

Softwareentwicklung (SW)

Einführung in die Webarchitektur und Spring Framework

Prof. Dr. Alixandre Santana
alixandre.santana@oth-regensburg.de

Wintersemester 2023/2024

- einen Überblick über die Entwicklung der Webentwicklung zu geben
- das Webcontainer-Konzept zu verstehen
- das MVC(S)-Modell zu verstehen
- die Spring-Architektur zu verstehen
- ein „Hello World“ mit Spring Boot zu implementieren

- Entwicklung der Webarchitektur
- Java-basierte Webarchitektur
- Das Model-View-Controller (MVC)-Muster
- Dependency Injection
- Das Spring Framework
- Ein Hello World-Projekt

.1995 : HTML 2.0

Es konnte das Ändern des Seitenhintergrunds, der Textfarbe, des Textgesichts, die Verwendung von Tabellen und Textfeldern usw. unterstützen.

Ungefähr zu dieser Zeit wurde W3C (The World Wide Web Consortium) gegründet – eine Organisation, die Webstandards entwickelt.



Members of the World Wide Web Consortium (W3, n.d.)

<https://medium.com/@jasmineharwood/the-evolution-of-html-837f85e6c1ee>





[Novos](#) [Inclusões](#)

[Cadê? Mail](#) [Veja](#)

[Cadê? Você](#)

[Download](#)

[AQUI!](#)

[Cadê? Livros](#)

[Cadê? EnglishTown](#)

Consulta

Busca

[Opções de busca](#)

[Lojas Americanas - Clique Aqui!!!](#)



PUBLICIDADE

[Ciência e Tecnologia](#)

[Institutos](#) [Publicações](#)

[Cultura](#)

[Museus](#) [Música](#) [MP3](#)

[Esportes](#)

[Automobilismo](#) [Futebol](#)

[Governo](#)

[Federal](#) [Estados](#) [Concursos](#)

[Informática](#)

[Empresas](#) [Software](#) [Vírus](#)

[Lazer](#)

[Carnaval](#) [Turismo](#) [Infantil](#)

[Referência](#)

[Bibliotecas](#) [Dicionários](#)

[Serviços](#)

[Beleza](#) [Companhias Aéreas](#)



[Compras Online](#)

[CD](#) [Informática](#) [Livros](#)



[Educação](#)

[Escolas](#) [Universidades](#)



[Finanças](#)

[Bancos](#) [Bolsas](#) [Seguros](#)



[Indústria e Comércio](#)

[Veículos](#) [Telecomunicações](#)



[Internet](#)

[E-mail Grátis](#) [Provedores](#)



[Notícias](#)

[Jornais](#) [Revistas](#)



[Saúde](#)

[Hospitais](#) [Instituições](#)



[Sociedade](#)

[ONGs](#) [Pessoais](#) [Religião](#)

[Noticias](#)

[LAUDO: CANTOR
ESTAVA EM ALTA
VELOCIDADE](#)

[Mais quatro
desenhos no Cartoon](#)

[Eliminatórias da
Copa começam
sábado](#)

[Mais...](#)

[Destaques](#)

[Sinal dos Tempos
Internet, o sintoma
de uma Nova Era](#)

[Febre Amarela
Um sério problema
colorido](#)

[Comportamento
Como seremos no
próximo milênio?](#)

.1997 : HTML 3.2

HTML 3.2 tables, applets, text flow around images, subscripts and superscripts

.1999: HTML 4.01 (W3C)

cascading style sheets (css)

.2014: HTML5

"HTML5 can be used to write web applications that still work when you're not connected to the net; to tell websites where you are physically located; to handle high definition video; and to deliver extraordinary graphics."

<https://medium.com/@jasmineharwood/the-evolution-of-html-837f85e6c1ee>



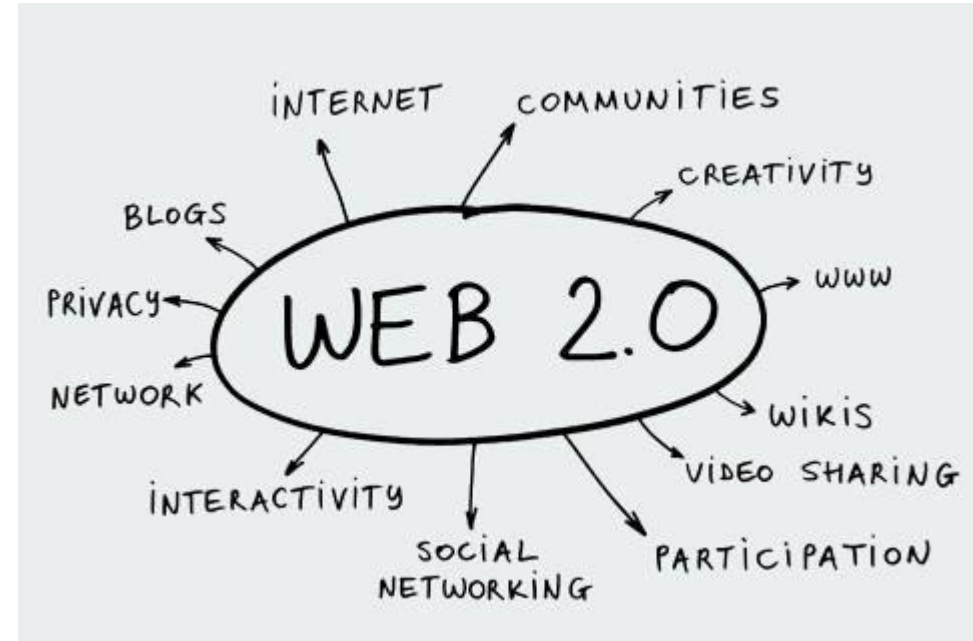
Entwicklung der Browser



Wir haben die Art und Weise, wie wir das Internet nutzen, verändert

Web 2.0 (Tim O'Reilly, 2004)

- Dynamic websites.
- die Revolution von Blogs und Chats, kollaborative soziale Medien, **soziale Netzwerke**, neue Möglichkeiten zur Monetarisierung der von Internetnutzern selbst produzierten Inhalte.



Web 3.0?

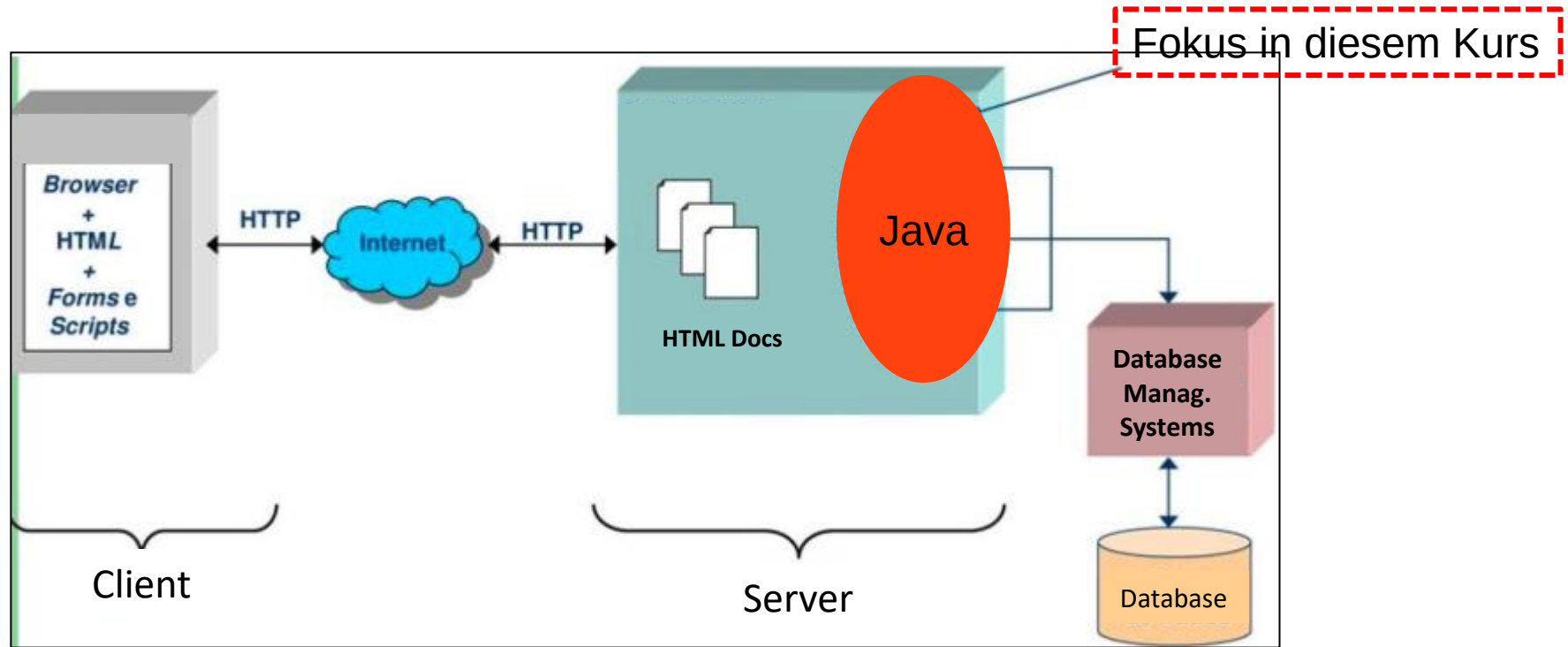
*There's definitely something new brewing, but I bet we will call it **something other than Web 3.0**. And it's increasingly likely that it will be far **broader and more pervasive than the web, as mobile technology, sensors, speech recognition**, and many other new technologies make computing far more ambient than it is today.*

Tim O'Reilly (2007).

Agenda

- Entwicklung der Webarchitektur
- Java-basierte Webarchitektur
- Das Model-View-Controller (MVC)-Muster
- Dependency Injection
- Das Spring Framework
- Ein Hello World-Projekt

Generische WEB-Architektur



JAVA-Architektur



Java Micro Edition (JME) is a stripped-down version of the Java programming language that is specifically designed for resource-constrained devices, such as mobile phones, sensors and embedded systems.

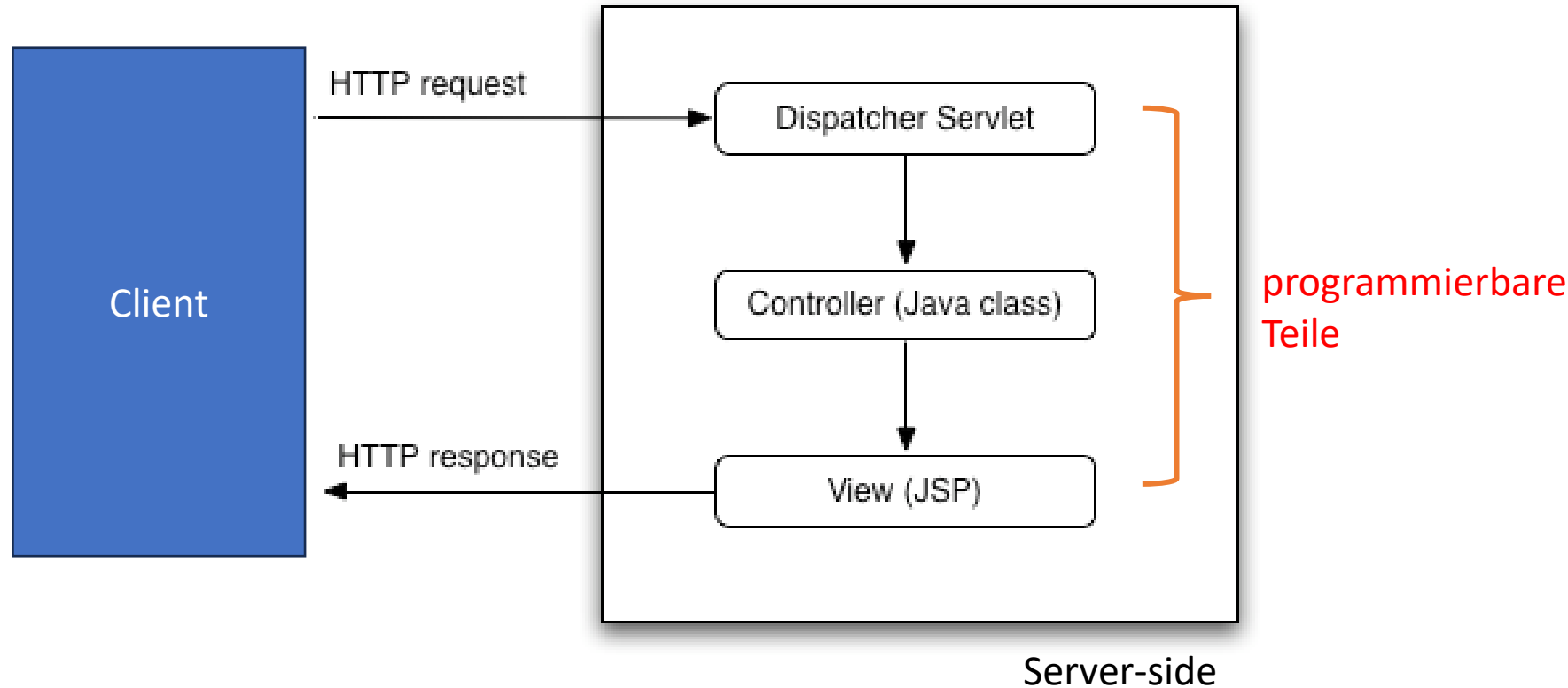


is the premier platform for rapidly developing and deploying secure, portable applications that run on server and desktop systems spanning most operating systems.

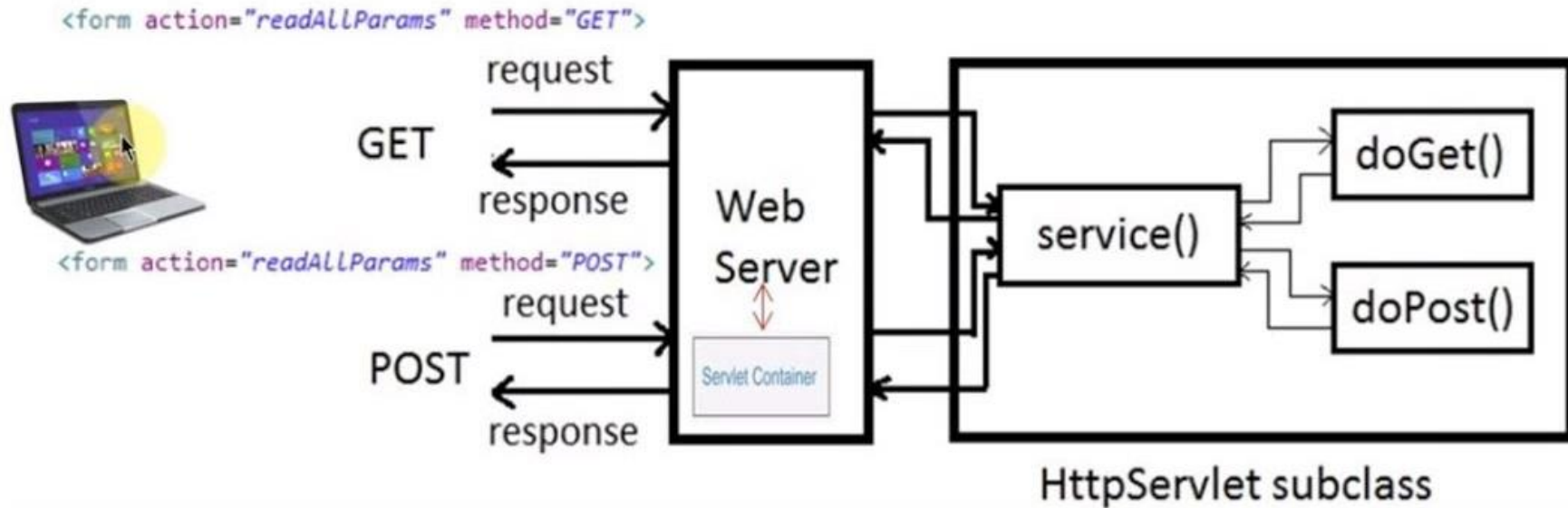


- ▶ **Web applications**
- ▶ **Distributed applications**
- ▶ **Transactional Applications**

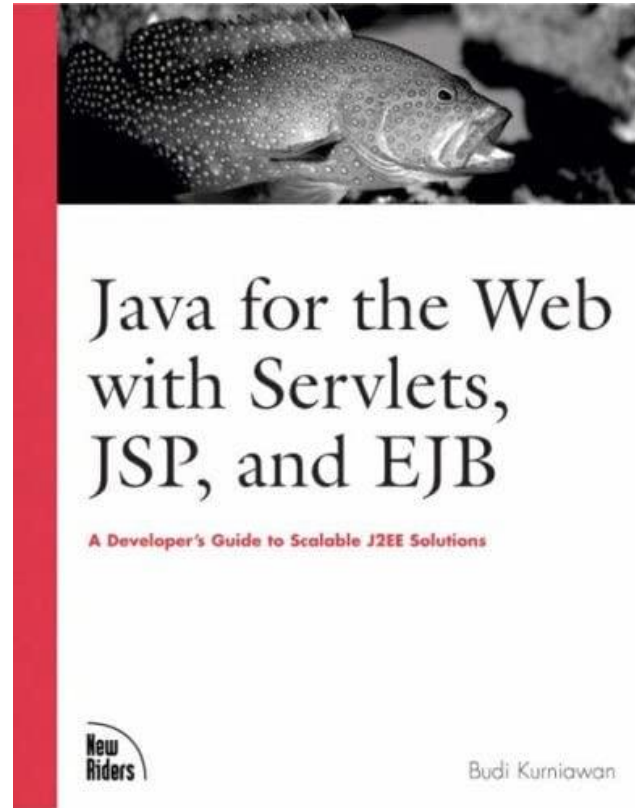
Web-Container



Im Hintergrund: Servlets...



Java-Lösungen für das Web



Back to the JSP Old Days...

```
BeispieleJSP.jsp
1  <%@page import="java.sql.*"%>
2  <%@page import="com.mysql.jdbc.PreparedStatement"%>
3  <%@page import="com.mysql.jdbc.Connection"%>
4  <%@ page language="java" contentType="text/html; charset=UTF-8"
5      pageEncoding="UTF-8"%>
6  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
7  <html>
8  <head>
9  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10 <title>Insert title here</title>
11 </head>
12 <body>
13 <%
14 Connection con;
15 Class.forName("com.mysql.jdbc.Driver");
16 con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost:3306/java", "java", "123456");
17 PreparedStatement ps=(PreparedStatement)con.prepareStatement("SELECT * from users");
18 ResultSet rs=ps.executeQuery();
19 while(rs.next()){
20     String username=rs.getString("username");
21     String password=rs.getString("password");
22     out.println("Username "+username+" Pass "+password+"<hr>");
23 }
24 %>
25 </body>
26 </html>
27
```

Heute: Java-Lösungen für das Web



EJB 3



JDBC



JUnit

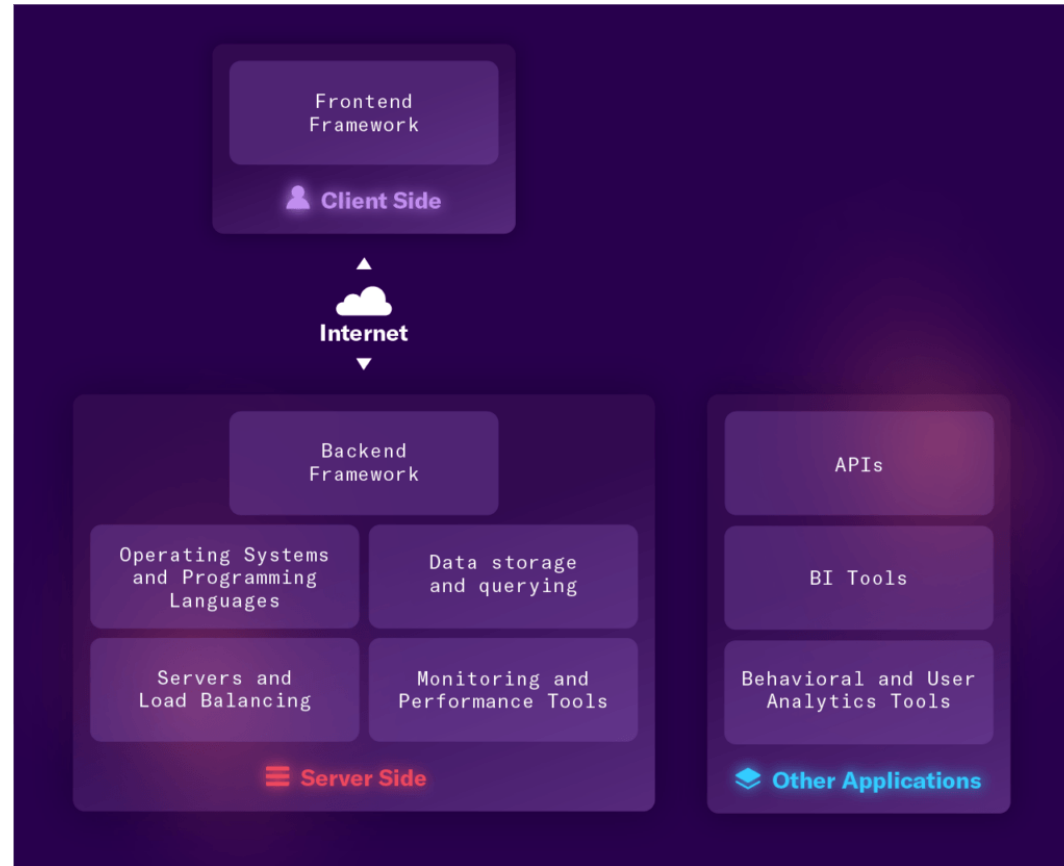
Struts²



maven

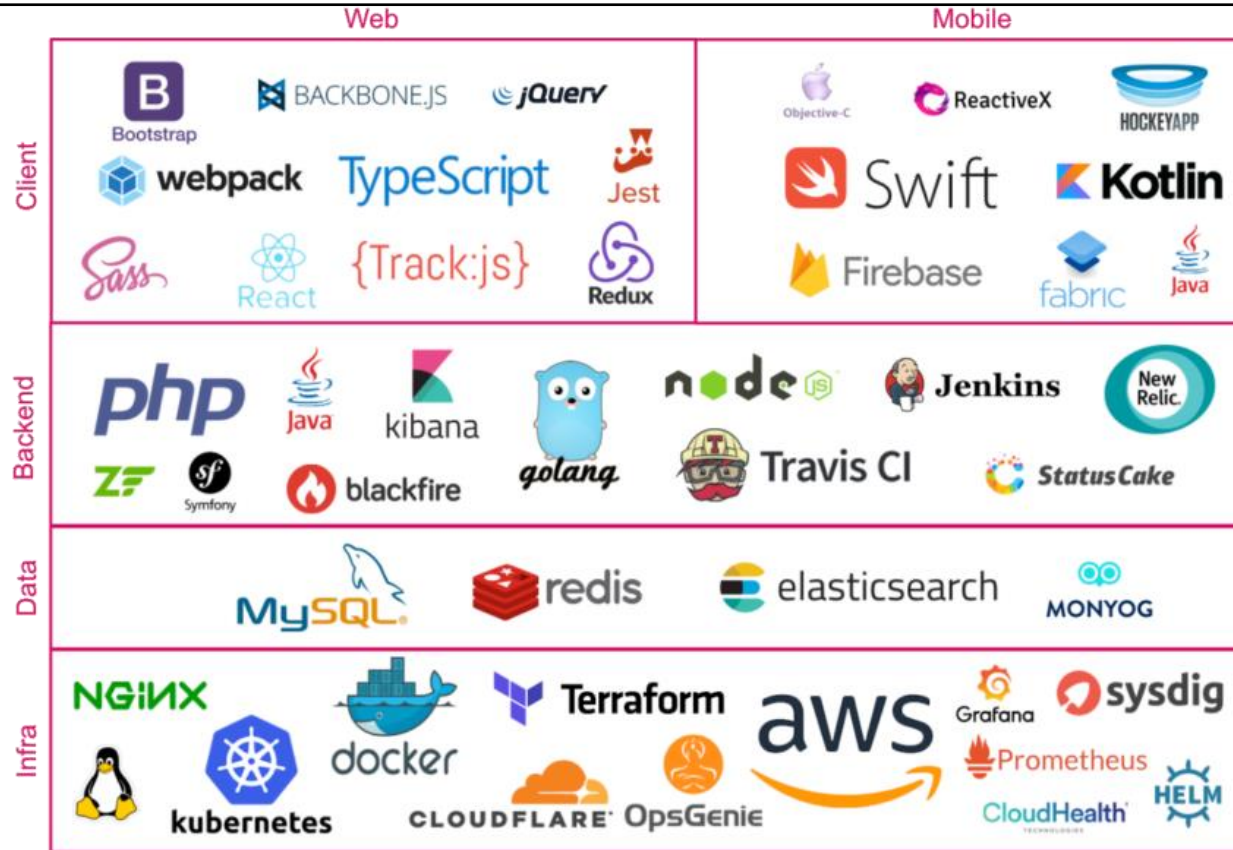


Was ist Ihr Tech Stack?



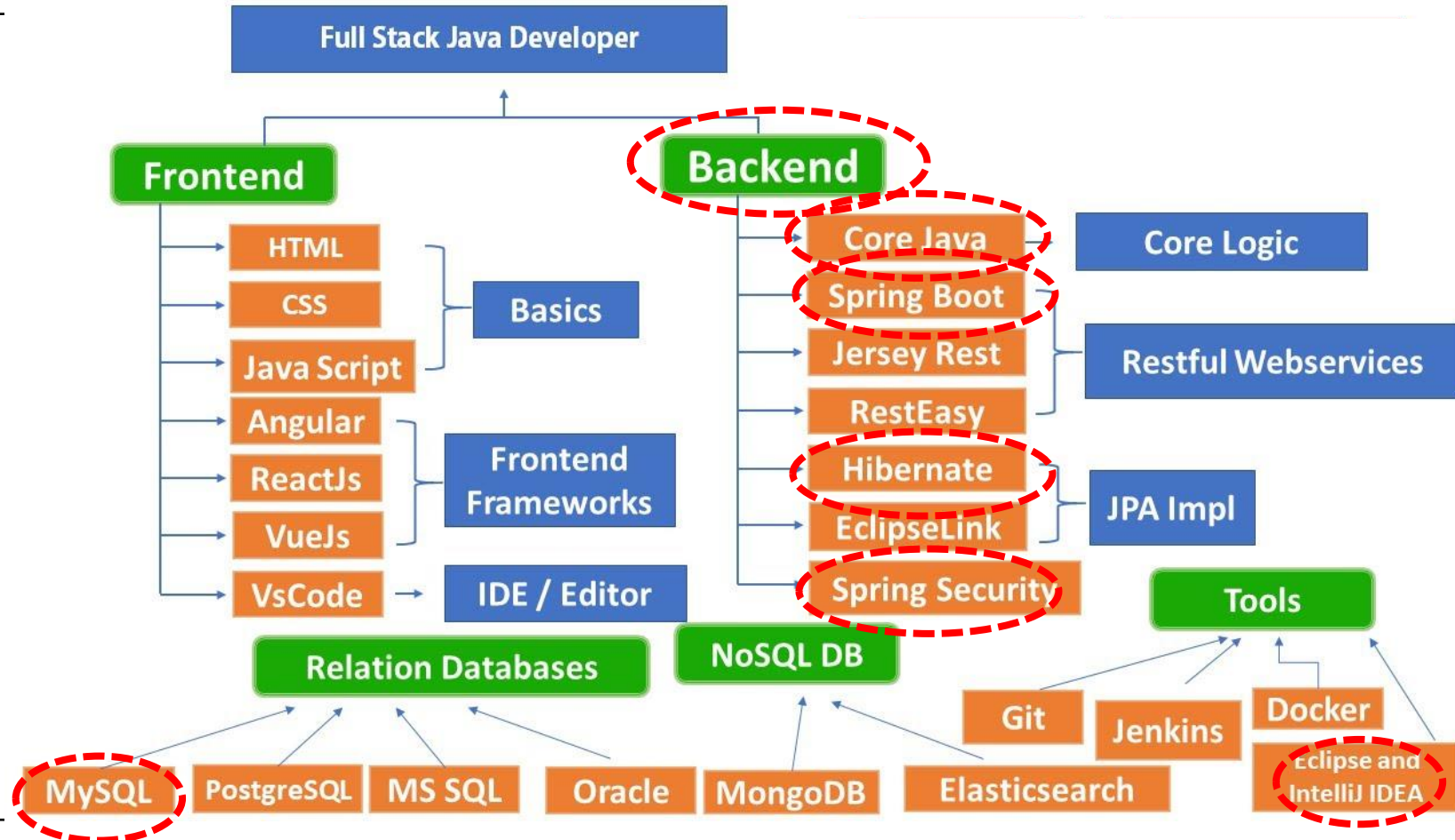
<https://heap.io/topics/what-is-a-tech-stack>

Was ist Ihr Tech Stack?



<https://tahanima.github.io/2021/10/30/tech-stack-of-software-companies-of-bangladesh/>
<https://survey.stackoverflow.co/2023/#most-popular-technologies-language-prof>

Was ist Ihr Tech Stack?



Agenda

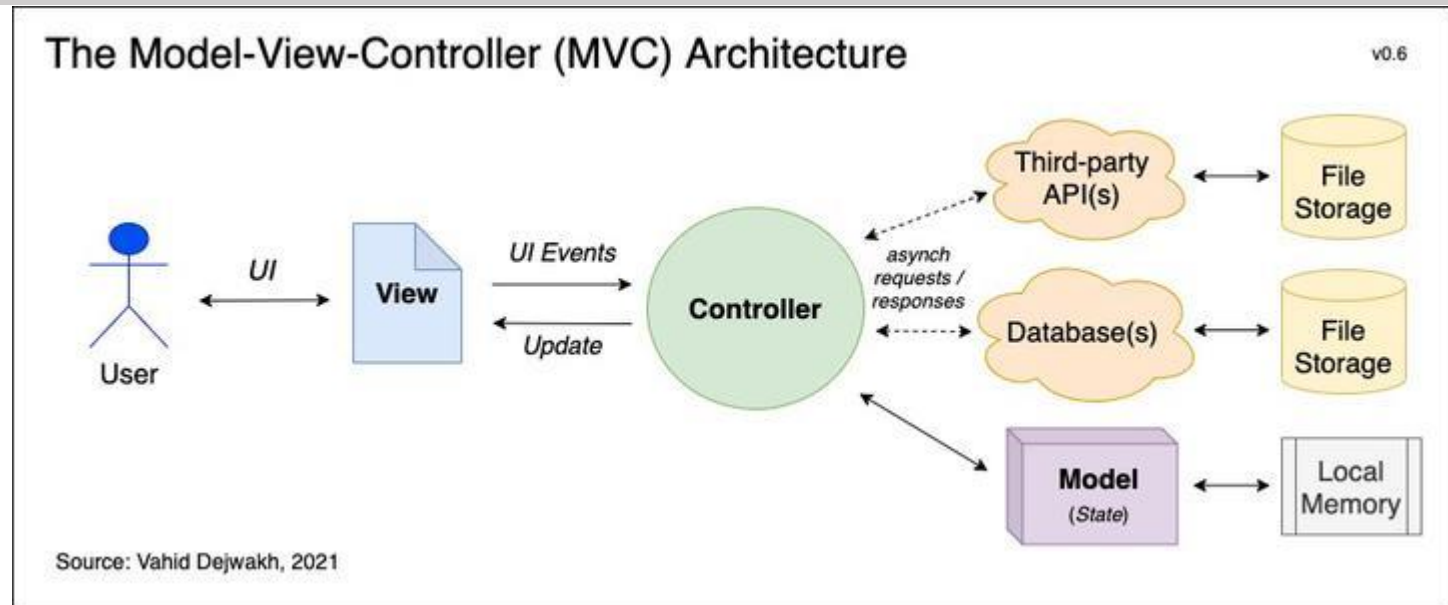
- Entwicklung der Webarchitektur
- Java-basierte Webarchitektur
- Das Model-View-Controller (MVC)-Muster
- Dependency Injection and Inversion of Control
- Das Spring Framework
- Ein Hello World-Projekt

Back to the JSP Old Days...

```
BeispieleJSP.jsp
1  <%@page import="java.sql.*"%>
2  <%@page import="com.mysql.jdbc.PreparedStatement"%>
3  <%@page import="com.mysql.jdbc.Connection"%>
4  <%@ page language="java" contentType="text/html; charset=UTF-8"
5      pageEncoding="UTF-8"%>
6  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
7  <html>
8  <head>
9  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10 <title>Insert title here</title>
11 </head>
12 <body>
13 <%
14 Connection con;
15 Class.forName("com.mysql.jdbc.Driver");
16 con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost:3306/java", "java", "123456");
17 PreparedStatement ps=(PreparedStatement)con.prepareStatement("SELECT * from users");
18 ResultSet rs=ps.executeQuery();
19 while(rs.next()){
20     String username=rs.getString("username");
21     String password=rs.getString("password");
22     out.println("Username "+username+" Pass "+password+"<hr>");
23 }
24 %>
25 </body>
26 </html>
27
```

Model-View-Controller

Projektdateien auf verschiedene Verantwortlichkeiten zu teilen, was eine bessere Organisation und einfachere Wartung ermöglicht.



<https://vahid.blog/post/2021-04-16-understanding-the-model-view-controller-mvc-pattern/>

MVC-Model

- Vollständige, gekapselte Darstellung des Objektes/der Daten, das/die von der Anwendung bearbeitet wird/werden (z.B. ein Customer)
- Stellt gewisse Dienstleistungen (services) zur Verfügung, um die Daten zu manipulieren (z.B. Berechnung, Abspeichern)
- Verantwortlich für die Durchführung von Berechnungen

```
1 // Java bean for Person
2 public class Person {
3     // Private variables
4     private String firstName;
5     private String lastName;
6
7     // Getters and setters for variables
8     public String getFirstName() {
9         return firstName;
10    }
11    public void setFirstName (String firstName) {
12        this.firstName=firstName;
13    }
14    public String getLastName() {
15        return lastName;
16    }
17    public void setLastName (String lastName) {
18        this.lastName=lastName;
19    }
20 }
```

MVC-View

- Zeigt eine spezielle Ansicht des Model-Objektes/der Model-Daten
- Verantwortlich für die Präsentation

```
addStudent.html
1 <!DOCTYPE HTML>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <title>Add Student</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6   <link type="text/css" rel="stylesheet" href="bootstrap.min.css" th:href="@{bootstrap.min.css}" />
7
8 </head>
9 <body>
10   <h1>New student</h1>
11   <div>
12     <form th:object="${student}" th:action="@{save}" action="#" method="post">
13       <input type="text" th:field="*{firstName}" class="form-control"
14         placeholder="Firstname" />
15       <div style="clear: both; display: block; height: 10px;"></div>
16       <input type="text" th:field="*{lastName}" class="form-control"
17         placeholder="Lastname" />
18       <div style="clear: both; display: block; height: 10px;"></div>
19       <input type="text" th:field="*{email}" class="form-control"
20         placeholder="Email" />
21       <div style="clear: both; display: block; height: 10px;"></div>
22       <input type="submit" class="btn btn-success" value="Save"></input>
23     </form>
24   </div>
25 </body>
26 </html>
```

Model-View Interaktion

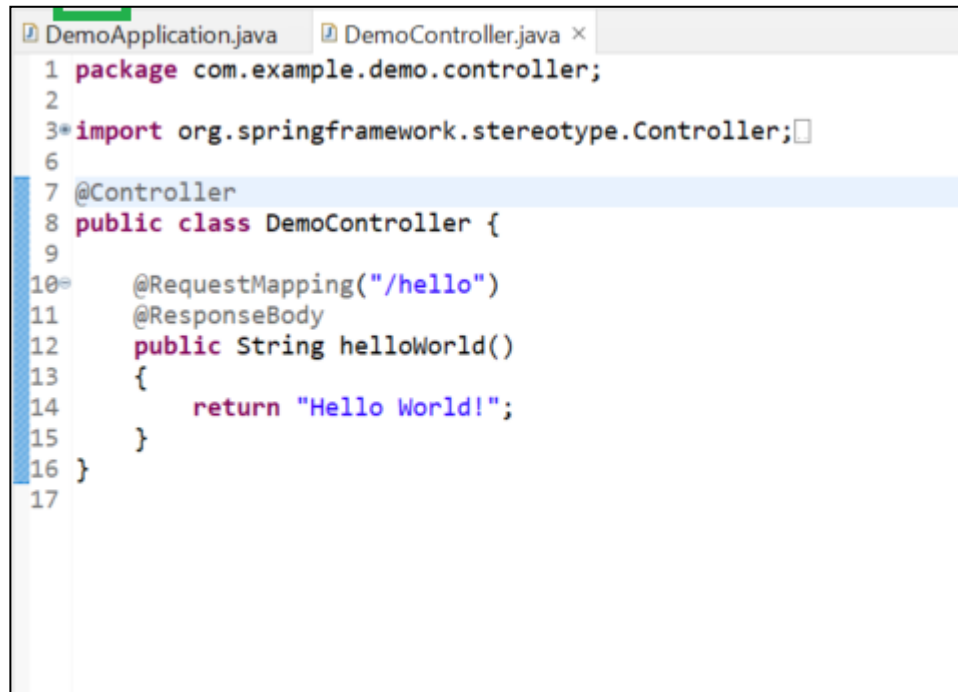
- Model könnten wir auch mit beliebigen **anderen View-Technologien** kombinieren, wie zum Beispiel:

- JSP - JavaServer Pages : <https://www.oracle.com/java/technologies/jspt.html>
- Groovy Markup Templates : http://groovy-lang.org/templating.html#_the_markuptemplateengine
- JSF - Java Server Faces : <https://www.oracle.com/java/technologies/javaserverfaces.html>

Es gibt noch viele mehr, hier werden **wir aber ausschließlich auf Thymeleaf konzentrieren.**

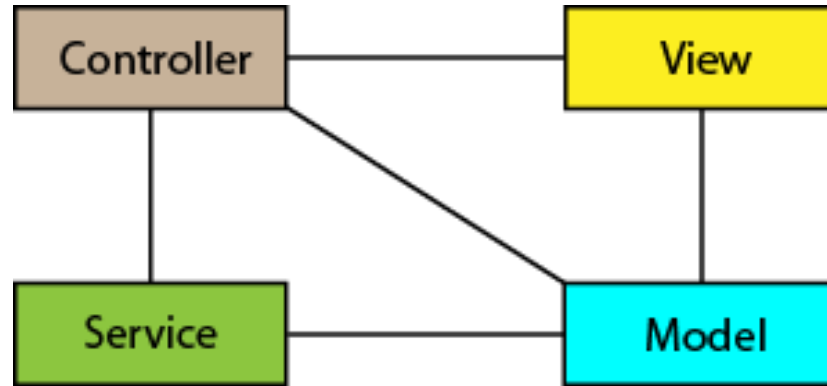
MVC-Controller

- Reagiert auf Benutzereingaben und verwendet Informationen aus dem View um das Model-Objekt/die Model-Daten zu modifizieren
- Verantwortlich für die Interaktion



```
DemoApplication.java DemoController.java x
1 package com.example.demo.controller;
2
3 import org.springframework.stereotype.Controller;
4
5
6
7 @Controller
8 public class DemoController {
9
10     @RequestMapping("/hello")
11     @ResponseBody
12     public String helloWorld()
13     {
14         return "Hello World!";
15     }
16 }
17
```

Model-View-Service-Controller



Quelle: <https://glossar.hs-augsburg.de/Model-View-Controller-Service-Paradigma>

Die Idee von MVCS besteht darin, eine Serviceschicht zwischen dem Controller und dem Modell zu haben, um die gesamte **Geschäftslogik zu kapseln, die sich im Controller befinden könnte**.

Auf diese Weise sind die Controller lediglich dazu da, die Ausführung weiterzuleiten und zu steuern. Und Sie können einen Dienst in vielen Controllern (z. B. einer Website und einem Webservice) aufrufen, ohne Code zu duplizieren.

Model-View-Service-Controller

Die Serviceschicht wäre dann verantwortlich für:

- Abrufen und Erstellen Ihres „Modells“ aus verschiedenen Datenquellen (oder Datenzugriffsobjekten).
- Aktualisieren von Werten über verschiedene Repositorys/Ressourcen hinweg.
- Durchführen anwendungsspezifischer Logik und Manipulationen usw.

Quelle: <https://glossar.hs-augsburg.de/Model-View-Controller-Service-Paradigma>

DAO Pattern

- **Data Access Object (DAO**, [englisch](#) für *Datenzugriffsobjekt*) ist ein [Entwurfsmuster](#), das den Zugriff auf unterschiedliche Arten von Datenquellen (z. B. Datenbanken, Dateisystem) so kapselt, dass die angesprochene Datenquelle ausgetauscht werden kann, ohne dass der aufrufende Code geändert werden muss.

```
public interface Dao<User> {  
  
    Optional<User> get(long id);  
  
    List<User> getAll();  
  
    void save(User u);  
  
    void update(User u, String[] params);  
  
    void delete(User u);  
}
```

```
public class UserApplication {  
  
    private static Dao<User> userDao;  
  
    public static void main(String[] args) {  
        userDao = new UserDao();  
  
        User user1 = getUser(0);  
        System.out.println(user1);  
        userDao.update(user1, new String[]{"Jake", "jake@domain.com"});  
  
        User user2 = getUser(1);  
        userDao.delete(user2);  
        userDao.save(new User("Julie", "julie@domain.com"));  
  
        userDao.getAll().forEach(user -> System.out.println(user.getName()));  
    }  
}
```

Quelle: <https://www.baeldung.com/java-dao-pattern>
https://de.wikipedia.org/wiki/Data_Access_Object

Agenda

- Entwicklung der Webarchitektur
- Java-basierte Webarchitektur
- Das Model-View-Controller (MVC)-Muster
- Dependency Injection
- Das Spring Framework
- Ein Hello World-Projekt

Dependency Injection

```
public class UserApplication {  
  
    private static Dao<User> userDao;  
  
    public static void main(String[] args) {  
        userDao = new UserDao();  
  
        User user1 = getUser(0);  
        System.out.println(user1);  
        userDao.update(user1, new String[]{"Jake", "jake@domain.com"});  
  
        User user2 = getUser(1);  
        userDao.delete(user2);  
        userDao.save(new User("Julie", "julie@domain.com"));  
  
        userDao.getAll().forEach(user -> System.out.println(user.getName()));  
    }  
}
```

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects or “dependencies”. Simply put, this allows for loose coupling of components and moves the responsibility of managing components on the container.

Dependency Injection

```
public class Store {  
    private Item item;  
  
    public Store() {  
        item = new ItemImpl1();  
    }  
}
```

```
public class Store {  
    private Item item;  
  
    public Store(Item item) {  
        this.item = item;  
    }  
}
```

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects or “dependencies”. Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.

Inversion of Control (IoC)

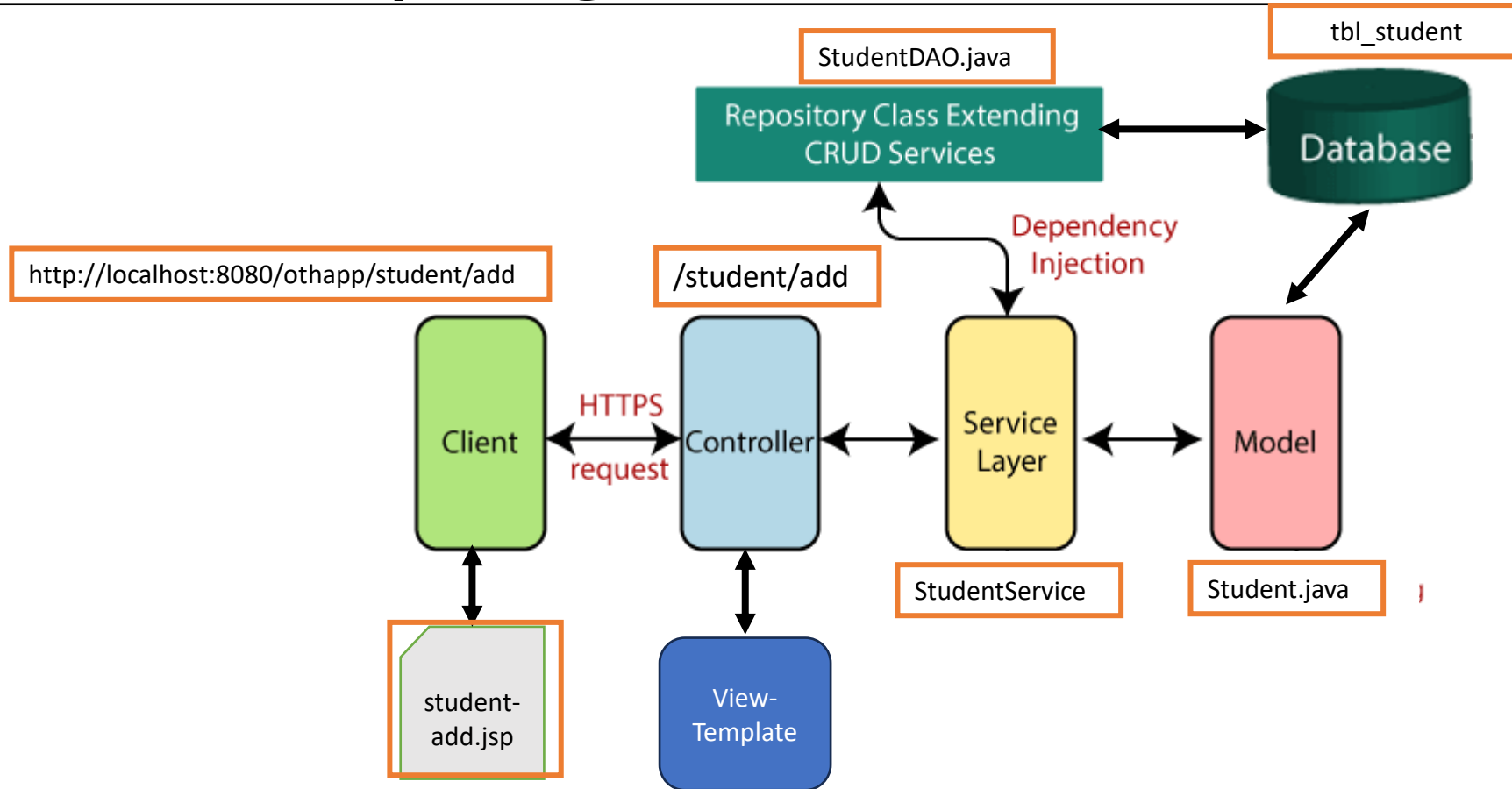
- Inversion of Control ist ein Prinzip in der Softwareentwicklung, das die Kontrolle über Objekte oder Teile eines Programms auf einen Container oder ein Framework überträgt.
- Der Spring-Container implementiert Dependency Injection- und Inversion of Control.

```
public class Store {  
    private Item item;  
  
    public Store(Item item) {  
        this.item = item;  
    }  
}
```

Agenda

- Entwicklung der Webarchitektur
- Java-basierte Webarchitektur
- Das Model-View-Controller (MVC)-Muster
- Dependency and Inversion of Control
- Das Spring Framework
- Ein Hello World-Projekt

Spring MVC Architecture



Spring MVC Architecture

- Einfache und robuste Anwendungen können implementiert werden
- Integration mit JSTL, JS, **Thymeleaf** und anderen View-Frameworks
- Datenkonvertierung
- Anmerkungen zur Unterstützung der Datenvalidierung
- Unterstützung für REST-APIs

Spring Annotations - @Component

@Component ist eine Annotation, die es Spring ermöglicht, unsere benutzerdefinierten Beans automatisch zu erkennen.

Ohne expliziten Code schreiben zu müssen, wird Spring Folgendes tun:

- unsere Anwendung nach Klassen, die mit @Component annotiert sind durchsuchen
- alle angegebenen Abhängigkeiten instanziiieren und einfügen
- Inject them, wo immer es nötig ist

```
package com.baeldung.component.scannedscope;  
@Component  
public class ScannedScopeExample {  
}
```

<https://www.baeldung.com/spring-component-annotation>

Spring Annotations - @Controller

Die Annotation @Controller gibt an, dass eine bestimmte Klasse die Rolle eines Controllers übernimmt.

Es kann nur auf Klassen angewendet werden. Es wird verwendet, um eine Klasse als Web-Request-Handler zu markieren.

```
@Controller
// Main class
public class DemoController {

    @RequestMapping("/hello")
    @ResponseBody

    // Method
    public String helloWorld()
    {
        return "Hello World!!!!";
    }
}
```

<https://www.geeksforgeeks.org/spring-controller-annotation-with-example/>

Spring Annotations - @RequestMapping

Simply put, the annotation is used to map web requests to Spring Controller methods.

```
@RequestMapping(value = "/ex/foos", method = POST)
@ResponseBody

public String postFoos() {

    return "Post some Foos";

}
```

1. **@GetMapping** - shortcut for `@RequestMapping(method = RequestMethod.GET)`
2. **@PostMapping** - shortcut for `@RequestMapping(method = RequestMethod.POST)`
3. **@PutMapping** - shortcut for `@RequestMapping(method = RequestMethod.PUT)`
4. **@DeleteMapping** - shortcut for `@RequestMapping(method = RequestMethod.DELETE)`
5. **@PatchMapping** - shortcut for `@RequestMapping(method = RequestMethod.PATCH)`

<https://www.baeldung.com/spring-requestmapping>

Spring Boot

- Ziel ist es, bei der Konfiguration von MVC-, Persistenz-, Vorlagen- und Sicherheitsmodulen usw. zu helfen

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.10.RELEASE</version>
</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-jdbc</artifactId>
  </dependency>
</dependencies>

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

Spring Boot

- Ziel ist es, bei der Konfiguration von MVC-, Persistenz-, Vorlagen- und Sicherheitsmodulen usw. zu helfen

```
@SpringBootApplication
public class DemoMvcApplication {

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

Agenda

- Entwicklung der Webarchitektur
- Java-basierte Webarchitektur
- Das Model-View-Controller (MVC)-Muster
- Dependency and Inversion of Control
- Das Spring Framework
- Ein Hello World-Projekt

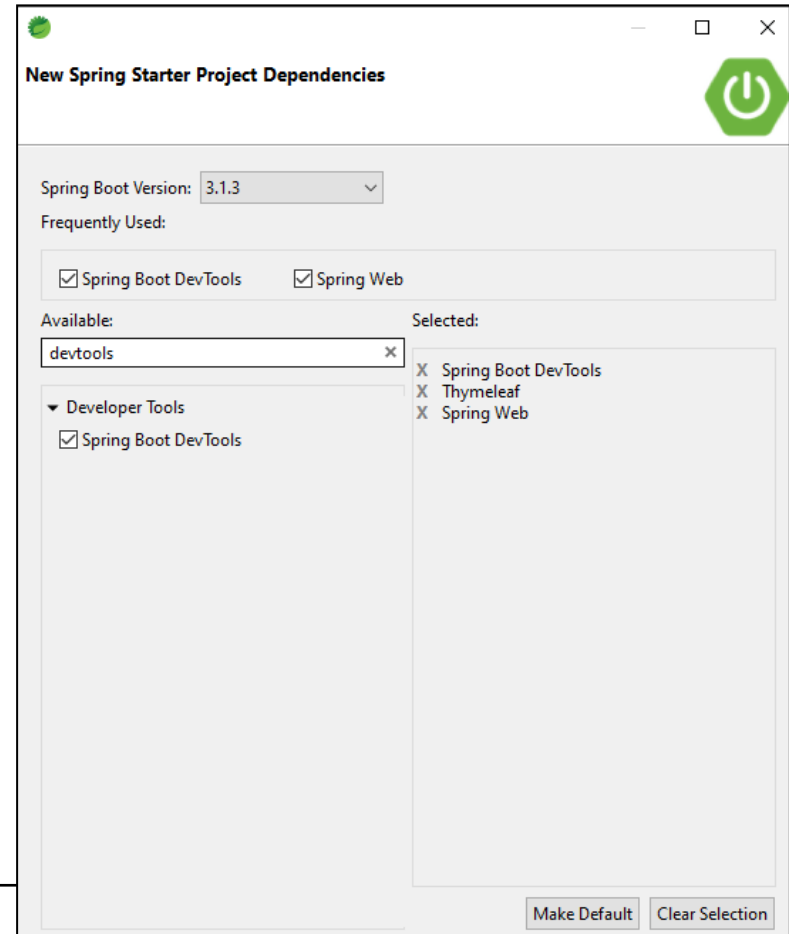
Creating the "HelloWorld" Project

Creating the Project

Folgen Sie die:

<https://www.javatpoint.com/creating-spring-boot-project-using-sts>

Ausgewählt Modulen: Spring Web, Spring Boot DevTools, Thymeleaf

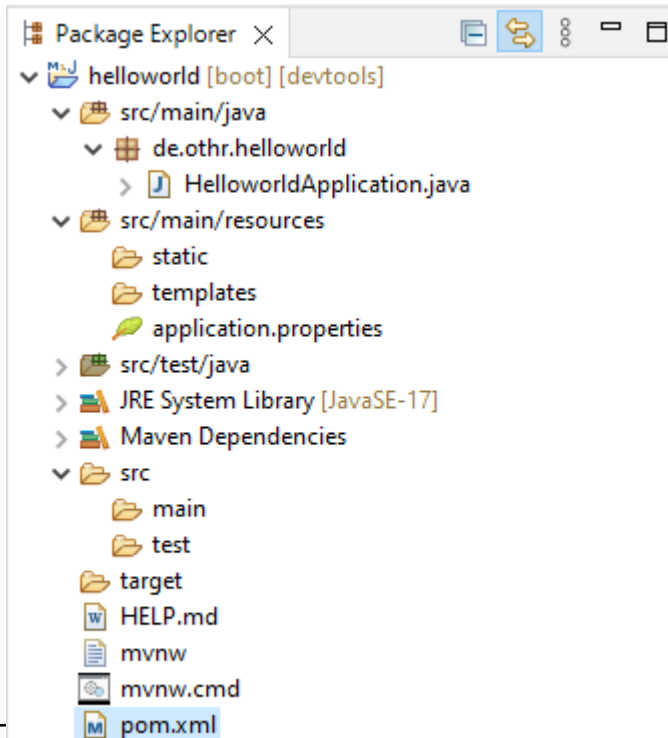


Creating the Project

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
<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>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Creating the Project

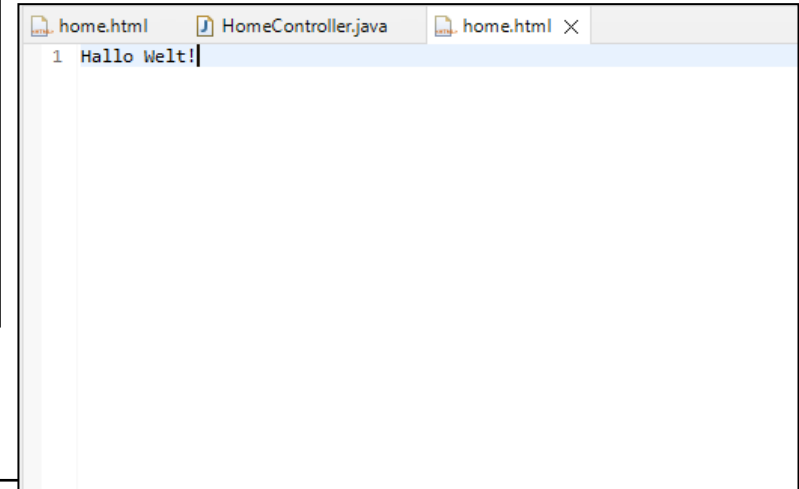
- Typical Structure:



Creating the Code...

- HomeController.java in de.othr.helloworld
- home.html in “templates”

```
1 package de.othr.helloworld.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RequestMethod;
6
7 @Controller
8 public class HomeController {
9
10     @RequestMapping(method = RequestMethod.GET, value = "/")
11     public String getHome () {
12
13         return "home";
14     }
15 }
16 }
```



Adding CSS und JS

Search at <https://www.webjars.org/> for :

- JQuery
- Bootstrap
- jquery-mask-plugin
- open-iconic

Add the dependency:

```
<dependency>
<groupId>org.webjars</groupId>
<artifactId>webjars-locator</artifactId>
<version>0.30</version>
</dependency>
```

```
<dependency>
<groupId>org.webjars</groupId>
<artifactId>webjars-locator</artifactId>
<version>0.30</version>
</dependency>
<dependency>
<groupId>org.webjars</groupId>
<artifactId>bootstrap</artifactId>
<version>4.1.0</version>
</dependency>
<dependency>
<groupId>org.webjars</groupId>
<artifactId>jquery</artifactId>
<version>3.7.1</version>
</dependency>
<dependency>
<groupId>org.webjars.npm</groupId>
<artifactId>jquery-mask-plugin</artifactId>
<version>1.14.16</version>
</dependency>
<dependency>
<groupId>org.webjars.bowergithub.iconic</groupId>
<artifactId>open-iconic</artifactId>
<version>1.1.1</version>
</dependency>
```

References

- <http://www.oreillynet.com/go/web2>
- <https://www.forbes.com/sites/forbestechcouncil/2020/01/06/what-is-web-3-0/?sh=634cbc4a58df>
- https://blogs.gartner.com/anthony_bradley/2009/04/28/dont-be-fooled-web-3-0-doesnt-exist/
- <https://www.w3schools.com/tags/default.asp>
- <http://www.inf.fu-berlin.de/lehre/SS03/aws/mvc.pdf>
- <https://spring.io/projects/spring-boot>
- <https://spring.io/tools>
- <https://docs.spring.io/spring-framework/docs/current/javadoc-api/>
- <https://www.baeldung.com/spring-mvc-model-model-map-model-view>
- <https://www.baeldung.com/spring-mvc-and-the-modelattribute-annotation>