

Verteilte Systeme – Web-Programmierung Übungen

Diplom Wirtschaftsinformatiker (BA) Thomas Pohl & Christian Romeyke
<https://github.com/chromey/dhbw-exercises>

Version: 09

Agenda

1 Setup

2 Spring Boot Übungen

3 Servlet Übungen

4 „Chat Forum“

5 HTML, CSS

6 Anderes

S01 Einrichten der Entwicklungsumgebung

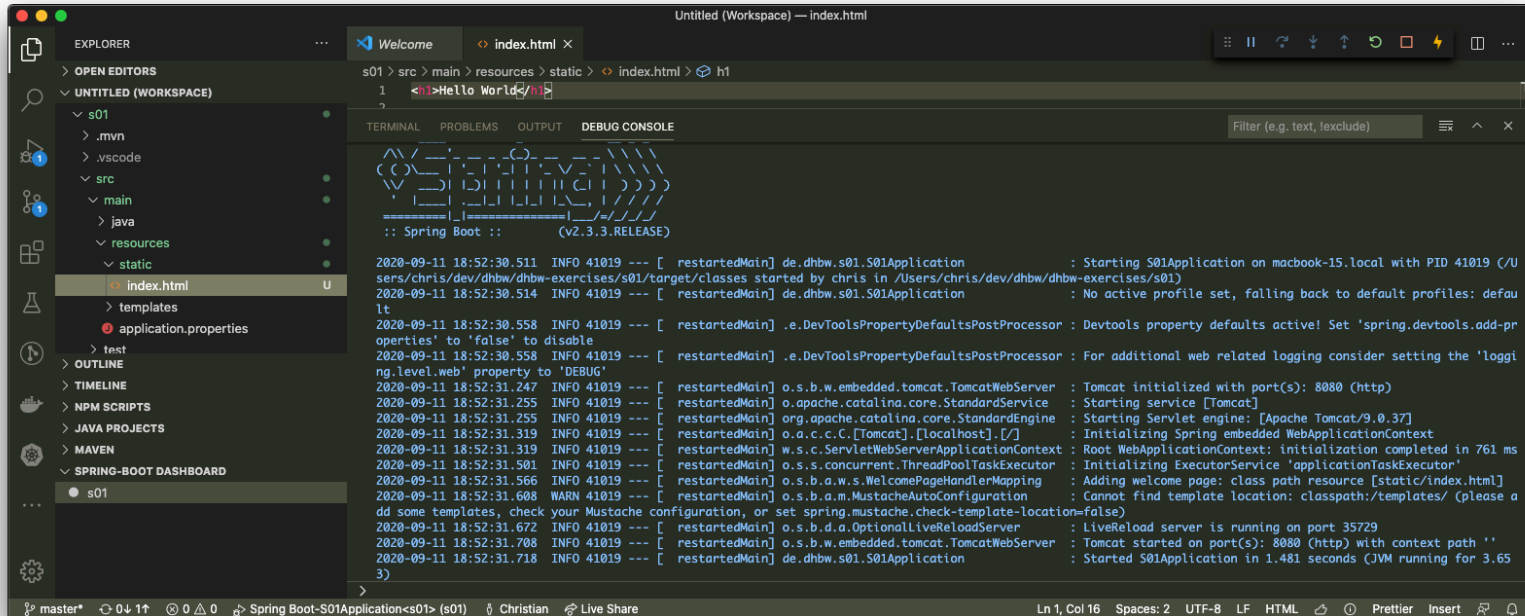
- In VS Code
 - File (Mac: Code) -> Preferences -> Extensions
 - Suchen und Installieren von
 - Java Extension Pack
 - Spring Boot Extension Pack
 - FreeMarker
 - Installation abwarten, dann View -> Command Palette -> Java: Configure Runtime
 - Ein JDK in der Version ≥ 11 wird benötigt. Wird unter „Configure“ keines angezeigt (Problem sollte rot hervor gehoben sein), dem Download-Button unter „Install“ folgen, herunterladen und installieren.
 - Windows: im Installer die Option „JAVA_HOME-Variable konfigurieren setzen“
 - VS Code beenden und neu starten (Reload Window reicht nicht immer)

S01 Anlegen einer neuen Applikation

- View -> Command Palette (oder Ctrl / Cmd + Shift + P)
- (Jeweils suchen und mit Enter bestätigen):
 - Spring Initializr -> Create a Maven Project ...
 - Specify Spring Boot version: 2.3.3
 - Specify project language: Java
 - Input Group Id for your project: de.dhbw
 - Input Artifact Id for your project: s01
 - Specify packaging type: JAR
 - Specify Java version: 1.8
 - Search for dependencies:
 - Spring Boot DevTools
 - Spring Web
 - Apache FreeMarker
- Wählen Sie ein Verzeichnis aus, in welches das Projekt (und später weitere Projekte) generiert werden soll
- Im erscheinenden Popup auf „Add to Workspace“ klicken **OR OPEN**
- Die beiden Popups („Import Java projects?“, „Trust Maven Wrapper“) mit **„Always“** oder „Yes“ bestätigen

S01 Ändern und Ausführen der Anwendung

- Im Projekt unter src/main/resources/static eine Datei namens „index.html“ anlegen mit beliebigem (HTML) Inhalt
- Über das Spring Boot Dashboard im Explorer die Anwendung starten und die Ausgabe in der Debug Console beobachten



```
s01 > src > main > resources > static > index.html > h1
1 <h1>Hello World</h1>

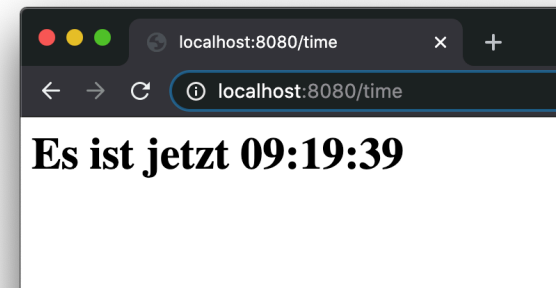
:: Spring Boot :: (v2.3.3.RELEASE)

2020-09-11 18:52:30.511 INFO 41019 --- [ restartedMain] de.dhbw.s01.S01Application : Starting S01Application on macbook-15.local with PID 41019 (C:/Users/chris/dev/dhbw/dhbw-exercises/s01/target/classes started by chris in /Users/chris/dev/dhbw/dhbw-exercises/s01)
2020-09-11 18:52:30.514 INFO 41019 --- [ restartedMain] de.dhbw.s01.S01Application : No active profile set, falling back to default profiles: default
2020-09-11 18:52:30.558 INFO 41019 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-pr
2020-09-11 18:52:30.558 INFO 41019 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'loggi
2020-09-11 18:52:31.247 INFO 41019 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-09-11 18:52:31.255 INFO 41019 --- [ restartedMain] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-09-11 18:52:31.255 INFO 41019 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.37]
2020-09-11 18:52:31.319 INFO 41019 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-09-11 18:52:31.319 INFO 41019 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 761 ms
2020-09-11 18:52:31.501 INFO 41019 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-09-11 18:52:31.566 INFO 41019 --- [ restartedMain] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page: class path resource [static/index.html]
2020-09-11 18:52:31.608 WARN 41019 --- [ restartedMain] o.s.b.a.m.MustacheAutoConfiguration : Cannot find template location: classpath:/templates/ (please a
2020-09-11 18:52:31.672 INFO 41019 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2020-09-11 18:52:31.708 INFO 41019 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-09-11 18:52:31.718 INFO 41019 --- [ restartedMain] de.dhbw.s01.S01Application : Started S01Application in 1.481 seconds (JVM running for 3.65
3)
```

- Im Browser <http://localhost:8080> aufrufen: Ihre HTML-Seite sollte zu sehen sein

SB02 – Zeitanzeige

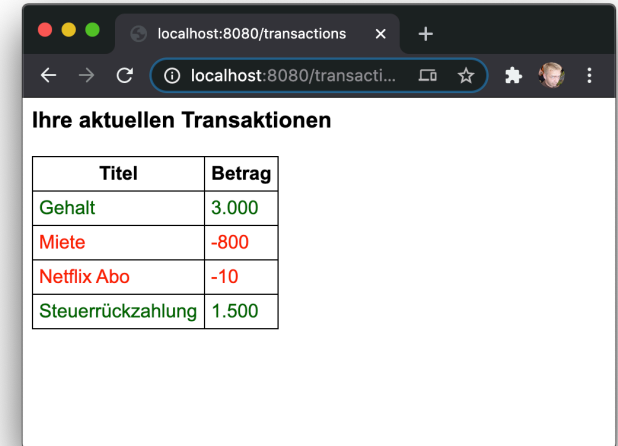
Erstellen Sie eine Anwendung, die bei jedem Zugriff die aktuelle Zeit anzeigt.



- Wiederholen Sie die Schritte aus „**S01 Anlegen einer neuen Applikation**“ mit der Artifact ID „s02“
 - die richtigen Dependencies sollten schon vorausgewählt sein
 - Generieren Sie die Applikation in den gleichen Ordner wie S01 (nicht *in* S01) und fügen Sie es zum Workspace hinzu
- Legen Sie einen Controller an, der für den HTTP-Pfad /time die aktuelle Uhrzeit ermittelt und in das Model setzt
 - **Siehe** `java.util.Date`, `java.text.SimpleDateFormat`
- Erstellen Sie ein zugehöriges FreeMarker Template
- Starten Sie die Anwendung

SB03 – Konto

Erstellen Sie eine Anwendung, die Kontotransaktionen (Einzahlungen, Auszahlungen) tabellarisch auflistet und farblich abhebt (Einzahlungen grün, Auszahlungen rot).



Ihre aktuellen Transaktionen

Titel	Betrag
Gehalt	3.000
Miete	-800
Netflix Abo	-10
Steuerrückzahlung	1.500

- Eine Transaktion können wir als einfaches POJO mit den Attributen „titel“ und „betrag“ modellieren
- Die Liste der Transaktionen kann direkt im Controller erstellt werden
- Zusatzaufgabe: Lagern Sie das notwendige CSS in eine andere Datei aus. Hierfür kennen wir jetzt zwei alternative Möglichkeiten. Welche sind das und wie unterscheiden Sie sich?

Agenda

1 Setup

2 Spring Boot Übungen

3 Servlet Übungen

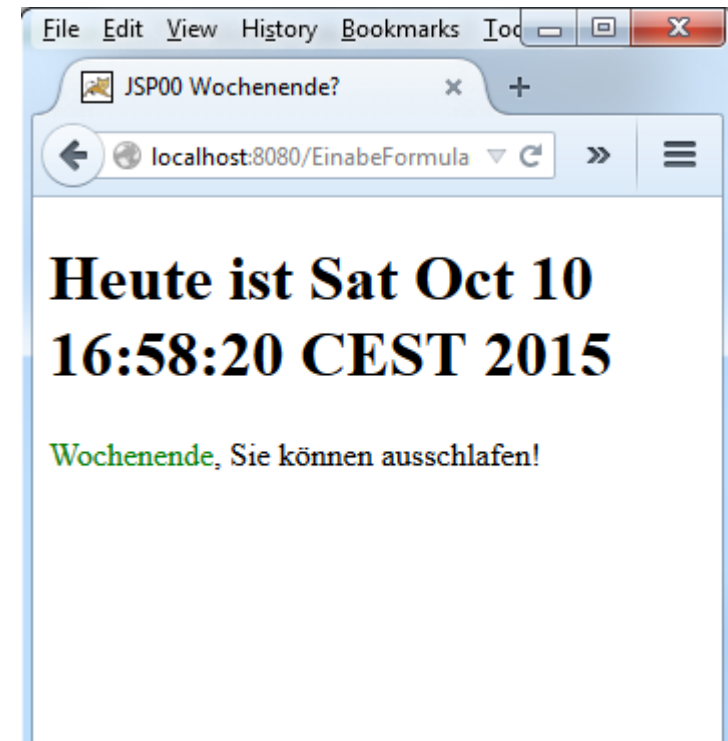
4 „Chat Forum“

5 HTML, CSS

6 Anderes

JSP00 Wochenende?

- Erstellen eines Java Web/Web Application-Projekts
- Erstellen einer „index.jsp“
- Abhängig vom Wochentag gibt die Anwendung aus:
 - Wochenende, Sie können ausschlafen! oder
 - Arbeitstag, Sie müssen aufstehen!
- Ausliefern des Projekts
- Start der ausgelieferten Seite im Browser
- HTML & JSP Kommentar einfügen und das Verhalten untersuchen!



JSP00 Wochenende? - Quellcode

```
...  
<body>  
<%!  
private static boolean isWeekend(){  
    java.util.Calendar calendar = new java.util.GregorianCalendar();  
    int day = calendar.get(Calendar.DAY_OF_WEEK);  
    return (day == Calendar.SATURDAY) | (day == Calendar.SUNDAY);  
}  
%>  
  
<!-- Dies ist ein JSP-Kommentar, der nicht übertragen wird --%>  
<!-- Dies ist ein HTML-Kommentar, der übertragen wird -->  
  
<h1>Heute ist <%= new java.util.Date() %></h1>  
  
<% if(isWeekend()){ %>  
    <span style="color: green">Wochenende</span>, Sie können ausschlafen!  
<% } else { %>  
    <span style="color: red">Arbeitstag</span>, Sie müssen aufstehen!  
<% } %>  
<br/>  
</body>  
...
```

Deklaration

JSP/ HTML Kommentar

Anweisung

Scriptlet

JSP02 - Ein JSP mit Scripting-Elementen (Zugriff auf implizite Objekte)

Der Zugriff auf die sogenannten „impliziten Objekte“ steht im Mittelpunkt dieser Aufgabe. Die Liste der impliziten Objekte finden sich auf den „JSP Reference Cards“. Es handelt sich um vordefinierte Objekte, auf die mittels (vordefinierter) Methoden zugegriffen werden kann. Vordefinierte oder implizite Variablen sind z.B. *request*, *response*, *session*, *out*, *application*.

Teil 1

- Erstellen Sie ein „Dynamic Web Application“ – Projekt, das aus einer *index.jsp* Datei besteht.
- Editieren Sie die *index.jsp* Datei und implementieren Sie den Zugriff auf eine vordefinierte bzw. implizite Variable.
- Wie kann das Ergebnis des Zugriffs angezeigt werden?
- Probieren Sie den Zugriff auf weitere Objekte und die Ausgabe der Informationen.

Tipp: Zugriffsmethode auf *session* ist z.B. *getId()* oder auf *application* ist z.B. *getServerInfo()*. Versuchen Sie zuerst diese zu implementieren.

Teil 2: Suchen Sie im Unterverzeichnis des Tomcat: „work\Catalina\localhost\<projekt>\org\apache\jsp“ das erzeugte Servlet und versuchen sie den Code zu verstehen.

JSP04 - Nutzung von Variablen

Die Deklaration und die Nutzung einer (lokalen) Variable ist das Ziel dieser Aufgabe. Die Variable notiert bzw. speichert jeden Zugriff auf die Seite, d.h. jeder Seitenzugriff wird (ab)gespeichert.

1. Erstellen Sie ein „Web Application“ – Projekt, das aus einer index.jsp Datei besteht.
2. Deklarieren Sie die Variable für die Speicherung der Anzahl der Seitenzugriffe.
3. Implementieren Sie die automatische Erhöhung der Anzahl der Seitenzugriffe.
4. Lesen Sie Daten aus den impliziten, vordefinierten Objekten, speichern Sie diese in Variablen, verändern Sie diese (z.B. in Kombination mit anderen) und geben Sie diese dann aus.

JSP05 - Ausgabe von Zufallszahlen

Erzeugen Sie eine konstante Anzahl von Zufallszahlen, die Sie jeweils in einer Liste ausgeben. Danach geben Sie eine zufällige Anzahl von Zufallszahlen in einer Liste aus.

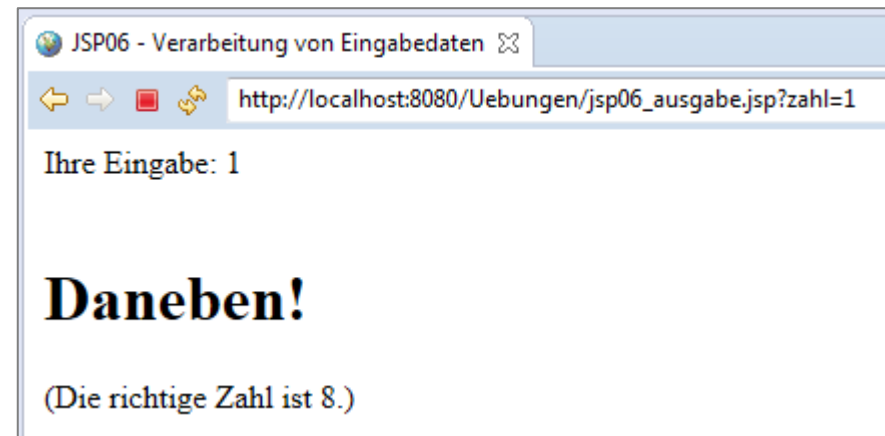
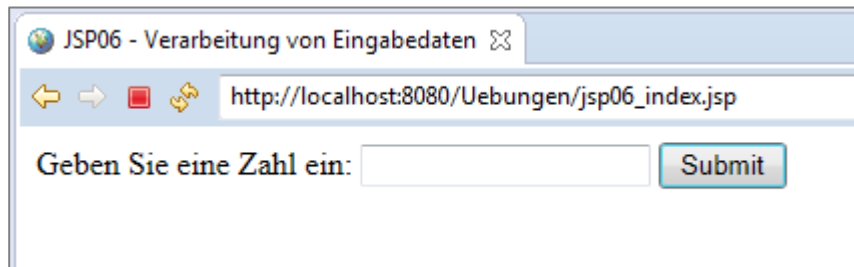
Erstellen Sie ein „Dynamic Web Application“ – Projekt, das aus einer index.jsp Datei besteht.

1. Erzeugen Sie eine (konstante) Anzahl von Zufallszahlen und geben Sie diese dann aus.
2. Variieren Sie das Programm, in dem die Anzahl der Zufallszahlen zufallsabhängig ist.
3. Variieren Sie im Programm: „HTML mit Java“ vs. „Java mit HTML“

```
public static int ganzeZufallszahl(int bereich) {  
    return (1 + ((int) (Math.random() * bereich)));  
}
```

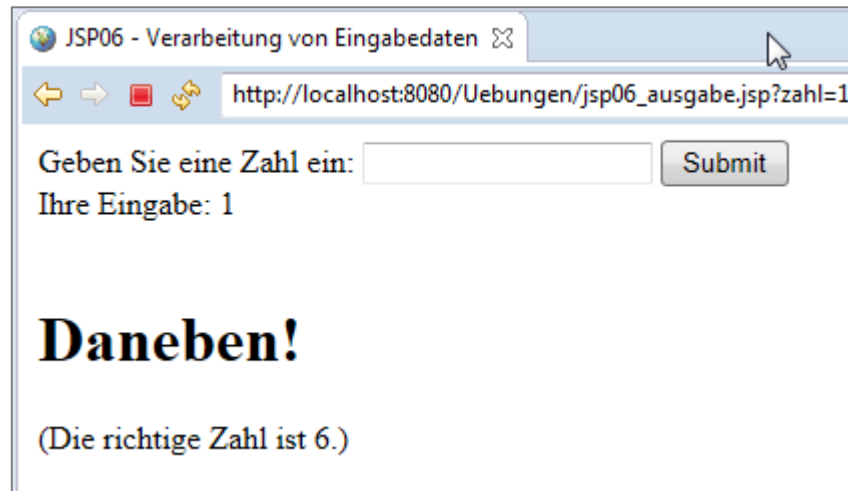
JSP06 – Verarbeitung von Eingabedaten

- Erstellen eines Java Web/Web Application-Projekts
- Erstellen einer „index.jsp“ mit einem Eingabeformular, welches den Parameter „zahl“ übermittelt
- Erstellen einer ausgabe.jsp, welche
 - Den Parameter ausliest und anzeigt
 - In eine Zahl umwandelt
 - Mit einer Zufallszahl vergleicht das Ergebnis ausgibt



JSP06 – Verarbeitung von Eingabedaten - Erweiterung

- Verändern Sie die Methode (get/ post) und untersuchen Sie das Verhalten
- Fügen Sie das Eingabeformular in der „ausgabe.jsp“ ein, so dass nur noch eine JSP Benötigt wird. Was ist die Herausforderung?



JSP06 - Verarbeitung von Eingabedaten

http://localhost:8080/Uebungen/jsp06_ausgabe.jsp?zahl=1

Geben Sie eine Zahl ein: Submit

Ihre Eingabe: 1

Daneben!

(Die richtige Zahl ist 6.)

JSP06 – Verarbeitung von Eingabedaten - Quellcode

```
<title>JSP06 - Verarbeitung von Eingabedaten</title>
<%!
public static int ganzeZufallszahl(int bereich) {
return (1 + ((int) (Math.random() * bereich)));
}
%>
</head>
<body>
<form method=get>
Geben Sie eine Zahl ein: <input type=text name=zahl> <input
type=submit value="Submit">
</form>

Ihre Eingabe: <%=request.getParameter("zahl")%><br/><br/>

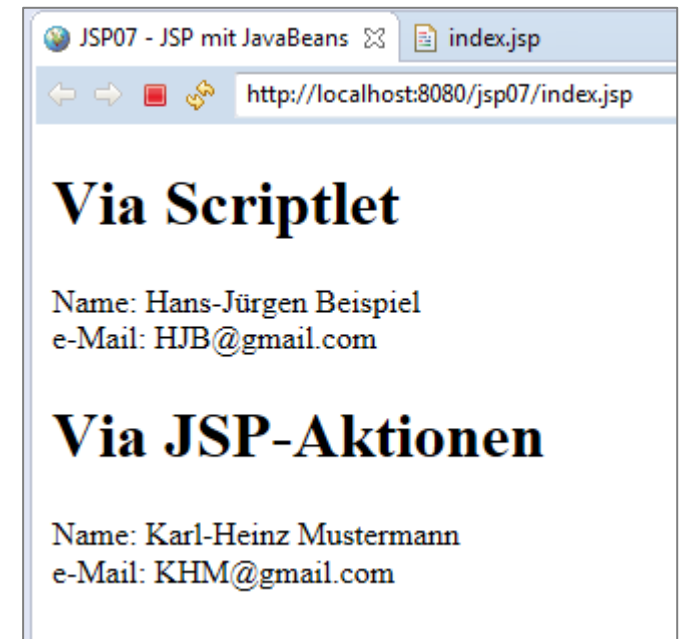
<%
int zufallszahl = ganzeZufallszahl(10);

int eingabeWert = 0;
try {
eingabeWert = Integer.parseInt(request.getParameter("zahl"));
} catch (NumberFormatException ex) {
%>
Fehlerhafte Eingabe!
```

```
<%
}
if (zufallszahl == eingabeWert) {
%>
<h1>Treffer!</h1>
<%
} else {
%>
<h1>Daneben!</h1>
(Die richtige Zahl ist <%=zufallszahl%>.)
<%}%>
</body>
```


JSP07 - JSP mit JavaBeans

- JavaBean „Person“ anlegen
 - Attribute: „Name“ und „eMail“
 - Package: org.dhbw.jee.bean
 - Schreiben Sie einen parameterlosen Standard-Konstruktor und get-/set-Zugriffsmethoden für alle Attribute.
- „Index.jsp“ erstellen welche:
 - Das „Person“ instanziiert und mit Werten belegt und
 - Diese Werte als HTML ausgibt.
 - Jeweils eine Instanz von Person: „normal“ via Scriptlet und via JSP-Aktion
- Mischen Sie die beiden Varianten und untersuchen das Verhalten.



JSP07 - JSP mit JavaBeans - Quellcode

<h1>Via Scriptlet</h1>

```
<% Person personABC = new Person();  
personABC.setName("Hans-Jürgen Beispiel");  
personABC.setEmail("HJB@gmail.com"); %>
```

Name: <%=personABC.getName() %>

e-Mail: <%=personABC.getEmail() %>

<h1>Via JSP-Aktionen</h1>

```
<jsp:useBean id="person0815" scope="session" class="org.dhbw.jee.bean.Person"/>  
<jsp:setProperty property="name" name="person0815" value="Karl-Heinz Mustermann"/>  
<jsp:setProperty property="email" name="person0815" value="KHM@gmail.com"/>
```

Name: <jsp:getProperty property="name" name="person0815"/>

e-Mail: <jsp:getProperty property="email" name="person0815"/>

JSP07 - JSP mit JavaBeans - Benutzereingaben

Erweitern Sie das Beispiel: Auslesens eines Parameters der Benutzerabfrage und direkte Zuweisung an eine JavaBean Instanz

- Separate „index.html“ erstellen, welche ein Eingabeformular enthält für „Name“ und „e-Mail“
- „index.jsp“ so anpassen, dass die Werte aus dem Eingabeformlar angezeigt werden
 - Das Auslesen der Parameter einmal via Scriptlet und einmal via JSP-Aktion



JSP07 - JSP mit JavaBeans - Benutzereingaben

```
<h1>Aus Benutzereingabe</h1>
<jsp:useBean id="personEingabe" scope="session"
class="org.dhbw.jee.bean.Person"/>
<jsp:setProperty name="personEingabe" property="name"
param="vorname"/>
<jsp:setProperty name="personEingabe" property="email"
value='<%=request.getParameter("e-mail")%>' />

Name: <jsp:getProperty property="name" name="personEingabe"/><br/>
e-Mail: <jsp:getProperty property="email" name="personEingabe"/>
```

- Variieren Sie die Nutzung der Parameter von „setProperty“
 - `<jsp:setProperty name=".." property=".." param=".."/>`
 - (*param weglassen, property="*" , ...*)
- *Erweitern Sie die Bean um ein Attribut was nicht vom Typ String ist*

JSP10 - JSP mit Direktiven [include]

Die Direktive include erlaubt die Integration von HTML-Seiten.

Erstellen Sie ein „Web Application“ – Projekt, das aus einer index.jsp Datei besteht.

1. Implementieren Sie in der index.jsp-Seite eine include-Direktive, die eine andere JSP-Seite aufnimmt, in der – über ein Skriptlet – die aktuelle Systemzeit ausgegeben wird.

JSP11 - „isErrorPage“ und JSP<>Servlet

- index.html
 - Formular mit Name/ E-Mail
- Greetings.jsp
 - Überprüft ob Eingabe Länger als „0“
 - Hat „error page“ definiert
 - Gibt im Erfolgsfall Eingabedaten aus
- errorPage.jsp
 - Flag „isErrorPage“ auf true
 - Gibt Details des Fehlers aus
- Suchen Sie im Unterverzeichnis des Tomcat: „work\Catalina\localhost\<projekt>\org\apache\jsp“ das erzeugte Servlet und finden Sie die Abbruchbedingung.. Warum wird die ErrorPage nicht angezeigt?
- Fügen sie in der errorPage.jsp eine Ausgabe ein „`System.out.println("Der Code wird durchlaufen!");`“ und debuggen Sie die JSP
- Fügen Sie „`<% response.setStatus(javax.servlet.http.HttpServletResponse.SC_OK); %>`“ in die errorPage.jsp ein

greetings.jsp

```
<%@page errorPage="errorPage.jsp" %>
...
userName.charAt(0);
...
```

errorPage.jsp

```
<%@ page isErrorPage="true" %>
...
<%= exception.getMessage() %>
```

JSP12 - Nutzen des TCP/IP Monitors

- Debuggen der http-Kommunikation
 - Window > Show view > Other > TCP/IP Monitor
 - Context Menü im oberen Fenster „Properties“
 - Local port: 80
 - Host name: localhost
 - Port: 8080 (Tomcat port)
 - Type: TCP/IP
 - Start monitor automatically: checked
 - Im web browser die gleiche URL wie zuvor aufrufen – jedoch mit dem Port 80
- (Basis Übung JSP 11) Untersuchen Sie die „errorPage“ mit und ohne „responst.setStatus ...“. Achten Sie auf die Informationen des HTTP heads
- Vergleichen Sie die Anzeige „errorPage“ ohne „responst.setStatus ...“ im Firefox und im Internet Explorer
- Optional: Nutzen Sie den TCP/IP Monitor überkreuz mit ihrem Nachbar

JSP13 – Forward JSP (mit JavaBeans)

- Erstellen Sie: index.html, fillbean.jsp, greetings.jsp und eine JavaBean
- Index.html – enthält ein Eingabeformlar
- fillbean.jsp – erzeugt eine Bean und füllt diese mit den Eingabeparametern
 - Forward nach greetings.jsp
- greetings.jsp – gibt die Werte der Bean aus
- Wie wird die Bean von fillbean nach greetings transportiert?
 - Testen Sie greetings.jsp und fillbeans.jsp direkt aufzurufen.
 - Ändern Sie den verwendeten Scope.

JSP14 – Tabelle

- Über eine Eingabeseite werden die Anzahl Zeilen/ Spalten festgelegt
 - Eine JSP stellt eine entsprechende Tabelle dar
 - In jeder Zelle werden die Koordinaten ausgegeben
 - Sollten die eingegebenen Koordinaten keine ganzen Zahlen sein, wird der Request auf die Ursprungsseite zurückgeleitet. Diese zeigt dann einen Fehler an.
-
- `Int columns = (new Integer(request.getParameter("columns"))).intValue();`

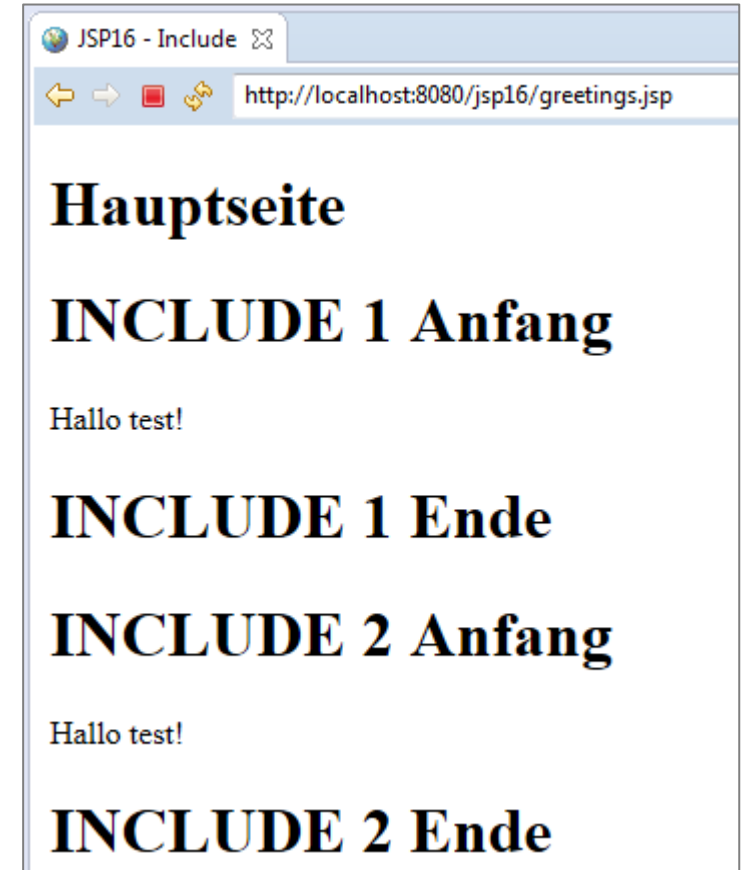
JSP15 – Kontexte

- Erstelle Sie eine JSP mit einem Text-Eingabefeld und einem Submit-Button
- Die gleiche JSP verarbeitet die Anfrage und legt den übertragenen Parameter jeweils im Request, Session und Application-Kontext ab.
- Bei jeder weiteren Anfrage die in den Kontexten abgelegten Attribute jeweils um den Eingabewert erweitert.
- Untersuchen Sie das Verhalten mit wiederholten Anfragen
 - Aus unterschiedlichen Browsern
 - Browser im Incognito-Modus
 - Aus unterschiedlichen Tabs eines Browser
 - Nach dem Schließen eines Browsers

```
...  
String parameter = request.getParameter("test");  
  
request.setAttribute("test", concatAttribute((String)request.getAttribute("test"),  
parameter));  
...
```

JSP16 – Include

- Übergeben Sie von einer index.html einen Parameter „name“ an eine greetings.jsp
- In greetings.jsp sind zwei verschiedene weitere JSPs eingebunden:
 - mit der Direktive „include“ und
 - mit der jsp action „jsp:include“
- In den eingebunden JSPs erzeugen Sie in einem Scriptlet einen „Gruß“
 - `String gruss = "Hallo " + request.getParameter("name") + "!";` und geben diesen aus.
- Untersuchen Sie das Verhalten!
- Was passiert wenn beide Include mehrfach in greetings.jsp verwendet werden?
- Nutzen Sie die Parameterübergabe der jsp-action.
- Untersuchen Sie den erzeugten Servlet-Code im Verzeichnis des Tomcat-Servers: `work\catalina\...`



JSP16 – Include - Quelltext

Greeting.jsp

```
...  
<title>JSP16 - Include</title>  
</head>  
<body>  
<h1>Hauptseite</h1>  
<%@ include file="include1.jsp" %>  
  
<jsp:include page="include2.jsp"/>  
  
<jsp:include page="include2.jsp"><jsp:param  
value="abc" name="name"/></jsp:include>  
  
</body>  
</html>
```

include1.jsp

```
<h1>INCLUDE 1 Anfang</h1>  
  
<% String hallo = "Hallo " +  
request.getParameter("name") + "!"; %>  
  
<%=hallo %>  
  
<h1>INCLUDE 1 Ende</h1>
```

include2.jsp

```
<h1>INCLUDE 2 Anfang</h1>  
<% String hallo = "Hallo " +  
request.getParameter("name") + "!"; %>  
  
<%=hallo %>  
  
<h1>INCLUDE 2 Ende</h1>
```

JSP17 – Forward / Redirect

- Erstellen Sie eine index.html mit zwei gleichen Eingabefeldern.
 - Jeweils: Name (Text) + Submit Button
- Erstellen Sie zwei verarbeiten1/2.jsp
 - Beide erzeugen aus der Eingabe einen Gruß inklusive Uhrzeit (z.B. Hallo *Du!* Uhrzeit: Sun Feb 16 20:23:10 CET 2014) und legen diesen in der Session ab.
 - Beide verweisen zur Verarbeitung auf die gleiche ergebnis.jsp, welche den Gruss anzeigt.
 - verarbeiten1.jsp → Verweis via Forward
 - verarbeiten2.jsp → Verweis via Redirect
- Untersuchen Sie das Verhalten – insbesondere bei Benutzung des Reload- und Back-Buttons des Browsers!
- Zusatzaufgabe:
 - Fügen sie Text vor der dem Forward in verarbeiten1.jsp ein. Was geschieht mit der Ausgabe?
 - Fügen Sie sehr, sehr viel Text ein. Analysieren Sie das Verhalten und ggf. die Fehlermeldung.
`<jsp:forward page="ergebnis.jsp"></jsp:forward>`
`response.sendRedirect("ergebnis.jsp");`

Agenda

- 1 Setup
- 2 Spring Boot Übungen
- 3 Servlet Übungen**
- 4 „Chat Forum“
- 5 HTML, CSS
- 6 Anderes

SERV00 – Hello World Servlet

Erstellen eines eigenen Servlets in der Eclipse Entwicklungsumgebung.

- File > New Dynamic Web Project ...
- Servlet Erstellen
 - Suchen Sie den „Java Resources“ Ordner des neu erstellten Projektes
 - Wählen Sie über das Context-Menü New > Servlet
 - Package „org.dhbw.jee“ Class „HelloWorld“
 - Implementieren sie die Methode „doGet“ welche eine Web-Seite ausgibt.
Greetings.java
- Erstellen sie eine Seite „index.html“
 - Implementieren sie eine HTML-Seite welche einen Link enthält, der das Servlet aufruft.

```
protected void doGet(...  
    response.setContentType("text/html");  
    PrintWriter out = response.getWriter();  
    out.println("<html>");  
    ...
```

SERV01 - Servlet „Greetings“

- Neues Projekt mit „Greetings.java“, „index.html“
- „Index.html“ enthält ein Eingabeformular.
- Erzeugen Sie mit dem Wizard ein Servlet „org.dhbw.jee.Greetings“
- Greifen Sie mittels dem request-Objekt auf die von der index.html übergebenen Parameter zu und geben die Werte im html aus.

Greetings.java

```
protected void doPost(...  
response.setContentType("text/html");  
PrintWriter out = response.getWriter();  
out.println("<html>");  
...  
out.println(request.getParameter("....  
....
```


SERV02 - Erweiterung Servlet „Greetings“

- Erweitern Sie die Index.html von 3.) um
 - Select/ Multiselect
 - Checkbox
 - Radiobox
 - ... → siehe selfhtml.org
- Geben Sie in einer Schleife alle Request-Parameter aus
 - `request.getParameterNames()` liefert alle Parameter
- Was stellen Sie fest über? Welche Informationen fehlen?

Greetings.java

```
...  
Enumeration<String> e =  
request.getParameterNames();  
....
```

SERV03 - Lifecycle methoden/ logging mit „Greetings“ (1/2)

- Gleiches Projekt wie bei SERV02
- Fügen Sie die Methode „init“ ein, welche die Standardimplementierung überschreibt
- Geben Sie via „System.out“ eine Nachricht aus und testen die Implementierung
 - Wann/ wie oft wird die Methode Init aufgerufen?
- Fügen Sie eine Ausgabe mittels der Methode „log“ hinzu.
 - Was stellen Sie fest?
 - Analysieren Sie das Problem mit dem Debugger
- Korrigieren Sie das Problem

Greetings.java

```
public void init(ServletConfig config)
throws ServletException {
    System.out.println("Greetings wird
initialisiert");
    log("Greetings wird initialisiert");
}
```

SERV03 - Lifecycle methoden/ logging mit „Greetings“ (2/2)

- Vergleichen Sie die Ausgaben von „System.out“ und „log“
 - Diskussion wofür benötigt man logging?
- Fügen Sie die Methode „Destroy“ hinzu
 - Testen Sie, wann diese Methode aufgerufen wird
- In welchen Fällen benötigt man die „Init“/ „Destroy“ methode?

Greetings.java

```
public void destroy() {  
    log("Und Tschüß!");  
}
```

SERV05 - doGet/ doPost mit „Greetings“

- Implementieren Sie die doGet Methode analog der doPost Methode
- Fügen Sie ein „log“-statement ein um nachzuvollziehen welche Methode aufgerufen wird
 - Nutzen Sie den TCP/IP Monitor um die Kommunikation zu analysieren
 - In welchen Fällen wird das Servlet nicht erneut aufgerufen?
- Wdh.: Warum wird zwischen Get und Post unterschieden?

SERV06 - Header Informationen

- Neues Projekt mit separaten Servlet
- Implementieren Sie die doGet Methode welche alle Header Informationen ausgibt
 - Vergleichen Sie die Ausgabe bei der Benutzung verschiedener Browser
 - Untersuchen Sie den Datenverkehr mittels des TCP/IP Browsers
- FYI - google: „Understanding User-Agent Strings“

HeaderInformationen.java

```
...  
Enumeration<String> e =  
request.getHeaderNames();  
...
```

SERV07 - Ein Servlet, das fehlende oder falsche Parameter toleriert

- Erstellen Sie eine Webseite, die numerische Daten (= Zahlen), z.B. Personendaten, einliest und dann das zugehörige Servlet aufruft. Die Besonderheit ist hierbei, dass fehlende oder falsche Parametereingaben durch (vorgegebene) Standardwerte ergänzt werden.
 - **Tipp:** Die Zahlenwerte der Parameter werden im String-Format eingelesen und müssen dann wahlweise in Integer- oder Double-Werte konvertiert werden.
1. Erstellen Sie ein „Web Application“ – Projekt, das – wie üblich – aus einer index.jsp Datei (für die Auslösung des Servlets per HTML-Code) und einer Servlet-Datei besteht, analog zur Erstellung der vorherigen Projekte bzw. Aufgaben.
 - Eingabewerte: z.B. Alter, Personalnummer, Gewicht
 2. Lesen Sie die numerischen Werte durch eine Benutzerabfrage ein. Verwenden Sie eine allgemeine Methode, die die Parameterwerte konvertiert und ggf. für fehlende (oder falsche) Parameter einen Standardwert vorschlägt.

SERV08 Servlet Init Parameter

- Erstellen Sie ein Servlet mit Hilfe des Wizards. Dabei:
 - Füge Sie einen Init-Parameter „shop_name“ hinzu und
 - lassen Sie die „init“-Methode überschreiben.
- Initialisieren Sie in der „init“-Methode eine Instanzvariable „headline“ des Servlets (z.B. „Willkommen im <shop_name>!“)
- Geben Sie in der Get-methode eine einfache Web-Seite mit der „headline“ aus.

SERV08 Servlet Init Parameter

- Erzeugen eines Deployment Descriptors
 - Im Project-Explorer – Context Menu
 - Recherchieren Sie die Syntax des web.xml Files und übertragen Sie die Servlet-Konfiguration aus der Annotation in den Descriptor
 - Testen Sie die Lauffähigkeit der Anwendung
- Starten im Standalone-Tomcat
 - Exportieren Sie die Anwendung als war-File
 - Start von Tomcat ohne eclipse (\bin\startup.bat)
 - Richten Sie den Zugang zur „Manager App“ ein
 - tomcat-users.xml > rolle „manager-gui“ und user hinzufügen
 - Deployment des war-Files über die Admin-GUI

SERV09 REST Servlet & Client

- Ziel: Ein REST-Service der eine HTML/ JavaScript Client mit Daten versorgt.
- Neues dynamisches Webprojekt
- Erstellen sie die folgenden Dateien (siehe nächste Seite)
 - hello.js
 - Index.html
- Erstellen sie ein Servlet welches folgendes JSON zurück gibt:
 - {"id":1,"content":"Hello, World!"} (in ersten Schritt statisch)
 - Content-Type ist „application/json“
- Erweiterung:
 - Die ID wird über alle Anwender hochgezählt.

```
...  
this.getContext().setAttribute("zaehler",zaehler);  
...  
int zaehler = (int)this.getContext().getAttribute("zaehler");
```

- Sequenzdiagramm zeichnen!

SERV09 - ... Web client

```
angular.module('demo', [])  
.controller('Hello', function($scope, $http) {  
    $http.get('http://localhost:8080/testtest/Greeting').  
        then(function(response) {  
            $scope.greeting = response.data;  
        });  
});
```

hello.js

index.html

```
<!doctype html>  
<html ng-app="demo">  
<head>  
<title>Hello AngularJS</title>  
<script  
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.3/angular.min.js"></script>  
    <script src="hello.js"></script>  
</head>  
  
<body>  
<div ng-controller="Hello">  
<p>The ID is {{greeting.id}}</p>  
<p>The content is {{greeting.content}}</p>  
</div>  
</body>  
</html>
```

Agenda

1 Setup

2 Spring Boot Übungen

3 Servlet Übungen

4 „Chat Forum“

5 HTML, CSS

6 Anderes

CH01 - HTML-Entwurf

- Entwerfen Sie eine index.html mit folgendem Inhalt:
 - Liste von Forumseinträgen (jeweils: Name des Erstellers, Datum, Nachricht)
 - Angeordnet z.B. durch HTML-Tabellen
 - Eingabeformular (Name, Nachricht, Senden-Button)

CH02 - Erweiterung um Servlet und JavaBean

- Erstellen Sie eine JavaBean die jeweils einen Forumseintrag repräsentiert
 - Package: org.dhbw.jee Class name: ChatEntry
 - Zeitpunkt des Chat-Eintrags wird im Constructor der Bean initialisiert
- Erstellen Sie ein Servlet, welches die Eingabewerte empfängt und dannach auf die html-Seite weiterleitet
 - Package: org.dhbw.jee Class name: AddEntry
 - Parameter werden im „log“ ausgegeben und ChatEntry Instanz wird erzeugt

Servlet Weiterleitung zur index.html

```
...
RequestDispatcher rd =
request.getRequestDispatcher("/index.html");
rd.forward(request, response);
....
```

CH03 - Speicherung Liste der Chat-Einträge

- Wo werden die Chat-Einträge am besten abgelegt? (Sie sollte nicht nur für einen Benutzer sichtbar sein ...)
 - Setzen Sie die notwendige Änderung im Servlet um.
- Die statische Ausgabe soll nun die tatsächliche Liste der Chat-Einträge enthalten.
 - Machen Sie aus der html- eine jsp-Datei.
 - Implementieren Sie die Ausgabe.

CH04 - Anwendung „merkt“ sich Namen

- Wenn ein Benutzer einen Eintrag geschrieben hat, wird in der neu geladenen Seite sein Name im Eingabefeld vorbefüllt.
 - In welchem Context muss diese Information abgelegt werden?
 - An welcher Stelle wird diese Information gespeichert/ wieder ausgelesen?

CH05 - Redirect vs. forward

- Untersuchen Sie das Verhalten der Implementierung bei der Benutzung des Reload Buttons.
- Ändern Sie den Forward in ein Redirect um ...
- Wie ändert sich das Verhalten?
 - Erläutern Sie mittels eines Sequenzidiagramms

```
...  
response.sendRedirect("index.jsp");  
...
```


CH06 - Überprüfung der Eingabe

- Die Eingabe wird auf unzulässige Schlüsselworte untersucht. Sollte eine „unanständige“ Eingabe erfolgen, wird diese nicht angezeigt. Stattdessen wird der Benutzer über eine Meldung darauf hingewiesen.
- An welcher Codestelle wird die Eingabe überprüft?
- Wie wird die Meldung an den Benutzer am besten zwischen den Programmteilen übergeben?

CH07 – Löschen einzelner Einträge

- In der Tabelle neben den Chat-Einträgen wird jeweils „Löschen“ angezeigt
- Implementieren Sie ein separates Servlet „DelEntry“ welches den ausgewählten Chat-Eintrag entfernt.

Agenda

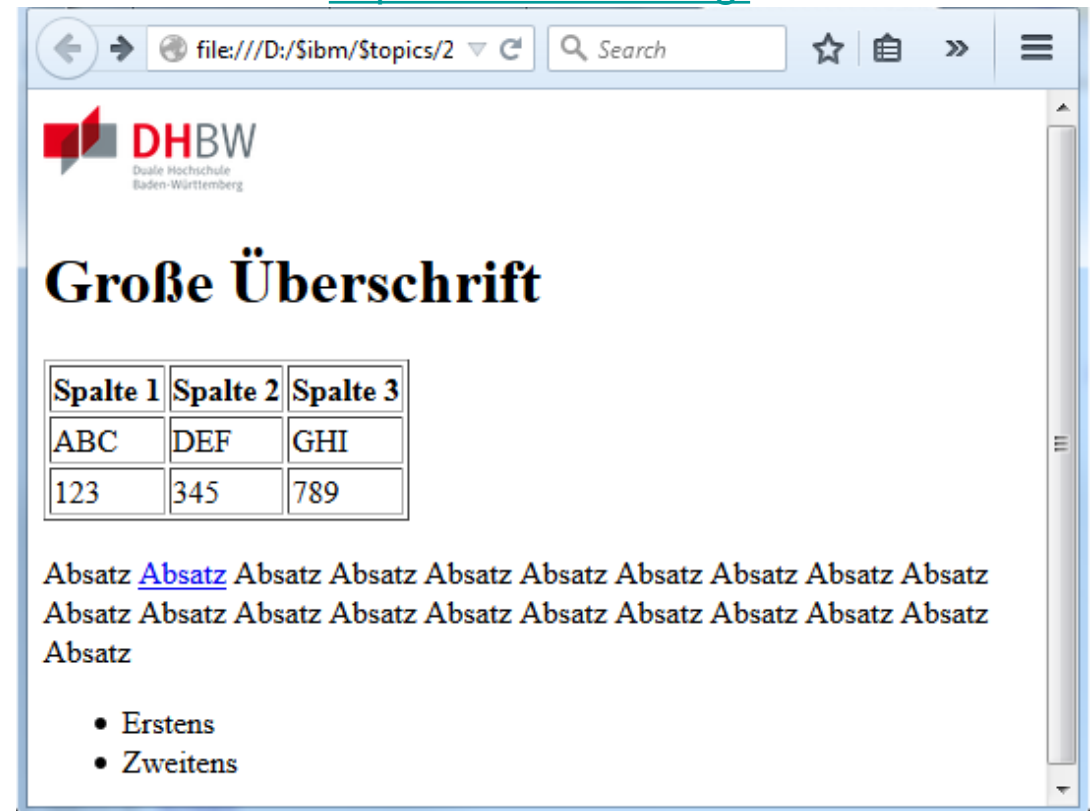
- 1 Setup
- 2 Spring Boot Übungen
- 3 Servlet Übungen
- 4 „Chat Forum“
- 5 HTML, CSS**
- 6 Anderes

•Quellen

- <https://code.visualstudio.com/download>
- <http://www.html-seminar.de>
- <http://www.w3schools.com/html>
- <http://wiki.selfhtml.org/>

HTML01 HTML ausprobieren

- Download und Installation von VS Code
- Textdatei erstellen (ausprobieren.html)
- HTML-Grundgerüst einfügen
- Html Elemente ausprobieren
 - Überschrift
 - Tabelle
 - Absatz
 - Aufzählung
- Untersuchen Sie das Verhalten wenn sich die Größe des Browserfensters ändert
- Platzieren Sie einen Link auf eine Ergebnisseite einer Google-Suche. In welchem Teil der URL steht der Suchbegriff?



HTML01 HTML ausprobieren - Quelltext

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Sample HTML5 Grundger&uuml;st</title>
    <meta name="description" content="Beschreibung" />
</head>
<body>

<h1> Gro&szlig;e &Uuml;berschrift </h1>

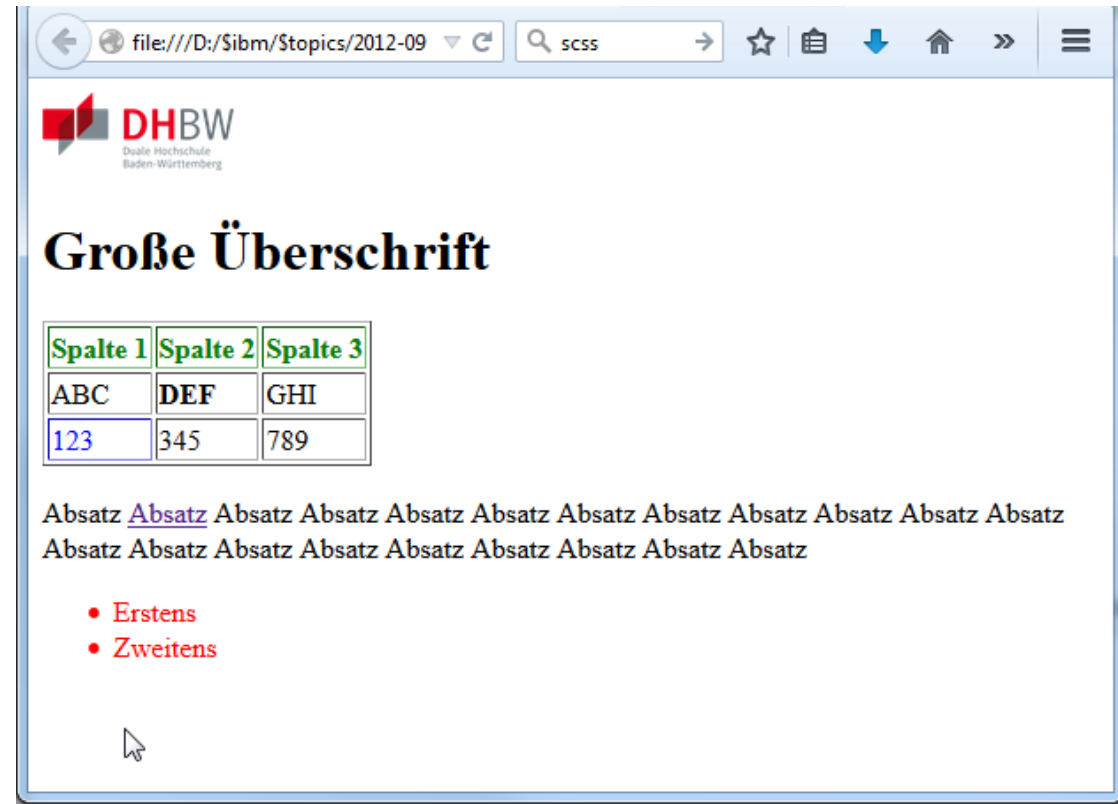
<table border="1">
<tr>
<th>Spalte 1</th> <th>Spalte 2</th> <th>Spalte 3</th></tr>
<tr>
<td>ABC</td><td>DEF</td><td>GHI</td></tr>
<tr>
<td>123</td><td>345</td><td>789</td></tr>
</table>

<p>Absatz <a href="http://google.de">Absatz</a> Absatz Absatz Absatz Absatz Absatz Absatz Absatz Absatz </p>
<ul>
<li>Erstens</li>
<li>Zweitens</li>
</ul>
</body>
</html>
```

- Zeilenumbrüche im Quelltext werden ignoriert
- Verlinkungen können sich auf andere Web Seiten oder andere Dateien auf dem gleichen Server beziehen
- Umlaute müssen „maskiert“ werden

HTML02 CSS Ausprobieren

- Beispiel von HTML01 kopieren und weiterentwickeln.
- Formatierung entsprechend des Beispiels → dabei inline & internal Definitionen verwenden und sinnvolle Selektoren-Typen



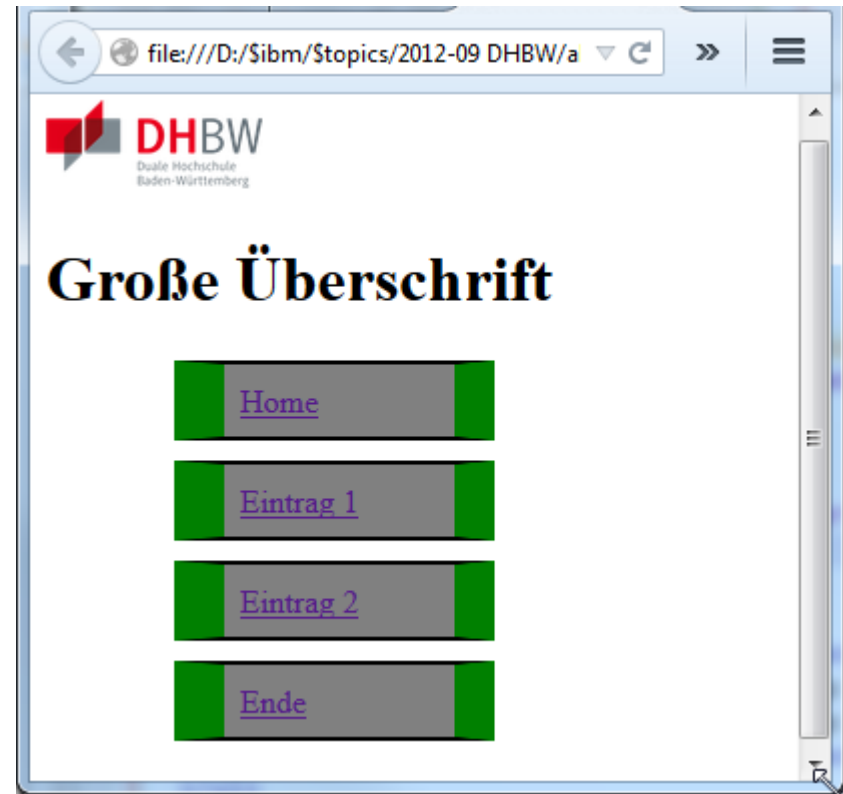
HTML02 CSS Ausprobieren - Quelltext

```
<head>
    ...
    <style type="text/css">
        li {color:red}
        .headline {color:green}
        #blue {color:blue}
    </style>
</head>

<body>
    ...
    <th class="headline">Spalte 1</th>
    <th class="headline">Spalte 2</th>
    <th class="headline">Spalte 3</th>
    ...
    <td style="font-weight: bold;" >DEF</td>
    ...
    <td id="blue">123</td>
    ...
    <ul>
    <li>Erstens</li>
    <li>Zweitens</li>
    </ul>
```

HTML03 CSS Layout

- Html Seite erstellen
- Navigation mittels `` & `` erstellen
- Externe CSS Datei erstellen und in das HTML einbinden
- Navigationseinträge mittels CSS formatieren (Rahmen Farbe/ Stärke (border), Außenabstand (margin), Innenabstand (padding))



HTML03 CSS Layout - Quelltext

```
<html>
<head>
...
<link href="externes_CSS.css" type="text/css" rel="stylesheet"/>
</head>

<body>


<div id="navigation">
  <ul>
    <li><a href="ausprobieren.html">Home</a></li>
    <li><a href="ausprobieren.html">Eintrag 1</a></li>
    <li><a href="ausprobieren.html">Eintrag 2</a></li>
    <li><a href="ausprobieren.html">Ende</a></li>
  </ul>

</div>
</body>

</html>
```

```
#navigation {
  width: 200px;
  text-align: left;
  margin-top: 22px;
  margin-bottom: 23px;
  margin-left: 24px;
  margin-right: 25px;
}

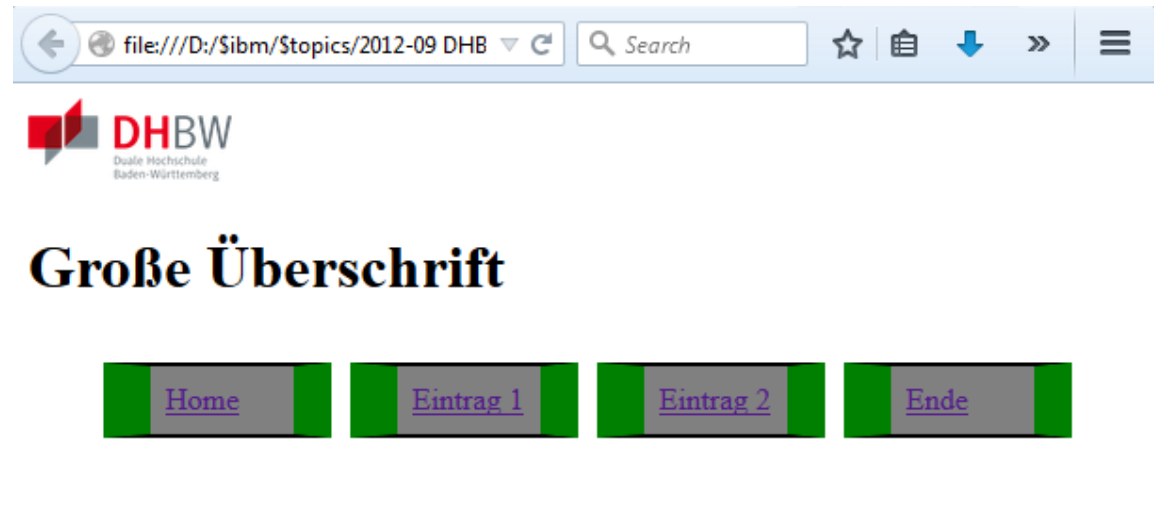
#navigation ul {
  list-style: none;
}

#navigation li {
  background-color: gray;
  border-top: 2px black solid;
  border-left: 25px green solid;
  border-bottom: 2px black solid;
  border-right: 20px green solid;
  margin-top: 10px;
  padding-top: 8px;
  padding-bottom: 8px;
  padding-left: 8px;
  padding-right: 8px;
}
```

HTML04 CSS Layout - erweitert

- Browser Tools kennenlernen (F12)
 - Inspector (Box-Model)
 - Console
 - Style Editor
- Navigation horizontal ausrichten (float: left; für li)
- Für Geräte mit kleiner Breite anpassen:
 - Navigation vertikal
 - Einträge breiter

```
@media(...) {  
  // selektor ...  
}
```



HTML04 CSS Layout erweitert - Quelltext

```
#navigation {  
→  
    text-align: left;  
    margin-top: 22px;  
    margin-bottom: 23px;  
    margin-left: 24px;  
    margin-right: 25px;  
}  
#navigation ul {  
    list-style: none;  
}  
#navigation li {  
→ float: left;  
→ width: 60px;  
→ margin-right: 10px;  
  
    background-color: gray;  
    border-top: 2px black solid;  
    border-left: 25px green solid;  
    border-bottom: 2px black solid;  
    border-right: 20px green solid;  
    margin-top: 10px;  
    padding-top: 8px;  
    padding-bottom: 8px;  
    padding-left: 8px;  
    padding-right: 8px;  
}
```

HTML05 Web Seite in Eclipse

- Erstellen eines Java Web/Web Application-Projekts
 - Erstellen einer Datei `index.html` und `hello.html`
 - Einfügen eines Links von `index` → `hello`
- “hello.html” soll Daten an den Server übermitteln.
 - Formular
 - Text-Eingabefeld
 - Mehrzeilige Texteingabe
 - Submit-Button
- Ausliefern des Projekts
- Start der ausgelieferten Seite im Browser



HTML05 Erweiterung des Eingabeformular

- Selbstständig um weitere Eingabeelemente erweitern (select box, radio-button, check-box, ...)

Index.html

```
...
<body>

<h1> Gro&szlig;e &Uuml;berschrift </h1>
<p>Standard Text</p>
<a href="hello.html">Link</a> auf Hello
</body>
...
```

Hello.html

```
...
<form action="meinserver.de/mache_etwas" id="" method="GET">
  Name: <input type="text" name="firstname"/><br/>
  Comment: <textarea type="text" name="comment"></textarea><br/>
  <input type="submit" name="submit" value="send!"/>
</form>
...
```

HTML06 ToDo Liste mit DHTML

- Erstellen einer HTML-Datei
 - Ein Eingabefeld und ein Button zum Hinzufügen eines ToDo Items
 - Hinweis: kein `<form>` Tag benutzen
 - Eine (leere) Liste mit Items
- Dynamisches Ändern der Liste
 - Beim Click auf den Button (-> Event) Hinzufügen des Inhalts des Eingabefelds zur Liste
 - JQuery oder natives JavaScript möglich
- Get Creative!
 - Löschen eines Items beim Click darauf
 - Checkbox „wichtig“ neben dem Input Feld
 - macht das Item GROSS und **rot**.
 - ...

To Do List

- Einkaufen

AJAX01 – STAR WARS! Trivia

- Wir bauen eine Trivia Seite zu Star Wars Charakteren
- Die Daten stammen von <https://swapi.dev/>
- Beim Click auf den Button wird zufällig einer von 82 Charakteren abgerufen und einige Attribute der Antwort angezeigt (z.B. Name, Augenfarbe ...)
 - Zufallszahl zw. 0 und 1: `Math.random()`
- Außerdem wollen wir auch den Namen der Heimatwelt des Charakters anzeigen
- Wie würde das Sequenzdiagramm aussehen?

STAR WARS!

Random Character!

Name:

Augenfarbe:

Heimatwelt:

```
fetch('...URL...')
  .then(response => response.json())
  .then(data => {
    // Schreibe Daten ins DOM
  });
```

Agenda

- 1 Setup
- 2 Spring Boot Übungen
- 3 Servlet Übungen
- 4 „Chat Forum“
- 5 HTML, CSS
- 6 **Anderes**

Übung zu AJAX

- Erstellen Sie:
 - Ein neues Projekt mit einer index.html die ein Servlet aufruft
 - Die Seite enthält
 - Einen Auswahlliste (Inhalt egal)
 - Und einen Button, der das Servlet aufruft
 - Das Servlet gibt Datum & Uhrzeit aus.

Übung zu AJAX – index.html um JavaScript erweitern

```
<script type="text/javascript">
function ajaxRequest(reqURL)
{
//Creating a new XMLHttpRequest object
var xmlhttp;
if (window.XMLHttpRequest){
xmlhttp = new XMLHttpRequest(); //for IE7+, Firefox, Chrome, Opera, Safari
} else {
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP"); //for IE6, IE5
}
xmlhttp.open("GET", reqURL, true);
xmlhttp.send(null);

if (xmlhttp.readyState == 4) {
if (xmlhttp.status == 200)
{
document.getElementById("message").innerHTML = xmlhttp.responseText;
}
else
{ alert('Something is wrong !!');}
}
}
</script>
```

Übung zu AJAX

- Der Button:
 - `<input type="button" value="Show Server Time" onClick='ajaxRequest("get-current-time")' />`
- Die Position der Nachrichtenausgabe:
 - `*** Message from server ***`

AN01 – Analyse Eingabeparameter

- zip-Datei aus moodle herunterladen
 - Index.html
 - MyBean.java
 - Servlet1.java
 - Ausgabe.jsp
- Ermitteln Sie die zu erwartenden Ausgaben durch Analyse des Codes!
 - Was wird angezeigt, wenn bei Variante 1-3 im jeweils die folgende Eingabe getätigt wird:
 - Feld A: „aa“
 - Feld B: „bb“

AN01 – Analyse Eingabeparameter

Index.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Analyse</title>
</head>
<body>
<h1>Variante 1:</h1>
  <form action="Servlet1" method="get">
    <p>Feld A: <input type="text" name="param1" /></p>
    <p>Feld B: <input type="text" name="param2" /></p>
    <input type="submit" value="Los!" />
  </form>
<h1>Variante 2:</h1>
  <form action="Analyse" method="post">
    <p>Feld A: <input type="text" name="param1" /></p>
    <p>Feld B: <input type="text" name="param2" /></p>
    <input type="submit" value="Los!" />
  </form>
<h1>Variante 3:</h1>
  <form action="submit" method="get">
    <p>Feld A: <input type="text" name="param1" /></p>
    <p>Feld B: <input type="text" name="param2" /></p>
    <input type="submit" value="Los!" />
  </form>
</body>
</html>
```

Ausgabe.jsp

```
...
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Analyse</title>
</head>
<body>

<jsp:useBean id="ausgabe" scope="request" class="org.dhbw.jee.MyBean"/>
<% ausgabe.processMessage(); %>
<%=ausgabe.getMessage() %><br/>

<% System.out.println("Alles richtig nachvollzogen?"); %>

</body>
</html>
```

AN01 – Analyse Eingabeparameter

MyBean.java

```
package org.dhbw.jee;

public class MyBean {
    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String processMessage() {
        return message = "Ergebnis: " + message;
    }
}
```

Servlet1.java

```
...
@WebServlet({"/Servlet1", "/Analyse"})
public class Servlet1 extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        MyBean ausgabeBean1 = new MyBean();
        ausgabeBean1.setMessage(request.getParameter("param1") + request.getParameter("param2"));
        request.setAttribute("ausgabe", ausgabeBean1);

        PrintWriter out = response.getWriter();
        out.println("<h1>Ergebnis:</h1>");

        RequestDispatcher rd = getServletContext().getRequestDispatcher("/ausgabe.jsp");
        rd.forward(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        MyBean ausgabeBean1 = new MyBean();
        ausgabeBean1.setMessage(request.getParameter("param2") + request.getParameter("param2"));
        request.setAttribute("ausgabe", ausgabeBean1);
        ausgabeBean1.processMessage();
        request.getSession().setAttribute("ausgabe", ausgabeBean1);

        response.sendRedirect("ausgabe.jsp");
    }
}
```