

Laboratorio Integrador. Aplicación para validación y registro con funcionalidad de base de datos.

Objetivos.

Conocer como instalar el manejador de bases de datos en un proyecto web

Definir las propiedades del manejador en el archivo de configuración web.xml

Completar la aplicación de validación y registro de usuario para que tenga acceso a base de datos utilizando JDBC

Pasos a seguir

1. Crear una base de datos que almacene la siguiente información sobre un usuario en una tabla:
 - Nombre
 - Apellido Paterno
 - Apellido Materno
 - Teléfono
 - Email
 - RFC
 - Login
 - Password
2. Importar el Laboratorio que contiene la navegación entre páginas completa para el registro y la validación de un usuario
3. Agregar el controlador para el manejo de Bases de Datos
4. Registrar la información para el manejo de Bases de Datos en el archivo web.xml
5. Desarrollar o modificar las clases necesarias

1. Crear una base de datos

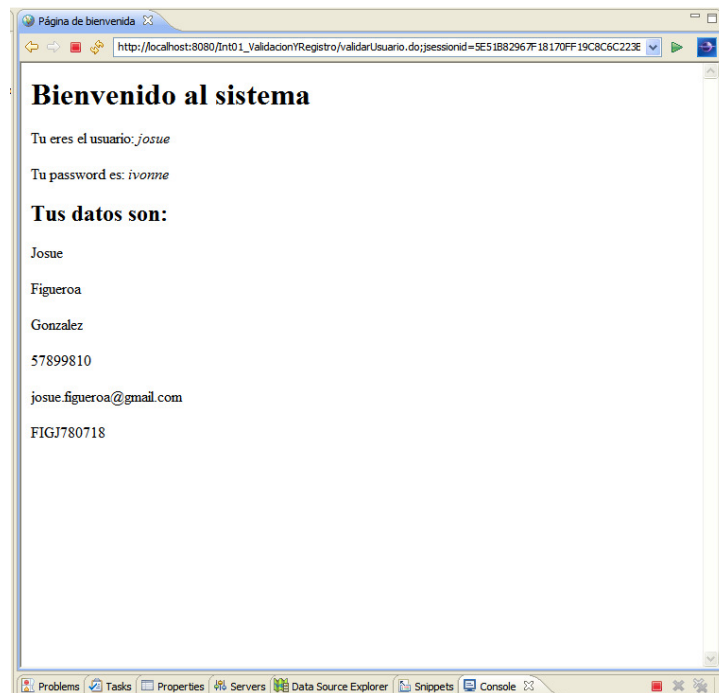
```
CREATE DATABASE integrador_1
```

Crear una tabla para almacenar los datos de un usuario

```
CREATE TABLE datosusuario(  
  nombre varchar(20),  
  apellidoPaterno varchar(20),  
  apellidoMaterno varchar(20),  
  telefono varchar(15),  
  email varchar(20),  
  RFC varchar(15),  
  login varchar(15),  
  passwd varchar(15)  
)
```

2. Importar y probar la aplicación completa de validación y registro





Página de registro

http://localhost:8080/Int01_Validacion/Registro/paginas/nuevoRegistro.jsp;jsessionid=SE51B82967F18170FF19C8C6C223E

Por favor introduce tus datos

Nombre:

Apellido paterno:

Apellido materno:

Teléfono:

e-mail:

RFC:

Login:

Password:

Problems Tasks Properties Servers Data Source Explorer Snippets Console

3. Agregar el controlador (Driver) para el manejo de Base de Datos

Contrario a una aplicación Java normal, en una aplicación Web no hay que agregar la JAR del manejador de base de datos para que sea reconocido.

Solución

Copiar el archivo del manejador (en este caso **mysql-connector-java-5.1.7-bin.jar**) a la carpeta **/lib/ext** del directorio en donde se encuentra instalado el JRE que se está utilizando.

4. Registrar la información del Driver en el archivo web.xml

Para que toda la aplicación tenga acceso a la base de datos, es conveniente declarar los datos en el archivo **web.xml** (contrario a declarar la información como se hizo en el Laboratorio de Introducción a MySQL), esto puede hacerse en cualquier lugar del archivo siempre y cuando se encuentre dentro de las llaves de `<web-app>`

Para esto es necesario agregar la siguiente información sobre el driver en el archivo **web.xml**

```
<context-param>
<param-name>driver</param-name>
<param-value>com.mysql.jdbc.Driver</param-value>
</context-param>
```

Aquí el valor entre las etiquetas de `<param-name>` representa un nombre con el que se hará referencia al valor entre `<param-value>`

La información correspondiente a la base de datos también debe agregarse en el archivo **web.xml**

```
<context-param>
<param-name>cadenaCon</param-name>
<param-value>jdbc:mysql://localhost/integrador_1</param-value>
</context-param>
```

Aquí el valor entre las etiquetas de `<param-name>` representa un nombre con el que se hará referencia al valor entre `<param-value>`, en particular el nombre de la Base de Datos con la que se va a trabajar.

De esta manera el contenido del archivo web.xml es el siguiente

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <display-name>validarUsuarioStruts</display-name>

    <servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-
class>
        <init-param>
            <param-name>config</param-name>
            <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
        <init-param>
            <param-name>debug</param-name>
            <param-value>2</param-value>
        </init-param>
        <init-param>
            <param-name>detail</param-name>
            <param-value>2</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file></welcome-file>
    </welcome-file-list>
    <context-param>
        <param-name>driver</param-name>
        <param-value>com.mysql.jdbc.Driver</param-value>
    </context-param>
    <context-param>
        <param-name>cadenaCon</param-name>
        <param-value>jdbc:mysql://localhost/integrador_1</param-value>
    </context-param>
</web-app>
```

5. Desarrollar o modificar las clases necesarias

Crear una clase para el manejo de la conexión a base de datos llamada **ManejoDeBaseDeDatos** en el paquete **ts.struts.modelo**.

Para crear la conexión a base de datos se utiliza la instrucción:

```
try{
Class.forName(driver).newInstance();
conexion=DriverManager.getConnection(informacion, "root", "96301423");
}catch(Exception e){
    e.printStackTrace();
}
```

En donde **driver** e **informacion** son variables que se obtendrán del archivo **web.xml**, se incluye el usuario ("root") y el password ("96301423").

ManejoDeBaseDeDatos.java

```
package ts.struts.modelo;

import java.sql.Connection;
import java.sql.DriverManager;

public class ManejoDeBaseDeDatos {

    private Connection conexion=null;
    private String driver;
    private String informacion;

    public ManejoDeBaseDeDatos(String d, String info) {
        this.driver=d;
        this.informacion=info;
    }

    public Connection abrirConexion(){

        try{
            Class.forName(driver).newInstance();
            conexion=DriverManager.getConnection(informacion,
"root", "96301423");
        }catch(Exception e){
            e.printStackTrace();
        }

        return conexion;
    }

    public void cerrarConexion() {
        try {
            conexion.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

Obteniendo información para la conexión

Como la información para la conexión se encuentra en el archivo **web.xml**, es necesario acceder a ella a través de una clase que permita conocer la información del archivo, en este caso la clase **ValidarAction**

Esto se realiza con la instrucción:

```
String  
driver=this.getServlet().getServletContext().getInitParameter("driver");
```

En donde “**driver**” es el nombre con el que se identifico al parámetro que contiene la información del controlador

De manera similar para la información de la base de datos

```
String  
informacion=this.getServlet().getServletContext().getInitParameter("cadenaCon");
```

Como el encargado de crear la conexión es el método validar() de la clase ValidarUsuario, es necesario enviarle la información necesaria:

- El objeto de tipo **ValidarUsuarioForm**
- La cadena con información del driver
- La cadena con información de la base de datos

ValidarAction.java

```
package ts.struts.servlets;  
import javax.servlet.http.*;  
import org.apache.struts.action.*;  
  
import ts.struts.DTO.PersonaDTO;  
import ts.struts.javabeans.*;  
import ts.struts.modelo.*;  
  
public class ValidarAction extends Action{  
  
public ActionForward execute(ActionMapping mapping,  
                             ActionForm form,  
                             HttpServletRequest request,  
                             HttpServletResponse response) throws Exception{  
  
    ValidarUsuario v = new ValidarUsuario();  
    ValidacionUsuarioForm vf = (ValidacionUsuarioForm)form;  
    PersonaDTO r;  
  
    String  
    driver=this.getServlet().getServletContext().getInitParameter("driver");  
  
    String  
    informacion=this.getServlet().getServletContext().getInitParameter("cadenaCon");
```

```

        r=v.validar(vf,driver,informacion);

        if(r!=null){
            request.setAttribute("usuario",r);
            return mapping.findForward("bienvenida");
        }
        else
            return mapping.findForward("errorValidacion");

    }

}

```

Validación de un usuario

Es necesario modificar la clase **ValidarUsuario** en el paquete **ts.struts.modelo**

```

package ts.struts.modelo;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

import ts.struts.javabeans.ValidacionUsuarioForm;
import ts.struts.DTO.*;

public class ValidarUsuario {

    public PersonaDTO validar(ValidacionUsuarioForm vf, String d, String i){

        PersonaDTO registro = new PersonaDTO();
        ManejoDeBaseDeDatos base = new ManejoDeBaseDeDatos(d,i);

        Connection con=base.abrirConexion();
        Statement s;

        try{
            s=con.createStatement();
            ResultSet rs = s.executeQuery("SELECT * FROM datosusuario WHERE login='" + vf.getUsuario() + "'");

            if(!rs.first()){
                registro=null;
            }
            else{
                rs.beforeFirst();
                String pb = rs.getString("passwd");

                if(pb.compareTo(vf.getPassword())==0){
                    registro.setNombre(rs.getString("nombre"));
                    registro.setApellidoPaterno(rs.getNString("apellidoPaterno"));
                    registro.setApellidoMaterno("apellidoMaterno");
                    registro.setEmail("email");
                    registro.setTelefono("telefono");
                    registro.setRFC("RFC");
                    registro.setLogin("login");
                }
            }
        }
    }
}

```



```

        registro.setPassword("passwd");
    }
    else
        registro=null;
    }
    base.cerrarConexion();
    return registro;
}
} catch (Exception e) {
    e.printStackTrace();
    return null;
}
}
}

```

Registro de un nuevo usuario

Los datos sobre el Driver y la base de datos también deben ser obtenidos para registrar a un nuevo usuario, en este caso se deben agregar las mismas líneas para su obtención a la clase **RegistrarUsuarioAction** en el paquete **ts.struts.servlets**

RegistrarUsuarioAction.java

```

package ts.struts.servlets;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.*;

import ts.struts.javabeans.RegistroUsuarioForm;
import ts.struts.modelo.RegistrarUsuario;

public class RegistrarUsuarioAction extends Action{

    public ActionForward execute(ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception{

        RegistroUsuarioForm rf = (RegistroUsuarioForm)form;
        RegistrarUsuario registro = new RegistrarUsuario();

        String driver=this.getServlet().getServletContext().getInitParameter("driver");
        String
        informacion=this.getServlet().getServletContext().getInitParameter("cadenaCon");

        registro.registraUsuario(rf,driver,informacion);

        return mapping.findForward("registroCompleto");
    }
}

```

La clase RegistrarUsuario del paquete **ts.struts.modelo** debe ser modificada para que inserte un elemento en la base de datos.

RegistrarUsuario.java

```
package ts.struts.modelo;

import java.sql.Connection;
import java.sql.Statement;

import ts.struts.DTO.PersonaDTO;
import ts.struts.javabeans.RegistroUsuarioForm;

public class RegistrarUsuario {

    public void registraUsuario(RegistroUsuarioForm rf, String d,
                                String i){

        ManejoDeBaseDeDatos base = new ManejoDeBaseDeDatos(d,i);
        Connection con=base.abrirConexion();
        Statement s;
        PersonaDTO persona=convierte(rf);

        try{
            s=con.createStatement();
            String parametro = construirParametros(persona);
            s.executeUpdate("INSERT INTO datosusuario" +
                "(apellidoPaterno,apellidoMaterno,nombre," +
                "email,telefono,RFC,login,passwd)" + parametro);
            s.close();
            base.cerrarConexion();
        } catch (Exception el) {
            el.printStackTrace();
        }

    }

    public PersonaDTO convierte(RegistroUsuarioForm rf){

        PersonaDTO persona=new PersonaDTO();

        persona.setApellidoMaterno(rf.getApellidoMaterno());
        persona.setApellidoPaterno(rf.getApellidoPaterno());
        persona.setEmail(rf.getEmail());
        persona.setLogin(rf.getLogin());
        persona.setNombre(rf.getNombre());
        persona.setPassword(rf.getPassword());
        persona.setRFC(rf.getTelefono());
        persona.setTelefono(rf.getTelefono());

        return persona;

    }

}
```

```
public String construirParametros(PersonaDTO p){  
    String parametro="";  
    parametro = parametro.concat(" VALUES (" +  
        "'" + p.getApellidoPaterno() + "', " +  
        "'" + p.getApellidoMaterno() + "', " +  
        "'" + p.getNombre() + "', " +  
        "'" + p.getEmail() + "', " +  
        "'" + p.getTelefono() + "', " +  
        "'" + p.getRFC() + "', " +  
        "'" + p.getLogin() + "', " +  
        "'" + p.getPassword() + "'" );  
    return parametro;  
}
```

```
}
```