

Creación de servicios Web con Apache Axis2 desde una clase Java

En este ejercicio se creará un Servicio Web desde la plataforma de desarrollo Eclipse con el plugin Web Tool Project (WTP) utilizando el framework ApacheAxis 2. Además, como contenedor que de soporte a la ejecución de Apache Axis2 usaremos el contenedor Web Tomcat.

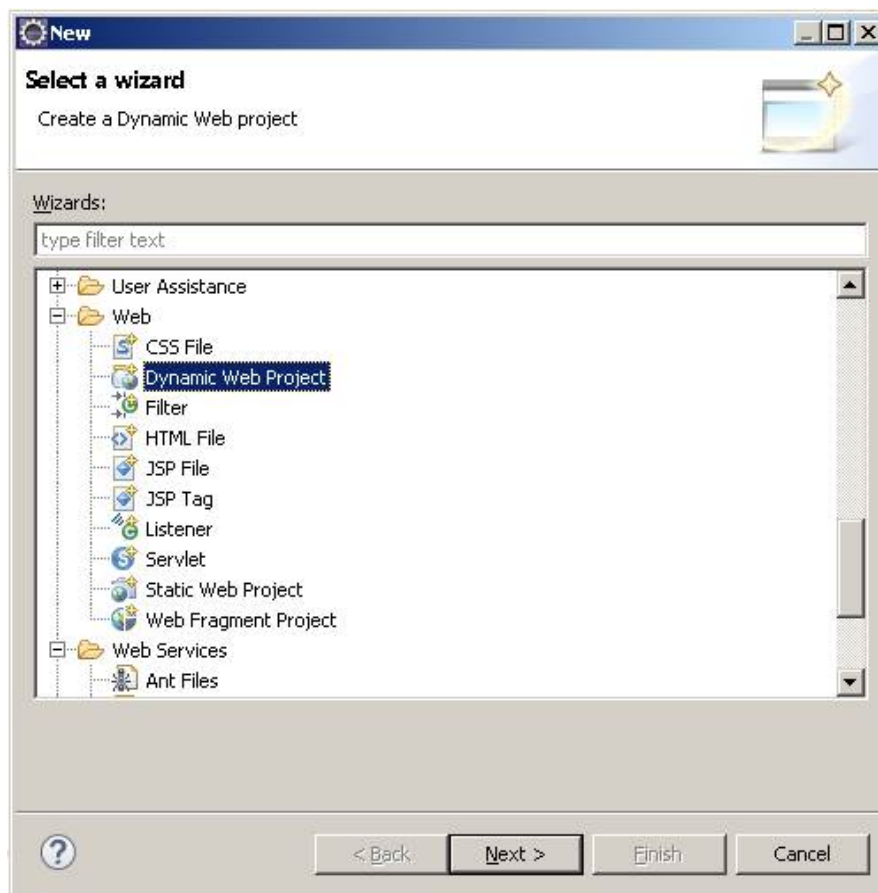
Prerrequisitos

1. Los mismos que en el ejercicio de la configuración de AXIS 2 en Eclipse.

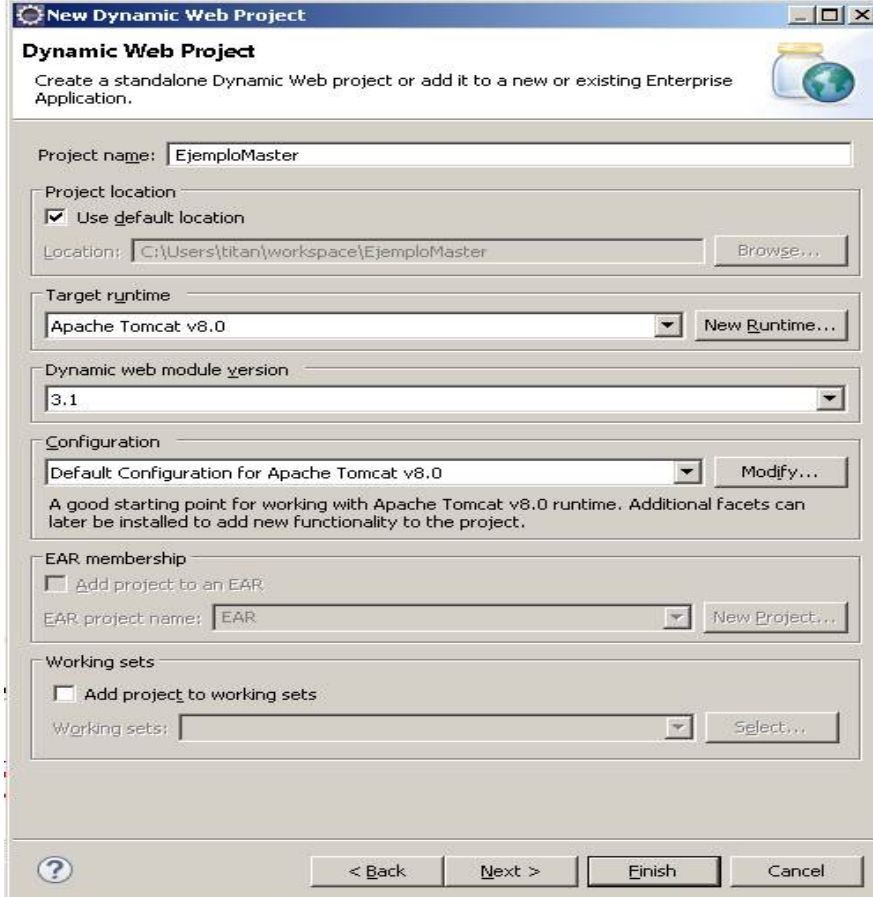
Pasos a seguir

1. Creación de proyecto Web dinámico

- 1.1. Iniciar Eclipse (si no ha sido ejecutado previamente)
- 1.2. Abrir *File* → *New* → *Other* → *Web* → *Dynamic Web Project*



- 1.3. Pulsar sobre *Next*
- 1.4. Completar los datos que se solicitan en el formulario (nombre del proyecto, entorno de ejecución, etc.)



New Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location:

☒ Use default location

Location:

Target runtime:

Dynamic web module version:

Configuration:

A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership:

☐ Add project to an EAR

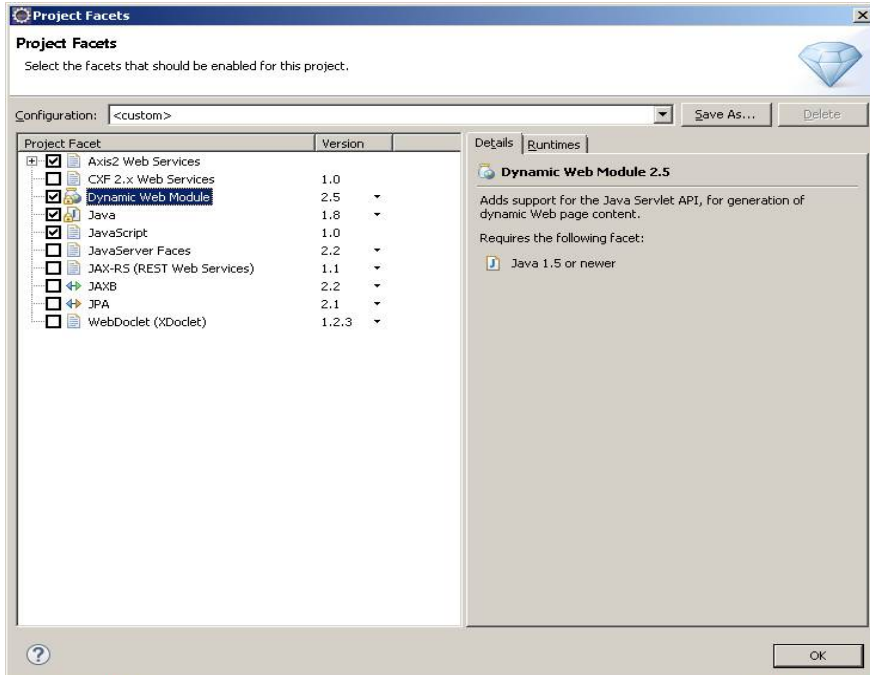
EAR project name:

Working sets:

☐ Add project to working sets

Working sets:

- 1.5. Pulsamos *Modify* y completamos los datos del siguiente formulario seleccionando la opción de Apache AXIS 2. Axis 2 Web Services Core 1.1 requires Dynamic Web Module necesitará alguna de las siguientes versiones para funcionar 2.2, 2.3, 2.4, 2,5.



Project Facets

Select the facets that should be enabled for this project.

Configuration:

Project Facet	Version
<input checked="" type="checkbox"/> Axis2 Web Services	
<input type="checkbox"/> CXF 2.x Web Services	1.0
<input checked="" type="checkbox"/> Dynamic Web Module	2.5
<input checked="" type="checkbox"/> Java	1.8
<input checked="" type="checkbox"/> JavaScript	1.0
<input type="checkbox"/> JavaServer Faces	2.2
<input type="checkbox"/> JAX-RS (REST Web Services)	1.1
<input checked="" type="checkbox"/> JAXB	2.2
<input type="checkbox"/> JPA	2.1
<input type="checkbox"/> WebDoclet (XDoclet)	1.2.3

Details | Runtimes

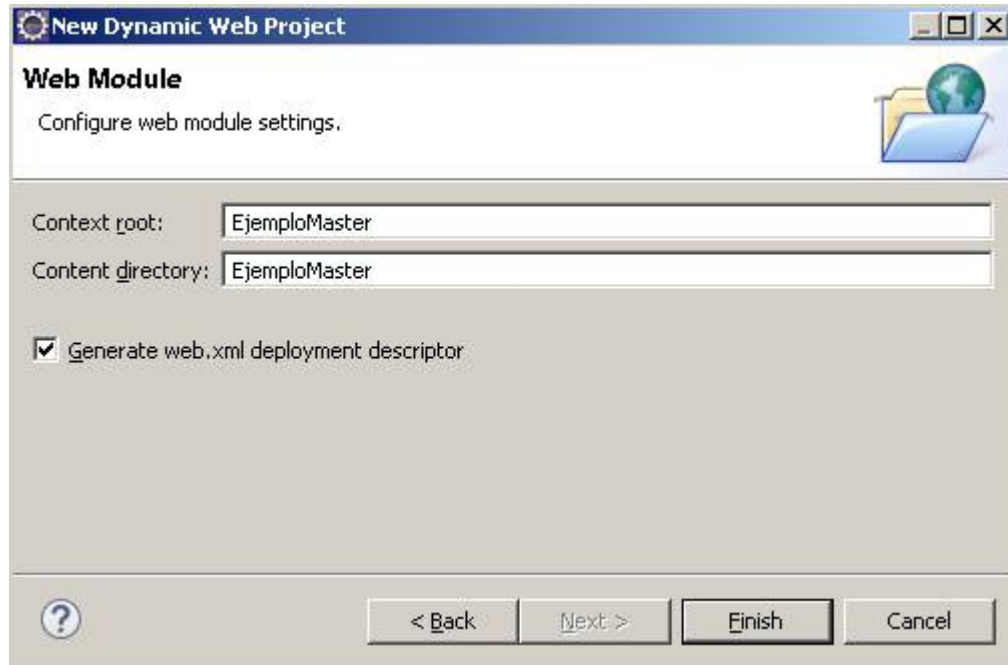
Dynamic Web Module 2.5

Adds support for the Java Servlet API, for generation of dynamic Web page content.

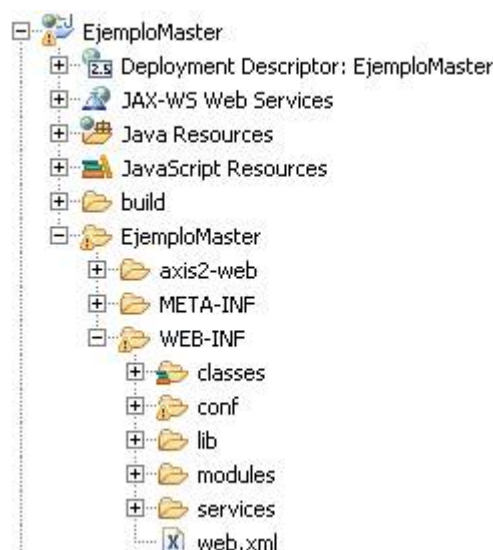
Requires the following facet:

- ☒ Java 1.5 or newer

- 1.6. Pulsamos *Ok* y para ser coherentes con el despliegue de la aplicación sobre el contendor Apache Tomcat cambiamos el nombre del *Content Directory* y en lugar de *WebContent* le daremos el nombre del contexto con el cual lo vayamos a desplegar posteriormente en el servidor Tomcat (por ejemplo, EjemploMaster).



- 1.7. Para concluir pulsamos *Finish*.



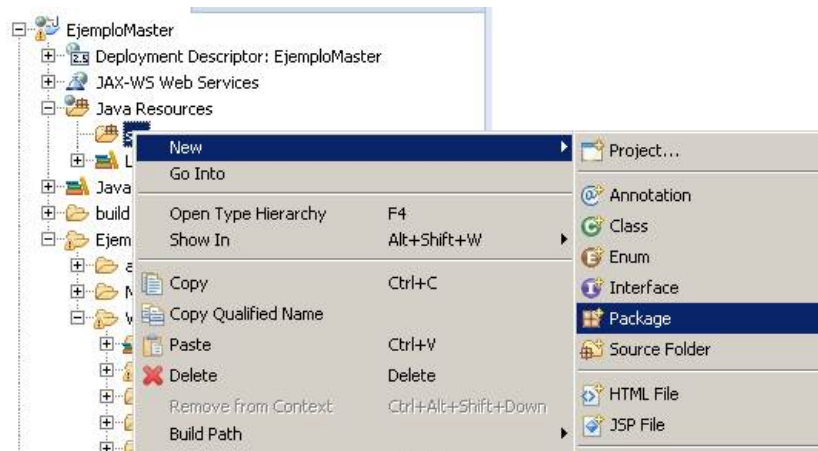
La estructura del proyecto se ha generado correctamente. Aunque aparezcan algunos avisos, no influyen sobre el funcionamiento de la aplicación.

2. Generar Servicio Web desde una clase Java

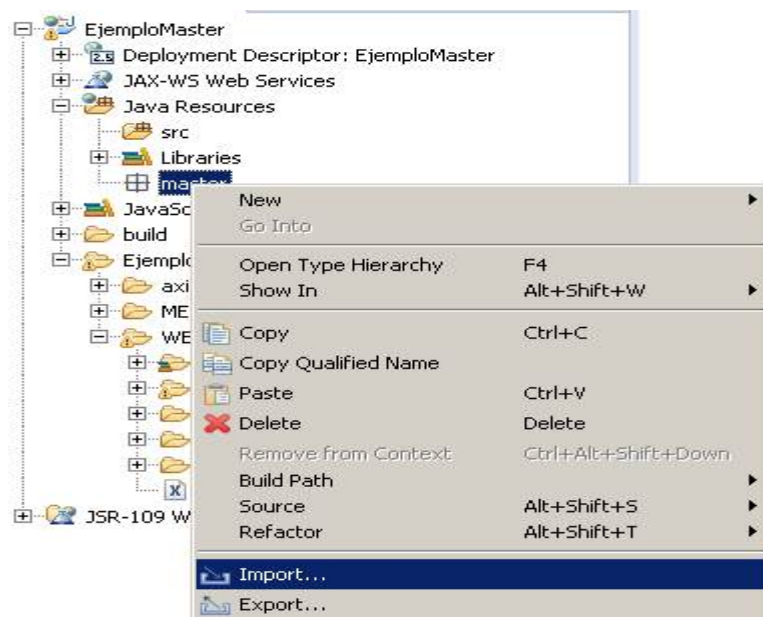
- 2.1. Importar la clase Java de ejemplo *converter.java* al proyecto creado anteriormente. Esta clase simplemente convierte de grados Celsius a Fahrenheit

y viceversa. Como ejercicio se puede añadir el convertir de los dos anteriores a Grados Centígrados y viceversa.

- Creamos primero un paquete llamado *master*.

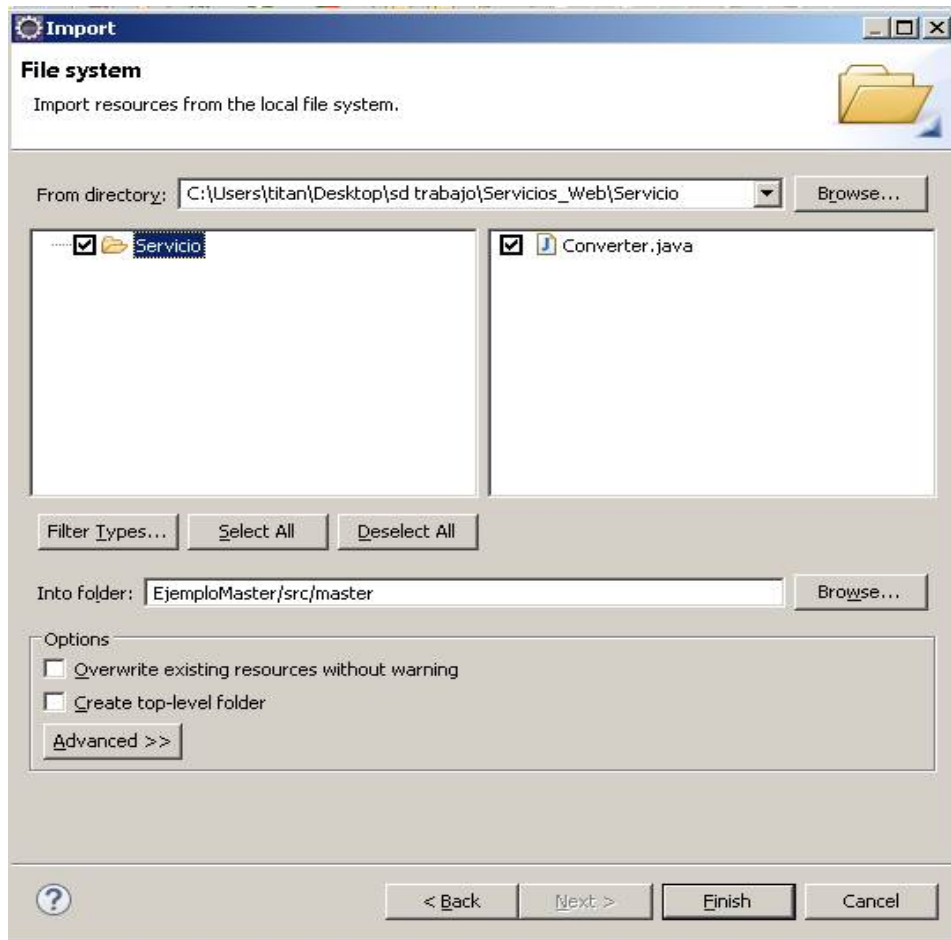


- Y posteriormente importamos la clase *Converter.java* al proyecto dentro del paquete. Para ello pulsamos sobre el paquete derecho con el botón derecho y seleccionamos la opción *import...*



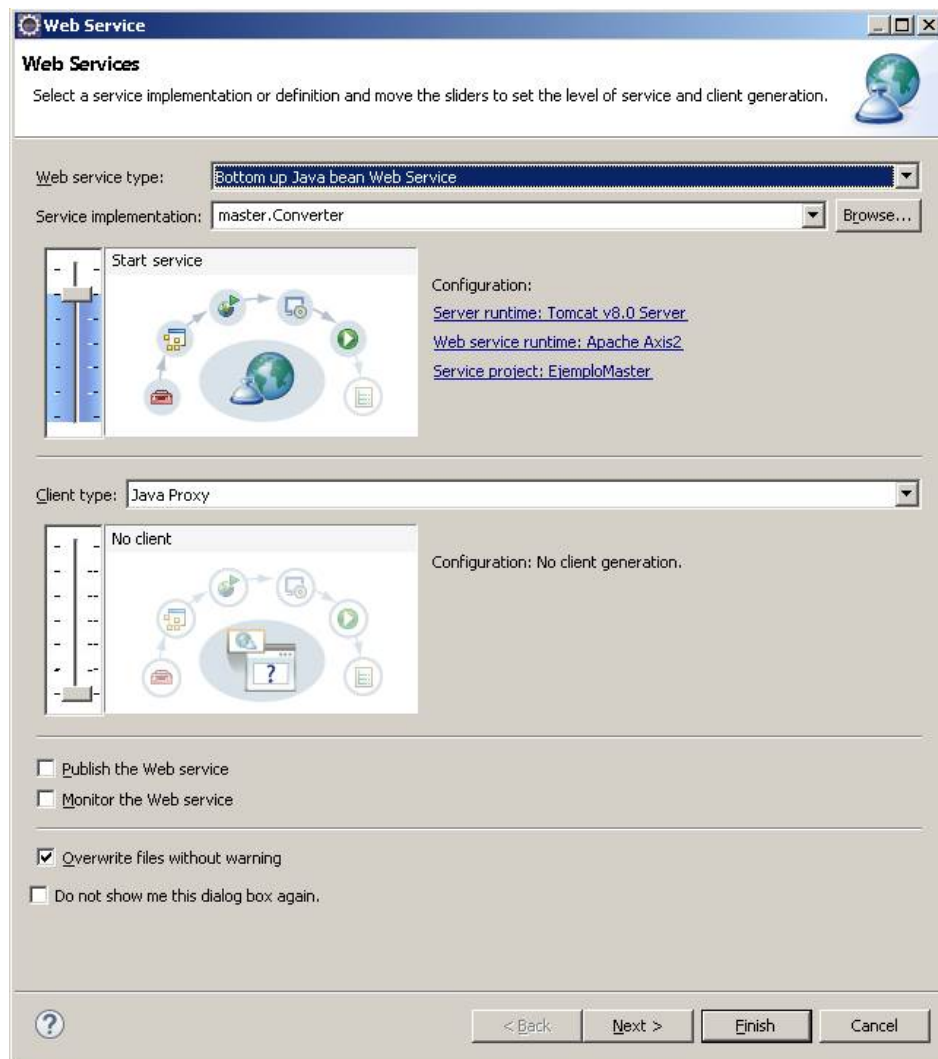
- Seleccionamos importar *General* → *File System* y elegimos el archivo que queremos importar *Converter.java* de donde lo hayamos ubicado.





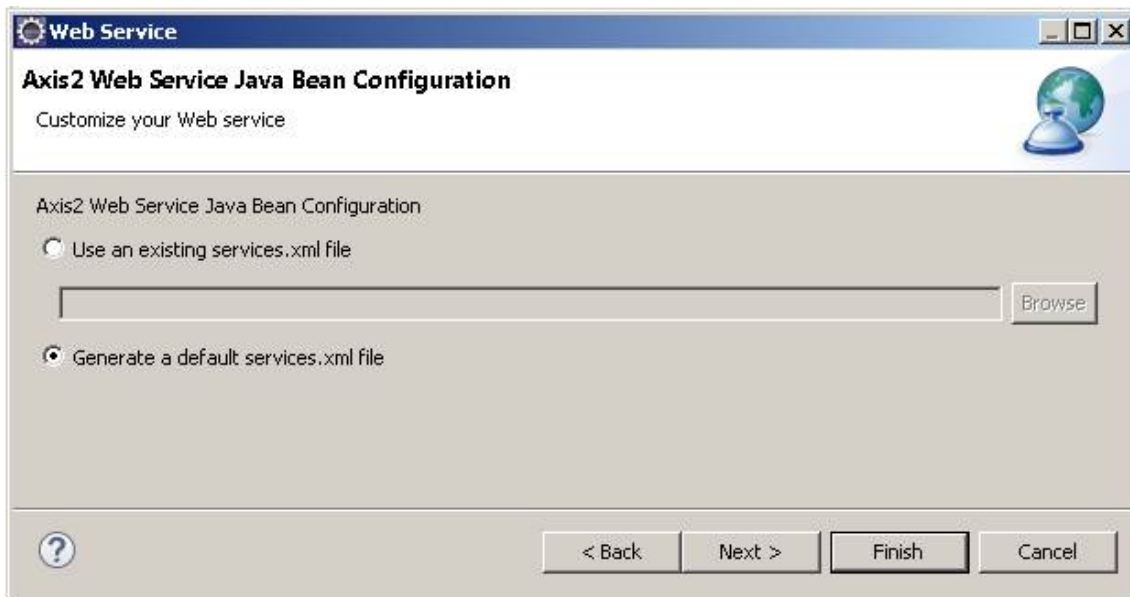
NOTA: Para utilizar el generador automático de servicios Web de Eclipse para AXIS se debe seguir la convención de nombrar los paquetes en minúsculas y las clases Java deben comenzar por mayúscula. Si no el generador no podrá trabajar pero tampoco nos indicará cuál es el error.

- 2.2. Seleccionado sobre la clase *Converter.java* y pulsando el botón derecho aparece una opción llamada *Web Service* → *Create Web Service*. Otra forma de hacerlo es desde *File* → *New* → *Other...* → *Web Services* → *Web service*. La primera opción selecciona la clase automáticamente, en la segunda se debería especificar.

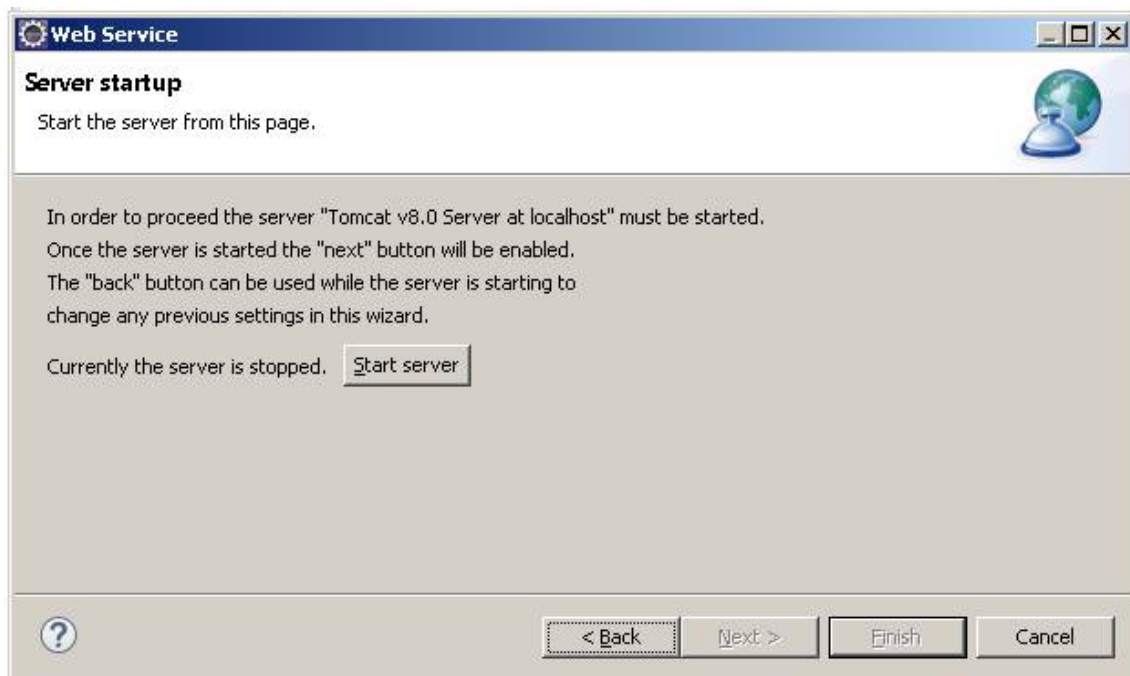


En el momento de generar el Servicio Web podemos elegir el nivel de generación del Servicio (Develop, Assemble, Deploy, Install, Start y Test). También nos permite generar un cliente dentro del proyecto para probar el servicio con los mismos niveles de generación que el servicio. En este caso el cliente lo genera dentro de un proyecto Web (para el ejemplo el cliente lo generaremos más adelante). Podríamos modificar los entornos de ejecución que aparecen a la derecha, simplemente pulsando sobre el enlace y seleccionando el adecuado.

2.3. Pulsamos en *Next >*.



2.4. Vamos avanzando con las opciones que aparecen por defecto. Hasta que se nos indique que arranquemos el servidor Apache Tomcat. Esta opción aparece porque hemos indicado como opción despliegue y prueba. Eclipse lanza una instancia nueva de Apache Tomcat, por lo que **si tenemos arrancado el servidor previamente debemos pararlo**. Posteriormente pulsamos *Start Server*.



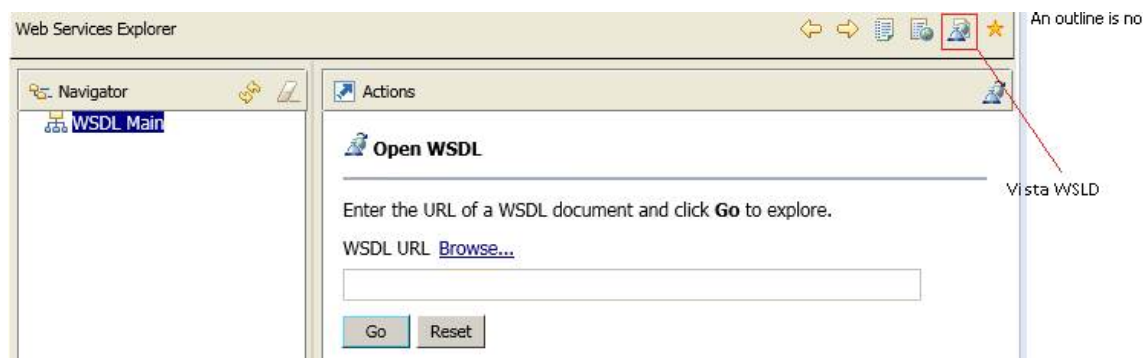
2.5. Una vez arrancado el servidor pulsaremos *Finish*.

3. Validación del Servicio Web

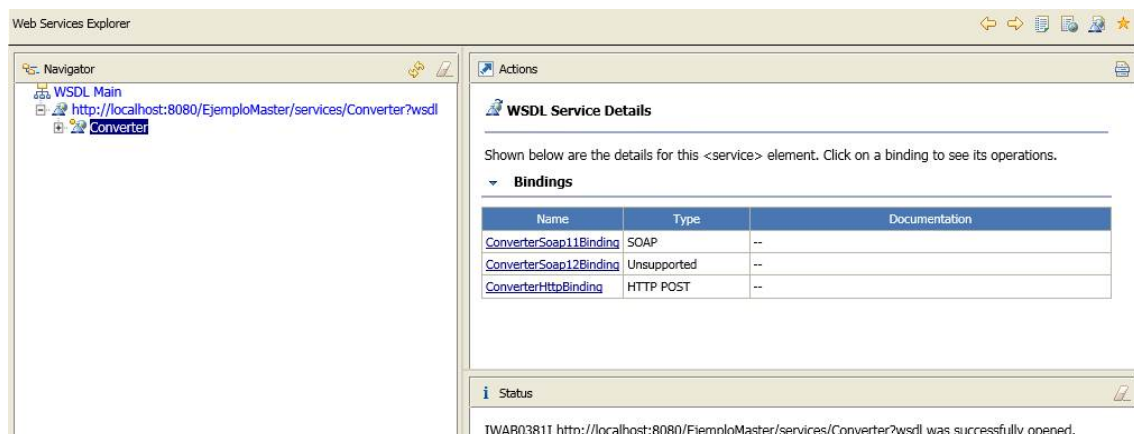
3.1. Lanzamos el explorador de Servicios Web situado en el menú de iconos en la parte superior.



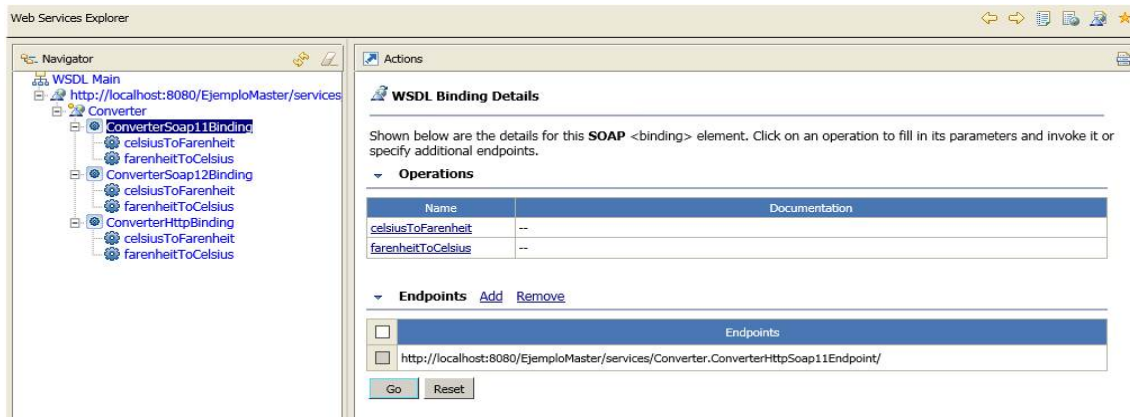
3.2. Seleccionamos la vista WSDL.



3.3. Introducimos la URL que apunta a la WSDL del servicio, en este caso <http://localhost:8080/EjemploMaster/services/Converter?wsdl>, y clicamos GO.



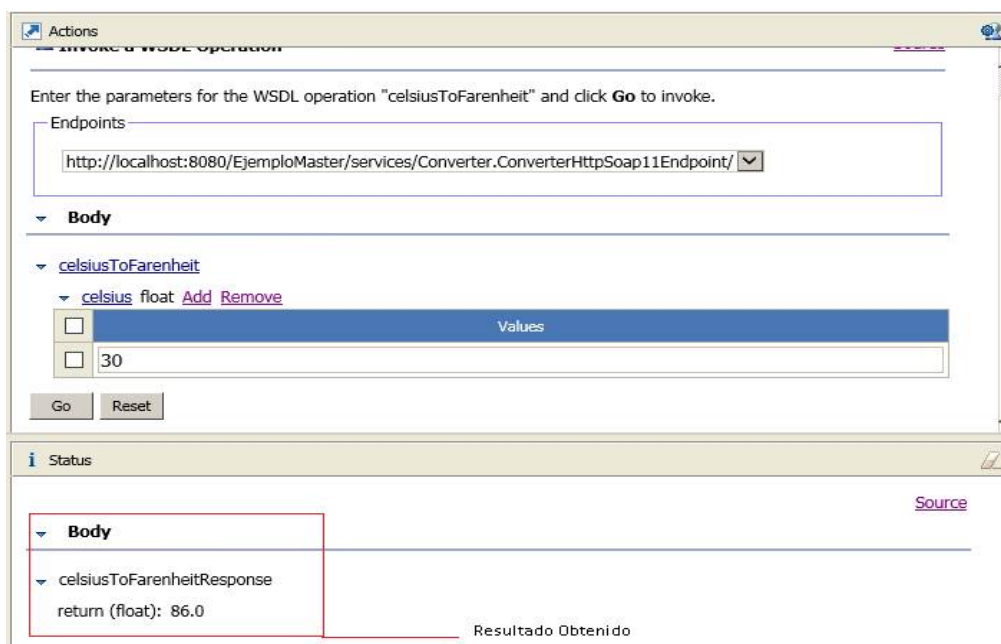
3.4. Seleccionando el enlace ConverterSoap11Binding que accede a las operaciones del Servicio.



3.5. Invoco a una de las operaciones introduciendo los parámetros de entrada pulsando en la opción Add.



3.6. Hago click en Go y obtengo el resultado.



Si clicamos sobre la Source podremos observar los elementos SOAP Request Envelope y SOAP Response Envelope.

▼ SOAP Request Envelope:

30

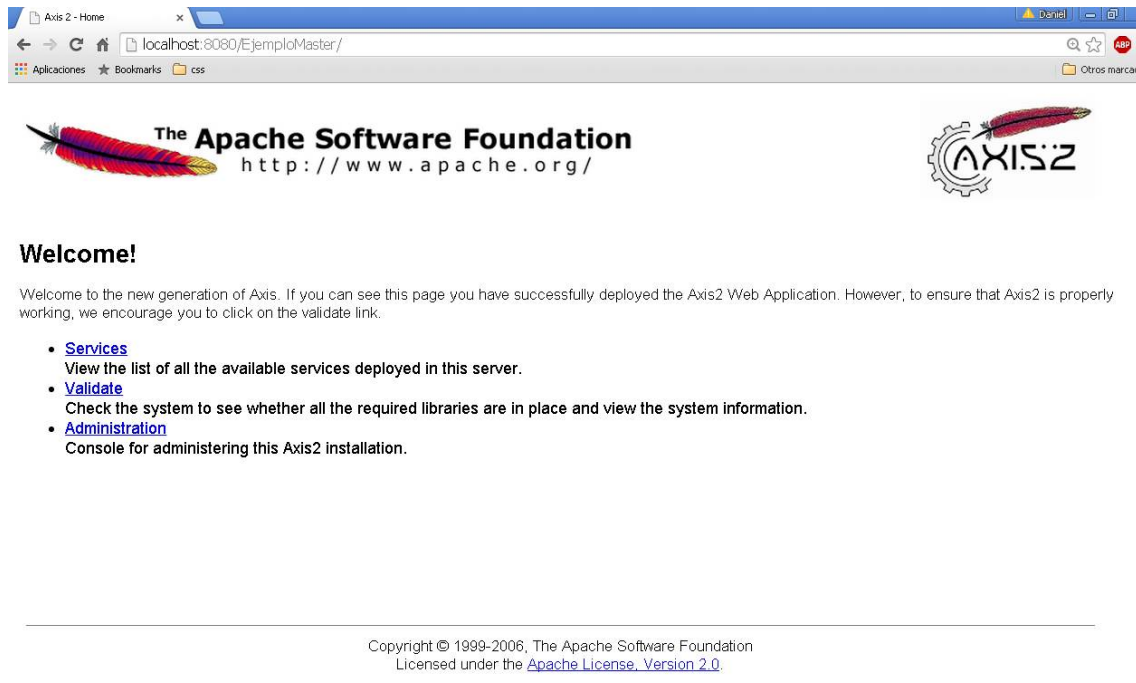
▼ SOAP Response Envelope:


86.0

4. Despliegue del Servicio Web en el servidor Apache Tomcat externo.

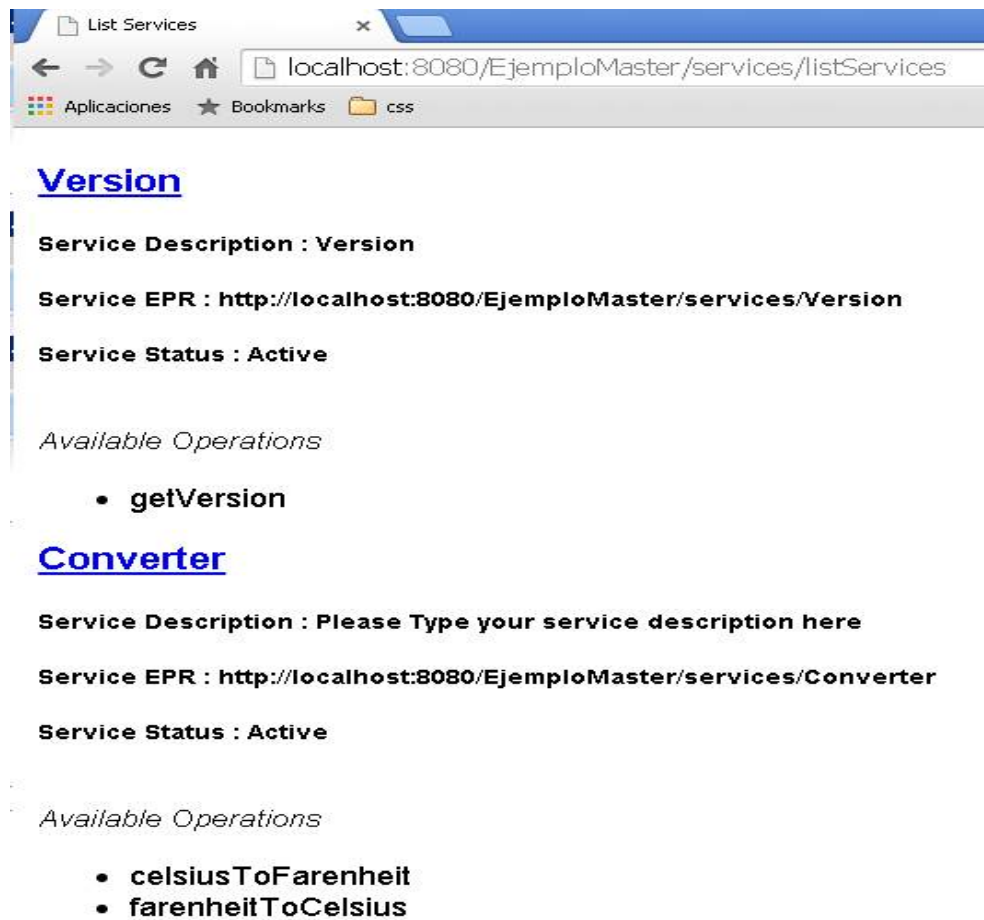
Para desplegar el Servicio Web en la distribución de apache Tomcat que tengamos instalada, sin que se lance desde el entorno Eclipse se deben seguir los siguientes pasos.

- 4.1. Crearemos una carpeta en el directorio *webapps* del servidor Apache Tomcat con el nombre del contexto que se considere apropiado (como comentamos anteriormente por ejemplo, EjemploMaster).
- 4.2. Copiaremos en esta carpeta el contenido de la carpeta *webapps* (o en su defecto, si le hemos cambiado el nombre EjemploMaster) generada en el proyecto.
- 4.3. Iniciamos el servidor Apache Tomcat y una vez arrancado comprobamos si el servicio ha sido desplegado. Para ello introducimos en el navegador la siguiente dirección URL: <http://localhost:8080/EjemploMaster/> .



También se puede acceder desde el navegador de eclipse, para ello se debe clicar el siguiente icono  e introducir la URL anterior.

- 4.4. Accedemos al enlace *Services* para ver la lista de servicios y comprobar que el servicio *Converter* se encuentra disponibles, como se muestra en la imagen inferior.



- 4.5. Pulsando sobre el nombre del Servicio accederemos al documento WSDL que describe el servicio.

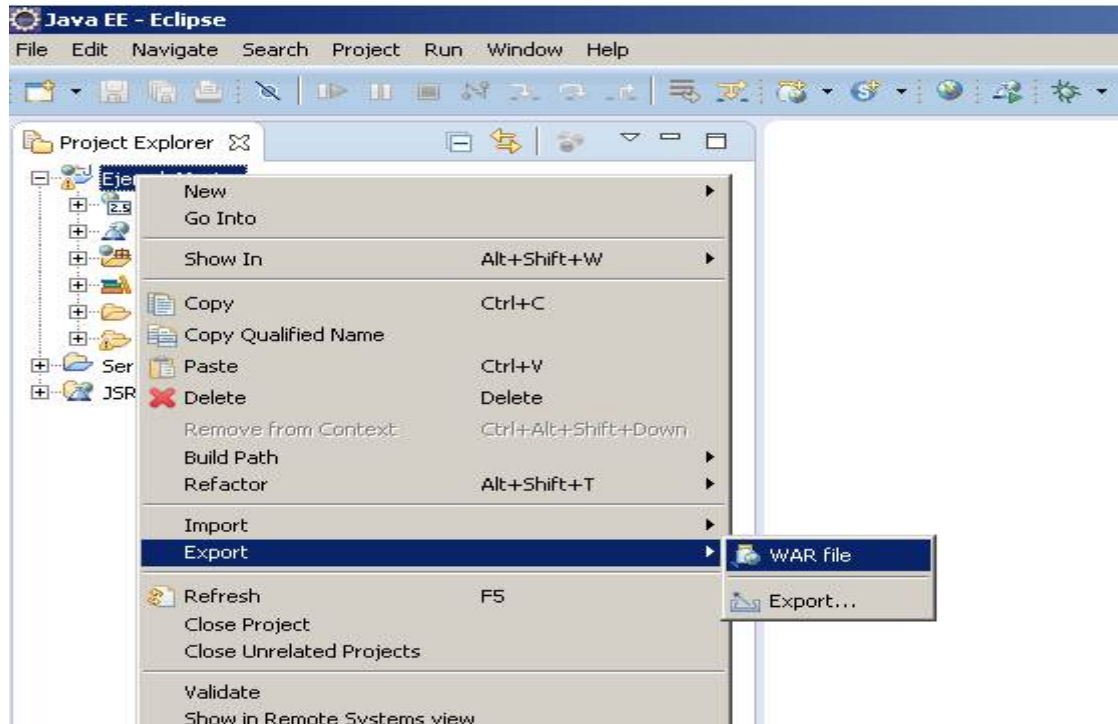
```
<?xml version='1.0' encoding='UTF-8' ?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:ns="http://master"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" targetNamespace="http://master">
  <wsdl:documentation>Please Type your service description here</wsdl:documentation>
  <wsdl:types>...</wsdl:types>
  <wsdl:message name="celsiusToFarenheitRequest">...</wsdl:message>
  <wsdl:message name="celsiusToFarenheitResponse">...</wsdl:message>
  <wsdl:message name="farenheitToCelsiusRequest">...</wsdl:message>
  <wsdl:message name="farenheitToCelsiusResponse">...</wsdl:message>
  <wsdl:portType name="ConverterPortType">...</wsdl:portType>
  <wsdl:binding name="ConverterSoap11Binding" type="ns:ConverterPortType">...</wsdl:binding>
  <wsdl:binding name="ConverterSoap12Binding" type="ns:ConverterPortType">...</wsdl:binding>
  <wsdl:binding name="ConverterHttpBinding" type="ns:ConverterPortType">...</wsdl:binding>
  <wsdl:service name="Converter">...</wsdl:service>
</wsdl:definitions>
```

NOTA: Se ha configurado Eclipse para poder trabajar con Apache AXIS 2, pero por defecto Eclipse WTP incluye Apache AXIS. En principio, y como entrenamiento se puede utilizar Apache AXIS para generar los servicios Web, aunque el rendimiento en apache AXIS 2 se haya mejorado y se hayan añadido algunas características de WS-*.

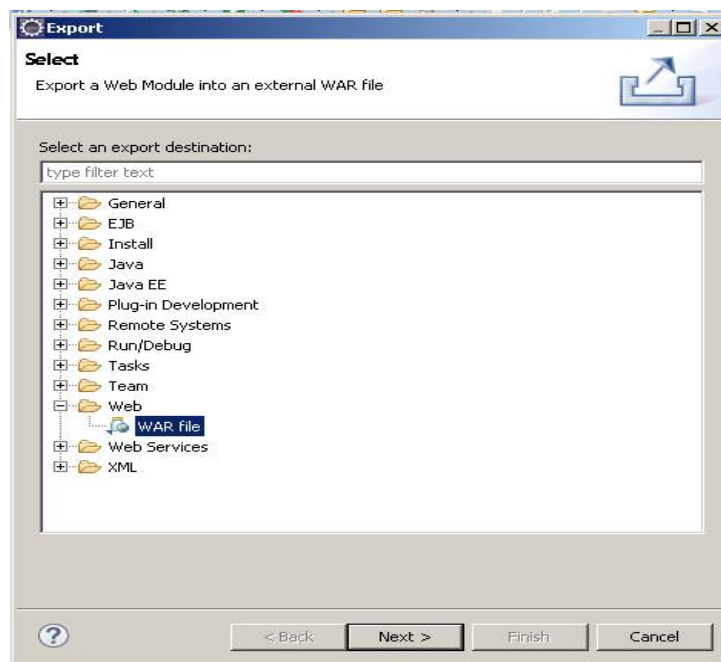
5. Despliegue del Servicio Web en el servidor Apache Tomcat de producción.

En entornos de producción directamente podemos desplegar la aplicación mediante un paquete WAR.

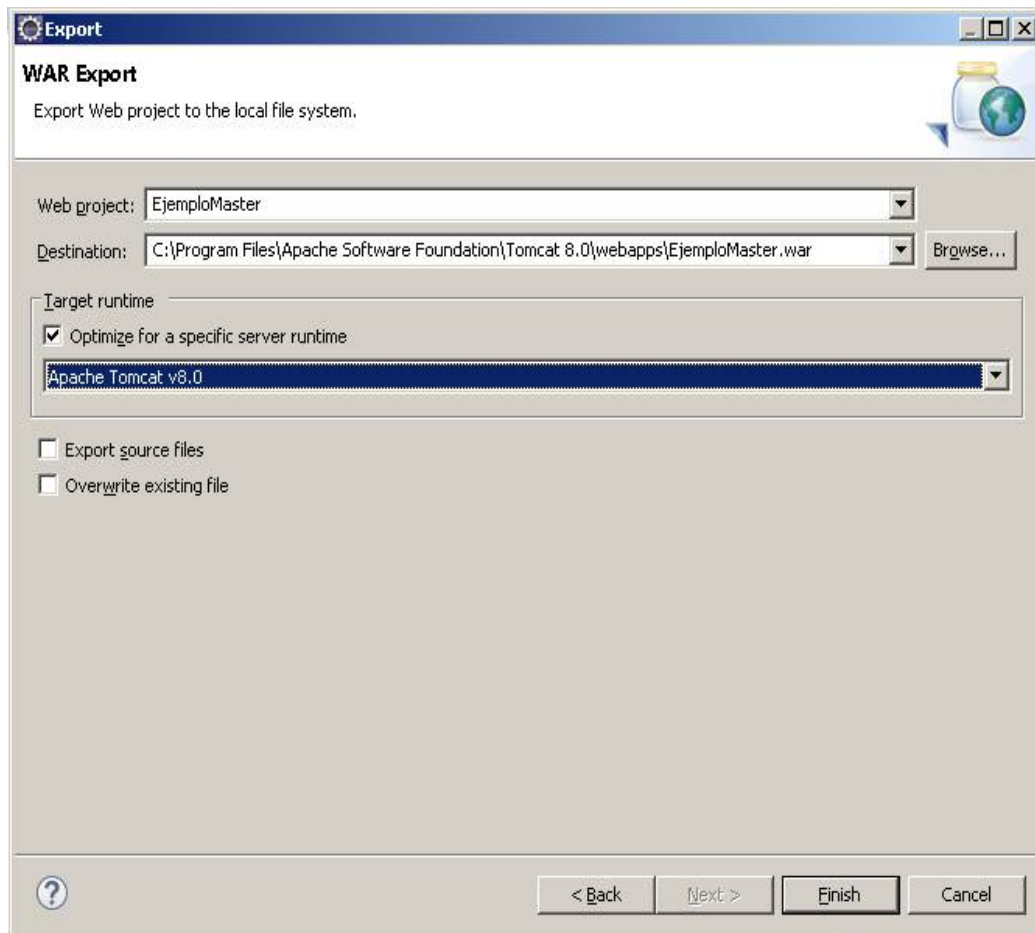
5.1. Pulsando en con el botón derecho sobre la carpeta EjemploMaster seleccionamos la opción de exportar → WAR File.



5.2. En el caso de que no se desplegara de forma automática la exportación de tipo WAR File, podremos seleccionar la opción *WAR file* en la carpeta *web*



5.3. Seleccionamos la localización del servidor Apache Tomcat donde desplegaremos la aplicación y pulsamos *Finish*. Recuerda que el destino tiene que acabar en .war.



5.4. Ya podemos acceder al servicio desplegado.