EBook Gratis

APRENDIZAJE

jsf

Free unaffiliated eBook created from
Stack Overflow contributors.

#jsf

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jsf

It is an unofficial and free jsf ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jsf.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con jsf

Observaciones

JavaServer Faces (JSF) es un marco de presentador de vista de modelo que se utiliza normalmente para crear aplicaciones web basadas en formularios HTML. Usando los componentes estándar y el kit de procesamiento, las vistas HTML con estado pueden definirse usando las etiquetas Facelets o JSP y conectarse a los datos del modelo y la lógica de la aplicación mediante beans de respaldo.

Versiones

Versión	Notas de lanzamiento	Fecha de lanzamiento
2.3	anuncio de Arjan Tijms	2017-03-28
2.2		2013-05-21
2.1		2010-11-22
2.0		2009-07-01
1.2		2006-05-11
1.1		2004-05-27
1.0		2004-03-11

Examples

Instalando JSF

El contenido se ha movido de nuevo a la buena página de wiki JSF

Hola Mundo

El contenido se ha movido de nuevo a la buena página de wiki JSF

Requerimientos mínimos

El contenido se ha movido de nuevo a la buena página de wiki JSF

Lea Empezando con jsf en línea: https://riptutorial.com/es/jsf/topic/916/empezando-con-jsf

Capítulo 2: Anotaciones JSF

Observaciones

Obtengo muchas informaciones de estos sitios web:

- http://www.jmdoudoux.fr/java/dej/chap-annotations.html
- http://docs.oracle.com/javaee/6/tutorial/doc/girch.html

Examples

Introducción a las anotaciones.

¿Por qué anotaciones?

En general, utilizamos anotaciones para facilitar el desarrollo y hacer que el código sea más claro y limpio.

¿Qué son las anotaciones?

Las anotaciones de Java 5 proporcionan la estandarización de los metadatos en un objetivo general. Estos metadatos asociados con las características de Java pueden ser explotados en la compilación o ejecución.

Java fue modificado para permitir la implementación de anotaciones:

- Se agregó una sintaxis dedicada en Java para permitir la definición y el uso de anotaciones.
- bytecode se ha mejorado para permitir el almacenamiento de anotaciones.

¿Dónde se pueden utilizar las anotaciones?

Las anotaciones se pueden utilizar con:

Paquetes, clases, interfaces, constructores, métodos, campos, parámetros, variables o anotaciones en sí.

Categorías de anotación.

Hay tres categorías de anotación:

- **Marcadores** : Estas anotaciones no tienen un atributo.

Por ejemplo `@Deprecated` , `@Override` ...

- **Anotación de valor único** : estas anotaciones tienen un solo atributo.

Por ejemplo, `@MyAnnotation ( "test")`

- **Anotaciones completas** : estas anotaciones tienen múltiples atributos.

Por ejemplo, `@MyAnnotation (arg1 = "test 3", arg2 = "test 2", arg3 = "test3")`

Como vemos antes, puedes crear tu propia anotación.

Anotación de alcance de bean gestionado

Crear bean gestionado

Para crear un bean de gestión necesita la anotación `@ManagedBean`

por ejemplo:

```
@ManagedBean
public class Example {}
```

Necesitas el paquete:

```
import javax.faces.bean.ManagedBean;
```

Alcance del frijol administrado

Utilizamos anotaciones para definir el alcance en el que se almacenará el bean.

Hay muchos alcances de bean administrado: `@NoneScoped`, `@RequestScoped`, `@ViewScoped`, `@SessionScoped`, `@ApplicationScoped` , ...

- Aplicación (`@ApplicationScoped`): el alcance de la aplicación persiste en todas las interacciones de los usuarios con una aplicación web.
- Sesión (`@SessionScoped`): el alcance de la sesión persiste en varias solicitudes HTTP en una aplicación web.
- Vista (`@ViewScoped`): el alcance de la vista persiste durante la interacción de un usuario con una sola página (vista) de una aplicación web.
- Solicitud (`@RequestScoped`): el alcance de la solicitud persiste durante una única solicitud HTTP en una aplicación web.
- Ninguno (`@NoneScoped`): indica que el alcance no está definido para la aplicación.
- Personalizado (`@CustomScoped`): un alcance no estándar definido por el usuario. Su valor debe estar configurado como `java.util.Map` . Los ámbitos personalizados se utilizan con poca frecuencia.

Lea Anotaciones JSF en línea: https://riptutorial.com/es/jsf/topic/7021/anotaciones-jsf

Capítulo 3: Comentarios en JSF

Introducción

JSF como lenguaje de marcado, admite comentarios de algunas partes del código, pero debemos ser cuidadosos, porque si usamos un código de comentario HTML normal como este: <! - Quiero comentar el siguiente botón -> <! - <h: commandButton value = "Push" onclick = "alert ('Hello');" /> -> Es posible que no haya comentado nada. Esto se debe a que JSF procesa este código como predeterminado, incluso si se comenta entre las etiquetas `<!--` y `-->` . Hay dos soluciones para comentar cualquier código JSF.

Sintaxis

- <ui: eliminar> código JSF que desea comentar </ ui: eliminar>

Observaciones

Puede encontrar más información en la documentación de Oracle:

- <ui: eliminar> en oracle.com
- SKIP_COMMENTS en facelets.java.net

Examples

Usar etiqueta

Necesitamos usar las etiquetas `<ui:remove>` y `</ui:remove>` entre cualquier código JSF que queramos comentar.

```
<ui:remove>
    <h:outputLabel value="Yeah, I'm really commented" />
</ui:remove>
```

Por supuesto, necesita agregar estos xmlns a su etiqueta html de encabezado. Compruebe este ejemplo completo mínimo:

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

    <ui:remove>
        <h:outputLabel value="Yeah, I'm really commented" />
    </ui:remove>

</html>
```

Configurar facelets.SKIP_COMMENTS

Debe agregar a web.xml una etiqueta de configuración como esta:

```
<context-param>
    <param-name>facelets.SKIP_COMMENTS</param-name>
    <param-value>true</param-value>
</context-param>
```

Ahora puedes usar la etiqueta de comentarios HTML normal `<!--` y `-->`

```
<!--
    <h:outputLabel value="Yeah, I'm really commented" />
-->
```

Anteriormente, el ejemplo completo con facelets.SKIP_COMMENTS configurado en web.xml será:

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

    <!--
        <h:outputLabel value="Yeah, I'm really commented" />
    -->

</html>
```

Lea Comentarios en JSF en línea: https://riptutorial.com/es/jsf/topic/8164/comentarios-en-jsf

Capítulo 4: El alcance del flash en JSF 2

Observaciones

El concepto de Flash se toma de Ruby on Rails y proporciona una forma de pasar objetos temporales entre las vistas de usuario generadas por el ciclo de vida de las caras. Al igual que en Rails, cualquier cosa que se coloque en el flash se expondrá a la siguiente vista encontrada por la misma sesión de usuario y luego se borrará. Es importante tener en cuenta que la "vista siguiente" puede tener el mismo ID de vista que la vista anterior.

La implementación de JSF debe garantizar que el comportamiento correcto de la memoria flash se conserve incluso en el caso de un `<navigation-case>` que contenga un `<redirect />` . La implementación debe garantizar que el comportamiento correcto del flash se mantenga incluso en el caso de solicitudes GET adyacentes en la misma sesión. Esto permite que las aplicaciones de Faces utilicen completamente el patrón de diseño "Publicar / Redirigir / Obtener".

Examples

Demostración del uso del Flash Scope

Bean1

```
@ManagedBean
@ViewScoped
public class Bean1 implements Serializable {

    /**
     * Just takes the given param, sets it into flash context and redirects to
     * page2
     *
     * @param inputValue
     * @return
     */
    public String goPage2(String inputValue) {
        FacesContext.getCurrentInstance().getExternalContext().getFlash()
                .put("param", inputValue);
        return "page2?faces-redirect=true";
    }

}
```

page1.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:c="http://java.sun.com/jsp/jstl/core">
<h:head />
<h:body>
```

```
    <!-- Sets the first flash param at the action method -->
    <h:form>
        <h:inputText value="#{inputValue}" />
        <h:commandButton action="#{bean1.goPage2(inputValue)}"
            value="Go Page 2" />
    </h:form>

    <!-- Sets the second flash param -->
    <c:set target="#{flash}" property="param2" value="Myparam2" />

    <!-- Tries to retrieve both of the params.
    Note none of them is displayed at the first page hit.
    If page refreshed, the second param which has been already set, will be displayed -->
    <p>Param1: #{flash['param']}</p>
    <p>Param2: #{flash['param2']}</p>
</h:body>
</html>
```

Bean2

```
@ManagedBean
@ViewScoped
public class Bean2 implements Serializable {

    public String getParam() {
        /**
         * Takes the parameter from the flash context
         */
        return (String) FacesContext.getCurrentInstance().getExternalContext()
                .getFlash().get("param");
    }

}
```

page2.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core">
<h:head />
<!-- This page just displays the received params -->
<h:body>
    <!-- Different ways to retrieve the params from the flash scope -->
    <p>Param1: #{bean2.param}</p>
    <p>Param1: #{flash.param}</p>
    <p>Param1: #{flash['param']}</p>
    <p>Param2: #{flash['param2']}</p>

    <!-- Keep the first param for next redirection -->
    #{flash.keep.param}

    <!-- Return to page1 and see how the first param is retained -->
    <h:button outcome="page1?faces-redirect=true" value="return to 1" />
</h:body>
</html>
```

Lea El alcance del flash en JSF 2 en línea: https://riptutorial.com/es/jsf/topic/3906/el-alcance-del-

flash-en-jsf-2

Capítulo 5: Integración Ajax

Examples

Actualizar parcialmente la vista

Realiza una solicitud ajax y actualiza solo parte de la vista.

Bean.java

```
@ManagedBean
@ViewScoped
public class Bean {

    public Date getCurrentDate() {
        return new Date();
    }

}
```

sample.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<h:head />
<h:body>
    <h:form>
        <h:commandButton value="Execute ajax">
            <f:ajax render="output" />
        </h:commandButton>
        <p>
            <h:outputText id="output" value="Ajax date: #{bean.currentDate}" />
        </p>
        <p>
            <h:outputText id="output2" value="Non-Ajax date: #{bean.currentDate}" />
        </p>
    </h:form>
</h:body>
</html>
```

Envíe las partes del formulario y escuche la solicitud.

Hace una solicitud solo enviando parte del formulario. El valor de `text1` se establece, pero no `text2` , como indica el oyente.

Bean.java

```
@ManagedBean
@ViewScoped
public class Bean {
```

```
    private String text1;

    private String text2;

    public String getText1() {
        return text1;
    }

    public void setText1(String text1) {
        this.text1 = text1;
    }

    public String getText2() {
        return text2;
    }

    public void setText2(String text2) {
        this.text2 = text2;
    }

    public void listener() {
        System.out.println("values: " + text1 + " " + text2);
    }

}
```

sample.xhtml

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<h:head />
<h:body>
    <h:form>
        <h:inputText id="my_input" value="#{bean.text1}" />
        <h:inputText value="#{bean.text2}" />
        <h:commandButton value="Execute ajax">
            <f:ajax execute="@this my_input" listener="#{bean.listener}" />
        </h:commandButton>
    </h:form>
</h:body>
</html>
```

Ajax en evento javascript

La fecha se actualiza cada vez que el usuario escribe en el campo de entrada:

Bean.java

```
@ManagedBean
@ViewScoped
public class Bean {

    public Date getCurrentDate(){
```

```
        return new Date();
    }

}
```

sample.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<h:head />
<h:body>
    <h:form>
        <h:inputText>
            <f:ajax event="keyup" render="output" />
        </h:inputText>
        <p>
            <h:outputText id="output" value="Ajax date: #{bean.currentDate}" />
        </p>
    </h:form>
</h:body>
</html>
```

Retrasar

Ejecuta las solicitudes con el retraso especificado en milisegundos, lo que significa que si alguna solicitud posterior ocurre después de que se haya puesto en cola la anterior, la primera se omitirá. La característica está disponible a partir de JSF 2.2:

Bean.java

```
@ManagedBean
@ViewScoped
public class Bean {

    public Date getCurrentDate(){
        return new Date();
    }

}
```

sample.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<h:head />
<h:body>
    <h:form>
        <h:inputText>
            <f:ajax event="keyup" render="output" delay="2000" />
        </h:inputText>
        <p>
```

```
            <h:outputText id="output" value="Ajax date: #{bean.currentDate}" />
        </p>
    </h:form>
</h:body>
</html>
```

Lea Integración Ajax en línea: https://riptutorial.com/es/jsf/topic/4916/integracion-ajax

Capítulo 6: Plantillas JSF

Observaciones

JSF proporciona etiquetas especiales para crear un diseño común para una aplicación web llamada etiquetas **facelets** . Estas etiquetas ofrecen flexibilidad para administrar partes comunes de varias páginas en un solo lugar.

Espacios de nombres:

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

Examples

Cómo crear una plantilla

Configuración de una plantilla para una aplicación

Cree un archivo llamado `template.xhtml` debajo de la carpeta `/WEB-INF` , de esa manera los archivos de plantilla estarán accesibles solo para el marco.

`/WEB-INF/template.xhtml`

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

    <h:head>
        <title><ui:define name="title">Default title</ui:define></title>
    </h:head>

    <h:body>
        <!-- Some styles that we might include for our whole application -->
        <h:outputStylesheet name="css/template.css" />

        <!-- Shared content for the application, e.g. a header, this can be an include -->
        <div>
            Application Header
        </div>

        <!-- The content we want to define in our template client -->
        <div>
            <ui:insert name="content" />
        </div>
    </h:body>
</html>
```

Este archivo actuará como plantilla para la aplicación. Ahora definiremos una vista específica en nuestro directorio de vistas.

`/home.xhtml`

```
<ui:composition template="/WEB-INF/template.xhtml"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

    <ui:define name="title">Home</ui:define>

    <ui:define name="content">
        Welcome to the application!
    </ui:define>
</ui:composition>
```

Eche un vistazo al atributo de `template` en esta vista de cliente, esto le indica a JSF que use la plantilla que queremos. Luego, utilizando `<ui:define>` definimos contenido específico para insertarlo donde se indica en la plantilla. Al acceder a `/home.xhtml` en el cliente, se mostrará el resultado completo.

Lea Plantillas JSF en línea: https://riptutorial.com/es/jsf/topic/5033/plantillas-jsf

Creditos

S. No	Capítulos	Contributors
1	Empezando con jsf	BalusC, Community, Emil Sierżęga, Pixelstix
2	Anotaciones JSF	Right leg, YCF_L
3	Comentarios en JSF	albertoiNET
4	El alcance del flash en JSF 2	Xtreme Biker
5	Integración Ajax	Xtreme Biker
6	Plantillas JSF	BalusC, DimaSan, Emil Sierżęga, Xtreme Biker