

Tema 1

Java EE – Primeros pasos

Instalación, REST, EJB, JSP, AJAX

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

Introducción

- ▶ Objetivos
 - Primera aproximación a Java EE
 - Instalación de componentes necesarios
 - Aprender los elementos necesarios para construir aplicaciones empresariales por capas
 - Conocer las principales APIs de Java EE
 - Crear una primera aplicación sencilla en Java EE
 - Aprender a desplegar y ejecutar la aplicación en un servidor de aplicaciones

¿Qué necesitamos?

- ▶ JDK
 - Necesitamos JDK 8 para que funcione el servidor de aplicaciones
- ▶ IDE
 - Vamos a usar NetBeans por ser la más fácil para aprender (la versión 12.0, que es la última estable, soportada por Apache, y se puede añadir el “bundle” para Java EE y los últimos servidores de aplicaciones)
 - Se podría usar cualquier otro IDE como IntelliJ IDEA o Eclipse, pero, a día de hoy, NetBeans es más sencillo para aprender los fundamentos de Java EE.
- ▶ Servidor de aplicaciones
 - Payara (la instalaremos por defecto con el IDE)
 - Se podría configurar un nuevo servidor (otra versión de Payara, Glassfish, Wildfly, etc...)
- ▶ Base de datos, etc... (los veremos en el siguiente tema)

Java JDK

Versión 8

Java JDK

- ▶ El servidor de aplicaciones requiere la versión 8 del JDK
- ▶ Si no la tenéis, descargadla de la web de Oracle (os tendréis que registrar, hacedlo con un email ficticio) e instaladla en vuestro ordenador.

[Java Platform, Standard Edition 8](#)

Java SE 8u281 is the latest release of Java SE 8 Platform.

Oracle strongly recommends that all Java SE 8 users upgrade to this release.

JDK for ARM releases are available on the same page as the downloads for other platforms

[Download](#)

[Release Notes](#)

NetBeans IDE

Versión 12.0

NetBeans IDE - para la versión 12.0

- ▶ Descargad la última versión del IDE NetBeans
 - <https://netbeans.apache.org/download/index.html>
- ▶ Aquí elegimos “Download” para descargarnos la versión 12
- ▶ La versión 12 ya viene con instalador, así que descargamos e instalamos la correspondiente a nuestro sistema operativo

NetBeans IDE - para la versión 12.0

The screenshot shows the Apache NetBeans website. At the top, there's a navigation bar with links for Community, Participate, Blog, Get Help, Plugins, and Download. Below the header, a blue banner highlights the "Latest release" Apache NetBeans 12.3, with a "Find out more" button.

Apache NetBeans Releases

Apache NetBeans is released four times a year. For details, see [full release schedule](#).

Apache NetBeans 12 feature update 3 (NB 12.3)

Latest version of the IDE, released on March 3, 2021.

[Download](#)

Apache NetBeans 12 LTS (NB 12.0)

Latest LTS version of the IDE, released on June 4, 2020.

[Features](#) [Download](#)

Older releases

Older Apache NetBeans releases and pre-Apache NetBeans releases can still be downloaded, but are no longer supported.

[Find out more](#)

NetBeans IDE - para la versión 12.0



Community Participate Blog Get Help Plugins Download

Downloading Apache NetBeans 12.0

Apache NetBeans 12.0 was released on June 4, 2020. See [Apache NetBeans 12.0 Features](#) for a full list of features.

Apache NetBeans 12.0 is available for download from your closest Apache mirror.

- Binaries: [netbeans-12.0-bin.zip \(SHA-512, PGP ASC\)](#)
- Source: [netbeans-12.0-source.zip \(SHA-512, PGP ASC\)](#)
- Installers:
 - [Apache-NetBeans-12.0-bin-windows-x64.exe \(SHA-512, PGP ASC\)](#)
 - [Apache-NetBeans-12.0-bin-linux-x64.sh \(SHA-512, PGP ASC\)](#)
 - [Apache-NetBeans-12.0-bin-macosx.dmg \(SHA-512, PGP ASC\)](#)
- Javadoc for this release is available at <https://bits.netbeans.org/12.0/javadoc>

[Deployment platforms](#)

[Community approval](#)

[Known problems](#)

[Earlier releases](#)

Officially, it is important that you [verify the integrity](#) of the downloaded files using the PGP signatures (.asc file) or a hash (.sha512 files). The PGP signatures should be matched against the [KEYS](#) file which contains the PGP keys used to sign this release.

Apache NetBeans can also be installed as a self-contained [snap package](#) on Linux.

Deployment platforms

Apache NetBeans 12.0 runs on the JDK LTS releases 8 and 11, as well as on JDK 14, i.e., the current JDK release at the time of this NetBeans release.

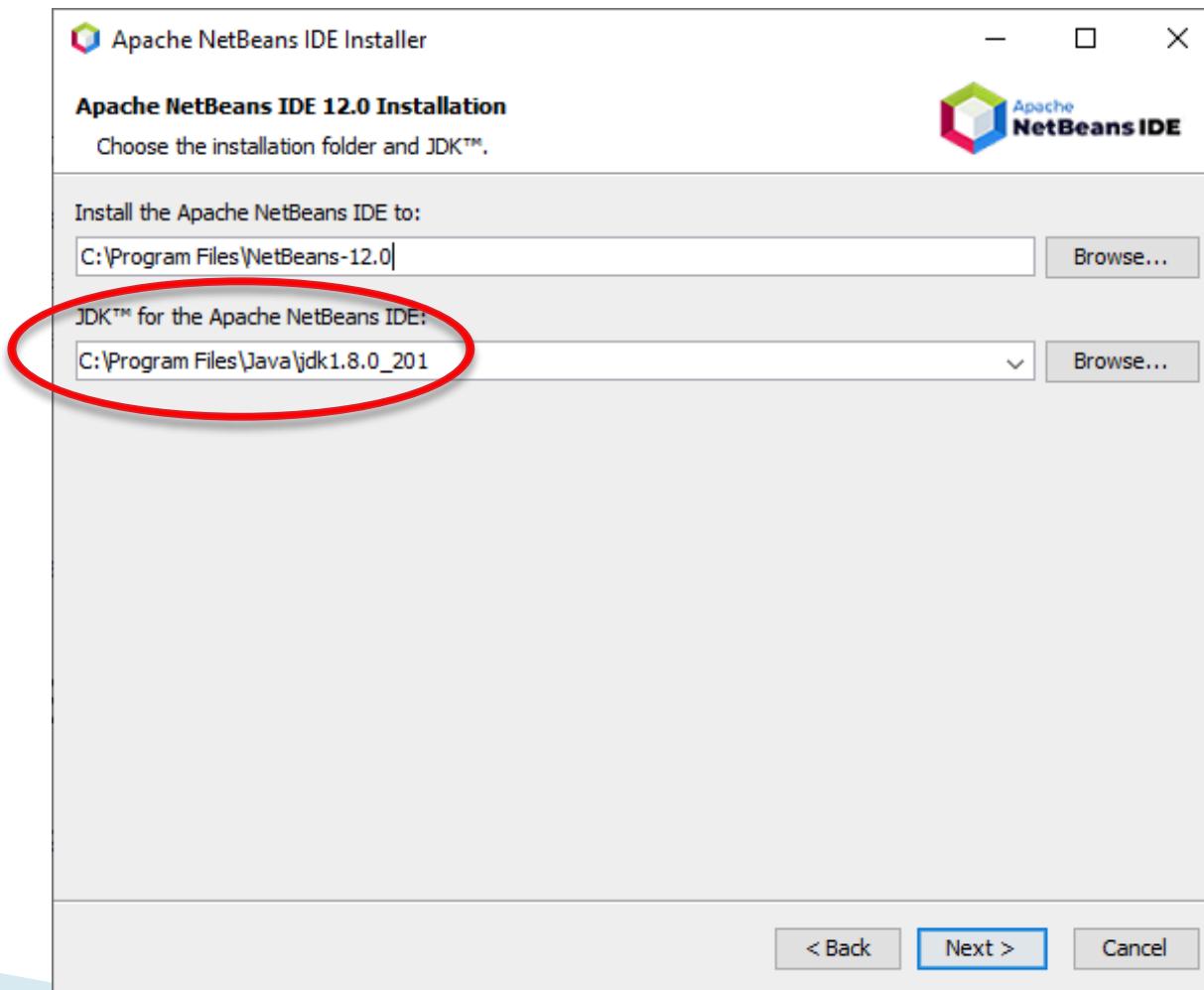
Community approval

As in any other Apache Project, the Apache NetBeans Community approved this release through the following voting processes in our [mailing lists](#):

- [PMC vote](#)
- [PMC vote result](#)

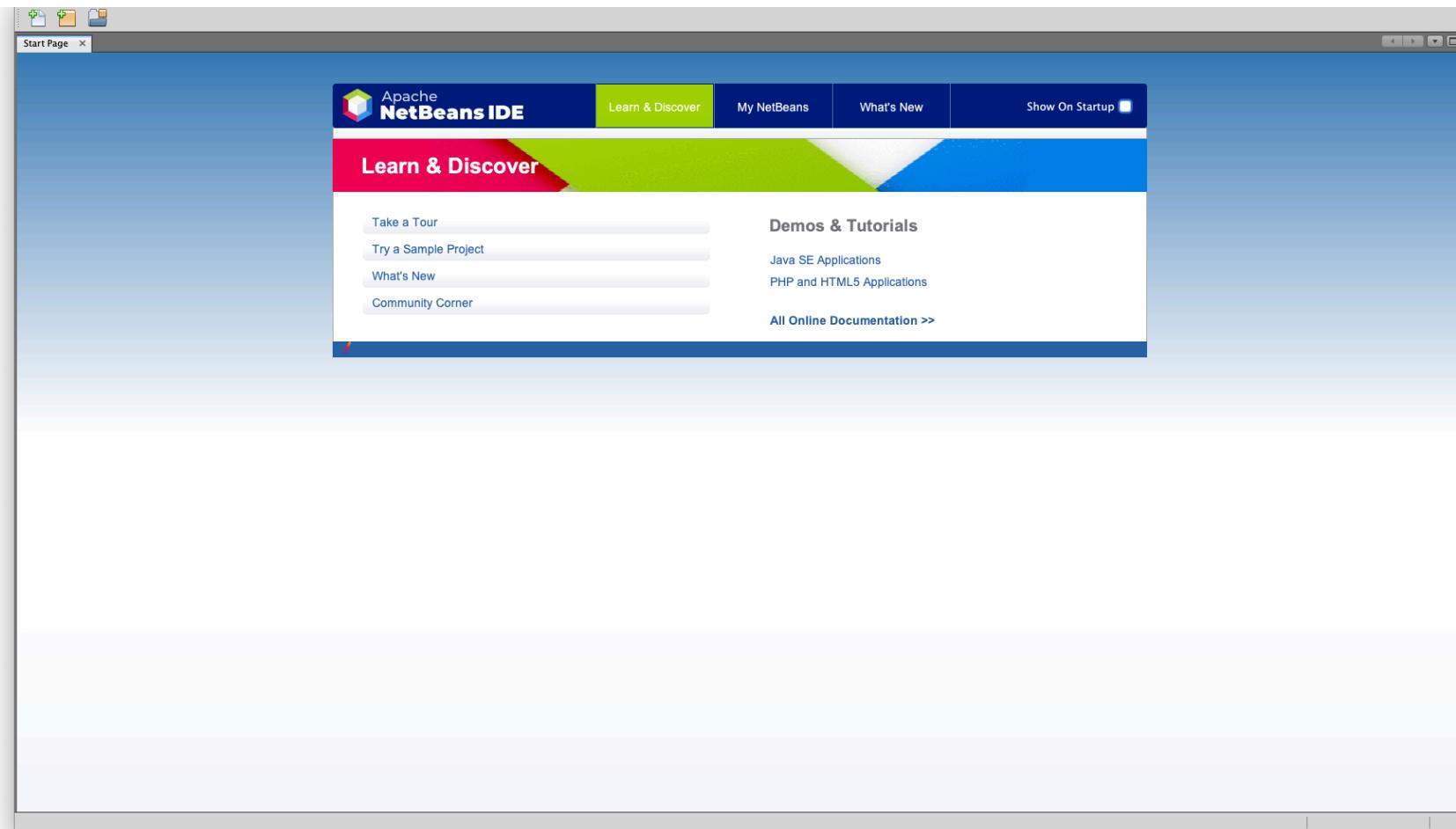
NetBeans IDE - para la versión 12.0

- ▶ Al instalarlo aseguraros de seleccionar la ruta a vuestro JDK 8



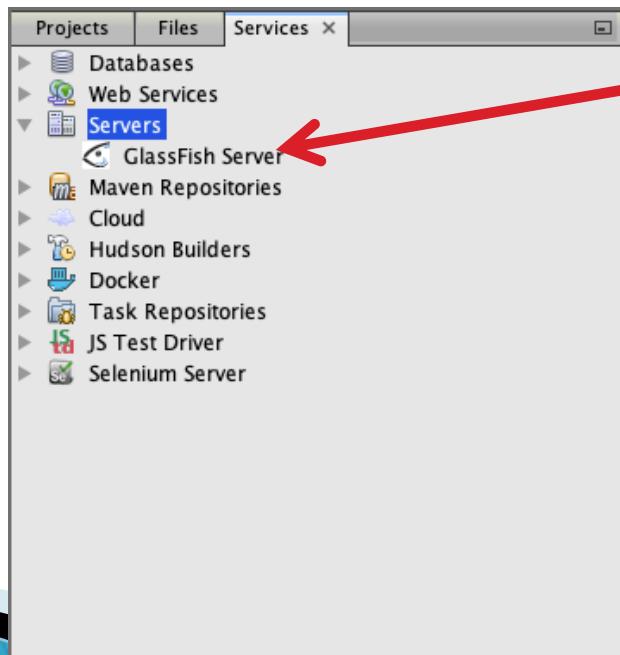
NetBeans IDE - para la versión 12.0

- ▶ Despues de instalarlo, arrancamos Netbeans

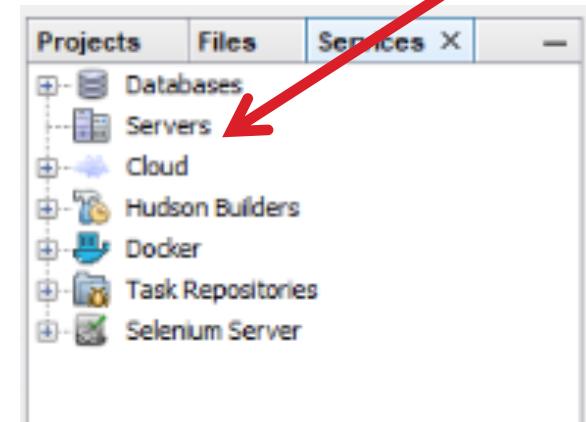


NetBeans IDE - para la versión 12.0

- ▶ En las distintas pestañas de la izquierda (cerrad la página de inicio para verlas – o sino Window/Projects y Window/Services) tenemos:
 - Los proyectos en desarrollo
 - Los ficheros
 - Los servicios (sobre todo bases de datos y servidores de aplicaciones)
- ▶ Dentro de los servicios podemos ver los servidores de aplicaciones

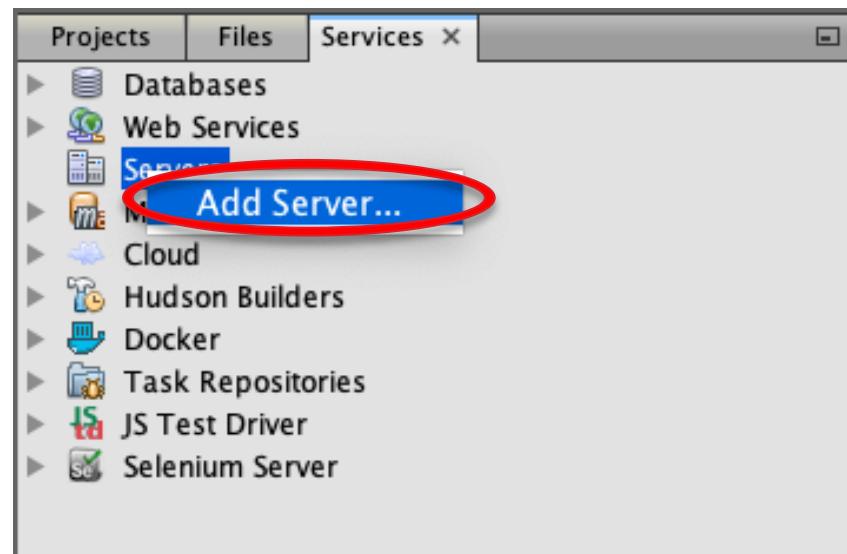


Dependiendo del SO y de la versión puede que esto os aparezca con un servidor por defecto **o que venga vacío**



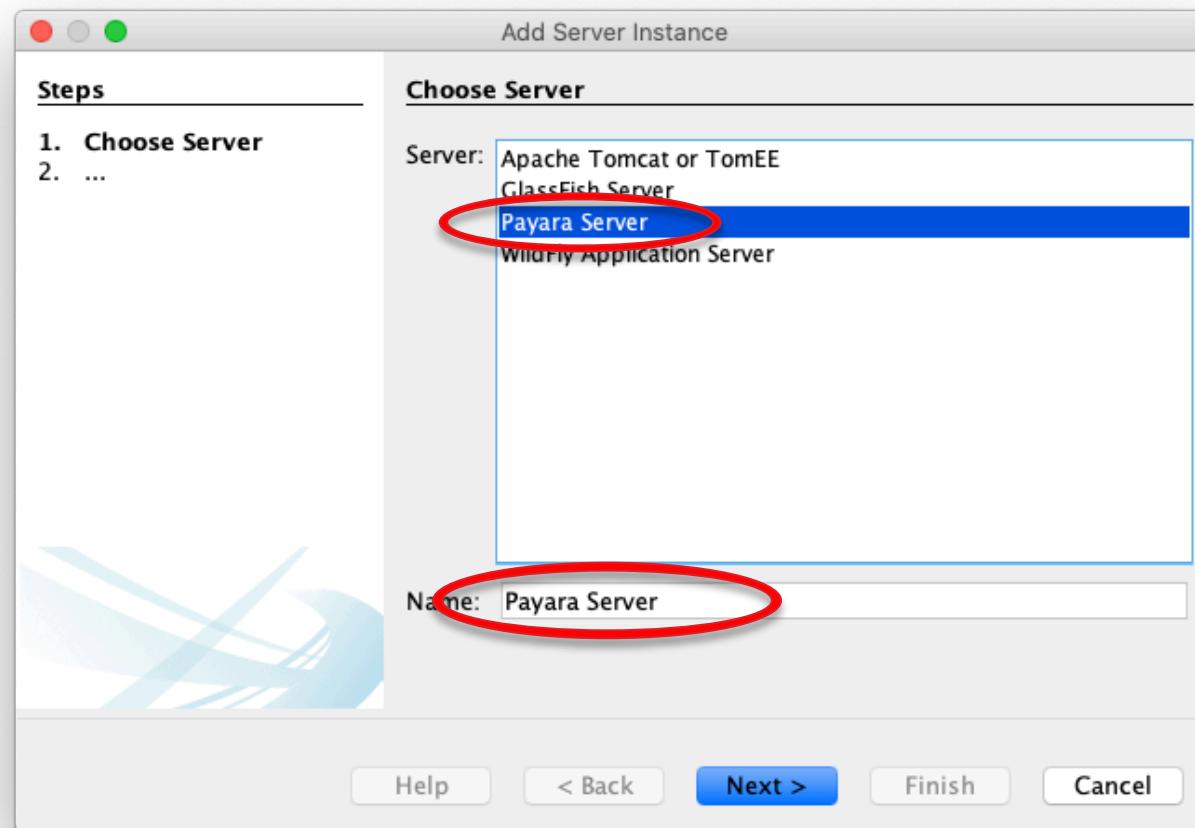
Configurar un nuevo servidor

- ▶ Si queremos instalar nuestro servidor de aplicaciones (**Payara**) lo podemos hacer directamente desde Netbeans
- ▶ En el menú Tools, seleccionamos Servers y pulsamos Add Server
- ▶ Esto también se puede hacer directamente pulsando con el botón derecho del ratón en “Servers” dentro de la pestaña “Services”)



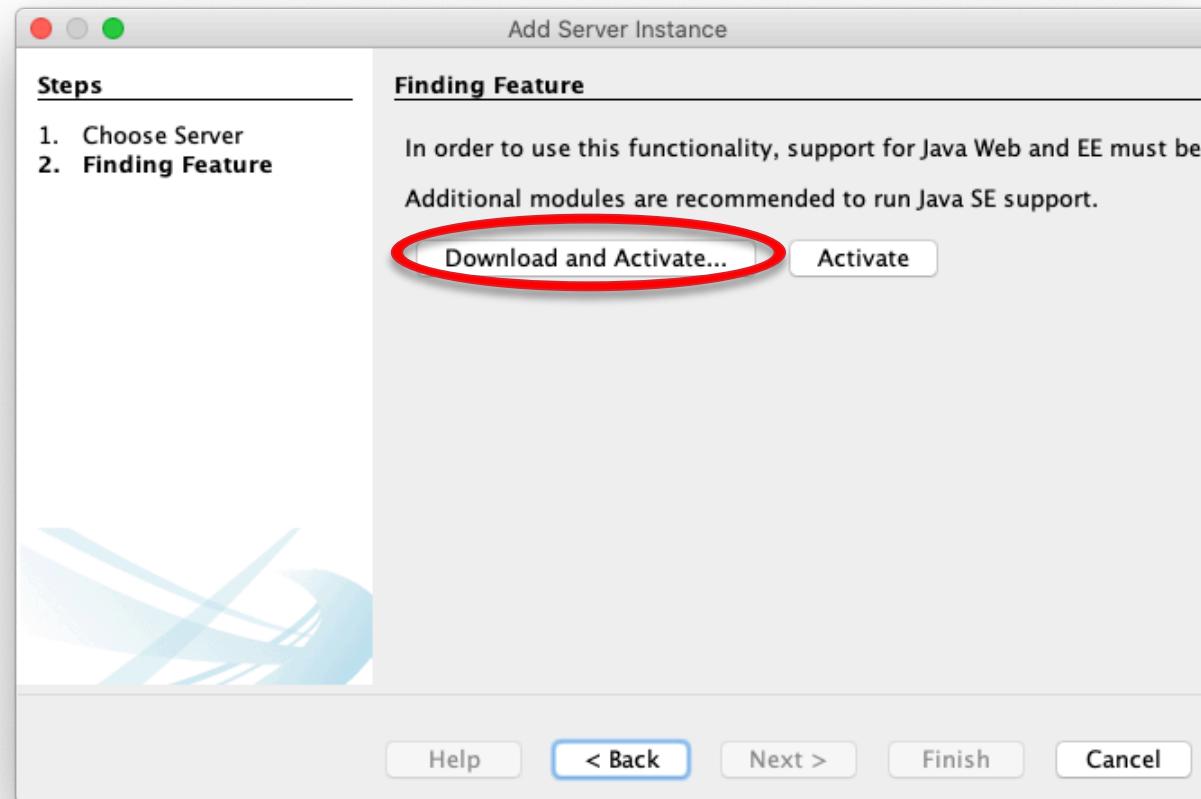
Configurar un nuevo servidor

- Elegimos el servidor Payara Server a instalar y le damos el nombre que queramos (por ejemplo “Payara Server” si no lo tenemos, si ya tenemos ese servidor, dadle otro nombre)



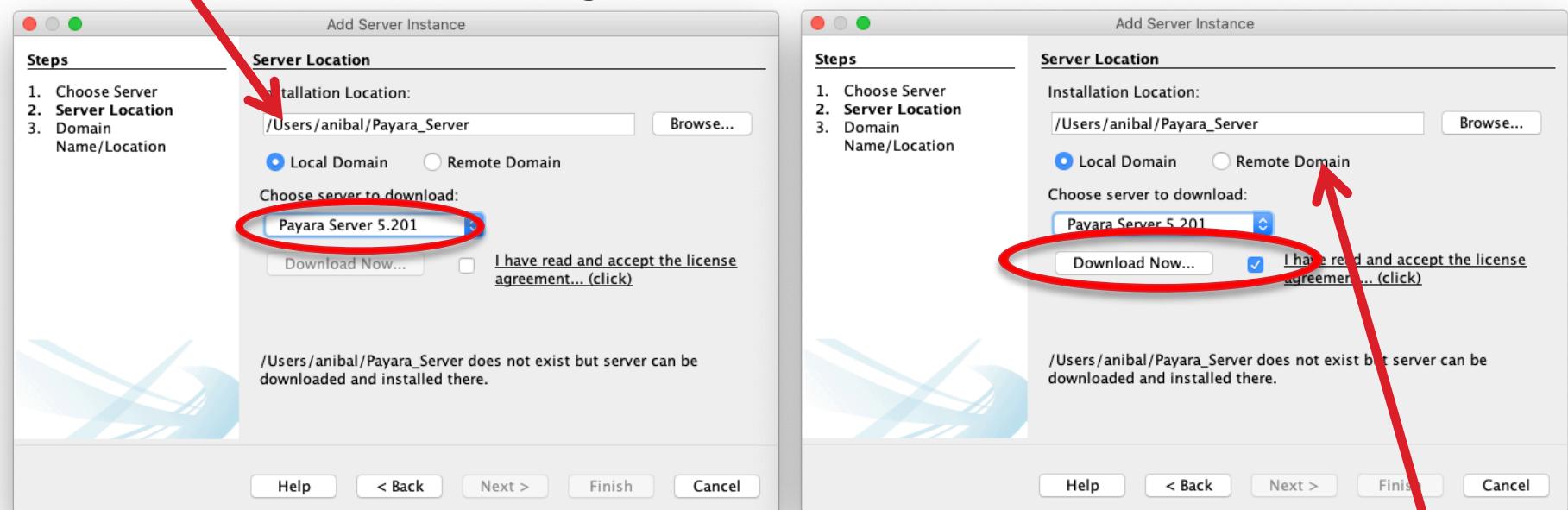
Configurar un nuevo servidor

- Si es la primera vez que lo instaláis y no lo habéis ejecutado nunca, os pedirá descargar y activar el soporte para Java Web y Java EE, que en Netbeans 12 no viene activado por defecto



Configurar un nuevo servidor

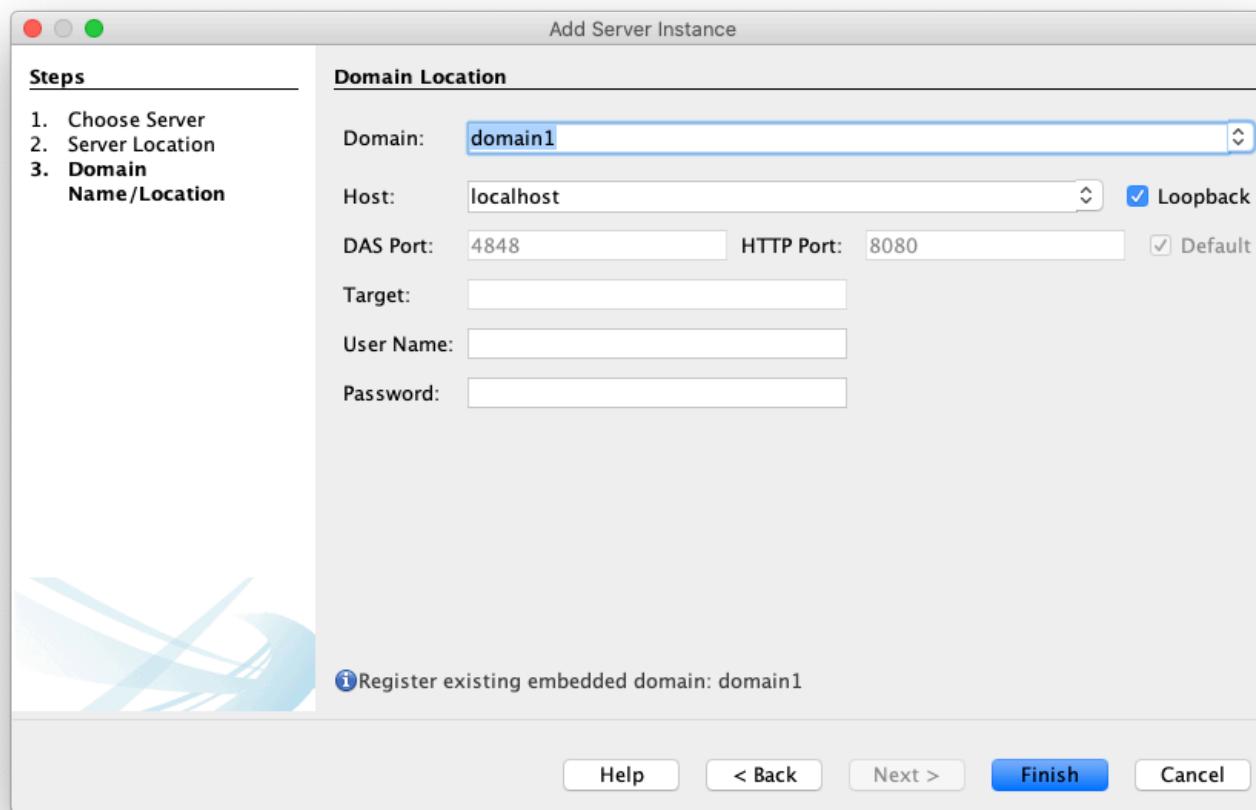
- ▶ Si nos hubiéramos descargado Payara por separado tendríamos que indicar la localización en la que lo hemos guardado
- ▶ En caso de no tener descargado ninguno, seleccionar uno del desplegable (**la versión, la 5.201**), aceptamos la licencia y presionamos en “Download Now...” y nos descargará el servidor de aplicaciones Payara en la ruta indicada
- ▶ Cuando termine de descargarla hacemos click en “Next”



También podríamos configurar un servidor remoto si lo necesitamos

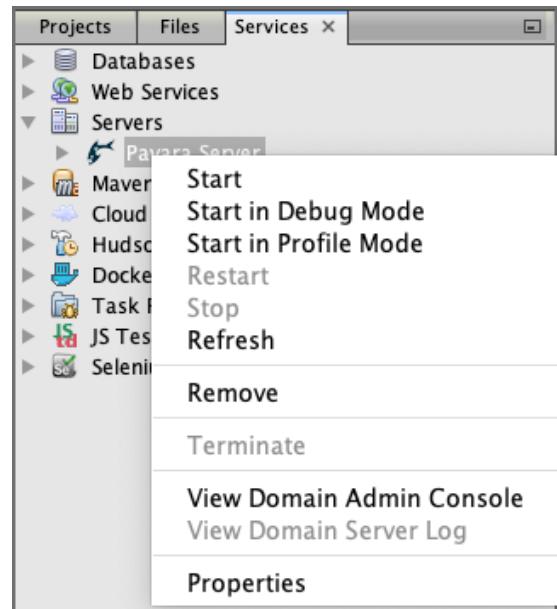
Configurar un nuevo servidor

- Terminamos la instalación registrando el dominio “domain1” (debería apareceros por defecto)



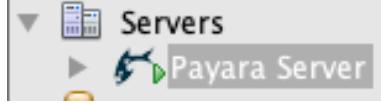
Servidor de aplicaciones

- ▶ Ahora ya debería aparecer el servidor de aplicaciones Payara → vamos a arrancarlo (pulsando el botón derecho sobre el servidor y Start)

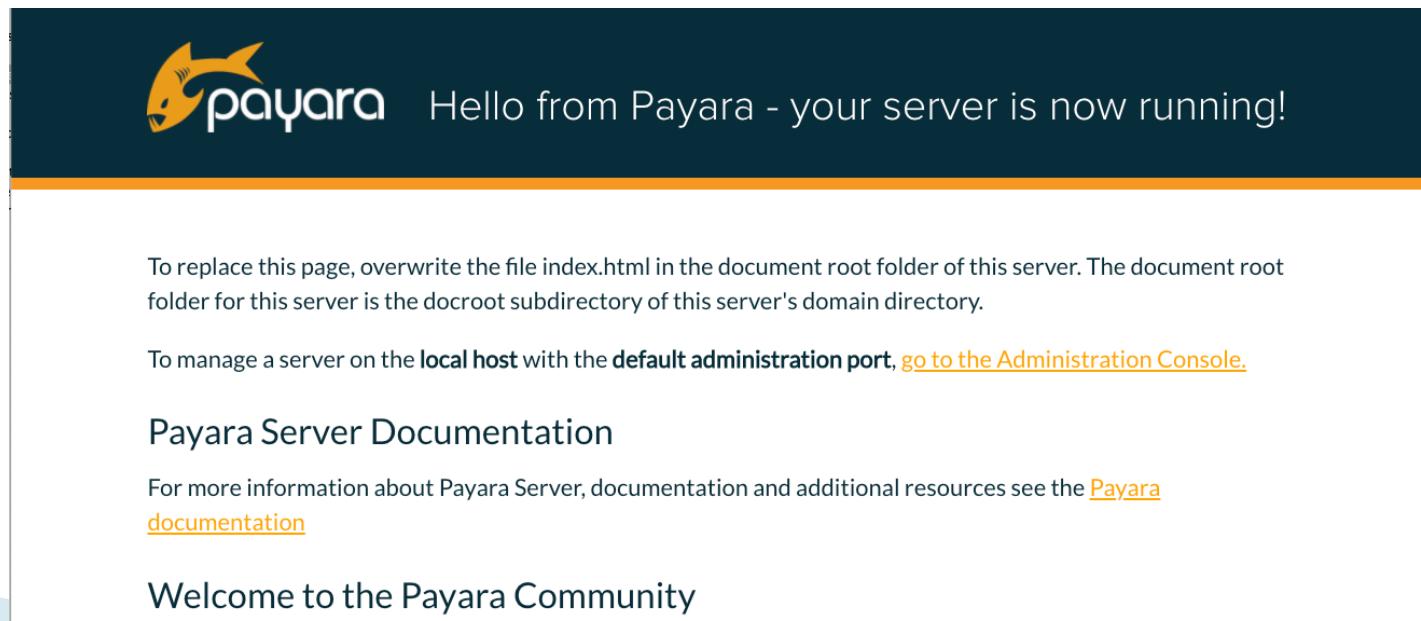


- ▶ Si tenéis cualquier otro servidor (Apache, etc...) ejecutándose en el puerto 8080 tenéis que pararlo!!

Servidor de aplicaciones

- ▶ Si aparece el símbolo del “play” en verde es que ha arrancado correctamente → 
- ▶ Podemos comprobar que ha arrancado correctamente entrando en:

localhost:8080



Hello from Payara - your server is now running!

To replace this page, overwrite the file index.html in the document root folder of this server. The document root folder for this server is the docroot subdirectory of this server's domain directory.

To manage a server on the **local host** with the **default administration port**, [go to the Administration Console](#).

Payara Server Documentation

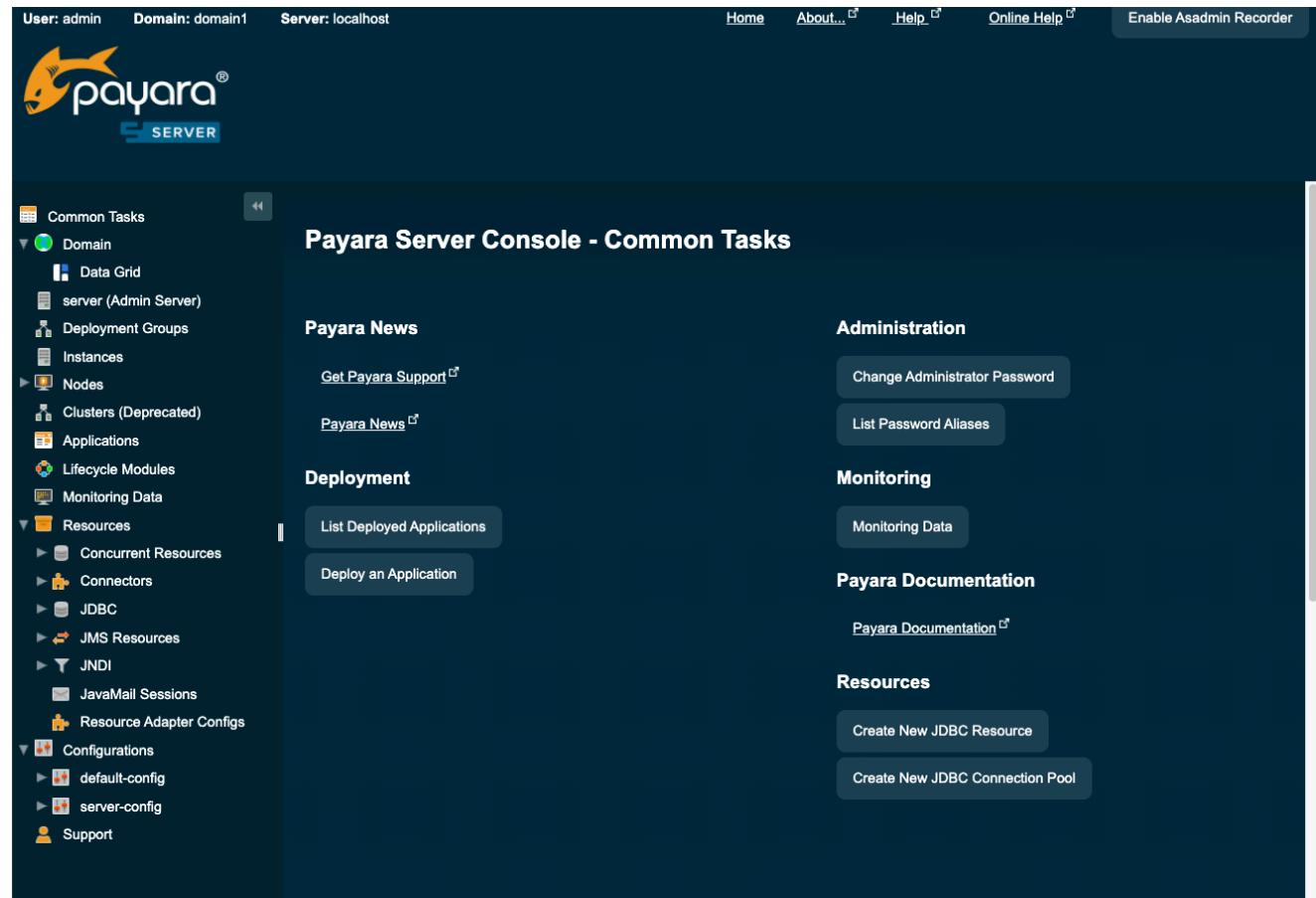
For more information about Payara Server, documentation and additional resources see the [Payara documentation](#)

Welcome to the Payara Community

Servidor de aplicaciones

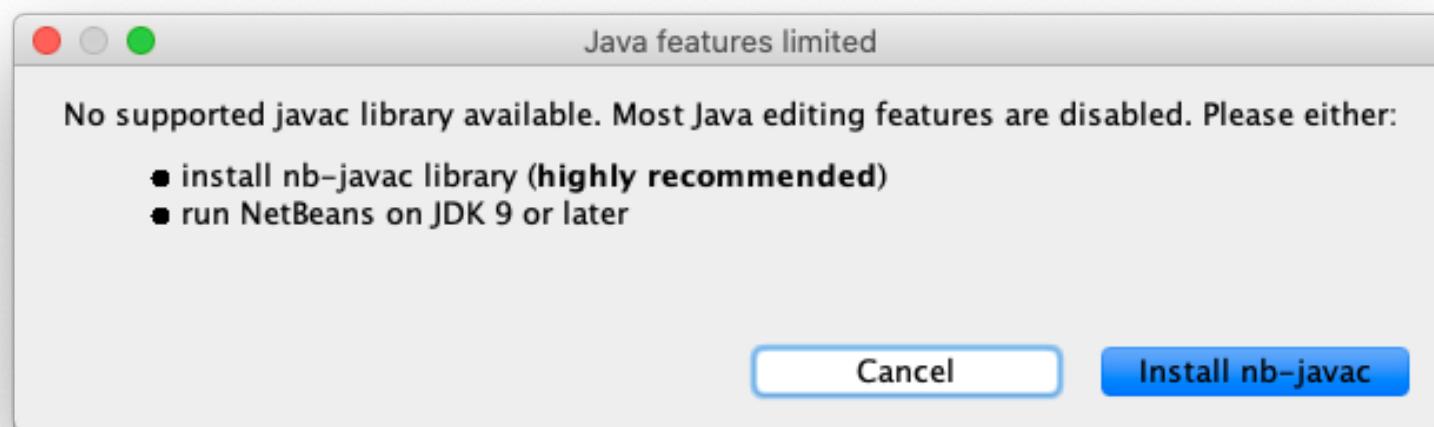
- ▶ O en el puerto de configuración de Payara (lo necesitaremos más adelante):

localhost:4848



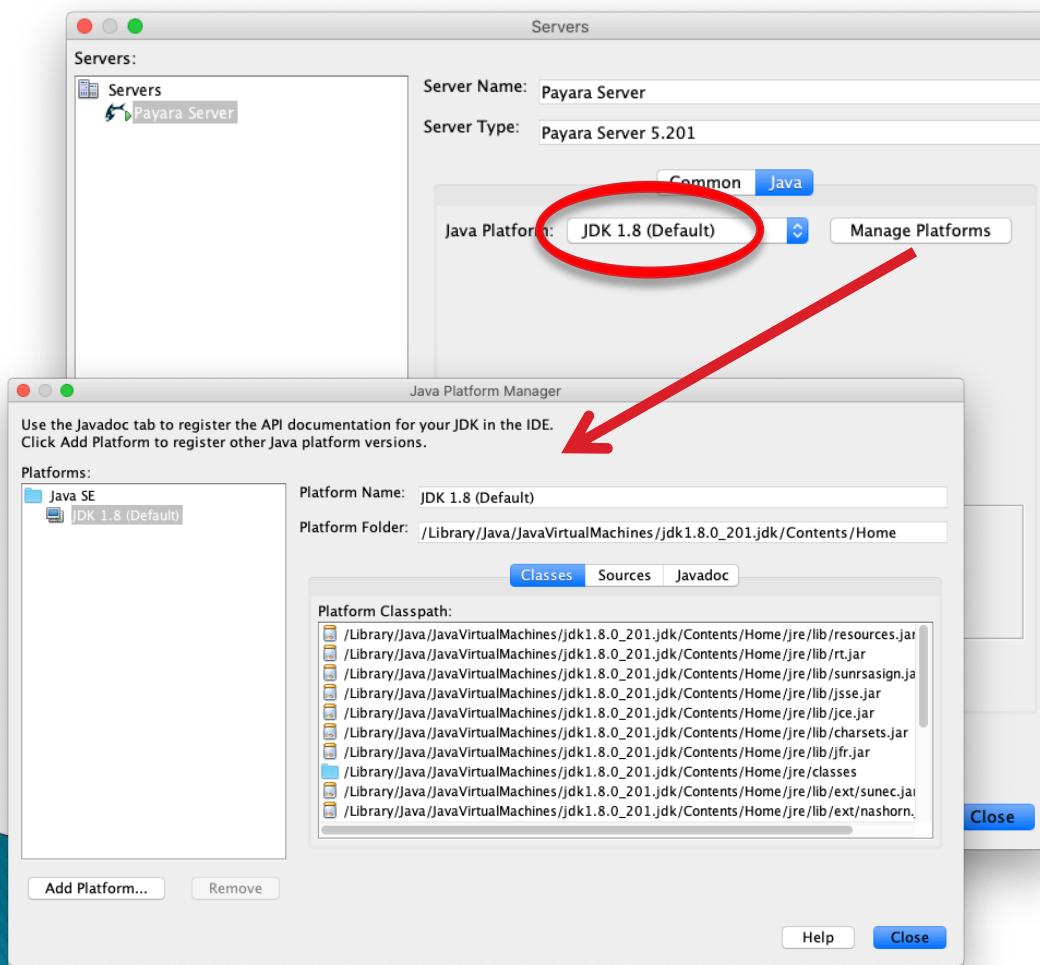
Para terminar...

- Como hemos visto, el servidor de aplicaciones sólo funciona con Java 8, así que si en algún momento (quizá al reiniciar el ordenador y Netbeans) puede que os pida instalar algún plugin, hacedlo, pero no cambiéis el JDK



Para terminar...

- ▶ Podéis aseguraros de la versión del JDK que estáis usando mirando en las propiedades de los servidores



- ▶ Aseguraros de que sea la versión 8 (que la habréis instalado en el primer punto)
- ▶ Si no tenéis el JDK 8 instalado o configurado en Netbeans, os lo tenéis que descargar y añadirlo pulsando el botón Manage Platforms

Otros servidores de aplicaciones – Wildfly

- ▶ WildFly/JBoss es otro de los servidores de aplicaciones más utilizados en Java EE:
 - <http://wildfly.org/downloads/>

The screenshot shows the WildFly website homepage. At the top, there is a navigation bar with links for Home, Downloads (which is highlighted in orange), News, About, Contribute, Docs, and a 'Fork' button. The main headline reads "Wild**Fly** 22.0.1 is now available". Below the headline are two yellow buttons labeled "DOWNLOAD THE ZIP" and "DOWNLOAD THE TGZ". A horizontal line separates this from a promotional text block. The text block contains the following message: "The technology behind Wild**Fly** is also available in [JBoss Enterprise Application Platform 7](#). JBoss EAP is a hardened enterprise subscription with Red Hat's world-class support, long multi-year maintenance cycles, and exclusive content. [Sign-up with Red Hat](#) to download a no-cost 1-year development license." At the bottom of this block is another yellow button labeled "DOWNLOAD RED HAT JBOSS EAP".

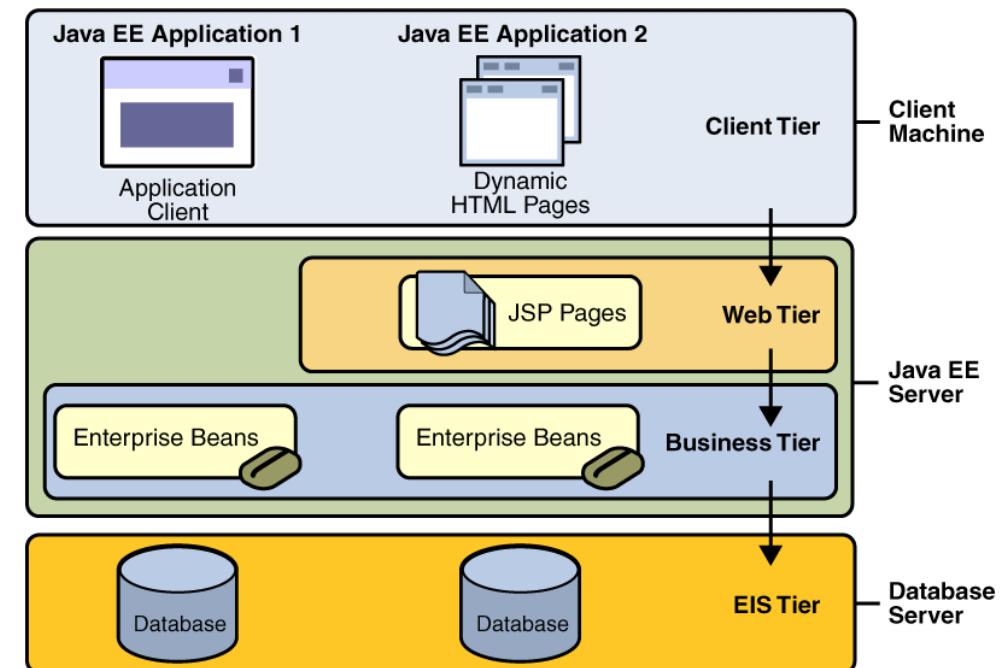
Otros servidores de aplicaciones – Glassfish

- ▶ Glassfish es el servidor de aplicaciones libre "oficial" de Oracle
- ▶ Payara Server deriva de GlassFish, pero cuenta con actualizaciones y parches más frecuentes

The screenshot shows the official GlassFish website. At the top, there's a dark header bar with the GlassFish logo and the text "The Open Source Java EE Reference Implementation". To the right is the Java™ Enterprise Edition logo. Below the header, the main content area has a light gray background. It features several sections: "Java EE 8 - GlassFish 5 Download" with links to "GlassFish 5.0.1 - Web Profile" and "GlassFish 5.0.1 - Full Platform"; a note about newer versions available from the Eclipse Foundation; "GlassFish Docker Images" with a link to details; "Java EE 8 RI" with a link to "downloads for Java EE 8"; and "Older GlassFish versions (archive)" with links to "GlassFish 5.0 - downloads", "GlassFish 4.1.2 - Web Profile | GlassFish 4.1.2 - Full Platform", and "GlassFish 4.1", "GlassFish 4.0", and "GlassFish 4.0". On the right side of the content area, there's a sidebar with links to "Sources", "Documentation", "Download", "Issue Tracker", "Mailing List", "Contribute", and "License". At the bottom right, it says "Sponsored by ORACLE".

Plataforma Java EE

- ▶ Ya hemos visto en clase las características principales de Java EE
- ▶ Ahora vamos a crear aplicaciones multi-capa utilizando Java EE.
- ▶ ¿Qué capas tenemos?
 - Cliente – navegador
 - La capa Web
 - La capa de Negocio
 - La capa de Persistencia
 - EIS – Enterprise Information System



Plataforma Java EE

▶ La capa cliente

- Clientes de las aplicaciones que acceden al servidor Java EE
- Los clientes hacen peticiones al servidor, que son procesadas y devueltas al cliente
- Los clientes pueden ser navegadores web, aplicaciones de escritorio, otros servidores.

Plataforma Java EE

- ▶ La capa Web – tecnologías Java EE
 - **Servlets**: Clases Java que procesan peticiones de manera dinámica y construyen las respuestas (normalmente como páginas HTML).
 - **JavaServer Pages (JSP)**: Documentos de texto que se compilan en servlets y definen cómo añadir contenido dinámico a páginas estáticas (como HTML).
 - **JavaServer Pages Standard Tag Library**: Librería de etiquetas que encapsula la funcionalidad común en páginas JSP.

Plataforma Java EE

- ▶ La capa Web – tecnologías Java EE
 - **JavaServer Faces**: Framework de componentes de interfaz de usuario para aplicaciones web que permite incluir componentes de la IU en una página, convertir y validar datos, almacenar los datos de los componentes de la IU en el lado del servidor, y mantener el estado.
 - **JavaServer Faces Facelets**: Tipo de aplicación JavaServer Faces. Usa XHTML en lugar de JSP.
 - **Expression Language**: Lenguaje de etiquetas usado en páginas JSP y Facelets para referirse a componentes Java EE.
 - **Y multitud de frameworks, componentes, etc: PrimeFaces, Bootsfaces, etc...**

Plataforma Java EE

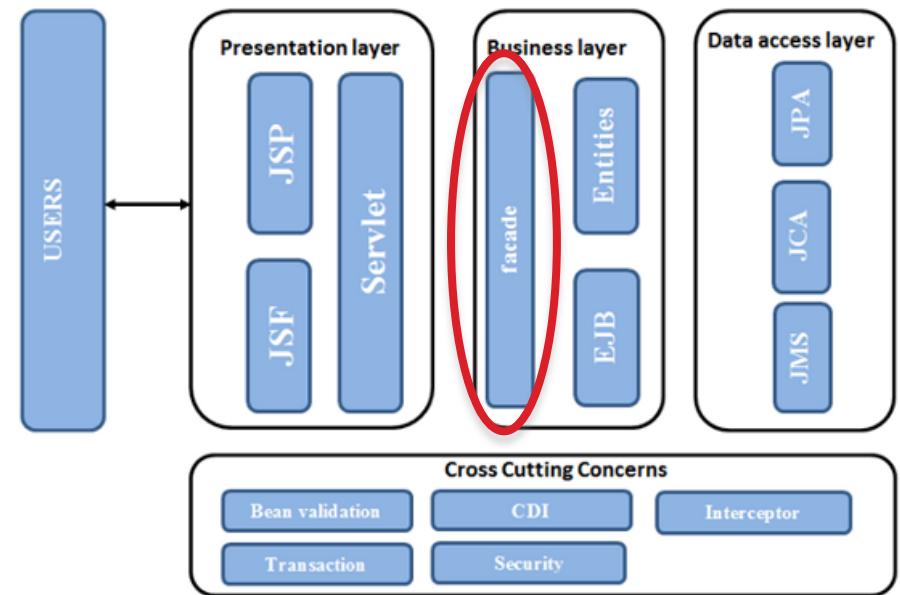
- ▶ La capa de negocio – tecnologías Java EE
 - Enterprise JavaBeans (EJBs): encapsulan la lógica de negocio
 - JAX-RS RESTful web services: servicios REST
 - JAX-WS web services: servicios SOAP
 - Java Persistence API entities: entidades de negocio
 - JSON-B: procesamiento de JSON
 - JAAS: autenticación y autorización

Plataforma Java EE

- ▶ La capa de persistencia – tecnología Java EE
 - Java Database Connectivity API (JDBC)
 - Java Persistence API (JPA)
 - Java Transaction API (JTA)

Primera aplicación Java EE

- ▶ Vamos a crear un servicio REST para calcular el **factorial de un número**
- ▶ Los servicios web se diseñan para ser independientes de los clientes
- ▶ Son accesibles públicamente y distintos tipos de clientes, a lo largo de Internet, pueden acceder a ellos
- ▶ Los servicios se tienen que desarrollar y desplegar de manera independiente



Primera aplicación Java EE

- ▶ **Servicios Web**
 - Estamos haciendo una aplicación empresarial y necesitamos saber la cotización de las distintas monedas... por ejemplo es una web de compra y queremos ver la tasa de cambio actual del Bitcoin... ¿cómo lo hacemos?
 - Tenemos que usar un servicio REST que nos proporcione la tasa de cambio instantáneo. Para ello necesitamos la URI que nos proporcione dicha información:

`api.coindesk.com/v1/bpi/currentprice.json`
 - Nuestra aplicación empresarial hará una petición GET a esa URI y el devolverá la tasa de cambio (lo podemos probar directamente en el navegador o usando aplicaciones como Postman, que veremos en detalle más adelante).

Primera aplicación Java EE

- ▶ **Servicios Web**
 - Otro ejemplo, ¿cómo hacemos una aplicación móvil o una página web que nos muestre la información meteorológica?
 - Nos vamos a Accuweather, que es uno de los proveedores meteorológicos más habituales y buscamos la información de su API REST:

<https://developer.accuweather.com/>

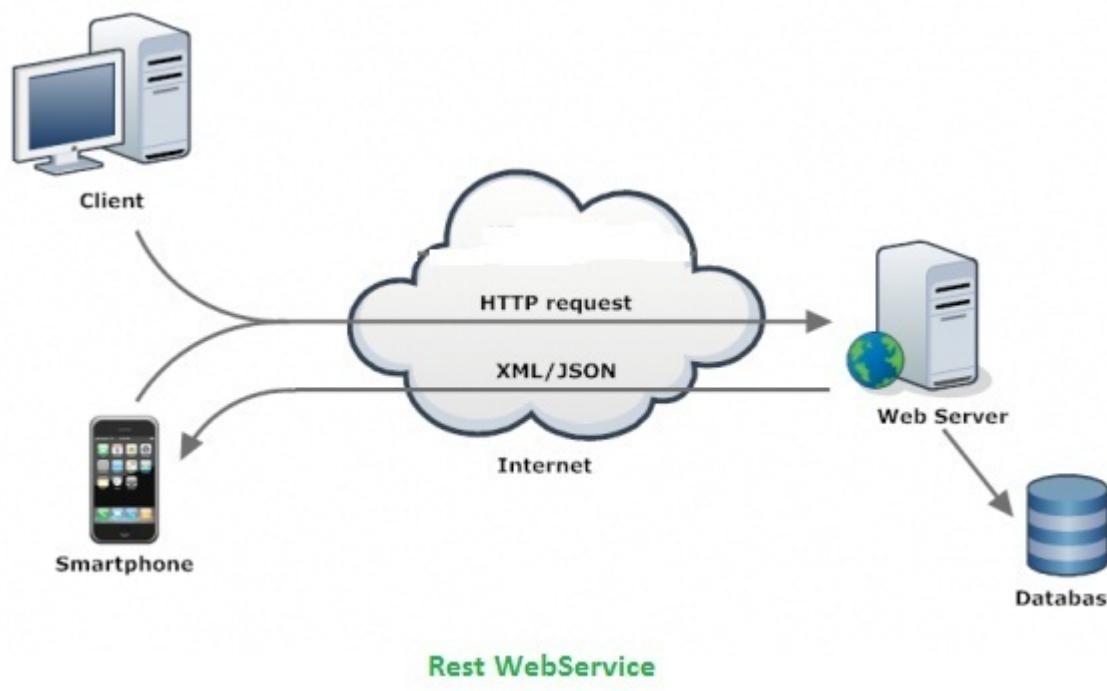
- Aquí podéis ver toda la documentación del API REST de Accuweather, con toda la información para acceder a los datos meteorológicos, las consultas que podemos hacer (mediante las distintas rutas), cómo son los datos que devuelve (qué formato tienen y cómo parsearlos), etc...
 - Notas: Accuweather, como la mayor parte de los API REST públicos (como Twitter, por ejemplo) requieren registro y nos proporcionan unas API Key para poder hacer llamadas al servicio REST.

Primera aplicación Java EE

- ▶ **Servicios Web**
 - Son **programas con funcionalidad accesible a través de protocolo HTTP** que permiten la ejecución remota de métodos y funciones
 - Los clientes envían solicitudes de ejecución (**GET, POST, PUT, DELETE**) de métodos y **funciones y sus argumentos**
 - Los servidores responden enviando los resultados solicitados
 - Desde Java EE 6 se incluyen “grandes” servicios web (basados en la especificación JAX-WS) y servicios web más ligeros (basados en la especificación JAX-RS)
 - Los servicios ligeros optimizan el ancho de banda y son más sencillos de desarrollar
 - Mediante servicios web, programas escritos con tecnologías diferentes (Java EE, .NET) pueden compartir información y procedimientos de acceso estandarizados

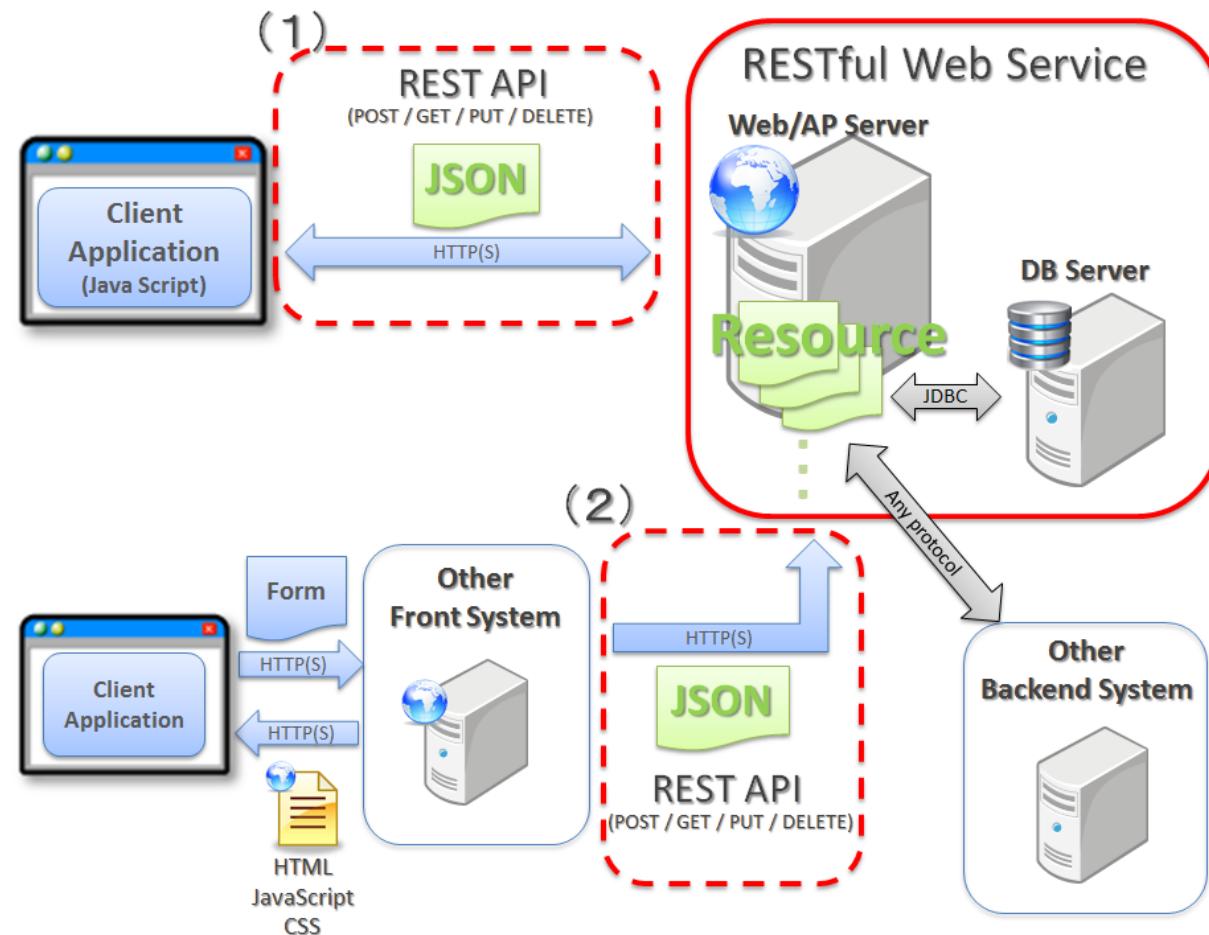
Primera aplicación Java EE

▶ Servicios Web REST



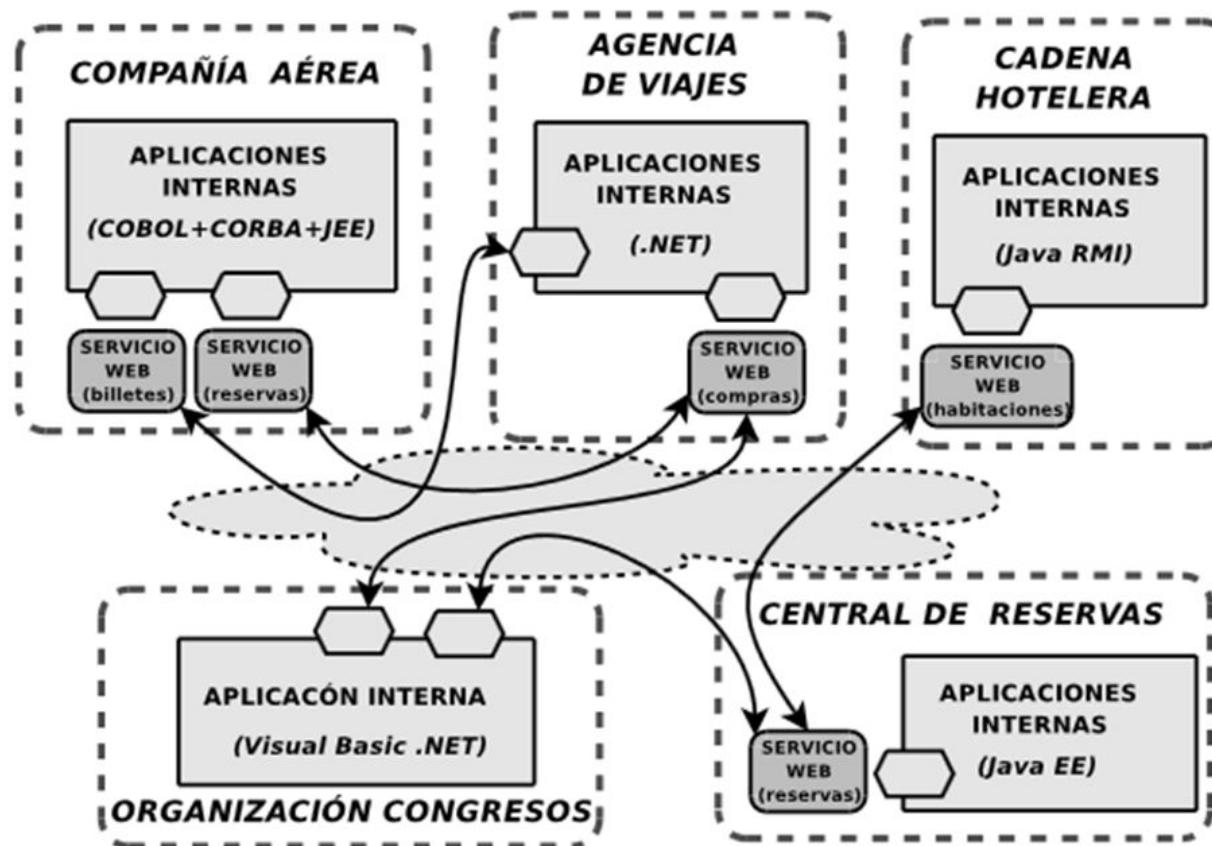
Primera aplicación Java EE

▶ Servicios Web REST



Primera aplicación Java EE

▶ Servicios Web



Primera aplicación Java EE

- ▶ Vamos a utilizar el API JAX-RS
- ▶ Al ser un servicio REST, será más sencillo y eficiente que SOAP, tenemos métodos definidos, y se accederá desde una URI única.
- ▶ RESTful utiliza el protocolo HTTP para acceder, modificar, o borrar la información contenida en un recurso.
- ▶ Tiene los siguiente métodos:
 - GET: Para obtener un valor. Puede ser un listado de objetos.
 - POST: Para guardar un nuevo objeto en la aplicación.
 - DELETE: Para eliminar un objeto.
 - PUT: Para actualizar un objeto.

Primera aplicación Java EE

▶ Pasos:

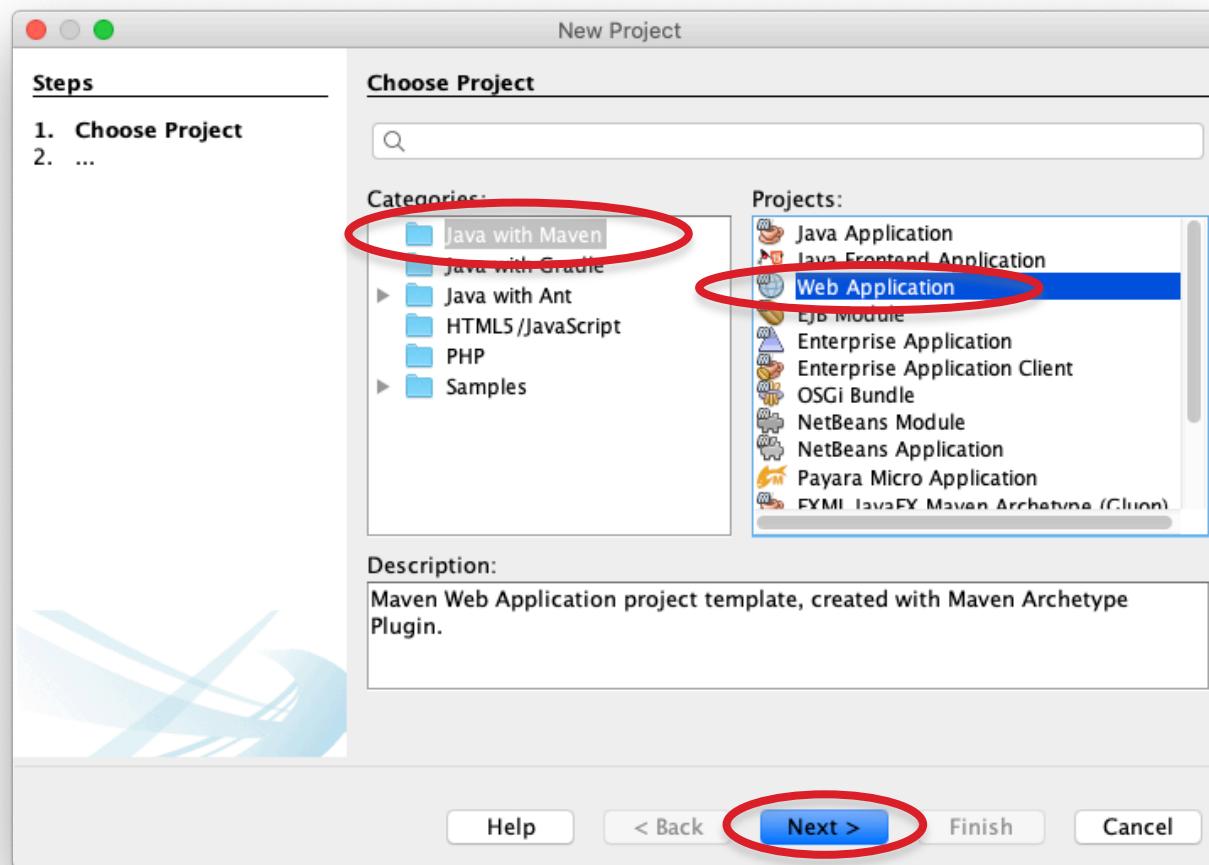
- 1.- Creamos una aplicación Web de Java EE.
- 2.- Creamos un objeto RESTful.
- 3.- Implementamos la lógica de negocio.
- 4.- Lo probamos.

Primera aplicación Java EE

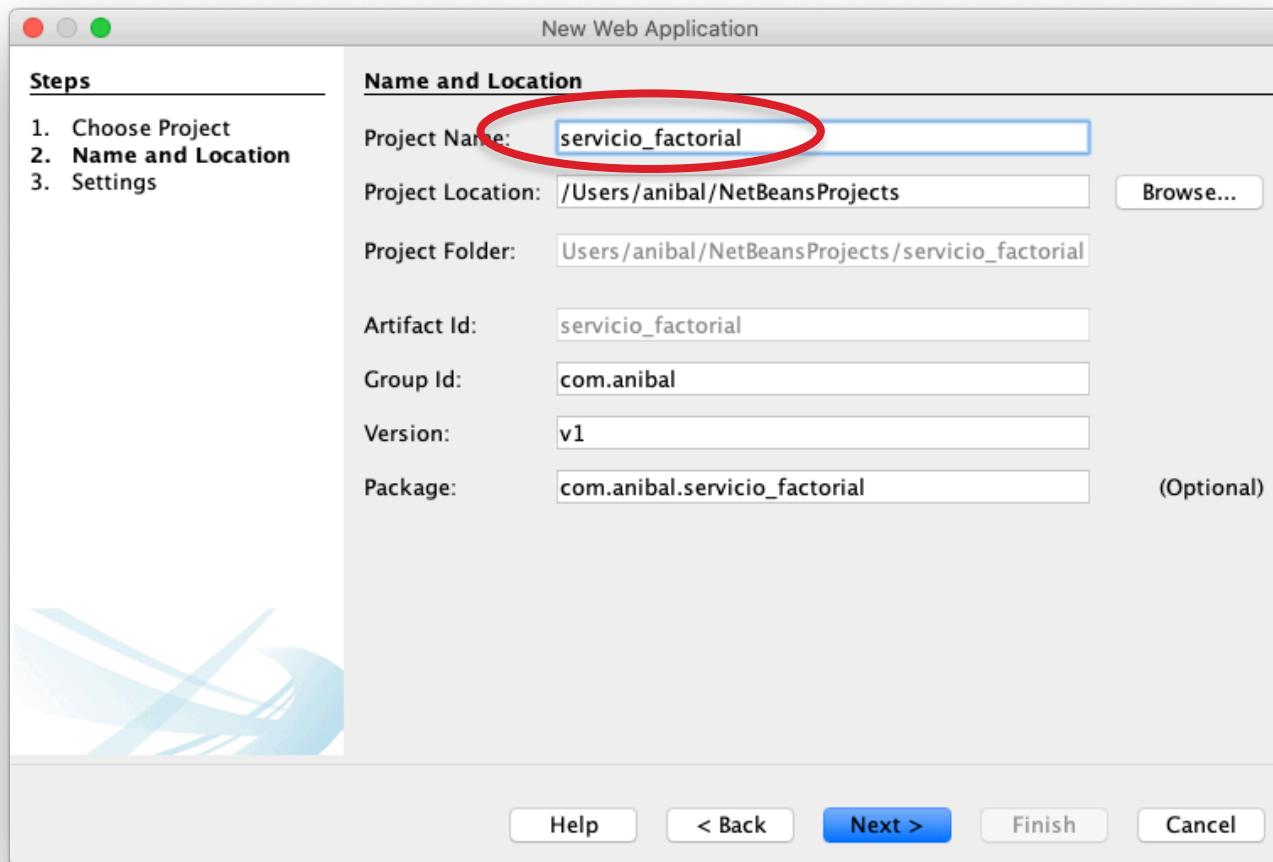
- ▶ Nos vamos a File – New Project
- ▶ Elegimos un arquetipo de Maven para aplicaciones Web (esto nos da una plantilla como base para escribir nuestra aplicación)
- ▶ Le damos un nombre (ej: servicio_factorial)
- ▶ Elegimos el servidor de Payara que acabamos de crear. El despliegue se realizará en dicho servidor de aplicaciones.

Primera aplicación Java EE

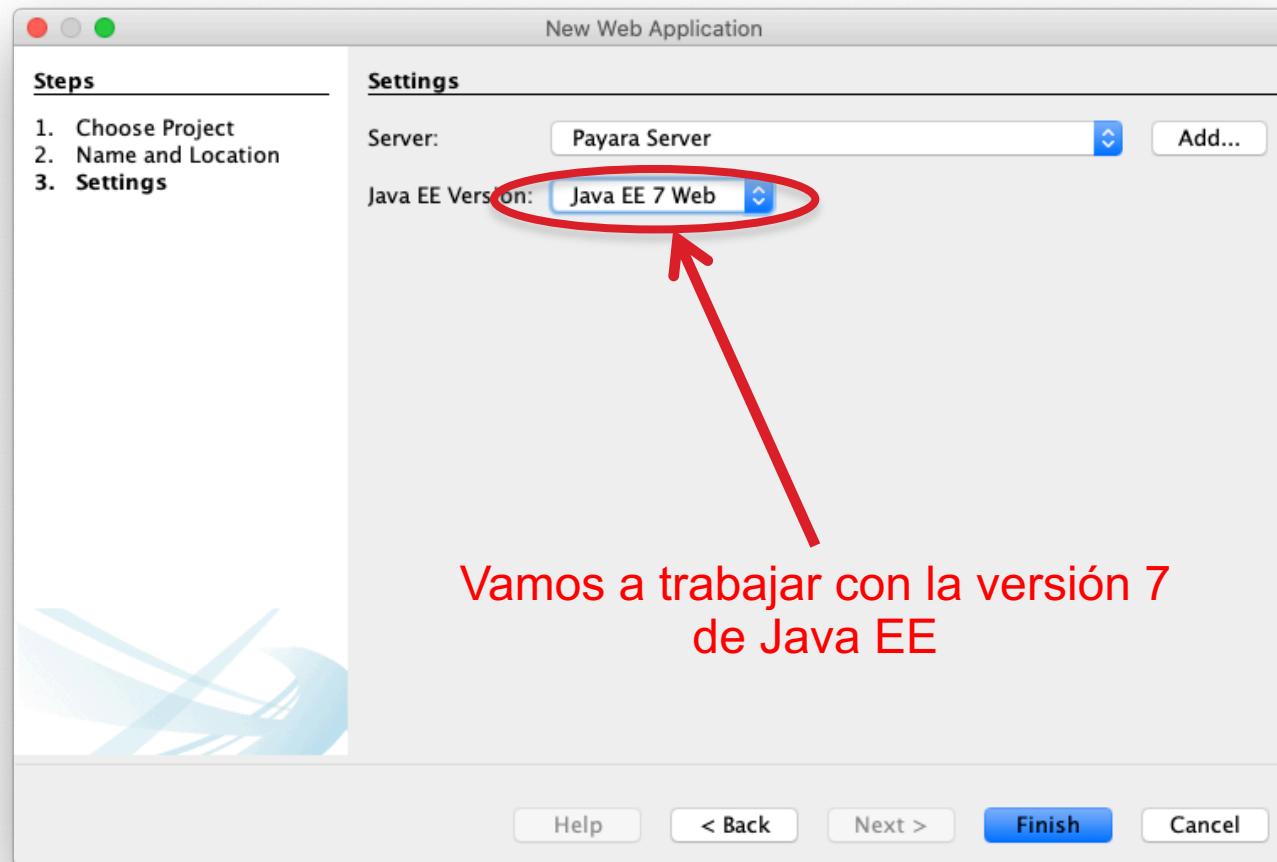
- ▶ Dependiendo de la versión de Netbeans (o de si habéis tenido algún bundle de Netbeans anteriormente instalado), la primera vez que lo utilicemos puede que nos pida activar las características de Java EE



Primera aplicación Java EE

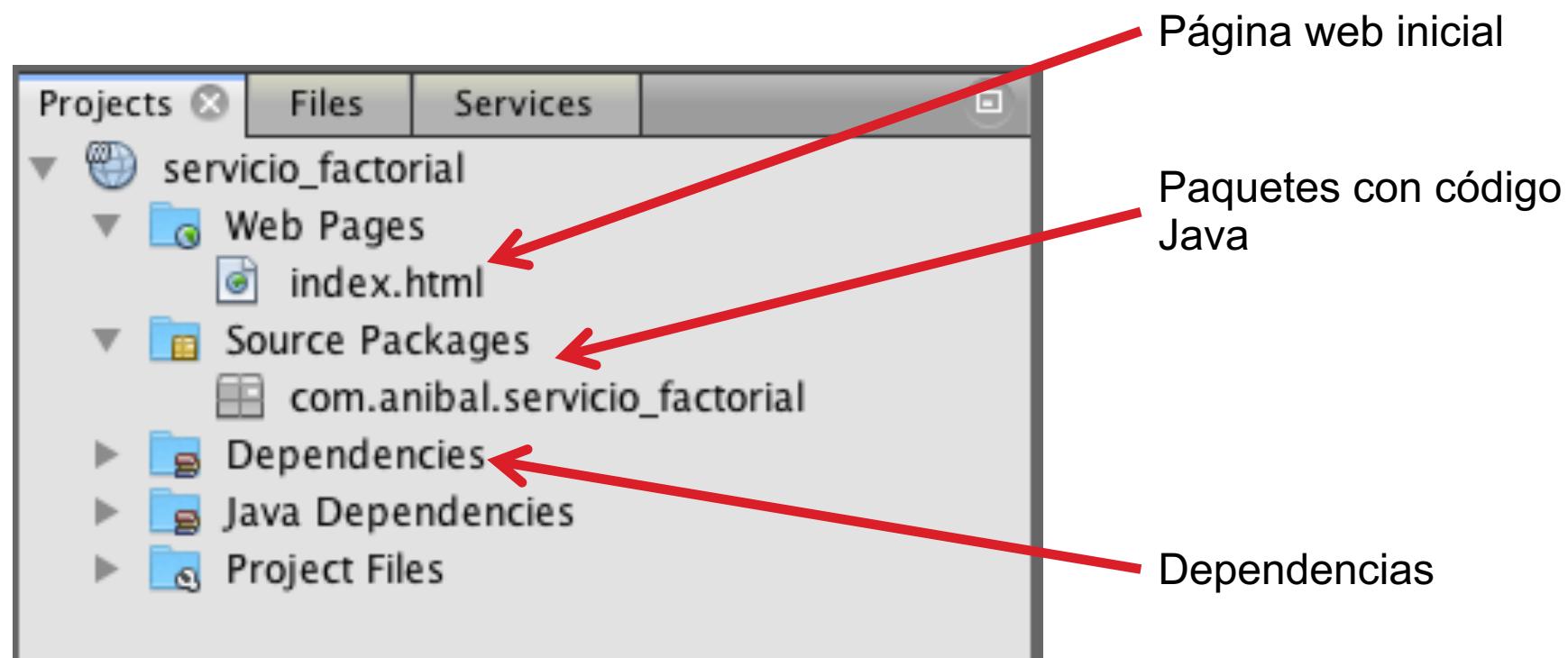


Primera aplicación Java EE



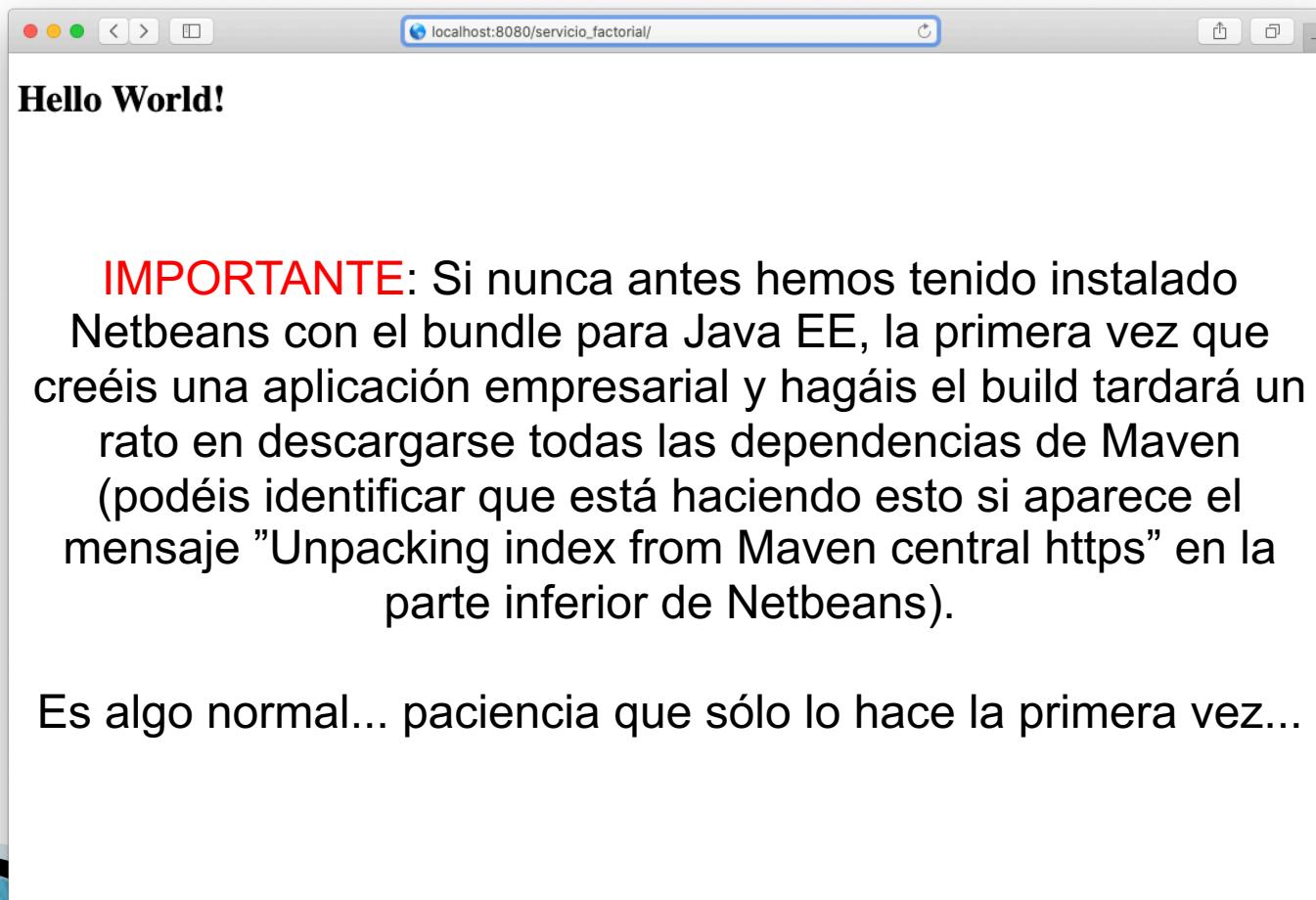
Primera aplicación Java EE

Maven ya nos crea la estructura básica de una aplicación Web Java EE con sus dependencias básicas y configuración por defecto



Primera aplicación Java EE

- ▶ Si el servidor Payara está arrancado, podemos compilarlo y desplegarlo (dándole al play ). Nos abrirá una ventana del navegador con un “Hello World!”.



Primera aplicación Java EE

- ▶ Si nos fijamos en el navegador, una vez hecho el despliegue siempre podremos acceder mediante la dirección:

http://localhost:8080/servicio_factorial/

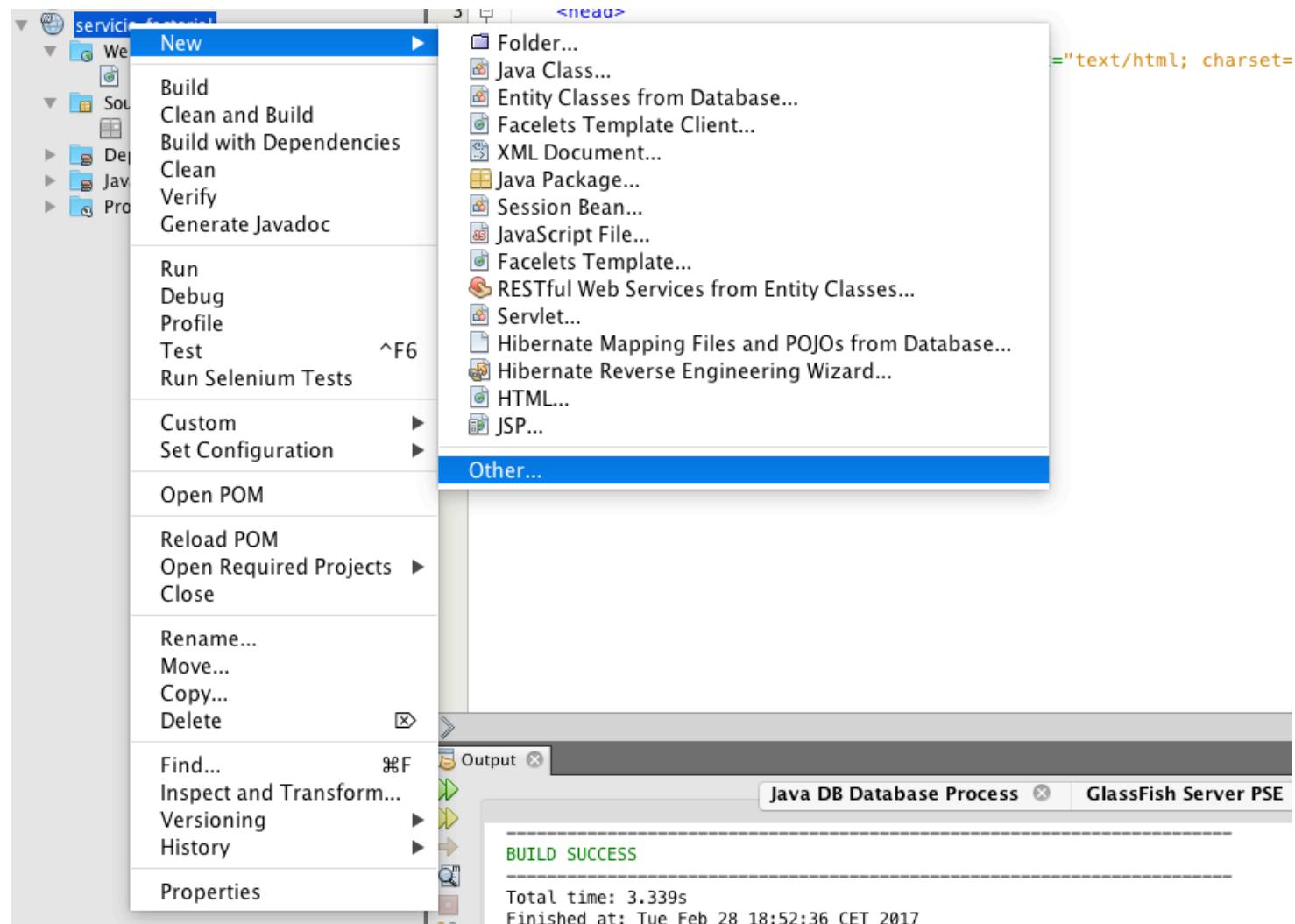
- ▶ Primero tenemos el nombre del host y el puerto
- ▶ Luego tenemos el nombre del proyecto que acabamos de crear
- ▶ El fichero que se carga por defecto es el index.html que ha creado.
- ▶ También podemos configurar Payara accediendo al puerto 4848:

<http://localhost:4848/>

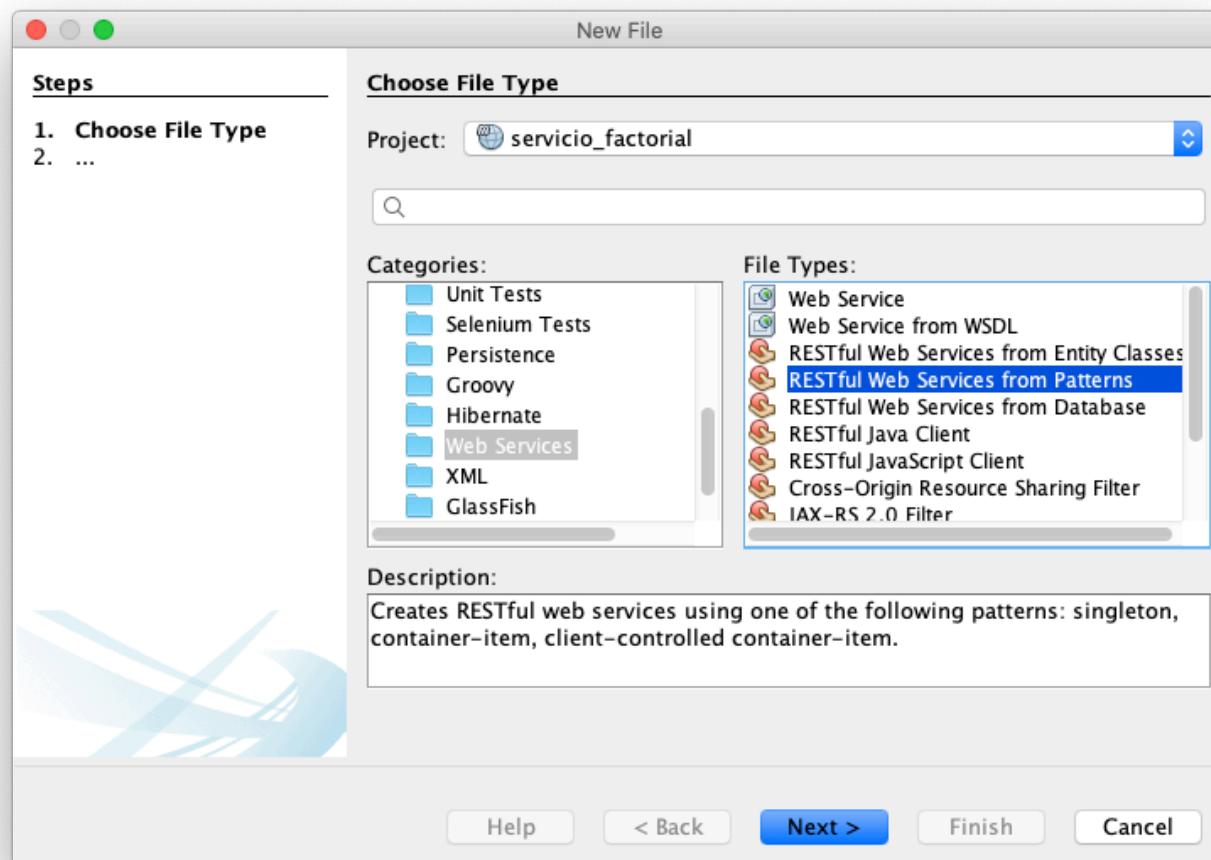
Primera aplicación Java EE

- ▶ Vamos a crear un servicio REST desde cero.
- ▶ Botón derecho sobre el proyecto que acabamos de crear – new – other...
- ▶ Buscamos Web Services y seleccionamos RESTful Web Services from Patterns (para que nos cree un patrón inicial del servicio)
- ▶ Elegimos “Simple Root Resource” (y pulsamos siguiente)
- ▶ En Path, elegimos el nombre del path – **esto es importante ya que la URI desde la que se llamará externamente al servicio REST estará identificada por este nombre**
- ▶ Elegimos el nombre de la clase
- ▶ Finalizar

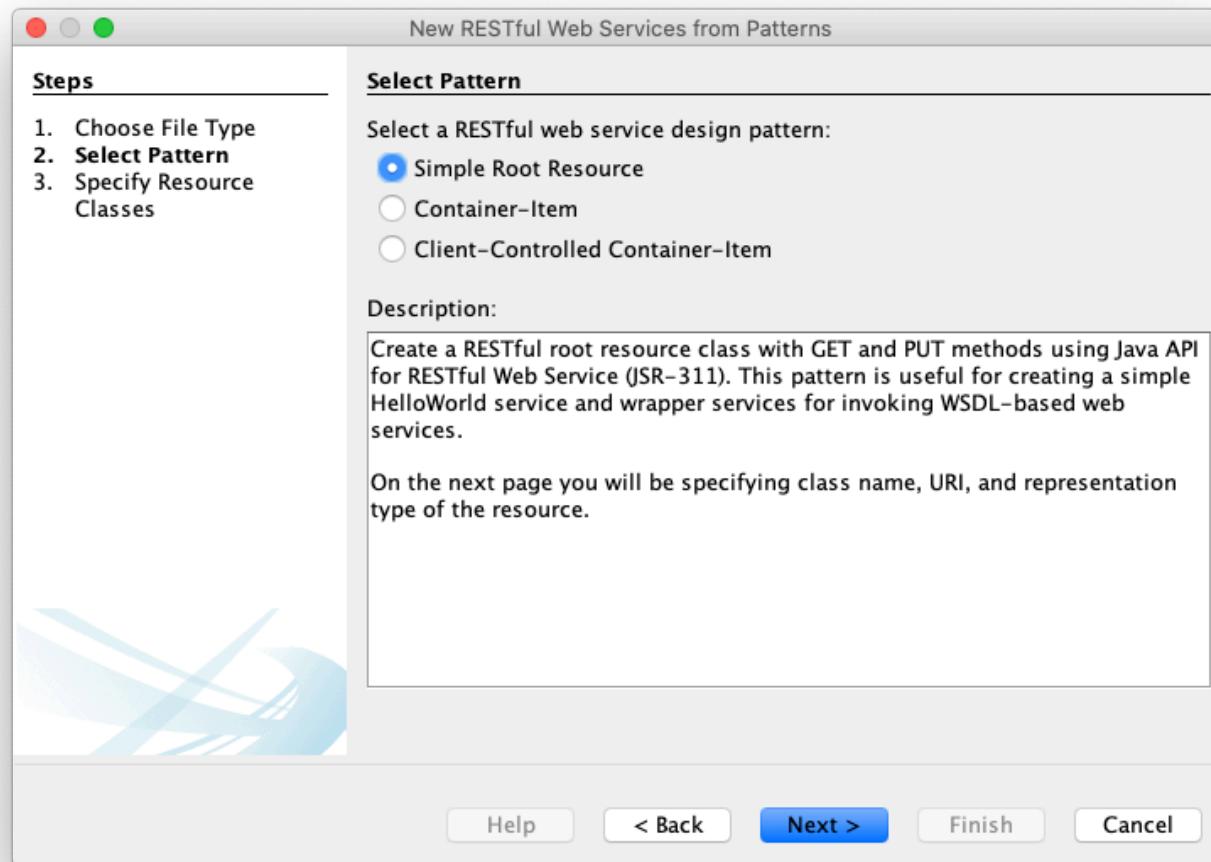
Primera aplicación Java EE



Primera aplicación Java EE

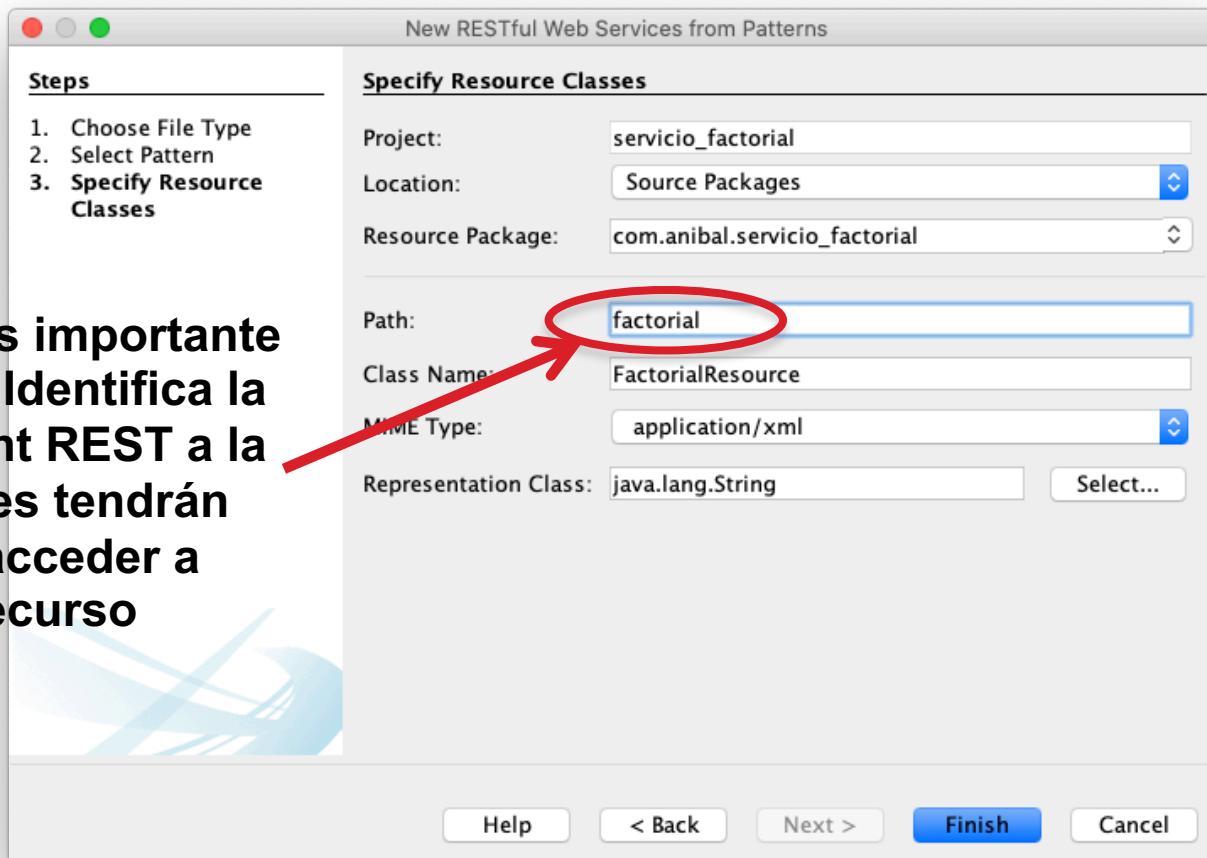


Primera aplicación Java EE

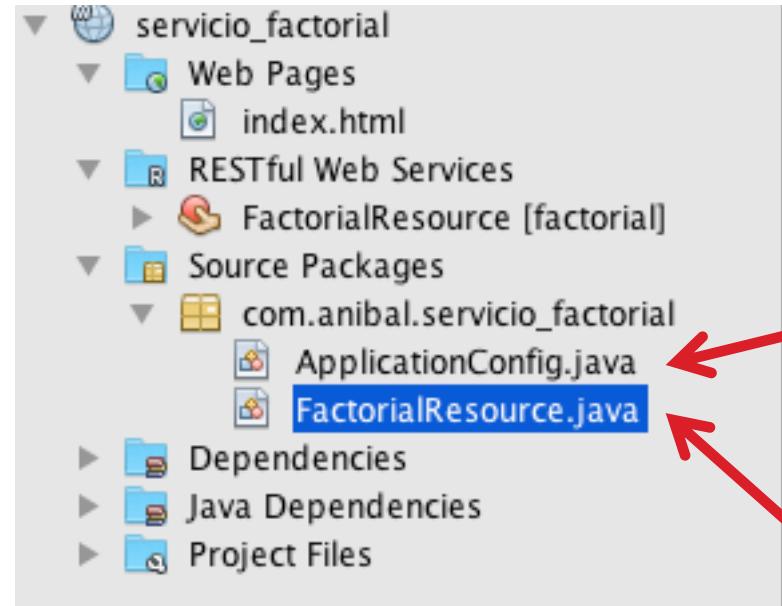


Primera aplicación Java EE

El path es lo más importante de esta página. Identifica la ruta del endpoint REST a la que los clientes tendrán llamar para acceder a nuestro recurso



Primera aplicación Java EE



**Clase Application
(lo vemos más adelante)**

Recurso REST que acabamos de crear

Vamos a ver con más detalle el código que nos ha creado...

Primera aplicación Java EE

Indicamos que vamos a declarar el método **GET de REST**
(Java EE funciona con “anotaciones” para dar información fácilmente al compilador y facilitarnos la programación y el despliegue de la aplicación)

Tipo de datos que devolvemos

```
package ...  
import javax.ws.rs.core.Context;  
...  
@Path("factorial")  
public class FactorialResource {  
  
    @Context  
    private UriInfo context;  
    public FactorialResource() {  
    }  
  
    @GET  
    @Produces(MediaType.APPLICATION_XML)  
    public String getXml() {  
        //TODO return proper representation object  
        throw new UnsupportedOperationException();  
    }  
  
    @PUT  
    @Consumes(MediaType.APPLICATION_XML)  
    public void putXml(String content) {  
    }
```

Cargamos las librería correspondientes - JAX RS

Ruta de acceso al recurso - se puede cambiar aquí posteriormente si es necesario

Método GET
(este método se ejecutará cuando el cliente haga una petición GET a la ruta factorial de nuestro endpoint REST)

A continuación tenemos el método PUT de REST

El PUT lleva la anotación Consumes porque no proporciona información, sino que la recibe para hacer algo con ella

Primera aplicación Java EE

- ▶ La clase **ApplicationConfig.java** es fundamental para que los servicios REST funcionen correctamente
- ▶ Define el conjunto de servicios REST que van a ser accesibles y **el punto desde el que cuelgan todos estos servicios REST** (en este caso “webresources”)

```
@javax.ws.rs.ApplicationPath("webresources")
```

- ▶ Debería crear esta clase de manera automática, pero no siempre lo hace...

Primera aplicación Java EE

- ▶ **Si no existiera**, tendríamos que añadirla – nos dirá que REST no está configurado:



A screenshot of the NetBeans IDE interface. On the left, there is a code editor window with the following Java code:

```
23
24  /**
25  * REST is not configured
26  * Create Subclass
27  * Create Test Class
28  * -----
29  * (Alt-Enter shows hints)
30  *     .cines.entities.movie")
31
32
```

The line at position 25, which contains the comment "REST is not configured", has a yellow tooltip box over it. The tooltip box contains the following text:

- REST is not configured
- Create Subclass
- Create Test Class
-
- (Alt-Enter shows hints)

The code editor also shows a cursor on the line starting with "public class MovieFacadeREST extends AbstractFacade<Movie> {".

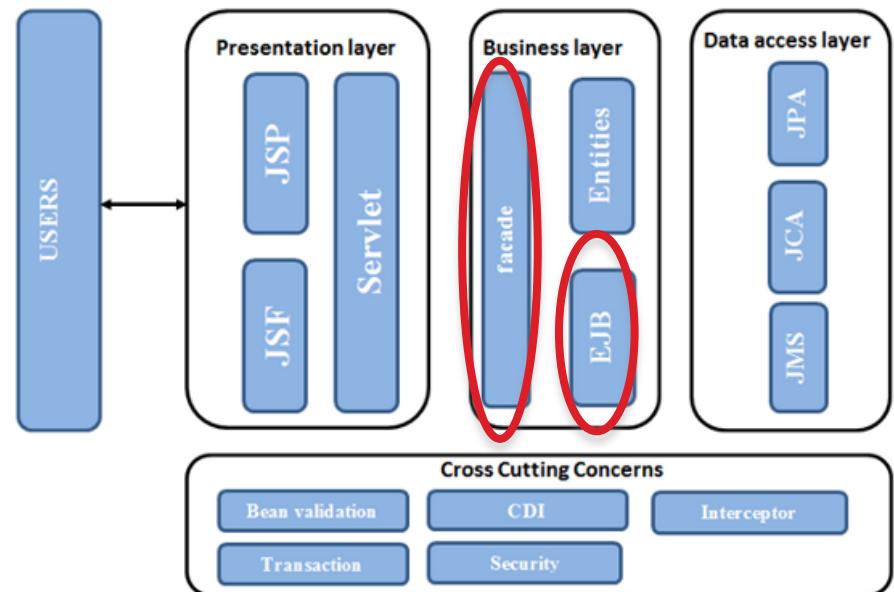
- ▶ Presionamos sobre la bombilla y seleccionamos “Configure REST using Java EE 6 specification”
- ▶ Nos habrá creado el paquete “org.netbeans.rest.application.config” con el ApplicationConfig. Podemos ver que ha creado el path de la clase Application para el servicio REST en “webresources”, que es la ruta de la que cuelgan todos nuestros servicios REST.

Primera aplicación Java EE

- ▶ Lo desplegamos y en teoría podríamos acceder mediante su URI:
http://localhost:8080/servicio_factorial/webresources/factorial
- ▶ Aunque de momento nos dará error!!! (lanza la excepción que tiene dentro del código – `UnsupportedOperationException`)
- ▶ Con esto ya tenemos toda la parte de la interfaz del servicio implementada, ahora vamos al siguiente componente, la lógica de negocio

Primera aplicación Java EE

- ▶ Vamos a implementar la lógica de negocio, que en este ejemplo sencillo será simplemente el cálculo del factorial de un número.



Primera aplicación Java EE

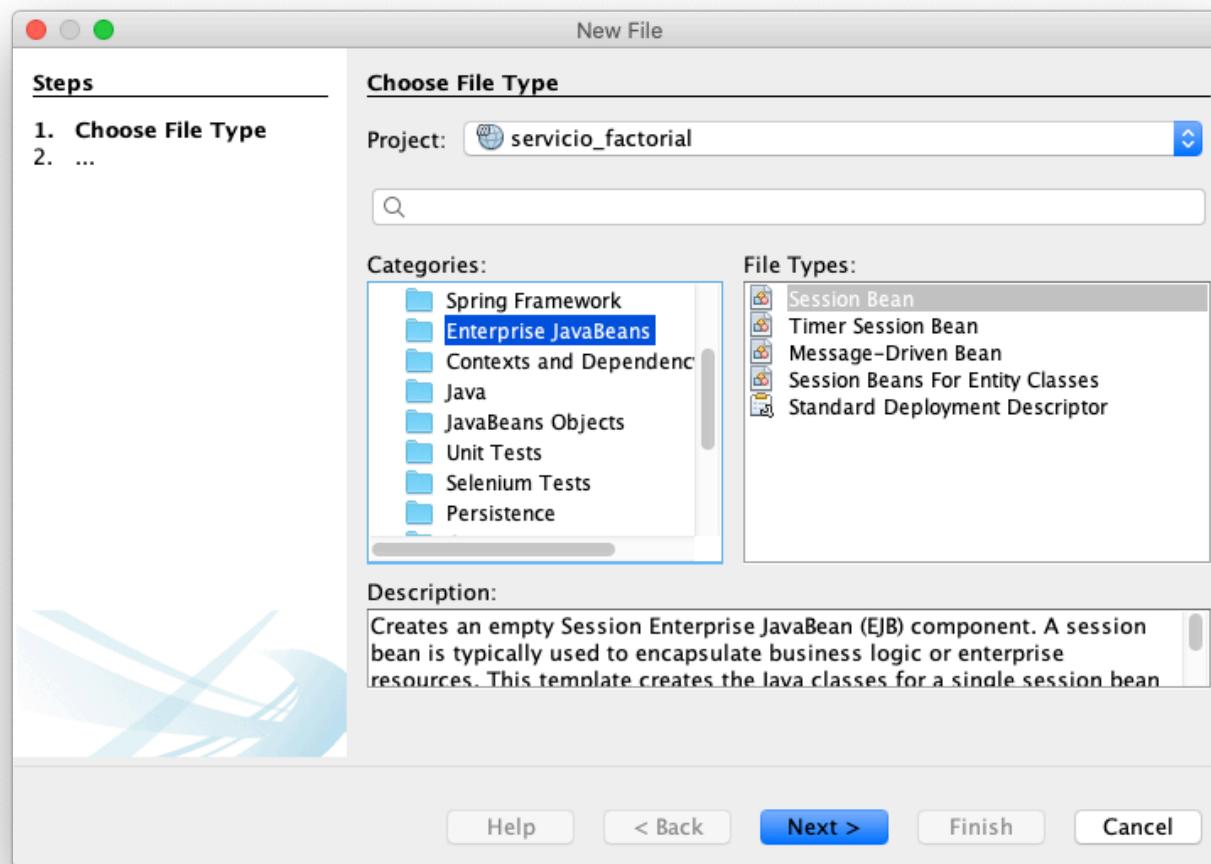
- La lógica de negocio que tenemos que implementar es el método para calcular el factorial de un número:

```
public long factorial(long base) {  
    if (base < 2) {  
        return 1;  
    } else {  
        return base * factorial(base - 1);  
    }  
}
```

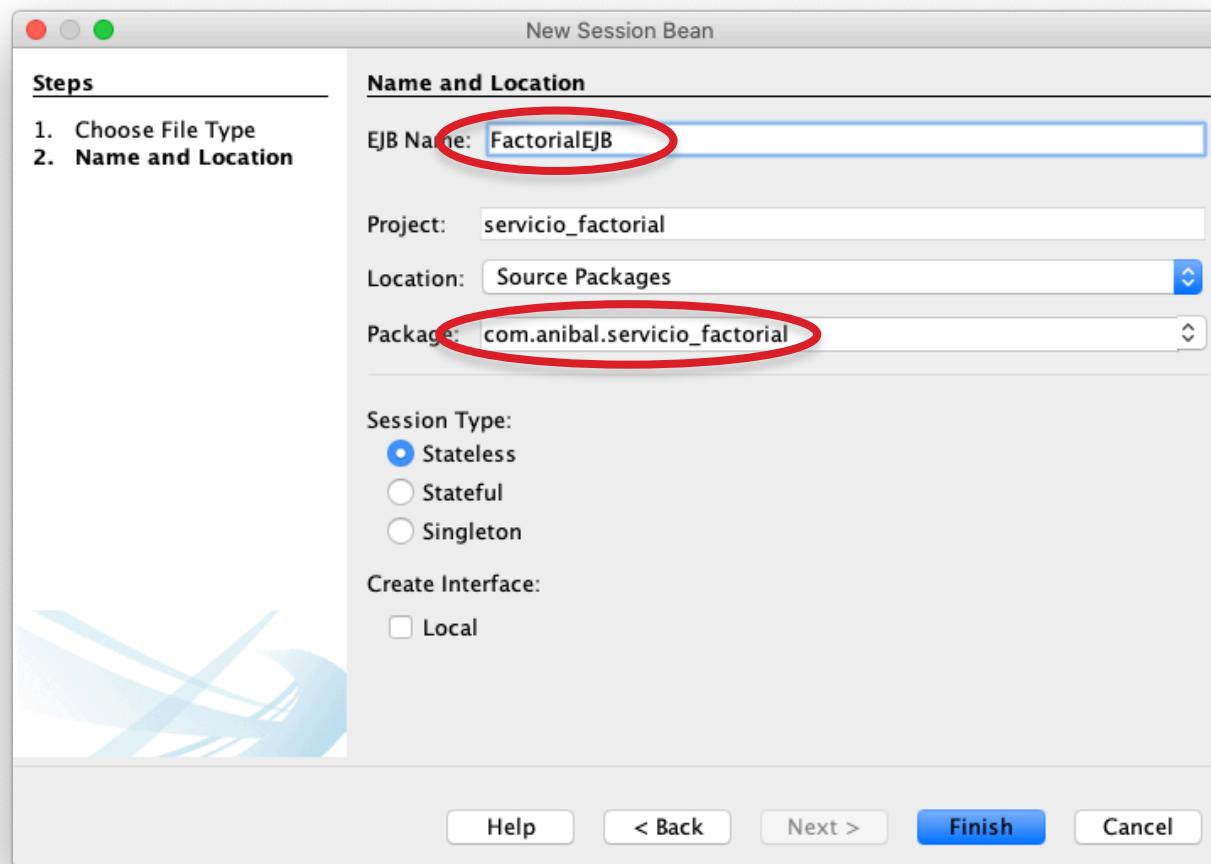
Primera aplicación Java EE

- ▶ Creamos un EJB
- ▶ Botón derecho sobre el paquete *servicio_factorial* – new – other...
- ▶ Buscamos Enterprise JavaBeans y seleccionamos Session Bean
- ▶ Le damos el nombre “FactorialEJB”
- ▶ Le declaramos Stateless (con esto le decimos que va a ser un EJB que no va a mantener el estado)

Primera aplicación Java EE



Primera aplicación Java EE



Primera aplicación Java EE

- ▶ Ahora implementamos la “lógica de negocio”
- ▶ Debería quedaros algo parecido a esto:

```
package com.anibal.servicio_factorial;  
  
import javax.ejb.Stateless;  
  
@Stateless  
public class FactorialEJB {  
  
    public long factorial(long base) {  
        if (base < 2) {  
            return 1;  
        } else {  
            return base * factorial(base - 1);  
        }  
    }  
}
```

Primera aplicación Java EE

- ▶ Modificamos nuestro servicio REST de la siguiente manera para integrarlo con el EJB:
 1. **Injectamos el EJB:** @EJB FactorialeEJB factorial_EJB;
 - Esto nos permitirá usar el EJB que hemos creado desde la clase del servicio REST
 2. Como no vamos a devolver un xml, tenemos que cambiar la etiqueta del @Produces. **Vamos a devolver un objeto, en nuestro caso pondremos un String**, por lo tanto será @Produces("text/plain")
 3. **Cambiamos el nombre del método getXml() por getFactorial()**
 - Esto es un simple cambio estético, podríamos mantener el getXml(), pero siempre es mejor acostumbrarse a usar nombres significativos para facilitar la reutilización.

Primera aplicación Java EE

- ▶ Modificamos nuestro servicio REST de la siguiente manera para integrarlo con el EJB:
 4. **Hacemos que pueda recibir parámetros de entrada** mediante la URI (mediante un parámetro de la query @QueryParam)
 - Este será el número para el que calculemos el factorial.
 5. **Hacemos que la salida sea String** en lugar de long.
 - Porque hemos indicado que el método va a devolver un texto plano.
 6. **Hacemos la llamada al método que calcula el factorial** en el EJB.
 7. Al añadir los distintos elementos tendremos que **incluir las bibliotecas de Java** correspondientes (Netbeans nos sugiere las más adecuadas – Ojo, que no siempre son las correctas!!)

▶ Nuestro servicio quedará de la siguiente manera:

Primera aplicación Java EE

```
@Path("factorial")
public class FactorialResource {

    @EJB FactorialEJB factorial_EJB;

    ...

    @GET
    @Produces("text/plain")
    public String getFactorial(@QueryParam("base") long base) {
        return Long.toString(factorial_EJB.factorial(base));
    }

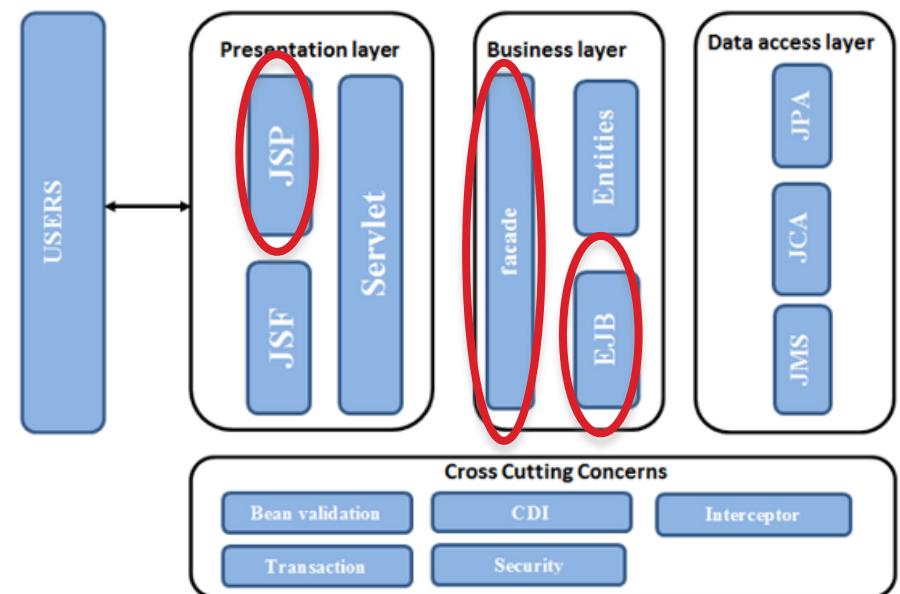
    ...
}
```

Primera aplicación Java EE

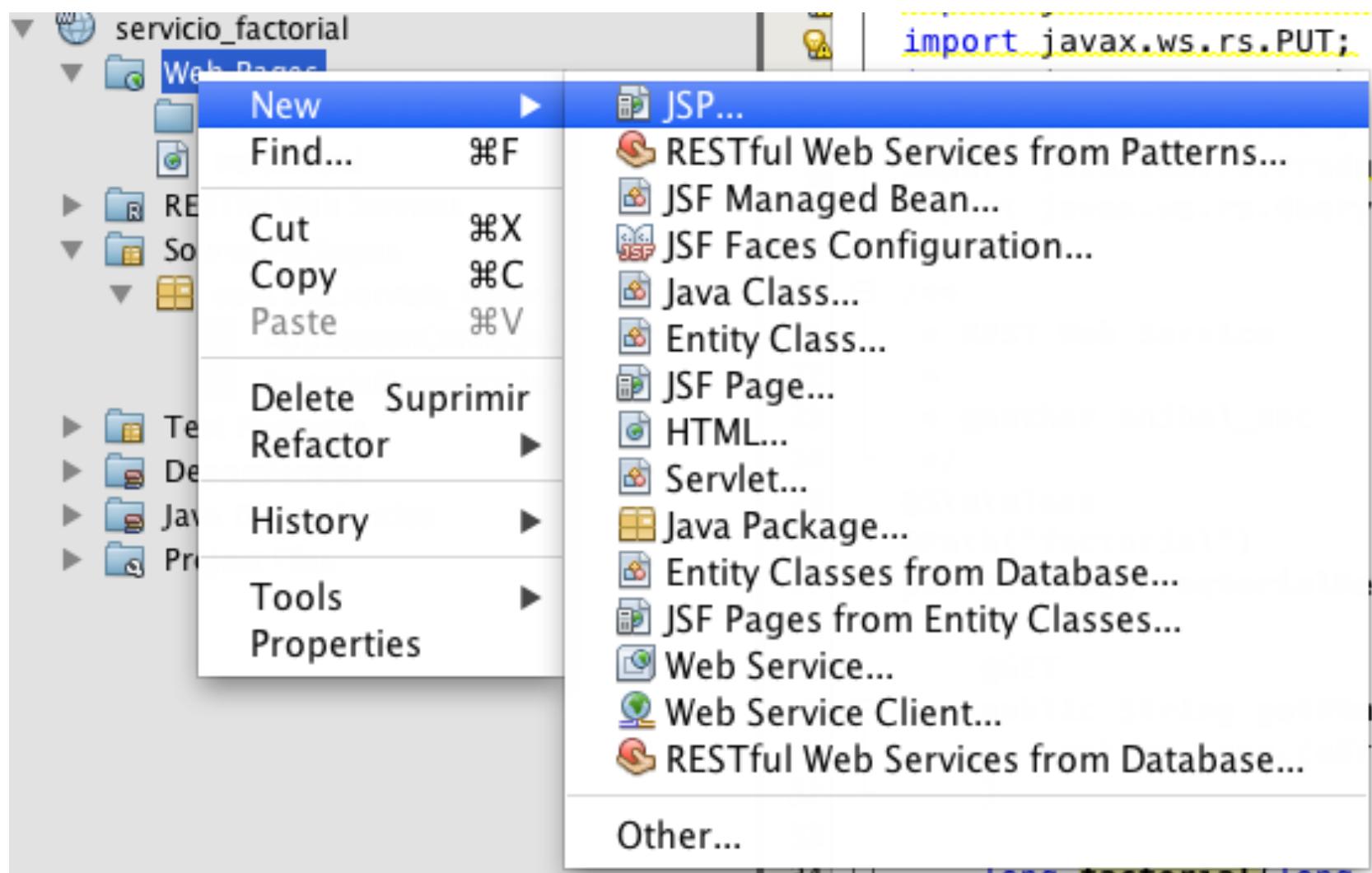
- ▶ Ahora ya podemos desplegar la aplicación empresarial y probarla.
 - **Cuidado**, a veces es necesario hacer un “Clean+Build” y volver a desplegarlo para que introduzca correctamente todos los artefactos en la inyección de dependencias.
- ▶ Con el método GET implementado con la lógica de negocio, podemos acceder a la URI del servicio REST de la siguiente manera:
http://localhost:8080/servicio_factorial/webresources/factorial?base=3
- ▶ No os olvidéis del ?base=3, que es la manera de pasarle parámetros al servicio REST mediante la query (sino siempre dará 1 porque al no pasarle valor del parámetro asume que es 0)
- ▶ Si desplegamos esta mini aplicación empresarial en un servidor de aplicaciones accesible desde Internet, se podría llamar a este servicio web desde cualquier sitio del mundo con cualquier aplicación (empresarial en .NET, móvil en Android, etc...)

Primera aplicación Java EE

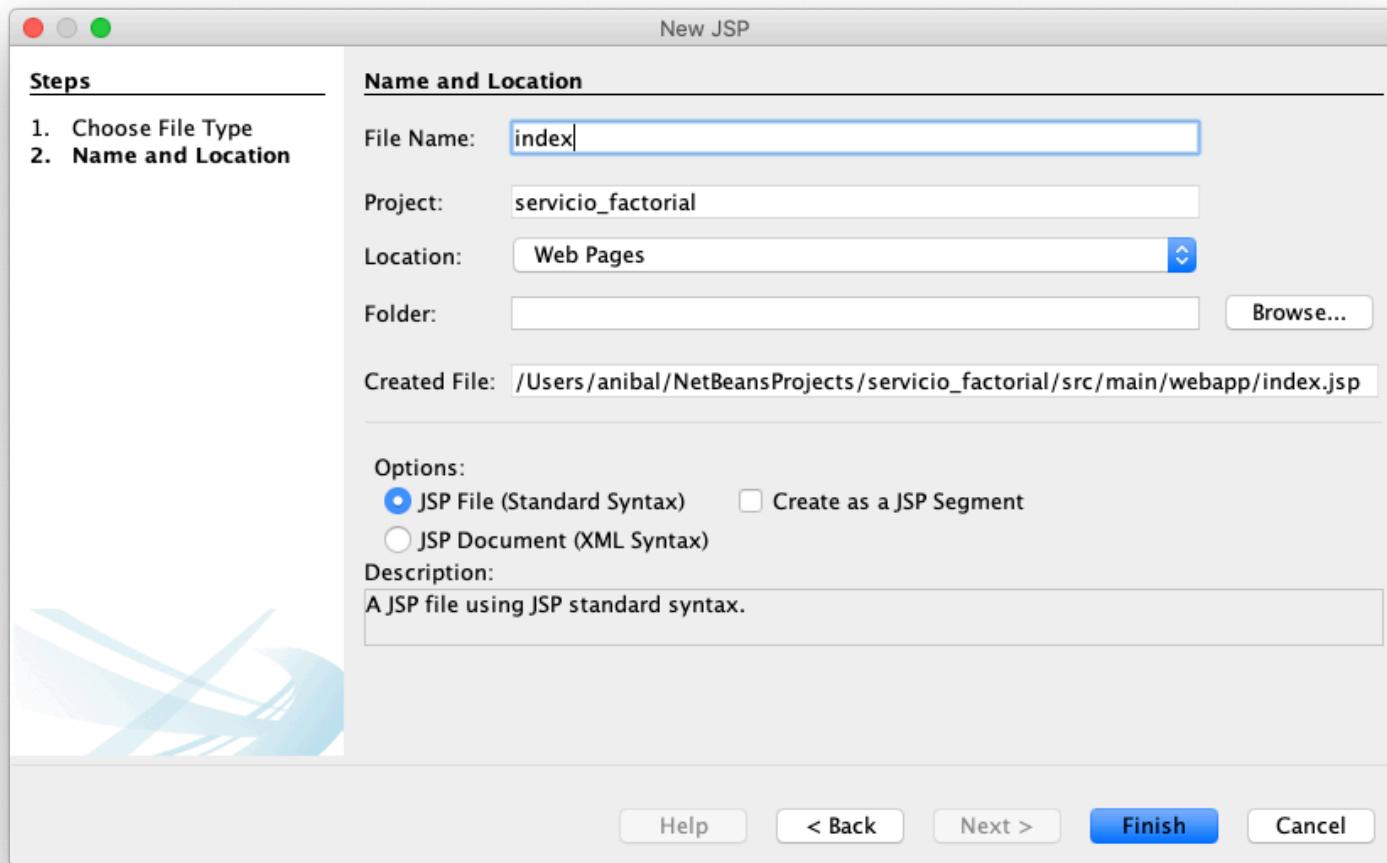
- ▶ Mediante la interfaz de servicios, se puede llamar al servicio, pero tenemos que hacerlo mediante la URI
- ▶ Ahora vamos a crear una interfaz de usuario que llame al servicio y se lo muestre al usuario.
- ▶ Usaremos JSP. Pasos:
 - Creamos una página JSP
 - Botón derecho → new → JSP
 - O Botón derecho → other → Web → JSP
 - Implementamos la IU para la aplicación factorial



Primera aplicación Java EE



Primera aplicación Java EE



Primera aplicación Java EE

```
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        La fecha de hoy es: <%=new Date().toString()%>
        <h1>Calculando factorial</h1>
        <form action="webresources/factorial">
            Base: <input name="base" type="text" />
            <button>Calcular</button>
        </form>
    </body>
</html>
```

Primera aplicación Java EE

- ▶ <%@page import="java.util.Date"%>
 - Usamos <% %> para introducir código Java dentro del JSP
- ▶ La fecha de hoy es: <%=new Date().toString()%>
 - Ejemplo con el JSP: nos muestra la fecha de hoy
- ▶ <form action="webresources/factorial">
 - Llamamos al recurso
- ▶ Base: <input name="base" type="text" />
 - Definimos la entrada del recurso
- ▶ <button>Calcular</button>
 - El botón

Primera aplicación Java EE

- ▶ Ahora borrad el fichero index.html, que ya no nos vale, y acceded de nuevo (recargad la página si es necesario)

Primera aplicación Java EE

- ▶ Desplegamos de nuevo...

La fecha de hoy es: Mon Feb 22 17:05:06 CET 2021

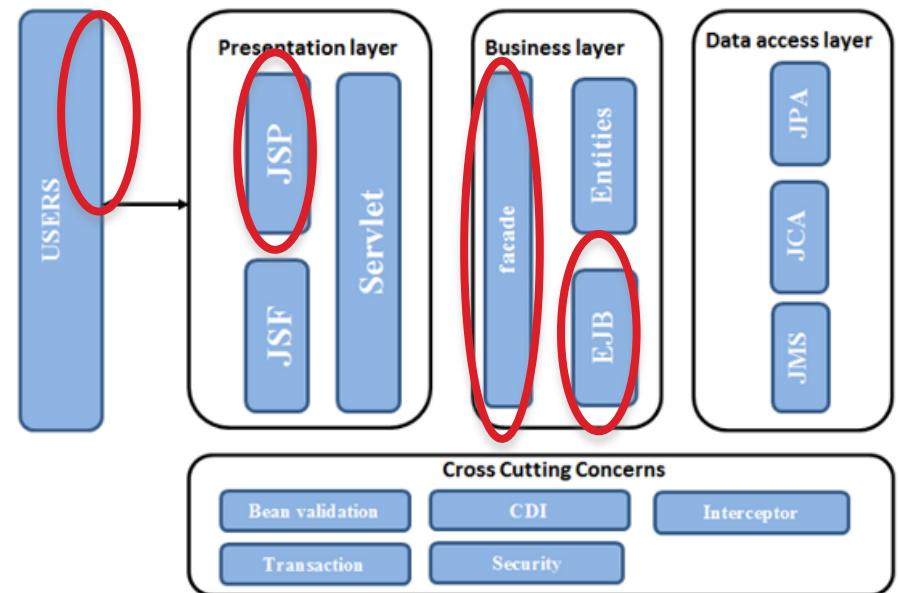
Calculando factorial

Base: Calcular

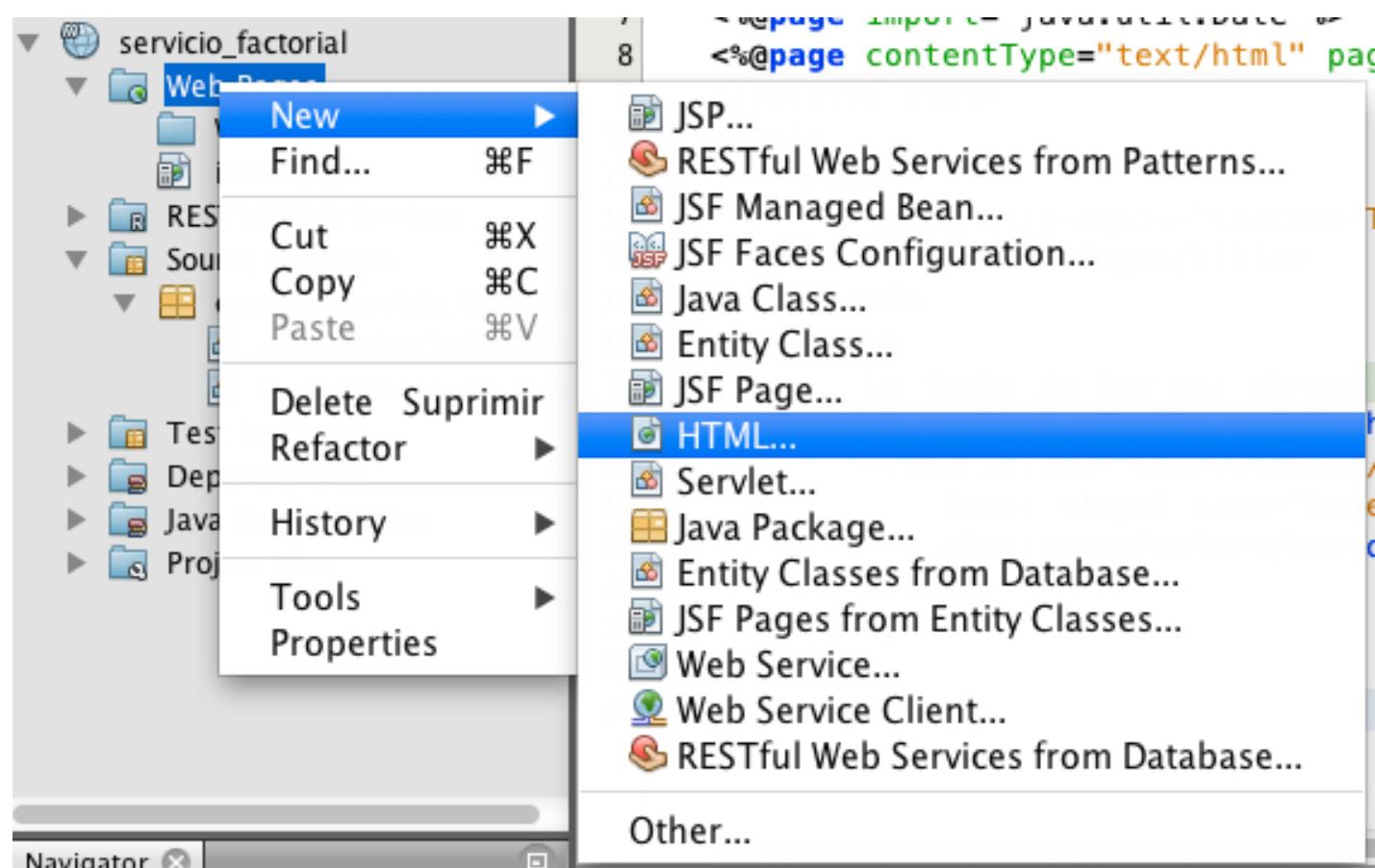
- ▶ Si introducimos un número y presionamos “Calcular”, se llama al GET del servicio REST, que calcula el factorial y nos llevará a una página con el número calculado
- ▶ Desde el punto de vista de PSE, esto tiene un problema importante → perdemos usabilidad!!!

Primera aplicación Java EE

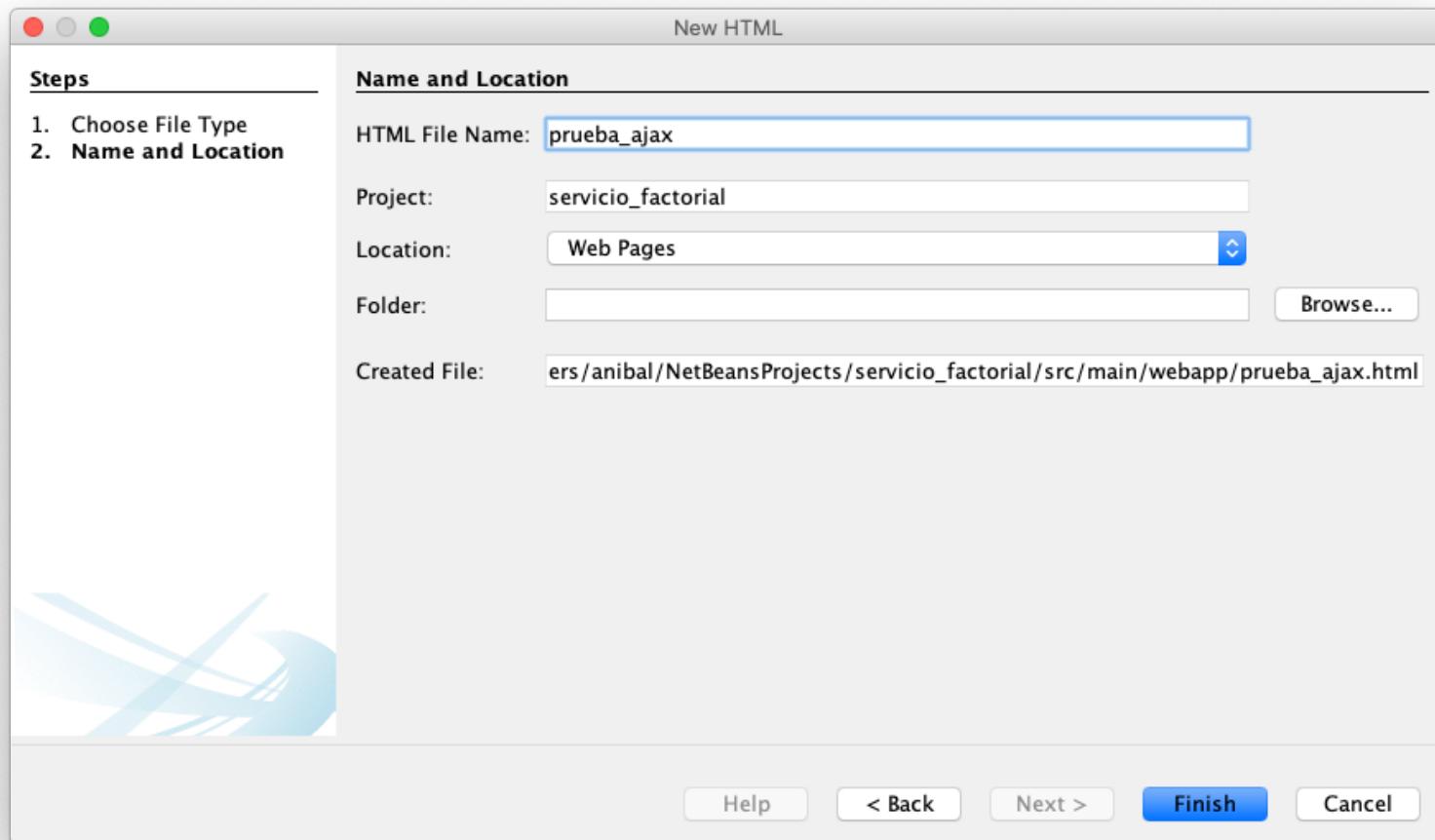
- ▶ Ahora queremos mejorar la disponibilidad y facilitar la navegabilidad al usuario.
 - Queremos que el resultado del calculo del factorial se cargue en la misma página del usuario, y que no tenga que esperar a mostrarlo en otra página.
- ▶ Para ello utilizaremos JavaScript con AJAX. Los datos y la llamada al recurso se enviarán al servidor a través de AJAX y no será necesario recargar la página.
- ▶ Para facilitar la tarea utilizaremos el Framework jQuery.



Primera aplicación Java EE



Primera aplicación Java EE



Primera aplicación Java EE

```
<!DOCTYPE html>
<html>
    <head>
        <title>Prueba Ajax</title>
        <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js">
        </script>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Calcular factorial</h2>
        Número:<input type="text" name="base" id="base"/>
        <button type="button" id="calcularBtn">Calcular</button>
        <div id="resultado">
            Resultado: <span></span>
        </div>
        <script type="text/javascript">
            jQuery("#calcularBtn").click(function(){
                var base = jQuery("#base").val();
                jQuery.get("http://localhost:8080/servicio_factorial/webresources/factorial", {
                    base:base
                },function(resultado){
                    jQuery("#resultado span").text(resultado);
                });
            });
        </script>
    </body>
</html>
```

Primera aplicación Java EE

- ▶ *localhost:8080/servicio_factorial/prueba_ajax.html*

Calcular factorial

Número:

Resultado: 120

¿PREGUNTAS?