15.–18. 09. 2008
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Dienstgesichter

Einführung in JavaServer Faces

Isabella Kneissl

MATHEMA Software GmbH

Pourya Harirbafan

Inhalt

- Einleitung
- Developer Roles
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- Facelets
- Conversion Model
- Internationalisierung (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- JSF 1.2 Neuerungen
- Erweiterungen
- Ausblick JSF 2.0
- Ressourcen

Orientierung

- Einleitung
- Developer Roles
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- …

JavaServer Faces (JSF)

- User interface framework für Java Web-Anwendungen
- Einfache Erstellung von UIs
- Erstellung durch Wiederverwendung von eigenen Oberflächenkomponenten
- Managed UI-Status über Server-Request hinweg
- Bietet einfaches Model für Verknüpfung von client- zu serverseitigen Ereignissen
- Vorbereitet zur Unterstützung von unterschiedlichen Markup-Sprachen (z.B. HTML, WML)

Java Server Faces

- Einfache Datenübertragung von Oberfläche zu Backend und umgekehrt
- Typkonvertierung (String zu Object und umgekehrt)
- Validierung von Benutzer-Eingaben
- Fehlerbehandlung und Rückmeldung an den Benutzer in lesbarer Form
- Seitennavigation in Abhängigkeit von Events und Model Interaktionen
- Unterstützt Internationalisierung
- Toolunterstützung

JavaServer Faces 1.1 & J2EE

- 2001 Spezifikation unter JSR-127
- Seit Juni 2004 in der Version JSF 1.1
- Standard für Java Webanwendungen
- JSF basiert auf:
  - Servlet 2.3 (JSR-53)
  - JSP 1.2 (JSR-53)
- JSF ist nicht Teil der J2EE 1.4
  - Aber im J2EE 1.4 SDK und im Tutorial zu finden
  - Bestandteil von Java EE 5.0

Java ServerFaces

- Java ServerFaces (JSF)
  http://java.sun.com/javaee/javaserverfaces/
- Version 1.2 Teil von Java EE 5.0 (=> Java 5.0)
- JSF 1.1 – JSR 127 (2004-05-27)
  http://www.jcp.org/en/jsr/detail?id=127
- JSF 1.2 – JSR 252 (2006-05-11)
  http://www.jcp.org/en/jsr/detail?id=252
- Referenzimplementierung 1.2_09 (2008-07-18)
  https://javaserverfaces.dev.java.net/
- GlassFish – Open Source Java EE 5 Application Server
  https://glassfish.dev.java.net/

Java ServerFaces - MyFaces

- Implementierung der JavaServer Faces Spezifikation (Java Specification Request 127)
- Apache Projekt http://www.myfaces.org/
- OpenSource, Apache Software License Version 2.0
- Aktuelle Version 1.2.4 (Stand August 2008), implementiert JSF 1.2
- Zusätzliche Funktionalität in **Tomahawk** und **Sandbox**
  - Komponenten
  - Validatoren
  - AJAX-Komponenten
  - ...

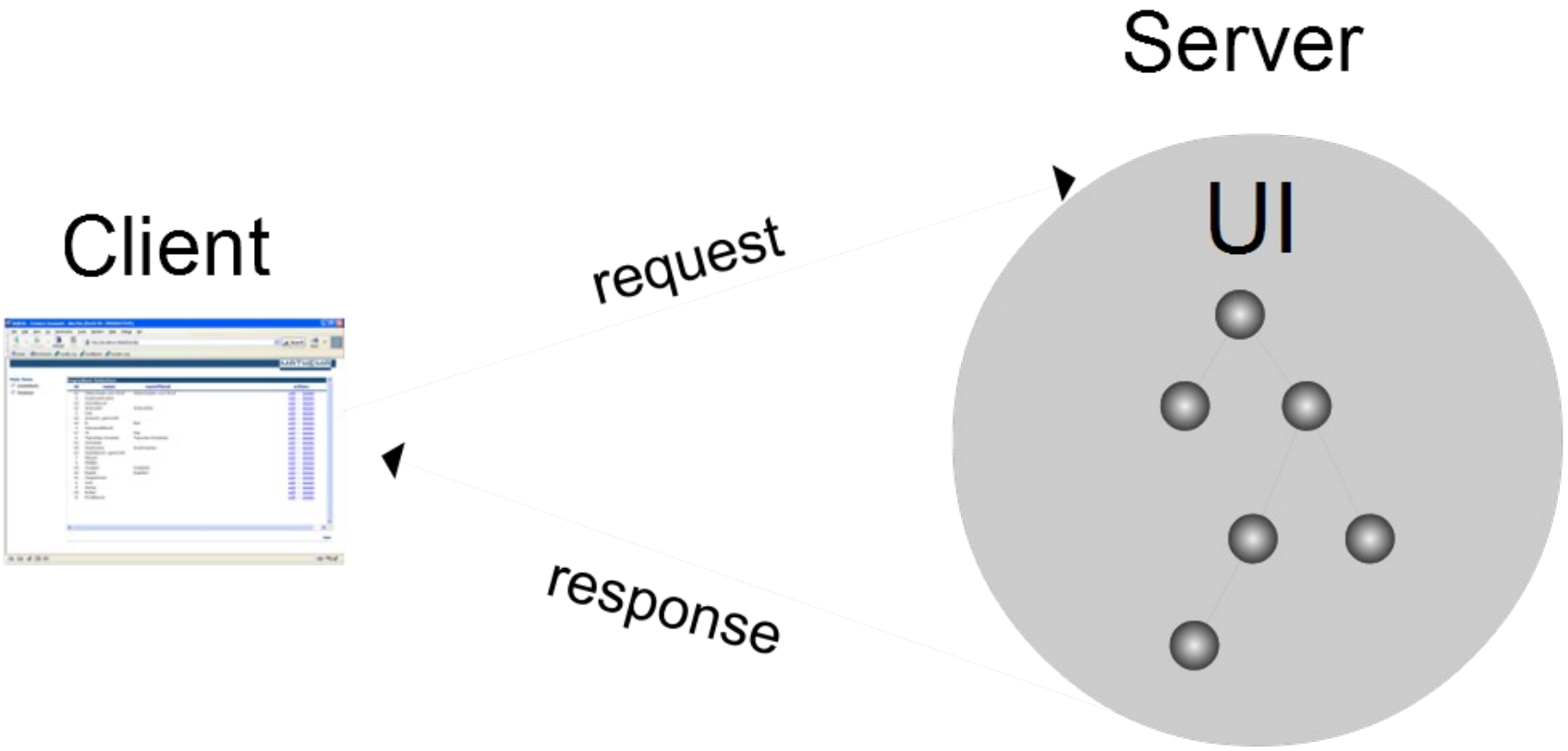
Was ist eine Java ServerFaces Anwendung?

- Webanwendung, wie jede andere Java-Webanwendung auch
- Ausführung in einem Servlet-Container (z.B. Tomcat)
- Besteht aus:
  - JavaBeans für Funktionalität (Controller) und Daten (modell)
  - Event-Listener (Controller)
  - Seiten als JSP (View)
  - Serverseitige Helper-Klassen, wie Datenbankzugriff, …

Was ist eine Java ServerFaces Anwendung?

- Tag-Library für UI-Komponenten
- Tag-Library zur Repräsentation von Event-Handler, Validatoren und anderen Aktionen
- UI-Komponenten repräsentieren ein statusbehaftetes Objekt auf dem Server
- Konfigurationsdatei (faces-config.xml)

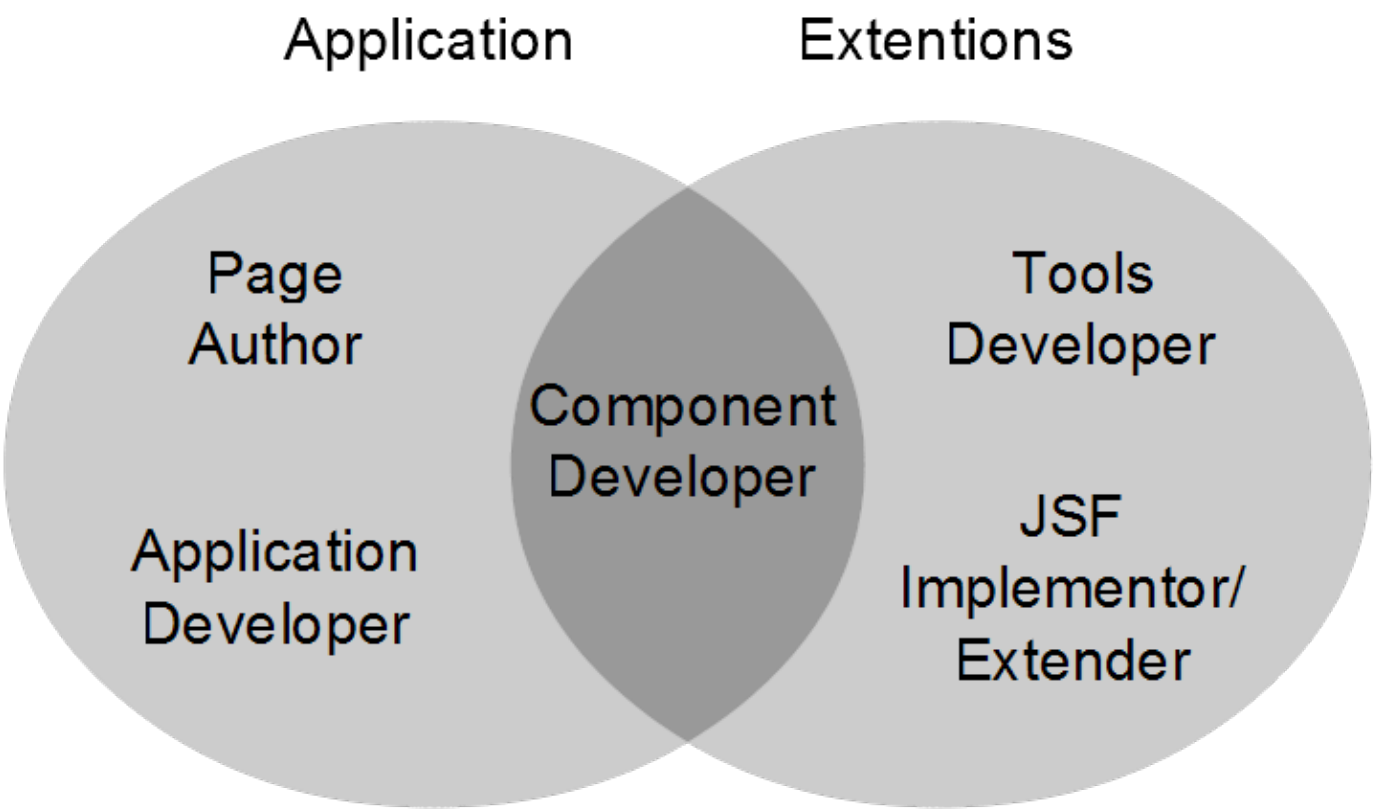
User Interface Framework



Orientierung

- Einleitung
- Developer Roles
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- …

JSF Developer Roles



JSF Developer Roles (Application)

- Page Author
  - Erstellt das User Interface einer Webanwendung
  - Benutzt Markup Language (JSP/HTML)
    - Standard JSF Tag Library
    - Vorgefertigte JSF Komponente
  - Erstellt das User Interface einer Webanwendung
  - Benutzer von Tools (Drag & Drop)
- Application Developer
  - Erstellt die serverseitige Funktionlität einer Webanwendung
  - Backing Bean, Validatoren, Konverter, Event Handler, Navigation
  - Business Logic / Persistence (JavaBean, EJB, …)

JSF Developer Roles (Extensions)

- Component Developer
  - Erstellt wiederverwendbare UI-Komponenten und Renderer
- Tools Developer
  - GUI-oriented page development tools
  - Unterstützt schnelle Entwicklung (Drag & Drop)
  - Beispiel: Exadel Studio
- JSF Implementor /Extender
  - Erstellt eine Umgebung (API) für eine JSF Applikation
  - Beispiel: MyFaces

Orientierung

- Einleitung
- Developer Roles
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- …

Einführendes Beispiel

- Einfaches Registrierungsformular mit
  - 3 Eingabefeldern mit Beschreibung
    - 2 mal <input type=„text“>
    - 1 mal <input type=„password“>
  - Link zur Datenübernahme

Beispiel JSP

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<html>
    <head><title>Beispiel JSP</title></head>
    <body>
        <f:view>
            <h:form>
                <h:outputText value="Email: " />
                <h:inputText id="email" value="#{registrationBean.email}"
                    validator="#{registrationBean.validateEmail}" />
                <h:outputText value="Name: " />
                <h:inputText id="name" value="#{registrationBean.name}" />
                <h:outputText value="Passwort: " />
                <h:inputSecret id="password"
                    value="#{registrationBean.password}" />
                <h:commandLink id="submitForm" value="Register"
                    action="#{registrationBean.registerUser}" />
            </h:form>
        </f:view>
    </body>
</html>
```

Beispiel web.xml

```
<web-app>
    <!-- Faces Servlet -->
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <!-- Faces Servlet Mapping -->
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>/jsp/index.jsp</welcome-file>
    </welcome-file-list>

</web-app>
```

Beispiel: faces-config.xml 1/2

```
<faces-config>

    <managed-bean>
        <description>
            Bean fuer User Registration
        </description>
        <manged-bean-name>registrationBean</managed-bean-name>
        <manged-bean-class>
            de.mathema.web.jsf.registration.RegistrationBean
        </managed-bean-class>
        <managed-bean-scope>session</managed-bean-scop>
    </managed-bean>
...
```

Beispiel: faces-config.xml 2/2

```
…

    <navigation-rule>
        <from-view-id>/jsp/registration.jsp</from-view-id>
        <navigation-case>
            <from-outcome>success</from-outcome>
            <to-view-id>/jsp/success.jsp</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-outcome>error</from-outcome>
            <to-view-id>/jsp/error.jsp</to-view-id>
        </navigation-case>

</faces-config>
```

Beispiel Backing Bean

```
public class RegistrationBean implements Serializable {

    private String email;
    private String name;
    private String password;

    // public getter und setter nach JavaBeans Konvention

    public String registerUser () {
        return "success";
    }

    public void validateEmail (FacesContext ctx, UIComponent comp,
        Object value) throws ValidatorException {

        // validate
    }
}
```

Orientierung

- Einleitung
- Developer Roles
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- …

Lifecycle Phasen in JSF 1/3

- Restore View
  - Wiederherstellung des UI-Komponenten-Baumes
  - Komponenten-Baum wird aus dem Faces Request wiederhergestellt und im FacesContext abgelegt
- Apply Request Values
  - Übernahme der Abfrage-Werte
  - Alle UI-Komponenten beziehen ihre Werte aus dem Request
  - Events für ValueChangeListener können hier abgefeuert werden (immediate=„true“)
  - Alle UI-Komponenten implementieren die Methode decode(javax.faces.context.FacesContext context)

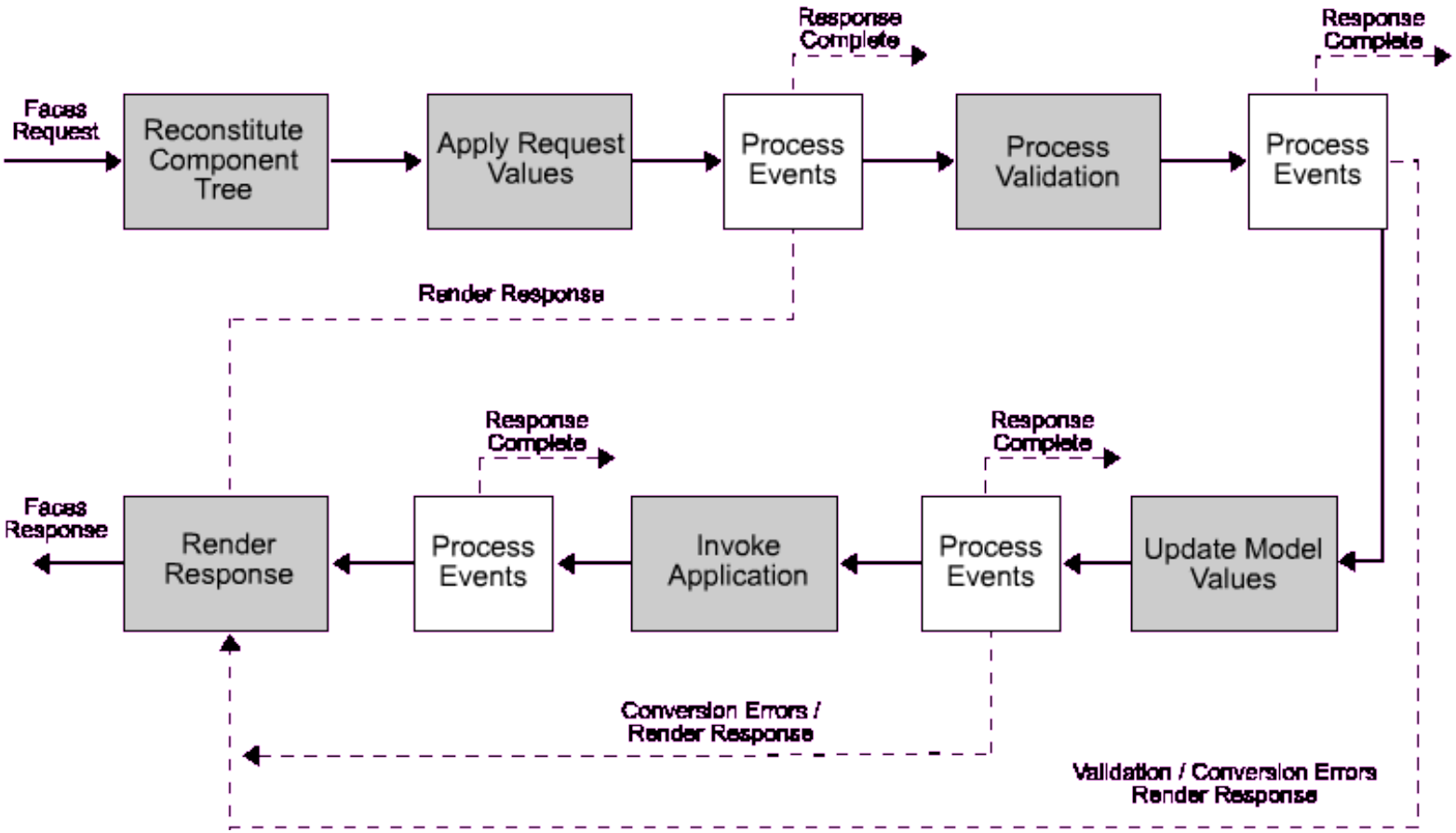
Lifecycle Phasen in JSF 2/3

- Process Validation
  - Verarbeitung der Validierung
  - Bei Fehler wird FacesContext.renderResponse() aufgerufen, um in die Render Response Phase zu springen
- Update Model Values
  - Aktualisierung der Modell-Objekte
  - Werte werden ins Modell übertragen
  - setter Aufrufe für Manages Beans

Lifecycle Phasen in JSF 3/3

- Invoke Application
  - Aufruf der Applikation (ActionListener)
  - Aufruf der Navigation Methode (Business-Logik)
  - Rückgabewert der Navigation Methode bestimmt die Navigation
- Render Response
  - Encode() Methode wird für jede Komponente, die sich im FacesContext Komponenten-Baum befindet, aufgerufen
  - Zustand des Komponenten-Baums wird gespeichert

Request Processing Lifecycle



Orientierung

- Einleitung
- Developer Roles
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- …

Konfiguration

- Konfiguration der JSF Anwendung erfolgt in
  - web.xml (Servlet Konfiguration)
  - faces-config.xml (JSF Konfiguration, vgl. mit struts-config.xml)

Konfiguration (faces-config.xml) 1/2

- faces-config.xml
  - XML
  - Semantik wird durch DTD festgelegt, ab JSF 1.2 durch XML-Schema
  - Konfiguration/Einstellungen
    - Konfiguration der Bean (Backing/Managed-Beans)
    - Registrierung von ResourveBundle/Ressourcen
    - Internationalisierung
    - Registrierung von Validatoren und Convertern
    - Konfiguration von Navigation Rules
    - Registrierung von Renderer durch Render Kit
    - Registrierung von eigenen Komponenten

Konfiguration (faces-config.xml) 2/2

- Mögliche Kindelemente von <faces-config>
  - <application>
  - <factory>
  - <component>
  - <converter>
  - <managed-bean>
  - <navigation-rule>
  - <referenced-bean>
  - <renderer-kit>
  - <lifecycle>
  - <validator>

Orientierung

- Einleitung
- Developer Roles
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- Facelets
- …

UI-Komponenten-Modell

- UI-Komponenten stellen die Benutzeroberfläche einer JSF-Anwendung dar
- JSF Spezifikation gibt die Architektur der Komponenten vor
  - Einige vordefinierte UI-Komponenten-Klassen mit bestimmten Zustand und Verhalten
  - Modell zum Rendering dieser Komponenten (Standard-Render-Kit: HTML)
  - Event- und Listener-Modell
  - Konvertierungs-Modell
  - Validierungs-Modell
- Alle UI-Komponenten-Klassen erweitern die Klasse javax.faces.component.UIComponent

UIViewRoot

- javax.faces.component.UIViewRoot
- JSP Tag: <f:view>
- Property: locale
- UIViewRoot hat kein Rendering
- Alle JSF Komponenten müssen in einem <f:view> Tag geschachtelt sein
- UIViewRoot ist das oberste Element des Komponentenbaumes
- In Restore View Phase wird der Komponenten-Baum wiederhergestellt

HTMLForm

- javax.faces.component.html.HTMLForm
- Family: javax.faces.component.UIForm
- JSP Tag: <h:form>
- Stellt ein Formular dar
- Alle javax.faces.component.UIInput Komponenten sollten in einem <h:form> Tag geschachtelt sein
- Eine JSP kann mehrere Forms enthalten
- Ein javax.faces.component.UICommand submittet die Form

HTMLOutputText

- javax.faces.component.html.HTMLOutputText
- Family: javax.faces.component.UIOutput
- JSP Tag: <h:outputText>
- Properties:
  - Value Wert zum Anzeigen
  - Escape (default=„true“)
  - …
- Dient nur zur Ausgabe (ReadOnly)
- Kann mit Property eines Models verbunden sein
- Eine Formatierung an Text kann nicht vorgenommen werden:
  - Für Formatierung: <h:outputFormat>

HTMLInputText

- javax.faces.component.html.HTMLInputText
- Family: javax.faces.component.UIInput
- JSP Tag: <h:inputText>
- Properties:
  - Value Wert zum Anzeigen/Bearbeiten
  - required (default=„false“)
  - …
- Generiert ein <input type=„text“> in HTML
- Sollte in einer <h:form> geschachtelt sein
- Registirierung von javax.faces.event.ValueChangeListener möglich
- Implementiert javax.faces.component.EditableValueHolder

HTMLCommandLink

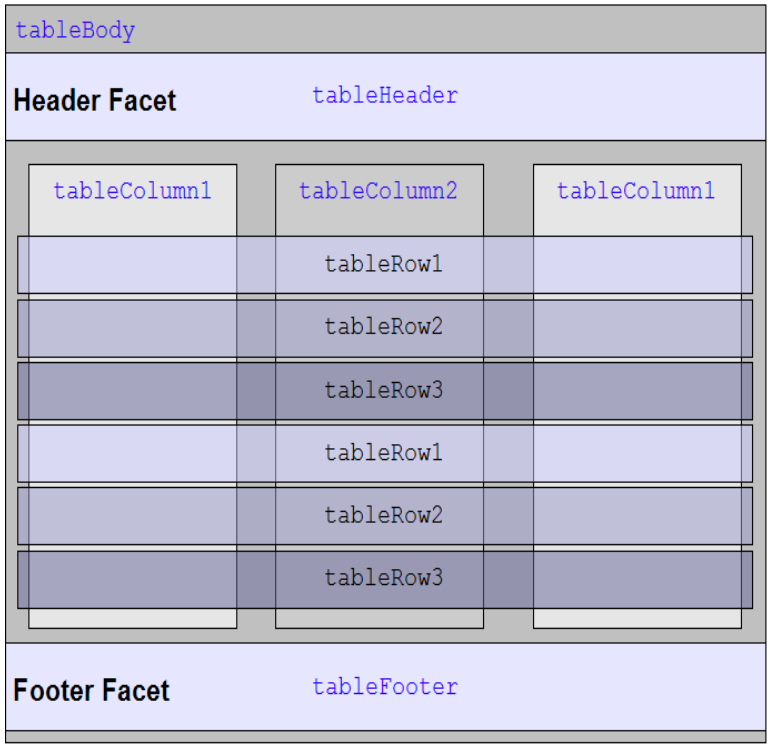
- javax.faces.component.html.HTMLCommandLink
- Family: javax.faces.component.UICommand
- JSP Tag: <h:commandLink>
- Properties:
  - action für Navigation (und Business Logic)
  - actionListener für Business Logic
  - immediate (default=„false“)
  - …
- Generiert ein <a href=„#“ onClick=„…“> in HTML
- Sollte in einer <h:form> geschachtelt sein, um die Form zu submitten
- Registrierung von ActionListener möglich

HtmlPanelGrid

- javax.faces.component.html.HtmlPanelGrid
- Family: javax.faces.component.UICommand
- JSP Tag: <h:panelGrid>
- Properties:
  - column Anzahl Spalten
  - ...
- Generiert eine <table> in HTML

HtmlPanelGrid

```
<h:panelGrid columns="3"
     styleClass="tableBody"
     headerClass="tableHeader"
     footerClass="tableFooter"
     columnClasses="tableColumn1, tableColumn2"
     rowClasses="tableRow1, tableRow2, tableRow3"
>
```



Orientierung

- …
- Einführendes Beispiel
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- Facelets
- Conversion Modell
- …

Backing Beans

- Backing Beans
  - JavaBeans, keine spezielle API (getter/setter)
  - Zugriff meist mit der JSF EL: #{bean.eigenschaft}
  - Repräsentiert FormBean + Action (vgl. Struts)
  - Zugriff auf Backend (EJB, DAO, JDO, …)
- Weitere optionale Methoden:
  - Validation Methode
  - Action event handler Methode
  - Value change event handler Methode
  - Navigation handling Methode (Action Methode)

Backing Bean/Managed Bean

- Unterschiede von Managed Bean zu Backing Bean:
  - Managed Bean wird von JSF bei Bedarf instanziiert und initialisiert
  - Managed Bean wird von JSF in den konfigurierten Scope gelegt
  - Managed Bean kann auch nur über Konfiguration erzeugt werden, keine Java-Klasse
  - Konfiguration von Managed Bean erfolgt in faces-config.xml

Konfiguration Managed Bean in faces-config.xml

```
<faces-config>
    …
    <managed-bean>
        <manged-bean-name>
            registrationBean
        </managed-bean-name>
        <manged-bean-class>
            de.mathema.web.jsf.registration.RegistrationBean
        </managed-bean-class>
        <managed-bean-scope>
            session
        </managed-bean-scop>
    </managed-bean>
    …
</faces-config>
```

Orientierung

- …
- Request Processing Lifecycle
- Konfiguration (faces-config.xml)
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- Facelets
- Conversion Modell
- Internationalisation (i18n)
- …

Backing Bean – Binding Arten

- Value Binding
- Method Binding
- Component Binding
- JSF Expression Language (EL) wird für Binding der Managed Bean benutzt
- JSF Expression Language:
  - Ähnlich wie JSTL 1.0 EL (statt $ ein #)
  - Format:
    - #{x.y}
    - #{x.y[3].z}
    - #{x[‚y‘].z}

Value Binding

- JSP

```
<h:outputText value="#{UserBean.name}" />
```

- Bean Deklaration in faces-config.xml:

```
<managed-bean>
    <manged-bean-name>registrationBean</managed-bean-name>
    <manged-property>
        <property-name>name</property-name>
        <value>someName</value>
    </managed-property>
</managed-bean>
```

Component Binding

- JSP

```
<h:outputText id="err"
    binding="#{registrationBean.errorComp}" />
```

- Leichter Zugriff auf die Komponente und ihre Werte in BB
- Dynamische Änderung von Attributen der Komponente möglich
- Eintrag in die RegistrationBean:

```
HtmlOutputText errorcomp;
// getter und setter
```

- Manipulation in Methode der RegistrationBean:

```
errorComp.setValue("so nicht !!!");
errorComp.setStyle("color: red");
```

Implizite Objekte (1/2)

- JSP

```
<h:outputText id="version" value="#{initParam.versionNo}" />
```

- Eintrag in die web.xml

```
<web-app>
    …
    <context-param>
        <param-name>versionNo</param-name>
        <param-value>1.05</param-value>
    </context-param>
    …
</web-app>
```

Implizite Objekte (1/2)

- Implizite Objekte:
  - applicationScope
  - Cookie
  - facesContext
  - Header
  - headerValues
  - initParam
  - Param
  - paramValues
  - requestScope
  - sessionScope
  - tree

Method Binding

- Aufruf beliebiger öffentlicher Methoden von Backing Beans
- Anwendungen
  - Validation Methode
  - Action event handler Methode
  - Value change event handler Methode
  - Navigation handling Methode (Action methods)

Method Binding – Validation Methode

- Validation Deklaration in JSP

```
<h:inputText id="email"
    value="#{registrationBean.validateEmail}"
    validator="#{validationBean.validateEmail}"
/>
```

- Validation Methode in RegistrationBean

```
public void validateEmail(
    FacesContext ctx, UIInput toValidate, Object value
) {
    String email = (String) value;
    if (email.indexOf('@') == -1) {
        toValidate.setValid(false);
        String message = "Keine gültige Emailadresse";
        ctx.addMessage(toValidate.getClientId(ctx),
            FacesMessage(message, "keine Details"));
    } else { toValidate.setValid(true); }
}
```

Method Binding – Action Event Handler Methode

- ActionListener anmelden in JSP

```
<h:commandLink id="de"
    action="#{registrationBean.doSomething}"
    actionListener="#{registrationBean.chooseLocaleFromLink}">
        <h:outputText value="#bundle.german" />
</h:commandLink >
```

- ActionListener Methode in RegistrationBean

```
public void chooseLocaleFromLink( ActionEvent event) {
    String current  = event.getComponent().getId();
    FacesContext ctx = FacesContext.getCurrentInstance();
    ctx.getViewRoot().setLocale( new Locale( current ) );
}
```

- Nur für UI-Komponente, die ActionSource implementiert: UICommand und UIButton

Method Binding – Value Change Event Handler

- ValueChangeListener anmelden in JSP

```
<h:selectOneListbox id="de"
    value="#{registrationBean.currentGroup}"
    valueChangeListener="#{registrationBean.groupChanged}">
        <f:selectItems value="#{registrationBean.groups}" />
</ h:selectOneListbox >
```

- ValueChangeListener Methode in RegistrationBean

```
public void groupChanged(ValueChengeEvent event) {
    if( event.getNewValue() != null ) {
        log.debug(event.getOldValue() + ">>"
            + event.getNewValue() );
    }
}
```

- Nur für UI-Komponente, die EditableValueHolder implementiert: UIInput

Method Binding – Navigation Methode

- Navigation Action Deklaration in JSP

```
<h:commandLink id="submitForm"
    action="#{registrationBean.registerUser}" >
        <h:outputText value="#{bundle.submit}" />
</h:commandLink>
```

- Navigation Action Methode in RegistrationBean

```
public String registerUser () {
    FacesContext ctx = FacesContext.getCurrentInstance();
    try {
        // do something
    } catch ( Exception e) {
        return "success";
    }
}
```

Orientierung

- …
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- Facelets
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- …

Facelets

- Templating für JSF (ähnlich Tapestry)
- Umsetztung erfolgt durch ViewHandler
- Standardkonform
- https://facelets.dev.java.net/
- Alternativen für Templating:
  - Tiles
  - Clay (Shale)
- XHTML als Dateiformat
- EL überall möglich
- Vermischung von HTML und JSF-Tags
- Eigene Taglib, nicht JSP konform (aber einfacher)

Vorraussetzungen

- JavaServer Faces RI (1.1 oder 1.2) oder Apache MyFaces
- JavaServer Faces 1.2 API
- EL API
- EL RI
- XML SAX

Templating mit Facelets

- Aufteilung der Seiten in mehrere separate Einheiten
- Parameterübergabe an die ausgelagerten Seitenfragmente
- Seiten in XHTML; Header:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
```

- Einfache Erstellung von eigenen Taglibs
- Einbindung von Funktionsaufrufen in Tags unkompliziert

Beispiel - Template

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets">
   <head>
      <title>Facelets: Number Guess Tutorial</title>
   </head>
   <body>
      <h1>
         <ui:insert name="title">Default Title</ui:insert>
      </h1>
      <ui:insert name="body">Default Body</ui:insert>
      <ui:insert src="/facelets/footer.xhtml" />
   </body>
</html>
```

externer Inhalt

Platzhalter für Bereiche

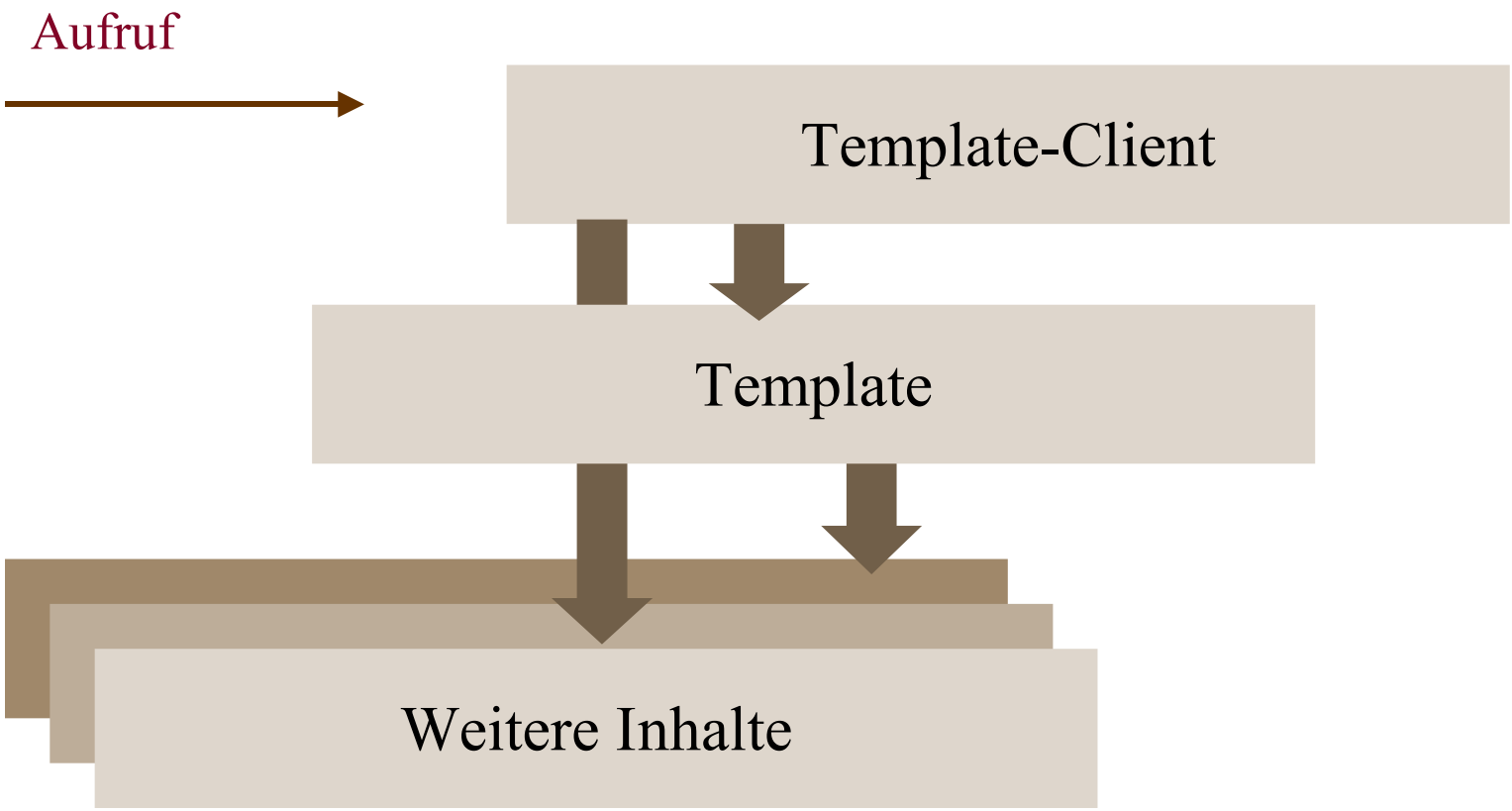
Beispiel – Template-Client

```
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:ui="http://java.sun.com/jsf/facelets„
   xmlns:h="http://java.sun.com/jsf/html">
      <body>
      This text above will not be displayed.
         <ui:composition template="/template.xhtml">
            This text will not be displayed.
            <ui:define name="title">
               Huhu wie gehst so
            </ui:define>
            This text will also not be displayed.
            <ui:define name="body">
               <h:form id="helloForm">
                  <h:inputText/>
                  <h:commandButton/>
               </h:form>
            </ui:define>
            This text will not be displayed.
        </ui:composition>
         This text below will also not be displayed.
      </body>
</html>
```

Template-Angabe

Bereiche

Templating mit Facelets



Facelets - Tags

- <ui:insert>
- <ui:composition>
- <ui:define>
- <ui:include>
- <ui:param>
- <ui:debug>

Konfiguration

- web.xml

```
<context-param>
   <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
   <param-value>.xhtml</param-value>
</context-param>
```

- faces-config.xml

```
<faces-config>
   <application>
      <view-handler>
         com.sun.facelets.FaceletViewHandler
      </view-handler>
   </application>
</faces-config>
```

Taglib

- Konfiguration eigener Taglibs in der web.xml

```
<context-param>
   <param-name>facelets.LIBRARIES</param-name>
   <param-value>
      /META-INF/tomahawk.taglib.xml;/META-INF/sandbox.taglib.xml;
      /taglibs/meine-eigene-taglib.taglib.xml
   </param-value>
</context-param>
```

Taglib – Beispiel Komponente

```
<!DOCTYPE facelet-taglib PUBLIC
      "-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"
      "facelet-taglib_1_0.dtd">
   <facelet-taglib>
      <namespace>http://myfaces.apache.org/tomahawk</namespace>
      <tag>
         <tag-name>commandButton</tag-name>
            <component>
              <component-type>
                 org.apache.myfaces.HtmlCommandButton
              </component-type>
              <renderer-type>
                 org.apache.myfaces.Button
              </renderer-type>
           </component>
       </tag>
   ...
   </facelet-taglib>
```

Taglib – Beispiel Facelets-Funktion

```
<!DOCTYPE facelet-taglib PUBLIC
      "-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"
      "facelet-taglib_1_0.dtd">
   <facelet-taglib>
      <namespace>http://myfaces.apache.org/tomahawk</namespace>
      <function>
               <function-name>
                        logout
               </function-name>
               <function-class>
                        demo.FunctionUtil
               </function-class>
               <function-signature>
                        void invalidateSession()
               </function-signature>
        </function>   ...
   </facelet-taglib>
```

Orientierung

- …
- JSF UI Komponenten
- Backing Bean / Managed Bean
- Binding
- Facelets
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- …

Conversion Model - Überblick

- Dient zur Konvertierung zwischen den Sichten
- Standard Converter
- Plugin-Mechanismus zum Hinzufügen eigener Converter
- Registrierung von Converter in faces-config.xml
  - Standard Converter
  - eigene Converter

Model view	Presentation view
Server	Client /Browser
Object/JavaBean	String/Zeichenkette

Standard Converter

Klasse (Converter)	ID	Class/Typ
BigDecimalConverter	javax.faces.BigDecimal	java.math.BigDecimal
BigIntegerConverter	javax.faces.BigInteger	java.math.BigInteger
BooleanConverter	javax.faces.Boolean	java.lang.Boolean
ByteConverter	javax.faces.Byte	java.lang.Byte
CharacterConverter	javax.faces.Character	java.lang.Character
DateTimeConverter	javax.faces.DateTime	
DoubleConverter	javax.faces.Double	java.lang.Double
FloatConverter	javax.faces.Float	java.lang.Float
IntegerConverter	javax.faces.Integer	java.lang.Integer
LongConverter	javax.faces.Long	java.lang.Long
NumberConverter	javax.faces.Number	
ShortConverter	javax.faces.Short	java.lang.Short

Standard-Converter – Anwendung

- Beispiel für java.util.Date

```
<h:inputText id="birthdate"
    value="#{visitBB.userBB.birthdate}" >
    <f:convertDateTime type="date" />
</h:inputText>
```

bzw.

```
<h:inputText id="birthdate"
    value="#{visitBB.userBB.birthdate}" >
    converter="javax.faces.DateTime" />
```

#{visitBB.userBB.birthdate/Geburtstag ist vom Typ java.util.Date
Vorteil der 1. Lösung: Möglichkeit zur Konfiguration des Converters

Eigener Converter

- Interface javax.faces.convert.Converter
- Methoden:
  - Object getAsObject(FacesContext context,
    UIComponent component,
    String value) throws ConverterException
  - String getAsString(FacesContext context,
    UIComponent component,Object value) throws
    ConverterException
- ConverterException extends RuntimeException
- Beispiel in Anwendung:
  de.mathema.web.jsf.faces.converter.MoneyConvert
  er

Eigener Converter – Registrierung in faces-config.xml

- <converter-for-class>

```
…
<converter>
 <description>
  Convert classes of type MoneyInterface
 </description>
 <display-name>MoneyConverter</display-name>
 <converter-for-class>
  de.mathema.web.struts.weisswurst.backend.bizobj.MoneyInterface
 </converter-for-class>
 <converter-class>
  de.mathema.web.jsf.faces.converter.MoneyConverter
 </converter-class>
</converter>
…
```

Eigener Converter – Registrierung in faces-config.xml

- <converter-id>

```
...
<converter>
 <description>Convert classes type MoneyInterface</description>
 <display-name>MoneyConverter</display-name>
 <converter-id>moneyConverter</converter-id>
 <converter-class>
  de.mathema.web.jsf.faces.converter.MoneyConverter
 </converter-class>
</converter>
...
```

JSP:

```
<h:inputText id="birthdate" … >
    <f:converter converterId="moneyConverter" />
</h:inputText>
```

Eigene Converter – MoneyConverter

- Von String nach Object/Money

```
public Object getAsObject(FacesContext context,
        UIComponent component, String value ) {
    if( value == null ) {
        return null;
    }
    String stringValue = value.toString().trim();
    if( stringValue.length() == 0 ) {
        return null;
    }
    ...
    //Ermittlung des currencyCode/Currency
    String currencyCode = ....
    Currency currency = Currency.getInstance( currencyCode );
    //Ermittlung des Betrag
    String amount = ...
    //Money erzeugen und zurückgeben
    return new Money( currency, amount );
}
```

Eigene Converter – MoneyConverter

- Von Object/Money nach String

```
public String getAsString( FacesContext context,
        UIComponent component, Object value ) {

    if( value == null ) {
        return null;
    }
    if( value instanceof MoneyInterface ) {
        return
            ( (MoneyInterface) value ).toString(
            context.getViewRoot().getLocale() );
    } else {
        return value.toString();
    }
}
```

Orientierung

- …
- Backing Bean / Managed Bean
- Binding
- Facelets
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- JSF 1.2 Neuerungen
- …

Locale

- java.util.Locale legt Sprache, Land und Variante fest
- Locale wird in facesContext.getViewRoot(). setLocale(locale) gespeichert
- Zugriff: facesContext.getViewRoot().getLocale()
- Konfiguration der unterstützten Locales in faces-config.xml

```
<faces-config>
...
 <application>
  <locale-config>
   <default-locale>de_DE</default-locale>
   <supported-locale>en</supported-locale>
   <supported-locale>fr</supported-locale>
  </locale-config>
 </application>
...
</faces-config>
```

Bestimmung der aktiven Locale

- Wiederherstellung des vorhergehenden *View*/UIViewRoot (*Restore View Phase*)
- Holen der Locale und in neuen *View* setzen
- Keine Locale vorhanden, dann Locale aus *Request* holen
- Überprüfen, ob Locale in Konfiguration unterstüzt wird
- Keine unterstütze Locale gefunden, dann Locale aus <default-locale> verwenden
- <default-locale> ist nicht konfiguriert, dann Locale.getDefault()
- (Setzen der Locale in neuen View/UIViewRoot)

Internationalisation (i18n) – Ressourcen/*Properties*

- Ressourcen werden in ResourceBundle/*Property*-Dateien verwaltet
- <f:loadBundle> – Laden eines ResourceBundles
  Attribute:
  - basename (*required*) - Basisname des ResourceBundle
  - var (*required*) - Name unter den das *Bundle* angesprochen werden kann

Beispiel

- JSP

```
<f:loadBundle
    basename="de.mathema....resource.ApplicationResource"
    var="shopResource" />
<h:outputText value="#{shopResource.label_profil_firstname}"/>
```

- ApplicationResource.properties

```
label_profil_id=Benutzer Id
label_profil_userName=Benutzername
label_profil_firstname=Vorname
label_profil_lastname=Nachname
label_profil_phonenumber=Telefonnummer
...
```

Orientierung

- …
- Binding
- Facelets
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- JSF 1.2 Neuerungen
- Erweiterungen
- Ausblick JSF 2.0
- …

Einführung (Fehler-)Meldunngen

- Welche Mechanismen bietet JSF um Hinweise/Fehler
- an den Benutzer weiterzureichen?
- Welche Klassen/Tags sind beteiligt?
- Wie werden (Fehler-) Meldungen erzeugt?
- Wie werden (Fehler-) Meldungen angezeigt?

Orientierung – Validation Model

- Klassen und Methoden für (Fehler-)Meldungen (Messages)
- Taglibunterstützung zur Darstellung

FacesMessages

- javax.faces.application.FacesMessage
- Repräsentiert Hinweis bzw. Fehlermeldung; Konstruktoren:
  - FacesMessage()
  - FacesMessage(FacesMessage.Severity severity,String summary,String detail)
  - FacesMessage(String summary)
  - FacesMessage(summary, java.lang.String detail)
- FacesMessage.Severity: SEVERITY_INFO (*default*), SEVERITY_WARN, SEVERITY_ERROR,SEVERITY_FATAL
- Internationalisierung der Meldung muss selbst vorgenommen werden

Internationalisierung der FacesMessage

Notwendige Schritte (siehe Implementierung in de.mathema.web.jsf.faces.message.MessageFactor)

- Bestimmung der aktuellen Locale
- Laden des ResourceBundle (Application().getMessageBundle())
- Finden von summary durch messageId/*Key* in ResourceBundle
  - Wenn das Finden scheitert, dann in ResourceBundle repräsentiert durch FacesMessage.FACES_MESSAGES suchen
- Finden von detail durch **messageId** + **"_detail"**/*Key* in ResourceBundle
  - Beim Scheitern, wie bei summary

<message-bundle>

- Konfiguration des <message-bundle> in faces-config.xml

```
<faces-config>
 ...
 <application>
  <message-bundle>
   de.mathema.web.jsf.webshop.resource.MessageResource
  </message-bundle>
  <locale-config>
   <default-locale>de_DE</default-locale>
   <supported-locale>en</supported-locale>
  </locale-config>
 </application>
...
</faces-config>
```

FacesMessage an FacesContext weiterreichen

- FacesMessage muss an FacesContext weitergereicht werden, damit darstellbar
- clientId:
  - ID der UIComponent, der die Meldung zugeordnet wird
  - Darf auch null sein, dann globale Meldung

```
String clientId = ...;
FacesMessage message = ...;
FacesContext context = FacesContext.getCurrentInstance();

context.addMessage( clientId, message );
```

Orientierung – Validation Model

- Klassen und Methoden für (Fehler-)Meldungen (Messages)
- Taglibunterstützung zur Darstellung

Taglibunterstützung zur Darstellung

- <h:message>
  - Stellt Meldung für eine spezifische Komponente dar
- <h:messages>
  - Stellt alle Meldungen dar

Beispiel

- <h:messages>, Anzeige aller Fehler

```
<h:messages layout="table" styleClass="error_messages" />
```

- clientId:
  - ID der UIComponent, der die Meldung zugeordnet wird
  - Darf auch null sein, dann globale Meldung

```
<h:inputText
  id="email"
  value="#{visitBB.userBB.user.email}"
  validator="#{visitBB.userBB.validateEmail}">
</h:inputText>
<h:message for="email" styleClass="error_messages" />
```

<h:message> – wichtige Attribute

- for (*required*)
  - ID für Komponente für die die Meldung dargestellt werden soll
- showDetail (*default*: true)
  - Zeigt die *Detail*-Meldung an
- showSummary (*default*: false)
  - Zeigt die *Summary*-Meldung an
- xxxClass
  - *Stylesheet-Class*, xxx steht für info, warn, error, fatal oder style
- xxxStyle
  - *Stylesheet*, xxx steht für info, warn, error, fatal oder nur style

<h:messages> – wichtige Attribute

- globalOnly (*default*: false)
  - Nur Meldungen ohne clientId (= null)
- layout (*default*: list)
  - Darstellung der Meldungen als table (HTML-Tabelle) oder list (HTML-Liste)
- showDetail (*default*: false)
  - Zeigt die *Detail*-Meldung an
- showSummary (*default*: true)
  - Zeigt die *Summary*-Meldung an
- xxxClass
  - *Stylesheet-Class*, xxx steht für info, warn, error, fatal oder style
- xxxStyle
  - *Stylesheet*, xxx steht für info, warn, error, fatal oder nur style

Orientierung

- …
- Binding
- Facelets
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- Erweiterungen
- Ausblick JSF 2.0
- Ressourcen

Validation Model – Überblick

- Arten der Validierung:
  - *Input*-UI-Komponente (Tag) durch Attribut required="true"
  - Standard-*Validator*en
  - Validierung durch Methode
  - Erweiterbar durch eigene *Validator*en
- Anzeige der Validierungsmeldungen durch <h:messages> (alle) bzw. <h:message>

Orientierung – Validation Model

- Validierung – *Required*
- Standard-*Validator*
- Validierung durch Methode
- Eigener *Validator*

Validierung - Required

- Validierungsfehler wird durch <h:message> dargestellt

```
<h:inputText
    id="birthdate"
    value="#{visitBB.userBB.birthdate}"
    required="true"
>
  <f:convertDateTime type="date" />
</h:inputText>
<h:message
    for="birthdate"
    styleClass="error_messages" />
```

Orientierung – Validation Model

- Validierung – *Required*
- Standard-*Validator*
- Validierung durch Methode
- Eigener *Validator*

Standard-Validator

- Verwendung

```
<h:inputText value="#{productViewItem.count}" >
    <f:validateLongRange minimum="0" maximum="99"/>
</h:inputText>
```

count muss zwischen 0 und 99 liegen

Validator Class	Tag	ID
DoubleRangeValidator	<validateDoubleRange>	javax.faces.DoubleRange
LengthValidator	<validateLength>	javax.faces.Length
LongRangeValidator	<validateLongRange>	javax.faces.LongRange

Orientierung – Validation Model

- Validierung – *Required*
- Standard-*Validator*
- Validierung durch Methode
- Eigener *Validator*

Validierung durch Methode - Implementierung

- Methode muss Parameter-Signatur wie im folgenden Beispiel aufweisen: Implementierung in UserBB

```
public void validateEmail(FacesContext context,
        UIComponent uicomponent, Object value ) {
    if( value == null || !EmailValidator.getInstance()
        .isValid( value.toString())) {
    throw new ValidatorException(
        new FacesMessage( "Keine gültige Email") );
    }
}
```

Verwendung:

```
<h:inputText id="email„ value="#{visitBB.userBB.user.email}"
    validator="#{visitBB.userBB.validateEmail}"/>
<h:message for="email" styleClass="error_messages" />
```

Orientierung – Validation Model

- Validierung – *Required*
- Standard-*Validator*
- Validierung durch Methode
- Eigener *Validator*

Eigener Validator

- Interface javax.faces.validator.Validator
- Methode:
  - void validate(FacesContext context, UIComponent component, Object value) throws ValidatorException
- Konstante:
  - static final String NOT_IN_RANGE_MESSAGE_ID
- Beispiel in Anwendung: de.mathema...validator.DateSmallerThanCurrentDateValidator

Eigener Validator – Registrierung in faces-config.xml

```
...
<validator>
  <description>
    Validate that the date is smaller than the current date
  </description>
  <validator-id>
    de.mathema...validator.DateSmallerThanCurrentDateValidator
  </validator-id>
  <validator-class>
    de.mathema...validator.DateSmallerThanCurrentDateValidator
  </validator-class>
</validator>
...
```

Eigener Validator - Anwendung

```
<h:inputText id="birthdate" value="#{visitBB.userBB.birthdate}">
  <f:validator
    validatorId=
  "de.mathema...validator.DateSmallerThanCurrentDateValidator"/>
  <f:convertDateTime type="date" />
</h:inputText>
```

#{visitBB.userBB.birthdate}/Geburtstag ist vom Typ java.util.Date

Eigener Validator - DateSmallerThanCurrentDateValidator

```
public void validate( FacesContext facesContext,
    UIComponent uicomponent, Object value )
    throws ValidatorException {
  if( value == null ) { return; }
  if( ! ( value instanceof Date ) ) {
      throw new ValidatorException(this.createMessage(
        facesContext,VALUE_IS_NOT_INSTANCEOF_DATE,
        new Object[]{ value } )
      );
  }
  Date date = (Date) value;
  Date currentDate = new Date();
  if( currentDate.before( date ) ) {
      throw new ValidatorException(
        this.createMessage(facesContext,
            DATE_NOT_SMALLER_THAN_CURRENT_DATE, new Object[]{ value } )
        );
  }
}
```

Orientierung

- …
- Facelets
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- JSF 1.2 Neuerungen
- Erweiterungen
- Ausblick JSF 2.0
- Ressourcen

JSF 1.2 Neuerungen

- https://javaserverfaces-spec-public.dev.java.net/servlets/ProjectIssues
- Browser Back Button Problem ist behoben
- Validierung faces-config.xml mit XML Schema anstatt DTD
-
```
<h:input
   requiredMessage="Hallo ich bin required"
   converterMessage="Konvertierung ist fehlgeschlagen"
   validatorMessage="Validierung ist fehlgeschlagen"
/>
```
- Benutzung von Java EE 5 Generics
- Annotations @PostConstruct und @PreDestroy bei ManagedBean
- Annotations @Resource, @EJB, @WebServiceRef, @WebServiceRefs, @PersistenceContext, @PersistenceUnit (und jeweils der Plural-Version)
- Support Java EE 5 enum
- Unified Expression Language (EL) from JSP 2.1.

Orientierung

- …
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- Facelets
- JSF 1.2 Neuerungen
- Erweiterungen
- Ausblick JSF 2.0
- Ressourcen

Orientierung - Erweiterungen

- MyFaces
  - Tomahawk
  - Sandbox
- RichFaces

MyFaces - Tomahawk

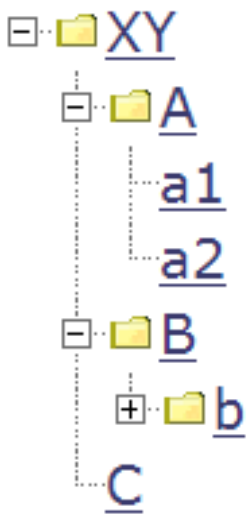
Tomahawk von MyFaces ergänzt JSF um zusätzliche Komponenten

- zusätzliche Validatoren
- *Other Goodies*

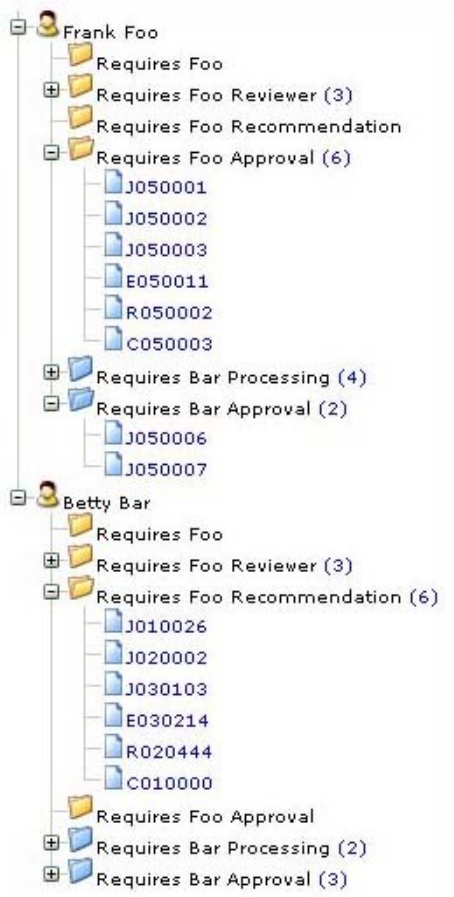
Verwendung auch unter anderen JSF-Implementierungen.

MyFaces – Tomahawk: Tree, Tree2, Tree Table

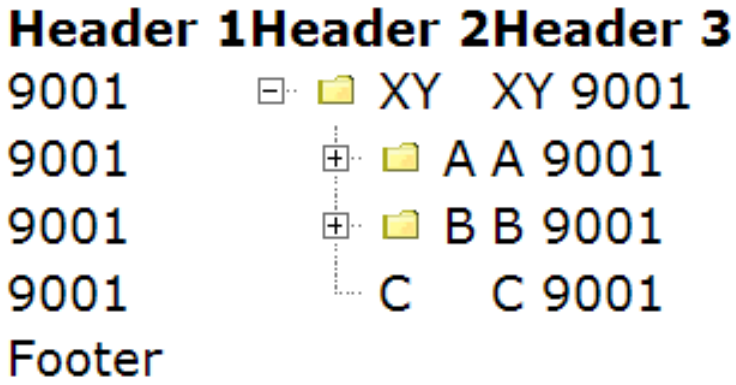
Tree



Tree2



TreeTable



MyFaces – Tomahawk: DataList

Einfache Liste

AUSTRIA, AZERBAIJAN, BAHAMAS, BAHRAIN, BANGLADESH, BARBADOS

Unsortierte Liste

- AUSTRIA
- AZERBAIJAN
- BAHAMAS
- BAHRAIN
- BANGLADESH
- BARBADOS

Nummerierte Liste

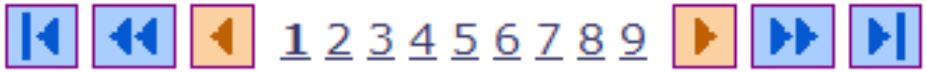
1. AUSTRIA
2. AZERBAIJAN
3. BAHAMAS
4. BAHRAIN
5. BANGLADESH
6. BARBADOS

MyFaces – Tomahawk: DataTable, SortHeader

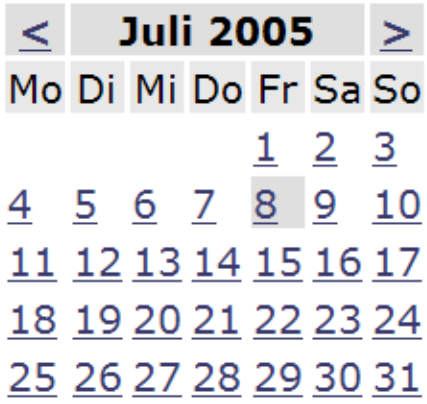
(header table)	
Typ ↑	Farbe
car A	red
car B	blue
car C	green
car D	yellow
car E	orange
(footer col1)	(footer col2)
(footer table)	

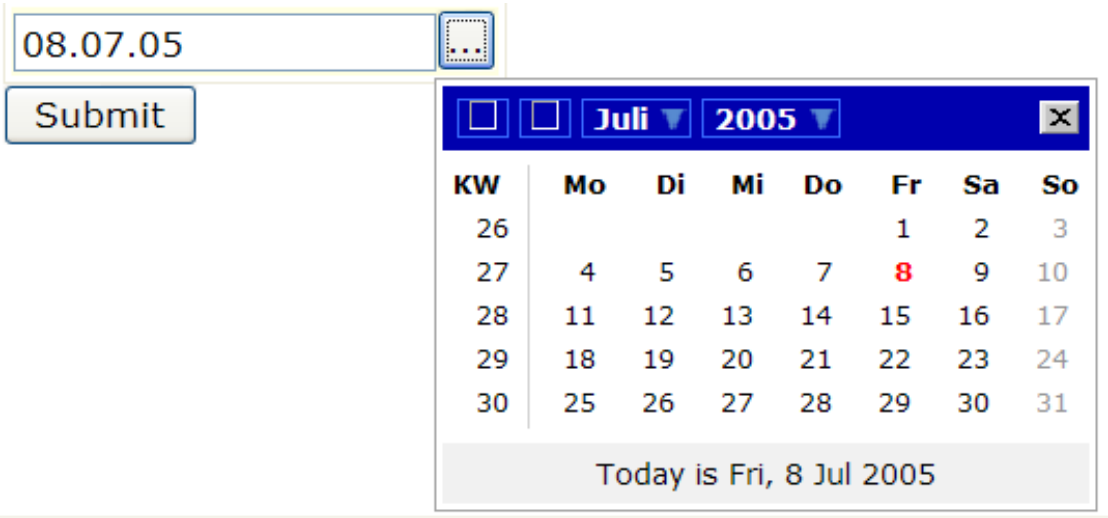
MyFaces – Tomahawk: DataScroller

	Modell	Farbe
1	Car Type 1	blue
2	Car Type 2	blue
3	Car Type 3	blue
4	Car Type 4	blue
5	Car Type 5	blue
6	Car Type 6	blue
7	Car Type 7	blue
8	Car Type 8	blue
9	Car Type 9	blue
10	Car Type 10	blue



MyFaces – Tomahawk: Calendar





MyFaces – Tomahawk: InputDate

Datumskomponente

Geben Sie ein Datum an 08 Juli 2005

Das Datum ist: Fri Jul 08

Juli 2005

Wk	Mo	Di	Mi	Do	Fr	Sa	So
26					1	2	3
27	4	5	6	7	8	9	10
28	11	12	13	14	15	16	17
29	18	19	20	21	22	23	24
30	25	26	27	28	29	30	31

Today is Fri, 8 Jul 2005

MyFaces – Tomahawk: Newspaper Table

AL ALABAMA	LA LOUISIANA	OH OHIO
AK ALASKA	ME MAINE	OK OKLAHOMA
AZ ARIZONA	MD MARYLAND	OR OREGON
AR ARKANSAS	MA MASSACHUSETTS	PA PENNSYLVANIA
CA CALIFORNIA	MI MICHIGAN	RI RHODE ISLAND
CO COLORADO	MN MINNESOTA	SC SOUTH CAROLINA
CT CONNECTICUT	MS MISSISSIPPI	SD SOUTH DAKOTA
DE DELAWARE	MO MISSOURI	TN TENNESSEE
FL FLORIDA	MT MONTANA	TX TEXAS
GA GEORGIA	NE NEBRASKA	UT UTAH
HI HAWAII	NV NEVADA	VT VERMONT
ID IDAHO	NH NEW HAMPSHIRE	VA VIRGINIA
IL ILLINOIS	NJ NEW JERSEY	WA WASHINGTON
IN INDIANA	NM NEW MEXICO	WV WEST VIRGINIA
IA IOWA	NY NEW YORK	WI WISCONSIN
KS KANSAS	NC NORTH CAROLINA	WY WYOMING
KY KENTUCKY	ND NORTH DAKOTA	

MyFaces – Tomahawk: Popup

MyFaces – Tomahawk: Komponenten

- JSCook Menu
- Tree2
- Newspaper Table
- Alias Bean
- Buffer
- File Upload
- TabbedPane
- Calendar
- Popup
- Javascript Listener
- Date
- Html Editor
- Data List
- Tree
- Tree Table
- Panel Stack
- Style Sheet
- Sort Header
- Data Scroller
- Extended Data Table
- Panel Navigation
- UI Save State
- Columns
- Column

MyFaces – Tomahawk: Validatoren & Other Goodies

Validatoren

- validateCreditCard
- validateUrl
- validateEmail
- validateEqual
- validateRegExpr

Other Goodies

- forceId
- Tiles Support
- J2EE basierte Rolen-Berechtigung
  - enabledOnUserRole
  - visibleOnUserRole
- Darstellung der Input-Komponenten als (nur) Text, nicht als Eingabe-*Widget*
  - displayValueOnly
  - displayValueOnlyStyle
  - displayValueOnlyStyleClass

Orientierung - Erweiterungen

- MyFaces
  - Tomahawk
  - Sandbox
- RichFaces

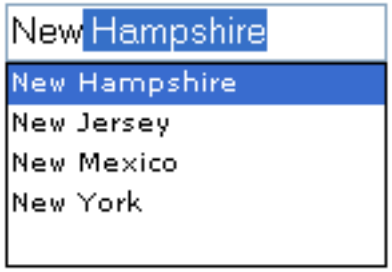
MyFaces - Sandbox

- Brutkasten (*incubator*) von neuen Komponent, Validatoren und *Other Goodies*
- Spielwiese
- Zur Zeit enthaltene Komponenten (Stand August 2008)

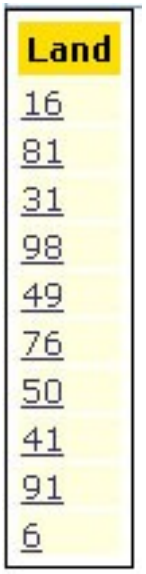
  - Input Suggest Ajax Menu
  - Auto Update DataTable
  - Input Suggest
  - Focus
  - Form
  - SubForm
  - Picklist
  - Dynamic Image
  - FishEye Navigation Menu
  - Excel Export
  - SecurityContext
  - Limit Rendered
  - Rounded Div
  - PasswordStrength

MyFaces – Sandbox: InputSuggest Ajax, Auto Update DataTable, InputSuggest

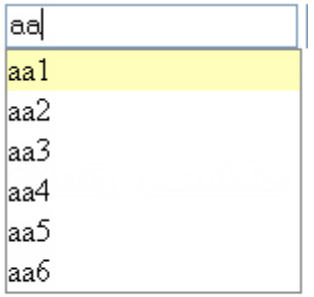
Input Suggest Ajax



Auto Update DataTable



Input Suggest



Orientierung - Erweiterungen

- MyFaces
  - Tomahawk
  - Sandbox
- RichFaces

RichFaces

- 17. Oktober 2007: RichFaces 3.1.2
- Maßgeblich entwickelt von JBoss/Red Hat
- Seit September 2007 ist AJAX4JSF ins Projekt integriert
- Sehr umfangreich, viele Komponenten
- Gute Doku
- Tutorials
- Sehr gute Live-Demo
- Support
- Sehr aktive Community
- **http://labs.jboss.com/jbossrichfaces/**

RichFaces

- AJAX4JSF und RichFaces seit März 2007 von Exadel zu JBoss übergegangen
- AJAX4JSF seit September 2007 in JBoss das RichFaces Projekt eingegliedert worden
- Trennung deutlich durch zwei Namespaces: a4j und rich
- AJAX-Erweiterungsmöglichkeiten für vorhandene Komponenten
- AJAX-isierung von ganzen Seiten
- Neue, eigene Komponenten mit AJAX-Funktionalität
- Schützt den JSF-Entwickler vor dem Zwang, für den Einsatz von AJAX selbst JavaScript schreiben zu müssen
- Skinnable Komponenten

RichFaces - Installation/Konfiguration

- Download von der Webseite: http://labs.jboss.com/jbossrichfaces/downloads/
- Entpacken der drei Jar-Dateien richfaces-api, -impl und -ui in den WEB-INF/lib Ordner der Anwendung
- Konfiguration der web.xml
- Eintragen der Taglib Namespaces in den JSP / XHTML Seiten

  <%@ taglib uri="http://richfaces.org/a4j" prefix="a4j" %>
  <%@ taglib uri="http://richfaces.org/rich" prefix="rich" %>
  oder
  <xmlns:a4j="http://richfaces.org/a4j">
  <xmlns:rich="http://richfaces.org/rich">

RichFaces - Installation/Konfiguration

```
...
    <context-param>
        <param-name>org.richfaces.SKIN</param-name>
        <param-value>blueSky</param-value>
    </context-param>
    <filter>
        <display-name>RichFaces Filter</display-name>
        <filter-name>richfaces</filter-name>
        <filter-class>org.ajax4jsf.Filter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>richfaces</filter-name>
        <servlet-name>Faces Servlet</servlet-name>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>FORWARD</dispatcher>
        <dispatcher>INCLUDE</dispatcher>
    </filter-mapping>
```

RichFaces - Komponenten

- Einsatz von Prototype, Sarissa und script.aculo.us
- Verschiedene Möglichkeiten, AJAX Requests zu senden:
  - a4j:commandButton und a4jCommandLink:onclick
  - a4j:poll sendet in regelmäßigen Abständen
  - a4j:support erweitert andere JSF-Komponenten, Event konfigurierbar
- Was gesendet wird, einschränkbar durch: a4j:region
- Was neu gerendert wird, mit dem Attribut reRender

RichFaces – Komponenten: Einsatz

- Beispiel: **rich:suggestionbox**
- Standardparameter:
  - **for** – zugehöriges Feld
  - **var** – Laufvariable für die Vorschläge
  - **suggestionAction**


```
<h:inputText required="true" id="name" value="#{person.name}" />
<rich:suggestionbox for="name" var="suggest"
    suggestionAction="#{nameController.suggestNames}"
    width="100" height="60">
    <h:column>
        #{suggest}
    </h:column>
</rich:suggestionbox>
```

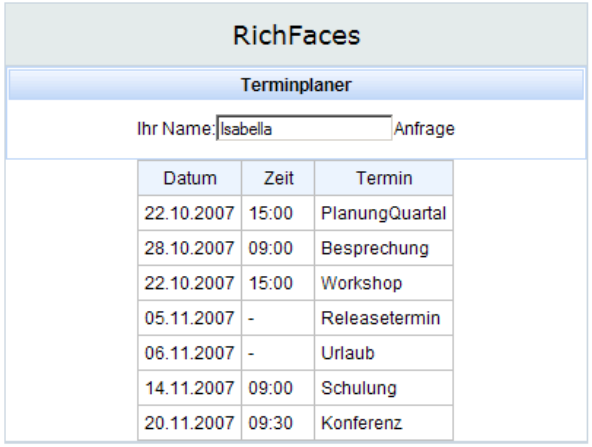
RichFaces – Komponenten: Einsatz

- Beispiel: **rich:suggestionbox**
- suggestionAction: Eingabeparameter Object; Ergebnistyp verarbeitbar für DataTable

```
public List<String> suggestNames(Object input) {
    List<String> nameErg = new ArrayList<String>();
    for(String tmp: getNameList()) {
        if (tmp.startsWith((String)input)) {
            nameErg.add(tmp);
        }
    }
    return nameErg;
}
```

RichFaces – Komponenten: Einsatz

- Beispiel: **a4j:support**
- Standardparameter:
  - **event** – JavaScript Event, bei dem der Aufruf erfolgen soll
  - **reRender** – Id (oder Liste von Ids) von Komponenten, die nach Aufruf neu gerendert werden sollen
  - **action** – ActionBinding




```
<h:inputText id="name" value="#{terminCalendarMgr.name}" >
    <a4j:support event="onblur" reRender="termine"
        action="#{terminCalendarMgr.readCal}" />
</h:inputText>
```

Orientierung

- …
- Conversion Model
- Internationalisation (i18n)
- (Fehler-)Meldungen (Messages)
- Validation Model
- Facelets
- JSF 1.2 Neuerungen
- Erweiterungen
- Ausblick JSF 2.0
- Ressourcen

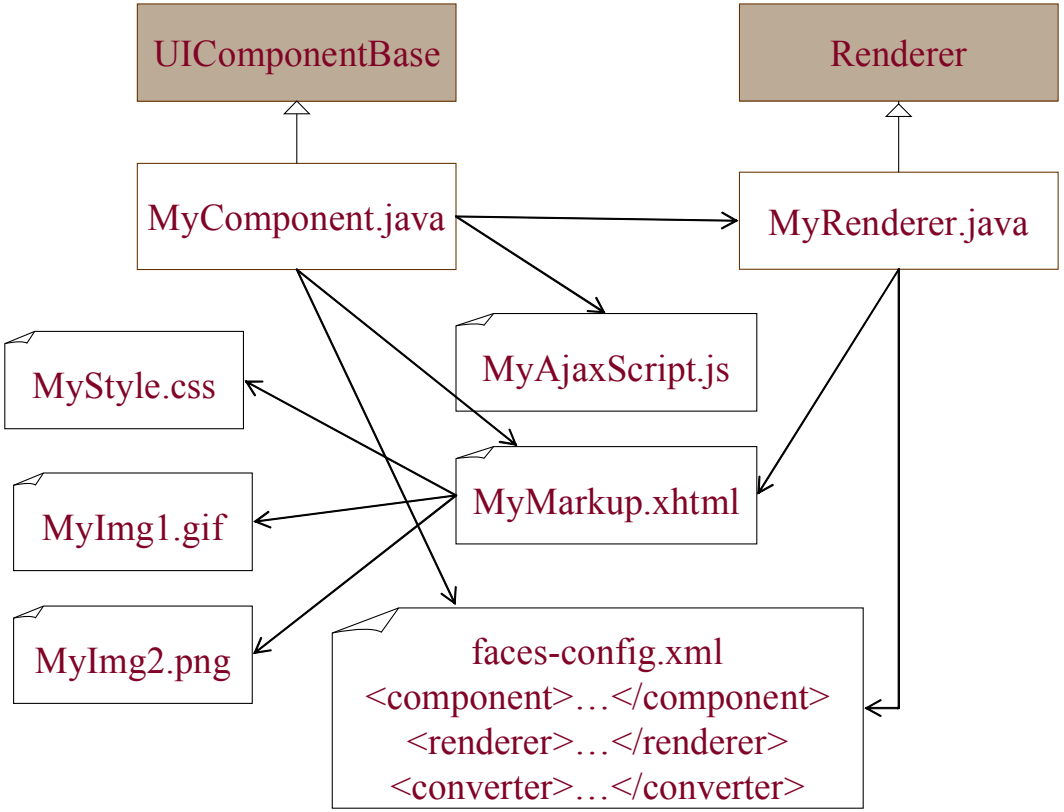
Ziele

- 5 Hauptziele für JSF 2.0
  - Leichtere Entwicklung eigener Komponenten
  - Ajax Support
  - Page Description Language (PDL)
  - Weniger Konfiguration
  - Bessere Kompatibilität zwischen den Komponentenbibliotheken verschiedener Hersteller
- Weitere Ziele u.a.:
  - Bookmarkable URLs
  - Null Deployment-Zeit
  - Traversieren des Komponentenbaumes
  - Scopes
  - Besseres Fehlerreporting

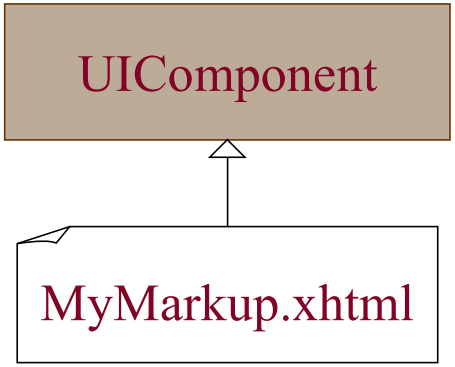
Quelle: Herbstcampus - Ed Burns' Vortrag - Hinter der Maske

Leichtere Entwicklung eigener Komponenten

Von so…



… zu so?



Quelle: Herbstcampus - Ed Burns' Vortrag - Hinter der Maske

Ajax in JSF

- Ein Mechanismus für Ressourcenerhalt
- Partielles Traversieren des Komponentenbaumes
- Partielles Update einer Seite
- Möglichkeit zur „Ajaxifizierung“

In JSF 2.0 Spec

- Ajax-fähige Komponenten

In Komponenten-bibliothek

Quelle: Herbstcampus - Ed Burns' Vortrag - Hinter der Maske

Ressourcen (1/2)

- JavaServer Faces Website http://java.sun.com/j2ee/javaserverfaces/
- Official Standard Implementation for JavaServer(TM) Faces https://javaserverfaces.dev.java.net/
- MyFaces – Open Source Implementierung der Spezifikation http://myfaces.apache.org/
- JSF Community Website http://jsfcentral.com
- Holmes' JavaServer Faces Resources http://www.jamesholmes.com/JavaServerFaces/
- JBoss RichFaces: http://labs.jboss.com/jbossrichfaces/

Ressourcen (2/2)

- Ed Burns Blog http://weblogs.java.net/blog/edburns/
- JSF Spezifikation 2.0 (draft) https://javaserverfaces-spec-public.dev.java.net/proposals/JSF-2_0-draft.html
- JSF Requirements Scratchpad http://wiki.java.net/bin/view/Projects/Jsf2RequirementsScratchpad

Vielen Dank!

Isabella Kneissl

MATHEMA Software GmbH

Pourya Harirbafan