



# CRUD EJB, JPA, ORACLE 11G + NETBEANS 8.0.2



## **Introducción**

A continuación en este pequeño tutorial se explica cómo podemos crear un CRUD utilizando las tecnologías “JPA y EJB” junto con los framework JSF en Java, haciendo uso de Netbeans como IDE de desarrollo y como gestor de base de datos Oracle 11G. Durante el proceso, se integrará las tecnologías ya mencionadas, además, aprenderemos a crear un pool de conexiones en Glassfish Server y a gestionar dicho pool para poder establecer la conexión con la base de datos de Oracle desde Netbeans.

Es importante contar con conocimientos Básico a Intermedios en el uso de estas herramientas, aunque en el tutorial todo se hace paso a paso; pero se da por entendido que ya tienes nociones básicas de Oracle 11g, servidor de aplicaciones Glassfish y programación en JAVA al menos básico para entender toda la lógica que engloba la construcción de aplicaciones empresariales con JAVA.

## 1. DISEÑO DE LA BASE DE DATOS

- **Crear una nueva conexión en Oracle con los siguientes datos:**

Usuario: neo, password: 12345, rol: connet, privilegios: Todos menos, sysdba, sysoper y Admin resouse manager. (Esto si lo haces desde SQL Developer), sino crear un workspace con el usuario y pass ya mencionado en Oracle11g XE

Nombre conexión: conexionNeo

Usuario: neo

Pass: 12345

SID: orcdb, según instalación de oracle

**Nota:** para realizar este punto es importante tener al menos los conocimientos básicos en Oracle 11g instalación y creación de conexión y usuarios.

- **Crear la tabla alumno con los siguiente campos**

```
create table alumno(  
    idAlumno number primary key,  
    nombres varchar2(50) not null,  
    apellidos varchar2(50)not null,  
    direccion varchar2(100)not null,  
    telefono VARCHAR2(15) not null  
);
```

- **Definir una secuencia para el campo idAlumno**

```
CREATE SEQUENCE incremento_id_alumno  
INCREMENT BY 1  
START WITH 1;
```

- **Crear un Trigger para invocar la secuencia al crear nuevos registros**

**Nota:** Este trigger únicamente nos servirá para insertar registros desde la consola de Oracle 11g. No lo utilizaremos en el desarrollo de la aplicación en netbeans.

```
CREATE TRIGGER TRG_ID_ALUMNO  
BEFORE INSERT ON ALUMNO  
FOR EACH ROW  
BEGIN  
    SELECT INCREMENTO_ID_ALUMNO.NEXTVAL INTO:NEW.IDALUMNO FROM DUAL;  
END;
```

**Nota:** al crear el trigger desmarcar la opción null que aparece en el cuadro de dialogo.

- Insertar los siguientes registros

```
insert into alumno(nombres, apellidos, direccion, telefono)values('JOSE ERNESTO','ZURITA NUÑEZ','AV. LA COLINA 2, CASA#12','7845-9623');
```

```
insert into alumno(nombres, apellidos, direccion, telefono)values('DAVID ANTONIO','LOPEZ CRUZ','CALLE INDEPENDENCIA, CASA#11','7158-6242');
```

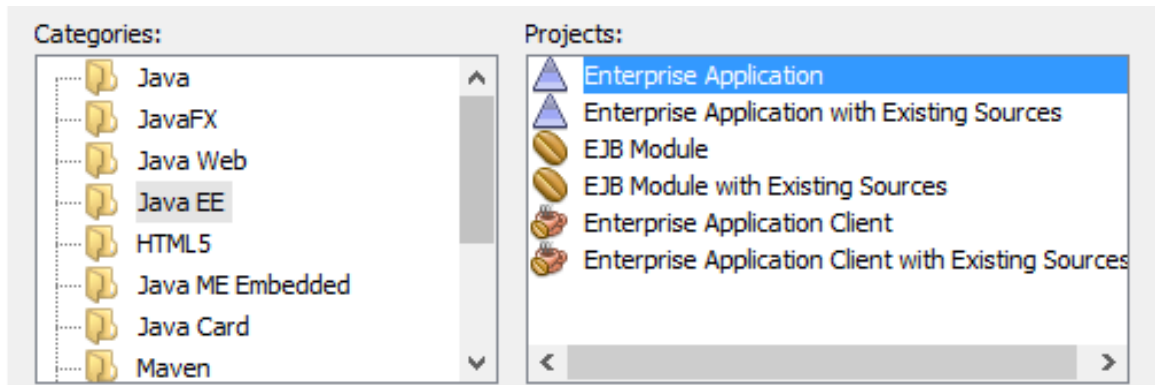
```
insert into alumno(nombres, apellidos, direccion, telefono)values('BLANCA MARGARITA','CARIAS RUIZ','AV. LAS MAGNOLIAS SUR, CASA#8','7348-958');
```

```
insert into alumno(nombres, apellidos, direccion, telefono)values('STAYCU CECILIA','MENDEZ SUEZ','COL. EL ZONTLE, CASA#2','7956-8542');
```

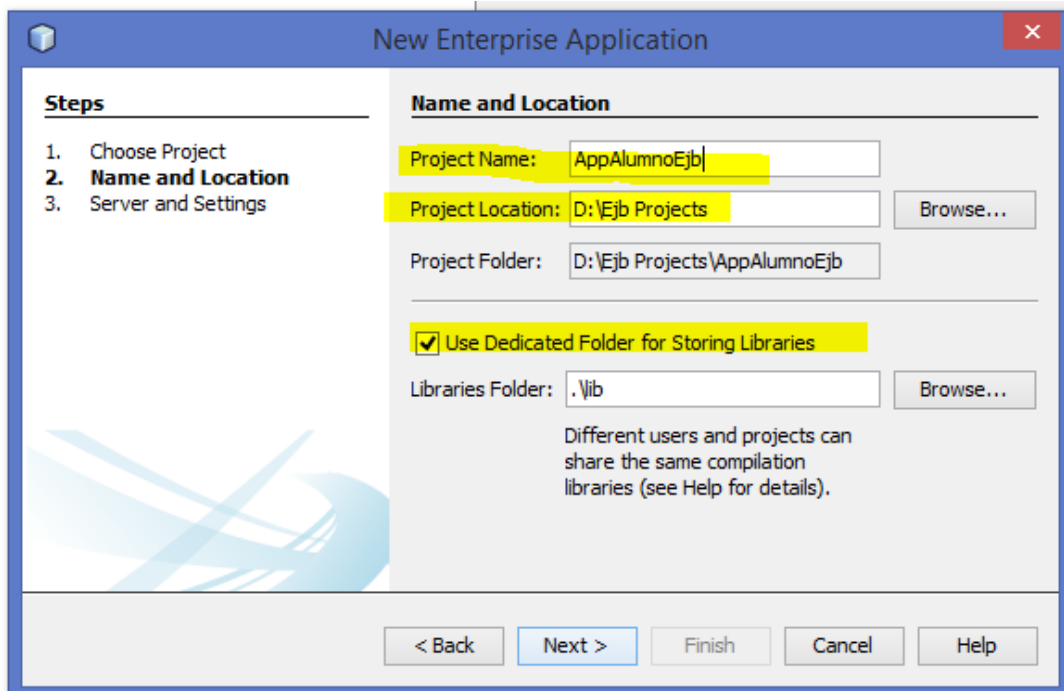
```
insert into alumno(nombres, apellidos, direccion, telefono)values('REBECA ABIGAIL','SERMEÑO AGUIRRE','RES. BOSQUE DE LA PAZ, POL A, CASA#29','7412-5836');
```

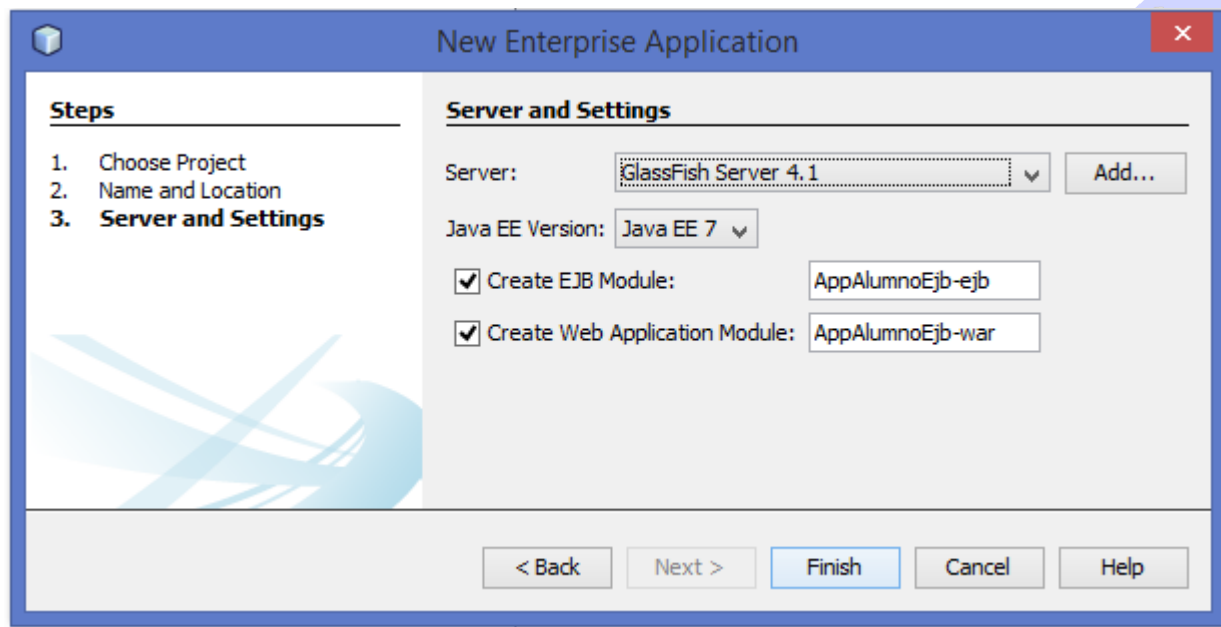
## 2. CREACION DEL PROYECTO USANDO EJB EN NETBEANS

- Crear un nuevo proyecto JavaEE, Enterprise Application.

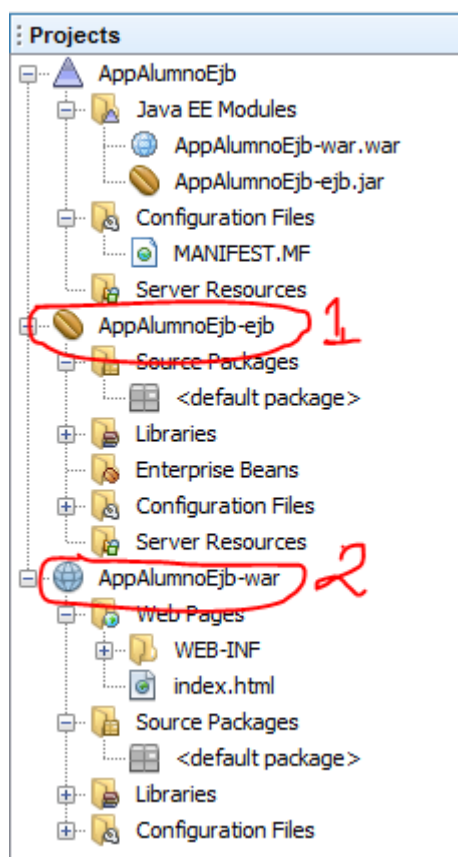


- Configurar según las siguientes pantallas





- Al finalizar la creación del proyecto tendremos esta estructura de directorios y paquetes



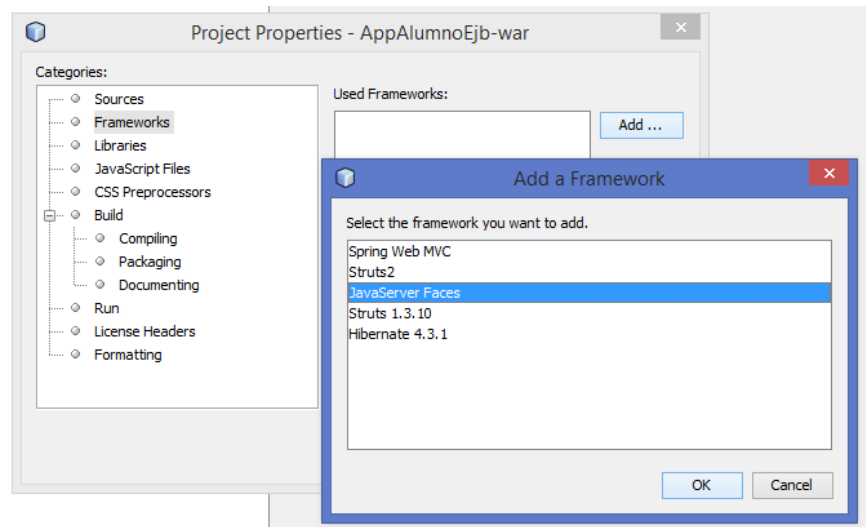
Nuestro trabajo estará centrado en los módulos ejb<sup>1</sup> y war<sup>2</sup>, en el primero es donde tendrá lugar la lógica de negocio y de desarrollo de nuestra aplicación y en la segunda parte todo lo que tiene que ver con las vistas para el usuario final (las paginas JSF). El primer elemento que vemos en forma de triángulo sirve para gestionar ambos módulos ejb y war y solo lo utilizaremos para deployar o ejecutar la aplicación.

### 3. AGREGAR SOPORTE PARA JSF Y PRIMEFACES AL PROYECTO

Java Code  
503

- **Asignando el framework de JSF**

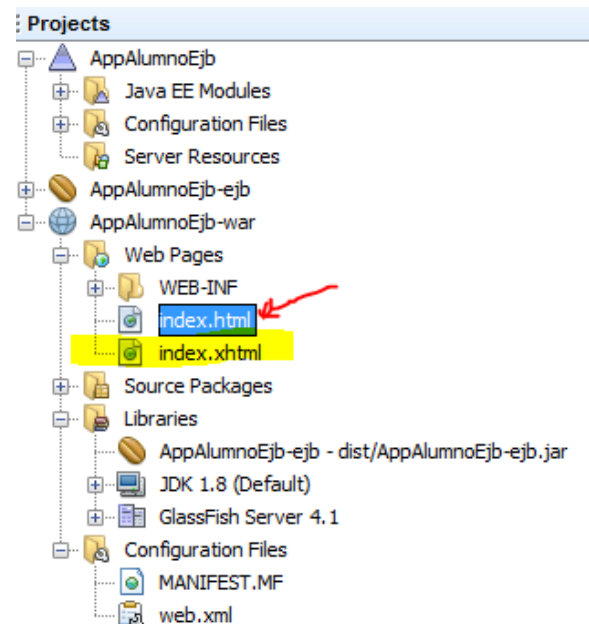
1. Hacer clic derecho sobre el módulo war (El mundo de color azul)
2. Seleccionar propiedades
3. Buscar y seleccionar en categorías: framework
4. Hacer clic al botón Add
5. Seleccionar el framework Java Server Faces
6. Clic en ok dos veces



Como se puede ver se ha creado una página index.xhtml propia de jsf, por lo que se deberá eliminar la otra index.html (clic derecho sobre el nombre y delete) ya que no se utilizara.

- **Agregar biblioteca de primefaces al proyecto**

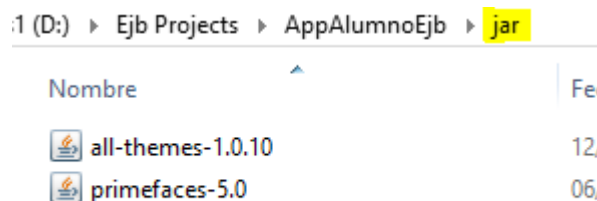
1. Crear dentro de la ubicación del proyecto (Esto hazlo donde está guardado el proyecto, no en netbeans) un folder llamado jar y guarda dentro las bibliotecas de primefaces y all-themes



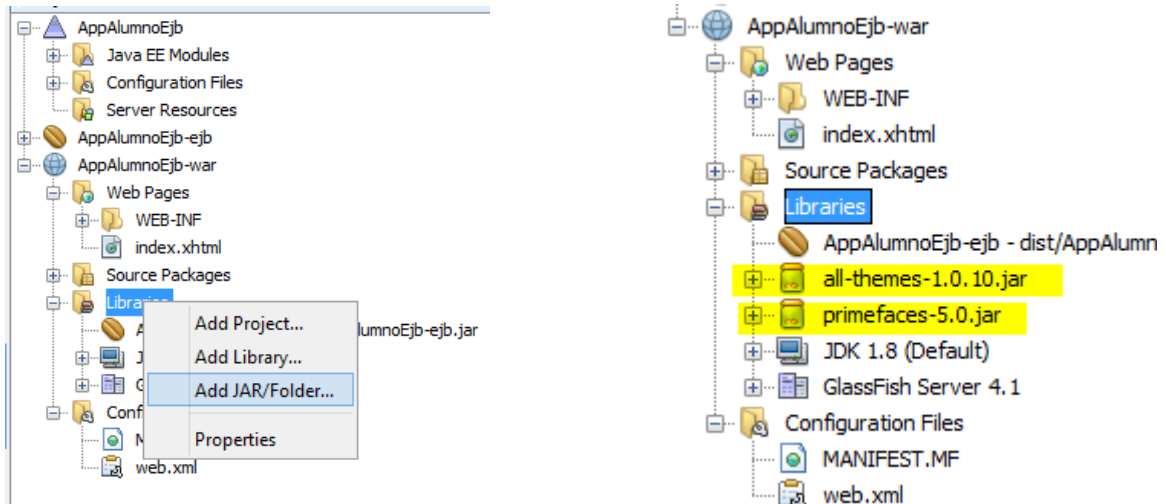
Descargar:

Primefaces 5.0 <http://primefaces.org/downloads>

All.-themes: <http://repository.primefaces.org/org/primefaces/themes/>



2. Siempre en el módulo war, hacer clic derecho sobre Libraries y seleccionar Add JAR/folder... luego buscar en el proyecto el folder jar y seleccionar ambas librerías, así como se aprecia en las imágenes.



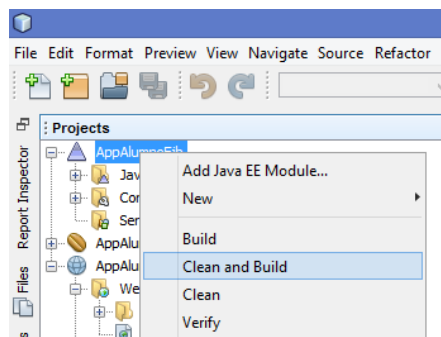
3. Ir a la página index.xhtml y colocar el siguiente código para probar que funciona jsf y primefaces.

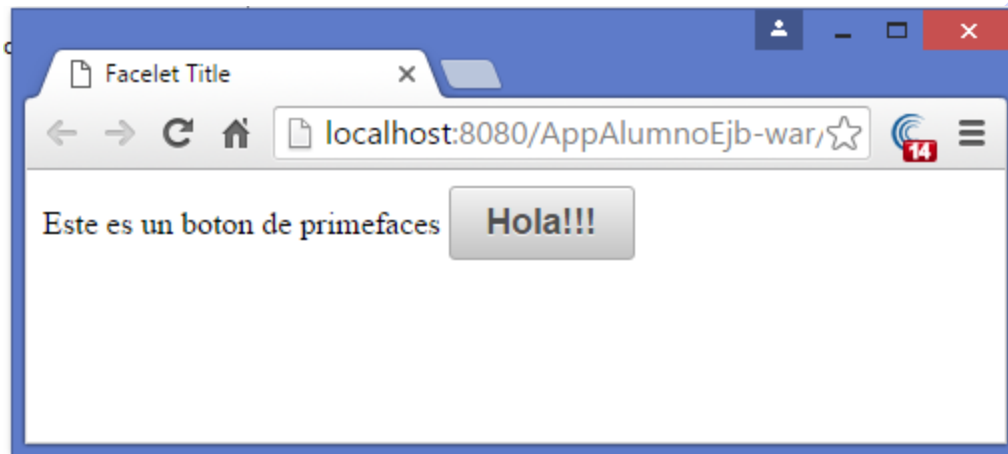
```
Este es un boton de primefaces
<h:form>
    <p:commandButton value="Hola!!!"/>
</h:form>
```

4. Guardar los cambios, si todo esta correcto no debe aparecer ningún error al crear los tag <h:form> de jsf y el tag <p:commanButtom> de primefaces.

#### 4. DEPURAR Y EJECUTAR EL PROYECTO

- Hacer un clean al Build al proyecto, desde el modulo principal (triangulo), luego hacer un deploy y por ultimo cuando todo lo anterior haya finalizado, correr el proyecto ... Run.
- Si todo salió correcto, debemos ver en nuestro navegador la página index.xhtml con el contenido siguiente:

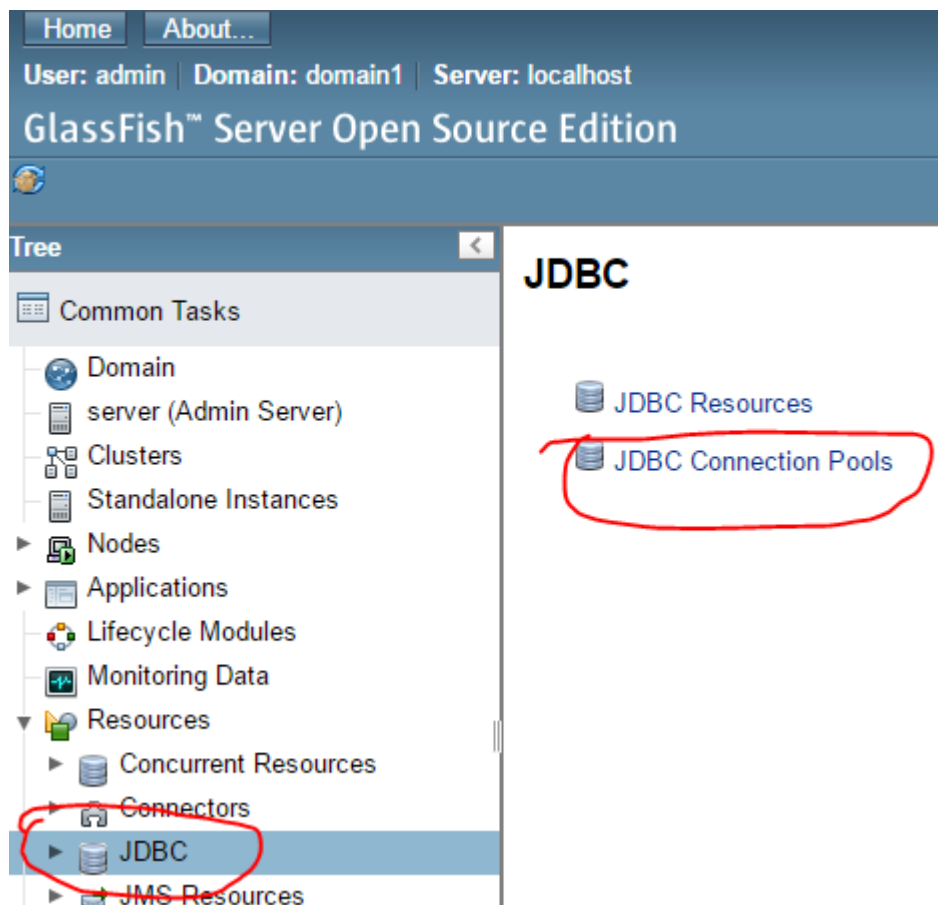




## 5. CREAR DATASOURCE Y POOL DE CONEXIONES EN GLASSFISH

- CREAREMOS PRIMERO UN POOL DE CONEXIONES

1. Ingresar a la consola de administración de Glassfish y buscar la opción JDBC



2. Clic en JDBC Connection Pools y crear un nuevo pool de nombre: alumnoPool, asignar la configuración que se ve en la imagen, hacer clic al botón next para continuar con la configuración.



## New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

Next Cancel

\* Indicates required field

### General Settings

Pool Name: \*

alumnoPool

Resource Type:

javax.sql.DataSource

Must be specified if the datasource class implements more than 1 of the interface.

Database Driver Vendor:

Oracle

Select or enter a database driver vendor

Introspect:

☐ Enabled

If enabled, data source or driver implementation class names will enable introspection.

3. En la siguiente pantalla, asignar los datos de conexión en la parte de Additional Properties, según la imagen

Additional Properties (18)		
<input type="checkbox"/> <input type="checkbox"/>   <input type="button" value="Add Property"/> <input type="button" value="Delete Properties"/>		
Select	Name	Value
<input type="checkbox"/>	TNSEntryName	
<input type="checkbox"/>	Description	
<input type="checkbox"/>	User	neo
<input type="checkbox"/>	MaxStatements	0
<input type="checkbox"/>	DatabaseName	orcdB
<input type="checkbox"/>	ImplicitCachingEnabled	false
<input type="checkbox"/>	NetworkProtocol	tcp
<input type="checkbox"/>	URL	jdbc:oracle:thin:neo/12345@localhost:1521:orc
<input type="checkbox"/>	ConnectionCacheName	
<input type="checkbox"/>	DataSourceName	OracleDataSource
<input type="checkbox"/>	LoginTimeout	0
<input type="checkbox"/>	ServiceName	
<input type="checkbox"/>	ServerName	
<input type="checkbox"/>	ONSConfiguration	
<input type="checkbox"/>	DriverType	
<input type="checkbox"/>	PortNumber	1521
<input type="checkbox"/>	ExplicitCachingEnabled	false
<input type="checkbox"/>	Password	12345

4. Hacer clic al botón finalizar para terminar con la configuración y se cree el pool.
5. Para comprobar que la configuración esta correcta, en la pantalla de la ficha general hacer clic al botón Ping y se deberá mostrar un mensaje como el siguiente:

**Edit JDBC Connection Pool**

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular

Load Defaults Flush **Ping**

**General Settings**

Pool Name: alumnoPool

Resource Type: javax.sql.ConnectionPoolDataSource ▼  
Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname: oracle.jdbc.pool.OracleDataSource  
Vendor-specific classname that implements the DataSource and/or XADataSource APIs

Driver Classname:   
Vendor-specific classname that implements the java.sql.Driver interface.

6. Si aparece un error en color rojo, puede ser porque no hemos copiado en la carpeta lib dentro de directorio de instalación de glassfish el driver ojdbc6.jar u ojdbc7.jar correspondiente para Oracle.  
C:\Servidores\_Web\glassfish4\glassfish\lib en mi caso

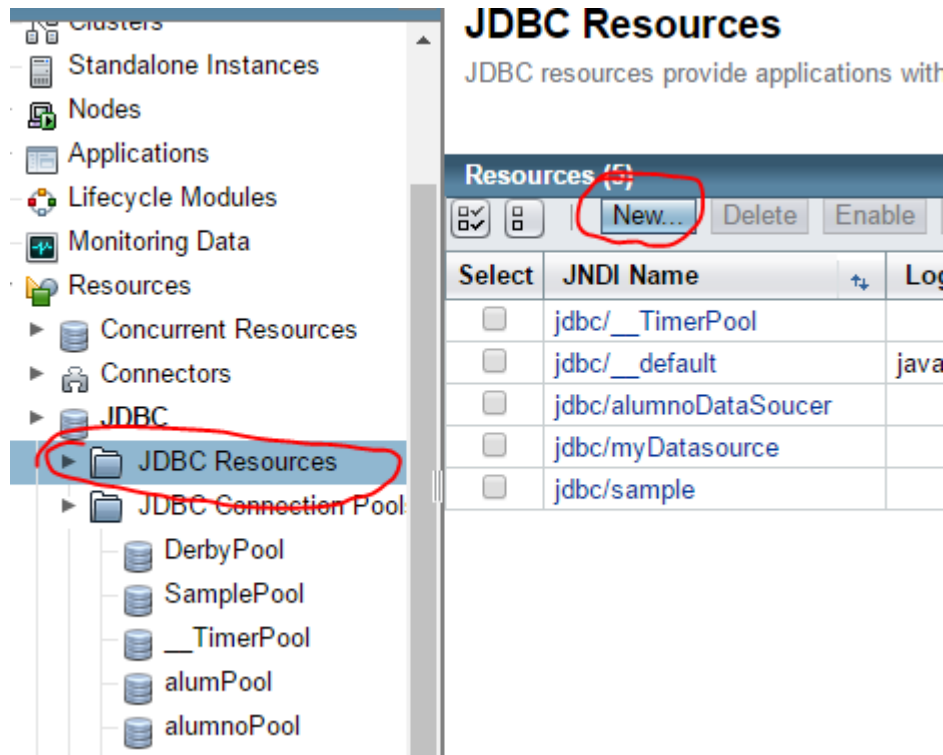
Descargar el driver

<http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html>

7. Listo

## • CREAR EL DATASOURCE

1. Con el pool ya creado, corresponde crear un datasource que hará uso del pool ya mencionado, así que debemos abrir la consola de administración de glassfish (Si es que la has cerrado) y buscar en JDBC la opción JDBC Resource, y hacer clic al botón New...



2. Colocar un nombre al Recurso según la imagen y seleccionar el pool creado anteriormente en la opción siguiente, luego hacer clic al botón ok

### New JDBC Resource

Specify a unique JNDI name that identifies the JDBC resource you want to create. The name must contain only alphanumeric, underscore, dash, or dot characters.

JNDI Name: \* jdbc/alumnoDataSource

Pool Name: alumnoPool

Use the [JDBC Connection Pools](#) page to create new pools

Description:

Status: ☒ Enabled

#### Additional Properties (0)

Add Property Delete Properties

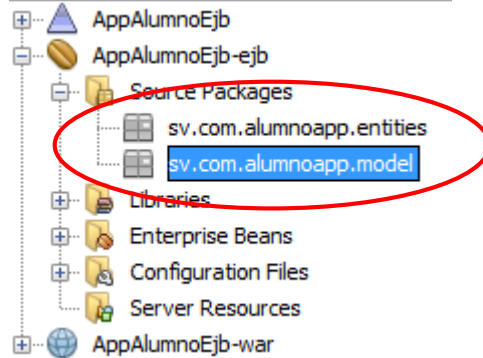
Select	Name	Value	Description
No items found.			

3. Listo con esto ya tenemos la configuración necesaria para utilizarla en la aplicación.

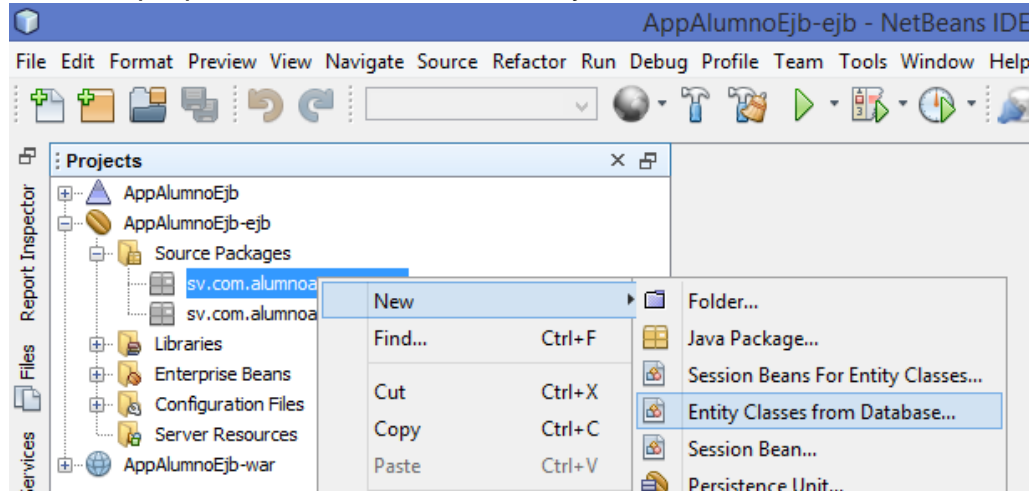
## 6. DESARROLLO DEL CRUD

### ➤ Creación del Entity Class

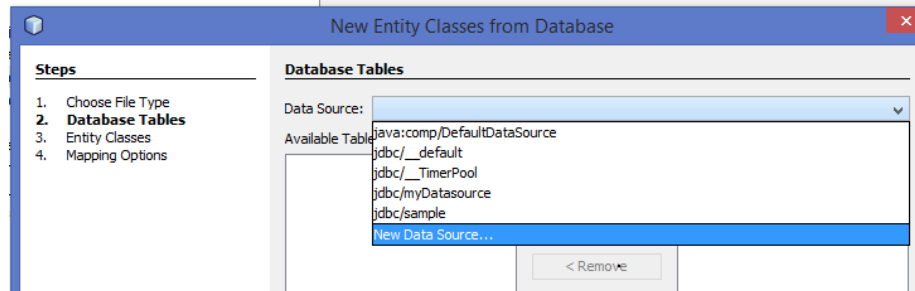
- En el módulo ejb crear los siguientes 2 paquetes



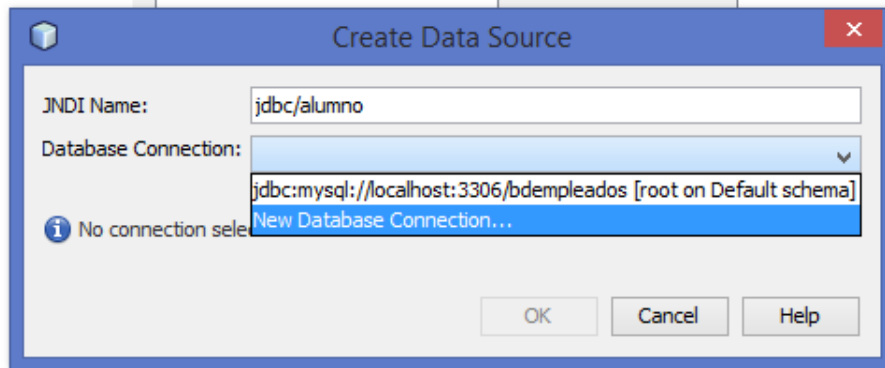
- Sobre el paquete entities, crear un Entity Class From Database



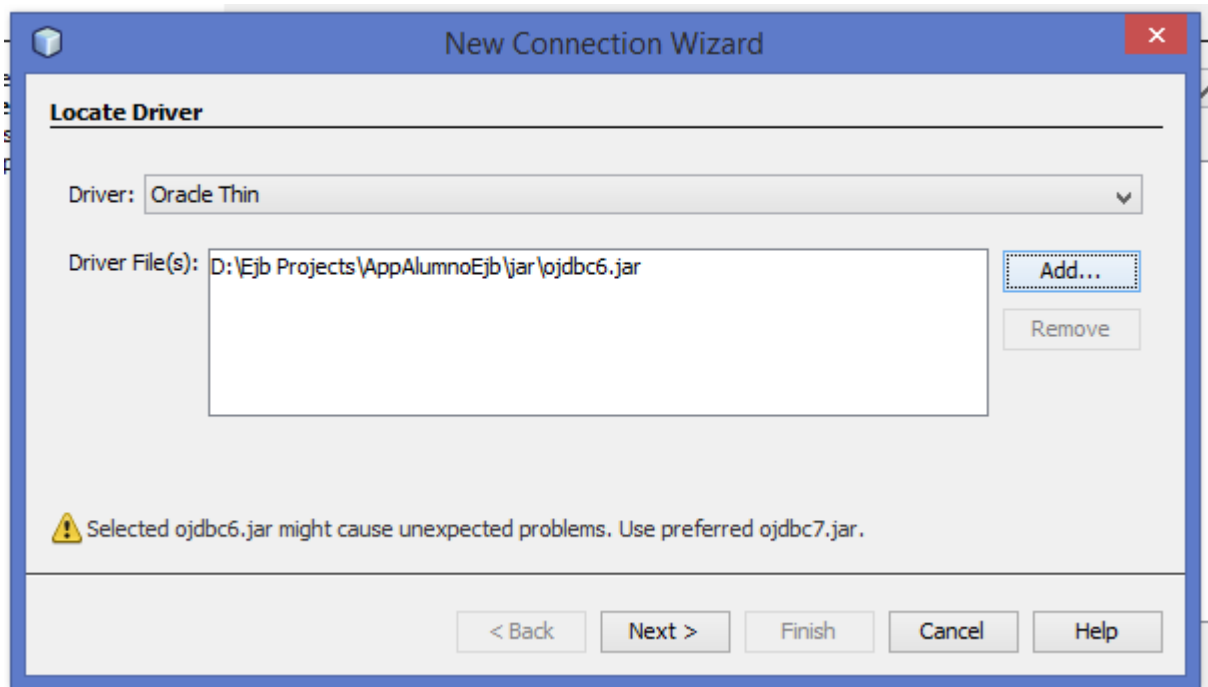
- Seleccionar un nuevo DataSource, New Data Source...



- Colocar el nombre al JNDI y en DatabaseConnection seleccionar New Database Connection



- En la siguiente pantalla, seleccionar Oracle Thin en Driver y agregar el driver correspondiente para Oracle 11g desde el botón Add... y clic en next



- En esta pantalla, especificar los datos para la conexión, recordar hacer clic al botón Test Connection para verificar que todo está correcto.

**New Connection Wizard**

**Customize Connection**

Driver Name: Oracle Thin (Service ID (SID))

Host: localhost Port: 1521

Service ID (SID): orclb

User Name: neo

Password: •••••

☒ Remember password

Connection Properties Test Connection

JDBC URL: jdbc:oracle:thin:@localhost:1521:orclb

**Connection Succeeded.**

< Back Next > Finish Cancel Help

- Hacer clic en next dos veces y aparecería la siguiente pantalla a la cual daremos clic a finalizar

**New Connection Wizard**

**Choose name for connection**

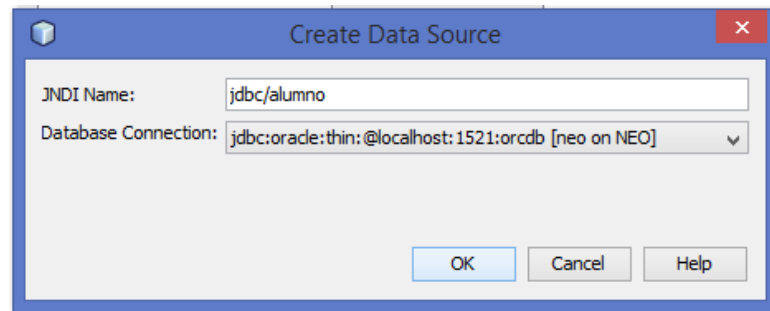
Override the default name for the connection. The name should be descriptive about the connection you are creating.

Input connection name:

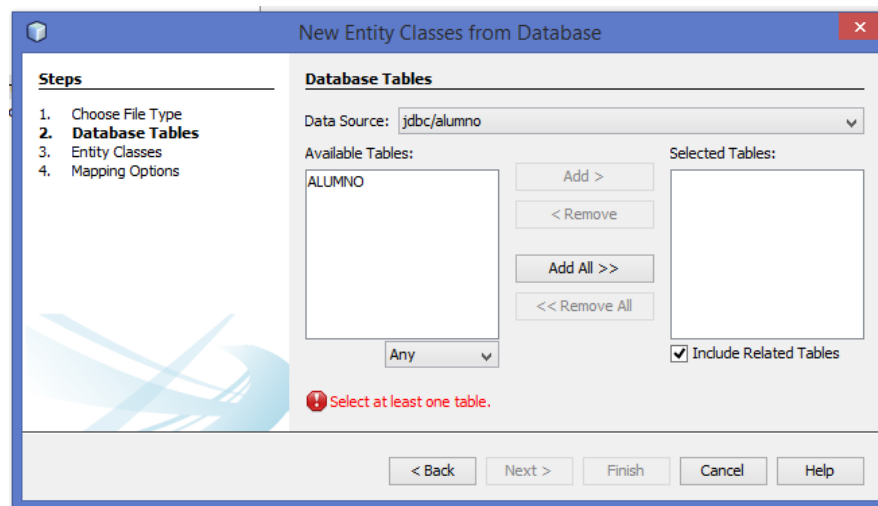
jdbc:oracle:thin:@localhost:1521:orclb [neo on Default schema]

< Back Next > Finish Cancel Help

- Clic en ok a esta pantalla

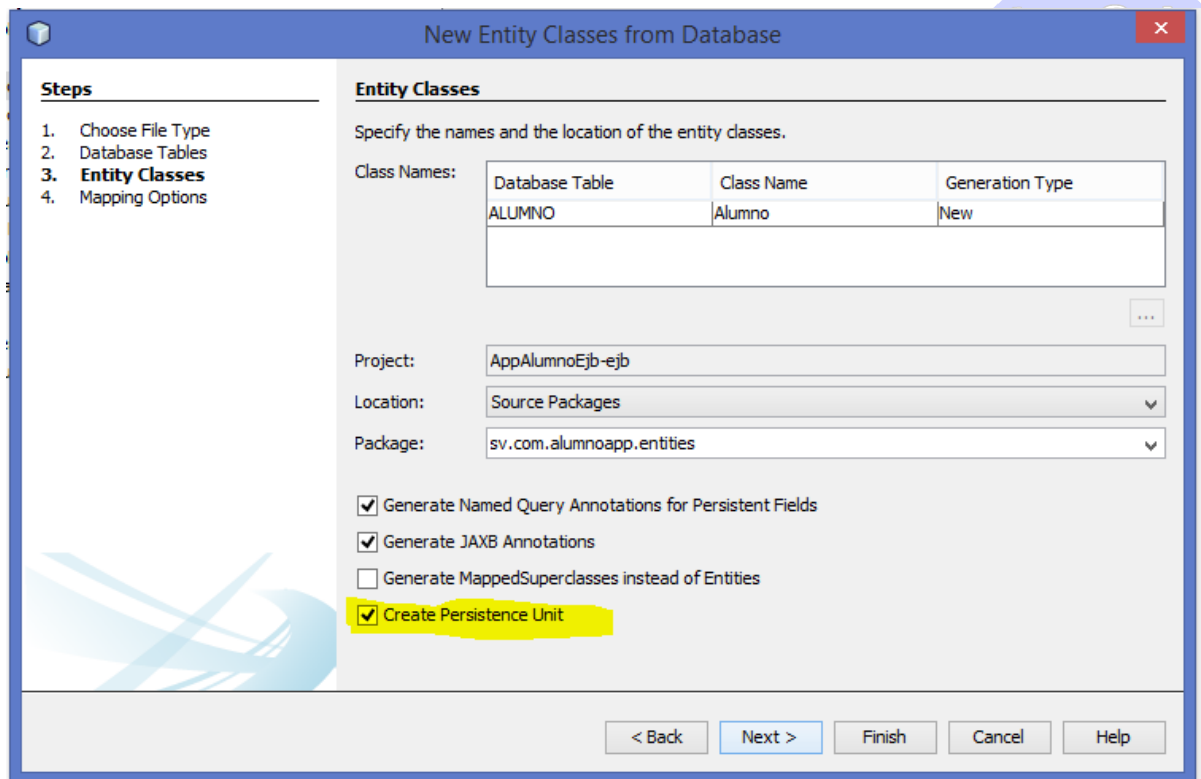


- Esperar a que el datasource liste las tablas de la base de datos, en este caso solo tenemos la tabla alumno

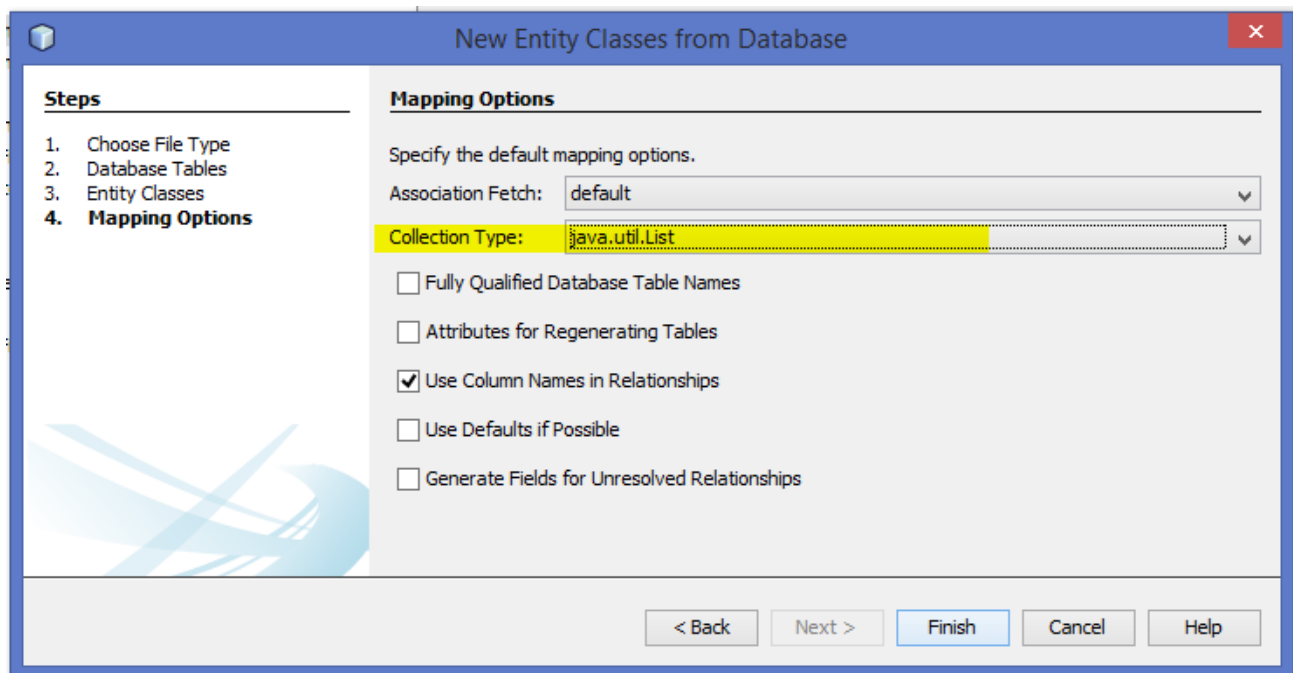


- Clic al botón Add All>> y luego clic en next, nos aparecerá la siguiente pantalla dejar la configuración según la imagen.

Es importante que la opción Create Persistence Unit esté seleccionado. Clic en Next.

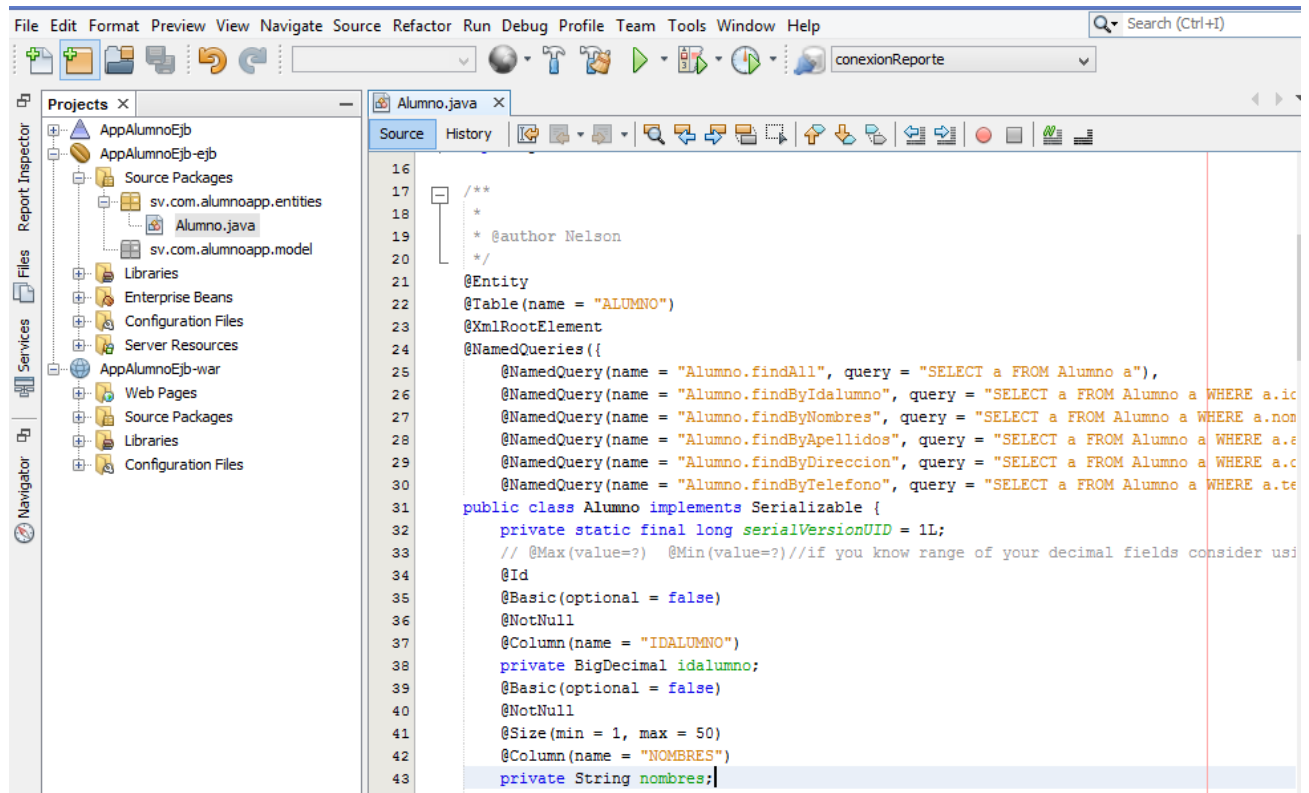


- En esta pantalla cambiar el Collection Type por java.util.list, el cual posee una clase más sencilla de utilizar. Clic en finalizar.



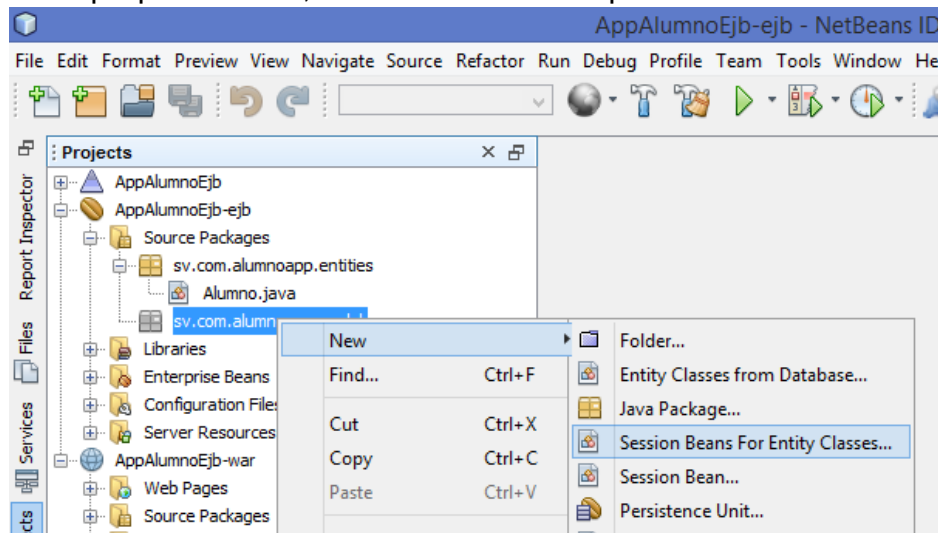


- Si todo salió bien, podremos ver la clase Alumno creada con una serie de anotación @NamedQuery, que permitirán extraer la información de la tabla correspondiente

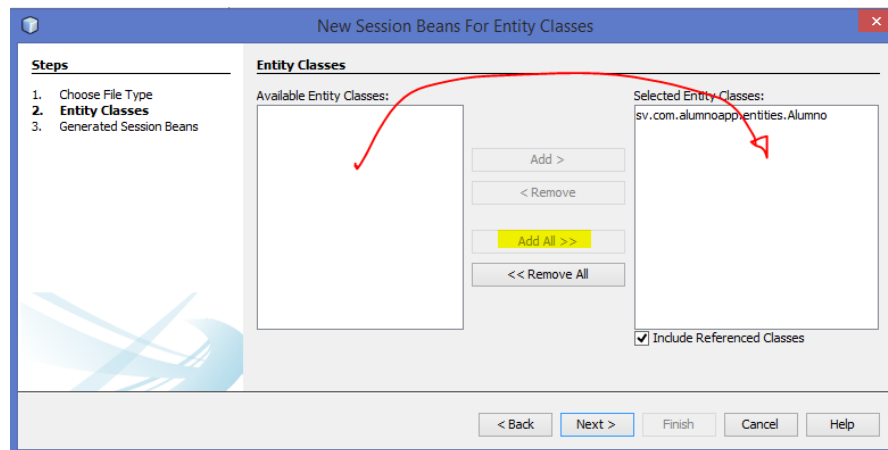


## ➤ Creación de la lógica de Negocio

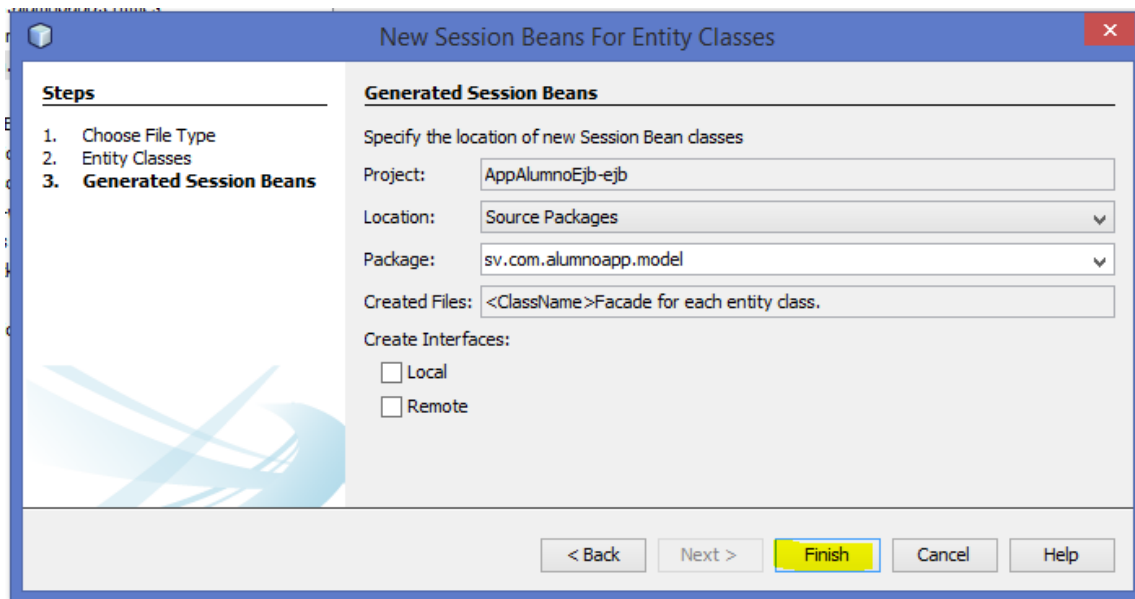
- En el paquete model, crear una clase de tipo: Session Beans For Entity Classes



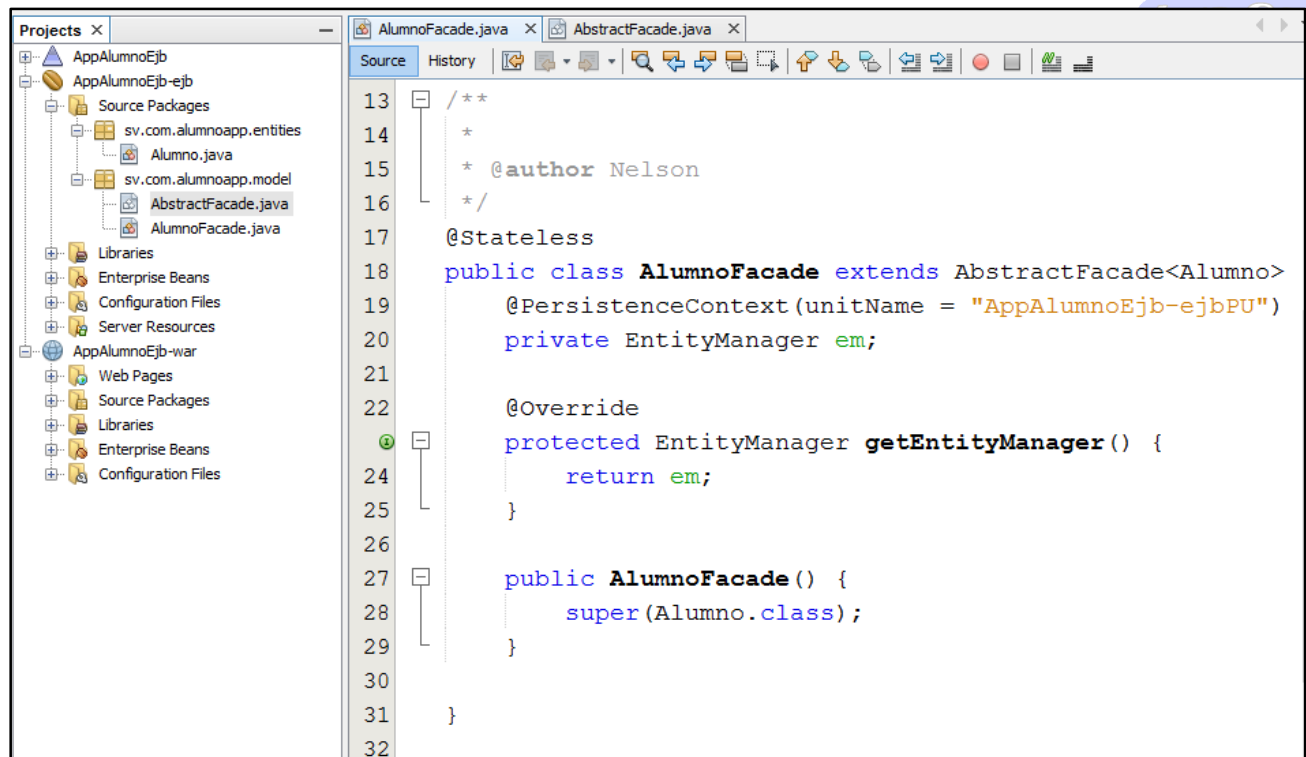
- Se mostrará la entidad creada, hacer clic al botón Add All, para trasladarlo hacia la derecha, clic en next



- Dejar las opciones como aparecen y clic en Finish



- Se crearan las Clases AbtracFacade.java y AlumnoFacade.java, los cuales poseen la lógica de negocios que manejara nuestra aplicación



### AbstractFacade.java:

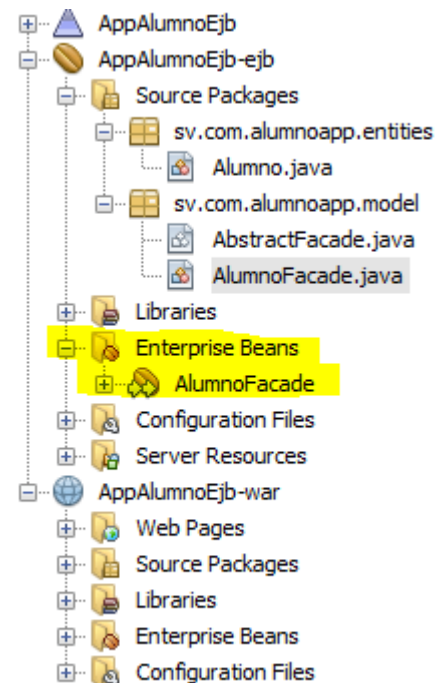
Contiene los métodos Crud y las operaciones que se podrán realizar sobre los datos de la entidad Alumno.

### AlumnoFacade.java

Esta clase posee un EntityManager, el cual maneja los métodos crud y demás operaciones que se crearon en la clase AbstractFacade; así también hará la persistencia de la información.

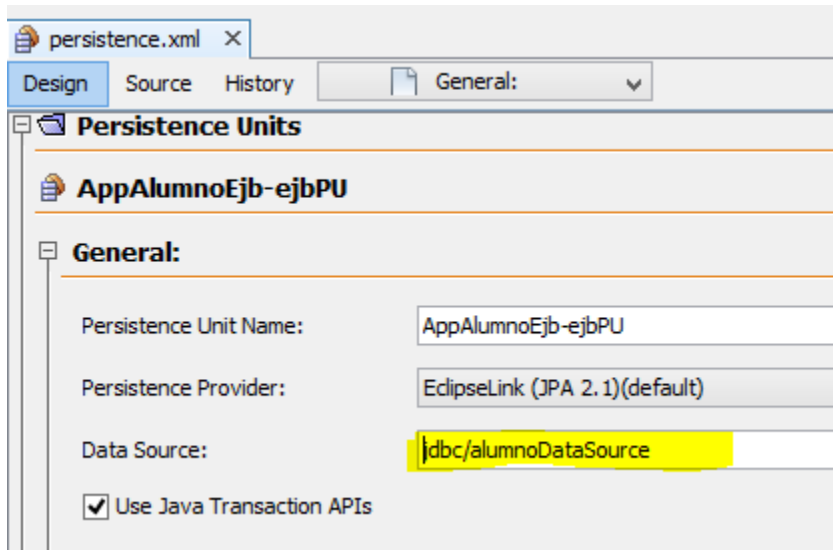
### Nota:

En nuestro folder Enterprise Beans, se debió crear el Bean AlumnoFacade



## 7. CORREGIR EL ARCHIVO persistence.xml

- Buscar y corregir el nombre del Data Source por el que creamos en glassfish, ya que al dejarlo así, se genera un error al momento de ejecutar la aplicación y por ende no funciona



- Guardar los cambios y continuamos

## 8. DEFINIENDO LOS METODOS CRUD PARA LA ENTIDAD ALUMNO

- **Creación De Un Managedbean De Java Server Faces**

1. En el Modulo WAR, crear un paquete nuevo llamado controller, siguiendo el estándar: sv.com.alumnoapp.controller.
2. Crear una clase tipo ManagedBean de Java Server Faces con los datos según la pantalla siguiente:

**New JSF Managed Bean**

**Steps**

1. Choose File Type
2. Name and Location

**Name and Location**

Class Name: AlumnoController

Project: AppAlumnoEjb-war

Location: Source Packages

Package: sv.com.alumnoapp.controller

Created File: mnoEjb\AppAlumnoEjb-war\src\java\sv\com\alumnoapp\controller\AlumnoController.java

☐ Add data to configuration file

Configuration File:

Name: alumnoController

Scope: session

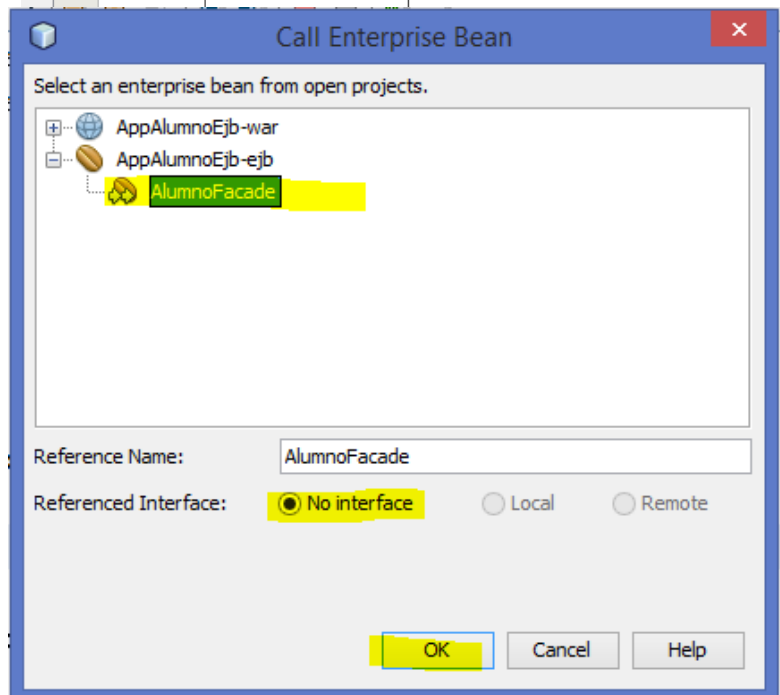
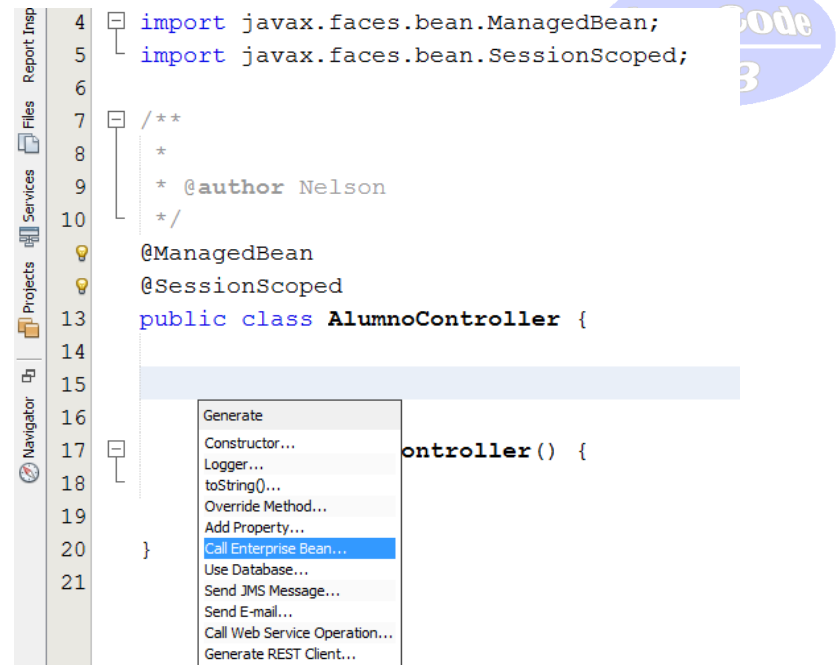
< Back   Next >   Finish   Cancel   Help

3. Clic en finalizar

**Nota:** Antes de proceder con la creación de los métodos, se debe crear en el controller una instancia hacia la clase AlumnoFacade para poder obtener acceso a los métodos u operaciones crud definidas en ellos.

Clic derecho sobre la clase, seleccionar insert code, en el menú que se muestra seleccionar Call Enterprise Bean

En esta pantalla, buscar el Enterprise Bean "AlumnoFacade", seleccionarlo y hacer clic al botón ok, no es necesario crear interfaces.



Deberá aparecer en la clase estas dos líneas de código, que indica que estamos enlazados al Enterprise Bean para poder hacer uso de sus métodos

```
@ManagedBean
@SessionScoped
public class AlumnoController {
    @EJB
    private AlumnoFacade alumnoFacade;
}
```

- **Creación del Metodo ListarAlumnos**

1. Bajo el método constructor, crear el siguiente método con el cual listaremos todos los datos de los alumnos que existen en la tabla alumno de la base de datos

```
public List<Alumno> listaTodosAlumnos() {
    return alumnoFacade.findAll();
}
```

2. Guardar los cambios y listo.

- **Creación del Método NuevoAlumno**

1. Crear una variable privada de tipo objeto:

```
private Alumno alumno= new Alumno();
```

2. Generar para esta variable los métodos Getter y Setter

```
public Alumno getAlumno() {
    return alumno;
}
```

```
public void setAlumno(Alumno alumno) {
    this.alumno = alumno;
}
```

3. Luego crear el método addAlumno, según el código siguiente:

```
//Agregar un nuevo alumno
public String addAlumno() {
    alumnoFacade.create(alumno);
    this.alumno=new Alumno();
    return "index";
}
```

4. Guardar los cambios y Listo.

- **Creación del Método Editar Alumno**

1. Crear un método para preparar la edición de la entidad alumno, para ello crear el siguiente código:

```
//Editar datos de un alumno existente
public String prepareEditAlumno(Alumno a) {
    this.alumno = a;
    return "edit";
}
```

Con este método estamos únicamente obteniendo los datos del alumno que será actualizado por medio del parámetro "a", por lo tanto se le pasa dicho

valor a la variable de tipo objeto “alumno” y se muestra la vista correspondiente con los datos a ser editados que se recuperaron por medio de este metodo.

2. Crear el método donde se efectuara la edición de los datos y posterior almacenado en la bd.

```
public String editAlumno() {
    this.alumnoFacade.edit(this.alumno);
    this.alumno=new Alumno();
    return "index";
}
```

3. Guardamos los cambios y listo

- **Creación del Método Eliminar Alumno**

1. Creamos un método tipo void con un parámetro para poder eliminar los datos del alumno seleccionado, el código es el siguiente:

```
//Eliminar datos de un alumno existente
public void deleteAlumno(Alumno a) {
    this.alumnoFacade.remove(a);
}
```

2. Guardamos los cambios y listo.



## 4. CREACION DE FORMULARIOS Y VISTAS

- **Formulario para listar Alumnos**

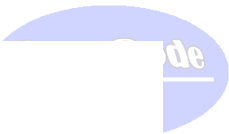
1. En index.xhtml crear el siguiente código para desplegar los datos de todos los alumnos en una tabla:

```
<h:body>
  <h:form id="frmListaAlumnos">
    <h:link value="Nuevo Alumno" outcome="add"/>
    <br></br>
    <p:dataTable var="al" value="#{alumnoController.listaTodosAlumnos()}"
      paginator="true" paginatorPosition="top" rows="10"
      emptyMessage="No hay datos de alumnos"
      style="font-size: 12px;">
      <p:column headerText="ID">
        <h:outputText value="#{al.idalumno}"/>
      </p:column>
      <p:column headerText="NOMBRE DEL ALUMNO">
        <h:outputText value="#{al.nombres} #{al.apellidos}"/>
      </p:column>
      <p:column headerText="DIRECCION">
        <h:outputText value="#{al.direccion}"/>
      </p:column>
      <p:column headerText="TELEFONO">
        <h:outputText value="#{al.telefono}"/>
      </p:column>
    </p:dataTable>
  </h:form>
```

2. Notar como en primer lugar se ha creado un link, el cual permitirá invocar a otra página llamada "add" el cual contendrá un formulario vacío para que pueda ser ingresado un nuevo alumno.
3. En la parte del dataTable se está llamando a nuestro método "listaTodosAlumnos()", creado en el bean "alumnoController"

- **Formulario para crear nuevo alumno**

1. En web pages, crear una nueva página jsf page llamada "add.xhtml"
2. Crear el siguiente formulario:



```

<h:form>
  <h1><h:outputText value="Resgistrar Nuevo Alumno"/></h1>
  <h:panelGrid columns="2">
    <h:outputLabel value="Nombres:" for="nombres" />
    <h:inputText id="nombres" value="#{alumnoController.alumno.nombres}"
      title="Nombres" required="true"
      requiredMessage="El Nombre es requerido"/>
    <h:outputLabel value="Apellidos:" for="apellidos" />
    <h:inputText id="apellidos" value="#{alumnoController.alumno.apellidos}"
      title="Apellidos" required="true"
      requiredMessage="Los Apellidos son requeridos"/>
    <h:outputLabel value="Direccion:" for="direccion" />
    <h:inputText id="direccion" value="#{alumnoController.alumno.direccion}"
      title="Direccion" required="true"
      requiredMessage="La Direccion es requerida"/>
    <h:outputLabel value="Telefono:" for="telefono" />
    <h:inputText id="telefono" value="#{alumnoController.alumno.telefono}"
      title="Telefono" required="true"
      requiredMessage="El Telefono es requerido"/>
    <h:commandButton value="Guardar" action="#{alumnoController.addAlumno()}" />
  </h:panelGrid>
</h:form>

```

3. Notar como se ha creado un botón para poder guardar la información por medio del método **addAlumno** creado en el bean.
4. Listo guarda los cambios.

- **Formulario para editar datos de un Alumno existente**

1. Crear una nueva página jsf page, llamada edit.xhtml
2. Crear el siguiente formulario

```
<h:form>
    <h1><h:outputText value="Editar Alumno"/></h1>
    <h:panelGrid columns="2">
        <h:outputLabel value="Idalumno:" for="idalumno" />
        <h:inputText id="idalumno" value="#{alumnoController.alumno.idalumno}"
            readonly="true"/>
        <h:outputLabel value="Nombres:" for="nombres" />
        <h:inputText id="nombres" value="#{alumnoController.alumno.nombres}"
            title="Nombres" required="true"
            requiredMessage="El Nombre es requerido"/>
        <h:outputLabel value="Apellidos:" for="apellidos" />
        <h:inputText id="apellidos" value="#{alumnoController.alumno.apellidos}"
            title="Apellidos" required="true"
            requiredMessage="Los Apellidos son requeridos"/>
        <h:outputLabel value="Direccion:" for="direccion" />
        <h:inputText id="direccion" value="#{alumnoController.alumno.direccion}"
            title="Direccion" required="true"
            requiredMessage="La Direccion es requerida"/>
        <h:outputLabel value="Telefono:" for="telefono" />
        <h:inputText id="telefono" value="#{alumnoController.alumno.telefono}"
            title="Telefono" required="true"
            requiredMessage="El Telefono es requerido"/>
        <h:commandButton value="Guardar" action="#{alumnoController.editAlumno()}" />
    </h:panelGrid>
</h:form>
```

3. Notar que se ha dejado el id pero como solo lectura para que no pueda ser modificado, así también en el botón estamos llamando al método editAlumno() creado en nuestro bean.
4. Listo guardar los cambios

- **Crear las opciones para editar y eliminar**

1. En index.xhtml, crear una nueva columna al final de la última que existe
2. Colocar el siguiente código

```
<p:column headerText="Opciones">
    <h:commandLink value="Edit"
        action="#{alumnoController.prepareEditAlumno(al) }"/>
    |
    <h:commandLink value="Delete"
        action="#{alumnoController.deleteAlumno(al) }"
        onclick="return confirm('Esta seguro??')"/>
</p:column>
```

3. Notar como se están utilizando los otros métodos prepareEditAlumno y deleteAlumno, a ambos se les pasa como parámetro la variable de la tabla.
4. Guardar los cambios y listo

## 5. EJECUTAR EL PROYECTO

- Hacer un clean al build al modulo principal
- Luego hacer un deploy
- Ejecutar el proyecto

Nuevo Alumno

ID	NOMBRE DEL ALUMNO	DIRECCION	TELEFONO	Opciones
1	JOSE ERNESTO ZURITA NUÑEZ	AV. LA COLINA 2, CASA#12	7845-9623	<a href="#">Edit</a>   <a href="#">Delete</a>
2	DAVID ANTONIO LOPEZ CRUZ	CALLE INDEPENDENCIA, CASA#11	7158-6242	<a href="#">Edit</a>   <a href="#">Delete</a>
3	BLANCA MARGARITA CARIAS RUIZ	AV. LAS MAGNOLIAS SUR, CASA#8	7348-958	<a href="#">Edit</a>   <a href="#">Delete</a>
4	STAYCU CECILIA MENDEZ SUEZ	COL. EL ZONTLE, CASA#2	7956-8542	<a href="#">Edit</a>   <a href="#">Delete</a>
5	REBECA ABIGAIL SERMEÑO AGUIRRE	RES. BOSQUE DE LA PAZ, POL A, CASA#29	7412-5836	<a href="#">Edit</a>   <a href="#">Delete</a>

## Resgistrar Nuevo Alumno

Nombres:

Apellidos:

Direccion:

Telefono:

### Importante:

Para que el registro de un nuevo alumno sea exitoso, es necesario anexar las dos anotaciones en el campo idAlummno, para que utilice la sequencia creada en nuestra bd y de esa forma el idAlumno se autogenere.

```

@Id
@Basic(optional = false)
@NotNull
@Column(name = "IDALUMNO")
@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "seq_alumno")
@SequenceGenerator(name = "seq_alumno", sequenceName = "incremento_id_alumno", allocationSize = 1)
private BigDecimal idalumno;

```

Diagrama de anotaciones:

- 1: `GeneratedValue`
- 2: `strategy = GenerationType.SEQUENCE`
- 3: `generator = "seq_alumno"`
- 4: `sequenceName = "incremento_id_alumno"`

Vemos acá en **@GeneratedValue** que el tipo de generación del id es secuencial, así también le indicamos un **generator** el cual puede ser cualquier nombre; mientras en el **@SequenceGenerator** le indicamos en el nombre el mismo valor que colocamos para **generator** en la anotación anterior y luego por medio del **sequenceName** le indicamos el nombre de la secuencia tal cual se creó en la **base de datos**, es importante mencionar que si el nombre no es el mismo se genera un error indicando que no se localiza el sequence; así también se indica el **allocationSize**, esto es el tamaño de asignación con un valor de 1, para que el incremento sea cada vez de 1 en 1.

## Editar Alumno

Idalumno:

Nombres:

Apellidos:

Direccion:

Telefono:

## Eliminar un registro

[Nuevo Alumno](#)

ID	NOMBRE	DIRECCION	TELEFONO	Opciones
452	SALOMON CHAVEZ	AV. LA ROCA FUERTE	8888-9999	<a href="#">Edit</a>   <a href="#">Delete</a>
453	LUIS EDILBERTO	AV. LA ROCA FUERTE	8888-9999	<a href="#">Edit</a>   <a href="#">Delete</a>
454	LUIS EDILBERTO ROBLES LOPEZ	AV. LA ROCA FUERTE	8888-9999	<a href="#">Edit</a>   <a href="#">Delete</a>

localhost:8080 dice:  
Esta seguro???

## CONCLUSION:

Espero este material se de mucha utilidad y sirva para despejar dudas en cuanto al desarrollo de aplicaciones empresariales con java, como se mencionaba en un principio este material es para un uso a nivel intermedio en lo que respecta a programación en JAVA.

Para más Sigue mis Web y suscríbete:

<http://www.javacode503.net/> 

[Facebook](#) 

[google plus](#) 

[YouTube](#) 