

SPEED PROJECT

Realizado por:
Antonio Calzado
Javier Gomez
Alberto Adame

Curso: 2º DAW

INDICE

- Script creación de tablas
- Campos que se reservan para los datos
- Clases y métodos Java(Comentados)
- Enlace de github con video y todos los demás campos

CREACION DE TABLAS

Este sería nuestro script para la creación de las tablas en nuestra base de datos, dicho script con todos los datos ya listos para insertarlos en la tabla y comenzar a trabajar sobre estas.



CAMPOS DE LA TABLA

```
CREATE TABLE USUARIO(  
  USERNAME VARCHAR(50),  
  PASSWORDD VARCHAR(50),  
  CONSTRAINT PK_USUARIO PRIMARY KEY(USERNAME)  
);  
  
CREATE TABLE CATALOGO(  
  ID_CATALOGO NUMERIC(5),  
  NOMBRE VARCHAR(50),  
  DESCRIPCION VARCHAR(150),  
  CONSTRAINT PK_CATALOGO PRIMARY KEY(ID_CATALOGO)  
);  
  
CREATE TABLE ZAPATILLA(  
  ID NUMERIC(5),  
  NAME VARCHAR(50),  
  PRICE FLOAT(5,2),  
  SIZES NUMERIC(2),  
  RELEASEDATE DATE,  
  STOCK BOOLEAN,  
  IDCATALOGO_ZAP NUMERIC(5),  
  CONSTRAINT PK_ZAPATILLA PRIMARY KEY(ID),  
  CONSTRAINT FK_ZAPATILLA FOREIGN KEY (IDCATALOGO_ZAP) REFERENCES CATALOGO(ID_CATALOGO)  
);
```

Como podemos ver la tabla Zapatilla, tenemos un campo numérico entero recogido por el Id de las zapatillas, el nombre es un campo de tipo String, el precio es un campo numérico decimal que recoge 2 decimales nada más, la talla recoge un campo numérico entero, la fecha recoge un campo de tipo date, y el stock recoge el campo boolean para comprobar si hay stock o no. Se incluye una nueva Tabla Catálogo que contendrá las Zapatillas También incluimos la unión entre las dos tablas como son Zapatilla y Catálogo.

CLASES Y METODOS JAVA

```
<%  
String bienvenida="";  
HttpSession sesion=request.getSession();  
String isSesion = (String) sesion.getAttribute("login");  
String userSesion= (String) sesion.getAttribute("usuario");  
if(isSesion != null && userSesion!=null && isSesion.equals("True")){  
    bienvenida=("Sesion: "+userSesion);  
}  
else{
```

```
%> <jsp:forward page="errorPage.html"></jsp:forward> <%  
}  
%>
```

Este método lo utilizamos para comprobar que el usuario existe en la base de datos.

```
<%int idCatalogo=Integer.parseInt(request.getParameter("idCatalogo"));  
Catalogo=CatalogoBD.readCatalogo(idCatalogo);  
Set<Shoes>listaZapatos=Catalogo.getShoesList();  
%>
```

Esto nos permite volver a la pagina anterior cuando entramos en una zapatilla ya que le pasamos el id por parametro, si no esto daría un fallo.

```
<%  
    String stockValor;  
    for(Shoes s:listaZapatos){  
        if (s.isStock()==true){  
            stockValor="En stock";  
        }  
        else{  
            stockValor="No hay stock";  
        }  
    }  
%>  
    <tr>  
        <td>  
            <%=s.getIdShoes() %>  
        </td>  
        <td>  
            <%=s.getName() %>  
        </td>  
        <td>  
            <%=s.getPrice() %>  
        </td>  
        <td>  
            <%=s.getSizes() %>  
        </td>
```

```
 <%=s.getReleaseDate() %>     </td>     <td>         <%=stockValor %>     </td>     <td>         <a href="editarZap.jsp?idZap=<%=s.getIdShoes()%>"></a>         <a href="borrarZap.jsp?idZap=<%=s.getIdShoes()%>"></a><br>     </td> </tr> |
```

Para leer las Zapatillas.

```

<%
    int id=Integer.parseInt(request.getParameter("Id"));

    Boolean entrar=true;
    List<Catalogo> lista=CatalogoBD.loadList();
    for(Catalogo c:lista){
        if(c.getId()==id){
            entrar=false;
        }
    }

    if(entrar==false){
        response.sendRedirect("annadirCatalogo.jsp?error=1");
    }
    else{
        String name=request.getParameter("name");
        String description=request.getParameter("descripcion");

        Catalogo=new Catalogo(id,name,description);
    }
%>

```

```
CatalogoBD.addCatalogo(Catalogo);

String redirect="mainCatalogo.jsp";
response.sendRedirect(redirect);
}
```

%>

Este es el metodo que utilizamos para añadir un Catalogo comprobando que no exista el id.

<%

```
Catalogo=CatalogoBD.readCatalogo(Integer.parseInt(request.getParameter("id")));
CatalogoBD.deleteCatalogo(Catalogo);
```

```
String redirect="mainCatalogo.jsp";
response.sendRedirect(redirect); %>
```

Este es el método que hemos utilizado para borrar un Catalogo, este método recibe un id lo busca en la bd y lo elimina.

<%

```
Catalogo=CatalogoBD.readCatalogo(Integer.valueOf(request.getParameter("id")));
```

```
CatalogoBD.updateUser(Catalogo, request.getParameter("name"),
request.getParameter("descrip"));
String redirect="mainCatalogo.jsp";
response.sendRedirect(redirect);
%>
```

Para editar el catalogo utilizamos este método en el que tambien lo cambiamos a través del id.


```
<%  
  
    String usuarioCadena= request.getParameter("usuario");  
    String password= request.getParameter("password");  
  
    List<Users>listaUsuarios=usuarioBD.loadList();  
    for(Users u: listaUsuarios){  
        if (usuarioCadena.equals(u.getUserName())){  
            usuario=usuarioBD.readUser(usuarioCadena);  
        }  
    }  
  
    if(password.equals(usuario.getPassWord())){  
        System.out.println("Si");  
        HttpSession sesion=request.getSession();  
        sesion.setAttribute("login", "True");  
        sesion.setAttribute("usuario", usuarioCadena);  
        String redirect="mainCatalogo.jsp";  
        response.sendRedirect(redirect);  
    }  
    else{  
        response.sendRedirect("errorPage.html");  
    }  
%>
```

Método para comprobar que el usuario existe dentro de la base de datos.

```
@Entity (name="ZAPATILLA")  
public class Shoes {  
  
    @Id  
    @Column (name="ID")  
    private int idShoes;  
    @Column (name="NAME")  
    private String name;  
    @Column (name="PRICE")  
    private double price;  
    @Column (name="SIZES")
```

```
private int sizes;
@Column (name="RELEASEDATE")
private LocalDate releaseDate;
@Column (name="STOCK")
private boolean stock;
@Column (name="IDCATALOGO_ZAP")
private int idCatalogo;
```

Esta es la clase Zapato la cual mostramos como la hemos mapeado respecto a los campos de la base de datos.

```
public class CRUDShoe {
    private StandardServiceRegistry sr;
    private SessionFactory sf;
    private Session session;

    public CRUDShoe() {
        super();
        sr = new
StandardServiceRegistryBuilder().configure().build();
        sf = new
MetadataSources(sr).buildMetadata().buildSessionFactory();
        session = sf.openSession();
    }
    public Shoes readShoe(int id) {
        Shoes p=null;
        try {
            p= (Shoes) session.get(Shoes.class,id);

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return p;
    }
    public boolean addShoe(Shoes s) {
        boolean resultado=false;
        try {
            session.getTransaction().begin();
            session.saveOrUpdate(s);
            session.getTransaction().commit();
        }
```

```
        resultado=true;

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return resultado;
}

public boolean deleteShoe(Shoes s) {
    boolean resultado=false;
    try {
        Shoes sNew= (Shoes)
session.get(Shoes.class,s.getIdShoes());
        session.getTransaction().begin();
        session.delete(sNew);
        session.getTransaction().commit();
        resultado=true;

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return resultado;
}

public List<Shoes> loadList(){

    List<Shoes> list= new ArrayList<>();
    Query query=session.createQuery("SELECT z FROM
ZAPATILLA z");
    list= query.getResultList();
    return list;

}

public boolean updateShoe(Shoes s, String nombre, double price,
int sizes, LocalDate releasedate, boolean stock) {
    boolean resultado=false;
    try {
        Shoes sNew= (Shoes)
session.get(Shoes.class,s.getIdShoes());
        sNew.setName(nombre);
        sNew.setPrice(price);
```

```
sNew.setSizes(sizes);
sNew.setReleaseDate(releasedate);
sNew.setStock(stock);

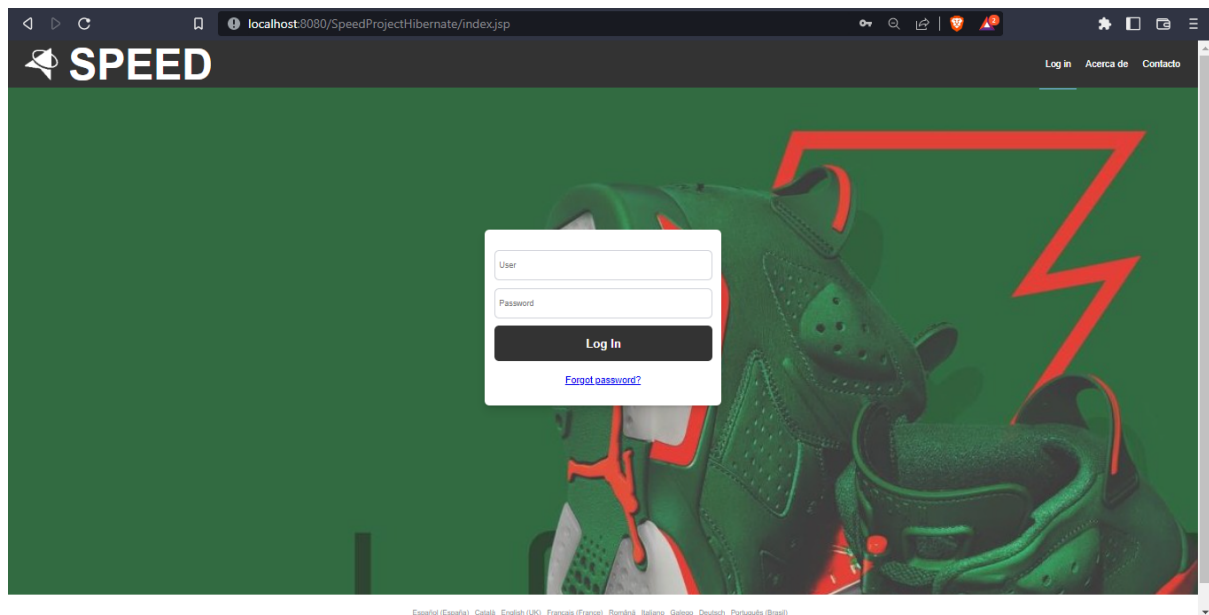
session.getTransaction().begin();
session.saveOrUpdate(sNew);
session.getTransaction().commit();
resultado=true;

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return resultado;
}
}
```

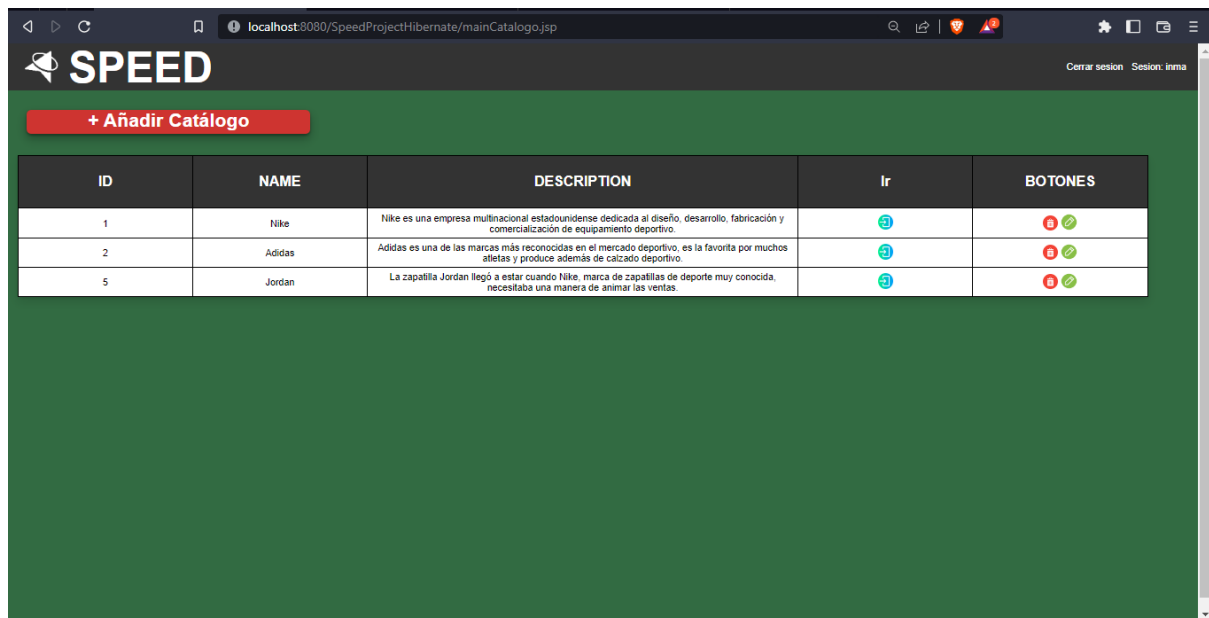
Esto es un ejemplo de la clase CRUD la cual es de zapatilla en la cual encontramos los métodos que necesitamos para añadir, borrar y actualizar esta tabla en la base de datos

Diseño de nuestra pagina

Login:



Main de Catálogos:













































Main de Zapatillas

localhost:8080/SpeedProjectHibernate/main.jsp?idCatalogo=1

SPEED Cerrar sesión Sesión: inma

[+ Añadir Producto](#)

ID	NAME	PRICE	SIZE	RELEASE DATE	STOCK	BOTONES
3	Legaan, water	178.41	49	2014-11-01	En stock	 
36	Macaw, red and blue	173.72	43	2016-09-07	No hay stock	 
7	Bateleur eagle	161.31	39	2016-06-26	En stock	 
72	Magellanic penguin	153.63	46	2001-05-11	No hay stock	 
9	Pronghorn	158.11	43	2022-09-01	No hay stock	 
10	Gray duiker	161.28	48	2018-07-18	No hay stock	 
42	Bulbul, black-eyed	144.05	36	2020-05-12	No hay stock	 
44	Squirrel, thirteen-lined	144.1	36	2005-06-16	No hay stock	 
76	African darter	50.24	48	2022-01-28	No hay stock	 
15	Raven, white-necked	165.31	40	2018-10-26	No hay stock	 
16	Heron, yellow-crowned night	162.61	40	2001-07-01	En stock	 
50	Giant armadillo	100.45	45	2017-11-08	No hay stock	 
82	North American river otter	83.82	38	2003-04-06	En stock	 
19	Wolf spider	140.62	45	2022-04-19	No hay stock	 
83	Wild turkey	98.47	46	2006-11-18	No hay stock	 
53	Porcupine, crested	184.52	46	2012-11-20	No hay stock	 
89	Polar bear	103.03	45	2000-12-02	No hay stock	 
90	Giant otter	155.62	44	2010-05-04	No hay stock	 
59	Owl, snowy	68.15	43	2009-05-28	En stock	 
91	Teal, hottentot	160.99	46	2013-11-10	En stock	 
61	Fox, grey	89.04	35	2017-02-15	En stock	 

Pagina de error:

