



M03	Material	Componentes de acceso a datos
------------	-----------------	-------------------------------

Contenidos

- Concepto de componente. Características.
- Propiedades.
- Atributos.
- Eventos; asociación de acciones a eventos.
- Introspección. Reflexión.
- Persistencia del componente.
- Herramientas para desarrollo de componentes no visuales.
- Empaquetado de componentes.
- Tipos de EJB.
- Instalaciones previas al ejemplo.
- Ejemplo de EJB con Netbeans.
- JPA.



Definición de componente

- Un componente es una unidad de software que encapsula partes de código con una funcionalidad determinada.
 - Los componentes pueden ser visuales del estilo a los proporcionados por lo entornos de desarrollo para ser incluidos en interfaces de usuario, o no-visuales, los cuales tienen funcionalidad como si fueran librerías remotas.
 - Trataremos los Enterprise Java Beans (EJB) como ejemplo y su uso en el contexto del acceso a datos.
-

Características

- Es una unidad ejecutable que puede ser instalada y utilizada independientemente.
 - Puede interactuar y operar con otros componentes desarrollados por terceras personas.
 - No tiene estado, al menos su estado no es externamente visible.
 - Un componente y la programación orientada a componentes puede, metafóricamente, asociarse a los componentes electrónicos.
-

Propiedades

- Las propiedades de un componente determinan su estado y lo diferencian del resto.
 - Las propiedades de un componente se dividen en simples, indexadas, compartidas y restringidas.
 - Las propiedades de un componente se pueden examinar y modificar mediante métodos o funciones de acceso:
 - El método *get*, que sirve para consultar o leer.
 - El método *set*, que sirve para asignar o cambiar su valor.
-

Simples e indexadas

- Las propiedades simples son aquellas que representan un único valor. Por ejemplo, suponiendo un botón de una interfaz gráfica, las propiedades simples serían aquellas relacionadas con su tamaño, su color de fondo, su etiqueta, etc.
 - Otro tipo de propiedades más complejas y muy similares a un conjunto de valores: indexadas. Los elementos de este tipo de propiedades comparten todos ellos el mismo tipo y a ellos se accede mediante un índice.
-

Compartidas y restringidas

- Las propiedades compartidas son aquéllas que cuando cambian notifican a todas las partes interesadas en esa propiedad, y sólo a ellas, la naturaleza del cambio.
- Las propiedades restringidas buscan la aprobación de otros componentes antes de cambiar su valor.



Atributos

- Los atributos son uno de los aspectos relevantes de la interfaz de un componente ya que son los elementos que forman parte de la vista externa del componente (los representados como públicos).
 - Estos elementos observables son particularmente importantes desde el punto de vista del control y del uso del componente, y son la base para el resto de los aspectos que caracterizan a un componente.
-

Eventos

- Una interfaz contiene sólo información sintáctica de las signatures de las operaciones entrantes y salientes de un componente, con las cuales otros componentes interactúan con él.
 - A este tipo de interacción se le conoce con el nombre de control proactivo, es decir, las operaciones de un componente son activadas mediante las llamadas de otro.
 - Existe otra forma que se denomina control reactivo y que se refiere a los eventos de un componente.
-

Herramientas

- Actualmente, los entornos de desarrollo de componentes no visuales más utilizados son:
 - NetBeans, Eclipse para el desarrollo de componentes en Java (por ejemplo, EJB).
 - Microsoft .NET para el desarrollo de ensamblados (COM+, DCOM).
-

Empaquetado de componentes

- Los cuatro tipos de módulos de JEE para aplicaciones web con EJBs son los siguientes:
 - Módulos EJB (.jar).
 - Los módulos web (.war).
 - Los módulos de aplicaciones de cliente (.jar).
 - Los módulos adaptadores de recursos que contienen todas las interfaces Java, clases, librerías nativas y otra documentación junto con su descriptor de despliegue.
-

Empaquetado de componentes



Java EE

- Java Platform, Enterprise Edition o Java EE es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.
 - Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.
 - La plataforma Java EE está definida por una especificación.
-

Java EE

- Java EE tiene varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos.
 - Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web.
-

Enterprise Java Bean (EJB)

- Los Enterprise JavaBeans (también conocidos por sus siglas EJB) son una de las interfaces de programación de aplicaciones (API) que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE) de Oracle Corporation (inicialmente desarrollado por Sun Microsystems).
-

Enterprise Java Bean (EJB)

- Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor, que son precisamente los EJB:
 - Comunicación remota utilizando CORBA.
 - Transacciones.
 - Control de la concurrencia.
 - Eventos utilizando JMS (Java Messaging Service).
 - Servicios de nombres y de directorio.
 - Seguridad.
 - Ubicación de componentes en un servidor de aplicaciones.
-

Enterprise Java Bean (EJB)

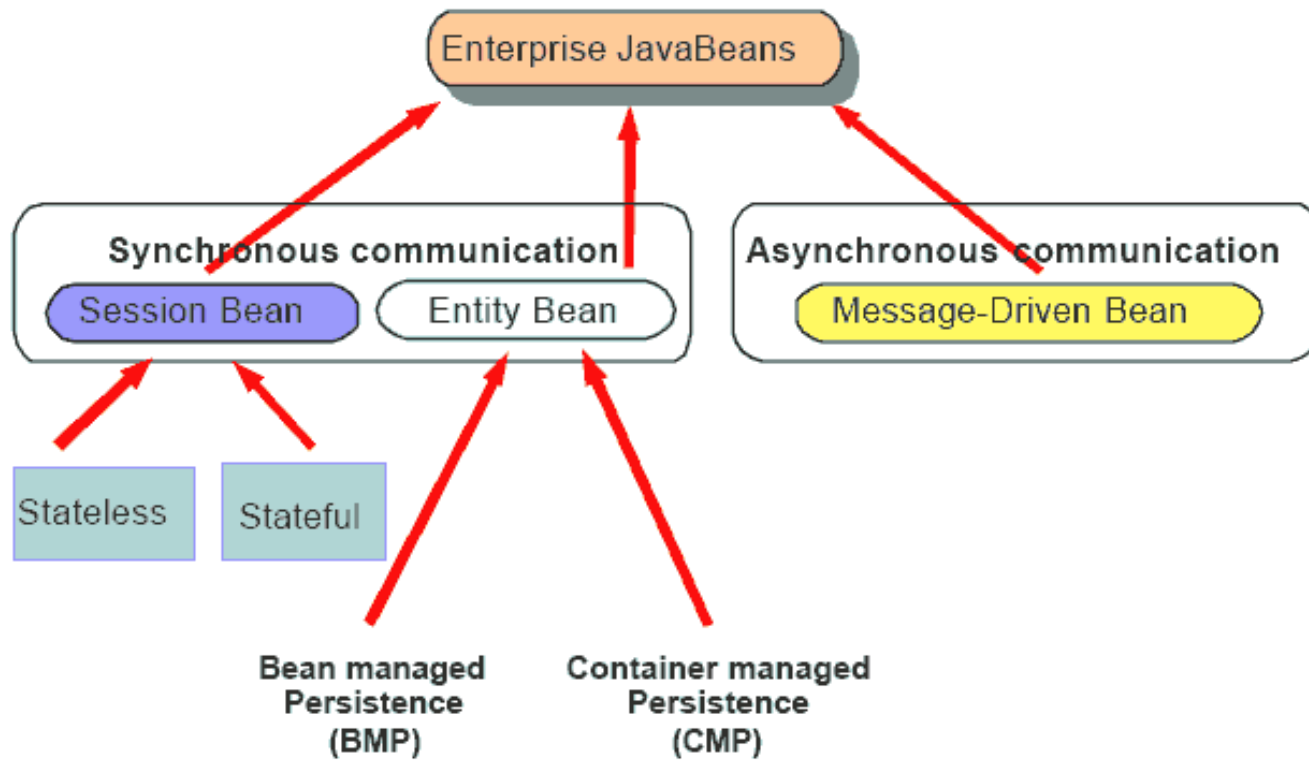
- Los EJB proporcionan un modelo de componentes distribuido estándar del lado del servidor.
 - El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (conurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí.
 - El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables.
-

Tipos de EJB

- EJBs de entidad (Entity EJBs)
 - Su objeto es encapsular los objetos de lado de servidor que almacenan los datos. Los EJBs de entidad presentan la característica fundamental de la persistencia.
 - EJBs de sesión (Session EJBs)
 - Gestionan el flujo de la información en el servidor. Generalmente sirven a los clientes como una fachada de los servicios proporcionados por otros componentes disponibles en el servidor.
 - EJBs dirigidos por mensaje (Message-driven EJBs)
 - Los únicos EJB con funcionamiento asíncrono. Usando el Java Messaging System (JMS), se suscriben a un tópico (topic) o una cola (queue) y se activan al recibir un mensaje dirigido a dicho tópico o cola. No requieren de su instanciación por parte del cliente.
-

Tipos de EJB

Enterprise JavaBeans



Persistencia del componente.

- En el caso de EJB la persistencia está facilitada con la librería Java Persistence API (JPA).
- Las clases que componen JPA:
 - Persistence
 - EntityManagerFactory
 - EntityManager
 - Entity
 - EntityTransaction
 - Query

Java
Persistence



JPA

- JPA o Java Persistence API es el standard de Java encargado de automatizar dentro de lo posible la persistencia de nuestros objetos en base de datos.
 - De esta forma tendremos a nuestra disposición un EntityManagerFactory con el que empezar a gestionar las entidades que se encuentran definidas. Ahora bien muchas aplicaciones JEE se conectan a varias bases de datos y generan distintos EntityManagerFactorys.
-

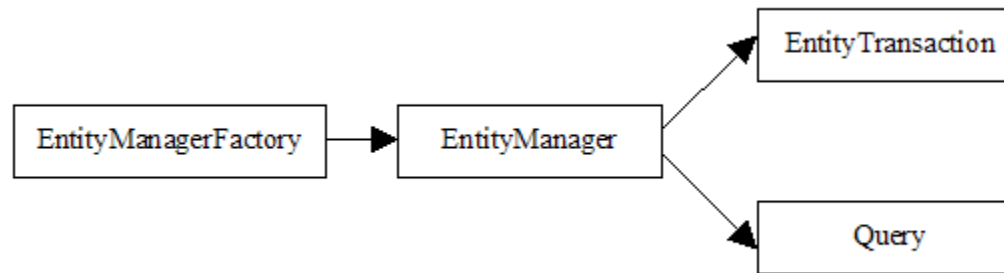
Entity Manager Factory

- En un primer lugar un EntityManagerFactory es único y es con el que nosotros gestionamos todas las entidades . Ahora bien si tenemos varias conexiones a base de datos deberemos definir un nuevo concepto que nos permite clarificar que tenemos dos EntityManagerFactories distintos. Este concepto es el que se conoce como PersistenceUnit. Cada PersistenceUnit tiene asociado un EntityManagerFactory diferente que gestiona un conjunto de entidades distinto.
-

Entity Manager

- Una vez disponemos de un EntityManagerFactory este será capaz de construir un objeto de tipo EntityManager que como su nombre indica gestiona un conjunto de entidades o objetos.
 - En principio estas entidades son objetos POJO normales con los cuales estamos trabajando en nuestro programa Java .
 - El EntityManager será el encargado de salvarlos a la base de datos , eliminarlos de la base de datos etc .
-

Entity Manager



- `persist()` – encargado de almacenar entidades en la base de datos.
 - `find()` - se encarga de localizar una Entidad a través de su clave primaria. Para ello necesita que le pasemos la clave y el tipo de Entidad a buscar.
 - `remove()` – se encarga de eliminar una entidad de la base de datos.
-

Entity Manager

- `flush()` - Este método es el encargado de sincronizar el `PersistenceContext` contra la base de datos. El `EntityManager` irá almacenando las modificaciones que nosotros realizamos sobre las entidades, para más adelante persistirlas contra la base de datos todas de golpe ya sea invocando `flush` o realizando un `commit` a una transacción.
 - `close()` – Cierra la `EntityManager`.
 - `getTransaction()` – devuelve un objeto de tipo `EntityTransaction`.
-

JPQL

- Java Persistence Query Language (JPQL) es un lenguaje de consulta orientado a objetos independiente de la plataforma definido como parte de la especificación Java Persistence API (JPA).
 - JPQL es usado para hacer consultas contra las entidades almacenadas en una base de datos relacional. Está inspirado en gran medida por SQL, y sus consultas se asemejan a las consultas SQL en la sintaxis, pero opera con objetos entidad de JPA en lugar de hacerlo directamente con las tablas de la base de datos.
-

JPQL

- Estructura:

```
SELECT . . . FROM . . .
```

```
[WHERE . . .]
```

```
[GROUP BY . . . [HAVING . . .]]
```

```
[ORDER BY . . .]
```

JPQL - Ejemplos

// Query for a List of objects.

```
Query query = em.createQuery("Select e FROM Employee e  
WHERE e.salary > 100000");
```

```
List<Employee> result = query.getResultList();
```

// Query for a single object.

```
Query query = em.createQuery("Select e FROM Employee e  
WHERE e.id = :id");
```

```
query.setParameter("id", id);
```

```
Employee result2 = (Employee) query.getSingleResult();
```

JPQL - Ejemplos

// Query for a List of data elements.

```
Query query = em.createQuery("Select e.firstName FROM  
Employee e");
```

```
List<String> result4 = query.getResultList();
```

// Query for a List of element arrays.

```
Query query = em.createQuery("Select e.firstName, e.lastName  
FROM Employee e");
```

```
List<Object[]> result5 = query.getResultList();
```

Requisitos para el Ejemplo

- El primer paso es asegurarnos de que tenemos instalado todo lo que necesitamos. Para este ejemplo:
 - MySQL (ya lo deberíamos tener de UF anteriores)
 - Glassfish
 - Módulo de aplicaciones Web de NetBeans.
 - Y.... NetBeans, claro ;)

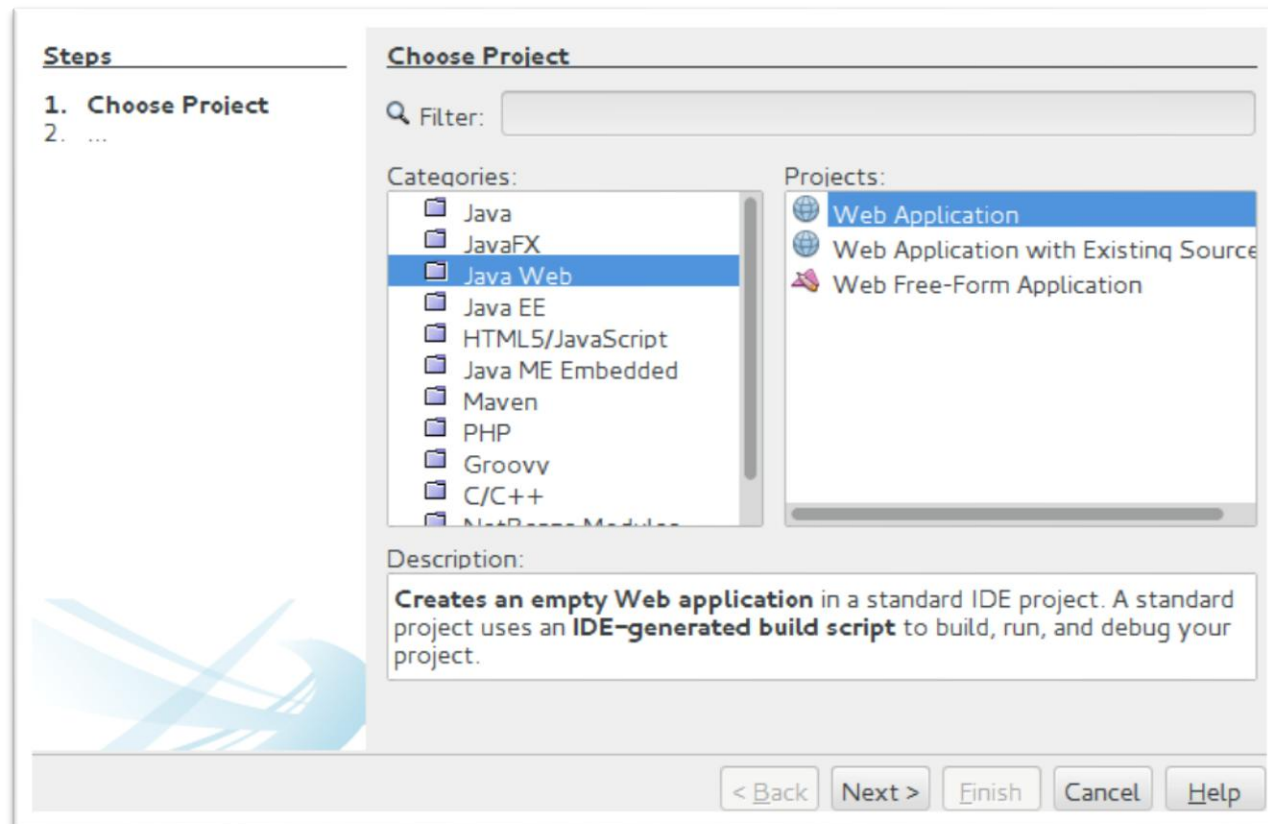


Ejemplo de EBJ con NetBeans

- A continuación mostraré un proyecto de ejemplo, basado en la tecnología de componentes EJB, gestionando la parte de la vista con un Servlet que será invocado desde un fichero html.
 - La base de datos ya está creada en MySQL. Cada tabla tiene una clave primaria simple.
-

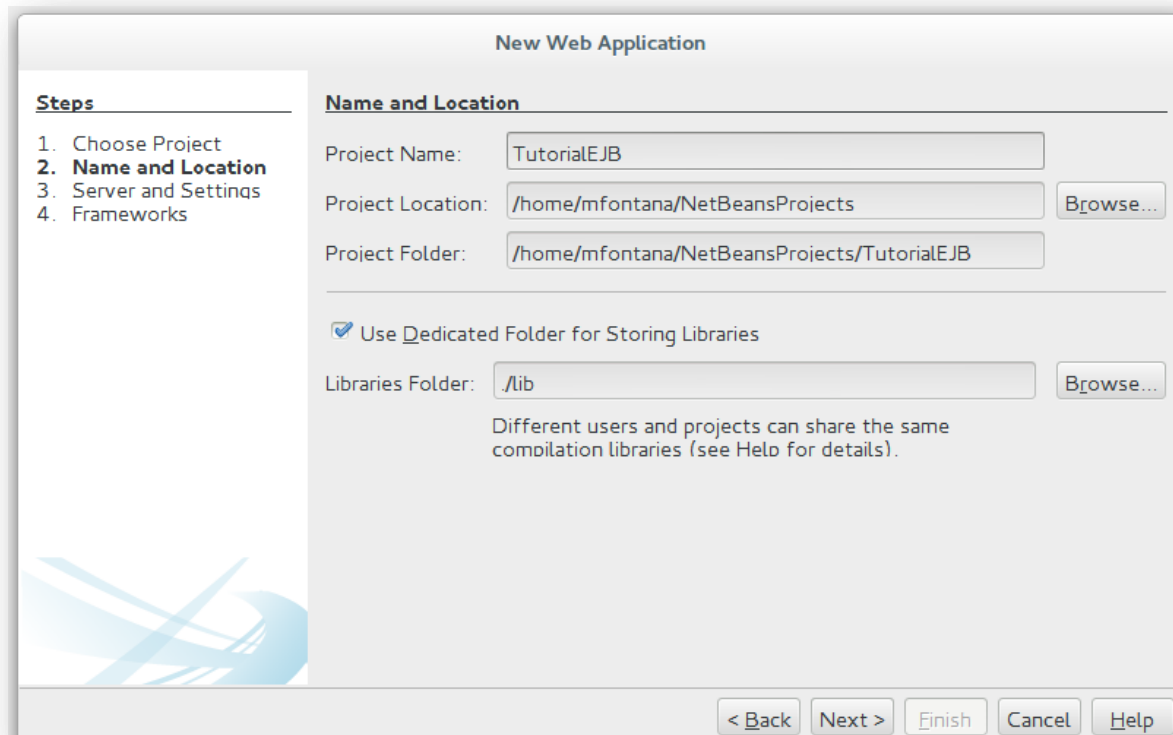
Creación del proyecto

- Nuevo Proyecto -> Dentro de la carpeta web se encuentra un tipo de proyecto llamado Web Application.



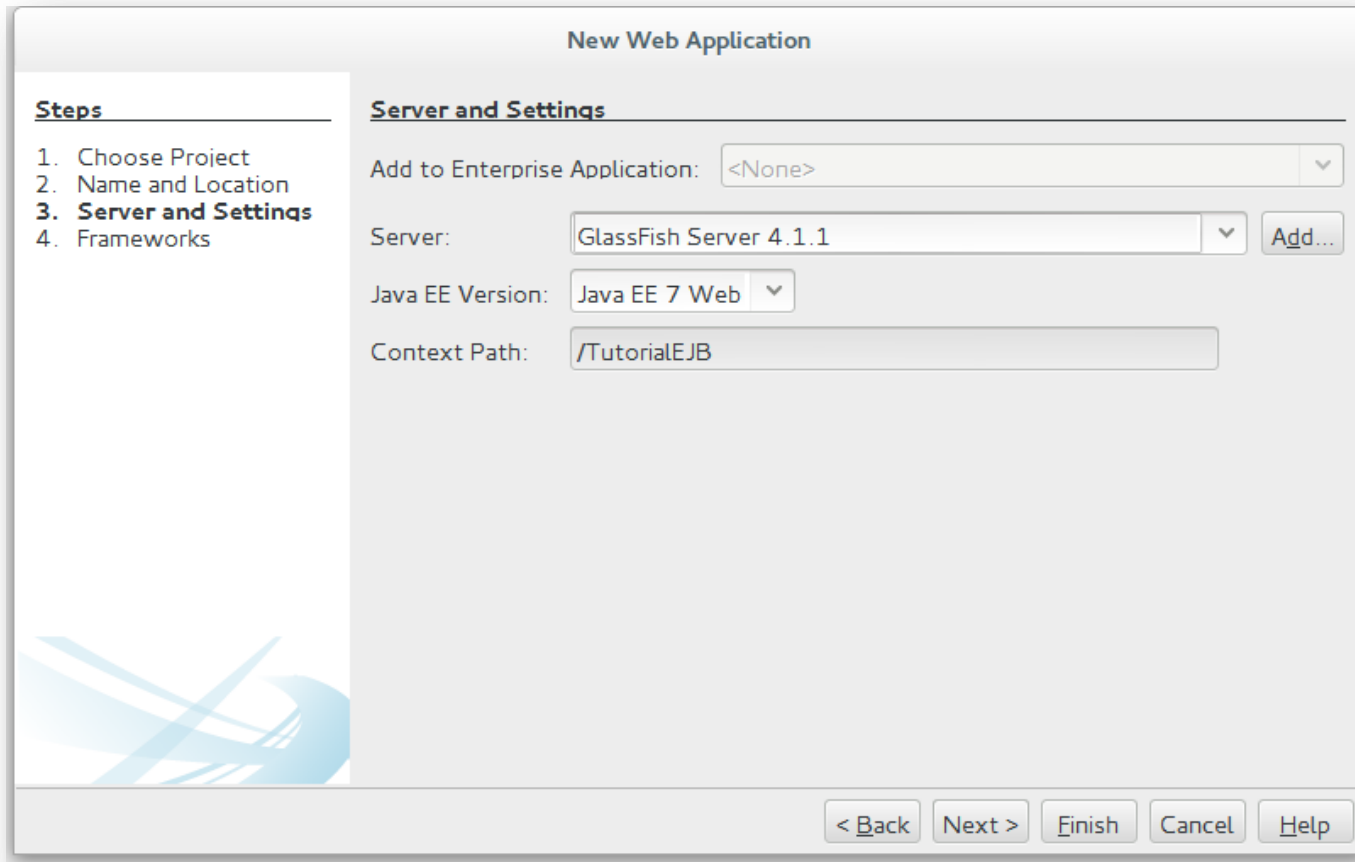
Creación del proyecto

- En el apartado dónde se indica el nombre del proyecto se debe marcar la opción Use Dedicated Folder for Storing Libraries y dejar por defecto la ruta .\lib.



Creación del proyecto

- Seleccionaremos GlassFish como servidor.



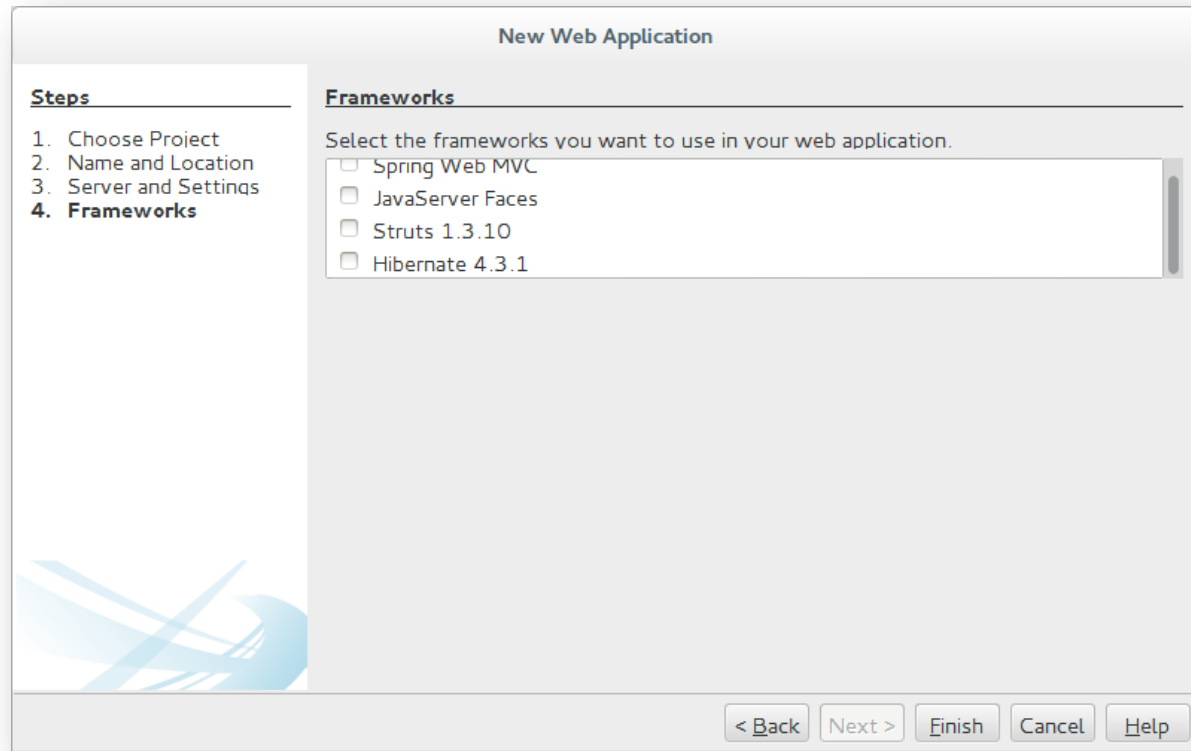
The screenshot shows the 'New Web Application' wizard in the NetBeans IDE. The window has a title bar 'New Web Application'. On the left, there is a 'Steps' panel with a list of four steps: 1. Choose Project, 2. Name and Location, 3. **Server and Settings** (highlighted), and 4. Frameworks. The main area is titled 'Server and Settings' and contains the following fields:

- 'Add to Enterprise Application:' with a dropdown menu showing '<None>'.
- 'Server:' with a dropdown menu showing 'GlassFish Server 4.1.1' and an 'Add...' button to the right.
- 'Java EE Version:' with a dropdown menu showing 'Java EE 7 Web'.
- 'Context Path:' with a text field containing '/TutorialEJB'.

At the bottom of the wizard, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

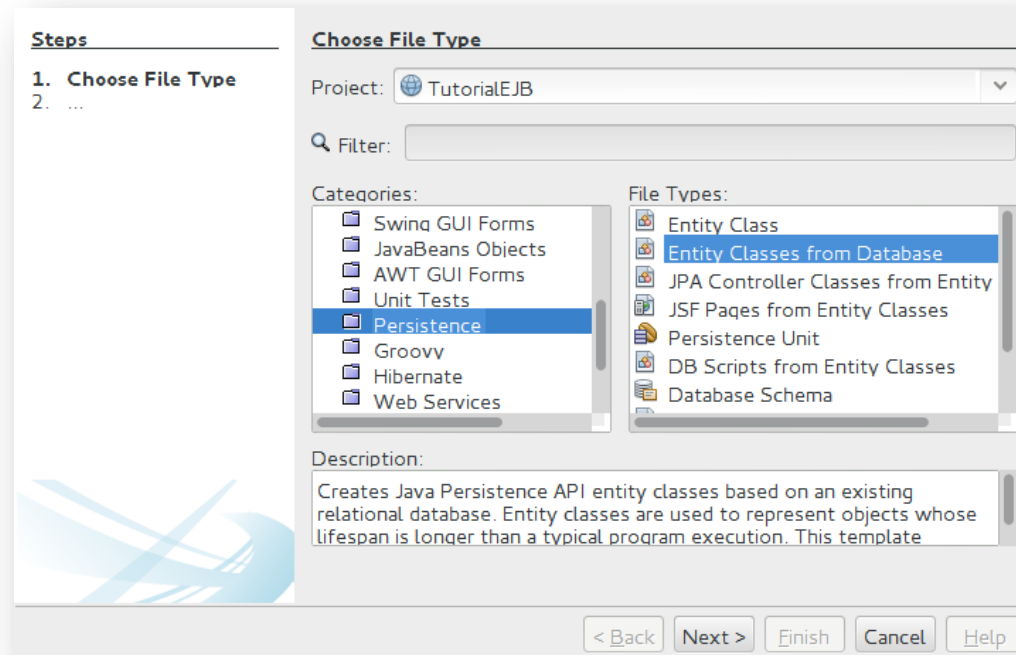
Creación del proyecto

- En el último paso podemos seleccionar un ORM como Hibernate. Si no seleccionamos ninguno JPA utiliza uno por defecto (EclipseLink).



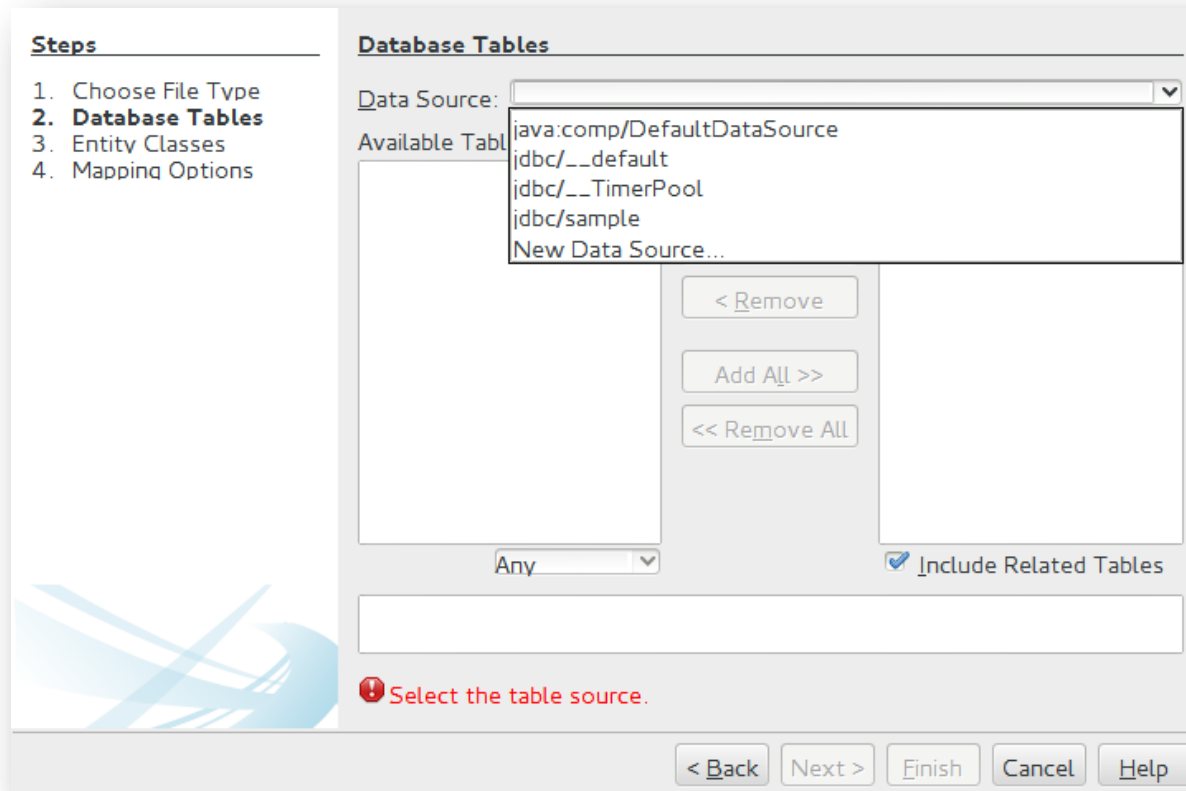
Soporte para JPA

- JPA permite la persistencia de objetos Java.
- En la raíz del proyecto, con el botón derecho del ratón y se selecciona Nuevo y dentro de la carpeta Persistencia -> Entity Classes from Database.



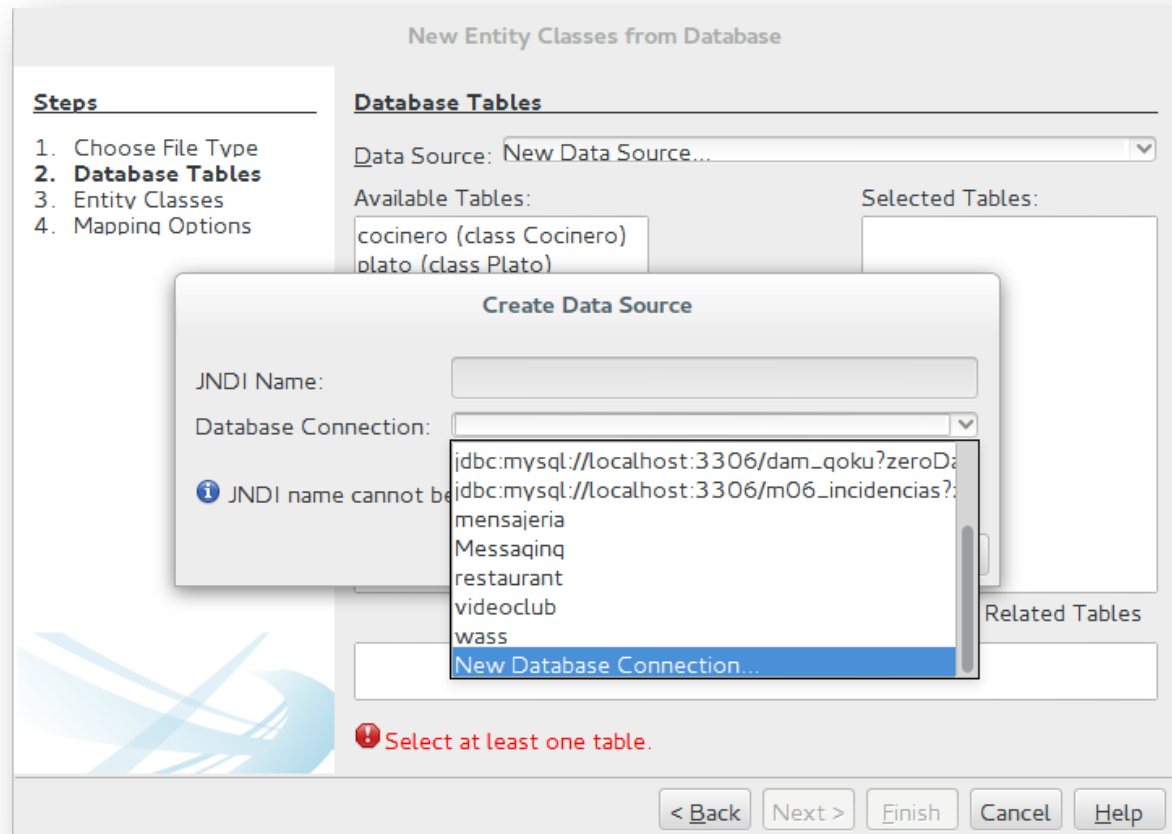
Soporte para JPA

- Se deberá escoger la conexión a la base de datos que se quiera acceder. Si no la tenemos, podemos crearla.



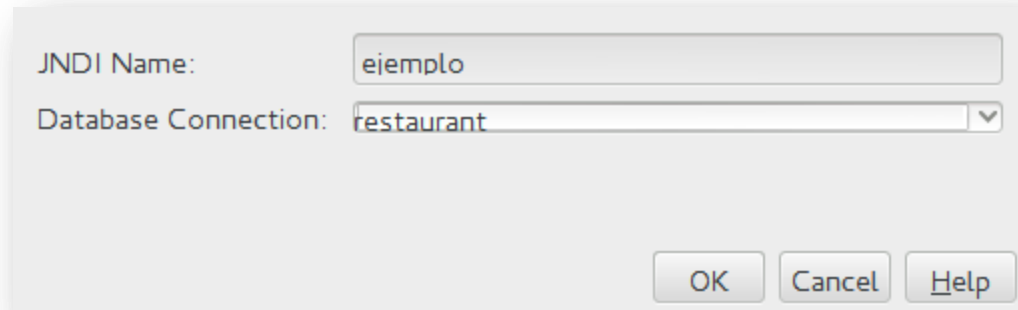
Soporte para JPA

- Al crearla se le indica un nombre (JDNI) y se selecciona la conexión.



Soporte para JPA

- Al crearla se le indica un nombre (JDNI) y se selecciona la conexión.

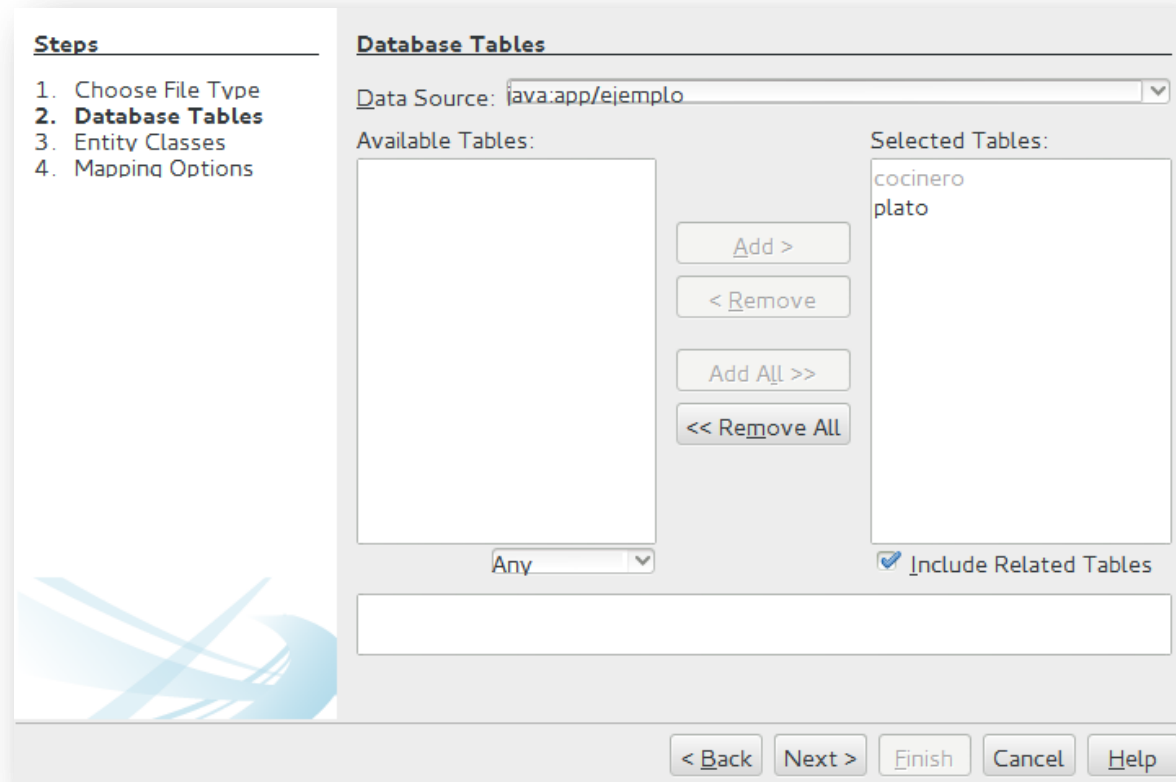


A screenshot of a configuration dialog box for JPA. It features two input fields: 'JNDI Name' with the text 'ejemplo' and 'Database Connection' with a dropdown menu showing 'restaurant'. At the bottom right, there are three buttons: 'OK', 'Cancel', and 'Help'.

JNDI Name:	ejemplo
Database Connection:	restaurant ▼
OK Cancel Help	

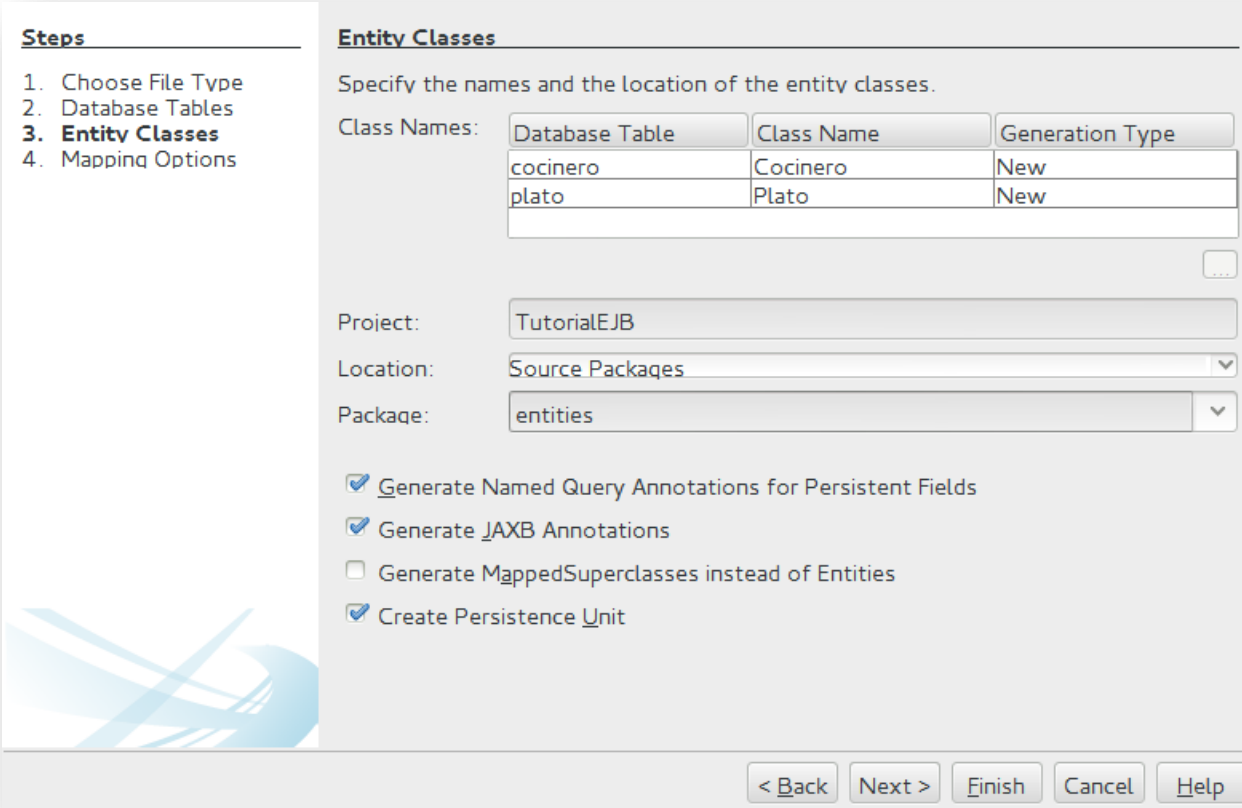
Soporte para JPA

- Seleccionar la/s tabla/s que se quieran añadir al proyecto.



Soporte para JPA

- Es recomendable escoger un package como entidades, por ejemplo.



Steps

1. Choose File Type
2. Database Tables
3. **Entity Classes**
4. Mapping Options

Entity Classes

Specify the names and the location of the entity classes.

Class Names:

Database Table	Class Name	Generation Type
cocinero	Cocinero	New
plato	Plato	New

Project: TutorialEJB

Location: Source Packages

Package: entities

☒ Generate Named Query Annotations for Persistent Fields

☒ Generate JAXB Annotations

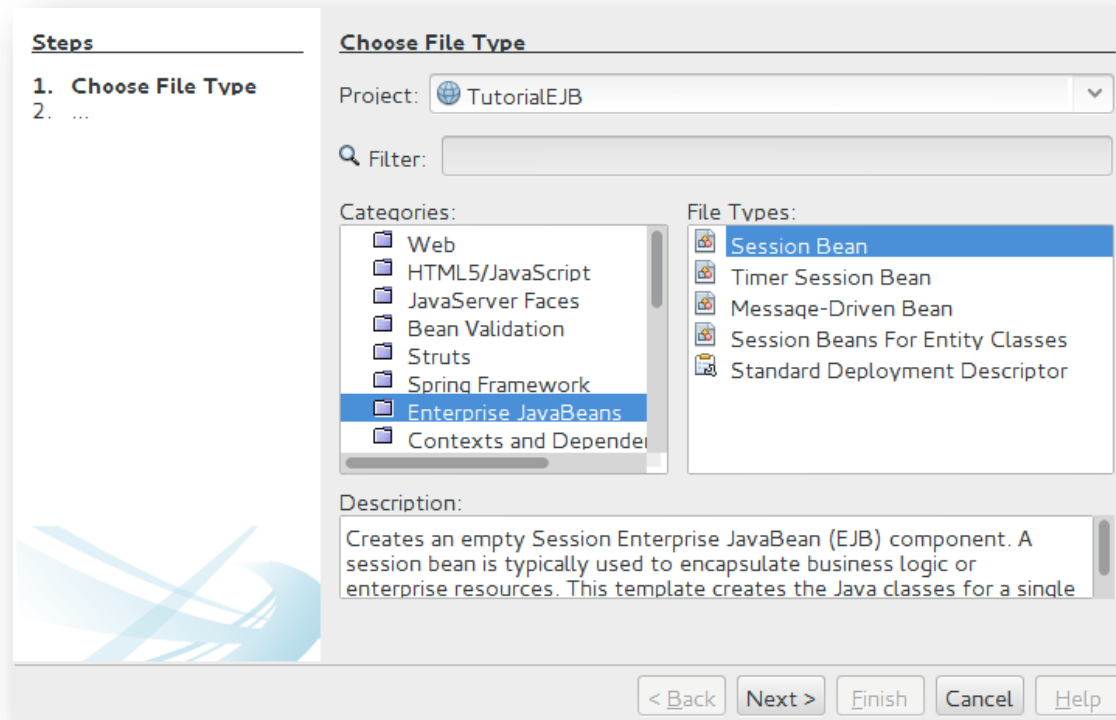
☐ Generate MappedSuperclasses instead of Entities

☒ Create Persistence Unit

< Back Next > Finish Cancel Help

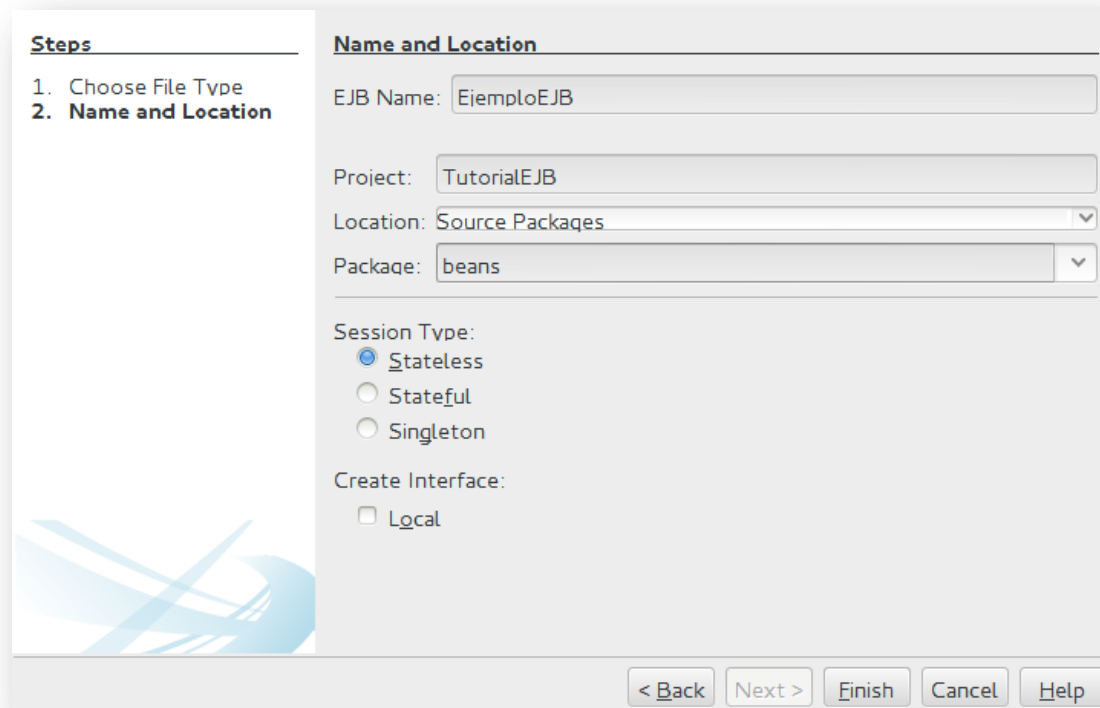
EJB de sesión sin estado

- En la raíz del proyecto, con el botón derecho del ratón se selecciona Nuevo y dentro de la carpeta Enterprise JavaBean se escoge Session Bean.



EJB de sesión sin estado

- Se le pone nombre al archivo, y se marca como tipo de sesión Stateless.
- Es recomendable guardarlo en el package beans.



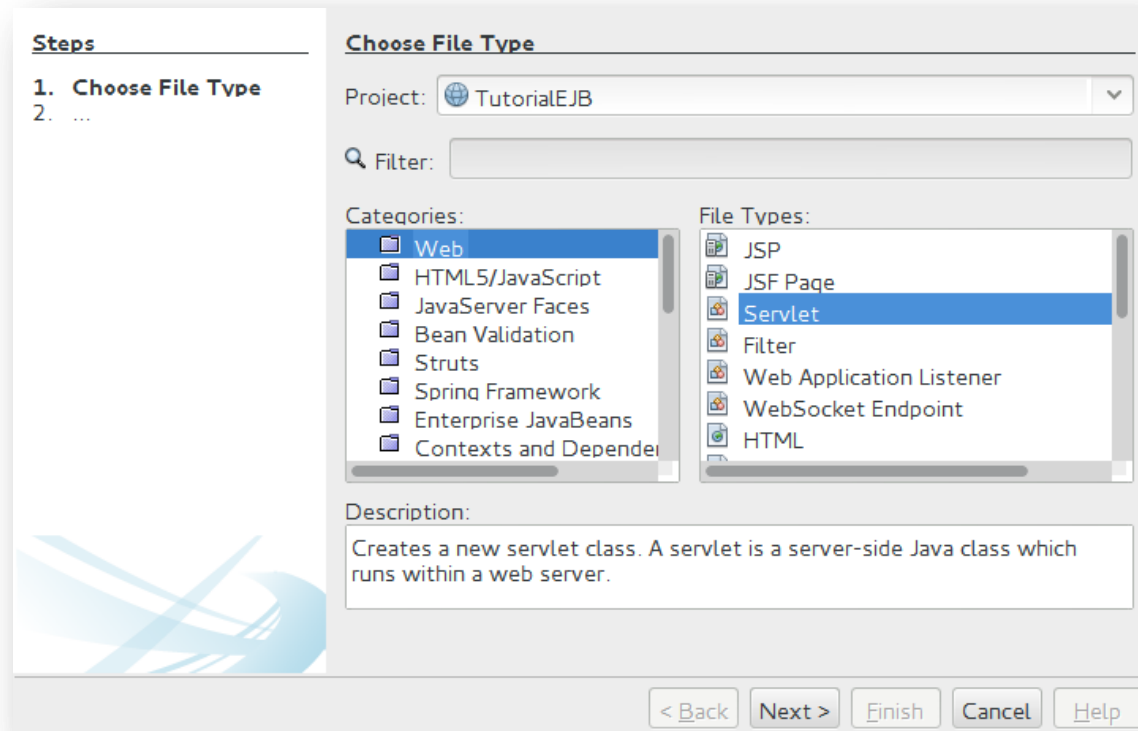
The screenshot shows a wizard dialog box for creating an EJB. On the left, a 'Steps' pane lists '1. Choose File Type' and '2. Name and Location'. The main area is titled 'Name and Location' and contains the following fields:

- EJB Name:** EjemploEJB
- Project:** TutorialEJB
- Location:** Source Packages (dropdown menu)
- Package:** beans (dropdown menu)
- Session Type:**
 - ☒ Stateless
 - ☐ Stateful
 - ☐ Singleton
- Create Interface:**
 - ☐ Local

At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

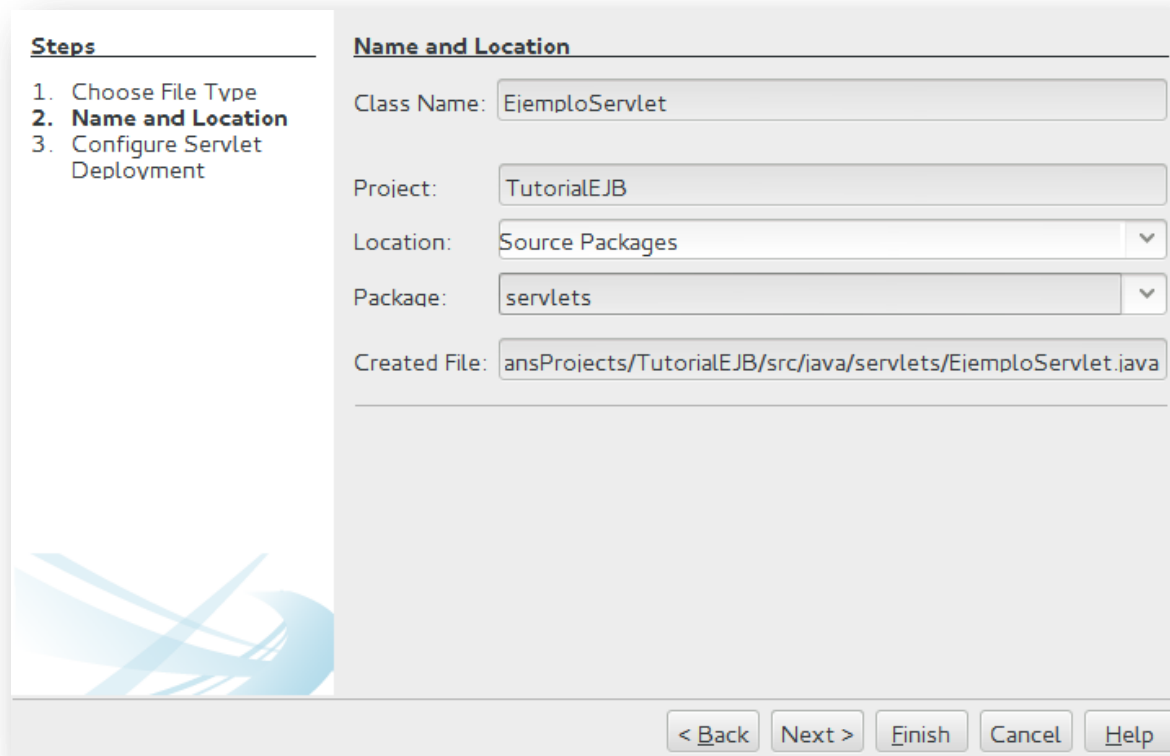
Creando el servlet

- En la raíz del proyecto, con el botón derecho del ratón se selecciona Nuevo y dentro de la carpeta Web se escoge el tipo de archivo Servlet.



Creando el servlet

- Se le pone nombre al servlet.
- Es recomendable guardarlo en el package servlets.



The screenshot shows a dialog box titled "Name and Location" with a "Steps" panel on the left. The "Steps" panel lists three steps: 1. Choose File Type, 2. Name and Location (which is the current step), and 3. Configure Servlet Deployment. The "Name and Location" panel contains the following fields:

- Class Name:** EjemploServlet
- Project:** TutorialEJB
- Location:** Source Packages (dropdown menu)
- Package:** servlets (dropdown menu)
- Created File:** ansProjects/TutorialEJB/src/java/servlets/EjemploServlet.java

At the bottom of the dialog box, there are five buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".