# Guia JSP con JPA y EJB por medio de JEE

## Definición de conceptos:

* **JSP:**

JavaServer Pages es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en [HTML](https://es.wikipedia.org/wiki/HTML" \o "HTML) y [XML](https://es.wikipedia.org/wiki/XML" \o "XML), entre otros tipos de documentos. JSP es similar a [PHP](https://es.wikipedia.org/wiki/PHP" \o "PHP), pero usa el lenguaje de programación [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)" \o "Java (lenguaje de programación)).

* **JPA (Hibernate)**

JAP es Java Persistence API (Framework ORM) es la propuesta estándar que ofrece Java para implementar un Framework Object Relational Mapping (ORM), que permite interactuar con la base de datos por medio de objetos, de esta forma, JPA es el encargado de convertir los objetos Java en instrucciones para el Manejador de Base de Datos (MDB).

* **EJB:**

Las Enterprise JavaBeans son una de las [interfaces de programación de aplicaciones](https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones" \o "Interfaz de programación de aplicaciones)(API) que forman parte del estándar de construcción de aplicaciones empresariales [J2EE](https://es.wikipedia.org/wiki/J2EE" \o "J2EE) (ahora JEE) de [Oracle Corporation](https://es.wikipedia.org/wiki/Oracle_Corporation)

* **JEE:**

Java Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)" \o "Java (lenguaje de programación)). Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un [servidor de aplicaciones](https://es.wikipedia.org/wiki/Servidor_de_aplicaciones" \o "Servidor de aplicaciones).

* **JDBC:**

es una [API](https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones" \o "Interfaz de programación de aplicaciones) que permite la ejecución de operaciones sobre [bases de datos](https://es.wikipedia.org/wiki/Base_de_datos" \o "Base de datos) desde el [lenguaje de programación Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java" \o "Lenguaje de programación Java), independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

* **JTA:**

JTA gestiona dos conceptos fundamentales: Uno es el de**Transactional Manager** que define como ha de implementarse un gestor de transacciones para ser compatible con JTA.

* **Servlets:**

es una [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)" \o "Clase (informática)) en el [lenguaje de programación Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java" \o "Lenguaje de programación Java), utilizada para ampliar las capacidades de un servidor. Aunque los servlets pueden responder a cualquier tipo de solicitudes, estos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como [applets de Java](https://es.wikipedia.org/wiki/Applet_Java" \o "Applet Java) que se ejecutan en servidores en vez de navegadores web. Este tipo de servlets son la contraparte Java de otras tecnologías de contenido dinámico Web, como [PHP](https://es.wikipedia.org/wiki/PHP" \o "PHP) y [ASP.NET](https://es.wikipedia.org/wiki/ASP.NET" \o "ASP.NET).

* **Entity class**

las clases especiales llamadas "Entity class" corresponde a tablas en una base de datos. El mapeo entre estas clases y tablas se realiza usando herramientas ORM externas como Hibernate o EclipseLink, estas herramientas proporcionan algunos medios para generar automáticamente las Entity class desde la tabla modelo

* **Persistence unit**

La unidad de persistencia es la responsable de todas las operaciones de la base de datos, en este objeto se generará todo el código SQL que está por debajo.

* **Session Bean**

Un Session Bean es un componente del lado del servidor que proporciona servicios a los clientes. Puede proporcionar los diferentes pasos para el flujo de trabajo de una aplicación:

* Autenticación
* Listado de entidades
* CRUD, etc ...

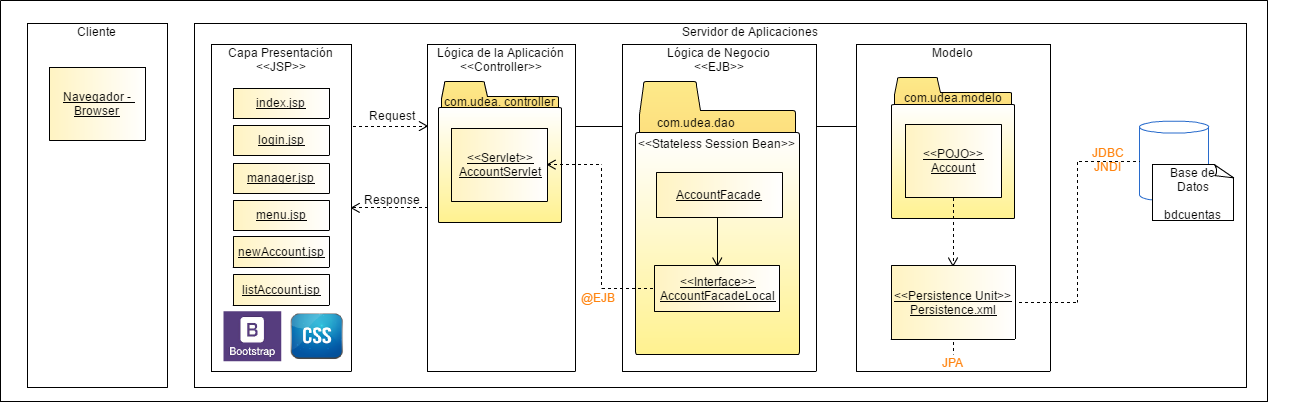
## Funcionalidad de la App

Se creará una simple aplicación de tipo CRUD para la gestión de un concesionario utilizando recursos JDBC, anotaciones JPA, EJB, unidades de persistencia, Servlets y JSP por medio de la plataforma JEE.

## Objetivos específicos

* Comprender el uso de los recursos JDBC y pool de conexiones para el manejo de conexiones y concurrencia en una base de datos.
* Conocer y comprender las anotaciones JPA para simplificar la persistencia y el mapeo de una base de datos objeto-relacional.
* Comprender el uso de EJB-Session Beans: Stateful, Stateless y Singleton para la seguridad de una aplicación transaccional.
* Utilizar Servlets para la lógica del negocio y JSP para la vista del cliente.
* Conocer la utilidad de los objetos request y response que nos proporcionan los Servlets.
* Comprender el manejo de JSP para la creación de una aplicación web.
* Aplicar una arquitectura básica de 3 o N capas para el desarrollo de una aplicación web.
* Aplicar y entender el uso de los patrones de diseño DTO y DAO.

## Arquitectura de la App



## Estructura Front End y Back End de la app

**Backend:** El Backend está compuesta de 3 capas las cuales son:

* Capa Datos: Corresponde al motor de base de datos MySQL, que tendrá la instancia ACCOUNT, previamente creada.

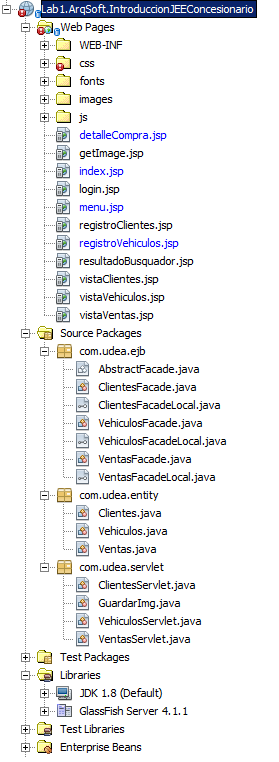
Acá se almacenarán los respectivos datos que lleguen por medio del patrón DAO (Data Access Object), generado en la capa de la lógica del negocio.

* Capa del Modelo: Contiene un Entity Bean que en esencia es un simple POJO (Plain Old Java Object) con anotaciones JPA para facilitar el acceso al modelo del dominio y poder enlazar los datos serializados usando el patrón DTO (Data Transfer Object). Hay que tener en cuenta que al momento de generar el Entiy Bean se realiza un mapeo objeto relacional (ORM), de forma transparente al desarrollador, lo que facilita el acceso a los datos. El otro componente es la llamada Unidad de Persistencia (P.U), que permite a través de la clase Entity Manager, realizar la conexión y transaccionalidad hacia la base de datos usando los API JPA y JTA respectivamente. Esto es importante porque facilita al desarrollador acceder al modelo del dominio a través de métodos predefinidos sin preocuparse de la complejidad que implica emplear llamadas por medio de JDBC.
* Capa de Lógica del Negocio: En esta capa se usará el patrón DAO, por medio de un Stateless Session Bean, el cual es una clase que contendrá la implementación de los principales métodos CRUD que serán expuestos a través de un Facade. La implementación se realizará en una clase genérica llamada Abstract Facade.

**FrontEnd:** Está compuesta de 3 capas las cuales son:

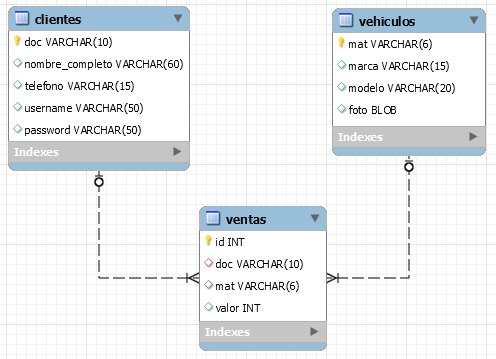
* Capa Cliente: Corresponde a donde estará el web browser elegido por el usuario para interactuar con el sistema.
* Capa Presentación: Corresponde a las diferentes páginas generadas en JSP y usando etiquetas de HTML5 y Twitter Bootstrap que contienen los formularios y las vistas principales del sistema.
* Capa Lógica de la Aplicación: Acá se tendrá la clase Servlet que actuará como un controlador de las vistas y genera una comunicación con los métodos del DAO por medio de la inyección de dependencias usando la anotación @EJB.

# Repositorio en Netbeans



# Pasos de Creación

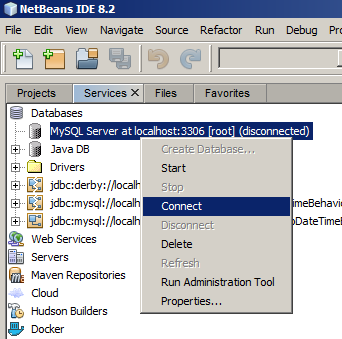
# Crear la base de datos

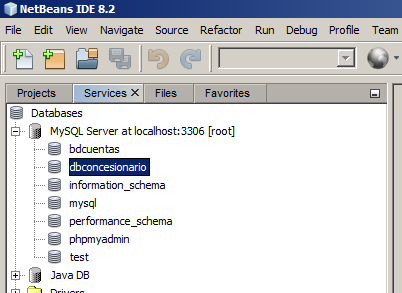


NOTA: El Script SQL está adjunto

# Conectar MySQL con Netbeans

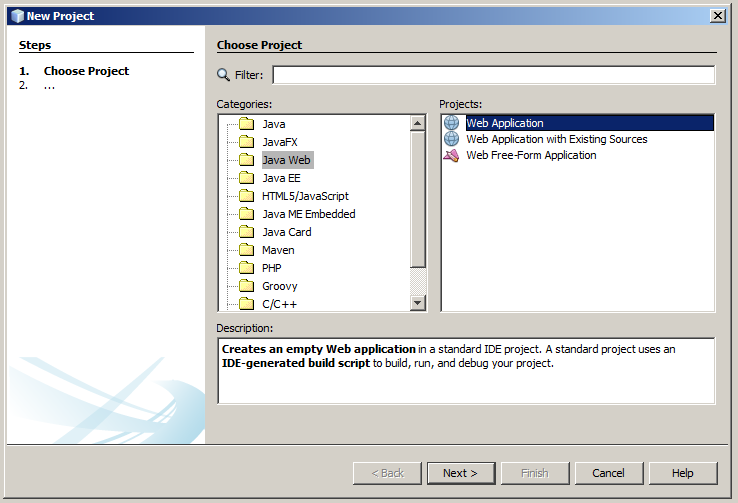
Recuerda tener xampp corriendo y con la base de datos en phpmyadmin

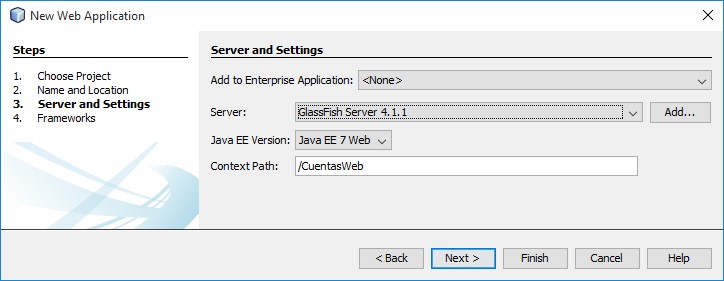




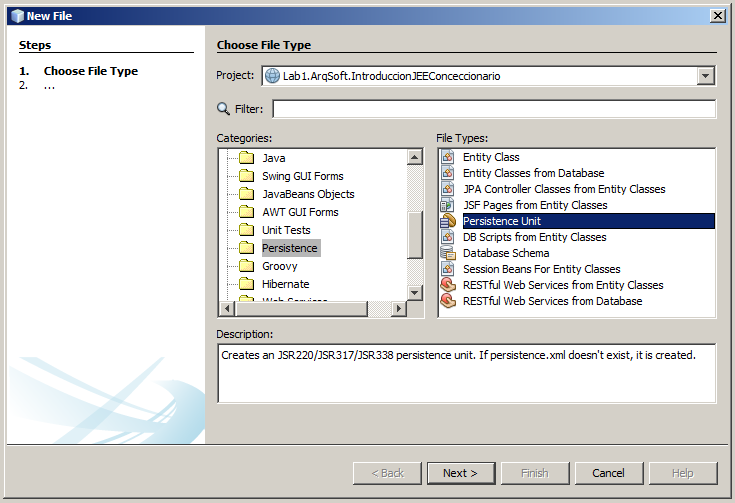
Nota: verificar que Glassfish tenga conexión con MySQL, si no. Se crea la base de datos en Derby

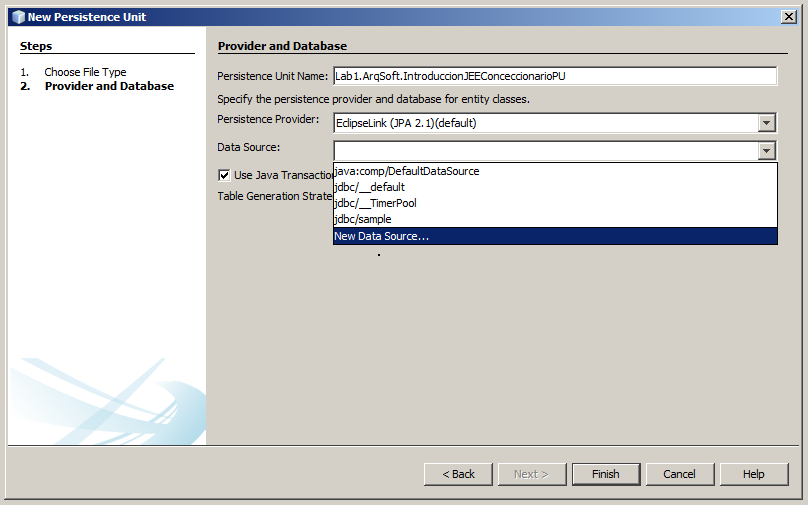
# Crear la aplicación en Netbeans

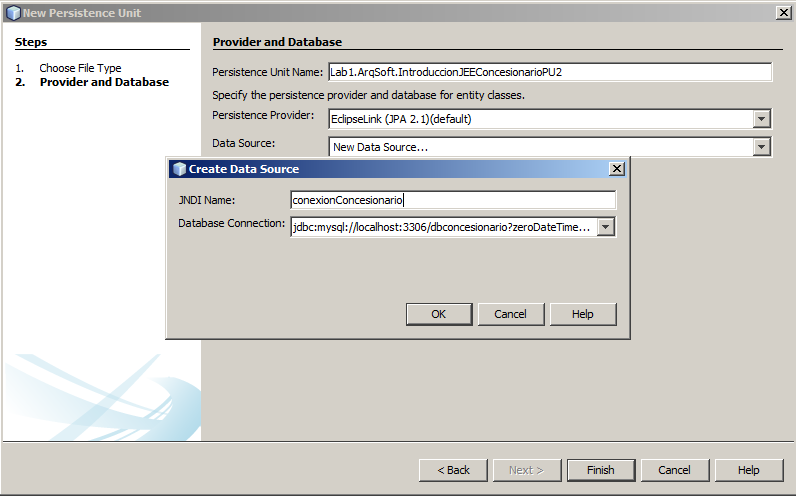




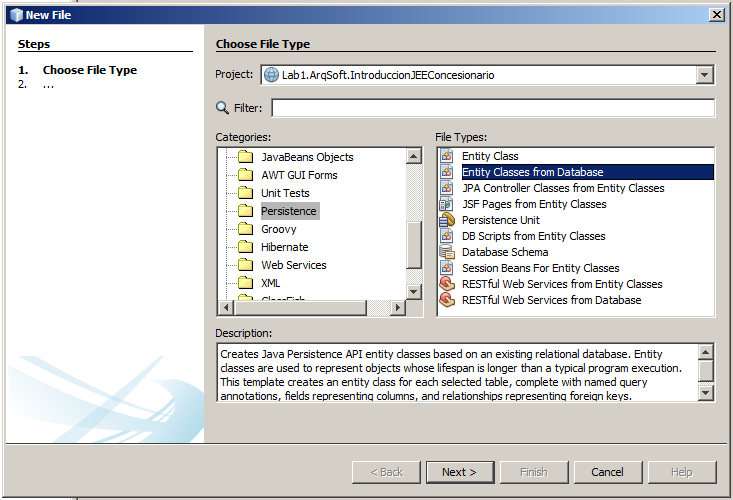
# Crear la unidad de persistencia (JPA) (JDBC)

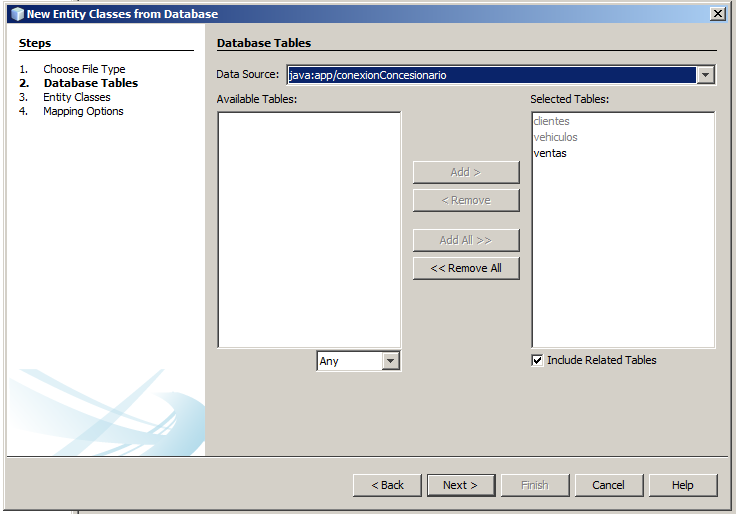


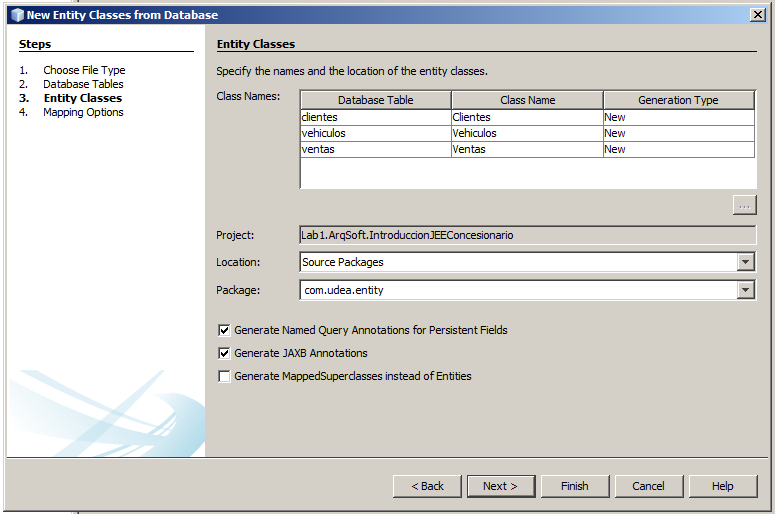


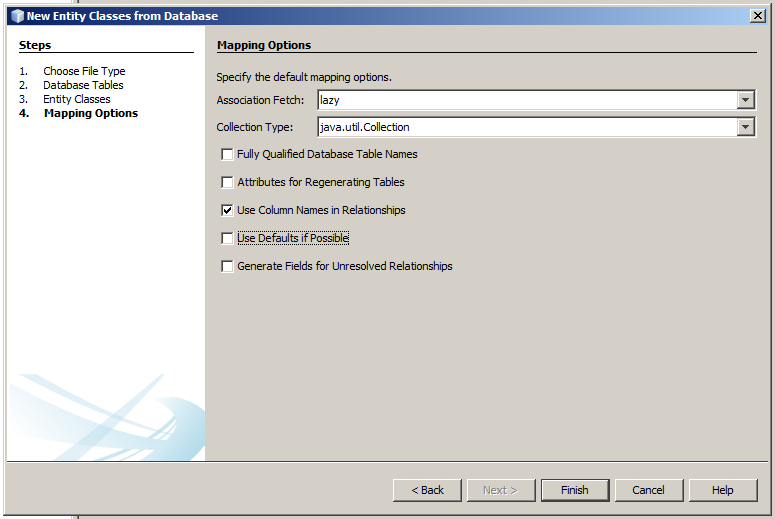


# Crear Entity Bean (Entity Classes From Database) (JPA) (Entidad)

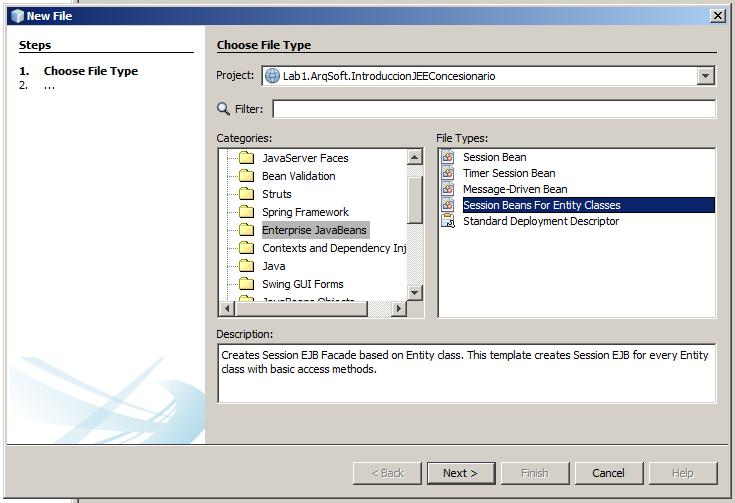




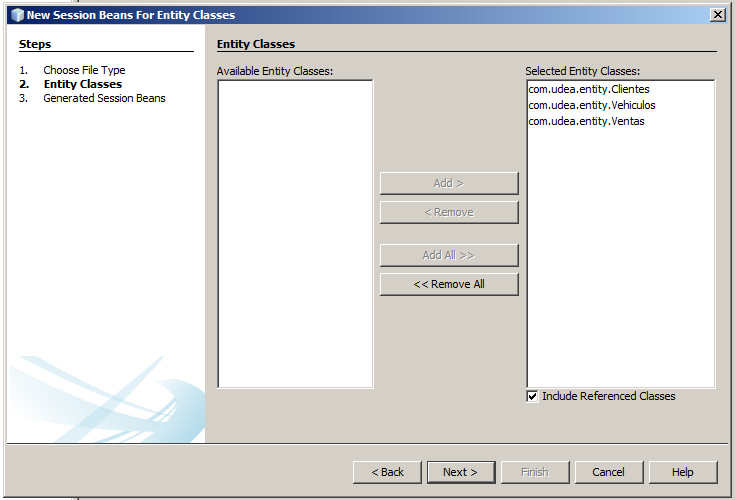


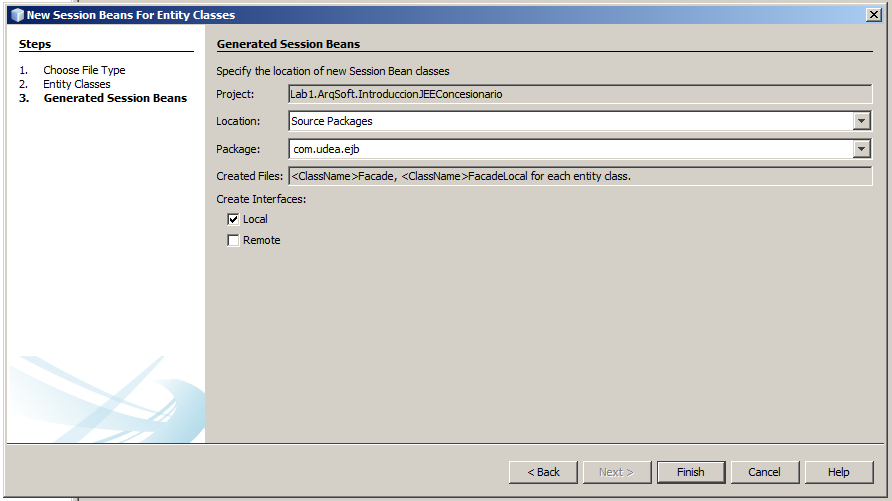


# Agregar EJB de tipo Session Bean (Session Bean From Entity Classes) (JPA) (Modelo)



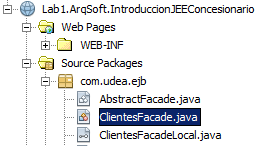
NOTA: Si no aparecen las clases entity, cierra y vuelve a abrir el netbeans



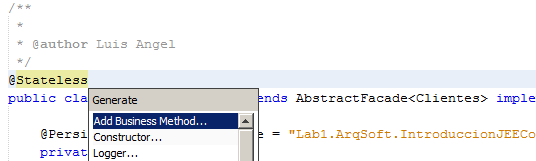


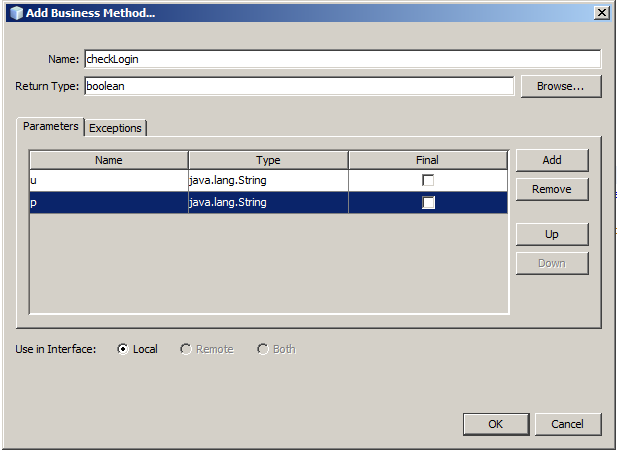
# Desarrollar una consulta con JSTL en una clase EJB

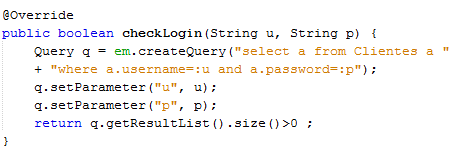
por ejemplo ClientesFacade nos queda de esta manera el EJB:



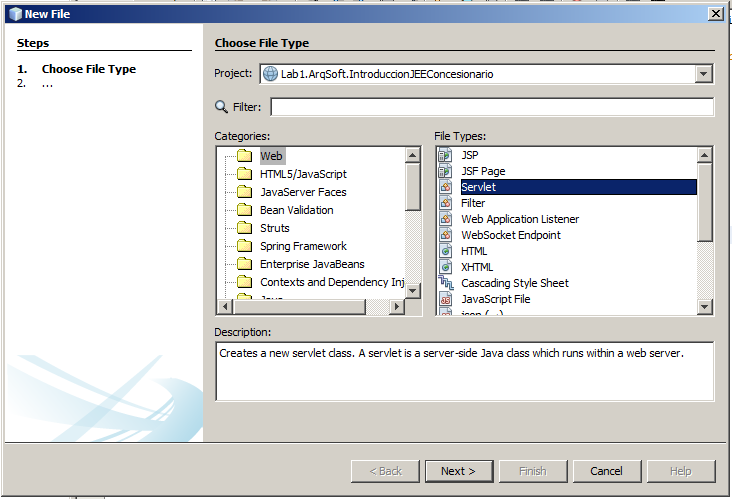
Nos paramos dentro de la clase y le damos clic derecho “insert code”:

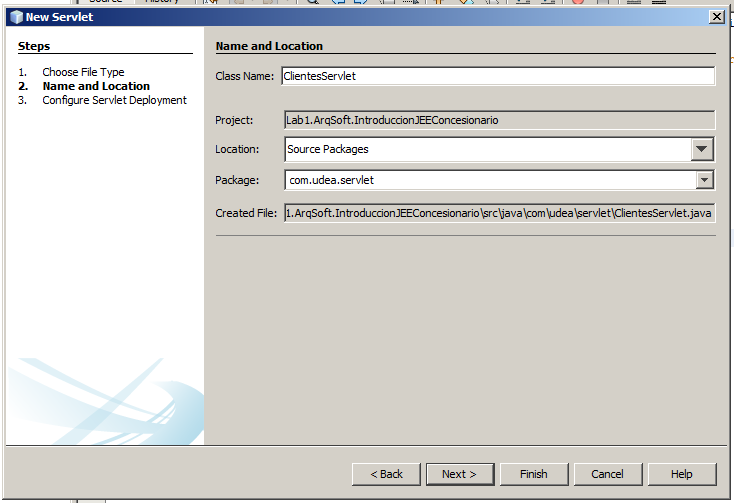


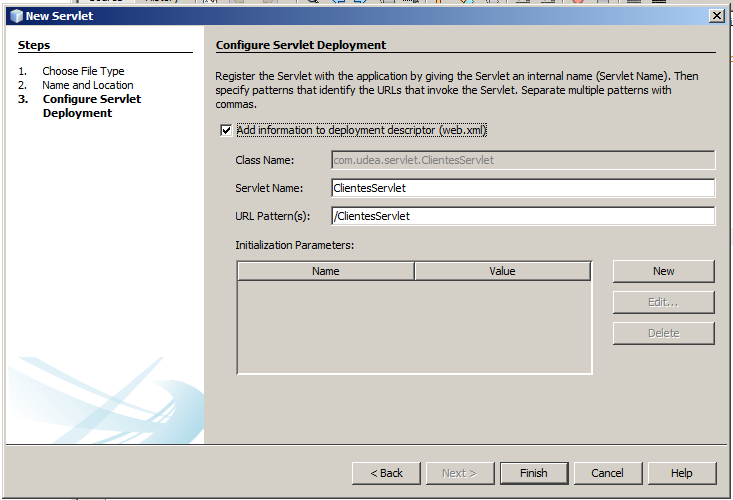




# Crear el Servlet

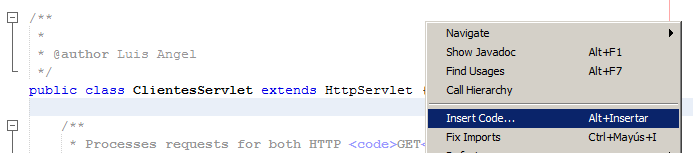


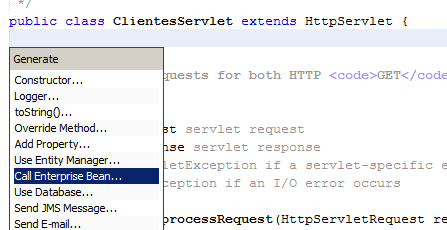


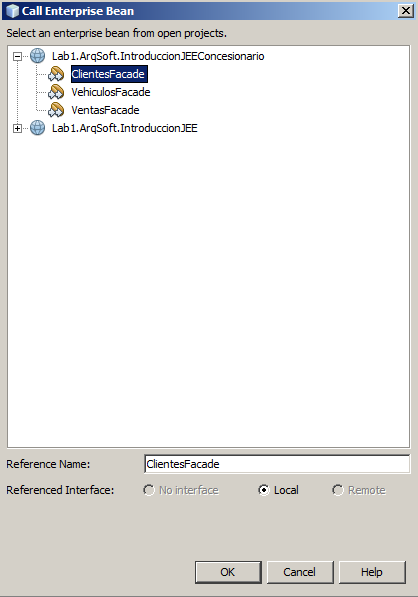


Hacemos una llamada al Enterprise Bean para agregar la interfaz de AccountFacade.

Dentro del servlet damos clic derecho:







Asegurase de que esté referenciado el acceso a los métodos del negocio:



# Desarrollar el método processRequest del servlet

|  |
| --- |
| protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  response.setContentType("text/html;charset=UTF-8");  try (PrintWriter out = response.getWriter()) {  String action = request.getParameter("action");  String url = "index.jsp";  if ("list".equals(action)){  List<Clientes> findAll = clientesFacade.findAll();  request.getSession().setAttribute("clientes", findAll);  url = "listAccounts.jsp";  }else if ("login".equals(action)){  String u = request.getParameter("username");  String p = request.getParameter("password");  boolean checklogin = clientesFacade.checkLogin(u, p);  if(checklogin){  request.getSession().setAttribute("login", u);  url="manager.jsp";  }else{  url = "login.jsp?error=1";  }  }else if("insert".equals(action)){  Clientes c = new Clientes();  c.setUsername(request.getParameter("username"));  c.setPassword(request.getParameter("password"));  c.setDoc(request.getParameter("doc"));  c.setNombreCompleto(request.getParameter("nombre"));  c.setTelefono(request.getParameter("telefono"));  clientesFacade.create(c);  url = "login.jsp";  }else if("delete".equals(action)){  String id = request.getParameter("id");  Clientes c = clientesFacade.find(Integer.valueOf(id));  clientesFacade.remove(c);  url = "ClientesServlet?action=list";    }else if("logout".equals(action)){  request.getSession().removeAttribute("login");  url = "login.jsp";  }  response.sendRedirect(url);  }finally {  out.close();  }  } |

# Crear las Vistas

NOTA: El proyecto se encuentra en GitHub con nombre de Concesionario-JEE-JSP