SpringBoot - Magie?
Oder doch nur mit Wasser gekocht.

Birgit Kratz, betterCode() Spring 2024

About me

Birgit Kratz

Freelancing IT Consultant

Java-Backend

SpringBoot Trainer

Co-Organizer of Softwerkskammer Düsseldorf and Cologne

Co-Organizer of SoCraTes-Conference Germany

https://www.birgitkratz.de

Common opinions about SpringBoot

Ease of Use

simple setup and configuration process, little boilerplate, use of auto-configuration

Quick start

get a new application up and running quickly, use of embedded servers contributes to a faster development cycle

Integration

integrates well with other Spring projects and third-party libraries

Documentation

well documented with plenty of explanations and example code

Production-Ready

Actuator, out-of-the-box tools for monitoring and managing applications

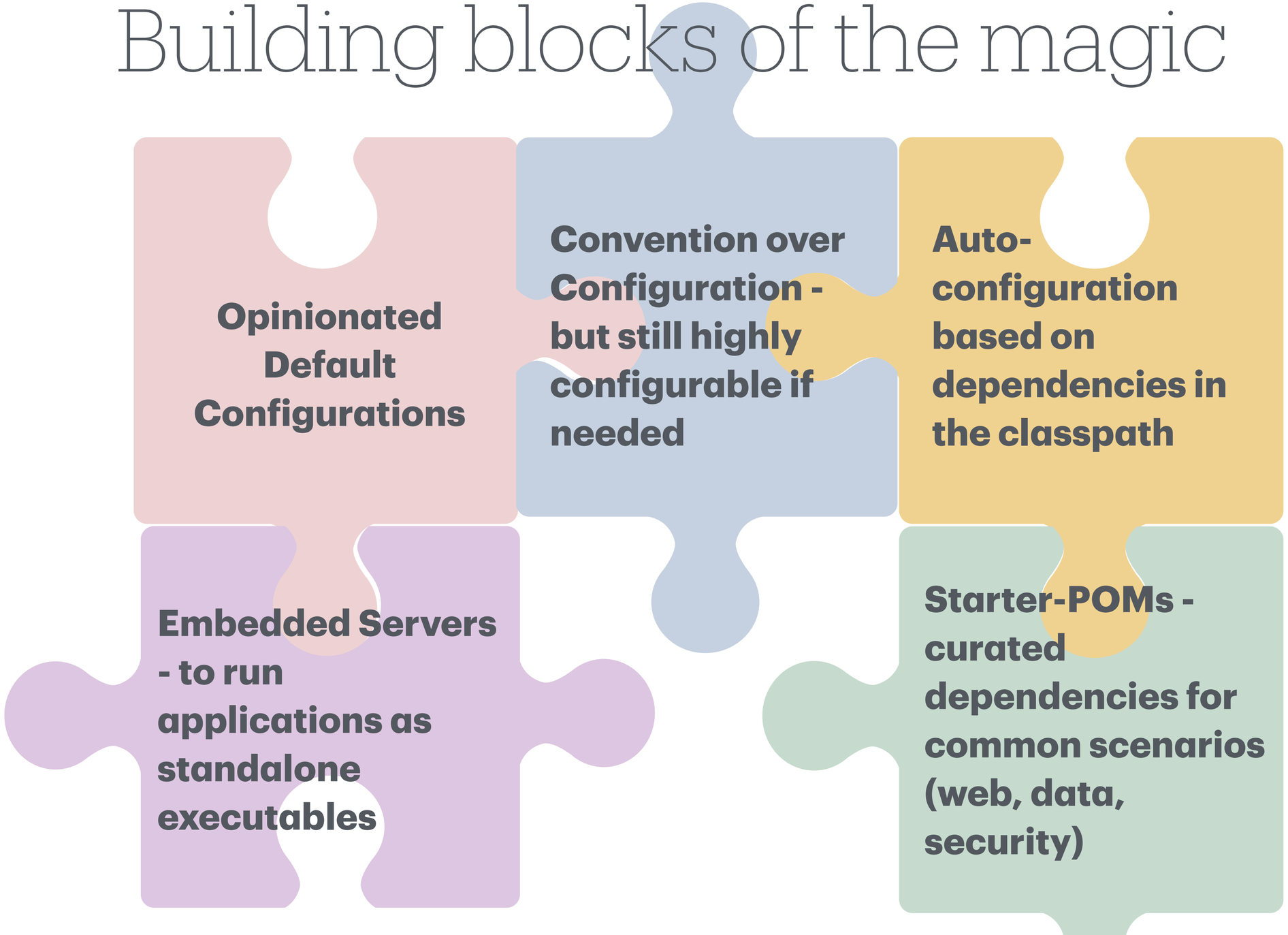
Common opinions about SpringBoot

Learning Curve

While SpringBoot makes many things easier, some developers feel **overwhelmed** by the number of features and options available.

Understanding the **magic** behind **auto-configuration** and embedded servers can be **challenging** initially.

Building blocks of the magic



How exactly does auto-configuration work?

@SpringBootApplication

```java
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
        excludeFilters = {@Filter(
                type = FilterType.CUSTOM,
                classes = {TypeExcludeFilter.class}
        ), @Filter(
                type = FilterType.CUSTOM,
                classes = {AutoConfigurationExcludeFilter.class}
        )}
)
public @interface SpringBootApplication {
```

```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
        excludeFilters = {@Filter(
                type = FilterType.CUSTOM,
                classes = {TypeExcludeFilter.class}
        ), @Filter(
                type = FilterType.CUSTOM,
                classes = {AutoConfigurationExcludeFilter.class}
        )}
)
public @interface SpringBootApplication {
```

@Service

@Repository

@Component

@Controller

@RestController

@Configruation

@Bean

@Repository

@Component

@Service

@Configruation

@Controller

@Bean

@RestController

Bild von anskuvartiainen auf Pixabay

Application Context

User configuration

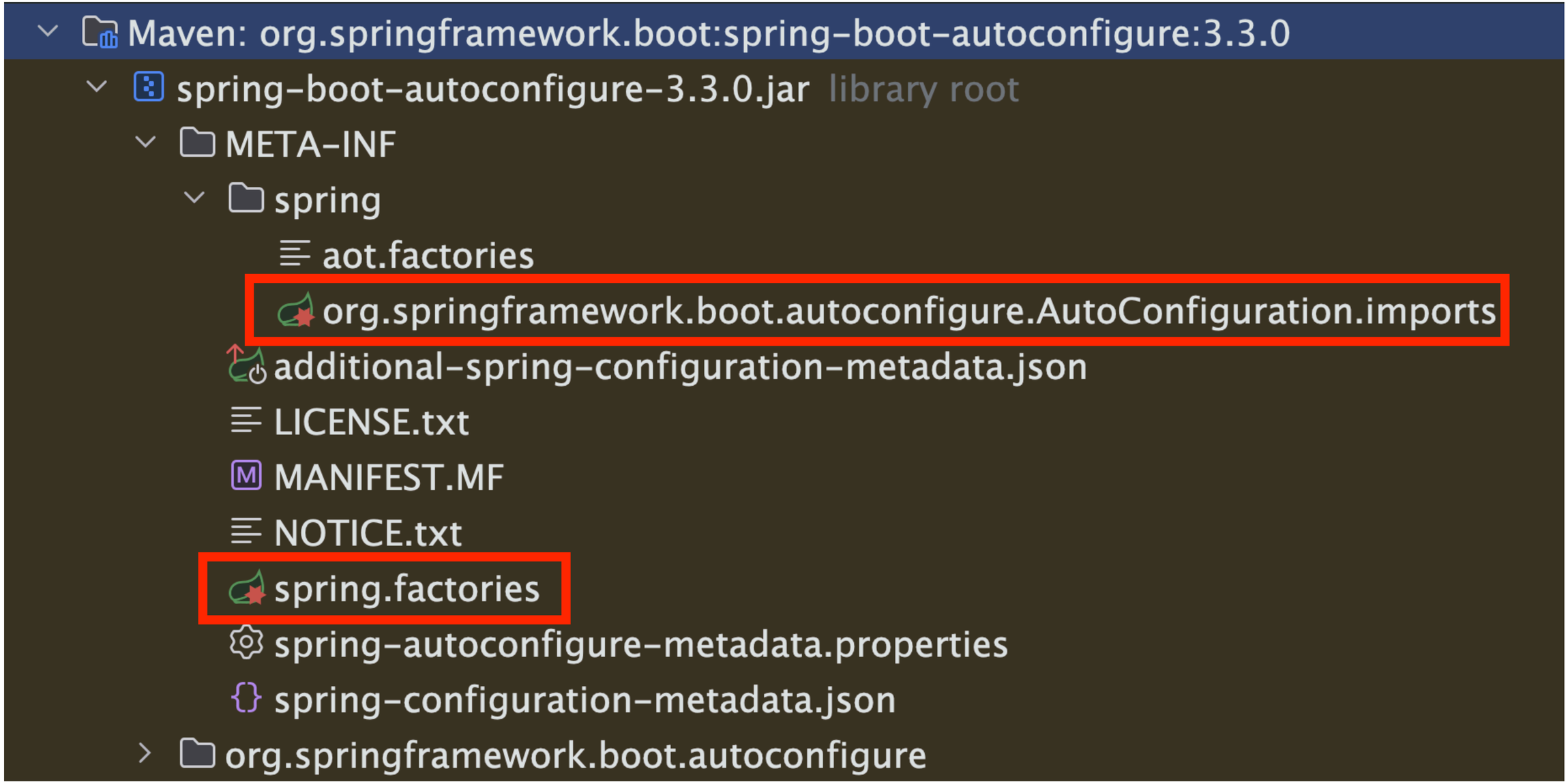
@ComponentScan

```java
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
        excludeFilters = {@Filter(
                type = FilterType.CUSTOM,
                classes = {TypeExcludeFilter.class}
        ), @Filter(
                type = FilterType.CUSTOM,
                classes = {AutoConfigurationExcludeFilter.class}
        )}
)
public @interface SpringBootApplication {
```

SpringBoot auto configuration

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

> Maven: org.springframework.boot:spring-boot-autoconfigure:3.3.0


Maven: org.springframework.boot:spring-boot-autoconfigure:3.3.0
spring-boot-autoconfigure-3.3.0.jar library root
META-INF
spring
aot.factories
org.springframework.boot.autoconfigure.AutoConfiguration.imports
additional-spring-configuration-metadata.json
LICENSE.txt
MANIFEST.MF
NOTICE.txt
spring.factories
spring-autoconfigure-metadata.properties
spring-configuration-metadata.json
org.springframework.boot.autoconfigure

Application Context

User configuration

@ComponentScan

Auto configuration

@EnableAutoConfiguration

Overview of all auto-configuration classes inside spring-boot-autoconfigure module

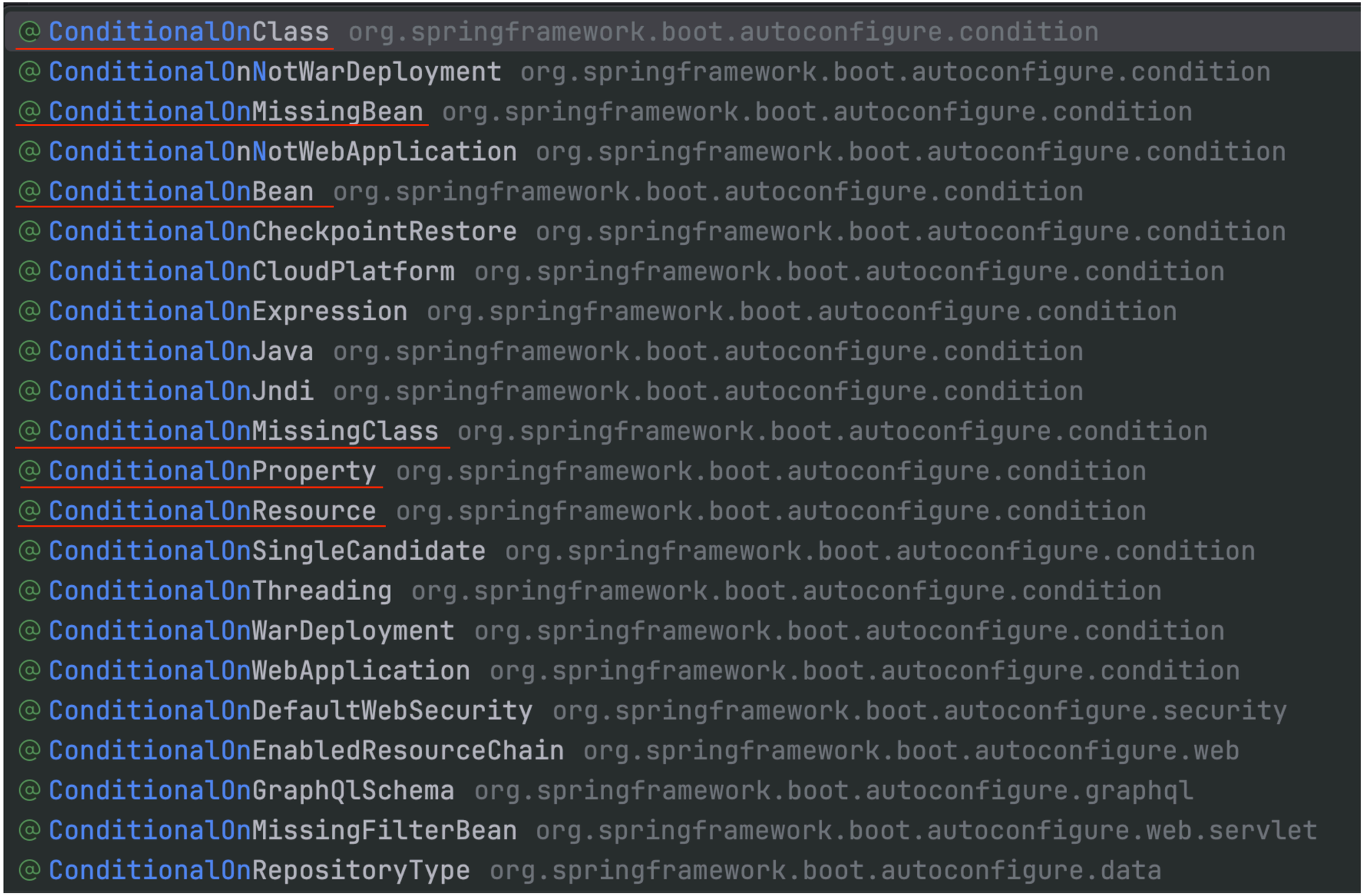
https://docs.spring.io/spring-boot/appendix/auto-configuration-classes/core.html

Demo

@AutoConfiguration

```
@Configuration(
        proxyBeanMethods = false
)
@AutoConfigureBefore
@AutoConfigureAfter
public @interface AutoConfiguration {
```

Make it conditional


@ConditionalOnClass org.springframework.boot.autoconfigure.condition
@ConditionalOnNotWarDeployment org.springframework.boot.autoconfigure.condition
@ConditionalOnMissingBean org.springframework.boot.autoconfigure.condition
@ConditionalOnNotWebApplication org.springframework.boot.autoconfigure.condition
@ConditionalOnBean org.springframework.boot.autoconfigure.condition
@ConditionalOnCheckpointRestore org.springframework.boot.autoconfigure.condition
@ConditionalOnCloudPlatform org.springframework.boot.autoconfigure.condition
@ConditionalOnExpression org.springframework.boot.autoconfigure.condition
@ConditionalOnJava org.springframework.boot.autoconfigure.condition
@ConditionalOnJndi org.springframework.boot.autoconfigure.condition
@ConditionalOnMissingClass org.springframework.boot.autoconfigure.condition
@ConditionalOnProperty org.springframework.boot.autoconfigure.condition
@ConditionalOnResource org.springframework.boot.autoconfigure.condition
@ConditionalOnSingleCandidate org.springframework.boot.autoconfigure.condition
@ConditionalOnThreading org.springframework.boot.autoconfigure.condition
@ConditionalOnWarDeployment org.springframework.boot.autoconfigure.condition
@ConditionalOnWebApplication org.springframework.boot.autoconfigure.condition
@ConditionalOnDefaultWebSecurity org.springframework.boot.autoconfigure.security
@ConditionalOnEnabledResourceChain org.springframework.boot.autoconfigure.web
@ConditionalOnGraphQlSchema org.springframework.boot.autoconfigure.graphql
@ConditionalOnMissingFilterBean org.springframework.boot.autoconfigure.web.servlet
@ConditionalOnRepositoryType org.springframework.boot.autoconfigure.data

@ConditionalOnClass

@ConditionalOnMissingClass

@ConditionalOnBean

@ConditionalOnMissingBean

@ConditionalProperty

@ConditionalResource

Make it configurable

Custom auto-configuration

- create auto configuration class and annotate it with @AutoConfiguration and @ConditionalOn.. (if needed)
- register auto configuration class under src/main/resources in file META-INF/spring/ org.springframework.boot.autoconfigure.AutoConfiguration.imports
- register any other factories under META-INF/spring.factories

Show applied auto-configuration

Command-line:	- -debug
application.properties:	debug=true
Actuator Endpoint:	conditions

Demo

Common opinions about SpringBoot

Learning Curve

While SpringBoot makes many things easier, some developers feel ~~**overwhelmed**~~ *embrace* by the number of features and options available.

Understanding the ~~**magic**~~ behind **auto-configuration** and embedded servers can be *interesting and fun*.

Questions?

Resources

Slide content was partially inspired by ChatGPT answers to my rather unspecific questions

Talk content heavily inspired by (a must see, despite it's age): https://youtu.be/uof5h-j0IeE?si=EIXoeESG4A2i0jIf

Sample repository: https://github.com/bkratz/spring-magic

Thank you