



## Struts2 - eine Einführung

Labor für Verteilte Informationssysteme  
Fachgebiet Informatik  
Hochschule Karlsruhe

Dipl.-Inform.(FH) Adelheid Knodel

- 
- 
- 
- 

## Inhalt

- MVC Design
- Was ist Struts2?
- MVC Komponenten in Struts2
- Funktionsweise von Struts2
- Beispiele

## Architektureigenschaften

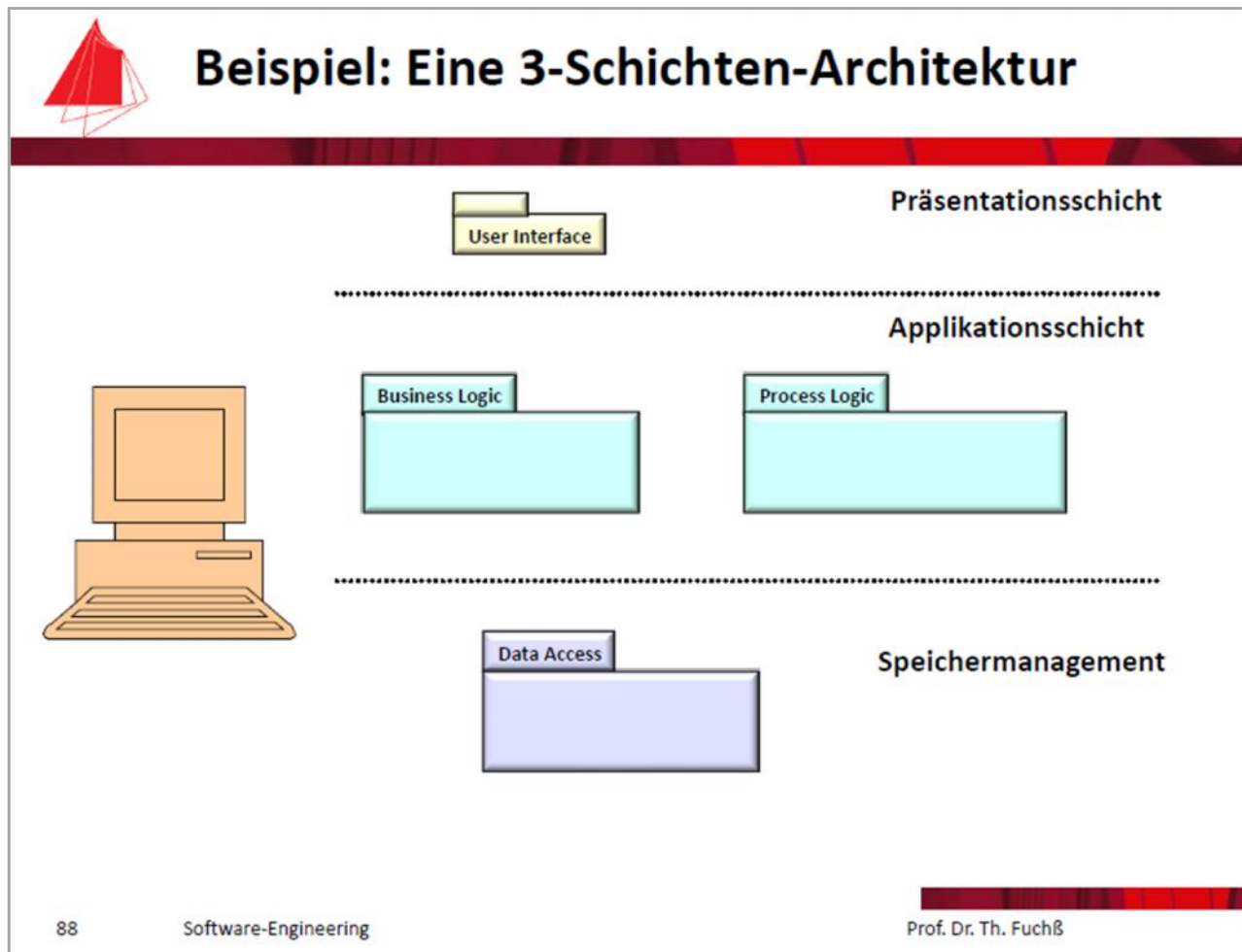


### Eigenschaften einer Architektur

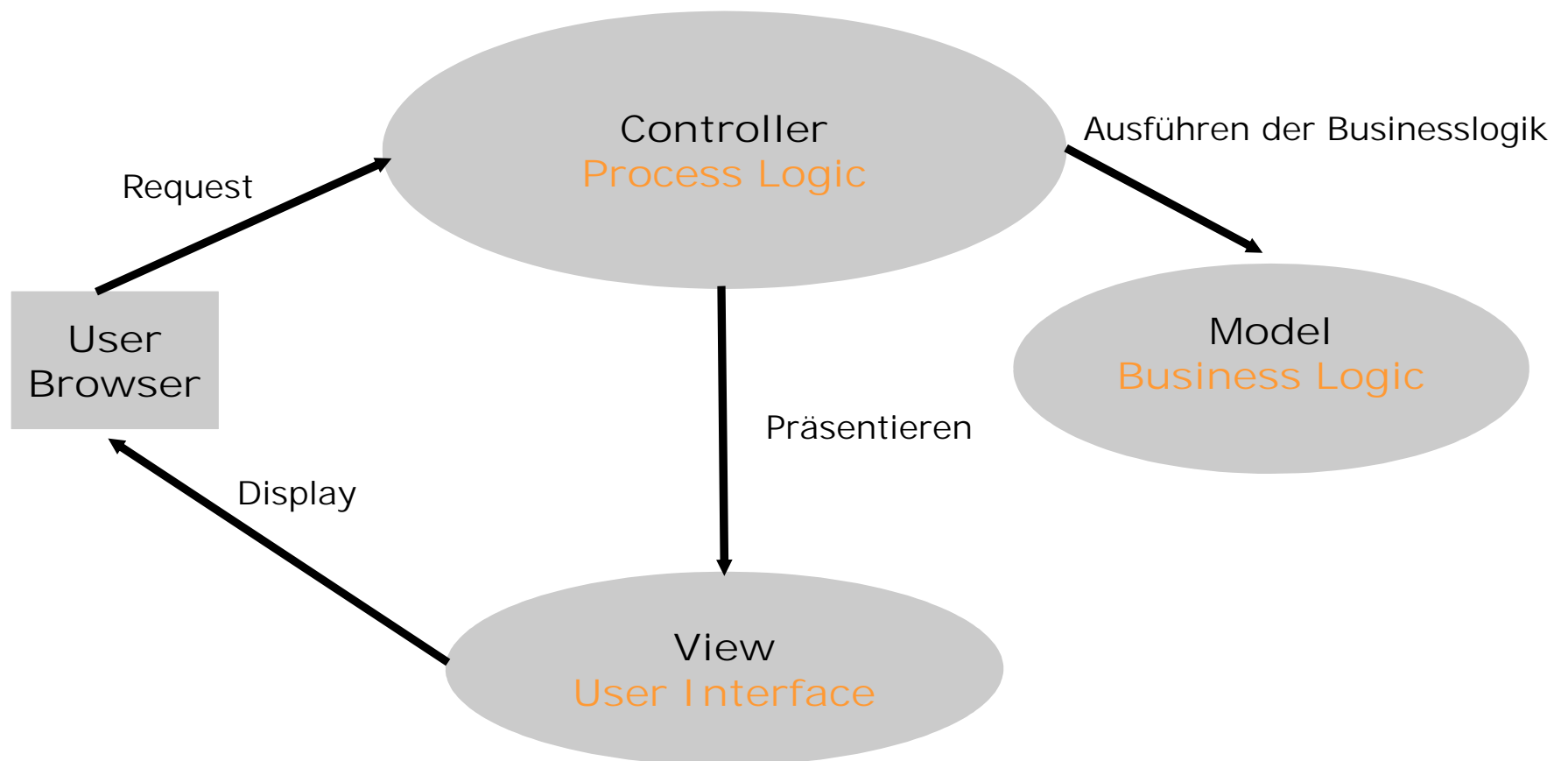
**Eine gut entworfene Architektur hat folgende Eigenschaften:**

- sie besitzt mehrere Schichten
- sie reduziert die Kopplung zwischen Subsystemen
- sie hat wohldefinierte Schnittstellen zwischen Systemen und Subsystemen
- sie ist leicht verständlich
- sie ist robust und skalierbar
- sie hat eine Vielzahl wiederverwendbarer Komponenten
- sie basiert auf den wichtigsten Use-Cases

## 3-Schichten-Architektur



## MVC Architektur



## Vorteile des MVC Designs

- Änderung der Funktionalität ohne Änderung der Seiten
- Änderung der Seiten ohne Änderung der Logik
- Bessere Strukturierung und damit Überschaubarkeit

## Was ist Struts2 ?

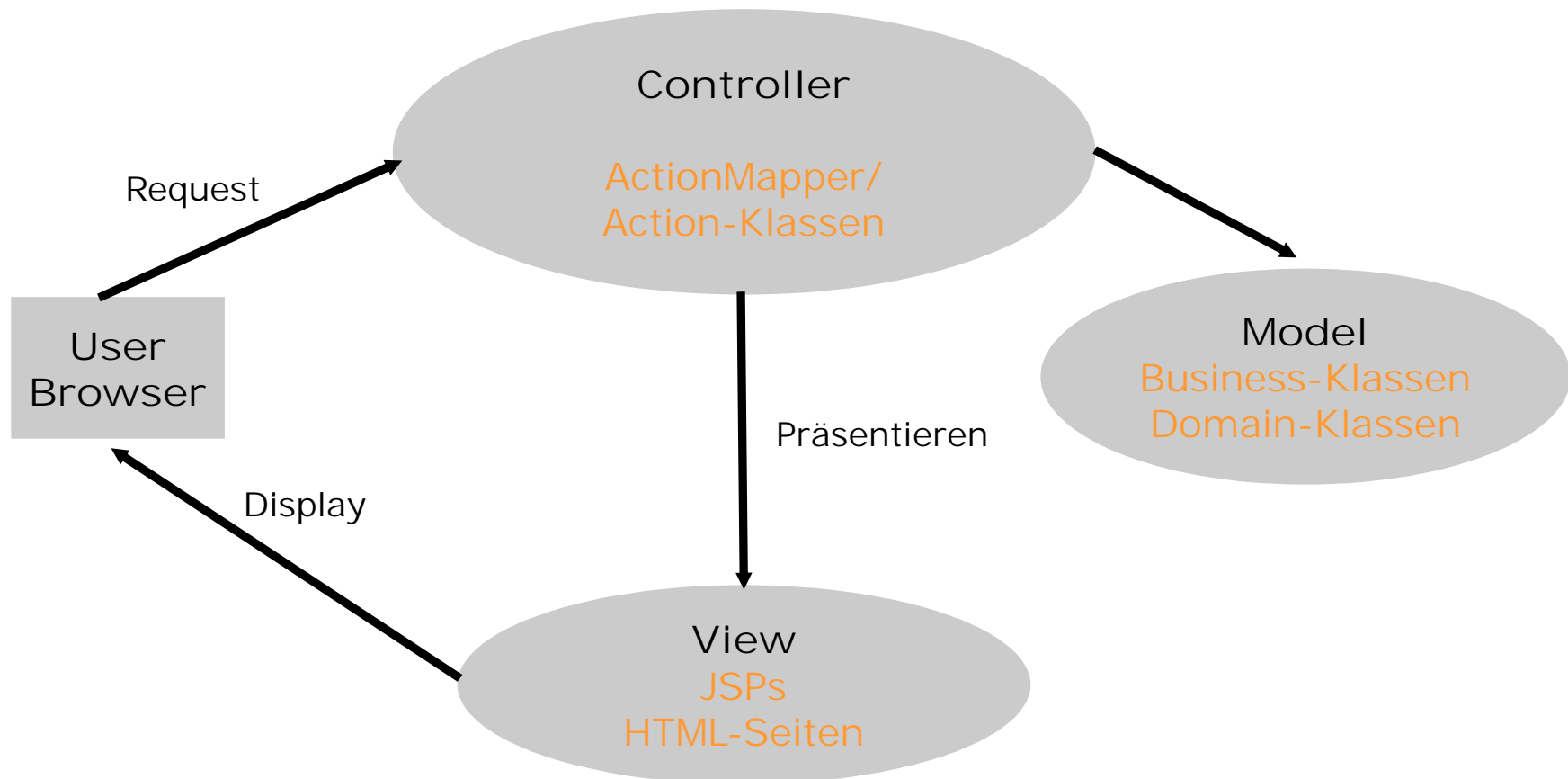
Struts2...

- ist ein Framework, das die Entwicklung von Java-Web-Applikationen entsprechend dem MVC-Prinzip unterstützt

Merkmale:

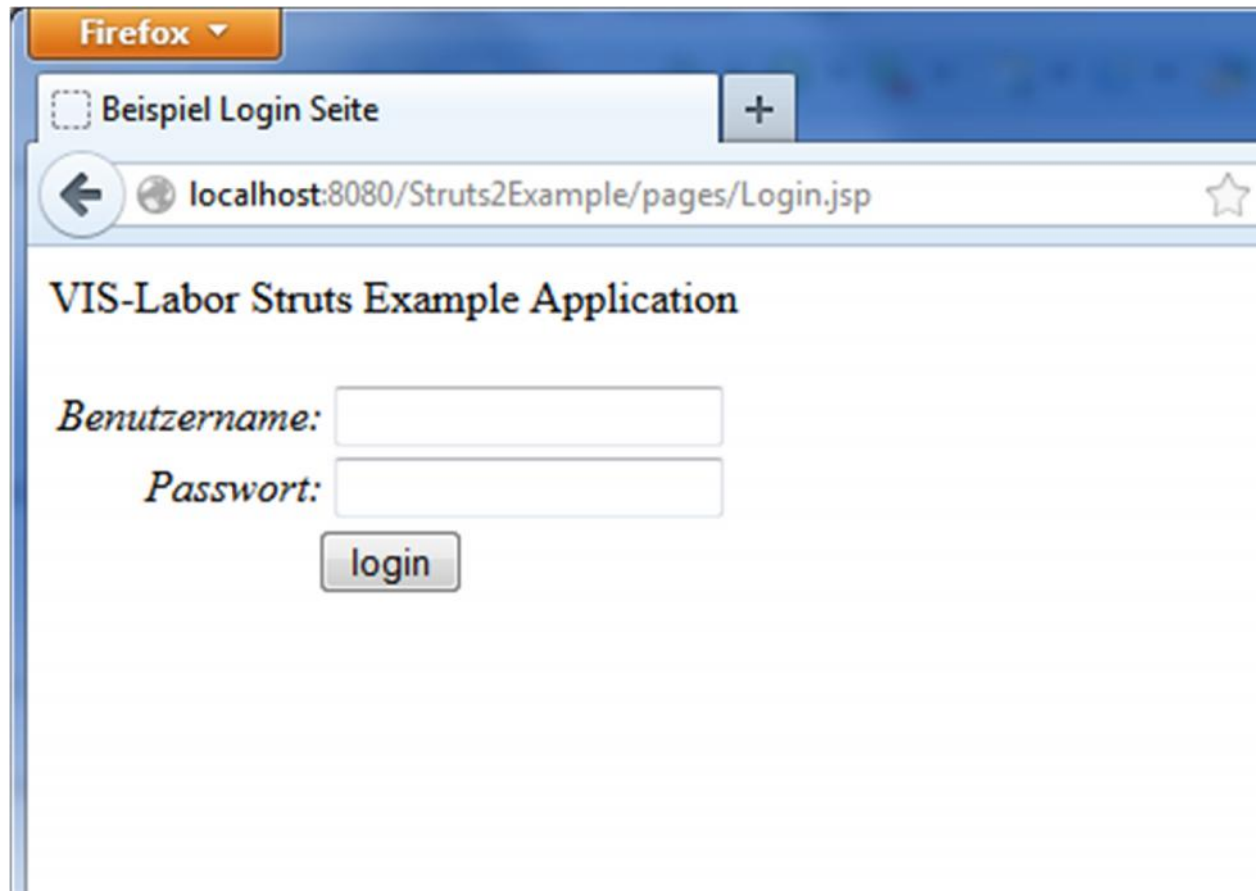
- Open Source, implementiert in Java
- Schlüsseltechnologien
  - Http Request/ Response, Session
  - Java Servlets, Java Filter
  - Java Server Pages, JSP Tag Libraries
  - Java Beans, Property Files, Resource Bundles, XML
- MVC Design Paradigma

## MVC Komponenten in Struts2



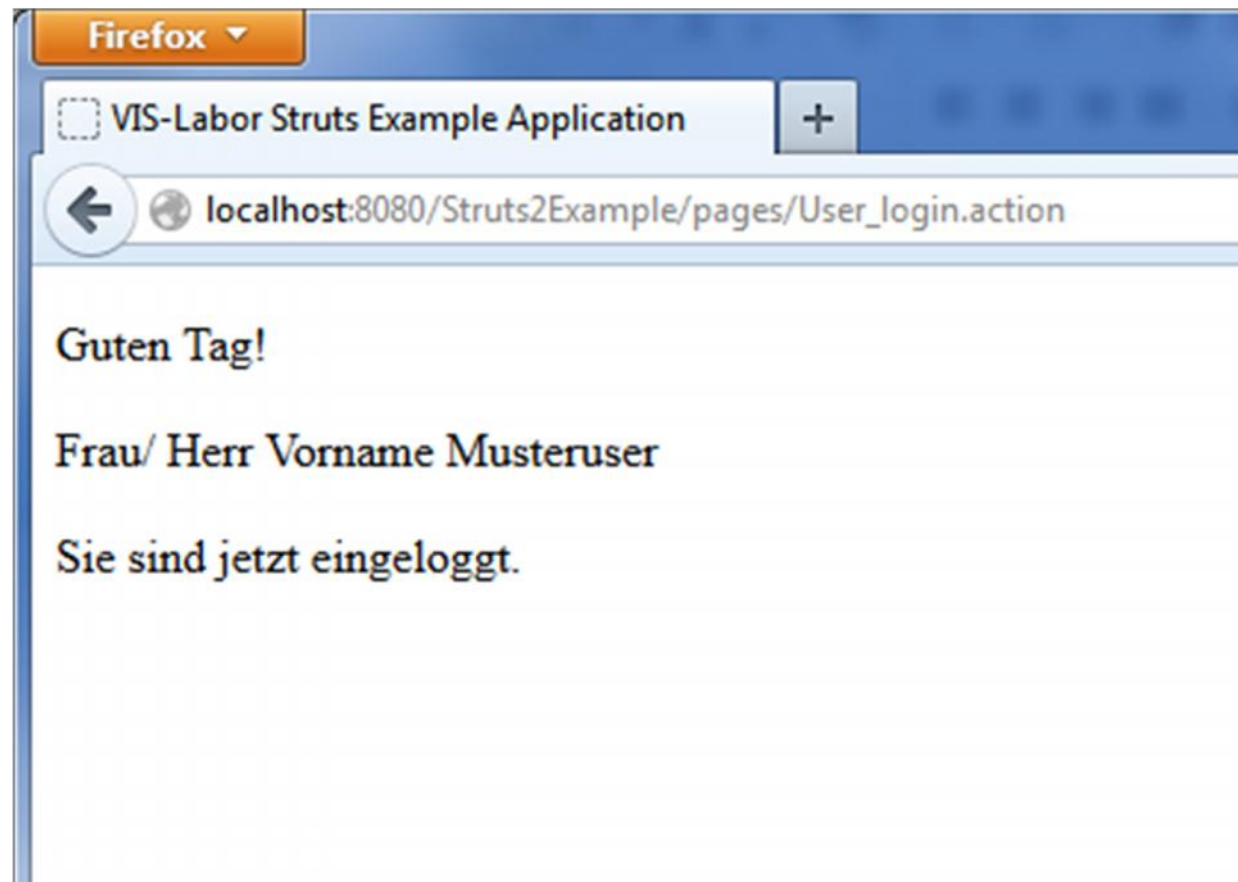


## Beispiel View: JSP/ Browser

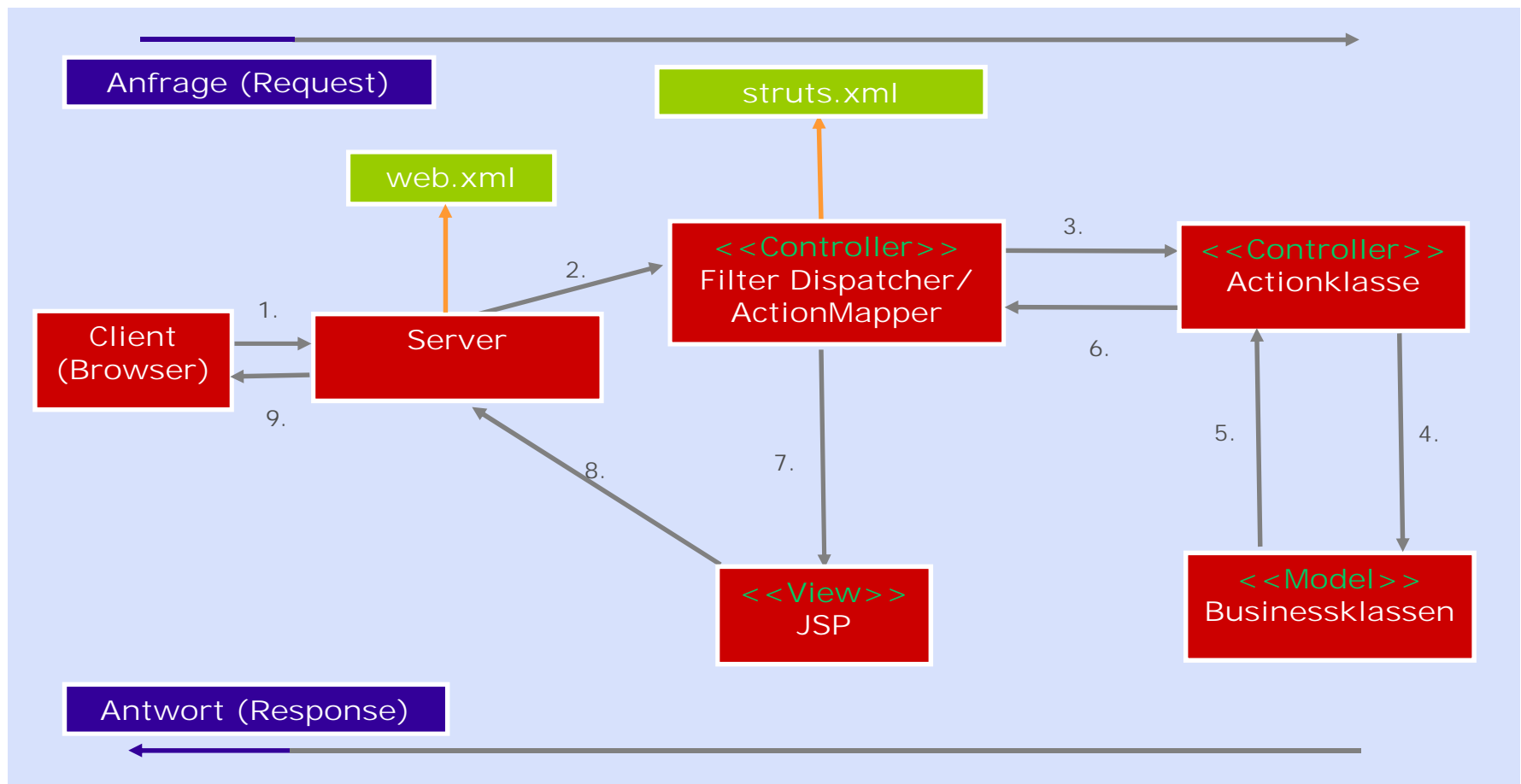


The screenshot shows a Firefox browser window with a single tab titled "Beispiel Login Seite". The address bar displays "localhost:8080/Struts2Example/pages/Login.jsp". The page content includes the title "VIS-Labor Struts Example Application", followed by two input fields labeled "Benutzername:" and "Passwort:". Below these fields is a "login" button.

## Beispiel View: JSP /Browser



## Funktionsprinzip und Ablauf



## Struts2 Controller Komponenten

- Hauptkomponente  
`org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter`  
`org.apache.struts2.dispatcher.mapper.ActionMapper`
- Konfiguration des Filters in `web.xml` des Projekts
- Definition des `ActionMapping` in `struts.xml`

## web.xml konfigurieren des Struts Controller

web.xml Deployment Descriptor File, beschreibt wie eine Web-Applikation in einem Servlet Container, wie z.B. Tomcat, konfiguriert werden soll.

```
<!-- Filter Dispatcher Configuration -->
```

```
<filter>
```

```
  <filter-name>struts2</filter-name>
```

```
  <filter-class>
```

```
    org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
```

```
  </filter-class>
```

```
</filter>
```

```
<!-- Filter Mapping -->
```

```
<filter-mapping>
```

```
  <filter-name>struts2</filter-name>
```

```
  <url-pattern>/*</url-pattern>
```

```
</filter-mapping>
```

## struts.xml

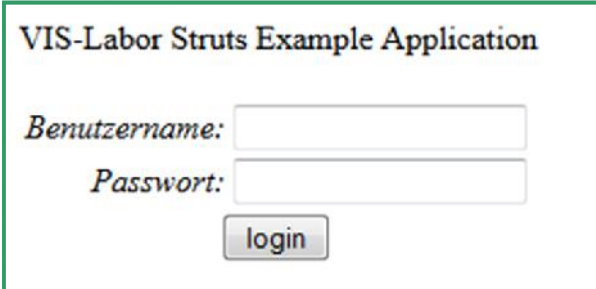
struts.xml enthält das Action Mapping

```
<struts>
  ...
  <!-- Action Definitions -->
    <package name= "package1" extends="struts-default" >
      <action name="action1" >
        .
      </action>
      .
      .
      <action name="actionn" >
        .
      </action>
    </package>
    .
    .
    <package name= "packageM" extends="struts-default" >
    </package>
  ...
</struts>
```

## Beispiel action in Login.jsp

```
<%@ page contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@ taglib prefix="s" uri="/struts-tags" %>
```

```
<html>
  <body>
    <s:text name="welcome.title" />
```



```
<s:form action="User_login" focusElement="username">
```

```
  <s:textfield name="username" label="Benutzername" />
```

```
  <s:password name="password" label="Passwort" /> <br>
```

```
  <s:submit method="execute" label="login" align="center" />
```

```
</s:form>
```

```
</body>
```

```
</html>
```

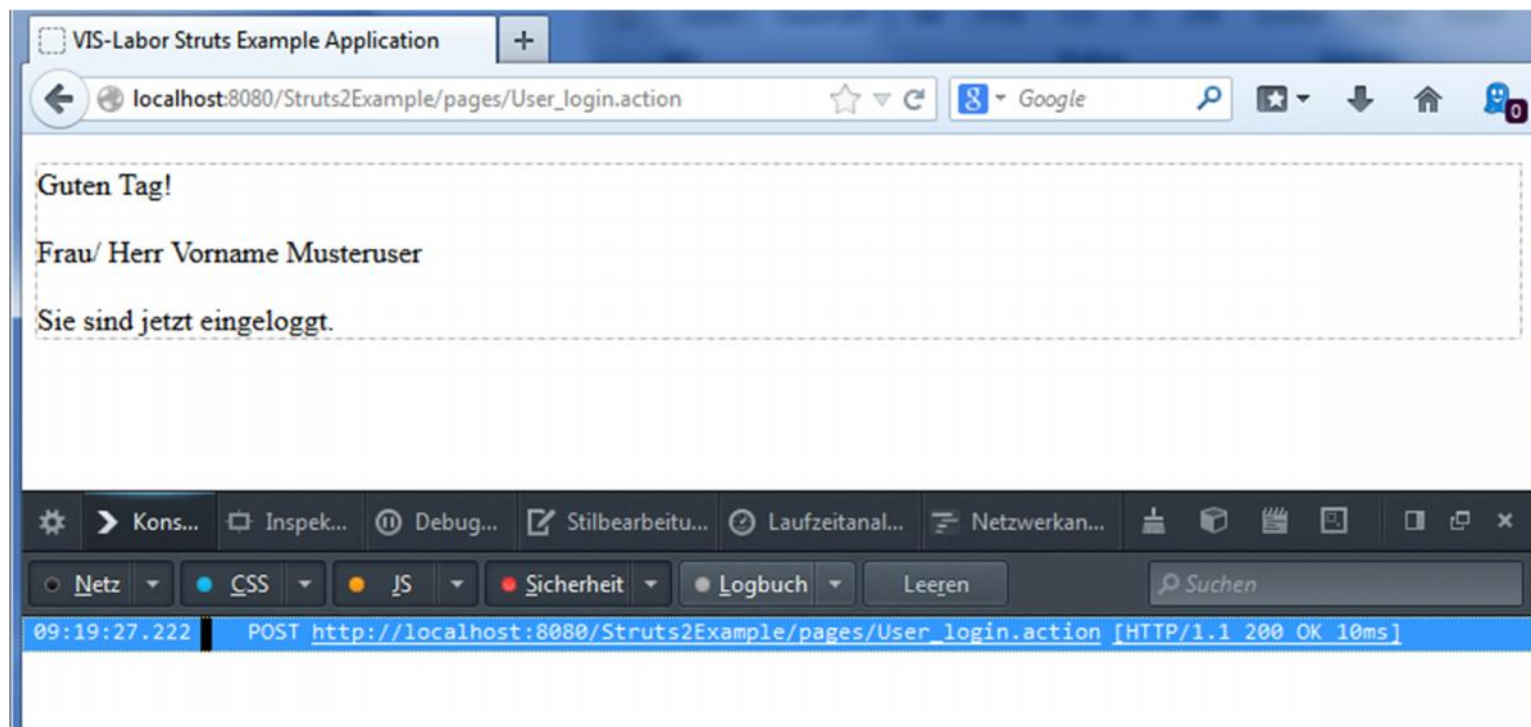
## Ablauf

- Properties aus JSP werden zu Request-Parametern  
`<s:textfield name="username"/>`  
`<s:password name="password"/>`
- Request 'User\_login' mit Parametern wird an Server gesendet  
`<s:form action="User_login"`

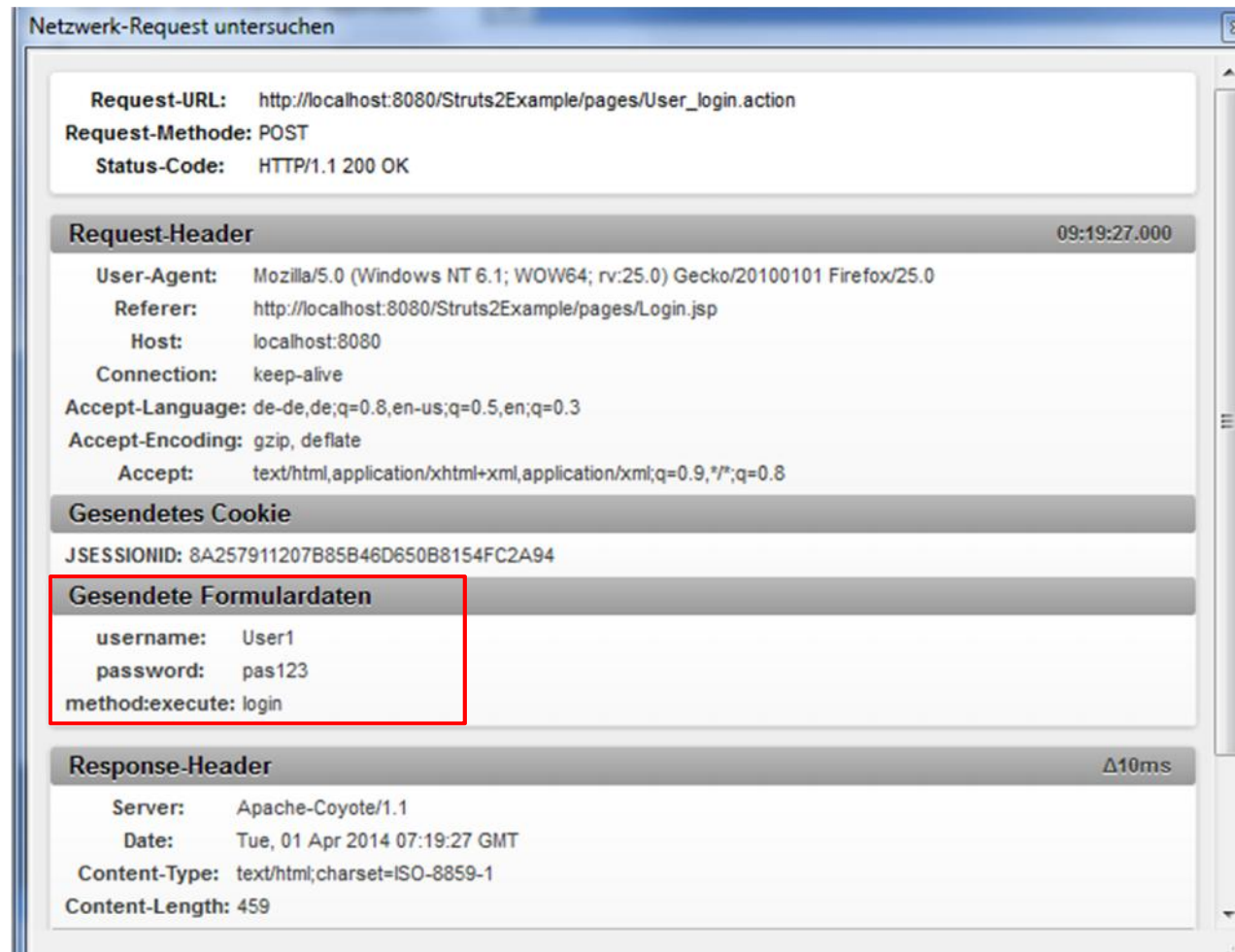


## Request-Parameter anzeigen

Im Firefox unter Web-Entwickler → Werkzeuge ein-/ausblenden



## Request-Parameter anzeigen



## Ablauf

- Properties aus JSP werden Request-Parameter
- Request User\_login an Server
- Server entscheidet anhand der web.xml:  
die Anfrage, wird an den Struts2 FilterDispatcher weitergeleitet

### Auszug web.xml

```
<!-- Filter Dispatcher Configuration -->
<filter>
  <filter-name>struts2</filter-name>
  <filter-class>
    org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
  </filter-class>
</filter>

<!-- Filter Mapping -->
<filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

- FilterDispatcher/ ActionMapper entscheidet anhand der struts.xml welche  
**ActionKlasse** aufgerufen wird

## struts.xml

struts.xml ist einer der zentralen Punkte des Struts2 Frameworks

```
<action name="action1"  
    class = "package1.subpackage1.ActionClass1" method="execute" >  
    <result name="success" >mypage.jsp</result>  
    <result name="input" >    input.jsp</result>  
    <result name="result1" >myresultpage1.jsp</result>  
</action>
```

execute method = default

## Ausschnitt struts.xml

(...)

```
<package name="vislabExample" extends="struts-default">  
  <action name="User_login"  
          class="vislabExample.controller.action.LoginAction"  
          <result name="input">/pages/Login.jsp</result>  
          <result name="success">/pages/welcome.jsp</result>  
        </action>  
</package>
```

(...)

in Login.jsp  
<s:form action="User\_login" >

## Ausschnitt struts.xml

(...)

```
<package name="vislabExample" extends="struts-default">
  <action name="User_login"
    class="vislabExample.controller.action.LoginAction"
    method="execute">
    <result name="input">/pages/Login.jsp</result>
    <result name="success"> /pages/success.jsp</result>
  </action>
</package>
```

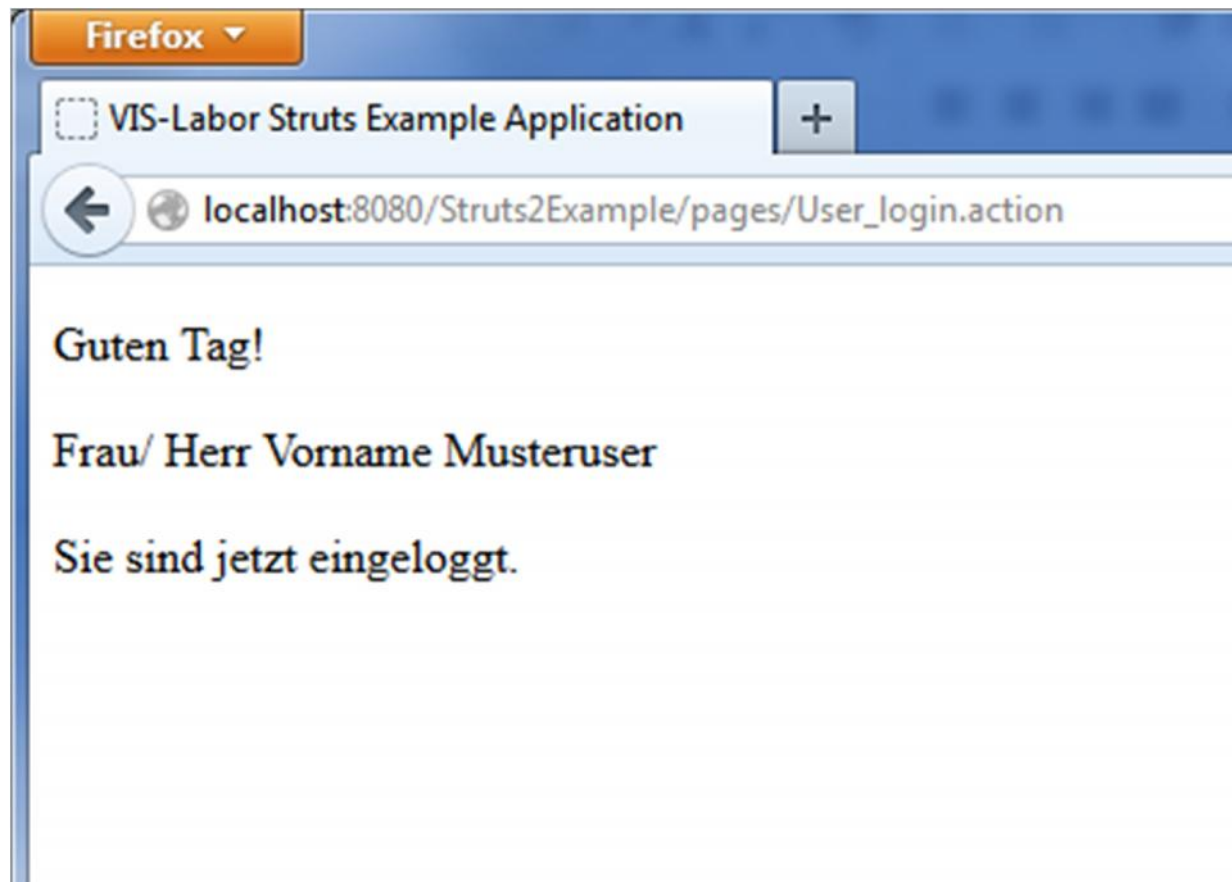
(...)

Die Methode "execute " ist die default-Methode und kann deshalb auch weggelassen werden.

## Ablauf

- Properties aus JSP werden Request-Parameter
- Request User\_login an Server
- Server entscheidet anhand der web.xml:  
alle Anfragen, werden an den Struts2 FilterDispatcher weitergeleitet
- FilterDispatcher entscheidet anhand der struts.xml welche  
ActionKlasse aufgerufen wird, die Action Klasse bearbeitet  
die Eingabe, produziert Ergebnisse und
- entscheidet anhand der Ergebnisse und der Definitionen in  
der struts.xml welche **View**, sprich Seite ausgegeben wird

## Beispiel View: JSP /Browser





## Struts Action Klasse

Action Klassen sind ein weiterer zentraler Punkt des Struts2 Frameworks

- Jeder Request/URL ist auf eine Methode einer Action gemappt
- Transfer der Daten von und zur View
- Validierung von Eingabedaten
- Aufruf der Methoden zur Verarbeitung der Anforderung (Businesslogik)
- Welche Seite soll als Ergebnis dargestellt werden

- 
- 
- 
- 

## • Struts Action Klasse

- muss eine execute – Methode definieren
- muss für jedes Eingabe-/ Ausgabefeld eine Variable (Property) mit Get-/Set-Methode definieren  
Struts wandelt HTTP Parameter in JavaBean Properties und füllt HTML Felder aus JavaBean Properties
- kapselt die Verarbeitung der Daten für einen Request
- Muss als Return-Wert einen String zurückgeben

## Beispiel property in Login.jsp

```
<%@ page contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <body>
    <s:text name="welcome.title" />

    <s:form action="User_login" focusElement="username">
      <s:textfield name="username" label="Benutzername"/>
      <s:password name="password" label="Passswort" /> <br>

      <s:submit method="execute" value="login" align="center"/>
    </s:form>
  </body>
</html>
```

## Action Class Beispiel

```
public class LoginAction extends ActionSupport {

    private String username;      // getter und setter fehlen aus Platzgründen
    private String password;      // getter und setter fehlen aus Platzgründen

    public String execute() throws Exception {

        if (getUsername().equals("user") {
            if (getPassword().equals("password")) {
                ..
                return SUCCESS;
            } else {
                addActionError(getText("error.user.passwordforgotten"));
                addActionError("Bitte geben Sie das richtige Passwort ein!");
                return "input";
            }
        }
        else { addActionError(getText("error.username.register"));
              return INPUT;
        }}
    }
```

## Action Class Beispiel

```
public class LoginAction extends ActionSupport {

    private String username;           // getter und setter fehlen aus Platzgründen
    private String password;           // getter und setter fehlen aus Platzgründen

    public String execute() throws Exception {

        if (getUsername().equals("user") {
            if (getPassword().equals("password")) {
                ..
                ..
                return SUCCESS;
            } else {
                addActionError(getText("error.user.passwordforgotten"));
                addActionError("Bitte geben Sie das richtige Passwort ein!");
                return "input";
            }
        }
        else { addActionError(getText("error.username.register"));
              return INPUT;
        }
    }
}
```

## Ausschnitt struts.xml

```
<action name="User_login" class="vislabExample.controller.LoginAction">  
    <result name="success">/pages/welcome.jsp</result>  
    <result name="input">/pages/Login.jsp</result>  
</action>
```

## Error Messages

In Action-Klasse

```
} else {  
    addActionError(getText("error.user.passwordforgotten"));  
    addActionError("Bitte geben Sie das richtige Passwort ein!");  
    return "input";  
}  
}  
else {  
    addFieldError("username", getText("error.username.register"));  
    return INPUT;  
}
```

In JSP

```
<font color="red">  
    <s:actionerror />  
</font>
```

## Beispiel Fehlerausgabe

Beispiel Login Seite

localhost:8080/Struts2Example/pages/User\_login.action

VIS-Labor Struts Example Application

Benutzername:

Passwort:

- Haben Sie Ihr Passwort vergessen?
- Bitte geben Sie das richtige Passwort ein!

Beispiel Login Seite

localhost:8080/Struts2Example/User\_login.action

VIS-Labor Struts Example Application

**Benutzername ist nicht registriert!**

Benutzername:

Passwort:



## Validierung der Eingabedaten

### Verschiedene Möglichkeiten zur Validierung

- XML Validierungskonfigurationsfiles  
Die Files müssen folgendermaßen benannt werden  
<action class>-validation.xml und müssen im selben Verzeichnis stehen, wie die Aktionklasse
- Definieren einer validate-Methode in der Action Klasse

## Beispiel Validierung XML File

```
<validators>
  <field name="username">
    <field-validator type="requiredstring">
      <message key="errors.required"/>
    </field-validator>
  </field>
  <field name="password">
    <field-validator type="requiredstring">
      <message>Bitte Passwort eingeben!</message>
    </field-validator>
  </field>
</validators>
```

Properties in Login.jsp

Validator Typen z.B.: requiredstring, email, date, int, url  
Für mehr siehe <http://struts.apache.org/docs/validation.html>

- 
- 
- 
- 

## • Beispiel validate – Methode in Action-Klasse

```
public void validate() {  
    if (!this.username.startsWith("Us")){  
        addFieldError("username", "Username muss mit Us beginnen!");  
    }  
}
```

## Beispiel message keys in JSP

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
.
<body>
  <s:text name="welcome.title" />

  <s:form action="User_login" focusElement="username">
    <s:textfield name="username" label="Benutzername"/>
    <s:password name="password" label="Passwort" /> <br>

    <s:textfield name="username" key="prompt.username" size="20"/>
    <s:password name="password" key="prompt.password" size="20"/>

    <s:submit method="execute" value="login" align="Center"/>
  </s:form>

  <font color="red">
    <s:actionerror label="label" />
  </font>
</body>
</html>
```

## MessageResource Files

muss im Classpath des Projekts stehen, wo wird in der struts.xml festgelegt,  
definiert key / value Paare

ApplicationResources\_de.properties

```
prompt.username=Benutzername  
prompt.password=Passwort  
button.submit=senden  
error.username.required=Benutzername eingeben  
error.username.register=Benutzername ist nicht registriert!  
error.benutzer.notrightuser=dieser User ist nicht zugelassen
```

ApplicationResources.properties

```
prompt.username=username  
prompt.password=password  
button.submit=submit
```

## struts.xml

```
<struts>
  .
  <constant name="struts.custom.i18n.resources" value="ApplicationResources" />
  .
  <!-- Action Definitions -->
    <package name="package1" extends="struts-default">
      </package>
      .
      .
      .
    <package name="packageM" extends="struts-default">
      </package>
    .
  .
</struts>
```

Struts Tag Libraries stellen Tags zur Verfügung, um JSP-Seiten mit Formularen, Listen, Tabellen aufzubauen und zu strukturieren:

- Form tags, z.B.:
  - s:form
  - s:submit
  - s:textfield
  - s:password
- Data Tags , z.B.:
  - s:property
  - s:text
- Control Tags , z.B.:
  - s:if zum Testen von Werten
  - s:else
  - s:iterator um Listen auszugeben
- nonForm UI Tags , z.B.:
  - s:actionerror
  - s:actionmessage

Werden über folgende Anweisung in die JSP-Seiten eingefügt:  
`<%taglib prefix="s" uri="/struts-tags" %>`

<http://struts.apache.org/docs/struts-tags.html>

## Beispiel Login.jsp

```
<%@ page contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <body>
    <s:text name="welcome.title" />

    <s:form action="User_login" focusElement="username">

      <s:textfield name="username" key="prompt.username" />
      <s:password name="password" key="prompt.password" /> <br>

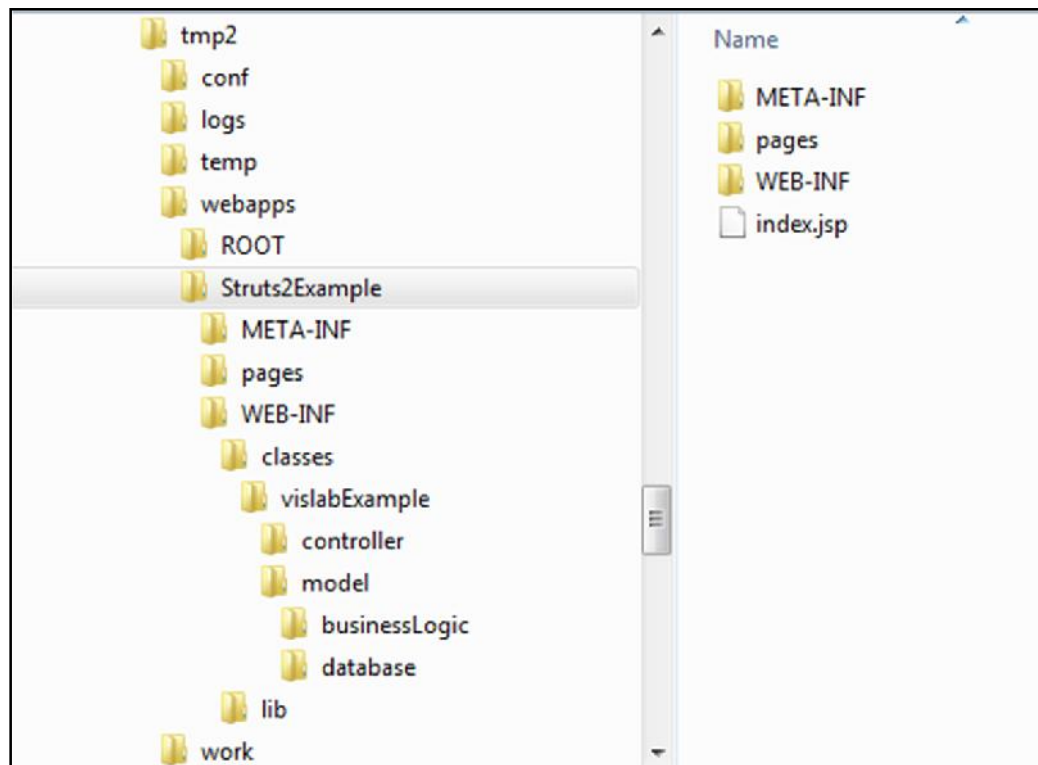
      <s:submit method="execute" label="Login" align="center"/>
    </s:form>
  </body>
</html>
```



## Apache Tomcat Verzeichnisstruktur

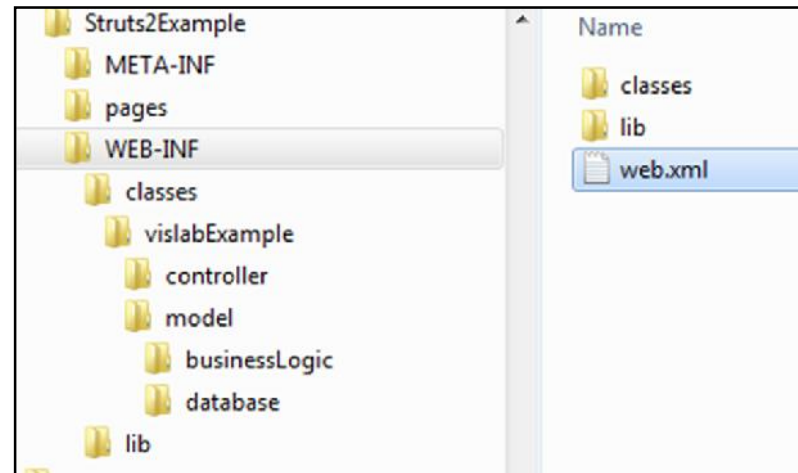
Apache Tomcat ist ein Open Source Webserver und Webcontainer, mit dem in Java geschriebene Web-Anwendungen auf Servlet- bzw. JSP-Basis ausgeführt werden können.

### Verzeichnisstruktur einer Web-Applikation



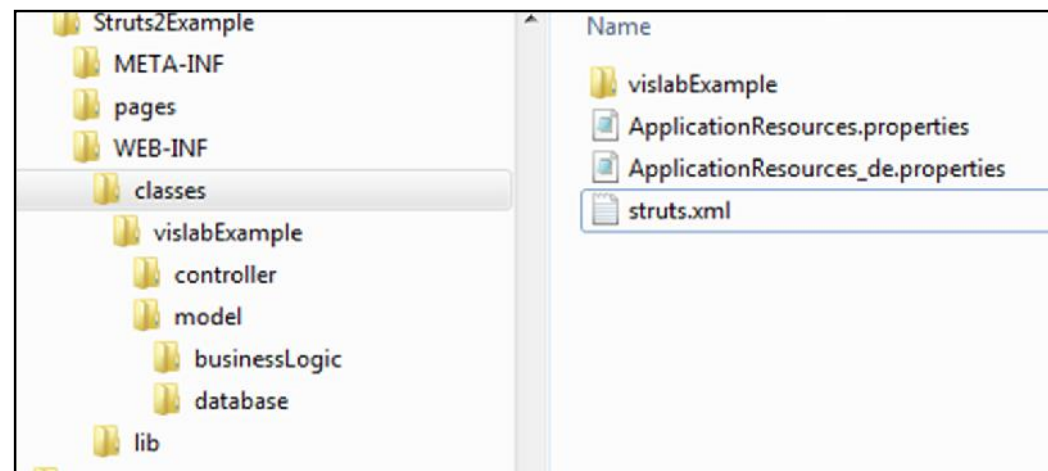
## Struts configuration files

web.xml



struts.xml

Resource Files



## Zusammenfassung

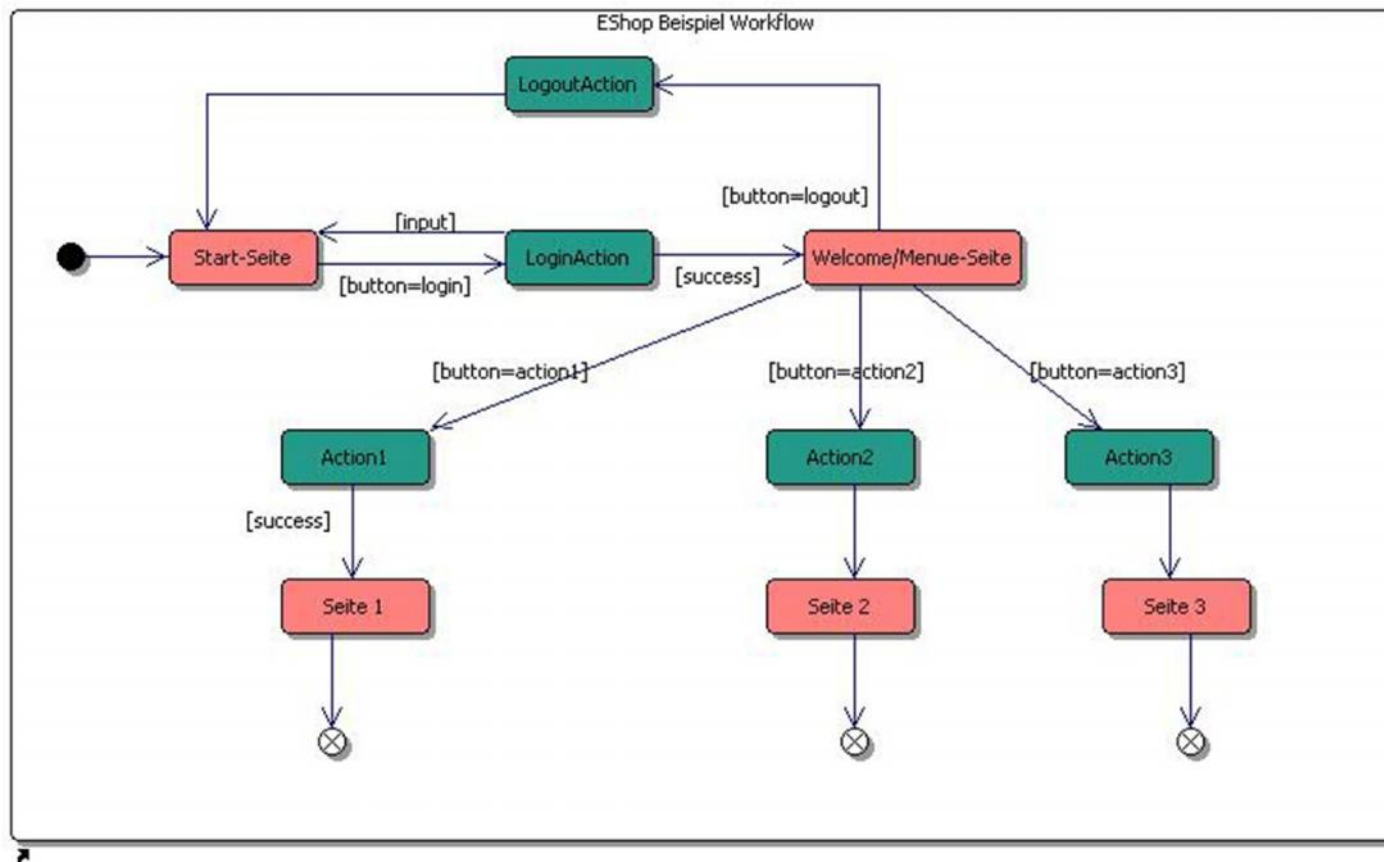
Das Model View Controller Design erfordert die Entwicklung von 3 Komponenten:

- Model: Geschäftslogik mit Domain-Objekten
- View: User Interface, d.h. Präsentation der Daten
- Controller: verbindet Model und View, steuert den Ablauf der Anwendung

## Vorgehensweise

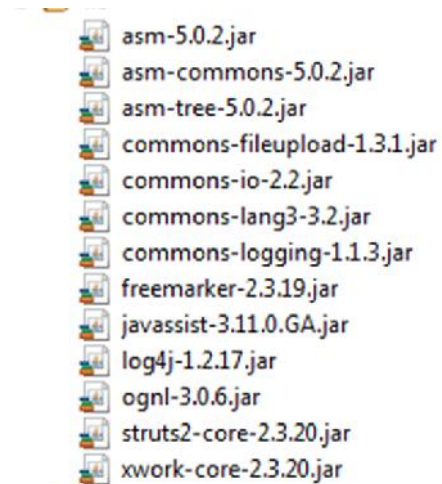
- Welche JSP-Seiten werden benötigt?
- Welche Eingabefelder (Properties) werden in den jeweiligen Seiten benötigt?
- Zu jedem Request eine ActionClass bzw. Methode erstellen.
- Zu allen Eingabefeldern der JSP-Seite in der zugehörigen ActionClass Properties mit get- und set-Methoden erstellen
- Erstellen bzw. konfigurieren der struts.xml  
Wie soll der Ablauf aussehen, d.h. welche Eingaben oder Verarbeitungsergebnisse haben welche Seite als Folge
- In der ActionClass die Eingabedaten verarbeiten, abhängig vom Ergebnis bzw. des Ablaufs ein result erzeugen

## Beispiel Workflow



## Benötigte Software für Struts-Teil der Aufgabe

Struts2 (neueste Version 2.3.20)



Apache Tomcat (neueste Version 8.0.20)

## Links

<https://struts.apache.org/index.html>

[http://www.tutorialspoint.com/struts\\_2/index.htm](http://www.tutorialspoint.com/struts_2/index.htm)

<http://viralpatel.net/blogs/2009/12/tutorial-create-struts-2-application-eclipse-example.html>

<http://www.roseindia.net/struts/index.shtml>

<http://www.roseindia.net/struts/struts2.3.15.1/index.shtml>

<http://www.javajazzup.com/issue5/page31.shtml>

<http://struts.apache.org/2.x/docs/validation.html>

<https://cwiki.apache.org/confluence/display/WW/Tag+Reference>

<http://www.mkyong.com/struts2/struts-2-i18n-or-localization-example/>

<http://www.brucephillips.name/blog/index.cfm/2008/10/10/Using-the-Struts-2-Key-Attribute>

<http://www.mkyong.com/struts2/struts-2-iterator-tag-example/>

<http://www.mkyong.com/struts2/struts-2-hello-world-example/>

Beispiel-Projekt in Ilias unter >> Labor Verteilte Informationssysteme