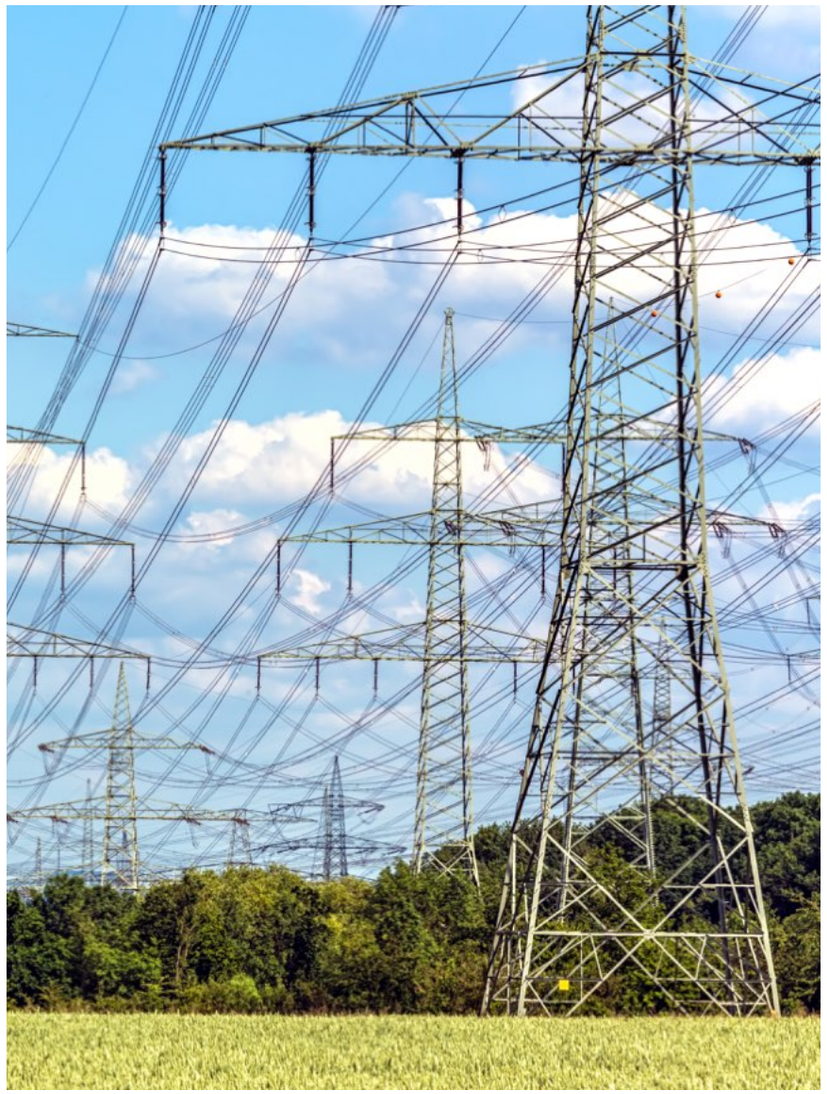
Michael Simons | ENERKO INFORMATIK GmbH

Keine Magie: Individuelle Spring-Boot-Module

Über mich

- Entwickler bei ENERKO INFORMATIK in Aachen
  - Datenbankzentrische Anwendungen im Energiemarkt
- Leiter Euregio JUG
- Bloggt zu Java und Spring Themen unter info.michael-simons.eu
- Co-Autor arc(42) by example
- @rotnroll666 auf Twitter



```
        <property name="fallbackToSystemLocale" value="false"/>
    </bean>
    <!-- Configure the JPA Adapter -->
    <bean id="jpaDialect" class="org.springframework.orm.jpa.vendor.HibernateJpaDialect"/>
    <bean id="jpaVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
      <property name="database" value="MYSQL"/>
      <property name="databasePlatform" value="org.hibernate.dialect.MySQLDialect"/>
      <property name="generateDdl" value="false"/>
      <property name="showSql" value="false"/>
    </bean>
    <!-- Configure the local Entity Manager Factory -->
    <bean id="entityManagerFactory"
  class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
      <property name="persistenceUnitName" value="theEntities"/>
      <property name="dataSource" ref="theDataSource"/>
      <property name="jpaDialect" ref="jpaDialect"/>
      <property name="jpaVendorAdapter" ref="jpaVendorAdapter"/>
      <property name="loadTimeWeaver">
        <bean class="org.springframework.instrument.classloading.InstrumentationLoadTimeWeaver"/>
      </property>
    </bean>
    <!-- Enable Spring JPA Transactions -->
    <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager" p:entity-
manager-factory-ref="entityManagerFactory"/>
    <tx:annotation-driven transaction-manager="transactionManager"/>
    <bean id="exampleBean" class="examples.ExampleBean">
      <!-- setter injection using the nested <ref/> element -->
      <property name="beanOne">
        <ref bean="anotherExampleBean"/>
      </property>
      <!-- setter injection using the neater 'ref' attribute -->
      <property name="beanTwo" ref="yetAnotherBean"/>
      <property name="integerProperty" value="1"/>
    </bean>
    <bean id="anotherExampleBean" class="examples.AnotherBean"/>
    <bean id="yetAnotherBean" class="examples.YetAnotherBean"/>
</beans>
```

A long time ago, in a framework far,
far away

A new hope

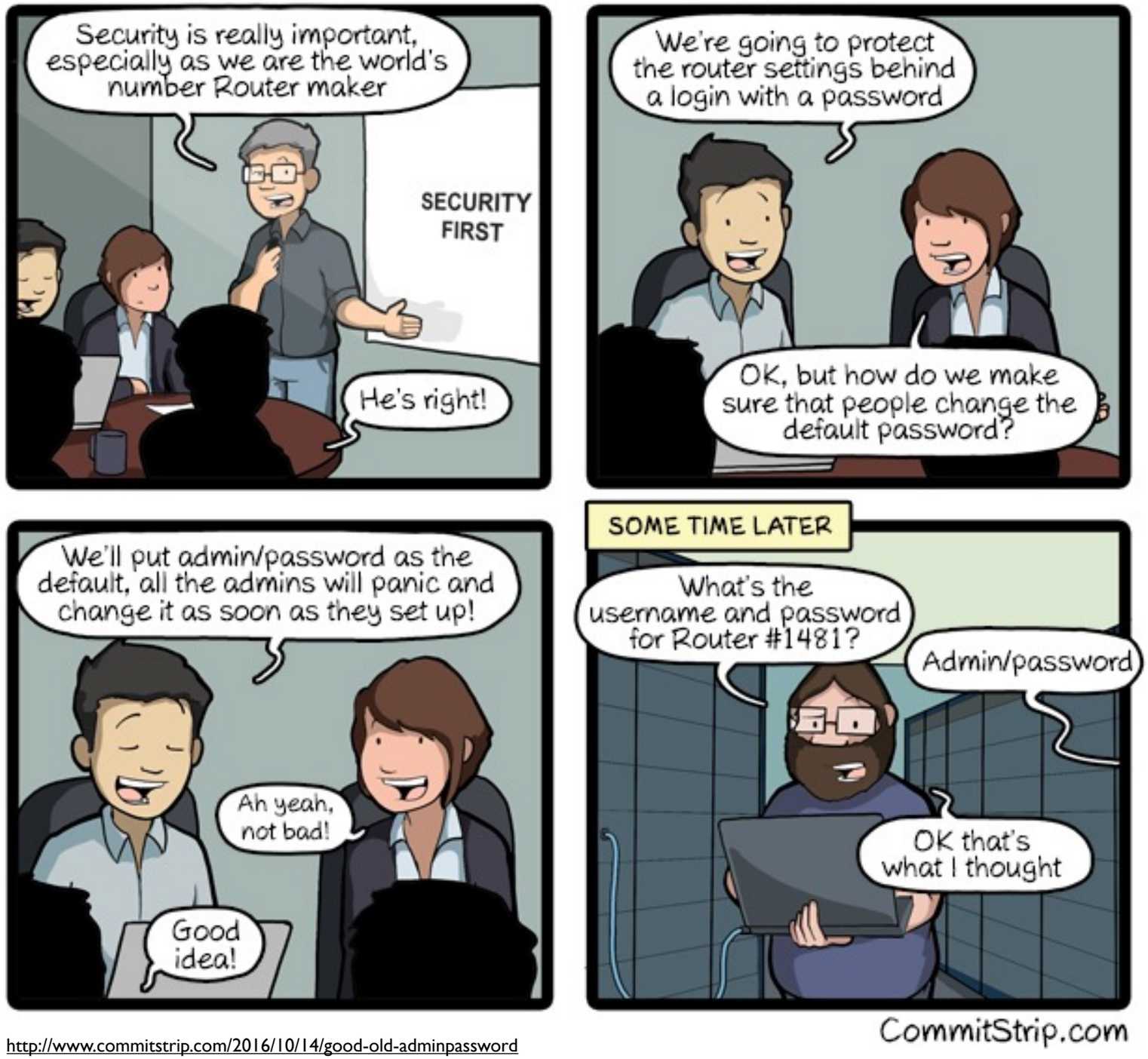
```
package ac.simons.wjax2016.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

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

Agenda

1. Was ist Spring Boot?
2. Was sind Spring Boot Starter?
3. Architektur eines Starters
    i. Autokonfiguration
    ii. Bedingungen und Struktur
    iii. Analyse

Was ist Spring Boot?

- Ziel: Schneller Start für Entwicklung mit Spring
- Großer Umfang nicht-funktionaler Eigenschaften
- Extern konfigurierbar
- Sinnvolle Defaults
  - kein Code oder Konfigurationsgenerator
  - solange wie nötig!



http://www.commitstrip.com/2016/10/14/good-old-adminpassword

Autokonfiguration, wie sie nicht sein sollte ;)

Besser:

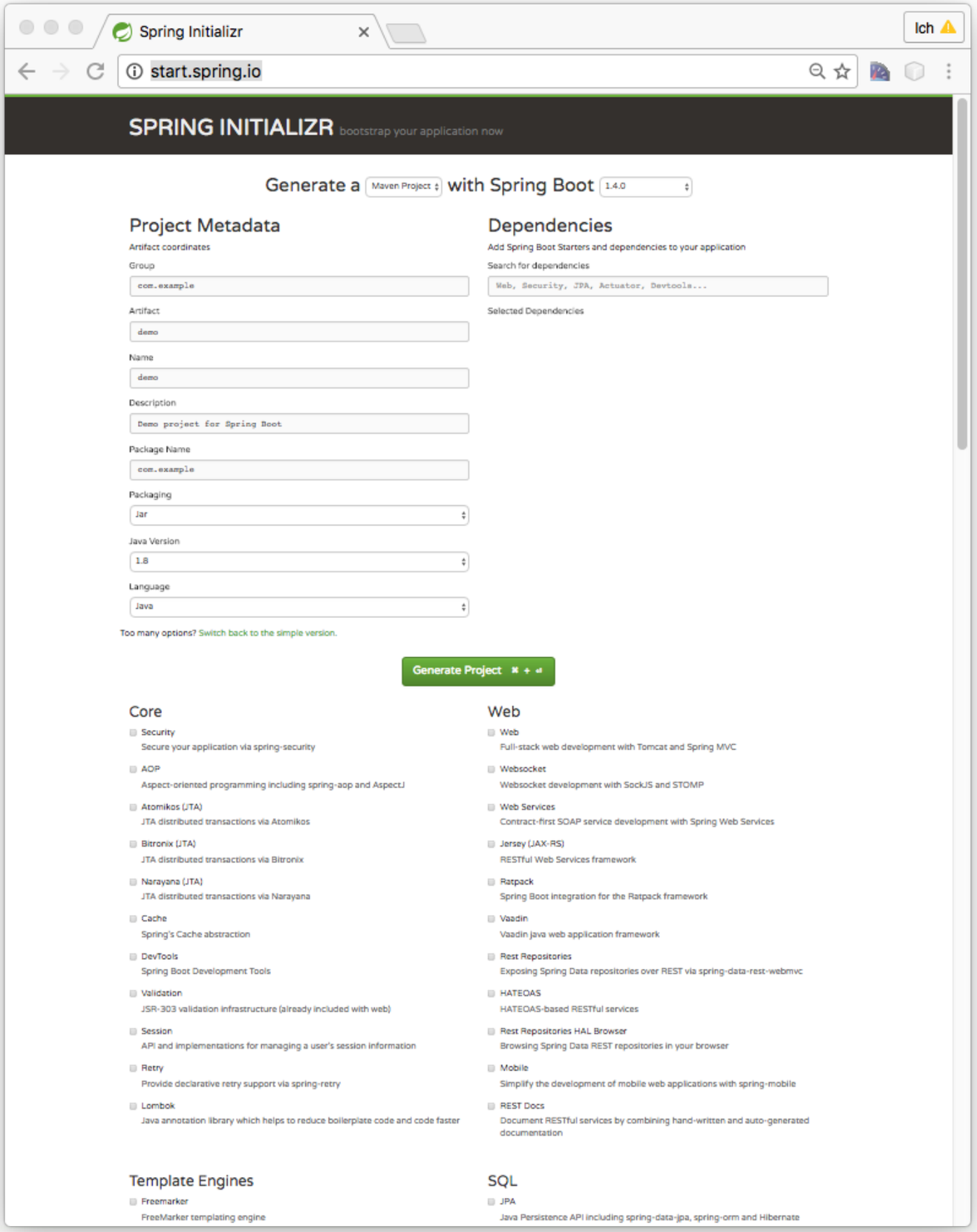
```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
</dependencies>
```

```
2016-11-04 09:44:52.436  INFO 67251 --- [           main] b.a.s.AuthenticationManagerConfiguration :

Using default security password: 9b3cebaf-11a8-4984-8870-1705995b5356



2016-11-04 09:47:32.890  INFO 67386 --- [           main] b.a.s.AuthenticationManagerConfiguration :

Using default security password: 2f607633-51cc-4d32-89f5-9c684e1e4a50
```

Modularisierung: Kein „Fat-Jar“

Was genau „starten“?

- Alle Arten von Spring Boot Startern
  - Security
  - Datenbanken
  - Template Engines
  - Validation
  - Service Discovery
- Viele mehr: https://github.com/spring-projects/spring-boot/tree/master/spring-boot-starters


Spring Initializr
Ich
start.spring.io
SPRING INITIALIZR bootstrap your application now
Generate a Maven Project with Spring Boot 1.4.0
Project Metadata
Artifact coordinates
Group
com.example
Artifact
demo
Name
demo
Description
Demo project for Spring Boot
Package Name
com.example
Packaging
Jar
Java Version
1.8
Language
Java
Too many options? Switch back to the simple version.
Dependencies
Add Spring Boot Starters and dependencies to your application
Search for dependencies
Web, Security, JPA, Actuator, Devtools...
Selected Dependencies
Generate Project
Core
Security
Secure your application via spring-security
AOP
Aspect-oriented programming including spring-aop and AspectJ
Atomikos (JTA)
JTA distributed transactions via Atomikos
Bitronix (JTA)
JTA distributed transactions via Bitronix
Narayana (JTA)
JTA distributed transactions via Narayana
Cache
Spring's Cache abstraction
DevTools
Spring Boot Development Tools
Validation
JSR-303 validation infrastructure (already included with web)
Session
API and implementations for managing a user's session information
Retry
Provide declarative retry support via spring-retry
Lombok
Java annotation library which helps to reduce boilerplate code and code faster
Web
Web
Full-stack web development with Tomcat and Spring MVC
Websocket
Websocket development with SockJS and STOMP
Web Services
Contract-first SOAP service development with Spring Web Services
Jersey (JAX-RS)
RESTful Web Services framework
Ratpack
Spring Boot integration for the Ratpack framework
Vaadin
Vaadin java web application framework
Rest Repositories
Exposing Spring Data repositories over REST via spring-data-rest-webmvc
HATEOAS
HATEOAS-based RESTful services
Rest Repositories HAL Browser
Browsing Spring Data REST repositories in your browser
Mobile
Simplify the development of mobile web applications with spring-mobile
REST Docs
Document RESTful services by combining hand-written and auto-generated documentation
Template Engines
Freemarker
FreeMarker templating engine
SQL
JPA
Java Persistence API including spring-data-jpa, spring-orm and Hibernate

Architektur eines Starters

- Autokonfiguration
  - JavaConfig
  - spring.factories
- Starter Modul

Autokonfiguration - Eine Art Magie?

JavaConfig plus @Conditional / @xxxCondition Annotationen

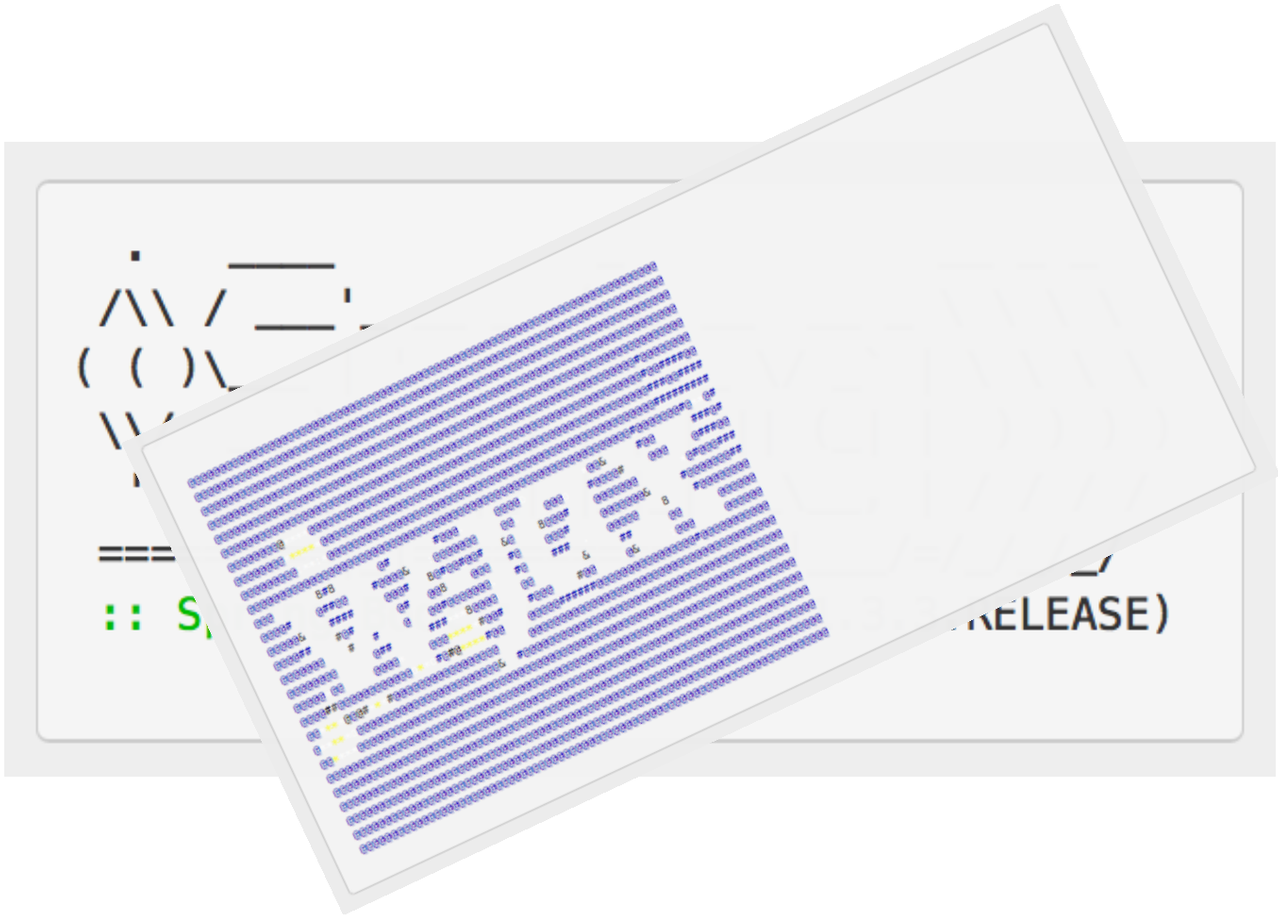
- OnClassCondition / OnMissingClassCondition
- OnBeanCondition / OnMissingBeanCondition
- OnPropertyCondition
- OnResourceCondition
- OnExpressionCondition
- OnJavaCondition
- OnJndiCondition
- OnWebApplicationCondition

Demo:
thymeleaf-banner-spring-boot-starter

Aus

```
<banner:show />
```

wird



Grundlegende Autokonfiguration

- @SpringBootApplication schaltet automatische Konfiguration ein
- spring.factories für @Configuration Klassen nicht vergessen
  - @ConditionalOnClass
  - @AutoConfigureBefore
  - @Bean

Mehr Bedingungen und Struktur

- @ConditionalOnBean / @ConditionalOnMissingBean
- @ConditionalOnProperty
- Ebenfalls gesehen: Spring Diagnostics

Analyse der Konfiguration

- Entweder mit dem --`debug` parameter
- Hinzufügen des `spring-boot-starter-actuator` stellt den REST Endpunkt `/autoconfig` bereit

Eigene Bedingungen

- Implementiere o.s.c.annotation.Condition
- Erweitere o.s.boot.autoconfigure.SpringBootCondition
- Verschachtelte Bedingungen mit
  - AllNestedConditions
  - AnyNestedCondition
  - NoneNestedCondition

Zusammenfassung

- Widerstandsfähige (resillient) Erweiterungen
- Nutzung der vorhandenen Spring Boot Infrastruktur
- Servicelocator
- Anwendungsfälle
  - unterschiedliche Repositories je nach Umgebung
  - Shared Kernel
  - Sammlung eigener UI Komponenten / tags

Ressourcen

- Dieser Vortrag: https://github.com/michael-simons/W-JAX2016
- Einer meiner nützlicheren Starter: wro4j-spring-boot-starter
- Kontakt: michael-simons.eu
- Gutschein für **arc42 by example** http://leanpub.com/arc42byexample/c/WJAX2016