

Tema 3

JSF – Parte 1

Plataformas de Software Empresariales
Grado en Ingeniería Informática de Servicios y Aplicaciones
Curso 2020/2021

Introducción

▶ Objetivos

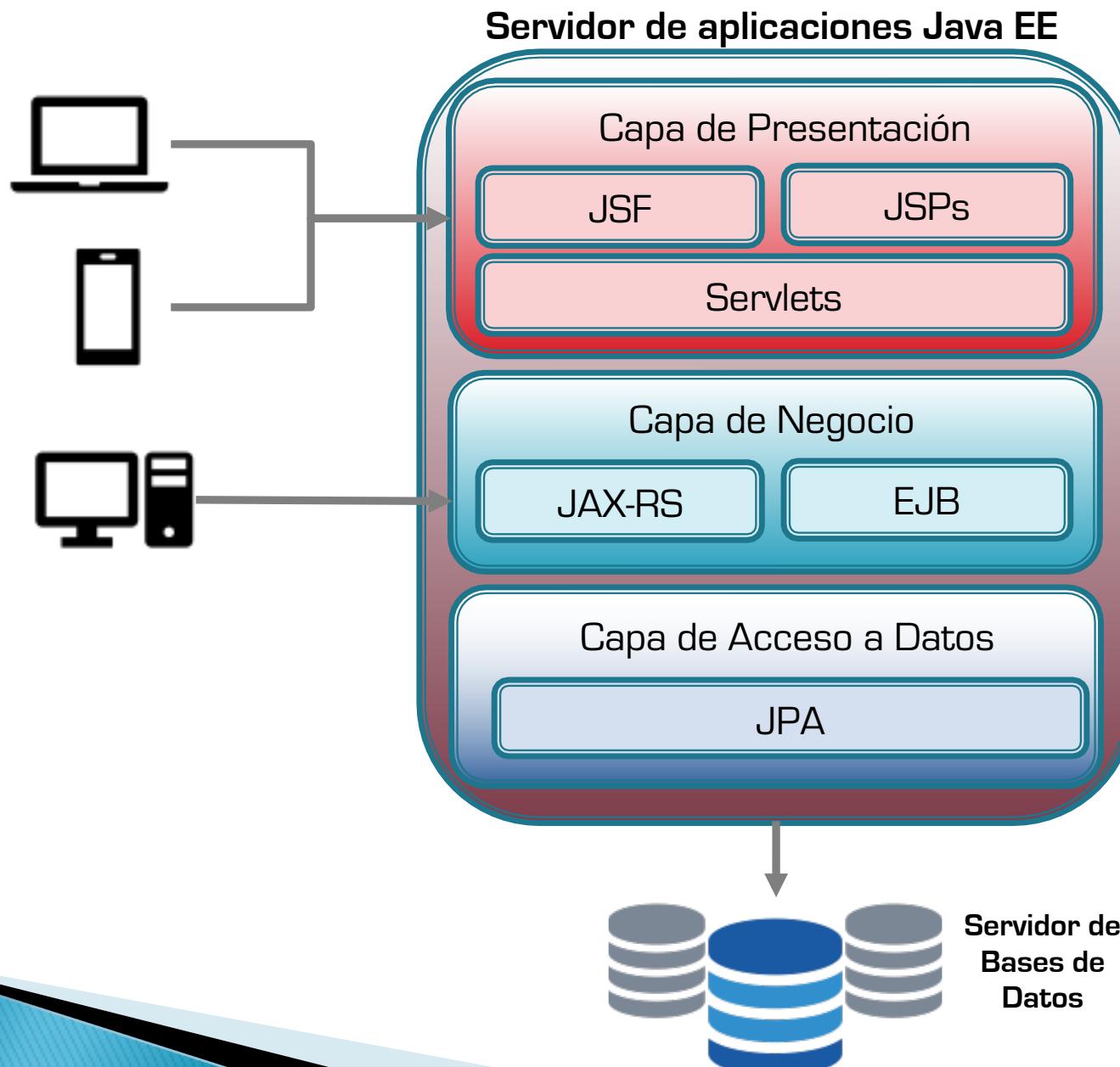
- Aprender qué es JSFs
- Aprender qué es Facelets
- Elementos fundamentales de una aplicación web MVC con JSFs y Facelets
- Templates en Facelets
- Recursos en Facelets
- Navegación en Facelets
- Extensiones de Facelets
- Aplicar todos estos conocimientos en el proyecto Cines Luz de Castilla

Introducción JSF

- ▶ JavaServer Faces (JSF) es...

Un **framework** del lado del servidor para desarrollar aplicaciones web Java **que proporciona distintos componentes para simplificar el desarrollo de interfaces de usuario**

Introducción JSF



Introducción JSF

- ▶ Es un estándar **claro y potente** para poder hacer aplicaciones visuales más potentes
- ▶ **Se trata de una tecnología que se ejecuta en el lado del servidor** y no en el lado del cliente
- ▶ Implementa Modelo-Vista-Controlador
- ▶ Al igual que Struts (evolucionó de él), JSF pretende normalizar y estandarizar el desarrollo de aplicaciones web... **pero Struts no es estándar** (aunque lo use mucha gente), no tiene componentes de IU, etc...

Introducción JSF

- ▶ ¿Qué es JSF?
 - Una especificación **e implementación** para desarrollo de aplicaciones web que proporciona:
 - Componentes de IU (ej: templates, calendarios deplegables, tablas,...)
 - En la práctica manejaremos otra librería más avanzada de componentes de IU para JSF
 - Manejo de eventos (ej: click, que haga algo al introducir texto (ej: validar calidad password), etc...)
 - Validadores y conversores de datos (ej: currency)
 - Navegación entre páginas
 - Integración con los datos del backend (EL)

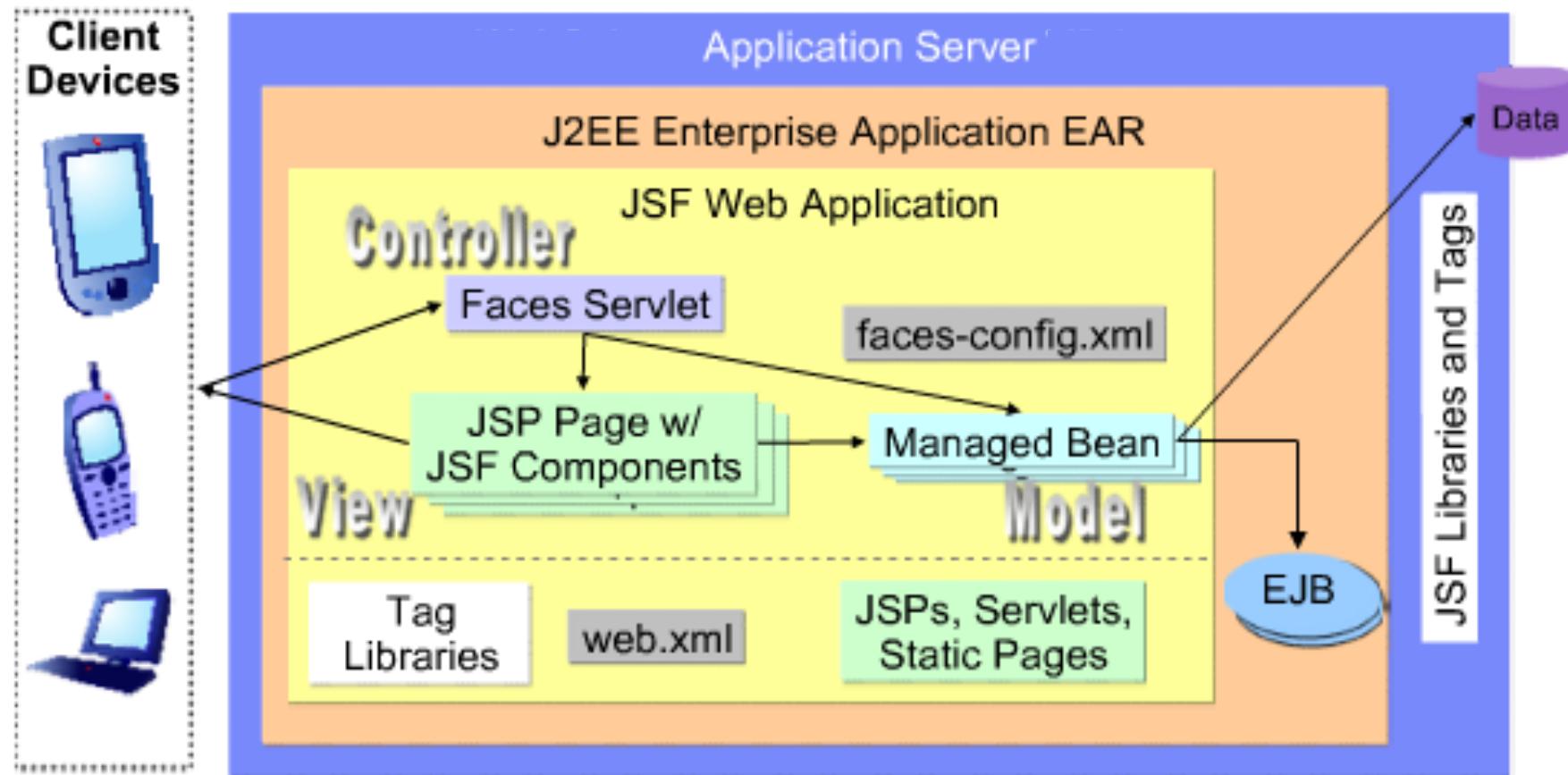
Introducción JSF

- ▶ **Inicialmente JSF usaba JavaServer Pages (JSP)** como la tecnología que permite hacer el despliegue de las páginas (ahora permite varias tecnologías y prima el uso de **Facelets**)
- ▶ JSF tiene **un Servlet como entrada** de las llamadas a su arquitectura (implementa MVC)
- ▶ JSF trata la vista (la interfaz de usuario) a través de **componentes y basada en eventos** (se pulsa un botón, cambia el valor de un campo, ...).

Introducción JSF - componentes JSF

- ▶ Una aplicación JSF está formada por los siguientes elementos:
 - Un **Servlet** (Faces Servlet)
 - Un conjunto de **páginas web** con componentes de la IU
 - Un conjunto de **Managed Beans** (o Backing Beans)
 - Se encargan de unir los componentes de la IU con un modelo del servidor (mediante CDI)
 - Un descriptor de despliegue opcional: `web.xml`
 - Un fichero de configuración opcional: `faces-config.xml`
 - Un conjunto opcional de objetos como conversores, listeners, etc... creados por el desarrollador

Introducción JSF - componentes JSF



Introducción JSF – MVC

- ▶ **Vista**
 - Conjunto de
 - ficheros JSP con las librerías de tags de JSF
 - **Facelets (ficheros xhtml – sustituyen a JSP)**
 - otros PDLs (Page Declaration Languages) [p.ej. XUL]
 - Describen la **jerarquía de componentes** JSF que conforman cada una de las páginas (pantallas) del interfaz de usuario de la aplicación.
 - Vinculan los componentes JSF con los **Managed Beans** (objetos de respaldo)
 - Se hace uso de la sintaxis del **Unified Expression Language (EL)** para referenciar los Managed Beans y sus atributos (`#{{objeto.atributo}}`)

Introducción JSF – MVC

► Modelo

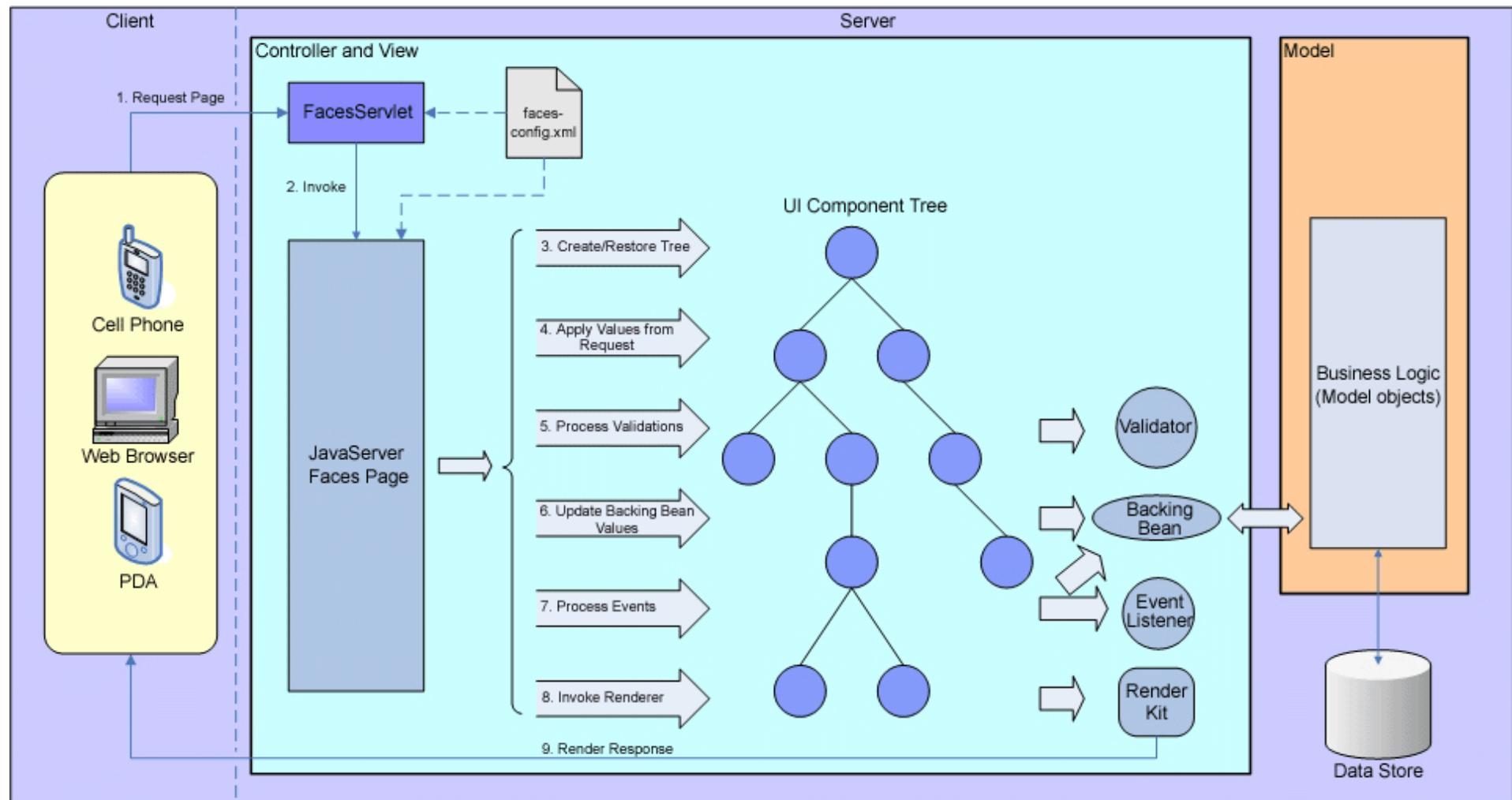
- Managed Beans
- Objetos Java (Java Beans) responsables de la lógica de negocio...
 - implementada completamente con los Managed Beans (deprecated)
 - delegando la lógica de negocio en componentes de negocio (EJBs, Beans SPRING, Servicios Web, etc,...) y dejando los managed/backing beans para almacenar la información de la interfaz
- Responden a los eventos generados por los componentes JSF

Introducción JSF – MVC

▶ Controlador

- Faces Servlet (configurado en faces-config.xml [opc. a partir de JSF 2.0]) + métodos de acción de los Managed Beans.
- Todas las peticiones HTTP del usuario pasan por el Faces Servlet
- Faces Servlet examina las peticiones recibidas, actualiza la representación del interfaz del cliente y los datos de los Managed Beans e invoca los manejadores de eventos y las acciones sobre el modelo a través de los métodos de los Managed Beans

Introducción JSF – Arquitectura



Introducción JSF

- ▶ Pasos del Proceso de Desarrollo de aplicaciones en JSF:
 - Crear las páginas utilizando las etiquetas de componentes UI y las etiquetas “core” (usando **Facelets**)
 - Desarrollar los objetos del modelo, los que contendrán los datos (**ManagedBeans/Backing Beans**)
 - Definir la navegación entre las páginas
- ▶ Estas tareas se pueden realizar simultáneamente o en cualquier orden.

Facelets

- ▶ Facelets es el lenguaje de declaración de la vista para JSF
- ▶ Reemplaza a JSP
- ▶ Además de los componentes estándar, a partir de la versión 2 se incluyó la posibilidad de crear componentes compuestos y añadir Ajax
- ▶ Principales ventajas: sistema de templates muy potente (y sencillo), fácil desarrollo y reutilización, reporte de errores, etc...

Facelets

- ▶ Ejemplo de página Facelets usando xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>My Facelet Page Title</title>
    </h:head>

    <h:body>
        Hello from Facelets
    </h:body>
</html>
```

Facelets

- ▶ En el ejemplo anterior se combinan tags html y tags html de Facelets (los que comienzan con h:)
- ▶ Es necesario declarar la librería de tags:

`xmlns:h="http://xmlns.jcp.org/jsf/html"`

- ▶ Una vez hecho esto ya podemos utilizar todos los tags de la librería html

Facelets – Librerías de tags estandar

- ▶ **h:** <http://xmlns.jcp.org/jsf/html>
 - h:head, h:inputText, ...
- ▶ **f:** <http://xmlns.jcp.org/jsf/core>
 - f:facet, f:actionListener, ...
- ▶ **c:** <http://xmlns.jcp.org/jsp/jstl/core>
 - c:forEach, c:if, ...
- ▶ **fn:** <http://xmlns.jcp.org/jsp/jstl/functions>
 - fn:toUpperCase, fn:contains, ...
- ▶ **ui:** <http://xmlns.jcp.org/jsf/facelets>
 - ui:component, ui:insert, ...
 - (buscad en Internet para más información sobre cada librería de tags
– he dejado una “chuleta” en Campus Virtual)

Facelets – EL

- ▶ Facelets también proporciona integración con el Lenguaje de Expresión (EL)
- ▶ Esto permite la “conexión” bi-direccional entre los beans y la interfaz de usuario
- ▶ Supongamos que tenemos el siguiente bean con información del usuario:

```
public class Name {  
    private String value;  
  
    public String getValue() {  
        return value;  
    }  
  
    public void setValue(String value) {  
        this.value = value;  
    }  
}
```

Facelets – EL

- ▶ Podemos mostrar, en la interfaz de usuario, la información almacenada en el bean “Name” de la siguiente manera

```
...
<body>
    Hello, my name is #{name.value}!
</body>
```

- ▶ **#{name.value}** es un EL que se refiere al método “**getValue**” del **backing bean “Name”**
 - Es importante respetar las mayúsculas/minúsculas si queremos utilizar los nombres CDI por defecto
- ▶ Aunque para ello tenemos que modificar ligeramente el bean

Facelets – EL

```
@Named  
@RequestScoped  
public class Name {  
    private String value; //... getValue...  
}
```

- ▶ Es importante añadir **@Named** para permitir que el bean sea accesible mediante EL
- ▶ **@RequestScoped** (Ámbito de Petición) se utiliza para indicar que las instancias solamente existen durante la petición HTTP. Una vez que termina el ciclo de vida de la petición, también termina su contexto.

Facelets – EL

- ▶ Cuidado a la hora de declarar y llamar a los beans:
 - Java EE distingue entre mayúsculas y minúsculas
 - Tenemos dos opciones:
 - Las clases se declaran comenzando en mayúscula y se llaman con minúscula → es la convención en Java EE

```
...  
<body>  
    Hello from Facelets, my name is #{name.value}!  
</body>  
...
```

```
@Named  
@RequestScoped  
public class Name {  
    private String value; //... getValue...  
}
```

- Si lo hacemos así podemos usar la anotación **@Named** sin especificar un nombre

Facelets – EL

- ▶ Cuidado a la hora de declarar y llamar a los beans:
 - Java EE distingue entre mayúsculas y minúsculas
 - Tenemos dos opciones:
 - O bien se define el nombre que queremos utilizar en el Named



```
@Named("name")
@RequestScoped
public class Name {
    private String value; //... getValue...
}
```

- A partir de este momento, la clase Name se introduce en CDI con el nombre “name” (o cualquier otro que queramos)

Facelets – EL

- ▶ Del mismo modo, un EJB también puede ser injectado en una expresión EL

```
@Stateless  
@Named  
public class CustomerSessionBean {  
    public List<Name> getCustomerNames() {  
        //...  
    }  
}
```

- ▶ Este EJB sin estado tiene un método de la lógica de negocio que devuelve una lista de todos los nombres de los clientes

Facelets – EL

- ▶ Para usar el EJB dentro de Facelets:

```
<h:dataTable value="#{customerSessionBean.customerNames}" var="c">
    <h:column>#{c.value}</h:column>
</h:dataTable>
```

- ▶ Vemos con este ejemplo qué fácilmente se puede acceder a la lógica de negocio, obtener toda la lista de clientes, y mostrarlo mediante una tabla

Facelets – Templates

The image shows a digital menu board for McDonald's. At the top, five different burgers are displayed: McPollo*, Cuarto de Libra, Big Mac*, McRoyal Deluxe*, and CBO*. Below them, the text "McMENÚ® CLÁSICOS" is prominently displayed. A table of prices for the McMENÚ is provided, along with options to add complements or upgrade to a large meal.

McMENÚ*	con Patatas o Ensalada de la Huerta y Bebida
Big Mac*	6,10€
McPollo*	6,10€
Cuarto de Libra	6,10€
McRoyal Deluxe*	6,10€
CBO*	6,75€
9 McNuggets*	6,10€
McWrap*	6,40€

AÑADE UN COMPLEMENTO POR
+1€ Hamburguesa con Queso
Café con Helado
+2€ 6 McNuggets*
Le Grand Plaisir

HAZLO GRANDE POR
+0,60€

Facelets – Templates



Facelets – Templates



Facelets – Templates

- ▶ Facelets proporciona un sistema de templates muy potente que permite obtener un look and feel consistente a lo largo de múltiples páginas
- ▶ La **página básica, llamada template**, se crea usando los tags para templates de Facelets
- ▶ El template **define la estructura por defecto de la página**, incluyendo los lugares donde irá el contenido que se definirá en las distintas páginas

El template es la creación de la bandeja con los compartimentos

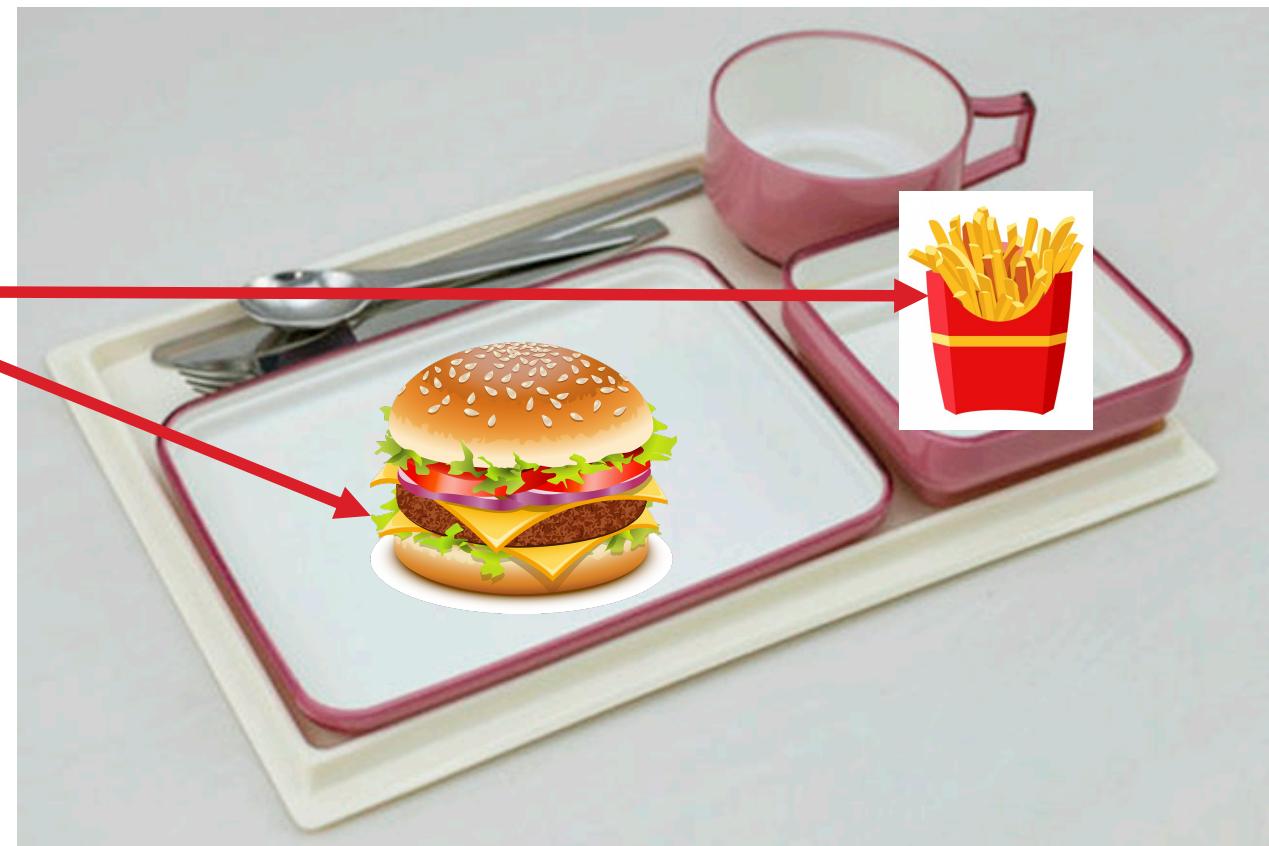


Facelets – Templates

- ▶ Una **página cliente del template** utiliza el template y proporciona contenido para todas o alguna de las partes del template

La página cliente sólo tiene que definir estos dos componentes

(es decir, cojo una bandeja que ya tengo hecha y le añado únicamente una hamburguesa y patatas fritas)

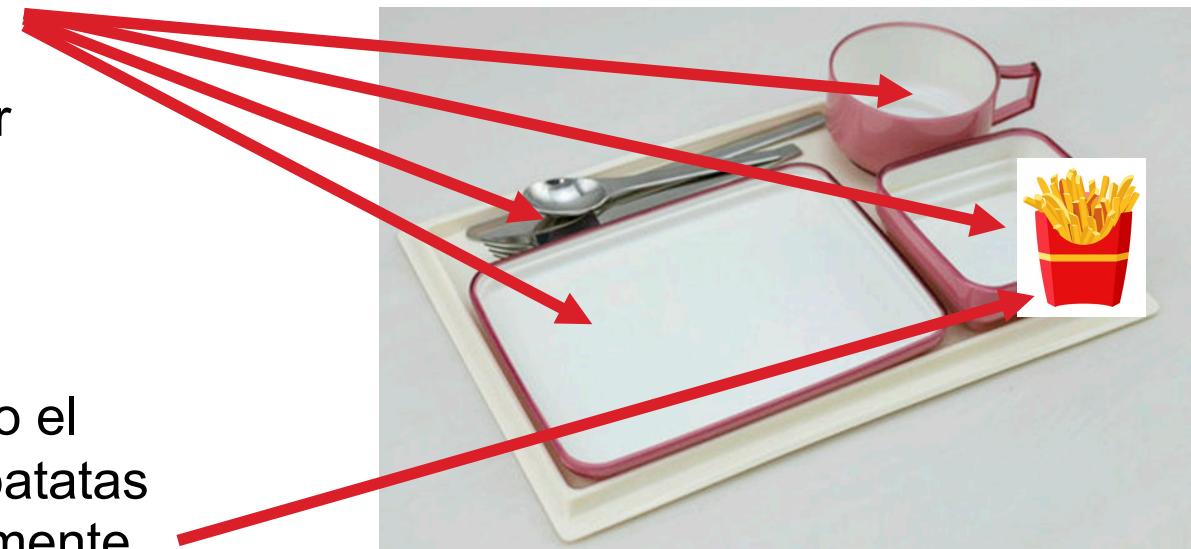


Facelets – Templates

- ▶ Los principales tags para templates de Facelets son los siguientes:
 - **ui:insert**: usado en la [página template](#); declara un contenido nombrado que debe ser definido en otra página.

ui:insert define los
compartimentos
que vamos a tener

Además, como todo el
mundo va a pedir patatas
fritas, pues directamente
las pongo en su lugar de
la bandeja



Facelets – Templates

- ▶ Los principales tags para templates de Facelets son los siguientes:
 - **ui:composition**: usado en la [página cliente](#); envuelve un conjunto de componentes para ser reutilizados en otra página, es la etiqueta que:
 - envuelve o puede envolver la plantilla.
 - se utiliza para hacer referencia a una plantilla.
 - todo lo que quede fuera de la etiqueta ui:composition no será renderizado.

ui:composition dice... en la página web que estoy creando quiero utilizar esta bandeja que has creado antes



Facelets – Templates

- ▶ Los principales tags para templates de Facelets son los siguientes:
 - **ui:define**: usado en una [página cliente](#); define el contenido que reemplaza el contenido definido en el template con una etiqueta ui:insert con el mismo nombre

ui:define añade la comida elegida por cada uno...
pero sólo hace eso...
coge la bandeja con las secciones indicada en ui:composition y pone la hamburguesa... NO
tienes que crear la bandeja desde cero ni añadir las patatas fritas!!



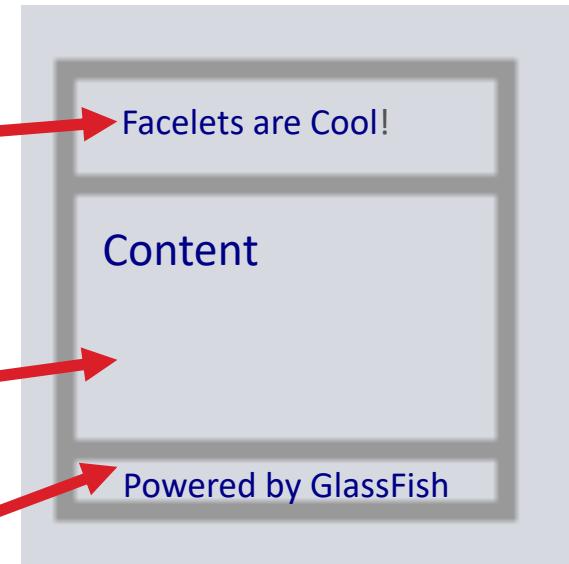
Facelets – Templates

- ▶ Los principales tags para templates de Facelets son los siguientes:
 - Tenemos otros componentes...
 - ui:include: es una etiqueta que sirve para incluir en una página el contenido de otra, como si el contenido de esta última formase parte de la primera.
 - ui:component, ui:fragment, etc...

Facelets – Templates

- ▶ Ejemplo de página template (`template.xhtml`):

```
<h:body>
    <div id="top">
        <ui:insert name="top">
            <h1>Facelets are Cool!</h1>
        </ui:insert>
    </div>
    <div id="content" class="center_content">
        <ui:insert name="content">
            Content
        </ui:insert>
    </div>
    <div id="bottom">
        <ui:insert name="bottom">
            <center>Powered by GlassFish</center>
        </ui:insert>
    </div>
</h:body>
```

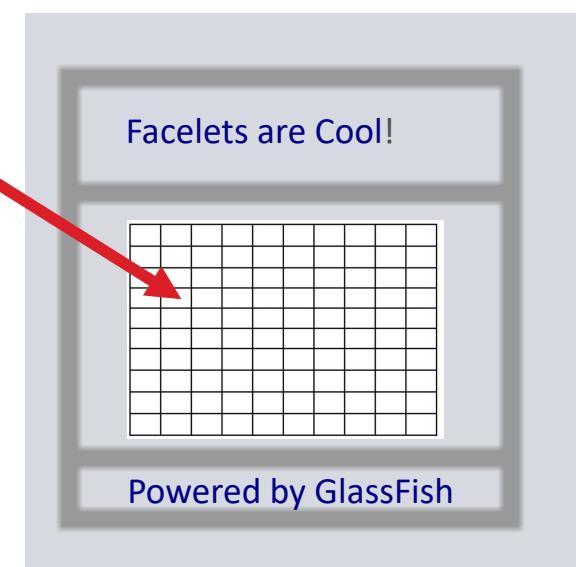
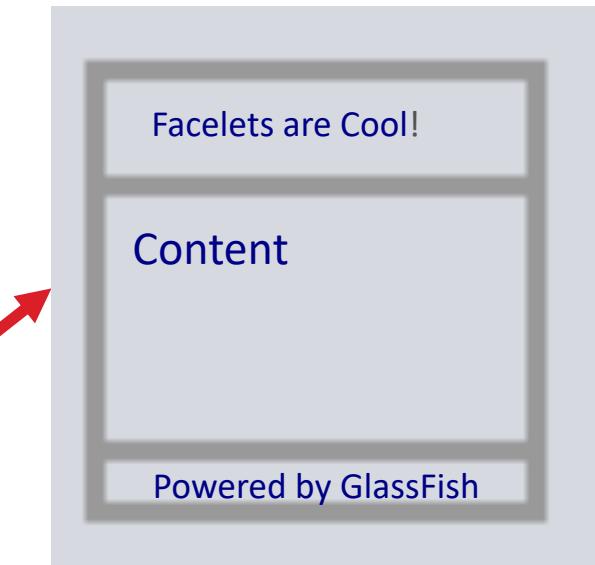


Facelets – Templates

- ▶ Ejemplo de cliente de template:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

    <body>
        <ui:composition template=".template.xhtml">
            <ui:define name="content">
                <h: dataTable value="#{customerSessionBean.customerNames}" var="c">
                    <h:column>#{c.value}</h:column>
                </h: dataTable>
            </ui:define>
        </ui:composition>
    </body>
</html>
```



Facelets – Recursos

- ▶ JSF define un mecanismo estándar para manejar recursos (imágenes, CSS, ficheros JavaScript, etc...)
- ▶ Los recursos tienen que estar en el directorio /resources de la aplicación web (o en /META-INF/resources del classpath)
- ▶ Los recursos son referenciados mediante EL:
``
- ▶ O si está dentro de una librería (corp):
`<h:graphicImage library="corp" name="header.jpg" />`

Facelets – Recursos

- ▶ Para incluir JavaScript:

```
<h:outputScript  
    name="myScript.js"  
    library="scripts"/>
```

- ▶ myScript.js es un recurso JavaScript empaquetado dentro del directorio scripts del directorio resources

Facelets – Recursos

- ▶ Para incluir CSS:

```
<h:outputStylesheet name="myCSS.css" library="css" />
```

Facelets – Navegación

- ▶ La navegación se puede hacer de manera implícita y explícita:
 - **Implícita:** buscan directamente la salida de una acción (ej: click en un link o un botón)

`<h:commandButton action="login" value="Login"/>`
 - Pulsando el botón se irá a la página login.xhtml dentro del directorio actual

Facelets – Navegación

- **Explícita:** se pueden definir reglas de navegación explícitas usando `<navigation-rule>` dentro del fichero faces-config.xml

```
<navigation-rule>
    <from-view-id>/index.xhtml</from-view-id>
    <navigation-case>
        <from-action>login</from-action>
        <from-outcome>success</from-outcome>
        <to-view-id>/login.xhtml</to-view-id>
        <if>#{user.isPremium}</if>
    </navigation-case>
    ...
</navigation-rule>
```

Cuando en la página index.xhtml se presione el botón con el action igual a “login” entonces irá a login.xhtml SI el usuario es premium

En el siguiente tema veremos Faces Flow para casos más complejos!!

Introducción JSF – Extensiones

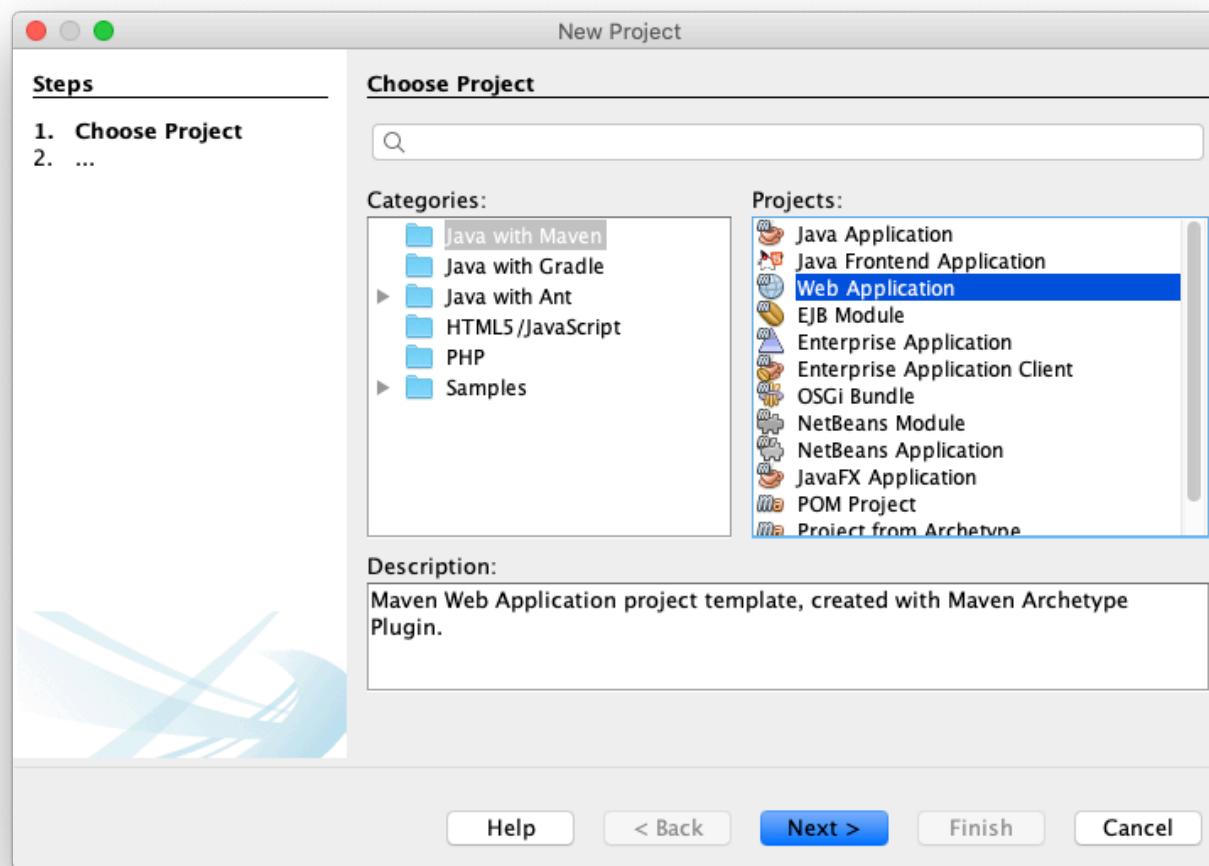
- ▶ Algunas extensiones de JSF son:
 - RichFaces: Agrega componentes visuales y soporte para AJAX (desarrollo detenido en 2016)
 - ICEfaces: Contiene diversos componentes para interfaces de usuarios más enriquecidas, tales como editores de texto enriquecidos, reproductores de multimedia, entre otros
 - jQuery4jsf: Contiene diversos componentes sobre la base de uno de los más populares framework javascript jQuery
 - **PrimeFaces: Es una librería muy liviana y de las más completas que existe. Proporciona muchas extensiones sencillas de implementar.**
 - OpenFaces: Librería open source que contiene diferentes componentes JSF, un Framework Ajax y un Framework de validación por parte del cliente
 - Otros muchos: GISFaces, etc...

Proyecto de desarrollo

Aplicación Empresarial para los
Cines Luz de Castilla (Segovia)

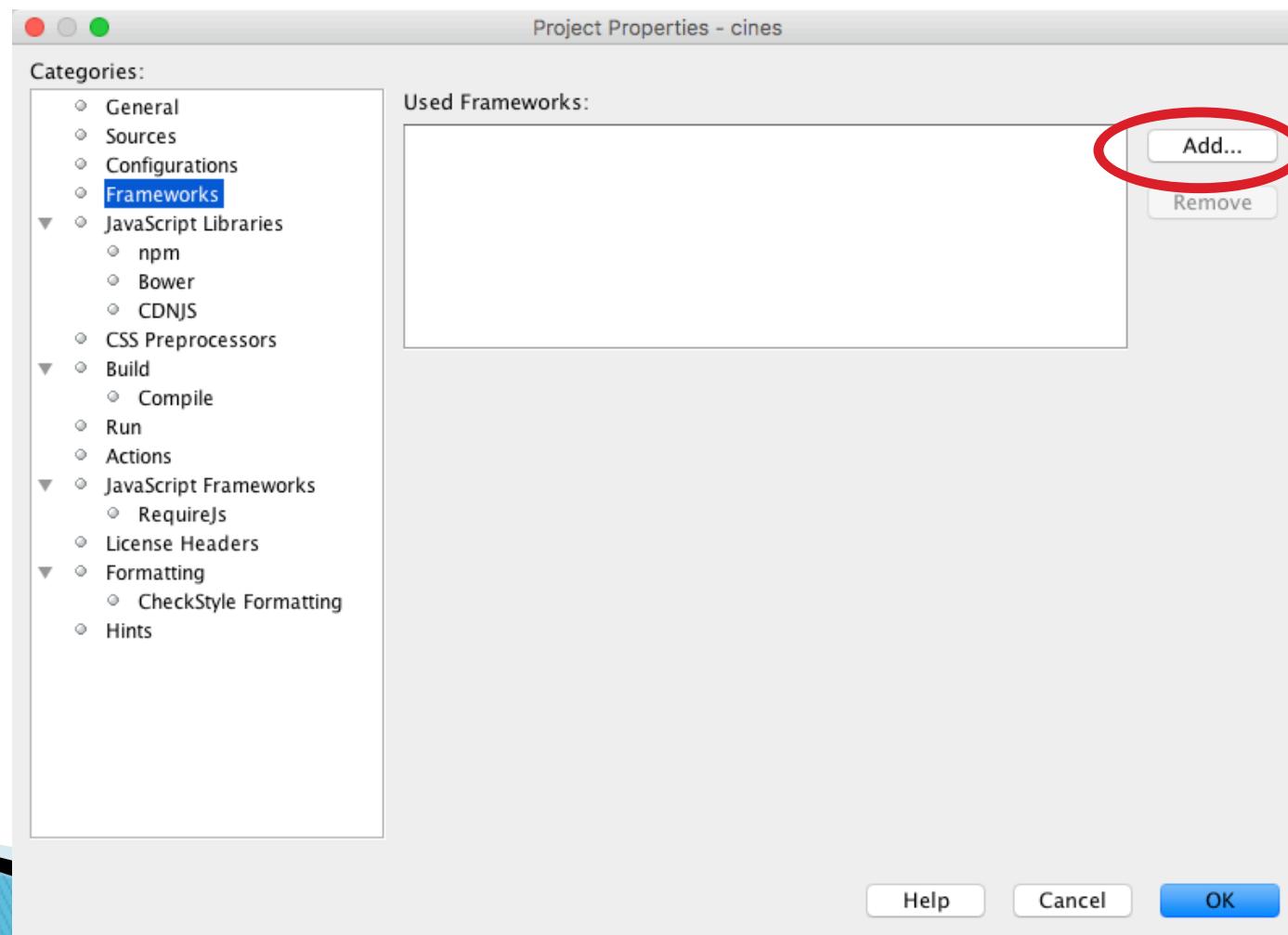
Proyecto Cines Luz de Castilla

- ▶ Creamos nuevo proyecto web (**hacedlo como ya vimos en el tema de Git**)



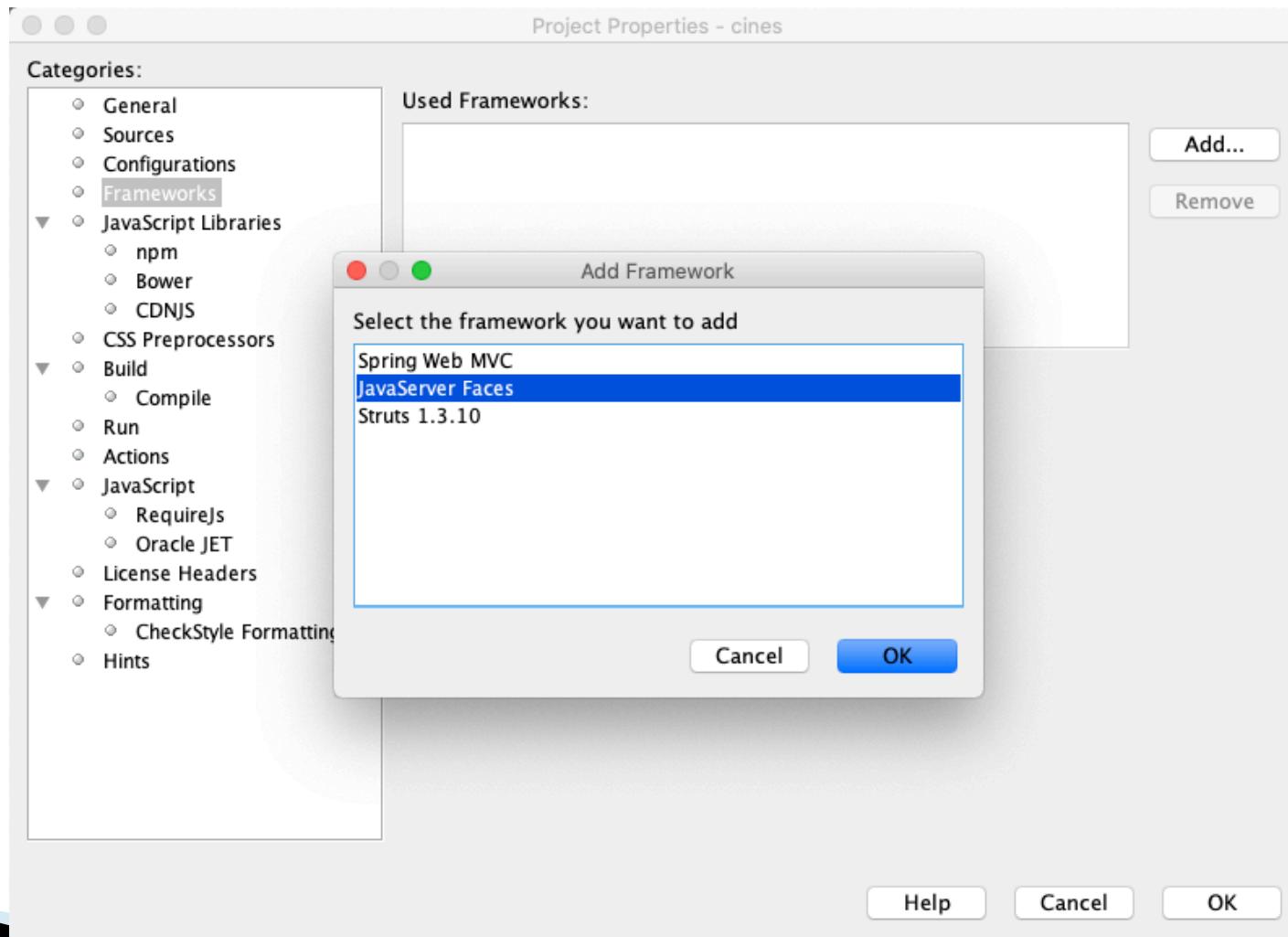
Añadimos Framework JSF

- ▶ Botón derecho sobre la carpeta del proyecto → Properties → Frameworks → Add... → JavaServer Faces



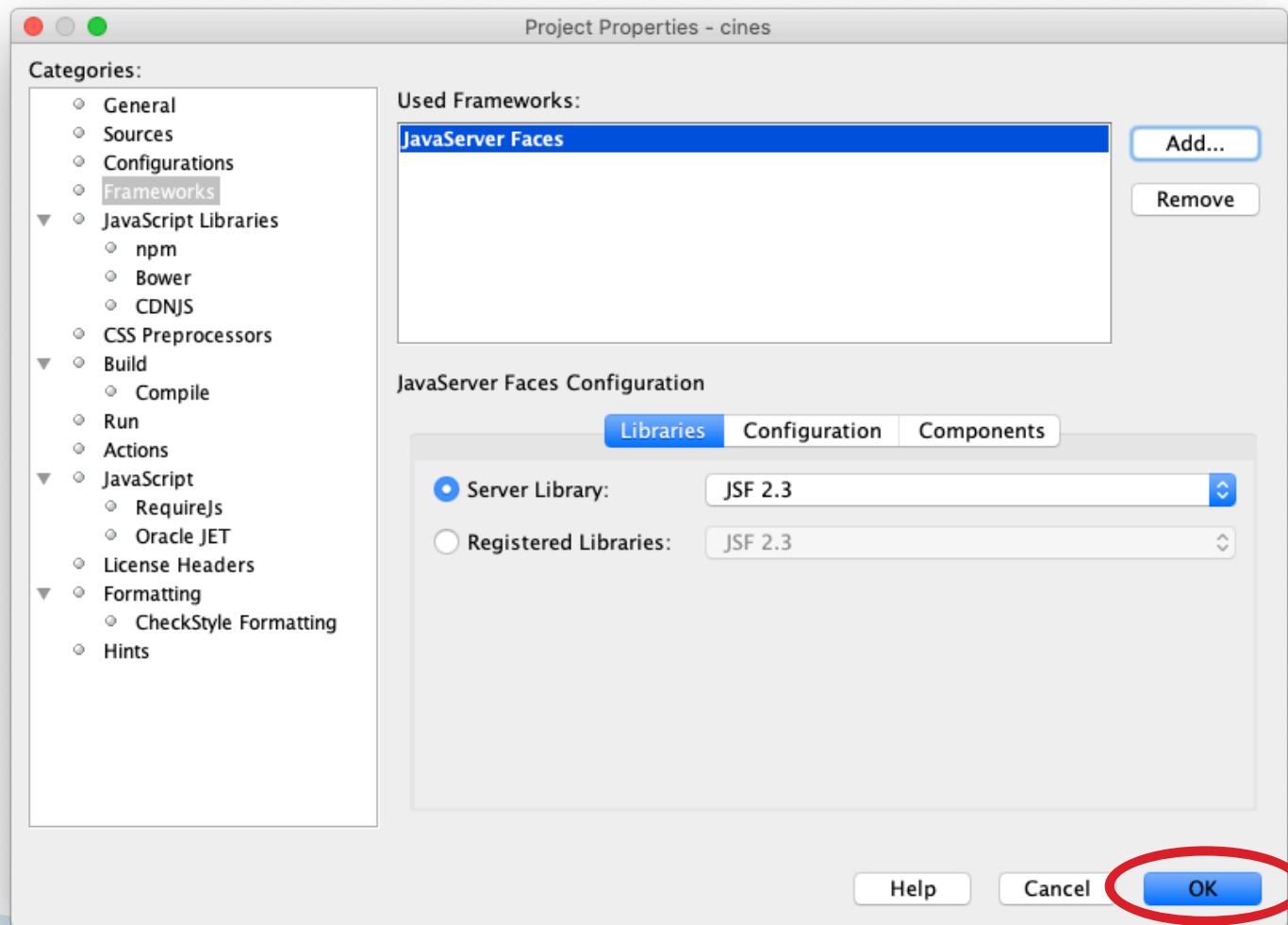
Añadimos Framework JSF

▶ Añadimos JavaServer Faces



Añadimos Framework JSF

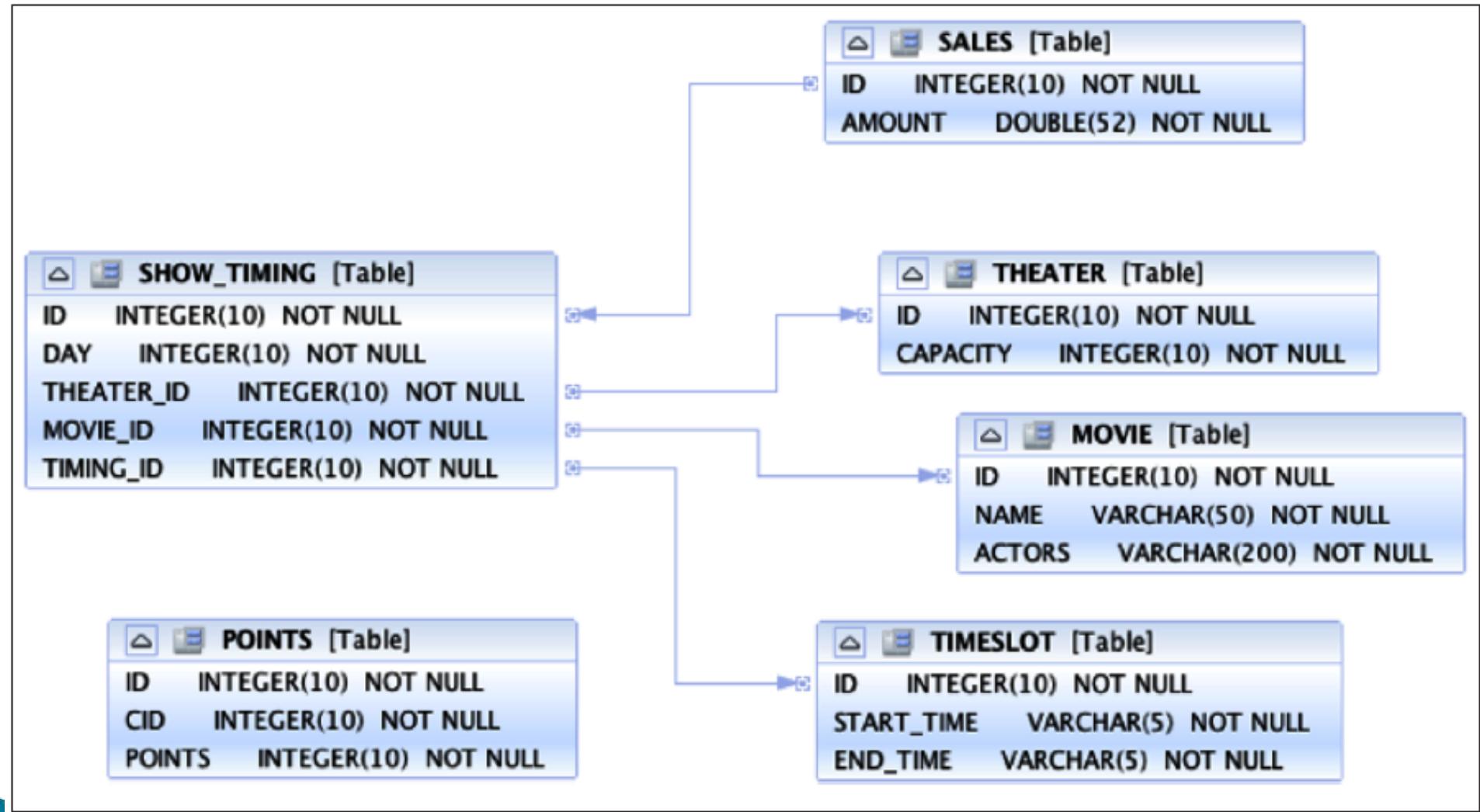
▶ Versión JSF 2.3



Añadimos Framework JSF

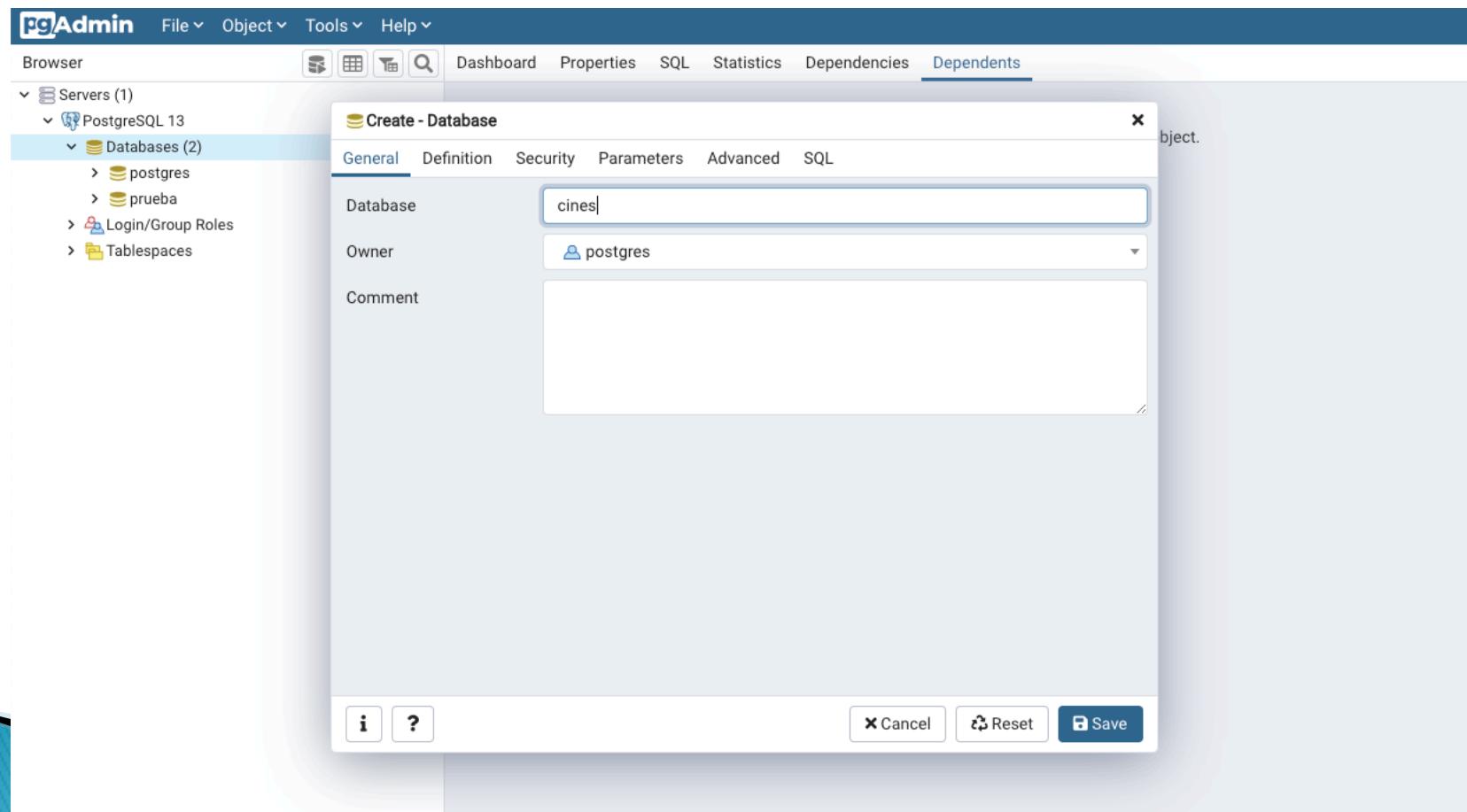
- ▶ Entre otras cosas nos va a crear el fichero index.xhtml
- ▶ Podemos eliminar el index.html que teníamos de antes
- ▶ Podemos ejecutar y comprobar que funciona
 - Tiene que decir “Hello from Facelets”

Base de datos (versión simplificada)



Base de datos

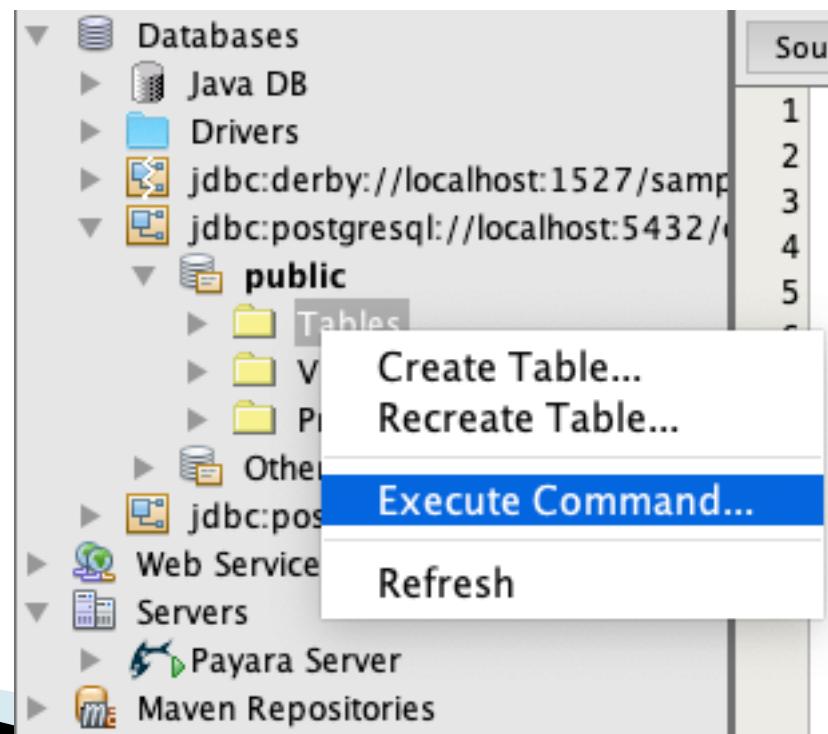
- ▶ Creamos la base de datos como hicimos en el tema anterior (en este caso todos **llamadla “cines”, aquí no tenéis que poner el número del grupo**).



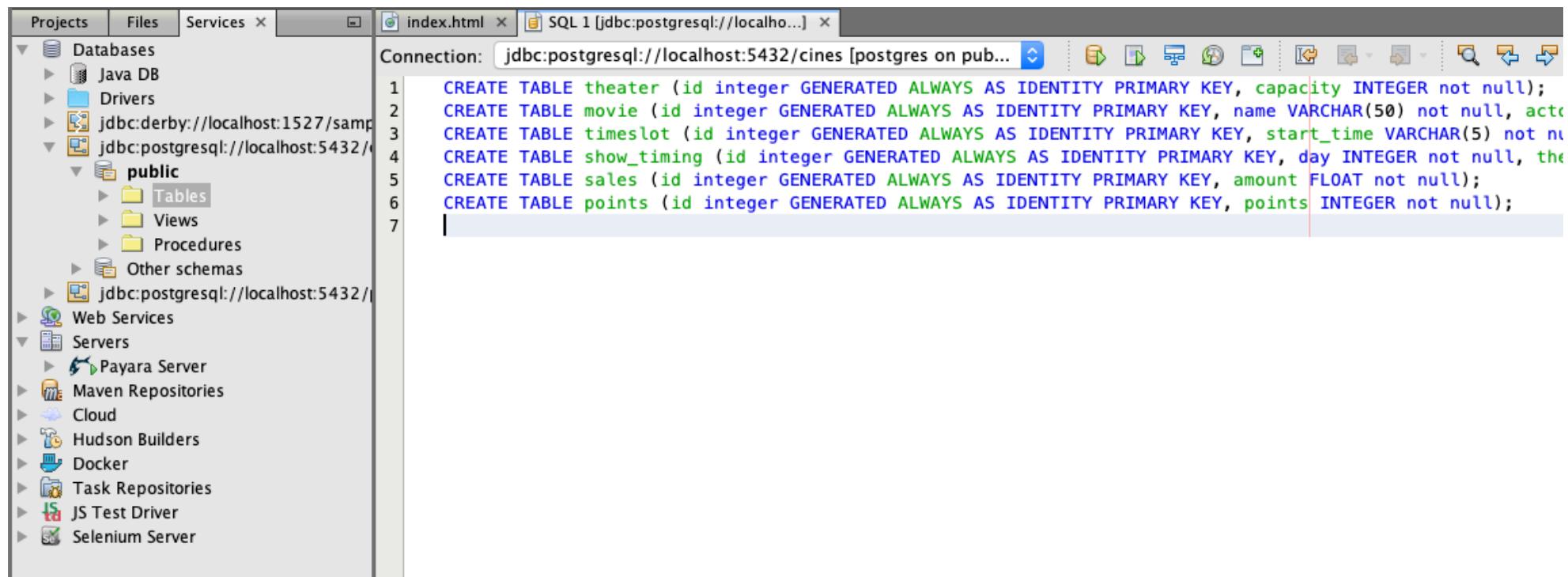
Base de datos

▶ Creamos las tablas y la poblamos

- Creáis una nueva conexión a la nueva tabla Postgres (como hicimos en el tema anterior)
- Execute command y ponemos las queries de los ficheros create.sql y load.sql que os he dejado en Campus Virtual (también podéis hacerlo directamente con la herramienta de administración)



Base de datos



The screenshot shows a software interface with a toolbar at the top and a central workspace. The workspace contains a tree view on the left and a code editor on the right.

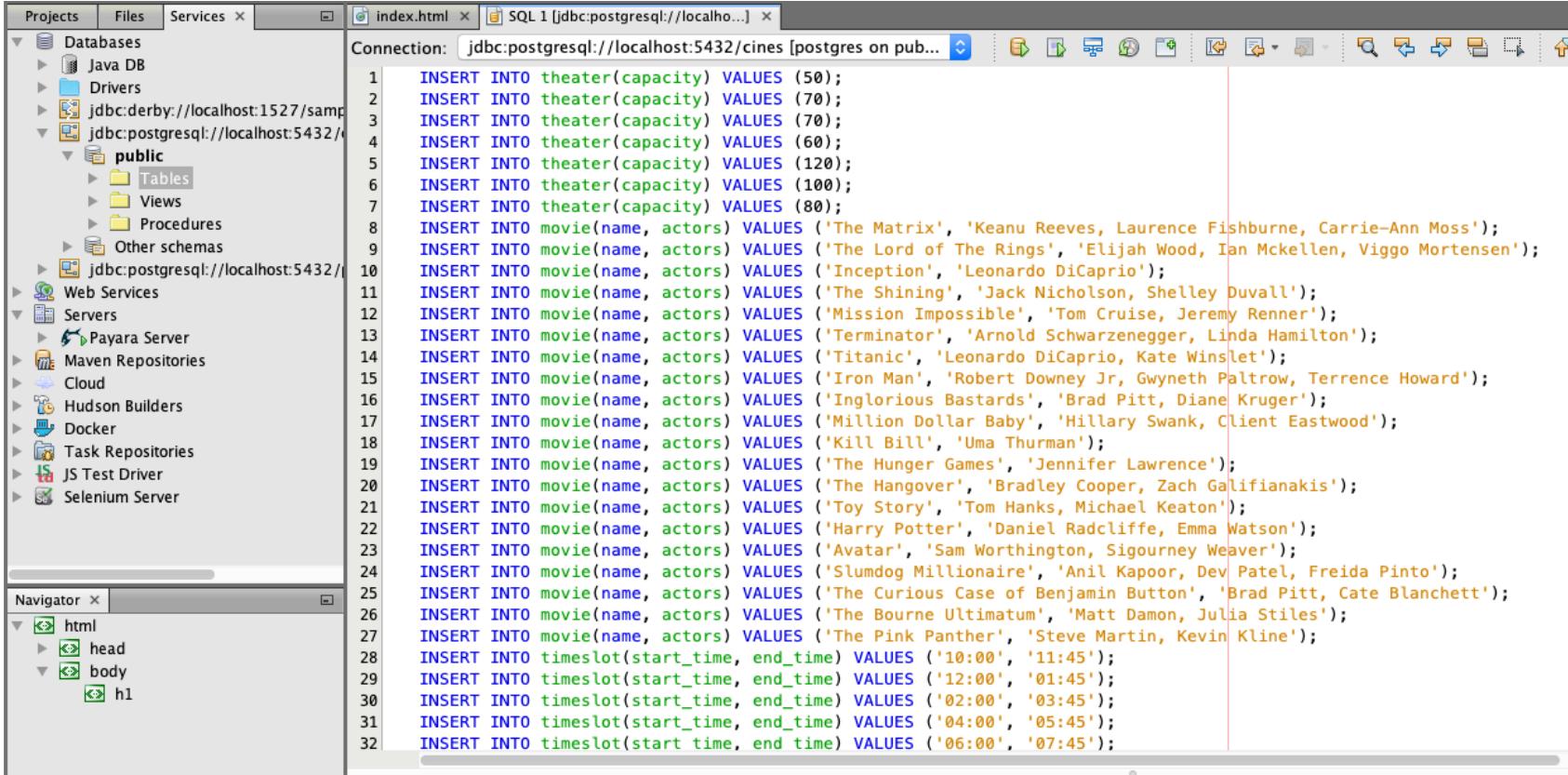
Tree View (Left):

- Projects
- Files
- Services
- Databases
 - Java DB
 - Drivers
 - jdbc:derby://localhost:1527/samp...
 - jdbc:postgresql://localhost:5432/cines
 - public
 - Tables
 - Views
 - Procedures
 - Other schemas
 - jdbc:postgresql://localhost:5432/...
- Web Services
- Servers
 - Payara Server
- Maven Repositories
- Cloud
- Hudson Builders
- Docker
- Task Repositories
- JS Test Driver
- Selenium Server

Connection: jdbc:postgresql://localhost:5432/cines [postgres on pub...]

```
1 CREATE TABLE theater (id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY, capacity INTEGER not null);
2 CREATE TABLE movie (id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY, name VARCHAR(50) not null, actor VARCHAR(50) not null, genre VARCHAR(50) not null, release_date DATE not null);
3 CREATE TABLE timeslot (id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY, start_time VARCHAR(5) not null, end_time VARCHAR(5) not null);
4 CREATE TABLE show_timing (id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY, day INTEGER not null, theater_id integer not null, movie_id integer not null, timeslot_id integer not null);
5 CREATE TABLE sales (id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY, amount FLOAT not null);
6 CREATE TABLE points (id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY, points INTEGER not null);
```

Base de datos



The screenshot shows a software interface with a toolbar at the top, a left sidebar for 'Projects' and 'Services', and a central workspace divided into two panes.

Left Sidebar (Projects):

- Databases
 - Java DB
 - Drivers
 - jdbc:derby://localhost:1527/samp
 - jdbc:postgresql://localhost:5432/cines
 - public
 - Tables
 - Views
 - Procedures
 - Other schemas
 - jdbc:postgresql://localhost:5432/
- Web Services
- Servers
 - Payara Server
- Maven Repositories
- Cloud
- Hudson Builders
- Docker
- Task Repositories
- JS Test Driver
- Selenium Server

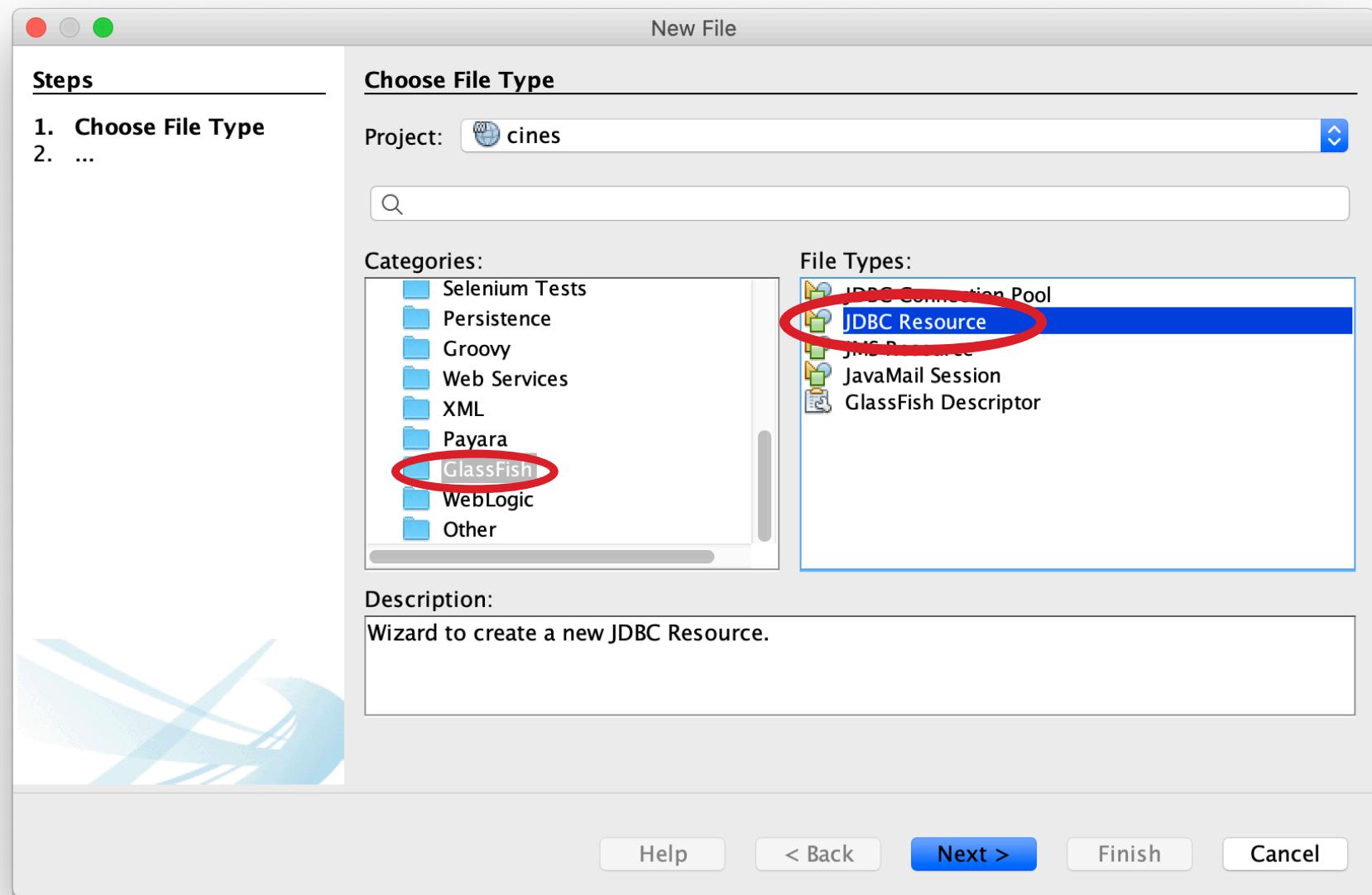
Connection: jdbc:postgresql://localhost:5432/cines [postgres on public]

```
1 INSERT INTO theater(capacity) VALUES (50);
2 INSERT INTO theater(capacity) VALUES (70);
3 INSERT INTO theater(capacity) VALUES (70);
4 INSERT INTO theater(capacity) VALUES (60);
5 INSERT INTO theater(capacity) VALUES (120);
6 INSERT INTO theater(capacity) VALUES (100);
7 INSERT INTO theater(capacity) VALUES (80);
8 INSERT INTO movie(name, actors) VALUES ('The Matrix', 'Keanu Reeves, Laurence Fishburne, Carrie-Ann Moss');
9 INSERT INTO movie(name, actors) VALUES ('The Lord of The Rings', 'Elijah Wood, Ian Mckellen, Viggo Mortensen');
10 INSERT INTO movie(name, actors) VALUES ('Inception', 'Leonardo DiCaprio');
11 INSERT INTO movie(name, actors) VALUES ('The Shining', 'Jack Nicholson, Shelley Duvall');
12 INSERT INTO movie(name, actors) VALUES ('Mission Impossible', 'Tom Cruise, Jeremy Renner');
13 INSERT INTO movie(name, actors) VALUES ('Terminator', 'Arnold Schwarzenegger, Linda Hamilton');
14 INSERT INTO movie(name, actors) VALUES ('Titanic', 'Leonardo DiCaprio, Kate Winslet');
15 INSERT INTO movie(name, actors) VALUES ('Iron Man', 'Robert Downey Jr, Gwyneth Paltrow, Terrence Howard');
16 INSERT INTO movie(name, actors) VALUES ('Inglourious Bastards', 'Brad Pitt, Diane Kruger');
17 INSERT INTO movie(name, actors) VALUES ('Million Dollar Baby', 'Hillary Swank, Clint Eastwood');
18 INSERT INTO movie(name, actors) VALUES ('Kill Bill', 'Uma Thurman');
19 INSERT INTO movie(name, actors) VALUES ('The Hunger Games', 'Jennifer Lawrence');
20 INSERT INTO movie(name, actors) VALUES ('The Hangover', 'Bradley Cooper, Zach Galifianakis');
21 INSERT INTO movie(name, actors) VALUES ('Toy Story', 'Tom Hanks, Michael Keaton');
22 INSERT INTO movie(name, actors) VALUES ('Harry Potter', 'Daniel Radcliffe, Emma Watson');
23 INSERT INTO movie(name, actors) VALUES ('Avatar', 'Sam Worthington, Sigourney Weaver');
24 INSERT INTO movie(name, actors) VALUES ('Slumdog Millionaire', 'Anil Kapoor, Dev Patel, Freida Pinto');
25 INSERT INTO movie(name, actors) VALUES ('The Curious Case of Benjamin Button', 'Brad Pitt, Cate Blanchett');
26 INSERT INTO movie(name, actors) VALUES ('The Bourne Ultimatum', 'Matt Damon, Julia Stiles');
27 INSERT INTO movie(name, actors) VALUES ('The Pink Panther', 'Steve Martin, Kevin Kline');
28 INSERT INTO timeslot(start_time, end_time) VALUES ('10:00', '11:45');
29 INSERT INTO timeslot(start_time, end_time) VALUES ('12:00', '01:45');
30 INSERT INTO timeslot(start_time, end_time) VALUES ('02:00', '03:45');
31 INSERT INTO timeslot(start_time, end_time) VALUES ('04:00', '05:45');
32 INSERT INTO timeslot(start_time, end_time) VALUES ('06:00', '07:45');
```

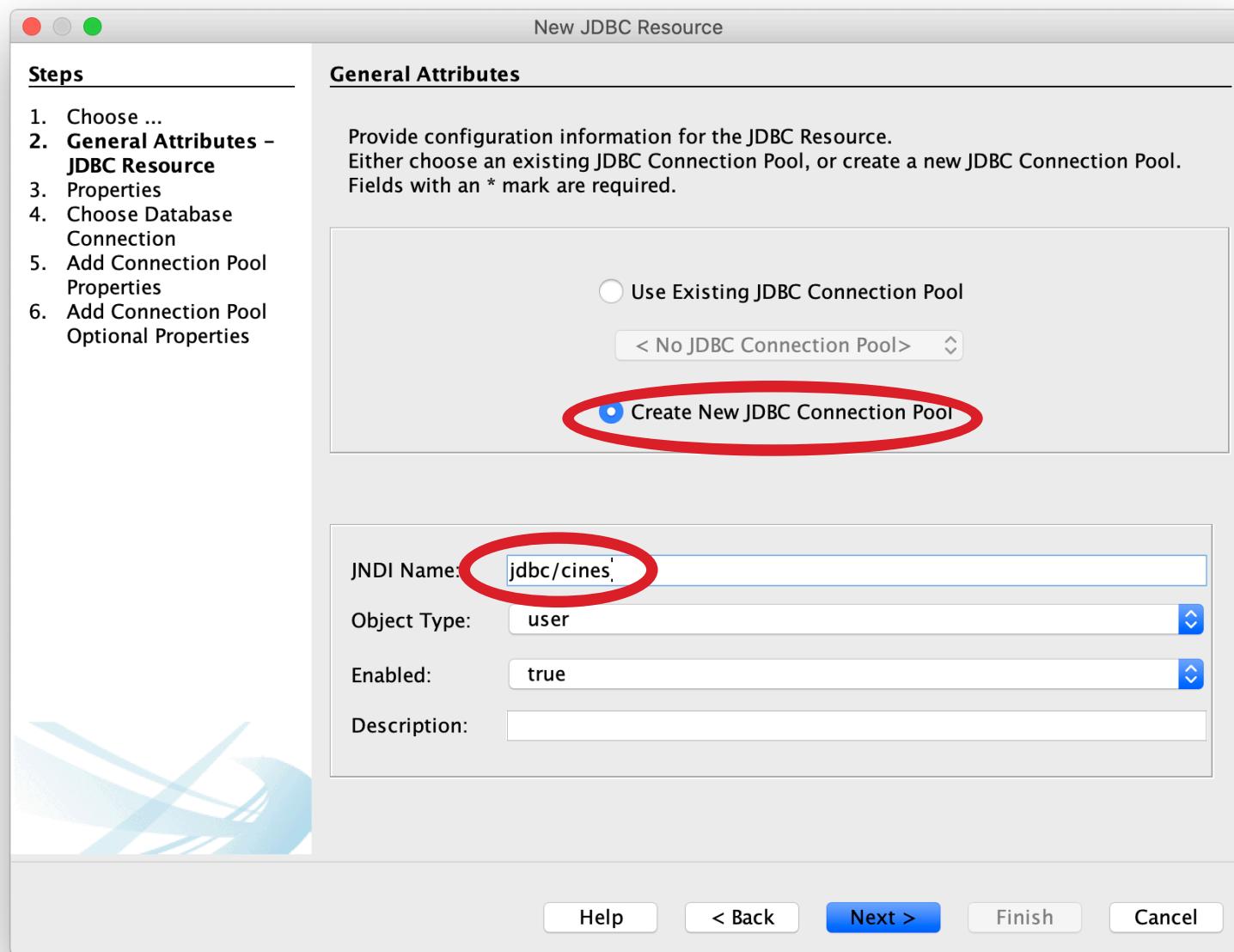
Recurso JDBC y Pool de conexiones

- ▶ Para gestionar las conexiones a la base de datos, lo mejor es crear un pool de conexiones en el servidor de aplicaciones y un recurso JDBC que sea el que se conecte con el pool de conexiones y nos ofrezca el acceso a la base de datos
- ▶ Netbeans nos permite crear los ficheros de recursos JDBC y pool de conexiones de manera sencilla para luego importarlos al servidor de aplicaciones
- ▶ Para ello hacemos click con el botón derecho sobre la carpeta del proyecto, buscamos new – other – Glassfish – JDBC Resource

Recurso JDBC y Pool de conexiones



Recurso JDBC y Pool de conexiones



Recurso JDBC y Pool de conexiones

New JDBC Resource

Steps

1. Choose ...
2. General Attributes – JDBC Resource
- 3. Properties**
4. Choose Database Connection
5. Add Connection Pool Properties
6. Add Connection Pool Optional Properties

Additional Properties

Add additional configuration information for the resource jdbc/cines.
Hit the Enter key to save values in the Properties table.

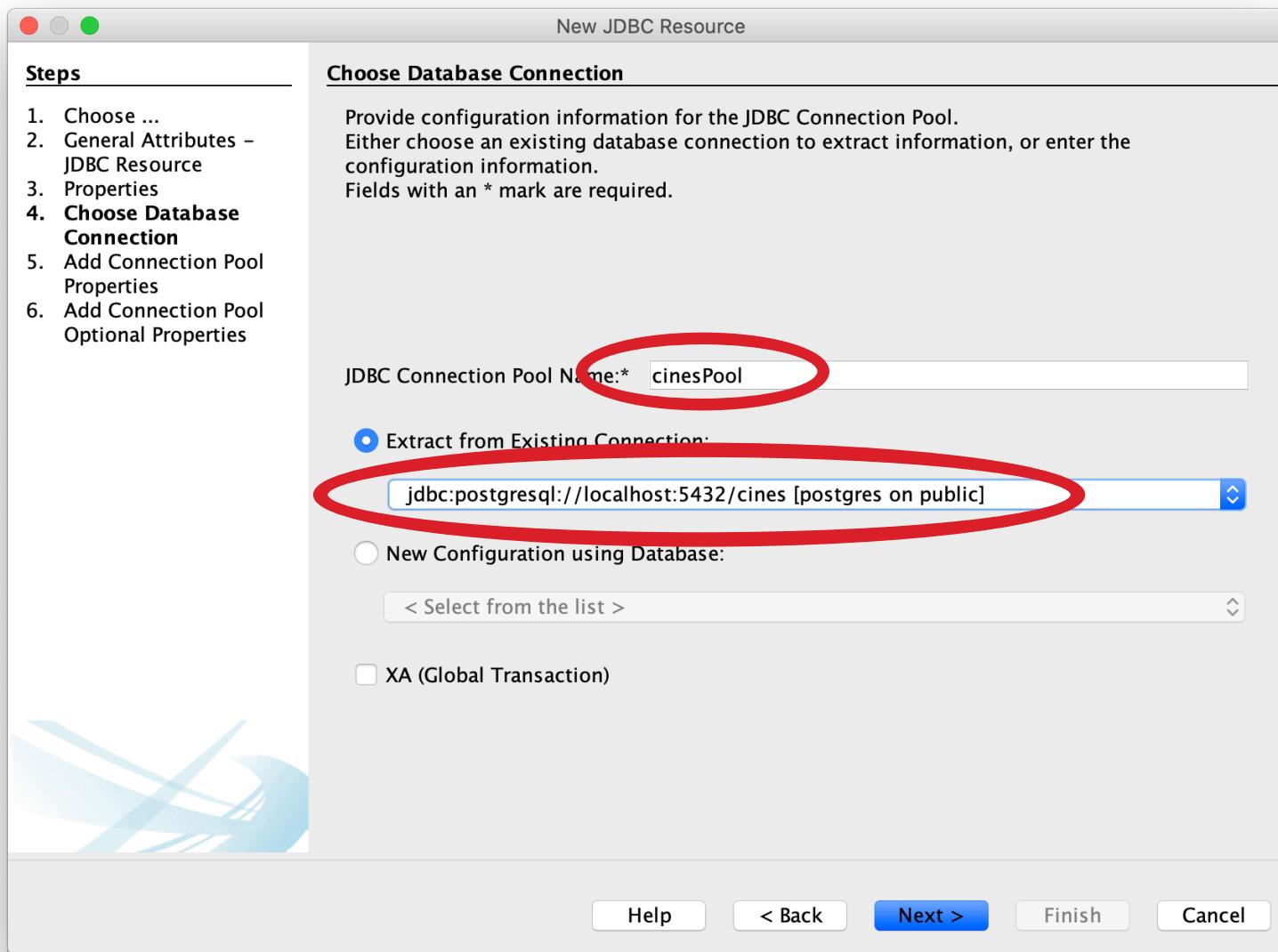
Properties:

Name	Value
------	-------

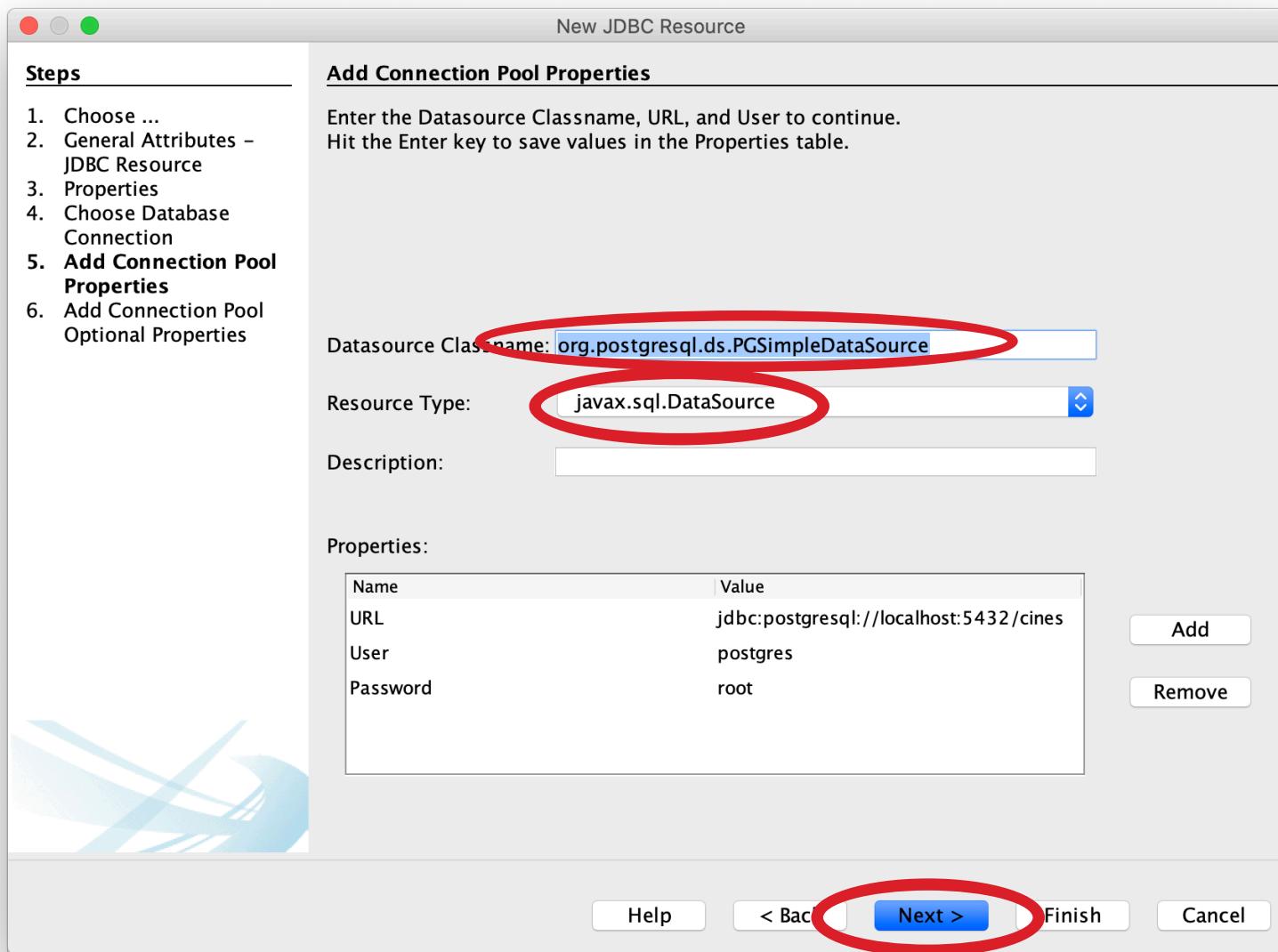
Add Remove

Help < Back **Next >** Finish Cancel

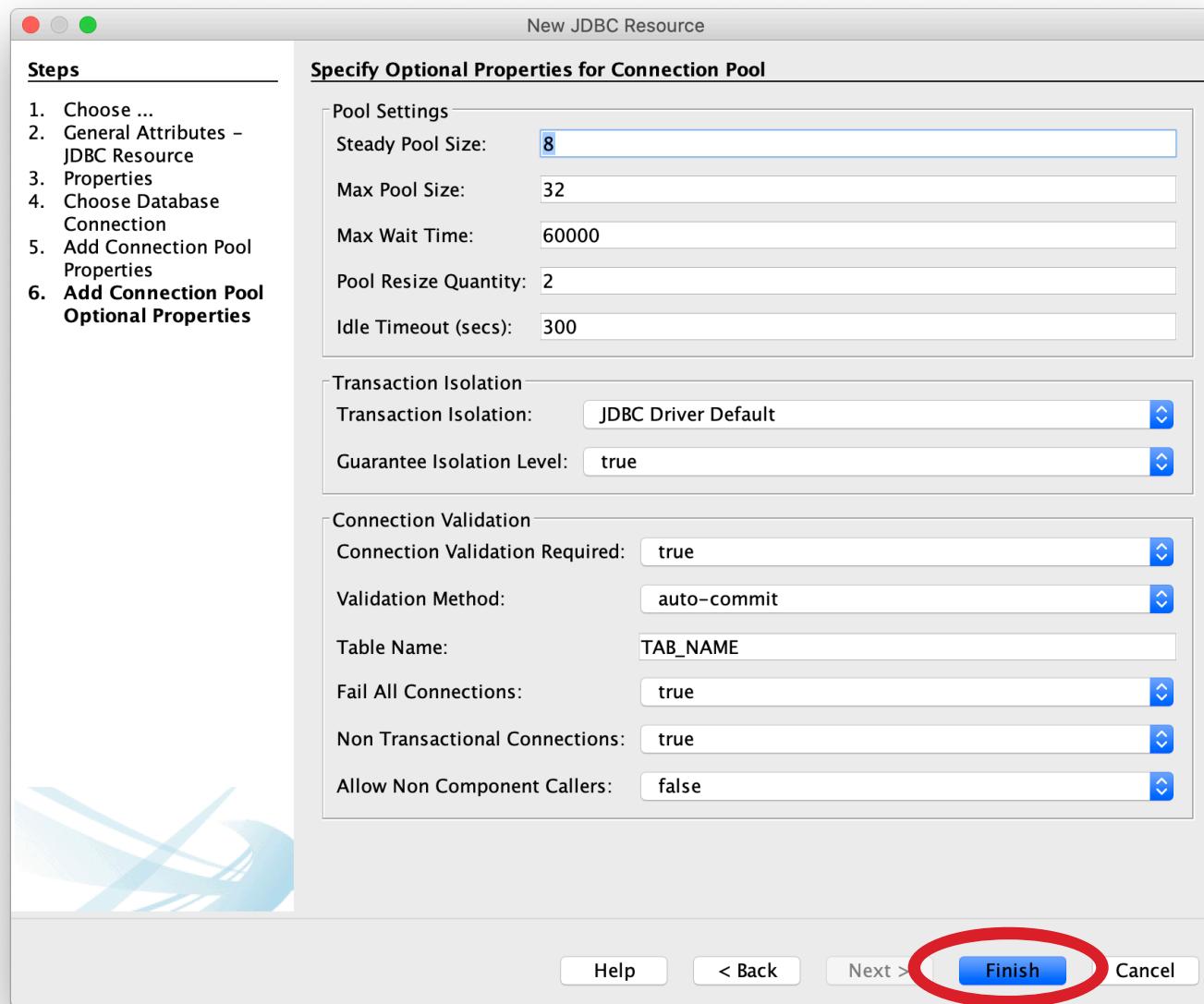
Recurso JDBC y Pool de conexiones



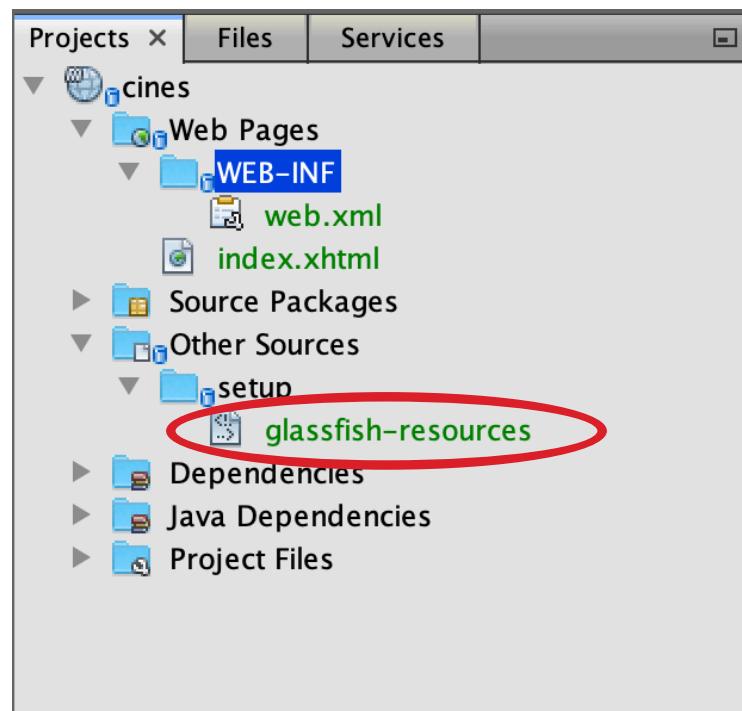
Recurso JDBC y Pool de conexiones



Recurso JDBC y Pool de conexiones

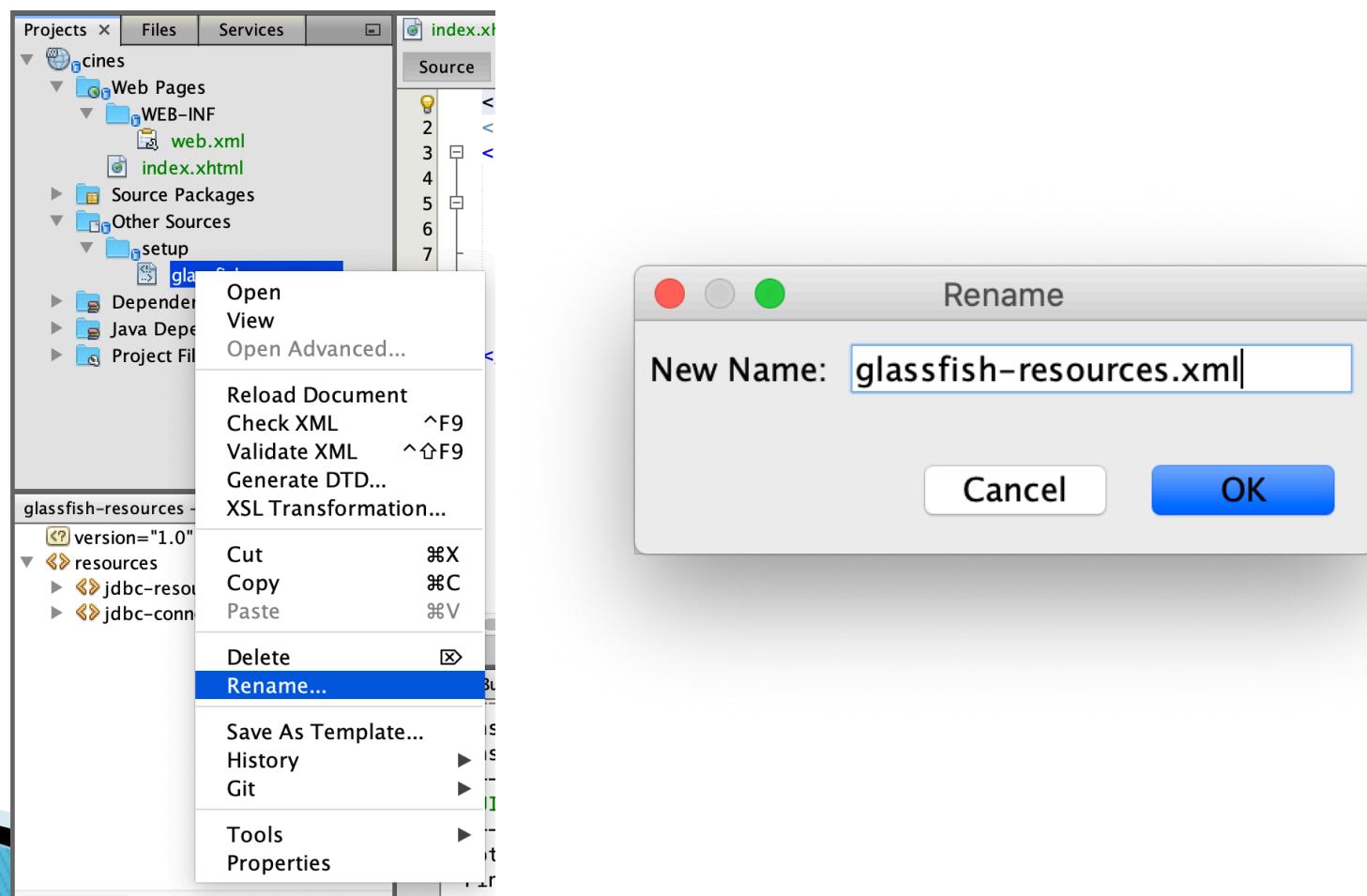


Recurso JDBC y Pool de conexiones



Recurso JDBC y Pool de conexiones

- ▶ Cambiamos el nombre para añadir la extensión xml si no aparece por defecto



Recurso JDBC y Pool de conexiones (lo importamos a Payara – puerto 4848)

User: admin Domain: domain1 Server: localhost

Hor

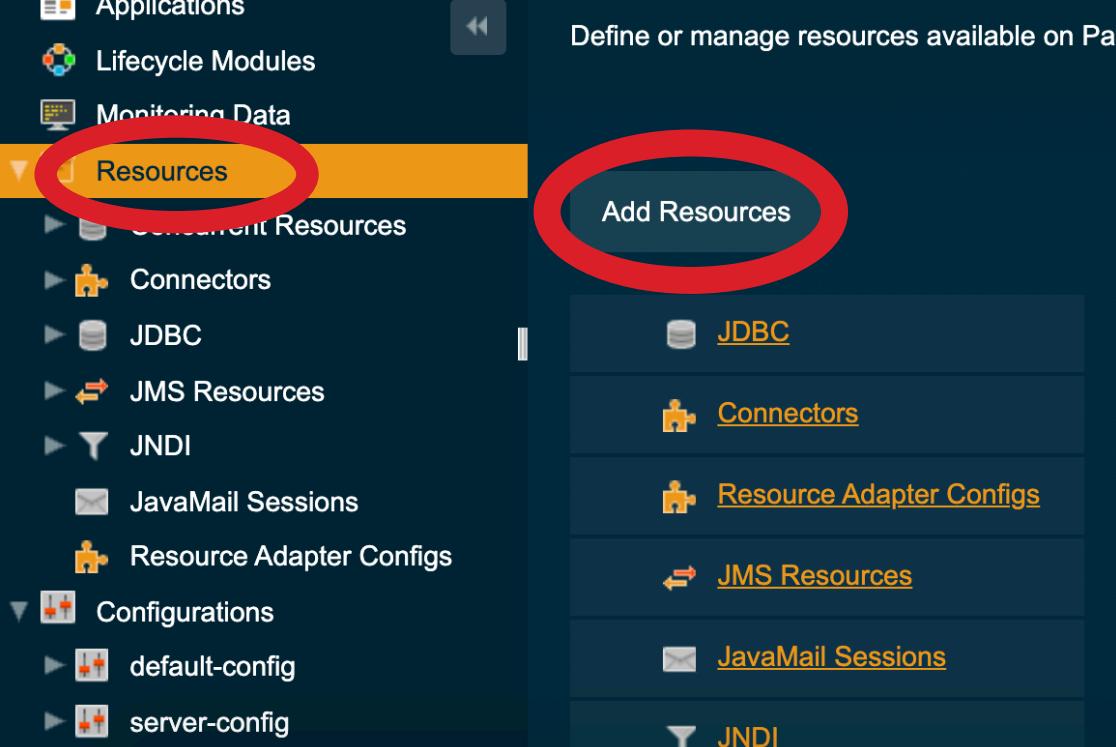


Define or manage resources available on Payara Server.

Applications
Lifecycle Modules
Monitoring Data
Resources
Concurrent Resources
Connectors
JDBC
JMS Resources
JNDI
JavaMail Sessions
Resource Adapter Configs
Configurations
default-config
server-config

Add Resources

JDBC
Connectors
Resource Adapter Configs
JMS Resources
JavaMail Sessions
JNDI



Recurso JDBC y Pool de conexiones (lo importamos a Payara – puerto 4848)

User: admin Domain: domain1 Server: localhost Home About... Help Online Help Enable Asadmin Recorder

payara® SERVER

Applications Lifecycle Modules Monitoring Data Resources Concurrent Resources Connectors JDBC JMS Resources JNDI JavaMail Sessions Resource Adapter Configs Configurations default-config server-config

Add Resources

Add Resources specified in a file for all the selected targets.

* Indicates required field

Location: Seleccione el archivo para cargarlo al servidor

Seleccionar archivo Ningún archivo seleccionado

Local XML file that is accessible from Payara Server

Browse Files...

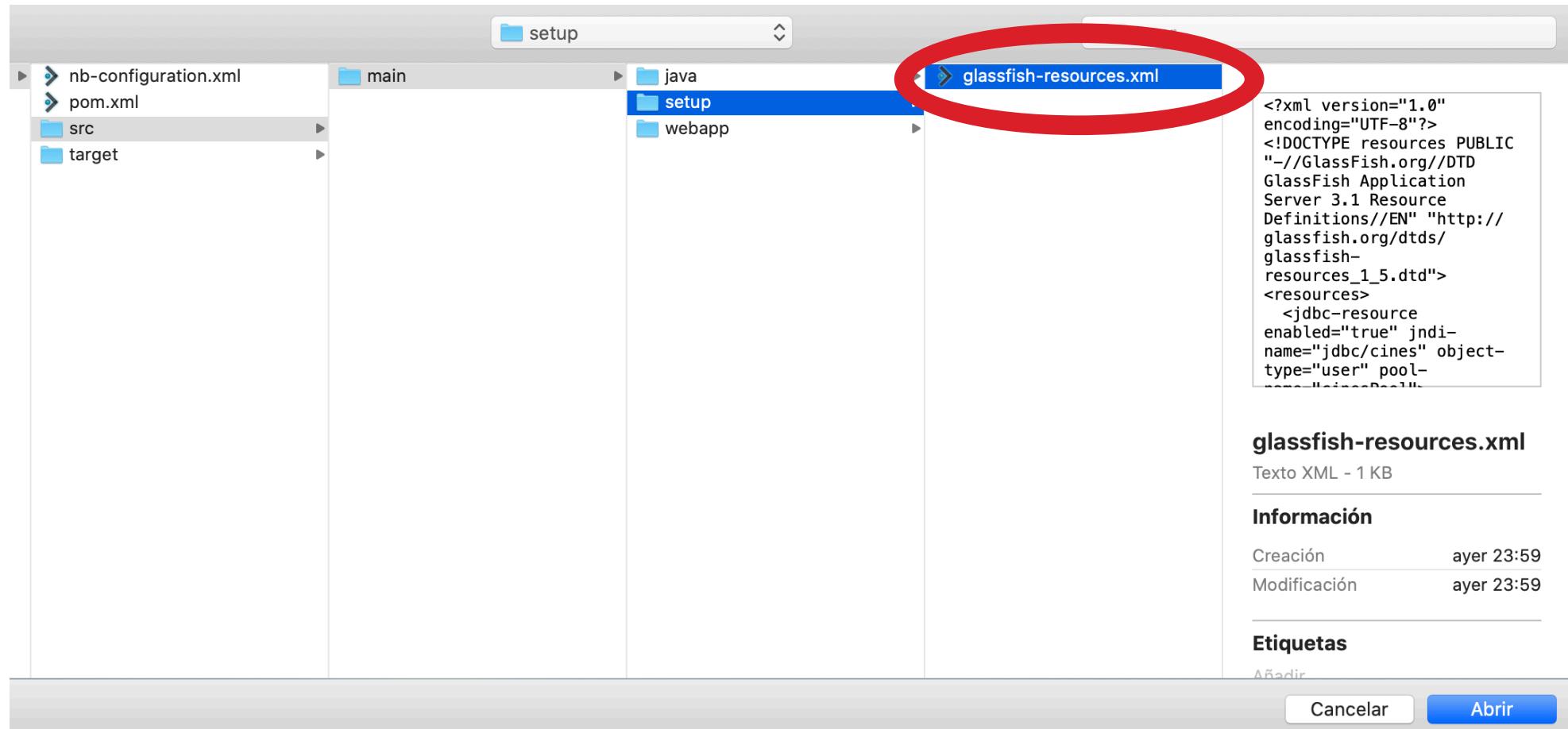
OK Cancel

Target: * server

Choose a target from the drop-down list.

The screenshot shows the Payara Server Administration Console interface. The left sidebar has a tree view with 'Resources' selected. The main panel shows the 'Add Resources' dialog. In the 'Location:' field, a file selection dialog is open, with the message 'Local XML file that is accessible from Payara Server' displayed above the file list. The 'OK' and 'Cancel' buttons are at the bottom right of the dialog. The top navigation bar shows 'User: admin', 'Domain: domain1', 'Server: localhost', and links for 'Home', 'About...', 'Help', 'Online Help', and 'Enable Asadmin Recorder'.

Recurso JDBC y Pool de conexiones (lo importamos a Payara – puerto 4848)



Recurso JDBC y Pool de conexiones (lo importamos a Payara – puerto 4848)

User: admin Domain: domain1 Server: localhost Home About... Help Online Help Enable Asadmin Recorder

payara® SERVER

Applications Lifecycle Modules Monitoring Data Resources Concurrent Resources Connectors JDBC JMS Resources JNDI JavaMail Sessions Resource Adapter Configs Configurations default-config server-config

Add Resources

Add Resources specified in a file for all the selected targets.

* Indicates required field

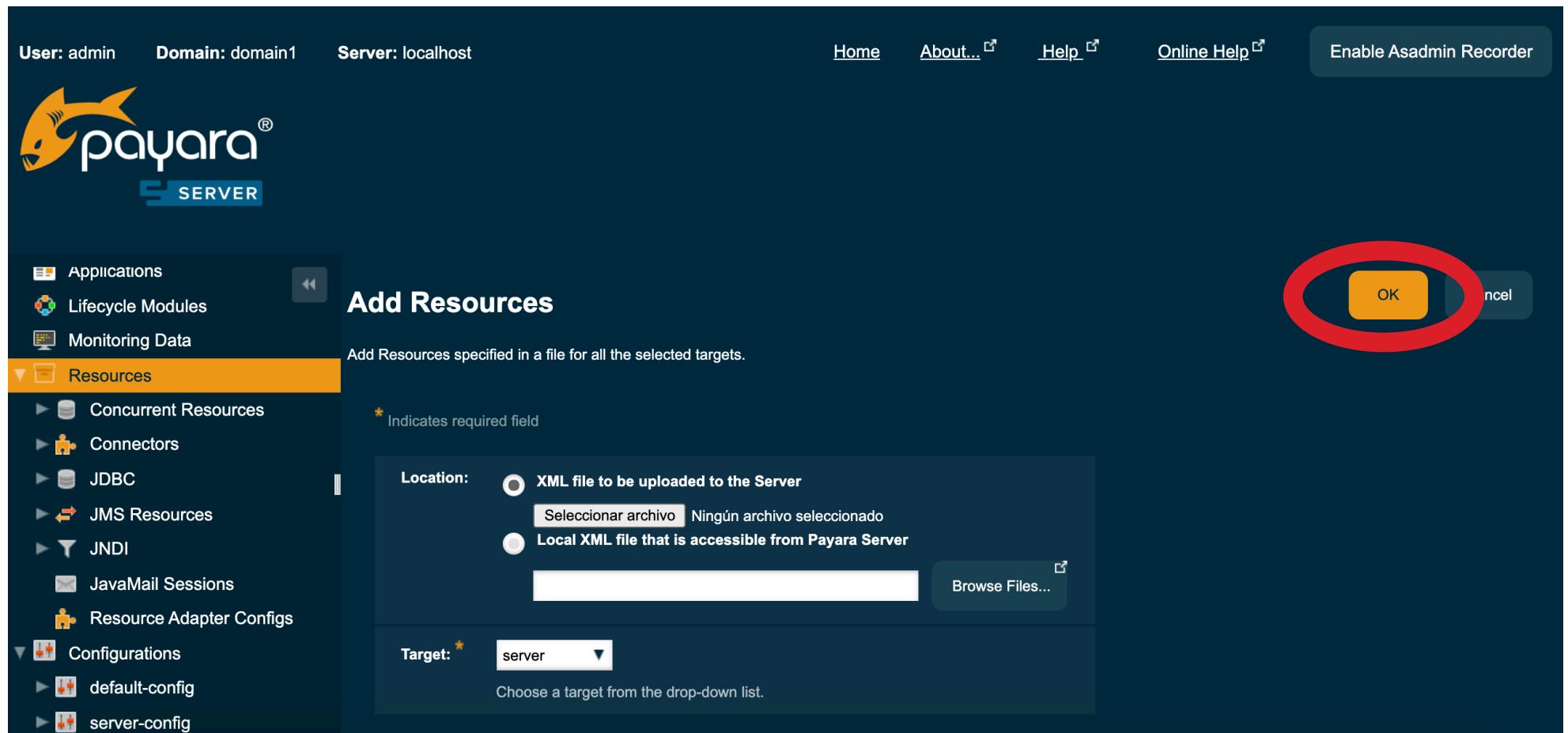
Location:

- XML file to be uploaded to the Server
Seleccionar archivo Ningún archivo seleccionado Browse Files...
- Local XML file that is accessible from Payara Server

Target: * server

Choose a target from the drop-down list.

OK Cancel



Recurso JDBC y Pool de conexiones (lo importamos a Payara – puerto 4848)

The screenshot shows the Payara Server Admin Console interface. The top navigation bar includes 'User: admin', 'Domain: domain1', 'Server: localhost', and links for 'Home', 'About...', 'Help', 'Online Help', and 'Enable Asadmin Recorder'. The main header features the Payara Server logo.

The left sidebar contains navigation links: Applications, Lifecycle Modules, Monitoring Data, Resources (selected), Concurrent Resources, Connectors, JDBC (highlighted), JMS Resources, JNDI, JavaMail Sessions, Resource Adapter Configs, Configurations (selected), default-config, and server-config.

The central 'Resources' page title is 'Define or manage resources available on Payara Server.' A prominent orange success message box is centered, stating 'Resources added successfully.' with a checkmark icon. Below this message, there are buttons for 'Add Resources' and links for 'JDBC' and 'Connectors'.

Recurso JDBC y Pool de conexiones (lo importamos a Payara – puerto 4848)

User: admin Domain: domain1 Server: localhost

Home About... Help Online Help Enable Asadmin Recorder

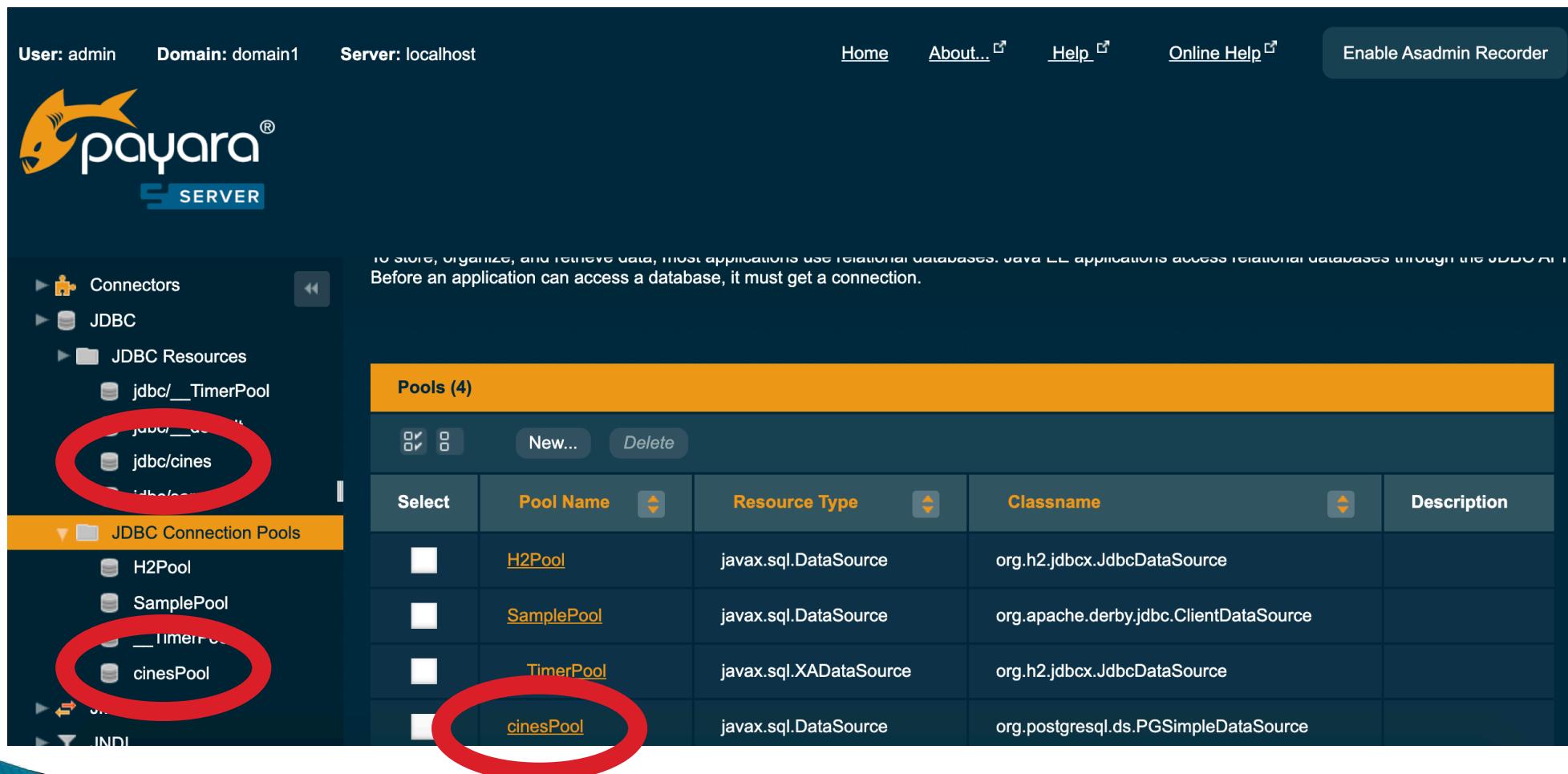
payara® SERVER

Connectors JDBC JDBC Resources jdbc/__TimerPool jdbc/_cines jdbc/cines JDBC Connection Pools H2Pool SamplePool __TimerPool cinesPool

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.

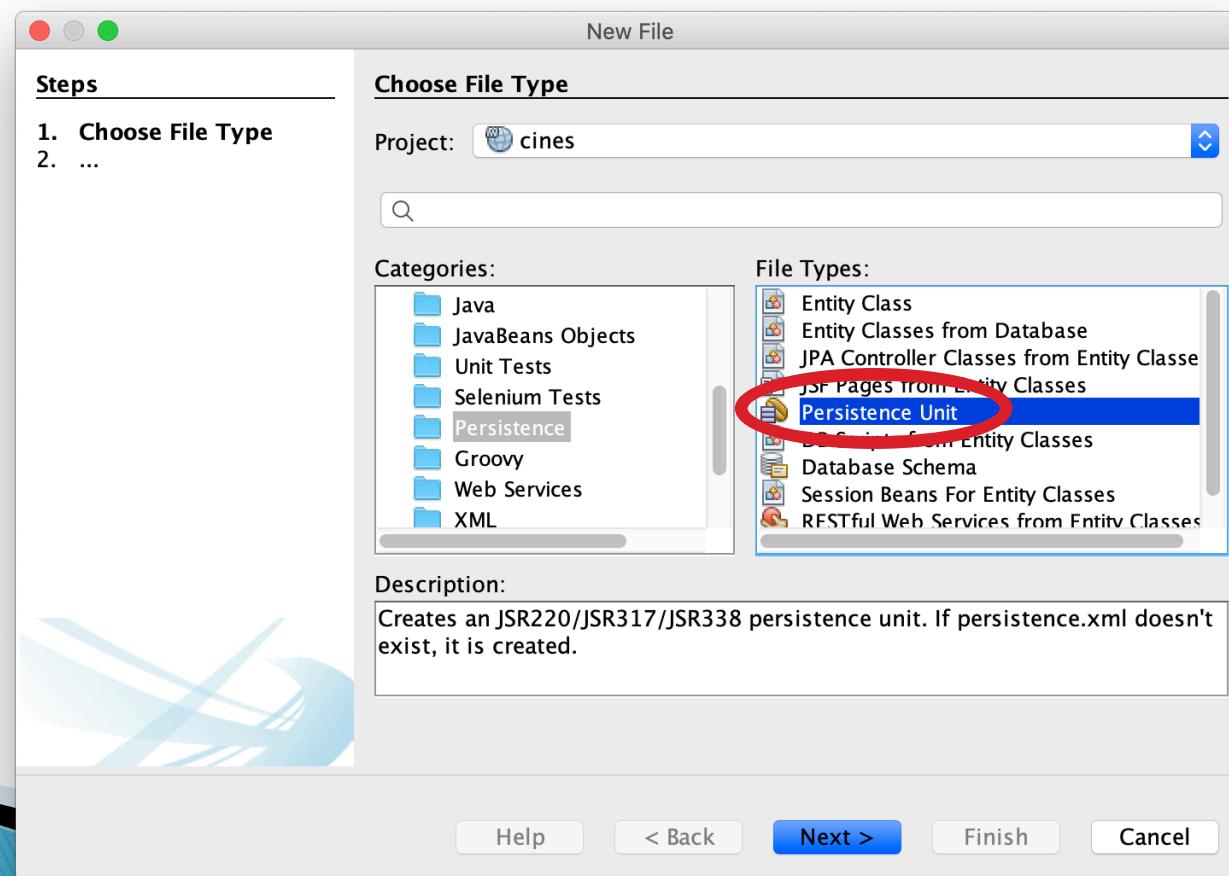
Pools (4)

Select	Pool Name	Resource Type	Classname	Description
	H2Pool	javax.sql.DataSource	org.h2.jdbcx.JdbcDataSource	
	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
	TimerPool	javax.sql.XADatasource	org.h2.jdbcx.JdbcDataSource	
	cinesPool	javax.sql.DataSource	org.postgresql.ds.PGSimpleDataSource	

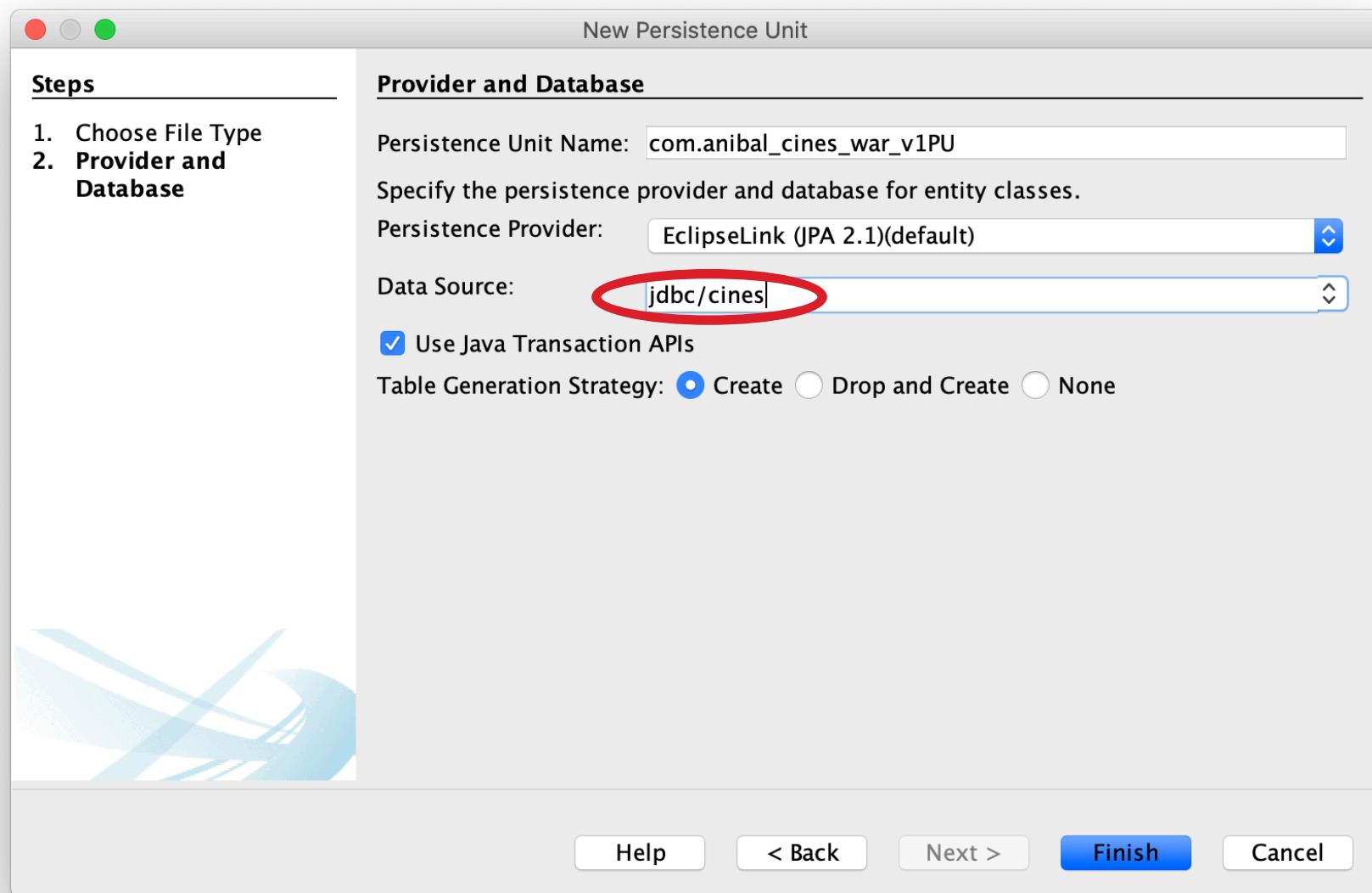


Unidad de persistencia

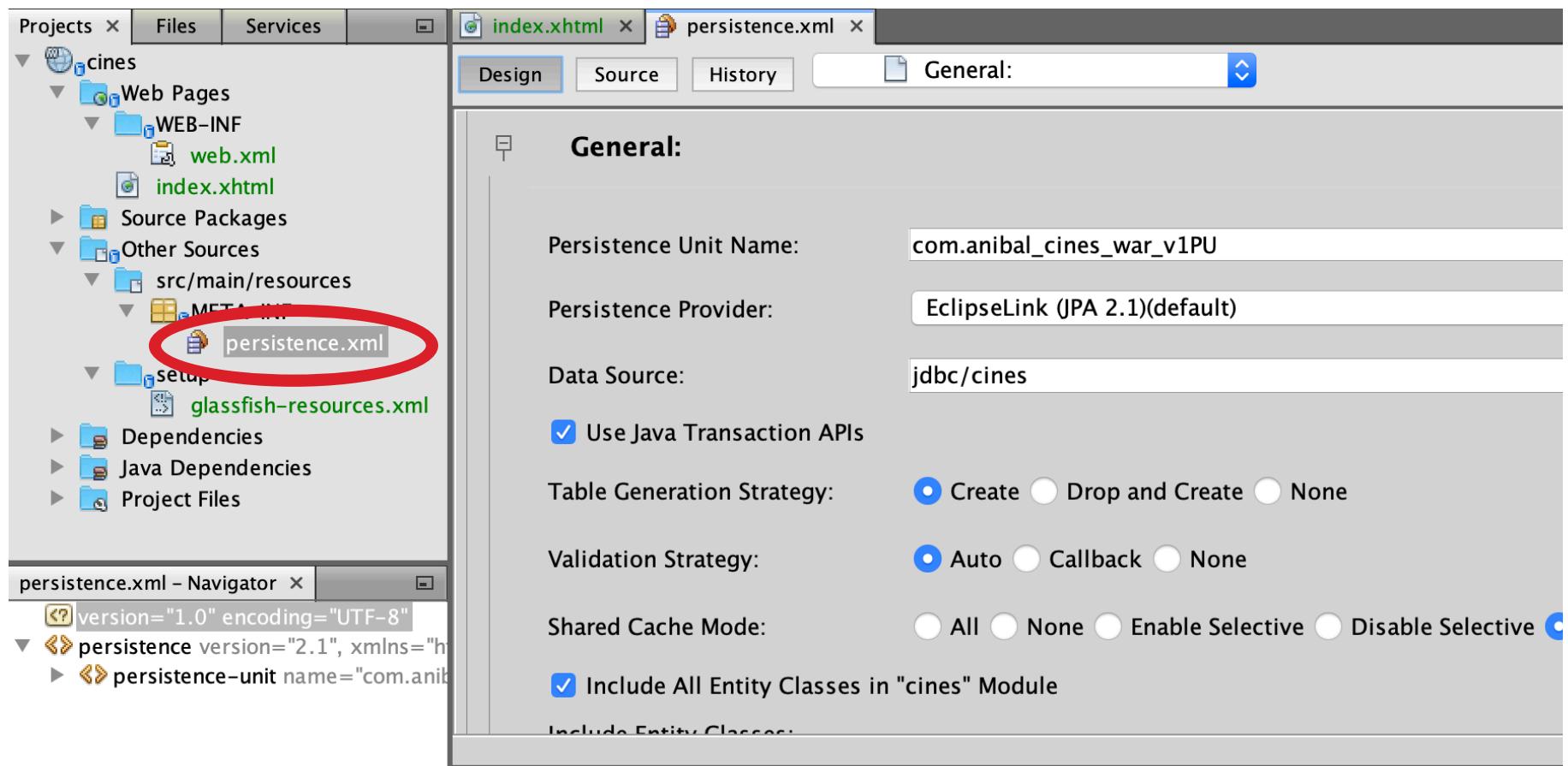
- Para terminar la configuración de la conexión a la base de datos, tenemos que configurar la unidad de persistencia para que use el recurso JDBC que acabamos de crear



Unidad de persistencia



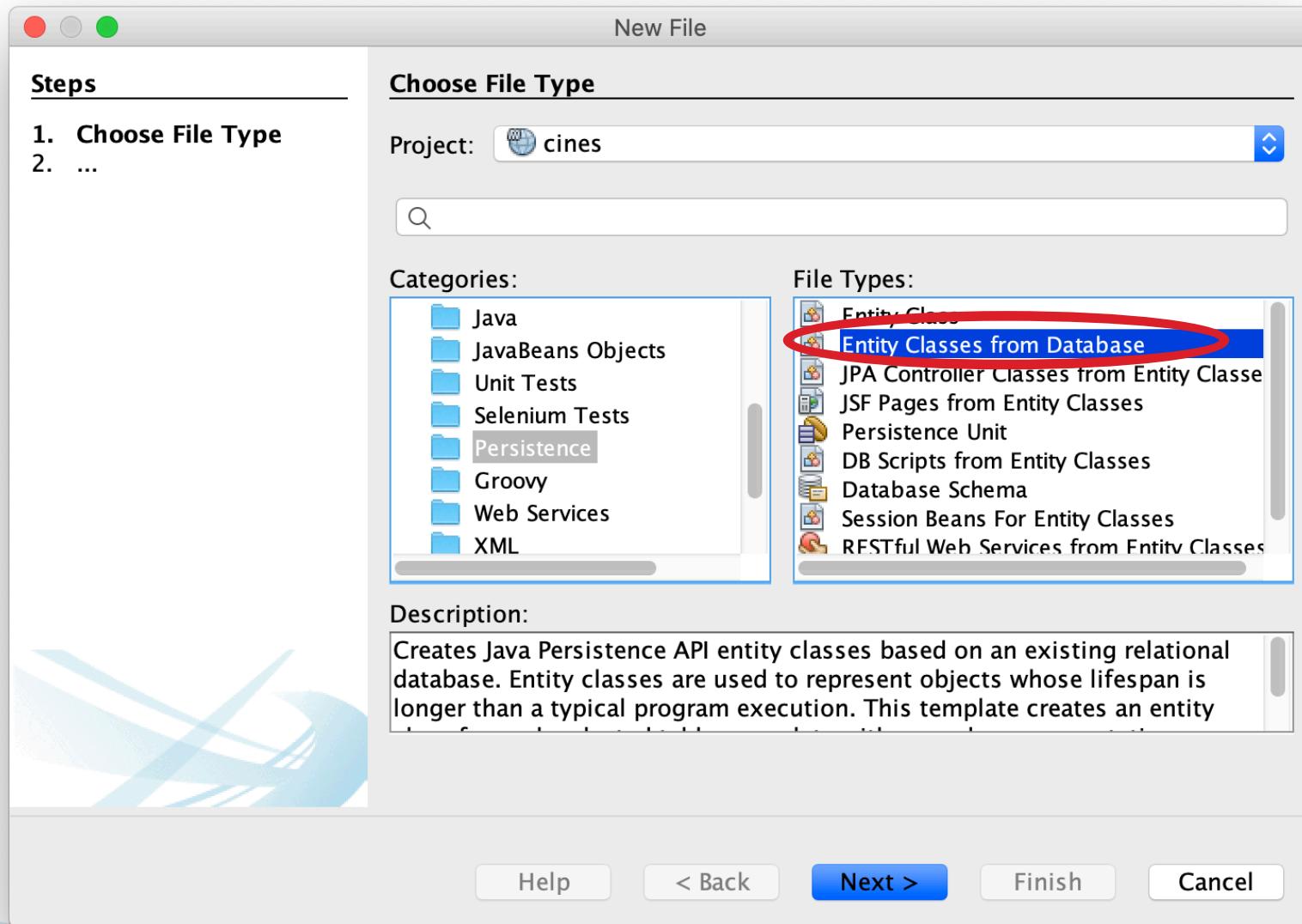
Unidad de persistencia



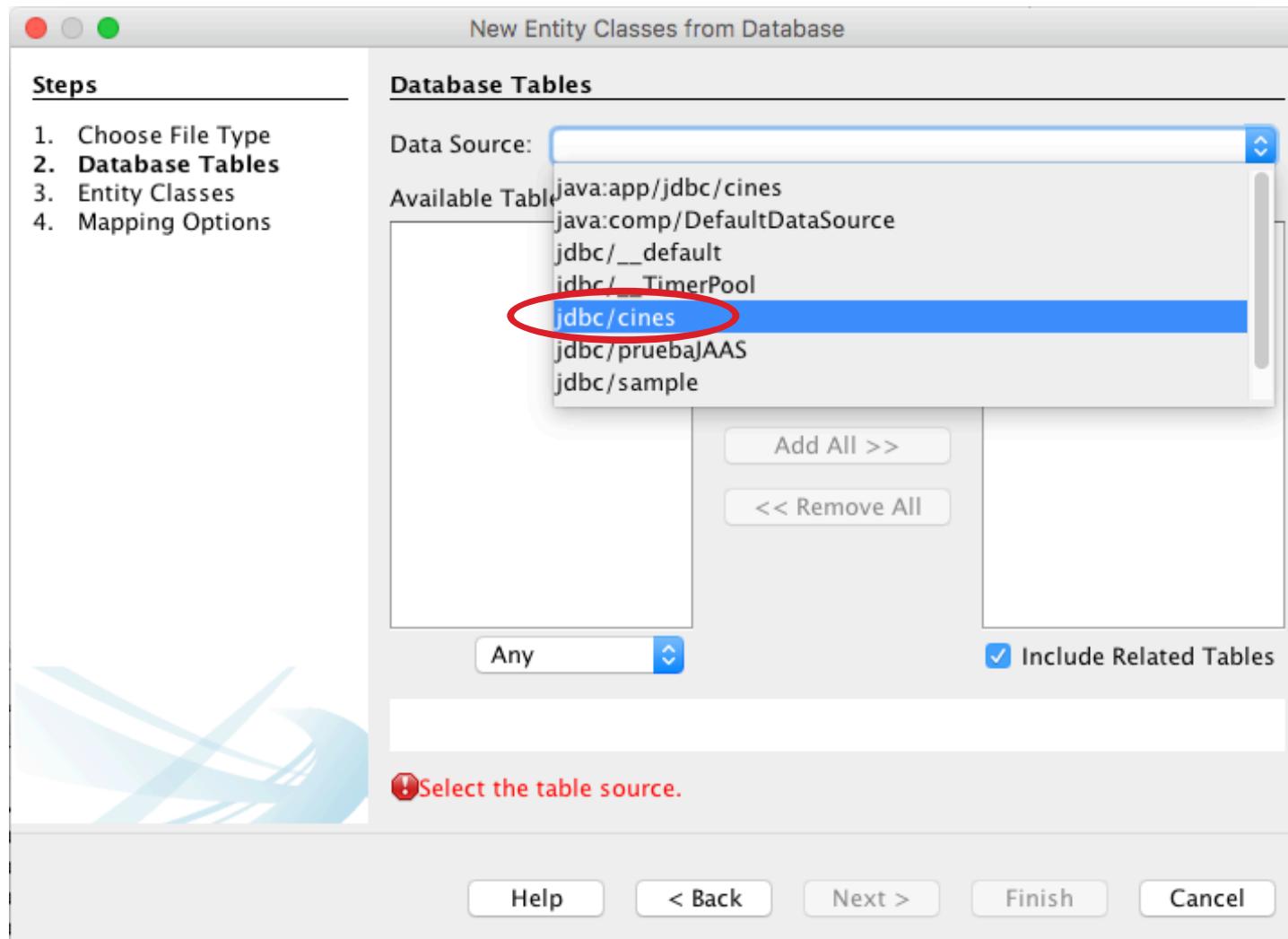
Clases entidad

- ▶ Y ahora creamos las clases entidad directamente desde las tablas de la base de datos
- ▶ New → others → Persistence → Entity Classes from Database

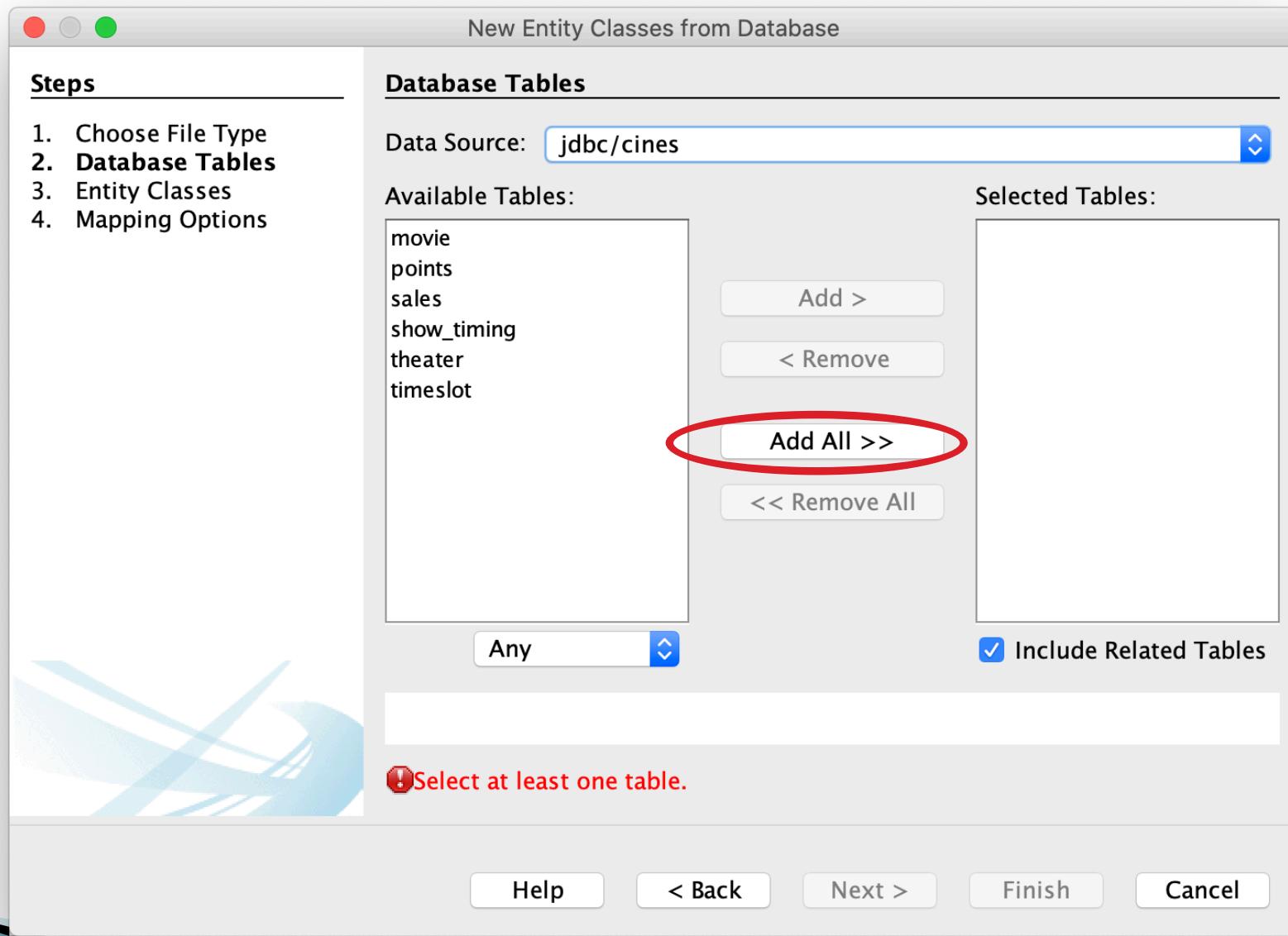
Clases entidad



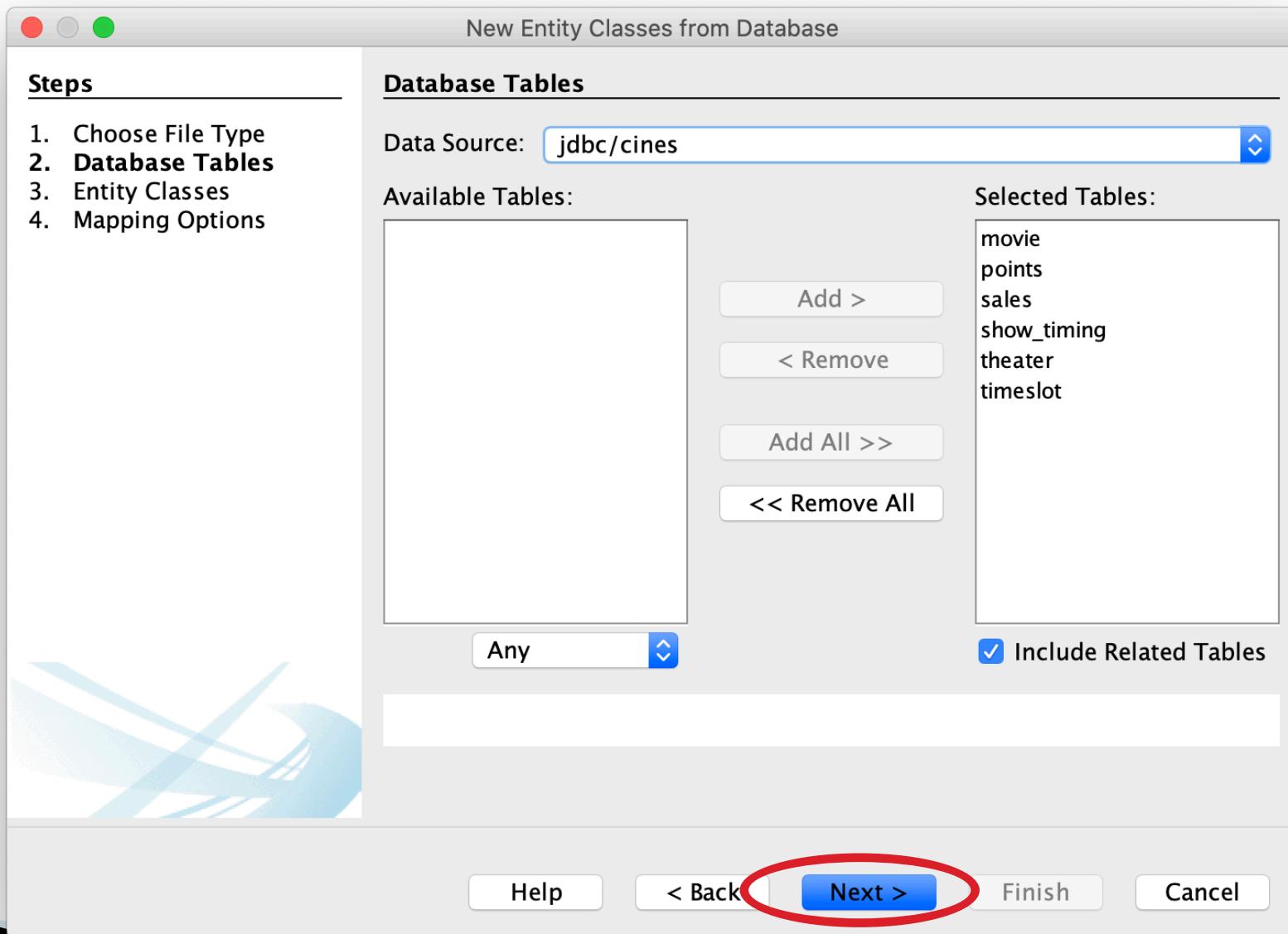
Clases entidad



Clases entidad



Clases entidad



Clases entidad

New Entity Classes from Database

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
movie	Movie	New
points	Points	New
sales	Sales	New
show_timing	ShowTiming	New
theater	Theater	New

Project: cines

Location: Source Packages

Package: com.anibal.cines.entities

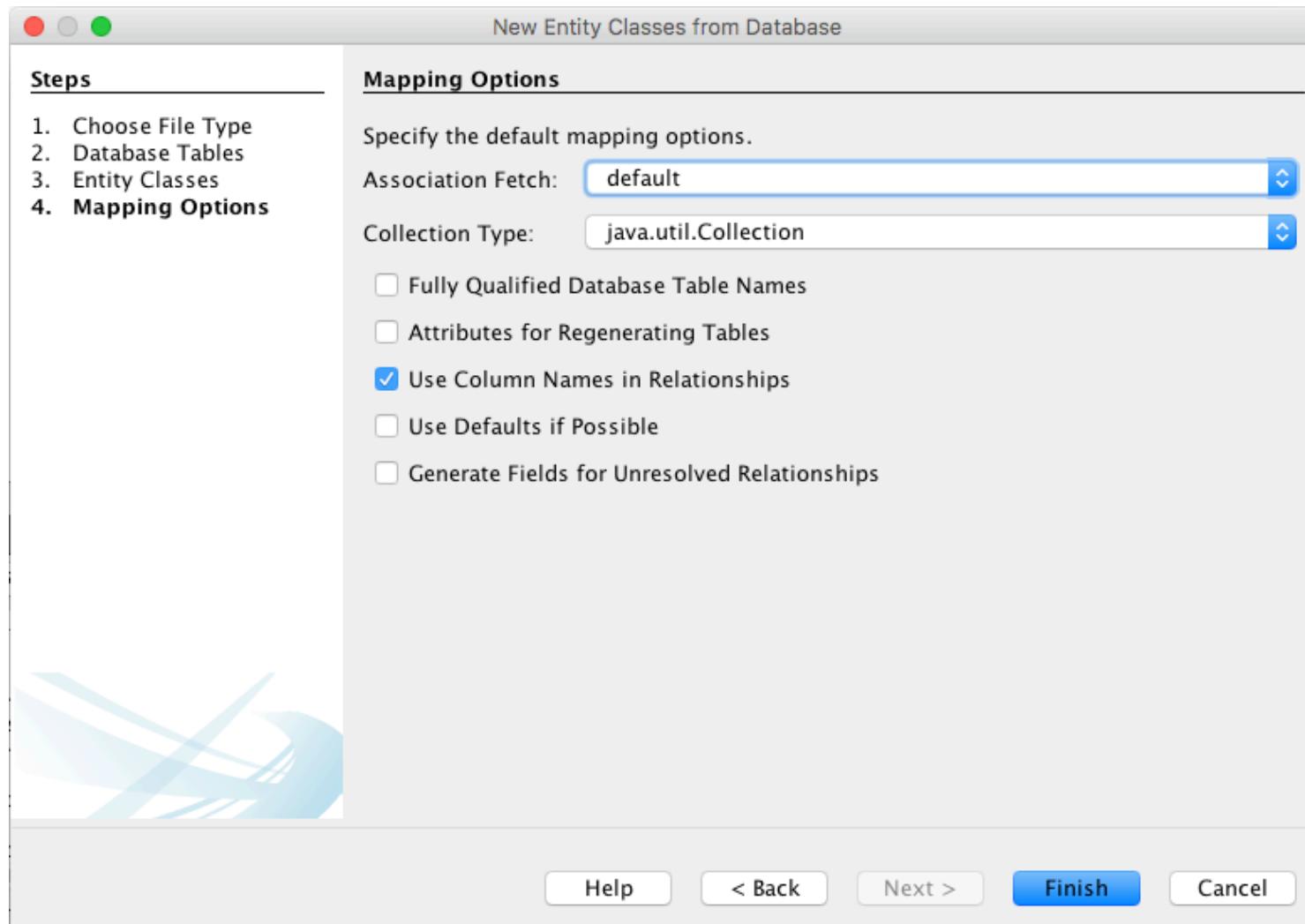
Generate Named Query Annotations for Persistent Fields

Generate JAXB Annotations

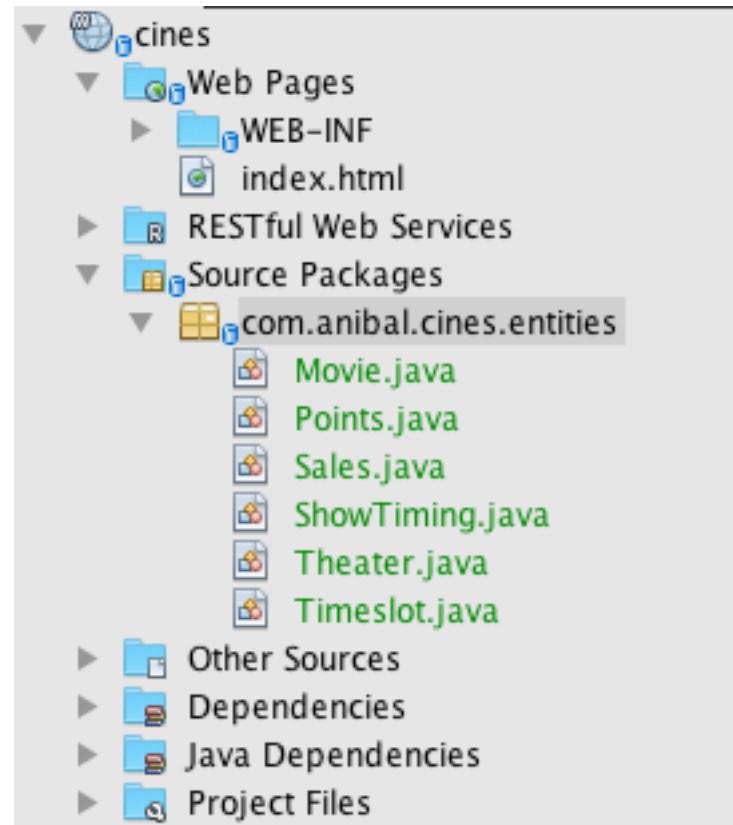
Generate MappedSuperclasses instead of Entities

Help < Back Next > Finish Cancel

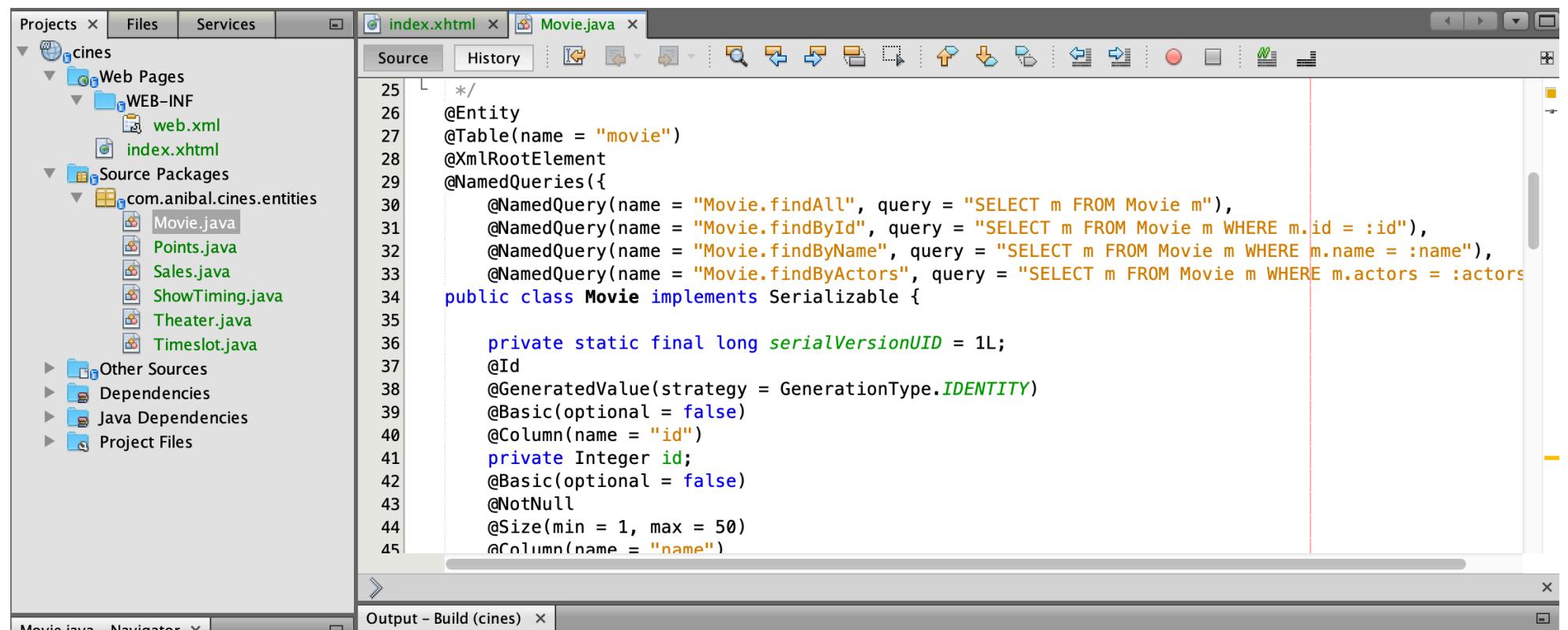
Clases entidad



Clases entidad



Clases entidad



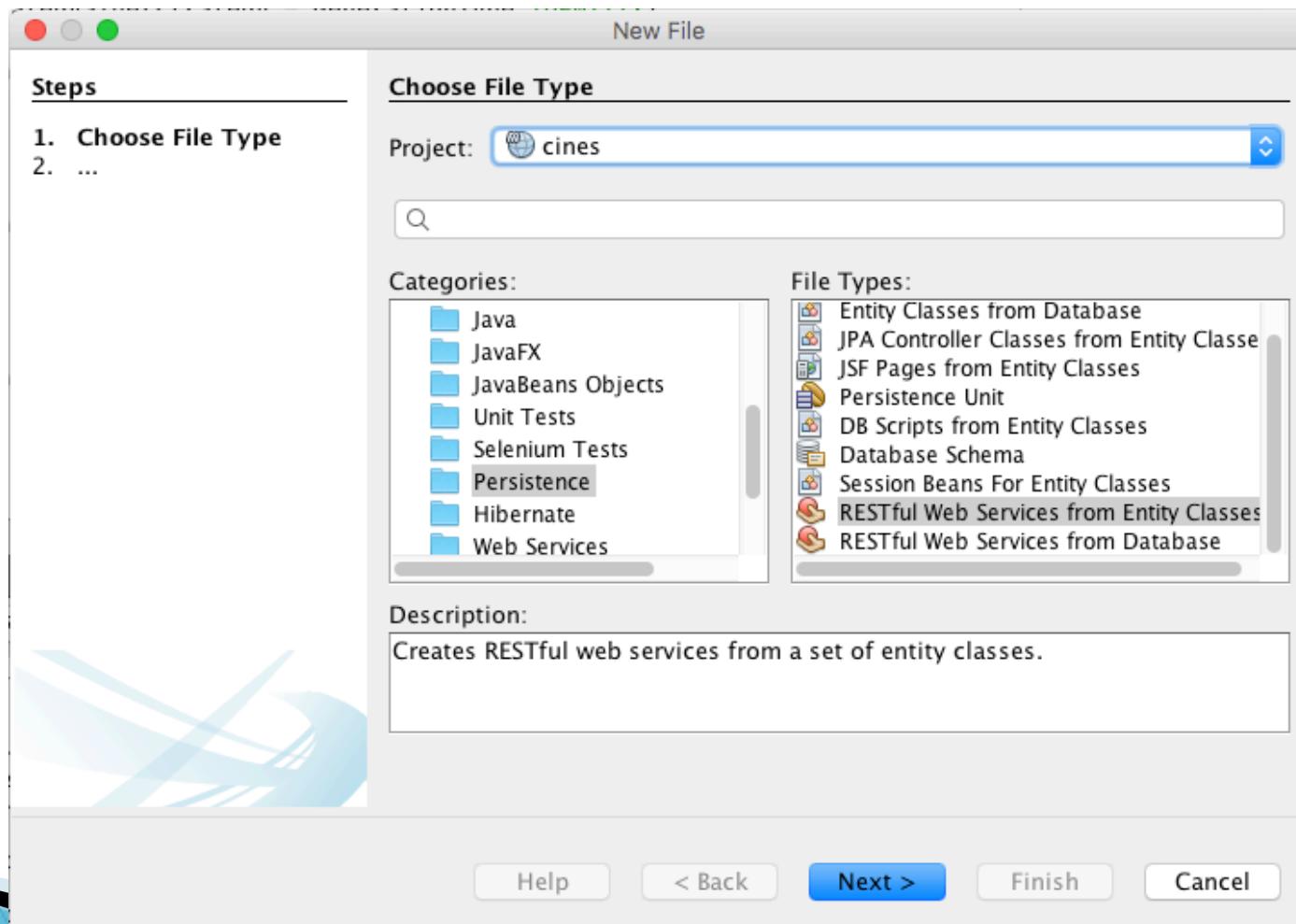
The screenshot shows a Java IDE interface with the following details:

- Project Explorer (Left):** Shows a project named "cines" containing:
 - Web Pages: index.xhtml
 - Source Packages: com.anibal.cines.entities (containing Movie.java, Points.java, Sales.java, ShowTiming.java, Theater.java, Timeslot.java)
 - Other Sources
 - Dependencies
 - Java Dependencies
 - Project Files
- Editor (Center):** Displays the content of Movie.java. The code defines a class Movie that implements Serializable. It uses annotations from javax.persistence and javax.xml.bind. The class has a primary key id of type Integer and a column name of type String named name. It includes four named queries: findAll, findById, findByName, and findByActors, each with a corresponding SQL SELECT statement.
- Bottom Navigation:** Shows tabs for "Movie.java - Navigator" and "Output - Build (cines)".

```
25  */
26  @Entity
27  @Table(name = "movie")
28  @XmlRootElement
29  @NamedQueries({
30      @NamedQuery(name = "Movie.findAll", query = "SELECT m FROM Movie m"),
31      @NamedQuery(name = "Movie.findById", query = "SELECT m FROM Movie m WHERE m.id = :id"),
32      @NamedQuery(name = "Movie.findByName", query = "SELECT m FROM Movie m WHERE m.name = :name"),
33      @NamedQuery(name = "Movie.findByActors", query = "SELECT m FROM Movie m WHERE m.actors = :actors")
34  })
35
36  public class Movie implements Serializable {
37
38      private static final long serialVersionUID = 1L;
39      @Id
40      @GeneratedValue(strategy = GenerationType.IDENTITY)
41      @Basic(optional = false)
42      @Column(name = "id")
43      private Integer id;
44      @Basic(optional = false)
45      @NotNull
        @Size(min = 1, max = 50)
        @Column(name = "name")
```

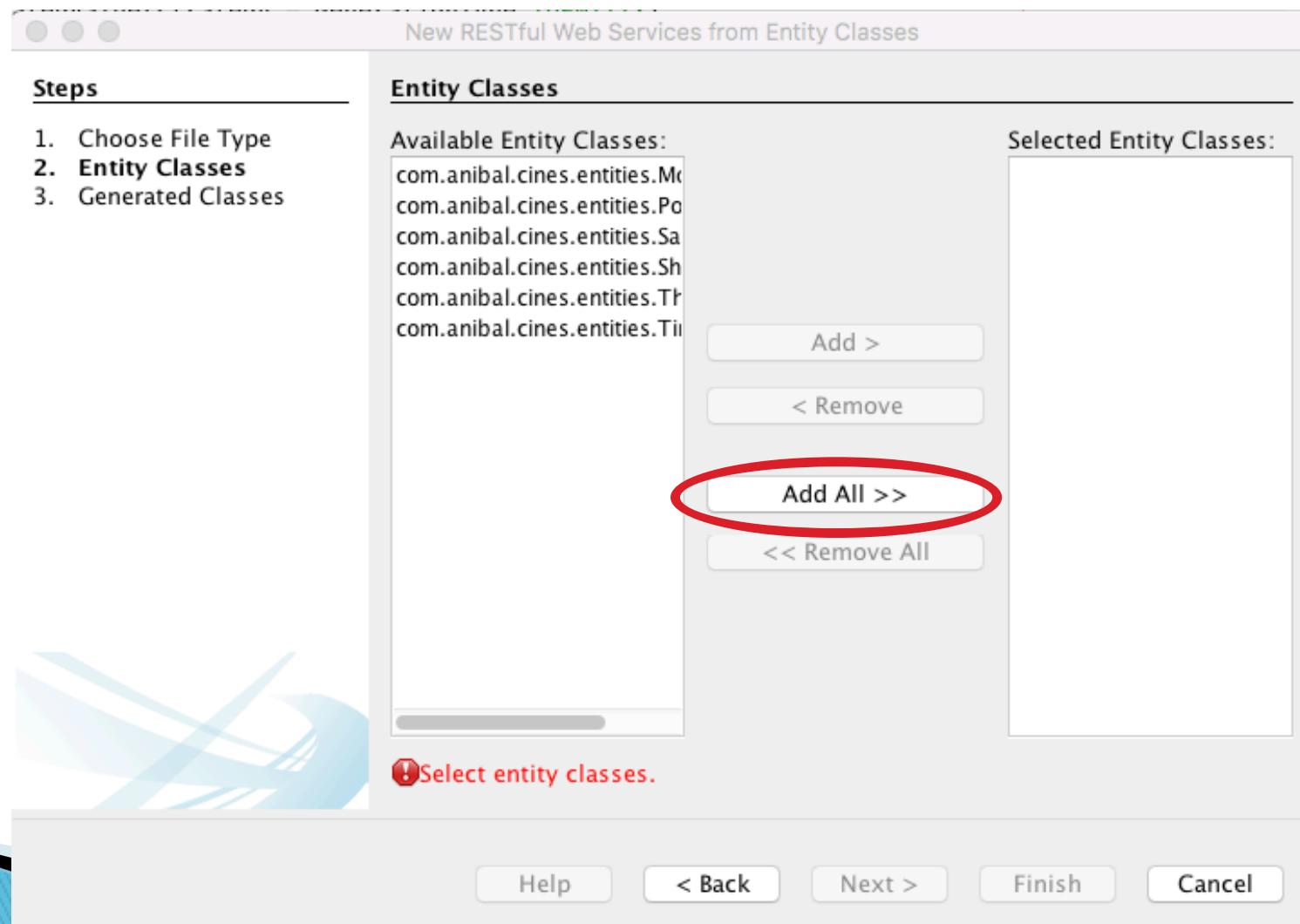
Servicios REST desde clases entidad

- ▶ Creamos servicios REST de clases entidad



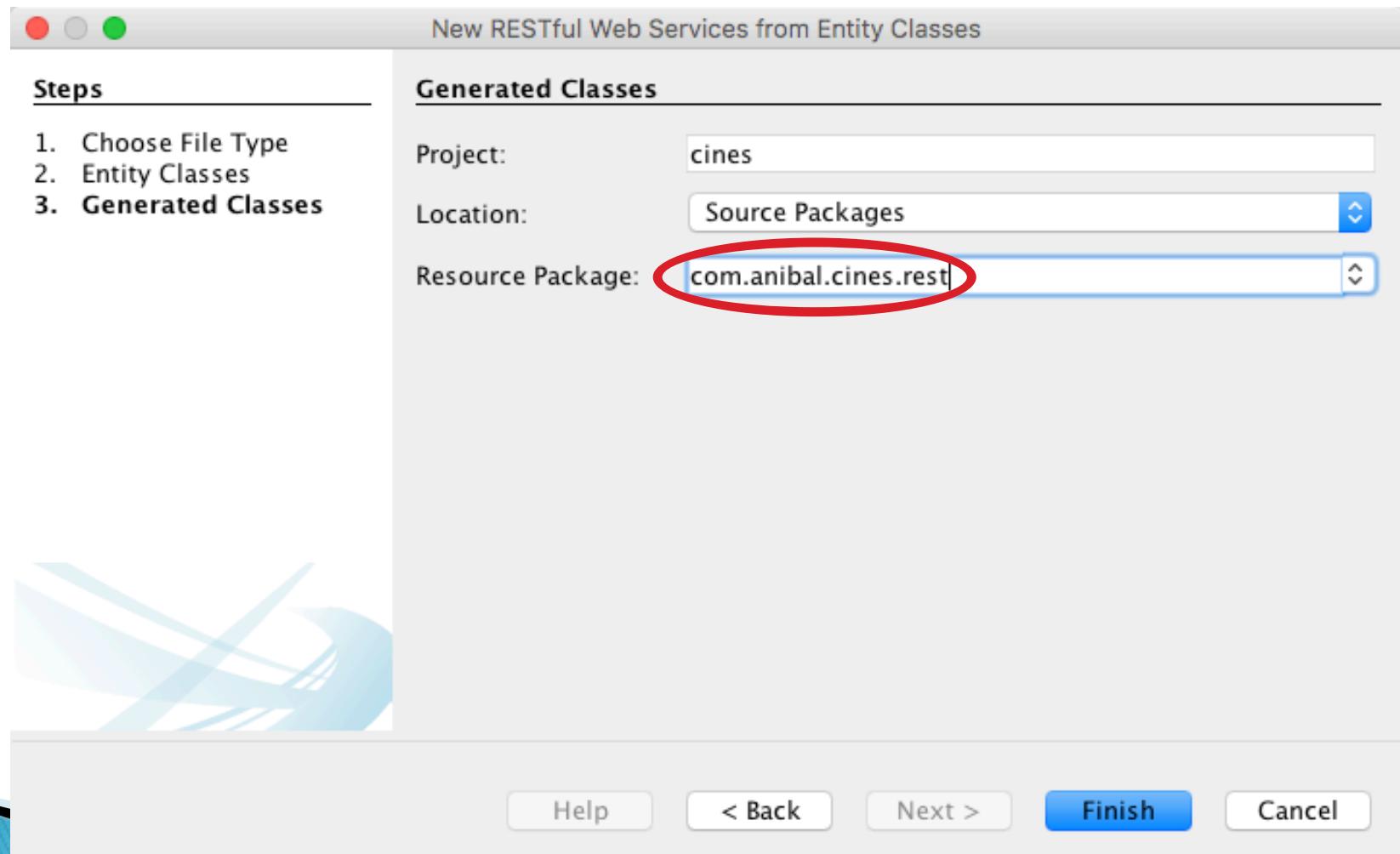
Servicios REST desde clases entidad

▶ Añadimos todas



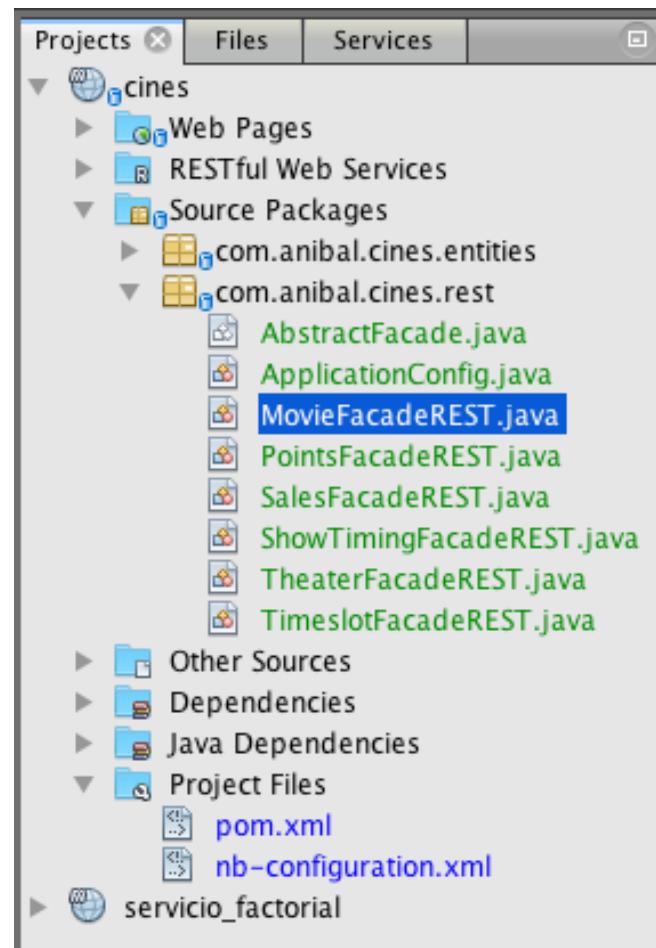
Servicios REST desde clases entidad

▶ Terminamos



Servicios REST desde clases entidad

▶ Resultado



JSF: template

- ▶ Vamos a definir el template con los siguientes componentes:

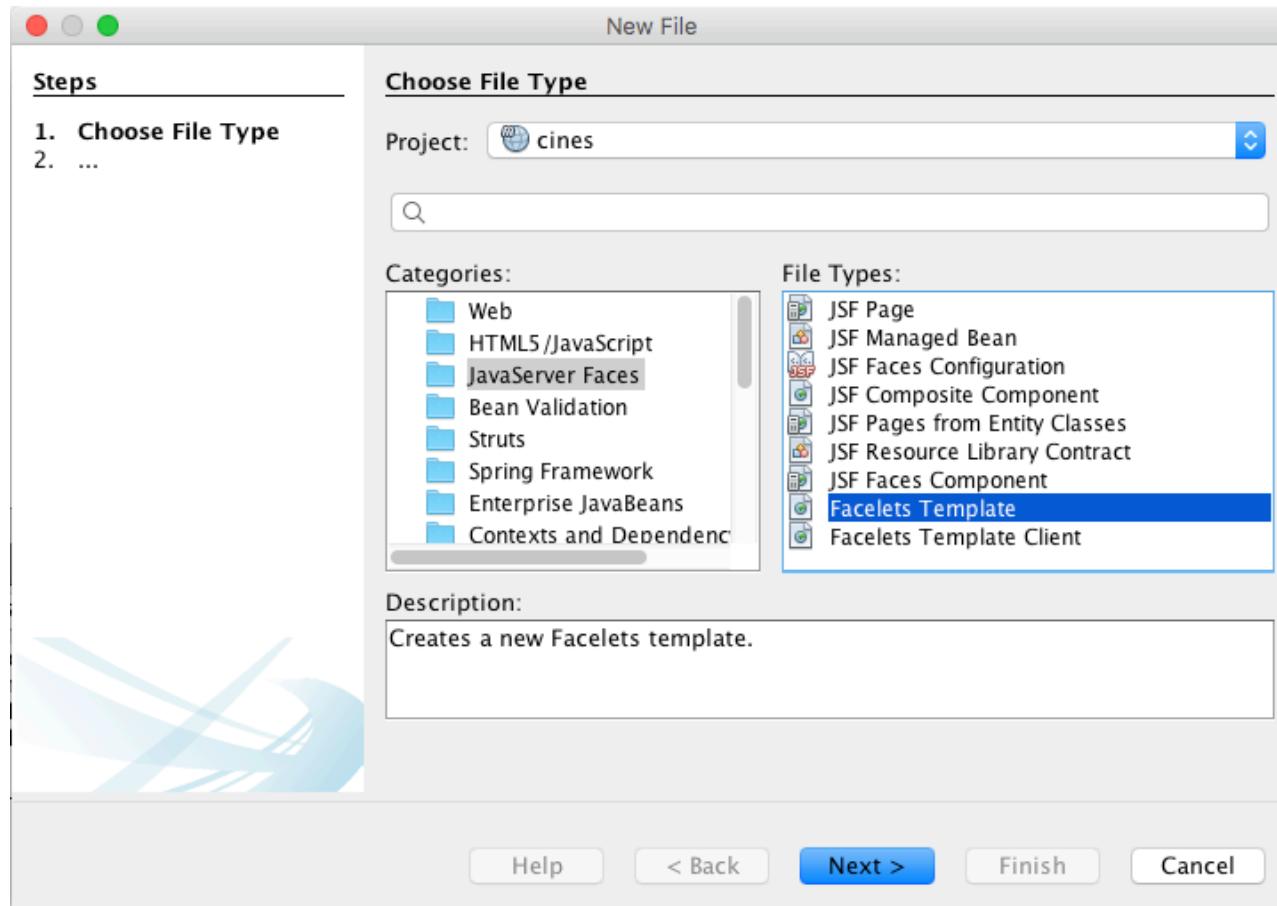
CABECERA

MENÚ

CONTENIDOS

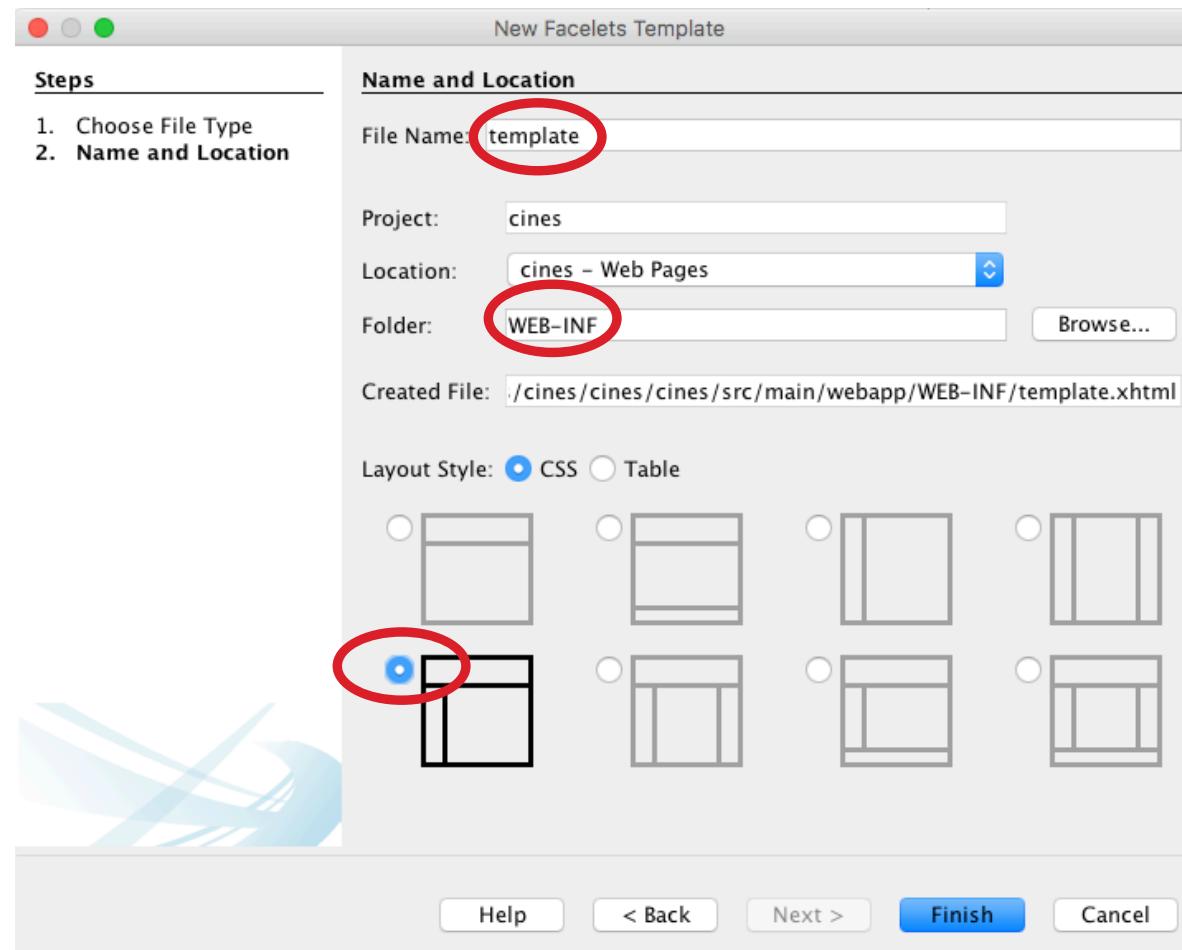
JSF: template

▶ Creamos un template



JSF: template

- ▶ Elegimos nombre, carpeta y layout



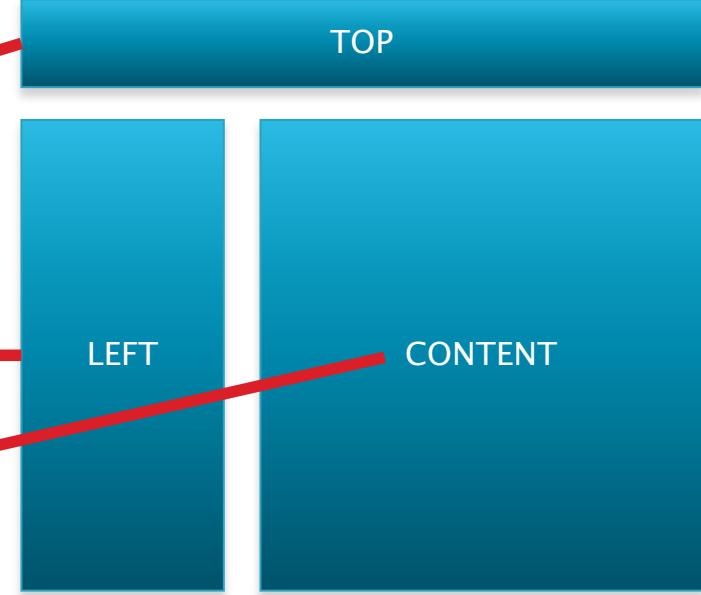
JSF: template

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

<h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <h:outputStylesheet name=".css/default.css"/>
    <h:outputStylesheet name=".css/cssLayout.css"/>
    <title>Facelets Template</title>
</h:head>

<h:body>
    <div id="top" class="top">
        <ui:insert name="top">Top</ui:insert>
    </div>
    <div>
        <div id="left">
            <ui:insert name="left">Left</ui:insert>
        </div>
        <div id="content" class="left_content">
            <ui:insert name="content">Content</ui:insert>
        </div>
    </div>
</h:body>

</html>
```



JSF: template

- ▶ Cambiar la ruta de los ficheros css por:

```
<link href="${facesContext.getExternalContext().getRequestContextPath()}/resources/css/default.css"  
rel="stylesheet" type="text/css" />  
<link href="${facesContext.getExternalContext().getRequestContextPath()}/resources/css/cssLayout.css"  
rel="stylesheet" type="text/css" />
```

- ▶ En la cabecera (*top*) poned lo siguiente (en lugar del ui:insert... ya que va a ser constante a lo largo de toda la aplicación):

```
<h:form>  
    <h1><h:commandLink value="Cines Luz de Castilla"  
    action="/index"/></h1>  
</h:form>
```

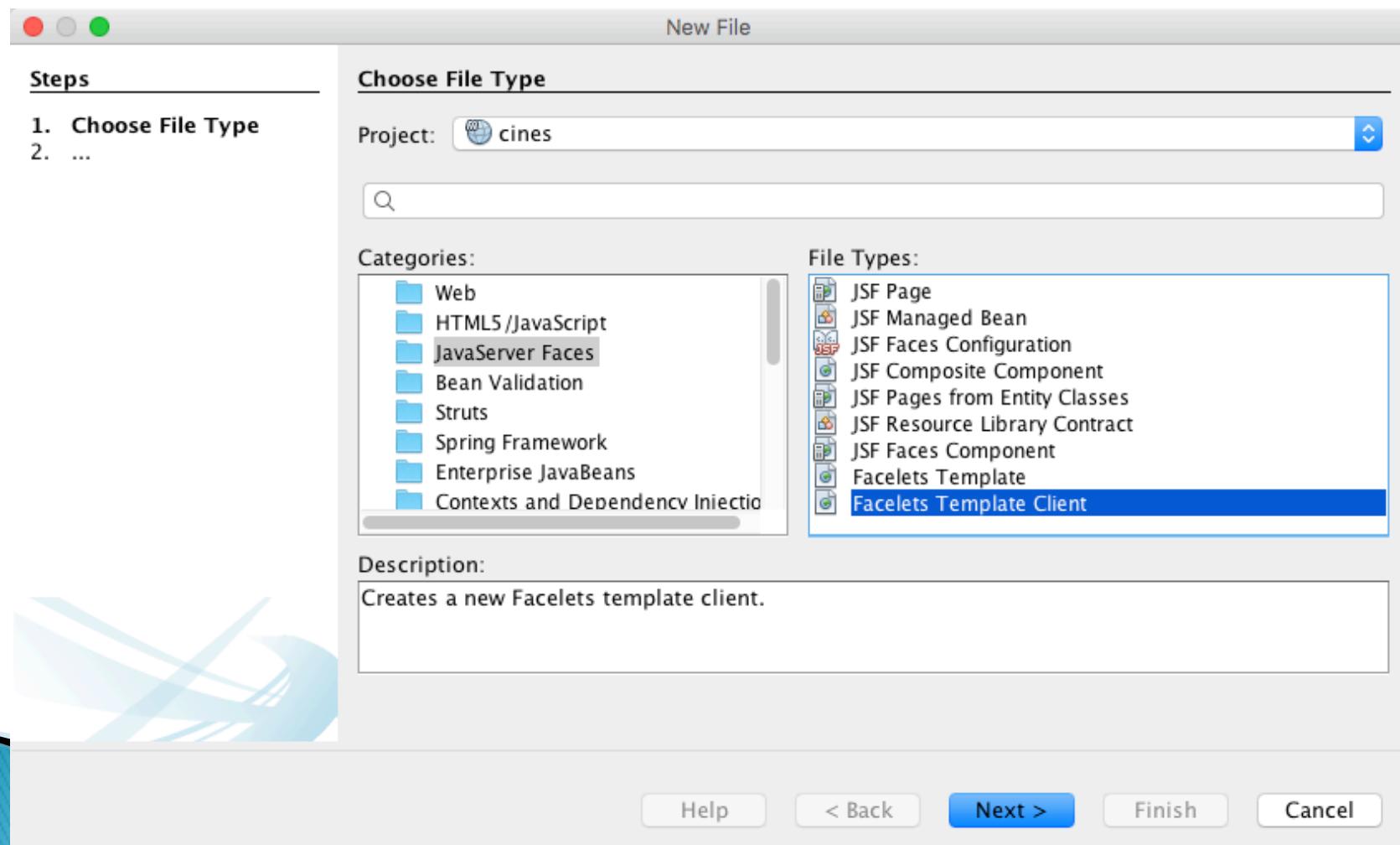
JSF: template

- ▶ En el menú de la izquierda poned lo siguiente (en lugar del ui:insert...):

```
<h:form>
    <h:commandLink action="booking">Comprar entrada</h:commandLink>
    <p/><h:outputLink
value="${facesContext.getExternalContext.requestContextPath}/faces/chat/chatroom.xhtml">
    Chat
</h:outputLink>
<p/><h:outputLink
value="${facesContext.getExternalContext.requestContextPath}/faces/client/movies.xhtml">
    Películas
</h:outputLink>
</h:form>
```

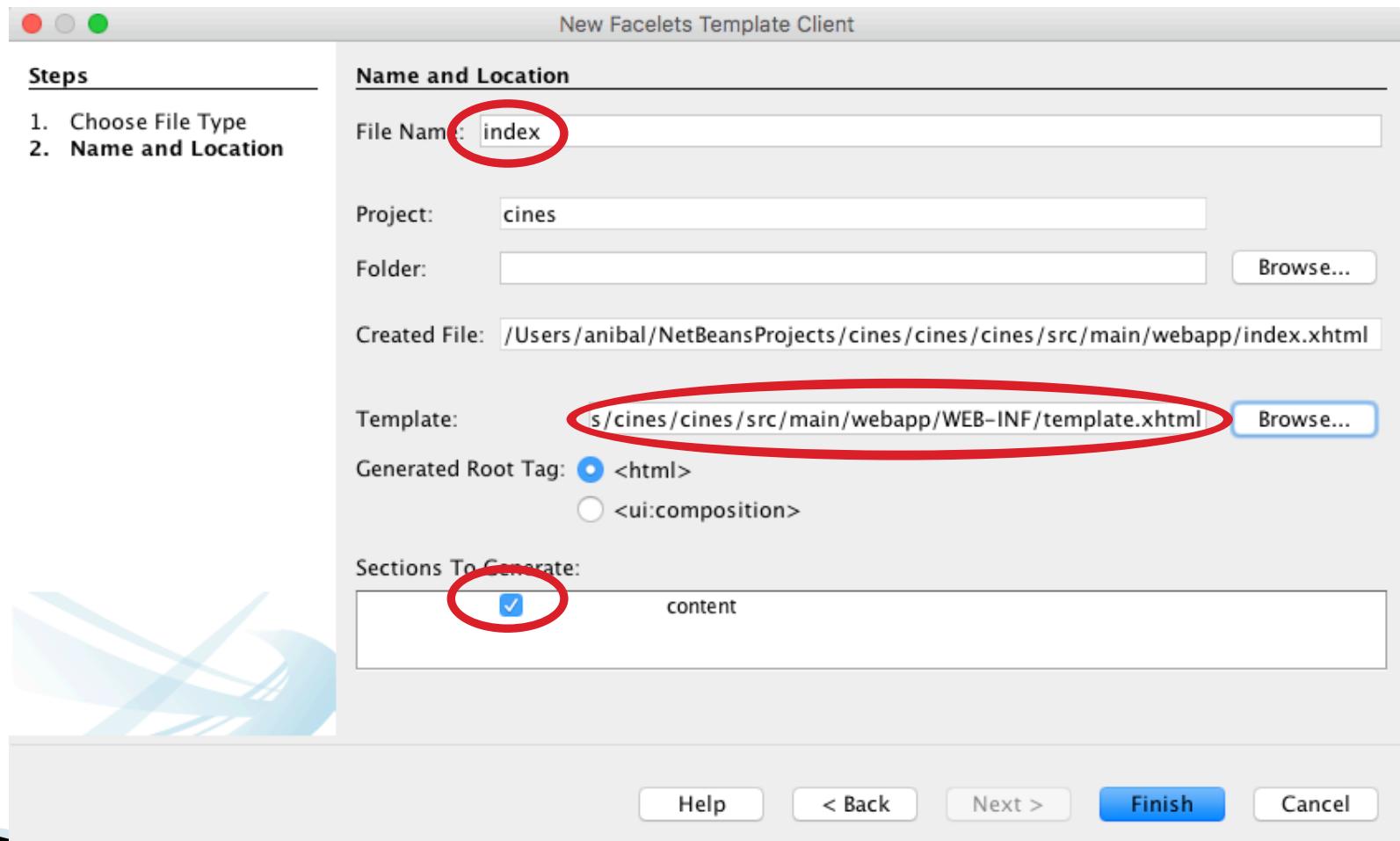
JSF: template

- ▶ Ahora hacemos que el index (template client) utilice el template
 - También lo podemos hacer de manera automática como sigue (**borrad antes el anterior index.xhtml**)



JSF: template

- ▶ Lo llamamos index.xhtml e indicamos el template a usar



JSF: template

▶ index.xhtml (así queda)

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<body>

    <ui:composition template=".//WEB-INF/template.xhtml">

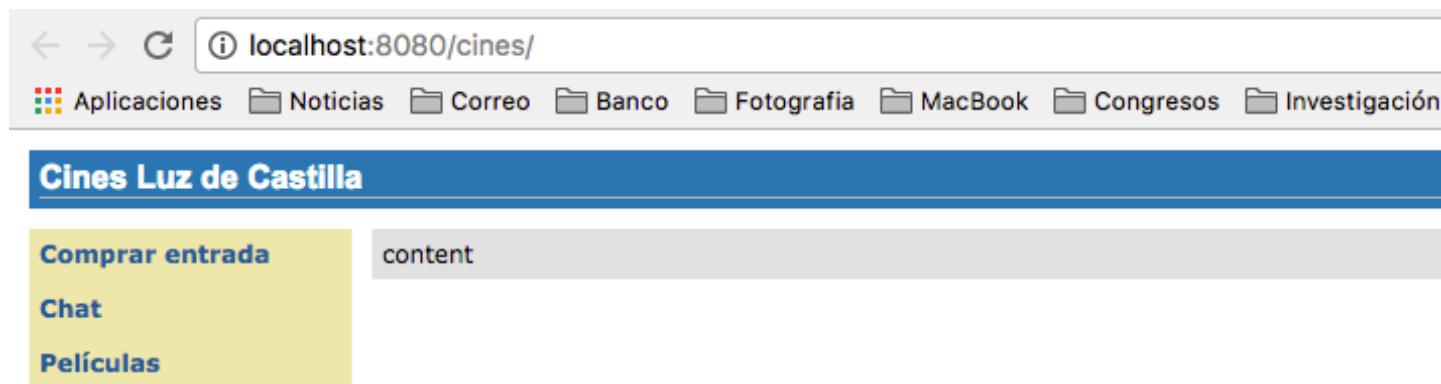
        <ui:define name="content">
            content
        </ui:define>

    </ui:composition>

</body>
</html>
```

JSF: template

- ▶ Si lo ejecutamos...



Mostrar la información de la BD

- ▶ Vamos a ver cómo acceder a la lógica de negocio desde la capa de presentación utilizando el **Expression Language (EL)**
- ▶ Para ello, haremos que en el index nos diga el número de películas y de cines que hay (y luego también añadiremos una tabla con las películas), para ello añadiremos lo siguiente:

```
<h:outputText value="Mostrando #{movieFacadeREST.countREST()}  
películas en #{theaterFacadeREST.countREST()} cines!"/>
```

- ▶ Añadir también el import necesario
- ▶ Después lo ejecutamos, y vemos que esto **NO** funciona... (no nos dice ni el número de películas ni de cines)... ¿a qué se debe?

Mostrar la información de la BD

- ▶ Nos falta inyectar las dependencias!!! (CDI)
 - Tenemos que poner @Named como anotación **a nivel de clase** en los servicios REST
- ▶ Lo probamos y vemos que funciona
 - Puede que nos toque hacer clean and build (y en casos extraordinarios quizá cerrar/abrir Netbeans) para que incluya las clases en el CDI.

Cines Luz de Castilla

[Comprar entrada](#)

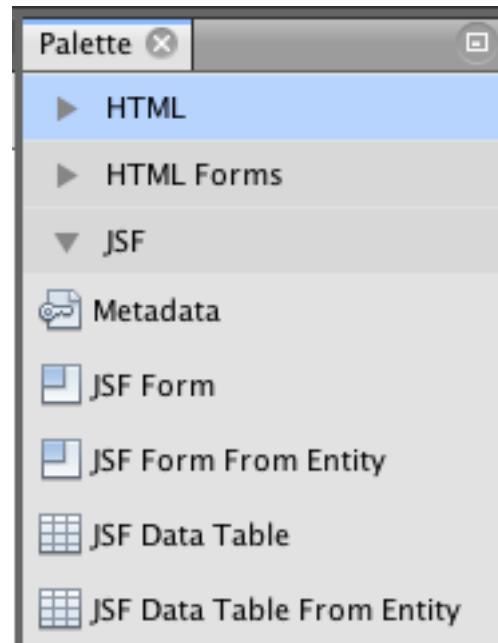
[Chat](#)

[Películas](#)

Mostrando 20 películas en 7 cines!

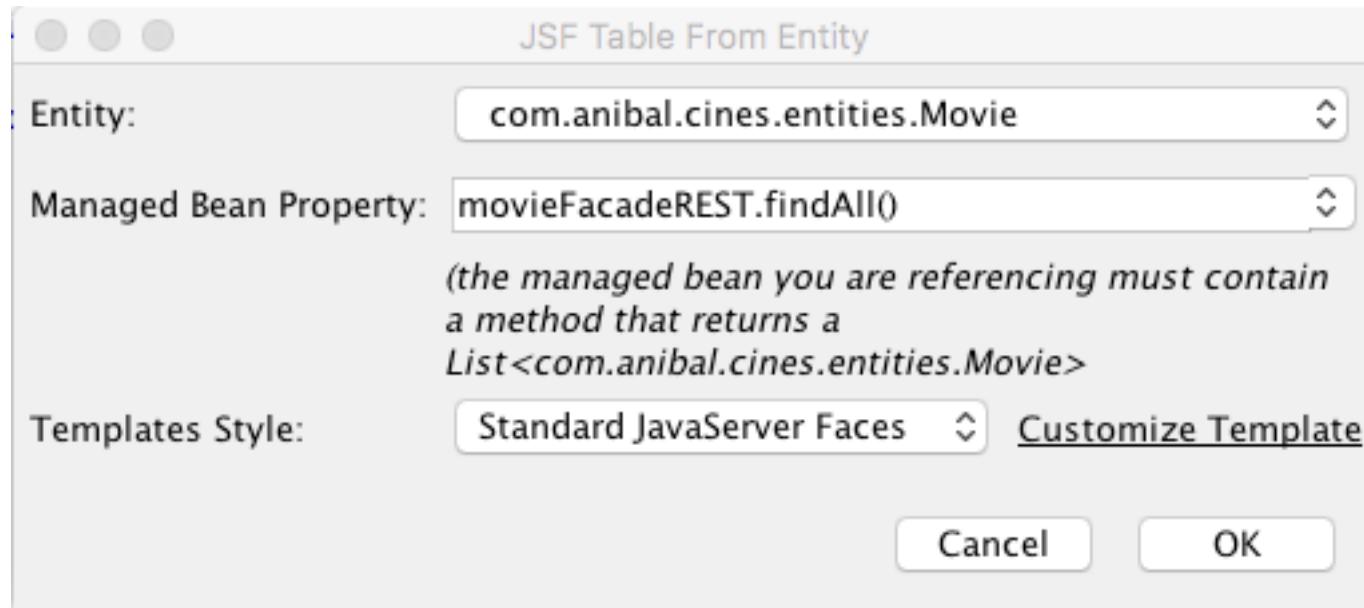
Mostrar la información de la BD

- ▶ Ahora vamos a añadir una tabla con la lista de películas
 - Podemos añadir una tabla desde las clases entidad directamente con la paleta (JSF Data Table From Entity)
 - Arrastramos y la ponemos en el xhtml (en donde queramos que aparezca la tabla)



Mostrar la información de la BD

- Decimos la clase entidad de la que obtener la información



Mostrar la información de la BD

```
<f:view>
  <h:form>
    <h1><h:outputText value="Lista de películas disponibles"/></h1>
    <h:dataTable value="#{movieFacadeREST.findAll()}" var="item">
      <h:column>
        <f:facet name="header">
          <h:outputText value="Id"/>
        </f:facet>
        <h:outputText value="#{item.id}"/>
      </h:column>
      <h:column>
        <f:facet name="header">
          <h:outputText value="Nombre"/>
        </f:facet>
        <h:outputText value="#{item.name}"/>
      </h:column>
      <h:column>
        <f:facet name="header">
          <h:outputText value="Actores"/>
        </f:facet>
        <h:outputText value="#{item.actors}"/>
      </h:column>
    </h:dataTable>
  </h:form>
</f:view>
```

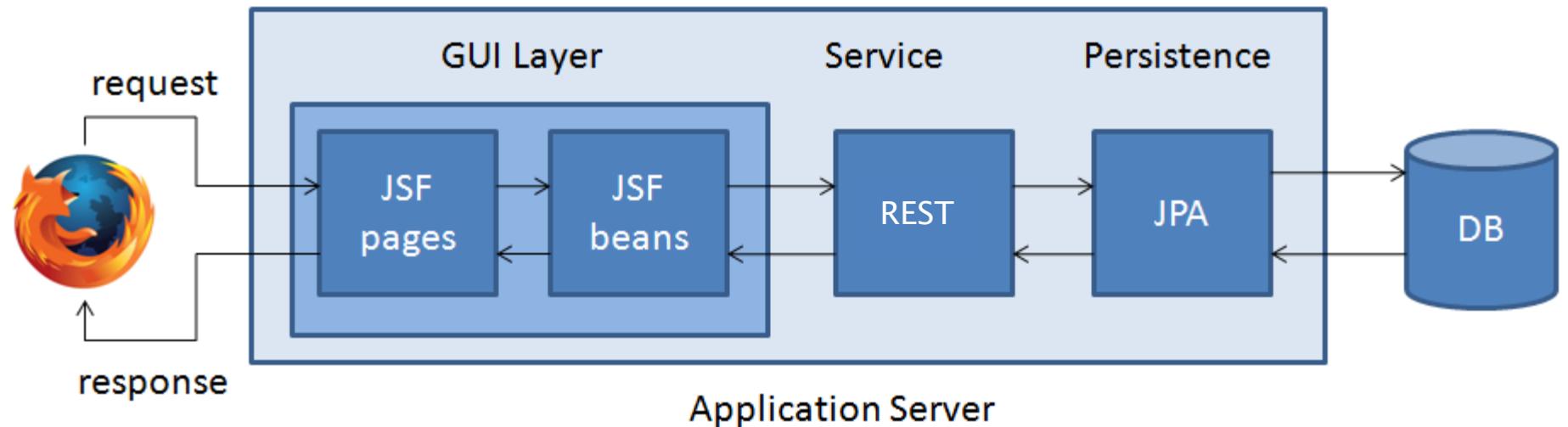
Mostrar la información de la BD

- ▶ Lo ejecutamos y comprobamos que funciona

Cines Luz de Castilla			
Comprar entrada	Mostrando 20 películas en 7 cines!		
Chat	Lista de películas disponibles		
Películas	Id	Nombre	Actores
	1	The Matrix	Keanu Reeves, Laurence Fishburne, Carrie-Ann Moss
	2	The Lord of The Rings	Elijah Wood, Ian Mckellen, Viggo Mortensen
	3	Inception	Leonardo DiCaprio
	4	The Shining	Jack Nicholson, Shelley Duvall
	5	Mission Impossible	Tom Cruise, Jeremy Renner
	6	Terminator	Arnold Schwarzenegger, Linda Hamilton
	7	Titanic	Leonardo DiCaprio, Kate Winslet
	8	Iron Man	Robert Downey Jr, Gwyneth Paltrow, Terrence Howard
	9	Inglourious Bastards	Brad Pitt, Diane Kruger
	10	Million Dollar Baby	Hilary Swank, Clint Eastwood
	11	Kill Bill	Uma Thurman
	12	The Hunger Games	Jennifer Lawrence
	13	The Hangover	Bradley Cooper, Zach Galifianakis
	14	Toy Story	Tom Hanks, Michael Keaton
	15	Harry Potter	Daniel Radcliffe, Emma Watson
	16	Avatar	Sam Worthington, Sigourney Weaver
	17	Slumdog Millionaire	Anil Kapoor, Dev Patel, Freida Pinto
	18	The Curious Case of Benjamin Button	Brad Pitt, Cate Blanchett
	19	The Bourne Ultimatum	Matt Damon, Julia Stiles
	20	The Pink Panther	Steve Martin, Kevin Kline

Mostrar la información de la BD

- ▶ Situación por capas de la aplicación empresarial:



Añadimos el complemento PrimeFaces

- ▶ En propiedades/frameworks añadimos el complemento PrimeFaces
- ▶ Utilizamos el fichero pom.xml de Maven para hacerlo fácilmente
- ▶ En <dependencies> añadimos lo siguiente:

```
<dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>8.0</version>
</dependency>
```

Añadimos el complemento PrimeFaces

- ▶ Nos añade el complemento Primefaces
- ▶ Ahora ya podemos usar Primefaces añadiendo lo siguiente a la cabecera (nos lo sugerirá entre las opciones cuando pongamos código de primefaces (<p:)):

`xmlns:p="http://primefaces.org/ui"`

- Todos los elementos de PrimeFaces comenzarán con <p:....
- Para la lista de los elementos de PrimeFaces y ejemplos de su uso mirad en (cuidado que aquí vienen cosas de la versión 10):
 - <http://www.primefaces.org/showcase/>

Añadimos el complemento PrimeFaces

- ▶ Modificamos nuestro fichero template para añadir PrimeFaces
 - Primero sustituimos todo el contenido dentro de <h:body></h:body> por el layout

```
<p:layout fullPage="true">
```

....

....

```
</p:layout>
```

- Y después, dentro del layout cada una de sus “unidades”

Añadimos el complemento PrimeFaces

- ▶ Para la “unidad” de la cabecera, lo cambiamos de la siguiente manera:

- Antes:

```
<div id="top" class="top">
    <ui:insert name="top">
        <h:form>
            <h1><h:commandLink value="Cines Luz de Castilla" action="/index"/></h1>
        </h:form>
    </ui:insert>
</div>
```

- Ahora:

```
<p:layoutUnit position="north" size="100" resizable="true" closable="true" collapsible="true">
    <h2><h:outputText value= "Aplicación Empresarial Cines Luz de Castilla - Segovia"/></h2>
</p:layoutUnit>
```

Añadimos el complemento PrimeFaces

- ▶ Para la “unidad” de la izquierda antes teníamos lo siguiente:

```
<div id="left">
    <ui:insert name="left">
        <h:form>
            <h:commandLink action="booking">Comprar entrada</h:commandLink>
            <p><h:outputLink
                value="${facesContext.getExternalContext.requestContextPath}/faces/chat/chatroom.xhtml">
                Chat
            </h:outputLink>
            <p><h:outputLink
                value="${facesContext.getExternalContext.requestContextPath}/faces/client/movies.xhtml">
                Películas
            </h:outputLink>
        </h:form>
    </ui:insert>
</div>
```

Añadimos el complemento PrimeFaces

- ▶ Ahora ponemos lo siguiente:

```
<p:layoutUnit position="west" size="200" header="Menú" collapsible="true">
    <h:form>
        <p:menu>
            <p:menuItem value="Inicio" outcome="/index.xhtml" icon="ui-icon-home" />
        </p:menu>
    </h:form>
</p:layoutUnit>
```

Añadimos el complemento PrimeFaces

- ▶ Añadid por vuestra cuenta el cambio necesario en la unidad “content” del template para que funcione con Primefaces
- ▶ Haced los cambios que sean necesarios en el index para que la tabla quede con el formato de Primefaces (no es necesario hacerlo a mano, podéis usar la paleta)
- ▶ Además, investigad por vuestra cuenta las distintas opciones de Primefaces:

<http://www.primefaces.org/showcase/>

**...y acordaros del Clean and Build si no funciona
cuando lo vayáis a probar!!**

Lo ejecutamos...

Aplicación Empresarial Cines Luz de Castilla - Segovia

Menú 

 Inicio

Mostrando 20 películas en 7 cines!
Lista de películas disponibles

Id	Name	Actors
1	The Matrix	Keanu Reeves, Laurence Fishburne, Carrie-Ann Moss
2	The Lord of The Rings	Elijah Wood, Ian Mckellen, Viggo Mortensen
3	Inception	Leonardo DiCaprio
4	The Shining	Jack Nicholson, Shelley Duvall
5	Mission Impossible	Tom Cruise, Jeremy Renner
6	Terminator	Arnold Schwarzenegger, Linda Hamilton
7	Titanic	Leonardo DiCaprio, Kate Winslet
8	Iron Man	Robert Downey Jr, Gwyneth Paltrow, Terrence Howard
9	Inglorious Bastards	Brad Pitt, Diane Kruger
10	Million Dollar Baby	Hillary Swank, Clint Eastwood
11	Kill Bill	Uma Thurman
12	The Hunger Games	Jennifer Lawrence
13	The Hangover	Bradley Cooper, Zach Galifianakis
14	Toy Story	Tom Hanks, Michael Keaton
15	Harry Potter	Daniel Radcliffe, Emma Watson
16	Avatar	Sam Worthington, Sigourney Weaver
17	Slumdog Millionaire	Anil Kapoor, Dev Patel, Freida Pinto
18	The Curious Case of Benjamin Button	Brad Pitt, Cate Blanchett
19	The Bourne Ultimatum	Matt Damon, Julia Stiles
20	The Pink Panther	Steve Martin, Kevin Kline

¿PREGUNTAS?

