



Universidad de Alcalá

Escuela Politécnica Superior

Universidad de Alcalá

PECL5

Arquitectura y Diseño de Sistemas Web y C/S

Servlets, JSP y bases de datos

Laboratorio Jueves 10:00 – 12:00

Grado en Ingeniería Informática & Ingeniería en Sistemas de
Información – Curso 2019/2020

Marcos Barranquero Fernández – 51129104N

Daniel Manzano Estébanez – 03220212M

CONSIDERACIONES PREVIAS

Los archivos anexos son proyectos de NetBeans que se pueden importar desde el propio programa.

Las bases de datos contenidas en la carpeta *BBDD – Derby* deben copiarse en la siguiente ruta para su correcto funcionamiento, y posteriormente conectarse a ellas en la pestaña Servicios – Databases – Java DB, haciendo click derecho sobre sus nombres. Se puede hacer una copia de seguridad de las que ya hubiera para que no se pierdan.

```
C:\Users\NOMBRE_USUARIO\AppData\Roaming\NetBeans\Derby
```

APARTADO 1 – SERVLETS

EJERCICIO 1

El ejercicio 1 consiste en ejecutar satisfactoriamente el Servlet. Se han añadido comentarios al código para explicar su funcionamiento:

```
package Servlets;

//Primer Servlet.
//Muy sencillo.
import java.io.*;
import static javafx.application.ConditionalFeature.WEB;
import javax.servlet.*;
import javax.servlet.http.*;

public class PrimerServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        // Establece tipo de contenido
        res.setContentType("text/html");

        // Instancia impresora para dar formato a la página
        PrintWriter out = new PrintWriter(res.getOutputStream());

        // Se crea una pequeña página HTML
        out.println("<html>");
        out.println("<head><title>HolaMundoServlet</title></head>");
        out.println("<body>");
        out.println("<h1><center>Hola Mundo desde el servidor WEB</center></h1>");
        out.println("</body></html>");
        out.close();
    }

    public String getServletInfo() {
        return "Crea una página HTML que dice HolaMundo";
    }
}
```

EJERCICIO 2

El ejercicio 2 propone enlazar un Servlet con un formulario, de forma que el servidor contenga funcionalidad e interactúe en función de este.

Es importante marcar la casilla que permite añadir la información del descriptor del Servlet al archivo xml:

Steps

1. Choose File Type
2. Name and Location
3. **Configure Servlet Deployment**

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☒ Add information to deployment descriptor (web.xml)

Class Name:

Servlet Name:

URL Pattern(s):

Initialization Parameters:

Name	Value
------	-------

New Edit... Delete

< Back Next > Finish Cancel Help

El código de la web html es el siguiente:

```
<html>

<head>
  <meta charset="UTF-8">
  <title>Segundo Servlet</title>
</head>

<body>
  <center>
    <h2>Segundo Servlet</h2>
  </center>

  <!-- Formulario asociado al Servlet llamado SegundoServlet -->
  <form action="SegundoServlet" method=POST>
    <center>
      Introduzca su nombre y pulse el botón de enviar
      <br>
      <br>
      <input type="text" name=NOMBRE>
      <br>
      <br>
      <input type="submit" value="Enviar Nombre">
      <input type="reset" value="Borrar">
    </center>
  </form>
</body>

</html>
```

Y el código del servlet asociado es el siguiente:

```
package Servlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

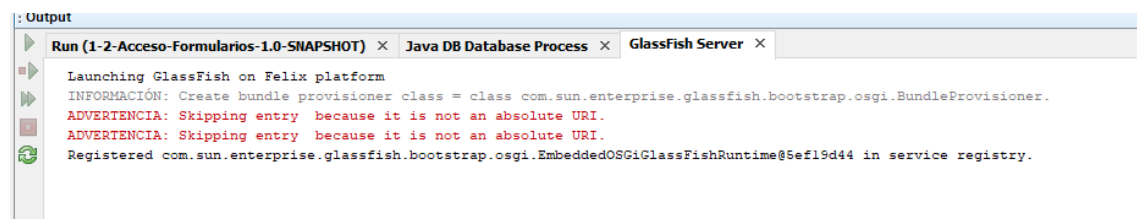
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SegundoServlet extends HttpServlet {
    String nombre;

    public void service(HttpServletRequest petition, HttpServletResponse respuesta)
        throws ServletException, IOException {
        // Recogemos parámetro nombre del formulario y lo guardamos
        // en una string
        nombre = petition.getParameter("NOMBRE");

        // Escribimos html mostrando el nombre:
        ServletOutputStream out = respuesta.getOutputStream();
        out.println("<html>");
        out.println("<head><title>HolaTalServlet</title></head>");
        out.println("<body>");
        out.println("<p><h1><center>Su nombre es: <B>" + nombre + "
</B></center></h1></p>");
        out.println("</body></html>");
        out.close();
    }
}
```

Si ejecutamos el proyecto, vemos que comienza a ejecutarse el servidor:



The screenshot shows the Eclipse IDE's Output window with three tabs: 'Run (1-2-Acceso-Formularios-1.0-SNAPSHOT)', 'Java DB Database Process', and 'GlassFish Server'. The 'GlassFish Server' tab is active, displaying the following log messages:

```
Launching GlassFish on Felix platform
INFORMACIÓN: Create bundle provisioner class = class com.sun.enterprise.glassfish.bootstrap.osgi.BundleProvisioner.
ADVERTENCIA: Skipping entry because it is not an absolute URI.
ADVERTENCIA: Skipping entry because it is not an absolute URI.
Registered com.sun.enterprise.glassfish.bootstrap.osgi.EmbeddedOSGiGlassFishRuntime@5ef19d44 in service registry.
```

Y tras unos instantes lanzándose todo, nos abre una nueva ventana en el navegador mostrando el index:

Enviar Nombre Borrar

← → ↺ ⓘ localhost:8080/1-2-Acceso-Formularios/SegundoServlet

```
<form action="CalculadoraServlet" method=POST>  
    <table border="3" WIDTH="400">  
        <tr>  
            <td>  
                <input name="operando1" type="text" style="text-align:right;">  
            </td>  
            <td>  
                <select name="operacion">  
                    <option value="1" selected>&nbsp;&nbsp;&nbsp;+ &nbsp;&nbsp;&nbsp;</option>  
                    <option value="2">&nbsp;&nbsp;&nbsp;- &nbsp;&nbsp;&nbsp;</option>  
                    <option value="3">&nbsp;&nbsp;&nbsp;* &nbsp;&nbsp;&nbsp;</option>  
                    <option value="4">&nbsp;&nbsp;&nbsp;/ &nbsp;&nbsp;&nbsp;</option>  
                </select>  
            <td>  
                <input name="operando2" type="text" style="text-align:right">  
            </td>  
        </tr>  
    </table>  
  
    <br>  
  
    <input name="Calcular" value="Calcular" type="submit">  
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
    <input name="limpiar" value="Limpiar" type="reset">  
</form>
```

CALCULADORA

+

▼

Calcular

Limpiar

El código del servlet es el siguiente.

```
package Servelts;

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

public class CalculadoraServlet extends HttpServlet {
    public void service(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        double op1, op2, result;
        int operacion;
        String simb_op[] = { "+", "-", "*", "/" };
        ServletOutputStream out = res.getOutputStream();
        op1 = Double.parseDouble(req.getParameter("operando1"));
        op2 = Double.parseDouble(req.getParameter("operando2"));
        operacion = Integer.parseInt(req.getParameter("operacion"));
        result = calcula(op1, op2, operacion);
        out.println("<html>");
        out.println("<head><title>Resultado de calcular conServlet</title></head>");
        out.println("<body BGCOLOR = \"#E0E0FF\" TEXT= \"blue\">");
        out.println("<h1><center>La operacion efectuada es:</center></h1>");
        out.println("<h2><b><center>" + op1 + " " + simb_op[operacion - 1] + " " + op2 + " = " + result + "</center></b></h2>");

        out.println("</body>");
        out.println("</html>");
        out.close();
    }

    public double calcula(double op1, double op2, int operacion) {
        double result = 0;
        switch (operacion) {
            case 1:
                return op1 + op2;
            case 2:
                return op1 - op2;
            case 3:
                return op1 * op2;
            case 4:
                return op1 / op2;
        }
        return result;
    }
}
```

Si introducimos "4 * 6", obtendremos lo siguiente.

La operacion efectuada es:

4.0 * 6.0 = 24.0

EJERCICIO 4

En este caso se trata de una primitiva: se eligen 6 números y se prueba suerte. En esencia, este programa es similar al anterior: se obtienen unos valores en un HTML que, mediante un form, se envían al servlet.

```
<form action="PrimitivaServlet" method=POST>
    <br>
    <br>
    <center>
        Introduce tu combinación y pulsa el botón de enviar<BR>
        <br>NUM1:<input type=text name=NUM1>
        <br>NUM2:<input type=text name=NUM2>
        <br>NUM3:<input type=text name=NUM3>
        <br>NUM4:<input type=text name=NUM4>
        <br>NUM5:<input type=text name=NUM5>
        <br>NUM6:<input type=text name=NUM6>
        <br><br>
        <input type=submit value="Enviar Combinación">
        <input type=reset value=Borrar>
    </center>
</form>
```

Una vez introducidos los 6 números, al pulsar "Enviar Combinación" se mandan al servlet. Este genera en el método init un array ordenado de 6 números aleatorios no repetidos que representarán la combinación ganadora. Posteriormente genera el código HTML donde muestra la combinación elegida y realiza la comprobación de aciertos.

Este es el código del servlet.

```
package Servlet;

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

public class PrimitivaServlet extends HttpServlet {

    int primi[] = new int[6], combiUsuario[] = new int[6];
    int i, contador = 0, aux, aciertos = 0;
    Random rand = new Random();

    public void init(ServletConfig config) throws ServletException {
```

```

        super.init(config);
//generamos los números
        while (contador < 6) {
            aux = rand.nextInt(48) + 1;
            if (!comprueba(primi, aux)) {
                primi[contador] = aux;
                contador++;
            }
        }
//ordenamos el array
        Arrays.sort(primi);
    }

    private boolean comprueba(int array[], int num) {
        for (int i = 0; i <= 5; i++) {
            if (primi[i] == num) {
                return true;
            }
        }
        return false;
    }

    public void service(HttpServletRequest petición, HttpServletResponse respuesta)
        throws ServletException, IOException {
        aciertos = 0;
        respuesta.setContentType("text/html");
        ServletOutputStream out = respuesta.getOutputStream();
        out.println("<html>");
        out.println("<head><title>Primitiva</title></head>");
        out.println("<body>");
        combiUsuario[0] = Integer.parseInt(petición.getParameter("NUM1"));
        combiUsuario[1] = Integer.parseInt(petición.getParameter("NUM2"));
        combiUsuario[2] = Integer.parseInt(petición.getParameter("NUM3"));
        combiUsuario[3] = Integer.parseInt(petición.getParameter("NUM4"));
        combiUsuario[4] = Integer.parseInt(petición.getParameter("NUM5"));
        combiUsuario[5] = Integer.parseInt(petición.getParameter("NUM6"));
        out.println("<center><h2>Primitiva Servlet</h2></center>");
//imprimimos todos los números de la combinación del usuario
        out.print("<p>Tu combinación es: </p><b>");
        for (i = 0; i < 6; i++) {
            out.print(" " + combiUsuario[i]);
        }
        out.print("</b>");
//comprobamos la combinación
        for (i = 0; i <= 5; i++) {
            if (Arrays.binarySearch(primi, combiUsuario[i]) >= 0) {
                out.println("<p>Número acertado: <b> "+combiUsuario[i]+" </b></p> ");
                aciertos++;
            }
        }
    }

```



```

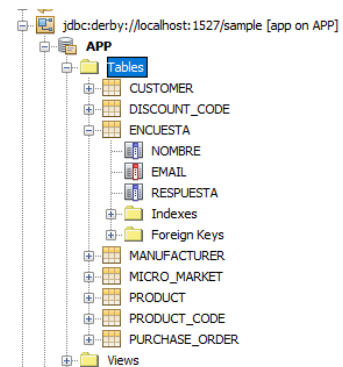
    }
}
out.println("<p>Números acertados: <b>" + aciertos + "</b></p>");
//imprimimos todos los números de la combinación ganadora
out.print("<p>La combinación ganadora es:</p><B>");
for (i = 0; i < 6; i++) {
    out.print(" " + primi[i]);
}
out.print("</B>");
out.println("</body></html>");
out.close();
}
}

```

EJERCICIO 5

Este ejercicio propone interactuar con una base de datos mediante un Servlet. El index contiene un formulario con un campo a recoger y una encuesta. El servlet recibe los datos del formulario y los añade a la base de datos de ejemplo. Después, lee los datos de la base de datos, realiza un recuento de la encuesta y muestra los resultados.

En primer lugar, debemos añadir la tabla Encuesta a la base de datos sample:



Tras esto, ya estamos listos para ejecutar el proyecto. Podemos ver el código comentado:

```

<html>

<head>
    <meta charset="UTF-8">
    <title>Ejemplo Encuesta</title>
</head>

<body bgcolor=white>
<center>
    <h2>Por favor rellene todos los datos</h2>
    <!-- formulario de la encuesta -->
    <form action="EncuestaServlet" method=POST>
        <br>
        Nombre:
        <br><br>
        <input type=text name=NOMBRE>
        <br><br>
        E-Mail:
        <br><br>
        <input type=text name=EMAIL>
        <br><br>
        <b>Pregunta:</b> ¿Piensas utilizar a los Servlets para los proyectos a parti
r de ahora?
        <br><br>
        Si<input type=radio name=RESPUESTA value=SI>

```

```

        No☐
        <br>
        <br>
        <input type="submit" value="Enviar">
        <input type="reset" value="Borrar">
    </form>
</center>
</body>

</html>

```

Respecto al servlet, es algo más complejo:

```

package Servlets;

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

public class EncuestaServlet extends HttpServlet {
    Statement mandato = null;
    Connection conexion = null;

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        // Compruebo que el driver funciona.
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver");
        } catch (Exception e) {
            System.out.println("Error al cargar el driver JDBC/ODBC.");
            return;
        }

        // Intento conectarme a la base de datos.
        try {
            conexion = DriverManager.getConnection("jdbc:derby://localhost:1527/sample",
"app", "app");
            mandato = conexion.createStatement();
        } catch (SQLException e) {
            System.out.println("Problemas al conectar con la base de datos");
        }
    }

    public void service(HttpServletRequest peticion, HttpServletResponse respuesta)
        throws ServletException, IOException {
        /* creación del flujo de salida hacia el cliente */
        ServletOutputStream out = respuesta.getOutputStream();
        respuesta.setContentType("text/html");

        /* recuperamos los valores que nos manda el cliente */
        String strNombre = peticion.getParameter("NOMBRE");
        String strEmail = peticion.getParameter("EMAIL");
        String strRespuesta = peticion.getParameter("RESPUESTA");

        /* insertamos los datos en la base de datos */
        try {
            mandato.executeUpdate("INSERT INTO ENCUESTA VALUES( '" + strNombre + "',
'" + strEmail + "', '"
                + strRespuesta + "')");
        } catch (SQLException e) {
            System.out.println("ERROR PORQUE LA TABLA NO ESTÁ CREADA. ");
            System.out.println(e);
            return;
        }

        /* leemos todos los registros para crear la estadística */try {
            int intSI = 0;

```

```

        int intNO = 0;
        ResultSet resultado = mandato.executeQuery("SELECT RESPUESTA FROM ENCUESTA");
;

        while (resultado.next()) {
            String resp = resultado.getString("RESPUESTA");
            if (resp.compareTo("SI") == 0)
                intSI++;
            else
                intNO++;
        }

        // Imprimo página HTML mostrando los resultados de la encuesta
        out.println("<h2><center>Encuesta Servlet</center></h2>");
        out.println("<BR>Gracias por participar en esta encuesta.");
        out.println("<BR>Los resultados hasta este momento son :");
        out.println("<BR>                SI : " + intSI);
        out.println("<BR>                NO : " + intNO);
        out.println("<a >");
    } catch (IOException e) {
        System.out.println(e);
        return;
    } catch (SQLException e) {
        System.out.println(e);
        return;
    }
}

public void destroy() {
    try {
        conexion.close();
    } catch (SQLException e) {
        System.out.println(e);
    }
}
}

```

Si ejecutamos el proyecto, tras esperar unos segundos, tenemos lista la página del index con la encuesta:

← → ↻ ⓘ localhost:8080/1-5-Acceso-datos/

Por favor rellene todos los datos

Nombre:

E-Mail:

Pregunta: ¿Piensas utilizar a los Servlets para los proyectos a partir de ahora?
 SI ☐ No ☒

Si pulsamos el botón de enviar, se ejecuta el servlet asociado y obtenemos la siguiente página:

← → ↻ ⓘ localhost:8080/1-5-Acceso-datos/EncuestaServlet

Encuesta Servlet

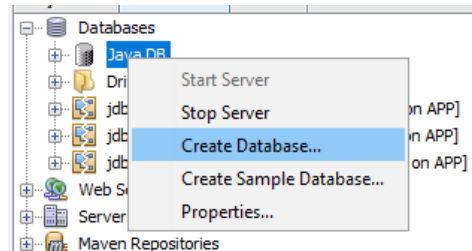
Gracias por participar en esta encuesta.
 Los resultados hasta este momento son :
 SI : 2
 NO : 2

APARTADO 2 – JSP'S Y BASES DE DATOS

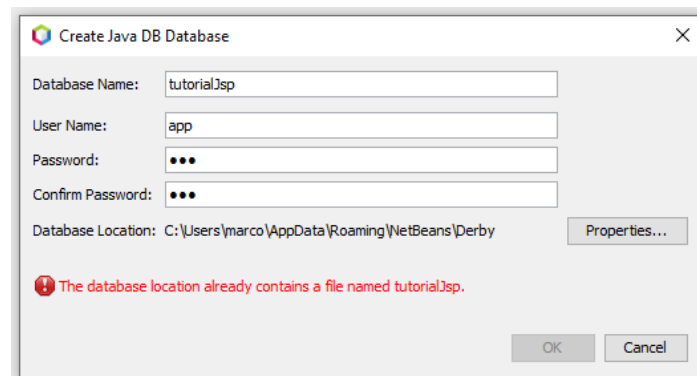
EJERCICIO 1

Este ejercicio propone interactuar con una base de datos, esta vez mediante un JSP.

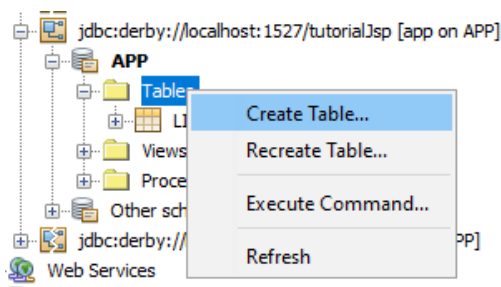
En primer lugar, debemos crear la base de datos propuesta. Para ello, debemos crear una nueva base de datos en el apartado servicios:



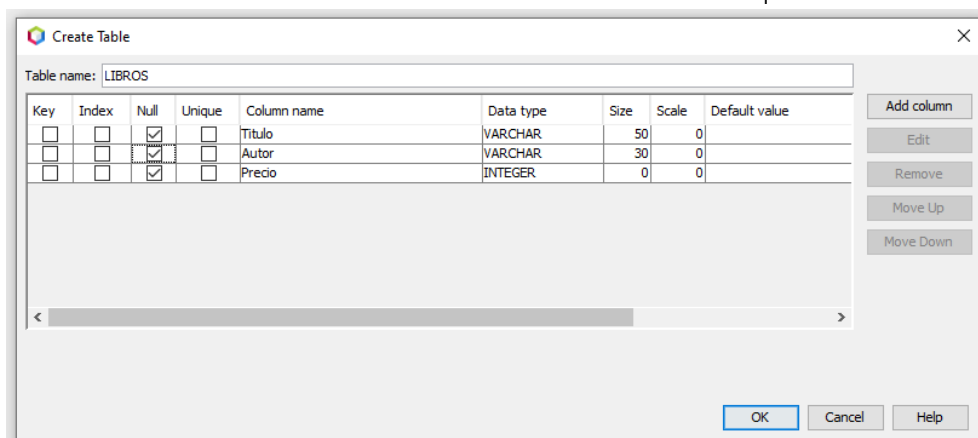
La nombraremos como viene en el código de ejemplo:



Y añadimos mediante la interfaz las tablas pertinentes:



Añadimos el esquema de Libro:



Podemos insertar varios valores para tener algo que mostrar mediante la herramienta *execute command*:

```
INSERT INTO LIBROS VALUES('Plan elecotoral del PP','Mariano Rajoy', 1)
INSERT INTO LIBROS VALUES('Juego de trolos',' Cervantes', 15)
INSERT INTO LIBROS VALUES('Star wars',' George R. Martin', 20)
```

El código del JSP queda de la siguiente manera:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>

<head>
  <title>Tutorial JSP, Base de Datos</title>
</head>

<body>
  <%@ page import="java.sql.*"%>
  <%!
    // Declaraciones de las variables utilizadas para la
    // conexión a la base de datos y para la recuperación de
    // datos de las tablas
    Connection c;
    Statement s;
    ResultSet rs;
    ResultSetMetaData rsmd;
  %>

  <%
    // Inicialización de las variables necesarias para la
    // conexión a la base de datos y realización de consultas
    Class.forName("org.apache.derby.jdbc.ClientDriver");
    c = DriverManager.getConnection("jdbc:derby://localhost:1527/tutorialJsp");
    s = c.createStatement();
    rs = s.executeQuery("SELECT * FROM LIBROS");
    rsmd = rs.getMetaData();%>

    <table width="100%" border="1">
      <tr>
        <% for( int i=1; i <= rsmd.getColumnCount(); i++ ) { %>
        <%-
-      Obtenemos los nombres de las columnas y los colocamos como cabecera de la tabla --%>

        <th>
          <%= rsmd.getColumnLabel( i ) %>
        </th><% } %>
      </tr><% while( rs.next() ) { %>
      <tr>
        <% for( int i=1; i <= rsmd.getColumnCount(); i++ ) { %>
        <%-
-      Recuperamos los valores de las columnas que corresponden a cada uno de los registros
        de la tabla.
        Hay que recoger correctamente el tipo de dato que contiene la columna -
        %>

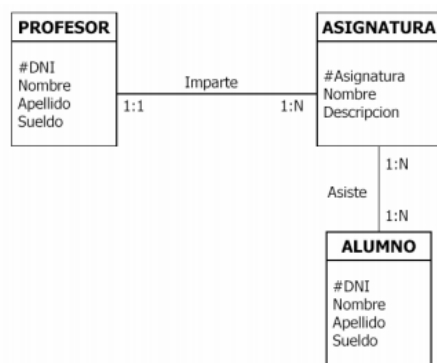
        <% if( i == 3 ) { %>
        <td>
          <%= rs.getInt( i ) %></td>
        <% } else { %>
        <td>
          <%= rs.getString( i ) %>
        </td><% } } %>
      </tr><% } %>
    </table>
  </body>
</html>
```

Y al ejecutar el archivo JSP, tras esperar a que se levante el servidor, obtenemos el siguiente resultado:

TITULO	AUTOR	PRECIO
Plan electoral del PP	Mariano Rajoy	1
Juego de trolos	Cervantes	15
Star wars	George R. Martin	20
Plan electoral del PP	Mariano Rajoy	1
Juego de trolos	Cervantes	15
Star wars	George R. Martin	20
Plan electoral del PP	Mariano Rajoy	1
Juego de trolos	Cervantes	15
Star wars	George R. Martin	20
Plan electoral del PP	Mariano Rajoy	1

EJERCICIO 2

En este ejercicio se debe modelar la siguiente base de datos:



Para ello, se ha escrito el siguiente script, que se puede encontrar en los archivos de la práctica:

```
-- Creo base de datos
CREATE DATABASE 'Universidad'

-- Añado tablas

-- -- Tabla profesor
CREATE TABLE Profesor (
  DNI varchar(15) NOT NULL,
  Nombre varchar(20) NOT NULL,
  Apellido varchar(30) NOT NULL,
  Sueldo FLOAT NOT NULL
);

-- -- Tabla asignatura
CREATE TABLE Asignatura (
  IdAsignatura varchar(15) NOT NULL,
  Nombre varchar(20) NOT NULL,
  Descripción varchar(30) NOT NULL,
  DNI varchar(15) NOT NULL
);
```

```
-- -- Tabla alumno
CREATE TABLE Alumno (
    DNI varchar(15) NOT NULL,
    Nombre varchar(20) NOT NULL,
    Apellido varchar(30) NOT NULL,
    Sueldo FLOAT NOT NULL
);

-- -- Tabla intermedia de relación "alumno asiste a clase"
CREATE TABLE Asiste (
    DNI varchar(15) NOT NULL, -- dni del alumno
    IdAsignatura varchar(15) NOT NULL -- id de asignatura
);

-- Añado PKs
ALTER TABLE Profesor ADD CONSTRAINT KEY_PROFESOR PRIMARY KEY(DNI);
ALTER TABLE Asignatura ADD CONSTRAINT KEY_ASIGNATURA PRIMARY KEY(IdAsignatura);
ALTER TABLE Alumno ADD CONSTRAINT KEY_AUMNO PRIMARY KEY(DNI);

-- Relación 1 profesor imparte N asignaturas
ALTER TABLE Asignatura ADD CONSTRAINT Imparte FOREIGN KEY(DNI) REFERENCES Profesor(DNI);

-- Relación 1:N alumnos asisten a 1:N asignaturas
-- -- Creo PK conjunta en Asiste
ALTER TABLE Asiste ADD CONSTRAINT KEY_ASISTE PRIMARY KEY(DNI, IdAsignatura);

-- -- Añado referencias de la PK de la tabla intermedia a las tablas relacionadas
ALTER TABLE Asiste ADD CONSTRAINT ALUMNO_ASISTE FOREIGN KEY(DNI) REFERENCES Alumno(DNI);
ALTER TABLE Asiste ADD CONSTRAINT ASIGNATURA_ASISTE FOREIGN KEY(IdAsignatura) REFERENCES Asignatura(IdAsignatura);

-- Valores añadidos a la BBDD para comprobar su funcionamiento.

INSERT INTO Profesor VALUES('123456789N','Roberto','Barchino',1500);

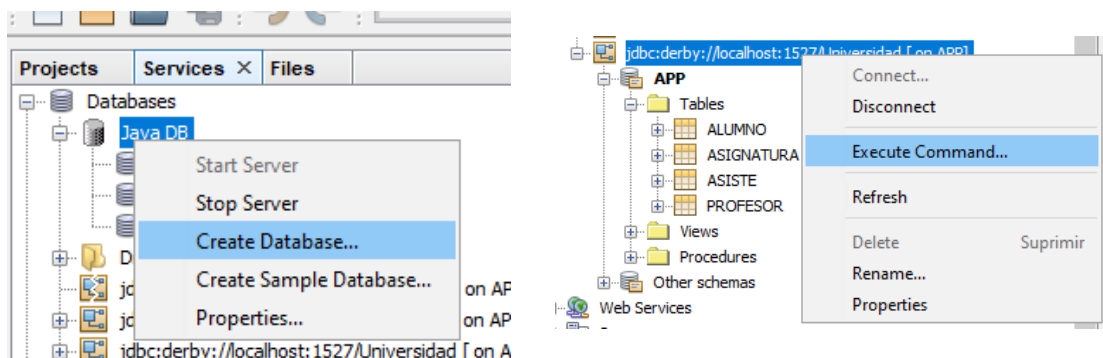
INSERT INTO Asignatura VALUES('ASIG9','Web','Asignatura de hacer webs','123456789N');

INSERT INTO Alumno VALUES('25395723F','Daniel','Manzano',-1);

INSERT INTO Asiste VALUES('25395723F','ASIG9');
```

Creamos una base de datos en Services → Databases → Java DB → Create Database.

Tras crearla, ejecutamos el script anterior para crear las tablas, relaciones e insertar los datos:



Tras esto, creamos 3 archivos:

- El html índice leerá el nombre del profesor y llamará a un Servlet.
- El Servlet llamará a un JSP pasándole por argumento el nombre del profesor.
- El JSP procesa la consulta SQL y muestra los resultados.

HTML

El HTML contiene un formulario que pide el nombre del profesor y llama al JSP al pulsar el botón del formulario.

```
<html>
<head>
  <title>Ejemplo Profesor</title>
</head>
<body>
  <h2>¿De que profesor quieres ver los alumnos?</h2>
  <form action="LlamadaAlJsp" method=POST>
    <label>Nombre del profesor:</label><input type=text name="profesor">
    <input type=submit value=Enviar>
  </form>
</body>
</html>
```

SERVLET

El servlet actúa de intermediario llamando al JSP. Es interesante ver el método de proceso de solicitud:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    // Tomo el nombre del profesor del formulario
    String strNombre = request.getParameter("profesor");

    // Lo añado al request
    request.setAttribute("profesor", strNombre);

    // Llamo al jsp
    RequestDispatcher dispatcher = request.getRequestDispatcher("/VerProfes.jsp");

    // y le paso el nombre del profesor
    dispatcher.forward(request, response);}
```

JSP

Finalmente, el JSP lee el nombre del profesor pasado por argumento en *request*, y realiza la consulta SQL:

```
<%
    Class.forName("org.apache.derby.jdbc.ClientDriver");
    c = DriverManager.getConnection("jdbc:derby://localhost:1527/Universidad");
    s = c.createStatement();
    String nombreProfesor = (String)request.getAttribute("profesor");
    rs = s.executeQuery("SELECT ALUMNO.DNI, ALUMNO.NOMBRE, ALUMNO.APELLIDO "
        + "FROM ALUMNO INNER JOIN ASISTE ON ASISTE.DNI = ALUMNO.DNI "
        + "INNER JOIN ASIGNATURA ON ASIGNATURA.IDASIGNATURA "
```



```

+ " = ASISTE.IDASIGNATURA INNER JOIN PROFESOR "
+ "ON PROFESOR.DNI=ASIGNATURA.DNI "
+ "WHERE PROFESOR.NOMBRE='"+ nombreProfesor +"';

rsmd = rs.getMetaData();
%>

```

La tabla de los datos queda similar a la generada por el ejercicio 1:

```

<table width="100%" border="1">

  <tr>

    <% for (int i = 1; i <= rsmd.getColumnCount(); i++) { %>

      <th>

        <%= rsmd.getColumnLabel(i) %>

      </th><% } %>

    </tr><% while (rs.next()) { %>

      <tr>

        <% for (int i = 1; i <= rsmd.getColumnCount(); i++) { %>

          <% if (i == 3) { %>

            <td>

              <%= rs.getString(i) %></td>

            <% } else { %>

              <td>

                <%= rs.getString(i) %>

              </td><% }

            } %>

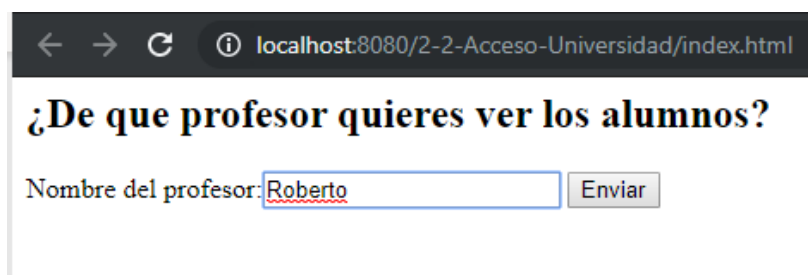
          </tr><% } %>

        </table>

```

EJECUCIÓN

Si ejecutamos en el navegador el archivo índice del proyecto 2-2-Acceso-Universidad, obtenemos la siguiente ventana:



¿De que profesor quieres ver los alumnos?

Nombre del profesor:

Si escribimos "Roberto" y pulsamos enviar, generamos la siguiente web:

DNI	NOMBRE	APELLIDO
25395723F	Daniel	Manzano

EJERCICIO 3

Finalmente tenemos la gestión de sesiones. Esta se realiza mediante un formulario en HTML que pide introducir el nombre.

```
<!DOCTYPE html>

<html>
  <head>
    <meta charset="UTF-8">
    <title>Ejemplo de Sesión</title>
  </head>
  <body>
    <h1>Ejemplo de sesión</h1>
    <form method="post" action="sesionEje.jsp">
      Por favor, introduce tu nombre:
      <input type="text" name="nombre">
      <input type="submit" value="Enviar información">
    </form>
  </body>
</html>
```

Ejemplo de sesión

Por favor, introduce tu nombre:

Mediante el submit, se redirige a un JSP que cargará el valor introducido al valor del elemento "session" mediante setAttribute.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo de Sesión</title>
  </head>
  <body>
    <%
      String val = request.getParameter("nombre");
      if (val != null) {
        session.setAttribute("Nombre", val);
      }
    %>
    <center>
      <h1>Ejemplo de Sesión</h1>
      <p>¿Dónde quieres ir?</p>
      <a href="sesionEje1.jsp">Ir a Página 1</a>
      <a href="sesionEje2.jsp">Ir a Página 2</a>
    </center>
  </body>
</html>
```

Con los enlaces de las páginas 1 y 2 (cuyo código es igual, salvo por el número de página), se obtiene el valor del elemento "session" con `getAttribute`.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo de Sesión</title>
  </head>
  <body>
    <center>
      <h1>Ejemplo de Sesión</h1>
      Hola, <%=session.getAttribute("Nombre")%>
      <p>Bienvenido a la página 1</p>
    </body>
  </html>
```

Ejemplo de Sesión

Hola, Lucas

Bienvenido a la página 1