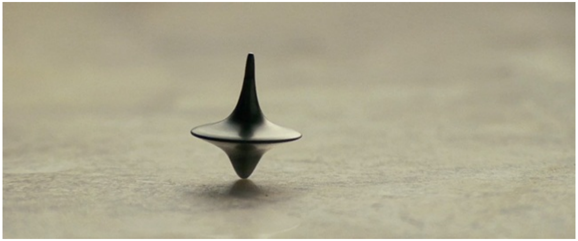
Spring Boot

JUGHRO-Treffen 10. Dezember 2015

Roland Ewald

Limbus Medical Technologies GmbH

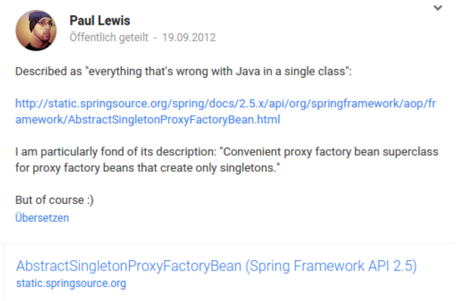
Spring Boot Inception



"[...] **Spring is now so complex that it has it's own framework, Spring Boot***. A framework for a framework. We are in Framework Inception, a film about Leonardo Di Caprio trying to find his long lost java code by going deeper and deeper through layers of XML and annotations before eventually giving up on life."*

Sam Atkinson, *Why I hate Spring*, `http://samatkinson.com/why-i-hate-spring/`

:-)



Paul Lewis
Öffentlich geteilt - 19.09.2012

Described as "everything that's wrong with Java in a single class":

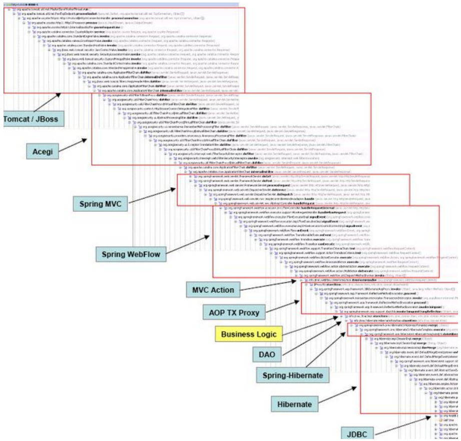
http://static.springsource.org/spring/docs/2.5.x/api/org/springframework/aop/framework/AbstractSingletonProxyFactoryBean.html

I am particularly fond of its description: "Convenient proxy factory bean superclass for proxy factory beans that create only singletons."

But of course :)
Übersetzen

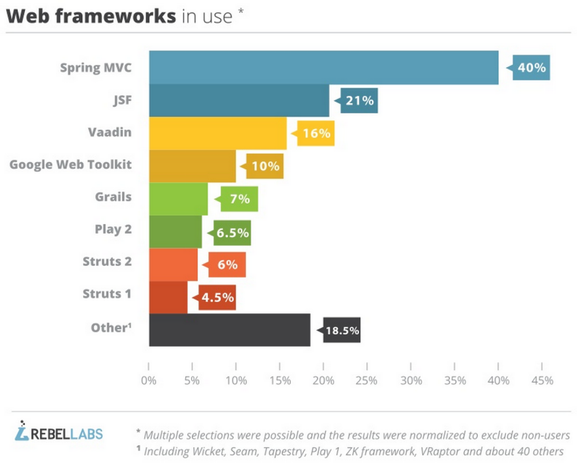
AbstractSingletonProxyFactoryBean (Spring Framework API 2.5)
static.springsource.org

https://plus.google.com/+aerotwist/posts/1QhcnQizuPc



https://twitter.com/vladon/status/659248116768645120

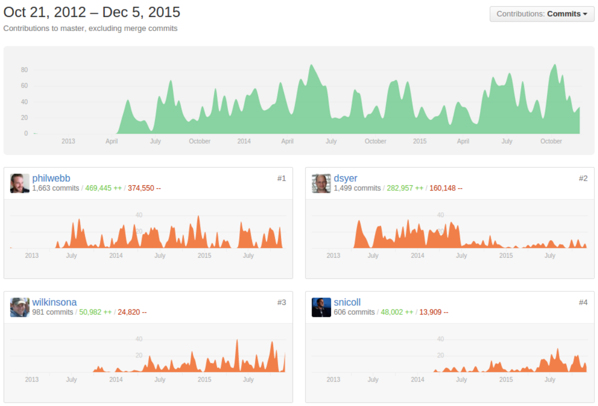
...aber:



Januar 2015, `http://zeroturnaround.com/rebellabs/`

`top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/`

Spring Boot: Entwicklungsstand


Oct 21, 2012 – Dec 5, 2015

Contributions to master, excluding merge commits

philwebb #1
1,663 commits / 469,445 ++ / 374,550 --

dsyer #2
1,499 commits / 282,957 ++ / 160,148 --

wilkinsona #3
981 commits / 50,982 ++ / 24,820 --

snicoll #4
606 commits / 48,002 ++ / 13,909 --

- Erstes Release im April 2014, aktuell bei 1.3, kontinuierlich weiterentwickelt
- Fast 40% Adoption unter Spring-Entwicklern
  (Stand Mai, http://www.baeldung.com/java-8-spring-4-and-spring-boot-adoption)

Spring Boot: Features

1. Kein XML ;-)
2. Convention over configuration
3. Kuratierte Abhängigkeiten
4. Konfigurierbarkeit
5. Einfache Integration unterschiedlicher Technologien
6. Einfaches Deployment (fat jar)

12 Faktoren für Web apps?

Von `http://12factor.net`:

1. Codebase One codebase tracked in revision control, many deploys
2. Dependencies Explicitly declare and isolate dependencies
3. **Config** Store config in the environment
4. Backing Services Treat backing services as attached resources
5. **Build, release, run** Strictly separate build and run stages
6. Processes Execute the app as one or more stateless processes
7. **Port binding** Export services via port binding
8. Concurrency Scale out via the process model
9. **Disposability** Maximize robustness with fast startup and graceful shutdown
10. **Dev/prod parity** Keep development, staging, and production as similar as possible
11. **Logs** Treat logs as event streams
12. **Admin processes** Run admin/management tasks as one-off processes

Maven-Setup[1]

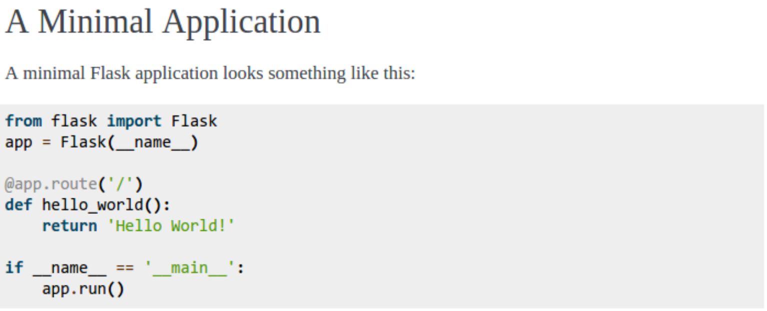
```xml
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.3.0.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<!-- ... -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
<!-- ... -->
```

[1] Gradle und Ant werden auch unterstützt

Minimalbeispiel

```java
@SpringBootApplication
@RestController
public class SpringDemoMinimal {

  @RequestMapping("/")
  public String testEndpoint(@RequestParam Optional<String> testParam) {
    return "Hello World!";
  }

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

Spring ist also ein Microframework! ;-)



A Minimal Application

A minimal Flask application looks something like this:

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

(von `http://flask.pocoo.org`)

Konfiguration

```java
@SpringBootApplication
@RestController
public class SpringDemoRest {

  @Value("${demo.test.parameter:testValue}")
  private String myTestParameter;

  @Value("${demo.test.mandatory.parameter}")
  private String myImportantParameter;

  @RequestMapping("/test")
  public List<String> testEndpoint(@RequestParam Optional<String> testParam) {
    return asList("test" + testParam.orElse("-"), "test2", "test3",
          myTestParameter, myImportantParameter);
  }

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

application.property Dateien

Anwendungsspezifische Konfiguration[2]:

```
#demo.test.parameter=This is the new test parameter
demo.test.mandatory.parameter=I am important
my.property.string=This is a string
my.property.integer=42
#Funktioniert nicht, falscher Typ: my.property.integer=42 test
```

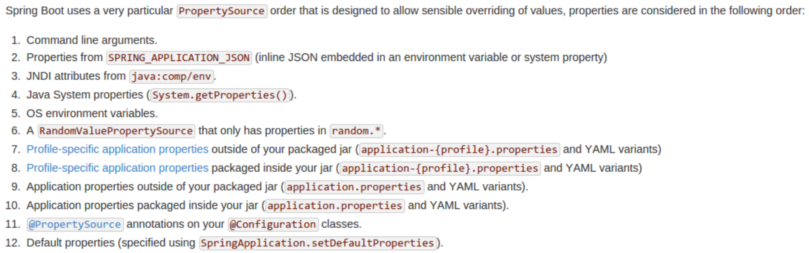
Spring Boot Handbuch, Appendix A:

```
# ===================================================================
# COMMON SPRING BOOT PROPERTIES
#
# This sample file is provided as a guideline. Do NOT copy it in its
# entirety to your own application.          ^^^
# ===================================================================

# ----------------------------------------
# CORE PROPERTIES
# ----------------------------------------

# BANNER
banner.charset=UTF-8 # Banner file encoding.
banner.location=classpath:banner.txt # Banner file location.

# ... etc. --- noch ca. 800 mehr :)
```

https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

[2]YAML wird auch unterstützt.

Wie Konfigurationselemente aufgelöst werden



Spring Boot uses a very particular `PropertySource` order that is designed to allow sensible overriding of values, properties are considered in the following order:

1. Command line arguments.
2. Properties from `SPRING_APPLICATION_JSON` (inline JSON embedded in an environment variable or system property)
3. JNDI attributes from `java:comp/env`.
4. Java System properties (`System.getProperties()`).
5. OS environment variables.
6. A `RandomValuePropertySource` that only has properties in `random.*`.
7. Profile-specific application properties outside of your packaged jar (`application-{profile}.properties` and YAML variants)
8. Profile-specific application properties packaged inside your jar (`application-{profile}.properties` and YAML variants)
9. Application properties outside of your packaged jar (`application.properties` and YAML variants).
10. Application properties packaged inside your jar (`application.properties` and YAML variants).
11. `@PropertySource` annotations on your `@Configuration` classes.
12. Default properties (specified using `SpringApplication.setDefaultProperties`).

`https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html`

- Bitte nicht alle 12 Arten benutzen! ;-)
- Checks & Logging der wichtigsten Parameter beim Start ist empfehlenswert (`z.B. via @PostConstruct`)

Zu viel Magic?

Auto-Konfiguration loswerden & ansehen

```
@SpringBootApplication
@RestController
@EnableAutoConfiguration(exclude={WebMvcAutoConfiguration.class, ...})
public class SpringDemoRest {
//...
```

`https://github.com/spring-projects/spring-boot/blob/master/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/WebMvcAutoConfiguration.java`

'Magic' ist also eigentlich:

- Intention des Entwicklers erkennen:
  - `@ConditionalOnClass`
  - `@ConditionalOnMissingBean`
  - `@ConditionalOnBean`
- Konfiguration lesen: `@ConfigurationProperties`

Testen

```java
// ...
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = SpringDemoRest.class)
@WebIntegrationTest("server.port=0") //Echter Wert steht dann in
     'local.server.port'
public class SpringDemoRestTest {

  @Autowired
  private WebApplicationContext wac;

  private MockMvc mockApp;

  @Before
  public void setUp() {
    this.mockApp = MockMvcBuilders.webAppContextSetup(this.wac).build();
  }

  @Test
  public void testWithMockMvc() throws Exception {
    MvcResult result =
          mockApp.perform(get("/test")).andExpect(status().isOk()).andReturn();
    assertTrue(result.getResponse().getContentAsString().contains("test2"));
  }
}
```

Profile

- Erlauben das Gruppieren von Beans und Konfiguration (`application-tests.properties`)
- Helfen bei der Trennung von Entwickling/Produktion und beim Deployment

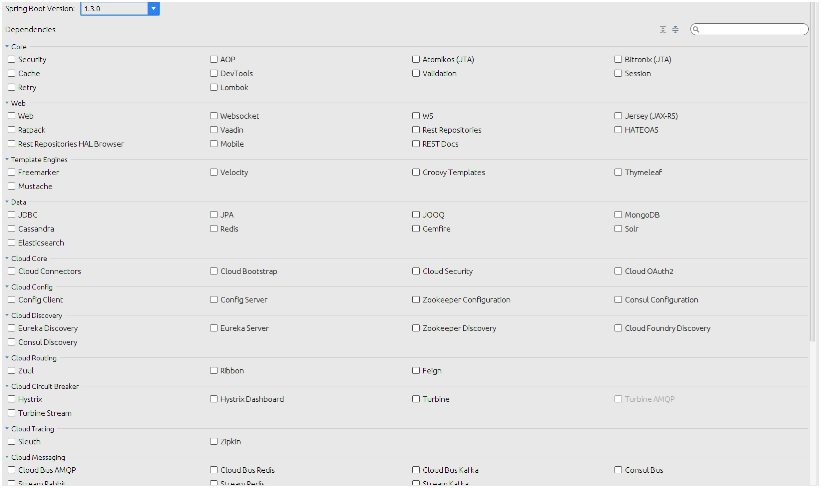
```java
@Component
@Profile("tests", "demo")
public class MyComplexTaskMockup {
  // ...
}
```

```java
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = SpringDemoRest.class)
@WebIntegrationTest("server.port=0")
@ActiveProfiles({"tests"})
public class TestSiteWithoutComplexTask {
  // ...
}
```

Weitere Beispiele

- 'Hello World' mit Spring Boot + Vaadin
- Metriken & Health
- Elasticsearch

… und es gibt noch viel mehr



Spring Boot Version: 1.3.0

Dependencies

Core

Security	AOP	Atomikos (JTA)	Bitronix (JTA)
Cache	DevTools	Validation	Session
Retry	Lombok		

Web

Web	Websocket	WS	Jersey (JAX-RS)
Ratpack	Vaadin	Rest Repositories	HATEOAS
Rest Repositories HAL Browser	Mobile	REST Docs	

Template Engines

Freemarker	Velocity	Groovy Templates	Thymeleaf
Mustache			

Data

JDBC	JPA	JOOQ	MongoDB
Cassandra	Redis	Gemfire	Solr
Elasticsearch			

Cloud Core

Cloud Connectors	Cloud Bootstrap	Cloud Security	Cloud OAuth2

Cloud Config

Config Client	Config Server	Zookeeper Configuration	Consul Configuration

Cloud Discovery

Eureka Discovery	Eureka Server	Zookeeper Discovery	Cloud Foundry Discovery
Consul Discovery			

Cloud Routing

Zuul	Ribbon	Feign	

Cloud Circuit Breaker

Hystrix	Hystrix Dashboard	Turbine	Turbine AMQP
Turbine Stream			

Cloud Tracing

Sleuth	Zipkin		

Cloud Messaging

Cloud Bus AMQP	Cloud Bus Redis	Cloud Bus Kafka	Consul Bus
Stream Rabbit	Stream Redis	Stream Kafka	

Zusammenfassung

Zusammenfassung

- Spring Boot ist immer noch Spring — no Silver Bullet
- Spring Boot macht die Entwicklung von "12-Faktor-Apps" in einigen Aspekten einfacher
- Spring-Anfänger: aufpassen, Spring Boot und Spring sind *verschiedene Projekte*
- Ein Blick auf die `*AutoConfigurer`-Quellen lohnt sich

Material zum Weitermachen

- Phil Webbs Antwort auf *Why I hate Spring — How not to hate Sprig in 2016*: `https://spring.io/blog/2015/11/29/how-not-to-hate-spring-in-2016`
- Handbuch: `http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/`
- Beispiele: `https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples`
- Die Zauberei: `https://github.com/spring-projects/spring-boot/tree/master/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure`
- Projektgeneratoren, z.B. `https://start.spring.io/` oder `https://jhipster.github.io/`

Lizenz