	UNIVERSIDAD DON BOSCO **ESCUELA DE COMPUTACIÓN**
CICLO I	**GUIA DE LABORATORIO** **Programación Orientada a Objetos** **JSP y JDBC**

I.OBJETIVOS

Que el estudiante:

- Pueda crear e implementar paginas JSP con IntelliJ Idea.
- Cree y manipule sesiones con JSP.

II. INTRODUCCION

Versiones y compatibilidad.

Antes de iniciar con el desarrollo de la presente guía de laboratorio, es necesario hablar un poco acerca de ciertos puntos relevantes sobre el entorno de desarrollo IntellJ IDEA. Como se le menciono en la guía anterior, debido a ciertos cambios en el lenguaje de java como tal, varios directorios conocidos bajo el nombre de Oracle, fueron cambiados por el nuevo dueño de esos empaquetados, que en este caso fue la fundación Apache, quien se vio obligada a cambiar el nombre de estos directorios; por ende entonces los paquetes que antes venían nombrados bajo el directorio javax, cambiaron a jakarta.

Hasta el momento entonces y como se notó en uno de los pasos de la guía anterior este cambio se ve no solo en las importaciones como tal, sino también en la implementación de ciertas tecnologías como las que se estarán viendo en la presente guía. Estamos hablando entonces de las **JSP**, que antes llevaban el nombre de **Java Server Pages**, y que ahora el nombre que reciben corresponde al nombre de los nuevos directorios, por ende serán llamadas **Jakarta Server Pages.**

Jakarta Server Pages (JSP) (http://java.sun.com/jsp) es una tecnología basada en el lenguaje Java que permite incorporar contenido dinámico a las páginas web. Los archivos JSP combinan HTML conetiquetas especiales y fragmentos de código Java.

Una página JSP es un tipo especial de servlet que puede poseer código HTML y que contiene ciertas etiquetas especiales (acciones) o fragmentos de código (scriptlets) para incluir contenido dinámico en dichas páginas. El código HTML se añade tal cual, a la respuesta, mientras que los

elementos dinámicos se evalúan con cada petición.

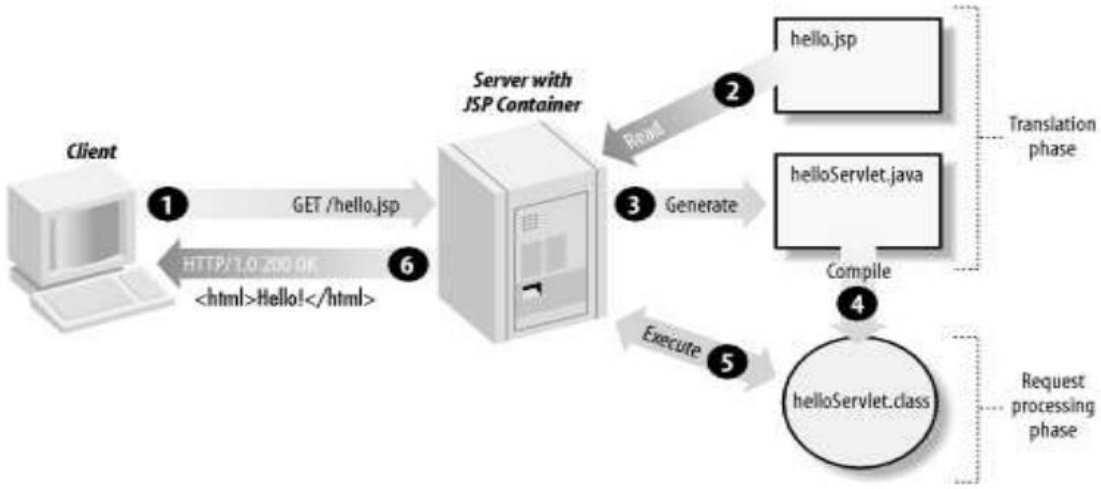


Figura 1: Petición de una página JSP

Cuando un cliente pide esa página al servidor de aplicaciones, se traduce a un fichero fuente de Java que implementa un Servlet, se compila y se ejecuta para devolver el resultado de la petición.

Una vez compilado el servlet, las peticiones posteriores se reducen a ejecutar dicho servlet en lugar de realizar el proceso una y otra vez. Esta es la razón por la que la primera petición tarda mucho más en responderse que las siguientes.
Dado que una JSP no es más que un servlet, hay que recordar que todo el código se ejecuta en el servidor, las respuestas son puramente páginas HTML.

Código Java
Poademos insertar código Java dentro de JSP de tres formas: Expresiones, scriptlets y declaraciones.

Expresiones: Son fragmentos de código Java, con la forma **<%= expresión %>** que se evalúan y se muestran en la salida del navegador. En general, dentro de una expresión podemos usar cualquier cosa que usaríamos dentro de un System.out.print(expr);

Scriptlets: Son fragmentos de código Java con la forma **<% código %>**, en general, podemos insertar cualquier código que pudiéramos usar dentro de una función Java. Para acceder a la salida del navegador, usamos el objeto implícito out.

Declaraciones: Contienen declaraciones de variables o métodos, con la forma **<%! declaración %>.** Estas variables o métodos serán accesibles desde cualquier lugar de la página JSP. Hay que tener en cuenta que el servidor transforma la página JSP en un servlet, y éste es usado por múltiples peticiones, lo que provoca que las variables conserven su valor entre sucesivas ejecuciones.

Directivas: Las directivas son elementos que proporcionan información al motor JSP, e influirán en la estructura del servlet generado. Hay tres tipos de directivas: page, taglib e include.

page: Se indica con la forma <%@ page atributo="valor">. Tiene diversos usos, entre los cuales destacaremos:

- Importar clases. Importar código, de la misma forma que se realiza en un programa en Java, se indica con el atributo import.
  Ejemplo:

<%@page import="java.io.*, miPackage.miClase"%>

- Indicar si la página tendrá acceso a la sesión. Se especifica con el atributo session.
  Ejemplo:

<%@page session="true" %>

- Gestión de errores. Permite redireccionar a una página cuando se produzca un error, se indica con los atributos errorPage e isErrorPage.

  Ejemplos:

```
<%@page errorPage="error.jsp">
[...]
<%@page isErrorPage="true">
<html>
<body>
Error, contacte con el administrador [...]
</body>
</html>
```

Include: Permite incluir un archivo en el lugar donde se especifique, al contrario que con la acción <jsp:include> que veremos más adelante, la directiva include simplemente copia el contenido del archivo byte a byte, siendo el resultado similar a si copiáramos el texto del archivo incluido y lo pegáramos en el JSP.
taglib: Se emplea para indicar que se van a emplear librerías de etiquetas. Se verá con más detalle en la siguiente sesión de clases.
Ejemplo:
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

Acciones

Las acciones tienen la forma <jsp:accion [parámetros]/>, y tienen diversos usos, entre los que destacan la inclusión de páginas y transferencia de control.

Inclusión de páginas

Se realiza con la acción **<jsp:include page="pagina.jsp">.** Incluye la salida de otra página JSP en la actual, al contrario que con la directiva <%@include file="fichero.ext"%>, la página incluida

se ejecuta y su salida se inserta en la página que la incluye, con la directiva se incluye el contenido del archivo (no su salida) y se ejecuta conjuntamente con la página principal.

La página incluida tiene acceso a los parámetros enviados a la principal, y podemos enviarle nuevosparámetros con la subetiqueta <jsp:param name="nombre" value="valor"/>.

Transferencia de control
Se realiza con la acción <jsp:forward page="pagina.jsp"/>. La petición es redirigida a otra página, y la salida de la actual se descarta. Al igual que con la inclusión, la página a la que se redirige tiene acceso a los parámetros pasados a la actual, y es posible el envío de nuevos parámetros.
Ejemplo:
<jsp:forward page="principal.jsp">
<jsp:param name="titulo" value="Principal"/>
</jsp:forward>

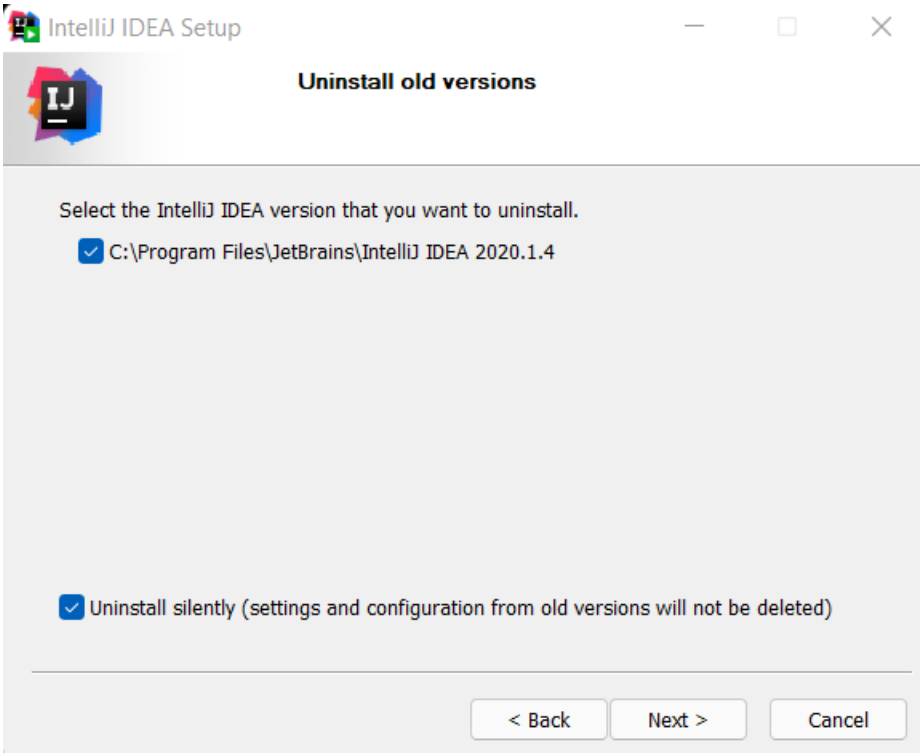
III. PROCEDIMIENTO

Cambio a una versión estable para Jakarta y sus nuevos empaquetados.

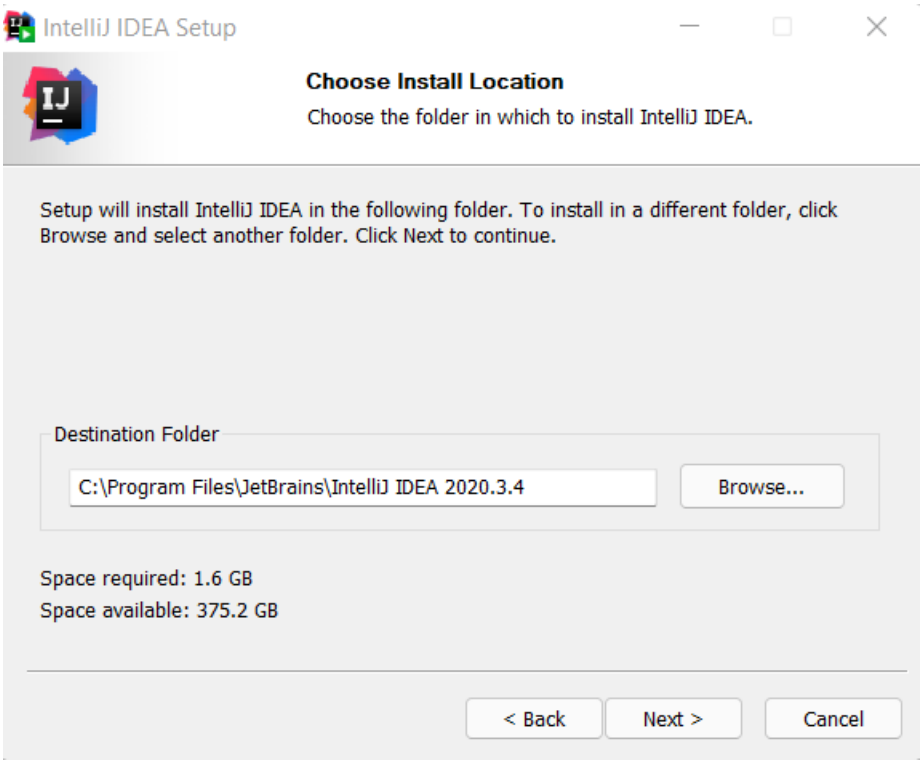
Paso 1: Descargar y actualizar el IDE a una nueva versión más estable y compatible con los directorios de Jakarta.

1- Hasta el momento la última versión estable para crear nuestras jakarta server pages es la versión 2020.3.4. Por lo cual será necesario ir al sitio oficial de IntelliJ IDEA y descargar esa versión.

2- Proceder a correr el archivo ejecutable, con lo cual se le solicitara desinstalar la versión anterior.



3- Podrá ver inmediatamente los pasos de instalación para la nueva versión. Para lo cual solo deberá seguir los pasos mostrados en la primera guía de instalación del IDE. Cuando la instalación haya terminado proceda a correr el entorno de desarrollo.

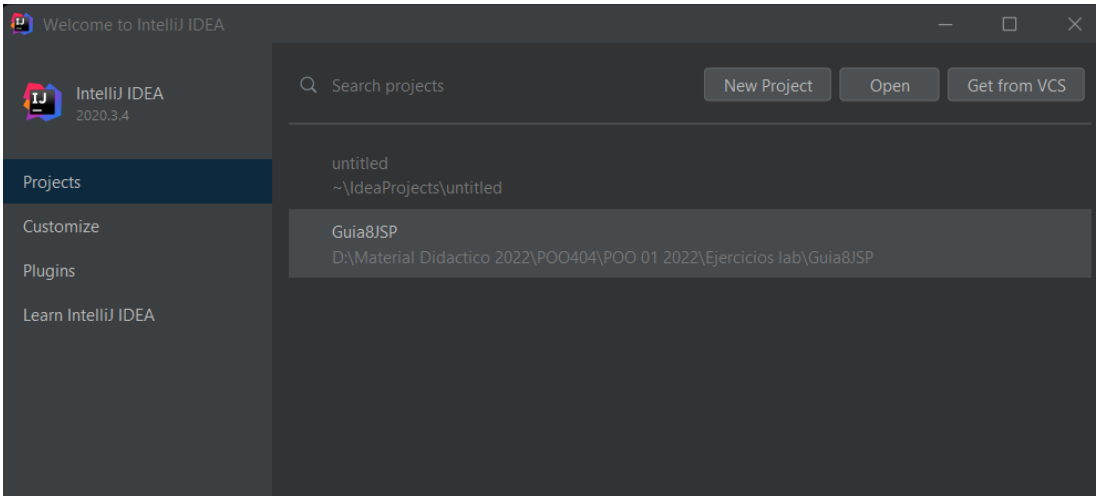


NOTA: Es necesario aclarar que, a pesar de cambiar la versión, todos sus proyectos anteriores no serán eliminados. También la sesión y activación del producto no es necesaria, dado que todo eso ya fue realizado en las primeras semanas. En caso se le solicite activar el producto, proceda a ingresar con las credenciales de su cuenta de JetBrains.

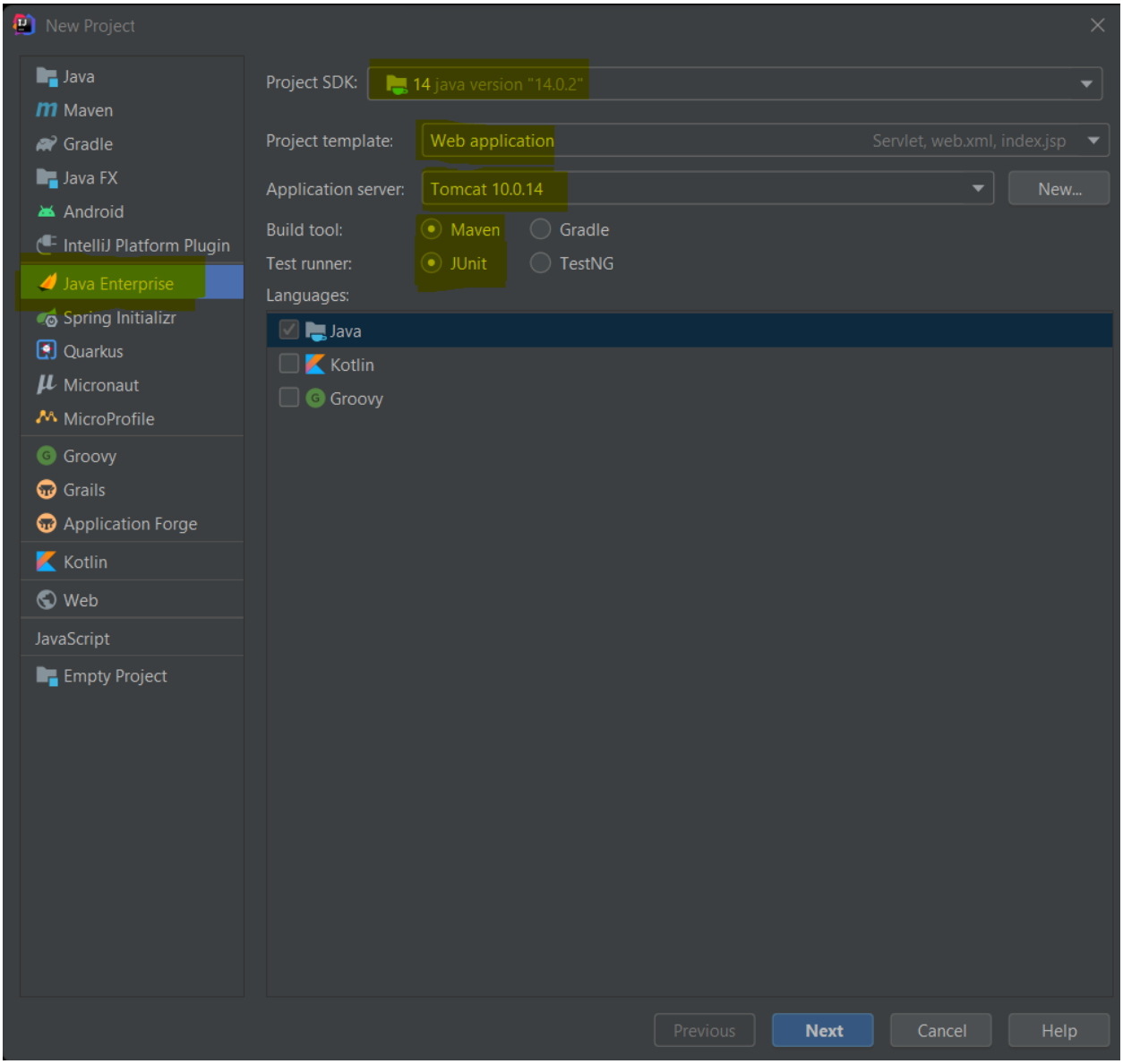
4- Su nueva versión esta lista para ser usada.

Paso 2: Creación de un nuevo proyecto web mediante la interfaz del nuevo IDE.

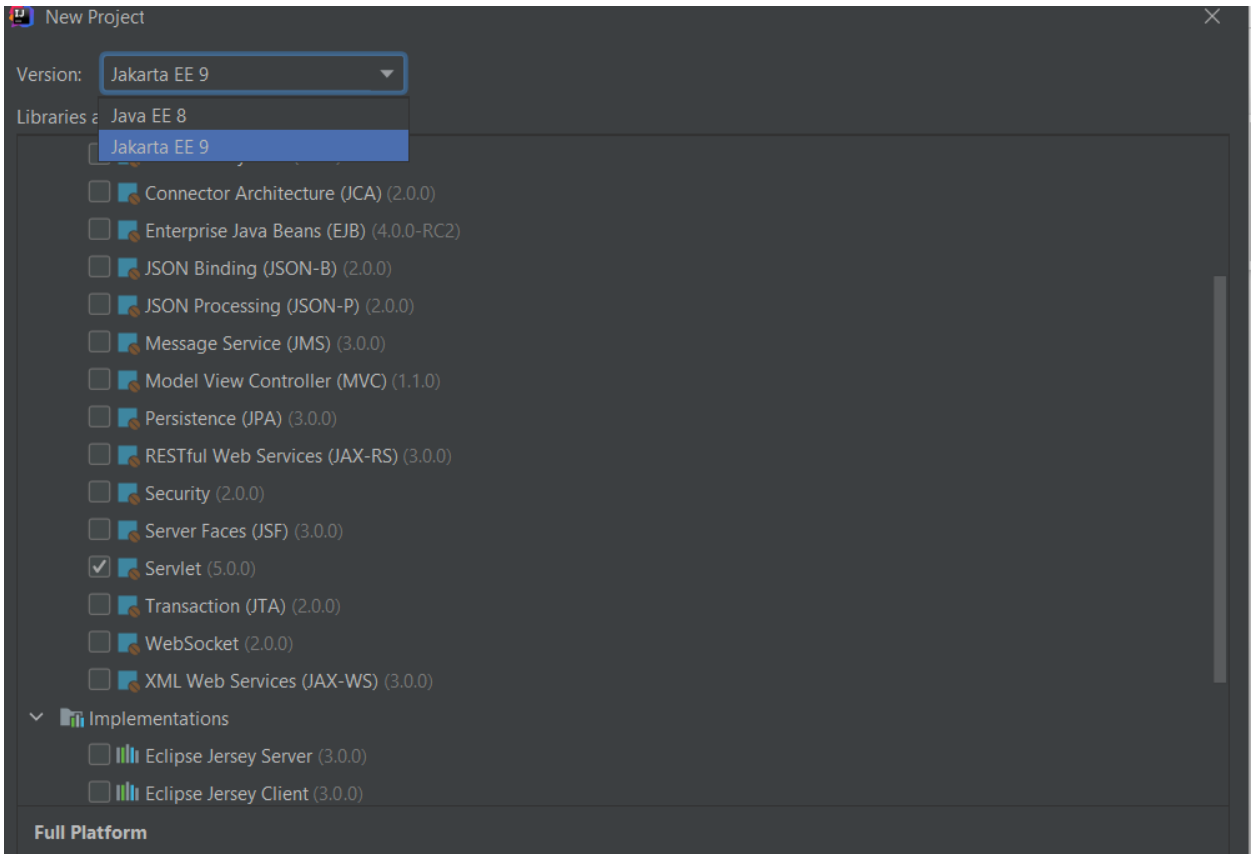
1- Dar click en **New Project**.



2- Para el siguiente paso deberá recordar ciertos conceptos importantes de la guía pasada. Deberá entonces crear una Java Enterprice para los proyectos web. También una versión de la JDK, la cual seguirá siendo la 14.0.2. En **Project Template** deberá seleccionar **Web Aplication.** Por último, dejar seleccionados los frameworks de Junit Y Maven. Finalmente dar click en **Next.**

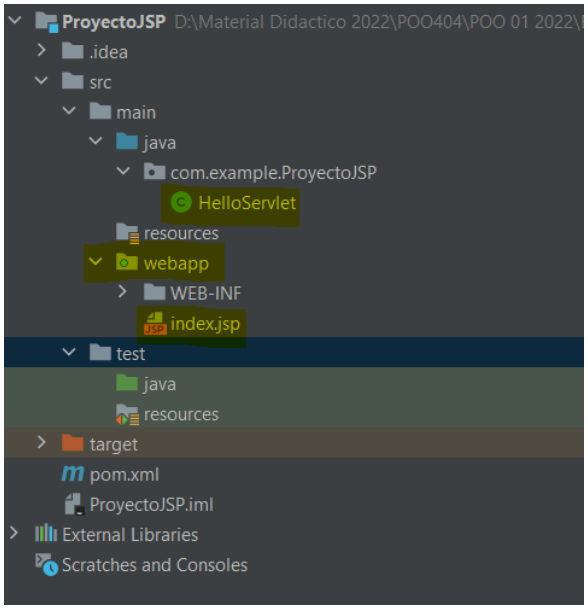


3- El siguiente paso es fundamental, así que deberá seleccionar estrictamente lo que podrá ver en la siguiente figura. Primera mente la versión a seleccionar debe ser la de **Jakarta EE 9.** Finalmente, y aunque no es obligatorio para la guía el **Servlet** como framework.

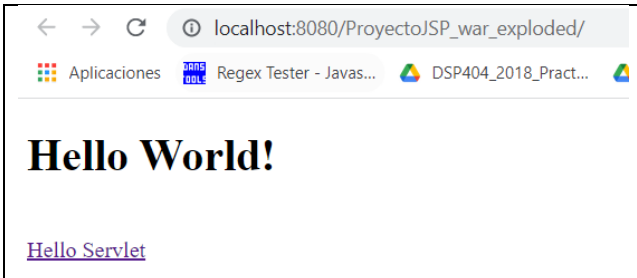


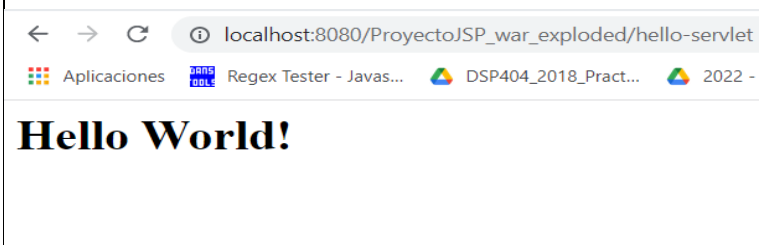
4- Ahora asignará un nombre a su proyecto, para este caso será **ProyectoJSP.** Deberá verse como en la siguiente figura. Dar click en **Finish.**

5- Su proyecto se habrá creado con éxito. Hay ciertos cambios en su estructura. A partir de este proyecto, todo nuestro contenido web estará alojado en **Web App.**



6- Lo primeo que notaremos es que por defecto se crean 2 recursos que ya conocemos. Un servlet el cual tiene por nombre **HelloServlet** y el archivo **index.jsp.** Procederemos a correr el proyecto para notar la interacción entre ambos.





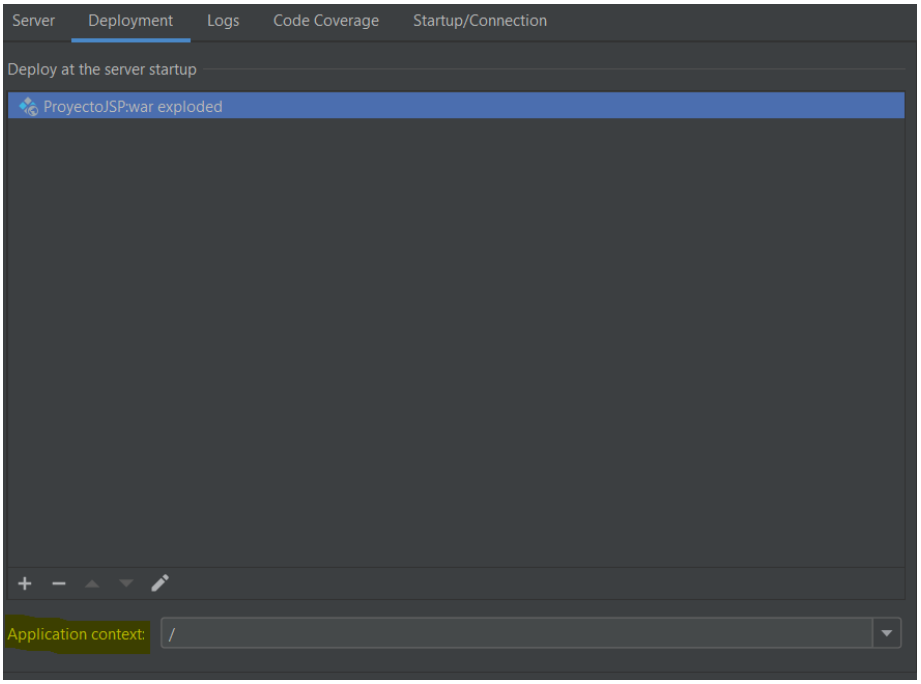

```jsp
<%@ page contentType="text/html;
charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>JSP - Hello World</title>
</head>
<body>
<h1><%= "Hello World!" %>
</h1>
<br/>
<a href="hello-servlet">Hello
Servlet</a>
</body>
</html>
```

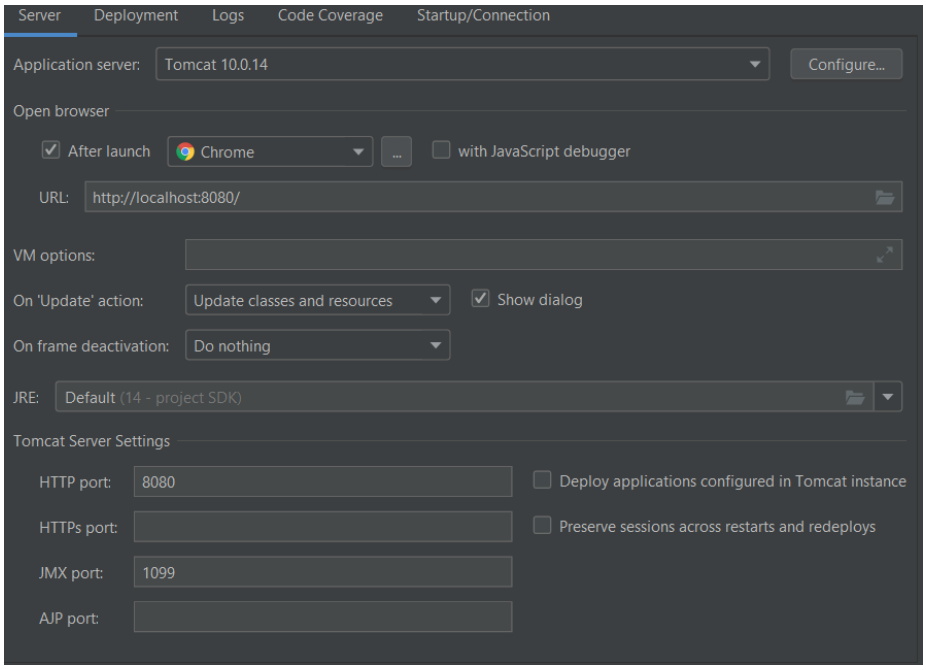
```java
@WebServlet(name = "helloServlet", value =
"/hello-servlet")
public class HelloServlet extends HttpServlet {
    private String message;

    public void init() {
        message = "Hello World!";
    }

    public void doGet(HttpServletRequest
request, HttpServletResponse response) throws
IOException {
        response.setContentType("text/html");

        // Hello
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>" + message +
"</h1>");
        out.println("</body></html>");
    }
```

Veremos entonces la interacción por defecto en un proyecto web, podrá notar que la jsp envía parámetros que posteriormente el servlet leerá, para este caso en particular será un mensaje. Nótese también el uso la scriplets y que el mediante una **expresión** pinta un mensaje en el cliente.

Paso 3: Preparación de mi proyecto web.

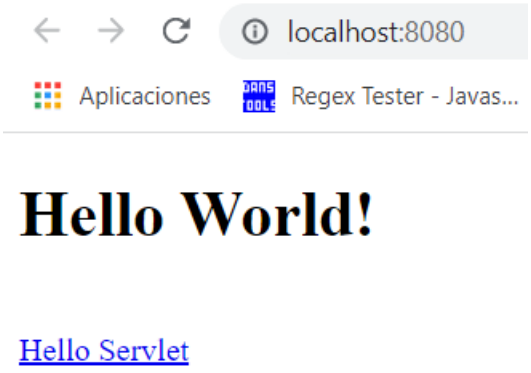
1- Por el momento configuraremos el proyecto para un navegador de preferencia y una url más cómoda. Primero cambiaremos el application context.



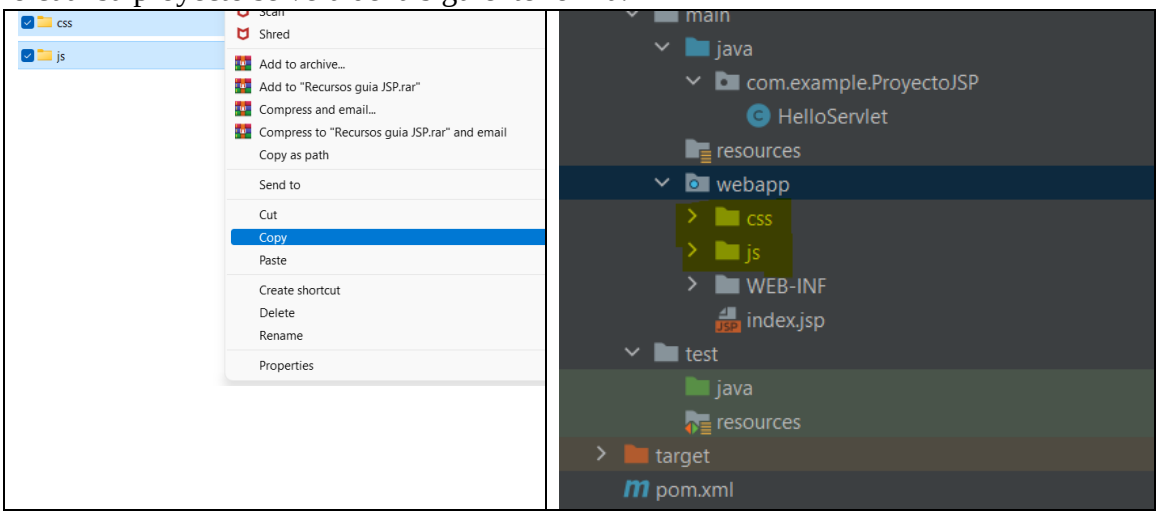


Notara que estas configuraciones ya fueron realizadas en la guía anterior y que no hay un cambio relevante en la configuración del servidor apache tomcat.

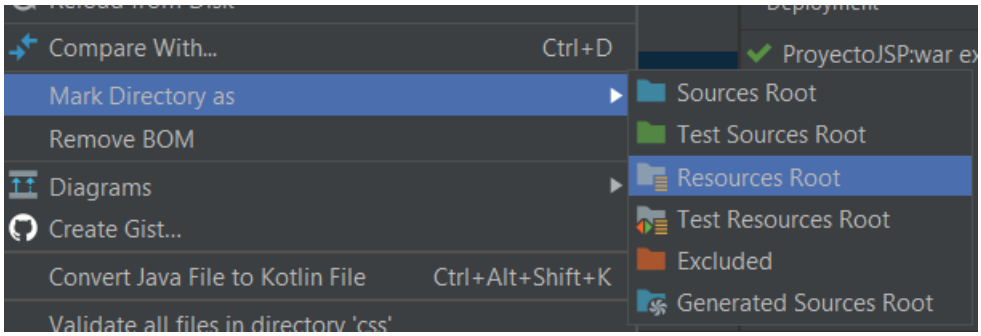
2- Aplique los cambios y ejecute el proyecto. En efecto corre en el navegador seleccionado y ya no muestra la descripción del aplication context.



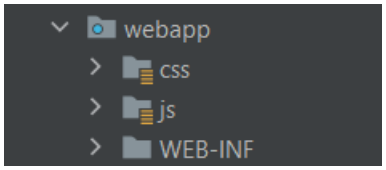
3- Colocaremos los recursos proporcionados para en el desarrollo de la guía en nuestro proyecto. Para ello copiara los archivos y los pegara dentro del directorio **Web App**, para lo cual su proyecto se vera de la siguiente forma.



4- Por último no olvide asignar estos directorios como recursos, de lo contrario no podrán ser leídos en la jsp. Dar click derecho sobre la carpeta **Mark Directory As > Resources Root.**



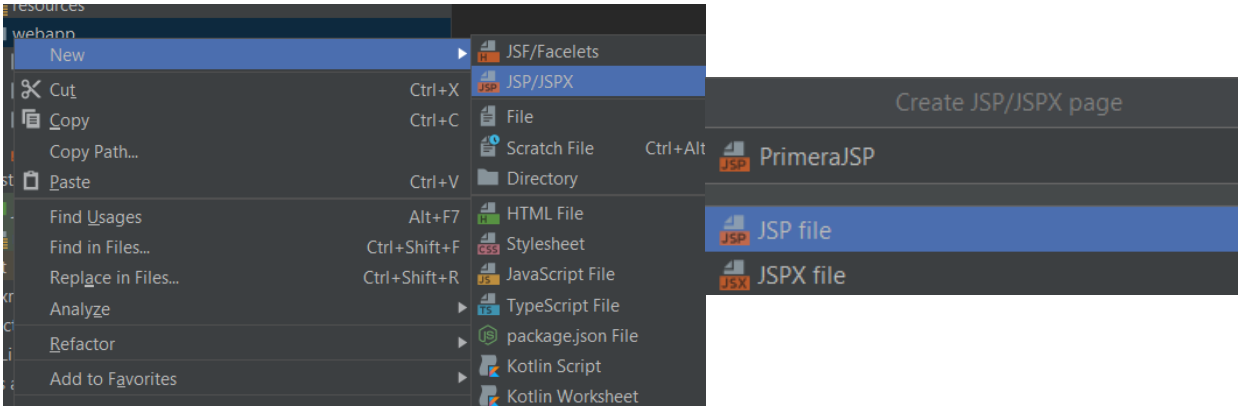
5- Deberá verse de la siguiente manera.



Creando las primer Jakarta Server Pages (JSP).

Paso 1: Crear las jsp.

1- Dar click derecho sobre la carpeta **Web App > New > JSP/JSPX**. Luego asignar un nombre el cual será **PrimeraJSP**

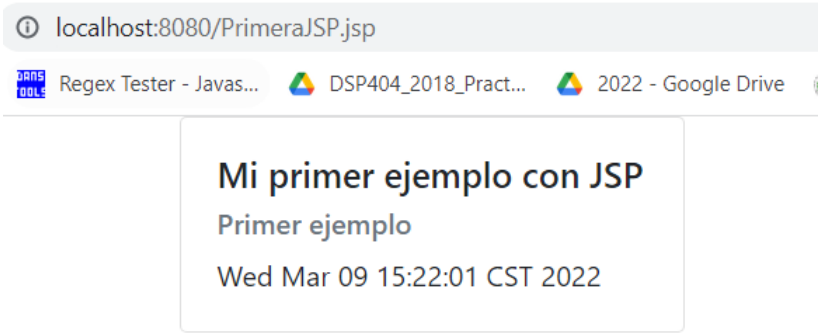


2- Digitar el siguiente código.

```
<%@page import="java.util.*"%>
<%@ page import="java.io.PrintWriter" %>
<%! String titulo = "Mi primer ejemplo con JSP";
   String cadena = "Primer ejemplo";
%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
   <title><%=titulo%></title>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
<body>
```

```
<div class="container">
<div class="card" style="width: 18rem;">
   <div class="card-body">
      <h5 class="card-title"><%=titulo%></h5>
      <h6 class="card-subtitle mb-2 text-muted"><%=cadena%></h6>
      <p class="card-text"><%

         out.println(new Date());
      %></p>

   </div>
</div>
</div>
</body>
</html>
```

3- Correr la aplicación para ver el resultado.

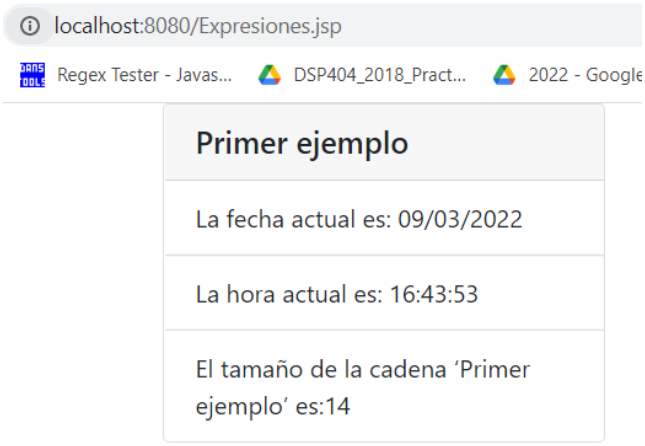


4- Ahora modificaremos un poco el código para poder utilizar Elementos de JSP, como son las expresiones, scriptlets y declaraciones. Para ello crearemos la página **Expresiones.jsp** el código de su página debe quedar de la siguiente manera.

```
<%@page import="java.text.SimpleDateFormat" %>
<%@page import="java.text.DateFormat" %>
<%@page import="java.util.*" %>
<%!
   String titulo = "Mi primer ejemplo con JSP";
   String cadena = "Primer ejemplo";
%>
<%@page contentType="text/html" pageEncoding="UTF-8" %>
<!DOCTYPEhtml>
<html>
<head>
   <title><%=titulo%>
   </title>
   <meta charset="utf-8">
```

```
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="css/bootstrap.min.css">
    </head>
    <body>
    <%
        //Obteniendo la fecha actual
        Date fechaActual = new Date();
    //Formateando la fecha
        DateFormat formatoHora = new SimpleDateFormat("HH:mm:ss");
        DateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
    %>


    <div class="container">
        <div class="card" style="width: 18rem;">
            <h5 class="card-header"><%=cadena%>
            </h5>
            <ul class="list-group list-group-flush">
                <li class="list-group-item"><%="La fecha actual es: " +
    formatoFecha.format(fechaActual)%>
                </li>
                <li class="list-group-item"><%="La hora actual es: " +
    formatoHora.format(fechaActual)%>
                </li>
                <li class="list-group-item"><%=" El tamaño de la cadena ‘" + cadena +
    "’ es:" + cadena.length()%>
                </li>
            </ul>

        </div>
    </div>
    </body>
</html>
```

5- Volver a correr para ver el resultado.



6- **Scriptlets en JSP.** Para esta parte crear una nueva página llamada **"forJSP"**, y luego digitar el siguiente código.

```
<%@page contentType="text/html" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
   <title>For JSP</title>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
<body>

<div class="container">
   <div class="row">
      <h3>Tablas de multiplicar</h3>
   </div>
   <%
      for (int i = 1; i <= 10; i++) {
   %>
   <div class="card">
      <div class="card-header"><%="Tabla del " + i%>
      </div>
      <ul class="list-group list-group-flush">
         <%
            for (int j = 1; j <= 10; j++) {
               out.println("<li class='list-group-item'>" + i + " x " + j + " = " + (i * j) + "</li>");
            }//Cerrando el for más interno
         %>
      </ul>
   </div>
```

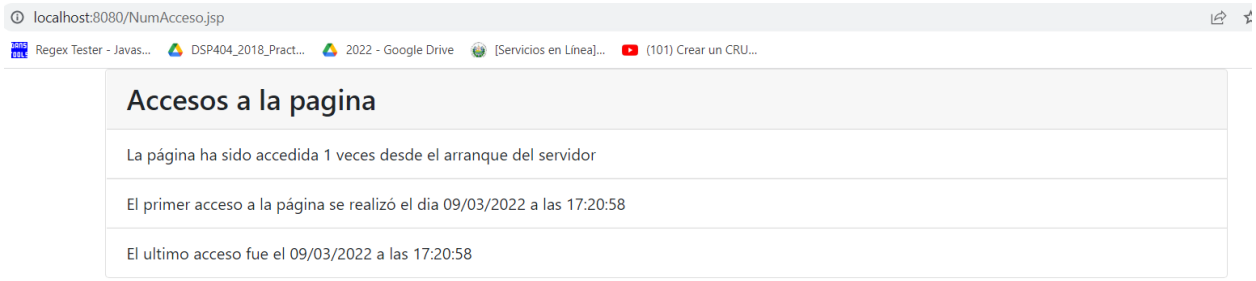
```
   <% }//Cerrando el for más externo
   %>
</div>
</body>
</html>
```

7- **Declaraciones en JSP.** Crear una nueva página llamada **"NumAcceso"** y luego digitar el siguiente código.

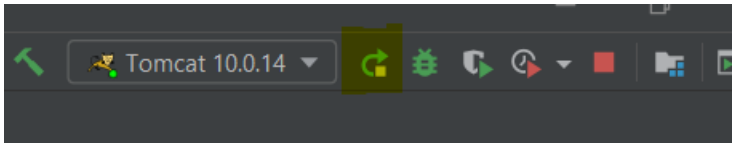
```
<%@page import="java.util.Date" %>
<%@page import="java.text.SimpleDateFormat" %>
<%@page import="java.text.DateFormat" %>
<%@page contentType="text/html" pageEncoding="UTF-8" %>

<%!
   int numeroAccesos = 0;
   DateFormat formatoHora = new SimpleDateFormat("HH:mm:ss");
   DateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
   java.util.Date primerAcceso = new java.util.Date();

   private Date ahora() {
      Date now = new Date();
      return now;
   }
%>
<!DOCTYPE html>
<html>
<head>
   <title>Ejemplo</title>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
<body>
<div class="container">
   <div class="card">
      <h3 class="card-header">Accesos a la pagina</h3>
      <ul class="list-group list-group-flush">
         <li class="list-group-item">
            <%="La página ha sido accedida " + (++numeroAccesos)
                  + " veces desde el arranque del servidor"%>
         </li>
         <li class="list-group-item">
            <%="El primer acceso a la página se realizó el dia "
                  + formatoFecha.format(primerAcceso) + " a las "
```

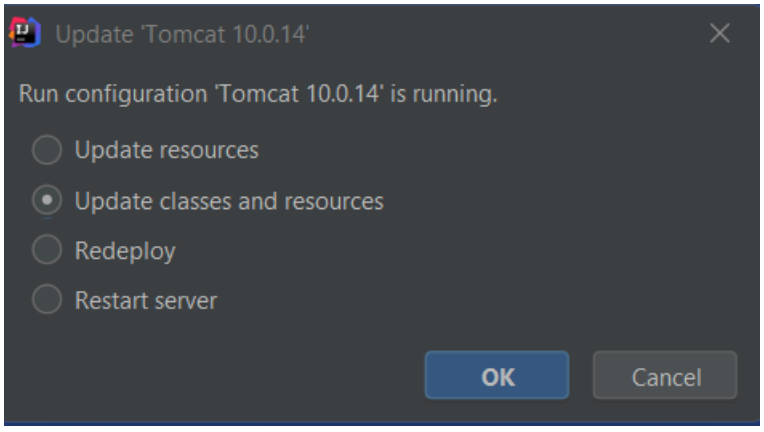
```
                + formatoHora.format(primerAcceso)%>
        </li>
        <li class="list-group-item">
           <%="El ultimo acceso fue el " + formatoFecha.format(ahora())
                + " a las " + formatoHora.format(ahora())%>
        </li>
      </ul>
   </div>
</div>
</body>
</html>
```

8- El resultado será.



Tip: Para poder ver los cambios que haga en las jsp o en archivos del cliente, deberá dar click en el icono de correr y vera las siguientes opciones.





Update resources: Actualiza recursos estáticos agregados al proyecto, archivos css, js y html.

Update clases and resources: Actualiza recursos estáticos y cambios a las clases o servlets del proyecto.

Redeploy y Restart Server: Limpian todo el cache de la web, y vuelven a construir dependencias y archivos necesarios para deploguear el proyecto.

JSP y JDBC

Ahora crearemos una aplicación en la cual nos conectaremos a una base de datos, para ello seguir los siguientes pasos.

Paso 1: Crear la siguiente base de datos.

```
CREATE DATABASE personabddg8;
USE personabddg8;

CREATE TABLE ocupaciones(
id_ocupacion int(11) NOT NULL AUTO_INCREMENT,
ocupacion varchar(50) NOT NULL,
PRIMARY KEY (`id_ocupacion`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT
CHARSET=latin1;

CREATE TABLE persona (
id_persona int(11) NOT NULL AUTO_INCREMENT,
nombre_persona varchar(100) NOT NULL,
edad_persona int(11) NOT NULL,
sexo_persona varchar(50) NOT NULL,
id_ocupacion int(11) NOT NULL,
fecha_nac date NOT NULL,
usuario varchar(50) NOT NULL,
contrasenia varchar(64) NOT NULL,
PRIMARY KEY (`id_persona`),
KEY id_ocupacion (id_ocupacion)
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT
CHARSET=latin1;

ALTER TABLE persona
ADD CONSTRAINT persona_ibfk_1
FOREIGN KEY (id_ocupacion)
REFERENCES ocupaciones (id_ocupacion);

INSERT INTO `ocupaciones` (`id_ocupacion`, `ocupacion`)
VALUES(1, 'Doctor'), (2, 'Emprendedor'), (3, 'Profesor');

INSERT INTO persona (id_persona, nombre_persona, edad_persona,
sexo_persona, id_ocupacion, fecha_nac, usuario, contrasenia) VALUES
(1, 'Alejandro Pineda', 22, 'Masculino', 1, '1999-01-05', 'usuario1',SHA2('usuario1',256)),
(2, 'Fernando Calderón', 19, 'Masculino', 2, '2001-05-07','usuario2',SHA2('usuario2',256)),
```

```
(3, 'Emerson Torres', 22, 'Masculino', 3, '1999-08-03','usuario3',SHA2('usuario3',256));
```

1- Vera que ahora estamos encriptando la contraseña, para mayor seguridad del sitio web.

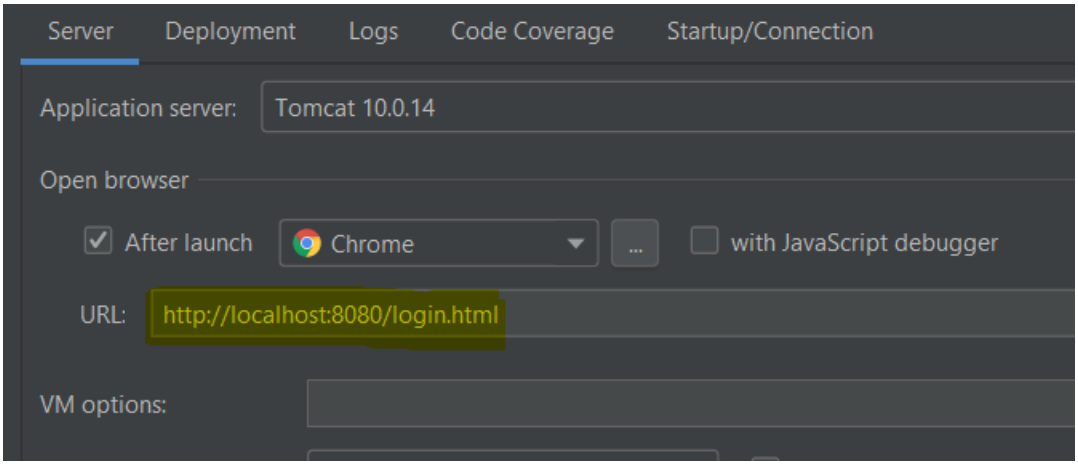
Paso 2: Creación del sitio.

1- Crear la siguiente página html llamada **login.html** y copiar el siguiente código.

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>login</title>
   <link rel="stylesheet" href="css/bootstrap.min.css">
   <link rel="stylesheet" href="css/estilo.css">
   <script src="js/bootstrap.min.js" ></script>
</head>

<body class="container" >

<div id="login">
   <h1 class="text-center text-white pt-5">Universidad Don Bosco</h1>
   <div class="container">
      <div id="login-row" class="row justify-content-center align-items-center">
         <div id="login-column" class="col-md-6">
            <div id="login-box" class="col-md-12">
               <form id="login-form" class="form" action="controller.jsp" method="post">
                  <h3 class="text-center text-info">Login</h3>
                  <div class="form-group">
                     <label for="username" class="text-info">Usuario:</label><br>
                     <input type="text" name="usuario" id="username" class="form-control">
                  </div>
                  <div class="form-group">
                     <label for="password" class="text-info">Password:</label><br>
                     <input type="password" name="password" id="password" class="form-
control">
                  </div>
                  <div class="form-group">
                     <input type="submit" name="operacion" class="btn btn-info btn-md"
value="logueo">
                     <input type="hidden" name="operacion"  value="logueo">
                  </div>
                  <br>

               </form>

            </div>
```

```
        </div>
      </div>
   </div>
</div>

</body>

</html>
```

2- Configuraremos el servidor tomcat para que esta sea la pagina a correr por defecto.



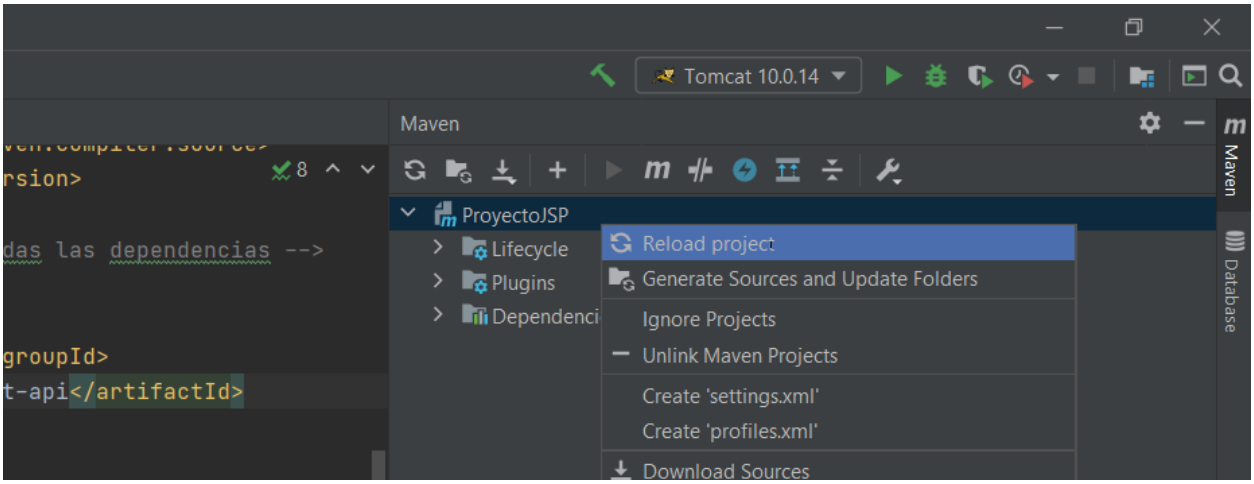
3- Importaremos las dependencias necesarias para poder conectarnos al servicio de MySQL. Ahora lo haremos desde el archivo de configuración Pom.xml. Pegar las siguientes líneas de código como se muestra en la figura.

```
        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <dependency>
          <groupId>mysql</groupId>
           <artifactId>mysql-connector-java</artifactId>
           <version>8.0.27</version>
        </dependency>
```

```
    <junit.version>5.7.0</junit.version>
</properties>
<!-- Inicio de importacion para todas las dependencias -->
<dependencies>
    <dependency>
        <groupId>jakarta.servlet</groupId>
        <artifactId>jakarta.servlet-api</artifactId>
        <version>5.0.0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>${junit.version}</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>${junit.version}</version>
        <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.27</version>
    </dependency>
</dependencies>
<!-- Fin de importacion para todas las dependencias -->
<build>
```

project

4- Notara que estas marcan un error, para lo cual deberá actualizar el proyecto y esperar que importe de forma automática todas las dependencias de mysqlconnector. Para ello bastara con que de click derecho sobre el nombre del proyecto y seleccione **Reload Project.**



5- Ahora creara el archivo **conexion.jsp** y pegara el siguiente código.

```
        <%@ page import="java.sql.*" %>
        <%@ page contentType="text/html;charset=UTF-8" language="java" %>

        <%
            Connection conexion =null;
           //private Statement s =null;
            ResultSet rs=null;
            PreparedStatement st =null;
           Class.forName("com.mysql.jdbc.Driver");
        // Se obtiene una conexión con la base de datos.
           conexion =
        DriverManager.getConnection("jdbc:mysql://localhost/personabddg8", "root",
        "");
           String valor;
%>
```

6- Notara que estamos seccionando nuestro proyecto, por lo cual ahora crearemos el recurso **menú.jsp**.

```
<%@ page import="jakarta.servlet.http.HttpSession" %>
<%

   HttpSession session_actual = request.getSession(false);
   String usuario = (String) session_actual.getAttribute("USER");
%>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top" role="navigation">
   <div class="container">
      <a class="navbar-brand" href="#">Guia JSP POO</a>
      <button class="navbar-toggler border-0" type="button" data-toggle="collapse" data-
target="#exCollapsingNavbar">
         &#9776;
      </button>
      <div class="collapse navbar-collapse" id="exCollapsingNavbar">

         <ul class="nav navbar-nav flex-row justify-content-between ml-auto">
            <li class="dropdown order-1">
               <button type="button" id="dropdownMenu1" data-toggle="dropdown"
                     class="btn btn-outline-secondary dropdown-toggle"><i class="fa fa-
user"></i> <%=usuario%>
                  <span class="caret"></span></button>
               <ul class="dropdown-menu dropdown-menu-right mt-2">
                  <a class="dropdown-item" href="controller.jsp?operacion=salir"><i
class="fa fa-sign-out"></i> Logout</a>
                  </li>
```

```
                </ul>
            </li>
          </ul>
        </div>
    </div>
</nav>
```

7- Haremos lo mismo con **consulta.jsp.**

```
<%@ include file="conexion.jsp" %>
<div class="">
   <table class="table table-striped table-bordered table-hover">
      <thead>
      <tr>
         <th>Id</th>
         <th>Nombres</th>
         <th>Edad</th>
         <th>Sexo</th>
         <th>Ocupacion</th>
         <th>Fecha de nacimiento</th>
         <th>Usuario</th>
         <th>Eliminar</th>
         <th>Modificar</th>
      </tr>
      </thead>
      <tbody>
      <%
         st = conexion.prepareStatement("SELECT
p.id_persona,p.nombre_persona,p.edad_persona,p.sexo_persona,p.id_ocupacion,p.fecha_n
ac, p.usuario, o.ocupacion FROM persona p INNER JOIN ocupaciones o on
p.id_ocupacion= o.id_ocupacion");
         rs = st.executeQuery();
         while (rs.next()) {
      %>
      <tr>
         <td><%=rs.getString("id_persona")%>
         </td>
         <td><%=rs.getString("nombre_persona")%>
         </td>
         <td><%=rs.getString("edad_persona")%>
         </td>
         <td><%=rs.getString("sexo_persona")%>
         </td>
         <td><%=rs.getString("ocupacion")%>
         </td>
         <td><%=rs.getString("fecha_nac")%>
```

```
            </td>
            <td><%=rs.getString("usuario")%>
            </td>
            <td><button class="btn btn-danger"
onclick="alerta('<%=rs.getString("id_persona")%>')">Eliminar</button></td>
            <td><button class="btn btn-info"
onclick="modificar('<%=rs.getString("id_persona")%>','<%=rs.getString("nombre_person
a")%>','<%=rs.getString("edad_persona")%>','<%=rs.getString("sexo_persona")%>','<%=
rs.getString("id_ocupacion")%>','<%=rs.getString("fecha_nac")%>','<%=rs.getString("usu
ario")%>')" >Modificar</button></td>

        </tr>
        <%
            }
            conexion.close();
        %>
        </tbody>
    </table>
</div>
<script>

    function alerta(id)
    {
        var mensaje;
        var opcion = confirm("Esta seguro de eliminar este registro");
        if (opcion == true) {
            location.href ="controller.jsp?operacion=eliminar&id="+id;
        }

    }

    function modificar(id, nombre,edad,sexo,ocupacion,fecha,usuario)
    {
        document.getElementById("idpersona").value=id;
        document.getElementById("nombre").value=nombre;
        document.getElementById("edad").value=edad;
        document.getElementById("sexo").value=sexo;
        document.getElementById("ocupacion").value=ocupacion;
        document.getElementById("fecha").value=fecha;
        document.getElementById("usuario").value=usuario;
        document.getElementById("operacion").value="modificar";
        hijo = document.getElementById("passhidden");
        padre = hijo.parentNode;
        padre.removeChild(hijo);

    }
```

```
</script>
```

8- Ahora crearemos la página principal y de bienvenida al usuario, esta página se llamará **principal.jsp.**

```
<%@ page import="jakarta.servlet.http.HttpSession" %>
<%@ include file="conexion.jsp" %>
<%


    HttpSession session_actual = request.getSession(false);
    String usuario = (String) session_actual.getAttribute("USER");
    String nombres = (String) session_actual.getAttribute("NAME");

    if (usuario == null) {
        response.sendRedirect("login.html");
    }
%>
<html lang="es">
<head>

    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bienvenido <%=nombres%>
    </title>
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
    <script src="js/jquery-3.2.1.slim.min.js"></script>
    <script src="js/bootstrap1.min.js"></script>


</head>

<body>
<jsp:include page="menu.jsp"/>

<h1 class="text-center">Bienvenido a POO404</h1>
<h2 class="text-center">
    Bienvenido: (<%=usuario%>) <%=nombres%>
</h2>
<div class="container">

    <div style="padding: 0;">
        <form role="form" action="controller.jsp" method="POST">
            <div class="col-md-12" id="conten">
```

```
            <input type="hidden"  name="id" id="idpersona">
            <input type="hidden" value="insertar" name="operacion" id="operacion">
            <div class="form-group">
              <label for="nombre">Ingrese el nombre de la persona:</label>
              <div class="input-group">
                <input type="text" class="form-control" name="nombre" id="nombre"
                     placeholder="Ingresa el nombre" required>

              </div>
            </div>
            <div class="form-group">
              <label for="edad">Ingrese la edad de la persona:</label>
              <div class="input-group">
                <input type="number" class="form-control" id="edad" name="edad"
                     placeholder="Ingresa la edad" required>
              </div>
            </div>

            <div class="form-group">
              <label for="sexo">Ingrese el sexo de la persona:</label>
              <div class="input-group">
                <select name="sexo" id="sexo" class="form-control" required>
                  <option value="Masculino">Maculino</option>
                  <option value="Femenino">Femenino</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <label for="ocupacion">Ingrese la ocupacion de la persona:</label>
              <div class="input-group">
                <select name="ocupacion" id="ocupacion" class="form-control" required>
                  <%
                     st = conexion.prepareStatement("SELECT * FROM ocupaciones");
                     rs = st.executeQuery();
                     while (rs.next()) {
                  %>
                  <option
value="<%=rs.getString("id_ocupacion")%>"><%=rs.getString("ocupacion")%>
                  </option>
                  <%}%>
                </select>
              </div>
            </div>
            <div class="form-group">
              <label for="fecha">Ingrese la fecha de nacimiento de la persona:</label>
              <div class="input-group">
```

```
                    <input type="date" class="form-control" id="fecha" name="fecha"
                        placeholder="Ingresa la fecha" required>
                </div>
            </div>
            <div class="form-group">
                <label for="fecha">Ingrese el usuario de la persona:</label>
                <div class="input-group">
                    <input type="text" class="form-control" id="usuario" name="usuario"
                        placeholder="Ingresa el usuario" required>
                </div>
            </div>
            <div class="form-group" id="passhidden">
                <label for="fecha">Ingrese el password de la persona:</label>
                <div class="input-group">
                    <input type="password" class="form-control" id="password"
name="password"
                        placeholder="Ingresa el password" required>
                </div>
            </div>
            <div style="margin-left: 30%;">
                <input type="submit" class="btn btn-success  col-md-6 " value="Guardar">
            </div>
        </div>
    </form>
    <%
        if(request.getParameter("exito") != null){
            out.println("<div class='alert alert-success' role='alert'>Operacion realizada con
exito</div>");
        }
    %>

  </div>
  <!--notese el uso de jsp:include -->
  <jsp:include page="consulta.jsp"/>



</div>
</body>
</html>
```

9- Finalmente crearemos el archivo que gestionara todas las peticiones y operaciones realizadas en nuestro proyecto. Este archivo se llamara **controller.jsp**

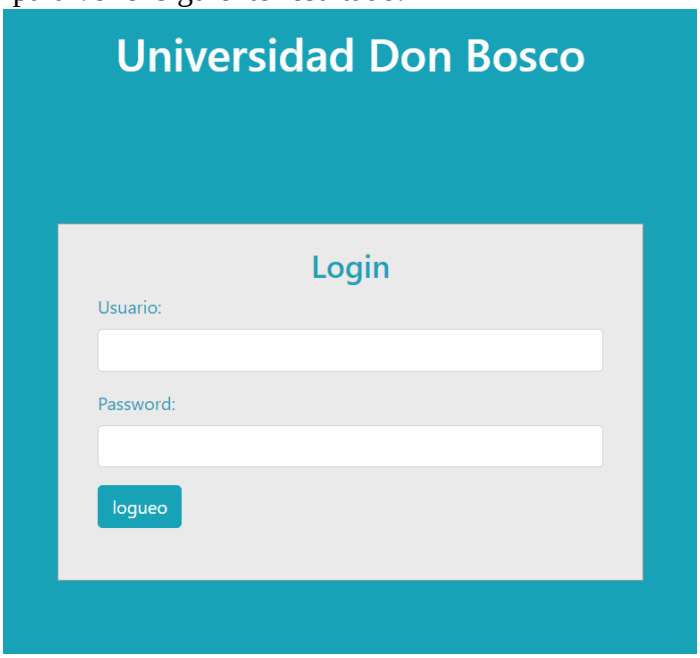
```
<%@ page import="jakarta.servlet.http.HttpSession" %>
<%@ page import="jakarta.servlet.http.HttpSession" %>
```

```
<%@ include file="conexion.jsp" %>
<html>
<head>
   <title>Title</title>
</head>
<body>
<h1>Generando sesion</h1>
<%
   HttpSession session_actual = request.getSession(true);
   String operacion = request.getParameter("operacion");

   out.println(operacion);
   if (operacion.equals("salir")) {
      session_actual.setAttribute("USER", null);
      session_actual.setAttribute("NAME", null);
      response.sendRedirect("login.html");
   } else if (operacion.equals("logueo")) {
      String usuario = request.getParameter("usuario");
      String password = request.getParameter("password");
      try {
         st = conexion.prepareStatement("select count(usuario),nombre_persona from
persona where usuario= ? and contrasenia=SHA2(?,256) ");
         st.setString(1, usuario);
         st.setString(2, password);
         rs = st.executeQuery();

         rs.next();
         if (rs.getInt(1) == 1) { //solo una coincidencia es permitida

            session_actual.setAttribute("USER", usuario);
            session_actual.setAttribute("NAME", rs.getString(2));
            response.sendRedirect("principal.jsp");
         } else {
            response.sendRedirect("login.html");
         }

         rs.close();
         conexion.close();


      } catch (SQLException throwables) {
         throwables.printStackTrace();
      }
   } else if (operacion.equals("insertar")) {
      String nombre=request.getParameter("nombre");
      int edad = Integer.parseInt(request.getParameter("edad"));
```

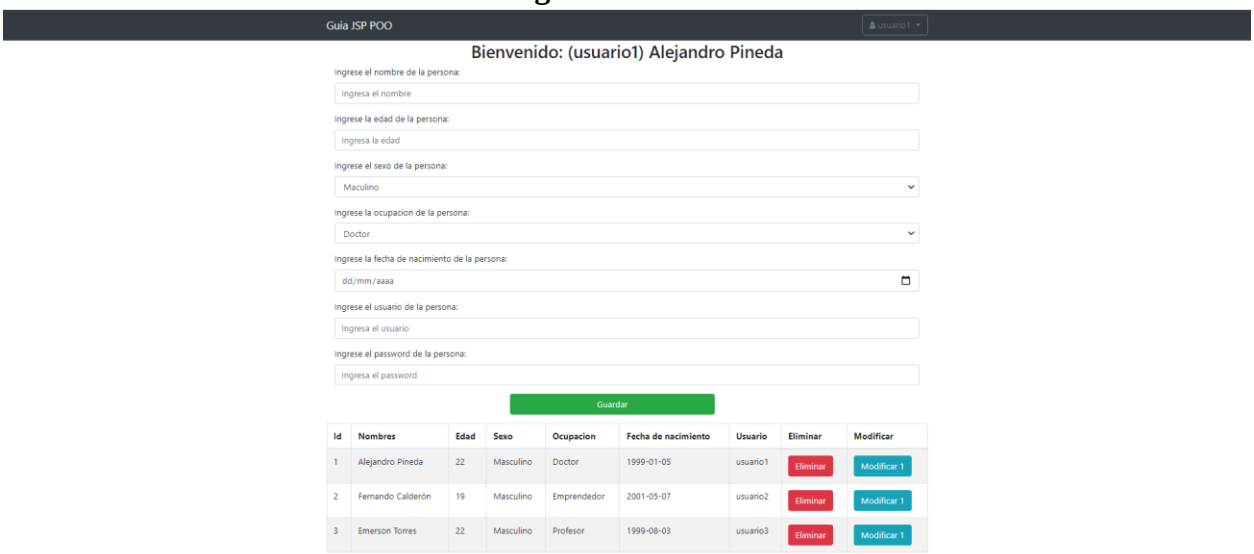
```
        String sexo=request.getParameter("sexo");
        String fecha=request.getParameter("fecha");
        int idocupacion = Integer.parseInt(request.getParameter("ocupacion"));
        String usuario=request.getParameter("usuario");
        String password=request.getParameter("password");
        st = conexion.prepareStatement("INSERT INTO persona(nombre_persona,
edad_persona, sexo_persona, id_ocupacion, fecha_nac, usuario, contrasenia) VALUES (?,
? , ?, ? ,?, ?, SHA2(?,256))" );
        st.setString(1,nombre);
        st.setInt(2, edad);
        st.setString(3, sexo);
        st.setInt(4,idocupacion);
        st.setString(5, fecha);
        st.setString(6, usuario);
        st.setString(7, password);
        st.executeUpdate();
        response.sendRedirect("principal.jsp?exito=si");
    }else if (operacion.equals("modificar")) {
        int id = Integer.parseInt(request.getParameter("id"));
        String nombre=request.getParameter("nombre");
        int edad = Integer.parseInt(request.getParameter("edad"));
        String sexo=request.getParameter("sexo");
        String fecha=request.getParameter("fecha");
        int idocupacion = Integer.parseInt(request.getParameter("ocupacion"));
        String usuario=request.getParameter("usuario");
        String password=request.getParameter("password");
        st = conexion.prepareStatement("UPDATE persona SET
nombre_persona=?,edad_persona=?,sexo_persona=?,id_ocupacion=?,fecha_nac=?,usuario
=? WHERE id_persona=?");
        st.setString(1,nombre);
        st.setInt(2, edad);
        st.setString(3, sexo);
        st.setInt(4,idocupacion);
        st.setString(5, fecha);
        st.setString(6, usuario);
        st.setInt(7, id);
        st.executeUpdate();
        response.sendRedirect("principal.jsp?exito=si");
    }else if (operacion.equals("eliminar")) {
        int id = Integer.parseInt(request.getParameter("id"));

        st = conexion.prepareStatement("DELETE FROM persona WHERE id_persona=?");
        st.setInt(1, id);
        st.executeUpdate();
        response.sendRedirect("principal.jsp?exito=si");
    }
```

```
%>
</body>
</html>
```

10- Corra su Proyecto para ver el siguiente resultado.



login.html



principal.jsp

En conclusión:

- A partir de la versión 2020.2 es que se predefine el uso de **Jakarta Server Pages** para el entorno de desarrollo IntelliJ IDEA, pero su última versión estable se encuentra dentro de la versión 2020.3 en adelante. Es necesario aclarar que, si podíamos crear una JSP en la versión 2020.1.4, pero estas implementaban una tecnología desfasada, las cuales eran las **Java Server Pages**, es por esta razón que en la guía pasada usted tenia el error en la línea request.getParameter(), esto pasaba porque ella siempre buscaba el directorio javax, pero este ya no existía.
- Notara que en la guía se hace la importación de las dependencias mediante el archivo Pom.xml, lo cual implica un mayor beneficio para nosotros, dado que no debemos buscar dichas dependencias por nosotros mismos; bastara entonces con ir al sitio https://mvnrepository.com/, buscar su dependencia y colocarla en el archivo de configuración. Vera que estas son importadas de forma automática.

IV. EJERCICIOS **COMPLEMENTARIOS**

1. **Tomando de base el proyecto del segundo periodo realizar el mantenimiento de la tabla usuarios.**
2. **Implementar el código de la guía pasada para calcular la edad a partir de la fecha de nacimiento.**