JavaServer Faces 2.0

- JSF 2.0 ist da -

Andy Bosch
Freier Berater und Trainer
Betreiber von **www.jsf-forum.de** und **www.jsf-portlets.net**
SENS-Experte (**www.SoftwareExperts.de**)

JavaServer Faces 2.0

- JSF 2.0 ist SO GUT WIE da -

Andy Bosch
Freier Berater und Trainer
Betreiber von **www.jsf-forum.de** und **www.jsf-portlets.net**
SENS-Experte (**www.SoftwareExperts.de**)

Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- Navigation
- Resource Handling
- Ajax
- SystemEvents
- ProjectStage
- Fazit

Agenda

- **Entstehungsgeschichte**
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- Navigation
- Resource Handling
- Ajax
- SystemEvents
- ProjectStage
- Fazit

Entstehungsgeschichte

- JSF 1.0 wurde im Sommer 2004 veröffentlicht

- JSF 1.2: Kleineres Update im Mai 2006
  Fokus: Alignment mit der JSP EL → Unified EL

- JSF 2.0 (JSR-314) begonnen im Sommer 2007
  Spec Lead: Ed Burns und Roger Kitain

- Aktueller Status: Spec ist „fertig"
  Mojarra: Beta 1
  MyFaces: Bald ein Alpha

Wie entstand die Spezifikation?

- Offene Diskussion

- Einholen von Ideen aus dem OpenSource Bereich

- **Keine** Lösung entwickelt im Elfenbeinturm

- Sehr aktive, kommunikative Expert Group.
- Viele bekannte Namen, die vormals Lösungen im JSF-Umfeld erstellt haben, z.B.
  - Gavin King (Seam),
  - Alexandr Smirnov (Rich Faces)
  - Ted Goddard (ICEFaces)
  - Ken Paulson (JSF Templating).
  - Jacob Hookom (Facelets)
  - …

Die Community befragt

- Convention over Configuration
- JSP war gestern, Facelets wird zum Standard
- Ajax / JavaScript Integration basierend auf den Erfahrungen von JBoss RichFaces, ICEfaces, Trinidad
- Bookmarkable URLs: Beeinflusst durch JBoss / Seam
- Validation: JSR-303 (Bean Validation)
- ...

Agenda

- Entstehungsgeschichte
- **Ein kurzer Überblick**
- Facelets-Integration
- Managed Beans
- Navigation
- Resource Handling
- Ajax
- SystemEvents
- ProjectStage
- Fazit

Tolle neue Feature-Welt ...

Ajax

Validation

Resources

Annotations

Config Annotations

Composite Components

Facelets

System Events

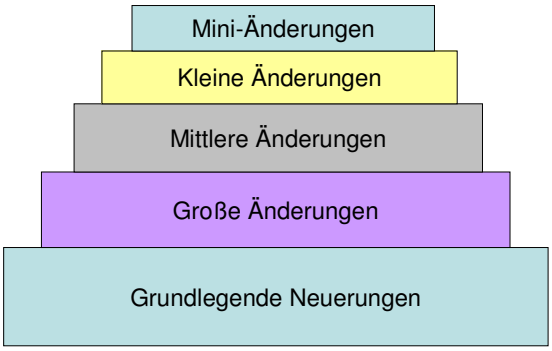
Project Stage

New Scopes

Tolle neue Feature-Welt ...

Es gibt sooooo viel Neues ... wo soll ich nur anfangen?

→ Einteilung der Features nach Ed Burns, JSF Spec Lead



Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- **Facelets-Integration**
- Managed Beans
- Navigation
- Resource Handling
- Ajax
- SystemEvents
- ProjectStage
- Fazit

PDL – Page Declaration Language

Standardization and improvement of Facelets.
We could call it Facelets 2.0.

- Kernfunktionalität von PDL wird geerbt von Facelets und JSFTemplating

- Facelets wird in den Standard als bevorzugte Viewtechnologie erhoben.

- Es erfolgt eine Trennung in eine API (Spec) und eine Impl (Details der Implementierung)

Was ist eigentlich Facelets?

- Ersatz für JSP → XHtml-Seiten
- Seiten werden nicht mehr kompiliert, sondern als spezieller Syntaxbaum aufbereitet
- Facelets bietet verschiedene Vereinfachungen für JSF, z.B.

```
<f:view>
   Guten Tag, #{SimpleBean.name}
</f:view>
```

- Templating ist direkt in Facelets integriert.

Composite Components (1)

- Neue Komponenten in XHtml definieren

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:jp="http://java.sun.com/jsf/composite/halloLib">
<h:head>
    <title>CC Demo</title>
</h:head>
<h:body>
  <h:form>
    <jp:hallo value="Andy Bosch" />
  </h:form>
</h:body>
</html>
```

Composite Components (2)

- hallo.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:composite="http://java.sun.com/jsf/composite">
<body>
  <composite:interface>
    <composite:attribute name="value" required="true" />
  </composite:interface>
  <composite:implementation>
    <h:outputText
        value="Hallo #{compositeComponent.attrs.value}" />
  </composite:implementation>
</body>
</html>
```


Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- **Managed Beans**
- Navigation
- Resource Handling
- Ajax
- SystemEvents
- ProjectStage
- Fazit

Managed Beans

- Convention over Configuration

- @ManagedBean

- Managed Bean Name = Klassenname by Default
  (erster Buchstabe wird klein)

- Default Scope = Request Scope

- Beans können in weiteren Scopes abgelegt werden.
→ ViewScope und FlashScope

Managed Beans mit Annotations

```
<managed-bean>
  <managed-bean-name>SimpleBean</managed-bean-name>
  <managed-bean-class>
     de.jsf.SimpleBean
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

```
@ManagedBean(name="StageContainer")
@SessionScoped
public class StageContainer {
  ...
}
```

Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- **Navigation**
- Resource Handling
- Ajax
- SystemEvents
- ProjectStage
- Fazit

Navigation per Konvention

- Weiterhin ist die Angabe von Navigationsregeln möglich.
- Es gibt jetzt jedoch auch die Möglichkeit, per Konvention zu navigieren.

```
<h:commandButton value=„Zeige Detail" action=„detail"/>
```

→Suche nach **detail.xhtml** im gleichen Verzeichnis

Auch möglich:

```
<h:commandButton value=„Zeige Detail"
   action=„/modul1/detail"/>

<h:commandButton value=„Zeige Detail"
   action=„/modul1/detail?faces-redirect=true„ />
```

Navigation: Jetzt auch GET

JSF alt: Immer nur POST

Jetzt endlich auch GET !

```
<h:link outcome=„meineSeite" value=„Klick mich">
    <f:param name="myParam" value="myParamValue"/>
</h:link>
```

Im Gegensatz zu h:outputLink wird der Lifecyle komplett durchlaufen.

In diesem Zusammenhang: ViewParameter

http://localhost:8080/meineAnwendung/faces/seite.xhtml?pId=4711

```
<f:metadata>
  <f:viewParam name="pId" value="#{cart.productId}" />
</f:metadata>

<h:body>

   Ausgewähltes Produkt: #{cart.productId}

</h:body>
```

Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- Navigation
- **Resource Handling**
- Ajax
- SystemEvents
- ProjectStage
- Fazit

Resource Handling

Was ist überhaupt eine Resource?

Resources are any artifacts that a component may need in order to be rendered properly to a user-agent. So think images, CSS, or JavaScript files.

In früheren Versionen gab es keinerlei Unterstützung für Resourcen in JSF.

→Jetzt viel bessere Unterstützung speziell für Komponentenentwickler

Resource Handling regelt, wie Resourcen gepackaged werden, wie man sie aus jar-Dateien lädt und managed.

Resource Handling

Es wird in folgenen Stellen nach Resourcen gesucht:

- **/resources**: Liegt direkt im Root der Webapplikation und wird im Projekt selbst gesucht
- **/META-INF/resources:** Weitere Resourcen im Classpath

Resource Identifiers:
• [localePrefix/][libraryName/][libraryVersion/]resourceName [/resourceVersion]

Vorteil: Da immer die aktuellste Version genommen werden muss, kann zur Laufzeit eine Resource aktualisiert werden.

Resource Relocation

- Komponenten benötigen in der Regel weitere Resourcen, um sich darzustellen.
- Komponenten müssen oftmals Angaben in den HEAD-Bereich einer Seite schreiben
- Beim Rendern einer Komponente ist der Head jedoch bereits geschrieben
- Im Head weiß der Seitenautor oftmals nicht, welche Resourcen benötigt werden.

→ Komponentenbibliotheken wie z.B. RichFaces oder Tomahawk reichern den Request über einen Filtermechanismus an.

Resource Relocation

Neue Tags:

- h:head
- h:body
- h:outputScript
- h:outputStylesheet

```
<h:outputScript  library="myjs" name="myscripts.js" target="body" />
```

Achtung:
Bei Stylesheets wird das Target-Attribut ignoriert und es erfolgt immer eine Ausgabe im Head !

Resource Relocation

```
<f:view contentType="text/html">

<h:head>
   <title>Resource Relocation</title>
</h:head>

<h:body>

   <h:form>
       <h:outputScript library="myjs"
              name="myscripts.js" target="body" />
   </h:form>

</h:body>

</f:view>
```

Resource Loading

```
@ResourceDependency
       (name=”greatJavaScript.js”,
        library=”GreatJS-Lib”, target=„head“)
public class MyGreatInputComponent extends UIInput {...}
```

```
Application app =
   FacesContext.getCurrentInstance().getApplication();

Resource resource =
         app.getResourceHandler().createResource(
                      “greatScript.js”,”GreatJS-Lib”);
```


Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- Navigation
- Resource Handling
- **Ajax**
- SystemEvents
- ProjectStage
- Fazit

Ajax

Erstaunliches zuerst:

Though not a new technology, the first time JavaScript APIs were introduced into Java EE.

Ajax

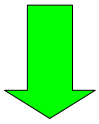
- Einheitliches Vorgehen bei JavaScript in einer Komponente

<h:outputScript name=*„jsf.js"* library=*"javax.faces"* target=*"head"*/>

```
<h:inputText id="eingabe"
   onchange="jsf.ajax.request(
      this, event, {execute:'eingabe',
                    render:'ausgabe'});
      return false;" />
```

Ajax-ifizieren

```
<h:commandButton value=„MachWas"
   action="#{MyUIController.machMalWasTollen}" />
```



```
<h:commandButton id="button1" value=„MachWas"
   action="#{MyUIController.machMalWasTollen}">

   <f:ajax execute="@this"
           render=„meinFeld1 meinFeld2"/>

</h:commandButton>
```


Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- Navigation
- Resource Handling
- Ajax
- **SystemEvents**
- ProjectStage
- Fazit

System Events

- Werden bei speziellen Punkten im Lebenszyklus der Anwendung aktiviert, z.B.:
  - PostAddToVievEvent
  - PreRenderComponentEvent
  - PreRenderViewEvent
  - PreValidateEvent
  - PostRestoreStateEvent
  - PostValidateViewEvent
- Bisherige Events waren auf Actions- oder ValueChanges beschränkt (naja, plus PhaseEvents)
- Damit kann z.B. der „Workaround" ServletContextListener innerhalb von JSF gelöst werden

System Events

```
<h:outputText id="beforeRenderTest1" >
  <f:event type="preRenderComponent"
     listener="#{SimpleEventController.processEvent}" />
</h:outputText>
```

Es wird ein entsprechender Event an der Komponente registriert.

Listeners können prinzipiell auf drei Ebenen registriert werden:

- component: UIComponent.subscribeToEvent()
- view: UIViewRoot.subscribeToEvent()
- application: Application.subscribeToEvent()

Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- Navigation
- Resource Handling
- Ajax
- SystemEvents
- **ProjectStage**
- Fazit

ProjectStage

- In JSF sind verschiedene Stages bekannt:
  - Production
  - Development
  - UnitTest
  - SystemTest
  - Extension

- Auf die verschiedenen Stages kann programmatisch reagiert werden

- Setzen des Stage fix in web.xml oder als JNDI

ProjectStage

```
<context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
</context-param>
```

```
FacesContext jsfContext =
       FacesContext.getCurrentInstance();

Application app = jsfContext.getApplication();
String sStage = app.getProjectStage().toString();
```


Agenda

- Entstehungsgeschichte
- Ein kurzer Überblick
- Facelets-Integration
- Managed Beans
- Navigation
- Resource Handling
- Ajax
- SystemEvents
- ProjectStage
- **Fazit**

Fazit

- Darauf hat die Community lange gewartet
- Viele coole neue Sachen
- Der „Hören" auf die Community hat der JSF-Version gut getan
- JSF 2.0 macht einen sehr umfangreichen und ausgereiften Eindruck
- Wird in JavaEE 6 mit aufgenommen, eventuell mit einer kleineren Erweiterung
- **Es macht einfach Spaß, die neuen Features auszutesten !!!**

Tutorials? Bücher? Wo gibt's Know how?

- JSF 2.0 Spezifikation (JSR-314)
- JSF 2.0 Java-Docs
- PDL Docs
- JS Docs (JavaScript API for JSF)
- Beispiele von der JSF RI
- Hoffentlich bald auch auf www.jsf-forum.de ;-)

Fragen ???

Weitere Tutorials und viel Material unter:

www.jsf-portlets.net
oder
www.jsf-forum.de

Gerne auch:
andy.bosch@jsf-forum.de

http://twitter.com/andybosch