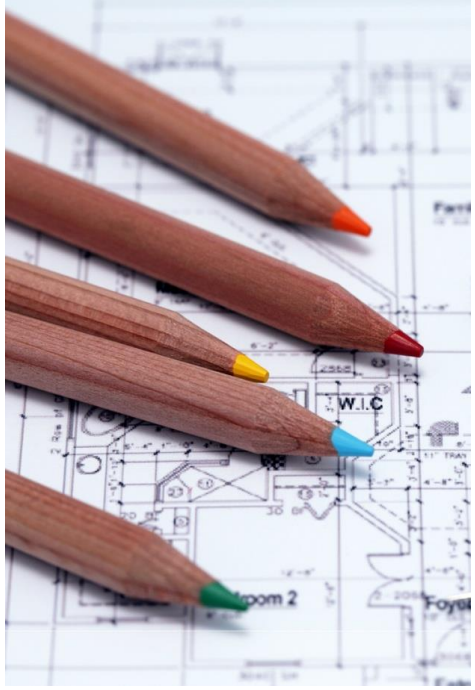


ARS

Java Server Faces

Facelets

Übersicht

- Definition
- Abgrenzung zu Java Server Pages
- Tag Libraries
- Besonderheiten
  - Templating
  - Composite Components
  - Component Aliasing

Definition

- Technologie für (Web-) Oberflächen
- Standard-Technologie für JSF ab Version 2.0
  - Verwendbarkeit in Kombination mit JSF ab Version 1.1
- XML-Datei mit Custom Tags
  - Oberflächenkomponenten
  - Logik
  - Ausgabe (z.B. XHTML)
- Ähnlichkeit zu Java Server Pages
  - Tag Libraries
    - JSTL
  - Unified Expression Language (`${...}` und `#{...}`)

Definition

```
<?xml version="1.0" encoding="UTF-8" ?>
<html
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Startseite</title>
  </h:head>
  <h:body>
    <h:form id="form1">

      <h1>Willkommen</h1>
      <h:outputText id="lblName" value="#{user.name}"/>

    </h:form>
  </h:body>
</html>
```

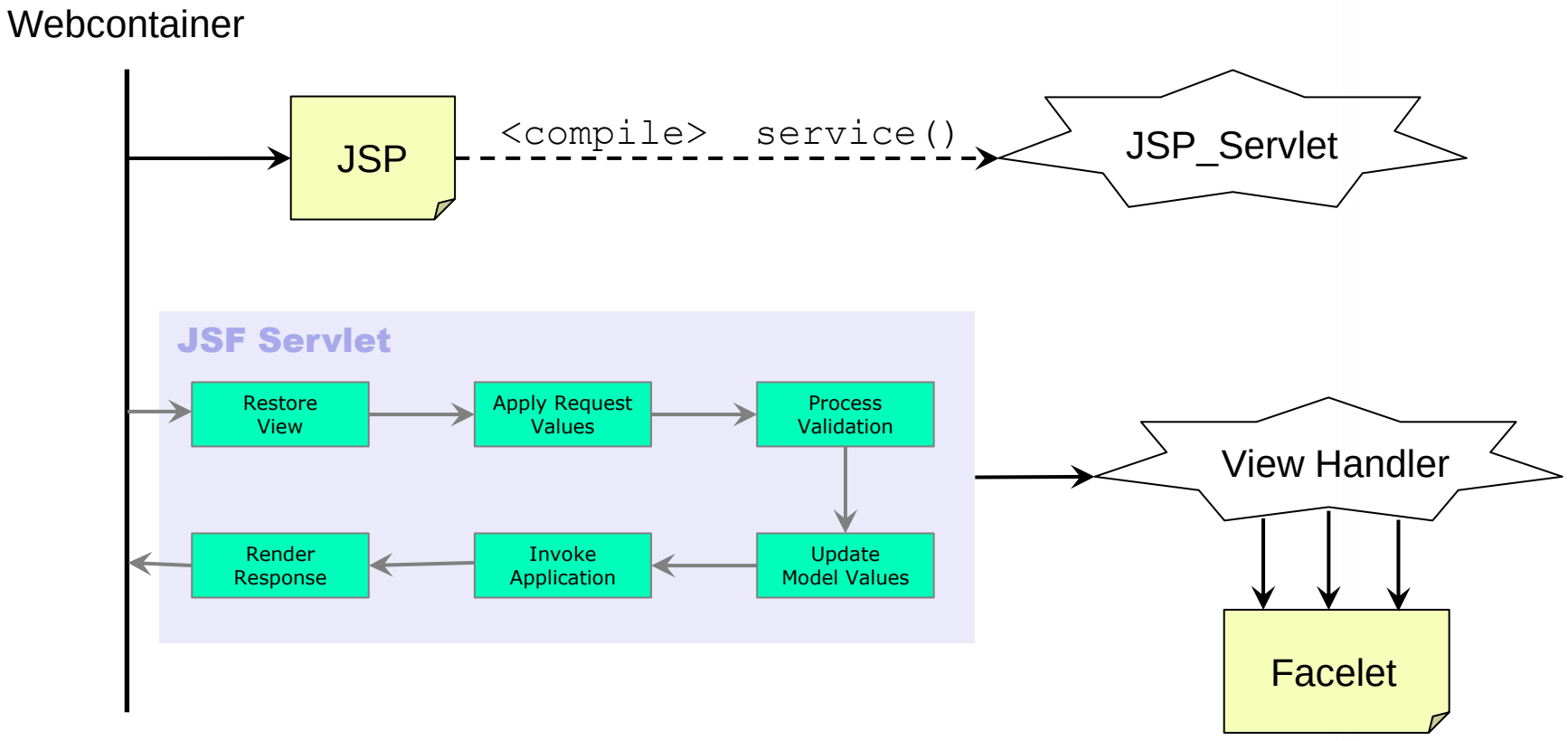
ARS

Abgrenzung zu Java Server Pages

	Java Server Pages	Facelets
Zweck	Erzeugen von statischen und dynamischen Inhalten durch Mix aus HTML und Scriptlets	Verwalten des JSF-Komponentenbaumes
Verarbeitung	Kompilierung in Servlet, einfache, zeilenweise Ausführung der Anweisungen	3-fache Verarbeitung jeder Komponente: Erzeugen, Eingaben verarbeiten, Rendern
JSTL	Vollständige Unterstützung (core, fn, fmt, sql, xml)	Teilweise Unterstützung (core, fn)
Tag Libraries	Tag Library Deskriptor (`META-INF/*.tld`) Tag-Handler als Unterklasse von `javax.servlet.jsp.tagext.SimpleTagSupport` oder als Tag File	Facelet Tag Library Deskriptor (`META-INF/*.taglib.xml`) Tag-Handler als Unterklasse von `javax.faces.view.facelets.TagHandler` oder als Tag File („Composite Component“)

Abgrenzung zu Java Server Pages

Keine Erweiterung des Webcontainers



Tag Libraries

In JSF verfügbare Bibliotheken

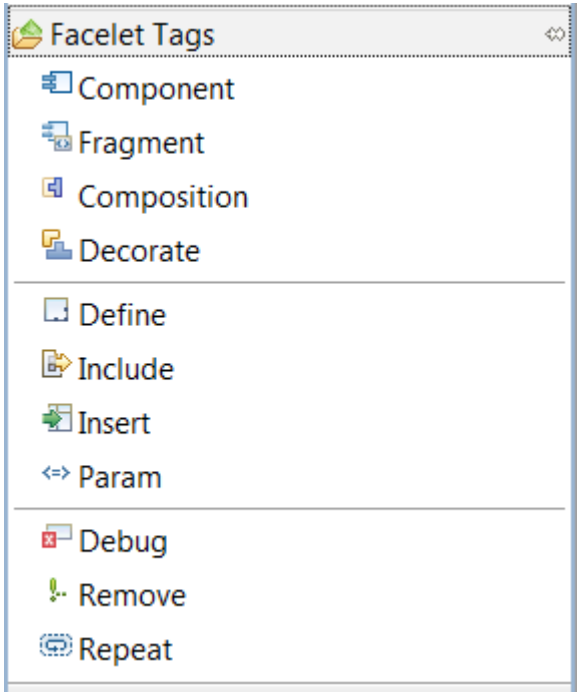
- Core Tags
  - Definition von Views
  - Konfiguration von UI-Komponenten (Konverter, Validatoren, …)
- HTML-Tags (siehe Appendix)
  - Bereitstellung der gängigen HTML-Komponenten
- Facelet-Tags (Templating)
- Composite Components

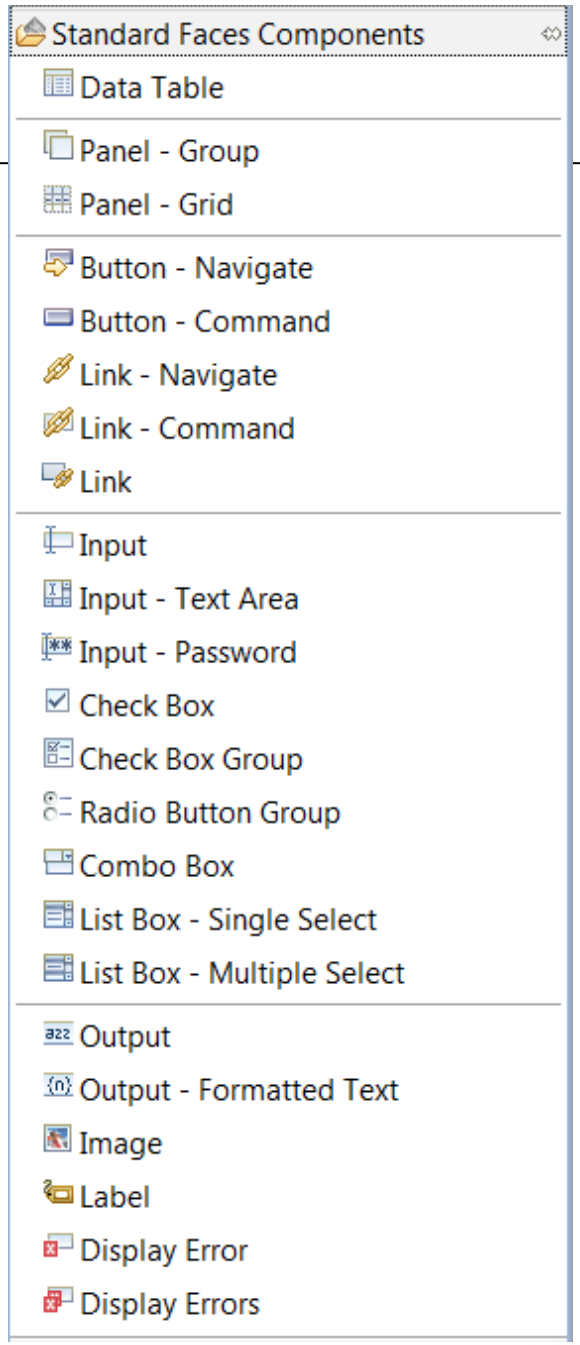
Verwendung

```
<html
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:composite="http://java.sun.com/jsf/composite">
```

Tag Libraries

In JSF verfügbare Bibliotheken





Tag Libraries

Attribute einer JSF-Komponente

- Durch die Komponente vorgegeben
- Allgemeine Attribute für alle JSF-Komponenten
  - `id`: Eindeutige Komponentenkennung
  - `binding`: Bindung an eine Backing Bean
  - `rendered`: Darstellung wird bei false unterdrückt
  - `styleClass`: Verknüpfung zum Cascading Stylesheet
- Verwendung von den Attributen aus HTML 4.0
  - `accesskey`
  - `alt`
  - `title`

Tag Libraries

Beispiel

```
<h:form>
  <h:outputText value="Vorname:" />
  <h:inputText id="firstname" value="#{booking.debitor.firstname}" />
  <h:outputText value="Nachname:" />
  <h:inputText id="lastname" value="#{booking.debitor.lastname}" />
  <h:outputText value="Firma:" />
  <h:inputText id="company" value="#{booking.debitor.company}" />
</h:form>
```

Tag Libraries

Facets

- Fragmente als (komplexe) Attribute von JSF-Komponenten
- Verwaltung durch JSF-Komponente selbst
  - Definition von Facets
  - Zeitpunkt der Auswertung
  - Anzahl der Auswertungen

```
<h:dataTable>
  <h:column>
    <f:facet name="header">
      <h:outputText value="Link zu Terminen" />
    </f:facet>
    <h:commandLink action="#{trainingTypeAction.execute}"/>
  </h:column>
</h:dataTable>
```

Besonderheiten

Templating

- Facelet Tag Library
  - Definition von Templates mit Platzhaltern
  - Verwendung von Templates mit konkreten Belegungen

```
<?xml version="1.0" encoding="UTF-8"?>
<f:view 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">
  <html>
    <h:head>
      <ui:insert name="header" />
    </h:head>
    <h:body>
      <ui:insert name="body />
    </h:body>
  </html>
</f:view>
```

(Template `template.xhtml`)

Besonderheiten

Templating

```
<?xml version="1.0" encoding="UTF-8"?>
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    template="/template.xhtml">

  <ui:define name="header">
    <meta name="description" content="Schulungen" />
    <title>Schulungen</title>
  </ui:define>

  <ui:define name="body">
    <h1>Schulungen:</h1>
    ...
  </ui:define>

</ui:composition>
```

Besonderheiten

Composite Components

- Gruppierung mehrerer UI-Komponenten zu einer neuen
  - Erfüllen einer bestimmten Funktion
  - Wiederverwendbarkeit
- Erstellung über Facelet
  - Keine Java-Kenntnisse notwendig

Besonderheiten

Composite Components

```
<?xml version="1.0" encoding="UTF-8" ?>
<html xmlns:composite="http://java.sun.com/jsf/composite"
    xmlns:h="http://java.sun.com/jsf/html">
  <composite:interface>
    <composite:attribute name="value" required="true"/>
  </composite:interface>
  <composite:implementation>
    <h1>
      <h:outputText value="#{cc.attrs.value}"/>
    </h1>
  </composite:implementation>
</html>
```

(/resources/html/title.xhtml)

```
<?xml version="1.0" encoding="UTF-8" ?>
<html xmlns:html="http://java.sun.com/jsf/composite/html">
  <html:title value="Schulungen" />
</html>
```

Besonderheiten

Component Aliasing

- Notation des Komponentenbaumes in HTML
- Mapping der zugehörigen JSF-Komponenten und deren Attribute über HTML-Attribute
- Statische Darstellung im Browser / Webdesign-Tool möglich
- Anwendungsfall: Vom Webdesigner gestaltete Seite wird vom Entwickler in dynamische Webapplikation überführt.

```
<h:form>
  <h:inputText value="..."/>
</h:form>
```

```
<form jsfc="h:form">
  <input type="text" jsfc="h:inputText" value="..." />
</form>
```

Kontrollfragen

- Was sind Facelets und worin unterscheiden sie sich von JSPs?
- Wofür werden Composite Components verwendet?

Appendix – HTML Tags

HTML Tags

Formulare

- Tag `<h:form>`
- Serverseitige Komponente verwaltet weitere Elemente
  - z.B. Eingabefelder, Buttons usw.
- Unterstützt nur `POST` als Anfragetyp
- Definiert kein `action`-Attribut
  - Navigation durch Buttons oder Links definiert

```
<h:form>
   […]
   <h:commandLink action="categories.start">
     <h:outputText value="#{msg.categories}" />
   </h:commandLink>
   […]
</h:form>
```

HTML Tags – Eingabekomponenten

Eingabefelder

- Tags `<h:inputText>`, `<h:inputSecret>`, `<h:inputTextarea>`
- Erforderliche Eingabe durch das Attribut `required="true"`
- Bindung an eine Managed Bean im Attribut `value`

Beispiele

```
<h:inputText id="firstname"
             value="#{booking.debitor.firstname}" />

<h:inputTextarea id="description" cols="5" rows="5"/>
<h:inputSecret id="password" redisplay="false" required="true"/>
```

Darstellung beim Neuladen unterdrücken

HTML Tags – Ausgabe (1/2)

Ausgabekomponenten

- Tags `<h:outputText>`, `<h:outputFormat>` und `<h:graphicImage>`
- Verwendung von `<h:outputFormat>` analog zu `java.text.MessageFormat`
  - Zusammengesetzte Ausgabe
- Maskieren von Sonderzeichen (Umlauten) mit `escape="true"`
- Bindung an eine Managed Bean im Attribut `value`

HTML Tags – Ausgabe (2/2)

```
<h:outputText value="Bitte entsprechende Felder ausfüllen"
escape="true" />

Bitte entsprechende Felder ausf\&uuml;llen

<h:outputFormat value="Name: {0} {1} {2}" escape="false">
  <f:param value="#{booking.debitor.formOfAddress}" />
  <f:param value="#{booking.debitor.firstname}" />
  <f:param value="#{booking.debitor.lastname}" />
</h:outputFormat>
```

HTML Tags – Steuerungselemente (1/2)

Buttons

- Tag `<h:commandButton>`
- Erzeugt Buttons vom Typ `button`, `submit`, `reset` und `image`
- Angabe des Typs im Attribut `type`
- Attribut `action` definiert Navigation
  - Zeichenkette mit dem Namen des Navigationsfalls
  - Ausdruck mit der Verknüpfung zu einer Methode

```
<h:commandButton action="registration.start"
                 value="Weiter zur Übersicht" />

<h:commandButton action="#{confirmationAction.execute}"
                 value="Anmeldung bestätigen"
                 type="submit" />
```

HTML Tags – Steuerungselemente (2/2)

Links

- Tag `<h:commandLink>`
- Funktionsweise analog zu einem Button
- Zusätzliche Parameter können mit verschickt werden
  - Verwendung des Tags `<f:param>`

```
<h:commandLink>
  <h:outputText value="Obst" />
  <f:param name="categoryName" value="Obst"/>
</h:commandLink>
```

HTML Tags – Auswahlmöglichkeiten (1/2)

h:selectBooleanCheckbox	
h:selectManyCheckbox	
h:selectOneRadio	
h:selectOneListbox	
h:selectManyListbox	
h:selectOneMenu	
h:selectManyMenu	

HTML Tags – Auswahlmöglichkeiten (2/2)

Eigenschaften

- Bindung an eine Managed Bean im Attribut `value`
  - Der ausgewählte Wert wird an die Bean übermittelt
- Darstellung von einem oder mehreren Werten
  - Verwendung des Tags `<f:selectItem>` für einen Wert
  - Verwendung des Tags `<f:selectItems>` für die Werteliste

```
<h:selectOneRadio id="formofaddress">
  <f:selectItem itemLabel="Frau" itemValue="Frau" />
  <f:selectItem itemLabel="Herr" itemValue="Herr" />
</h:selectOneRadio>
```

ARS

HTML Tags – Tabellarische Darstellung (1/2)

Tabellarische Darstellung von Daten

- Tag `<h:dataTable>`
- Verknüpfung des Attributs `value` an die darzustellende Liste
  - Objekt vom Typ `java.util.List` oder ein Array
  - Objekt vom Typ `javax.faces.model.DataModel`
  - Objekt vom Typ `java.sql.ResultSet` oder `javax.sql.RowSet`
  - Objekt vom Typ `javax.servlet.jsp.jstl.sql.ResultSet`
- Jeder Eintrag der Liste bildet eine Zeile
- Aktueller Eintrag wird im Sichtbarkeitsbereich `request` abgelegt
  - Attribut `var` definiert den Namen der Variablen
- Optische Darstellung durch Attribute konfigurierbar

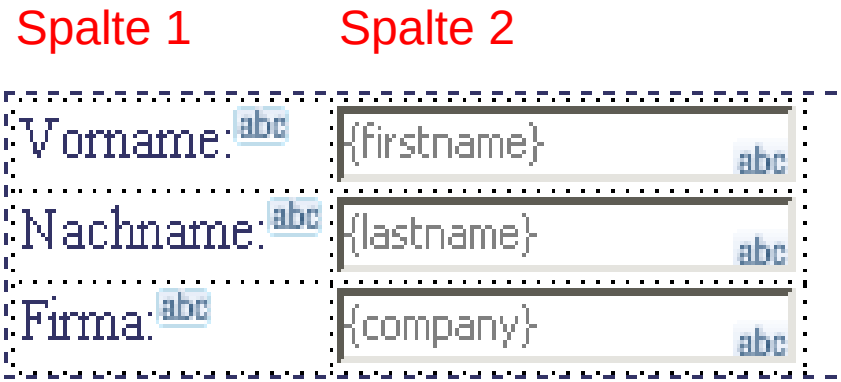
HTML Tags – Tabellarische Darstellung (2/2)

```
<h:dataTable value="#{searchAction.resultTrainingTypes}„ var="trainingType">
  <!-- Spalte mit dem Link zur Übersichtsseite -->
  <h:column>
    <f:facet name="header">
      <h:outputText value="Link zu Terminen" />
    </f:facet>
    <h:commandLink action="#{trainingTypeAction.execute}"> […]
    </h:commandLink>
  </h:column>
  <!-- Spalte mit Beschreibung des Schulungstyps -->
  <h:column>
    <f:facet name="header">
      <h:outputText value="Kursbeschreibung" />
    </f:facet>
    <h:outputText value="#{trainingType.shortDescription}"/>
  </h:column>
</h:dataTable>
```

HTML Tags – Layout (1/2)

Tabellarische Darstellung von Komponenten

- Tag `<h:panelGrid>` und <h:panelGroup>
- Spalten-Angaben im Attribut `columns`
- Umschließt alle Komponenten für die Darstellung
- Darstellung der Komponenten von links nach rechts und von oben nach unten
- Gruppieren von Komponenten zu eine Zelle mit `<h:panelGroup>`



ARS

HTML Tags – Layout (2/2)

```
<h:form>
<h:panelGrid columns="2">

<!-- Erste Zeile, Komponente 1 und 2 -->
<h:outputText value="Vorname:" />
<h:inputText id="firstname" value="#{booking.debitor.firstname}" />
<!-- Zweite Zeile Komponente 3 und 4 -->
<h:outputText value="Nachname:" />
<h:inputText id="lastname" value="#{booking.debitor.lastname}" />
<!-- Dritte Zeile Komponente 5 und 6 -->
<h:outputText value="Firma:" />
<h:inputText id="company" value="#{booking.debitor.company}" />

</h:panelGrid>
</h:form>
```