8 Guía primeros Servidores Aplicaciones

8.2 Servidores Aplicaciones

4. Instalar el IDE eclipse y elegir la opción de Enterprise Java and Web Developers.

   Nos descargamos el instalador en función del Sistema Operativo en el que estemos trabajando y lo instalamos. Los siguientes enlaces son para 64 bits.

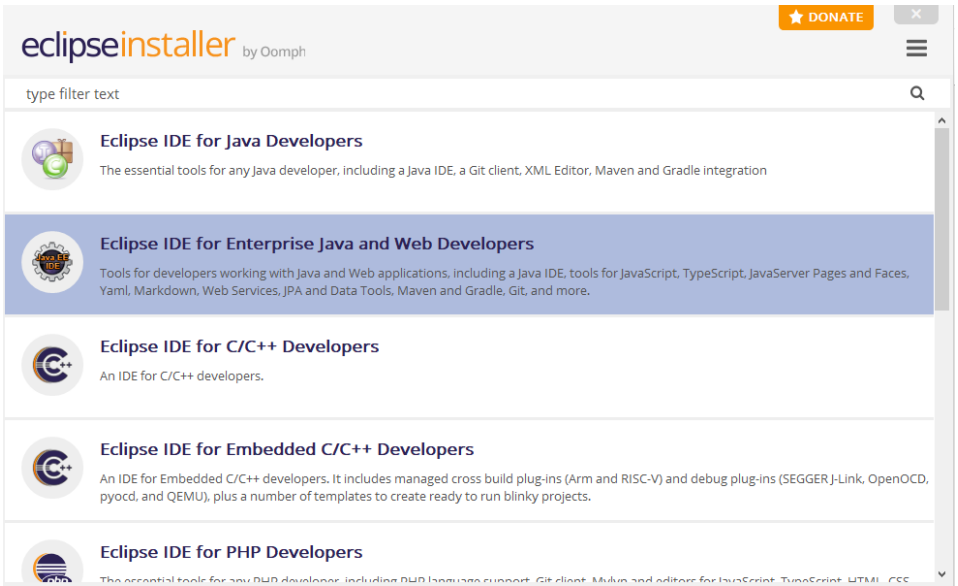
   Instalación Windows

   https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2023-09/R/eclipse-inst-jre-win64.exe&mirror_id=1313

   Instalación Linux (Ubuntu)

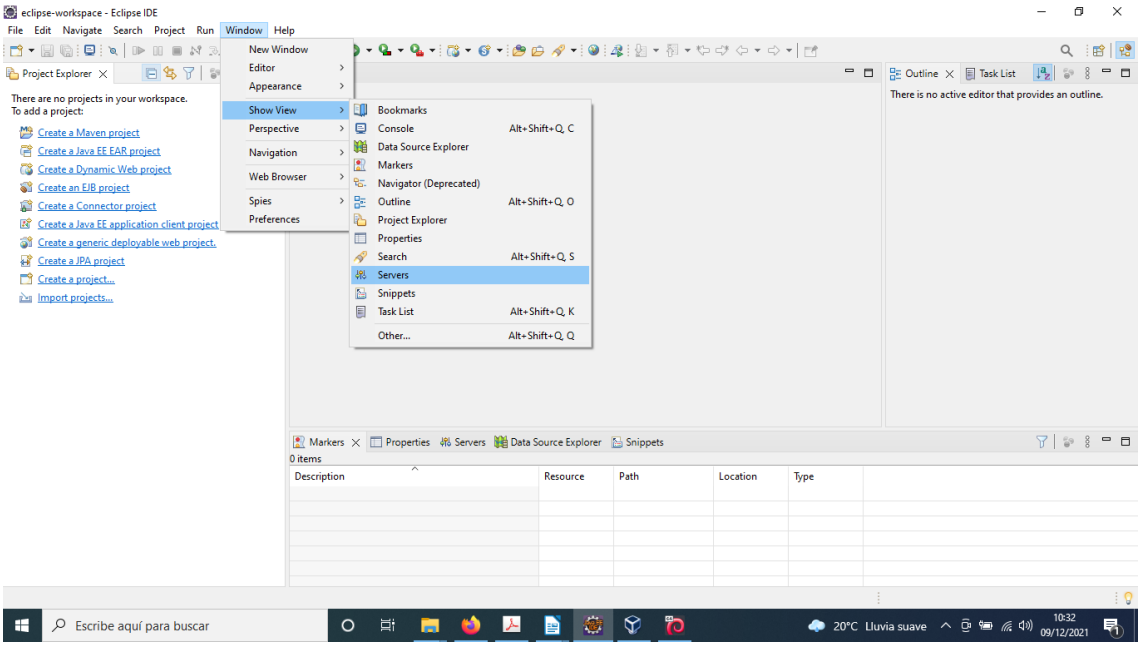
   https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2023-09/R/eclipse-inst-jre-linux64.tar.gz&mirror_id=1313

   Una vez descargado prodemos a instalar el Eclipse, cuando nos salga la siguiente ventana debemos seleccionar la instalación de "**Eclipse IDE for Enterprise Java and Web Developers**", la opción que se resalta.

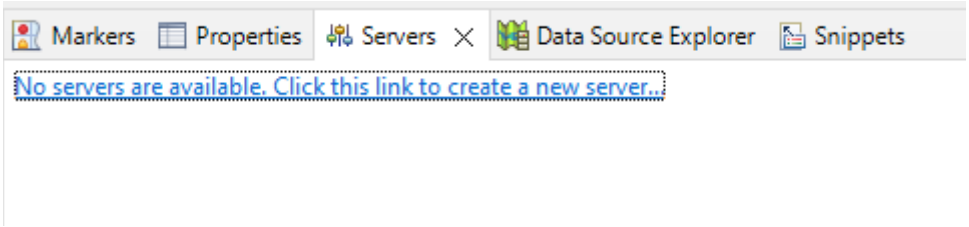


5. Realizar las primeras configuraciones de Eclipse para preparar nuestro entorno de trabajo para el desarrollo de aplicaciones.

   Una vez instalado vamos a proceder a realizar la integración del Servidor de aplicaciones para realizar las pruebas de nuestro código. Para ello debemos ir a Windows>Show View>Servers.

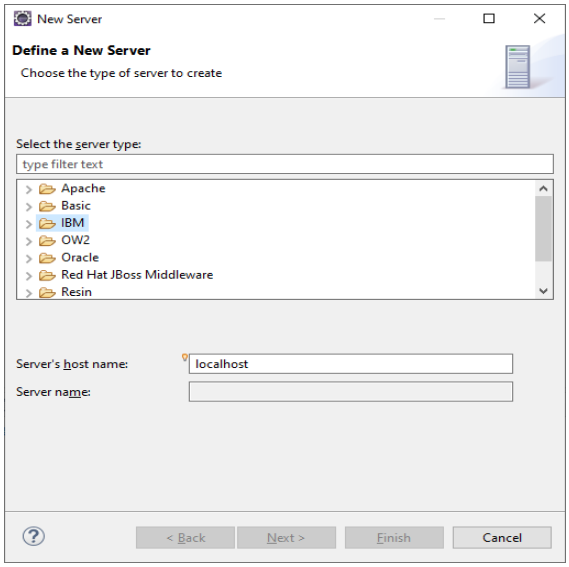


Se nos abre en la parte inferior, en los paneles de información la pestaña Servers, donde deberemos pinchar en el enlace de la parte inferior para vincular el servidor de aplicaciones o con el botón derecho.



Ambas opciones nos llevarán a la ventana para definir el Servidor de Aplicaciones que empleará nuestro IDE para realizar las pruebas sobre los desarrollos que realicemos.

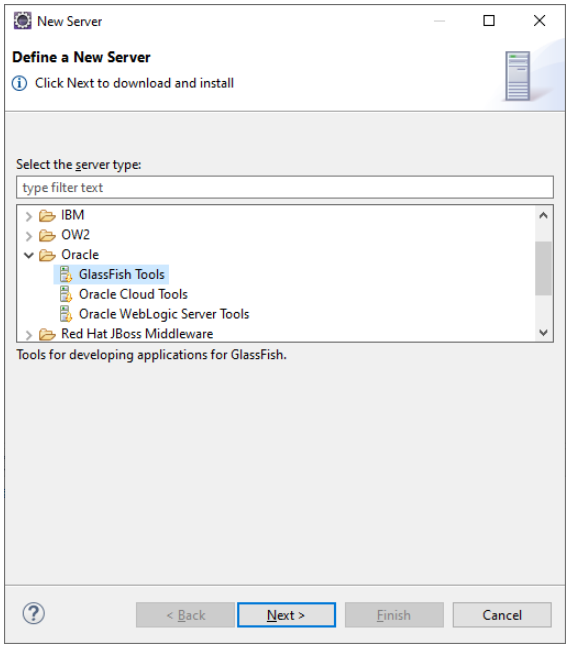
La ventana que se nos mostrará es la siguiente.



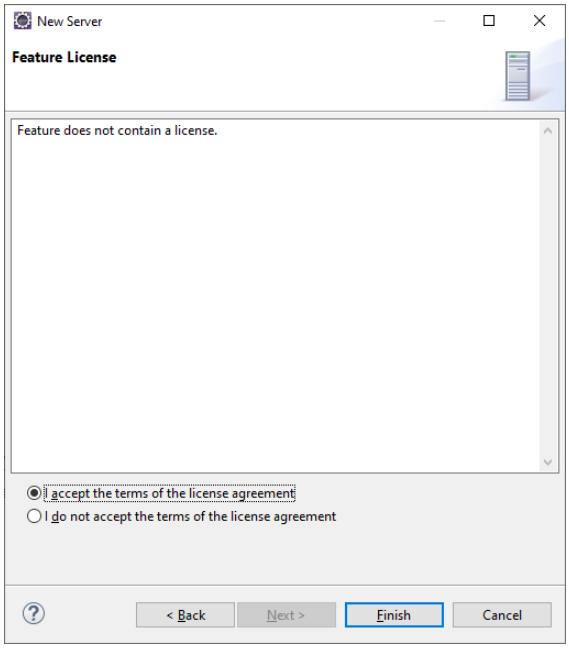
En esta ventana se nos ofrecen diversas opciones de servidores de aplicaciones que tiene disponibles.

En el caso de que quisiéramos conectar con el Tomcat que tenemos en nuestro servidor, tenemos que tener en cuenta si el IDE está en la misma máquina (Server's host name localhost) o si por el contrario lo tenemos en una máquina remota donde tendríamos que poner la IP de dicha máquina en Server's host name.

En nuestro caso vamos a configurar el Tomcat o el WildFly que tenemos instalado en nuestro servidor.



Pulsamos en Next.



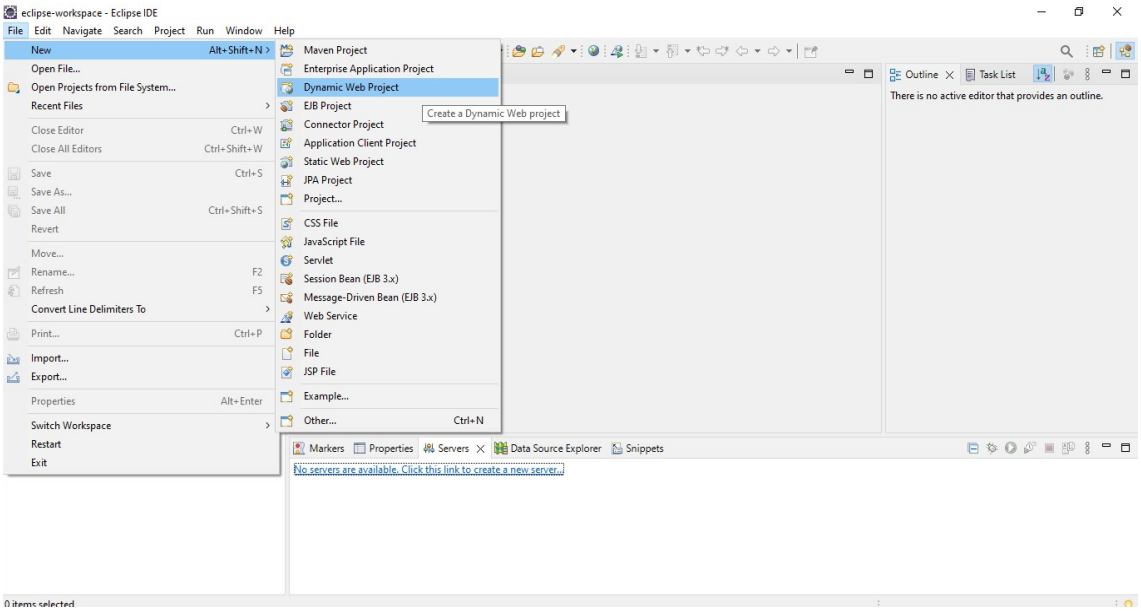
Aceptamos los términos de acuerdo de la licencia y pulsamos en Finish.

Por lo que ya tendríamos integrado el Servidor de Aplciaciones en el IDE Eclipse.

A continuación realizamos una prueba de integración con un simple ejemplo.

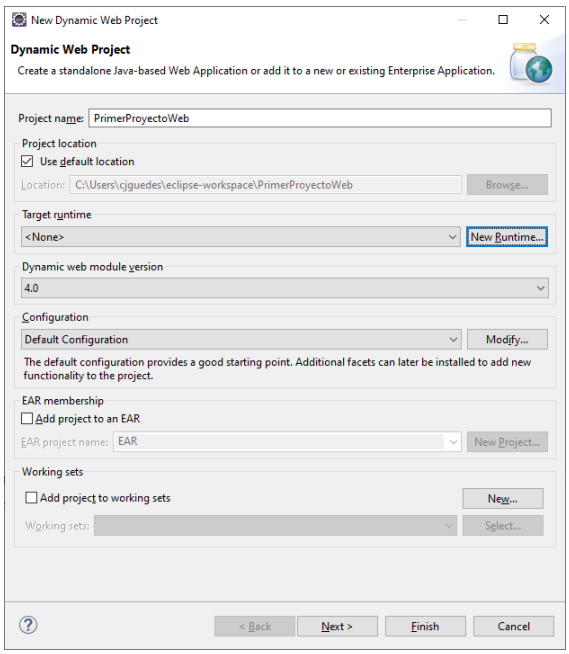
El primer paso es crear un proyecto, para lo que tenemos que seleccionar:

File>New>Dynamic Web Project



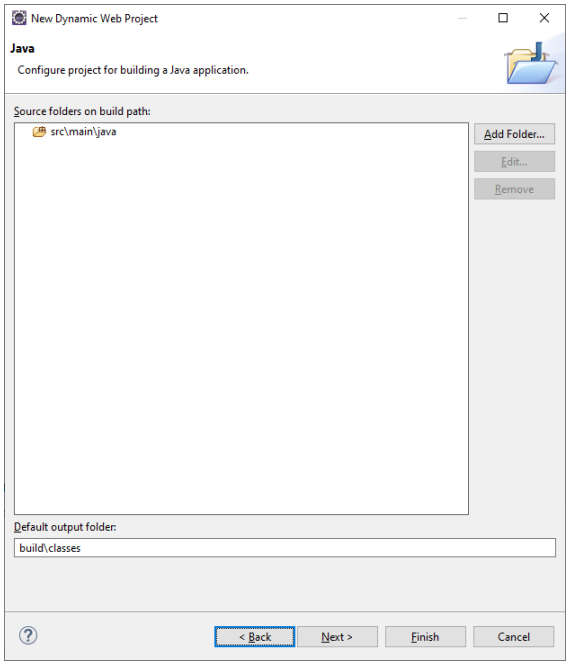
A partir de aquí es bastante intuitivo, de hecho es que si estamos familiarizados con la creación de aplicaciones Java es similar. Nos solicitará un nombre para el proyecto.

En este caso vamos a nombrarlo como “PrimerProyectoWeb”

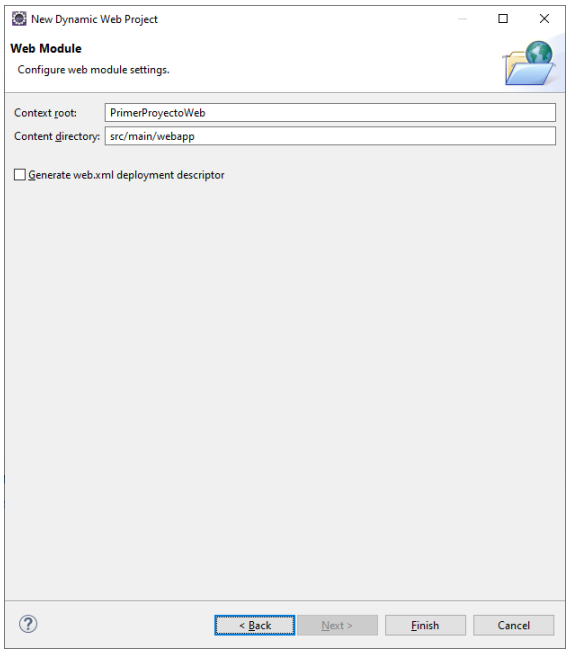


En los siguientes pasos nos pedirá la ubicación de los archivos fuente (src) y la ubicación de los archivos web.

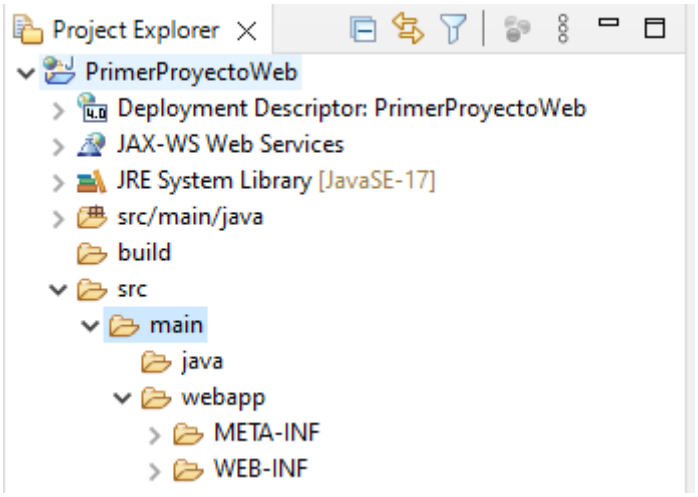
src



Archivos web



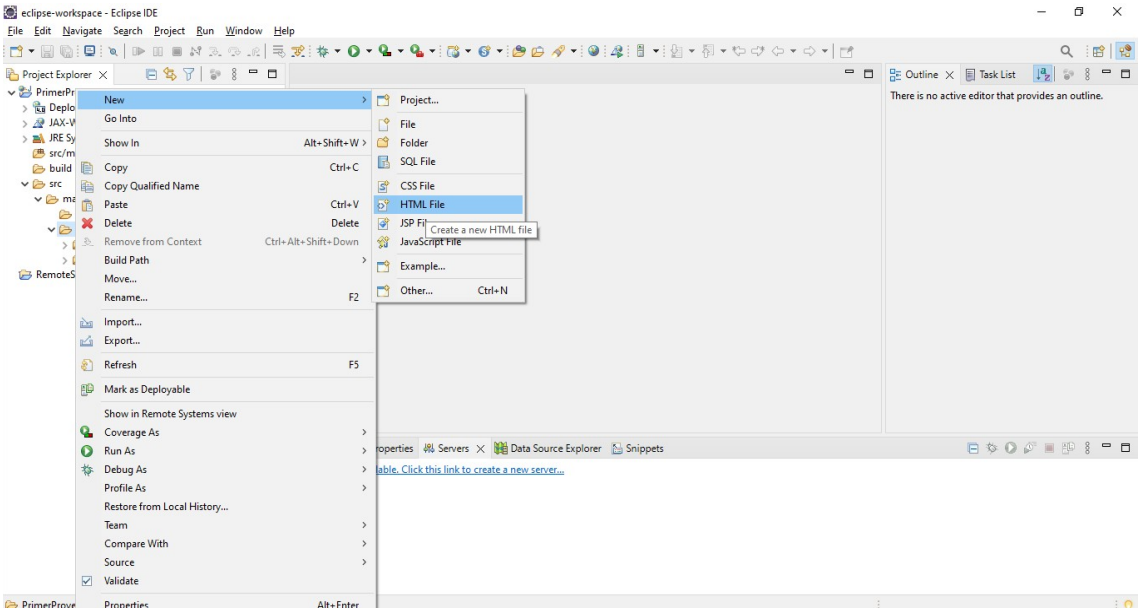
Quedando la estructura del proyecto de la siguiente manera:

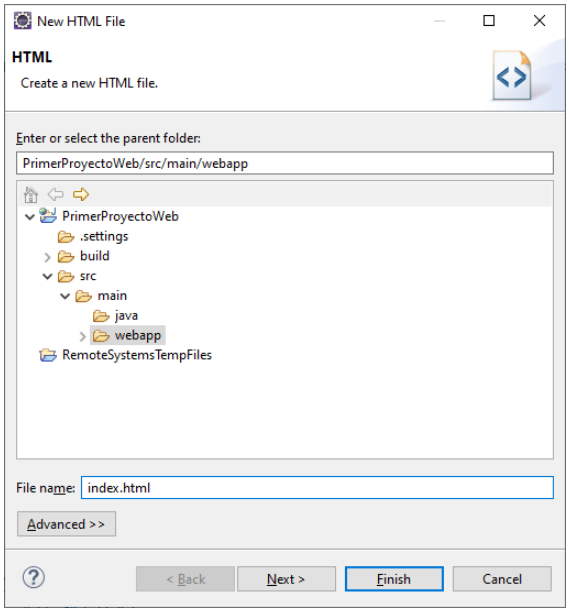


Donde encontramos una estructura reconocible de aplicación web con los siguientes directorios:

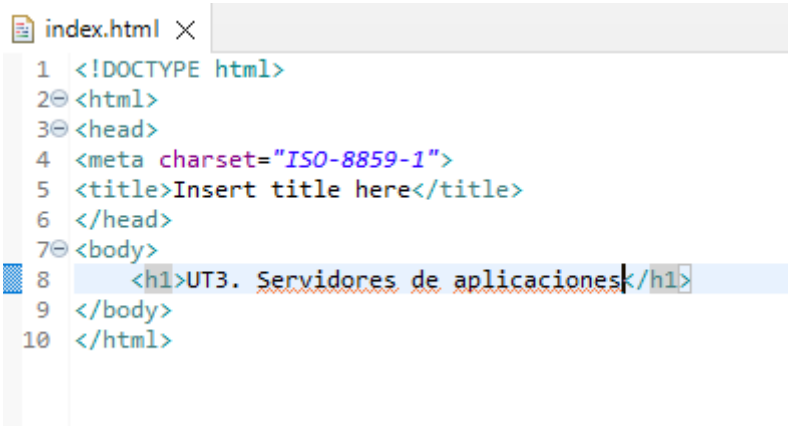
- **Java**, donde tendremos las fuentes Java, en src/main
- **build**, donde se guardarán los archivos precompilados
- **webapp**, será la raíz de nuestro proyecto web, donde se guardarán los archivos html, jsp, imágenes, etc. Estos archivos estarán disponibles de forma pública en la aplicación.

Vamos a añadir un archivo html a nuestro proyecto, para realizarlo hacemos clic con botón derecho sobre la carpeta webapp y seleccionamos New>HTML file al que llamaremos index.html



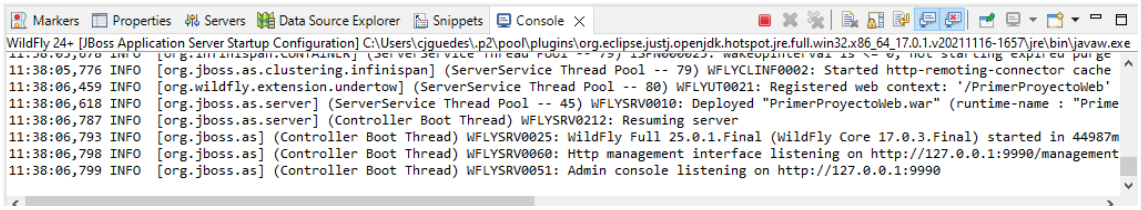


En el índice generado añadimos una etiqueta <h1></h1> a la que le pondremos el texto "UT3. Servidores de aplicaciones", quedando como se muestra en la siguiente imagen:



Ejecutamos seleccionando el nombre del proyecto en el explorador y pulsando en el boton de Run (icono verde con símbolo de play). Elegimos el servidor que hemos configurado anteriormente.

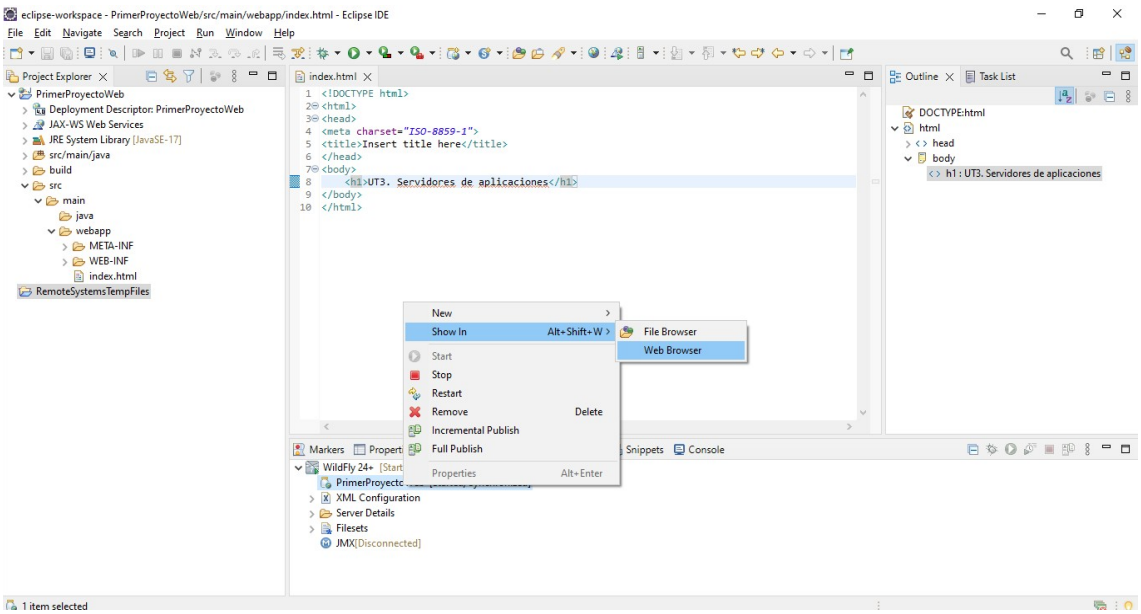
En la pestaña consola del panel de información veremos como se realizar el proceso de despliegue de la aplicación web en el servidor de aplicaciones.



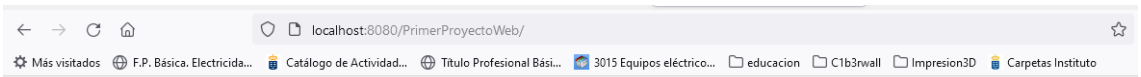
Para ver el resultado acudimos al navegador web y poniendo la URL de la aplicación web la podemos ver desplegada.

http://localhost:8080/PrimerProyectoWeb/

Lo anterior también lo podemos ver si vamos a la pestaña de servers del panel de información con el proyecto arrancado y pulsamos con botón derecho sobre el nombre del proyecto que aparece en nuestro servidor y seleccionamos Show in>web Browser



Si estamos en la máquina del servidor ponemos localhost o 127.0.0.1, en el caso de que el servidor esté en otra máquina la IP.



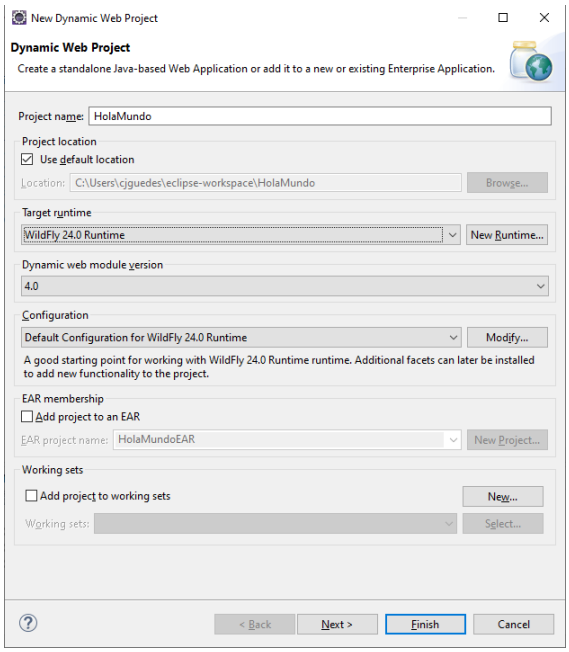
UT3. Servidores de aplicaciones

Depliegue de Servlets

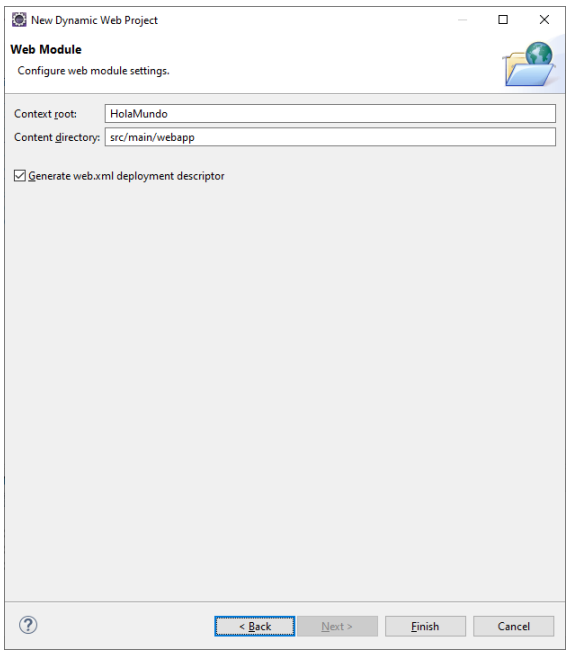
En este punto vamos a desplegar un Servlet.

Para ello vamos a crear un nuevo proyecto como ya hicimos antes File>New>Dynamic Web Project.

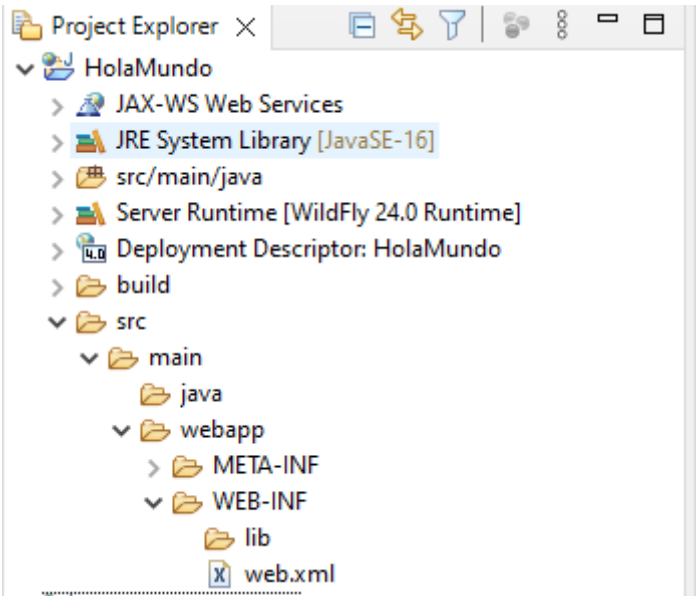
Este proyecto lo denominaremos **HolaMundo**.



Pulsamos en Next y dejamos por defecto la configuración de la ruta de las fuentes, carpeta SRC. Pulsamos en Next y en este caso debemos activar la creación del archivo descriptor de despliegue, web.xml.



Analizando la estructura de carpetas vemos que es similar a la anterior, lo que en este caso al activar la creación del descriptor de despliegue nos ha incluido el archivo web.xml



Añadimos una página html, HolaMundo.html que nos dará el típico mensaje de Hola Mundo!.

Seleccionando la carpeta webapp, pulsamos con botón derecho y seleccionamos New>HTML File, le asignamos el nombre indicado y pulsamos en Finish.

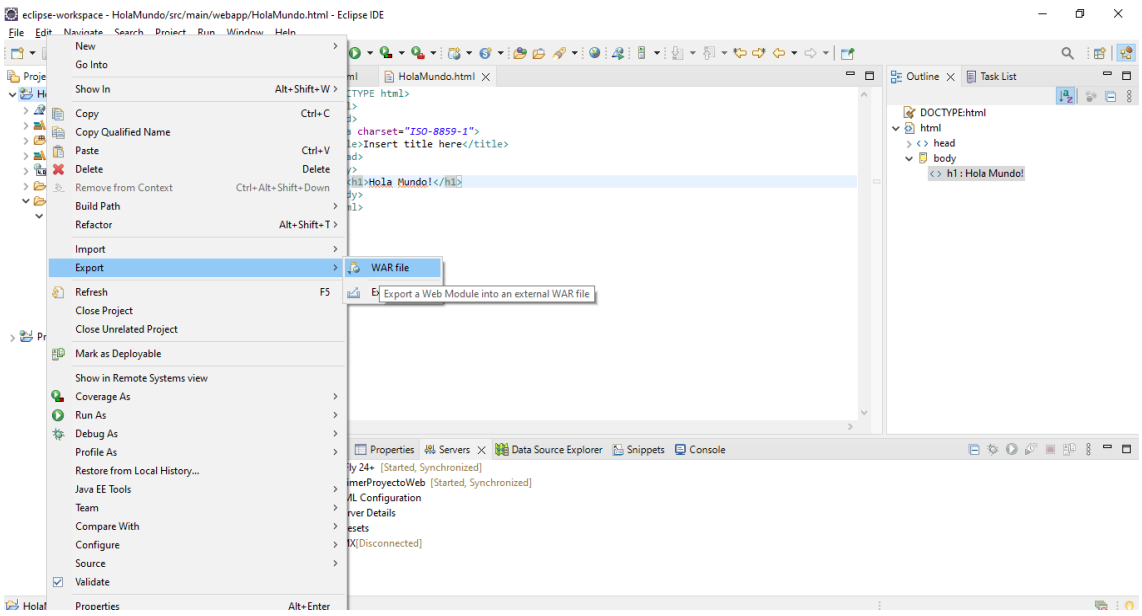
Editamos el archivo e incluimos una etiqueta de encabezado 1 con el mensaje.

index.html HolaMundo.html

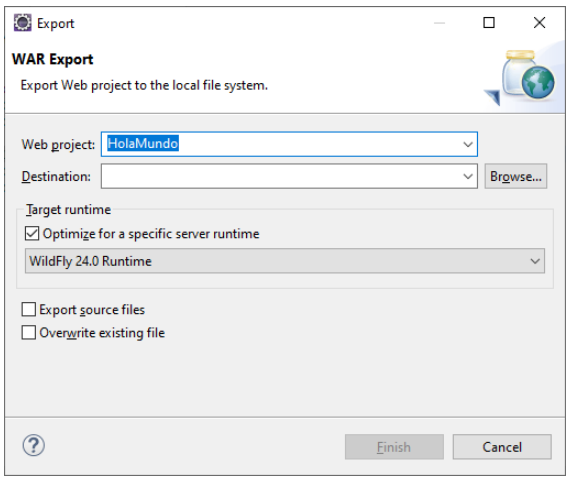
```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <h1>Hola Mundo!</h1>
</body>
</html>
```

En este caso en vez de ejecutar la aplicación web desde el IDE de Eclipse lo que vamos a realizar es la exportación del archivo HolaMundo.war para luego desplegarlo en el servidor de aplicaciones instalados en las actividades anteriores.

El procedimiento para obtener el archivo para el despliegue web es seleccionar la carpeta del proyecto y haciendo clic con botón derecho en el menú que se nos abre escogemos Export>WAR file



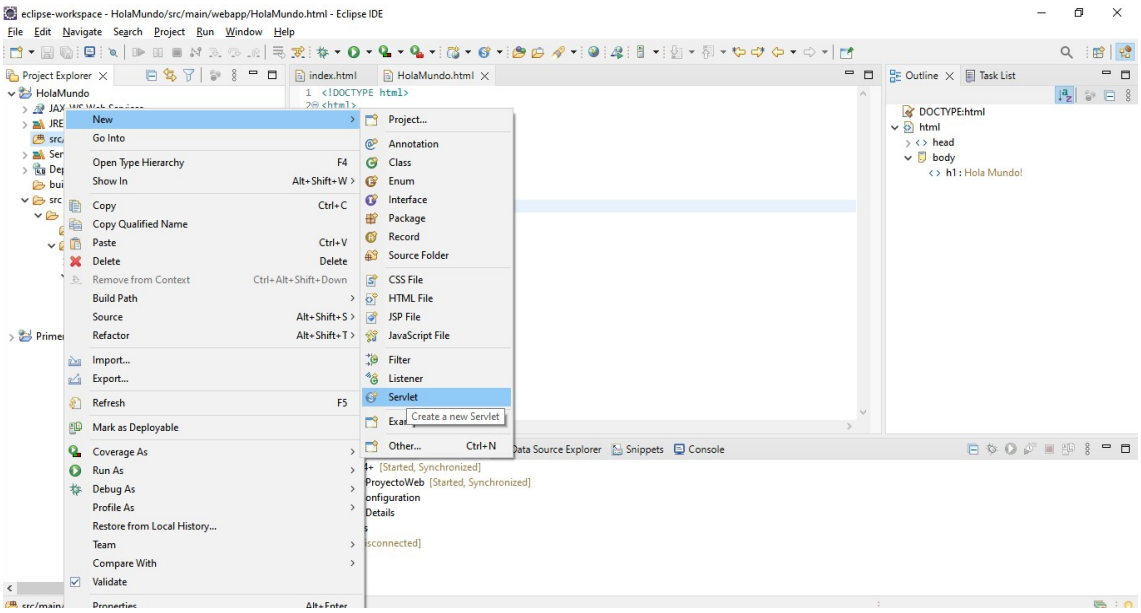
En la ventana que se nos abre nos pedirá la ubicación donde queremos guardar el archivo web empaquetado. Pulsamos en el botón browse y elegimos la ruta donde guardará el referido archivo.



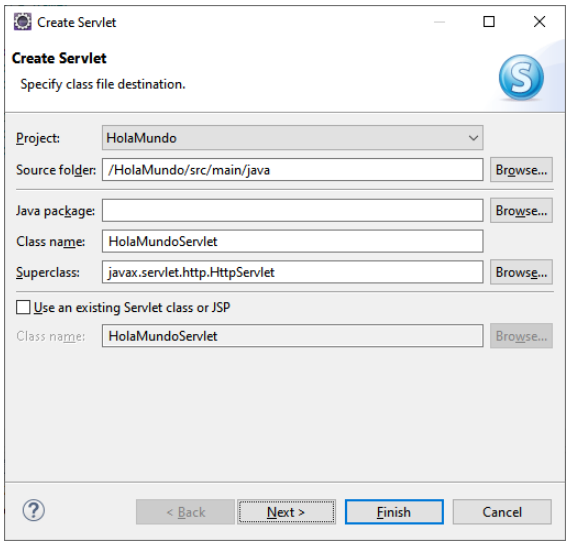
Pulsamos en Finish y obtenemos el archivo en la ruta que le hemos facilitado.

Con este archivo haciendo el procedimiento visto en las unidades anteriores podemos proceder a realizar el despliegue en los Servidores de Aplicaciones que hemos instalado y configurado.

Ahora vamos a crear el servlet ubicándonos en src/main/java y pulsamos en File>New>Servlet



Se nos abre la siguiente ventana para crear el Servlet y lo único que tenemos que poner es el Class Name “HolaMundoServlet”



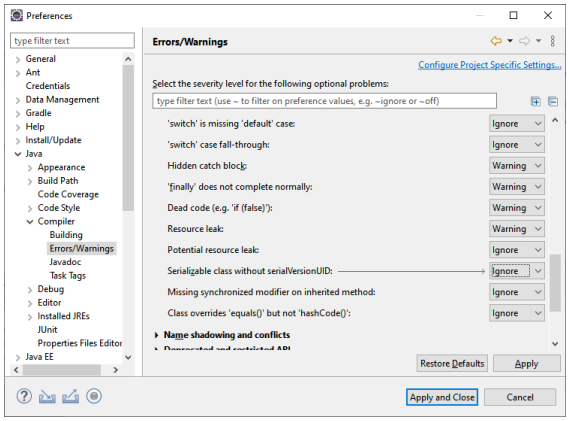
Podemos o pulsar en Next y podemos introducir la información del descriptor de despliegue, en el siguiente Next nos permite indicar los métodos a incluir, podemos darle a Finish directamente para que se cree el Servlet.

Después de que nos cree el Servlet vamos a proceder a editarlo con el siguiente contenido.

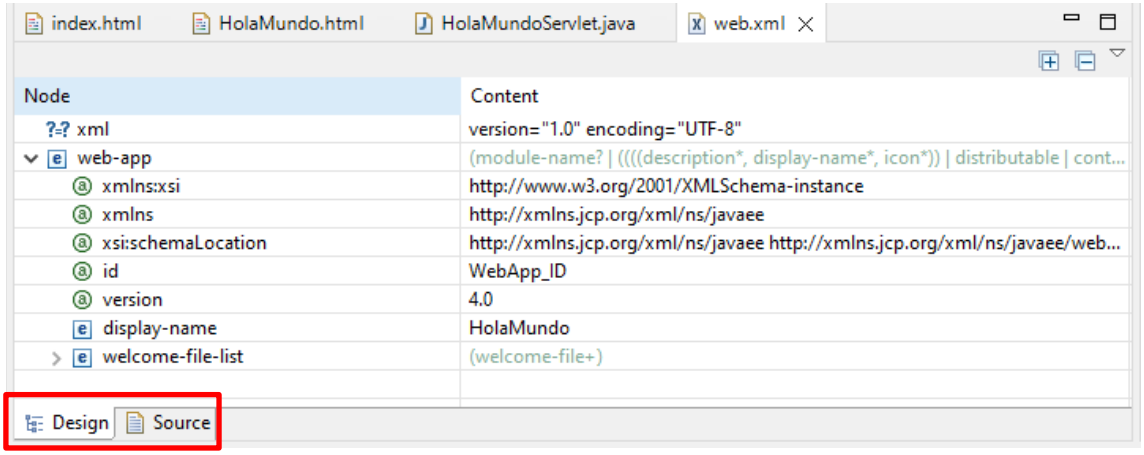
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/**
 * Servlet implementation class HolaMundoServlet
 */
public class HolaMundoServlet extends HttpServlet {
        @Override
        public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException{
                //Establece el tipo MIME del mensaje de respuesta
                response.setContentType("text/html");
                //Crea un flujo de salida para escribir la respuesta a la petición de cliente
                PrintWriter out = response.getWriter();

                //Escribe el mensaje de respuesta en una página html
                try {
                        out.println("<html>");
                        out.println("<head><title>Hola Mundo</head></title></head>");
                        out.println("<body>");
                        out.println("<h1>Hola Mundo</h1>"); //Encabezado con el mensaje Hola
```

```
Mundo
                        //Muestra información de la petición del cliente
                        out.println("<p>Request URL:"+request.getRequestURI()+"</p>");
                        out.println("<p>Protocolo:"+request.getProtocol()+"</p>");
                        out.println("<p>Dirección remota:"+request.getRemoteAddr()+"</p>");
                        //Genera un número aleatorio para cada petición
                        out.println("<p>Número                                    aleatorio:
<strong>"+Math.random()+"</strong></p>");
                        out.println("</body></html>");
                }finally {
                        out.close(); //Cierra el flujo de salida
                }
        }
}
```

Al introducir este código nos saltará una advertencia sobre la clase HolaMundoServlet podemos aplicar las soluciones que nos indica el IDE o ir a Widow>Preferences>Java>Compiler>Error Warnings y en Potential program problems Poner Ignore en “Serializable class without serailVersionUID:”.



Vamos a editar el descriptor de despliegue. Hacemos doble clic sobre él y nos saldrá un cuadro con los parámetros del web.xml, porque por defecto entra en el modo Design, en la parte inferior derecha tenemos unas pestañas cambiamos a Source y accedemos al código.



El código a introducir al final del archivo web.xml es el siguiente antes de la etiqueta de cierre de web-app:

```
<!--  Indicamos que la clase HolaMundoServlet se corresponde con el nombre de servlet

  HolaMundo-->

  <servlet>

       <servlet-name>HolaMundo</servlet-name>

       <servlet-class>HolaMundoServlet</servlet-class>

  </servlet>

  <!-- Indicamos que el nombre del Servlet HolaMundo corresponde a la URL / DiHola -->

  <servlet-mapping>

       <servlet-name>HolaMundo</servlet-name>

       <url-pattern>/DiHola</url-pattern>

  </servlet-mapping>
```

index.html HolaMundo.html HolaMundoServlet.java web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/
  <display-name>HolaMundo</display-name>
    <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
  <!--  Indicamos que la clase HolaMundoServlet se corresponde con el nombre de servlet
  HolaMundo-->
  <servlet>
    <servlet-name>HolaMundo</servlet-name>
    <servlet-class>HolaMundoServlet</servlet-class>
  </servlet>
  <!-- Indicamos que el nombre del Servlet HolaMundo corresponde a la URL /DiHola -->
  <servlet-mapping>
    <servlet-name>HolaMundo</servlet-name>
    <url-pattern>/DiHola</url-pattern>
  </servlet-mapping>
</web-app>
```

Design Source

Se debe tener en cuenta que en el descriptor de despliegue para cada servlet declaramos el elemento servlet y un elemento servlet-mapping en el que ambos comparten el servlet-name para relacionarlo con el servlet correspondiente.

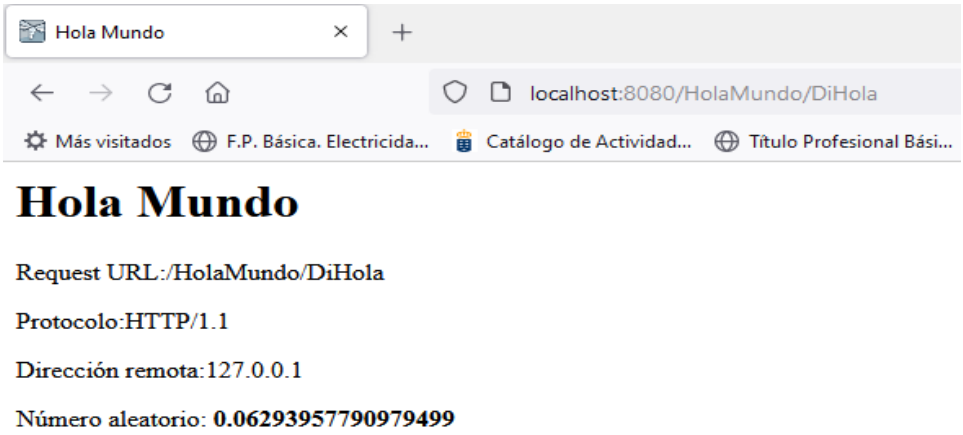
Empaquetamos la aplicación y la probamos en el servidor de aplicaciones.

index.html HolaMundo.html HolaMundoServlet.java web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/
  <display-name>HolaMundo</display-name>
    <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>default.htm</welcome-file>
  </welcome-file-list>
  <!--  Indicamos que la clase HolaMundoServlet se corresponde con el nombre de servlet
  HolaMundo-->
  <servlet>
    <servlet-name>HolaMundo</servlet-name>
    <servlet-class>HolaMundoServlet</servlet-class>
  </servlet>
  <!-- Indicamos que el nombre del Servlet HolaMundo corresponde a la URL /DiHola -->
  <servlet-mapping>
    <servlet-name>HolaMundo</servlet-name>
    <url-pattern>/DiHola</url-pattern>
  </servlet-mapping>
</web-app>
```

Design Source

El resultado sería el siguiente. Poniendo la URL: **http://localhost:8080/HolaMundo/DiHola**

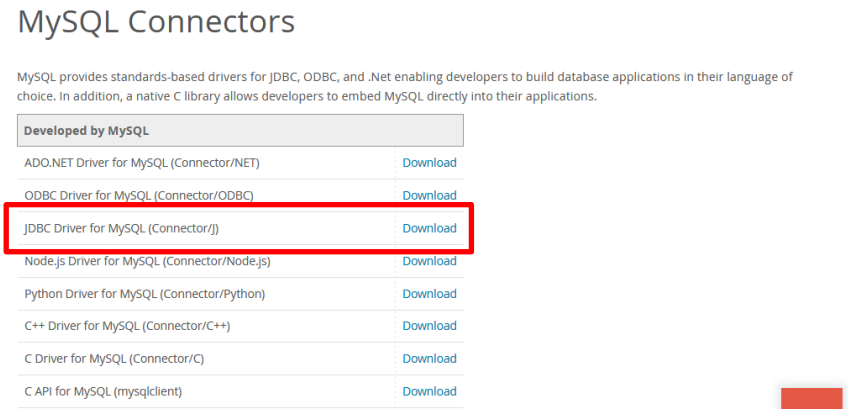


Conexión con Bases de Datos

En este caso vamos a realizar la conexión con MySQL, para ello necesitamos tener instalado el Servidor MySQL y habilitar el conector JDBC para efectuar la conexión entre el IDE de Eclipse y la base de datos.

MySQL ya lo deberíamos de tener instalado de las actividades anteriores y para obtener el conector JDBC debemos ir a descargarlo a la página del propio MySQL.

https://www.mysql.com/products/connector/



De las opciones que nos ofrece nos descargamos la del Driver JDBC. Esto nos llevará a la página de descarga del referido conector que tiene un desplegable

que nos permite elegir entre las diferentes opciones de Sistemas Operativos que tiene disponibles el conector. En nuestro caso elegimos "Platform Independent", ya que nos dará acceso a descargar un archivo comprimido que en su interior tiene el archivo "**mysql-connector-java-8.0.27.jar**"

Para realizar la prueba vamos a crear una nueva base de datos en MySQL, crearemos una tabla y cargaremos los datos.

Estos son los pasos a realizar en MySQL:

```
create database TiendaLibros;

use TiendaLibros;


create table Libros (

id int,

titulo varchar(50),

autor varchar(50),

precio float,

cantidad int,

primary key (id));


insert into Libros values

(1001, 'Servicios de Red e Internet', 'Alvaro García', 25, 100);

insert into Libros values

(1002, 'Apache Tomcat 7', 'Aleska Vukotic', 22.22, 22);

insert into Libros values

(1003, 'Beginning JSP, JSP and Tomcat Web Development', 'Giulio Zambon', 33.33, 33);

insert into Libros values

(1004, 'Tomcat, the definitive guide', 'Jason Brittain', 55.55, 55);

insert into Libros values
```

(1005, 'Profesional Apache Tomcat', 'Vivek Chopra', 66.66, 66);

Ahora vamos a crear un nuevo proyecto web dinámico, al que llamaremos AccesoDatos, al crearlo marcaremos la opción web.xml.

Añadiremos una página html que llamaremos ConsultaLibros.html.

```
<!DOCTYPE html>
<html>
<head>
 <title>TiendaLibros</title>
</head>
<body>
 <h2>Tienda Libros</h2>
 <form method =”get” action=”http://localhost:8080/AccesoDatos/consulta”>
  <b>Elige autor:</b>
  <input type=”checkbox” name=”autor” value=”Alvaro Garcia”>Alvaro Garcia
  <input type=”checkbox” name=”autor” value=”Aleska Vukotic”> Aleska Vukotic
  <input type=”checkbox” name=”autor” value=”Giulio Zambon”> Giulio Zambon
  <input type=”submit” value=”Buscar”>
 </form>
</body>
</html>
```

ConsultaLibros.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>TiendaLibros</title>
5  </head>
6  <body>
7    <h2>Tienda Libros</h2>
8    <form method ="get" action="http://localhost:8080/AccesoDatos/consulta">
9      <b>Elige autor:</b>
10     <input type="checkbox" name="autor" value="Alvaro Garcia">Alvaro Garcia
11     <input type="checkbox" name="autor" value="Aleska Vukotic"> Aleska Vukotic
12     <input type="checkbox" name="autor" value="Giulio Zambon"> Giulio Zambon
13     <input type="submit" value="Buscar">
14   </form>
15 </body>
16 </html>
```

Ejecutamos el proyecto y accedemos a ella poniendo en la URL:

http://localhost:8080/AccesoDatos/ConsultaLibros.html

Si pulsamos en el botón buscar no realizará ninguna acción puesto que no hemos implementado el Servlet al que está relacionado.

Para hacer la conexión con la base de datos debemos de poner en el CATALINA_HOME en la carpeta lib el conector de MySQL "mysql-connector-java-8.0.27.jar" y reiniciar el servidor para que lo reconozca.

A continuación enumeramos los pasos que realiza una aplicación para conectarse a los datos de la base de datos con el conector JDBC:

1. Carga el driver JDBC.
2. Conecta a la Base de Datos empleando la clase *Connection*.
3. Crea las sentencias SQL, utilizando objetos de tipo *Statement*.
4. Ejecuta sentencias SQL a través de los objetos de tipo *Statement*.
5. En caso necesario, procesa conjunto registros resultante utilizando la clase *ResulSet*.

Visto lo anterior prodecemos a crear el Servlet ConsultServlet.java en la carpeta src/main/java

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

//@WebServlet("/ConsultaServlet")
public class ConsultaServlet extends HttpServlet {
	//El método doGet() se ejecuta una vez por cada petición HTTP GET.
	@Override
	public void doGet(HttpServletRequest request, HttpServletResponse
response)
	throws ServletException, IOException{
		//Establecemos el tipo MIME del mensaje respuesta
		response.setContentType("text/html");
		//Creamos un objeto para poder escribir la respuesta
		PrintWriter out = response.getWriter();

		Connection conn = null;
		Statement stmt = null;
		try {
			//Paso 1. Cargar el driver JDBC
			Class.forName("com.mysql.jdbc.Driver").newInstance();

			//Paso 2. Conectarse a la Base de Datos utilizando la
clase Connection
			String userName = "root";
			String password = "despliegue";
			//URL a la base de datos (equipo, puerto, base de datos)
```

```
                    String url="jdbc:mysql://localhost/TiendaLibros";
                    conn = DriverManager.getConnection(url, userName, pa-
ssword);

                    //Paso 3. Crear sentencias SQL, utilizando objetos de ti-
po statement
                    stmt = conn.createStatement();

                    //Paso 4. Ejecutar las sentencias SQL a través de los ob-
jetos Statement
                    String sqlStr = "Select * from libros where
autor="+"'"+request.getParameter("autor")+"'";

                    //Generar una página HTML como resultado de la consulta
                    out.println("<html><head><title>Resultado de la
consulta</html></head></title>");
                    out.println("<h3>Gracias por tu consulta</h3>");
                    out.println("<p>Tu consulta es: "+ sqlStr +"</p>");
                    ResultSet rset = stmt.executeQuery(sqlStr);

                    //Paso 5. Procesar el conjunto de registros resultante
utilizando ResultSet
                    int count = 0;
                    while (rset.next()) {
                          out.println("<p>"+ rset.getString("autor")+", "+
rset.getString("titulo") + rset.getString("precio") + "</p>");
                          count++;
                    }
                    out.println("<p>=== " + count + " registros encontrados
===</p>");
                    out.println("</body></html>");



             }catch (Exception ex) {
                    ex.printStackTrace();
             }finally {
                    out.close(); //Cerramos el flujo de escritura
                    try {
                          //Cerramos el resto de recursos
                          if(stmt !=null) stmt.close();
                          if(conn !=null) conn.close();
                    }catch (SQLException ex) {
                          ex.printStackTrace();
                    }
             }
      }
}
```

Para terminar editamos el descriptor de despliegue.

```
 <servlet>
      <servlet-name>ConsultaUsuario</servlet-name>
      <servlet-class>ConsultaServlet</servlet-class>
  </servlet>
  <!-- Indicamos que el nombre del Servlet Consulta corresponde a la URL /
consulta -->
  <servlet-mapping>
      <servlet-name>ConsultaUsuario</servlet-name>
      <url-pattern>/consulta</url-pattern>
  </servlet-mapping>
```

Lo explicado antes es una referencia para poder hacer los puntos 6, 7, 8 y 9 de la actividad 7, teniendo en cuenta los archivos que se adjuntan.