Spring

Konrad Raue, Oliver Scholz

21. Januar 2020

Gliederung

1. Framework vs Library
2. Spring
3. Nächste Woche

Framework vs Library

Framework vs Library

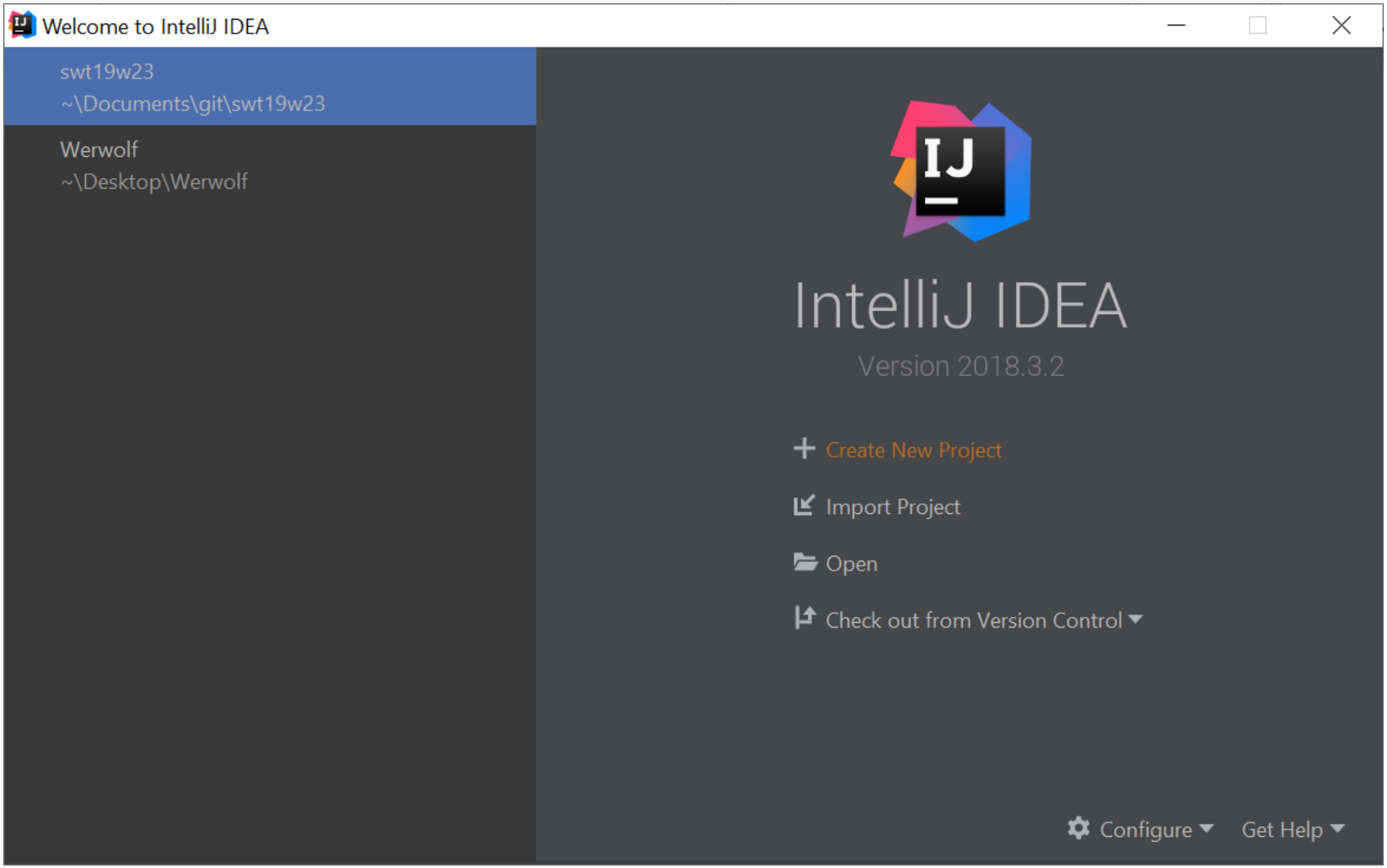
- Frameworks verwenden deinen Code
- dein Code verwendet Bibliotheken
- Frameworks sind (fast) eigenständig laufende Programme, in deren Lücken nur noch dein Code eingebettet wird
- Bibliotheken sind nur Klassendefinitionen um häufig verwendete Sachverhalte nicht immer neu zu implementieren

Spring

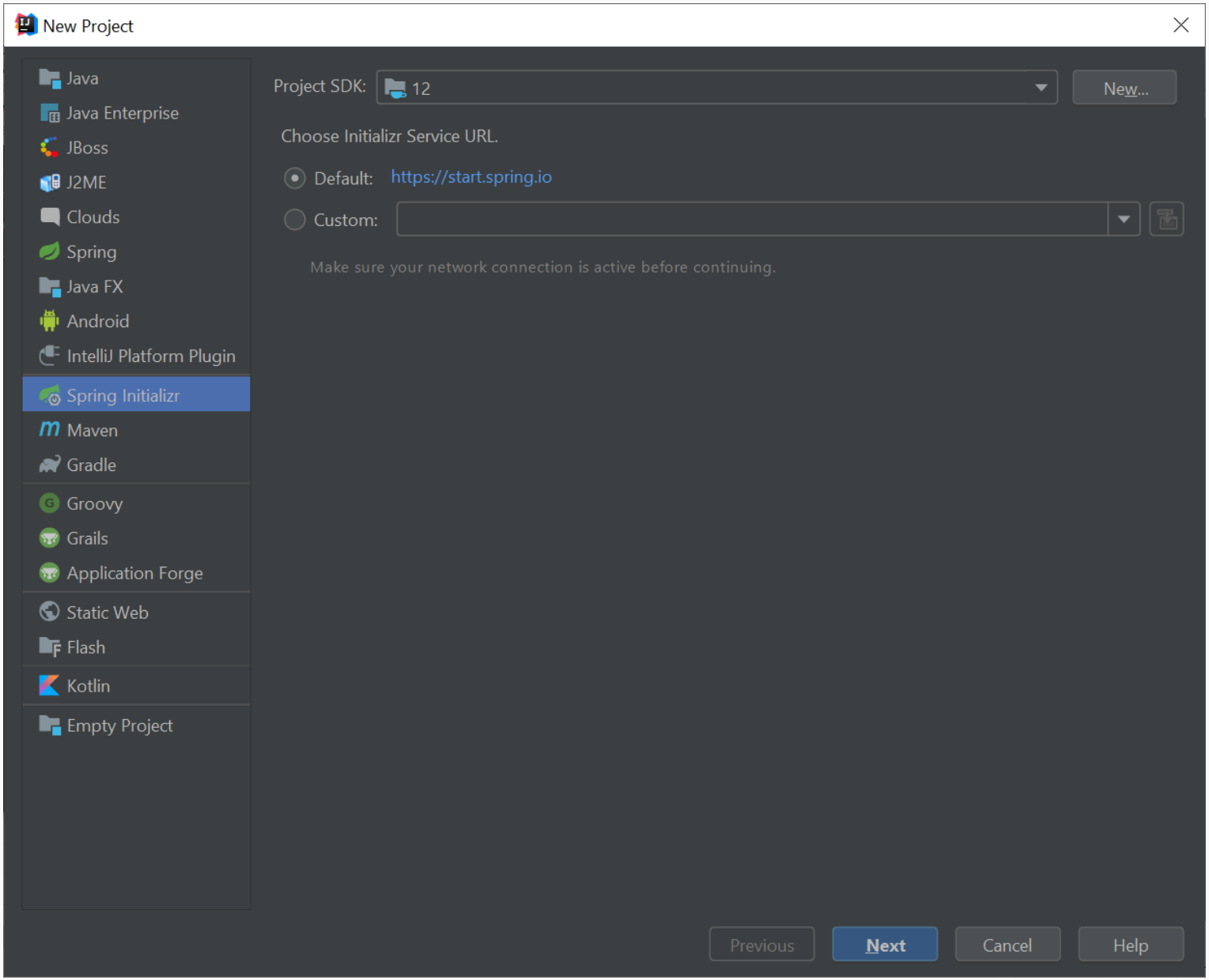
Spring

- Framework für Webapplikationen mit Java
- MVC (Model, View Controller)
- nutzt Maven

Projekt anlegen



Projekt anlegen



Projekt anlegen

New Project

Project Metadata

Group: com.example

Artifact: demo

Type: Maven Project (Generate a Maven based project archive.)

Language: Java

Packaging: Jar

Java Version: 8

Version: 0.0.1-SNAPSHOT

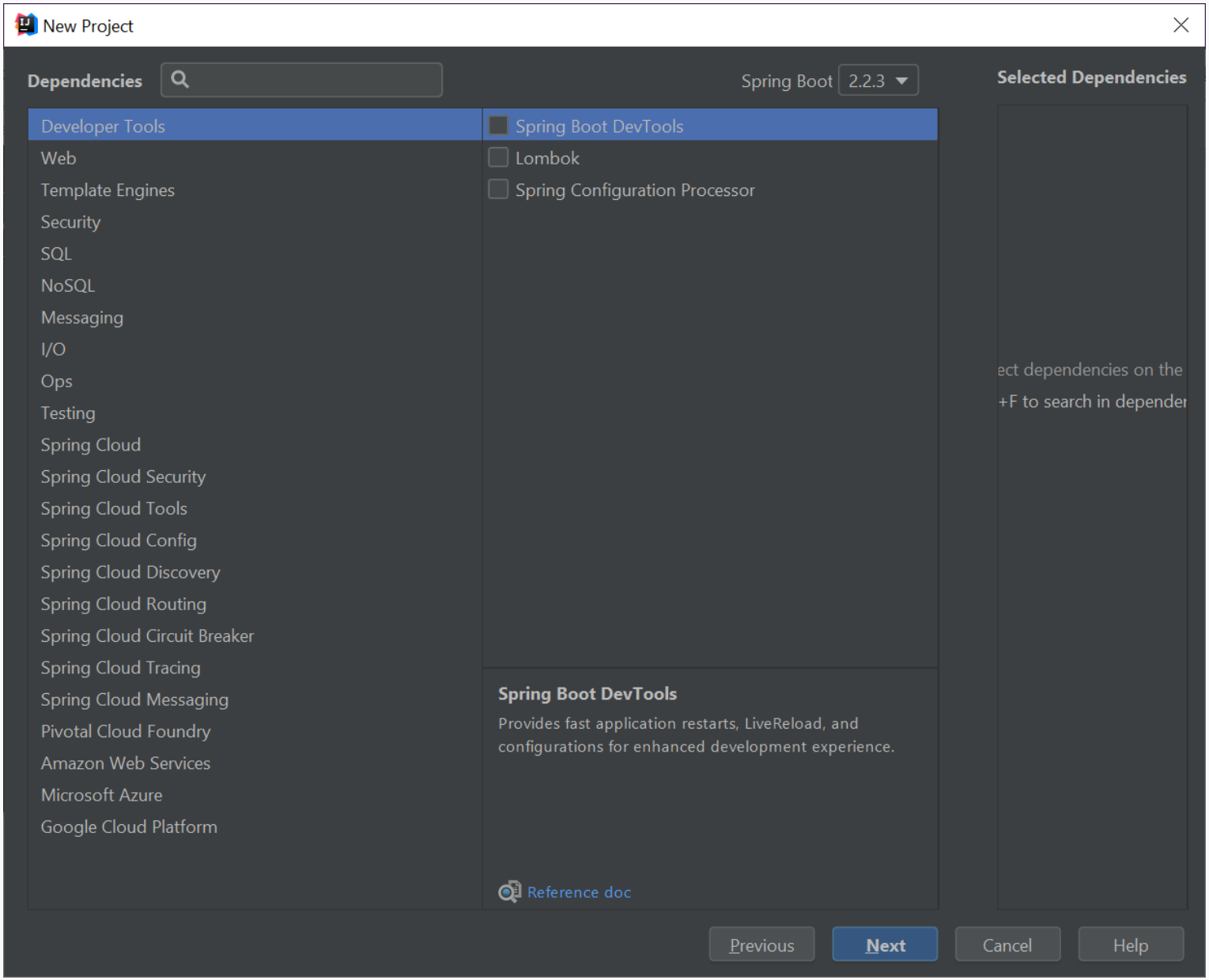
Name: demo

Description: Demo project for Spring Boot

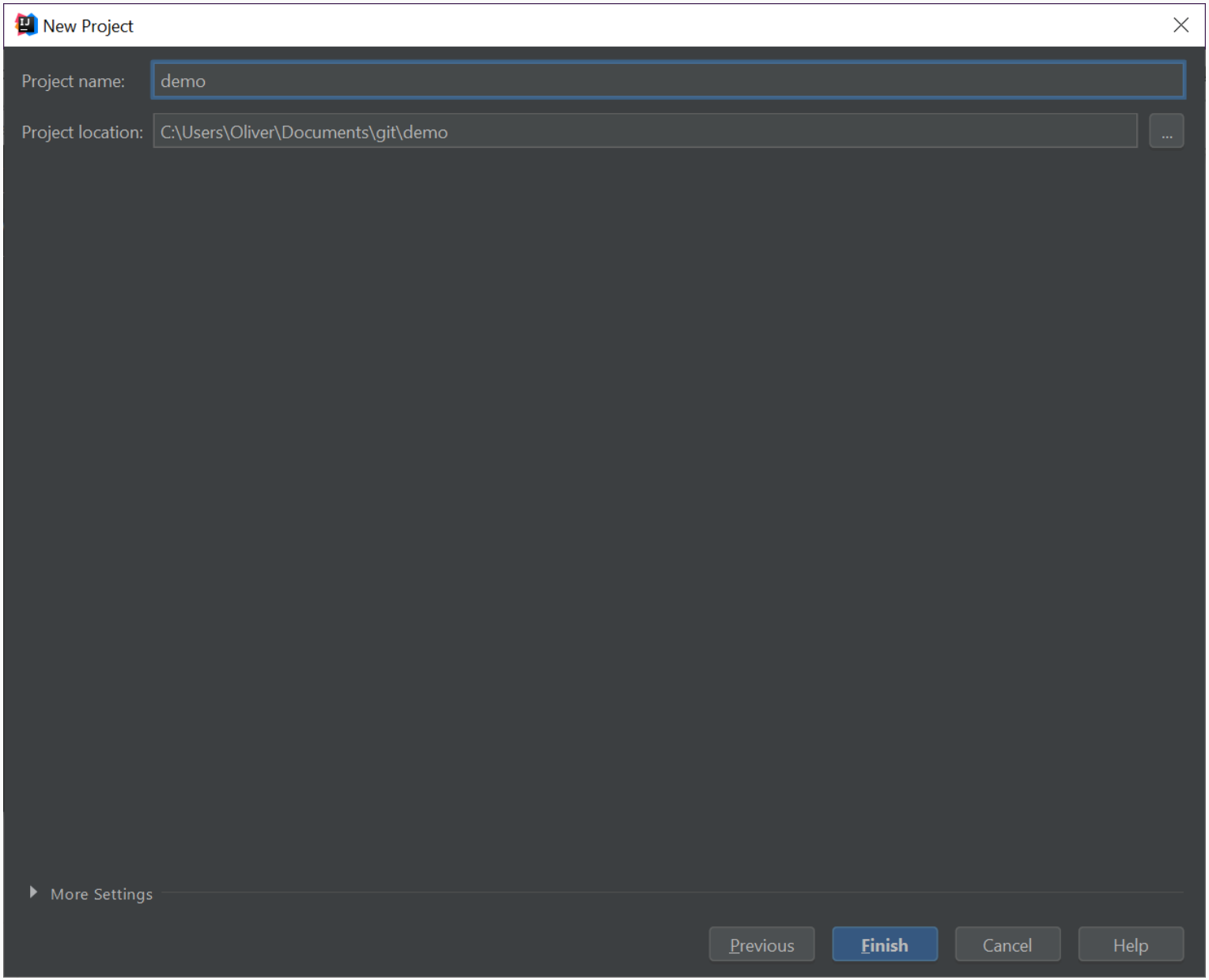
Package: com.example.demo

Previous | **Next** | Cancel | Help

Projekt anlegen



Projekt anlegen



erste Anwendung

demo [C:\Users\Oliver\Documents\git\demo] - ...\src\main\java\com\example\demo\DemoApplication.java [demo] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

demo 〉 src 〉 main 〉 java 〉 com 〉 example 〉 demo 〉 DemoApplication

Project

- demo C:\Users\Oliver\Documents\git\demo
  - .idea
  - .mvn
  - src
    - main
      - java
        - com.example.demo
          - DemoApplication
      - resources
        - application.properties
    - test
  - .gitignore
  - demo.iml
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
- External Libraries
- Scratches and Consoles

DemoApplication.java

```
package com.example.demo;

import ...

@SpringBootApplication
public class DemoApplication {

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

}
```

DemoApplication

Terminal Spring 6: TODO Event Log

12:1 LF UTF-8 4 spaces

erster Controller

WelcomeController.java:

```java
@Controller
public class WelcomeController{
    @GetMapping("/")
    public String index(){
        return "hallo";
    }
}
```

hallo.html:

```html
<!DOCTYPE html>
<html lang="de" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="UTF-8">
        <title>Hallo</title>
    </head>
    <body>
        <h1>Hallo</h1>
    </body>
</html>
```

@RestController für Daten (Ressourcen) statt richtiges HTML.

erster Controller

pom.xml (Auszug aus dependencies):

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Controller erweitern

```
@Controller
public class WelcomeController{
    @GetMapping("/{name}")
    public String index(@PathVariable String name, Model model){
        model.addAttribute("name", name);
        return "hallo";
    }
}
```

```
<!DOCTYPE html>
<html lang="de" xmlns="http://www.w3.org/1999/xhtml"
        xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>Hallo</title>
    </head>
    <body>
        <h1 th:text="${'Hallo '+name}"></h1>
    </body>
</html>
```

Controller erweitern

Alternative:

```
@Controller
public class WelcomeController{
    @GetMapping("/")
    public String index(@RequestParam Optional<String> name, Model
     model){
        model.addAttribute("name", name.orElse(""));
        return "hallo";
    }
}
```

Controller erweitern

```java
@Controller
public class WelcomeController{
    @GetMapping("/{name}")
    public String index(@PathVariable String name, Model model){
        model.addAttribute("name", name);
        return "hallo";
    }

    @PostMapping("/name")
    public String name(Name name){
        return "redirect:/" + name.getName();
    }
}
```

```java
public class Name {
    private String name;

    public Name(String name){
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

Controller erweitern

```
<!DOCTYPE html>
<html lang="de" xmlns="http://www.w3.org/1999/xhtml"
        xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>Hallo</title>
    </head>
    <body>
        <h1 th:text="${'Hallo '+name}"></h1>
        <form method="post" role="form" id="form"
                th:action="@{/name}" th:object="${name}">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" />
        </form>
    </body>
</html>
```

Controller erweitern

```java
@Controller
public class WelcomeController{
    private ArrayList<Name> names;

    public WelcomeController(){
        names = new ArrayList<>();
    }

    @GetMapping("/{name}")
    public String index(@PathVariable String name, Model model){
        model.addAttribute("name", name);
        return "hallo";
    }

    @PostMapping("/name")
    public String name(Name name){
        names.add(name);
        return "redirect:/" + name.getName();
    }
}
```

Controller erweitern

```java
@Component
public class Name {
    private String name;

    public Name(String name){
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

Wenn die Klasse Name als Datenbankeintrag gespeichert wird: @Entity statt @Component und @Id bei einem einzigartigen Wert.

erster Service

```java
@Service
public class NameService {
    private ArrayList<Name> names;

    public NameService(){
        names = new ArrayList<>();
    }

    public void addName(Name name){
        names.add(name);
    }
}
```

erster Service

```java
@Component
public class Name {
    private String name;

    @SuppressWarnings("unused")
    private Name(){}

    public Name(String name){
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

erster Service

```java
@Controller
public class WelcomeController {
    private final NameService nameService;

    public WelcomeController(NameService nameService){
        this.nameService = nameService;
    }

    @GetMapping("/{name}")
    public String index(@PathVariable String name, Model model){
        model.addAttribute("name", name);
        return "hallo";
    }

    @PostMapping("/name")
    public String name(Name name){
        nameService.addName(name);
        return "redirect:/" + name.getName();
    }
}
```

erste Datenbank

```java
public interface NameRepository
        extends CrudRepository<Name, String> {
    @Override
    ArrayList<Name> findAll();
}
```

```java
@Entity
public class Name {
    private @Id String name;

    @SuppressWarnings("unused")
    private Name(){}

    public Name(String name){
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

erste Datenbank

```
@Service
@Transactional
public class NameService {
    private static NameRepository nameRepository;

    public NameService(NameRepository nameRepository){
        NameService.nameRepository = nameRepository;
    }

    public void addName(Name name){
        nameRepository.save(name);
    }
}
```

erste Datenbank

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<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>
    <exclusions>
        <exclusion>
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>5.2.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-commons</artifactId>
    <version>2.2.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

erste Datenbank

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>5.2.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-jpa</artifactId>
    <version>2.2.0.RELEASE</version>
</dependency>
```

erster Test

```java
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
@SpringBootTest
@Transactional
@AutoConfigureMockMvc
class DemoApplicationTests {
    @Autowired MockMvc mockMvc;
    @Autowired NameService nameService;

    @Test
    void contextLoads() throws Exception{
        mockMvc.perform(get("/"))
            .andExpect(status().isOk())
            .andExpect(model().attribute("name", ""))
            .andExpect(content().string(containsString("Hallo")));

        int before = nameService.findAll().size();
        mockMvc.perform(post("/name").param("name", "Oliver"));
        assertEquals(before + 1, nameService.findAll().size());
    }
}
```

Nächste Woche

Nächste Woche

- Übungsstunde