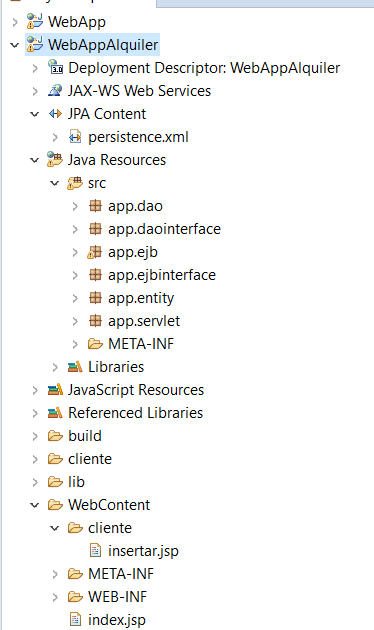
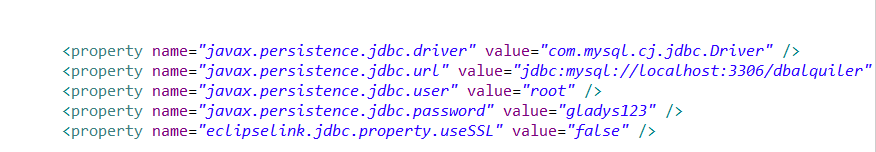
En este ejemplo se integra en una sola aplicación la conexión a los datos con JPA a través de un DAO, el uso de un EJB para la lógica del negocio y la interfaz a través de una página jsp y Servlet

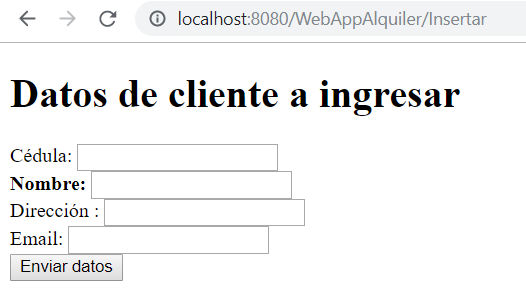
Abra en eclipse el proyecto WebAppAlquiler. El proyecto tiene la siguiente estructura



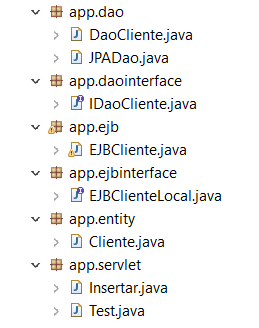
Cambie los datos de conexión del archivo persistence.xml



Ejecute la aplicación en el servidor de aplicaciones y llame al servlet Insertar. Debe visualizar un formulario de ingreso de datos como el siguiente.



Al dar click en Enviar datos la aplicación procesa a través de un EJB la persistencia de la información ingresada a la base de datos.

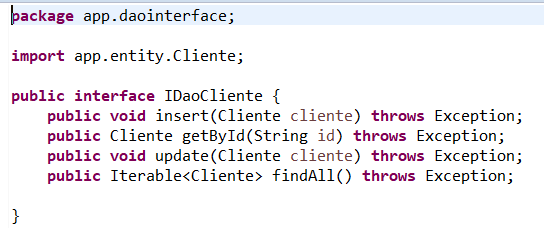


**Capa de datos**

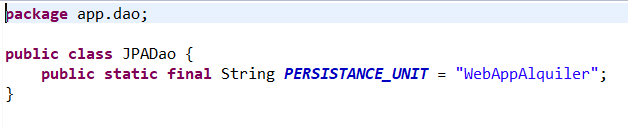
En el proyecto existe una entidad que modela la tabla cliente de la base de datos. Esta entidad permitirá manejar objetos Cliente.

Se ha creado también una clase para objetos DAO. Un objeto de acceso a datos (en inglés, data access object, abreviado DAO) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo. Esto permitirá que nuestra lógica de negocio no necesite saber cuál es la implementación de persistencia que se está manejando.

En la interfaz IDaoCliente se definen los métodos que se exponen.



Se ha creado también una clase para almacenar el nombre de la unidad de persistencia

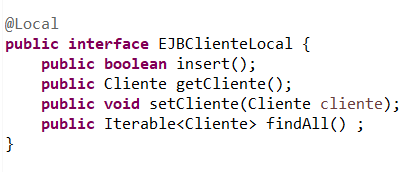


La clase DaoCliente implementa la interfaz escribiendo en los métodos el código correspondiente de acuerdo a la implementación de persistencia. En este caso con JPA.

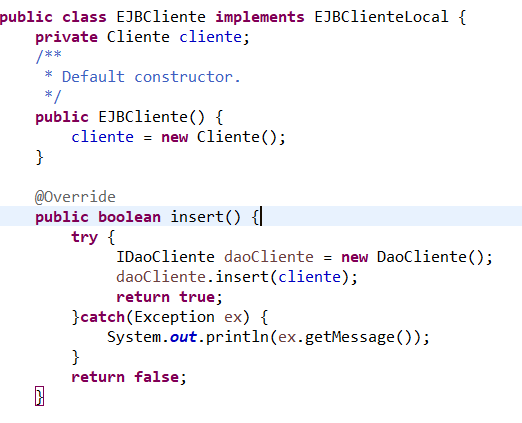


**Capa de negocios**

Un objeto de DaoCliente es el que utilizará el ejb para conectarse a los datos. El objeto EJB local provee métodos de manera similar a los indicados en el DAO



En la clase EJBCliente, por ejemplo, el método insert crea un objeto DaoCliente y llama al método insert con un objeto cliente.



**Capa Web**

El formulario que se muestra es una página jsp



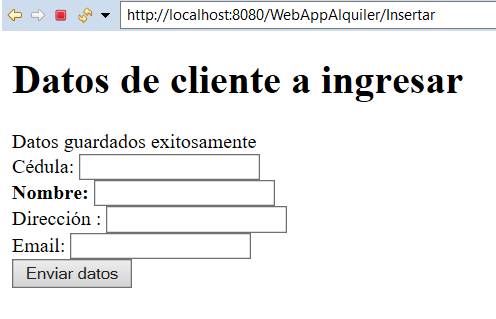
La acción del formulario la maneja el servletInsertar. Este servlet tiene un atributo ejb



En el método POST se recuperan los datos del formulario y se los asigna al objeto cliente del EJB y se llama al método insert



El resultado de la operación se enviará a la página jsp mediante un mensaje para indicar el éxito o fallo de la operación.



**Mostrando lista de clientes de la base de datos**

Ahora completaremos el findAll de clase ClienteDao. En ese método copiar el siguiente código

EntityTransaction et = em.getTransaction();

try {

et.begin();

TypedQuery<Cliente> query = em.createNamedQuery("Cliente.findAll",Cliente.class);

List<Cliente> lista = query.getResultList();

et.commit();

return lista;

} catch (Exception e) {

System.out.println("Error:" + e.getMessage());

throw e;

}finally {

if (em!=null) {

em.close();

em=null;

}

if (emf!=null) {

emf.close();

emf=null;

}

}

Y en la clase EJBCliente, la llamada a este método

@Override

public Iterable<Cliente> findAll() {

try {

IDaoCliente daoCliente = new DaoCliente();

return daoCliente.findAll();

}catch(Exception ex) {

System.out.println(ex.getMessage());

}

return null;

}

Crearemos un nuevo jsp para mostrar la información de los clientes

En la carpeta cliente del WebContent crear un nuevo jsp con nombre listar. Reemplazar con el siguiente contenido

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Listado de Clientes</title>

</head>

<body>

<h3>Lista de Clientes</h3>

<hr size="4" color="gray"/>

<table>

<tr>

<th>Cedula</th>

<th>Nombre</th>

<th>Direccion</th>

<th>Email</th>

</tr>

<%

java.util.List<app.entity.Cliente> clientes=(java.util.List<app.entity.Cliente>)request.getAttribute("clientes");

for (app.entity.Cliente cliente: clientes) {

%>

<tr>

<td><%=cliente.getCedula() %></td>

<td><%=cliente.getNombre() %></td>

<td><%=cliente.getDireccion() %></td>

<td><%=cliente.getEmail() %></td>

</tr>

<%}%>

</table>

</body>

</body>

</html>

En este jsp se usa un atributo clientes, que es la lista que se muestra. Este atributo lo enviará el servlet.

En el paquete app.servlet crear un nuevo Servlet llamado Listar

import app.ejb.EJBCliente;

import app.ejbinterface.EJBClienteLocal;

import app.entity.Cliente;

/\*\*

\* Servlet implementation class Listar

\*/

@WebServlet("/Listar")

public class Listar extends HttpServlet {

private static final long serialVersionUID = 1L;

private EJBClienteLocal ejbCliente;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public Listar() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

ejbCliente = new EJBCliente();

List<Cliente> clientes= (List<Cliente> )ejbCliente.findAll();

request.setAttribute("clientes", clientes);

request.getRequestDispatcher("cliente/listar.jsp").forward(request, response);

}

}

En este servlet se obtienen los clientes a través del ejb con el método findAll