

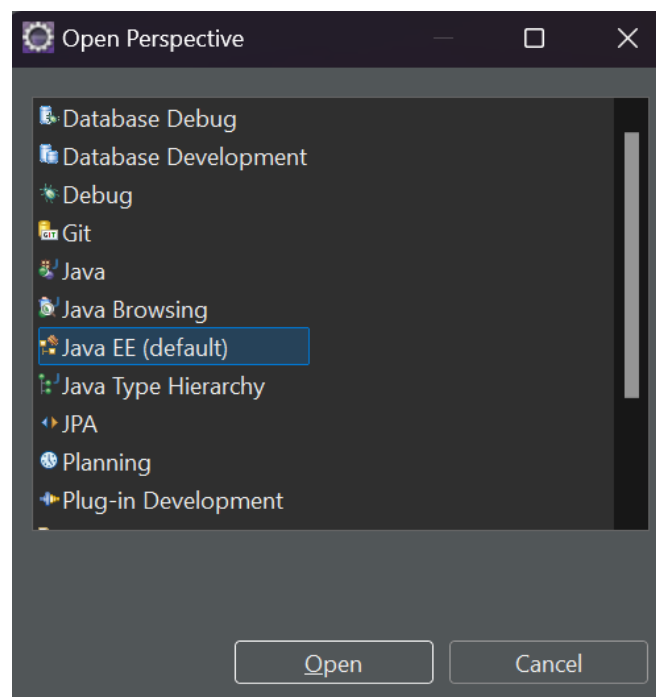
DAW Práctica 3.4: Despliegue de una aplicación con Apache Tomcat

En esta actividad vamos a realizar un despliegue de una aplicación básica con Apache Tomcat.

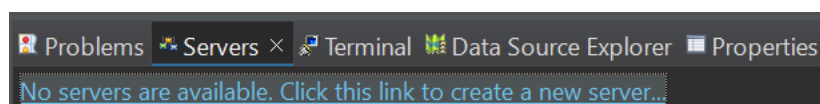
Procedimiento:

Instrucciones:

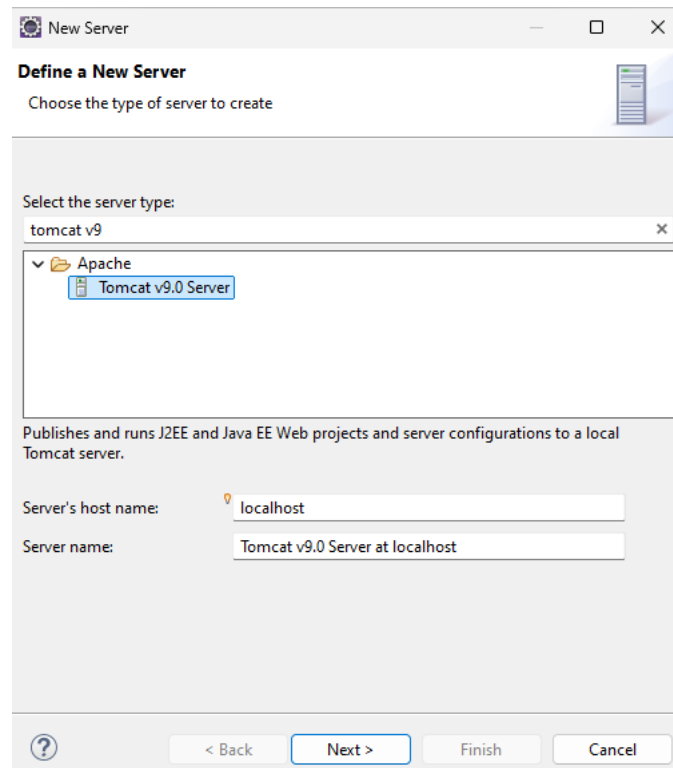
1. Si no lo tienes en tu equipo descarga el IDE Eclipse escogiendo la opción “Eclipse IDE for Enterprise Java and Web Developers”.
2. Integraremos Eclipse con Tomcat, para ello:
 - a. Por defecto aparece la perspectiva de Java EE (herramientas para el desarrollo de aplicaciones). En caso de no tenerla debemos entrar en el menú en Windows->Perspective->Open Perspective->Other y buscamos Java EE.



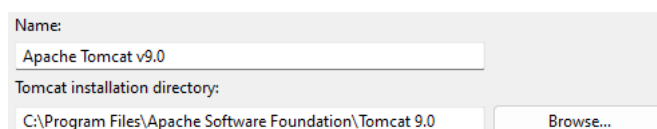
- b. En la parte inferior, en el menú **Window->Show View** haz click en la opción Servers.



- c. Añadir un servidor nuevo. Vamos a escoger Apache Tomcat 9. **Observa los servidores de aplicación que nos permite instalar.**



- d. Especifica el nombre que quieres dar al servidor, selecciona el directorio de instalación de Tomcat 9 (C:\Program Files\Apache Software Foundation\Tomcat 9.0) y haz click en Finalizar.



3. Realizaremos una aplicación en eclipse para su posterior despliegue con Tomcat.
- Para ello, abrimos Eclipse y creamos un nuevo Dyanamic Web Project llamado HolaMundo.

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 v9.0 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:

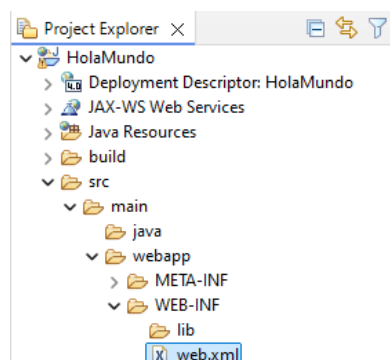
Haz click en siguiente hasta que aparezca la opción **“Generate web.xml deployment descriptor”**. Marca la opción para generar un descriptor web.xml.

Context root:

Content directory:

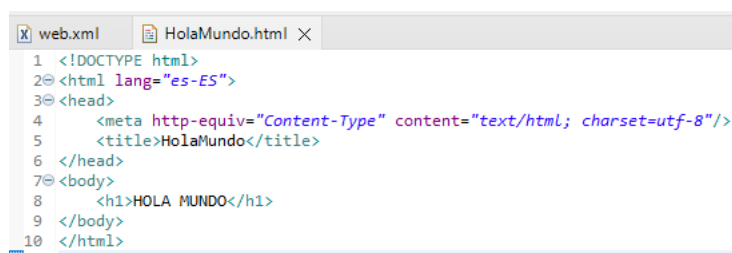
☒ Generate web.xml deployment descriptor

- b. Observa la estructura de carpetas del proyecto. Verás que dentro de la carpeta WEB-INF ha creado el archivo web.xml. **¿Cuál es la función de este archivo?**








También se puede acceder al contenido desde el desplegable “Deployment Descriptor: HolaMundo”.

- c. **META-INF** contiene el MANIFEST.MF mapea las clases de ficheros JAR existentes en otros proyectos pertenecientes al mismo Enterprise Application Project.
- d. En **Java Resources** irán los Servlets y los ficheros .java.
- e. Genera un archivo dentro de src->main->webapp que se llame HolaMundo.html (Botón derecho, New, HTML File) con un contenido similar al siguiente:



```
1 <!DOCTYPE html>
2 <html lang="es-ES">
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5   <title>HolaMundo</title>
6 </head>
7 <body>
8   <h1>HOLA MUNDO</h1>
9 </body>
10</html>
```

- f. Ahora exporta el proyecto como archivo WAR. Para ello, vamos a File -> Export y en el desplegable Web seleccionamos WAR file, el proyecto que deseamos exportar y el destino. **¿Qué es un archivo WAR?**
- g. Copia el archivo WAR en *RutaTomcat*\webapps.
- h. Observa que al poco de copiar el archivo WAR se crea la carpeta HolaMundo en la misma carpeta de webapps. Navega por la carpeta y comprueba que el contenido es el mismo que tenías en el proyecto de Eclipse.

 HolaMundo	15/12/2024 16:28	Carpeta de archivos
 host-manager	04/12/2024 9:51	Carpeta de archivos
 manager	04/12/2024 9:51	Carpeta de archivos
 ROOT	04/12/2024 9:51	Carpeta de archivos
 HolaMundo.war	15/12/2024 16:24	Archivo WAR

- i. Comprueba que la web ha sido desplegada automáticamente. Para ello intenta acceder a <http://localhost:8080/HolaMundo/HolaMundo.html>

4. Repetiremos el proceso utilizando un servlet.

- a. En el mismo proyecto nos colocamos sobre el directorio “Java Resources\src”. Hacemos click con el botón derecho, click en New y seleccionamos Servlet.
- b. Crearemos uno nuevo llamado HolaMundoServlet.
- c. Observa que ha creado un archivo .java con una plantilla de Servlet que nos podría servir para crear uno nuevo.

```
3 import java.io.IOException;
9
10 /**
11  * Servlet implementation class HolaMundoServlet
12  */
13 @WebServlet("/HolaMundoServlet")
14 public class HolaMundoServlet extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public HolaMundoServlet() {
21         super();
22         // TODO Auto-generated constructor stub
23     }
24
25     /**
26      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
27      */
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29         // TODO Auto-generated method stub
30         response.getWriter().append("Served at: ").append(request.getContextPath());
31     }
32
33     /**
34      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
35      */
36     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
37         // TODO Auto-generated method stub
38         doGet(request, response);
39     }
40 }
41 }
```

- d. Sin embargo, vamos a modificar el Servlet para que contenga el siguiente código:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class HolaMundoServlet extends HttpServlet{

    //Método para responder solicitudes HTTP GET
    public void doGet(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException {
        res.setContentType("text/html");//setting the content type
        PrintWriter pw=res.getWriter();//get the stream to write the data

        //writing html in the stream
        pw.println("<html><body>");
        pw.println("HolaMundo en modo servlet");
        pw.println("</body></html>");

        pw.close();//closing the stream
    }
}
```

- e. Modifica el archivo web.xml añadiendo el siguiente contenido dentro del elemento **<web-app>**:

```
13 <!-- Indicamos que la clase HolaMundoServlet se corresponderá con el servlet HolaMundo -->
14 <servlet>
15   <servlet-name>HolaMundo</servlet-name>
16   <servlet-class>HolaMundoServlet</servlet-class>
17 </servlet>
18 <!-- Indicamos que el servlet HolaMundo irá en la url /DiHola -->
19 <servlet-mapping>
20   <servlet-name>HolaMundo</servlet-name>
21   <url-pattern>/DiHola</url-pattern>
22 </servlet-mapping>
23
24 </web-app>
```

- f. Para cada servlet incluido en la aplicación habrá que crear un elemento similar.
- g. A continuación, exportamos el proyecto de nuevo a un archivo .WAR y repetimos el proceso visto en puntos anteriores relacionados con el archivo WAR. Comprobamos que ha funcionado accediendo desde un navegador a la ruta que habéis configurado.
5. Repite el proceso generando tu un propio servlet. Puedes fijarte en propuestas de internet.