



Verteilte Verarbeitung

Kapitel 12.2

REST mit SpringBoot

Spring Boot



Beispiel in unserem Repository unter `benekengerd`

Was ist SpringBoot?

Project Summary

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

Tags

java spring springmvc

In a Nutshell, Spring Boot...

- ... has had **15,872 commits** made by **498 contributors** representing **290,315 lines of code**
- ... is **mostly written in Java** with an average number of source code comments
- ... has a well established, mature codebase maintained by a very large development team with **stable Y-O-Y commits**
- ... took an estimated **76 years of effort** (COCOMO model) starting with its **first commit in October, 2012** ending with its **most recent commit 2 months ago**

Quick Reference

Project Links: [Homepage](#) [Download](#)

Code Locations: [git://github.com/spring-projects...](https://github.com/spring-projects...)

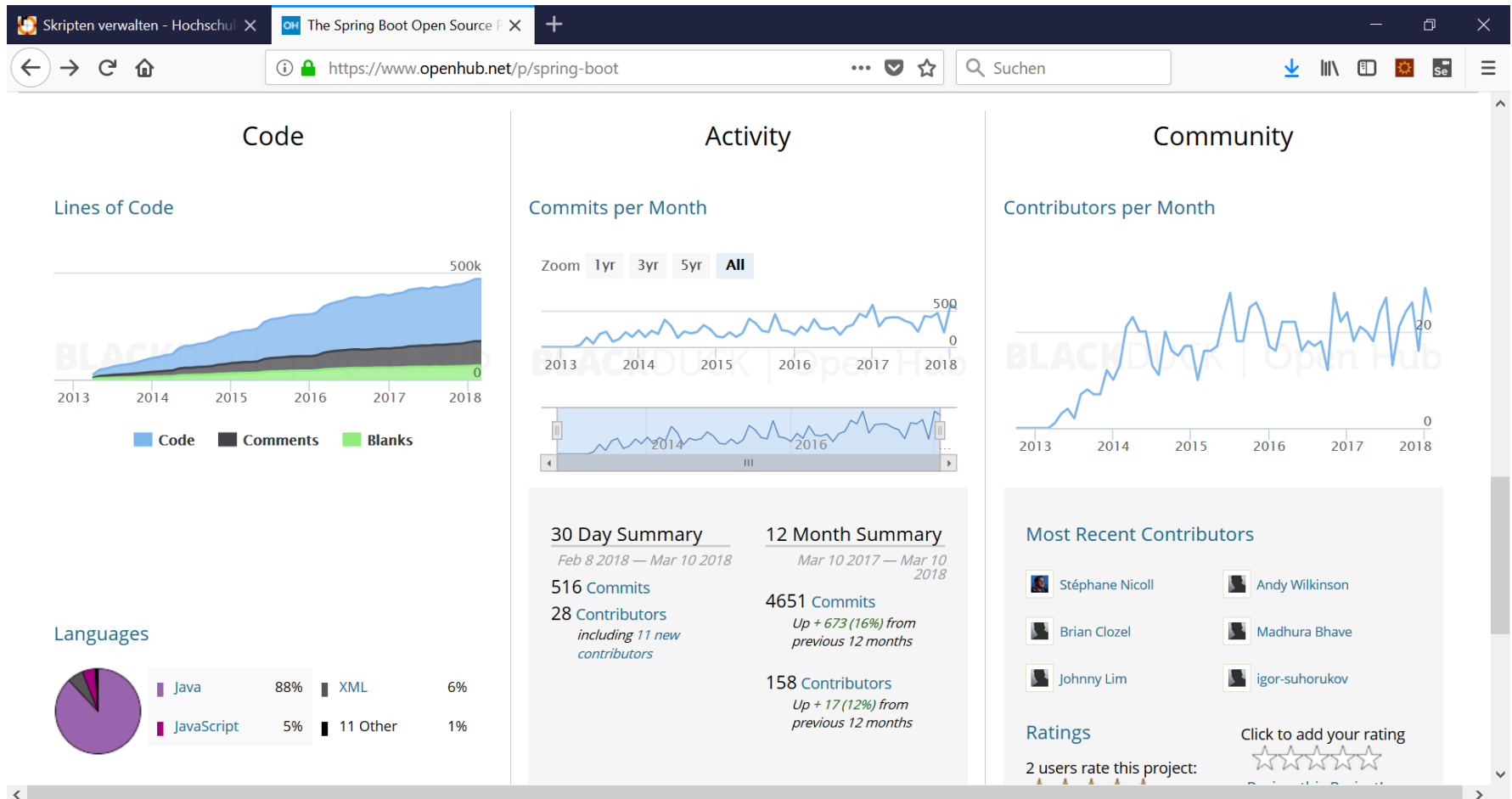
Similar Projects: [Mick Knutson...](#) [Spring MVC U...](#) [spring-jade4j](#) [YopCodes](#)

Managers: Stéphane Nicoll

Activity: ? Activity Not Available **16** I Use This!

Analyzed 2 months ago. based on code collected 2 months ago.

Einschätzung nach Openhub.net



Spring Boot und Sprint MVC

Bitte verwenden Sie Initializr im Projekt

The image shows the IntelliJ IDEA 'New Project' dialog box on the left. The 'Spring Initializr' option is highlighted with a red circle. In the background, the Spring Initializr website is visible in a browser window. The website shows the 'Generate a' dropdown set to 'Gradle Project' and 'with Spring Boot' set to '1.5.3'. Under 'Project Metadata', the 'Artifact' field contains 'rest'. Under 'Dependencies', the 'Selected Dependencies' section shows 'Web', 'Security', 'JPA', and 'H2' as selected options. A 'Generate Project' button is at the bottom right of the website interface.

IntelliJ Initializr

New Project

Name: boot6

Type: Gradle Project (Generate a Gradle based project archive)

Packaging: Jar

Java Version: 1.8

Language: Java

Group: de.fhr.inf.vv

Artifact: boot6

Version: 0.0.1-SNAPSHOT

Description: REST Projekt

Package: de.fhr.inf.vv

New Project

Spring Boot Version: 1.5.3

Dependencies

Core

- ☐ Security
- ☐ Narayana (JTA)
- ☐ Validation
- ☐ AOP
- ☐ Cache
- ☐ Session
- ☐ Atomikos (JTA)
- ☐ DevTools
- ☐ Retry
- ☐ Bitronix (JTA)
- ☐ Configuration Processor
- ☐ Lombok

Web

- ☒ Web
- ☐ Jersey (JAX-RS)
- ☐ Rest Repositories
- ☐ REST Docs
- ☐ Reactive Web
- ☐ Apache CXF (JAX-RS)
- ☐ HATEOAS
- ☐ Stormpath
- ☐ Websocket
- ☐ Ratpack
- ☐ Rest Repositories HAL Browser
- ☐ Keycloak
- ☐ Web Services
- ☐ Vaadin
- ☐ Mobile

Template Engines

- ☐ Freemarker
- ☐ Mustache
- ☐ Velocity
- ☐ Groovy Templates
- ☐ Thymeleaf

Database

- ☒ JPA
- ☒ H2
- ☐ JOOQ
- ☐ HSQLDB
- ☐ MyBatis
- ☐ Apache Derby
- ☐ JDBC
- ☐ MySQL

Previous Next Cancel Help

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) {
```

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

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

```
// ...
```

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

Rückgabewert:
Daten + Statuscode

RESTful WebServices mit Spring Boot

/kunden	
GET	<i>alle Kunden auflisten</i>
PUT	unused
POST	<i>neuen Kunden anlegen</i>
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	<i>Details zum Kunden</i>
PUT	<i>Kunden ändern</i>
POST	unused
DELETE	<i>Kunden löschen</i>

```
@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);
    }
}
```