Spring Boot

ANIL JOSEPH

Agenda

- Introduction to Spring Boot
- Overview of Spring Configuration Annotations
- Creating Spring Boot Applications
- Maven dependencies
- Building Spring MVC Applications
- Auto Configuration
- Deployment
- Configuring Security

Spring

- A light-weight enterprise framework
- Simplified framework for creating components based on POJO
  - IOC, AOP, Transaction Management, Security, and many more
- Based on configuration
- Annotations based configuration introduced with Spring 2.5
- Spring 3.0 and 4.0 enhanced Java configuration
  - @ComponentScan, @Profile, @Conditional etc.
- Configurations can be tedious and time consuming

Spring Boot

- Provide a radically faster and widely accessible getting started experience for all Spring development.
- Automatically configure Spring whenever possible
- Opinionated(out of the box)
  - Provides default configuration
- Customizable when requirements diverge from the defaults
- Provide a range of non-functional features
  - Security, Metrics, externalized configuration etc.
- Creates Standalone Applications
- Embed Tomcat, Jetty directly, facilitating deployments on to the cloud
- Based on Maven and Gradle
- No requirement for XML configuration
- No code generation

Getting Started

- Spring CLI
  - A command line tool
  - Allows Groovy scripts
  - Very less boilerplate code
- Spring Initializer(https://start.spring.io/)
  - A web interface to generate the Maven or Gradle projects
- Eclipse and IntelliJ plugins
  - Provide templates to create spring boot applications

Spring Boot Application

- @SpringBootApplication
  - The annotation is used on the main class of the application
  - Its an alternative to @Configuration, @ComponentScan & @EnableAutoConfiguration
- @Configuration
  - Indicates that a class declares on or more @Bean methods.
  - @Configuration classes are typically bootstrapped using
    - AnnotationConfigApplicationContext
    - AnnotationConfigWebApplicationContext
- @ComponentScan
  - Provides support parallel with Spring XML's <context:component-scan> element.

Spring Boot Application

- @EnableAutoConfiguration
  - Enable auto-configuration of the Spring Application Context
  - Automatically loads all the beans required by the application
- SpringApplication class
  - The SpringApplication class provides a convenient way to **bootstrap** a Spring application that will be started from a main() method
  - Will create an ***AnnotationConfigApplicationContext*** or ***AnnotationConfigEmbeddedWebApplicationContext***
  - Provides a set of events and listeners

Auto Configuration
Excluding Auto Configurations

```
@EnableAutoConfiguration(exclude = { SomeAutoConfig.class })
public class Application {}
```

```
@EnableAutoConfiguration(excludeName = { “a.b.SomeAutoConfig” })
public class Application {}
```

```
@SpringBootApplication(exclude = { SomeAutoConfig.class })
public class Application {}
```

application.properties

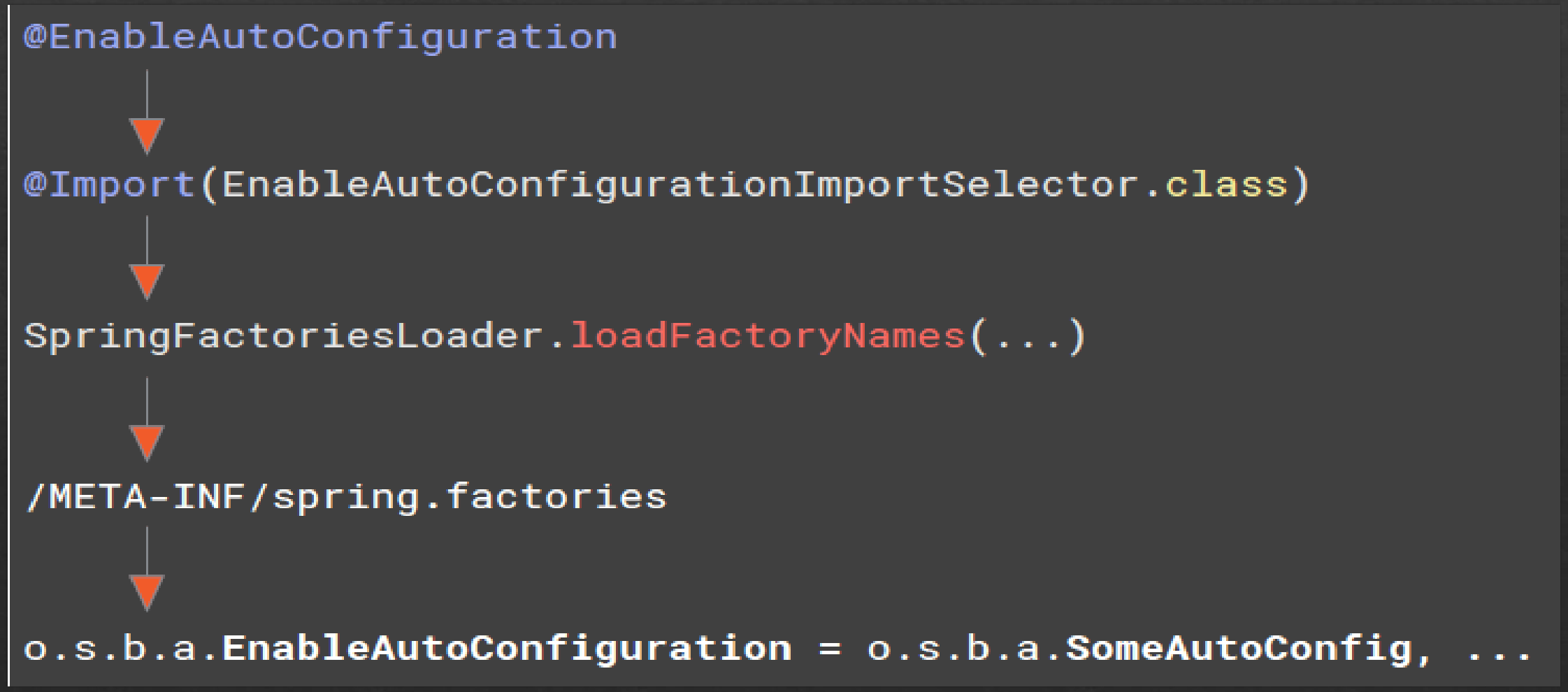
```
spring.autoconfigure.exclude = my.company.SomeAutoConfig
```

Auto Configuration
Reconfiguring Auto Configurations

spring.dao.exceptiontranslation.enabled = false

spring.mvc.favicon.enabled = false

Auto Configuration



Configuration Properties

- Properties can be in .properties files or .yml files
  - properties keys and values are Strings
  - .yml keys are Strings and values are their respective type
- Typesafe Configuration
  - Annotate with @ConfigurationProperties
  - Define getters & setters (JavaBean Spec)
  - Annotate with @Component
  - Can also use @EnableConfigurationProperties
  - Vaidate with JSR 303
- @ConfigurationProperties
  - turns all of your application configuration into typesafe POJOs

Resolving Configuration

- Command Line Arguments
- StandardServletEnvironment
- applicationProperties variants
  - Look for profile-specific configuration 1st
    - application-{profile}.properties
    - application-{profile}.yml
  - Look for generic configuration 2nd
    - application.properties / application.yml
- @PropertySource
  - @PropertySource("/some/path/foo.properties")