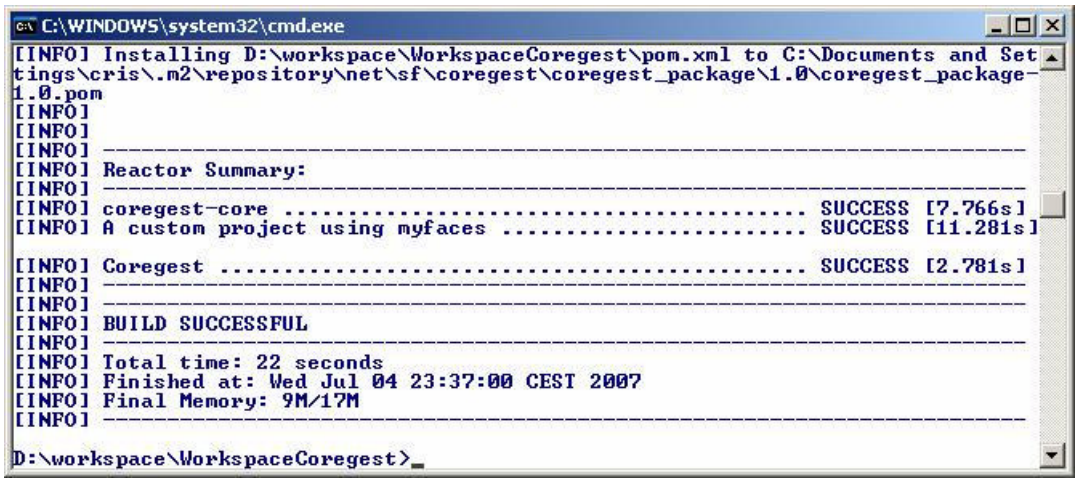
    <groupId>net.sf.coregest</groupId>
      <version>1.0</version>
    <artifactId>coregest_package</artifactId>
    <packaging>pom</packaging>

    <modules>
      <module>coregest-core</module>
      <module>coregest</module>
    </modules>
 </project>
```

Ahora compilamos todo con mvn install



Y nos da una salida final



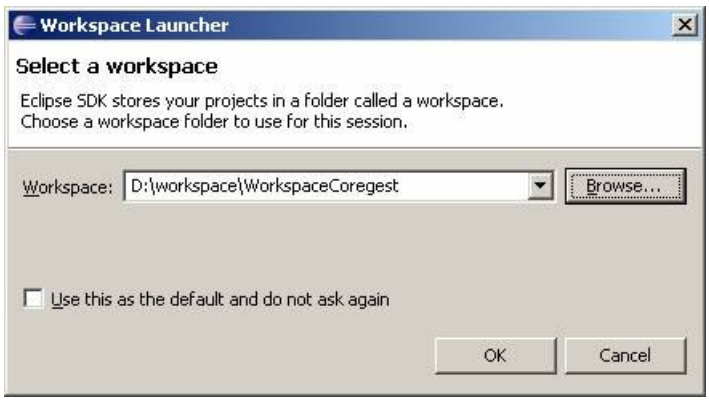
Como vemos se han compilado los dos proyectos (los comentarios de los proyectos no están todavía corregidos, no me ha dado tiempo…). De esta forma podríamos ir añadiendo todos los proyectos locales de los que depende el nuestro.

Preparando los módulos para Eclipse

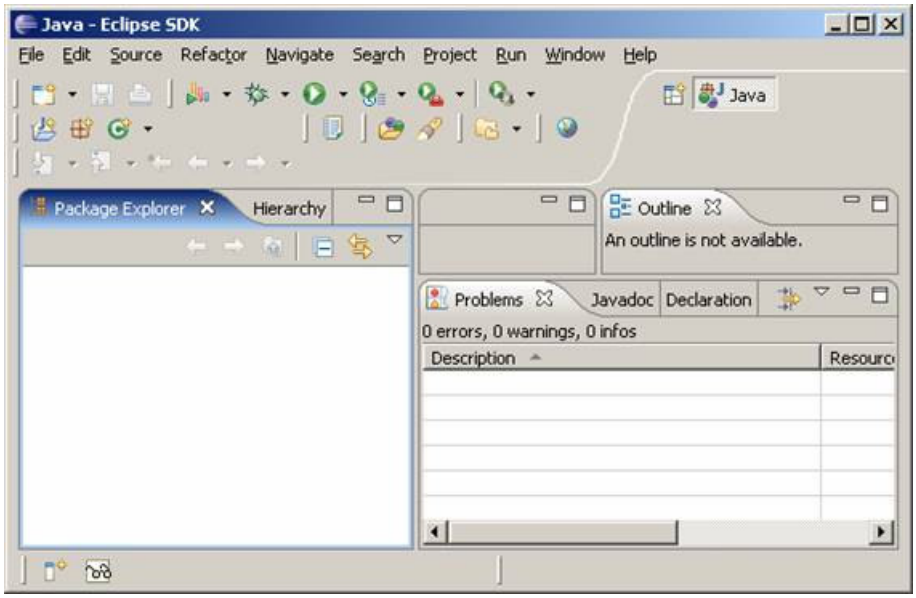
Ahora vamos a integrar los proyectos con Eclipse, como hicimos en el anterior tutorial

Añadiendo la variable del repositorio Maven local

Dado que hemos partido de un espacio de trabajo limpio, hay que prepararlo para Eclipse 3.2. Abrimos el Eclipse en nuestro espacio de trabajo nuevo:

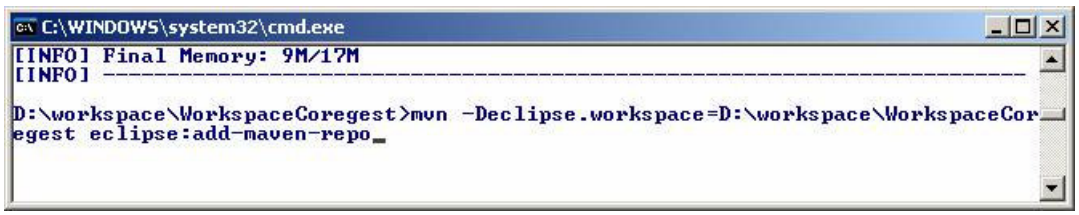


y nos abre un Eclipse limpio;

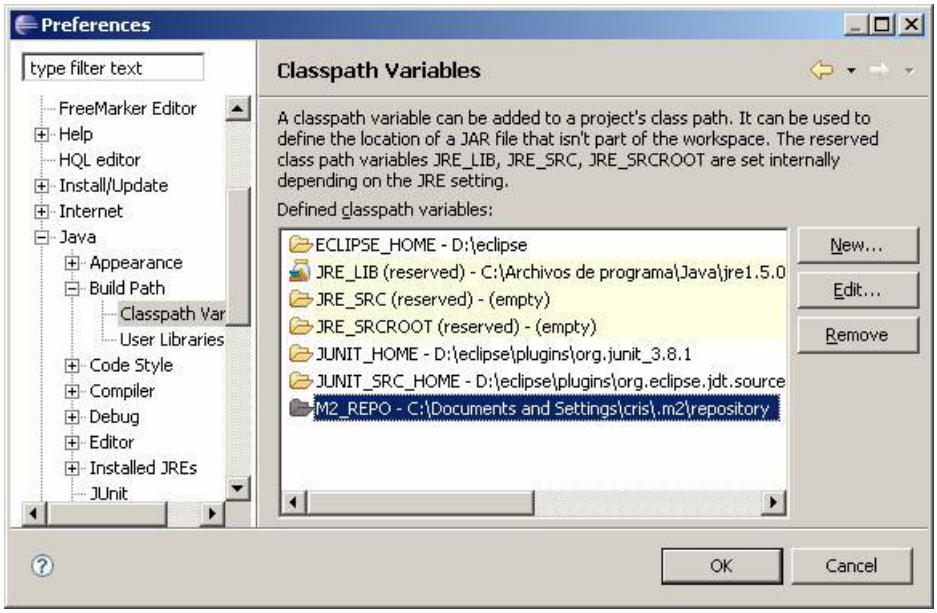


Lo cerramos para que se guarde la configuración. Se ha creado en nuestro espacio de trabajo la carpeta .metadata que es la que se necesita para configurar proyectos.

Ejecutamos mvn –Declipse.workspace= D:\workspace\WorkspaceCoregest eclipse:add-maven-repo, para añadir el repositorio local de Maven a este espacio de trabajo:

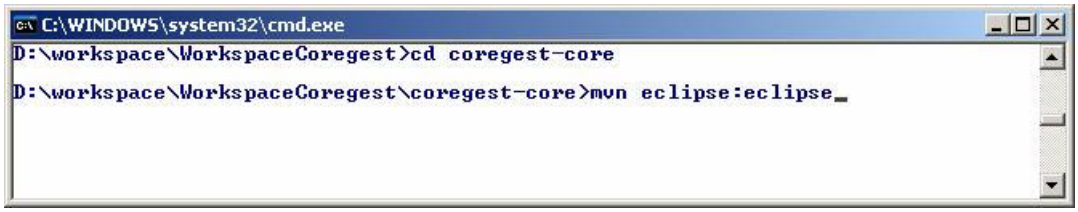


Abrimos de nuevo el Eclipse y comprobamos en Java\Build path\Classpath variables si se ha dado de alta correctamente el repositorio de Maven



Añadiendo los módulos al Eclipse

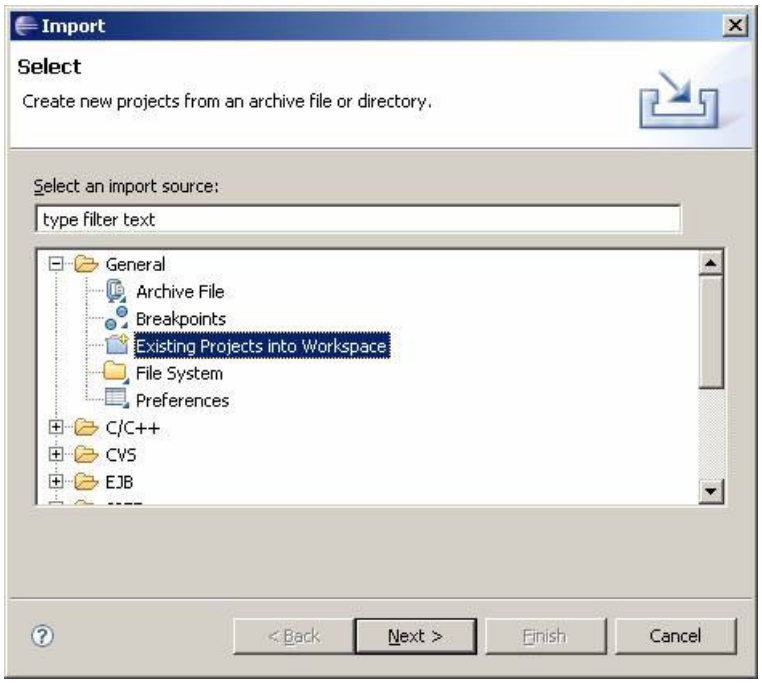
Ahora hay que preconfigurar los módulos para luego importarlos en el Eclipse. Vamos con el módulo de la lógica de negocio:



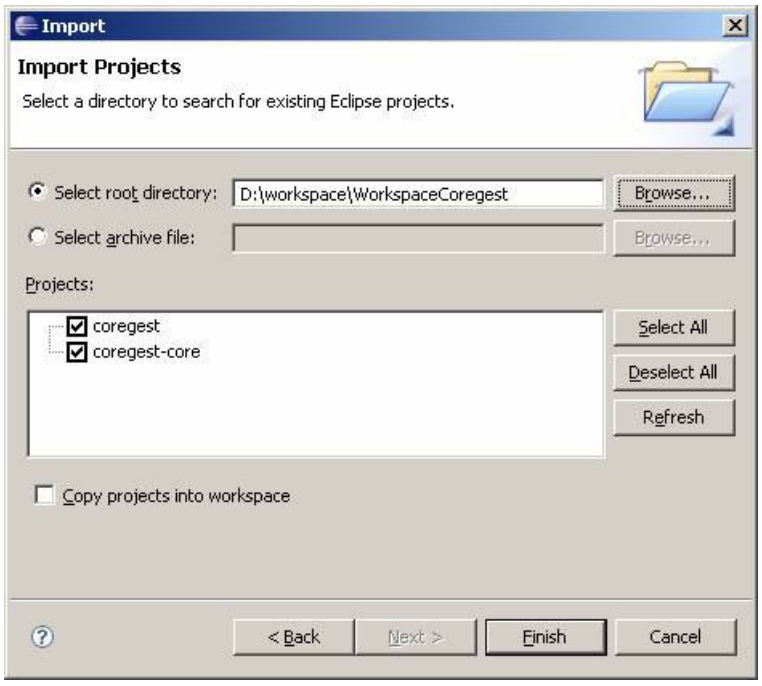
si ha ido bien vamos con el segundo, que es la aplicación web, especificando el tipo de plugin para wst



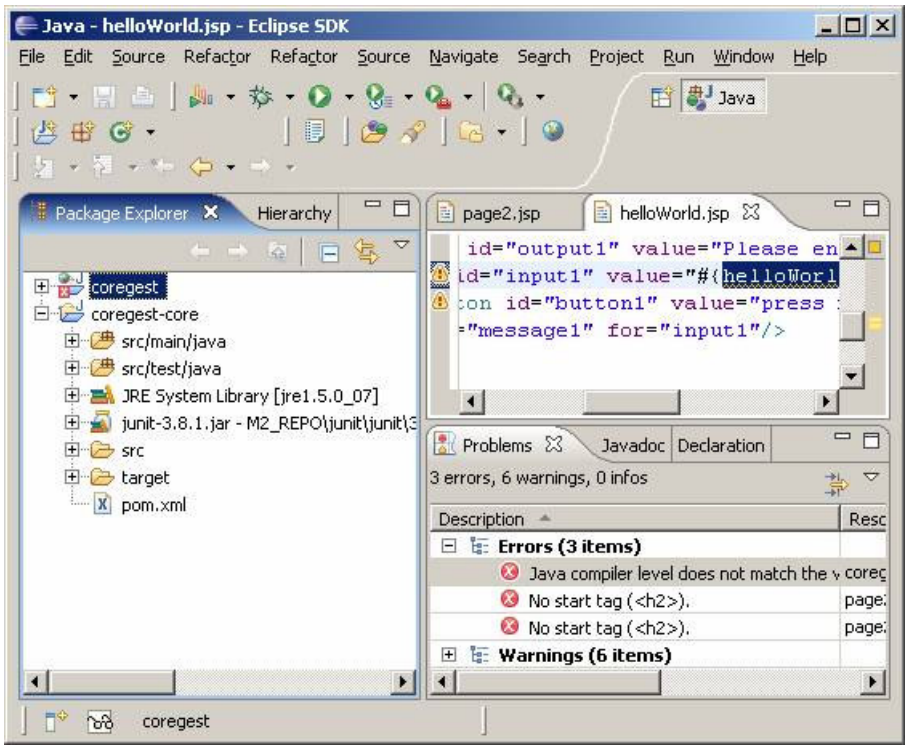
Ahora importamos los proyectos en el Eclipse, con File\Import...\Existing proyect into Workspace...



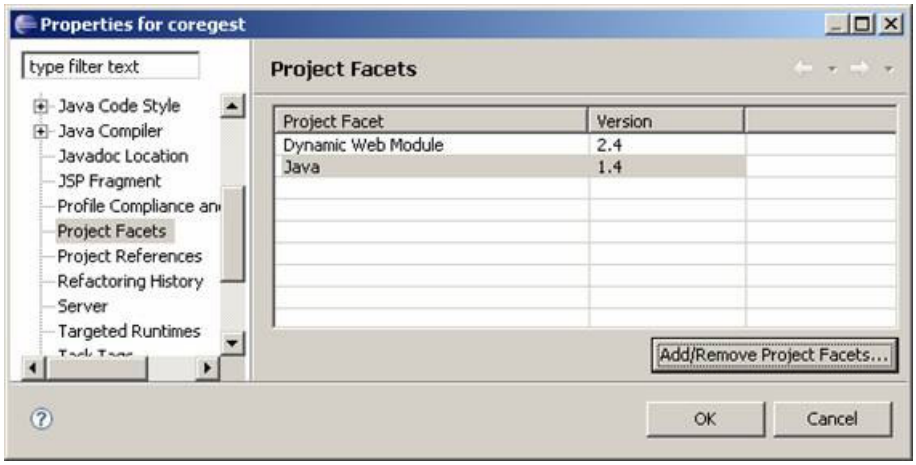
Pulsamos "next"



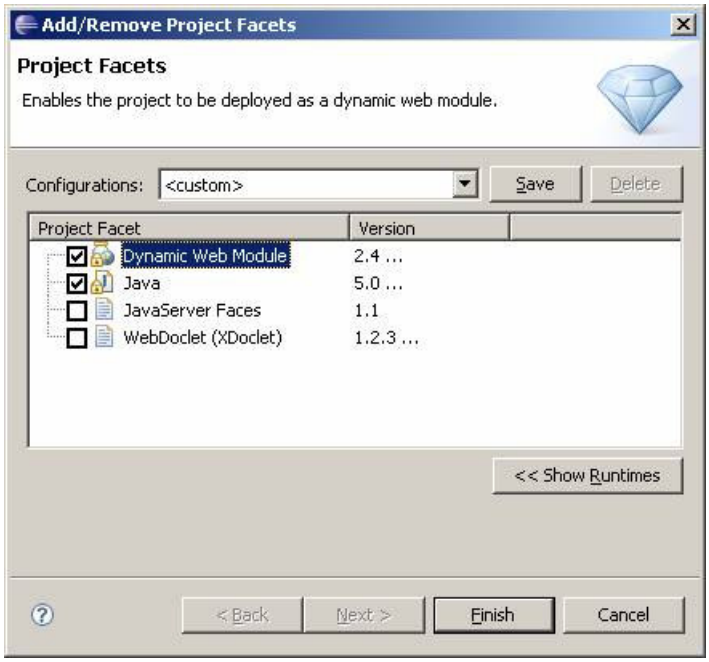
Primero pulsamos "Browse", seleccionamos nuestro espacio de trabajo y el cuadro de diálogo mostrará los dos proyectos. Los importamos con "Finish"



Aparentemente ambos proyectos se han cargado. Me sale un error, ya que el proyecto declara un Java Proyect Facet = 1.4 y estoy usando un runtime de Java 5.0. por lo que cambio el Java Proyect Facet en las opciones del proyecto:

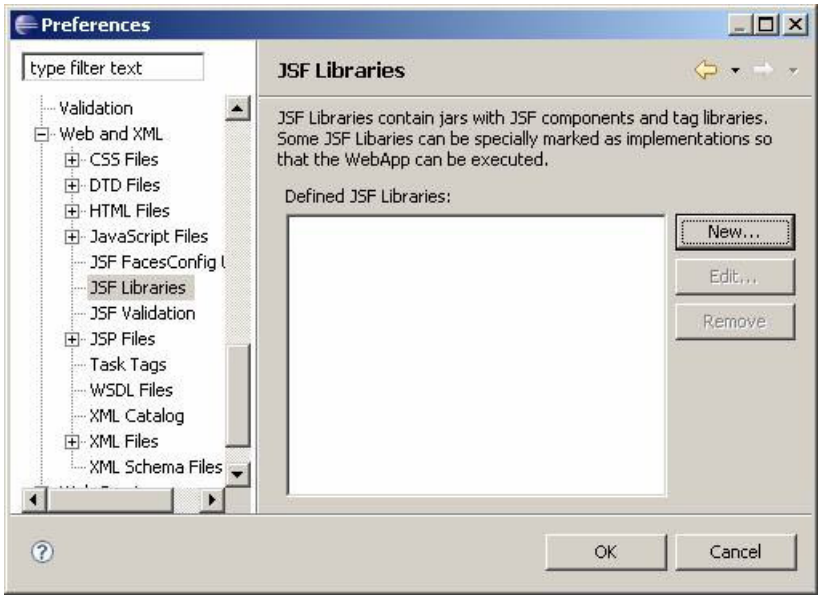


Pulso en "Add/Remove project facets..."

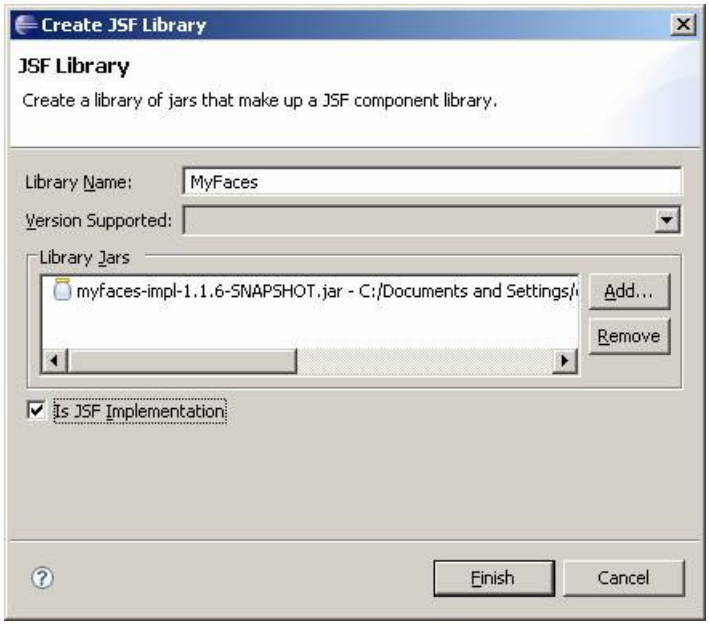


Añadiendo el Facelet de JSF

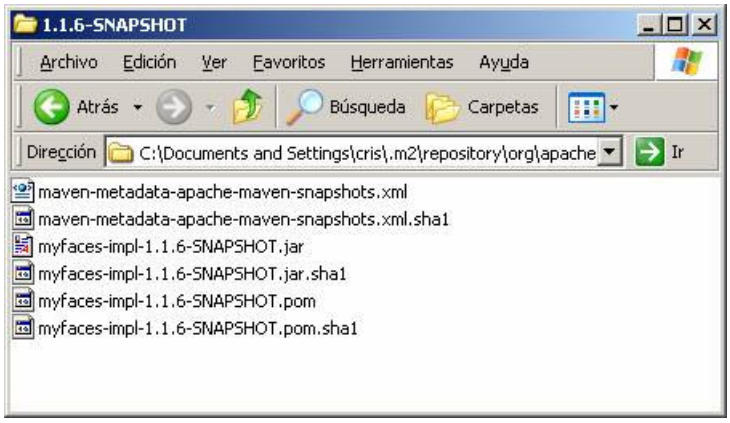
Primero vamos a añadir las MyFaces JSF a nuestro Eclipse. Para ello abrimos en Windows\Preferences\Web and XML\JSF Libraries una nueva biblioteca JSF



Pulsamos en "New..."



Debemos buscar el jar de Myfaces en nuestro repositorio local, que se llama myfaces-impl-1.1.6-SNAPSHOOT.jar.

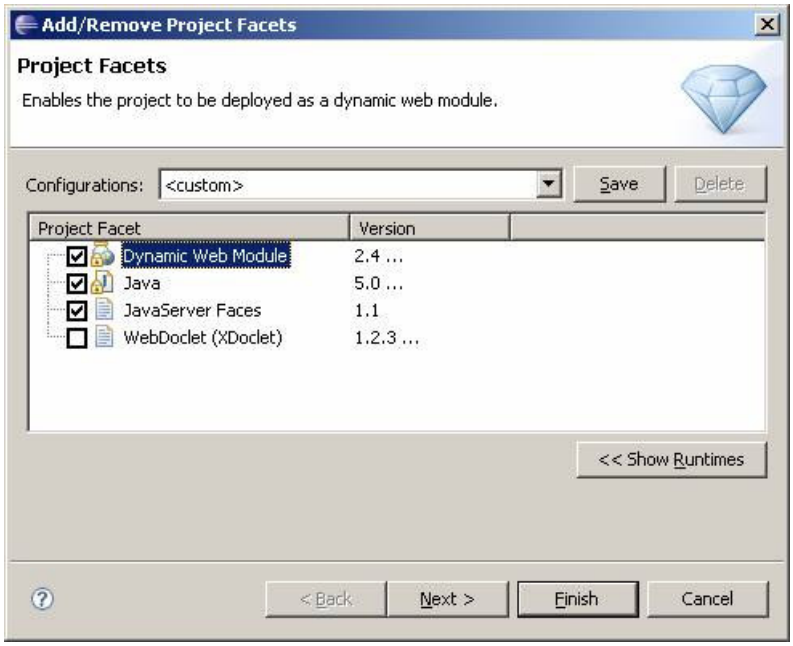


En mi caso, la ruta hasta el core de MyFaces ha sido C:\Documents and Settings\cris\.m2\repository\org\apache\myfaces\core\myfaces-impl\1.1.6-SNAPSHOT, que es muy largo…

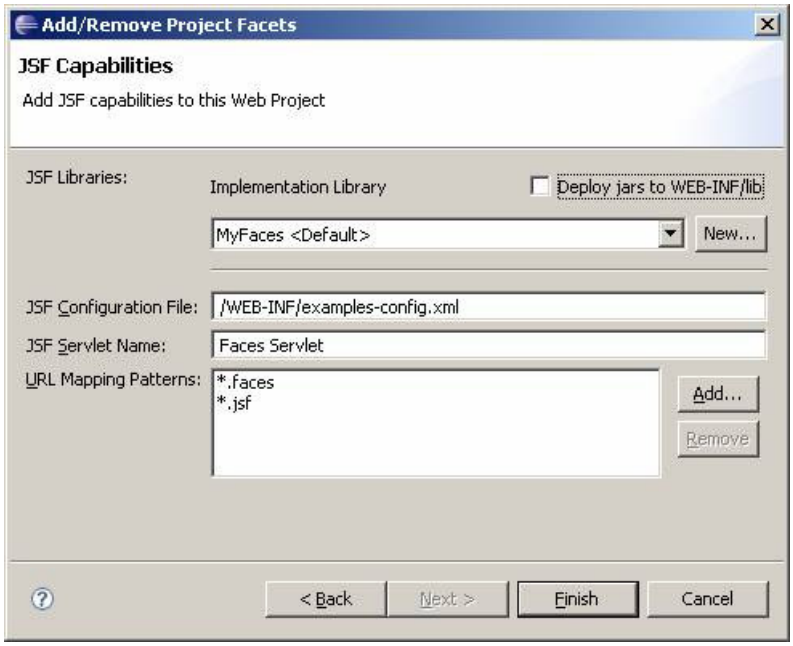
Y finalmente pulsamos "Finish"



Nos vamos a las opciones de nuestro proyecto:



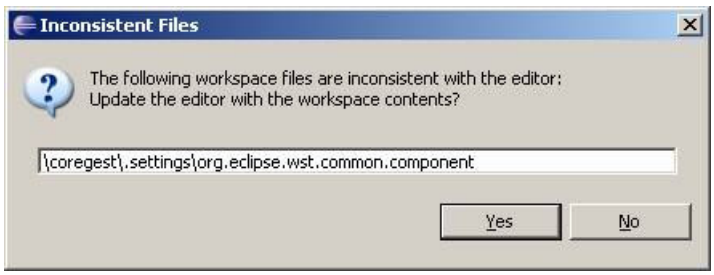
Marcamos Java Server Faces y pulsamos "Next"



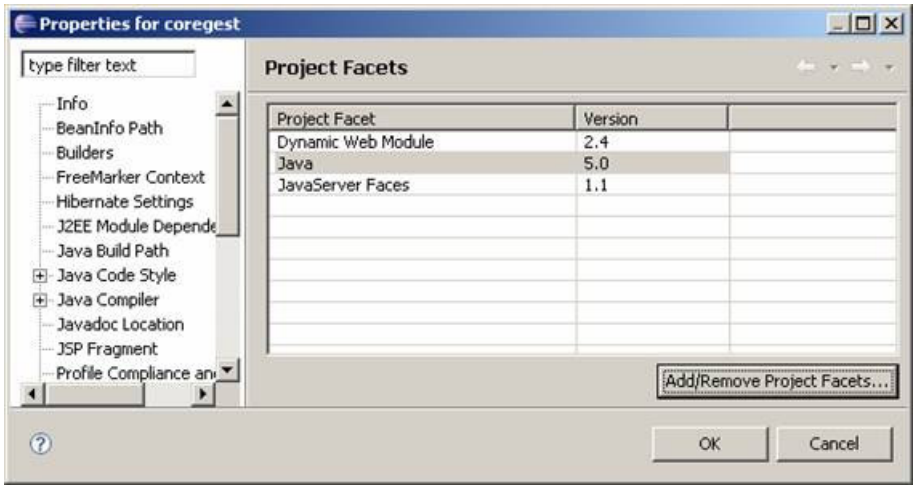
También en el cuadro de diálogo hemos hecho algunas modificaciones:

- Hemos añadido la extensión .jsf, ya que es la que usa el arquetipo por defecto.
- Hemos cambiado el nombre al archivo de configuración del jsf, por el mismo motivo. Más adelante podremos ponerlo a nuestro gusto.
- No queremos que los jar se copien al web-inf/lib, porque ya lo hace maven.

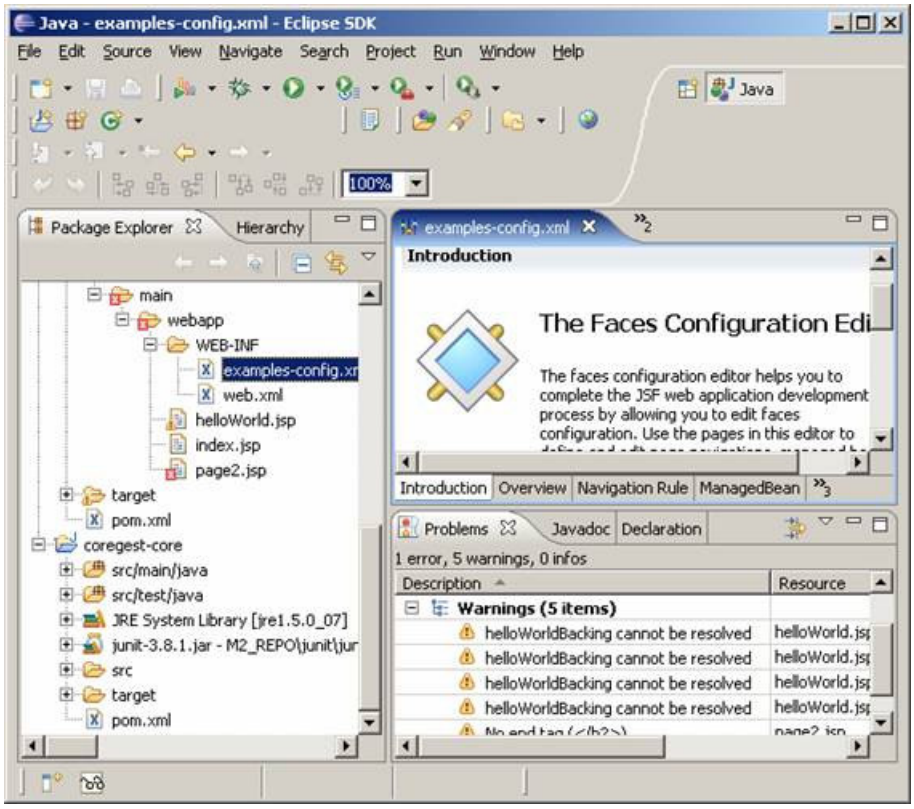
Pulsamos "Finish". A mí me ha salido este mensaje tan majo:



He pulsado "Yes", por si acaso, y me ha quedado una nueva configuración como sigue:



Ahora si pulso en el fichero "examples-config.xml" de mi proyecto Coregest, se lanza el editor de JSF



Conclusión

Sólo hemos empezado la aplicación. En el siguiente tutorial vamos a meterle Hibernate a la aplicación, pero no de cualquier manera, lo haremos al estilo Maven. Pronto empezaréis a "mavenizar" vuestros proyectos.

Poco a poco iremos avanzando en la construcción de una aplicación compleja con Maven.

SOME RIGHTS RESERVED This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 2.5 License.

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	Descripción
Pruebas de integración con Maven	Este tutorial nos muestra un ejemplo para lanzar las pruebas de integración "engañando" a Maven para que no se lanzen en la fase de test teniendo únicamente un módulo para ambas
Análisis de rendimiento (Profiling) de aplicaciones web con eclipse	En este tutorial se va a explicar como analizar el rendimiento de nuestras aplicaciones web con una herramienta propia de Eclipse, llamada Eclipse TPTP.
Manual Básico de Eclipse JET	Este tutorial nos introducirá en Java Emitter Template (JET) que nos sirve para generar automáticamente código fuente a partir de plantillas
Optimización Java con Eclipse Profiler Plugin	Alejandro Pérez nos enseña como analizar el rendimiento de nuestras aplicaciones con Eclipse Profiler Plugin.
Introducción a Eclipse 3.3 / Europa	En este tutorial se dará a conocer la nueva version de Eclipse, en la que se introducen nuevas funcionalidades, soporte para Java 6.
Crear un repositorio remoto y como hacer una 'release' con varios proyectos en Maven y Eclipse	En este tutorial vamos a explicar como podemos trabajar teniendo varios proyectos relacionados en Maven y en Eclipse
Java Server Faces con Eclipse	Este tutorial es un completo estudio de JSF, una moderna tecnología que nos permite desarrollar aplicaciones empresariales robustas.
Proyecto con JSF Myfaces, Maven y Eclipse	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.
Crear el sitio web de documentación del proyecto con Maven Site	En este tutorial vamos a ver cómo crear los sitios web de documentación del proyecto con Apache Maven.
Integración de Struts y eclipse	Alejandro Perez nos enseña como construir un entorno de alta eficiencia de desarrollo on Struts a través de plugins de eclipse

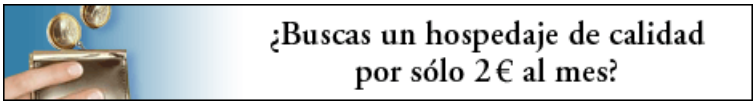
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

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 .... Hosting en Castellano con soporte Java/J2EE



www.AdictosAlTrabajo.com Opimizado 800X600