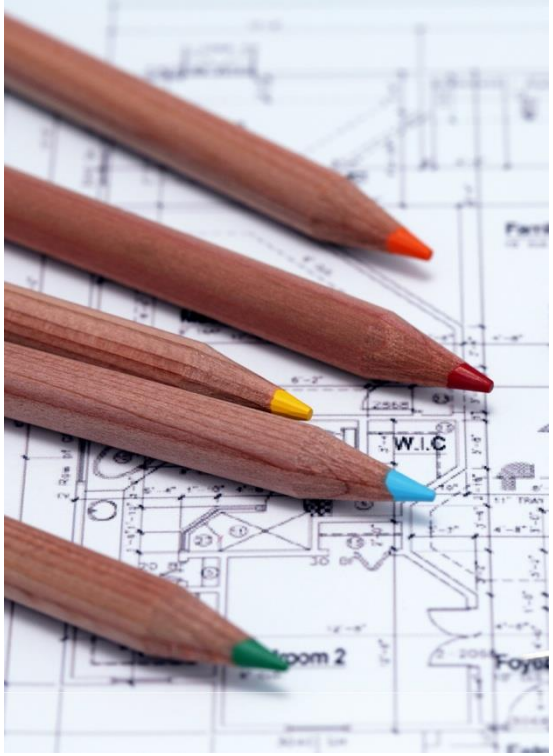


ARS

Java Server Faces

Ajax

Überblick

- Client Components
  - Motivation
  - Ansätze
- Ajax mit JSF 2
  - Partial View Processing
  - JSF JavaScript API
  - JSF Ajax Tag

Client Components

Motivation

- Verlagerung bestimmter Funktionen auf Client
  - Validierung von Benutzereingaben
  - Event-Handling
- Vorteile:
  - Verringern von Server- und Netzwerklast
  - Verringern von Antwortzeiten
  - „Desktop-Feeling“
- Umsetzung clientseitiger Ablauflogik
  - JavaScript
  - Applets (weniger praktikabel)

Client Components

Was bedeutet Client-seitig?

- Ausführen von Aktionen auf dem Client des Nutzers (Browser)
- Server erhält über diese Vorgänge keinerlei Informationen

Was bedeutet Server-seitig?

- Ausführen von Aktionen auf dem Server
- Client sendet für jede Aktion einen Request
- Kompletter Seitenneuaufbau aufgrund der Antwort vom Server

AJAX als Mischform

- Ausführen von Aktionen weitestgehend auf dem Client
- Werden Informationen vom Server benötigt, wird ein Request verschickt
- Antwort wird analysiert, um aktuelle Seite anzupassen

Asynchronous JavaScript and XML (AJAX, Beispiel)

```
<SCRIPT type="text/javascript">
  function doClick(node) {
    var title = document.getElementById('title');
    var req = new XMLHttpRequest();
    req.open('GET', '/MyServlet', true ); // true: asynchron
    req.onreadystatechange = function() {
      switch(req.readyState) {
        case 4: if(req.responseText == 'true')
                    title.parentNode.removeChild(title);
                break;
        case default: return false;
      };
    };
    req.setRequestHeader('Content-Type', 'text/plain');
    req.send('Das ist meine Nachricht.'); // null: Nachricht ohne Inhalt
  }
</SCRIPT>
```

Umsetzung in JSF

- Ziel: Verlagerung der Phasen auf den Client
- Problem: Implementierung der Komponenten in Java
- Status Quo:
  - JSF ab Version 2 bietet Mechanismen
  - JSF-Implementierungen ermöglichen vereinzelt Client Components
    - Unterschiedliche Strategien und Umsetzungen
  - IBM-Komponenten unterstützen teilweise JS/AJAX
- Ansätze:
  - Renderer der UI-Komponenten erzeugen JavaScript
  - Custom Tags erzeugen JavaScript
  - Custom Components

Validatoren

- Render-Ansatz

```
<h:inputText id="isbnInput"
                value="#{insertBean.isbn}" required="true"/>
```


```
<form onSubmit="return validate();" ...>
  <input id="isbnInput" name="isbnInput" value=""/>
</form>
<script type="text/javascript">
  function validate() {
    if(document.getElementById("isbnInput").value == "") {
      ...
    }
  }
</script>
```

Validatoren

- Custom Tag – Ansatz

```
<h:inputText id="isbnInput" value="#{insertBean.isbn}">
  <f:validateLength minimum="10" maximum="13">
</h:inputText>
```


```
<form onSubmit="return validate();" ...>
  <input id="isbnInput" name="isbnInput" value=""/>
</form>
<script type="text/javascript">
  function validate() {
    if(document.getElementById("isbnInput").value.length < 10) {
      ...
    }
  }
</script>
```

Validatoren

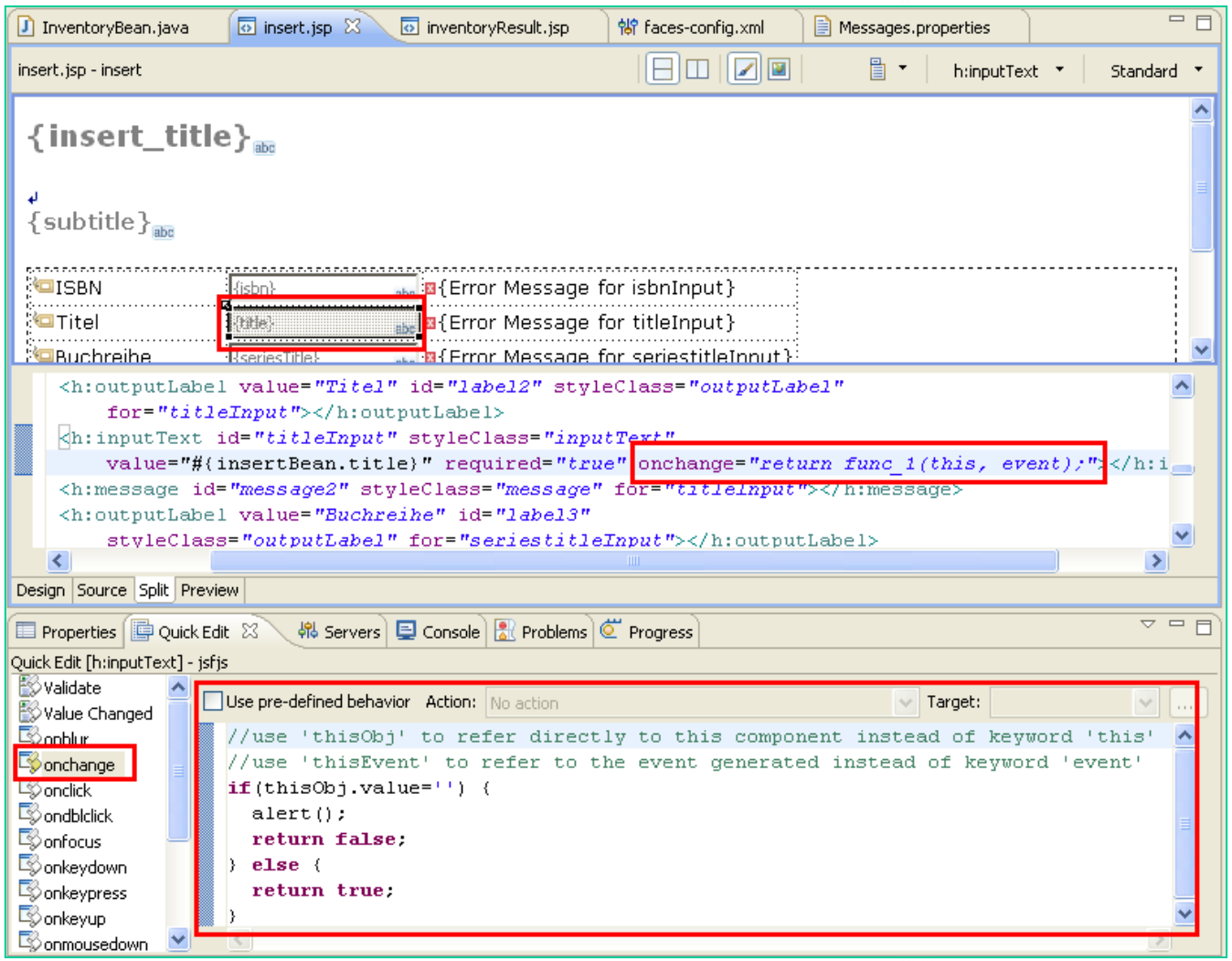
- Beispiel: Apache MyFaces
  - JSF-Implementierung
  - Verwendung der JSF-Tags
  - Umschalten der Renderer per Konfigurationsparameter im Web Deployment Descriptor

```
<context-param>
  <param-name>org.apache.myfaces.ENABLE_CLIENT_SIDE_VALIDATION</param-name>
  <param-value>true</param-value>
</context-param>
```

  - Apache MyFaces Trinidad
    - Eigene Validatoren in Java- und JavaScript-Versionen möglich
    - Voraussetzung: Client-Side-Converter

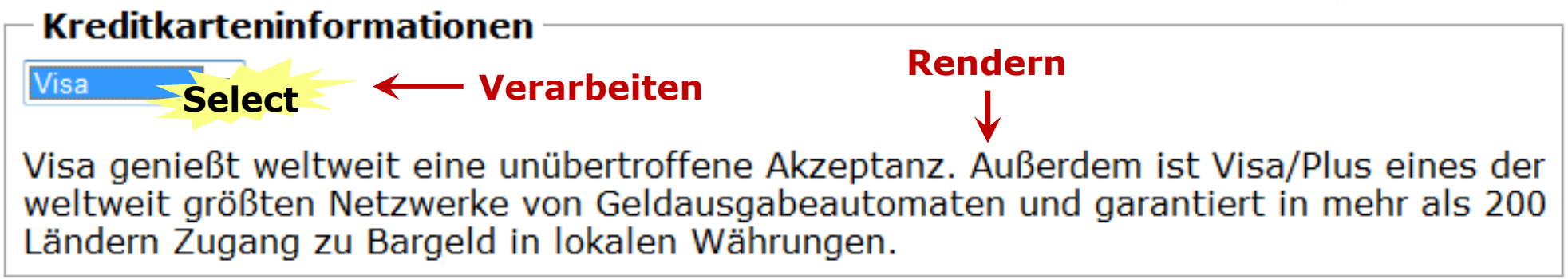
Event Handler / Aktionen

- Analoge Ansätze zu Validatoren
- `onXXX`-Attribute von Komponenten erlauben direkte Eingabe von JavaScript
- Unterstützung im RSA/RAD über QuickEdit-View



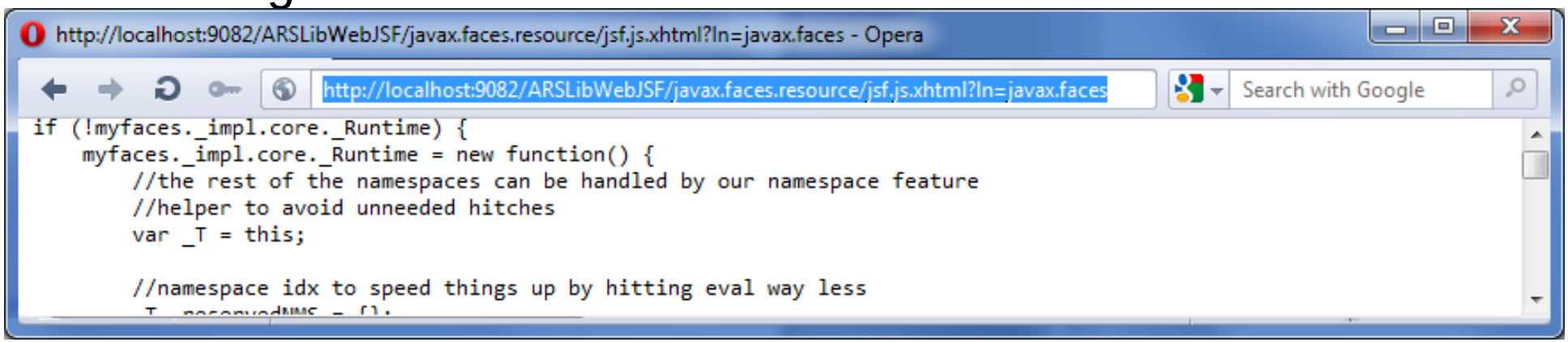
Partial View Processing

- Anwendung des Request-Lifecycle auf einen Teil des Komponentenbaumes
  - Verarbeitung von Benutzereingaben
  - Rendern
- Voraussetzung für „Ajaxifizierung“ einer Applikation



JSF JavaScript API

- API für clientseitige Operationen
  - Informationen über Umgebung
  - Ajax
- Auslieferung über FacesServlet



- Einbinden im Facelet

```
<h:outputScript name="jsf.js" library="javax.faces"/>
```

- Dokumentation unter
  http://javaserverfaces.java.net/nonav/docs/2.0/jsdocs/index.html

JSF JavaScript API

```
<h:form>
  <h:outputScript name="jsf.js" library="javax.faces" target="head"/>
  <h:inputText id="myinput" value="#{userBean.name}"/>
  <h:outputText id="outtext" value="Echo: #{userBean.name}"/>
  <h:commandButton id="submit" value="submit"
    onclick="jsf.ajax.request(
               this,
               event,
               {execute:'myinput',render:'outtext'}
             );
             return false;" />
</h:form>
```

Ajax mit JSF 2

JSF Ajax Tag `<f:ajax/>`

- Komponente zur Kapselung der Ajax-Funktionalität
- Vermeiden von JavaScript in Facelets

```
<h:form>
  <h:outputScript name="jsf.js" library="javax.faces" target="head"/>
  <h:inputText id="myinput" value="#{userBean.name}"/>
  <h:outputText id="outtext" value="Echo: #{userBean.name}"/>
  <h:commandButton id="submit" value="submit">
    <f:ajax event="click" execute="myinput" render="outtext" />
  </h:commandButton>
</h:form>
```

JSF Ajax Tag `<f:ajax/>`

- Hinzufügen eines clientseitigen Events für Komponente(n)
  - Einbetten in Komponente

```
<h:commandButton id="submit" value="submit">
  <f:ajax event="click" execute="myinput" render="outtext" />
</h:commandButton>
```

  - Umschließen der Komponente(n)

```
<f:ajax event="click" render="outtext">
  <h:outputText id="outtext" value="Echo: #{userBean.name}"/>
  <h:commandButton id="submit" value="submit"/>
</f:ajax>
```

JSF Ajax Tag `<f:ajax/>`

- Event Handling durch Binding von Listener-Methode
  - Methode mit Parameter `javax.faces.event.AjaxBehaviorEvent`

```
<f:ajax event="click" listener="#{userBean.onEvent}">
  <h:outputText id="outtext" value="Echo: #{userBean.name}"/>
  <h:commandButton id="submit" value="submit"/>
</f:ajax>
```

ARS

Ajax mit JSF 2

JSF Ajax Tag `<f:ajax/>`

- Angabe von IDs für Attribute execute und render
  - Liste von Komponenten-IDs
  - Value Expression mit Collection von Strings
  - Schlüsselwort

```
<f:ajax execute="text1 text2 text3"/>
<f:ajax execute="#{myForm.executeComponents}"/>
<f:ajax execute="@form"/>
```

Schlüsselwort	Bedeutung
`@all`	Alle Komponenten im View
`@this`	Nur Auslöser des Requests
`@form`	Alle Komponenten im Formular
`@none` (Standard)	Keine Komponente

Kontrollfragen

- Wie lassen sich auch in älteren JSF-Versionen Teile der Funktionalität von Komponenten auf den Client verlagern?
- Was bedeutet Partial View Processing?
- Wie können ab JSF 2 Komponenten „ajaxifiziert“ werden?