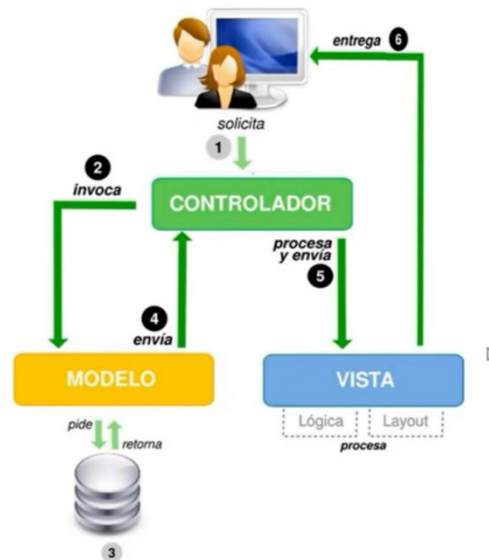
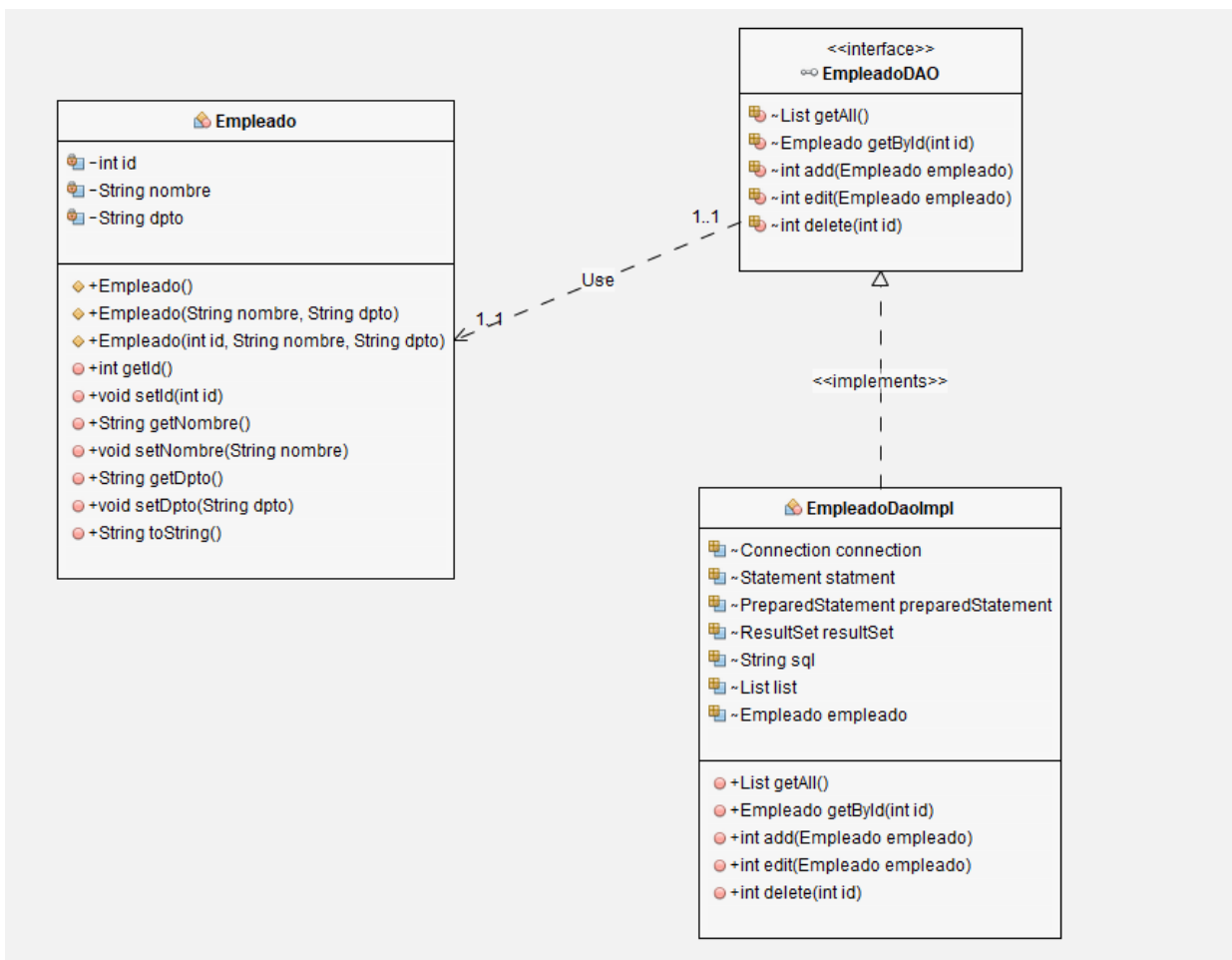


Tutorial ejemplo CRUD con JSP – Servlet – JDBC - MySQL - Maven

MVC



Patrón de diseño DAO



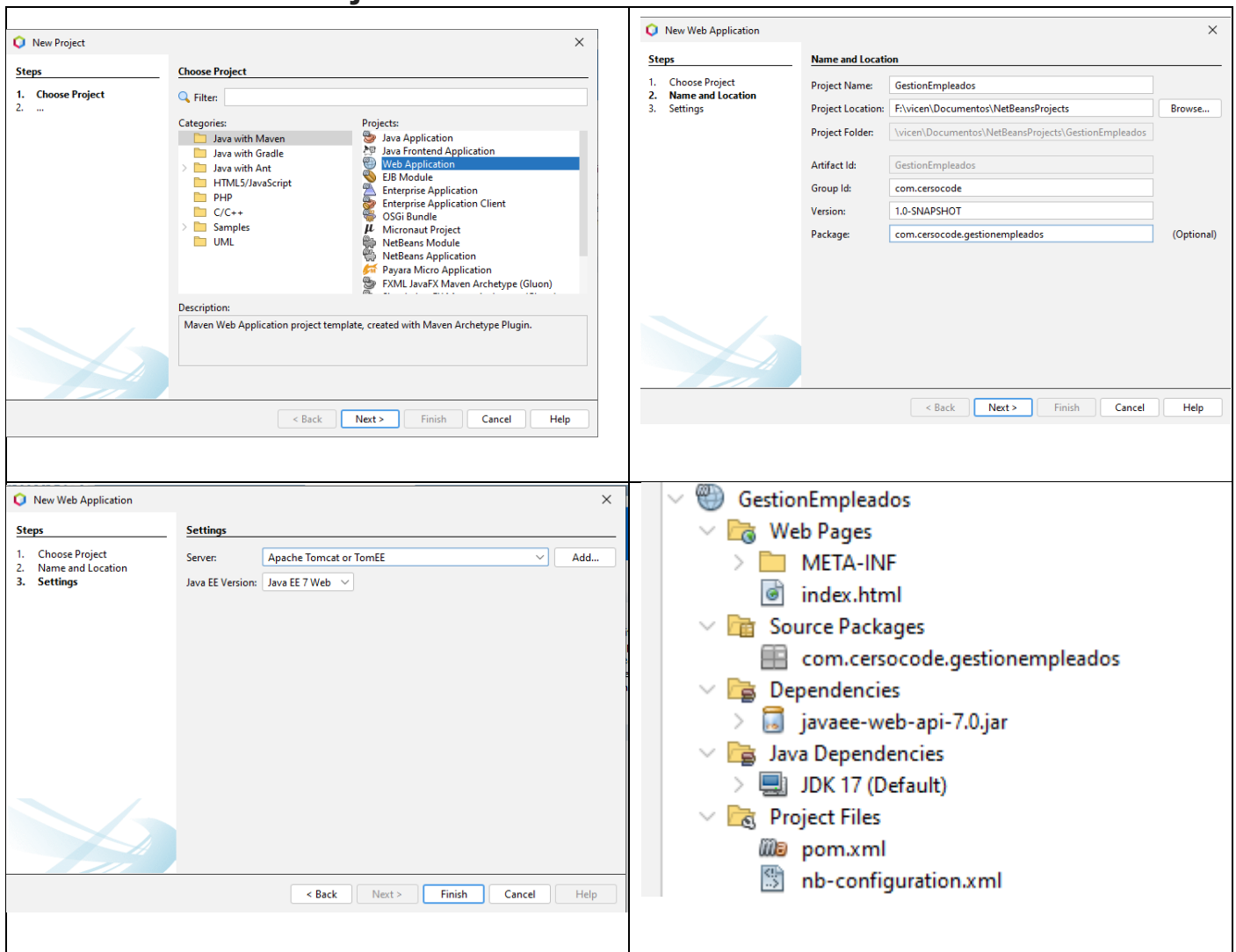
En este tutorial añadiremos el uso de **Maven**, que es una herramienta de software para la gestión de dependencias y construcción de proyectos.

Vamos a crear un pequeño CRUD a una tabla usando MVC, DAO. Creamos un nuevo (Maven Project). Después de ello, crearemos las dependencias y construiremos la estructura de nuestro proyecto que realizará un CRUD a una tabla usando el patrón DAO.

¿Qué es Maven?

Normalmente cuando trabajamos con Java/JavaEE el uso de librerías es algo común como en cualquier otro lenguaje de programación. Sin embargo, una librería puede depender de otras librerías para funcionar de forma correcta. Por ende, su gestión se puede volver tedioso a la hora de buscar las librerías y ver que versión exacta debemos de elegir. Así pues necesitamos más información para gestionarlo todo de forma correcta y para ellos existe MAVEN, que es una herramienta que nos va a permitir tener una gestión correcta de nuestra librerías, proyectos y dependencias. Dentro de un proyecto Maven el archivo *pom.xml* (*Project Object Model*) es el de mayor importancia ya que contiene toda la información del artefacto que usará nuestro proyecto. Un Artefacto puede verse como una librería (aunque agrupa mas conceptos). Contiene las clases propias de la librería pero además incluye toda la información necesaria para su correcta gestión (grupo, versión, dependencias etc).

1. Crea un Web Project con MAVEN



2. Configurar las de pendencias en el archivo pom.xml

Vamos a agregar 4 dependencias que necesitaremos para nuestro proyecto

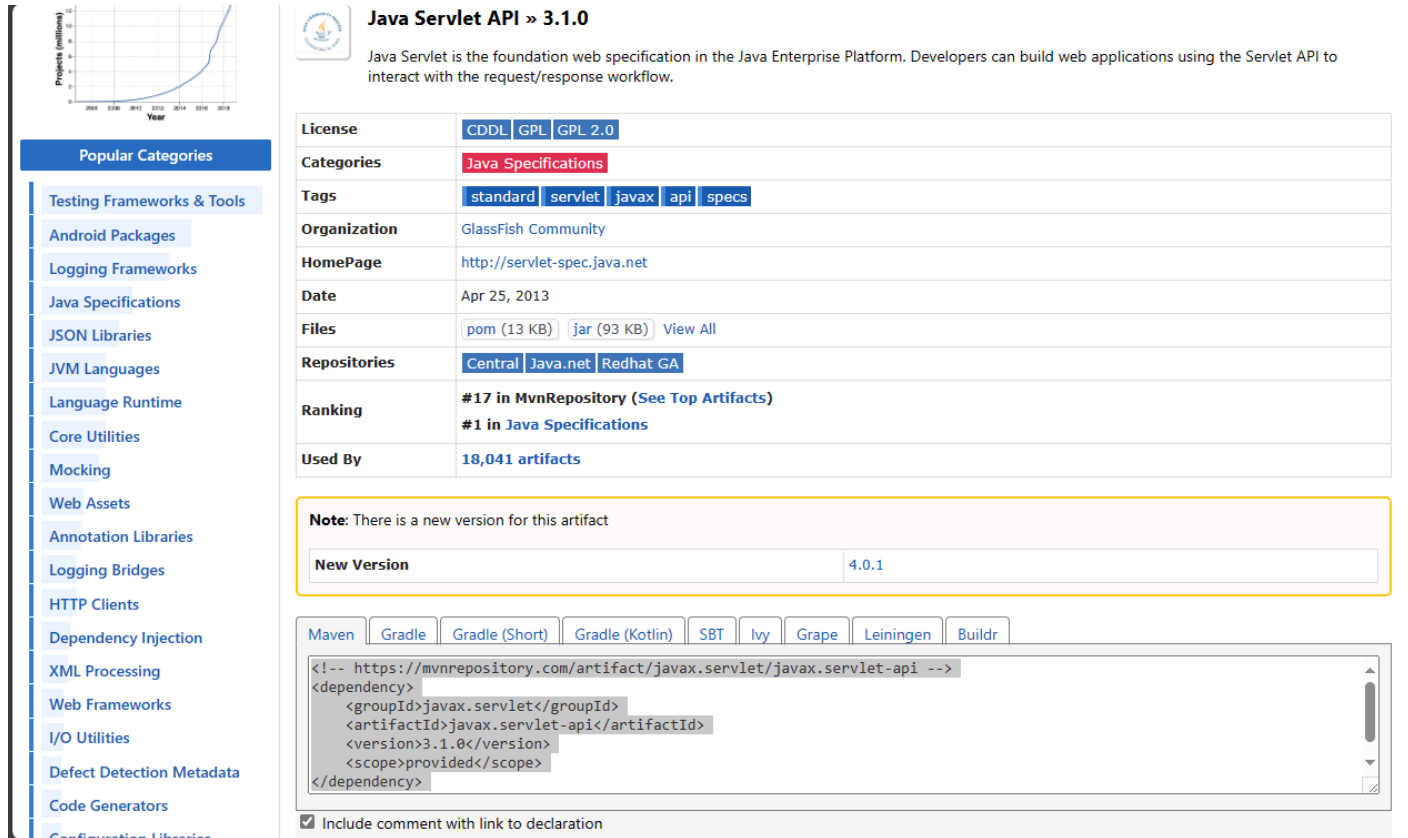
javax.servlet-api: para trabajar con serlvets 3.1.0

mysql-connector-java: driver de conexión mysql 8.0.33

commons-dbcp2: para trabajar con pool de conexiones 2.10.0

jstl: para trabajar con lenguaje de expresiones 1.2

Para agregar estas dependencias podemos hacerla directamente desde IDE o desde el [repositorio de Maven](#). En ambos casos solo tenemos que buscar el nombre del recurso que estamos queriendo agregar y seleccionar una determinada versión. Por ejemplo



The screenshot shows the Maven Central repository page for the **Java Servlet API » 3.1.0**. The page includes a sidebar with popular categories like Testing Frameworks & Tools, Android Packages, and Logging Frameworks. The main content area displays the artifact details, including the license (CDDL, GPL, GPL 2.0), categories (Java Specifications), tags (standard, servlet, javax, api, specs), organization (GlassFish Community), homepage (http://servlet-spec.java.net), date (Apr 25, 2013), files (pom (13 KB), jar (93 KB)), repositories (Central, Java.net, Redhat GA), ranking (#17 in MvnRepository, #1 in Java Specifications), and used by (18,041 artifacts). A note indicates a new version (4.0.1) is available. The code editor shows the dependency declaration in a pom.xml file:

```
<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>provided</scope>
</dependency>
```

Una vez seleccionada la agregamos al archivo **pom.xml** en el tag de **dependencies**

Importante Modificar la versión **maven-war-plugin** a **3.4.0**

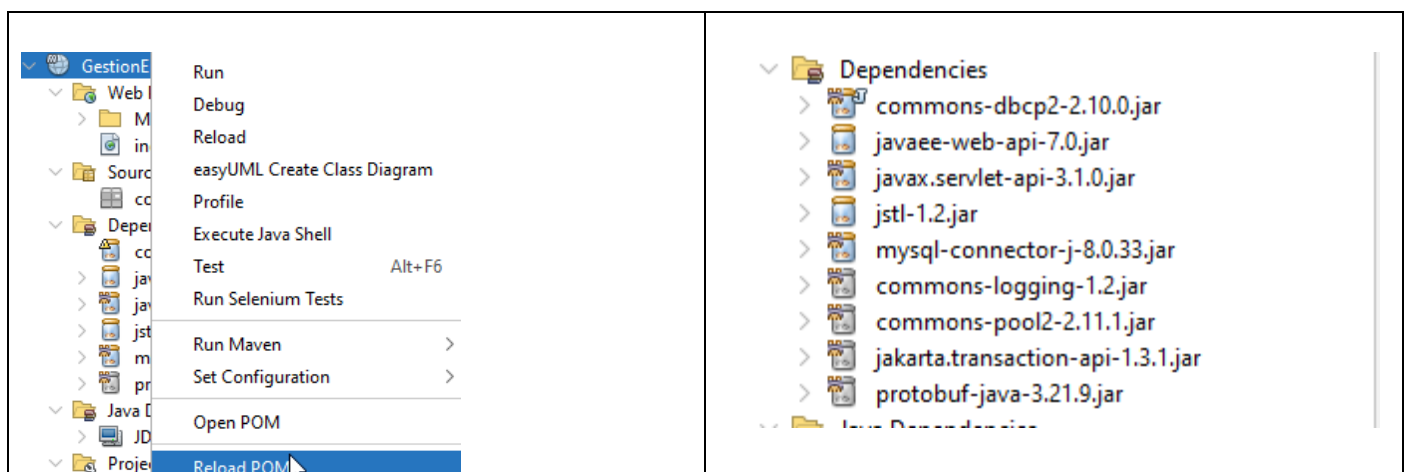
Debe quedar Así

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-war-plugin</artifactId>
  <version>3.4.0</version>
</plugin>
```

Dependencias

```
<dependencies>
  <dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>7.0</version>
    <scope>provided</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.33</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcp2</artifactId>
    <version>2.10.0</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
</dependencies>
```

Como podemos apreciar en el código se ha adicionado las 4 dependencias que necesitaremos para nuestro proyecto. Si necesitáramos de otras dependencias, debemos repetir el mismo proceso e ir agregándolas una por una. Guardamos y recargamos el archivo pom.xml



3) Creamos la pagina **index.jsp** y ejecutamos el proyecto



4. Crear la estructura del proyecto

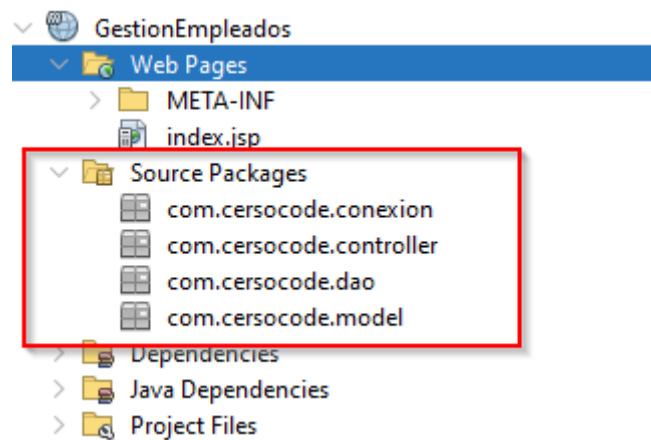
1) Creamos lo siguientes paquetes

com.cersocode.conexion: aquí estará nuestra conexión

com. cersocode.model: aquí irán los DTOs (Data Transfer Object)

com. cersocode.dao: aquí irán nuestros DAOs(Data Access Object)

com. cersocode.controller: aquí irán nuestros servlets



5. Crear la conexión a una base de datos MySQL

1) Crear la tabla **empleados**

```
CREATE TABLE empleados(  
  id int NOT NULL AUTO_INCREMENT,  
  nombre varchar(50) NULL,  
  departamento varchar(50), NULL,  
  PRIMARY KEY (id)  
);
```

2) Crear la clase **Conexion.java** (Se crea un pool de conexiones, investigá sobre este tema)

```
package com.cersocode.conexion;

import java.sql.Connection;
import java.sql.SQLException;
import javax.sql.DataSource;
import org.apache.commons.dbcp2.BasicDataSource;

public class Conexion {
    private static BasicDataSource dataSource = null;

    private static DataSource getDataSource() {
        if (dataSource == null) {
            dataSource = new BasicDataSource();
            dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
            dataSource.setUsername("root");
            dataSource.setPassword("");
            dataSource.setUrl("jdbc:mysql://localhost:3306/empresa?useTimezone=true&serverTimezone=UTC");
            dataSource.setInitialSize(20);
            dataSource.setMaxIdle(15);
            dataSource.setMaxTotal(20);
            dataSource.setMaxWaitMillis(5000);
        }
        return dataSource;
    }

    public static Connection getConnection() throws SQLException {
        return getDataSource().getConnection();
    }
}
```

6. Crear el modelo

1) Crear la clase **Empleado.java** con sus propiedades, constructores y métodos de acceso getters and setters

```
public class Empleado {

    private int id;
    private String nombre;
    private String departamento;

    //constructores
    //setter
    //getter
    //toString()
}
```

7. Crear el DAO

- 1) Crear la interface ***EmpleadoDAO.java*** con los métodos para el CRUD
- 2) Crear la clase ***EmpleadoDAOImpl.java*** que implemente la interface

EmpleadoDAO.java

```
import com.cersocode.model.Empleado;
import java.util.List;

public interface EmpleadoDAO {

    // guardar empleado
    boolean guardar(Empleado empleado);

    // editar empleado
    boolean editar(Empleado empleado);

    // eliminar empleados
    boolean eliminar(int idEmpleado);

    // obtener lista de empleados
    public List<Empleado> obtenerProductos();

    // obtener empleado
    Empleado obtenerProducto(int idEmpleado);

}
```

EmpleadoDAOImpl.java

```
import java.util.List;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import com.cersocode.conexion.Conexion;
import com.cersocode.model.Empleado;

/**
 *
 * @author vicec
 */
public class EmpleadoDAOImpl implements EmpleadoDAO {

    private Connection connection;
    private PreparedStatement statement;
```

```

private boolean estadoOperacion;

// obtener conexion pool
private Connection obtenerConexion() throws SQLException {
    return Conexion.getConnection();
}

@Override
public boolean guardar(Empleado empleado) {
    String sql = null;
    estadoOperacion = false;

    try {
        connection = obtenerConexion();
        connection.setAutoCommit(false);
        sql = "INSERT INTO empleados (id, nombre, dpto)VALUES(?,?,?)";
        statement = connection.prepareStatement(sql);
        statement.setInt(1, 0);
        statement.setString(2, empleado.getNombre());
        statement.setString(3, empleado.getDepartamento());
        estadoOperacion = statement.executeUpdate() > 0;
        connection.commit();
        statement.close();
        connection.close();
    } catch (SQLException e) {
        try {
            connection.rollback();
        } catch (SQLException ex) {
        }
        e.printStackTrace();
    }
    return estadoOperacion;
}

@Override
public boolean editar(Empleado empleado) {
    String sql = null;
    estadoOperacion = false;

    try {
        connection = obtenerConexion();
        connection.setAutoCommit(false);
        sql = "UPDATE empleados SET nombre=?, dpto=? WHERE id=?";
        statement = connection.prepareStatement(sql);
        statement.setString(1, empleado.getNombre());
        statement.setString(2, empleado.getDepartamento());
        statement.setInt(3, empleado.getId());
        estadoOperacion = statement.executeUpdate() > 0;
        connection.commit();
        statement.close();
        connection.close();
    } catch (SQLException e) {
        try {

```



```

        connection.rollback();
    } catch (SQLException ex) {

    }
    e.printStackTrace();
}

return estadoOperacion;
}

@Override
public boolean eliminar(int idEmpleado) {
    String sql = null;
    estadoOperacion = false;

    try {
        connection = obtenerConexion();
        connection.setAutoCommit(false);
        sql = "DELETE FROM empleados WHERE id=?";
        statement = connection.prepareStatement(sql);
        statement.setInt(1, idEmpleado);
        estadoOperacion = statement.executeUpdate() > 0;
        connection.commit();
        statement.close();
        connection.close();
    } catch (SQLException e) {
        try {
            connection.rollback();
        } catch (SQLException ex) {
        }
        e.printStackTrace();
    }

    return estadoOperacion;
}

@Override
public List<Empleado> obtenerEmpleados() {
    ResultSet resultSet = null;
    List<Empleado> listaEmpleados = new ArrayList<>();
    String sql = null;
    estadoOperacion = false;

    try {
        sql = "SELECT * FROM empleados";
        connection = obtenerConexion();
        statement = connection.prepareStatement(sql);
        resultSet = statement.executeQuery(sql);
        while (resultSet.next()) {
            Empleado e = new Empleado();
            e.setId(resultSet.getInt(1));
            e.setNombre(resultSet.getString(2));
            e.setDepartamento(resultSet.getString(3));

```

```

        listaEmpleados.add(e);
    }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return listaEmpleados;
}

@Override
public Empleado obtenerEmpleado(int idEmpleado) {
    ResultSet resultSet = null;
    Empleado emp = new Empleado();

    String sql = null;
    estadoOperacion = false;

    try {
        sql = "SELECT * FROM empleados WHERE id =?";
        connection = obtenerConexion();
        statement = connection.prepareStatement(sql);
        statement.setInt(1, idEmpleado);

        resultSet = statement.executeQuery();

        if (resultSet.next()) {
            emp.setId(resultSet.getInt(1));
            emp.setNombre(resultSet.getString(2));
            emp.setDepartamento(resultSet.getString(3));
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return emp;
}
}

```

8. Crear el servlet controlador

1) Crear la clase ***EmpleadoController.java*** con sus respectivos métodos **doGet()** y **doPost()** para administrar las peticiones para la tabla **empleados**

```
@WebServlet(name = "EmpleadoController", urlPatterns = {"/EmpleadoController"})
public class EmpleadoController extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String opcion = request.getParameter("opcion");

        if (opcion.equals("crear")) {
            System.out.println("Usted a presionado la opcion crear");
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("/views/crear.jsp");
            requestDispatcher.forward(request, response);

        } else if (opcion.equals("listar")) {
            EmpleadoDAO empleadoDAO = new EmpleadoDAOImpl();
            List<Empleado> lista = new ArrayList<>();
            lista = empleadoDAO.obtenerEmpleados(); // TODO Auto-generated catch block
            for (Empleado empleado : lista) {
                System.out.println(empleado);
            }
            request.setAttribute("lista", lista);
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("/views/listar.jsp");
            requestDispatcher.forward(request, response);

            System.out.println("Usted a presionado la opcion listar");

        } else if (opcion.equals("editar")) {
            int id = Integer.parseInt(request.getParameter("id"));
            System.out.println("Editar id: " + id);
            EmpleadoDAO empleadoDAO = new EmpleadoDAOImpl();
            Empleado emp = new Empleado();
            emp = empleadoDAO.obtenerEmpleado(id);
            System.out.println(emp);
            request.setAttribute("empleado", emp);
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("/views/editar.jsp");
            requestDispatcher.forward(request, response);

        } else if (opcion.equals("eliminar")) {
            EmpleadoDAO empleadoDAO = new EmpleadoDAOImpl();
            int id = Integer.parseInt(request.getParameter("id"));

            empleadoDAO.eliminar(id);
            System.out.println("Registro eliminado satisfactoriamente...");
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("/index.jsp");
            requestDispatcher.forward(request, response);

        }
    }
}
```

```

        // response.getWriter().append("Served at: ").append(request.getContextPath())
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        String opcion = request.getParameter("opcion");

        if (opcion.equals("guardar")) {
            EmpleadoDAO empleadoDAO = new EmpleadoDAOImpl();
            Empleado empleado = new Empleado();
            empleado.setNombre(request.getParameter("nombre"));
            empleado.setDepartamento(request.getParameter("departamento"));

            empleadoDAO.guardar(empleado);
            System.out.println("Registro guardado satisfactoriamente...");
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("/index.jsp");
            requestDispatcher.forward(request, response);

        } else if (opcion.equals("editar")) {
            Empleado empleado = new Empleado();
            EmpleadoDAO empleadoDAO = new EmpleadoDAOImpl();

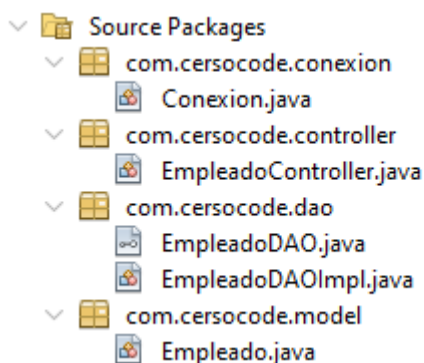
            empleado.setId(Integer.parseInt(request.getParameter("id")));
            empleado.setNombre(request.getParameter("nombre"));
            empleado.setDepartamento(request.getParameter("departamento"));

            empleadoDAO.editar(empleado);
            System.out.println("Registro editado satisfactoriamente...");
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("/index.jsp");
            requestDispatcher.forward(request, response);
        }

        // doGet(request, response);
    }
}

```

Hasta aquí hemos terminado toda parte de Java



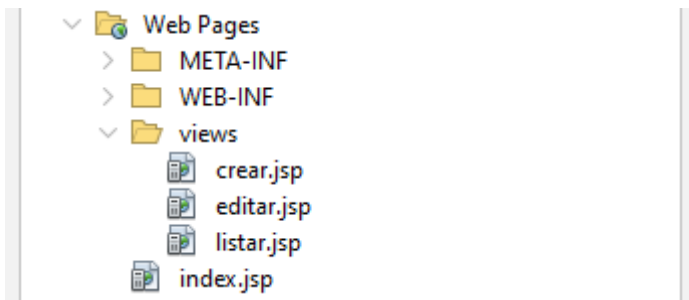
9. Crear los JSPs

1) Crear las siguientes páginas dentro de la carpeta **/views**

crear.jsp: interfaz para crear un nuevo producto

listar.jsp: interfaz para listar los productos

editar.jsp: interfaz para actualizar un producto



2) Actualizar la página **index.jsp** para invocar al servlet **EmpleadoController** y poder redireccionar a las diferentes páginas

```
<h1>Menu de opciones Empleados</h1>
<table border="1">
  <tr>
    <td><a href="EmpleadoController?opcion=crear"> Crear un Empleado</a></td>
  </tr>
  <tr>
    <td><a href="EmpleadoController?opcion=listar"> Listar Empleados</a></td>
  </tr>
</table>
```

3) Editar la página **crear.jsp**

```
<body>
  <h1>Crear Empleado</h1>
  <form action="EmpleadoController" method="post">
    <input type="hidden" name="opcion" value="guardar">
    <table border="1">
      <tr>
        <td>Nombre:</td>
        <td><input type="text" name="nombre" size="50"></td>
      </tr>
      <tr>
        <td>Departamento:</td>
        <td><input type="text" name="departamento" size="50"></td>
      </tr>
    </table>
    <input type="submit" value="Guardar">
  </form>
</body>
```

4) Editar la página *listar.jsp* (agregar el taglib *jstl*)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Listar Empleados</title>
    </head>
    <body>
        <h1>Listar Empleados</h1>
        <table border="1">
            <tr>
                <td>Id</td>
                <td>Nombre</td>
                <td>Departamento</td>
                <td>Accion</td>
            </tr>
            <c:forEach var="empleado" items="${lista}">
                <tr>
                    <td>
                        <a href="EmpleadoController?opcion=editar&id=<c:out value="${empleado.id}"></c:out>">
                            <c:out value="${empleado.id}"></c:out>
                        </a>
                    </td>
                    <td><c:out value="${empleado.nombre}"></c:out></td>
                    <td><c:out value="${empleado.departamento}"></c:out></td>
                    <td>
                        <a href="EmpleadoController?opcion=eliminar&id=<c:out value="${empleado.id}"></c:out>">
                            Eliminar
                        </a>
                    </td>
                </tr>
            </c:forEach>
        </table>
    </body>
</html>
```

5) Editar la página *editar.jsp* (agregar el taglib *jstl*)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Editar Empleado</title>
    </head>
    <body>
        <h1>Editar Empleado</h1>
        <form action="EmpleadoController" method="post">
            <c:set var="empleado" value="${empleado}"></c:set>
            <input type="hidden" name="opcion" value="editar">
            <input type="hidden" name="id" value="${empleado.id}">
            <table border="1">
                <tr>
                    <td>Nombre:</td>
                    <td><input type="text" name="nombre" size="50" value="${empleado.nombre}"></td>
                </tr>
                <tr>
                    <td>Departamento:</td>
                    <td><input type="text" name="departamento" size="50" value="${empleado.departamento}"></td>
                </tr>
            </table>
            <input type="submit" value="Guardar">
        </form>
    </body>
</html>
```

10. Ejecutar la Aplicación y probar las funcionalidades

Listar Empleados

Id	Nombre	Departamento	Accion
1	Jhon	Electomecanica	Eliminar
2	Ada	Informatica	Eliminar
3	Adela	Ventas	Eliminar
6	Mirtha	Desarrollo	Eliminar

Listar Empleados

Id	Nombre	Departamento	Accion
1	Jhon	Electomecanica	Eliminar
2	Ada	Informatica	Eliminar
3	Adela	Ventas	Eliminar
6	Mirtha	Desarrollo	Eliminar
7	Mony	Idiomas	Eliminar

Crear Empleado

Nombre:	<input type="text" value="Mony"/>
Departamento:	<input type="text" value="Idiomas"/>
<input type="button" value="Guardar"/>	

Listar Empleados

Id	Nombre	Departamento	Accion
1	Jhon	Electomecanica	Eliminar
2	Ada	Informatica	Eliminar
3	Adela	Ventas	Eliminar
7	Mony	Idiomas	Eliminar