Hochschule **Rosenheim**
University of Applied Sciences

Verteilte Verarbeitung

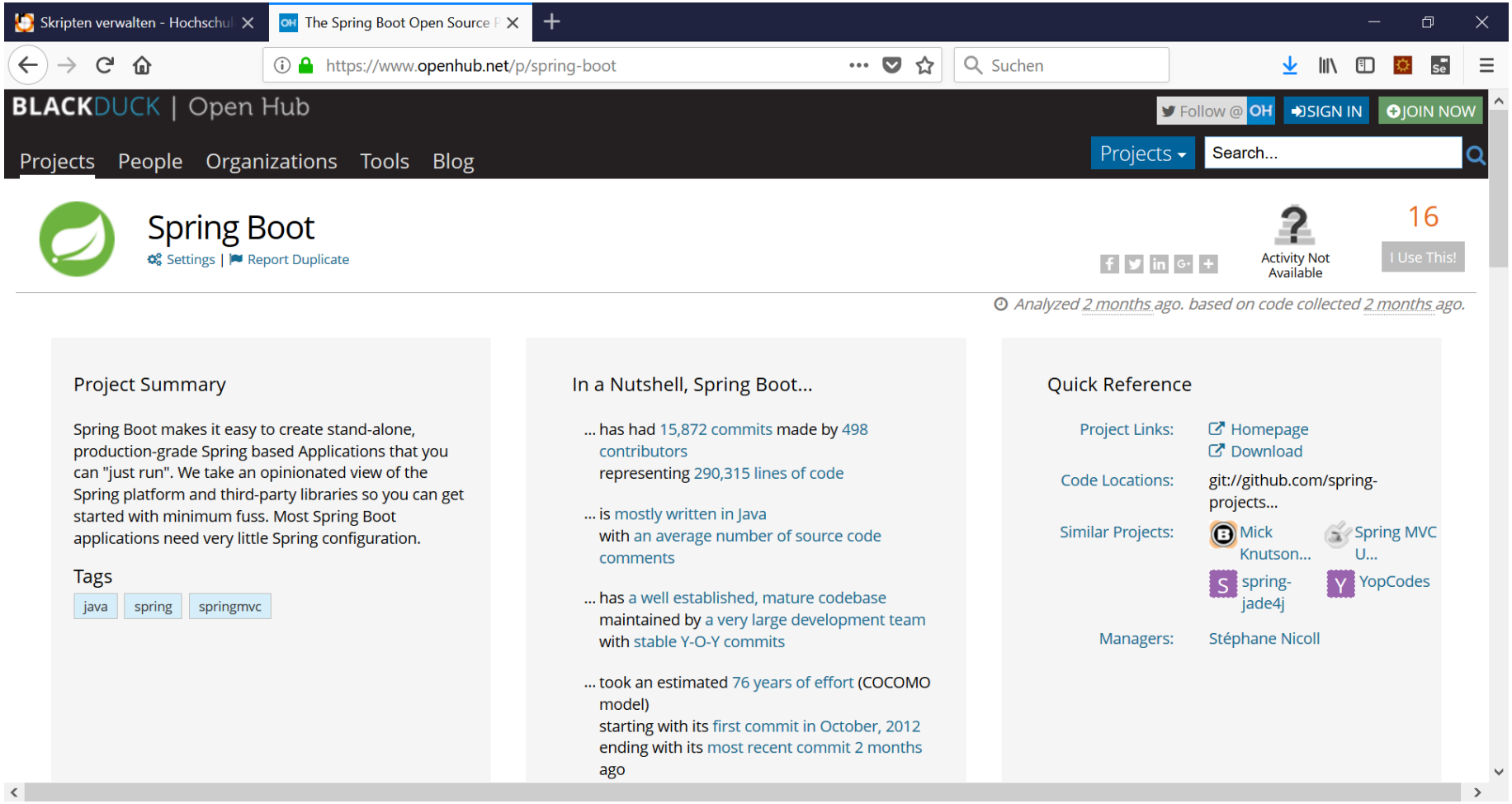
Kapitel 12.2

REST mit SpringBoot

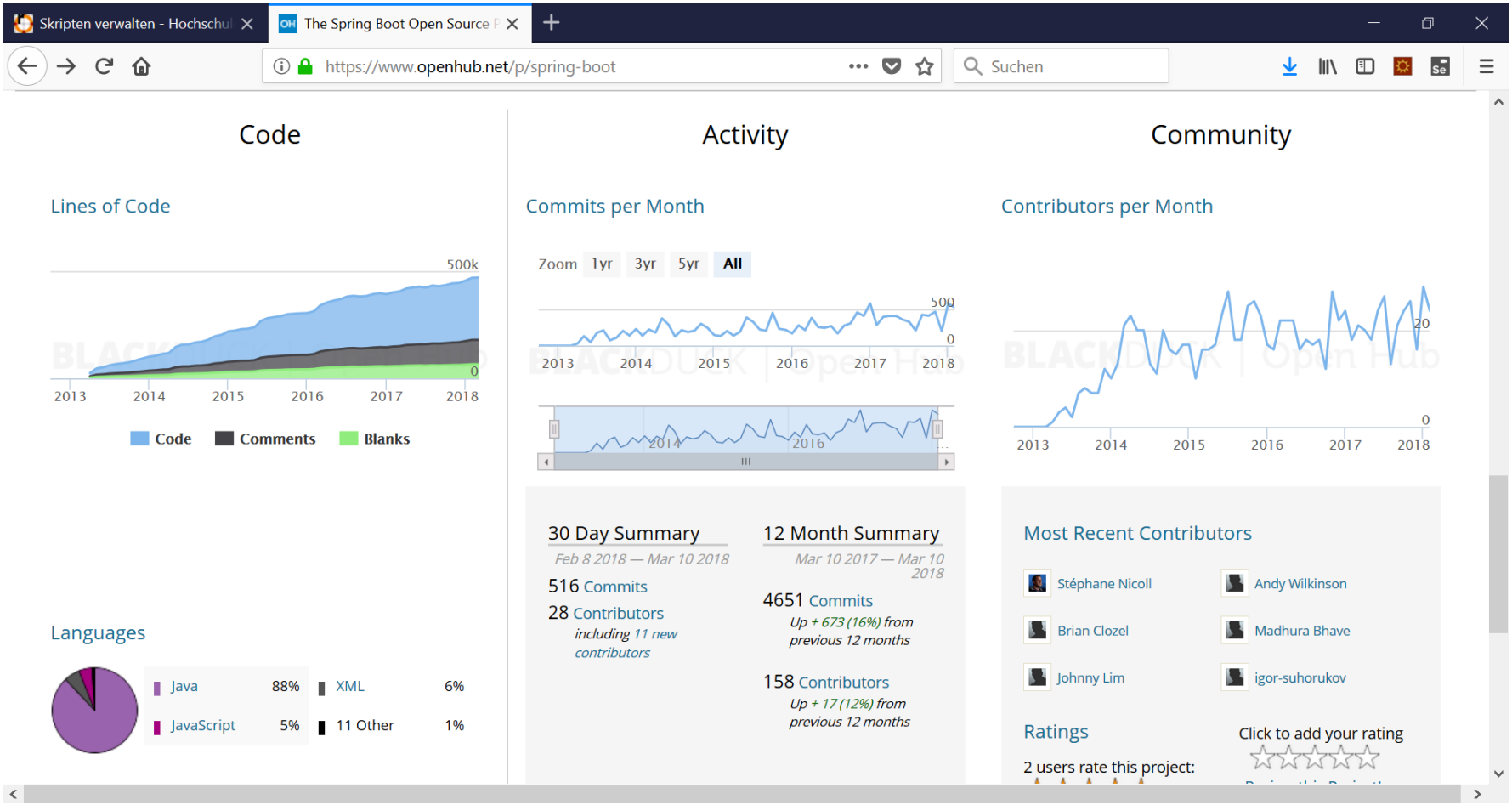
Spring Boot

Beispiel in unserem Repository unter benekengerd

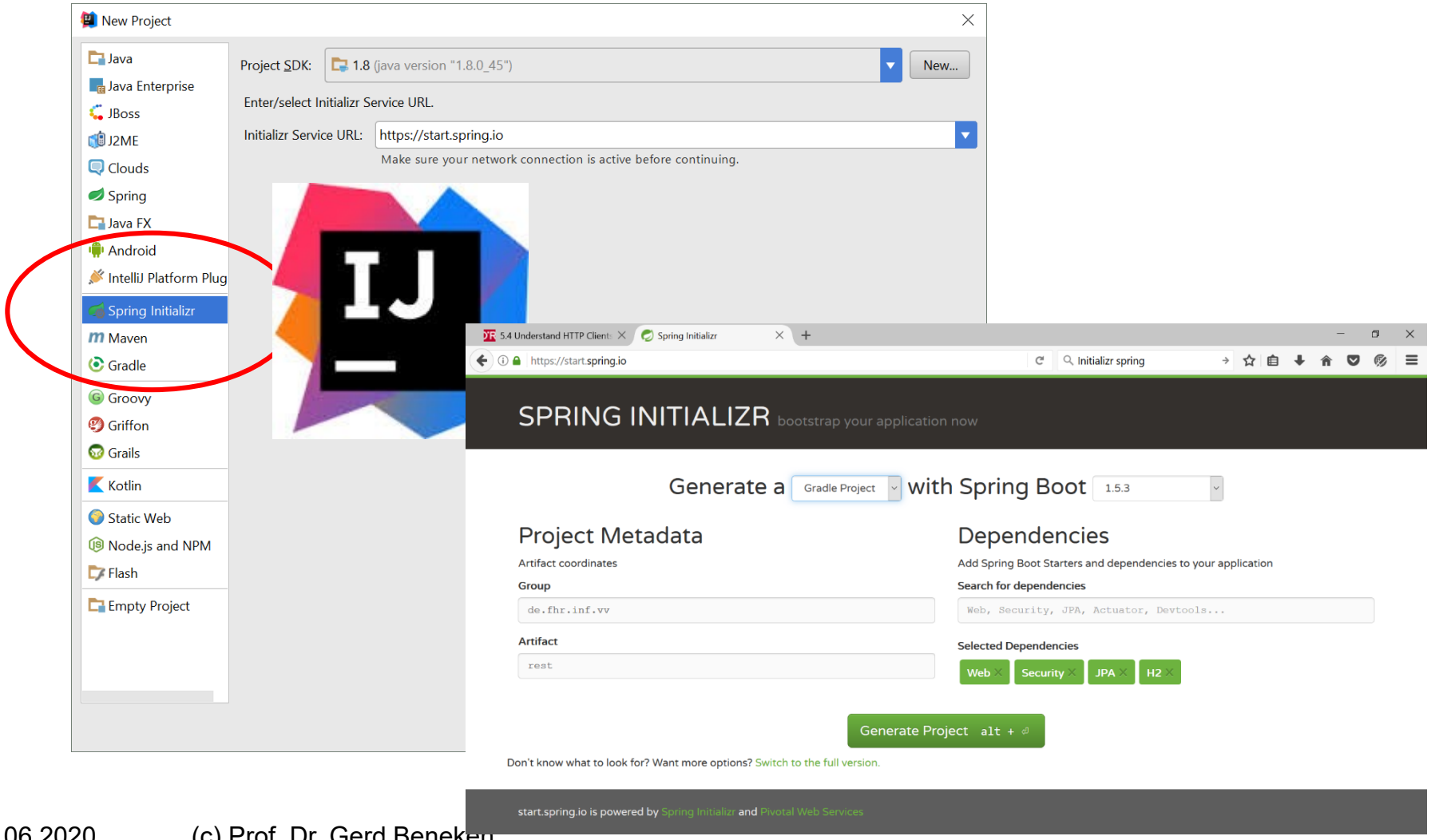
Was ist SpringBoot?



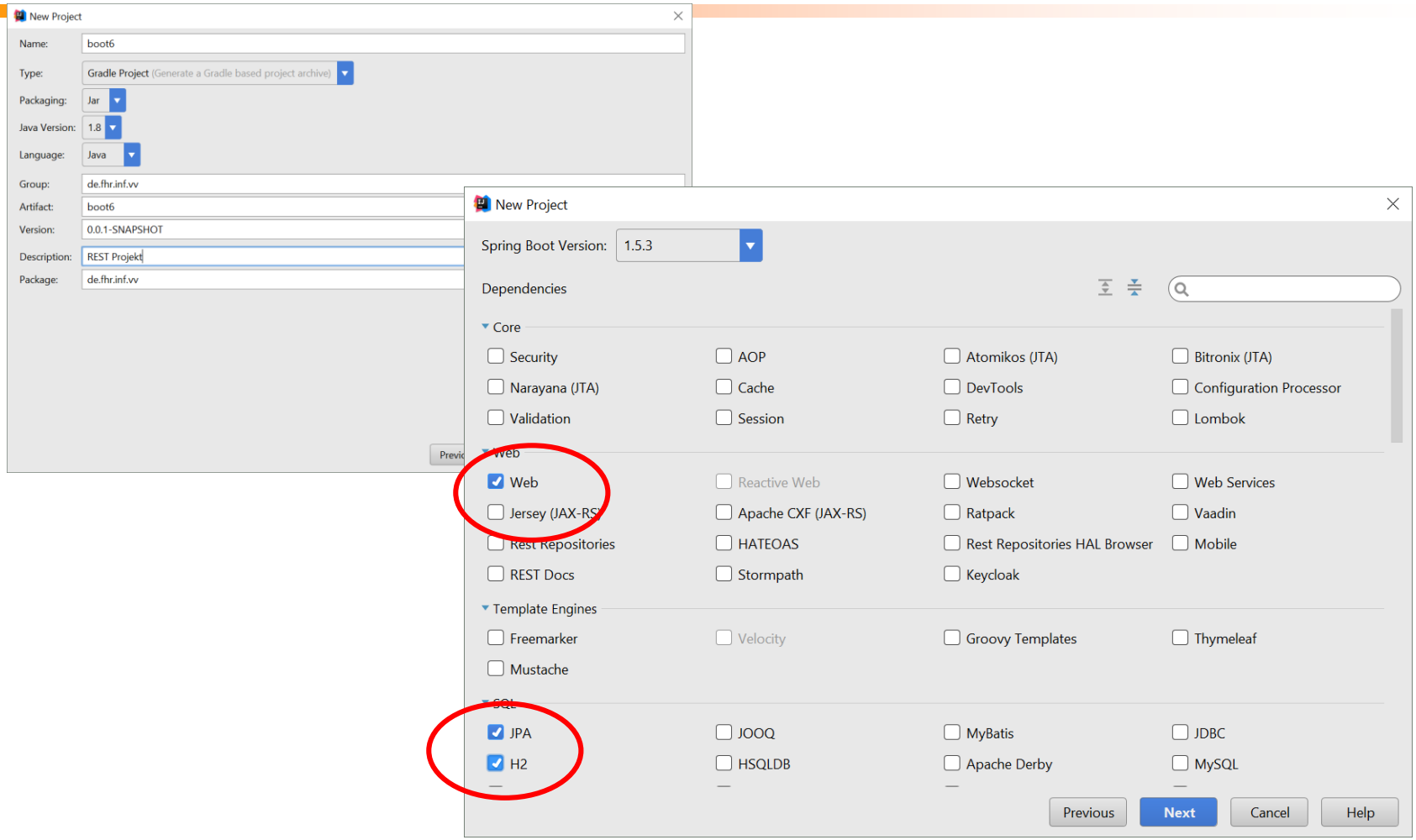
Einschätzung nach Openhub.net



Spring Boot und Sprint MVC
Bitte verwenden Sie Initializr im Projekt



IntelliJ Initializr



Generierte Gradle Build Datei

```
plugins {
    id 'org.springframework.boot' version '2.3.0.RELEASE'
    id 'io.spring.dependency-management' version '1.0.9.RELEASE'
    id 'java'
}

group = 'de.thro.inf'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    runtimeOnly 'com.h2database:h2'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}

test {
    useJUnitPlatform()
}
```

Mapping von HTTP-Requests auf Methoden

- **`@RequestMapping`**: Methode wird an einen HTTP-Request "angeschlossen"
  - **`Value`**: URL des Requests, ggf. mit Parametern, z.B.
    `value = "/kunden/{id}` (Requestparameter: id)
  - **`Method:`** HTTP-Verb (also GET, PUT, POST, DELETE), z.B.
    `method= RequestMethod.GET`
  - **`Produces:`** Welche Formate kann die Methode erzeugen? (JSON, ATOM, XML, …), Rückgabewert der Methode korrespondiert dazu z.B.
    `produces = MediaType.APPLICATION_JSON_VALUE`
  - **`Consumes:`** Welche Formate kann die Methode konsumieren? (JSON, ATOM, XML, …), ein Parameter der Methode **`@RequestBody`** korrespondiert dazu z.B.
    `consumes = MediaType.APPLICATION_JSON_VALUE`

Parameter und Rückgabewerte von REST-Methoden

```
@RequestMapping(
        value = "customers/{number}",
        method = {RequestMethod.PUT, RequestMethod.PATCH},
        produces = MediaType.APPLICATION_JSON_VALUE,
        consumes = MediaType.APPLICATION_JSON_VALUE
)
public ResponseEntity<?> modifyKunde(
      @PathVariable("number") String number,
      @RequestBody Customer customer) {

    Customer result = …;
    if (result == null) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    // …
    return new ResponseEntity<>(result, HttpStatus.OK);
}
```

Parameter:
Pfad (Teil der URI),
Query-Parameter,
Body des HTTP-Requests

Rückgabewert:
Daten + Statuscode

RESTful WebServices mit Spring Boot

/kunden	
GET	*alle Kunden auflisten*
PUT	unused
POST	*neuen Kunden anlegen*
DELETE	unused

```
@RestController
public class KundeService {


    @RequestMapping(value = "/kunden",
                    method= RequestMethod.GET,
                    produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity< List<Kunde>> findAll() { … }

    @RequestMapping( value = "/kunden",
                  method = RequestMethod.POST,
                  consumes = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Void>  neuerKunde(@RequestBody Kunde k,
                                  UriComponentsBuilder ucBuilder){…}

}
```

Im HTTP-Header die URI der neuen Ressource

Besonderheit des Post-Requests

```
@RequestMapping(
        value = "customers",
        method = RequestMethod.POST,
        produces = MediaType.APPLICATION_JSON_VALUE,
        consumes = MediaType.APPLICATION_JSON_VALUE

)
public ResponseEntity<?> createCustomer(
         @RequestBody Customer customer,
         UriComponentsBuilder ucBuilder) {
    // …
    HttpHeaders headers = new HttpHeaders();
    headers.setLocation(ucBuilder
            .path("api/v1/customers/{id}")
            .buildAndExpand(customer.getNumber()).toUri());
    return
       new ResponseEntity<String>(headers, HttpStatus.CREATED);
}
```

- Inhalt des HTTP-Headers:

```
Location: http://localhost:8080/api/v1/customers/7
```

RESTful WebServices mit Spring Boot

/kunden/{id}	
GET	*Details zum Kunden*
PUT	*Kunden ändern*
POST	unused
DELETE	*Kunden löschen*

```
@RequestMapping(value = "/kunden/{id}",
                method= RequestMethod.GET,
                produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Kunde> findById(@PathVariable String id) {…}


@RequestMapping( value = "/kunden/{id}",
                method = RequestMethod.PUT,
                consumes = MediaType.APPLICATION_JSON_VALUE,
                produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Kunde> aendereKunde(
            @PathVariable String id, @RequestBody Kunde kNeu) { … }


@RequestMapping( value = "/kunden/{id}",
                  method = RequestMethod.DELETE,
                  produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Kunde> loescheKunde(@PathVariable String id) {…}

}
```

Spring Boot Hauptprogramm

```
@SpringBootApplication
public class BootApplication {
    // Initialisierung fehlt noch
    public static void main(String[] args) {
        SpringApplication.run(BootApplication.class,
            args);
    }
}
```