Java EE
Ejercicio (MVC Parte 1)

Ejercicio 1 de Java EE. MVC

Ejercicio 1 de Modelo Vista Controlador

En este ejercicio, vamos a readaptar la aplicación del ejercicio anterior para utilizar el Modelo Vista Controlador. Para esto, vamos a crear un **nuevo proyecto web "JavaEEMVC"** y copiar en este proyecto aquellos recursos que vamos a necesitar de la aplicación anterior o del proyecto "JavaEEServlets", s**eguiremos usando la base de datos "musicadb2" y crearemos los nuevos componentes necesarios.**

Como en el ejercicio anterior, imaginemos que trabajas en una pequeña emisora musical de radio local que quiere dar otro servicio a sus oyentes, y para ello crea una pequeña web donde se pueda ver el listado de los grupos musicales con los que trabaja y el detalle o información de cada uno de ellos.

La aplicación tiene que funcionar igual que antes, es decir, muestra un listado con todos los grupos musicales que tenemos en nuestra bbdd, además, para cada uno de esos grupos podremos ver toda su información a través de un enlace, de la siguiente forma:

Si pulsamos sobre el enlace "Ver más..." de cada uno de los registros, nos muestra la siguiente ventana:

Y si pulsamos sobre el enlace "Volver al listado" nos lleva de regreso a la primera ventana.

En primer lugar, debemos **crear el nuevo proyecto web "JavaEEMVC"**.

Además, vamos a configurar nuestra aplicación Web creando **el descriptor de despliegue de la aplicación**, es decir, en el archivo **web.xml**. Este archivo contiene meta-datos, como la página por defecto a mostrar, Servlets a cargar, o restricciones de seguridad que imponer a los archivos.

El descriptor de despliegue de la aplicación Web sólo hace falta si un módulo Web contiene algún Servlet, filtro, escuchador, o algún parámetro inicial de la Aplicación; si sólo contiene páginas JSP y archivos estáticos no hará falta incluir el archivo web.xml. En nuestro caso vamos a crear el **fichero web.xml** para indicar nuestra página de arranque de la aplicación, que en este caso será el servlet **ServletEmisora**.

NOTA: Si no recuerdas como crear un proyecto web y configurar el servidor para ese proyecto, ver el apartado Crear proyecto web al final del documento. Además, en este apartado veras como crear el proyecto con el fichero web.xml.

Como vamos a trabajar con las librerías JSTL en nuestras JSPs debemos incluir dichas librerías en nuestro proyecto.

NOTA: Si no recuerdas como se hace, puedes ver el apartado Incluir librerías JSTL al final del documento.

Como vamos a trabajar con la BBDD, lo primero que vamos a hacer, es **crear un pool de conexiones**, para ello vamos a seguir los siguientes pasos:

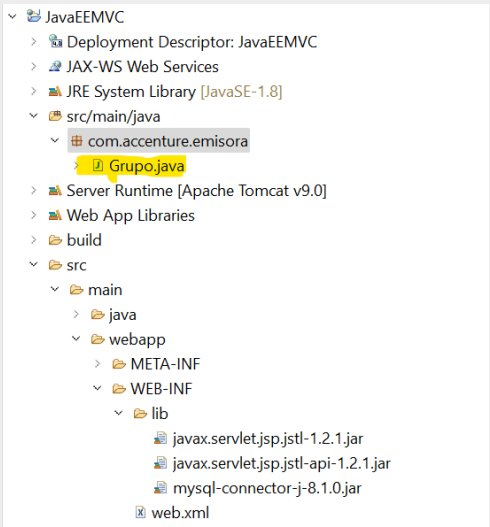
- **Incluir el driver** jdbc para MySQL en el classpath de nuestro proyecto para poder conectar con la BBDD.

  NOTA: Si no recuerdas como se hace, puedes ver el apartado Incluir driver JDBC al final del documento.

- **Definir o construir el pool de conexiones** dentro de la carpeta META-INF de nuestro proyecto, es decir, dentro de esta carpeta **creamos el fichero context.xml.** Posteriormente, crearemos una referencia a este pool de conexiones desde nuestro código Java.

  NOTA: Si no recuerdas como crear el pool de conexiones, ver el apartado Crear pool de conexiones al final del documento.

Como vamos a seguir trabajando con grupos musicales, debemos tener la clase **Grupo** que nos permita crear elementos de ese tipo, así que dentro de nuestro nuevo proyecto vamos a crear el paquete **es.accenture.emisora** y vamos a copiar la clase Grupo de proyecto anterior "JavaEEServlets".



Como vamos a trabajar con el patrón MVC, necesitamos crear el Modelo, es decir, vamos a crear el fichero java **ModeloGrupo** que se encargara de conectar con la BBDD para recuperar la información necesaria para nuestra aplicación. Que debe tener este fichero:

- Una variable encapsula de tipo DataSource, que será la encargada de almacenar el pool de conexiones.

```
private DataSource poolConexiones;
```

- El constructor para la clase, que recibirá como parámetro el pool de conexiones, es decir, un parámetro de tipo DataSource.

```
public ModeloGrupo(DataSource poolConexiones) {
        this.poolConexiones = poolConexiones;
}
```

- Como al arrancar la aplicación, lo que necesitamos es un listado de todos los grupos musicales que tenemos en la BBDD, necesitamos un método que nos devuelva ese listado, para ello, debemos **crear el método getGrupos** que retorne una lista de tipo Grupo, además, este método debe lanzar una excepción para obligar a que cuando llamemos a este método se haga dentro de un try-catch y controlar los posibles errores.

```
public List<Grupo> getGrupos() throws Exception
```

Este método debe realizar lo siguiente:

- Crear una lista de tipo Grupo

```
List<Grupo> listaGrupos = new ArrayList<>();
```

- Crear los objetos de tipo Connection, Statement y ResultSet

```
Connection miConesion = null;
Statement miStatement = null;
ResultSet resultado = null;
```

- Establecer la conexión usando el pool de conexiones

```
miConesion = poolConexiones.getConnection();
```

- Crear la sentencia SQL que nos permita obtener los registros y el statement

```
String instruccionSql = "SELECT * FROM grupos";
miStatement = miConesion.createStatement();
```

- Ejecutar la sentencia SQL y guardarla en el objeto ResultSet

```
resultado = miStatement.executeQuery(instruccionSql);
```

- Recorrer el ResultSet obtenido y guardarlo en nuestra lista

```
while (resultado.next()) {
    Grupo grupo = new Grupo(resultado.getInt(1), resultado.getString(2),
    resultado.getInt(3), resultado.getString(4), resultado.getString(5));
    listaGrupos.add(grupo);
    }
```

- Devolver la lista obtenida

```
return listaGrupos;
```

- Como además, cada vez que pulsemos al enlace "Ver más..." de GruposMusicales.jsp debemos recuperar la información de un grupo determinado, necesitamos un método que nos devuelva ese detalle de un grupo específico, así que debemos **crear el método getGrupo** que recibe como parámetro el id del grupo y que retorne el detalle de dicho grupo, es decir, retorna un objeto de tipo Grupo con la información de dicho grupo, además, este método debe lanzar una excepción para obligar a que cuando llamemos a este método se haga dentro de un try-catch y controlar los posibles errores.

```
public Grupo getGrupo(int idGrupo) throws Exception
```

Este método debe realizar lo siguiente:

- Crear un objeto de tipo Grupo

```
Grupo grupoEncontrado = null;
```

Crear los objetos de tipo Connection, PreparedStatement y ResultSet

```
Connection miConesion = null;
PreparedStatement consulta = null;
ResultSet resultado = null;
```

- Establecer la conexión usando el pool de conexiones

```
miConesion = poolConexiones.getConnection();
```

- Crear la sentencia SQL preparada y los parámetros para dicha consulta que nos permita obtener los datos de un grupo

```
consulta = miConesion.prepareStatement("SELECT * FROM grupos
WHERE grupoId=?");
consulta.setInt(1, idGrupo);
```

- Ejecutar la sentencia SQL y guardarla en el objeto ResultSet

```
resultado = consulta.executeQuery();
```

- Comprobar el resultado de nuestra consulta y guardarlo en nuestro objeto de tipo Grupo

```
if(resultado.next()){

    grupoEncontrado = new Grupo(resultado.getInt(1),
    resultado.getString(2), resultado.getInt(3),
    resultado.getString(4), resultado.getString(5));

} else {

    System.out.println("No hay datos para ese grupo");

}
```

- Devolver el registro o datos obtenidos

```
return grupoEncontrado;
```

Una vez que ya tenemos creado nuestro Modelo del patrón MVC, vamos a crear el Controlador, que no es más ni menos que un Servlet. Vamos a **crear el servlet** llamado **ServletEmisora** que se comunicará con nuestro modelo para intercambiar información.

NOTA: Si no recuerdas como crear el servlet, ver el apartado Crear servlet al final del documento.

Que debe tener este Servlet o Controlador:

- Una variable encapsula de tipo ModeloGrupo, que será la encargada de almacenar el el modelo creado anteriormente.

```
private ModeloGrupo modelo;
```

- También vamos a definir la referencia al pool de conexiones, para ello vamos a usar la directiva @Resource, ya que esta directiva nos permite utilizar recursos de nuestro proyecto, en este caso, el pool de conexiones. Para ello, a nivel de clase vamos a incluir la directiva indicando el nombre de nuestro pool de conexiones definido anteriormente.

```
@Resource(name="jdbc/emisora")
```

- Además, necesitamos una variable encapsulada de tipo tipo DataSource (javax.sql) donde almacenamos nuestro pool de conexiones.

```
private DataSource poolConexiones;
```

- Tenemos que incluir también el **método init()** del servlet, que es el método que se va a ejecutar en primer lugar (como el método main de un programa). Dentro de este método tenemos que conectar con el modelo, es decir, creamos un objeto de tipo ModeloGrupo y le pasamos nuestro pool de conexiones.

```
public void init(ServletConfig config) throws ServletException {
    try {
        modelo = new ModeloGrupo(poolConexiones);
    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
        throw new ServletException(e);
    }
}
```

- Por último, tenemos que ir a implementar el **método toGet** del servlet. Como desde este servlet podemos realizar dos acciones (cargar los grupos musicales para la Vista GruposMusicales.jsp y cargar el detalle del grupo para la Vista DetalleGrupo.jsp) vamos a usar algún parámetro para identificar la acción a realizar, y según esa acción realizaremos una tarea u otra. Este método deberá realizar los siguiente:
  - Vamos a crearnos una variable que recupere la acción a realizar de la request. Debemos tener en cuenta, que cuando arrancamos por primera vez, este parámetro no existe, valdrá null.

```
String accion = request.getParameter("accion");
if (accion == null) {
    accion="cargar";
}
```

  - Vamos a evaluar dicha variable, de tal forma que si tiene el valor “cargar” vamos a listar los grupos musicales y vamos a ir a la Vista GruposMusicales.jsp, si no, si el valor es “detalle”, vamos a recuperar el detalle del grupo e ir a la Vista DetalleGrupo.jsp.

```
switch (accion) {
    case "cargar":
        try {
            obtenerGrupos(request, response);
        } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
```

```
}
break;
case "detalle":
try {
        detalleGrupo(request, response);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    break;
    default:
    try {
        obtenerGrupos(request, response);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    break;
}
```

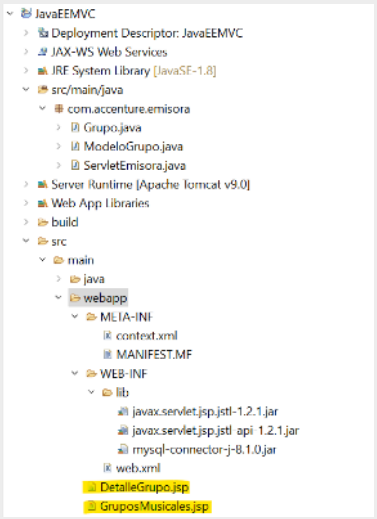
- Creamos el método obtenerGrupos: Este método debe obtener la lista de los grupos desde el modelo, agregar la lista de grupos a la request para poder usarla luego en la Vista y por último, enviar la request a la página jsp.

```
List<Grupo> listaGrupos = modelo.getGrupos();
request.setAttribute("gruposMusicales", listaGrupos);
RequestDispatcher dispatcher = request.getRequestDispatcher("/GruposMusicales.jsp");
dispatcher.forward(request, response);
```

- Creamos el método detalleGrupo: Este método debe obtener los datos de un grupo específico que pasaremos como parámetro en la request, agregar los datos del grupo a la request para poder usarla luego en la Vista y por último, enviar la request a la página jsp.

```
int id = Integer.parseInt(request.getParameter("idGrupo"));

Grupo grupo = modelo.getGrupo(id);

request.setAttribute("detalleGrupo", grupo);

RequestDispatcher dispatcher =
request.getRequestDispatcher("/DetalleGrupo.jsp");

dispatcher.forward(request, response);
```

Una vez que ya tenemos creado nuestro Modelo y Controlador (Servlet) del patrón MVC, vamos a necesitar la Vista, para ello vamos a usar los jsp's **GruposMusicales.jsp y DetalleGrupo.jsp** del proyecto anterior, así que vamos a copiar dichos archivos del proyecto anterior "JavaEEServlets" en nuestro proyecto de la siguiente forma:



Una vez copiados, debemos modificar el fichero **GruposMusicales.jsp** de la siguiente forma:

- Debemos eliminar las directivas import, ya que ahora no van a ser necesarias.
- Debemos eliminar el scriptlet con el código java que realizaba la consulta en la bbdd, ya que esto lo vamos a hacer ahora desde el servlet y modelo.
- Debemos modificar el enlace "Ver más...", ya que ahora en lugar de llamar al jsp DetalleGrupo debe llamar al servlet y pasar como parámetro el id del grupo de cada registro.

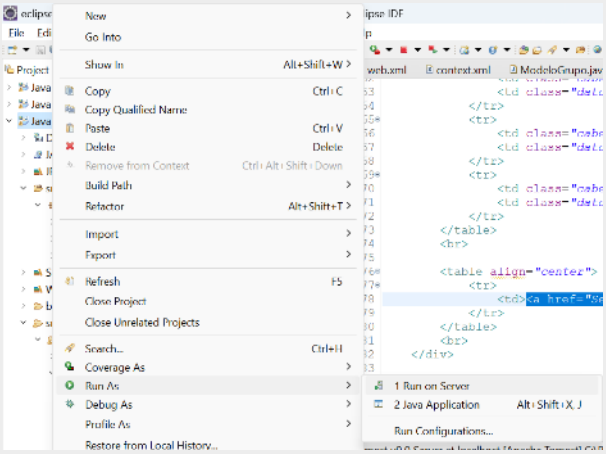
```
<a href="ServletEmisora?accion=detalle&idGrupo=${grupo.id}">Ver más...</a>
```

También debemos modificar el fichero **DetalleGrupo.jsp** de la siguiente forma:

- Debemos modificar el enlace "Volver al listado", ya que ahora en lugar de llamar al jsp GruposMusicales debe llamar al servlet para que se encargue de redirigirnos y cargar la lista de grupos.

```
<a href="ServletEmisora?accion=cargar">Volver al listado</a>
```

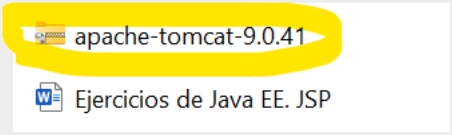
Una vez que tenemos todos nuestros componentes creados y modificados, ya podemos **probar aplicación**. Para probar la aplicación, simplemente debemos ejecutar el con el servidor el proyecto, es decir, sobre el nombre del proyecto con el botón derecho damos a **Run As > Run a Server.**



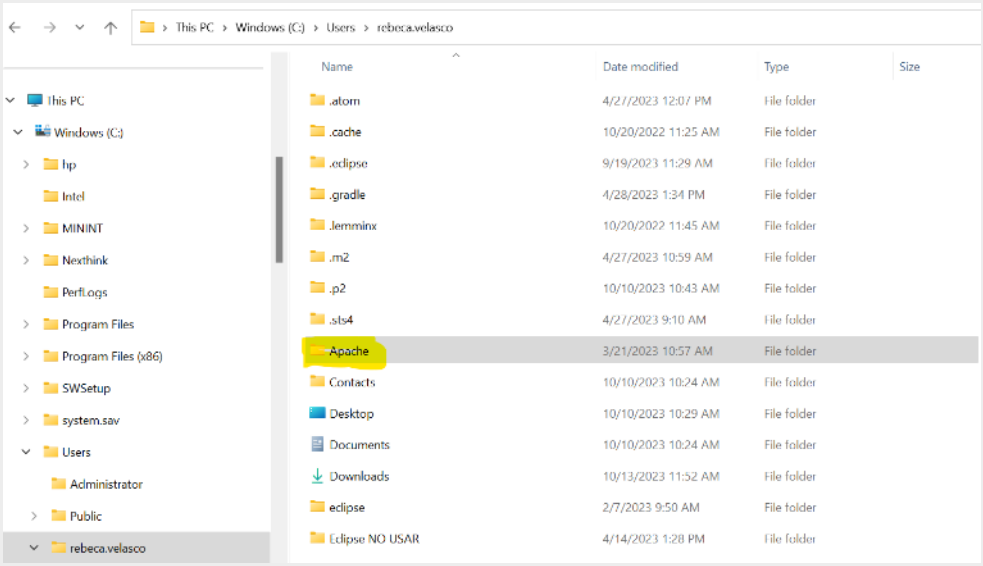
Instalar servidor Tomcat

Para la instalación del servidor, simplemente debes seguir los siguientes pasos:

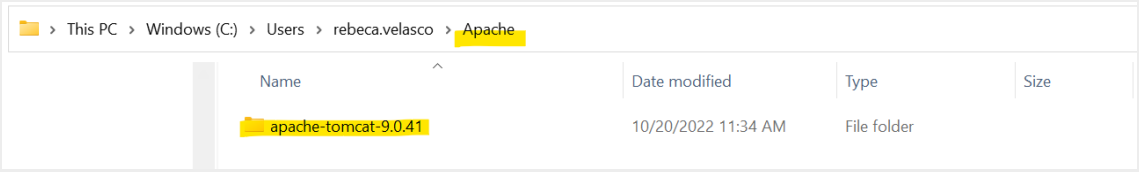
1. Descargar el archivo "apache-tomcat-9.0.41.zip" proporcionado junto a este documento



2. Crea una carpeta en tu local, de la siguiente forma:



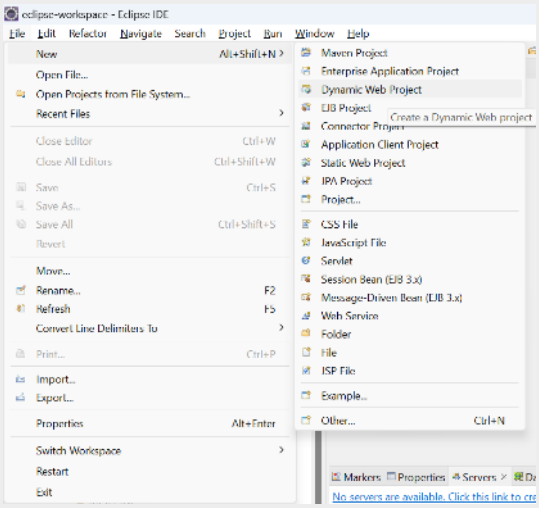
3. Descomprime el contenido del archivo del punto 1 en esa nueva carpeta, quedando de la siguiente forma:



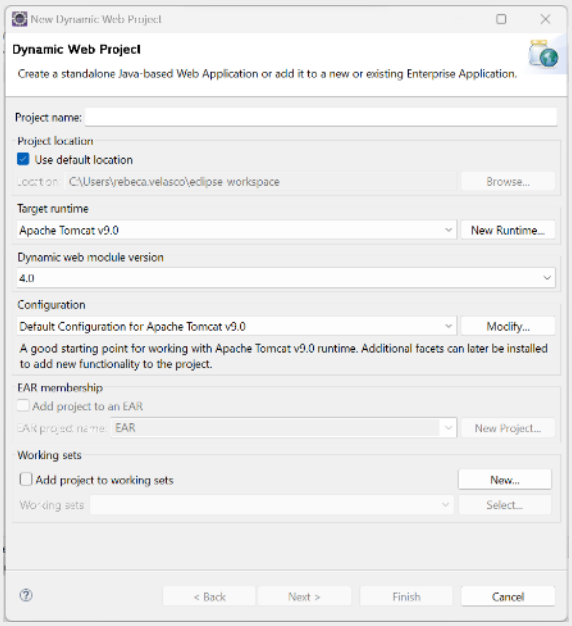
Y con esto, ya tendría el servidor Tomcat instalado y preparado para que lo configures después desde Eclipse al proyecto que necesites.

Crear proyecto web

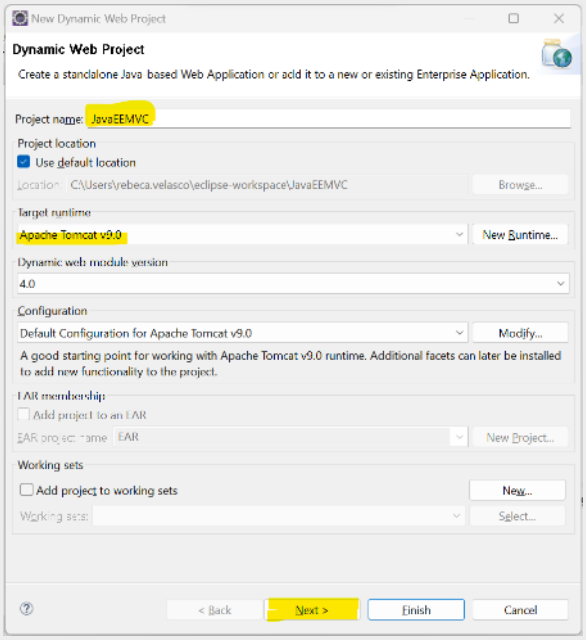
En Eclipse debemos seleccionar la opción de **File > New > Dynamic Web Project**.



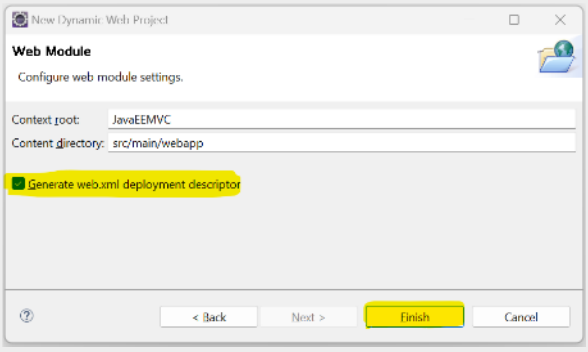
Se abre la siguiente ventana:



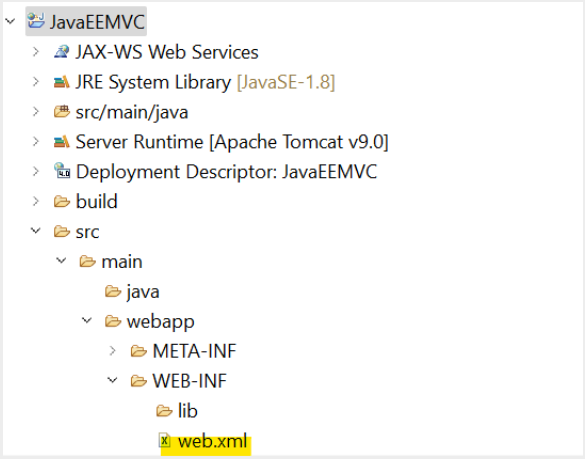
Donde debemos indicar el nombre de nuestro proyecto en Project name e indicar el servidor con el que vamos a ejecutar nuestro proyecto en Target runtime.



Pulsamos el botón Next> hasta llegar a la siguiente ventana:



Seleccionamos la opción de Generarte web.xml deployment descriptor y pulsamos Finish, y termina de construir nuestro proyecto web, con la siguiente estructura:

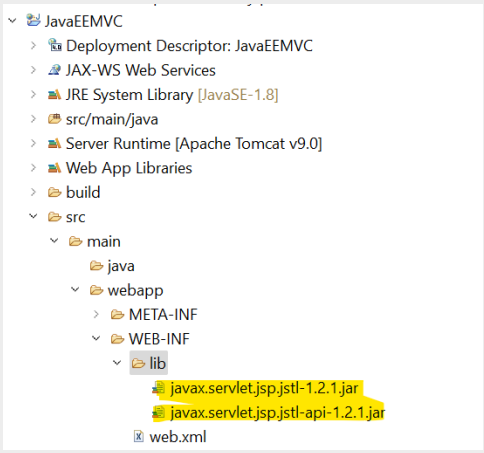


Debemos modificar este fichero web.xml, de tal forma, que indique que cual es el fichero de arranque de la aplicación.

```
<welcome-file-list>
    <welcome-file>ServletEmisora</welcome-file>
</welcome-file-list>
```

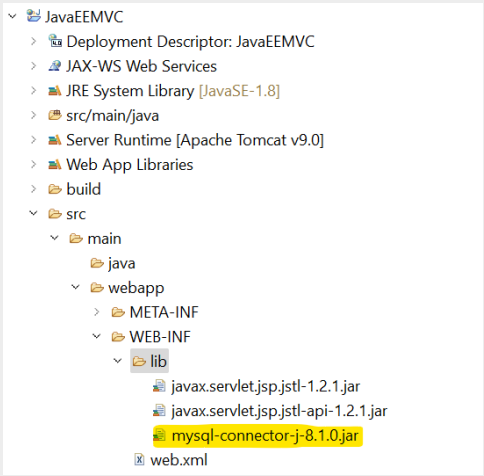
Incluir librerías JSTL

Para incluir las librerías JSTL en el proyecto, simplemente debes copiarlas en la carpeta lib que hay dentro de la carpeta WEB-INF de tu proyecto, quedando de la siguiente forma:



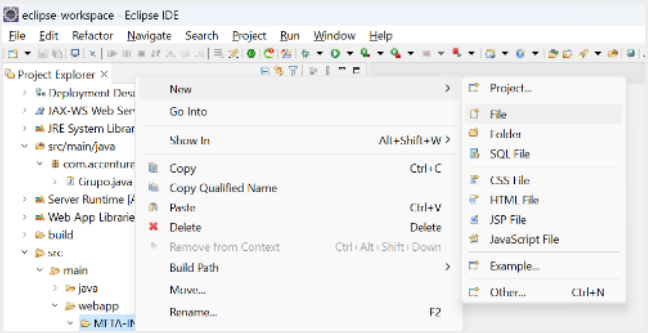
Incluir driver JDBC

Para incluir el driver en el proyecto, simplemente debes copiar el fichero .jar en la **carpeta** lib que hay dentro de la carpeta WEB-INF de tu proyecto, quedando de la siguiente forma:

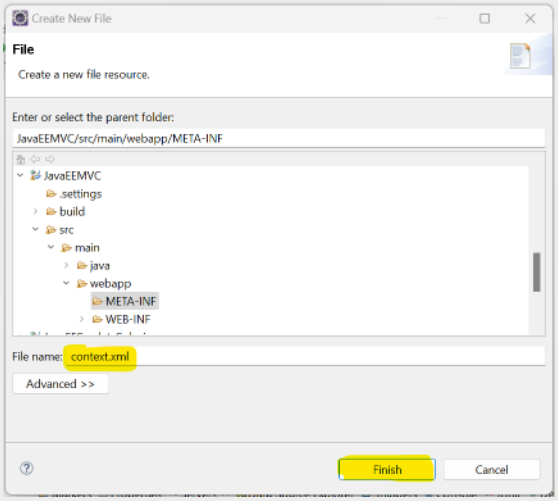


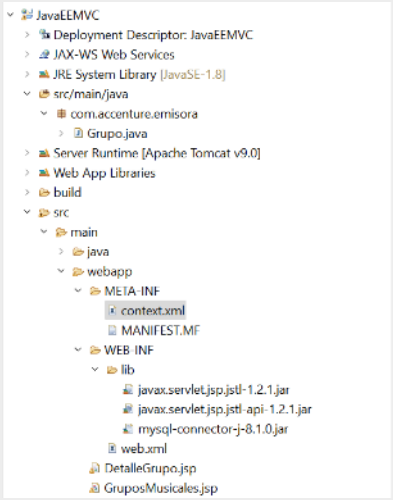
Crear pool de conexiones

Sobre la carpeta META-INF de tu proyecto, seleccionamos **File > New > File**



Le damos el nombre **context.xml** al fichero y pulsamos **Finish**:





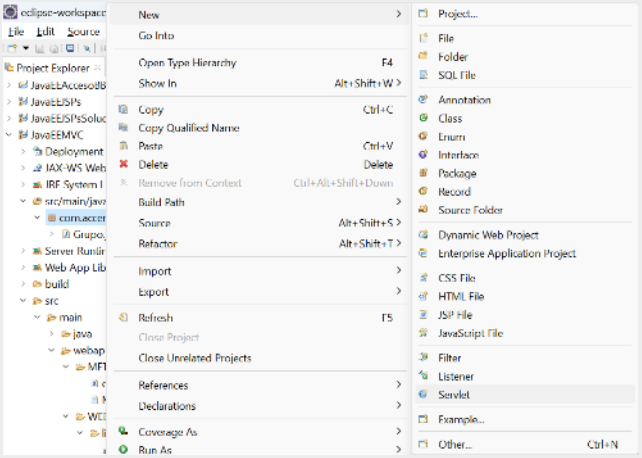
Y dentro del fichero podemos configurar o definir el pool de conexiones de la siguiente forma:

```
<Context>
    <Resource name="jdbc/emisora" auth="Container" type="javax.sql.DataSource"
                maxActive="15" maxIndle="3" maxWait="5000"
                username="root" password="1234abcd"
                driverClassName="com.mysql.cj.jdbc.Driver"
                url="jdbc:mysql//localhost:3306/musicadb2"> >
    </Resource>
</Context>
```

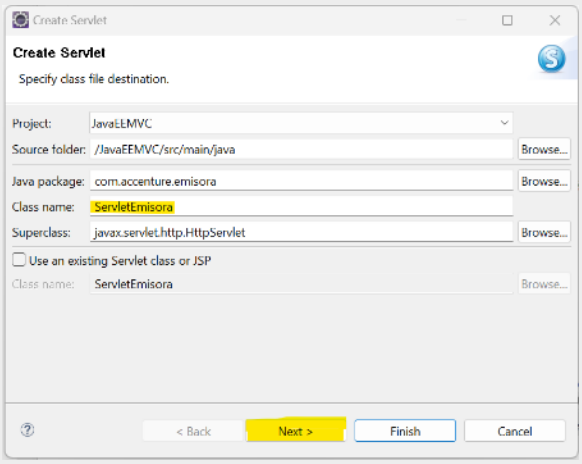
Ten en cuenta el nombre que le damos al pool (atributo name), el username y password (serán los que usamos en nuestra bbdd) y por último, la url de nuestra bbdd.

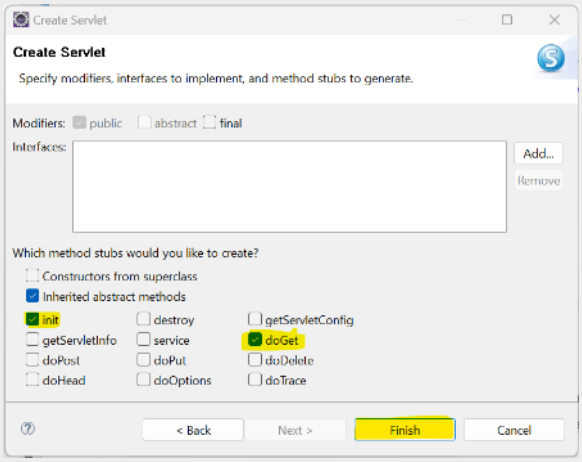
Crear servlet

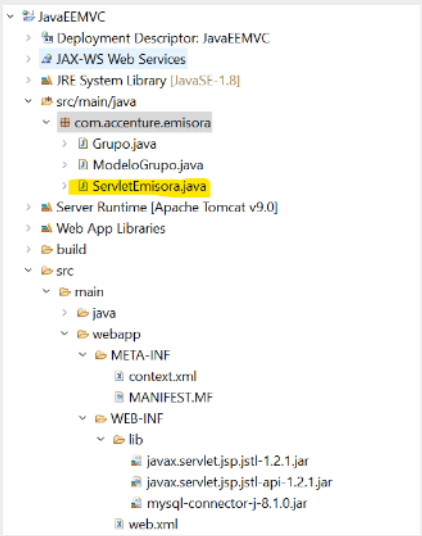
Sobre el paquete com.accenture.emisora, seleccionamos **File > New > Servlet**



Le damos el nombre **ServletEmisora** al fichero, pulsamos **Next >** y nos aseguramos de que están marcados los métodos que queremos implementar, y posteriormente pulsamos **Finish**:







Gracias