Бесплатная электронная книга

УЧУСЬ

jsf

Free unaffiliated eBook created from
Stack Overflow contributors.

#jsf

Около

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

глава 1: Начало работы с jsf

замечания

JavaServer Faces (JSF) - это модель-представление-презентатор, обычно используемая для создания веб-приложений на основе форм HTML. Используя стандартные компоненты и набор рендеринга, представления HTML в виде состояния могут быть определены с использованием меток Facelets или JSP и подключены для моделирования данных и логики приложений с помощью бэкэндов.

Версии

Версия	Примечания к выпуску	Дата выхода
2,3	Объявление от 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

Установка JSF

Содержимое было перенесено на страницу добрых ссылок на JSF

Привет, мир

Содержимое было перенесено на страницу добрых ссылок на JSF

Минимальные требования

Содержимое было перенесено на страницу добрых ссылок на JSF

Прочитайте Начало работы с jsf онлайн: https://riptutorial.com/ru/jsf/topic/916/начало-работы-

c-jsf

глава 2: JSF Аннотации

замечания

Я получаю много информации с этих веб-сайтов:

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

Examples

Введение в аннотации

Почему аннотации?

Как правило, мы используем аннотацию, чтобы облегчить разработку и сделать код более понятным и чистым.

Что такое аннотации?

Аннотации Java 5 обеспечивают стандартизацию метаданных в общей цели. Эти метаданные, связанные с функциями Java, могут быть использованы в компиляции или выполнении.

Java была изменена для реализации аннотаций:

- В Java был добавлен выделенный синтаксис, позволяющий определять и использовать аннотации.
- байт-код улучшен, что позволяет хранить аннотации.

Где можно использовать аннотации?

Аннотации можно использовать с:

пакеты, классы, интерфейсы, конструкторы, методы, поля, параметры, переменные или аннотации.

Категории аннотации

Существует три категории аннотаций:

- **Маркеры** : эти аннотации не имеют атрибута

Например `@Deprecated` , `@Override` ...

- **Аннотации с одним значением** : эти аннотации имеют только один атрибут

Например, `@MyAnnotation ( "test")`

- **Полные аннотации** : эти аннотации имеют несколько атрибутов

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

Как мы видим, прежде чем вы сможете создать свою собственную аннотацию

Аннотации управляемого управляемого компонента

Создание управляемого компонента

Чтобы создать управляющий компонент, вам понадобится аннотация `@ManagedBean`

например:

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

Вам нужен пакет:

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

Управляемый компонент боба

Мы используем аннотации для определения области, в которой будет храниться компонент.

Существует много возможностей управляемого компонента: `@NoneScoped`, `@RequestScoped`, `@ViewScoped`, `@SessionScoped`, `@ApplicationScoped` , ...

- Приложение (`@ApplicationScoped`): область применения сохраняется во всех взаимодействиях пользователей с веб-приложением.
- Session (`@SessionScoped`): Область сеанса сохраняется в нескольких HTTP-запросах в веб-приложении.
- View (`@ViewScoped`): область просмотра сохраняется во время взаимодействия пользователя с одной страницей (представлением) веб-приложения.
- Запрос (`@RequestScoped`): область запроса сохраняется в течение одного HTTP-запроса в веб-приложении.
- None (`@NoneScoped`): указывает, что область не определена для приложения.
- Пользовательский (`@CustomScoped`): Пользовательский, нестандартный объем. Его значение должно быть настроено как `java.util.Map` . Пользовательские области используются нечасто.

Прочитайте JSF Аннотации онлайн: https://riptutorial.com/ru/jsf/topic/7021/jsf-аннотации

глава 3: Интеграция Ajax

Examples

Частично обновить представление

Делает запрос ajax и обновляет только часть представления.

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>
```

Отправьте детали формы и послушайте запрос

Делает запрос только отправкой части формы. Значение `text1` устанавливается, но не `text2` , поскольку состояние прослушивателя.

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 на javascript событии

Дата обновляется всякий раз, когда пользователь вводит в поле ввода:

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>
```

задержка

Выполняет запросы с указанной задержкой в миллисекундах, а это означает, что если какой-либо последующий запрос произойдет после того, как предыдущий был поставлен в очередь, первый будет пропущен. Эта функция доступна с 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>
```

Прочитайте Интеграция Ajax онлайн: https://riptutorial.com/ru/jsf/topic/4916/интеграция-ajax

глава 4: Комментарии в JSF

Вступление

JSF как язык разметки, поддерживает комментарии некоторых частей кода, но мы должны быть осторожны, потому что если мы используем обычный код комментария HTML следующим образом: <! - Я хочу прокомментировать следующую кнопку -> <! - <h: commandButton value = "Push" onclick = "alert ('Hello');" /> -> Возможно, что он ничего не прокомментировал. Это связано с тем, что JSF обрабатывает этот код по умолчанию, даже если он прокомментирован между тегами `<!--` и `-->` . Есть два решения, чтобы прокомментировать любой код JSF

Синтаксис

- <ui: remove> Код JSF, который вы хотите прокомментировать </ ui: remove>

замечания

Дополнительную информацию вы можете найти в документации Oracle:

- <ui: remove> на oracle.com
- SKIP_COMMENTS на facelets.java.net

Examples

Использовать тег

Нам нужно использовать тег `<ui:remove>` и `</ui:remove>` между любым кодом JSF, который мы хотим прокомментировать.

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

Конечно, вам нужно добавить этот xmlns в свой HTML-тег заголовка. Проверьте этот минимальный полный пример:

```
<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>
```

Настроить facelets.SKIP_COMMENTS

Вы должны добавить в web.xml тег конфигурации, например:

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

Теперь вы можете использовать обычный тег комментариев HTML `<!--` и `-->`

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

Превосходно полный пример с facelets.SKIP_COMMENTS, настроенный в web.xml, будет:

```
<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>
```

Прочитайте Комментарии в JSF онлайн: https://riptutorial.com/ru/jsf/topic/8164/комментарии-в-jsf

глава 5: Область Flash в JSF 2

замечания

Концепция Flash взята из Ruby on Rails и обеспечивает способ передачи временных объектов между представлениями пользователей, сгенерированными жизненным циклом лиц. Как и в Rails, все, что помещается во флэш-память, будет отображаться в следующем представлении, обнаруженном одним и тем же сеансом пользователя, а затем очищается. Важно отметить, что «следующий вид» может иметь тот же идентификатор вида, что и предыдущий.

Реализация JSF должна гарантировать, что правильное поведение вспышки сохраняется даже в случае `<navigation-case>` который содержит `<redirect />` . Реализация должна гарантировать, что правильное поведение вспышки сохраняется даже в случае смежных запросов GET на том же сеансе. Это позволяет приложениям Faces полностью использовать шаблон дизайна «Post / Redirect / Get».

Examples

Демонстрация использования 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>
```

Прочитайте Область Flash в JSF 2 онлайн: https://riptutorial.com/ru/jsf/topic/3906/область-flash-в-jsf-2

глава 6: Шаблоны JSF

замечания

JSF предоставляет специальные теги для создания общей компоновки для веб-приложения, называемого тегами **facelets** . Эти теги предоставляют гибкость для управления общими частями нескольких страниц в одном месте.

Пространства имен:

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

Examples

Как создать шаблон

Настройка шаблона для одного приложения

Создайте файл с именем `template.xhtml` в папке `/WEB-INF` , таким образом, файлы шаблонов будут доступны только для фреймворка.

`/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>
```

Этот файл будет действовать как шаблон для приложения. Теперь мы определим конкретное представление в нашем каталоге.

`/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>
```

Посмотрите на атрибут `template` в этом представлении клиента, это говорит JSF использовать необходимый шаблон. Затем, используя `<ui:define>` мы определяем конкретный контент, который нужно вставить, где он указан в шаблоне. Доступ к `/home.xhtml` в клиенте даст весь результат.

Прочитайте Шаблоны JSF онлайн: https://riptutorial.com/ru/jsf/topic/5033/шаблоны-jsf

кредиты

S. No	Главы	Contributors
1	Начало работы с jsf	BalusC, Community, Emil Sierżęga, Pixelstix
2	JSF Аннотации	Right leg, YCF_L
3	Интеграция Ajax	Xtreme Biker
4	Комментарии в JSF	albertoiNET
5	Область Flash в JSF 2	Xtreme Biker
6	Шаблоны JSF	BalusC, DimaSan, Emil Sierżęga, Xtreme Biker