

## UD1 - Práctica 2. Comenzamos con aplicaciones web

Para esta actividad, vamos a crear nuestra primera aplicación web. Seguiremos usando XAMPP, ya que nos proporciona un servidor de aplicaciones como Tomcat, que nos va a ser útil, ya que será quien aloje nuestra aplicación. Para este primer proyecto no esperamos construir algo muy grande, por lo que trataremos con servlets.

Un servlet es un componente basado en Java que se utiliza en el desarrollo web para manejar el procesamiento de datos del lado del servidor HTTP y generar contenido web dinámico. Los servlets suelen formar parte de aplicaciones web basadas en Java y se ejecutan en un servidor web, como nuestro querido Tomcat (o Jetty).

Para crear nuestro servlet, necesitaremos instalar tanto JRE como Eclipse. Los enlaces de descarga se pueden encontrar [aquí](#) y [aquí](#). Una vez que tengamos todo listo, podemos volver a XAMPP y poner en marcha nuestro servidor Tomcat (figura 1).

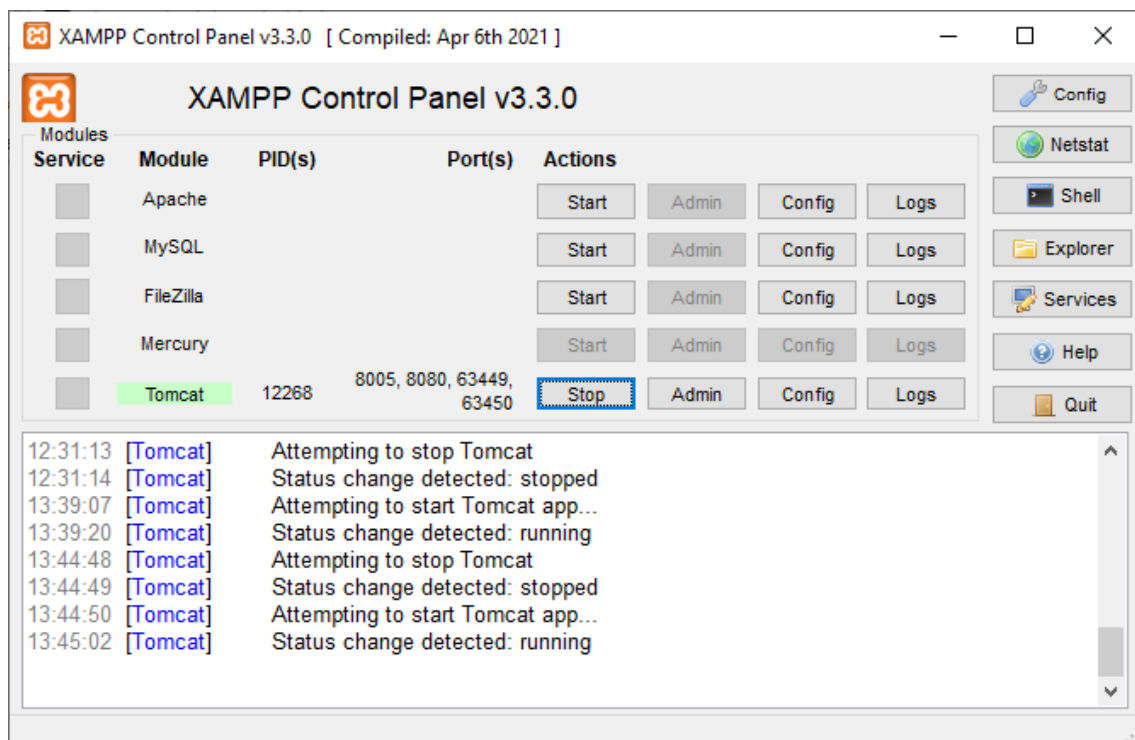


Figure 1. Levantamos Tomcat

Ahora podemos iniciar Eclipse y en primer lugar, debemos definir nuestro servidor de aplicaciones haciendo clic en Servers / No servers available, Click this link to create a new server..., que se puede ubicar en la parte inferior de la pantalla. A continuación, podremos seleccionar nuestro servidor (figura 2).

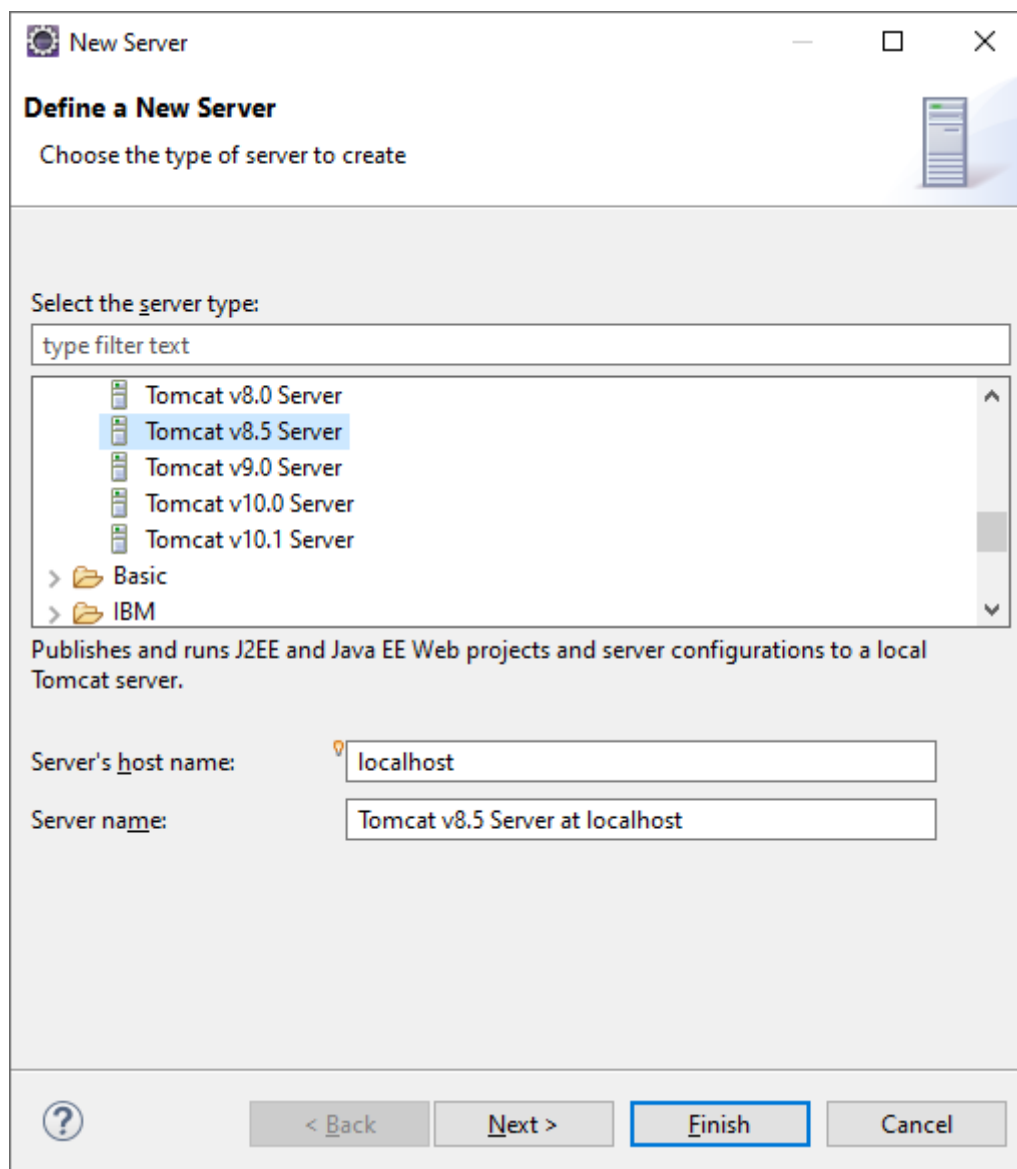


Figure 2. Configurando Tomcat en Eclipse

Antes de hacer clic en Finish, no olvides presionar Siguiente y proporcionar el directorio de instalación de Tomcat (generalmente C:\xampp\tomcat) y tu versión de JRE (que puedes encontrar en la lista desplegable). Una vez que hayas terminado, Tomcat estará listo para funcionar (figura 3).

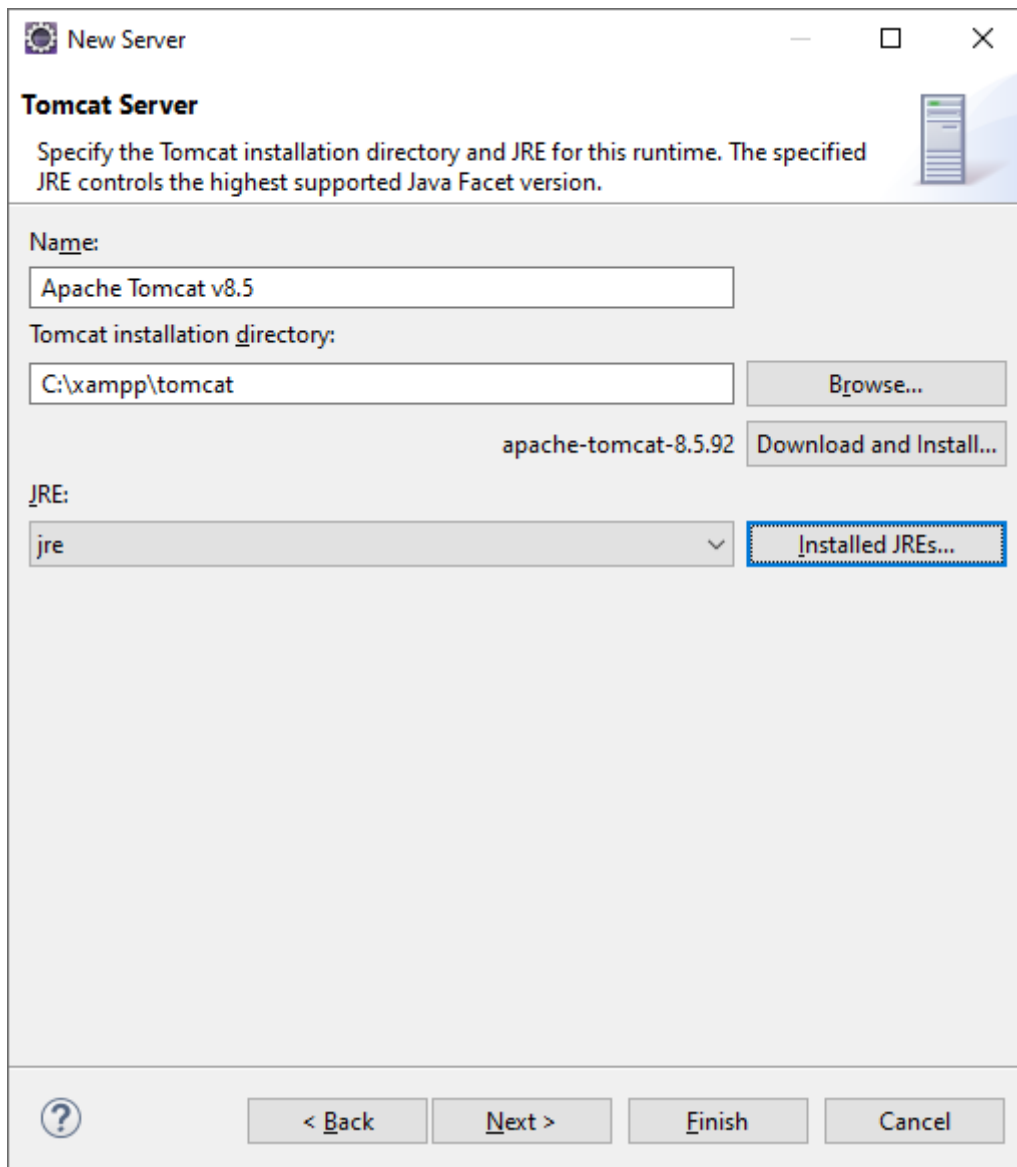
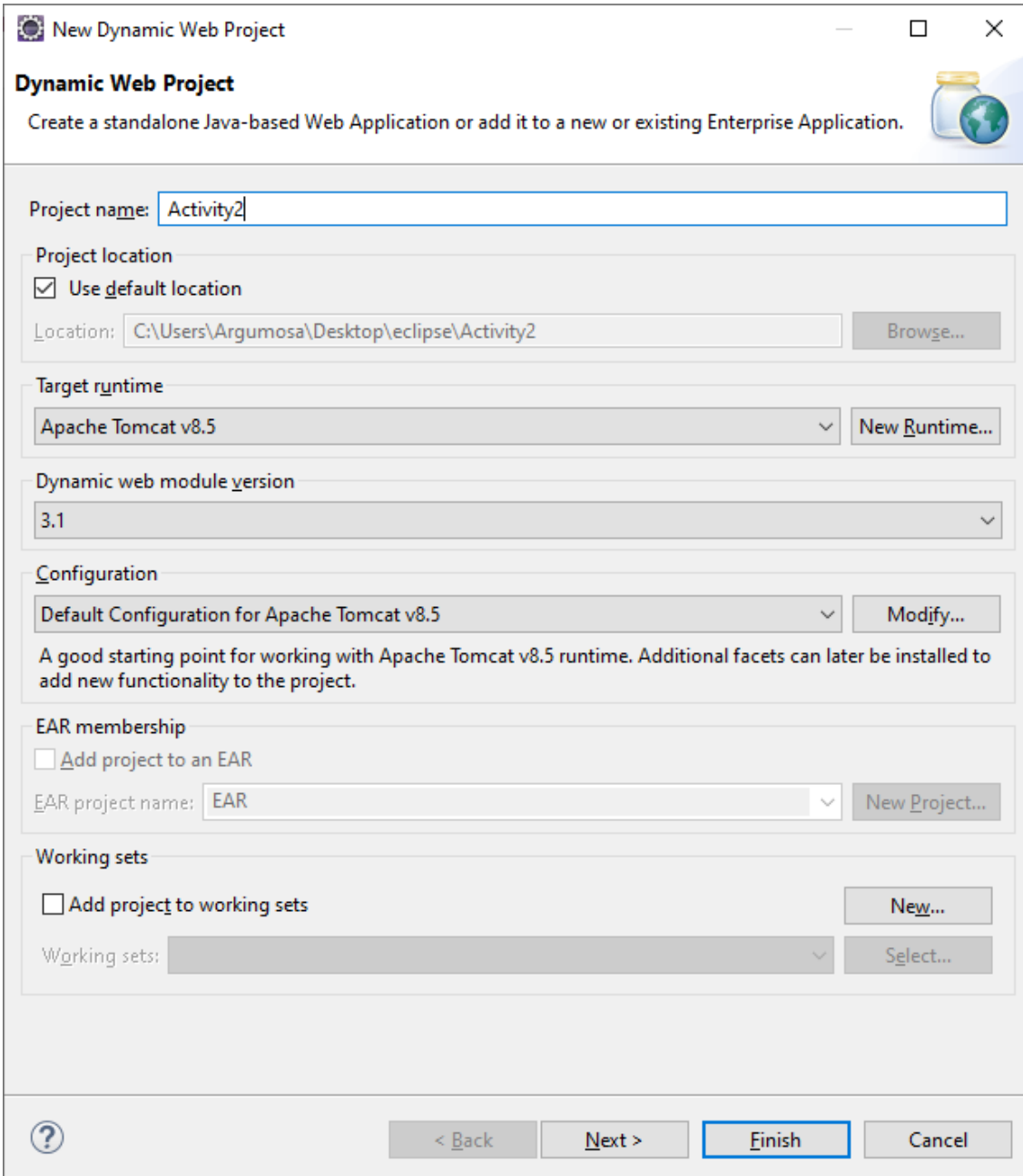


Figure 3. Últimas configuraciones de Tomcat en Eclipse

Ahora podemos crear un nuevo proyecto usando Eclipse. Vamos a File/New/Dynamic Web Project y veremos una pantalla emergente (figura 4) donde podemos ajustar los detalles del proyecto de la siguiente manera:

- **Project name:** algo relacionado, como por ejemplo "Actividad2"
- **Target runtime:** Tenemos que asegurarnos de que estamos desarrollando para nuestro servidor (Apache Tomcat v8.5)
- **Dynamic web module version:** 3.1
- **Configuration:** default for Apache Tomcat v8.5



**New Dynamic Web Project**

**Dynamic Web Project**  
Create a standalone Java-based Web Application 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 v8.5 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:

Figure 4. Creando un nuevo proyecto

Después de esto, podemos crear nuestro primer servlet, que nos permitirá desarrollar nuestro programa. Podemos ir a New/Servlet donde aparecerá una nueva pantalla (figura 5). Necesitamos proporcionar un nombre de clase, como por ejemplo “MyServlet”.

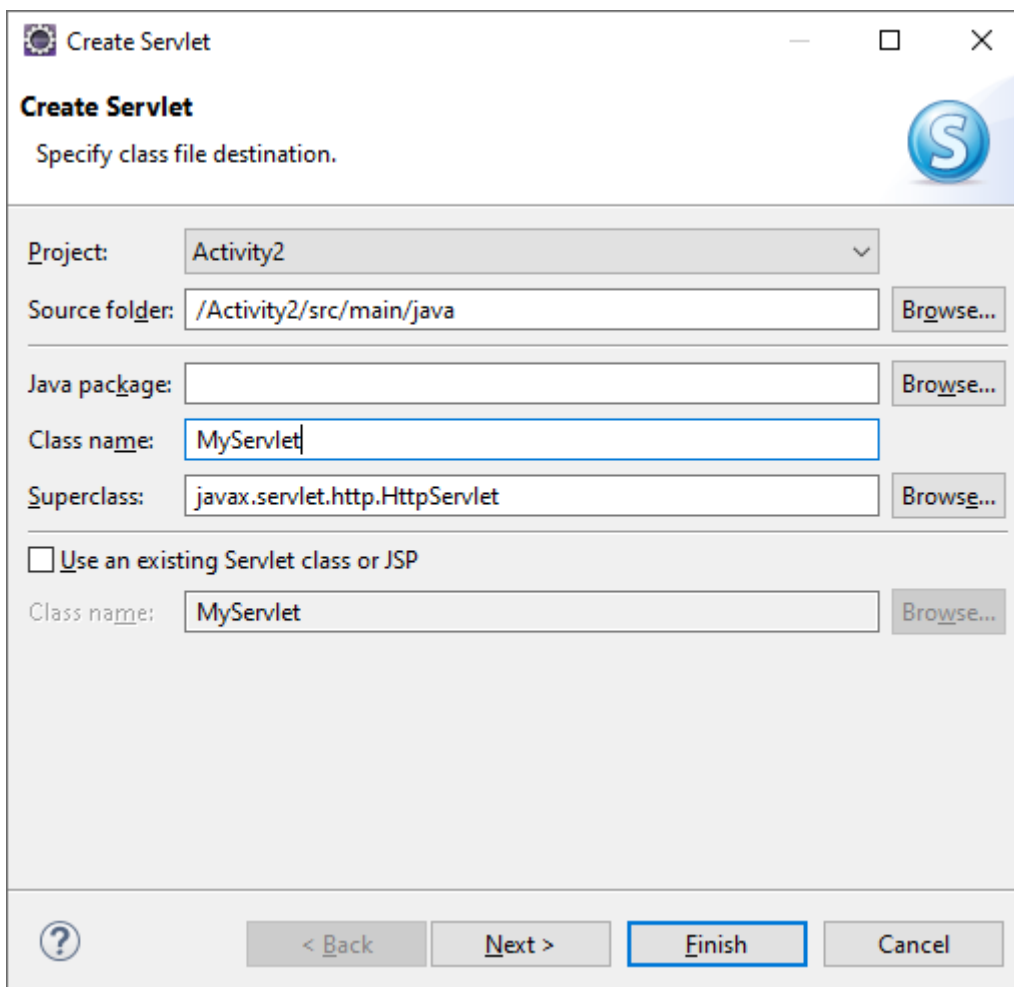


Figure 5. Creando un nuevo servlet

Ahora podemos comenzar a editar nuestro servlet. Veremos 2 métodos ya creados:

- **doGet** este método se utiliza para gestionar peticiones HTTP GET, normalmente se utiliza para recuperar información

**doPost** este método se utiliza para gestionar peticiones HTTP POST, Se utiliza habitualmente para procesar envíos de datos, como datos de formularios

Ahora, encontrarás un código corto que puedes usar para probar estos métodos rápidamente:

```

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class MyServlet
 */
@WebServlet("/MyServlet")
public class MyServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public MyServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Message from doGet method from MyServlet</h1>");

    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
    HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Message from doPost method from MyServlet</h1>");

    }

}

```

Después de hacer esto, podemos crear un JSP (Java Server Pages) para proporcionar una interfaz gráfica de usuario (GUI) para nuestra aplicación. Simplemente haz clic con el botón derecho en la carpeta webapp y presiona New / JSP. Veamos un código de prueba para este JSP:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.Date"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Activity 3</title>
</head>
<body>

    <p>Hello from the JSP file!</p>

    <form action="MyServlet" method="GET">
        <button type="submit">Get message</button>
    </form>
    <br>
    <form action="MyServlet" method="POST">
        <button type="submit">Post message</button>
    </form>

</body>
</html>
```

Cuando hayamos terminado, podemos exportar nuestro proyecto como un archivo WAR (figura 6). Un fichero WAR (Web Application Archive) es un formato estándar utilizado para empaquetar e implementar aplicaciones web. Es similar en concepto a un JAR (Java Archive) pero está diseñado específicamente para aplicaciones web. Después de seleccionar WAR, necesitamos proporcionar un destino válido (could be our Tomcat folder: C:\xampp\tomcat\webapps).

Si todo está correcto, veremos nuestro servlet funcionando al acceder a localhost:8080/Activity2. De lo contrario, es posible que debamos verificar problemas de compatibilidad entre la versión de Java de Eclipse y nuestra versión de JRE Java. Para solucionar rápidamente muchos de los problemas, podemos ir a nuestro explorador de proyectos, hacer clic con el botón derecho en nuestro proyecto y luego en propiedades. Debería aparecer una ventana donde seleccionaremos "Java Compiler" y luego estableceremos Java compliance al nivel 1.8. Esto debería solucionar todos los problemas de compatibilidad (figura 7).

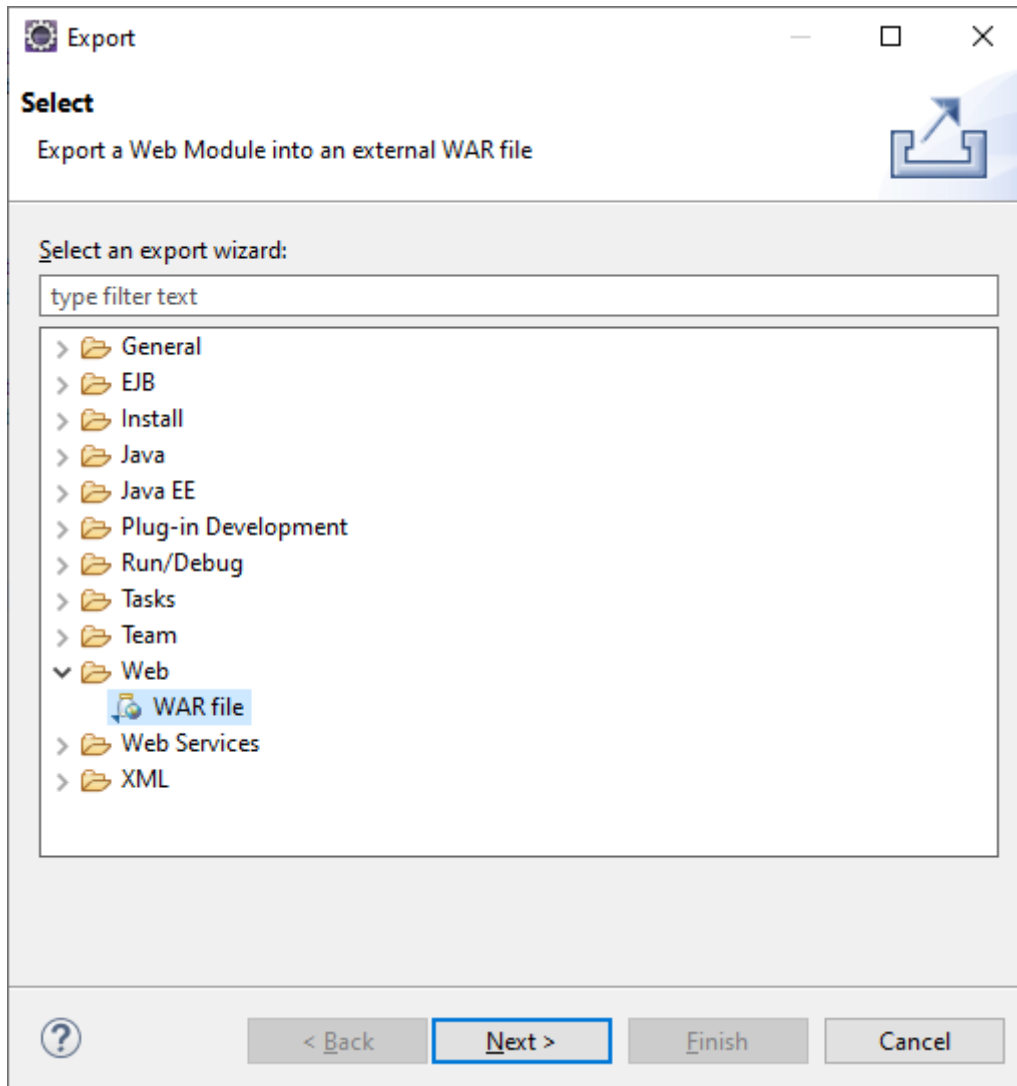


Figura 6. Exportando un fichero WAR

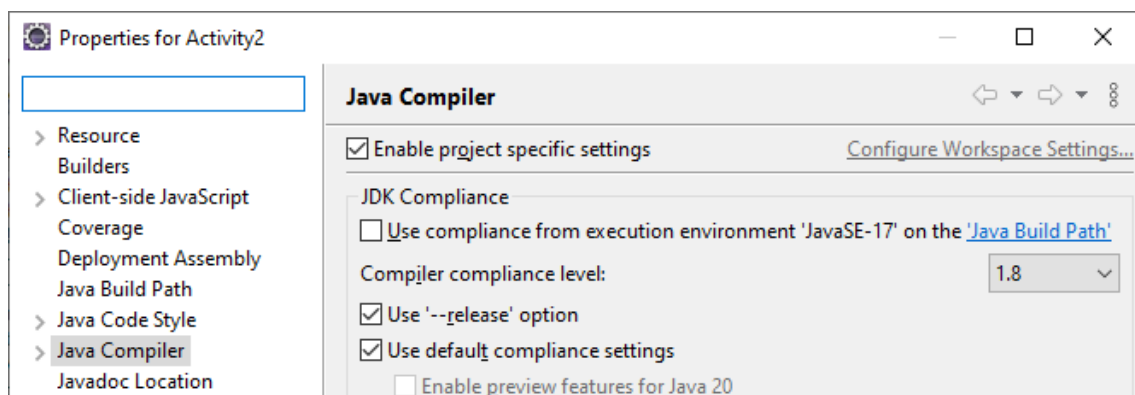


Figura 7. Solucionando problemas de compatibilidad



### ¿Qué hacer ahora?

1. Crea un programa simple que proporcione cierto nivel de interacción con el usuario final. Aquí tienes algunos ejemplos:
  - a. Calculadora de letra del DNI
  - b. Comprobador de números primos
  - c. Generador de números aleatorios entre 2 números dados
2. Crea una interfaz de usuario agradable para tu desarrollo web
3. Crea un informe en el que expliques el desarrollo de tu trabajo. Debe incluir, al menos, los siguientes elementos:
  - a. Portada
  - b. Tabla de contenidos
  - c. Capturas de pantalla y explicaciones sobre el funcionamiento del programa
4. Investiga sobre los siguientes temas y proporciona una respuesta corta (no más de 50 palabras cada uno):
  - a. ¿Cuál fue la parte más compleja de esta actividad? ¿Cómo lograste resolverla?
  - b. ¿Puedes encontrar otro servidor de aplicaciones y compararlo con Tomcat?

Entrega tu informe en nuestro Moodle en un ZIP junto a tu proyecto de Eclipse.

**SOLO SE ACEPTARÁN ARCHIVOS PDF. CUALQUIER .DOCX, .DOC U OTROS  
NO SERÁN EVALUADOS**