

Verteilte Systeme 2 Lab

Konstruktion Web-basierter Systeme

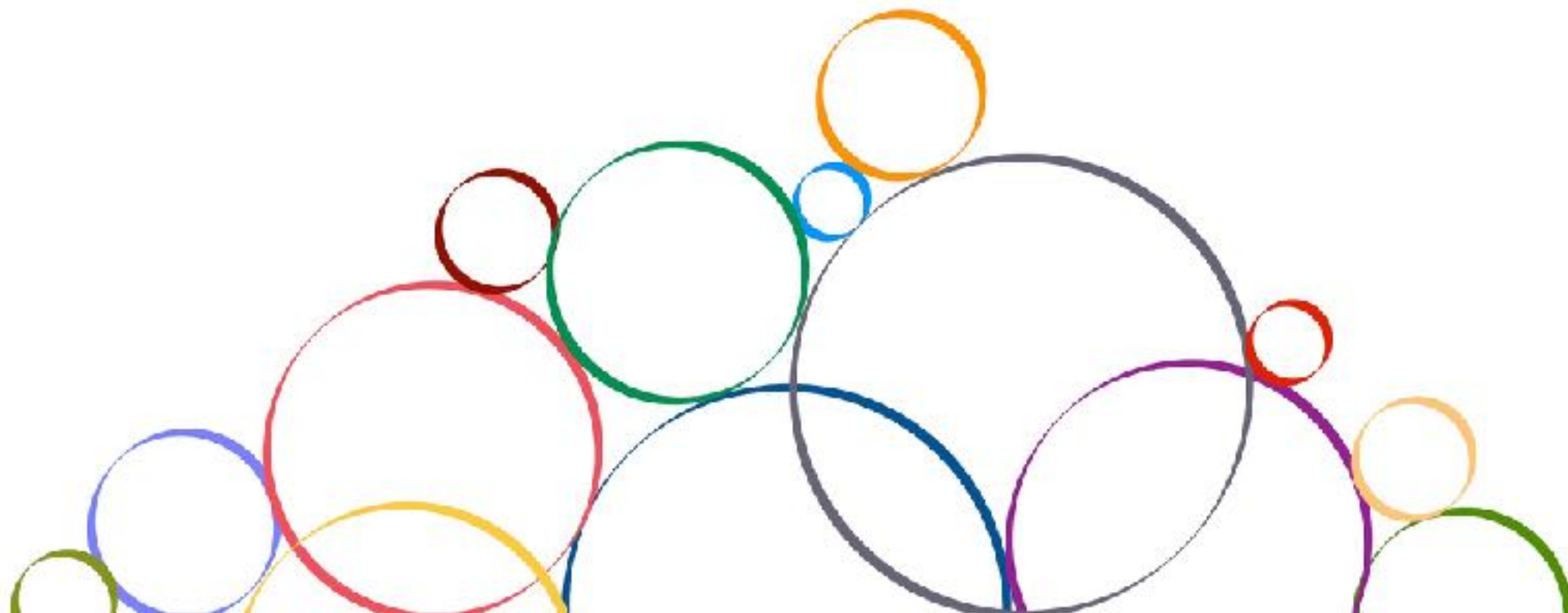
christian.zirpins@hs-karlsruhe.de

Überblick und Einführung



Hochschule Karlsruhe
Technik und Wirtschaft

UNIVERSITY OF APPLIED SCIENCES



Struktur — Überblick und Einführung

Überblick des Labors

- Einordnung
- Aufgabenstellung
- Organisation

Einführung in Spring MVC

- Spring Web MVC
- Spring Boot Beispiel

Einordnung in der Fakultät IWI (Informatik)

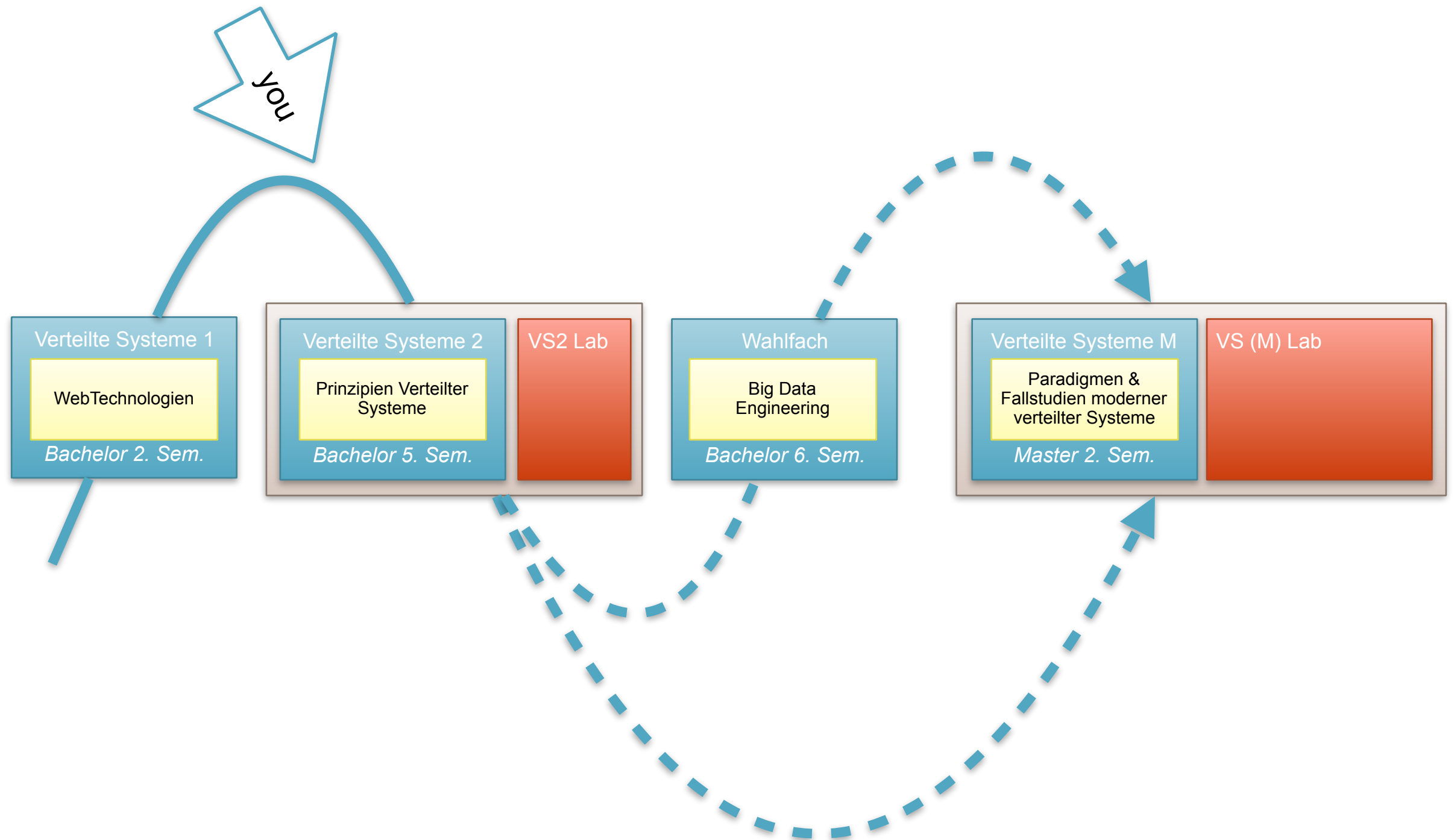
VS2-Lab Team

- Dipl.-Inf. (FH) **Adelheid Knodel**
- Prof. Dr. rer. nat. Dipl.-Inf. **Christian Zirpins**

Bereich Verteilte Systeme (VSYS)

- Schwerpunkte in Forschung und Lehre:
 - *Web Engineering*
 - *Datenintensive Systeme ("Big Data")*
 - *Service Computing ("Microservices")*
- Mit Bezügen zu...
 - *Cloud Computing, Internet of Things, Smart Systems u.a.*

Lehrpfade Verteilte Systeme



Laborveranstaltung

Ziele: Die Studierenden ...

- ... wenden *moderne Frameworks¹* zur *Konstruktion verteilter Systeme²* an.
- ... setzen *allgemeine Prinzipien³* verteilter Systeme praktisch um.

Inhalt

- ² Entwicklung eines Web-basierten Systems
 - Microblogging Social Media Anwendung ('Twitter-Klon')
- ³ Anwendung verschiedener Architekturstile und Kommunikationsmodelle
 - Multitier-Client/Server, geschichteter, objekt- und ereignisbasierter Stil
 - Synchrone/asynchrone Kommunikation, Pub/Sub Messaging
- ¹ Verwendung aktueller Softwaretechniken für Web Anwendungen
 - HTML5/CSS3/JavaScript, MVC, WebSockets, NoSQL
 - Bootstrap, Spring Boot, Redis

Aufgabe

Social Media Anwendung

- Microblogging ('Twitter-Klon')

Funktionale Anforderungen

- Verwaltung von Nutzerkonten
- Nutzer suchen, folgen
- Posts schreiben, Timelines lesen
- Push Benachrichtigungen

Nicht-funktionale Anforderungen

- Sicherheit durch Login/ Sessions
- Benachrichtigungen in Echtzeit
- Replikationstransparenz (Webserver)
- Mobilgerätetauglichkeit

Welcome to the LKIT Microblog

Please login or register

Login with existing account.

Username:

Password:

Submit

Reset

Create new account.

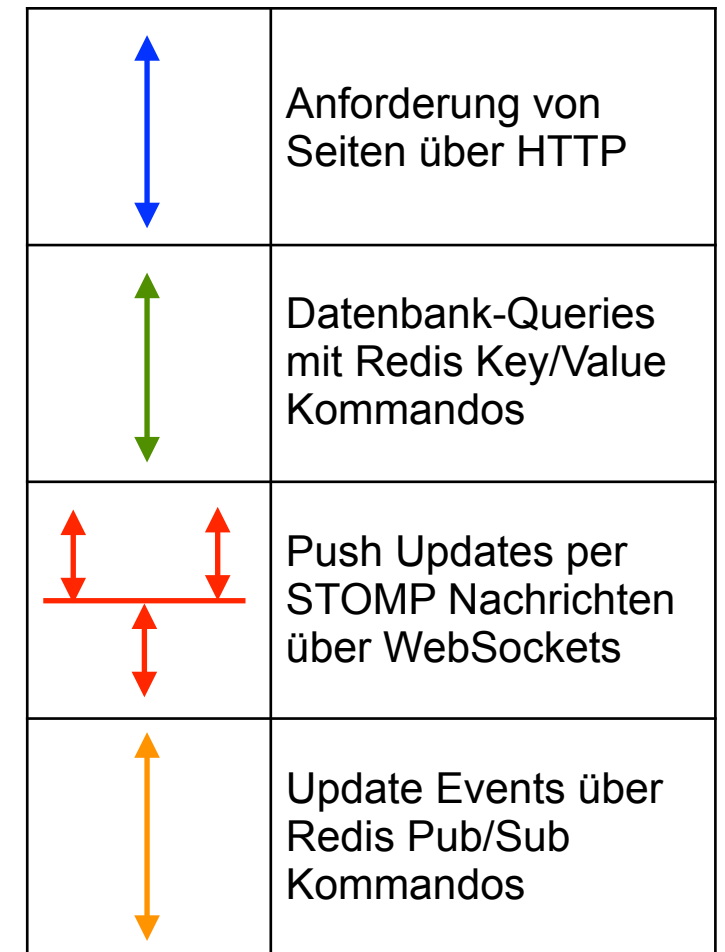
Username:

Password:

Submit

Reset

© 2016 HsKA, LKIT



Teilaufgaben Winter 2017

A1 Konzeption und Gestaltung (12.10 bis 9.11)

- Analyse der Anforderungen (Use Case Diagramm oder textuell)
- Entwurf der logische Seitenstruktur/Navigation (Zustandsdiagramm)
- Erstellen von Mockups (Spring Boot Projekt)
- Entwurf des Datenmodells (Redis Datenstrukturen und Key-Muster)

A2 Seitenbasierte Implementierung (16.11 bis 14.12)

- Erstellung von Repositories und Services
- Realisierung von Authentifizierung und Sessions
- Konstruktion von Controllern und Templates
- Clientseitige Programmierung von Layouts und Interaktionen

A3 Asynchrone Erweiterungen (21.12 bis 25.1)

- Realisierung von Client Push-Messages mit WebSockets
- Implementierung von Server Pub/Sub Messaging mit Redis

Projektplan Winter 2017

					<i>Aufwand (Stunden)</i>	
<i>Termin</i>	<i>Laboraufgabe</i>	<i>Meetup</i>	<i>Seminarteil</i>	<i>Abgabe</i>	<i>Präsenz</i>	<i>Eigenst.</i>
12.10		Intro			1	1
19.10	A1 Konzeption und Gestaltung		Web Frameworks/Spring		2	1
26.10			Persistenz/Redis		2	1
2.11					1	1
9.11		Diskussion A1		Entwürfe	1	
16.11					1	1
23.11	A2 Seitenbasierte Implementierung				1	1
30.11					1	1
7.12					1	1
14.12		Diskussion A2		System V1	1	
21.12			Messaging/WebSockets		1	1
11.1	A3 Asynchrone Erweiterungen				1	1
18.1					1	1
25.1		Demo Day		System V2	1	
					16	11

Ressourcen und Support Winter 2017

Präsenztermine

- Wöchentlich: [Seminarteile \(3x\)](#), [Meetups \(4x\)](#), [freie Arbeit \(7x\)](#)

Foliensätze / Begleitmaterial etc. auf ILIAS

- [Anmeldung](#) (pwd 'ic4ip')

<http://bit.ly/2dv1AY4>

Unterlagen / Material

- [Aufgabenblatt](#) im ILIAS
- [Demo Projekte](#) auf Github laden

```
# git clone https://github.com/zirpins/vs2lab.git
```

Email Support

- adelheid.knodel@hs-karlsruhe.de
- christian.zirpins@hs-karlsruhe.de

Struktur — Überblick und Einführung

Überblick des Labors

- Einordnung
- Aufgabenstellung
- Organisation

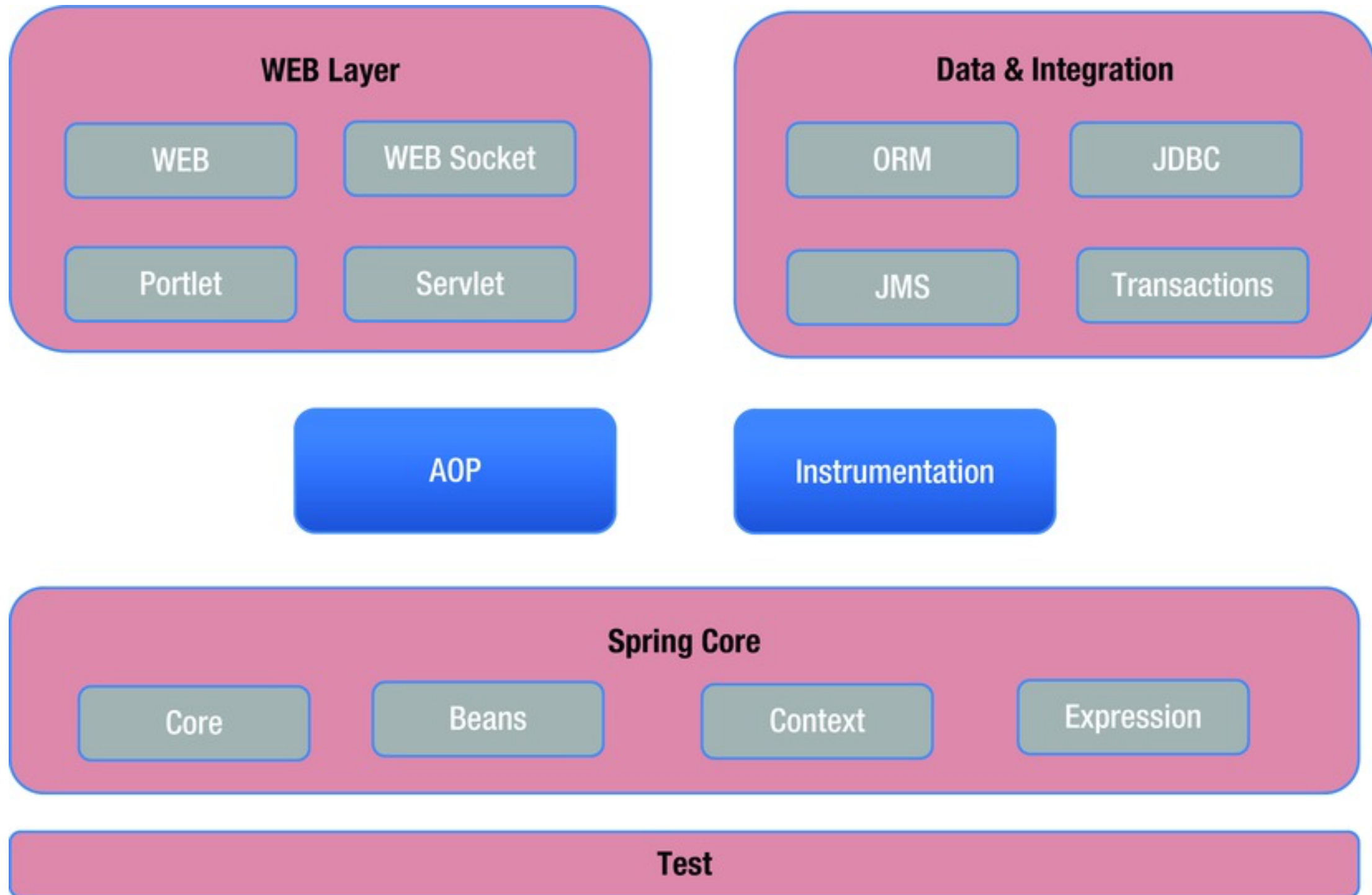
Einführung in Spring MVC

- Spring Web MVC
- Spring Boot Beispiel

Spring in a Nutshell

- Das Spring Framework ist der de facto Standard zur Entwicklung von Java/Java EE–basierten Enterprise Anwendungen.
 - Das Framework wurde erstmals 2002 von Rod Johnson vorgestellt.
- Das Spring Framework ...
 - ...enthält ein **Dependency Injection** (DI) Model zur **Reduktion von schematischem Code** bei der Anwendungsentwicklung,
 - ...unterstützt **Aspect Oriented Programming** (AOP) zur Implementierung von **Crosscutting Concerns (CCC)**, and
 - ...vereinfacht die **Integration mit anderen Frameworks** und Technologien.
- Das Spring Framework beinhaltet **Module**, die spezifische Dienste anbieten, u.a. für **Datenzugriff**, Instrumentation, Messaging, Testen und **Web Integration**.
 - Durch die Modularität von Spring können beliebige Module (und nur diese) je nach individuellen Anforderungen gewählt werden.

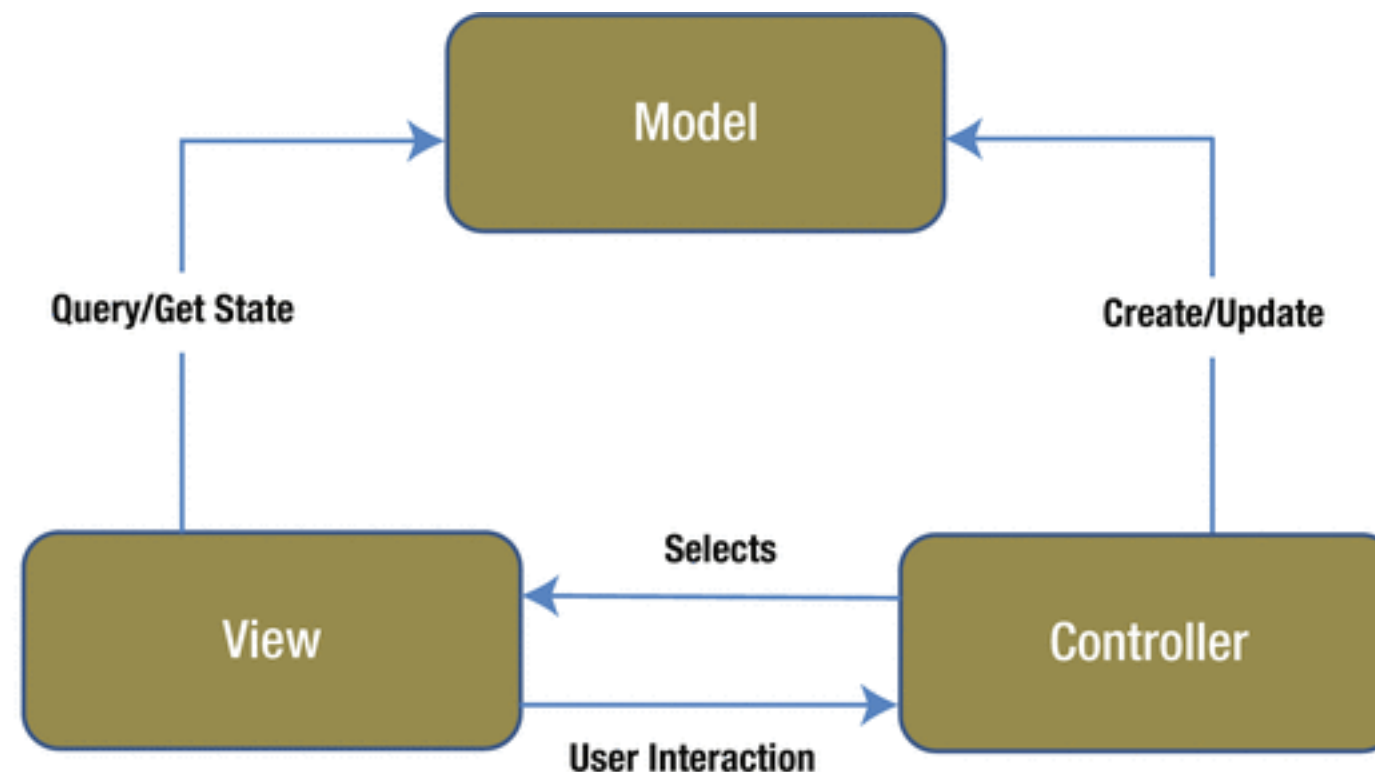
Spring Framework Module



Aus [Varanasi 2015]

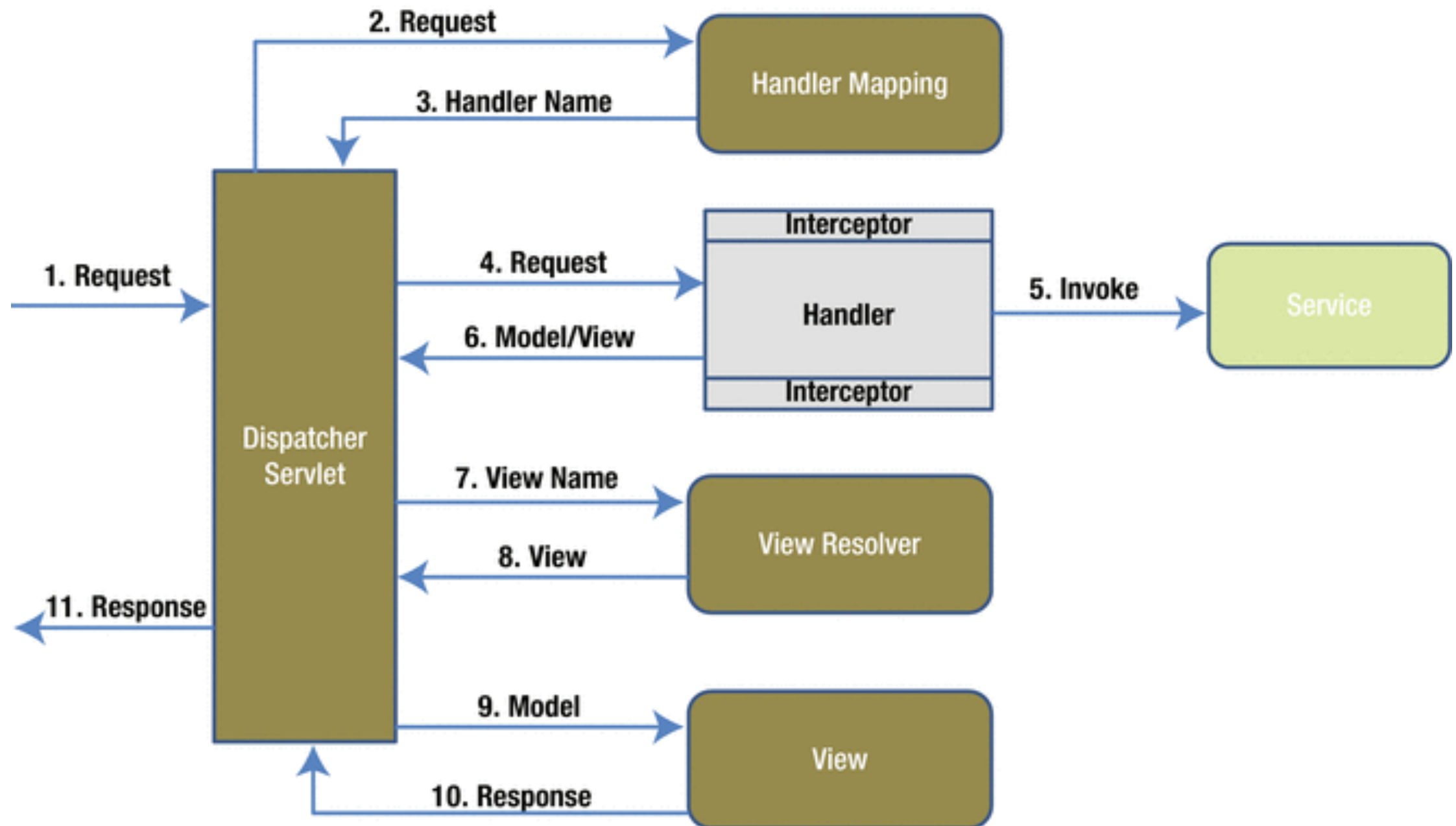
Spring Web MVC

- **Spring Web MVC** ist ein Teil des **Spring Web Modules** zur Entwicklung Web-basierter Anwendungen.
- Das Modul basiert auf der **Model-View-Controller** Architektur.
 - Ein **Modell** repräsentiert Daten oder Zustand.
 - Ein **View** bildet eine visuelle Repräsentation des Modells.
 - Ein **Controller** wickelt Benutzeraktivitäten ab.



Spring Web MVC Architektur

- Die Spring Web MVC Implementierung basiert auf dem Dispatcher Servlet, das als Einstiegspunkt für die Bearbeitung von Anfragen dient.



Web MVC Komponenten: *Controller*

- Controller werden mit dem **Stereotype** `org.springframework.stereotype.Controller` deklariert.
 - **Stereotypes** kennzeichnen Rollen oder Verantwortlichkeiten von Klassen oder Schnittstellen.

```
@Controller
public class HomeController {
    @RequestMapping("/home.html")
    public String showHomePage() {
        return "home";
    }
}
```

- **@Controller** kennzeichnet `HomeController` als MVC Controller.
- **@RequestMapping** bildet Web Requests auf Handler Klassen und Methoden ab.

Web MVC Komponenten: *Model*

- Das `org.springframework.ui.Model` Interface dient als Container für Modellattribute.

```
public interface Model {  
    Model addAttribute(String attributeName, Object attributeValue);  
    Model addAttribute(Object attributeValue);  
    Model addAllAttributes(Collection<?> attributeValues);  
    Model addAllAttributes(Map<String, ?> attributes);  
    Model mergeAttributes(Map<String, ?> attributes);  
    boolean containsAttribute(String attributeName);  
    Map<String, Object> asMap();  
}
```

- Die Methoden `addAttribute` und `addAllAttributes` Methoden fügen Attribute zum Model Objekt hinzu.

Web MVC Komponenten: *Model*

- Ein Controller kann mit einem Model arbeiten, indem dieses als Methodenparameter deklariert wird.

```
@RequestMapping("/home.html")
public String showHomePage(Model model) {
    model.addAttribute("currentDate", new Date());
    return "home";
}
```

- Alternativ kann `java.util.Map` verwendet werden.

```
@RequestMapping("/home.html")
public String showHomePage(Map model) {
    model.put("currentDate", new Date());
    return "home";
}
```

Web MVC Komponenten: *View*

- Web MVC unterstützt **View Techniken** (JSP, Velocity, XSLT u.a.) mit dem `org.springframework.web.servlet.View` Interface.

```
public interface View {
    String getContentType();
    void render(
        Map<String, ?> model,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception;
}
```

- Konkrete Implementierungen des `View` Interfaces realisieren das Rendering der Response. Beispielsweise:

<code>org.springframework.web.servlet.view.json.MappingJackson2JsonView</code>	View Implementierung zur Kodierung von ModellAttributen in JSON .
<code>org.springframework.web.servlet.view.InternalResourceView</code>	View Implementierung zur Delegation von Requests an eine JSP Seite.
<code>org.thymeleaf.spring4.view.ThymeleafView</code>	View Implementierung zur Delegation von Requests an ein Thymeleaf Template.

Web MVC Komponenten: *@RequestParam*

- Die *@RequestParam* Annotation bindet Servlet Request Parameter an Handler/Controller Methodenparameter.
- Der Request Parameterwert wird automatisch in den spezifizierten Methodenparameter konvertiert.

```
@RequestMapping("/search.html")
public String search(@RequestParam String query,
    @RequestParam("page") int pageNumber, Model model) {
    model.put("result", doSearch(query, pageNumber));
    return "showResults";
}
```

- Falls der Parameter nicht im Request enthalten ist, wird eine Exception geworfen.

Web MVC Komponenten: *@RequestMapping*

- Die `@RequestMapping` Annotation bildet einen Web Request auf eine Handler Klasse oder Handler Methode ab.
- `@RequestMapping` besitzt verschiedene Attribute, die die Abbildung einschränken:
 - **method** - Einschränkung auf HTTP Methoden wie GET, PUT etc.
 - **produces** - Einschränkung auf einen erzeugten Mediatype
 - **consumes** - Einschränkung auf einen empfangenen Mediatype
 - **headers** - Einschränkung auf bestimmte Header
 - **name** - Benennung einer Abbildung
 - **params** - Einschränkung auf Parameternamen und Wert

```
@RequestMapping(value="/saveuser.html", method=RequestMethod.POST)
public String saveUser(@RequestParam String username,
    @RequestParam String password) {
    // Save User logic
    return "success";
}
```

Web MVC Komponenten: *Pfad Variablen*

- Die @RequestMapping Annotation **unterstützt dynamische URIs** mit URI Templates.
- **URI Templates** sind URIs mit Platzhaltern/Variablen.
- Durch die **@PathVariable Annotation** können die **Platzhalter als Methodenparameter** genutzt werden.

```
@RequestMapping("/users/{username}")  
public User getUser(@PathVariable("username") String username) {  
    User user = null;  
    // Code to construct user object using username  
    return user;  
}
```

Struktur — Überblick und Einführung

Überblick des Labors

- Einordnung
- Aufgabenstellung
- Organisation

Einführung in Spring MVC

- Spring Web MVC
- Spring Boot Beispiel

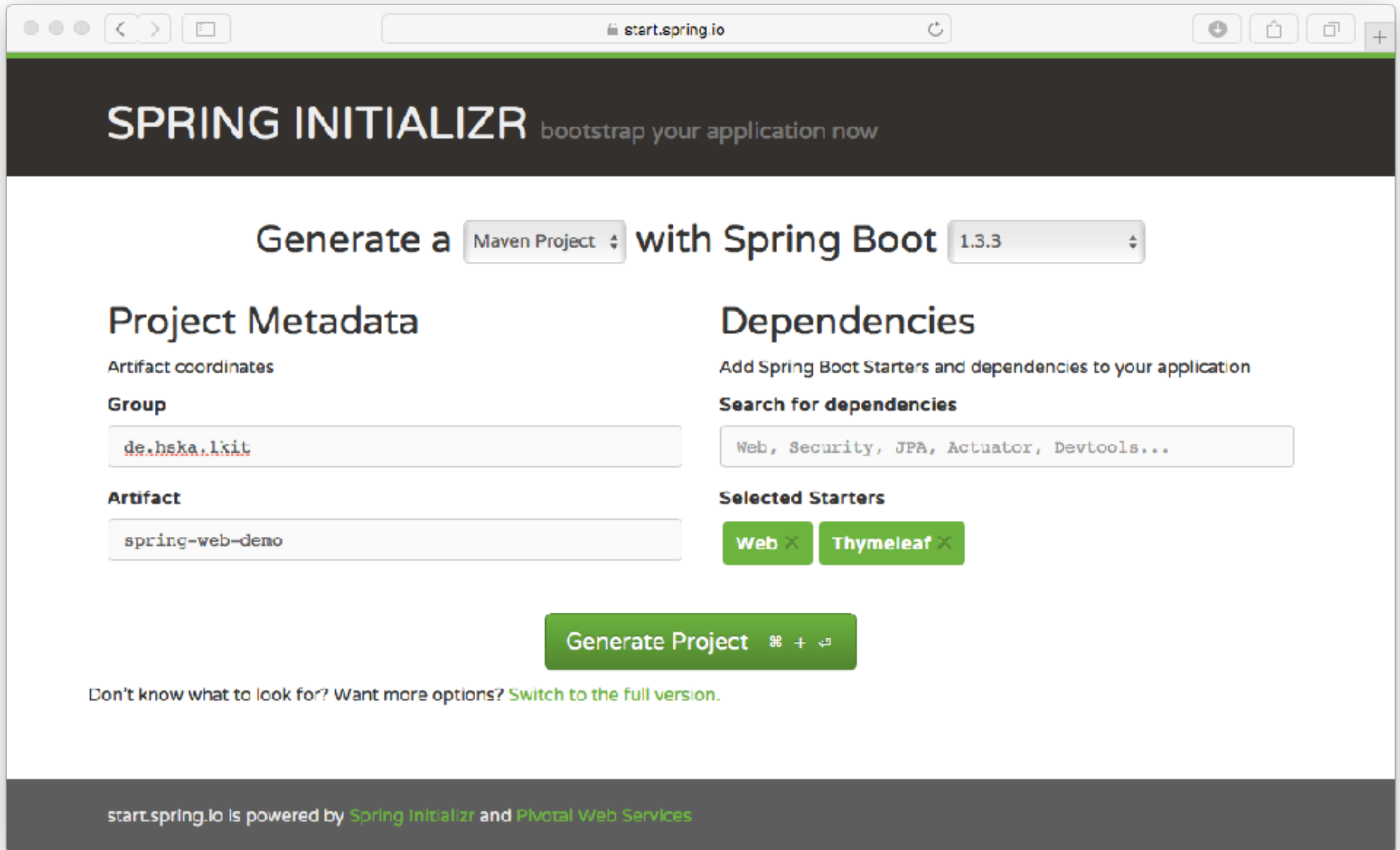
Spring Boot

- **Spring Boot** ist ein Spring Projekt, das das **Bootstrapping von Spring Anwendung** durch die Bereitstellung verschiedener **Starter Project Templates** vereinfacht.
 - Starter Projekte enthalten eine **saubere Zusammenstellung von Abhängigkeiten** basierend auf spezifischen Projekt **Features**.
 - Z.B. führt die Auswahl des **JPA Features** dazu, dass alle abhängigen JPA, Hibernate, and Spring JAR Dateien automatisch hinzugefügt werden.
- Spring Boot verfolgt einen „**eigensinnigen**“ **Ansatz** und stellt **Standardkonfigurationen** bereit, die die Entwicklung erleichtern.
 - Wenn Spring Boot z.B. JPA und MySQL JARs im Klassenpfad findet, würde es automatisch eine JPA Persistence Unit konfigurieren.
- Spring Boot unterstützt **Standalone Anwendungen** mit **eingebetteten Jetty/Tomcat Servern**, die auf jedem Rechner mit Java laufen.
- Zusätzlich enthält Spring Boot Produktivfunktionen wie **Metriken** und **Health Checks**.

Spring Tools

- Spring Boot unterstützt die zwei populärsten **Build Systeme**: **Maven** und **Gradle**.
- **Spring Tool Suite** bzw. **STS** ist eine freie **Eclipse-basierte Entwicklungsumgebung** mit Tooling zur Entwicklung Spring-basierter Anwendungen.
 - STS kann von Pivotal's Website bezogen werden:
<https://spring.io/tools/sts/all>
 - Die aktuelle Version von STS ist 3.8.3 (März 2017)
- Spring Boot bietet drei Möglichkeiten um **neue Projekte zu generieren**:
 - Spring Boot's **Starter Website** (<http://start.spring.io>)
 - **Spring Tool Suite (STS) IDE**
 - **Boot command line interface (CLI)**

SPRING INITIALIZR



The screenshot shows the Spring Initializr web application in a browser window. The address bar shows 'start.spring.io'. The page has a dark header with the 'SPRING INITIALIZR' logo and the tagline 'bootstrap your application now'. Below the header, there's a main section titled 'Generate a' followed by a dropdown menu set to 'Maven Project', then 'with Spring Boot' followed by a dropdown menu set to '1.3.3'. The page is divided into two columns. The left column is titled 'Project Metadata' and contains two input fields: 'Group' with the value 'de.hska.lkit' and 'Artifact' with the value 'spring-web-demo'. The right column is titled 'Dependencies' and contains a search bar with the text 'Web, Security, JPA, Actuator, Devtools...'. Below the search bar, there's a section titled 'Selected Starters' showing two green buttons: 'Web' and 'Thymeleaf', each with a close icon. At the bottom of the main content area, there's a large green button labeled 'Generate Project' with a download icon. Below this button, there's a link that says 'Don't know what to look for? Want more options? Switch to the full version.' The footer of the page states 'start.spring.io is powered by Spring Initializr and Pivotal Web Services'.

start.spring.io

SPRING INITIALIZR

bootstrap your application now

Generate a Maven Project with Spring Boot 1.3.3

Project Metadata

Artifact coordinates

Group

de.hska.lkit

Artifact

spring-web-demo

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Starters

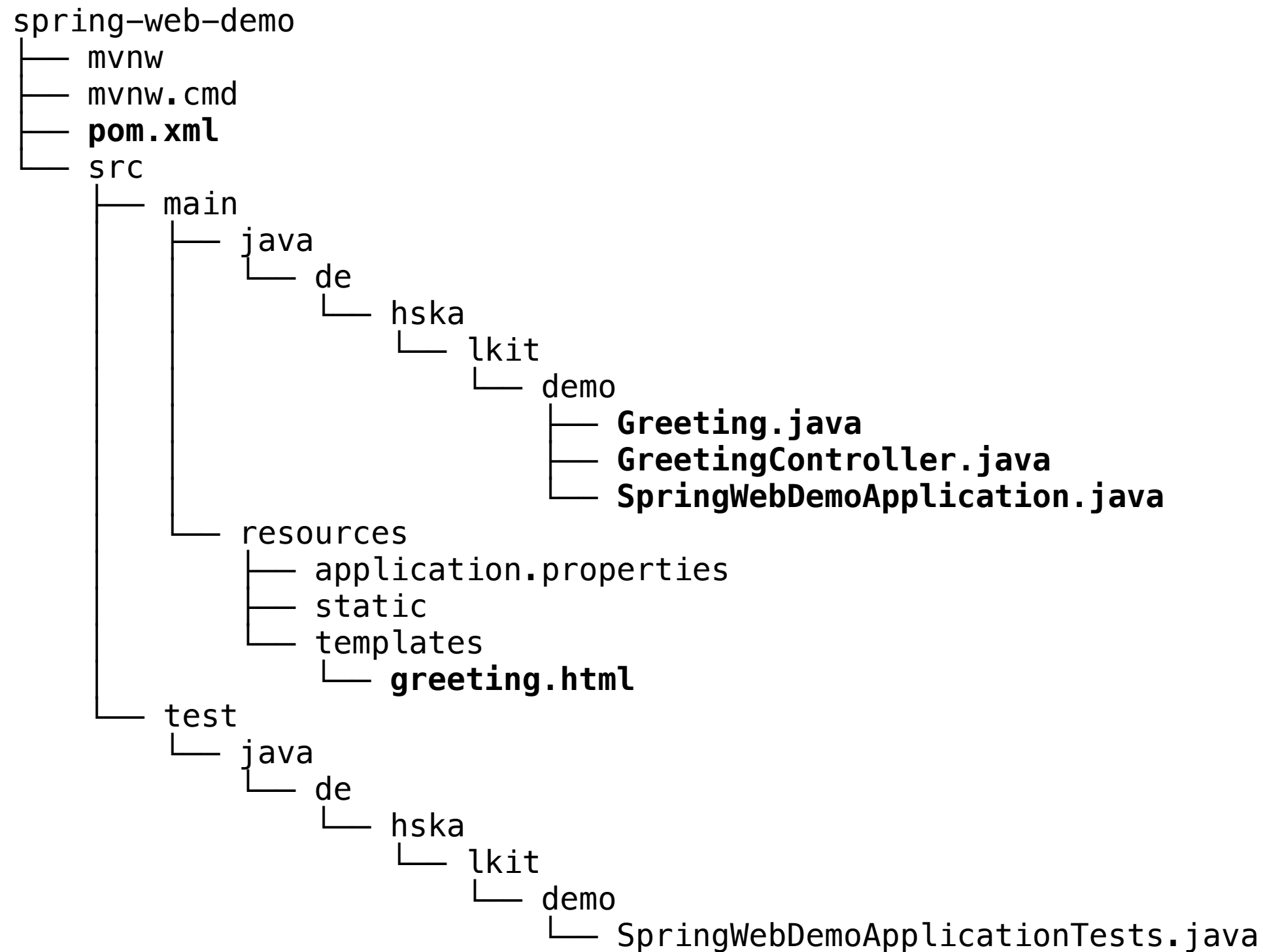
Web X Thymeleaf X

Generate Project

Don't know what to look for? Want more options? [Switch to the full version.](#)

start.spring.io is powered by [Spring Initializr](#) and [Pivotal Web Services](#)

Spring Boot Projekt Struktur



Maven POM

```
<project ... >
  <groupId>de.hska.lkit</groupId>
  <artifactId>spring-web-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>spring-rest-demo</name>
  ...
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    ...
  </parent>
  ...
  <dependencies> <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency> <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency> ... </dependencies>

  <build> <plugins> <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
  </plugin> </plugins> </build>
</project>
```

SpringWebDemoApplication.java

- SpringWebDemoApplication.java ist die **Basisklasse** der Anwendung und enthält die main() Methode.
- `@SpringBootApplication` ist equivalent zu `@Configuration`, `@ComponentScan` `@EnableAutoConfiguration`

```
package de.hska.lkit.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringWebDemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringWebDemoApplication.class, args);
    }
}
```

Greeting.java

- Ein POJO dient als Datencontainer.

```
package de.hska.lkit.demo;

public class Greeting {
    private String name;
    private String content;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    // weitere Getter und Setter nicht gezeigt
}
```

GreetingController.java

- Die `@Controller` Annotation weist auf einen **Spring MVC Controller** hin.
- `@RequestMapping` bestimmt die **URL**, der die Methode zugeordnet ist.
- `@ModelAttribute` bindet Teile des HTTP-Requests (Formulardaten).
- Der **Model**-Parameter ist später im Template verfügbar.

```
package de.hska.lkit.demo;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class GreetingController {

    @RequestMapping(value = "/greeting")
    public String greetingSubmit(@ModelAttribute Greeting greeting, Model model) {
        model.addAttribute("greeting", greeting != null ? greeting : new Greeting());
        return "greeting";
    }
}
```

Thymeleaf Template (Teil 1)

- Thymeleaf (TH) ist eine Java XML/XHTML/HTML5 **Template Engine**.
 - Thymeleaf kann **Transformationen** auf **Template Dateien** anwenden um Daten und Text anzuzeigen, die **von einer Anwendung erzeugt** wurden.
- Thymeleaf basiert auf **XML Tags und Attributen**, die die **Ausführung vordefinierter Logik** auf dem DOM definieren.
 - Templates enthalten keinen (wenig) Code und sind **wohlgeformt**.

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Getting Started</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
  <h1>Greeting!</h1>
  <p th:text="'Hello ' + ( ${greeting.name} != null ? ${greeting.name} : 'World')"/>
  <p th:text="${greeting.content} != null ? 'Your message: ' + ${greeting.content}"/>
```

TH-Namespace

Conditional Variable Expression ersetzt den Text des Elements

Thymeleaf Template (Teil 2)

- Thymeleaf enthält viele th: Attribute für **XHTML/HTML5 Attribute** mit entsprechenden Namen sowie **spezielle Attribute** (z.B. **th:text**).
- **Werte von Thymeleaf Attributen** werden oft als **Expressions** gesetzt.
 - **Variable Expressions** werden auf **Model Attributen** ausgeführt.
 - **Selection Expressions** werden auf zuvor gewählten Objekten ausgeführt.
 - **URL Expressions** können Kontextinformationen zu URLs hinzufügen.

...

```
<h3>Your Comment</h3>
```

TH-Attribut setzt HTML Action als **URL Expression**

```
<form action="#" th:action="@{/greeting}" th:object="${greeting}" method="post">
  <p> Name <input type="text" th:field="*{name}" /></p>
  <p> Message <input type="text" th:field="*{content}" /></p>
  <p> <input type="submit" value="Submit" /></p>
</form>
```

```
<a th:href="@{/greeting}">Reset</a>
```

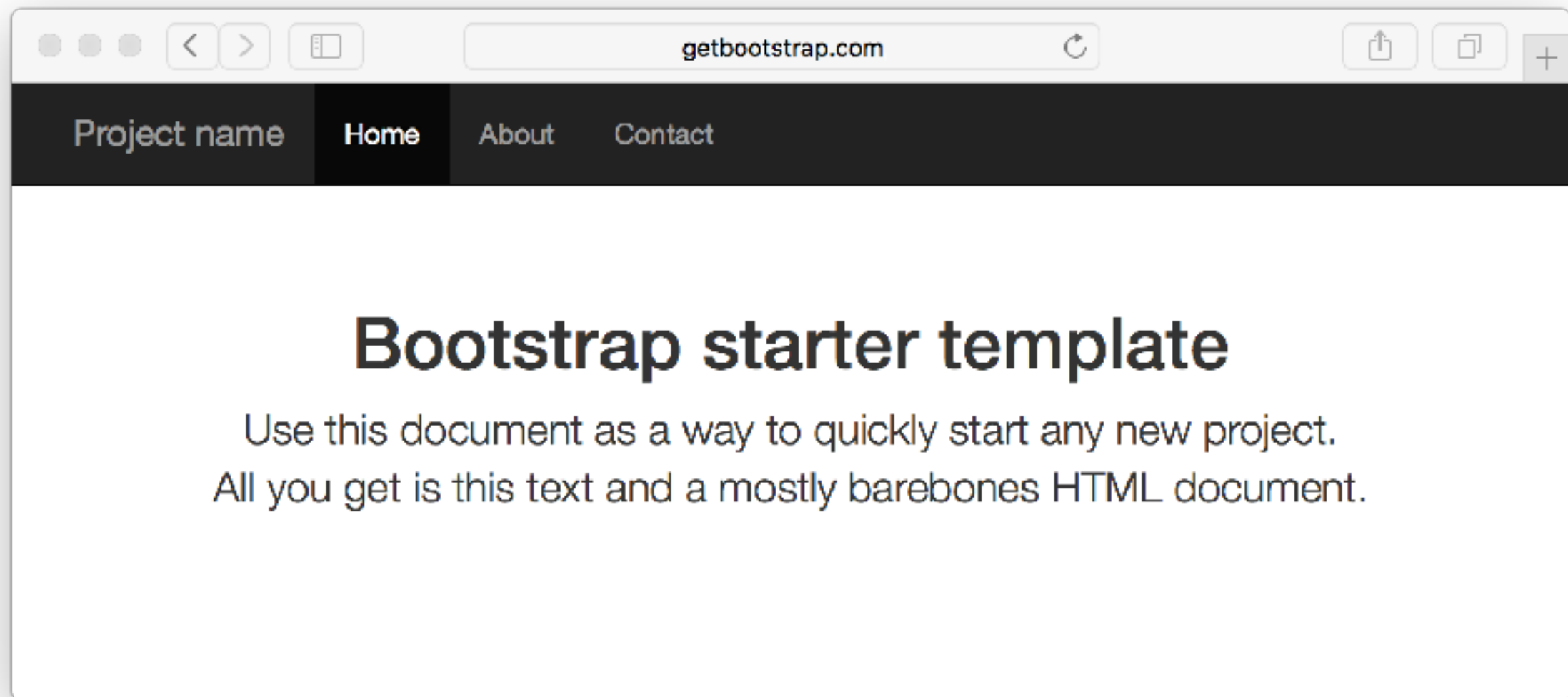
```
</body>
</html>
```

Selection Expression bindet ein Feld des selektierten Objekts an das Input element

Variable Expression selektiert ein Objekt im Model

Oberflächenentwicklung mit Bootstrap

- Das **CSS-Framework Bootstrap** wurde 2011 bei Twitter gestartet. Heute ist es **Open Source** und mit 93K GitHub Stars sehr populär.
- Es enthält allgemeine **CSS-Einstellungen** (Typographie, Grids...), Stile für viele **HTML-Elemente** und spezielle **Komponenten** (Dropdowns, Navbars ...) sowie **Javascript Erweiterungen** (Alerts, Popover ...) u.v.m.
- Bootstrap unterstützt **responsives Webdesign** ("mobile first").



WebJars - JAR-basierte Web-Bibliotheken per Maven verwalten und mit Spring Boot integrieren

```
...
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <!--WebJars -->
  <dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>3.3.4</version>
  </dependency>
  <dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>2.1.4</version>
  </dependency>
</dependencies>
...
```

Spring Boot stellt diese Bibliotheken als statische Ressourcen des internen Webserver zur Verfügung.

<http://www.webjars.org>

Template mit Bootstrap (Teil 1: HTML Skelett)

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Getting Started</title>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<!-- The above 3 meta tags *must* come first in the head -->
<link th:href="@{/webjars/bootstrap/3.3.4/css/bootstrap.min.css}"
      rel="stylesheet" media="screen" />
</head>
<body>
  <!-- content goes here (see next slide) -->

  <!-- Bootstrap core JavaScript
  ===== -->
  <!-- Placed at the end of the document so the pages load faster -->
  <script th:src="@{/webjars/jquery/2.1.4/jquery.min.js}"></script>
  <script th:src="@{/webjars/bootstrap/3.3.4/js/bootstrap.min.js}"></script>
</body>
</html>
```

Aktivierung des Bootstrap Stylesheet

Aktivierung von Javascript Erweiterungen

Template mit Bootstrap (Teil 2: erweiterter Inhalt)

```

<div class="container">
  <div class="jumbotron">
    <h1>Greeting!</h1>
    <p th:text="'Hello ' + (${greeting.name}!=null?${greeting.name}+'!': 'World!')"/>
    <p th:text="${greeting.content}!=null?' Your message: ' + ${greeting.content}'"/>
    <h3>Your Comment</h3>
    <form class="form-inline" action="#" th:action="@{/greeting}"
      th:object="${greeting}" method="post">
      <div class="form-group">
        <label for="f1"> Name </label>
        <input placeholder="Bob" id="f1" type="text" th:field="*{name}" />
      </div>
      <div class="form-group">
        <label for="f2"> Message </label>
        <input placeholder="hello" id="f2" type="text" th:field="*{content}" />
      </div>
      <input type="submit" value="Submit"/>
    </form>
  </div>
  <a class="btn btn-default" th:href="@{/greeting}">Reset</a>
</div>

```

Container Struktur mit Jumbotron Komponente

Formate für Formular Elemente

Button Komponente

Zum Nach- und Weiterlesen

Literatur

[Schaefer 2014a] C. Schaefer, C. Ho, R. Harrop, "Pro Spring", 4th ed., Apress, 2014

Sekundär: [Varanasi 2015] Balaji Varanasi, Sudha Belida, "Spring REST", Apress, 2015

Aus dem Web

Spring MVC

[Spring 2016a] Spring Reference: MVC,

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>

[Spring 2016b] Spring Getting Started Guides, "Serving Web Content with Spring MVC" und "Handling Form Submission", <https://spring.io/guides>

Thymeleaf

[Thymeleaf 2013a] Getting started with the Standard dialects in 5 minutes,

<http://www.thymeleaf.org/doc/articles/standarddialect5minutes.html>

[Thymeleaf 2014a] Tutorial: Using Thymeleaf,

<http://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html>

Bootstrap

[bootstrap 2016a] Bootstrap, <http://getbootstrap.com/>

[w3schools 2016a] Bootstrap 3 Tutorial, <http://www.w3schools.com/bootstrap/>

Nächste Aufgabe

A1 Konzeption und Gestaltung (bis 9.11)

- Analyse der **Anforderungen** (Use Case Diagramm oder textuell)
- Entwurf der logische **Seitenstruktur/Navigation** (Zustandsdiagramm)
- Erstellen von **Mockups** (Spring Boot Projekt)
- Entwurf des **Datenmodells** (Redis Datenstrukturen und Key-Muster)

Hinweis: vollständige Spezifikation und Aufgabenstellung als PDF im ILIAS