Java Server Faces

•••

Elementos Básicos

Contenido

- Introducción
- Nuevas Características en JSF v2.*
- Ventajas y Desventajas
- MVC en Java Server Faces (JSF)
- Vistas en Java Server Faces (JSF)
- Lenguaje de Expresiones EL
- Etiquetas
- Managed Beans
- Uso de los Managed Beans
- Alcance de los Managed Beans
- Plantillas en Java Server Faces (JSF)
- Uso de Plantillas en Java Server Faces (JSF)
- Reglas de Navegación:
  - Estática
  - Dinámica

Introducción

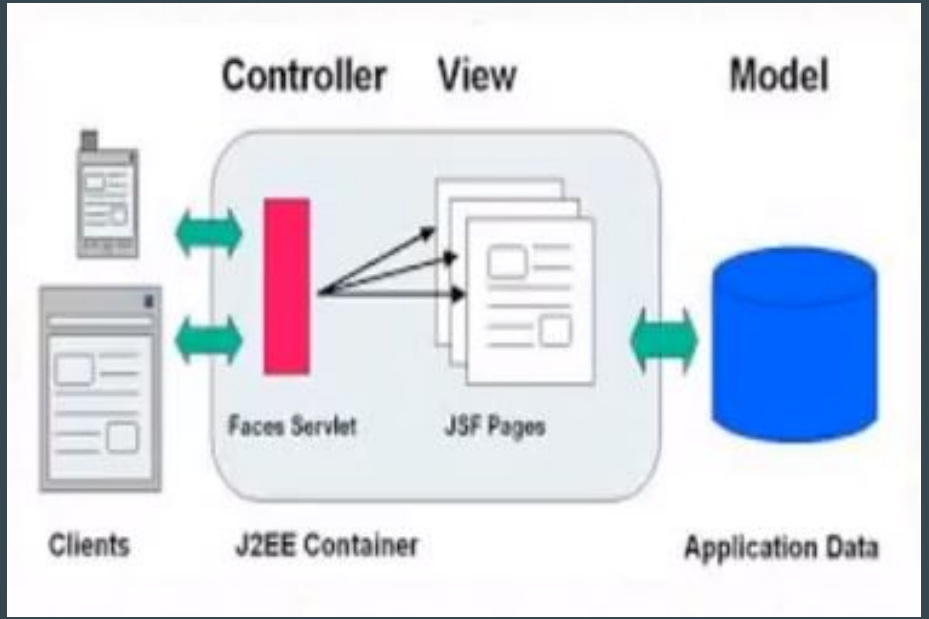
Características:

- Framework para crear aplicaciones web estándar para JEE
- Cuenta con soporte completo en IDEs (Netbeans, Eclipse, entre otros)
- Aplica el patrón de diseño Modelo Vista Controlador (MVC)
- Implementa el desarrollo de aplicaciones web ágiles (RAD)
- Utiliza el motor de render Render-Kids, los componentes pueden usarse en dispositivos móviles y otros dispositivos
- JSF es altamente extensible debido a su arquitectura

Nuevas Características en JSF v2.*

Nuevas características:

- Condiciones más inteligentes
- Anotaciones de configuraciones
- Soporte nativo para AJAX
- Soporte para Facelets
- Más componentes y validadores



Ventajas y Desventajas

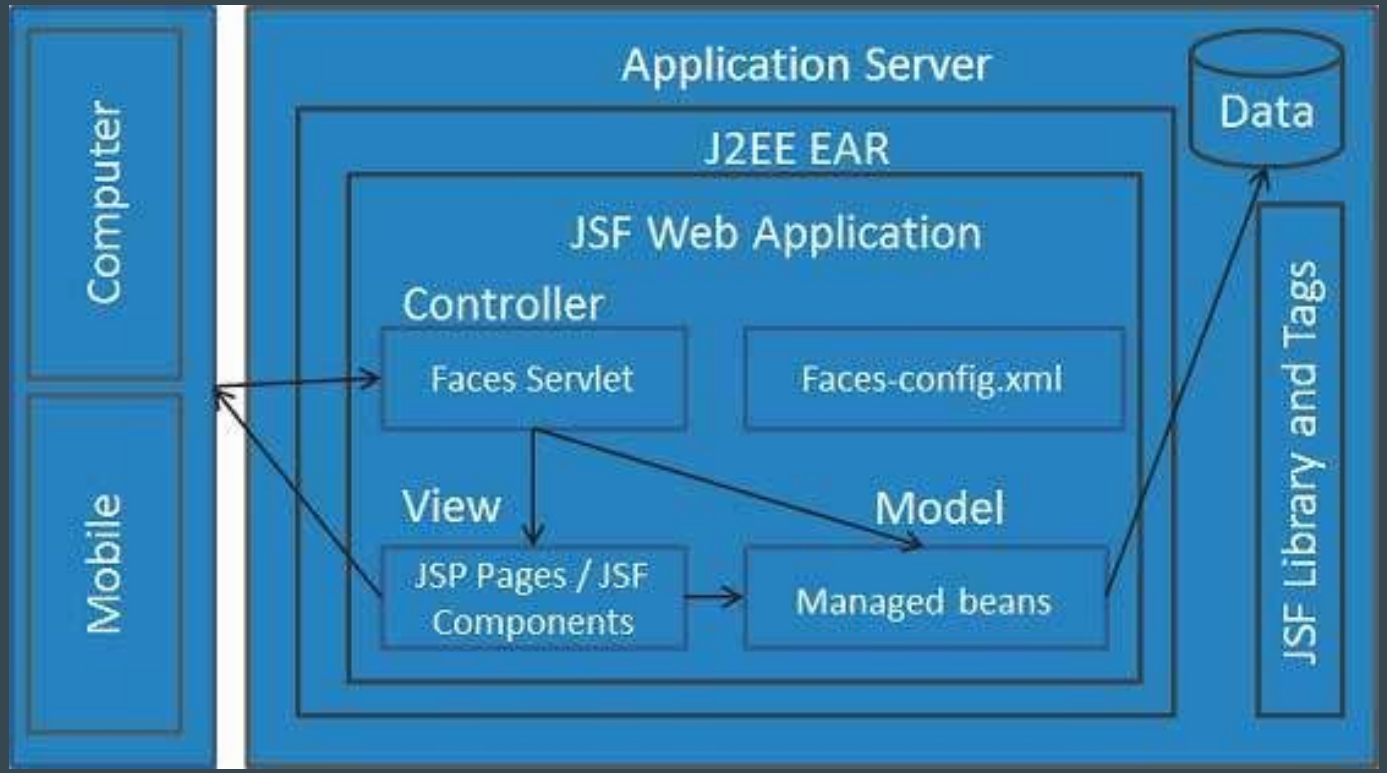
Ventajas:

- El código es muy parecido al HTML estándar
- Se encarga de la obtención y generación de los valores de los elementos de la página
- Resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n)
- Permite introducir de JavaScript
- Es extensible

Desventajas:

- Utilizar el alicate para clavar un clavo.
- Abuso del JavaScript
- La maquetación compleja (DataTables)

MVC en Java Server Faces (JSF)



Vistas en Java Server Faces (JSF)

Características:

- Es asociado a cada vista un conjunto de objetos provenientes de Java, manejados por el controlador (Managed Beans)
- Facilita la obtención, manipulación y visualización de los datos
- Se utiliza un archivo de configuración para el controlador en formato XML
- Es extensible

Lenguaje de Expresiones EL

Características:

- Permite acceder a las propiedades de los Managed Beans
- Se utiliza la siguiente estructura:
  - #{ Expresión a utilizar }
  - ${ Expresión a utilizar }
- Por ejemplo:

<h:outputText value="**#{ EXPRESIÓN }**" />

<h:inputText value="**${ EXPRESIÓN }**" />

Existen dos maneras para evaluar una expresión:

- Inmediata:
  - Es necesario el valor al inicial la página
  - **${ EXPRESIÓN }**
- Diferida:
  - Leer y escribir datos
  - Utilizar expresiones de método
  - **#{ EXPRESIÓN }**

Lenguaje de Expresiones EL

Expresión EL	Tipo de Propiedad	Valor
bean.property	String	El valor del String.
bean.myBoolean	boolean	Valores true o false o su cadena “true” o “false”.
bean.property.property2	property: Class property2: String	El valor de la propiedad 2 del objeto property del bean.

Etiquetas

Etiqueta	Descripción
h:commandButton	Un botón al que podemos asociar una acción.
h:commandLink	Un enlace hipertexto al que podemos asociar una acción.
h:dataTable	Crea una tabla de datos dinámica con los elementos de una propiedad de tipo Array o Map del bean.
h:form	Define el formulario JSF en la página JSP.
h:inputText	Incluye un campo de texto normal.

https://docs.oracle.com/javaee/7/javaserver-faces-2-2/vdldocs-facelets/toc.htm
https://tutorialspointexamples.com/jsf-tags-pdf-basic-html-simple-list-core-download-code/

Managed Beans

General:

- Es una clase Java que sigue la nomenclatura de JavaBeans
- Los Managed Beans no están obligados a extender de ninguna clase
- Nos ayudan a controlar el flujo de información y representar datos en las vistas

Tipos de Beans:

- Modelo: Representan el modelo
- Control: Representan el controlador
- Helpers: Contienen código, por ejemplo convertidores
- Utilerias: Tareas genéricas

Uso de los Managed Beans

Formas:

- Anotaciones, antes del nombre de la clase:
  - @ManagedBean
- Archivo faces-config.xml
  - <managed-bean>
  - ...
  - </managed-bean>
- CDI (Content and Dependency Injection), antes del nombre de la clase:
  - @Named(value = “NOMBRE”)

Alcance de los Managed Beans

Scope	Descripción
Application	Persiste la información durante toda la vida de la aplicación web.
Session	Guarda la información desde que el usuario comienza una sesión hasta que ésta termina (porque el tiempo expiró o se invocó al método invalidate sobre un objeto HttpSession).
View	El scope dura desde que se muestra una página JSF al usuario hasta que el usuario navega hacia otra página. Es muy útil para páginas que usan AJAX.
Request	Comienza cuando se envía una petición al servidor y termina cuando se devuelve la respuesta al usuario.

Ejemplo

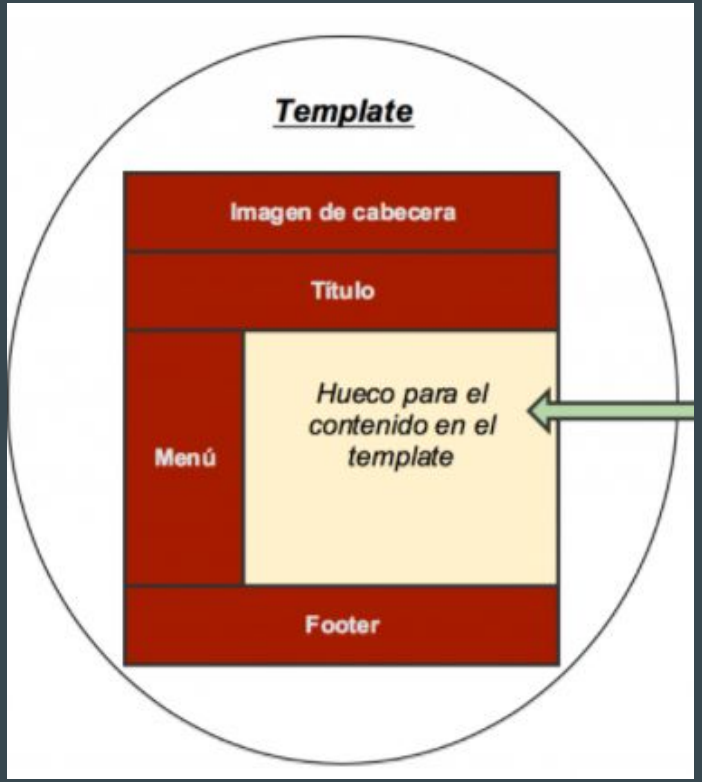
Plantillas en Java Server Faces (JSF)

Plantilla (Template):

- Archivo **xhtml**
- Define zonas de contenido dinámico
- Header, footer, menús, colocación de elementos, entre otros

Cliente (Client):

- Archivo **xhtml**
- Utiliza una plantilla definida
- Especifica el contenido que va a mostrar



Uso de Plantillas en Java Server Faces (JSF)

Plantilla (Template):

```
<h:body>
    <div id="top">
        <ui:insert name="top">
            <h1>Bienvenido a mi sitio web</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">
            <h1>Todos los derechos reservados</h1>
        </ui:insert>
    </div>
</h:body>
```

Client:

- Uso de la plantilla:

```
<ui:composition template=”URL”>
  <ui:define name="NOMBRE"></ui:define>
</ui:composition>
```

- Archivos externos:

```
<ui:include src=”URL”>
  <ui:param name="VARIABLE"
value="VALOR">
  </ui:param>
</ui:include>
```

Ejemplo

Reglas de Navegación (Navigation Rules): Estática

Implícita:

Directamente desde las vistas.

```
<h:form>
    <ul>
        <li>
            <h:commandLink value="NAME_TO_SHOW" action="PAGE_URL"/>
        </li>
    </ul>
</h:form>
```

Reglas de Navegación (Navigation Rules): Dinámica

Formas:

- Archivo **faces-config.xml**
- Managed Bean

```
<h:form>
    <ul>
        <li><h:commandLink value="Pagina1" action="page1" /></li>
        <li><h:commandLink value="Pagina2" action="#{ bean.metodo() }" /></li>
    </ul>
</h:form>
```

```
<navigation-rule>
    <from-view-id>FILE_ROOT</from-view-id>
    <navigation-case>
        <from-outcome>OUTPUT</from-outcome>
        <to-view-id>PAGE_URL</to-view-id>
    </navigation-case>
</navigation-rule>
```

Ejemplo