24. September 2019
Düsseldorf / JCON

Spring Boot entzaubert

INNOQ

MICHAEL VITZ

Senior Consultant
bei INNOQ Deutschland GmbH

Akt 1

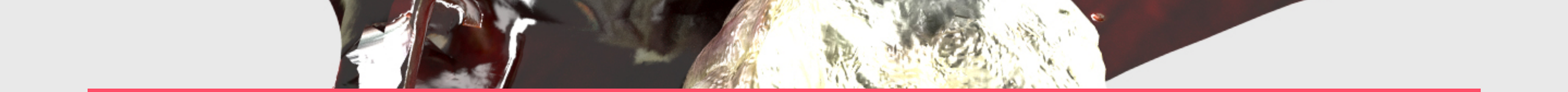
"The Pledge"

Akt 2

"The Turn"

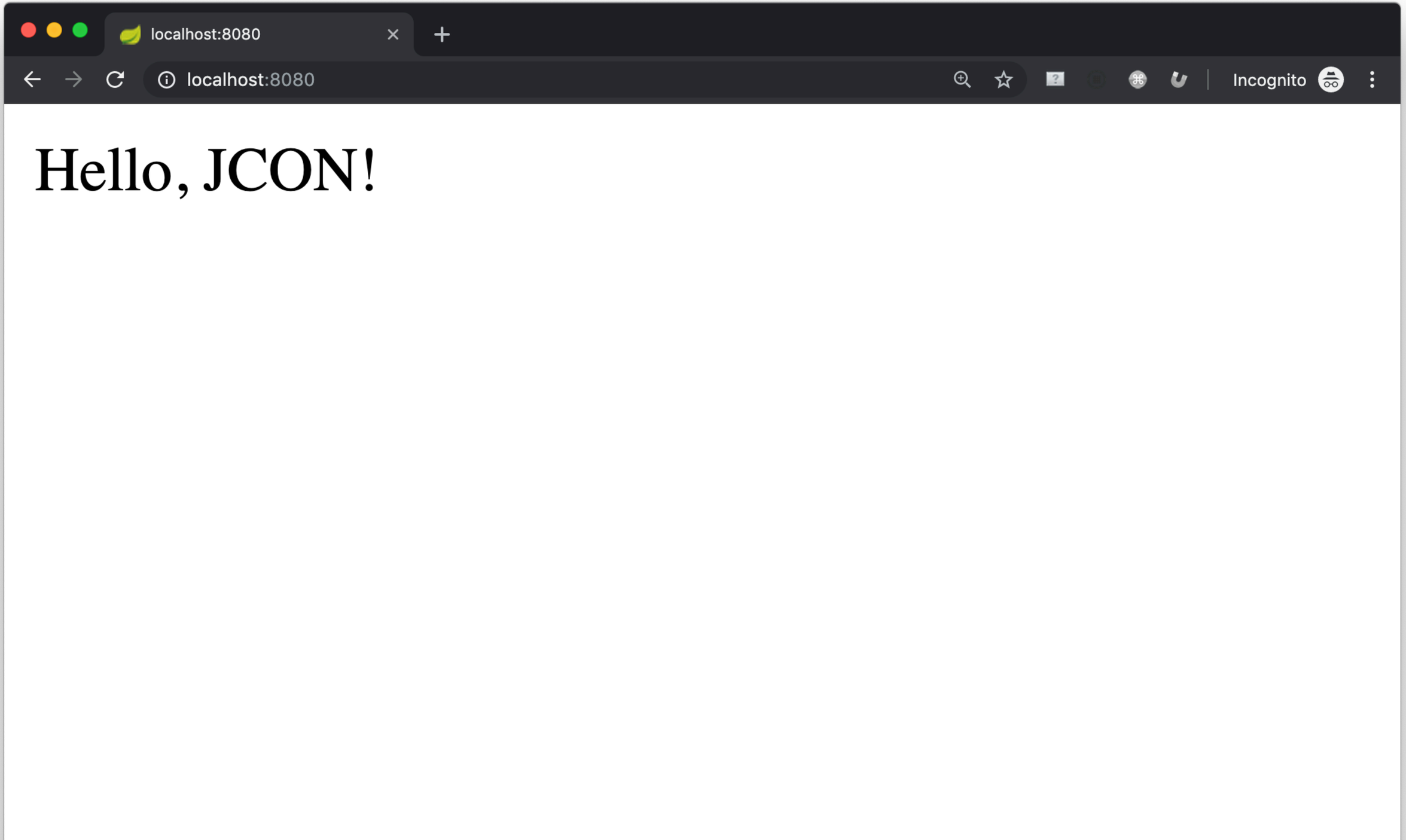
Akt 3

“The Prestige”



Trick 1
Embedded Web-Server

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

    @GetMapping
    public String index() {
        return "Hello, JCON!";
    }

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

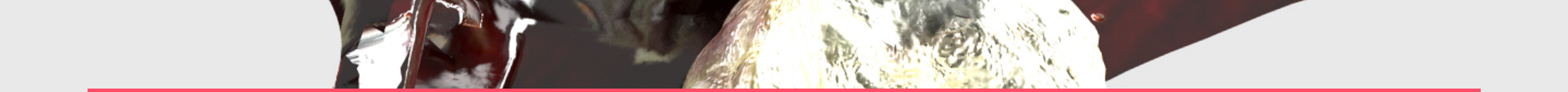

localhost:8080
localhost:8080
Incognito
Hello, JCON!

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

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

Embedded Web-Server Auswahl

- Tomcat (Default)
- Jetty
- Undertow

https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-developing-web-applications.html#boot-features-embedded-container

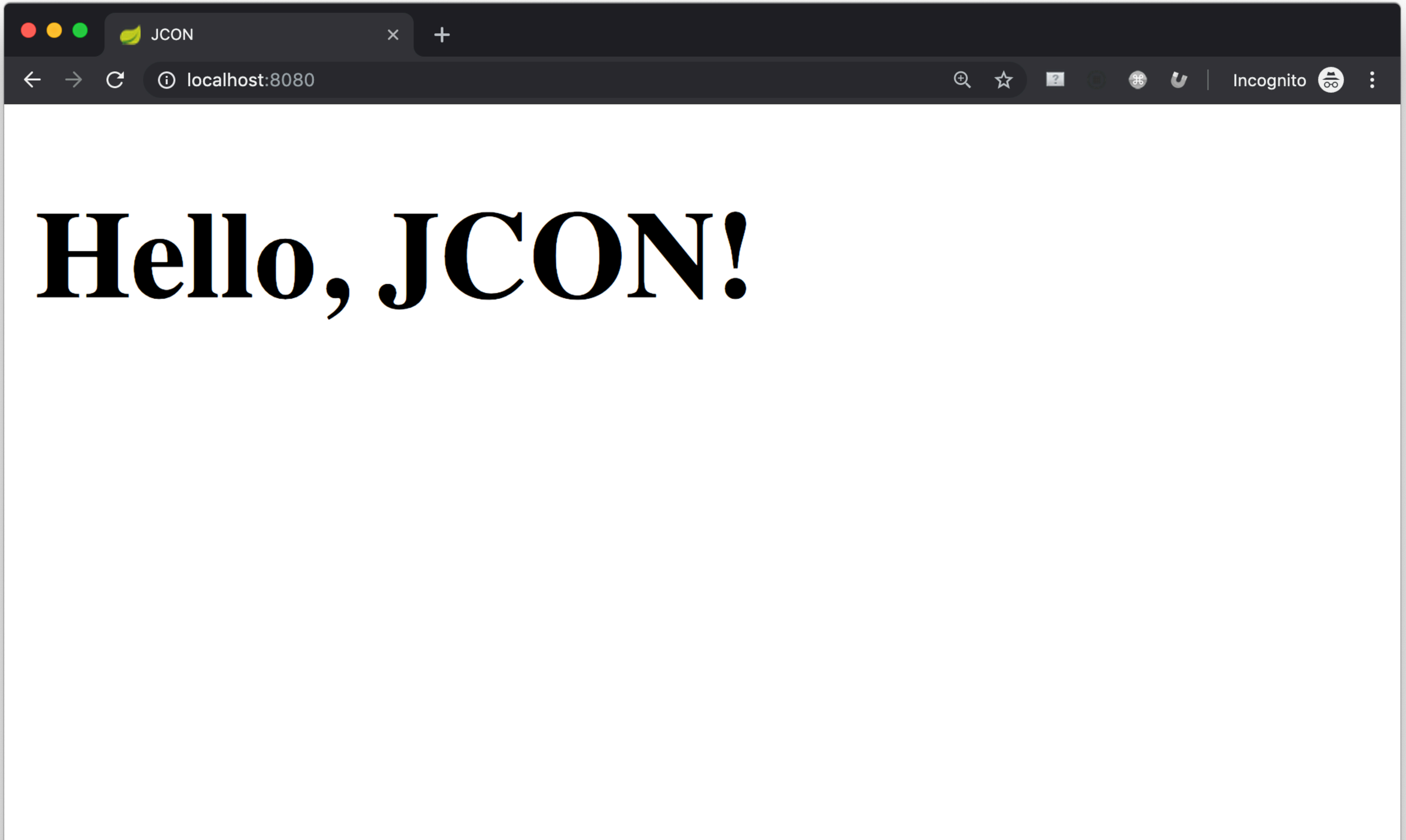


Trick 2
Automatische Konfiguration

```
@Controller
@SpringBootApplication
public class Application {

    @GetMapping
    public String index() {
        return "index";
    }

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

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="utf-8"/>

    <title>JCON</title>
  </head>
  <body>
    <h1>Hello, JCON!</h1>
  </body>
</html>
```

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


JCON
localhost:8080
Incognito
Hello, JCON!

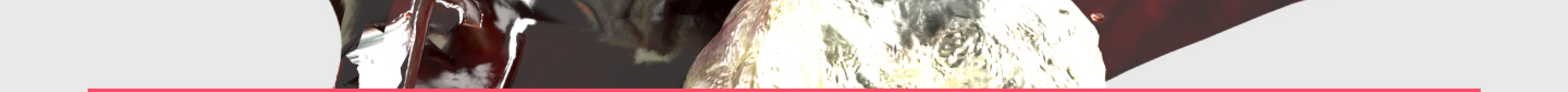
ThymeleafAutoConfiguration.java

```
78      */
79      @Configuration(proxyBeanMethods = false)
80      @EnableConfigurationProperties(ThymeleafProperties.class)
81      @ConditionalOnClass({ TemplateMode.class, SpringTemplateEngine.class })
82      @AutoConfigureAfter({ WebMvcAutoConfiguration.class, WebFluxAutoConfiguration.class })
83      public class ThymeleafAutoConfiguration {
84
85          @Configuration(proxyBeanMethods = false)
86          @ConditionalOnMissingBean(name = "defaultTemplateResolver")
87          static class DefaultTemplateResolverConfiguration { ... }
132
133         @Configuration(proxyBeanMethods = false)
134         protected static class ThymeleafDefaultConfiguration {
135
136             @Bean
137             @ConditionalOnMissingBean(ISpringTemplateEngine.class)
138 @           SpringTemplateEngine templateEngine(ThymeleafProperties properties,
139                     ObjectProvider<ITemplateResolver> templateResolvers, ObjectProvider<IDialect> dialects) { ... }
147
148         }
149
150         @Configuration(proxyBeanMethods = false)
151         @ConditionalOnWebApplication(type = Type.SERVLET)
152         @ConditionalOnProperty(name = "spring.thymeleaf.enabled", matchIfMissing = true)
153         static class ThymeleafWebMvcConfiguration {
```

Eigene Auto-Konfiguration

- Besteht aus
  - Klasse mit @Configuration und Bedingungen
  - Eintrag in META-INF/spring.factories der auf Klasse verweist
- Optional
  - Weitere Klassen
  - Tests für die eigene Auto-Konfiguration

https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-developing-auto-configuration.html



Trick 3
Abhängigkeitsmanagement

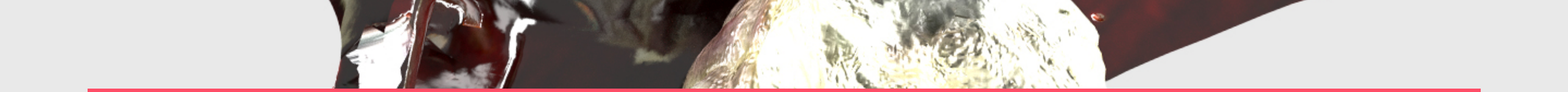
```
 <parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>2.2.0.M6</version>
   <relativePath/>
 </parent>

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

```
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------------< com.innoq:cfps >---------------------------
[INFO] Building cfps 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-dependency-plugin:3.1.1:tree (default-cli) @ cfps ---
[INFO] com.innoq:cfps:jar:0.0.1-SNAPSHOT
[INFO] +- org.springframework.boot:spring-boot-starter-thymeleaf:jar:2.2.0.M6:compile
[INFO] |  +- org.springframework.boot:spring-boot-starter:jar:2.2.0.M6:compile
[INFO] |  |  +- org.springframework.boot:spring-boot-starter-logging:jar:2.2.0.M6:compile
[INFO] |  |  |  +- ch.qos.logback:logback-classic:jar:1.2.3:compile
[INFO] |  |  |  |  \- ch.qos.logback:logback-core:jar:1.2.3:compile
[INFO] |  |  |  +- org.apache.logging.log4j:log4j-to-slf4j:jar:2.12.1:compile
[INFO] |  |  |  |  \- org.apache.logging.log4j:log4j-api:jar:2.12.1:compile
[INFO] |  |  |  \- org.slf4j:jul-to-slf4j:jar:1.7.28:compile
[INFO] |  |  \- org.yaml:snakeyaml:jar:1.25:runtime
[INFO] |  +- org.thymeleaf:thymeleaf-spring5:jar:3.0.11.RELEASE:compile
[INFO] |  |  +- org.thymeleaf:thymeleaf:jar:3.0.11.RELEASE:compile
[INFO] |  |  |  +- org.attoparser:attoparser:jar:2.0.5.RELEASE:compile
[INFO] |  |  |  \- org.unbescape:unbescape:jar:1.1.6.RELEASE:compile
[INFO] |  |  \- org.slf4j:slf4j-api:jar:1.7.28:compile
[INFO] |  \- org.thymeleaf.extras:thymeleaf-extras-java8time:jar:3.0.4.RELEASE:compile
[INFO] +- org.springframework.boot:spring-boot-starter-web:jar:2.2.0.M6:compile
[INFO] |  +- org.springframework.boot:spring-boot-starter-json:jar:2.2.0.M6:compile
[INFO] |  |  +- com.fasterxml.jackson.core:jackson-databind:jar:2.9.9.3:compile
[INFO] |  |  |  +- com.fasterxml.jackson.core:jackson-annotations:jar:2.9.0:compile
```

```
Archive:  spring-boot-starter-thymeleaf-2.2.0.M6.jar
  Length      Date    Time    Name
---------  ---------- -----   ----
       239  09-10-2019 11:38   META-INF/MANIFEST.MF
         0  09-10-2019 11:38   META-INF/
---------                     -------
       239                     2 files
```

Spring Boot Starter

- Verwalten “nur” die Abhängigkeiten
- Eigentliche Logik wird durch die Autoconfigure Abhängigkeit bereitgestellt
- über 50 offizielle Starter
  https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html#using-boot-starter
- zusätzliche von der Community gepflegte
  https://github.com/spring-projects/spring-boot/blob/master/spring-boot-project/spring-boot-starters/README.adoc



Trick 4
Externe Konfiguration

```
java -jar ./target/cfps.jar --server.port=8090
```

```
…
Tomcat started on port(s): 8090 (http) with context path ''
…
```

Externe Konfigurationsquellen

- Command Line Argumente
- JNDI
- Java System Properties
- Umgebungsvariablen
- Property (oder YAML) Dateien an diversen Stellen

Externe Konfiguration Benutzung

- Verwendung via **@Value("${my.property}")**
- oder über eigene **@ConfigurationProperties** Klasse
- Hunderte von bereits vorhandenen Properties https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html
- Mittels **spring-boot-configuration-processor** Auto-Completion für IDEs https://docs.spring.io/spring-boot/docs/current/reference/html/configuration-metadata.html#configuration-metadata-annotation-processor

Danke! Fragen?

Michael Vitz
michael.vitz@innoq.com
+49 151 19116015
@michaelvitz

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany
+49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
+49 2173 3366-0

Kreuzstr. 16
80331 München
Germany
+49 2173 3366-0

Hermannstrasse 13
20095 Hamburg
Germany
+49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 0116