Spring Boot

An Introduction

Marten Deinum

Spring Boot

An Introduction: Objectives

- Explain what Spring Boot is
- Explain how Spring Boot simplifies application development
- Explain Spring Boot Features (and how to use them)

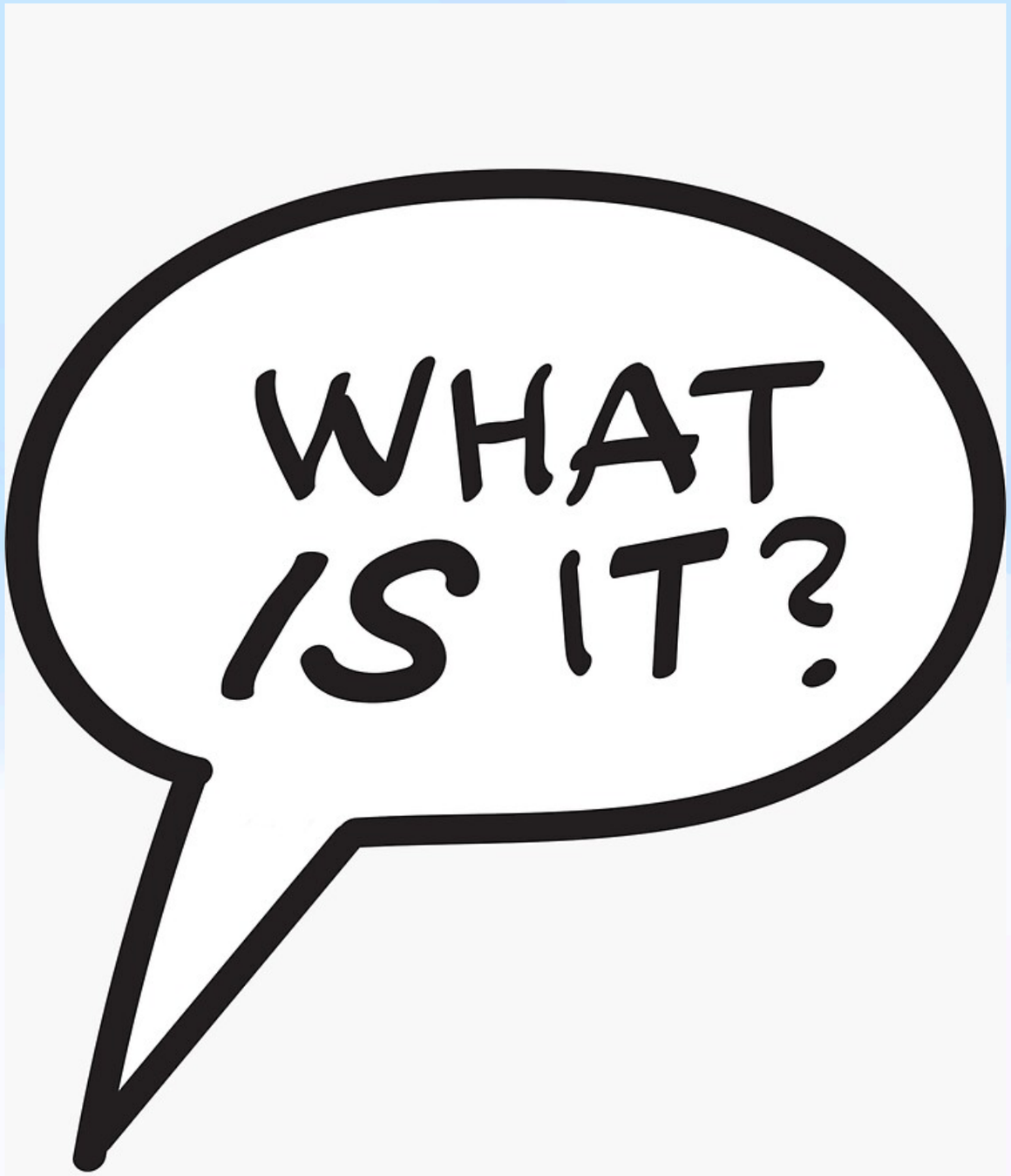
Spring Boot

An Introduction: Topics

- What is Spring Boot
- Features
- Getting Started
- Overview

Spring Boot

What is Spring Boot



Spring Boot helps you to create stand-alone, production-grade Spring-based applications that you can run. We take an opinionated view of the Spring platform and third-party libraries, so that you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

Spring Boot Reference Guide

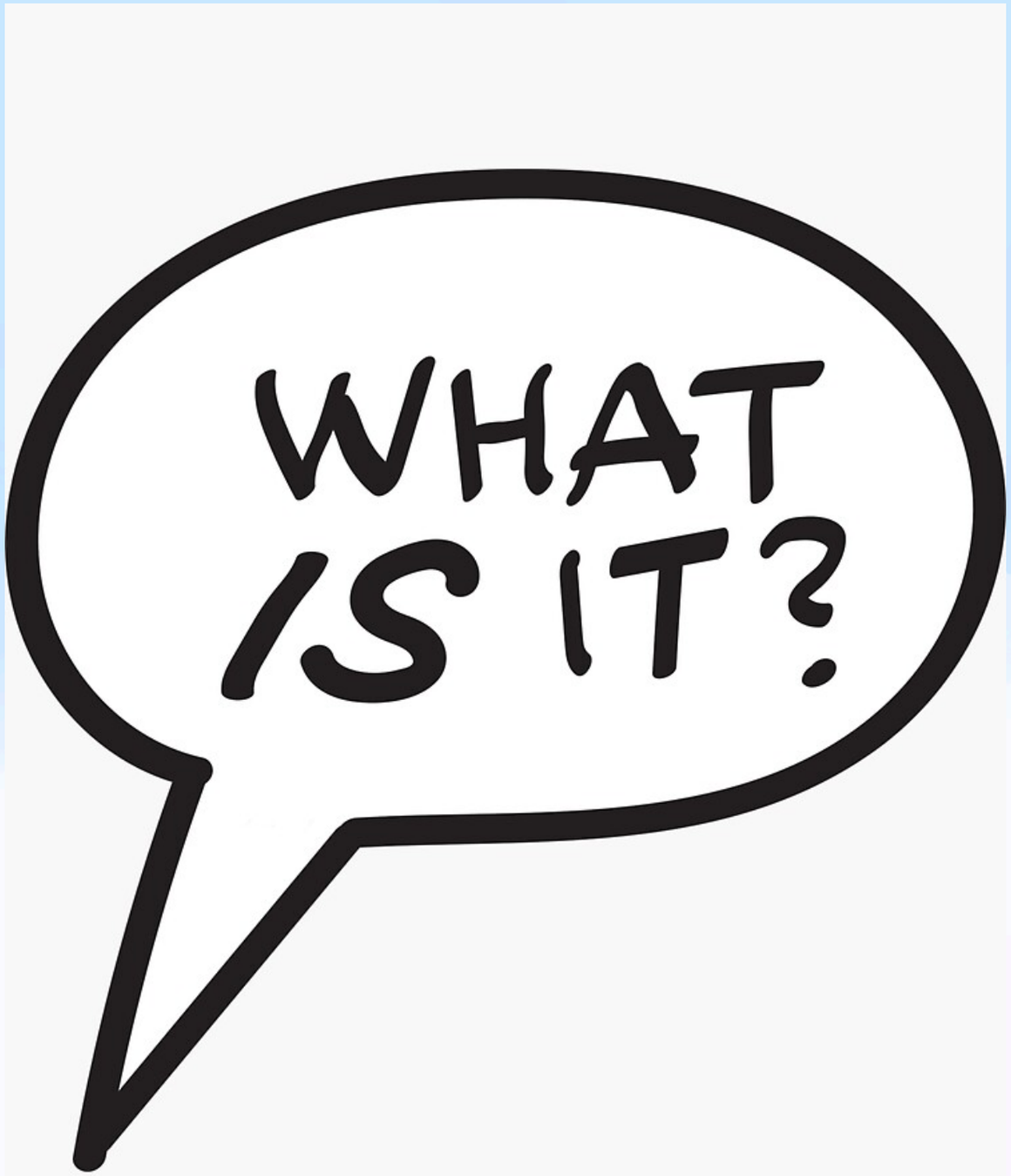
Spring Boot

What is Spring Boot

- Takes “opinionated” view of the Spring Portfolio **and** 3rd party libraries
- Support for different application types
  - Web
  - Batch
  - CLI
- Handles low-level configuration in a predictable way.

Spring Boot

Why use Spring Boot



Spring Boot

Why use Spring Boot

- Getting started quickly for all Spring based development
- Opinionated setup but can (and will) get out of your way quickly
- Large range of non-functional features depending on your needs
    - Metrics, Health, Config, Container support, Embedded Server (and more)

Spring Boot
Features

Spring Boot

Features

- Dependency Management
- Auto-Configuration
- Packaging / Runtime
- (Integration) Testing

Spring Boot

Features

- **Dependency Management**
- Auto-Configuration
- Packaging / Runtime
- (Integration) Testing

Spring Boot

An Introduction: Features - Dependency Management

- Java applications require a large number of dependencies.
  - How do you manage the correct and compatible versions
- Spring Boot Parent and/or Spring Boot Starters will help
  - Utilizes Maven or Gradle dependency management features
- Still able to do finer grained dependency management if wanted or needed
  - override dependency version
  - exclude dependency
  - define dependency explicitly

Spring Boot

An Introduction: Features - Spring Boot Parent

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.0</version>
</parent>
```

Defines properties for dependencies, like ${hibernate.version}

- Defines dependency versions
  - Utilizes Mavens' `dependencyManagement` feature
  - Uses `spring-boot-dependencies` internally
- Defines base configuration for Maven Plugins (incl. versions)
- Set up java version for compiler (override through `java.version`)

Spring Boot

An Introduction: Features - Spring Boot Starters

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

Resolves a lot of dependencies (about 18). **spring-boot*.jar, spring*.jar** but also **sfl4j** and **logback.**

Version can be omitted as it is defined in the parent!

- Brings in multiple, tested and coordinated dependencies.
  - including managing transitive dependencies

Spring Boot

An Introduction: Features - Spring Boot Starters

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

- Resolves common test libraries
  - Junit Jupiter
  - Spring Test Context (including Spring Boot Test extensions)
  - AssertJ (for assertions
  - Mockito
  - …

Spring Boot

An Introduction: Features - Spring Boot Starters

- Not required but recommended for getting started
- Managed dependencies for common Java frameworks
  - Pick the ones you need for your project (JPA, JDBC, Security)
- Examples:
  - `spring-boot-starter-data-jpa`
  - `spring-boot-starter-security`
  - `spring-boot-starter-web`
  - `spring-boot-starter-mail`

Spring Boot

Features

- Dependency Management
- **Auto-Configuration**
- Packaging / Runtime
- (Integration) Testing

Spring Boot

An Introduction: Features - Auto Configuration

- Spring Boot will create beans you need based on conditions
- `@EnableAutoConfiguration` is the secret annotation that enables auto-configuration.
- The `@SpringBootApplication` annotation includes the `@EnableAutoConfiguration` (and more)

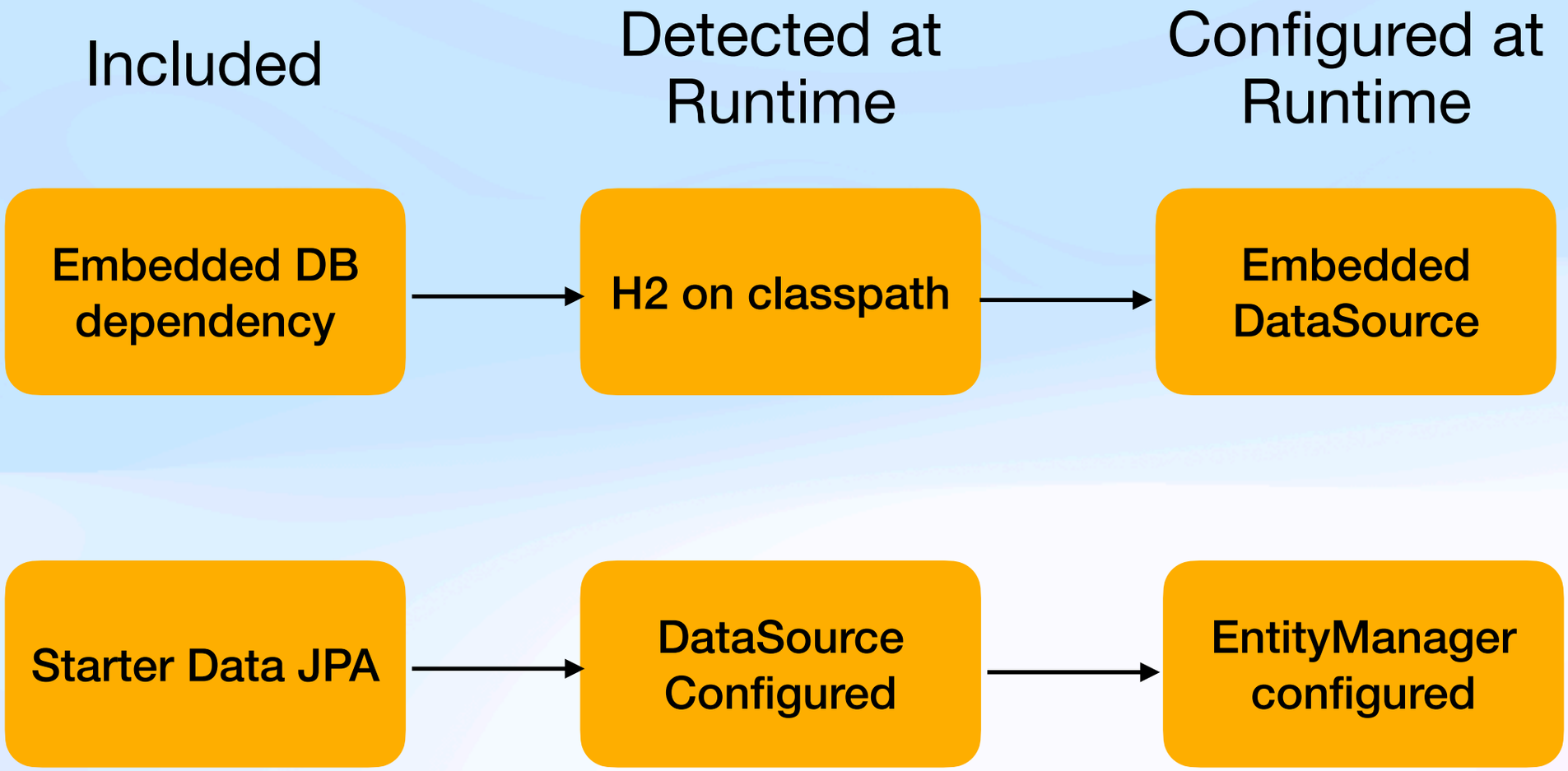
Spring Boot

An Introduction: Features - Auto Configuration

```
@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 {
```

Spring Boot

An Introduction: Features - Auto Configuration



Spring Boot

Features

- Dependency Management
- Auto-Configuration
- **Packaging / Runtime**
- (Integration) Testing

Spring Boot

An Introduction: Features - Packaging

- Spring Boot creates an executable jar file (aka “fat” jar) for your application
  - Can be run using `java -jar` ….
    - Which might not be the most efficient way to run it on a cloud platform.
- Luckily Spring Boot can help us build container images as well!

Spring Boot

An Introduction: Features - Packaging - Fat Jar

- Running `mvn package` will generate the “Fat” jar
  - The `spring-boot-maven-plugin` (or gradle variant) extends the package goal to do this
    - Creates 2 jars
      - `<your-app>.jar.original` (contains only your classes)
      - `<your-app>.jar` (the fat jar)

Spring Boot

An Introduction: Features - Packaging - Container

- Running `mvn spring-boot:build-image` will generate container image
    - Includes Java Runtime
    - Spring Boot libraries
    - Dependencies
    - Application Classes
    - Configuration
    - Creates entry point for starting

Spring Boot

Features

- Dependency Management
- Auto-Configuration
- Packaging / Runtime
- (Integration) Testing

Spring Boot

An Introduction: Features - Testing

- Spring Boot has several added testing features as compared to plain Spring (or at least makes it easier)
  - `@SpringBootTest` (for full integration tests)
  - Test slices (`@JdbcTest`, `@WebMvcTest` etc.)
  - Dedicated `TestExecutionListener`s to extend the default Spring Test Context features

Spring Boot
Getting Started

Spring Boot

Getting Started

- Need 3 files to get started
  - `pom.xml` or `build.gradle`
  - General configuration (`application.properties`)
  - An Application class (to launch everything)

Spring Boot

Getting Started: Kickstart Things

- Use Spring Initializr
  - Framework (and API) to create an quick start for Spring Boot projects.
    - https://start.spring.io
  - You can also create your own based on the sources.
    - https://github.com/spring-io/initializr
- Creates project for your Spring Boot project
  - Folder structure
  - Initial dependencies
  - Available through website, but also integrated into IDE (IntelliJ, Eclipse etc.)

Spring Boot

Summary

Spring Boot

Summary

- Spring Boot will kickstart your development
    - Setup application for you
    - Simpler dependency management
    - Auto-configuration with defaults (opinionated)
    - Builds executable jar (“fat” jar)
        - Or use a container
    - Provides enhanced testing support for partial of full blown integration testing

Spring Boot

Excercise: 30-spring-boot-introduction