

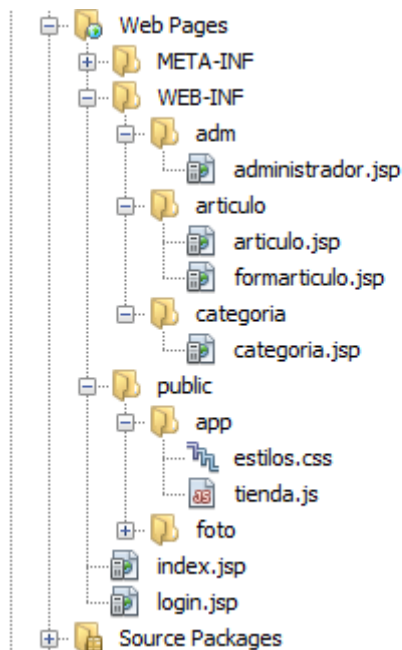
Lenguaje de Programación II

Docente: Ing. Díaz Leva Teodoro

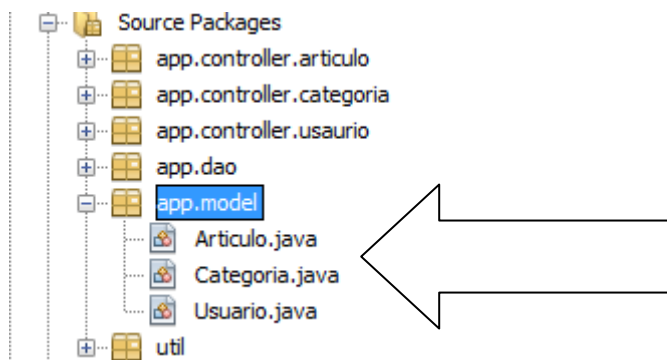
Tema: Mantenimiento de Tablas

En el siguiente laboratorio se desarrollan las operaciones de mantenimiento (create, read, update, delete) en la tabla artículos de la base de datos tienda.

Estructura de archivos del proyecto “webApp”



Crear clases artículo y categoria con los métodos setters and getters en el paquete app.model:



```
public class Artículo {
    private int idarticulo;
    private int idcategoria;
    private String nombre;
    private String descripcion;
    private double precio;
    private String foto;

    public int getIdarticulo() {
        return idarticulo;
    }
    public void setIdarticulo(int idarticulo) {
        this.idarticulo = idarticulo;
    }

    public int getIdcategoria() {
        return idcategoria;
    }

    public void setIdcategoria(int idcategoria) {
        this.idcategoria = idcategoria;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

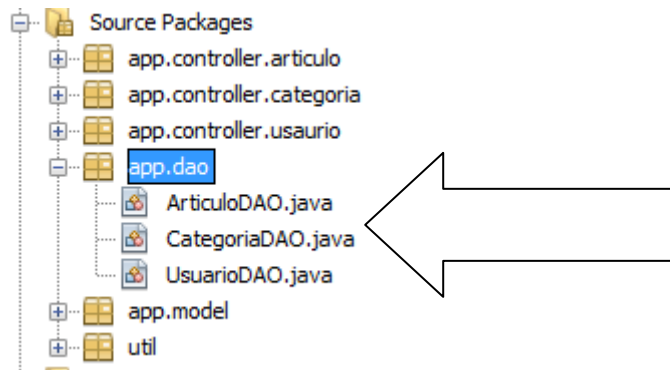
    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public String getFoto() {
        return foto;
    }

    public void setFoto(String foto) {
        this.foto = foto;
    }
}
```

```
public class Categoria {  
  
    private int idcategoria;  
    private String categoria;  
  
    public int getIdcategoria() {  
        return idcategoria;  
    }  
  
    public void setIdcategoria(int idcategoria) {  
        this.idcategoria = idcategoria;  
    }  
  
    public String getCategoria() {  
        return categoria;  
    }  
  
    public void setCategoria(String categoria) {  
        this.categoria = categoria;  
    }  
}
```



Crear clases articuloDAO y categoriaDAO con los métodos get, sabe, delete y udpate en el paquete app.dao:

```
import app.model.Articulo;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import util.AccesoBD;
```

public class ArtículoDAO {

```
    public static ArrayList<Articulo> lista() {
        //arreglo de objetos tipo articulo
        ArrayList<Articulo> lista = new ArrayList<Articulo>();

        //conexion a la BD
        Connection cn = AccesoBD.abrir();

        //variable tipo articulo
        Articulo ar;

        //instrucción sql para extraer todos los artículos de la BD
        String sql = "select*from articulo";

        //objeto resultset para almacenar en memoria los articulos
        ResultSet rs;

        //objeto preparastamet para ejecutar instrucción sql a traves de su método
        //executequery y la conexion cn
        PreparedStatement stm;
        try {
            stm = cn.prepareStatement(sql);

            //ejecutar objeto preparedstament
            rs = stm.executeQuery();

            //leer resulset
            while (rs.next()) {
                //crear objeto ar
                ar = new Articulo();

                //encapsular datos en obejto ar
                ar.setIldarticulo(rs.getInt(1));
            }
        }
    }
}
```

```
        ar.setIdcategoria(rs.getInt(2));
        ar.setNombre(rs.getString(3));
        ar.setDescripcion(rs.getString(4));
        ar.setPrecio(rs.getDouble(5));
        ar.setFoto(rs.getString(6));
        //asignar objeto al arreglo de objeto lista
        lista.add(ar);
    }
} catch (SQLException ex) {
    ex.printStackTrace();
}
return lista;
}

//-----
public static Artículo get(int id) {
    //conexion a la BD
    Connection cn = AccesoBD.abrir();

    //variable tipo articulo
    Artículo ar = null;

    //instrucción sql para extraer todos los artículos de la BD
    String sql = "select*from articulo where idarticulo=?";

    //objeto resultset para almacenar en memoria los articulos
    ResultSet rs;

    //objeto preparastamet para ejecutar instruccion sql a traves de su metodo
    //executequery y la conexion cn
    PreparedStatement stm;
    try {
        stm = cn.prepareStatement(sql);
        stm.setInt(1, id);

        //ejecutar objeto preparedstament
        rs = stm.executeQuery();

        //leer resulset
        if (rs.next()) {
            //crear objeto ar
            ar = new Artículo();

            //encapsular datos en obejto ar
            ar.setIdarticulo(rs.getInt(1));
            ar.setIdcategoria(rs.getInt(2));
            ar.setNombre(rs.getString(3));
            ar.setDescripcion(rs.getString(4));
            ar.setPrecio(rs.getDouble(5));
            ar.setFoto(rs.getString(6));
        }

    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    return ar;
}
```

```
//-----
public static void save(Articulo art) {
    String sql = "insert into articulo(nombre,idcategoria,descripcion,precio)values(?,?,?,?)";
    PreparedStatement ps;
    Connection cn = AccesoBD.abrir();
    try {
        ps = cn.prepareStatement(sql);
        ps.setString(1, art.getNombre());
        ps.setInt(2, art.getIdcategoria());
        ps.setString(3, art.getDescripcion());
        ps.setDouble(4, art.getPrecio());

        //ejecutar sentencia sql
        ps.executeUpdate();

        //cerrar objetos
        ps.close();
        cn.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

//-----
public static void update(Articulo art) {
    String sql = "update articulo set nombre=?,idcategoria=?,descripcion=?,precio=? where idarticulo=?";

    PreparedStatement ps;
    Connection cn = AccesoBD.abrir();
    try {
        ps = cn.prepareStatement(sql);
        ps.setString(1, art.getNombre());
        ps.setInt(2, art.getIdcategoria());
        ps.setString(3, art.getDescripcion());
        ps.setDouble(4, art.getPrecio());
        ps.setInt(5, art.getIdarticulo());

        //ejecutar sentencia sql
        ps.executeUpdate();

        //cerrar objetos
        ps.close();
        cn.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
```

```
public static void delete(int codigo) {
    String sql = "delete from articulo where idarticulo=?";
    PreparedStatement ps;
    Connection cn = AccesoBD.abrir();
    try {
        ps = cn.prepareStatement(sql);
        ps.setInt(1, codigo);

        //ejecutar sentencia sql
        ps.executeUpdate();

        //cerrar objetos
        ps.close();
        cn.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
```

```
import app.model.Categoria;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import util.AccesoBD;
```

```
public class CategoriaDAO {
    public static ArrayList<Categoria> lista() {

        //arreglo de objetos tipo articulo
        ArrayList<Categoria> lista = new ArrayList<Categoria>();

        //conexion a la BD
        Connection cn = AccesoBD.abrir();

        //variable tipo articulo
        Categoria ar;

        //instrucción sql para extraer todos los artículos de la BD
        String sql = "select*from categoria";

        //objeto resultset para almacenar en memoria los articulos
        ResultSet rs;

        //objeto preparastamet para ejecutar instrucción sql a través de su método
        //executequery y la conexion cn
        PreparedStatement stm;
        try {
            stm = cn.prepareStatement(sql);

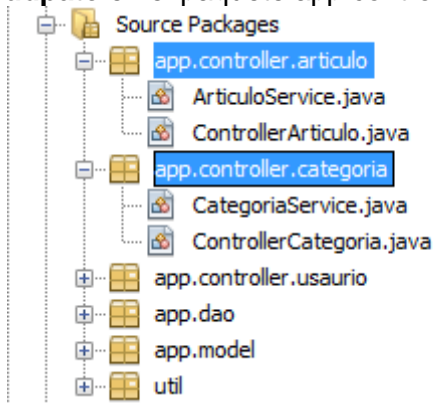
            //ejecutar objeto preparedstament
            rs = stm.executeQuery();
        }
    }
}
```

```
//leer resultSet
while (rs.next()) {
    //crear objeto ar
    ar = new Categoria();

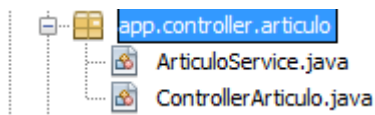
    //encapsular datos en objeto ar
    ar.setIIdcategoria(rs.getInt(1));
    ar.setCategoria(rs.getString(2));

    //asignar objeto al arreglo de objeto lista
    lista.add(ar);
}
} catch (SQLException ex) {
    ex.printStackTrace();
}
return lista;
}
```

Crear clases `ArticuloService` y `CategoriaService` con los métodos `get`, `sabe`, `delete` y `update` en el paquete `app.controller.articulo` y `app.controller.categoria`:



`ArticuloService` y `CategoriaService`, utilizarán los métodos de las clases `ArticuloDAO` y `CategoriaDAO` para servir a los servlets `ControllerArticulo` y `ControllerCategoria` respectivamente:



```
package app.controller.articulo;
```

```
import app.dao.ArticuloDAO;
import app.model.Articulo;
import java.util.ArrayList;
```

```
public class ArticuloService {

    public static ArrayList<Articulo> lista() {
        return ArticuloDAO.lista();
    }
}
```

ArticuloService retorna los métodos de la clase `ArticuloDAO`, que se utilizará en el servlet. También el servlet podría comunicarse directamente con el DAO y en este caso no se utilizaría el **ArticuloService**


```
}  
  
public static Artículo get(int id) {  
    return ArtículoDAO.get(id);  
}  
  
public static void save(Artículo ar) {  
    ArtículoDAO.save(ar);  
}  
  
public static void update(Artículo ar) {  
    ArtículoDAO.update(ar);  
}  
  
public static void delete(int id) {  
    ArtículoDAO.delete(id);  
}  
}
```

✓ **Creamos servlet ControllerArticulo:**

```
package app.controller.articulo;  
  
import app.controller.categoria.CategoriaService;  
import app.model.Artículo;  
import app.model.Categoria;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
@WebServlet(name = "ServletArticulo", urlPatterns = {"/articulos", "/editar", "/nuevo", "/grabar",  
"/eliminar"})  
public class ControllerArticulo extends HttpServlet {  
  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        //captura el valor del path, según este valor se realizan las operaciones de mantenimiento  
        String path = request.getServletPath();  
  
        //si el path es articulos  
        if (path.equals("/articulos")) {  
            //arreglo de articulos  
            ArrayList<Artículo> articulos = ArtículoService.lista();  
  
            //objeto articulos se asigna a la variable articulos  
            request.setAttribute("articulos", articulos);  
  
            // Redirigir (forward), funcionalmente semejante a sendRedirect(), se trata de redirigir la  
            //petición a articulo.jsp (en esta página se recogerá el valor de la variable artículos)
```

```
request.getRequestDispatcher("WEB-INF/articulo/articulo.jsp").forward(request,
response);
}

//si el path es editar
if (path.equals("/editar")) {
    //categorias para listar las categorias en la lista desplegable <select></select>
    //del formarticulo.jsp
    List<Categoria> categorias = CategoriaService.lista();
    request.setAttribute("categorias", categorias);

    //captura del valor del id del artículo que viene por la url,determinará
    //qué artículo editar
    int id = Integer.parseInt(request.getParameter("id"));

    //buscar articulo a editar
    Articulo articulo = ArticuloService.get(id);

    //objeto artículo encapsula datos del artículo a editar, se asigna a la variable
    //articulo
    request.setAttribute("articulo", articulo);

    // Redirigir al formulario formarticulo.jsp, en el cual se mostrará artículo a editar
    request.getRequestDispatcher("WEB-INF/articulo/formarticulo.jsp").forward(request,
response);
}

//si el path es nuevo
if (path.equals("/nuevo")) {
    //categorias para listar las categorías en la lista desplegable <select></select>
    //del formarticulo.jsp
    List<Categoria> categorias = CategoriaService.lista();
    request.setAttribute("categorias", categorias);

    //crear objeto artículo y se asigna a la variable artículo, sin ningún dato
    request.setAttribute("articulo", new Articulo());

    // Redirigir al formulario formarticulo.jsp, en el cual se ingresará nuevo articulo
    request.getRequestDispatcher("WEB-INF/articulo/formarticulo.jsp").forward(request,
response);
}

//si el path es grabar
if (path.equals("/grabar")) {
    //objeto articulo
    Articulo ar = new Articulo();

    //tanto para nuevo y editar se necesita los datos del artículo,
    //por lo tanto estos datos son capturados del formarticulo.jsp
    int id = Integer.parseInt(request.getParameter("txtid"));

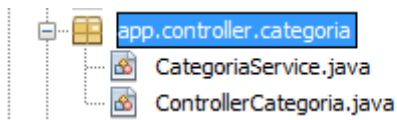
    ar.setIdcategoria(Integer.parseInt(request.getParameter("cat")));
    ar.setNombre(request.getParameter("txtnom"));
    ar.setDescripcion(request.getParameter("txtdesc"));
    ar.setPrecio(Double.parseDouble(request.getParameter("txtpre")));
}
```

```
//si el valor del id es mayor a cero, entonces se editara articulo
//de lo contrario se agregará nuevo artículo a la base de datos
if (id > 0) {
    ar.setIIdarticulo(id);
    ArtículoService.update(ar);
} else {
    ArtículoService.sabe(ar);
}

//después de editar o grabar nuevo artículo se redirige a la lista
//de articulo a través del servlet :/articulos

//request.getContextPath() :es el Path del contexto de la aplicación
//Es decir donde está instalada la aplicación Ej: /webApp
response.sendRedirect(request.getContextPath() + "/articulos");
}

//si el path es eliminar
if (path.equals("/eliminar")) {
    //recibe valor del id del articulo a eliminar que viene por la url
    ArtículoService.delete(Integer.parseInt(request.getParameter("id")));
    //redirigir a la lista de articulo a través del servlet :/articulos
    response.sendRedirect(request.getContextPath() + "/articulos");
}
}
```



```
package app.controller.categoria;

import app.dao.CategoriaDAO;
import app.model.Categoria;
import java.util.ArrayList;

public class CategoriaService {

    public static ArrayList<Categoria> lista() {

        return CategoriaDAO.lista();
    }
}
```

```
package app.controller.categoria;

import app.model.Categoria;
import java.io.IOException;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

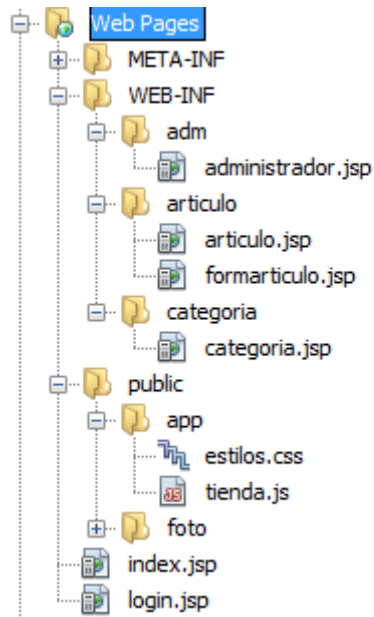
@WebServlet(name = "ControllerCategoria", urlPatterns = {"/lista"})
public class ControllerCategoria extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        String path = request.getServletPath();

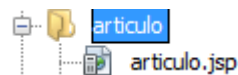
        if (path.equals("/lista")) {
            List<Categoria> categorias = (List<Categoria>) CategoriaService.lista();
            request.setAttribute("categorias", categorias);
            request.getRequestDispatcher("WEB-INF/categoria/categoria.jsp").forward(request,
response);
        }
    }
}
```

Nota : en este laboratorio no se está utilizando el servlet: ControllerCategoria, pero si se utiliza CategoriaService, ya que la tabla articulo se relaciona con la tabla categoría a través del campo idcategoria

La presentación de la aplicación:



Archivos articulo.jsp de la carpeta articulo:



```
<%@page import="java.util.ArrayList"%>
<%@page import="app.model.Articulo"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>

    <!-- vinculo al archivo tienda.js que contendrá funciones javascript-->
    <script type="text/javascript" src="public/app/tienda.js"></script>

    <!-- vinculo al archivo estilos.css que contendrá los estilos de la aplicación-->
    <link rel="stylesheet" type="text/css"
href="<%=request.getContextPath()%>/public/app/estilos.css">

  </head>
```

```
<body>
<h1>Articulos</h1>
<div>
<table class="tabla-articulos">

<tr>
<td>Articulo</td>
<td>Descripcion</td>
<td>Precio</td>
<td colspan="2" style="text-align: center"><a href="nuevo">
</a>
</td>
</tr>

<%

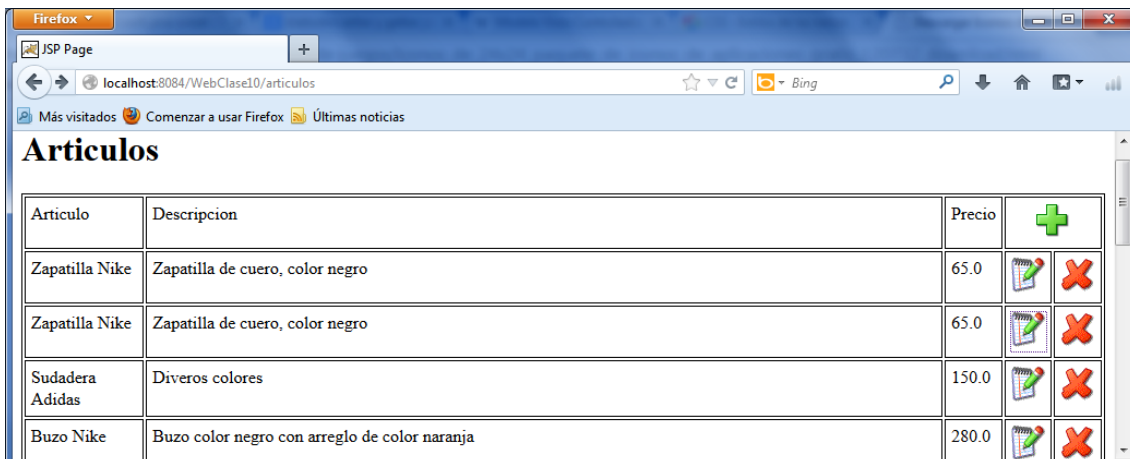
    ArrayList<Articulo> articulos = (ArrayList<Articulo>)
    request.getAttribute("articulos");
    for (Articulo x : articulos) {









%>

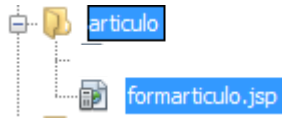
<tr>
<td><%= x.getNombre()%></td>
<td><%=x.getDescripcion()%></td>
<td><%= x.getPrecio()%></td>
<td>
<a href="editar?id=<%=x.getIdarticulo()%>">
</a>
</td>

<td>
<a href="eliminar?id=<%=x.getIdarticulo()%>" onclick="return eliminar();">
</a>
</td>

</tr>
<%}%>
</table>
</div>
</html>
```



Articulo	Descripcion	Precio		
Zapatilla Nike	Zapatilla de cuero, color negro	65.0		
Zapatilla Nike	Zapatilla de cuero, color negro	65.0		
Sudadera Adidas	Diveros colores	150.0		
Buzo Nike	Buzo color negro con arreglo de color naranja	280.0		

Archivos formarticulo.jsp de la carpeta articulo:

```
<%@page import="app.model.Categoria"%>
<%@page import="java.util.List"%>
<%@page import="app.model.Articulo"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
<%
//tanto las categorias como los articulos se utiliza para cargar los datos de la categoria
//en el <select name="cat"></select>
List<Categoria> categorias = (List<Categoria>) request.getAttribute("categorias");
Articulo articulo = (Articulo) request.getAttribute("articulo");
String cat="";
%>
</head>
<body>
<h1>Articulo</h1>
<form action="grabar" method="post">
<!-- lenguaje de expresiones (EL) para facilitar el tratamiento de información -->
<input type="hidden" value="${articulo.idarticulo}" name="txtid">
<table>
<tr>
<td>Nombre </td>
<td><input type="text" name="txtnom" id="txtnom" value="${articulo.nombre}" >
</td>
</tr>
</tr>
<tr>
<td>Categoria</td>
<td>
<select name="cat">
<%
for (Categoria x : categorias) {
if(x.getIdcategoria()==articulo.getIdcategoria()){
cat=x.getCategoria();
continue;
}
%>
<option value="<%= x.getIdcategoria()%>"> <%=
x.getCategoria()%></option>

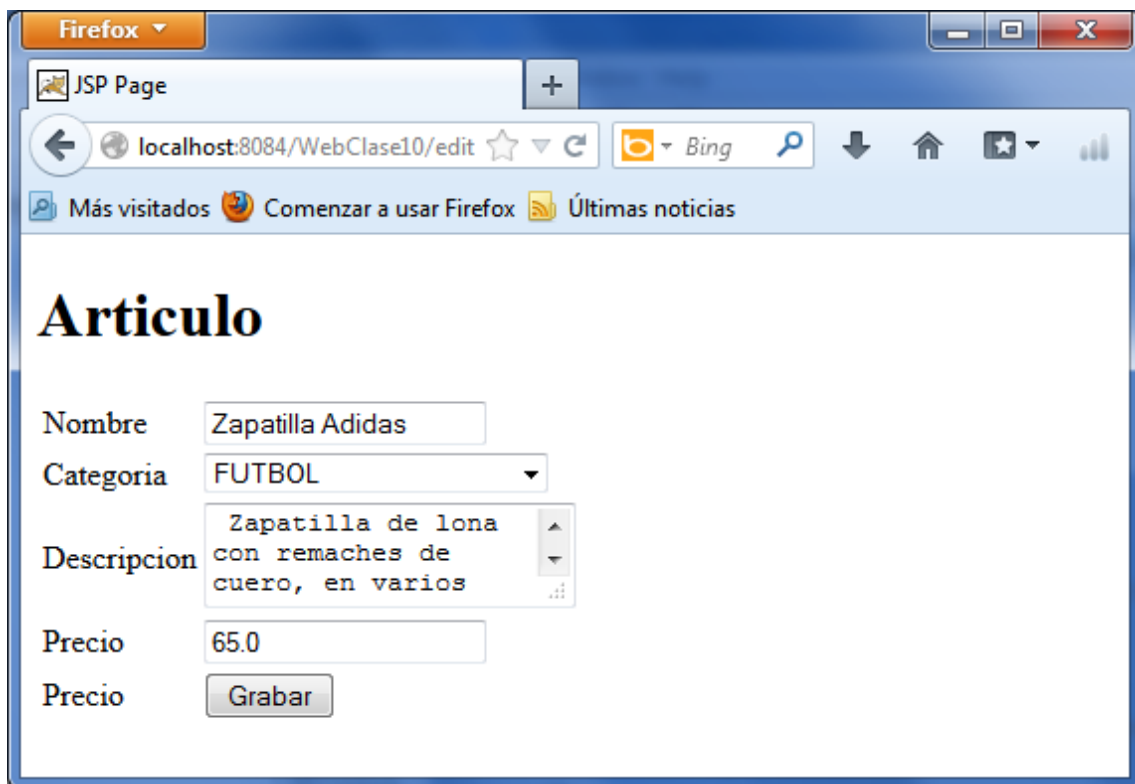
<% }%>

<% if(articulo.getIdcategoria()>0){%>
<!-- selecciona la categoria del articulo a editar-->
<option value="${articulo.idcategoria}" selected=""><%=cat%></option>
<% }else{%>
<!-- selecciona opcion [seleccione]para ingresar nuevo articulo-->
<option value="-1" selected="">[Seleccione]</option>
<% } %>
```

```
</select>

</td>
</tr>
<tr>
  <td>Descripcion</td>
  <td><textarea name="txtdesc"> ${articulo.descripcion}</textarea> </td>
</tr>
<tr>
  <td>Precio </td>
  <td> <input type="text" name="txtpre" id="txtpre" value="${articulo.precio}"> </td>
</tr>
<tr>
  <td>Precio </td>
  <td> <input type="submit" value="Grabar" ></td>
</tr>
</table>
</form>
</body>
</html>
```

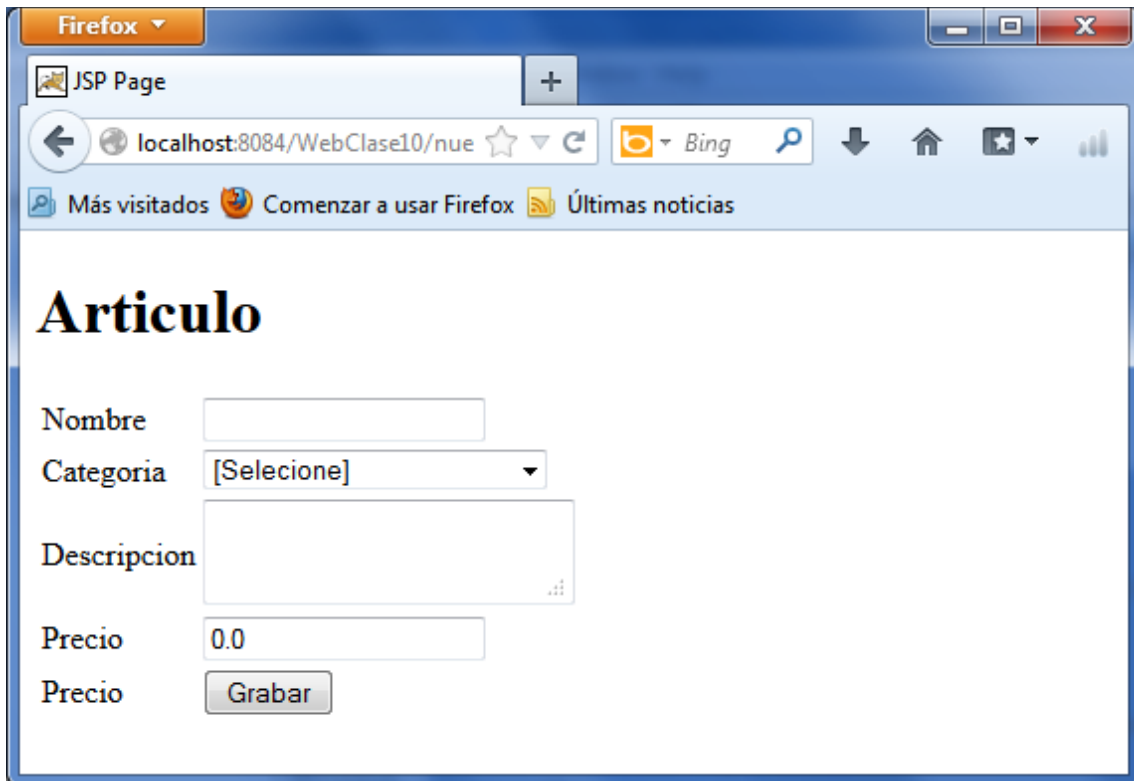
Vista para editar artículo:



The screenshot shows a Firefox browser window with the address bar displaying 'localhost:8084/WebClase10/edit'. The page title is 'JSP Page'. The main content area is titled 'Articulo' and contains a form with the following fields:

- Nombre:** Zapatilla Adidas
- Categoria:** FUTBOL (dropdown menu)
- Descripcion:** Zapatilla de lona con remaches de cuero, en varios (text area)
- Precio:** 65.0
- Precio:** Grabar (button)

Vista para ingresar nuevo artículo:



Firefox

JSP Page

localhost:8084/WebClase10/nue

Bing

Más visitados Comenzar a usar Firefox Últimas noticias

Articulo

Nombre

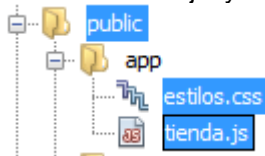
Categoria [Seleccione] ▼

Descripcion

Precio 0.0

Precio

Archivos tienda.js yestilos.css



Tienda.js (utilizado para confirmar cuando se desea eliminar artículo):

```
function eliminar(){
    if(confirm("Confirma eliminar articulo?")){
        return true;
    }
    return false;
}
```

estilos.css :

```
.login
{
  width: 20%;
  margin-left: 40%;
  margin-top: 150px;
}
.txtuser,.txtpass{
  width: 90%;
  border: 1px solid #D8EEFE;
}

#btn-aceptar,#btn-cancelar{
  background-color: #4682B4;
  border: 1px solid #4682B4;
  color: #D8EEFE;
  font-family: "Arial";
}

#botones-login{

  text-align: center;
}

#mensaje-login-error{
  color: red;
  font-family: "Arial";
  font-size: 8px;
}

/*tabla articulos*/

.tabla-articulos{
  width: 100%;
  border: 1px solid #000;
}

.tabla-articulos td{
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-collapse: collapse;
  padding: 0.3em;
  caption-side: bottom;
}
```