**PRUEBA DE JSF2.0 CON PRIMEFACES**

Guarde este archivo con su nombre y apellido

1. Coloque el ámbito que corresponde a cada definición

Sesión: @SessionScoped El objeto dura en memoria mientras dura la sesión web, cada usuario que se conecta tendrá su propia instancia de este objeto.

Request: @RequestScoped El objeto dura en memoria mientras dura el request, es decir por cada nuevo request que se envíe al servidor, se vuelve a crear un nuevo objeto

View: @ViewScoped El objeto dura en memoria mientras dura la vista, es decir mientras no cambie de página, no se va a crear un nuevo objeto.

Aplicación: @ApplicationScoped El objeto dura en memoria mientras dura la aplicación, el objeto se crea una sola vez y es común para todos los usuarios.

1. Coloque el número de la fase del ciclo de vida que corresponde a cada definición

6 Los valores pasan del objeto a la página, a través de la invocación a los métodos GET

4 Se invoca a los métodos SET, para pasar los valores de la página al objeto

3 Se ejecutan las conversiones

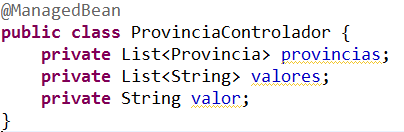
3 Se ejecutan las validaciones

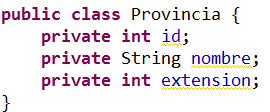
5 Se ejecutan los métodos invocados desde la página

1. Cuando se coloca la anotación @ManagedBean, el import correcto es del paquete:
2. **javax.faces.bean**
3. javax.annotation
4. javax.beans
5. Complete el siguiente cuadro que define los atributos process y update de Ajax

|  |  |  |
| --- | --- | --- |
| Atributo | Fases del ciclo de vida afectadas | Comportamiento por defecto, si no se incluye esta propiedad |
| Process | 3,4,5 | Procesa Todo |
| Update | 6 | No actualiza nada |

1. Se tiene las siguientes clases con sus respectivos getters y setters





Y dados los siguientes fragmentos de código:

*prov*

*#{provinciaControlador.provincias}*

*#{prov.id}*

*#{prov.nombre}*

*#{prov.extension}*

*#{provinciaControlador.valor}*

*#{provinciaControlador.valores}*

Complete el código para pintar una tabla, un componente de selección única y un componente de selección múltiple en JSF

<h:dataTable value=*"#{provinciaControlador.provincias}"*

var=*" prov "*>

<h:column>

<h:outputText value=*"#{prov.id}"* />

</h:column>

<h:column>

<h:outputText value=*"#{prov.nombre}"* />

</h:column>

<h:column>

<h:outputText value=*"#{prov.extension}"* />

</h:column>

</h:dataTable>

<h:selectOneMenu value=*"#{provinciaControlador.valor}"*>

<f:selectItems value=*"#{provinciaControlador.valores}"*

var=*"prov"*

itemLabel=*"#{prov.nombre}"*

itemValue=*"#{prov.id}"*

/>

</h:selectOneMenu>

<h:selectManyCheckbox value=*"#{provinciaControlador.valor}"*>

<f:selectItems value=*"#{provinciaControlador.valores}"*

var=*"prov"*

itemLabel=*"#{prov.nombre}"*

itemValue=*"#{prov.id}"*

/>

</h:selectManyCheckbox>

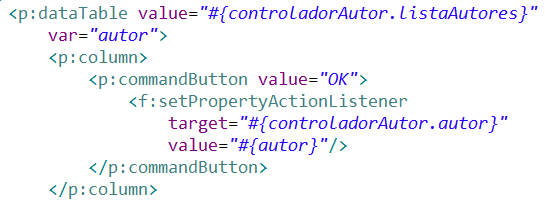
1. Si en el código se encuentra el siguiente error:  
   #{autorControlador.autor.nombre} autor devolvió nulo.

Cuál es la causa del error y cómo lo corrige?

No está instanciado el objeto autor. Se debe instanciar:

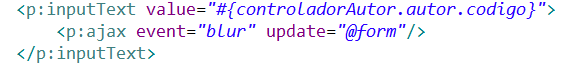
autor = new Autor();

1. Describa qué efecto tiene la etiqueta f:setPropertyActionListener, cuando se presiona el botón OK en el siguiente fragmento de código:

**

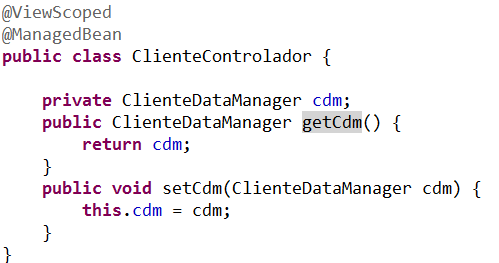
A la variable autor del controlador, se le asigna el valor la variable autor del DataTable

1. Describa qué efecto tiene la etiqueta p:ajax en el siguiente fragmento de código:



Se dispara el evento “Blur” (cuando se pierde el foco de la caja de texto) y se acutaliza todo el form.

1. Frente a cada afirmación indique si es Verdadera (V) o falsa(F)
2. Los métodos de una interface solo tienen firma, no implementación **(V)**
3. Considerando un submit sin ajax, en la fase 4 se hace SET solamente de los componentes de entrada de la página **(F)**
4. Un método abstracto tiene firma e implementación **(F)**
5. No es posible instanciar un objeto de una interface **(V)**
6. Un objeto puede ser apuntado con una variable del mismo tipo, de una clase padre o de una interface que implemente **(V)**
7. Si falla una conversión, de todas formas se ejecutan las 6 fases del ciclo de vida **(F)**
8. Considerando un submit sin Ajax, en la fase 4 se hace SET de todos los componentes de la página **(V)**
9. Considerando un submit sin Ajax, en la fase 6 se hace SET de todos los componentes de la página **(F)**
10. Si se lanza un submit con Ajax y no se especifica un atributo update, se actualizan todos los componentes, es decir se hace GET de todos **(F)**
11. Si se lanza un submit con Ajax y no se especifica un atributo process, se procesan todos los componentes, es decir todos entran a validación y a las faces 4 y 5. **(V)**
12. El ámbito que me representa menos consumo de memoria del servidor es el ámbito de sesión. **(F)**
13. Cuando se quiere hacer update de un diálogo, no se debe hacer update del componente diálogo directamente, sino del panel dentro del diálogo **(V)**
14. Para abrir un diálogo con la sentencia show(), se utiliza el id del diálogo, en lugar del widgetVar **(F)**
15. Se tiene el managedBean de sesión llamado: ClienteDataManager, y el siguiente código:



Complete el código para que ClienteControlador pueda utilizar ClienteDataManager

@ManagedProperty("#{usuarioDataManager}")

private UsuarioDataManager udm;