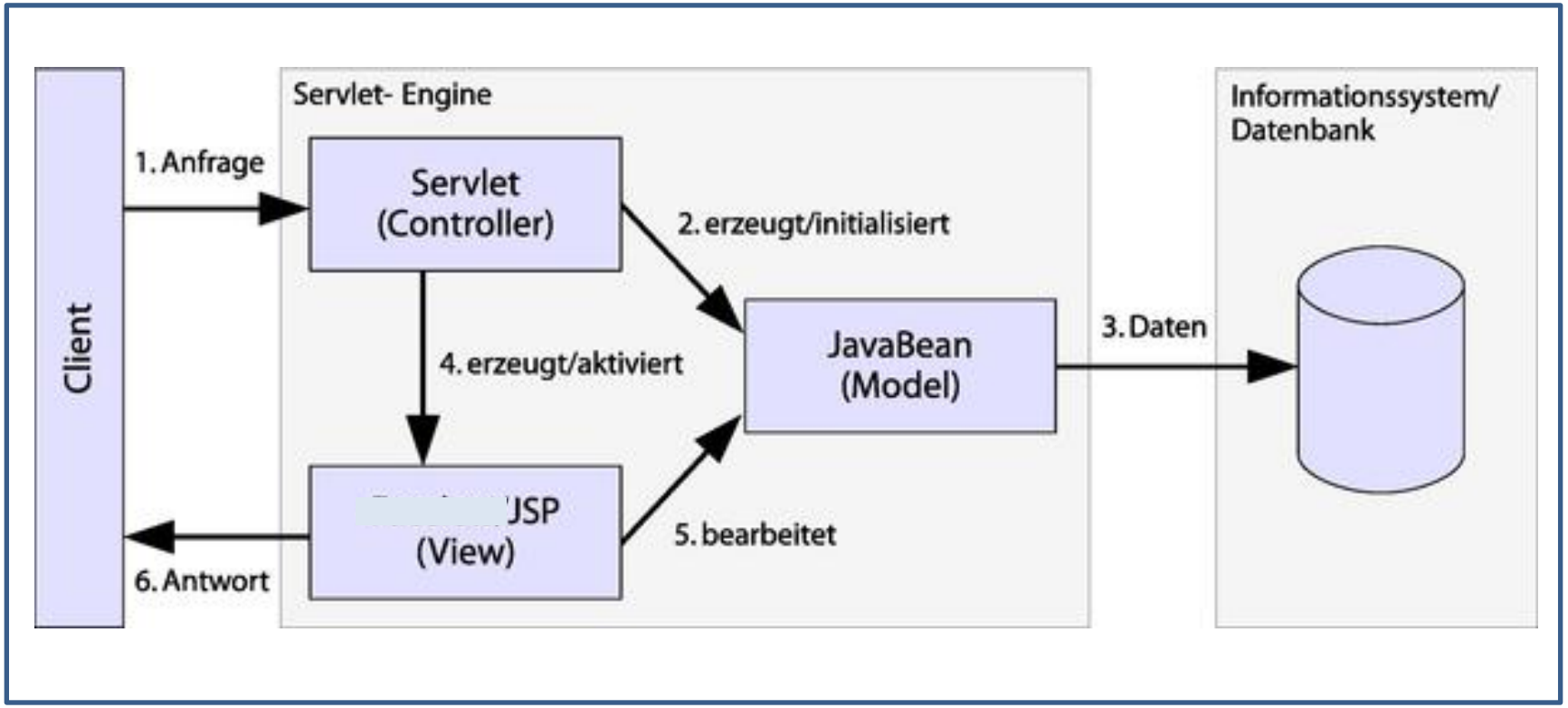
Java Server Faces (JSF)

Modul 7
JSF: Motivation



Model2 MVC Pattern

Quelle: http://jsfatwork.irian.at/semistatic/introduction.html

Modul 7
JSF: Motivation - MVC automatisieren

1) Aus JSP HTML generieren und HTML an Browser senden

2) Servlet wird von JSP-Seite aufgerufen. Servelt holt sich die request-Parameter
(`String userStr = request.getParamater("name");`)

3) Überprüfen der Parameter
(`if (userStr==null) { ... };`)

4) Übertragen der Daten in die entsprechende JavaBean (Model-Update)
(`userBean.name = userStr; …`)

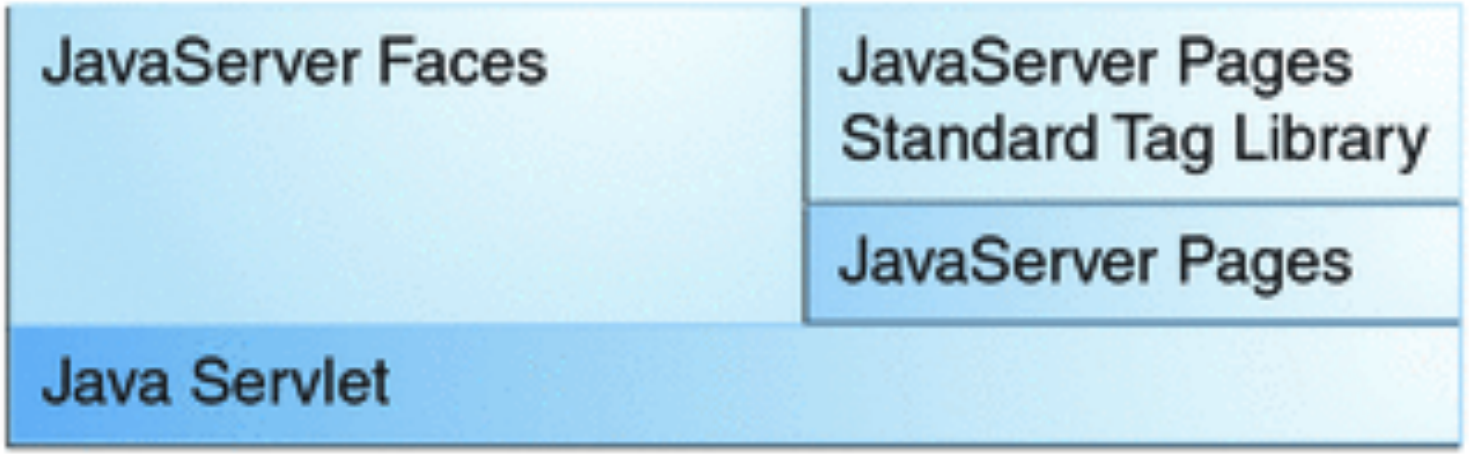
5) Ausführen von Methoden (Business-Logik)
(`userBean.checkName(...);`
`userBean.saveNameinDB(...);`)
Ermitteln und Aufrufen der darzustellenden Seite.
(`RequestDispatcher rd =`
`request.getRequestDispatcher("welcome.jsp"); )`

6) Weiter bei (1)

Modul 7
JSF: Motivation

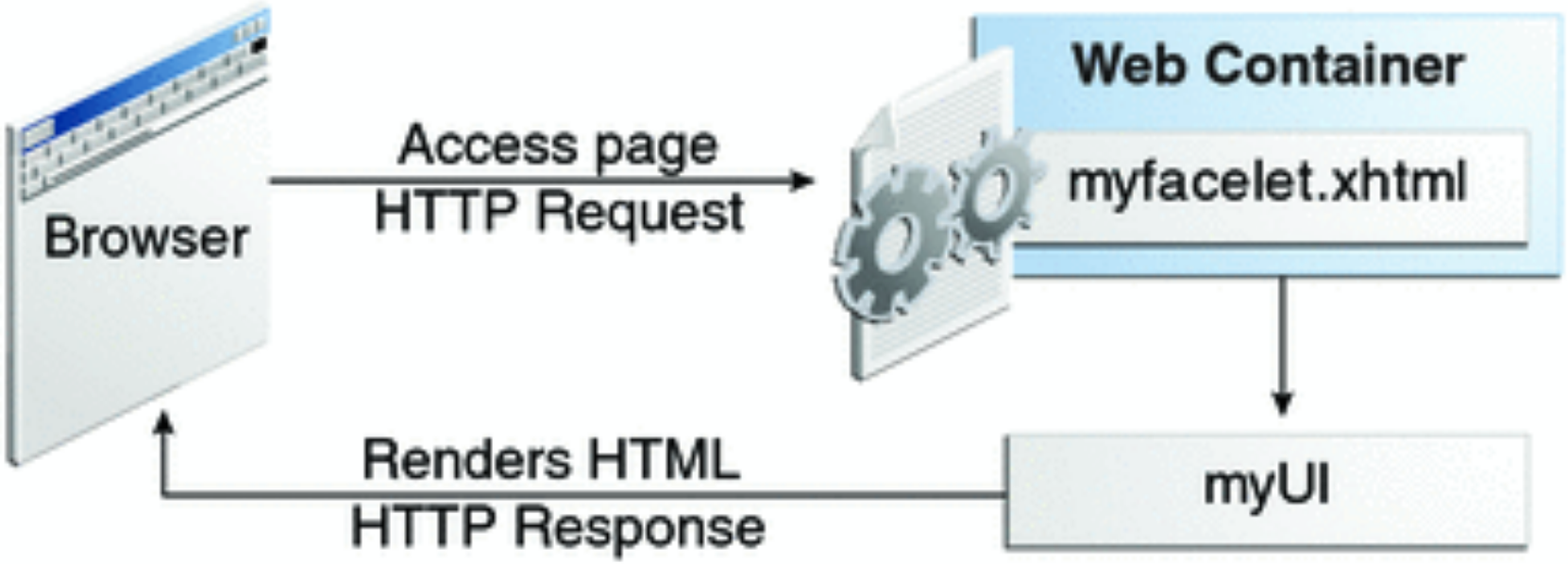
JSF 2.0

- New technolog stack
- „Facelets“ instead of JSP
- Version 2.0 fundamental changes towards JSF 1.2



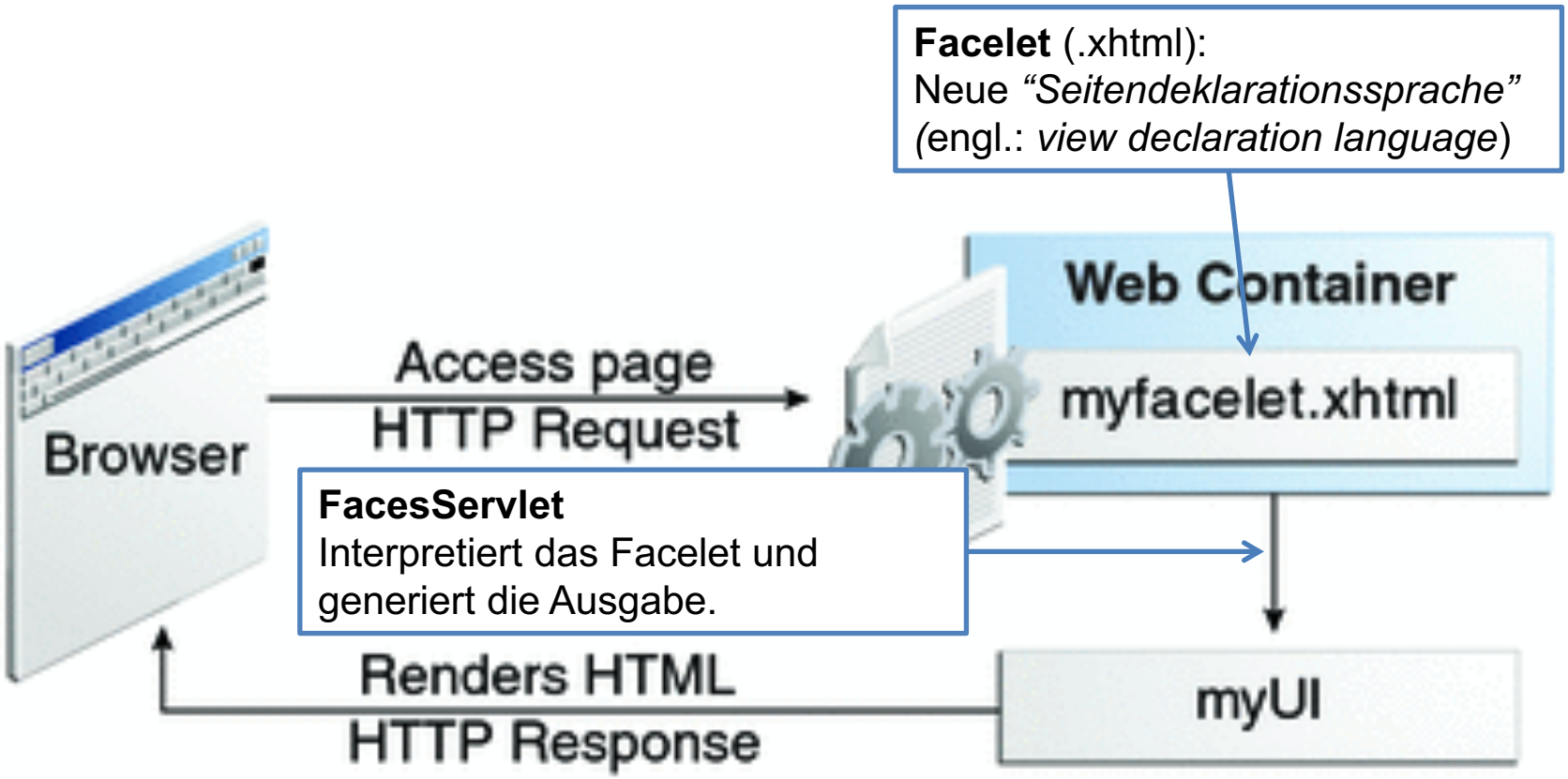
Quelle: http://docs.oracle.com/javaee/6/tutorial/doc/bnapk.html

Modul 7
JavaServer Faces Application



Quelle: http://docs.oracle.com/javaee/6/tutorial/doc/bnapk.html

Modul 7
JavaServer Faces Application



Quelle: http://docs.oracle.com/javaee/6/tutorial/doc/bnapk.html

Modul 7
JSF: Facelet

Bsp.:Hallo-World Facelet

```
<html lang="en"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Facelets Hello World</title>
    </h:head>
    <h:body>
        #{hello.world}
    </h:body>
</html>
```

Quelle: http://docs.oracle.com/javaee/6/tutorial/doc/bnapk.html

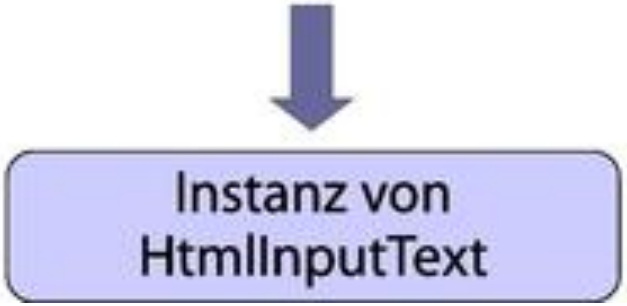
Ab jetzt eine Zusammenfassung nach:
http://jsfatwork.irian.at/book_de

Modul 7
JSF in Schlagworten

Komponente (auch Component, UIComponent oder Control)
Eine Komponente ist ein eigenständiger und wiederverwendbarer Baustein, der zusammen mit anderen Komponenten zum Aufbau einer Seite in einer JSF-Anwendung eingesetzt wird.
Bsp.: Ausgabekomponenten für Texte oder Bilder, etc.
(Seitendeklaration -> Komponenten)

Bsp.:

```
<h:inputText id="firstName" value="#{customer.firstName}"/>
```

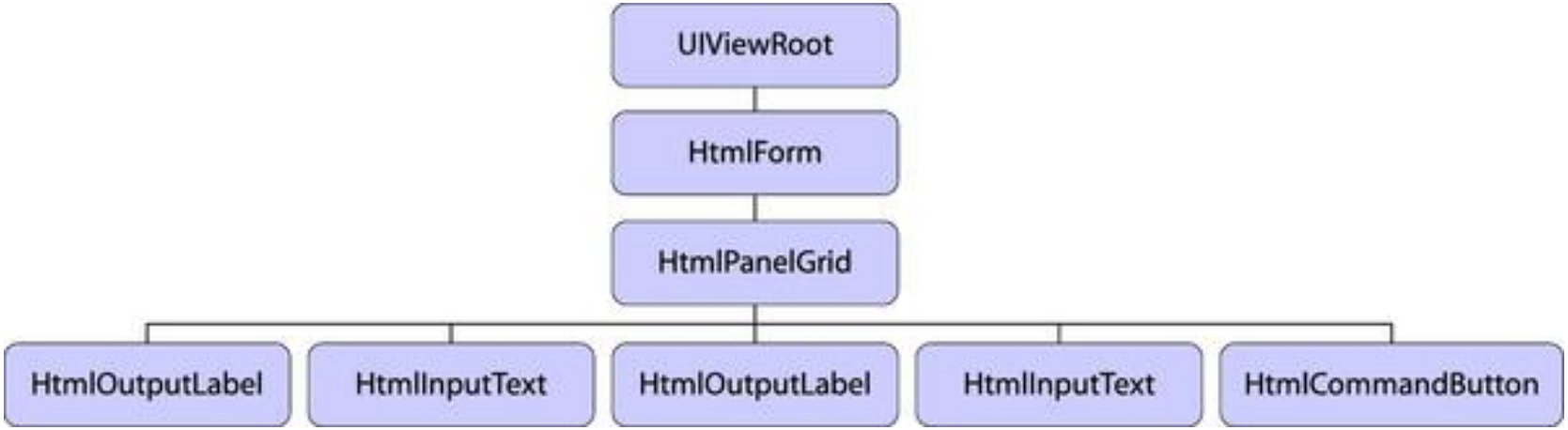


Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Ansicht und Komponentenbaum

Alle Komponenten einer Seite werden zusammen als Ansicht (engl.: *view*) bezeichnet und sind in Form eines Komponentenbaums miteinander verknüpft.

Bsp.:

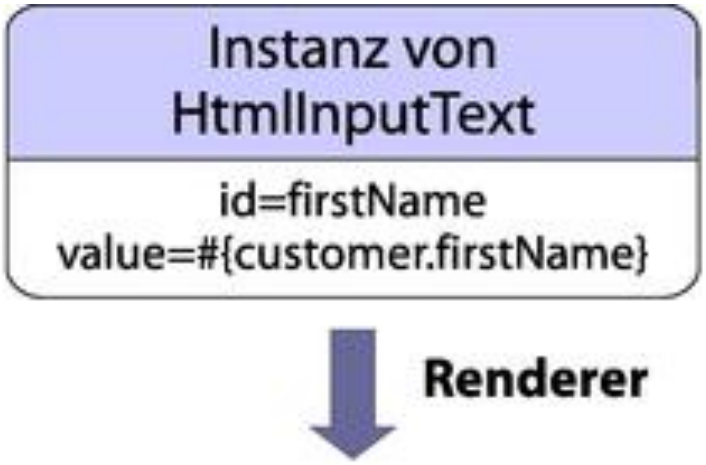


Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Renderer

Die eigentliche Ausgabe der Komponente und ihrer Daten und das Entnehmen der vom Benutzer am Client geänderten Daten wird vom *Renderer* erledigt.

Bsp.:



Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF in Schlagworten

Seitendeklarationssprache (auch View Declaration Language, VDL)

Eine Seitendeklarationssprache (VDL) ist eine Syntax, um Ansichten beziehungsweise Seiten für JSF zu deklarieren. In JSF 2.0: Facelets (.xhtml-Datei).

editCustomer.xhtml

```
<!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">
<h:head>
  <title>MyGourmet - Edit Customer</title>
</h:head>
<h:body>
  <h1><h:outputText value="MyGourmet"/></h1>
  <h2><h:outputText value="Edit Customer"/></h2>
  <h:form id="form">
    <h:panelGrid id="grid" columns="2">
      <h:outputLabel value="First Name:" for="firstName"/>
      <h:inputText id="firstName"
          value="#{customer.firstName}"/>
      <h:outputLabel value="Last Name:" for="lastName"/>
      <h:inputText id="lastName"
          value="#{customer.lastName}"/>
      <h:commandButton id="save" action="#{customer.save}"
        value="Save"/>
    </h:panelGrid>
  </h:form>
</h:body>
</html>
```

at/book_de/introduction.html

Modul 7
JSF in Schlagworten

Seitendeklarationssprache (auch View Declaration Language, VDL)

Eine Seitendeklarationssprache (VDL) ist eine Syntax, um Ansichten beziehungsweise Seiten für JSF zu deklarieren. In JSF 2.0: Facelets (.xhtml-Datei).

```
<?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">
<head>
  <title>MyGourmet - Show Customer</title>
</head>
<body>
  <h1><h:outputText value="MyGourmet"/></h1>
  <h2><h:outputText value="Show Customer"/></h2>
  <h:panelGrid id="grid" columns="2">
    <h:outputText value="First Name:"/>
    <h:outputText value="#{customer.firstName}"/>
    <h:outputText value="Last Name:"/>
    <h:outputText value="#{customer.lastName}"/>
  </h:panelGrid>
  <h:outputText value="Customer saved successfully!"/>
</body>
</html>
```

showCustomer.xhtml

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF in Schlagworten

Managed-Beans (Backing-Beans, Datenmodell, Geschäftsobjekte)

Simple Java-Klassen, auch POJOs (Plain Old Java Objects) genannt, die dem *JavaBeans - Standard* genügen müssen. In den Managed-Beans liegen die eigentlichen Werte zum Befüllen der Komponenten.

(Sollte serialisierbar sein, damit Session vom servlet-Container ggf. auf HDD zwischen-gespeichert werden können.)

```
package at.irian.jsfatwork.gui.page;

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

@ManagedBean
@SessionScoped
public class Customer implements Serializable {
  private String firstName;
  private String lastName;

  public String getFirstName() {
    return firstName;
  }
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  public String getLastName() {
```

JavaBean is just a standard

1. All properties private (use getters/setters)
2. A public no-argument constructor
3. Implements Serializable.

```
}
```

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF in Schlagworten

Managed-Beans (Backing-Beans, Datenmodell, Geschäftsobjekte)

Simple Java-Klassen, auch POJOs (Plain Old Java Objects) genannt, die dem *JavaBeans - Standard* genügen müssen. In den Managed-Beans liegen die eigentlichen Werte zum Befüllen der Komponenten.

(Sollte serialisierbar sein, damit Session vom servlet-Container ggf. auf HDD zwischen-gespeichert werden können.)

```
package at.irian.jsfatwork.gui.page;

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

@ManagedBean
@SessionScoped
public class Customer implements Serializable {
  private String firstName;
  private String lastName;

  public String getFirstName() {
    return firstName;
  }
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  public String getLastName() {
```

Serializable
Diese Schnittstelle enthält keine Methoden und ist nur eine Markierungsschnittstelle (engl. marker interface).

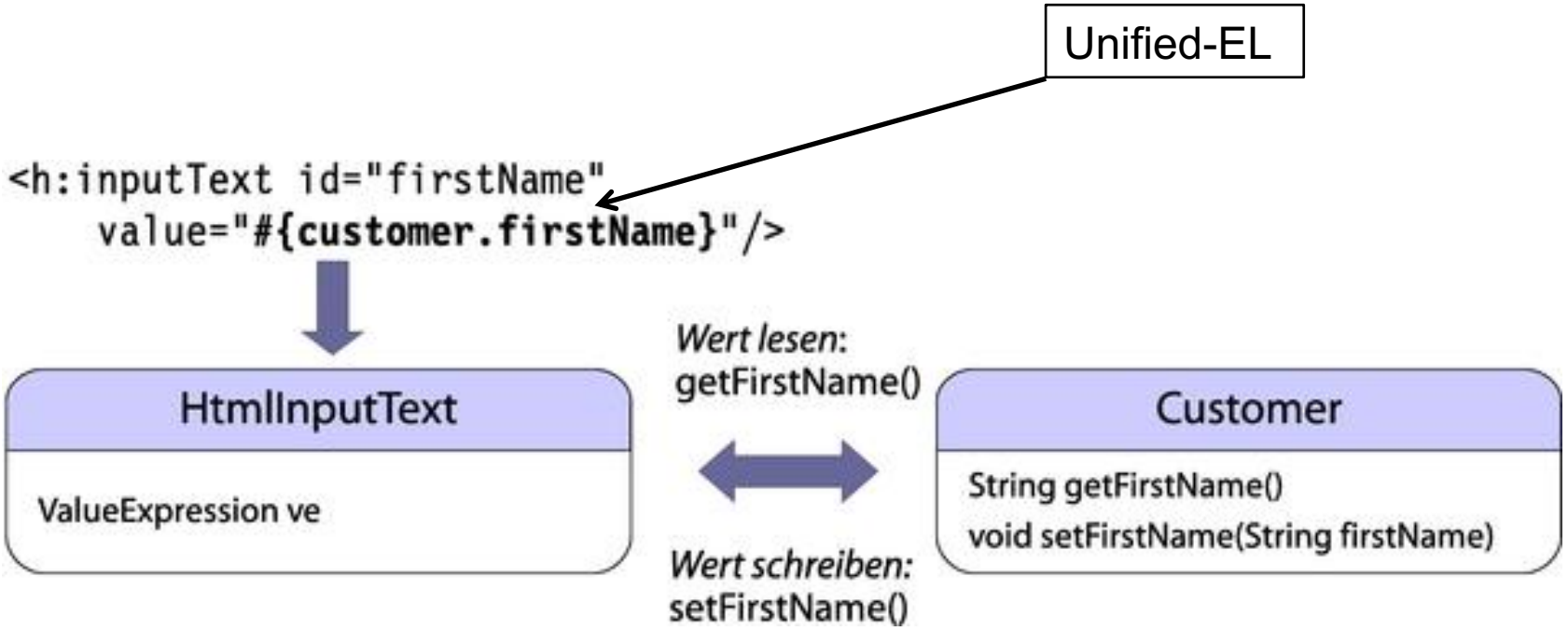
Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF in Schlagworten

Managed-Beans
(Backing-Beans)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

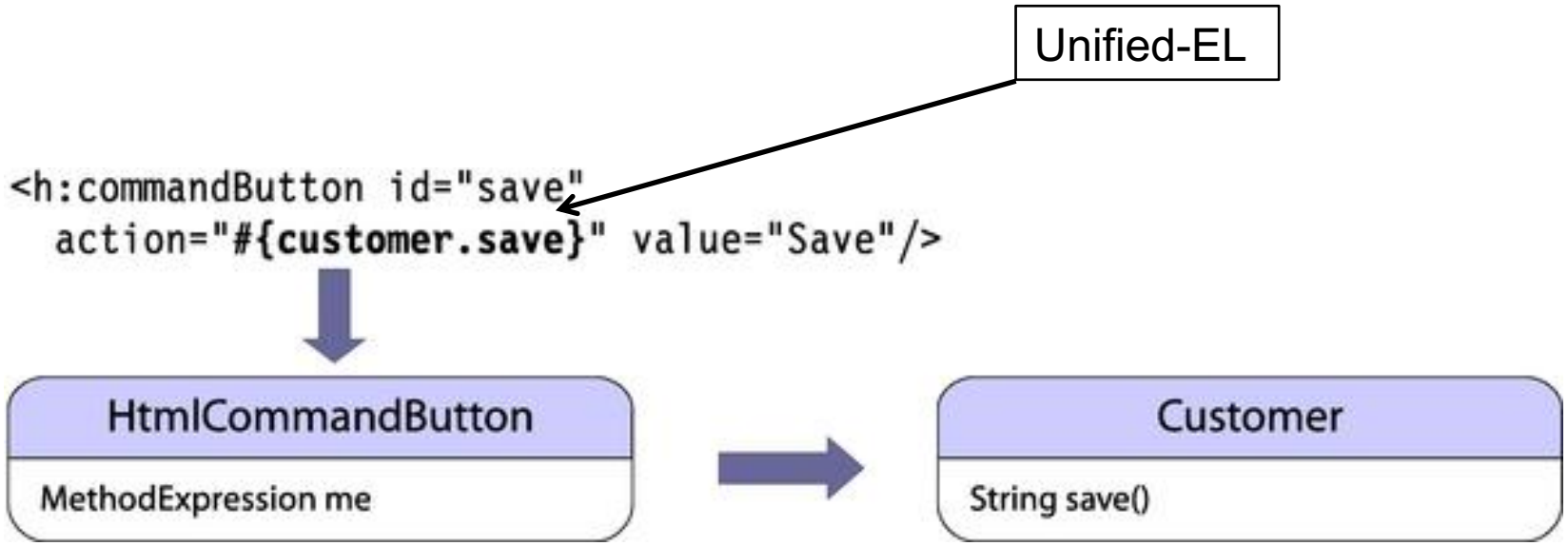
Unified Expression Language (auch Unified-EL)
Mit *Value-Expressions* werden Eigenschaften von Managed-Beans mit Attributen von Komponenten verbunden (Lesen & Schreiben von Beans.)



Bsp.: Gourment01->showCustomer.xhtml

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Unified Expression Language (auch Unified-EL)
Method-Expressions erlauben das Verknüpfen von Komponenten mit Methoden. Ein Konzept, das zum Beispiel bei der Validierung von Benutzereingaben und der Ereignisbehandlung eingesetzt wird.



Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF in Schlagworten

Navigation und Aktionen (navigation-rules, action)

JSF verwendet den Rückgabewert spezieller Methoden, nämlich der "Action"-Methoden, um eine Weiterleitung von einer auf die nächste Seite zu veranlassen. Dieser Rückgabewert kann der Name einer Navigationsregel (faces-config.xml) oder direkt der Name der nächsten Seite sein.

editCustomer.xhtml

```
…
<h:commandButton id="save" value="Save" action="#{customer.save}"/>
…
```

```
public class Customer {
…
  public String save() {
    return "/showCustomer.xhtml";
  }
…
}
```

Bsp.: Gourment01->editCustomer.xhtml

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Ereignisse und Ereignisbehandlung

Reaktion auf eine betätigte Schaltfläche oder eine Wertänderung. Jede Komponente kann Ereignisse auslösen und jede Managed-Bean kann als Interessent für solche Ereignisse registriert werden.

```
<h:selectBooleanCheckbox onclick="this.form.submit()"
    value="#{customer.useCreditCard}“ immediate="true"

valueChangeListener="#{customer.useCreditCardChanged}"/>
```

Bsp.: M07_Events

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Konverter

Konvertierung von Datentypen in eine Zeichenketten, die vom Browser dargestellt werden können und umgekehrt.

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

  String getAsString(FacesContext context,
      UIComponent component,
      Object value) throws ConverterException;
}
```

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Validator

Überprüfen der Gültigkeit der vom Benutzer eingegebenen Werte und unterbinden des Zurückschreibens von ungültigen Werten ins Modell.

```
public class Customer {
  @NotNull @Min(value = 1000) @Max(value = 99999)
  private Integer zipCode;

  @NotBlank
  private String city;

  @NotBlank
  private String street;

  ...
}
```

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Nachrichten

Sollten bei der Abarbeitung von Methoden oder beim Validieren und Konvertieren von Werten Fehler auftreten, müssen diese Fehler dem Benutzer angezeigt werden. Validierungs- und Konvertierungsfehler werden in JSF in Nachrichten umgewandelt, die dann auf der Seite angezeigt werden können.

```
if (date.after(new Date())) {
  FacesMessage msg = new FacesMessage(
      FacesMessage.SEVERITY_ERROR,
      "Birthday is in the future.", null);
  throw new ValidatorException(msg);
}
```

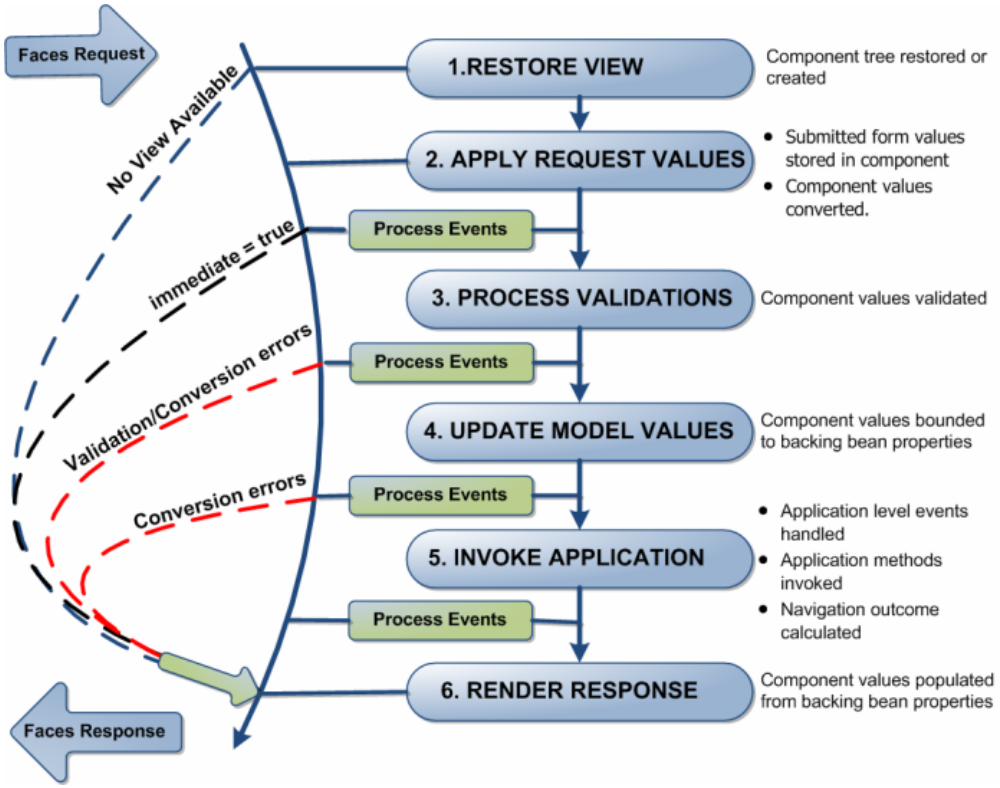
editCustomer.xhtml

```
  <h:messages showDetail="false" showSummary="true"
                                                layout="table"/>
```

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Lifecycle

Das zeitliche Zusammenspiel dieser einzelnen Objekte in der JSF-Technologie ist sehr genau geregelt, und zwar im "Request Processing Lifecycle" genannten Lebenszyklus einer HTTP-Anfrage.



Quelle: http://www.developersbook.com/jsf/jsf-tutorials/jsf-tutorials.php

Modul 7
JSF: Beispiel

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF 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:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html">
<h:head>
  <title>MyGourmet - Edit Customer</title>
</h:head>
<h:body>
  <h1><h:outputText value="MyGourmet"/></h1>
  <h2><h:outputText value="Edit Customer"/></h2>
  <h:form id="form">
    <h:panelGrid id="grid" columns="2">
      <h:outputLabel value="First Name:" for="firstName"/>
      <h:inputText id="firstName"
          value="#{customer.firstName}"/>
      <h:outputLabel value="Last Name:" for="lastName"/>
      <h:inputText id="lastName"
          value="#{customer.lastName}"/>
      <h:commandButton id="save" action="#{customer.save}"
        value="Save"/>
    </h:panelGrid>
  </h:form>
</h:body>
</html>
```

editCustomer.xhtml

Seitendeklaration

Komponenten

EL
(*value expression*)

EL
(*method expression*)

Action / Navigation

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF Beispiel

```
package at.irian.jsfatwork.gui.page;

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

@ManagedBean
@SessionScoped
public class Customer {
  private String firstName;
  private String lastName;

  public String getFirstName() {
    return firstName;
  }
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  public String getLastName() {
    return lastName;
  }
  public void setLastName(String lastName) {
    this.lastName = lastName;
  }
  public String save() {
    return "/showCustomer.xhtml";
  }
}
```

← **ManagedBean**

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF Lifecycle

1) Aus JSP HTML generieren und HTML an Browser senden

2) Servlet wird von JSP-Seite aufgerufen. Servelt holt sich die request-Parameter
(`String userStr = request.getParamater("name");`)

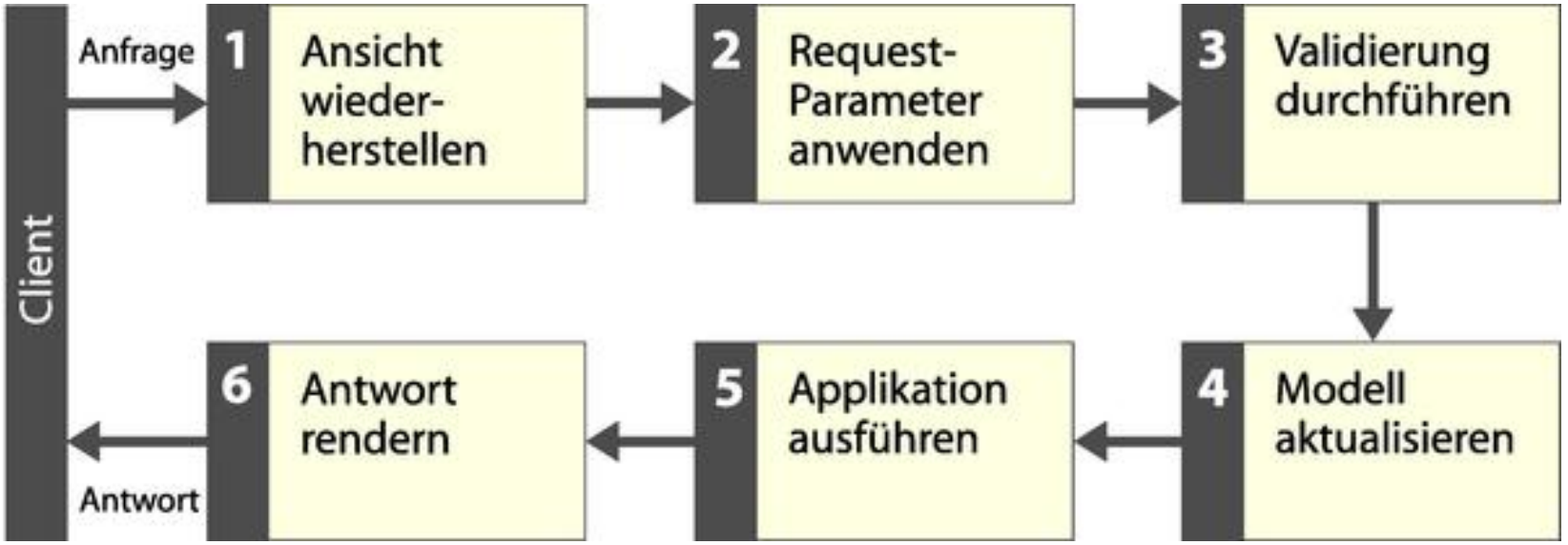
3) Überprüfen der Parameter
(`if (userStr==null) { ... };`)

4) Übertragen der Daten in die entsprechende JavaBean (Model-Update)
(`userBean.name = userStr; …`)

5) Ausführen von Methoden (Business-Logik)
(`userBean.checkName(...); userBean.saveNameinDB(...);`)
Ermitteln und Aufrufen der darzustellenden Seite.
(`RequestDispatcher rd =`
`request.getRequestDispatcher("welcome.jsp");`)

6) Weiter bei (1)

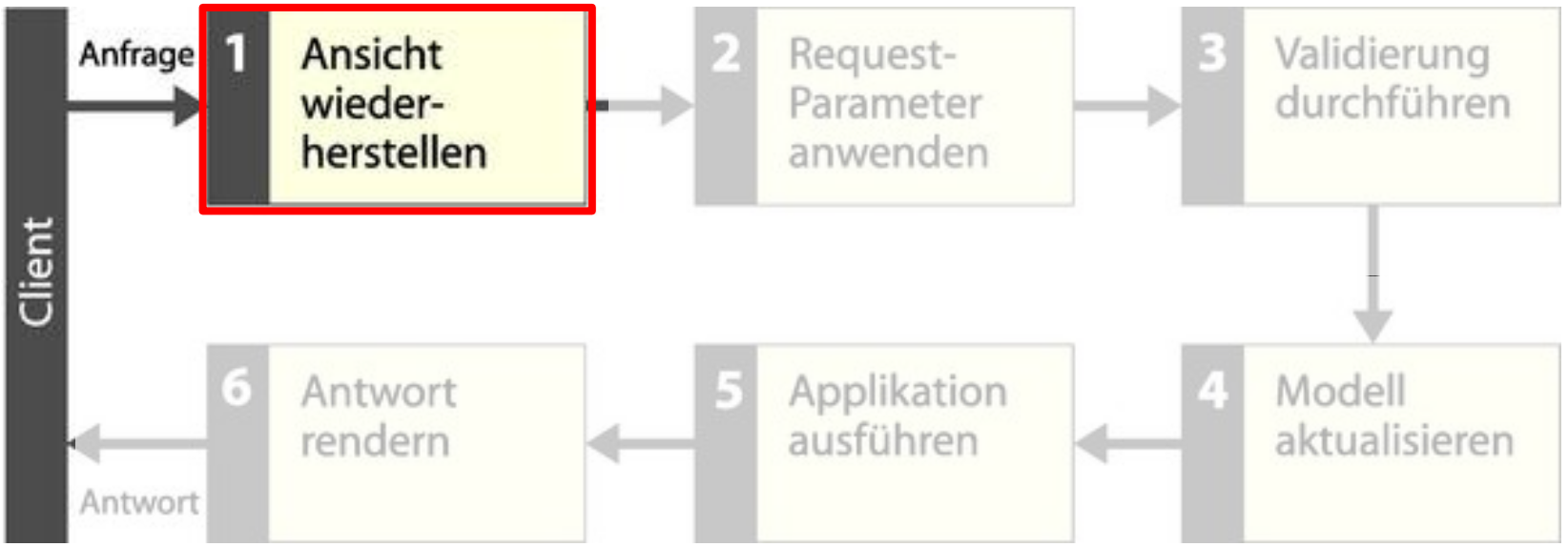
Modul 7
JSF Lifecycle



Phase 1: Ansicht wiederherstellen (Restore View)
Phase 2: Request-Parameter anwenden (Apply Request Values)
Phase 3: Konvertierung und Validierung durchführen (Process Validations)
Phase 4: Modell aktualisieren (Update Model Values)
Phase 5: Applikation ausführen (Invoke Application)
Phase 6: Antwort rendern (Render Response)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF Lifecycle: Phase 1



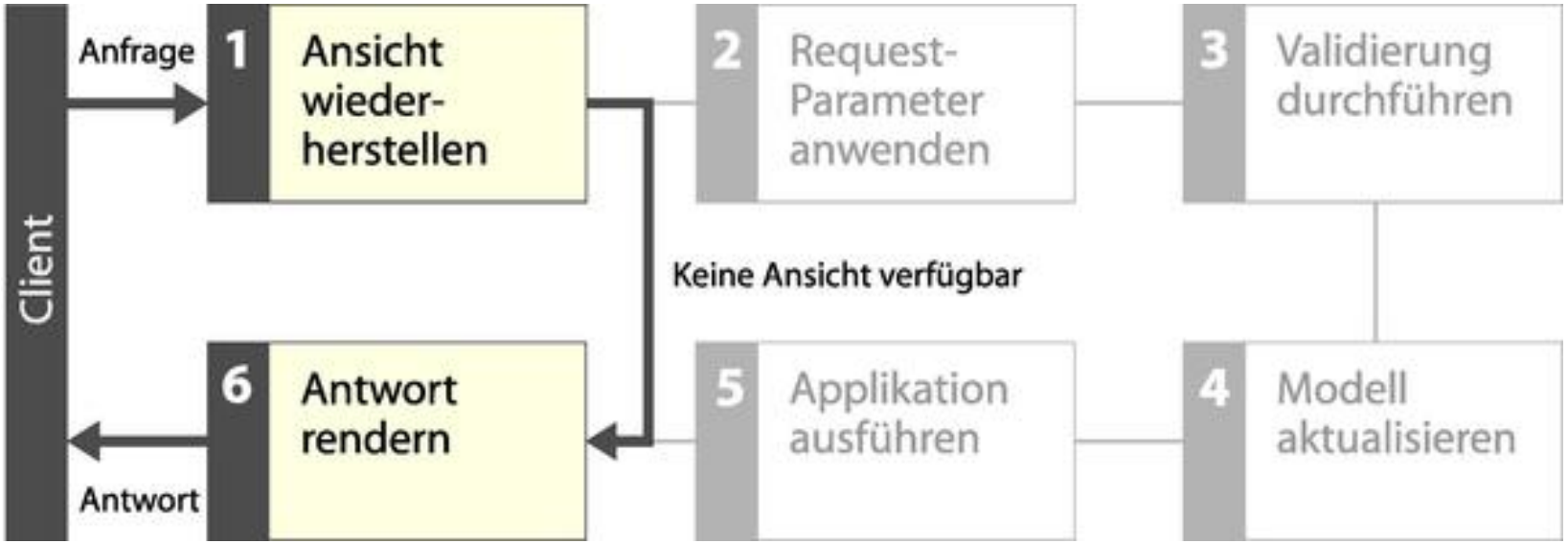
Initialer Aufruf: Erstellung des Komponentenbaums, weiter in (6)

oder

Erneuter Aufruf: Laden des bereits erstellten Komponentenbaums

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF Lifecycle: Phase 1 – Initaler Aufruf

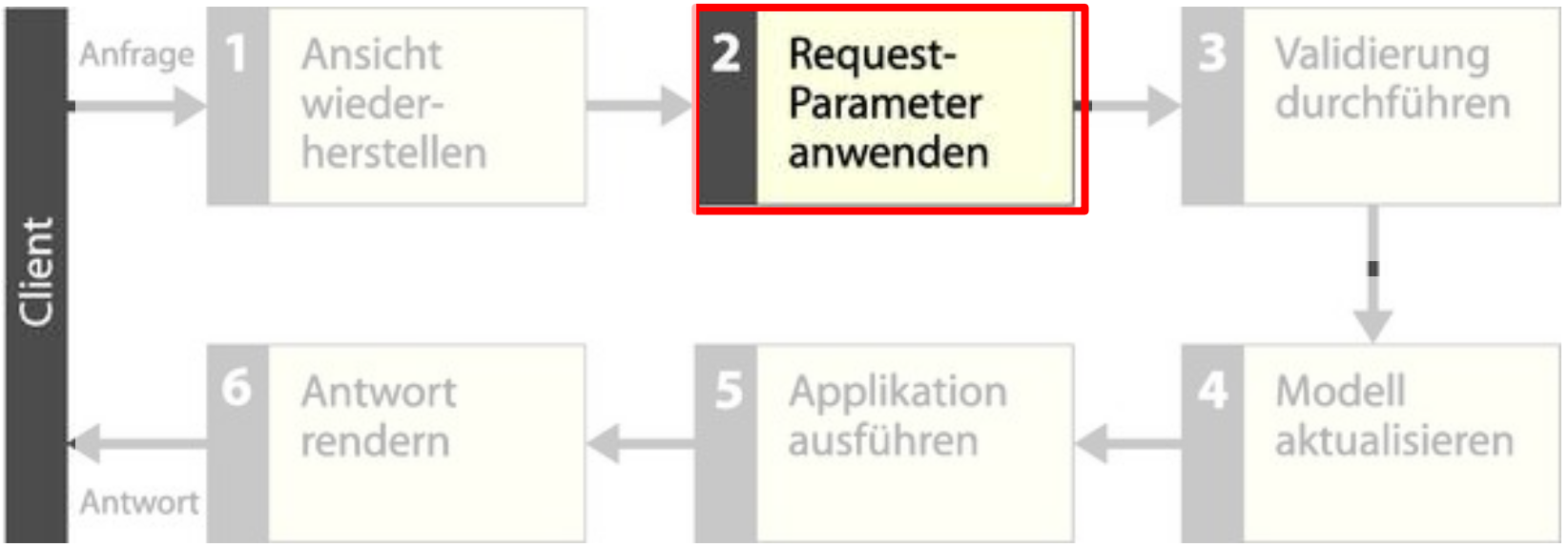


Initialer Aufruf: Erstellung des Komponentenbaums, weiter mit Phase 6

(Somit: Keine direkte Übergabe von get-Parametern beim ersten Aufruf möglich. Dafür Kap. 4.4 „Bookmarks und GET-Anfragen in JSF“ in irian.at)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

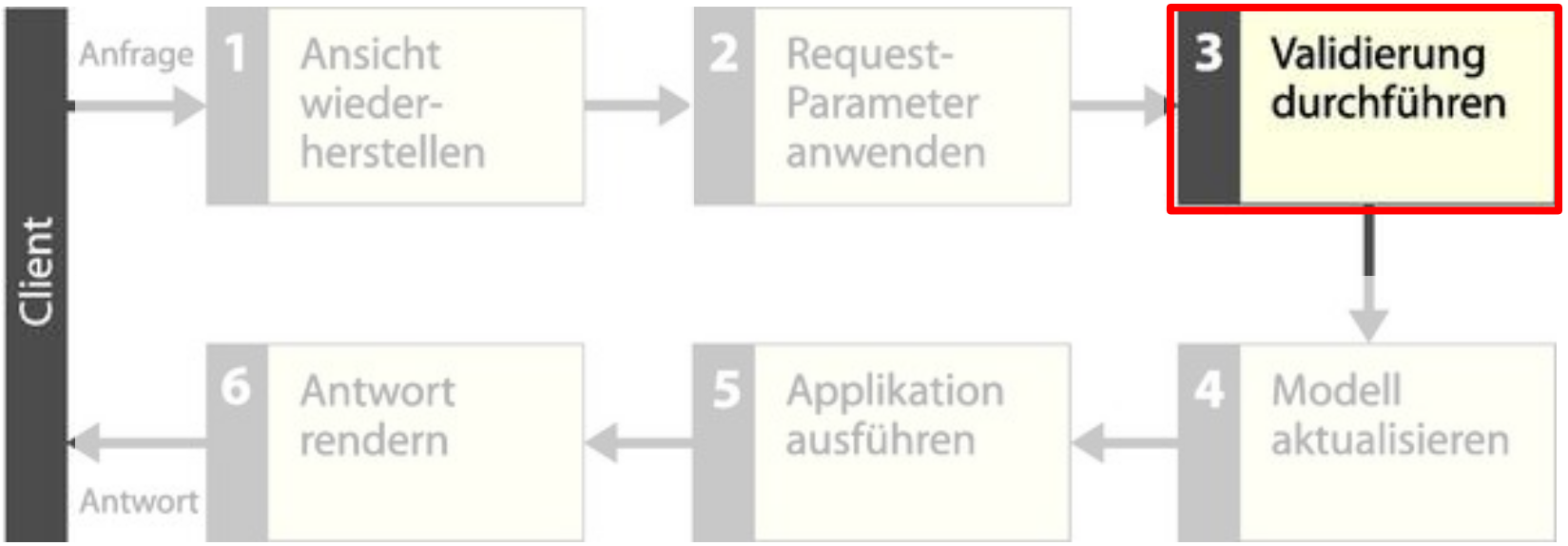
Modul 7
JSF Lifecycle: Phase 2



- Übernahme der vom Benutzer eingetragenen Werte
- Extraktion der Parameter aus dem Request (üblicherweise ein HTTP-Post-Request) und Zuweisung zu den entsprechenden JSF-Komponenten (z.B. Eingabefeldern) als s.g. „*submitted-values“ (String)*
- Weiter in (3)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

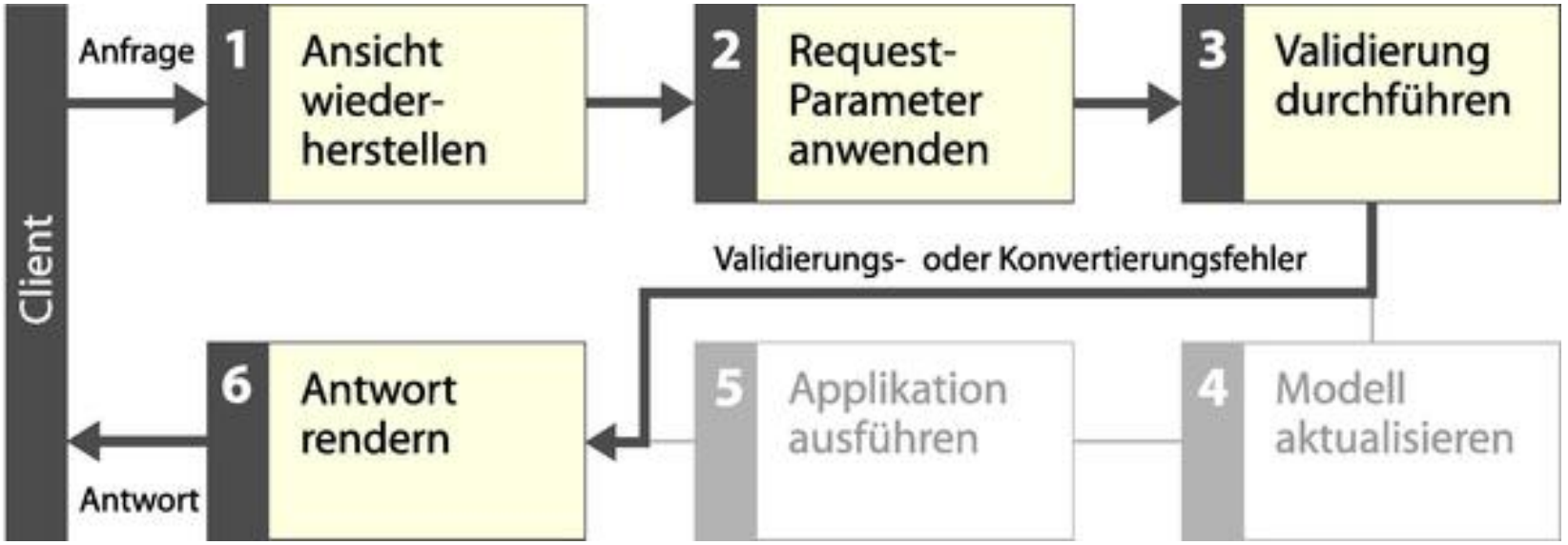
Modul 7
JSF Lifecycle: Phase 3



- Konvertieren der „*submitted values*“ (Strings) in den Datentyp des Feldes des dahinterliegenden ManagedBeans als „*local value*“
- Aufruf von Event-Listnern
- Validierung des „*submitted values*“ (z.B. @NotNull)
- Wenn Validierung ok, dann weiter bei (4), sonst zu (6)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

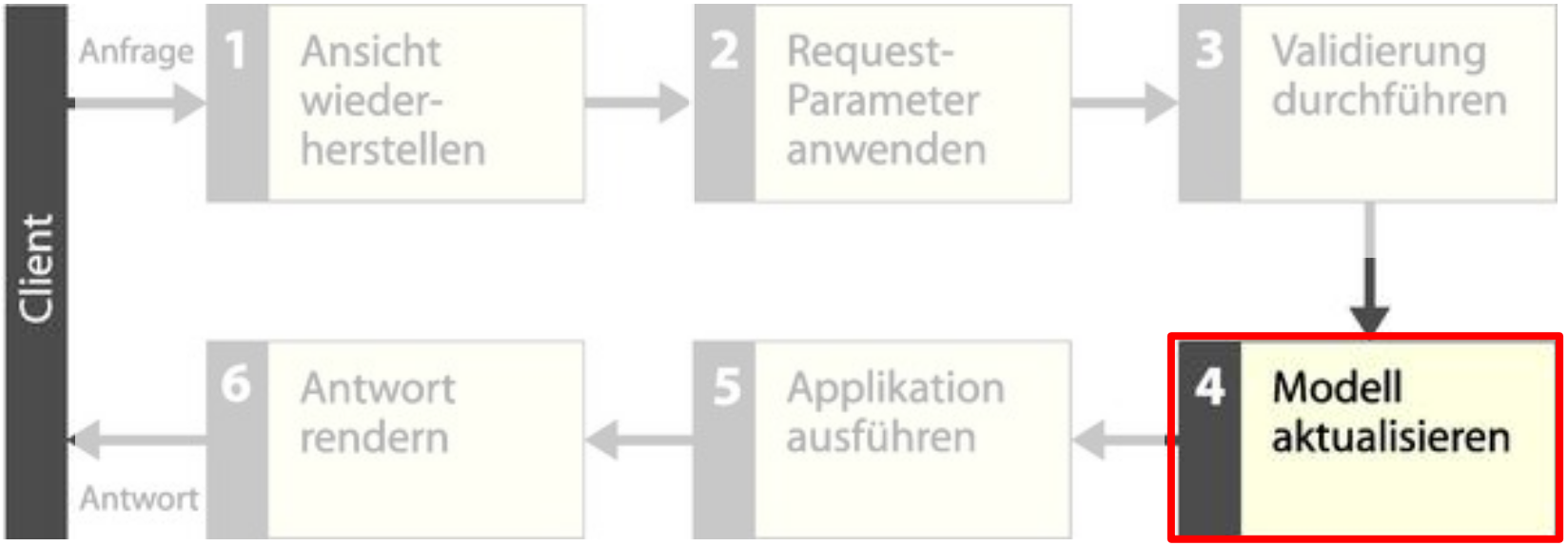
Modul 7
JSF Lifecycle: Phase 3 - Validierungsfehler



- Konvertieren der „*submitted values*“ (Strings) in den Datentyp des Feldes des dahinterliegenden ManagedBeans als „*local value*“
- Aufruf von Event-Listener (**`valueChangeListener`**)
- Validierung des „*submitted values*“ (z.B. @NotNull)
- Wenn Validierung ok, dann weiter bei (4), sonst zu (6)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

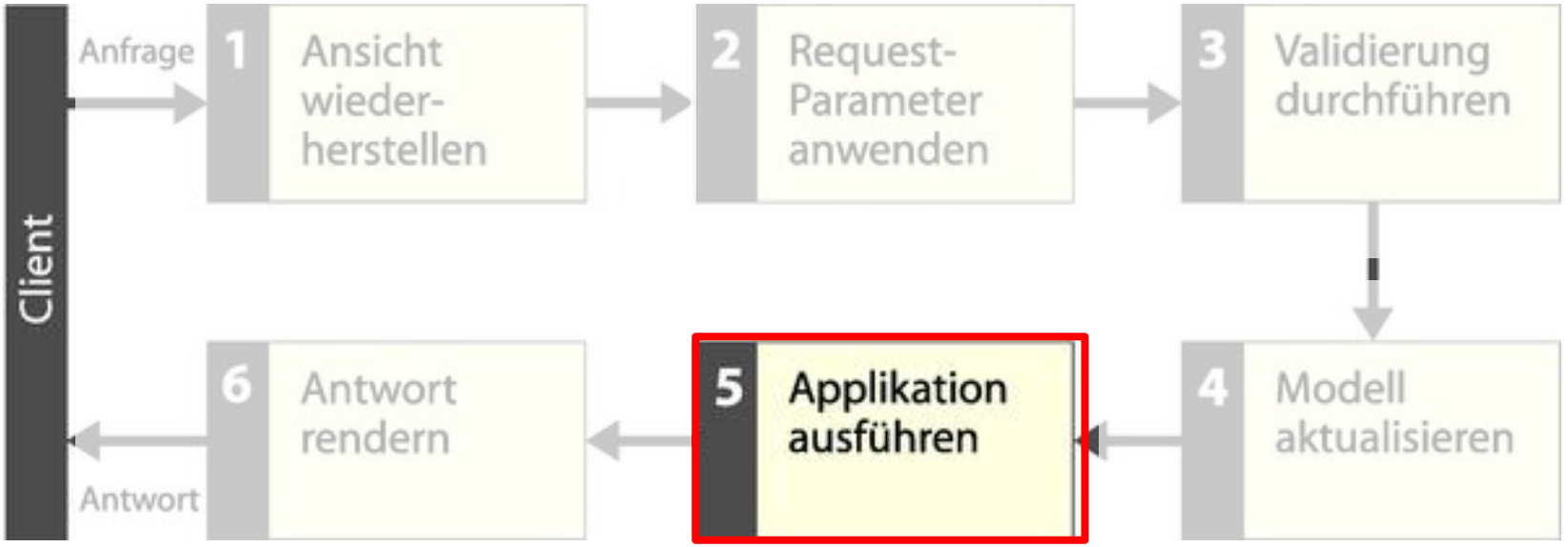
Modul 7
JSF Lifecycle: Phase 4



- Übertragen des „*local values*“ in das Feldes des dahinterliegenden ManagedBeans (z.B. **`#{customer.firstName}`**)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

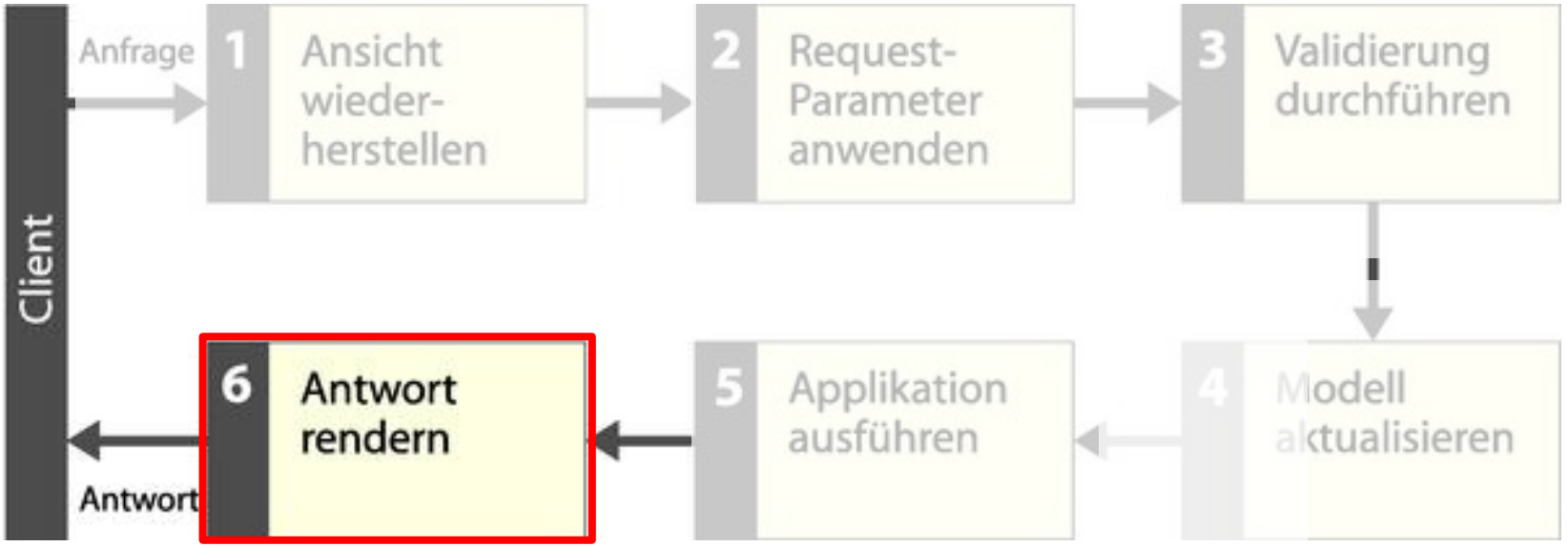
Modul 7
JSF Lifecycle: Phase 5



- Ausführen von Action-Events (**`action`** bzw. **`actionlistener`** durch Drücken eines Buttons **`h:commandButton`** oder Links **`h:commandLink`**)
- Das **`return`**-Wert der **`action`**-methode definiert, welche Seite als nächstes dargestellt wird. (Bei **`null`** die bisherige Seite).

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

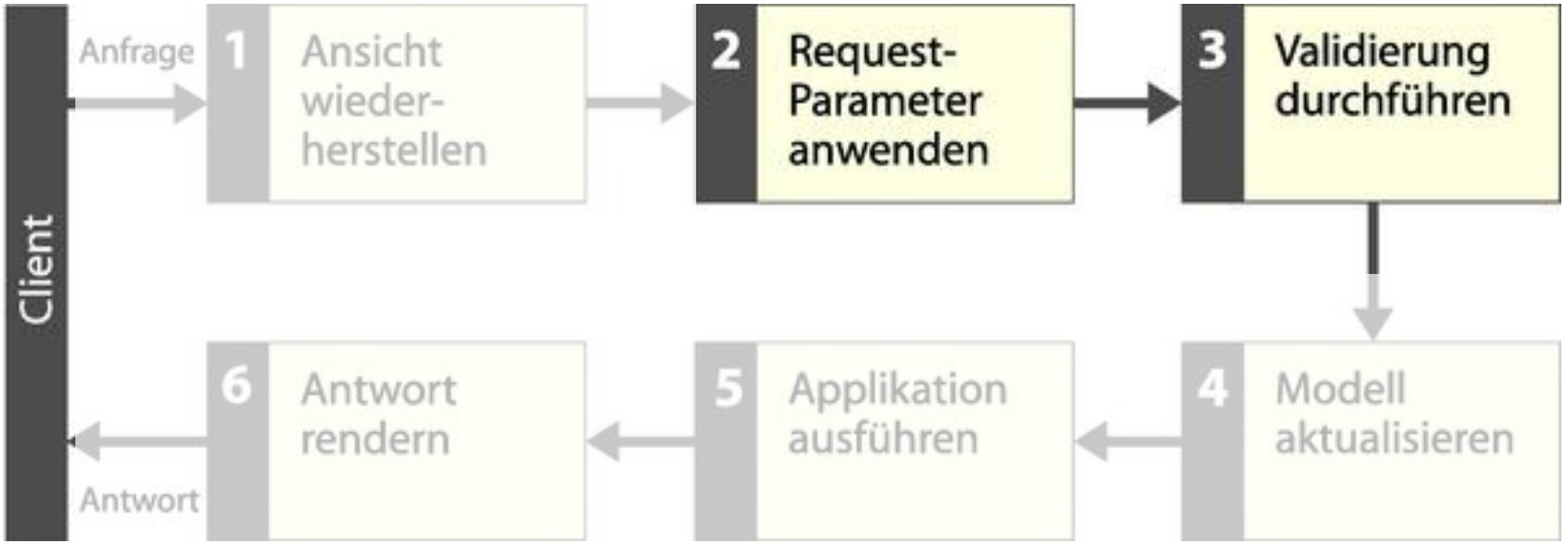
Modul 7
JSF Lifecycle: Phase 6



- Rendern (HTML-Erstellung) des darzustellenden Komponentenbaus (**`return`**-Wert der **`action`**-Methode)
- Speichern des Komponentenbaums

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

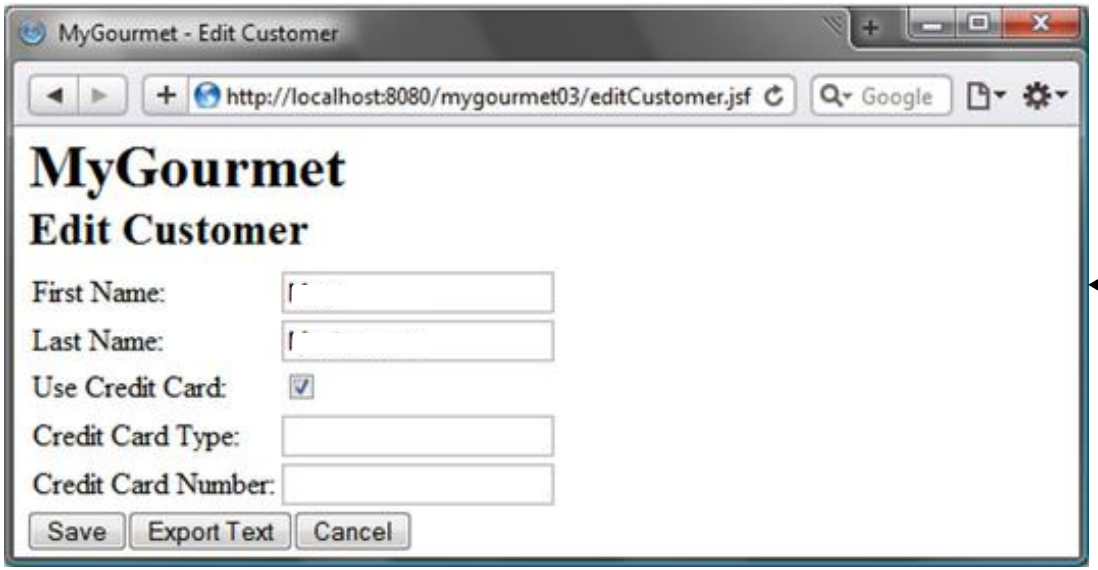
Modul 7
JSF Lifecycle: Ändern des LifeCycles



- In (3) werden **alle** Komponenten „überprüft“ (z.B. `@NotNull`)
- Es soll aber ggf. eine Änderung durchgeführt werden, sobald sich nur eine Komponente ändert.
  (Bsp.: Anklicken einer CheckBox führt zur Ansicht einer zusätzlichen Komponente)

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

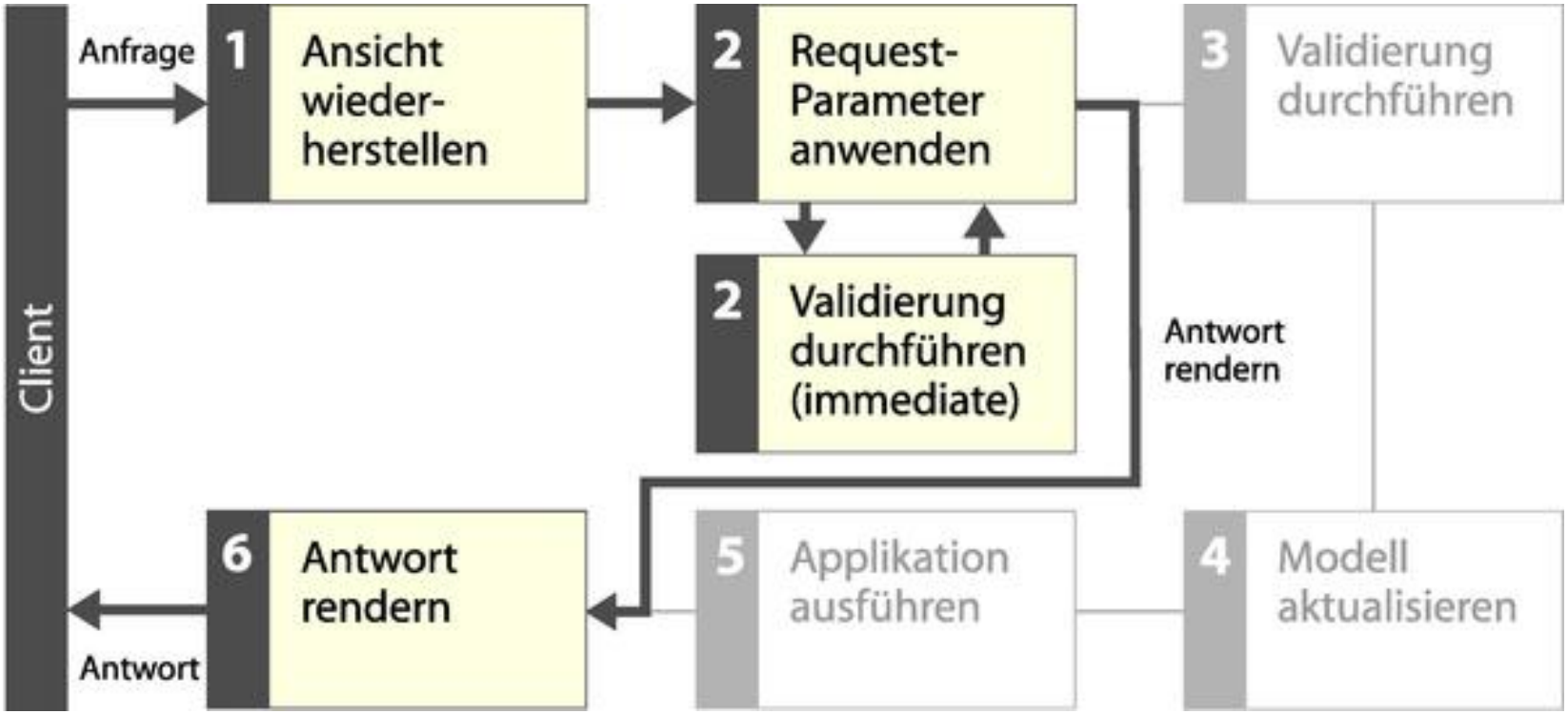
Modul 7
JSF Lifecycle: Ändern des LifeCycles



Erneuter Aufruf der Seite, obwohl *First Name* und *Last Name* leer sind.

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF Lifecycle: Ändern des LifeCycles



- **`immidiate`**-Attribut in der Komponente ist **`true`** :
  a) Validierung wird sofort für diese eine Komponente durchgeführt
  b) EventListener (**`valueChangeListener`**) wird direkt nach (2) aufgerufen

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF Lifecycle: Beispiel mit EventListener

```
<h:inputText value="#{customer.lastName}"
    required="true"/>
<h:selectBooleanCheckbox onclick="this.form.submit()"
    value="#{customer.useCreditCard}" immediate="true"
    valueChangeListener="#{customer.useCreditCardChanged}"/>
<h:inputText value="#{customer.creditCardType}"
    rendered="#{customer.useCreditCard}"/>
```

```
public void useCreditCardChanged(ValueChangeEvent ev) {
  Boolean useCreditCard = (Boolean) ev.getNewValue();
  if (useCreditCard != null) {
    this.useCreditCard = useCreditCard;
  }
  FacesContext.getCurrentInstance().renderResponse();
}
```

springt zu Phase 6

Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
JSF Lifecycle: Beispiel **myGourmet02**

```
…
 <h:panelGrid id="grid" columns="2">
            <h:outputLabel value="First Name:“ for="firstName"/>
            <h:inputText id="firstName" value="#{customer.firstName}"
                    required="true"/>
            <h:outputLabel value="Last Name:" for="lastName"/>
            <h:inputText id="lastName" value="#{customer.lastName}"
                    required="true"/>
 </h:panelGrid>
…

<h:commandButton id="cancel" value="Cancel"
        immediate="true" action="/cancelled.xhtml"/>
…
```

springt zu Phase 6

Quelle: http://jsfatwork.irian.at/book_de/introduction.html