JAKARTA SERVER FACES

Speaker notes

- Früher Java Server Faces
- gehört zu Jakarta EE (ehemals Java EE)
- Framework basiert auf Servlets und JSP
- Framework Standard

ZIELE

- Requestübergreifendes Management des States der UI Komponenten
- Stark typisiertes Eventmodel um serverseitig Client Events zu behandeln
- Kapselung der Unterschiede innerhalb der Clients und Browsern

- Handling der Navigation sowie Error und Exception Events
- Typ Konvertierung
- Validierung der Daten sowie entsprechendes Errorhandling

Speaker notes

Typ Konvertierung: vom Application Model zu Anzeigestrings und umgedreht

ABGRENZUNG ZU JSP

JSF

ist ein MVC Framework, welches zu dem ein Lifecycle enthält

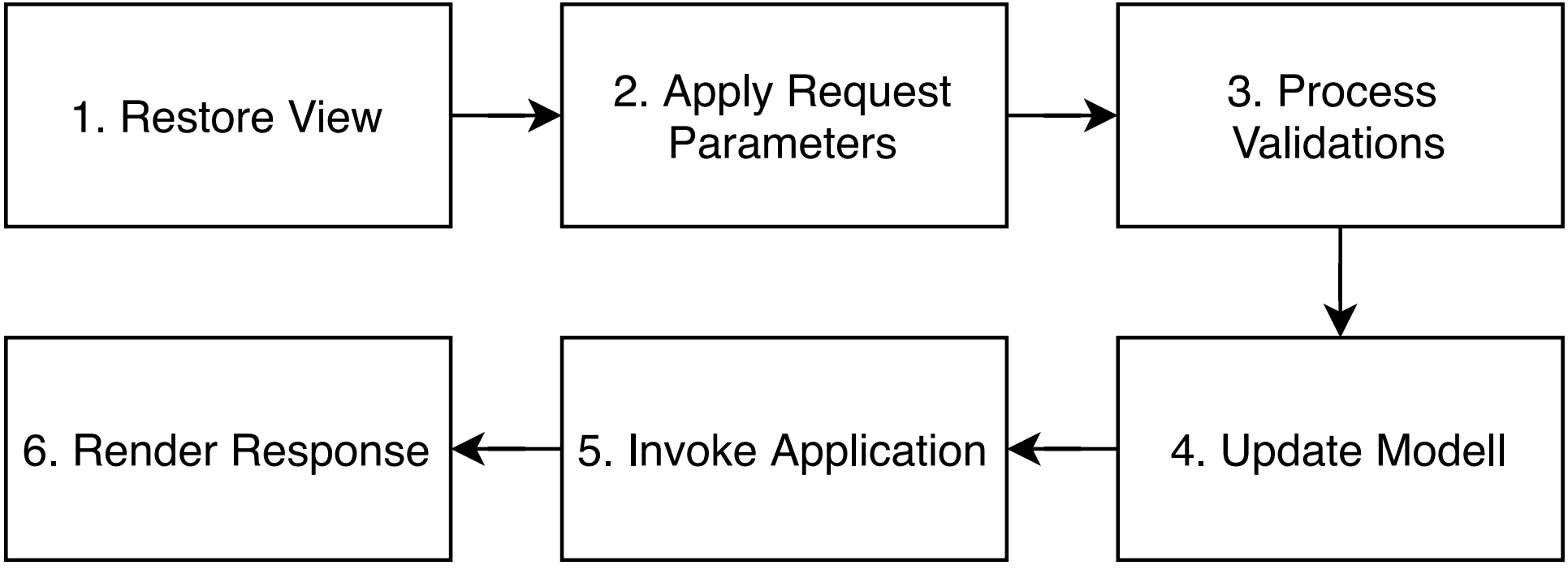
JSP

ermöglicht das dynamische Erzeugen von HTML Pages

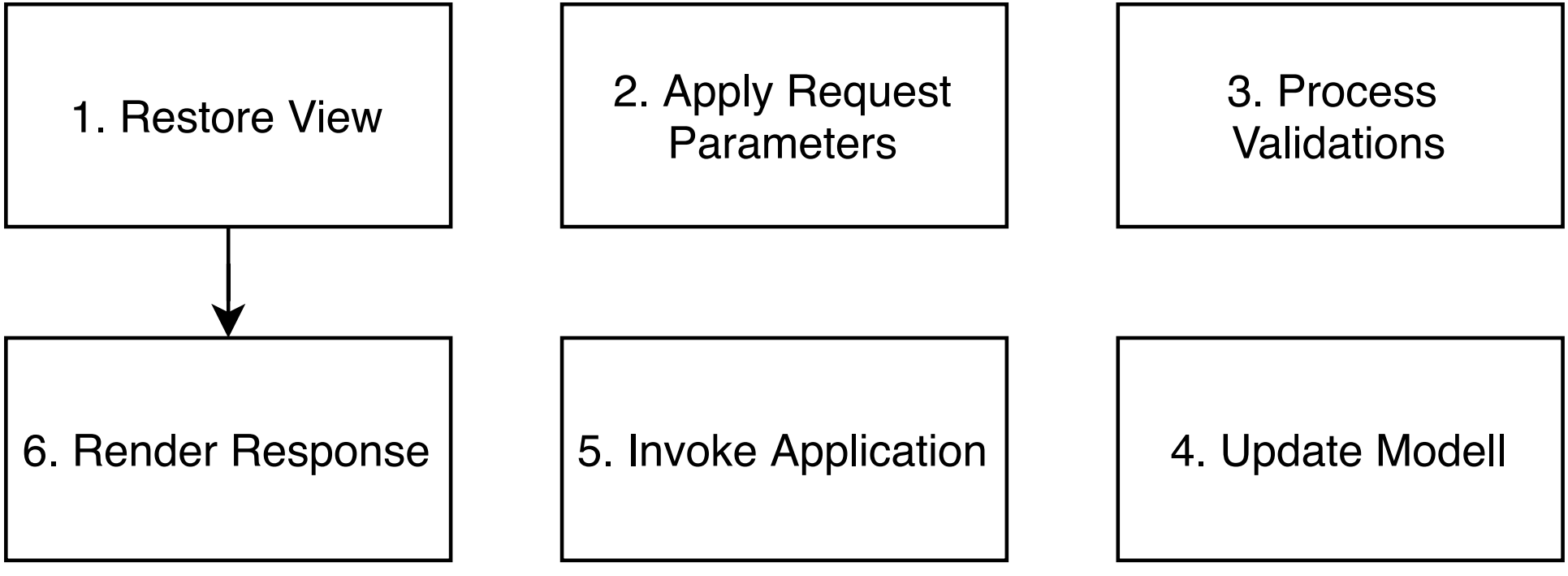
JSF KONFIGURATION

- Konfiguration in faces-config.xml im WEB-INF Ordner
- hier gibt es Definitionen zu:
  - Navigation
  - Listeners
  - Beans
  - etc.

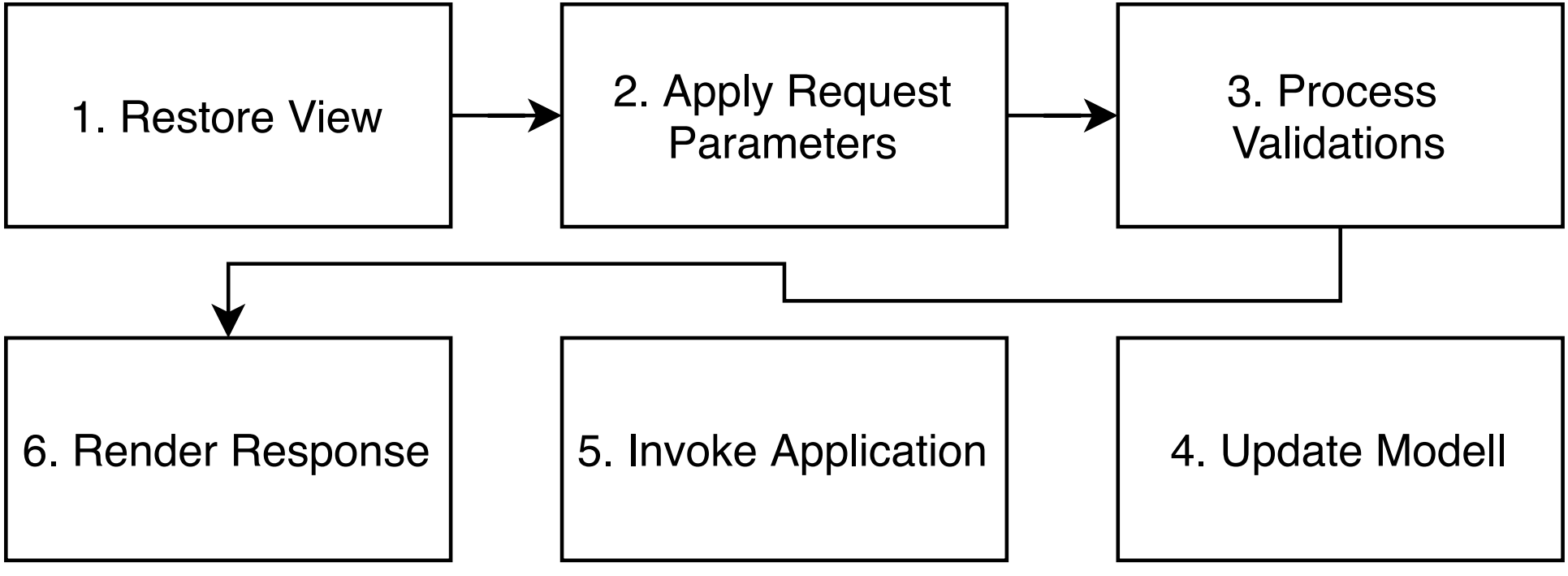
LIFECYCLE


1. Restore View
2. Apply Request Parameters
3. Process Validations
4. Update Modell
5. Invoke Application
6. Render Response

LIFECYCLE: INITIALER AUFRUF



LIFECYCLE: VALIDATION ERROR



ERSTE AUSGABE

```
<!DOCTYPE html>
<html lang="de"  xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html">
    <head>
        <meta charset="UTF-8"/>
        <title>Hello World of JSF</title>
    </head>
    <body>
        <h:outputText value="Hello World"/>
    </body>
</html>
```

ERSTE AUSGABE

```
1  <!DOCTYPE html>
2  <html lang="de"  xmlns="http://www.w3.org/1999/xhtml"
3      xmlns:f="http://java.sun.com/jsf/core"
4      xmlns:h="http://java.sun.com/jsf/html">
5      <head>
6          <meta charset="UTF-8"/>
7          <title>Hello World of JSF</title>
8      </head>
9      <body>
10         <h:outputText value="Hello World"/>
11     </body>
12 </html>
```

ERSTE AUSGABE

```
<!DOCTYPE html>
<html lang="de"  xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html">
    <head>
        <meta charset="UTF-8"/>
        <title>Hello World of JSF</title>
    </head>
    <body>
        <h:outputText value="Hello World"/>
    </body>
</html>
```

UNIFIED EXPRESSION LANGUAGE

- Grundsyntax

```
${expression} // immediate
#{expression} // deferred
```

Speaker notes

Immediate: direkte Ausführung in der ersten Phase
Deferred: wird in jeder Phase neu ausgeführt

LITERALE

```
${'hello ' + 'world'} // hello world
${21*2} // 42

${'\${' + '\#{'} // $ und # müssen escaped werden
```

Speaker notes

Literal: Konstante Ansonsten gleich wie bei JSP

ATTRIBUTZUGRIFF AUF BEANS

Rein lesend:

```
${bean.attribute} // => bean.getAttribute
```

Speaker notes

ATTRIBUTZUGRIFF AUF BEANS

Lesend und schreibend:

```
#{bean.attribute} // Funktion phasenabhängig
```

- Lesender Zugriff in den Phasen: `process validations` und `render response`
- Schreibender Zugriff in den Phasen: `update Model`
- Beispiel (Two-Way-Binding, lesend+schreiben):

```
<h:InputText value="#{bean.name}"/>
```

Speaker notes

BEANS

Speaker notes

- funktionieren grundlegend wie JSP Beans
- einige Unterschiede

BEANS DEFINIEREN

- Annotations:
  - Zugriff über Klassenname mit kleinem Anfangsbuchstabe

```
1 @ManagedBean
2 @ViewScope
3 public class SayHello
```

- Faces-config.xml:
  - Zugriff über Attributwert managed-bean-name

```
1 <managed-bean>
2   <managed-bean-name>sayHello</managed-bean-name>
3   <managed-bean-class>demo.SayHello</managed-bean-class>
4   <managed-bean-scope>request</managed-bean-scope>
5 </managed-bean>
```

SCOPES

- Request Scope `RequestScoped`
- View Scope `ViewScoped`
  - während der Nutzer auf dem gleichen View bleibt
  - gilt auch über mehrere Page-Reloads
- Session `SessionScoped`
  - gilt für die Session des Nutzers
- Application Scope `ApplicationScoped`
  - Gilt über die komplette Laufzeit

BAKING BEANS

- JSF ruft die Getter und Setter in jeder Phase neu auf
  - unnötige Resourcenverschwendung
- Aufwendige Berechnungen oder Aufrufe externer Dienste aus Beans sind teuer
- Füge Baking Beans zwischen View und Logik ein
  - Pufferung und Interpretation der Daten

NAVIGATION

- explizit über in `faces-config.xml` definierte Regeln
- implizit wenn keine explizite Regel greift

NAVIGATION RULE (EXPLIZITE NAVITION)

```
1 <navigation-rule>
2     <from-view-id>/login.xhtml</from-view-id>
3     <navigation-case>
4         <from-action>#{LoginForm.login}</from-action>
5         <from-outcome>success</from-outcome>
6         <to-view-id>/storefront.xhtml</to-view-id>
```

Speaker notes

- from-view-id: legt fest, für welchen View die Navigation Rule gilt. Es ist auch möglich Wildcards zu nutzen.
- navigation-case: Die Fälle werden von oben nach unten abgearbeitet. Sollte einer zutreffen, wird der erste Treffer zur Navigation genutzt.
- from-action und from-outcome sind optional.
  - Wenn beide spezifiziert sind, wird der Rückgabewert der Action (from-action) mit dem Wert in from-outcome verglichen
  - Ist nur from-outcome definiert, muss der Wert mit dem Rückgabewert der Action der jakarta.faces.component.UICommand Komponenten bzw. deren Implementierung übereinstimmen.
  - Sofern nur from-action definiert ist muss ein Component Tag existieren, welches dem Rückgabewert entspricht.
- weitere Attribute möglich

NAVIGATION RULE (EXPLIZITE NAVITION)

```
<navigation-rule>
    <from-view-id>/login.xhtml</from-view-id>
    <navigation-case>
        <from-action>#{LoginForm.login}</from-action>
        <from-outcome>success</from-outcome>
        <to-view-id>/storefront.xhtml</to-view-id>
```

Speaker notes

- from-view-id: legt fest, für welchen View die Navigation Rule gilt. Es ist auch möglich Wildcards zu nutzen.
- navigation-case: Die Fälle werden von oben nach unten abgearbeitet. Sollte einer zutreffen, wird der erste Treffer zur Navigation genutzt.
- from-action und from-outcome sind optional.
  - Wenn beide spezifiziert sind, wird der Rückgabewert der Action (from-action) mit dem Wert in from-outcome verglichen
  - Ist nur from-outcome definiert, muss der Wert mit dem Rückgabewert der Action der jakarta.faces.component.UICommand Komponenten bzw. deren Implementierung übereinstimmen.
  - Sofern nur from-action definiert ist muss ein Component Tag existieren, welches dem Rückgabewert entspricht.
- weitere Attribute möglich

NAVIGATION RULE (EXPLIZITE NAVITION)

```
1 <navigation-rule>
2     <from-view-id>/login.xhtml</from-view-id>
3     <navigation-case>
4         <from-action>#{LoginForm.login}</from-action>
5         <from-outcome>success</from-outcome>
6         <to-view-id>/storefront.xhtml</to-view-id>
```

Speaker notes

- from-view-id: legt fest, für welchen View die Navigation Rule gilt. Es ist auch möglich Wildcards zu nutzen.
- navigation-case: Die Fälle werden von oben nach unten abgearbeitet. Sollte einer zutreffen, wird der erste Treffer zur Navigation genutzt.
- from-action und from-outcome sind optional.
  - Wenn beide spezifiziert sind, wird der Rückgabewert der Action (from-action) mit dem Wert in from-outcome verglichen
  - Ist nur from-outcome definiert, muss der Wert mit dem Rückgabewert der Action der jakarta.faces.component.UICommand Komponenten bzw. deren Implementierung übereinstimmen.
  - Sofern nur from-action definiert ist muss ein Component Tag existieren, welches dem Rückgabewert entspricht.
- weitere Attribute möglich

NAVIGATION RULE (EXPLIZITE NAVITION)

```
1 <navigation-rule>
2     <from-view-id>/login.xhtml</from-view-id>
3     <navigation-case>
4         <from-action>#{LoginForm.login}</from-action>
5         <from-outcome>success</from-outcome>
6         <to-view-id>/storefront.xhtml</to-view-id>
```

Speaker notes

- from-view-id: legt fest, für welchen View die Navigation Rule gilt. Es ist auch möglich Wildcards zu nutzen.
- navigation-case: Die Fälle werden von oben nach unten abgearbeitet. Sollte einer zutreffen, wird der erste Treffer zur Navigation genutzt.
- from-action und from-outcome sind optional.
  - Wenn beide spezifiziert sind, wird der Rückgabewert der Action (from-action) mit dem Wert in from-outcome verglichen
  - Ist nur from-outcome definiert, muss der Wert mit dem Rückgabewert der Action der jakarta.faces.component.UICommand Komponenten bzw. deren Implementierung übereinstimmen.
  - Sofern nur from-action definiert ist muss ein Component Tag existieren, welches dem Rückgabewert entspricht.
- weitere Attribute möglich

NAVIGATION RULE (EXPLIZITE NAVITION)

```
1 <navigation-rule>
2     <from-view-id>/login.xhtml</from-view-id>
3     <navigation-case>
4         <from-action>#{LoginForm.login}</from-action>
5         <from-outcome>success</from-outcome>
6         <to-view-id>/storefront.xhtml</to-view-id>
```

Speaker notes

- from-view-id: legt fest, für welchen View die Navigation Rule gilt. Es ist auch möglich Wildcards zu nutzen.
- navigation-case: Die Fälle werden von oben nach unten abgearbeitet. Sollte einer zutreffen, wird der erste Treffer zur Navigation genutzt.
- from-action und from-outcome sind optional.
  - Wenn beide spezifiziert sind, wird der Rückgabewert der Action (from-action) mit dem Wert in from-outcome verglichen
  - Ist nur from-outcome definiert, muss der Wert mit dem Rückgabewert der Action der jakarta.faces.component.UICommand Komponenten bzw. deren Implementierung übereinstimmen.
  - Sofern nur from-action definiert ist muss ein Component Tag existieren, welches dem Rückgabewert entspricht.
- weitere Attribute möglich

NAVIGATION RULE (EXPLIZITE NAVITION)

```
1 <navigation-rule>
2     <from-view-id>/login.xhtml</from-view-id>
3     <navigation-case>
4         <from-action>#{LoginForm.login}</from-action>
5         <from-outcome>success</from-outcome>
6         <to-view-id>/storefront.xhtml</to-view-id>
```

Speaker notes

- from-view-id: legt fest, für welchen View die Navigation Rule gilt. Es ist auch möglich Wildcards zu nutzen.
- navigation-case: Die Fälle werden von oben nach unten abgearbeitet. Sollte einer zutreffen, wird der erste Treffer zur Navigation genutzt.
- from-action und from-outcome sind optional.
  - Wenn beide spezifiziert sind, wird der Rückgabewert der Action (from-action) mit dem Wert in from-outcome verglichen
  - Ist nur from-outcome definiert, muss der Wert mit dem Rückgabewert der Action der jakarta.faces.component.UICommand Komponenten bzw. deren Implementierung übereinstimmen.
  - Sofern nur from-action definiert ist muss ein Component Tag existieren, welches dem Rückgabewert entspricht.
- weitere Attribute möglich

IMPLIZITE NAVIGATION

Greift keine Navigation Regel dann wird der Rückgabewert der jeweiligen Action als relative URL genutzt.

ZUSÄTZLICHE INTERFACES

- Converters
- Events and Listeners
- Validators

Speaker notes

* Converter: Pluggable support class to convert the markup value of a component to and from the corresponding type in the model tier. * Events and Listeners: An event broadcast and listener registration model based on the design patterns of the JavaBeans Specification, version 1.0.1. * Validators: Pluggable support classes that can examine the local value of a component (as received in an incoming request) and ensure that it conforms to the business rules enforced by each Validator. Error messages for validation failures can be generated and sent back to the user during rendering.

COMMON EVENT PROCESSING

EVENT TYPES

- Application Events
  - ActionEvent
  - ValueChangeEvent
- System Events

Speaker notes

Application Events: Anwendungsspezifisch, werden von Anwendungsentwicklern geschrieben, immer an eine UIComponent gebunden System Events: representieren spezifische Zeitpunkte in der Anwendungsausführung

APPLICATION EVENTS

- extends `jakarta.faces.event.FacesEvent`
- enthält immer die triggernde Komponenten
- werden an eine Lifecycle Phase gebunden
- können gequeued werden

Speaker notes

LISTENER CLASSES APPLICATION EVENTS

- für jeden Application Event Type eine Listener
  - ActionListener
  - ValueChangeListener

SYSTEM EVENTS

- extends `SystemEvent`
- alle spezifizierte Events unter `jakarta.faces.event`
- Beispiel: `PostConstructApplicationEvent`

COMPONENT SYSTEM EVENTS

- spezielle System Events
- beinhalten einen Verweis auf eine Komponente
- Beispiel: `PreRenderViewEvent`

LISTENER CLASSES

- nur zwei Listener Classes nötig
- für System Events:
  `jakarta.faces.event.SystemEventListener`
- für Component System Events:
  `jakarta.faces.event.ComponentSystemEventListene`

VALIDATOREN

- Validatoren für Beans und Input Fields
- Standardvalidatoren für gängige Datentypen
  - einfache Einschränkungen für Standard-Datentypen
- eigene komplexe Validatoren möglich

STANDARDVALIDATOREN

```
<f:validateLength minimum="6" maximum="15"/>
<f:validateLongRange minimum="6" maximum="15"/>
<f:validatePattern pattern="<myPattern>"/>
```

VERWENDUNG

```
<h:inputText value="#{bean.variable}"
             validator="validatorMethod()" />
<h:inputText value="#{bean.variable}">
    <f:validateRequired/>
</inputText>
```

KONVERTER

- Validatoren für Beans und Input Fields
- Konverter für gängige Datentypen
  - BooleanConverter, BigIntegerConverter, DateTimeConverter etc.

VERWENDUNG

- wird automatisch zugeordnet durch den Typ der Beanvariable

```
<h:inputText value="#{bean.variable}"/>
```

JSF-TAGS

Speaker notes

- nur die für die Praxis relevanten
- die JSF Tags besitzen mindestens die gleichen Attribute wie das HTML Pendant

FORM

```
<h:form id="id">
    ...
</h:form>
```

Speaker notes

https://www.javatpoint.com/jsf-form Besitzt die gleichen Attribute wie das HTML Tag. Wird benutzt um Formulare zu wrappen

CHECKBOX

```
<h:selectBooleanCheckbox value="#{item.done}"
                         disabled="true" />
```

Speaker notes

https://www.javatpoint.com/jsf-form Besitzt die gleichen Attribute wie das HTML Tag. Wird benutzt um Formulare zu wrappen

INPUT TEXT

```
<h:inputText value="#{item.title}"/>
```

Speaker notes

https://www.javatpoint.com/jsf-inputtext

INPUT HIDDEN

```
<h:inputHidden value="#{item.title}"/>
```

Speaker notes

https://www.javatpoint.com/jsf-inputhidden

COMMANDBUTTON

```
<h:commandButton value="Speichern"
                 action="#{manager.save()}"/>
```

Speaker notes

https://www.javatpoint.com/jsf-commandbutton Der Return Wert der Methode muss ein String sein. Dieser wird dann zur Navigation genutzt.

REQUIREMENTS

- JDK
- Servlet-Container (bspw. Apache Tomcat)

Speaker notes

Technische Voraussetzungen: JDK und Servlet-Container zur Entwicklung

Für das JSF Verständnis wird allerdings auch Grundverständnis über HTTP und Java sowie grundlegende MVC Kenntnisse benötigt.

PRAXIS

- erstellen Todo Listen Anwendung

REQUIREMENTS

- Todo Liste
- erkennbar welche Todo bereits abgeschlossen
- Todos sollen hinzugefügt und editiert werden können
- Todos enthalten ID, Titel und Description