Benutzeroberflächen mit JSF 2.2

Studienarbeit zum FWP-Modul

Aktuelle Technologien zur Entwicklung verteilter Java-Anwendungen

Nikola Topalovic, IB4C

Agenda

- 1991 konnte erstmals die erste Seite in HTML über HTTP übertragen werden
- Webentwicklungstechnologien wurden seitdem immer vielseitiger und komplexer

  → Entwickler mussten unterstützt werden

Bisherige Technologien für Java-Entwickler

- 1997: Servlet-Technologie → HTML-Seiten über Java-Code erzeugen
- 1999: JSP-Technologie → Java-Code über HTML-Tags aufrufen

```
public class BeispielServlet extends HttpServlet
{
	public void doGet(HttpServletRequest req, HttpServletResponse res)
		throws ServletException, IOException
	{
		String name = req.getParameter("name");
		String text = req.getParameter("text");

		res.setContentType("text/html");

		ServletOutputStream out = res.getOutputStream();
		out.println("<html><head><title>");
		out.println(name);
		out.println("</title></head><body>");
		out.println(text);
		out.println("</body></html>");
		out.flush();
	}
}
```

(Ausschnitt) Einfaches Servlet Behandlung der GET – Methode

```
<html>
	<body>
		Heute ist der <%= new java.util.Date() %>.
	</body>
</html>
```

(Ausschnitt) Ein einfaches JSP – Beispiel

2.1 JSF - Grundlegendes

Definition

JSF ist „mittlerweile“ ein modernes Standard Java-Webframework zur Erstellung von dynamischen Weboberflächen.

Ziele

Den Entwicklungsprozess von Webanwendungen durch klare Code-Strukturierung und vereinfachtes Management zu verbessern, sowie nachhaltige Projekte zu fördern.

Versionsüberblick

Die Version JSF 1.0 wurde erstmals im Jahre 2004 von JCP spezifiziert. Die Version 1.2 aus dem Jahre 2006 war über mehrere Jahre als Teil von Java EE 5 stark vertreten.

Eine besondere Bedeutung hat die Einführung der JSF Version 2.0 als Teil von Java EE 6 im Jahre 2009. Diese Version veränderte das Arbeiten mit JSF grundlegend. Die aktuelle Version 2.2 vom Mai 2013 brachte abgesehen von der HTML5-Standardisierung vergleichsweise wenige neue Features mit sich.

2.2 JSF - Softwareumgebung

- Installation des JDKs der Version 6 oder 7
- Einsatz eines Anwendungsservers

  Referenzimplementierung durch Oracle Glassfish-Applicationserver

  Weitere Implementierungsmöglichkeiten: Apache MyFaces
- Einsatz einer modernen Entwicklungsumgebung wie Eclipse oder NetBeans
- Einsatz von Appache Maven zur Projektverwaltung (optional)

2.3 JSF - Konfigurationsdateien

Die Deployment-Descriptor ist die wichtigste Konfigurationsdatei und wird als web.xml im Ordner /WEB-INF abgelegt.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
      version="3.0">

      <description>MeinProjekt</description>
      <servlet>
            <servlet-name>Faces Servlet</servlet-name>
            <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
            <load-on-startup>1</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>Faces Servlet</servlet-name>
            <url-pattern>*.jsf</url-pattern>
      </servlet-mapping>
      <welcome-file-list>
            <welcome-file>index.html</welcome-file>
      </welcome-file-list>
      <context-param>
            <param-name>javax.faces.PROJECT_STAGE</param-name>
            <param-value>Development</param-value>
      </context-param>
</web-app>
```

→ *Konfiguration eines Servlets*

→ *Alle Dateien mit der Endung .jsf sollen von einem JSF Servlet bearbeitet werden*

→ *Definition der Startseite*

→ *Definition des Projektzustands*

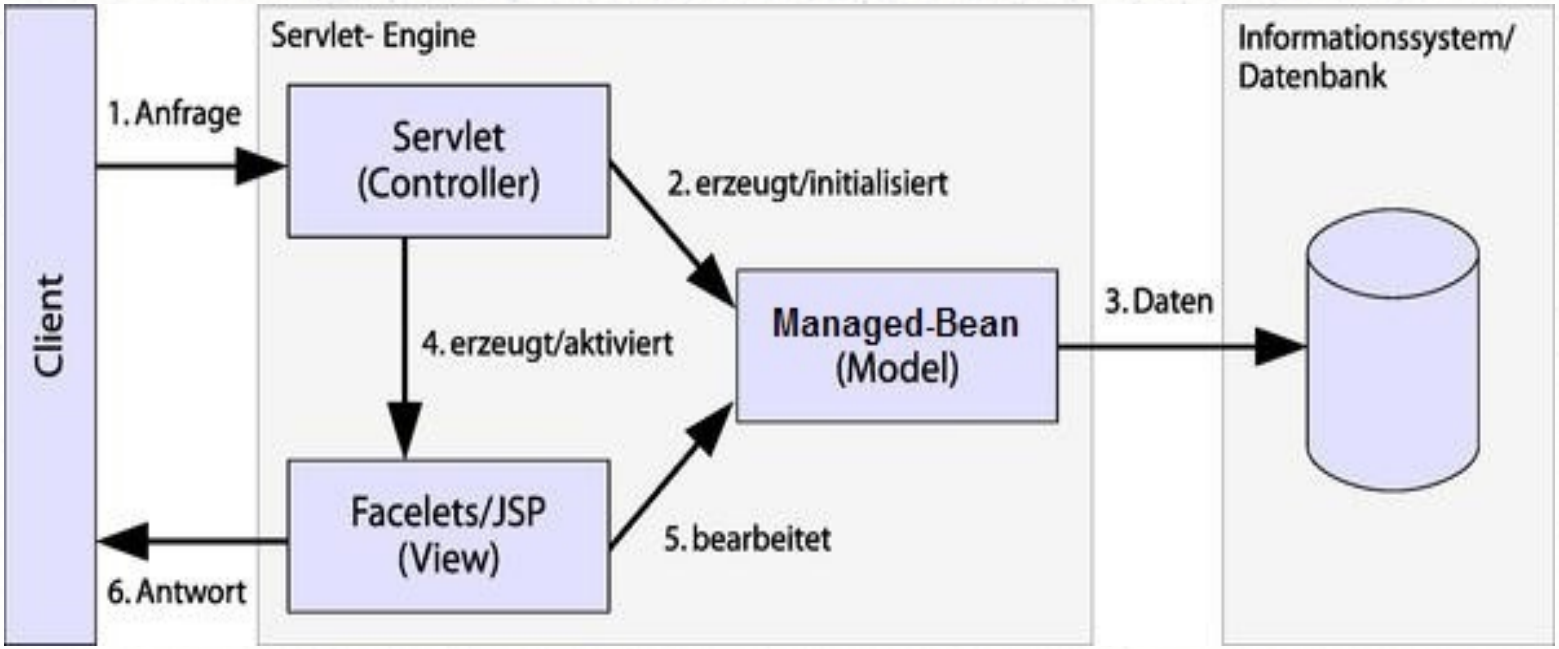
Weitere vornehmbare Konfigurationen: Filter, Security Constraints, Security Roles, Anmeldungsarten etc.

Weitere Konfigurationsdateien: faces-config.xml; pom.xml (Maven)

3 Das Model2-Prinzip

Eine spezialisierte Form des Model-View-Controller-Musters für den Einsatz im Web.

- Controller: Beinhaltet der Steuerungslogik
- Model: Beinhaltet die Daten der Anwendung und die Zugriffsregeln
- View: Stellt die Daten dem Benutzer dar



4 Komponenten

- Seiten in JSF-Anwendungen sind aus eigenständigen und wiederverwendbaren Komponenten aufgebaut.
- Alle Komponenten zusammen werden als Ansicht (View) bezeichnet und hängen in Form einer Baumstruktur an der unsichtbaren Wurzel „UIViewRoot“. Ein sogenannter Renderer übersetzt diese Komponenten in eine Ausgabesprache, wie beispielsweise HTML.

```
<h:form>
<h:panelGrid id="grid" columns="2">
      <h:outputLabel for="inputName" value="Name:"/>
      <h:inputText id="inputName" value="#{meineBean.meinName}"/>
      <h:commandButton id="btnAccept" value="Übernehmen"
                  action="/nächsteSeite.xhtml"/>
</h:panelGrid>
</h:form>
```

5 Facelets

Ein Facelet ist eine Seitendeklarationssprache (View Declaration Language; kurz: VDL), die zur Definition von Views verwendet wird.

- Seit JSF 2.0 sind Facelets standard
- JSP wird aber immer noch von JSF unterstützt

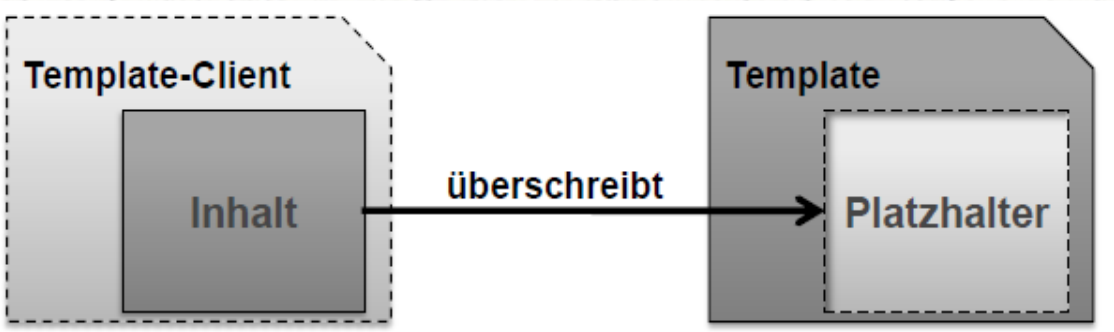
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
      <h:head>
            <title>Facelets</title>
      </h:head>
      <h:body>
            <h1><h:outputText value="meinFacelet"/></h1>
      </h:body>
</html>
```

→ *Einbindung der Komponentenbibliotheken*

Beispiel eines Facelets

6 Templating

Zentrale Erfassung von Layout und Design, dass von allen Views der Anwendung wiederverwendet und überschrieben werden kann.



Vorteile:

- Vermeidung von Redundanzen
- Steigert die Flexibilität, weil Änderungen zentral erfolgen und von allen benutzenden Views gleichzeitig übernommen werden

6 Templating

```xml
<?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:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<head>
      <title>MeinLayout</title>
      <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
      <div id="header">
            <ui:insert name="header">
                  <h:graphicImage value="/images/Header.png"/>
            </ui:insert>
      </div>
      <div id="content" style="margin: 50px">
            <ui:insert name="content"/>
      </div>
      <div id="footer">
            <ui:insert name="footer">
                  <h:graphicImage value="/images/Footer.png"/>
            </ui:insert>
      </div>
</body>
</html>
```

→ *Das Template*

ui : insert = Platzhalter definieren

Die Ausgabe

```xml
<ui:composition template="template.xhtml"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
      <ui:define name="content">
            <h1><h:outputText value="mein Inhalt"/></h1>
      </ui:define>
</ui:composition>
```

→ *Der Template – Client*

ui : composition = Einbindund des Templates

ui : define = Platzhalter üerschreiben

7 Ausgabekomponenten der Standardbibliothek

Die HTML-Custom-Tag-Library bietet viele Standard-Komponenten an, die über das Präfix h verwendet werden können.


```
<body>
	<h1><h:outputText value="Ausgabekomponenten"/></h1>
	<h:form id="form">
	<h:panelGrid id="gridOutput" columns="1">

		<h:outputText value="Ich bin ein &lt;em>h:outputText&lt;/em>"
			escape="false"/>

		<h:outputLabel value="Ich bin ein h:outputLabel"
			style="font-weight:bold"/>

		<h:outputLink value="http://google.de">
			<h:graphicImage id="graphic" url="/images/Beispielbild.png"/>
		</h:outputLink>

		<h:outputFormat value="{0} bin ein {1}">
			<f:param value="Ich"/>
			<f:param value="h:outputFormat"/>
		</h:outputFormat>

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

Beispiele von weiteren Komponententypen:

Befehlskomponenten (h:commandButton); Datenkomponenten (h:dataTable), Auswahlkomponenten (h:selectBooleanCheckbox), Eingabekomponenten (h:inputText); Dateiuploadkomponenten (h:inputFile) etc.

8 Managed-Beans

- Managed-Beans sind gewöhnliche Java-Klassen, die das Modell beziehungsweise die Verbindung zwischen dem Modell und der Geschäftslogik bilden.

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

@ManagedBean
@SessionScoped
public class MeineJSFBean {

	private String meinName;

	public String getMeinName() {
		return meinName;
	}

	public void setMeinName(String meinName) {
		this.meinName = meinName;
	}

	public String willkommen() {
		return "Hallo " + meinName;
	}
}
```

Eine einfache Managed – Bean

→ *Annotation (Registrierung einer Bean)*
→ *Lebensdauer einer Managed – Bean*

Name	Annotation	Eigenschaft
None-Scope	@NoneScoped	Die Managed-Bean wird bei jedem Aufruf neu erstellt.
Request-Scope	@RequestScoped	Die Managed-Bean lebt für die Zeitdauer einer HTTP-Anfrage.
View-Scope	@ViewScoped	Die Lebensdauer der Managed-Bean ist an die Ansicht geknüpft, in der sie verwendet wird.
Session-Scope	@SessionScoped	Die Managed-Bean lebt für die Dauer einer Sitzung, in der der Benutzer mit der Anwendung verbunden ist.
Application-Scope	@ApplicationScoped	Für die gesamte Lebensdauer der Anwendung ist nur eine für alle Benutzer gleiche Instanz dieser Managed-Bean vorhanden.

9 Unified Expression Language

Die Unified EL ist das Verbindungsglied zu den den Daten der Managed-Beans.

- **Value-Expressions**

  Binden eines Komponentenattributes an eine Managed-Bean.

  Syntax: `komponentenattribut = “#{bean.datenfeldname}“`

  `Bsp: value = “#{meineJSFBean.meinName}“`

- **Method-Expressions**

  Komponenten mit Methoden einer Managed-Bean verknüpfen.

  Syntax: `komponentenattribut = “#{bean.methodenname}“`

  `Bsp: value = “#{meineBean.willkommen}“`

10 Konvertierung

Konverter wandeln beim Absenden eines Formulars (vom Clienten), Strings in Java-Typen um und beim Rendern wieder den Java-Datentyp in einen String zurück.

Sofern kein benutzerdefinierter Konverter erstellt wurde, benutzt JSF automatisch den Standardkonverter (= implizite Konvertierung), den es für die meisten einfachen Datentypen gibt.

```
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.ConverterException;

public interface Converter
{
	Object getAsObject(FacesContext context, UIComponent component,
		String value) throws ConverterException;
	String getAsString(FacesContext context, UIComponent component,
		Object value) throws ConverterException;
}
```

Das Interface , das jeder Konverter implementieren muss

10 Konvertierung

Standardkonverter:

Bei Nummern- und Datumsformaten gibt es keine einheitlichen Standards, weshalb das Verhalten des Standardkonverters nicht ausreichend ist und ergänzt werden muss.

Beispiel `<f:convertNumber>`

Verwendung erfolgt über die View heraus mittels Konvertierungkomponenten der Core-Tag-Library unter dem Präfix f.

```
<h:outputText value="#{meineBean.meineNummer}" >
      <f:convertNumber groupingUsed="true" type="currency"/>
</h:outputText>
```

Eine Eingabe von „9999.906“ würde mit oben aufgeführter Konvertierung zur Anzeige von „9.999,91 €“ führen.

11 Validierung

Benutzereingaben sollten stets kontrolliert werden, um Systemabstürze und die Speicherung von fehlerhaften Daten zu vermeiden. JSF bietet für die Fehlerüberprüfung mehrere Möglichkeiten an.

Standardvalidatoren:

Standardvalidatoren werden ähnlich wie Standardkonverter über Validierungskomponenten der Core-Tag-Library verwendet.

```
<h:outputLabel for="inputName" value="Name:"/>
<h:inputText id="inputName" value="#{meineBean.meinName}">
      <f:validateRequired />
      <f:validateLength minimum="1" maximum="10"/>
</h:inputText>
```

→ *Eingabe darf nicht null sein*
→ *Eingabe ist mind.* 1 *und max.* 10 *Buchstaben lang*

Hinweis: Normalerweise ignoriert JSF Nullwerte. Daher muss noch eine Eintragung in der faces-config.xml vorgenommen werden, damit die Komponente f:validateRequired funktioniert.

11 Validierung

Bean Validation nach JSR-303:

Diese Art der Validierung ist erst seit JSF 2.0 möglich und ist besonders beliebt, da es den Code in der View reduziert und die Fehlerkontrolle zentral in der Klasse erfolgt.

```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Min;
import javax.validation.constraints.Max;
import javax.validation.constraints.Size;

@ManagedBean
@SessionScoped
public class MeinValidator {

      @NotNull @Size(min = 1, max = 10)
      private String meinName;
      …
```

Äquivalente Validierung zum vorherigen Beispiel

12 Navigation

Die Navigation zwischen den einzelnen Ansichten spielt bei Webanwendungen eine fundamentale Rolle. Benutzer navigieren über Befehlskomponenten, die das Absenden der aktuellen Seite an den Server veranlassen. JSF bietet für die Navigation zwei Varianten an.

Implizite Navigation:

Diese Art der Navigation ist erst seit JSF möglich und bietet eine äußerst einfache Möglichkeit über das action-Attribut einer Befehlskomponente zu navigieren.

```
<h:commandButton id="btnAccept" value="Übernehmen"
      action="/zweiteSeite.xhtml?faces-redirect=true"/>
```

Implizite Navigation mit Redirect.

12 Navigation

Regelbasierende Navigation:

Vor JSF 2.0 konnten lediglich Navigationsregeln verwendet werden, die in der Datei faces-config.xml konfiguriert wurden.

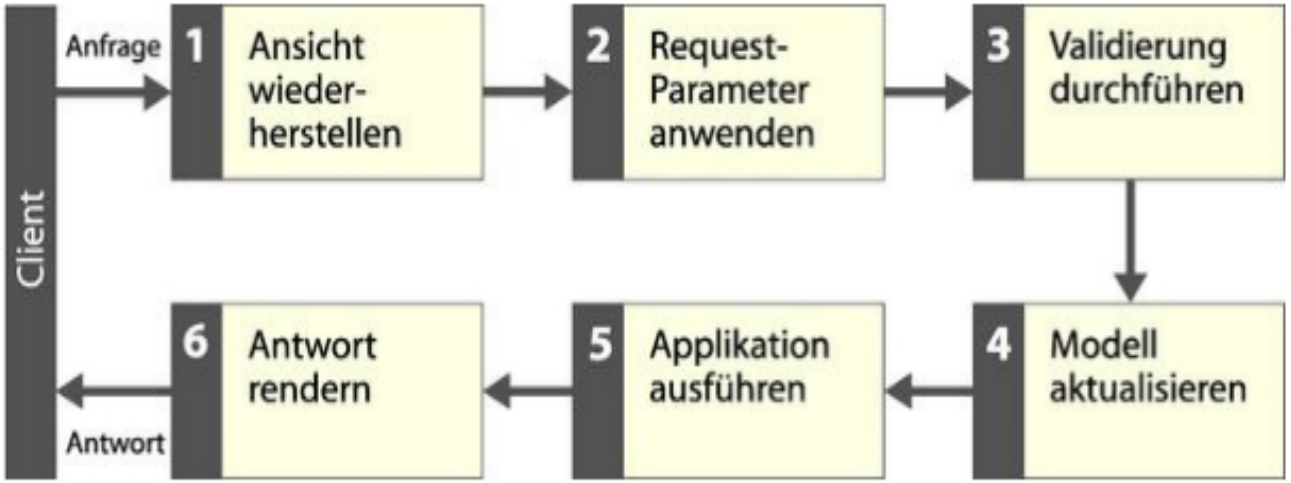
```
<navigation-rule>
      <from-view-id>/ersteSeite.xhtml</from-view-id>
      <navigation-case>
            <from-outcome>ok</from-outcome>
            <to-view-id>/zweiteSeite.xhtml</to-view-id>
      </navigation-case>
</navigation-rule>
```

→ *Beliebige Anzahl an Regeln festlegen*
→ *Von welcher Seite ist die Regel gültig*

→ *Der erwartete Rückgabewert*
→ *Definition der Zielseite*

13 JSF Lebenszyklus

Der Lebenszyklus einer JSF-Applikationen wird immer dann in Gang gesetzt, wenn der Benutzer die aktuelle Seite absendet oder das System selbst ein Event auslöst.



14 PrimeFaces

PrimeFaces ist eine Open-Source-Komponentenbibliothek mit etwa 100 aufeinander abgestimmten Komponenten, die viele Funktionalitäten weit über den JSF-Standard hinaus anbieten.

Die Einbindung erfolgt über eine Jar-Datei, die in das Projekt eingebunden werden muss. Bei Verwendung von Maven muss nur die Abhängigkeit in die pom.xml eingetragen werden.

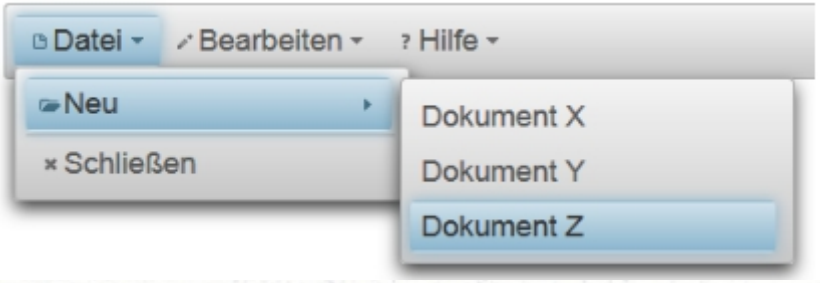
```
<repositories>
        <repository>
            <id>prime-repo</id>
            <name>PrimeFaces Maven Repository</name>

<url>http://repository.primefaces.org</url>
            <layout>default</layout>
        </repository>
    </repositories>

<dependencies>
      ...
      ...
      <dependency>
            <groupId>org.primefaces</groupId>
            <artifactId>primefaces</artifactId>
            <version>4.0</version>
        </dependency>
      ...
      ...
<dependencies>
```

14 PrimeFaces

Beispiel <p:menubar>




```
<p:menubar>
	<p:submenu label="Datei" icon="ui-icon-document">
		<p:submenu label="Neu" icon="ui-icon-folder-open">
			<p:menuitem value="Dokument X" url="#" />
			<p:menuitem value="Dokument Y" url="#" />
			<p:menuitem value="Dokument Z" url="#" />
		</p:submenu>
		<p:separator />
		<p:menuitem value="Schließen" url="#" icon="ui-icon-close" />
	</p:submenu>
	<p:submenu label="Bearbeiten" icon="ui-icon-pencil">
		<p:menuitem value="Rückgängig" url="#"
			icon="ui-icon-arrowreturnthick-1-w" />
		<p:menuitem value="Wiederholen" url="#"
			icon="ui-icon-arrowreturnthick-1-e" />
	</p:submenu>
	<p:submenu label="Hilfe" icon="ui-icon-help">
		<p:menuitem value="Problem melden" url="#" />
		<p:menuitem value="Suche" icon="ui-icon-search" url="#"/>
	</p:submenu>
</p:menubar>
```

14 PrimeFaces

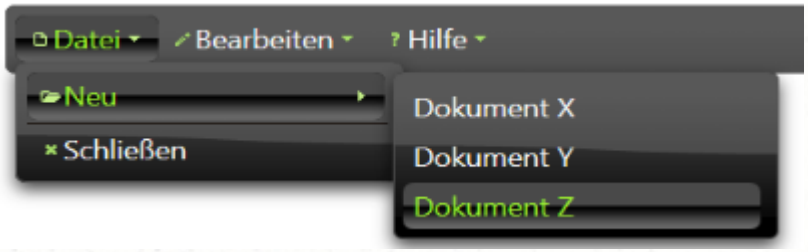
Eine besondere Stärke von PrimeFaces ist die Verwaltung von Themes. Es werden über 30 verschiedene Erscheinungsmuster angeboten, die über eine Jar-Datei in das Projekt eingebunden werden. Standardmäßig verwendet PrimeFaces das aristo-Theme.

```
<dependency>
	<groupId>org.primefaces.themes</groupId>
	<artifactId>all-themes</artifactId>
	<version>1.0.10</version>
</dependency>
```

Einbindung der Themes über die pom.xml

```
<context-param>
	<param-name>primefaces.THEME</param-name>
	<param-value>trontastic</param-value>
</context-param>
```

Konfiguration des aktuellen Themes



Das trontastic Theme

15 Fazit

Vorteile von JSF

- Ermöglicht die Erstellung von komplexen und umfangreichen Webapplikationen
- Intelligente Integration der Konzepte
- Klare Code-Trennung
- Umfangreicher Support von vielen Open-Source-Herstellern

Nachteile von JSF

- flache Lernkurve
- Umfangreiche Integration
- Viele Technologien müssen nachintegriert werden

Danke für Ihre Aufmerksamkeit!

Haben Sie noch Rückfragen?