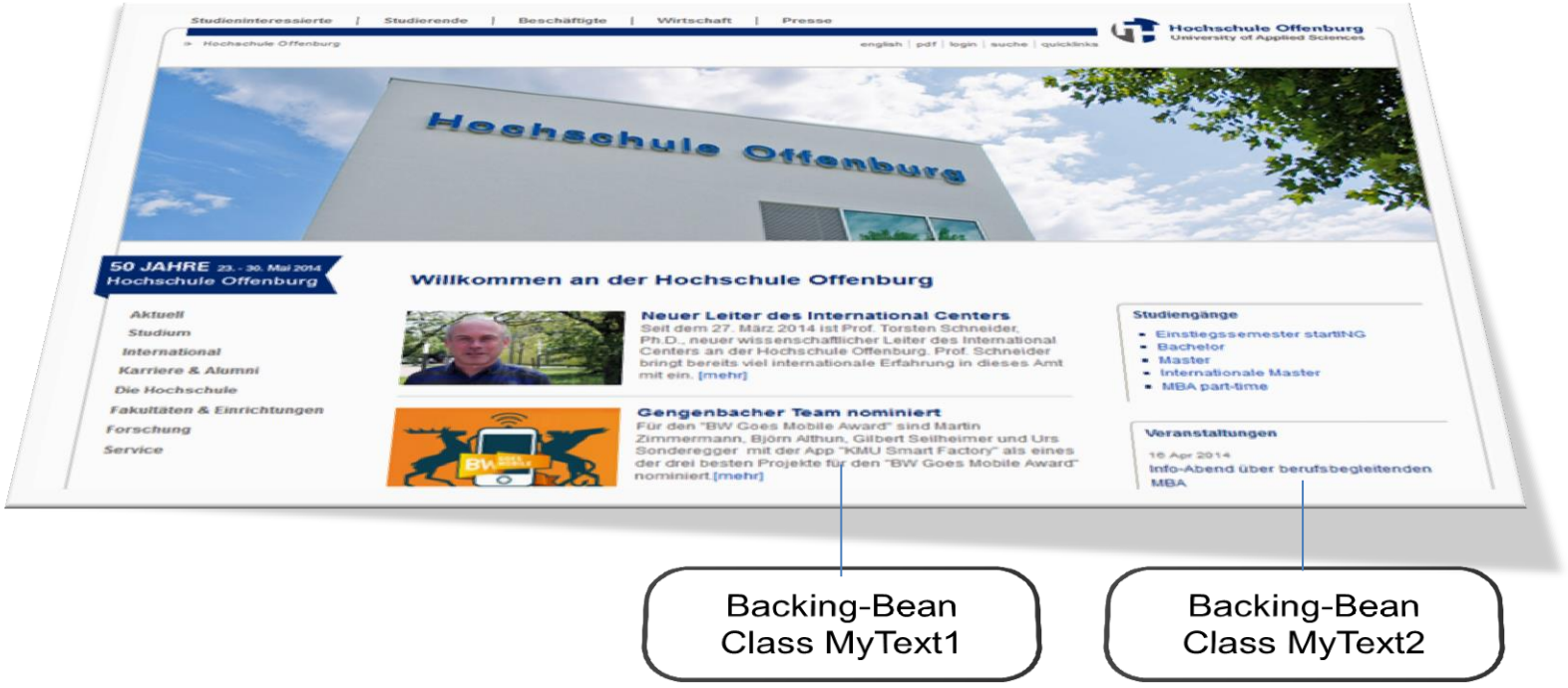
Java Server Faces (JSF)

Modul 7
Managed Beans

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

Modul 7
Managed Beans: Konfiguration

Annotation

```
@ManagedBean
@SessionScoped
public class Customer {
   ...
}
```

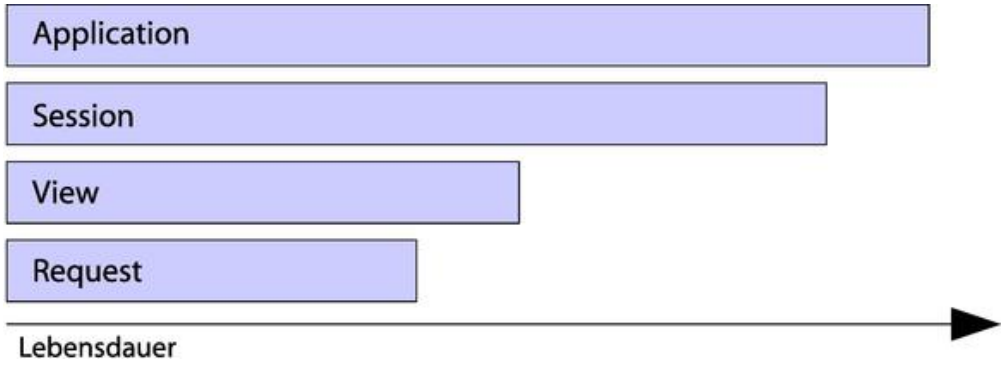
Alternativ: faces-config.xml

```
<managed-bean>
  <managed-bean-name>customer</managed-bean-name>
  <managed-bean-class>
    at.irian.jsfatwork.gui.page.Customer
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

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

Modul 7
Managed Beans: Scopes

- **None-Scope (`none , @NoneScoped`):**
  Die Managed-Bean wird nicht gespeichert und bei jedem Aufruf neu erstellt.
- **Request-Scope (`request , @RequestScoped`):**
  Die Managed-Bean lebt für die Zeitdauer einer HTTP-Anfrage.
- **View-Scope (`view , @ViewScoped`):**
  Die Lebensdauer der Managed-Bean ist an die Ansicht geknüpft, in der sie verwendet wird (z.B. gleiche Seite wird erneut geladen. Insbesondere für AJAX-Anfragen einzusetzen.)
- **Session-Scope (`session , @SessionScoped`):**
  Die Managed-Bean lebt für die Dauer einer Sitzung, in der der Benutzer mit der Anwendung verbunden ist.
- **Application-Scope (`application , @ApplicationScoped`):**
  Für die gesamte Lebensdauer der Anwendung ist nur eine für alle Benutzer gleiche Instanz dieser Managed-Bean vorhanden.



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

Modul 7
Managed Beans: Zugriff in der Unified-EL

```
@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";
  }
}
```

#{customer.firstName}

#{customer.lastName}

#{customer.save}

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

Modul 7
Managed Beans: Zugriff in der Unified-EL

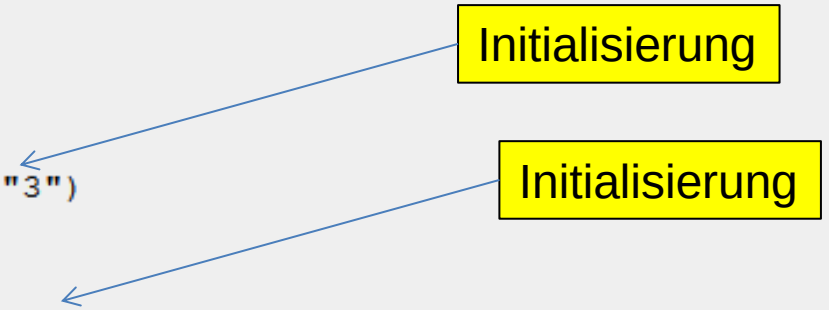
```
@ManagedBean
@SessionScoped
public class Customer {
  private String firstName;
  private String lastName;

  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";
  }
}
```

Keine Business- Logik in getter- oder setter-Methoden implementieren!!!

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

Modul 7
Managed Beans: Managed-Properties

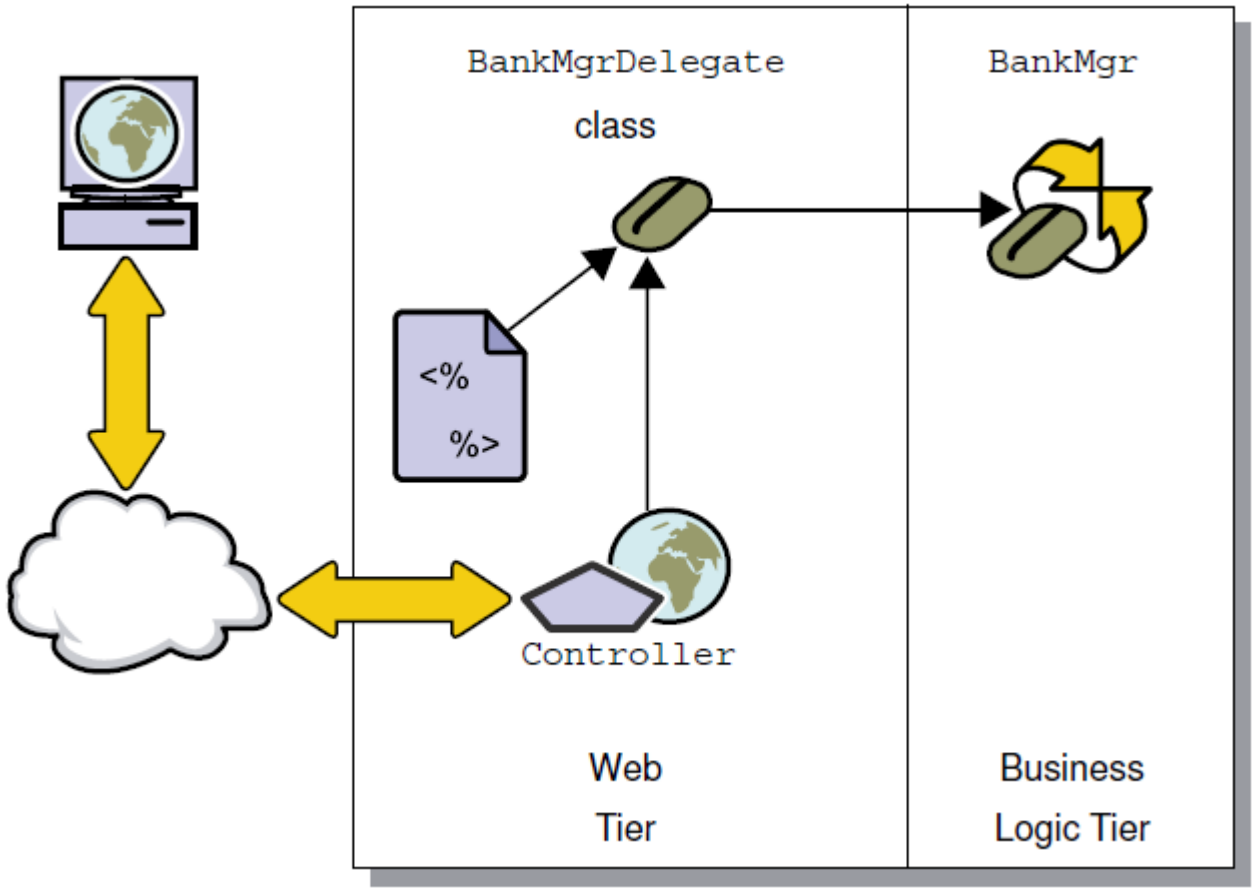
```
@ManagedBean
@SessionScoped
public class Login {
  @ManagedProperty(value = "3")
  private int loginRetries;
  @ManagedProperty(
    value = "#{roleResolver.defaultResolver}")
  private RoleResolver roleResolver;
  ...
}
```



Alternativ via faces-config.xml

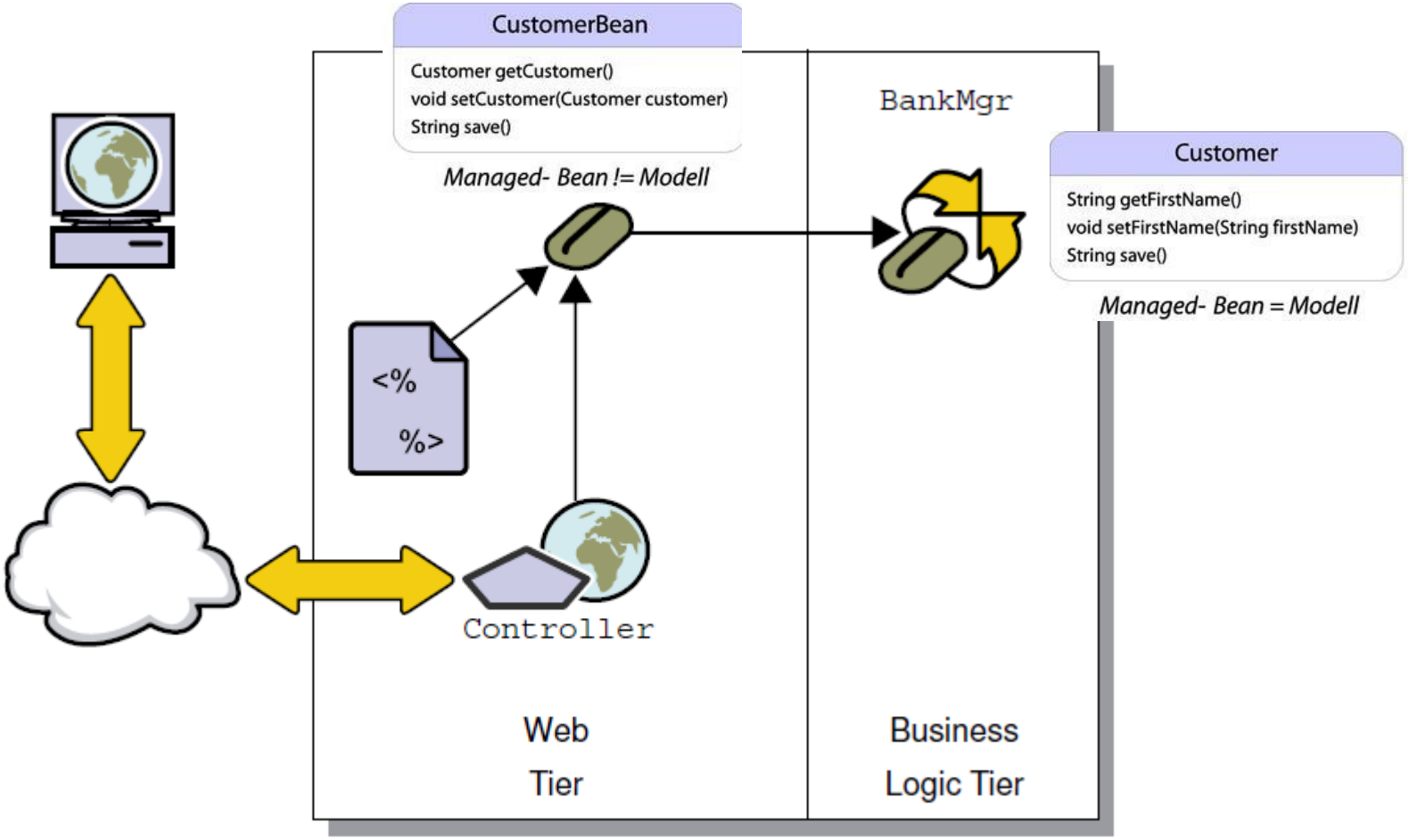
Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
Managed Beans als “Vermittler” (Delegates)



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

Modul 7
Managed Beans als “Vermittler” (Delegates)



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

Modul 7
Unified EL

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

Modul 7
Unified EL - Ausdrücke

- `value="#{myBean.myValue}"` `alt: value="${myBean.myValue}"`

- `value="#{user.username}"`
- `rendered="#{user.username != null}"`
- `value="#{bill.sum * 13,7603}"`
- `style="#{grid.displayed ? 'display:inline;, :display:none;'}"`
- `value=" Hallo Benutzer #{user.username}"`
- `action="#{user.storeUser}"`
- `value="#{mapBean['index']}"`
- `value="#{mapBean[user.username]}"`
- `value="#{listBean[5]}"`

Bsp.: MyGourmet03

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

Modul 7
Unified EL - Ausdrücke

- **`value="{}#{bean.list.size()}"{}`**
  Mit diesem Ausdruck ist es endlich möglich, die Anzahl der Elemente einer Liste ohne Umwege auszulesen.
- **`value="{}#{bean.text.replaceAll(':', '_')}"{}`**
  Dieser Ausdruck ruft auf der Eigenschaft text vom Typ String die Methode replaceAll() auf, um alle Doppelpunkte durch Unterstriche zu ersetzen, und liefert das Ergebnis zurück.
- **`value="{}#{bean.findOrders(otherBean.customer)}"{}`**
  Der Wert dieses Ausdrucks wird über einen Aufruf der Methode findOrders() auf der Bean bean bestimmt. Als Parameter kommt dabei die Eigenschaft customer der Bean otherBean zum Einsatz - auch das ist ohne Probleme möglich.
- **`value="{}#{bean.getName()}"{}`**
  Diese Value-Expression bindet den Rückgabewert der Methode getName() an die Komponente, kann aber nur gelesen werden.

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

Übergabe von Parametern

```
<h:commandLink value="Delete"
   action="#{customerBean.deleteAddress(address)}"/>
```

Facelet

```
public String deleteAddress(Address address) {
  // Adresse löschen und Zeichenkette für Navigation zurückgeben
}
```

ManagedBean

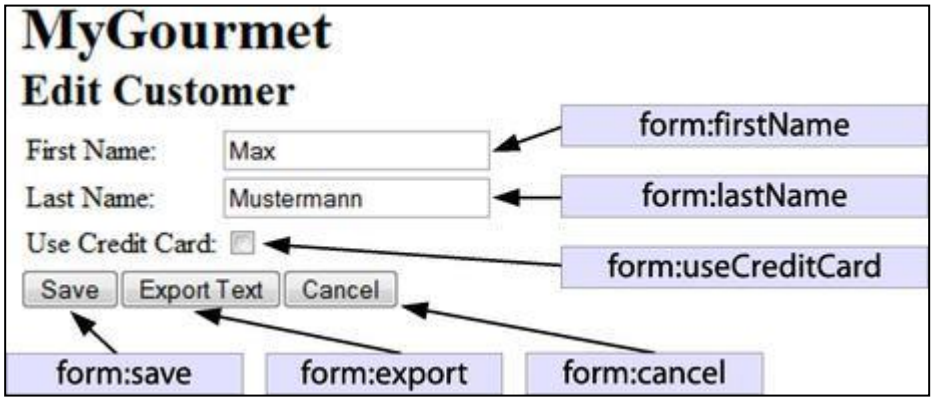
Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
ID-Attribute in JSF

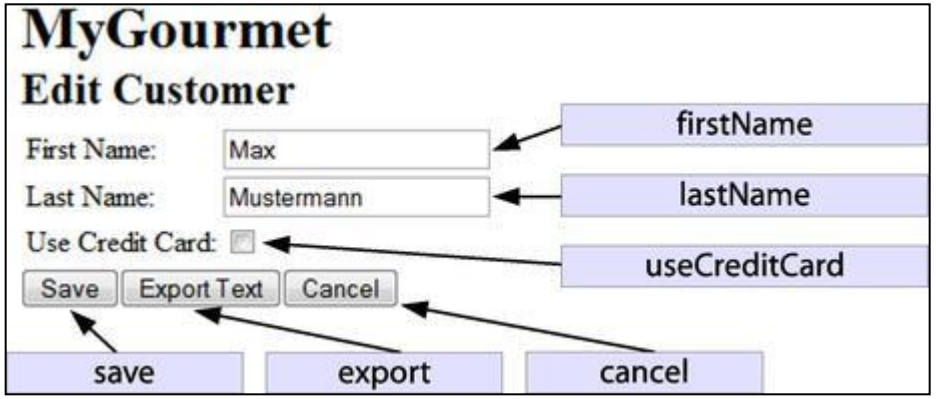
Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Modul 7
ID-Attribut in JSF

```
<h:form id="form">
   <h:inputText id="firstName" value="#{helloBean.name}">
```



Ohne Angabe wird automatisch eine ID erstellt: **`j_id`** …




```
<h:form id="form"
prependId="false">
```

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

Modul 7
Navigation

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

Modul 7
Navigation

- Navigation mittels Steuerkomponenten und action-Attribut
- Es gibt nur die **zwei** Steuerkomponenten **`h:commandButton`** und **`h:commandLink`**
- action–Attribut enthält die ViewID (“xhtml-Seite”, Facelet) oder eine Methode, die die ViewID zurückgibt

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

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

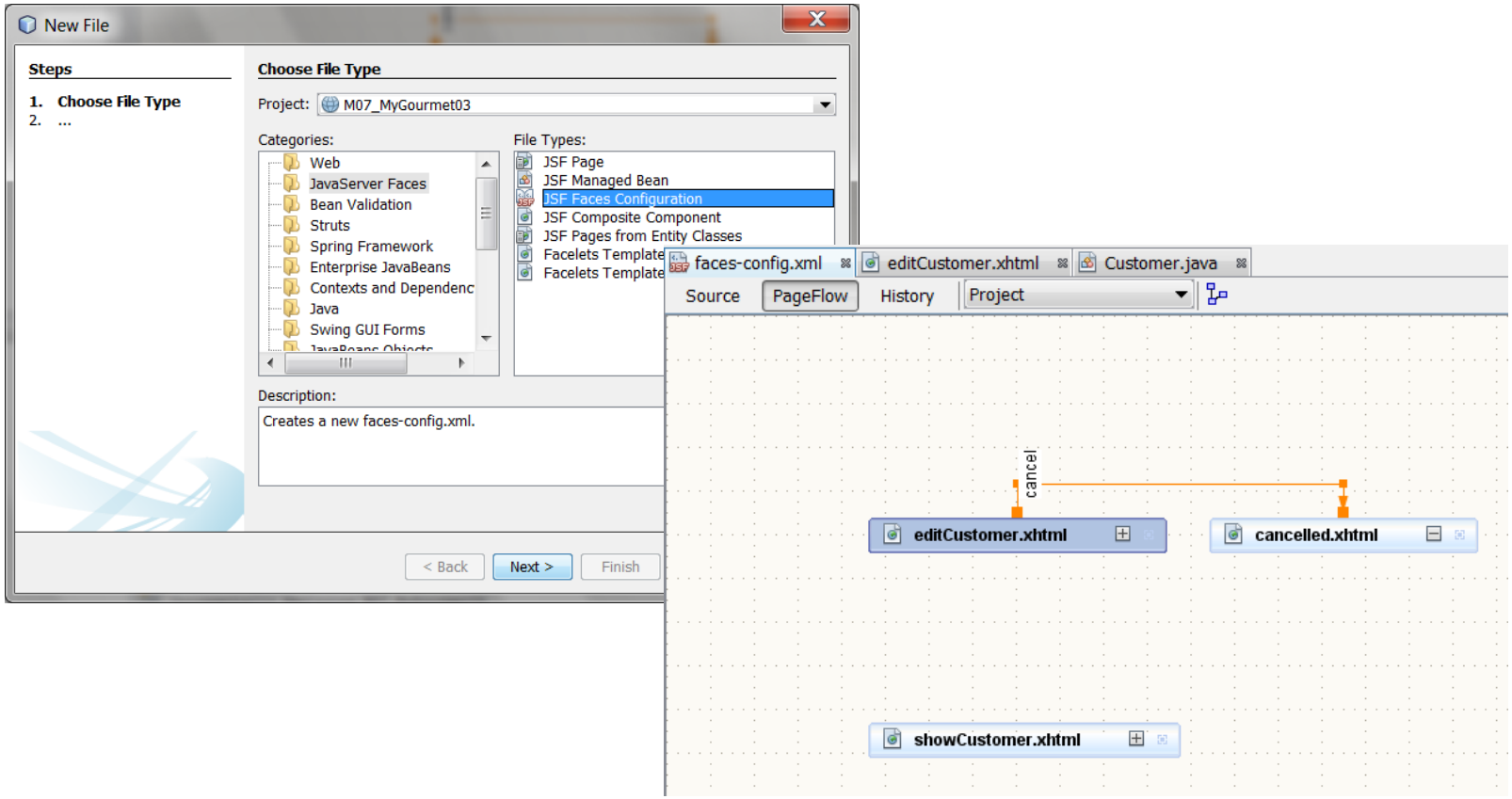
Quelle: http://jsfatwork.irian.at/book_de/introduction.html

Alternativ Definition der Navigation in der faces-config.xml Datei

```
 <navigation-rule>
  <from-view-id>/editCustomer.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>ok</from-outcome>
    <to-view-id>/showCustomer.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>cancel</from-outcome>
    <to-view-id>/cancelled.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

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

Bearbeitung der faces-config.xml-Datei in netbeans auch grafisch möglich.



MyGourment04

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

Modul 7
Events

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

Modul 7
Events

Durch den Benutzer ausgelöst:

Value-Change-Events (*zu implementieren: ValueChangeListener*)
werden ausgelöst, wenn sich der Wert einer Eingabekomponente ändert.

Action-Events (*zu implementieren: ActionListener*)
werden von Steuerkomponenten (Link oder Button) ausgelöst, wenn sie aktiviert werden.

Durch das System ausgelöst:

System-Events
werden vom System zu bestimmten Zeitpunkten im Lebenszyklus ausgelöst.

Phase-Events
werden vom System routinemäßig beim Abarbeiten des Lebenszyklus vor und nach jeder Phase ausgelöst.

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

Modul 7
Events – valueChangeListener

Variante 1: Als Ereignisbehandlungsmethode

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

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

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

Modul 7
Events – valueChangeListener

Variante 2: Als Kindelement und eigener Klasse

```
<h:selectBooleanCheckbox onclick="this.form.submit()"
    value="#{customer.useCreditCard}" immediate="true" >
  <f:valueChangeListener
      type="at.irian.CreditCardChangeListener"/>
</h:selectBooleanCheckbox>
```

```
public class CreditCardChangeListener
    implements ValueChangeListener {
  public void processValueChange(ValueChangeEvent e) {
    Boolean useCreditCard = (Boolean) e.getNewValue();
    FacesContext fc = FacesContext.getCurrentInstance();
    if (useCreditCard != null) {
      ELContext el = fc.getELContext();
      Customer customer = (Customer) el.getELResolver()
          .getValue(el, null, "customer");
      customer.setUseCreditCard(useCreditCard);
    }
    fc.renderResponse();
  }
}
```

Modul 7
Events – actionListener

Variante 1: Als Ereignisbehandlungsmethode

```
<h:commandButton id="submitButton" value="Submit"
                        action="#{userData.showResult}“
                 actionListener="#{userData.updateData}" />
</h:commandButton>
```

```
public void updateData(ActionEvent e){
        data="Hello World";
}
```

Variante 2: Als Kindelement mit eigener Klasse

http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm

Modul 7
Events - System-Events

Für bestimmte Komponenteninstanzen:

`PreRenderComponentEvent`
wird vor dem Rendern einer Komponente ausgelöst.

`PreRenderViewEvent`
wird vor dem Rendern der kompletten Seite für die Wurzelkomponente des Komponentenbaums ausgelöst.
…

System-Events unabhängig von einer Komponenteninstanz:

`PostConstructApplicationEvent`
wird beim Starten der Applikation ausgelöst nachdem die Konfiguration fertig geladen ist.

`PreDestroyApplicationEvent`
wird beim Beenden der Applikation ausgelöst.
…

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm

Modul 7
Events - Phase-Events

Vor und nach jeder Phase wird ein Event ausgelöst. Auf diesen kann mit einem „PhaseListener“ reagiert werden.

```
<lifecycle>
          <phase-listener>
                    at.irian.DebugPhaseListener
          </phase-listener>
</lifecycle>
```

Registrieren in **faces-config.xml**

```
public class DebugPhaseListener implements PhaseListener {
  private static final Logger log =
            Logger.getLogger(ApplicationListener.class.getName());

  public void afterPhase(PhaseEvent event) {
    log.info("After phase: " + event.getPhaseId());
  }

  public void beforePhase(PhaseEvent event) {
    log.info("Before phase: " + event.getPhaseId());
  }

  public PhaseId getPhaseId() {
    return PhaseId.ANY_PHASE;
  }
}
```

MyGourment05

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Direkte Ausgabe auf das Response-Objekt

```
public String export() {
  FacesContext fc = FacesContext.getCurrentInstance();
  try {
    HttpServletResponse resp = (HttpServletResponse)
        fc.getExternalContext().getResponse();
    resp.setContentType("text/plain");
    PrintWriter writer = resp.getWriter();
    writer.print("First Name: ");
    writer.println(firstName);
    writer.print("Last Name: ");
    writer.println(lastName);
    if (useCreditCard) {
      writer.print("Credit Card Type: ");
      writer.println(creditCardType);
      writer.print("Credit Card Number: ");
      writer.println(creditCardNumber);
    }
    fc.responseComplete();
  } catch (IOException e) {e.printStackTrace();}
  return null;
}
```

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm

Modul 7
Konverter

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Konverter

Standard-Konverter

```
<h:inputText id="birthday" size="30"
    value="#{customerBean.customer.birthday}">
  <f:convertDateTime pattern="dd.MM.yyyy"/>
</h:inputText>
```

Benutzerdefinierte-Konverter

```
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://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Bean-Validation

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Bean-Validation (JSR-303)

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

   @NotNull @Size(min=1)
   private String city;

   @NotNull
   private String street;

   ...
}
```

Auch benutzerdefineirte Bean-Validation möglich …

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Bean-Validation (JSF klassisch)

```
<h:inputText value="#{backingBean.property}">
       <f:validateLength minimum="3" maximum="7"/>
</h:inputText>
```

Auch benutzerdefineirte Bean-Validation möglich ...

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Messages

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Messages

Ausgabe von (Fehler-)Meldungen

```
…
<h:form>
    <h:messages errorStyle="color:red" />
    <h:inputText required="true" label="My Input" />
    <h:commandButton value="Click" />
</h:form>
…
```

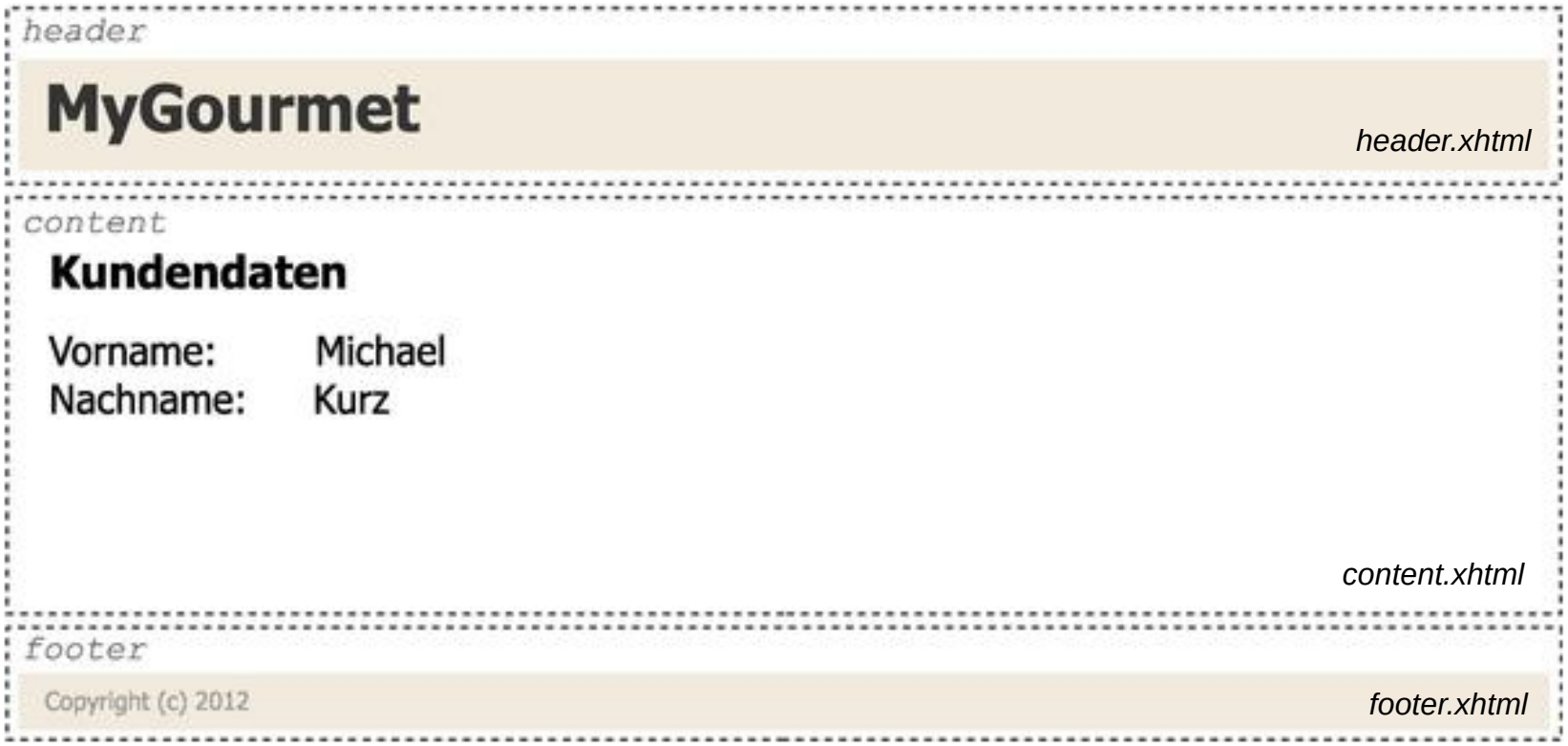
Erzeugen einer eigenen Meldungen in einer Bean

```
…
  String message = "submitted successfully !!";
  FacesContext.getCurrentInstance().addMessage(null,
                                  new FacesMessage(message)); }
…
```

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Wiederverwendung von Inhalten mit Facelets



Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Wiederverwendung von Inhalten mit Facelets

Definition einer „Komposition“ (wiederverwendbarer Block)

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets">
<head><title>MyGourmet header</title></head>
<body>
  <ui:composition>
    <h:panelGroup style="width: 100; height: 40px;"
        layout="block">
      <h:graphicImage value="/images/logo.png"
          style="float: left;"/>
      <h1 style="display: inline; margin-left: 5px;">
        #{msgs.title_main}
      </h1>
    </h:panelGroup>
    <h2>#{pageTitle}</h2>
  </ui:composition>
</body>
</html>
```

/WEB-INF/includes/header.xhtml

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Wiederverwendung von Inhalten mit Facelets

Verwenden einer „Komposition“

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets">
  <head>
    <title>#{msgs.title_main}</title>
  </head>
  <body>
    <ui:include src="/WEB-INF/includes/header.xhtml">
      <ui:param name="pageTitle"
          value="#{msgs.title_show_customer}"/>
    </ui:include>
    ...
  </body>
</html>
```

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Templating

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Templating: ein Template

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets">
<head>
  <title>MyGourmet</title>
  <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
  <div id="header">
    <ui:insert name="header">
      <h1>MyGourmet</h1>
    </ui:insert>
  </div>
  <div id="content">
    <ui:insert name="content"/>
  </div>
  <div id="footer">
    <ui:insert name="footer">
      <h:outputText value="Copyright (c) 2012"/>
    </ui:insert>
  </div>
</body>
</html>
```

template.xhtml

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Templating: ein Template-Client

Template-Client

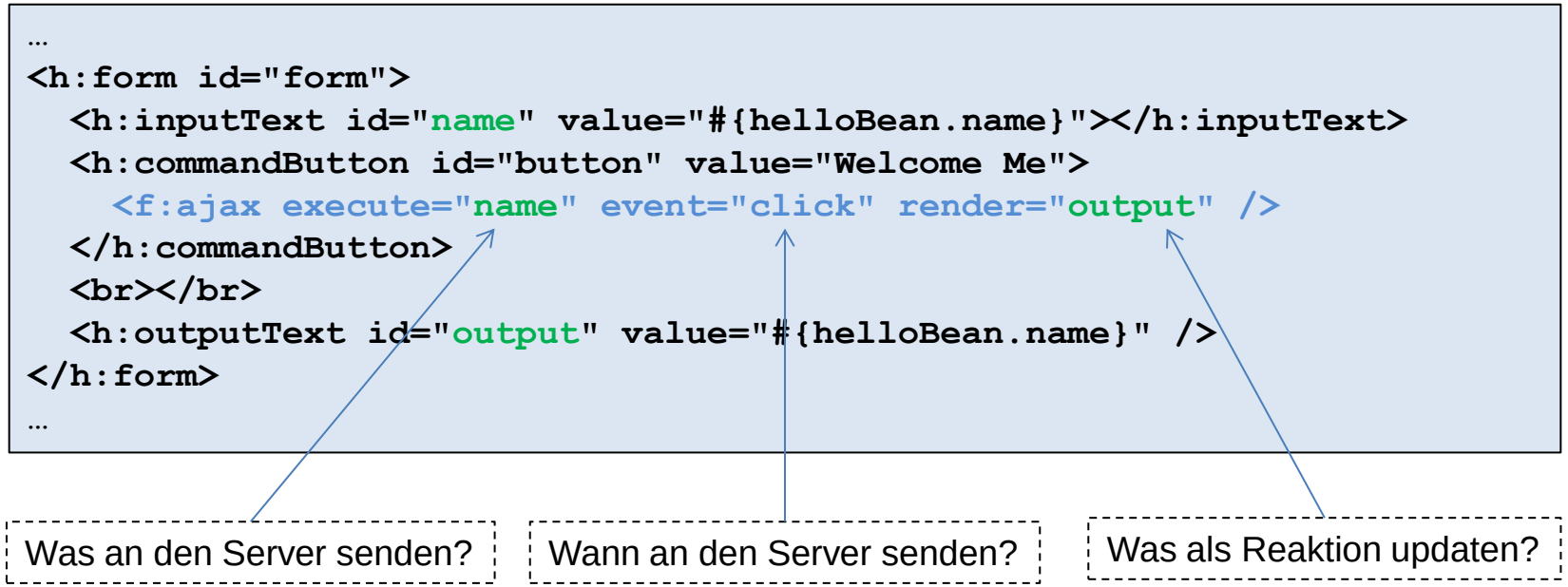
```
…
<ui:composition template="template.xhtml"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets">
  <ui:define name="content">
    <h2>Kundendaten</h2>
    <h:panelGrid id="grid" columns="2">
      <h:outputText value="Vorname:"/>
      <h:outputText value="#{customer.firstName}"/>
      <h:outputText value="Nachname:"/>
      <h:outputText value="#{customer.lastName}"/>
    </h:panelGrid>
  </ui:define>
</ui:composition>
…
```

showCustomer.xhtml

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

```
…
<h:form id="form">
  <h:inputText id="name" value="#{helloBean.name}"></h:inputText>
  <h:commandButton id="button" value="Welcome Me">
    <f:ajax execute="name" event="click" render="output" />
  </h:commandButton>
  <br></br>
  <h:outputText id="output" value="#{helloBean.name}" />
</h:form>
…
```



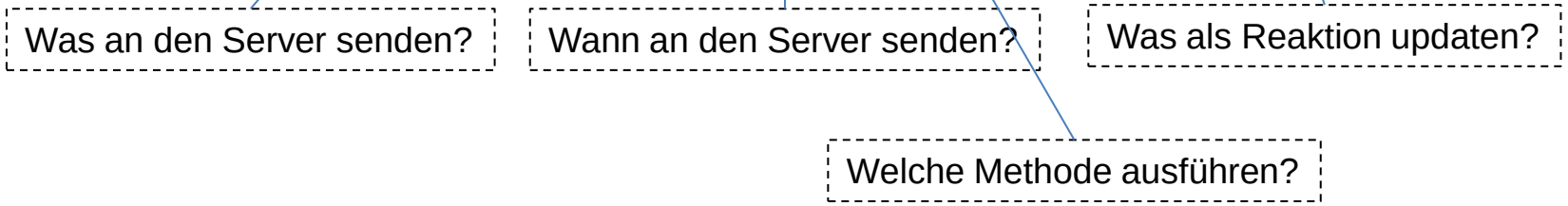
Kein **`action`**-Attribut. Warum?
- Es soll ja keine neue Seite geladen werden (Ajax).

Bsp.: SimpleAjax

Quelle: http://www.mkyong.com/jsf2/jsf-2-0-ajax-hello-world-example/

Modul 7
Ajax-Listener

```
…
<h:form id="form">
   <h:inputText id="name" value="#{helloBean.name}">
       <f:ajax execute="name" event="keyup" render="output"
          listener="#{helloBean.addTime()}"/>
   </h:inputText>
   <br></br>
   <h:outputText id="output" value="#{helloBean.name}" />
</h:form>
…
```



Bsp.: SimpleAjax

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Ajax onevent-Parameter

```
…
<script>
    function doSomeThing() {
        alert("OK");
    }
</script>
…
```

```
…
 <h:form id="form">
      <h:inputText id="name" value="#{helloBean.name}">
         <f:ajax execute="name" event="keyup" render="output"
                 listener="#{helloBean.addTime()}"
                 onevent="doSomeThing"  />
      </h:inputText>
      <br></br>
      <h:outputText id="output" value="#{helloBean.name}" />
 </h:form>
…
```

Welche JavaScript-Methode soll im jeweiligen Browser aufgerufen werden?

Bsp.: SimpleAjax

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Ajax-Parameter

`event`:
Name des Ereignisses, das die Ajax-Anfrage auslöst (*onclick, onkeypressed, …*)

`execute`:
IDs der Komponente, die durch die Ajax-Anfrage an den Server gesendet wird. Auch @this, @form (das Formular des Elements), **@all** (alle Elemente) und @none (kein Element)).

`render`:
Liste der IDs der Komponenten, die gerendert werden sollen. Auch (@this, @form, …)

`listener`:
Eine Methode einer Managed-Bean wird als Listener für den jeweiligen event registriert. Ausführung in der Invoke-Application-Phase (Phase 5).

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Ajax-Parameter

`onevent`:
Erlaubt das Registrieren einer clientseitigen JavaScript-Callback-Funktion für Ajax-Ereignisse. (Auch Parameterübergabe ist möglich …)

`onerror`:
Erlaubt das Registrieren einer clientseitigen JavaScript-Callback-Funktion für Fehler, die beim Bearbeiten der Ajax-Anfrage auftreten.

`disabled`:
Das Ajax-Verhalten wird "abgeschaltet", wenn dieses Attribut auf true gesetzt ist.

Beispiel: M07_SimpleAjax

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Tipps

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Tipps

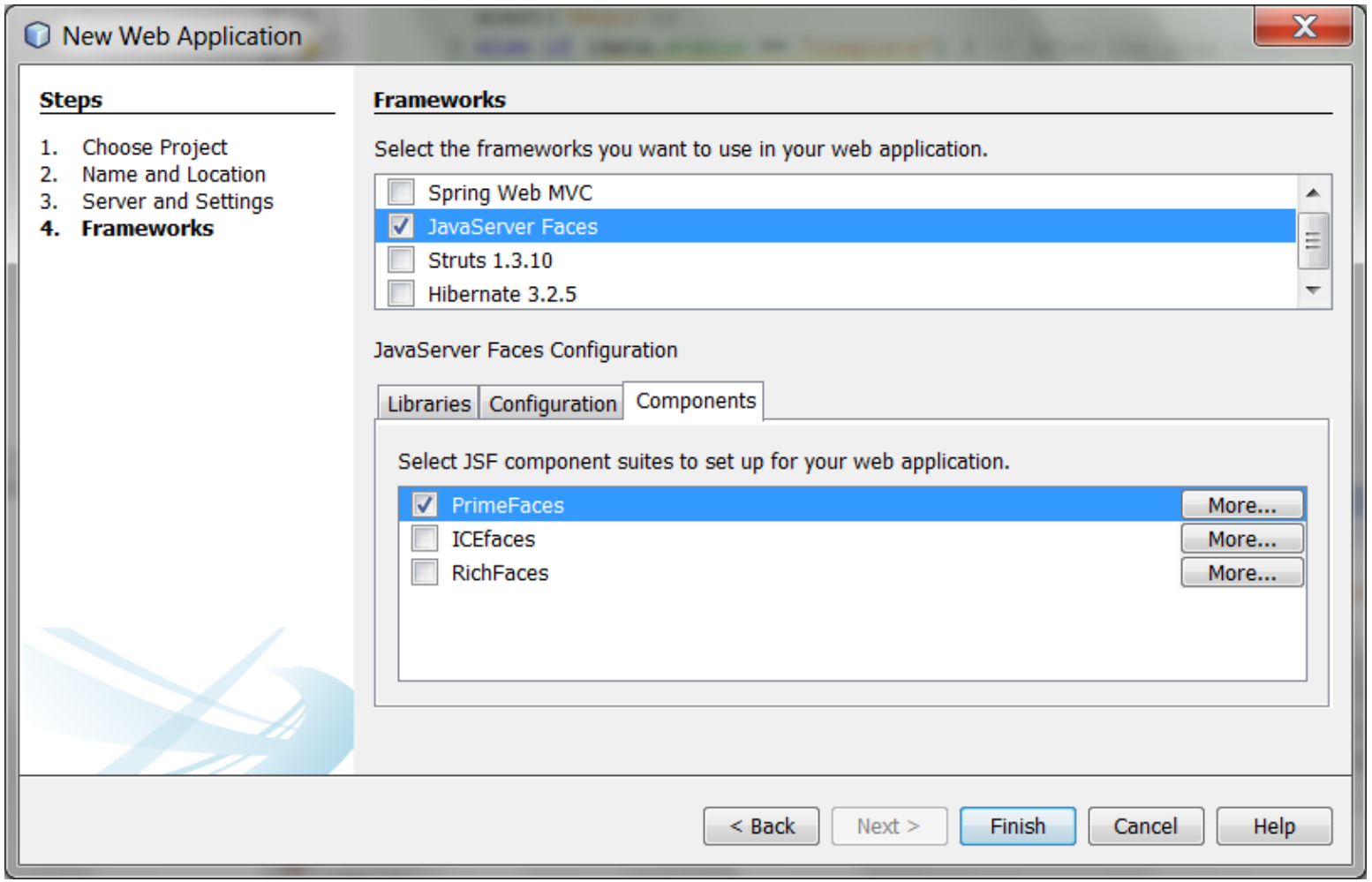
- Verwenden Sie in allen Seitendeklarationen **`h:head`** und **`h:body`** . Nur so kann JSF die für Ajax benötigten Skript-Ressourcen richtig einbinden.

- Auch mit Ajax muss weiterhin **`h:form`** in gewohnter Art und Weise eingesetzt werden.

- **`@ManagedBean: import javax.faces.bean.ManagedBean;`**
- **`@SessionScoped: import javax.faces.bean.SessionScoped;`**

Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl

Modul 7
Primefaces, Richfaces, ICEfaces



Quelle: http://www.tutorialspoint.com/jsf/jsf_actionlistener_tag.htm nl