

# Verteilte Systeme – Web-Programmierung Übungen

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

Version: 12

## Agenda

**1** Setup

**2** Spring Boot Übungen

**3** HTML, CSS

**4**

**5**

**6**

## S01 Einrichten der Entwicklungsumgebung

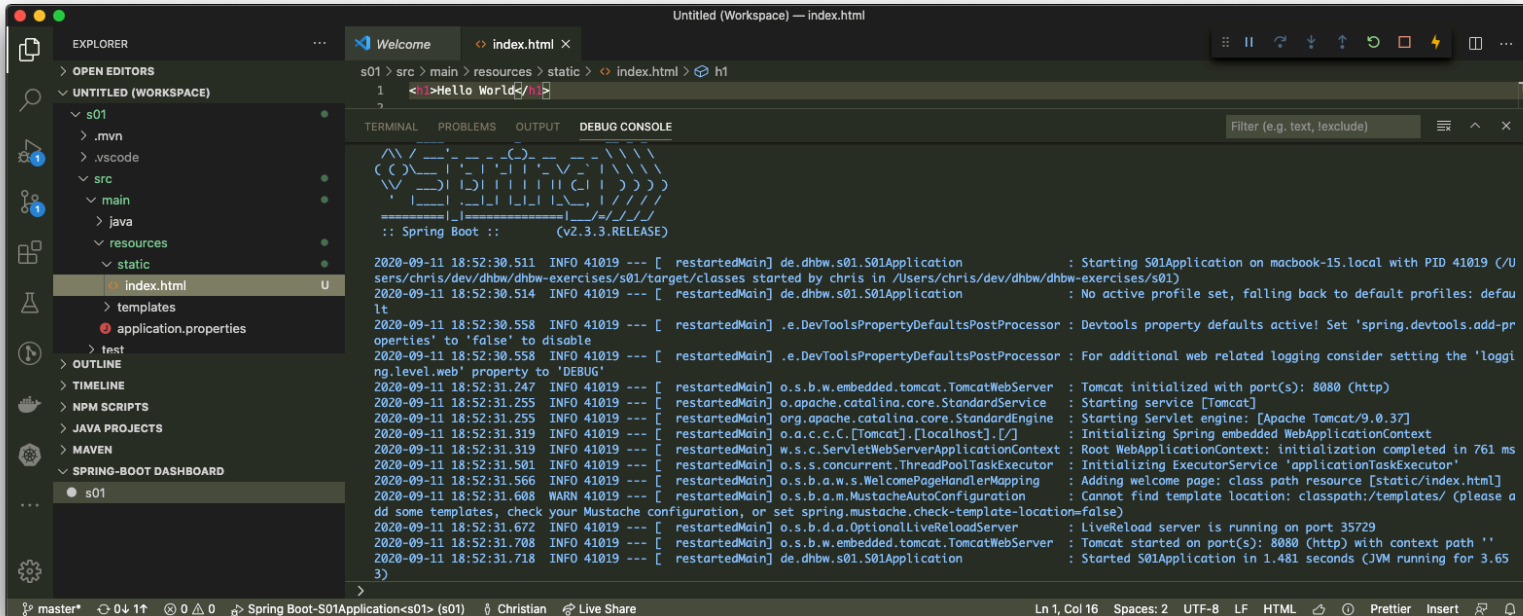
- In VS Code
  - File (Mac: Code) -> Preferences -> Extensions
  - Suchen und Installieren von
    - Java Extension Pack
    - Spring Boot Extension Pack
    - FreeMarker
  - Installation abwarten, dann View -> Command Palette -> Java: Configure Runtime
  - Ein JDK in der Version  $\geq 11$  wird benötigt. 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: z.B. 2.3.3 (latest version)
  - 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
properties' to 'false' to disable
2020-09-11 18:52:30.558 INFO 41019 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'loggi
ng.level.web' property to 'DEBUG'
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
dd some templates, check your Mustache configuration, or set spring.mustache.check-template-location=false)
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

## Agenda

1 Setup

2 Spring Boot Übungen

3 HTML, CSS

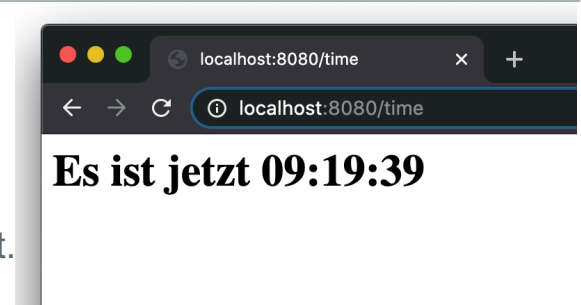
4

5

6

## SB02 – Zeitanzeige

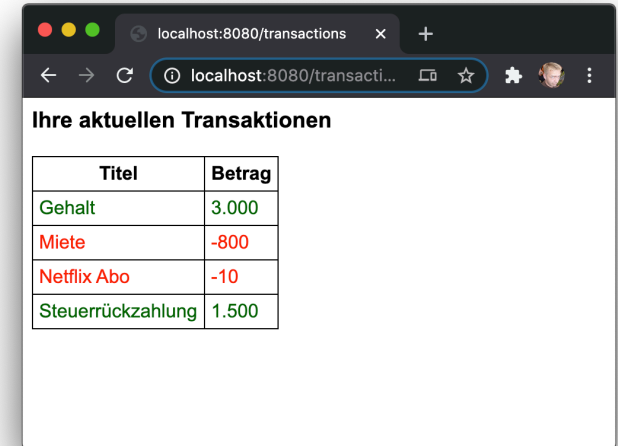
Erstellen Sie eine Anwendung, die bei jedem Zugriff die aktuelle Zeit anzeigt.



- Wiederholen Sie die Schritte aus „**S01 Anlegen einer neuen Applikation**“ mit der Artifact ID „sb02“
  - 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 eine Java-Klasse als Controller an (Ordner: src > main > java > de.dhbw.sb02 | Dateiname: z.B. „TimeController.java“), 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 z.B. „time-view.ftlh“ (Ordner: src > main > resources > templates)
- 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

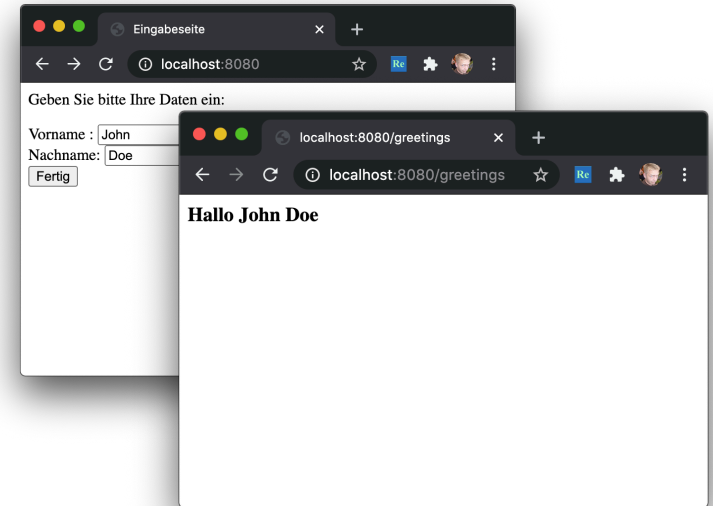
Titel	Betrag
Gehalt	3.000
Miete	-800
Netflix Abo	-10
Steuerrückzahlung	1.500

- Überlegen Sie welche Komponenten Sie benötigen: Controller, Template, .. und wie sie diese benennen.
- Eine Transaktion können wir als einfaches POJO (Java-Klasse) 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?



## SB04 – Formular

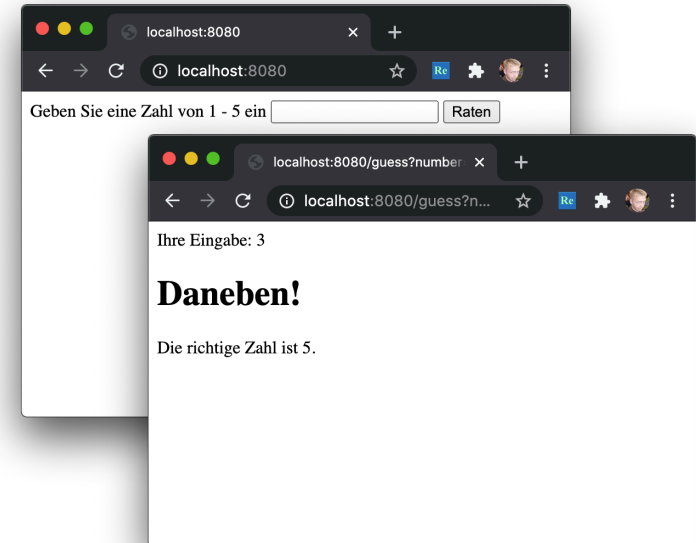
Erstellen Sie eine Anwendung, die in einem Formular Vor- und Nachname des Benutzers erfragt und ihn nach dem Absenden auf einer neuen Seite begrüßt.



- Das Formular kann auf einer statischen HTML-Seite definiert sein, die Begrüßungsseite sollte ein FreeMarker-Template sein.
- Was muss getan werden, um GET statt POST zu verwenden? Was ändert sich dadurch technisch und konzeptionell?
- Zusatzaufgabe: Experimentieren Sie mit weiteren Formularelementen (Auswahllisten, Checkboxes, Radiobuttons ...), s. <https://wiki.selfhtml.org/wiki/HTML/Formulare>

## SB05 – Zahlenraten

Erstellen Sie eine Anwendung, die den Benutzer eine Zahl zwischen 1 und 5 raten lässt. Der Server soll die Zahl mit einer Zufallszahl vergleichen und dem Benutzer sagen, ob er „gewonnen“ hat



- Welche Eingabevalidierungen sollten sinnvollerweise durchgeführt werden?
- Wo sollten diese passieren?

Code-Schnipsel:

```
Int randomNumber = ThreadLocalRandom.current().nextInt(1, 6);
```

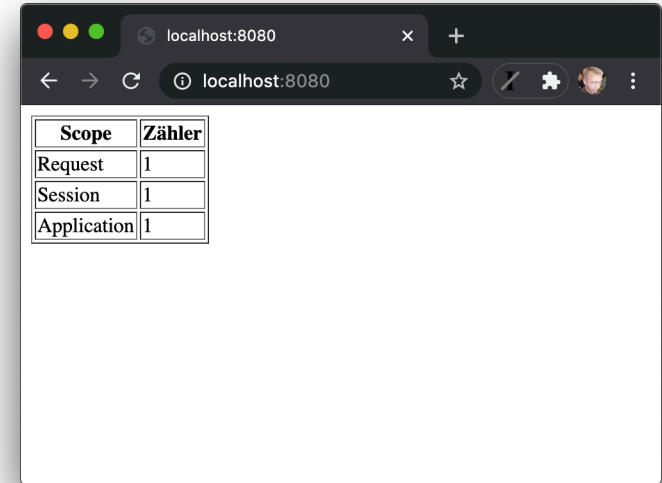
## SB06 – Kontexte

Erstellen Sie eine Anwendung, die bei jedem Aufruf je einen Zähler in den 3 Kontexten

- Request
- Session
- Application

um 1 erhöht. Legen Sie dafür 3 Model-Klassen für Zähler in dem jeweiligen Scope an.

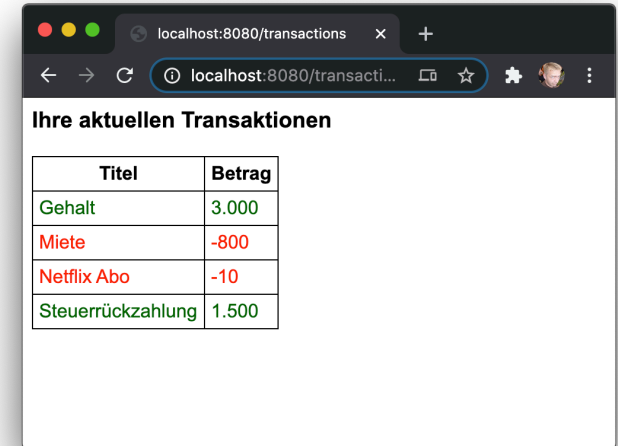
- Experimentieren Sie mit verschiedenen Arten, die Anwendung aufzurufen
  - Mehrmals in einem Browser-Tab
  - In einem anderen Browser-Tab
  - In einem Incognito-Tab
  - In einem anderen Browser
  - Nach dem Neustart der Anwendung



Scope	Zähler
Request	1
Session	1
Application	1

## SB07 – Konto (REST Style)

Schreiben Sie die Anwendung aus SB03 so um, dass sie ohne Server-seitiges Templating, sondern mit einem REST-API-Call funktioniert



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/transactions'. The page content is titled 'Ihre aktuellen Transaktionen' and contains a table with two columns: 'Titel' and 'Betrag'. The table lists four transactions: 'Gehalt' (3.000), 'Miete' (-800), 'Netflix Abo' (-10), and 'Steuerrückzahlung' (1.500). The values are color-coded: green for positive and red for negative.

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

- Viele Teile aus SB03 können wieder verwendet werden (das Model, CSS, Teile des Templates und des Controllers)
- Da das vom Server ausgelieferte HTML keine dynamischen (aus Serversicht) Inhalte mehr enthalten soll, muss es kein Freemarker-Template sein, sondern kann (wie in SB01) als statische HTML-Seite realisiert werden
- Um in der HTML-Seite die Daten anzeigen zu können, ist ein AJAX-Call (s. AJAX01) und DOM-Manipulation (s. HTML06) notwendig

## Agenda

1 Setup

2 Spring Boot Übungen

3 HTML, CSS

4

5

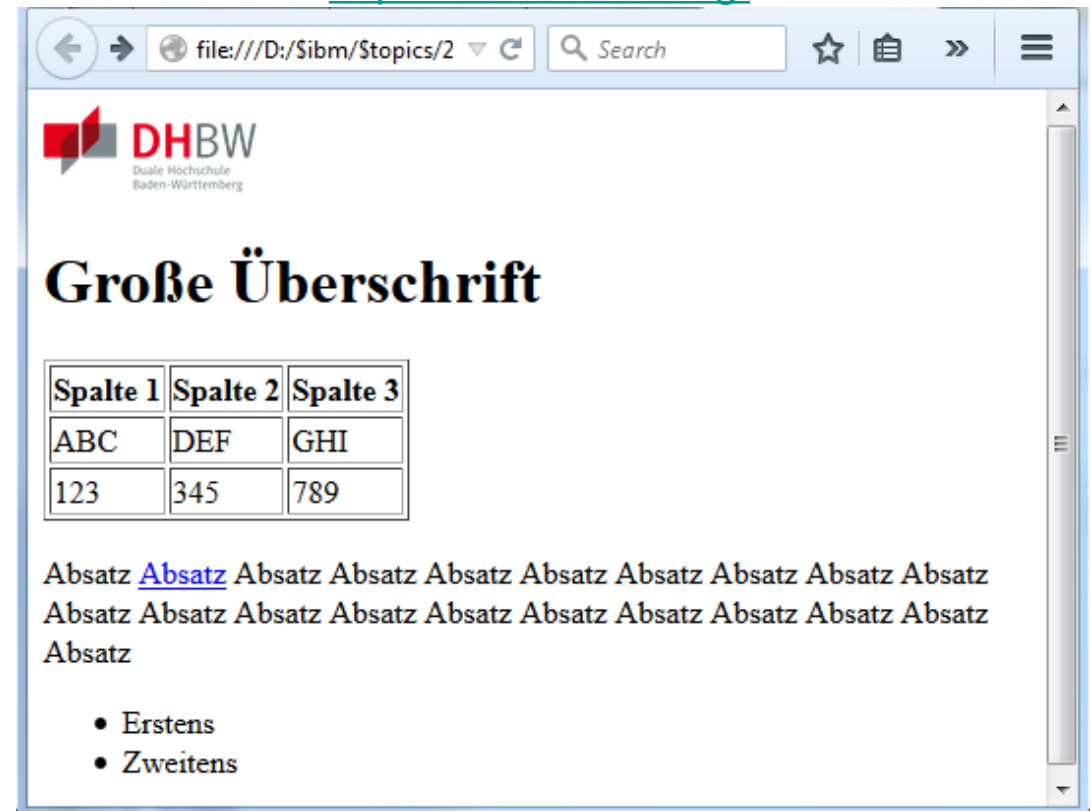
6

## •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





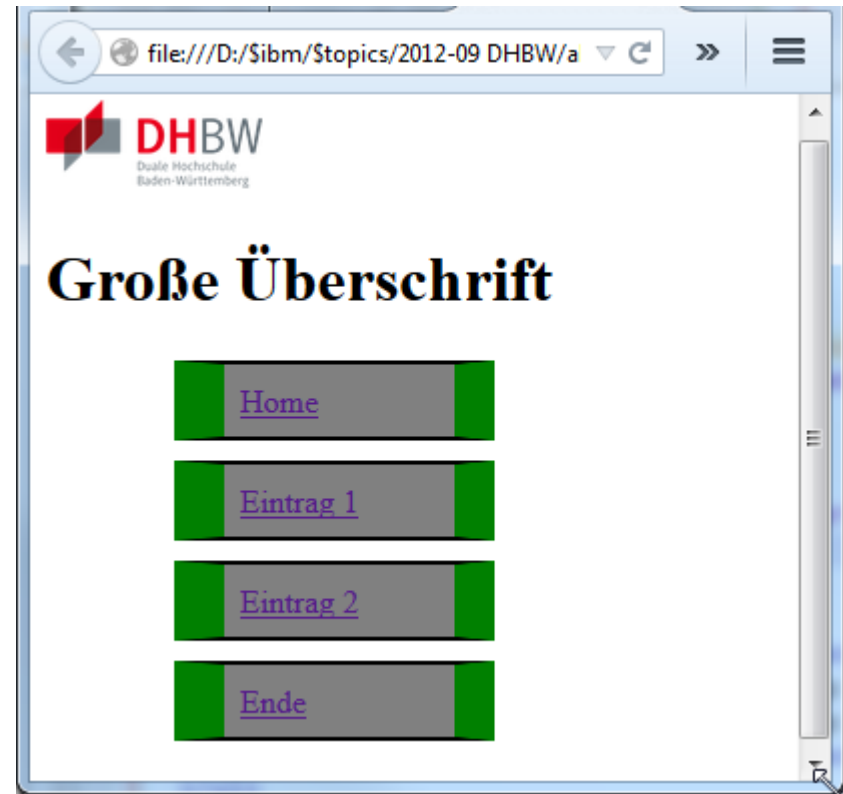
## HTML02 CSS Ausprobieren - Quelltext

```
<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 `<ul>` & `<li>` 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;
}

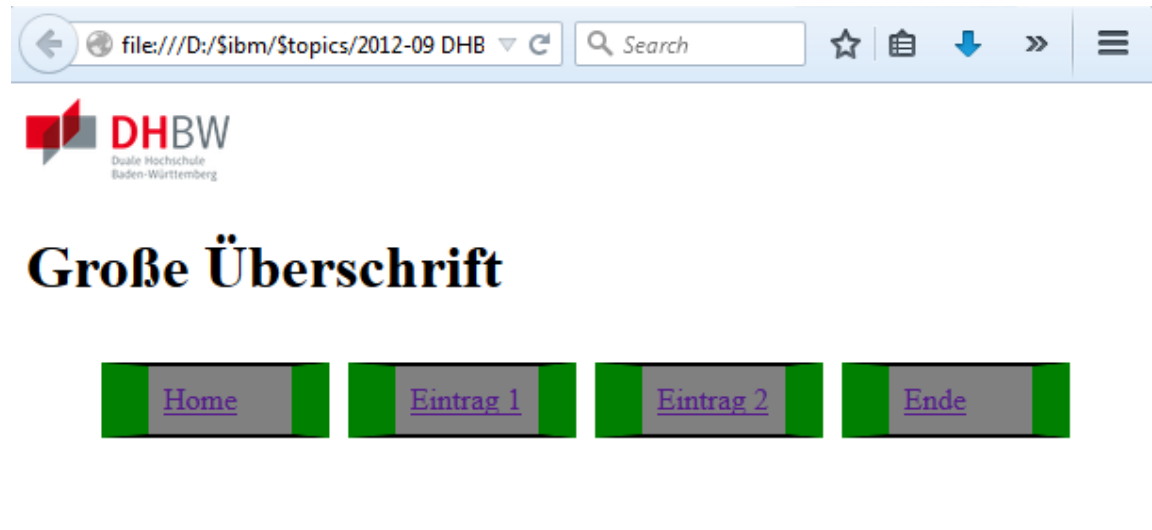
#navigation ul {
  list-style: none;
}

#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
  });
```