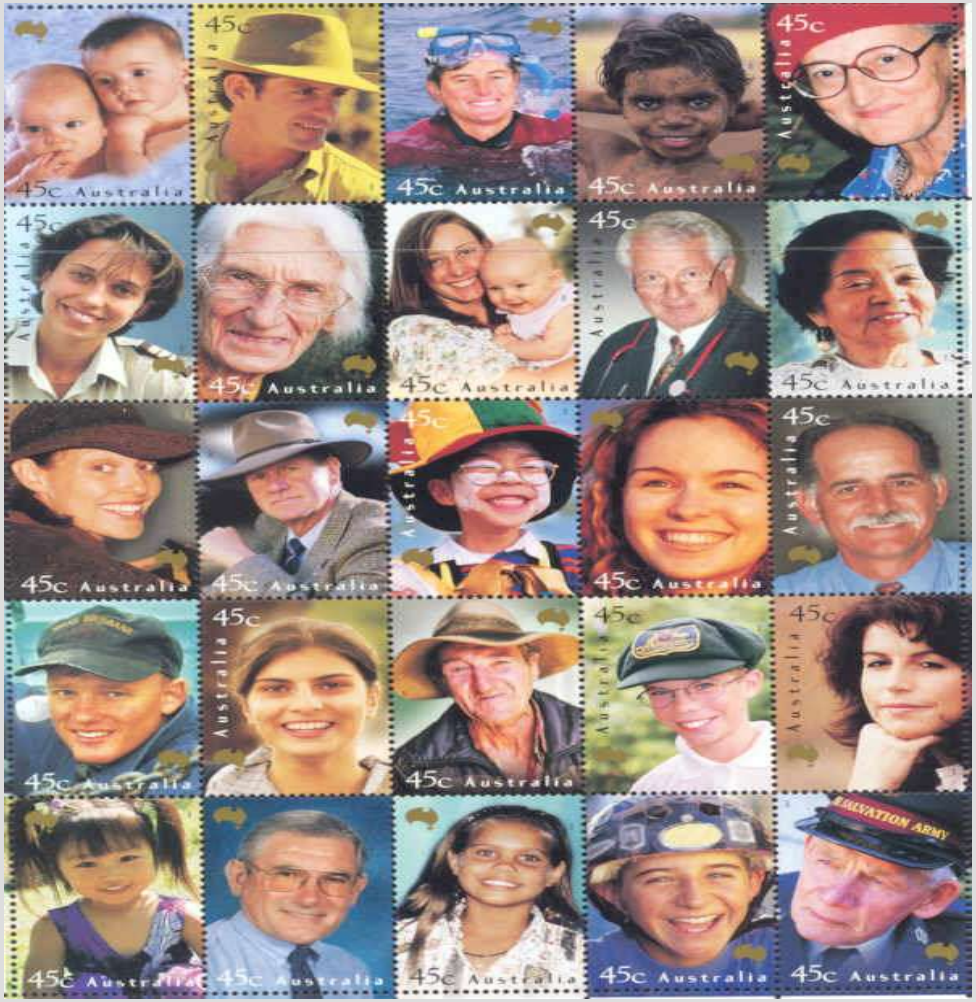
Facelets



Facelets

- Eine alternative View Technologie um JSF Applikationen OHNE JSP zu entwickeln
- Wird unter java.net gehostet
- Open Source, CDDL Lizenz

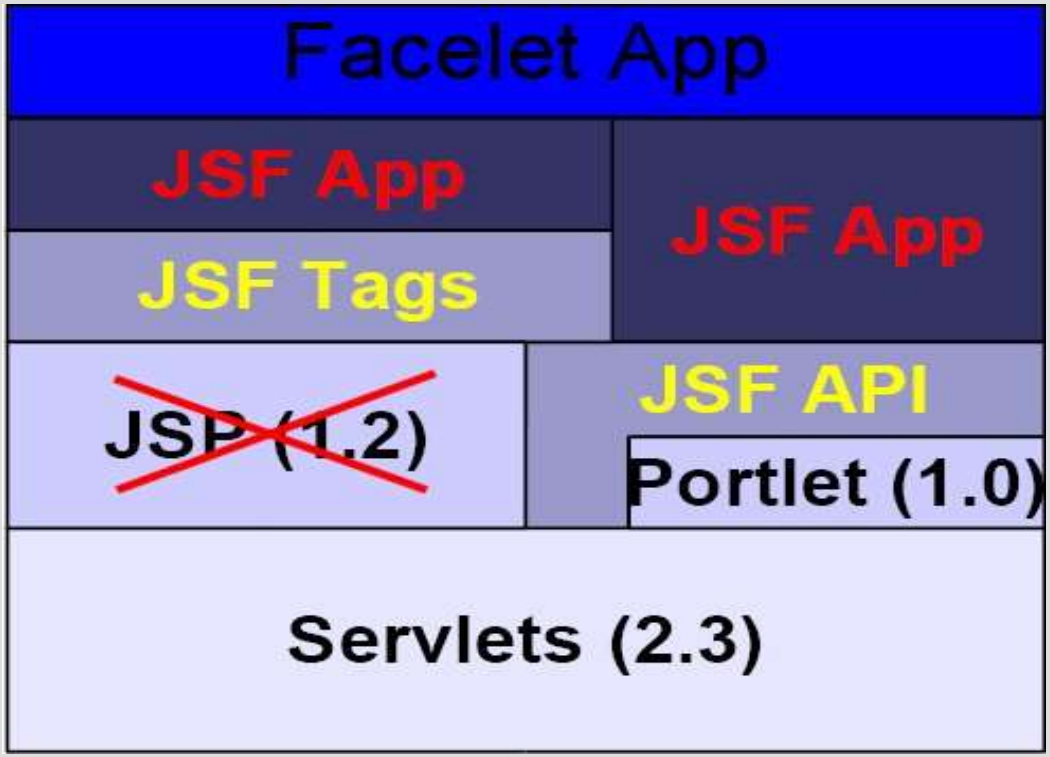
  (COMMON DEVELOPMENT AND DISTRIBUTION LICENSE)
- Von Jacob Hocom entwickelt
  - 24 Jahre alt, arbeitet ab dem Alter von 16 Jahren als Software Consultant.
    Mitglied der JSF Expert Group, Unified EL, Glassfish Entwickler

Status

- Aktuell liegt die Version 1.0d vor
- Es gibt eine ausführliche Dokumentation, Mailingliste + Demoapplikation
- Wird von vom Eclipse Plugin „Exadel Studio“ unterstützt

Architektur

- JSP Layer wird durch Facelets ersetzt!



Bestandteile

- Hauptbestandteile
  - (X)HTML basierte Seitenentwicklung ohne JSP (ähnlich wie Tapestry) durch „Component Aliasing“
  - Templating Unterstützung
- Goodies
  - Bei Fehlermeldungen, genaue Angabe von Zeile/Tag/Attribut
  - Basiert nicht auf einem bestimmten Web Container (JSF 1.2 läuft auf Tomcat 4.x, 5.x mit gewissen Einschränkungen)
  - Volle EL Unterstützung (keine Unterscheidung zwischen $ und #)
  - Vereinfachte Erstellung von Komponenten

Getting Started

- Wenig zusätzliche Konfiguration notwendig
  - facex-config.xml

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

  - web.xml

```
<!-- Use Documents Saved as *.xhtml -->
<context-param>
  <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
  <param-value>.xhtml</param-value>
</context-param>
```

Component Aliasing

- Seitenerstellung völlig ohne JSP
- Es werden normale HTML Tags benutzt
- JSF Komponenten können über das „jsfc“ Attribut eingebunden werden
- Vorteil: HTML zentrierter Ansatz => Designer können Ihr Lieblingstool (z.B. Dreamweaver) benutzen
- „Normaler“ Syntax für JSF Tags weiterhin möglich
  `<h:form ...>`
- Die Seiten werden normalerweise als XHTML abgespeichert

Component Aliasing

- Beispiel

```
<!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:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">
<body>
    <form jsfc="h:form">
        <input jsfc="h:commandButton" type="submit" id="back"
            value="Back" action="success"/>
    </form>
</body>
</html>
```

Component Aliasing

- Zusammenfassung
  - Neuer Ansatz: JSF basierte Anwendung, keine „Softmigration“ bestehender JSP Seiten
  - JSP Features wie `<jsp:include...>` etc. sind nicht verfügbar
  - Es gibt einen eigenen Standard für Taglibs, bestehende Taglibs können nicht eingebunden werden, JSF Komponenten Libraries nur mit Zusatzaufwand, eingeschränkte JSTL Unterstützung

Templating

- Da die JSP Mechanismen für Templates (`<jsp:include>`, tag files) aufgrund der Architektur nicht verfügbar und JSF (so gut wie) keinen Templating Support bietet (muß) Facelets einen eigenen Templating Mechanismus bieten
- Einfach zu benutzen, bietet speziellen Support für JSF
- Templates können beliebig tief ineinander verschachtelt werden
- Über UI Komponenten

Templating

- Beispiel: Template Dokument

```
<!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:ui="http://java.sun.com/jsf/facelets">
<head>
  <style type="text/css">
    body {
       font-family: Verdana, Arial, Helvetica, sans-serif;
    }
  </style>
</head>
<body>
  <h1>
    <ui:insert name="title">Default Title</ui:insert>
  </h1>
  <p>
    <ui:insert name="body">Default Body</ui:insert>
  </p>
</body>
</html>
```

Templating

- Aufrufendes Dokument

```
<!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:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">
<body>
<ui:composition template="/template.xhtml">
  <ui:define name="title">
    I'm thinking of a number from #{NumberBean.min} to #
        {NumberBean.max}.<br/>Can you guess it?
  </ui:define>
  <ui:define name="body">
    <h:form id="helloForm">
        ...
    </h:form>
  </ui:define>
</ui:composition>
</body>
</html>
```

Goodies

- Bei Fehlermeldungen, genaue Angabe von Zeile/Tag/Attribut (Demo)
- EL Literal beliebig wählbar (# oder $)
- Demo: Guess Number Applikation

Zusammenfassung

- „Revolutionäres“ Konzept
- Vorteile
  - JSP Balast wird über Bord geworfen
  - Baut auf Standards JSF und XHTML auf
  - Designer-freundlich, wird von jedem HTML-fähigen Tool unterstützt
  - Eingebautes Templating
  - JSF 1.2 kann mit älteren Container genutzt werden
- Nachteile
  - Softmigration von bestehenden Projekten schwierig
  - Propriätäres Komponentenmodell
  - Bestehende Taglibs können nicht eingebunden werden
  - Zukunft ungewiss