Programming III
By: Reza Shalchian

You will learn how to

- Develop Spring Boot applications
- Leverage Hibernate/JPA for database access
- Create a Spring MVC app with Spring Boot
- Connect Spring Boot apps to a Database for CRUD development
- Use Thymeleaf for the UI

Java Development Environment

- We assume that you are already have experience with Java
  - OOP, classes, interfaces, inheritance, exception handling, collections
- You should have the following items already installed
  - Java Development Kit (JDK) - *Spring Boot 3 requires JDK 17 or higher*
  - *IntellijIdea*
  - *MySql*

Spring in a Nutshell

- Very popular framework for building Java applications
- Provides a large number of helper classes and annotations

Spring Boot Solution

- Make it easier to get started with Spring development
- Minimize the amount of manual configuration
    - Perform auto-configuration based on props files
- Help to resolve dependency conflicts (Maven or Gradle)
- Provide an embedded HTTP server so you can get started quickly
    - Tomcat, Jetty, Undertow

Spring Boot and Spring

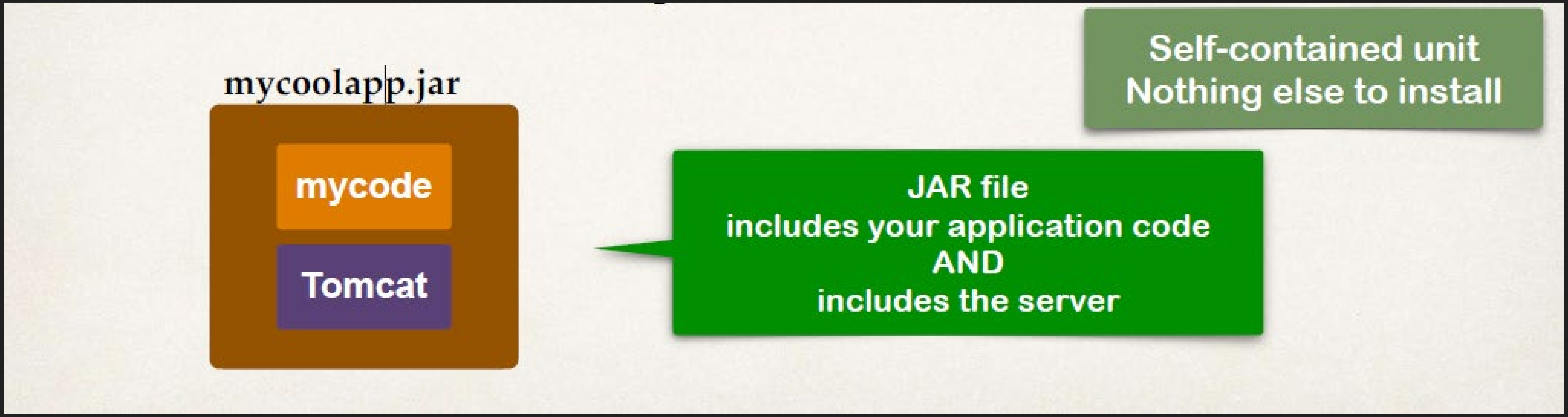
- Spring Boot uses Spring behind the scenes
- Spring Boot simply makes it easier to use Spring

Spring Initializr

- Quickly create a starter Spring Boot project -> **http://start.spring.io**
- Select your dependencies
- Creates a Maven/Gradle project
- Import the project into your IDE
    - Eclipse, IntelliJ, NetBeans

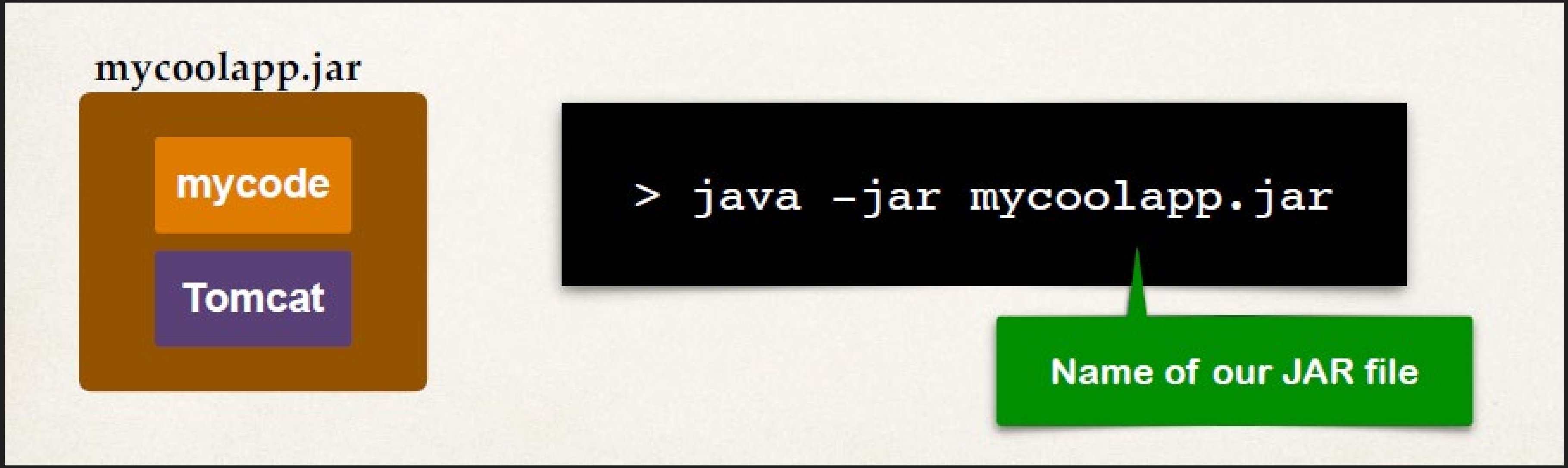
Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
- No need to install a server separately



Running Spring Boot Apps

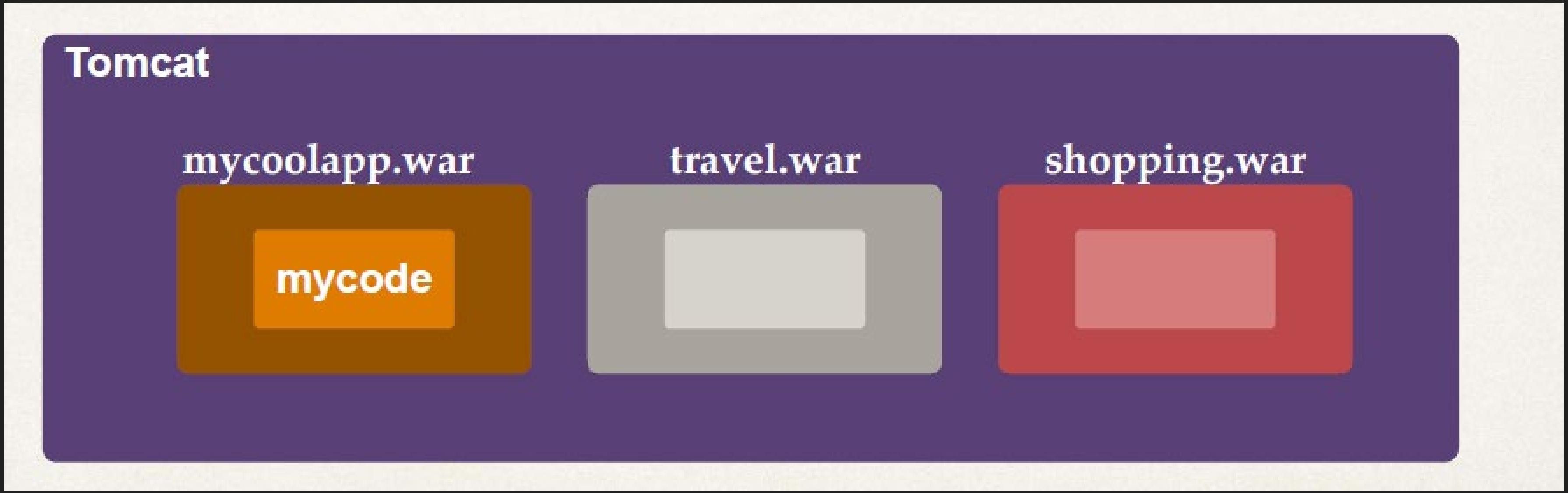
- Spring Boot apps can be run standalone (includes embedded server)
- Run the Spring Boot app from the IDE or command-line



```
> java -jar mycoolapp.jar
```

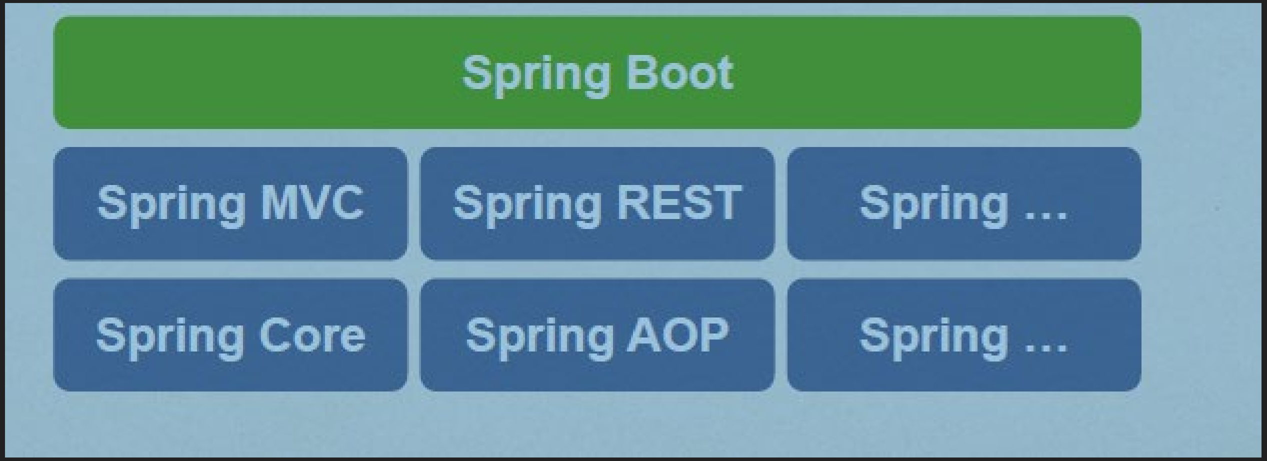
Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way
- Deploy **Web Application Archive (WAR) file** to an external server:
  - Tomcat, JBoss, WebSphere etc



Some FAQ

- Does Spring Boot replace Spring MVC, Spring REST etc …?
  - No. Instead, Spring Boot actually uses those technologies



- Does Spring Boot run code faster than regular Spring code?
  - No.
  - Behind the scenes, Spring Boot uses same code of Spring Framework
  - Remember, Spring Boot is about making it easier to get started
    - Minimizing configuration etc …

- Do I need a special IDE for Spring Boot?
  - No.
  - You can use any IDE for Spring Boot apps … even use plain text editor
  - The Spring team provides free *Spring Tool Suite (STS)* [IDE plugins]
  - Some IDEs provide fancy Spring tooling support

Maven

- When building your Java project, you may need additional JAR files
- For example: Spring, Hibernate, Commons Logging, JSON etc…
- One approach is to download the JAR files from each project web site
- Manually add the JAR files to your build path / classpath

Maven Solution

- Tell Maven the projects you are working with (dependencies)
  - Spring, Hibernate etc ….
- Maven will go out and download the JAR files for those projects for you
- And Maven will make those JAR files available during compile/run

Development Process

- Configure our project at Spring Initializr website
- Download the zip file
- Unzip the file
- Import the project into our IDE

Spring Framework Overview

- Why Spring?
  - Simplify Java Enterprise Development

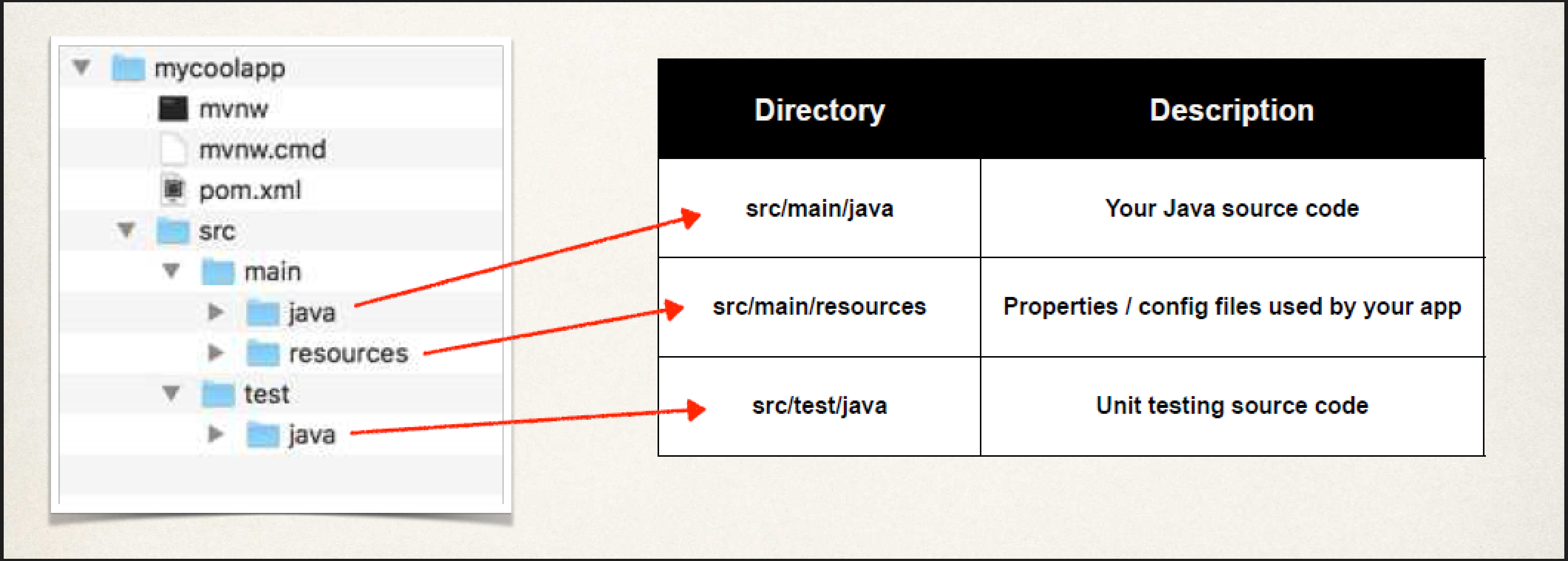
Goals of Spring

- Lightweight development with Java POJOs (Plain-Old-Java-Objects)
- Dependency injection to promote loose coupling
- Declarative programming with Aspect-Oriented-Programming (AOP)
- Minimize boilerplate Java code

Spring Projects

- Additional Spring ***modules*** built-on top of the core Spring Framework
- Only use what you need
  - Spring Cloud, Spring Data
  - Spring Batch, Spring Security

Spring Boot Project Structure



```
mycoolapp
    mvnw
    mvnw.cmd
    pom.xml
  src
    main
      java
      resources
    test
      java
```

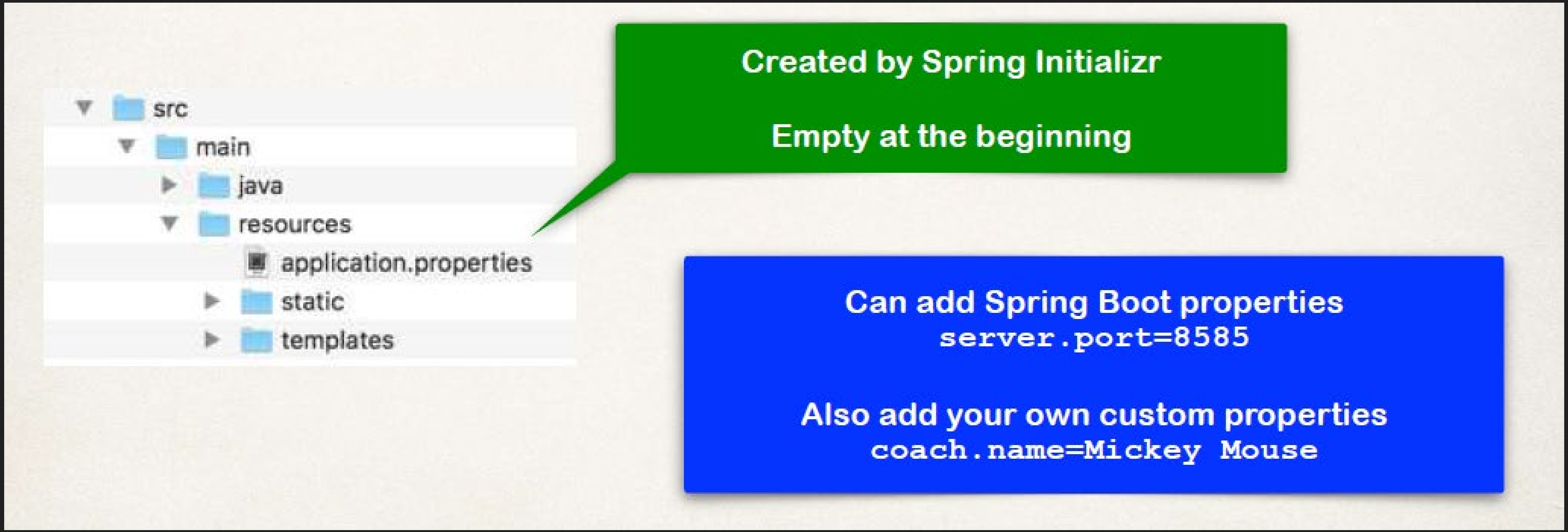
Directory	Description
src/main/java	Your Java source code
src/main/resources	Properties / config files used by your app
src/test/java	Unit testing source code

Maven POM file

Name	Description
Group ID	Name of company, group, or organization. Convention is to use reverse domain name: com.jac
Artifact ID	Name for this project: myapp
Version	A specific release version like: **1.0, 1.6, 2.0** … If project is under active development then: **1.0-SNAPSHOT**

Application Properties

- By default, Spring Boot will load properties from: **application.properties**



Created by Spring Initializr

Empty at the beginning

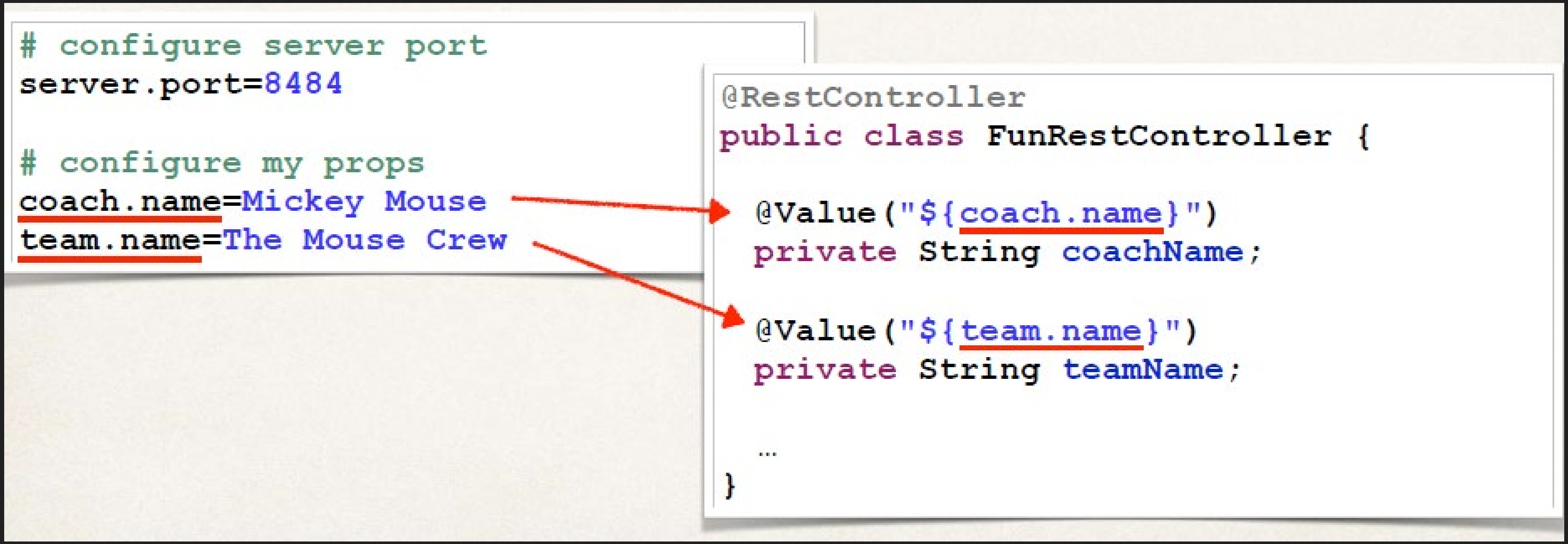
Can add Spring Boot properties

```
server.port=8585
```

Also add your own custom properties

```
coach.name=Mickey Mouse
```

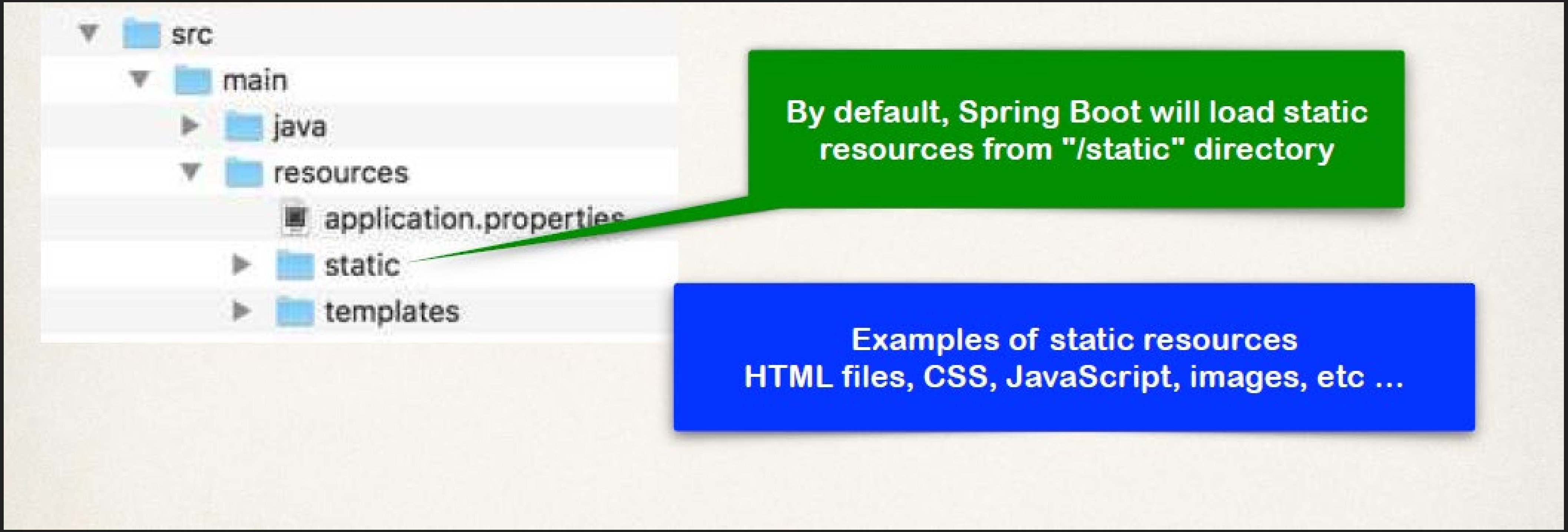
- Read data from: **application.properties**



```
# configure server port
server.port=8484

# configure my props
coach.name=Mickey Mouse
team.name=The Mouse Crew
```

```
@RestController
public class FunRestController {

  @Value("${coach.name}")
  private String coachName;

  @Value("${team.name}")
  private String teamName;

  ...
}
```

Static Content

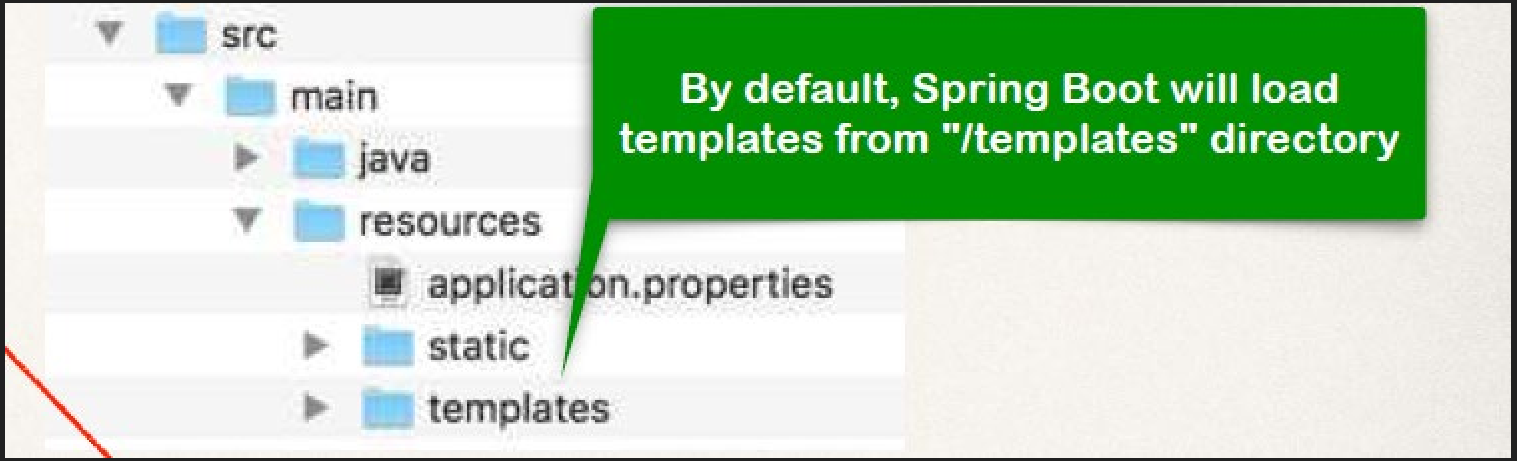


By default, Spring Boot will load static resources from "/static" directory

Examples of static resources
HTML files, CSS, JavaScript, images, etc ...

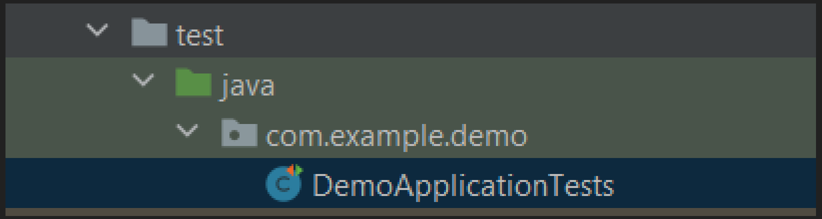
Templates

- Spring Boot includes auto-configuration for following template engines
  - FreeMarker
  - Thymeleaf
  - Mustache


src
main
java
resources
application.properties
static
templates
By default, Spring Boot will load templates from "/templates" directory

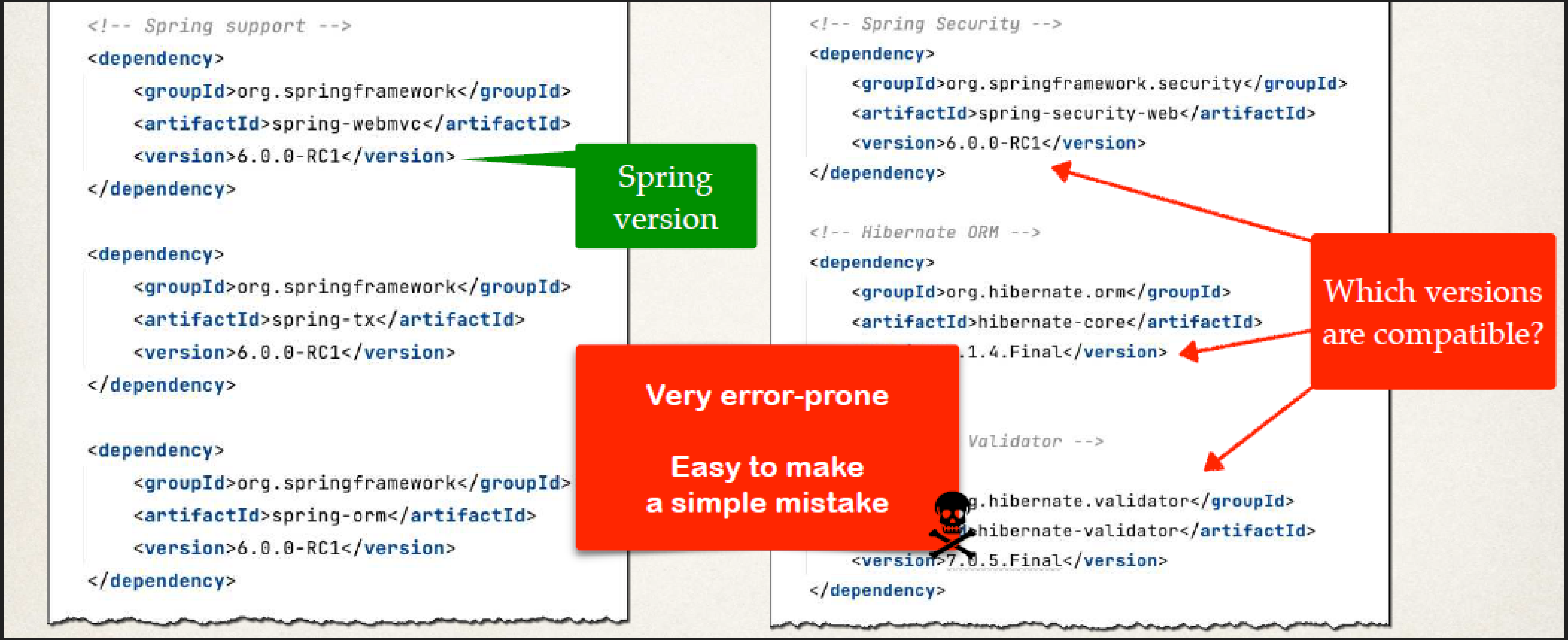
Unit Tests

- Springboot Unit test class Created by Spring Initializer
- We can add unit tests to the file



Spring Boot Starters

- **The Problem :** Building a Spring application is really HARD!!!
    - Which Maven dependencies do I need?



```xml
<!-- Spring support -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>6.0.0-RC1</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>6.0.0-RC1</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>6.0.0-RC1</version>
</dependency>
```

```xml
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>6.0.0-RC1</version>
</dependency>

<!-- Hibernate ORM -->
<dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-core</artifactId>
              .1.4.Final</version>

              Validator -->

              g.hibernate.validator</groupId>
              hibernate-validator</artifactId>
    <version>7.0.5.Final</version>
</dependency>
```

Spring version

Which versions are compatible?

Very error-prone

Easy to make a simple mistake

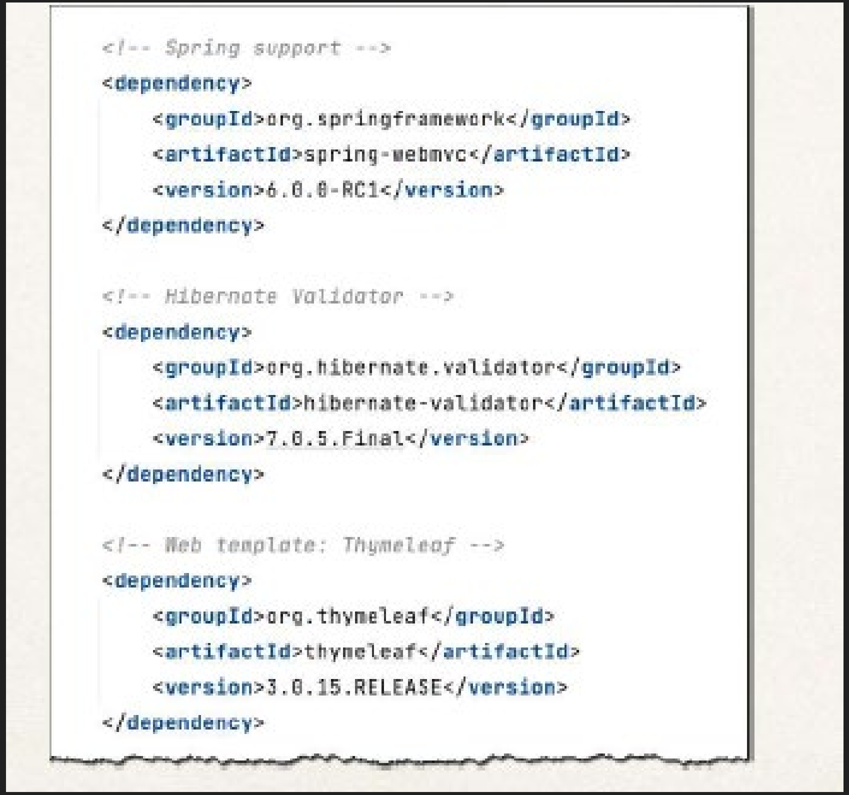
Why Is It So Hard?

- It would be great if there was a simple list of Maven dependencies
- Collected as a group of dependencies … one-stop shop
- So I don't have to search for each dependency

The Solution - Spring Boot Starters

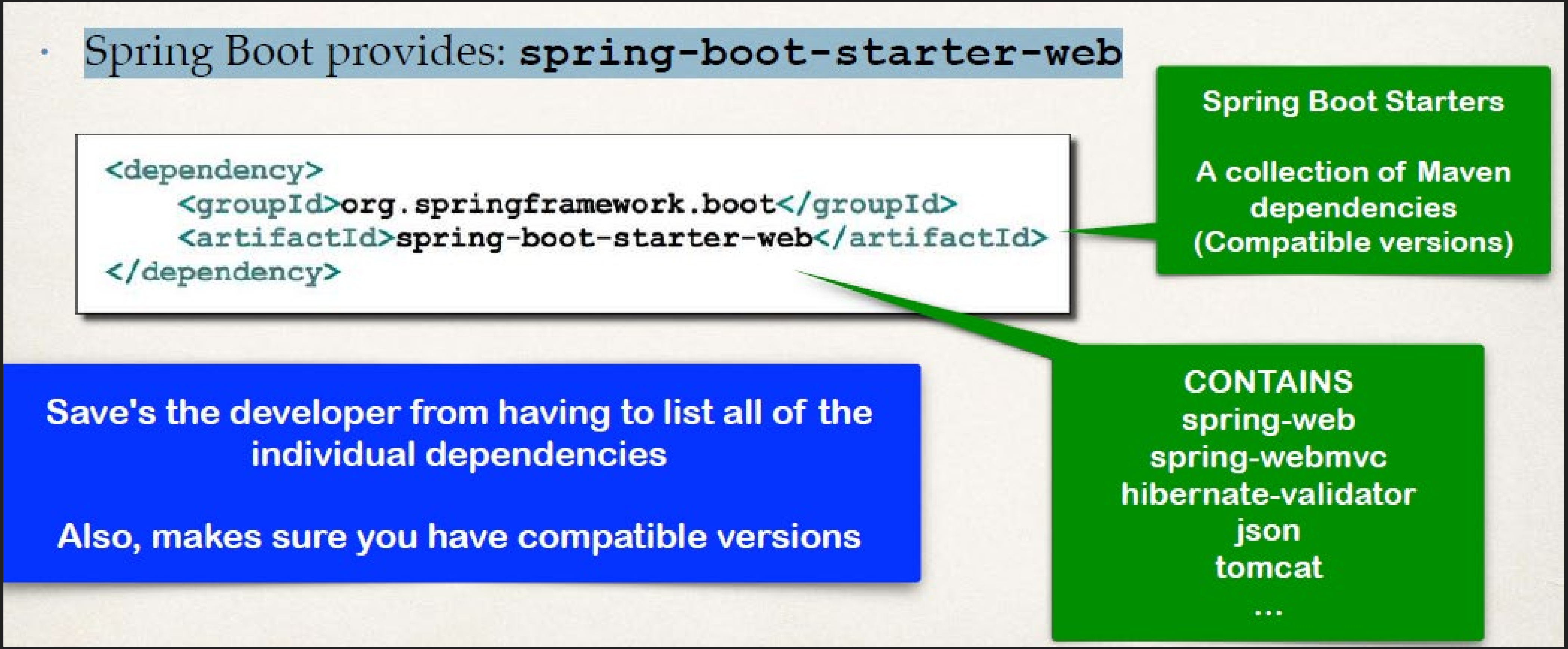
- A collection of dependencies grouped together
- Tested and verified by the Spring Development team
- Makes it much easier for the developer to get started with Spring
- Reduces the amount of Maven configuration

- For example, when building a Spring MVC app, you normally need



```
<!-- Spring support -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>6.0.0-RC1</version>
</dependency>

<!-- Hibernate Validator -->
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>7.0.5.Final</version>
</dependency>

<!-- Web template: Thymeleaf -->
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf</artifactId>
    <version>3.0.15.RELEASE</version>
</dependency>
```

Solution: Spring Boot Starter - Web



- Spring Boot provides: `spring-boot-starter-web`

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

Spring Boot Starters

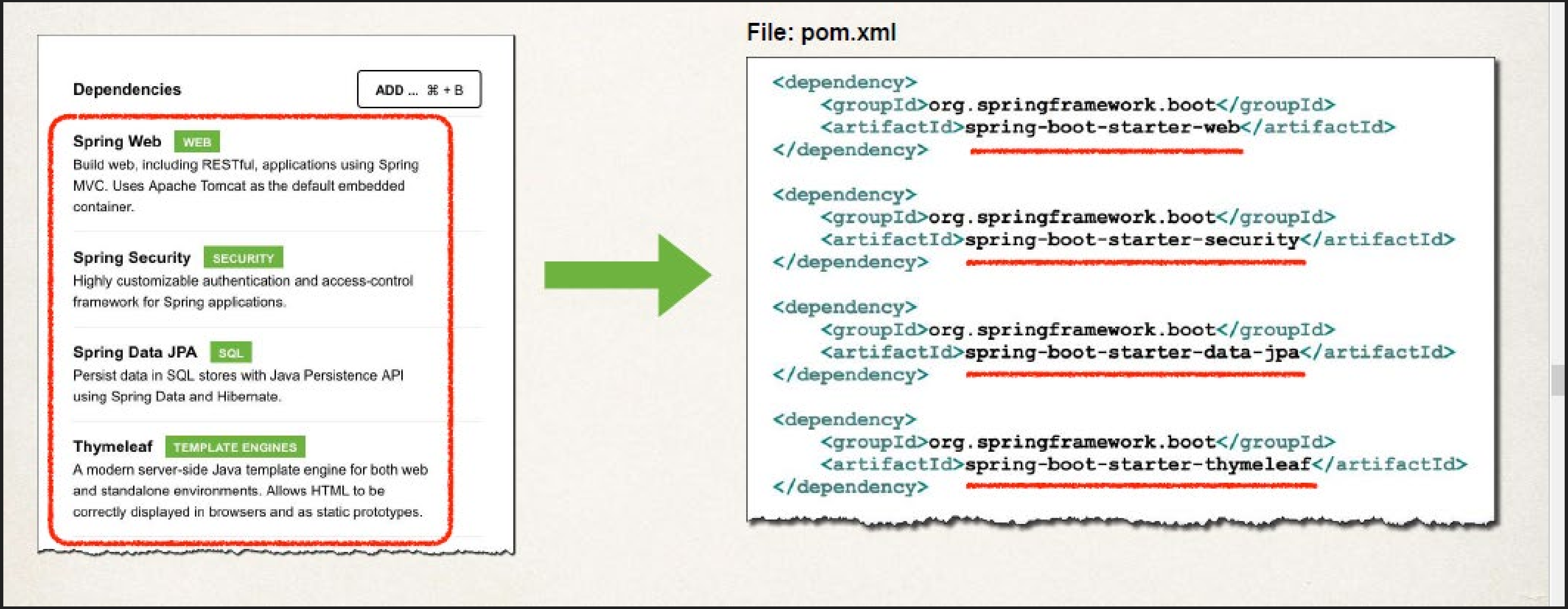
A collection of Maven dependencies (Compatible versions)

CONTAINS
spring-web
spring-webmvc
hibernate-validator
json
tomcat
...

Save's the developer from having to list all of the individual dependencies

Also, makes sure you have compatible versions

Spring Initializr



Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

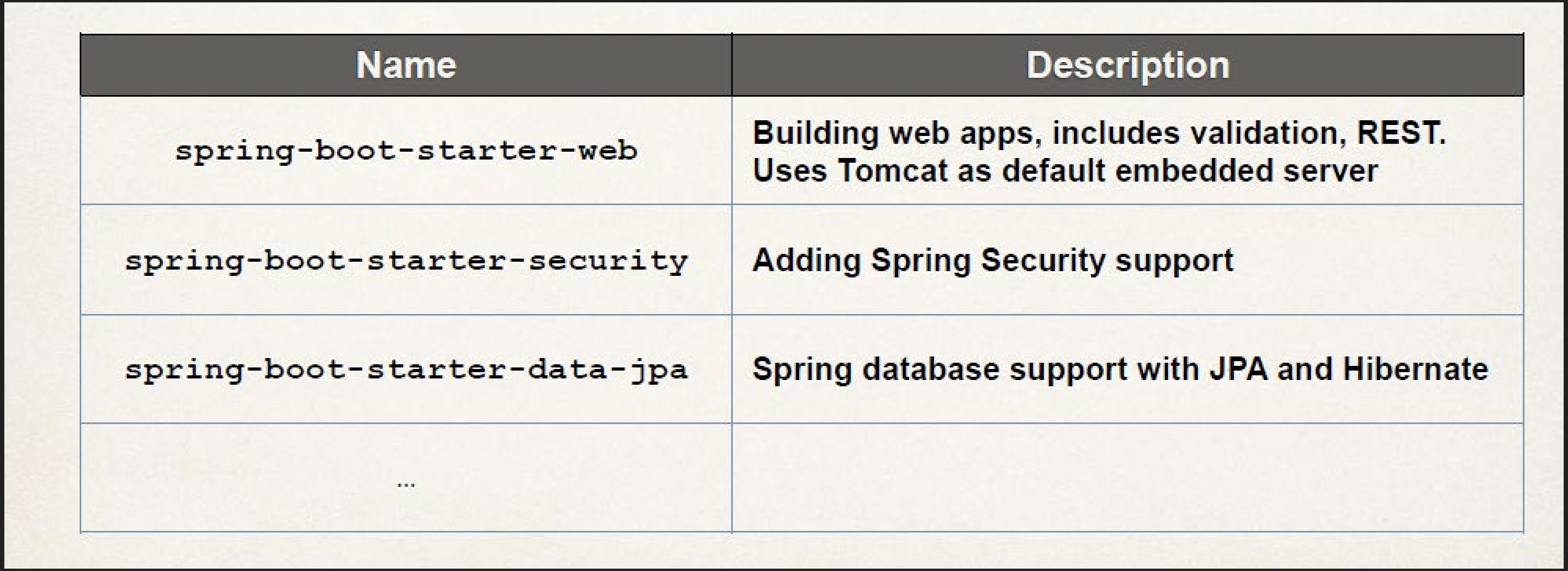
Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

File: pom.xml

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

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

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

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

Spring Boot Starters

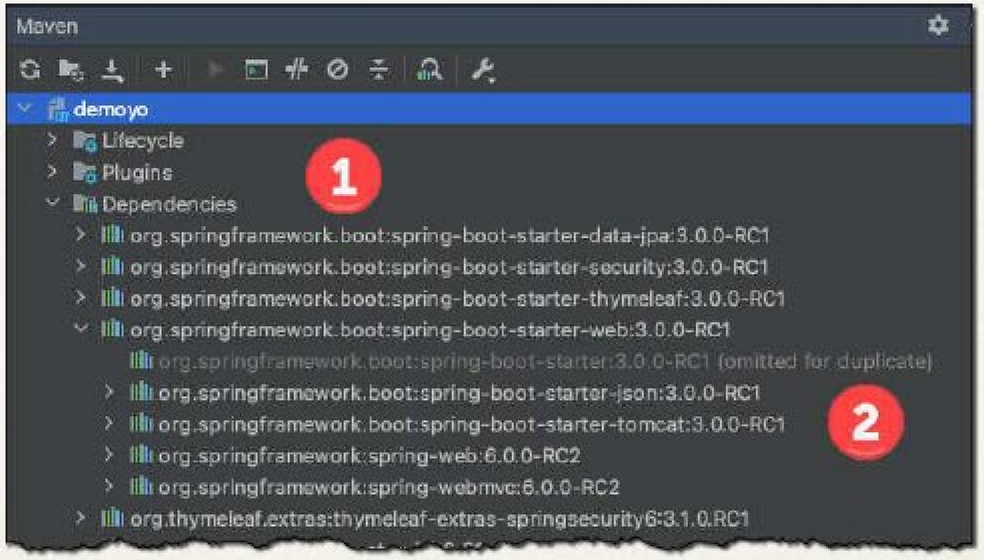
- There are 30+ Spring Boot Starters from the Spring Development team



Name	Description
`spring-boot-starter-web`	Building web apps, includes validation, REST. Uses Tomcat as default embedded server
`spring-boot-starter-security`	Adding Spring Security support
`spring-boot-starter-data-jpa`	Spring database support with JPA and Hibernate
...	

What Is In the Starter?

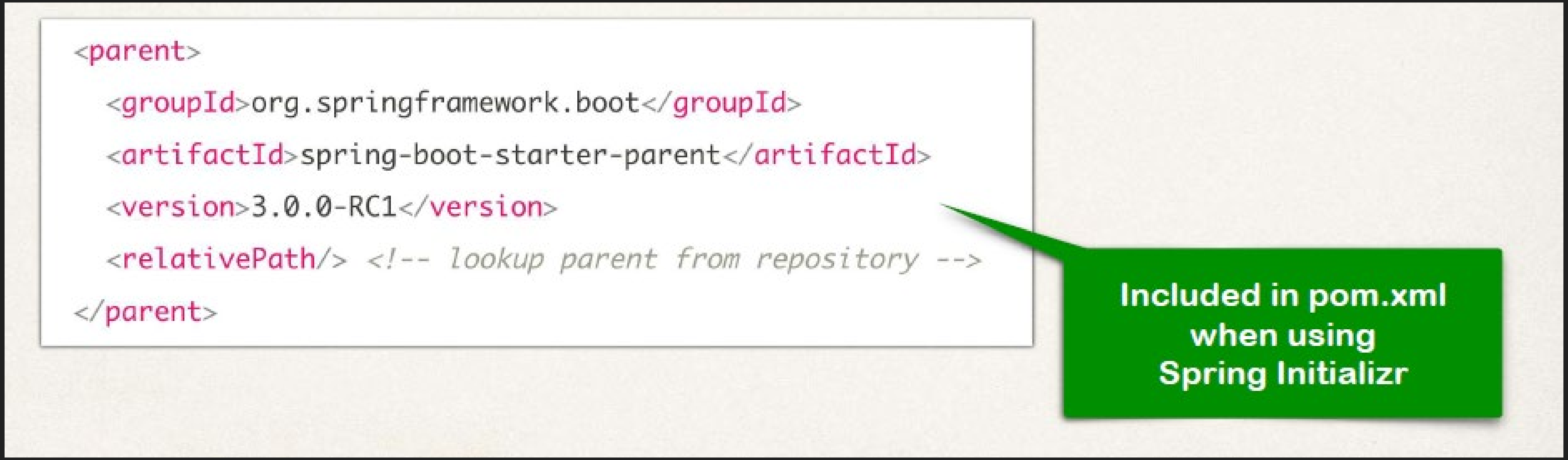
- For IntelliJ users
- Select **View > Tool Windows > Maven Projects > Dependencies**



```
Maven
demoyo
  Lifecycle
  Plugins
  Dependencies
    org.springframework.boot:spring-boot-starter-data-jpa:3.0.0-RC1
    org.springframework.boot:spring-boot-starter-security:3.0.0-RC1
    org.springframework.boot:spring-boot-starter-thymeleaf:3.0.0-RC1
    org.springframework.boot:spring-boot-starter-web:3.0.0-RC1
      org.springframework.boot:spring-boot-starter:3.0.0-RC1 (omitted for duplicate)
      org.springframework.boot:spring-boot-starter-json:3.0.0-RC1
      org.springframework.boot:spring-boot-starter-tomcat:3.0.0-RC1
      org.springframework:spring-web:6.0.0-RC2
      org.springframework:spring-webmvc:6.0.0-RC2
    org.thymeleaf.extras:thymeleaf-extras-springsecurity6:3.1.0.RC1
```

Spring Boot Starter Parent

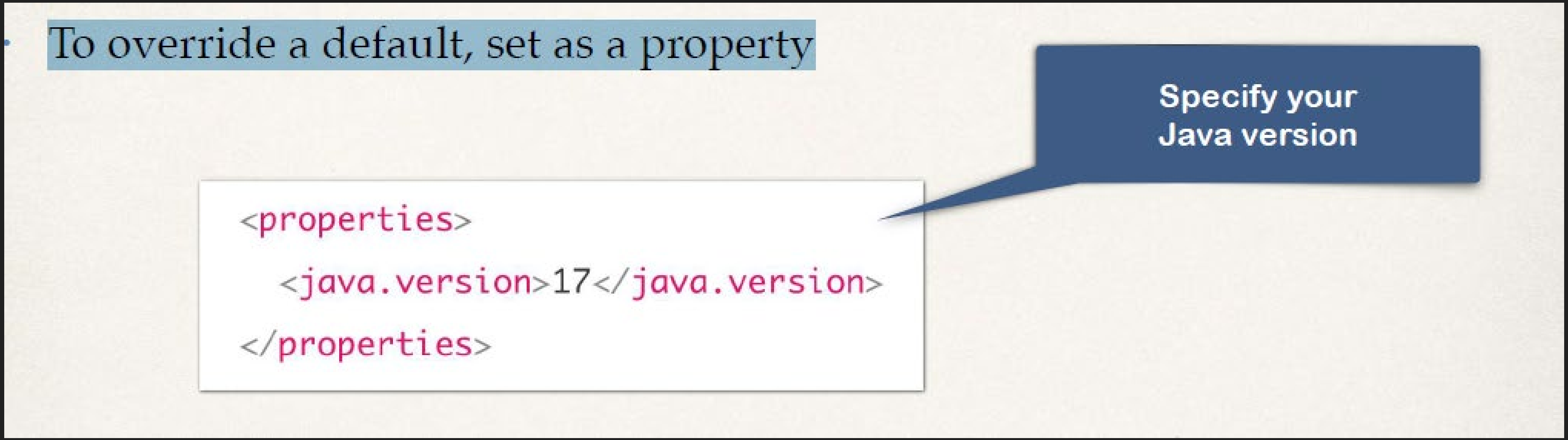
- Spring Boot provides a "Starter Parent“
- This is a special starter that provides Maven defaults



```xml
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.0.0-RC1</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

- Maven defaults defined in the Starter Parent
- Default compiler level
- UTF-8 source encoding
- *Others …*

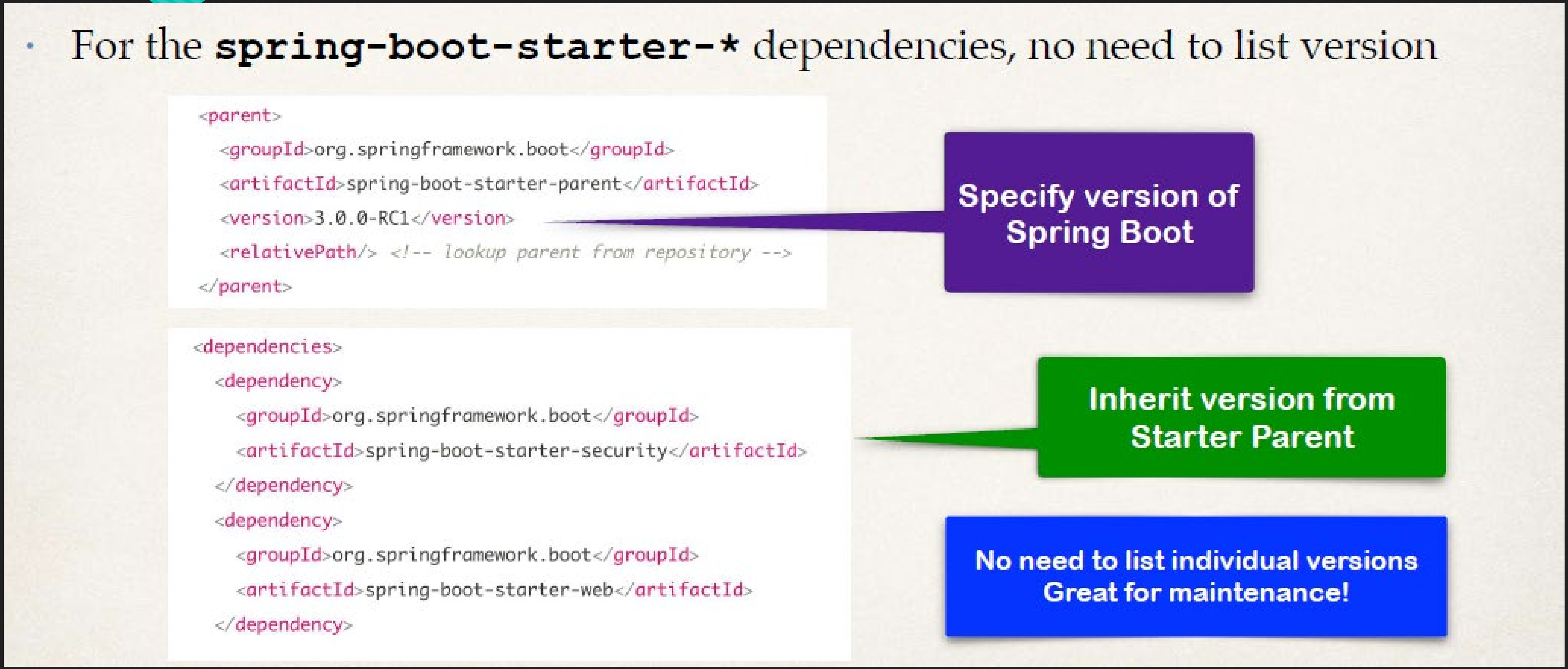
Spring Boot Starter Parent



- To override a default, set as a property

```
<properties>
  <java.version>17</java.version>
</properties>
```

Spring Boot Starter Parent



- For the **`spring-boot-starter-*`** dependencies, no need to list version

```xml
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.0.0-RC1</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

Specify version of Spring Boot

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

Inherit version from Starter Parent

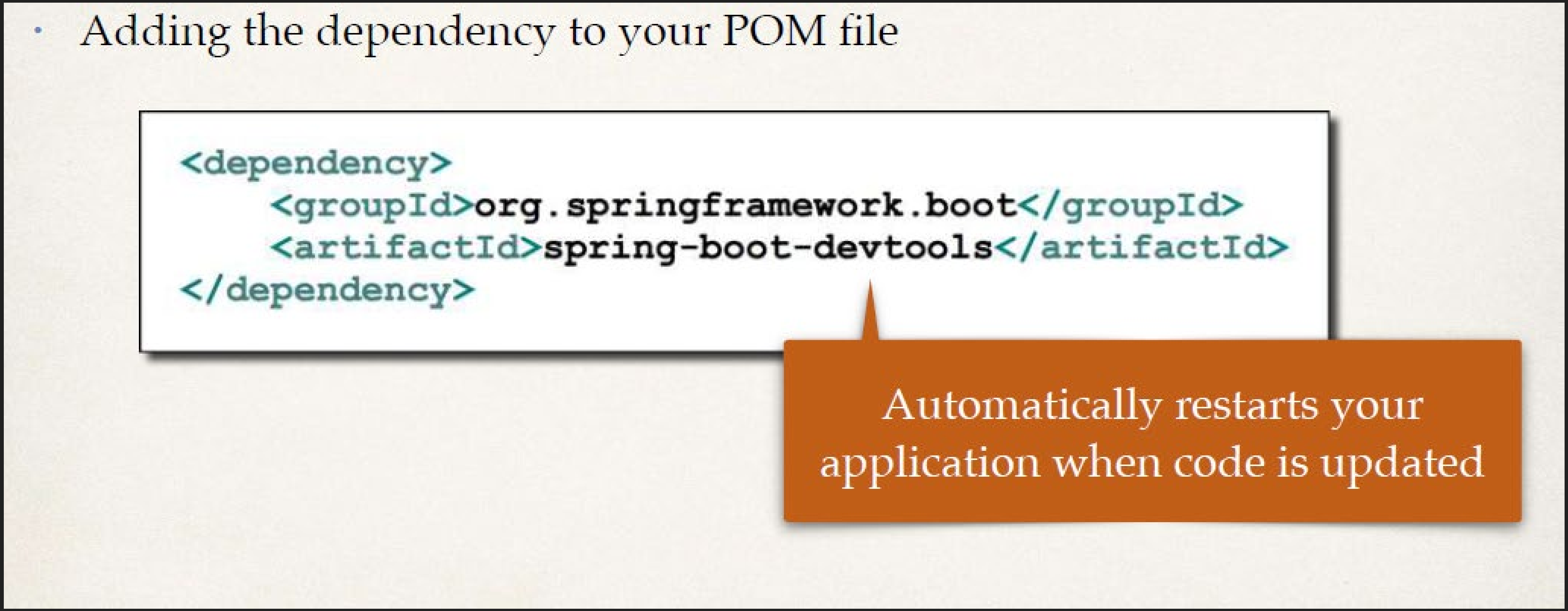
No need to list individual versions
Great for maintenance!

Spring Boot Dev Tools

- **The Problem**
  - When running Spring Boot applications
    - If you make changes to your source code
    - Then you have to manually restart your application :-(

- **spring-boot-devtools** to the rescue
- Automatically restarts your application when code is updated
- Simply add the dependency to your POM file
- No need to write additional code :-)

Spring Boot Dev Tools



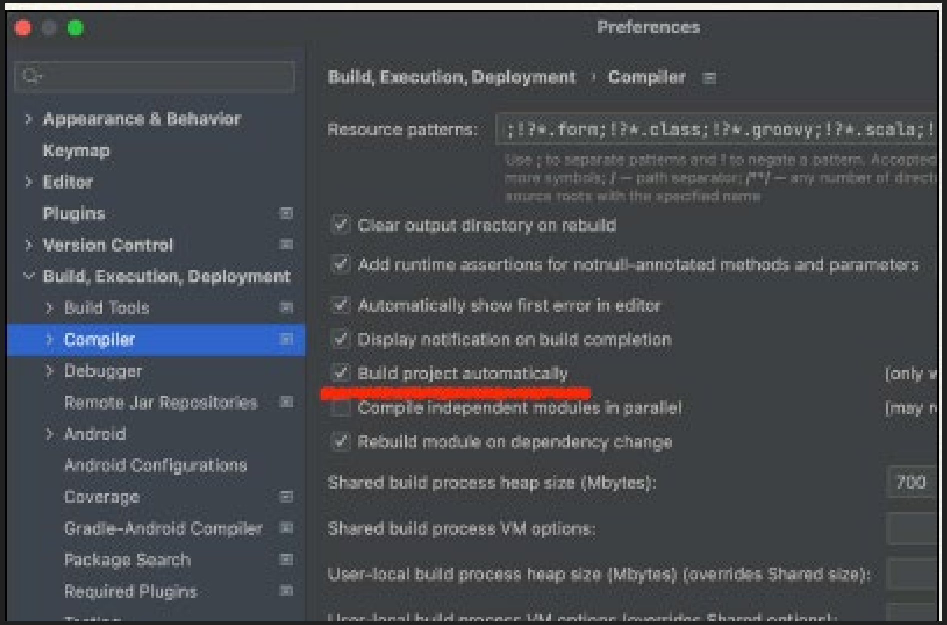
- Adding the dependency to your POM file

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

Automatically restarts your application when code is updated

IntelliJ Community Edition - DevTools

- IntelliJ Community Edition does not support DevTools by default
- Select: **Preferences > Build, Execution, Deployment > Compiler**
  - Check box: **Build project automatically**



Build, Execution, Deployment › Compiler

Resource patterns: `;!?*.form;!?*.class;!?*.groovy;!?*.scala;!`

Use ; to separate patterns and ! to negate a pattern. Accepted
more symbols; / — path separator; /**/ — any number of directi
source roots with the specified name

- [✓] Clear output directory on rebuild
- [✓] Add runtime assertions for notnull-annotated methods and parameters
- [✓] Automatically show first error in editor
- [✓] Display notification on build completion
- [✓] Build project automatically (only w
- [ ] Compile independent modules in parallel (may n
- [✓] Rebuild module on dependency change

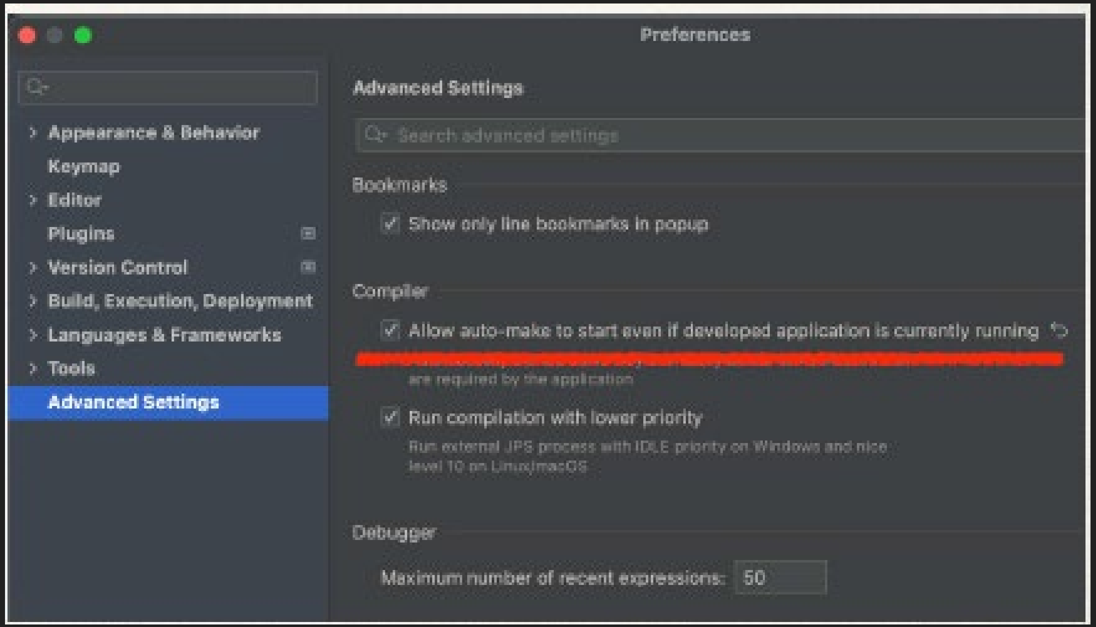
Shared build process heap size (Mbytes): 700

Shared build process VM options:

User-local build process heap size (Mbytes) (overrides Shared size):

IntelliJ Community Edition

- Additional setting
- Select: **Preferences > Advanced Settings**
  - Check box: **Allow auto-make to …**



Advanced Settings

Search advanced settings

Bookmarks

[✓] Show only line bookmarks in popup

Compiler

[✓] Allow auto-make to start even if developed application is currently running

[redacted]
are required by the application

[✓] Run compilation with lower priority

Run external JPS process with IDLE priority on Windows and nice level 10 on Linux/macOS

Debugger

Maximum number of recent expressions: 50

Spring Boot Actuator

- **Problem**
  - How can I monitor and manage my application?•
  - How can I check the application health?
  - How can I access application metrics?

Solution: Spring Boot Actuator

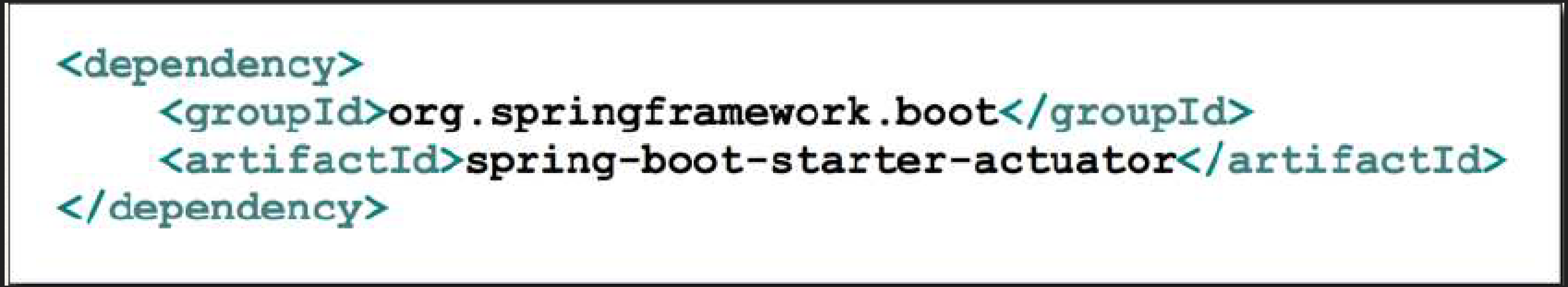
- Exposes endpoints to monitor and manage your application
- You easily get DevOps functionality out-of-the-box
- Simply add the dependency to your POM file
- REST endpoints are automatically added to your application

No need to write additional code!

You get new REST endpoints for FREE!

Spring Boot Actuator

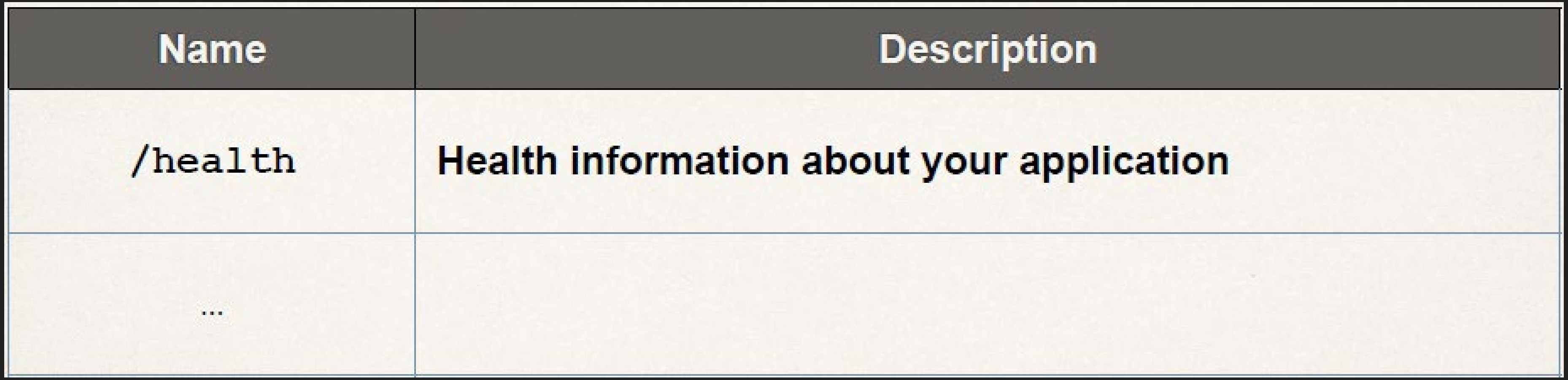
- Adding the dependency to your POM file



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

Spring Boot Actuator

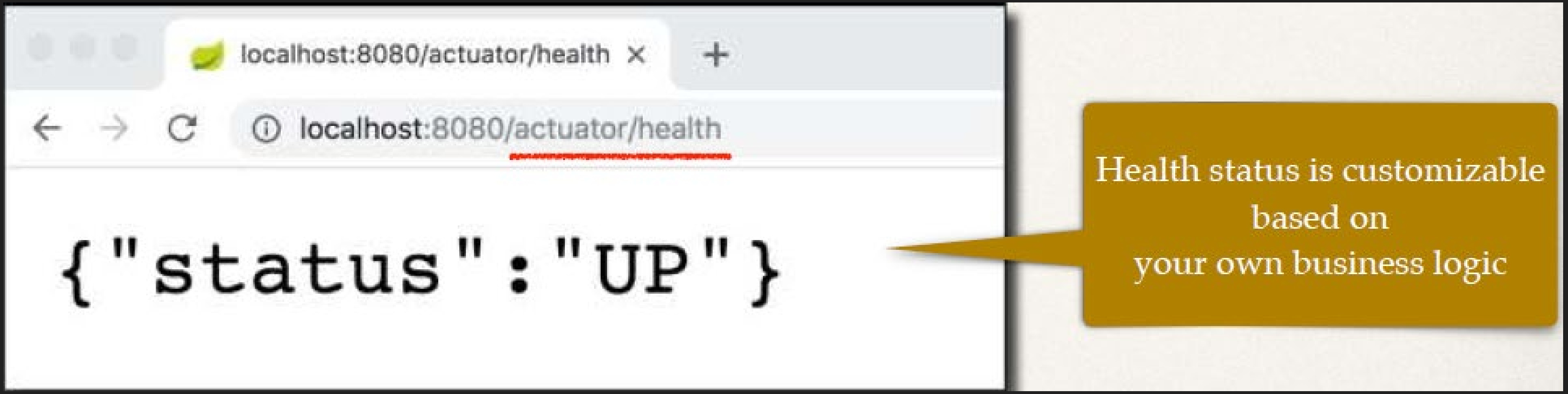
- Automatically exposes endpoints for metrics out-of-the-box
- Endpoints are prefixed with: **/actuator**



Name	Description
`/health`	Health information about your application
…	

Health Endpoint

- **/health** checks the status of your application
- Normally used by monitoring apps to see if your app is up or down

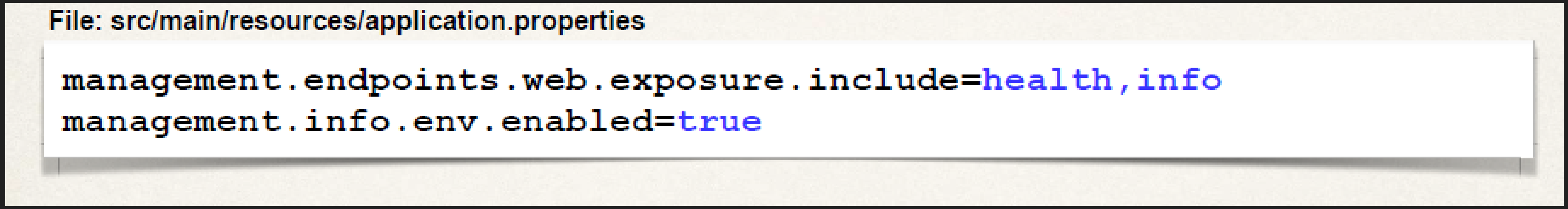

localhost:8080/actuator/health
localhost:8080/actuator/health

```
{"status":"UP"}
```

Health status is customizable
based on
your own business logic

Exposing Endpoints

- By default, only **/health** is exposed
- The **/info** endpoint can provide information about your application
- To expose **/info**

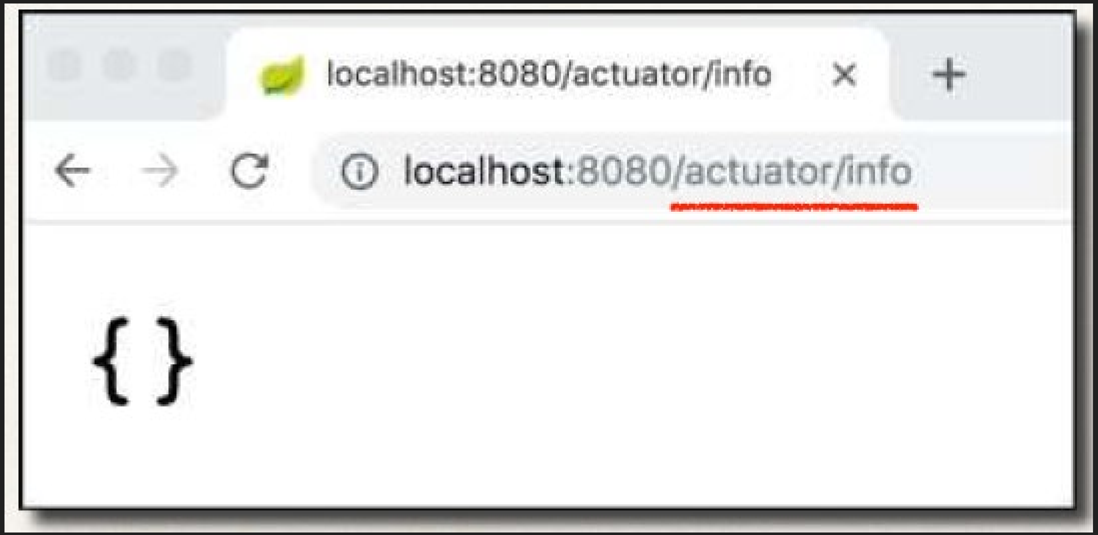


File: src/main/resources/application.properties

```
management.endpoints.web.exposure.include=health,info
management.info.env.enabled=true
```

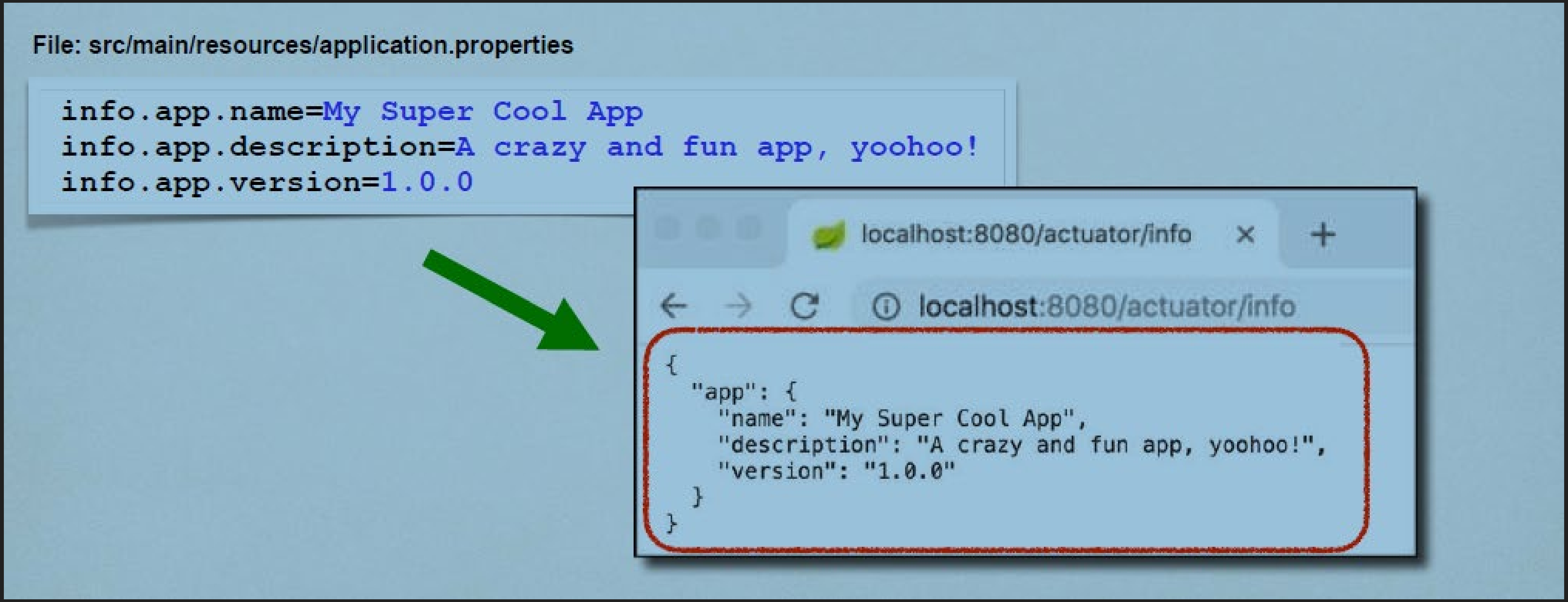
Info Endpoint

- **/info** gives information about your application
- Default is empty



```
{}
```

- • Update **application.properties** with your app info



File: src/main/resources/application.properties

```
info.app.name=My Super Cool App
info.app.description=A crazy and fun app, yoohoo!
info.app.version=1.0.0
```

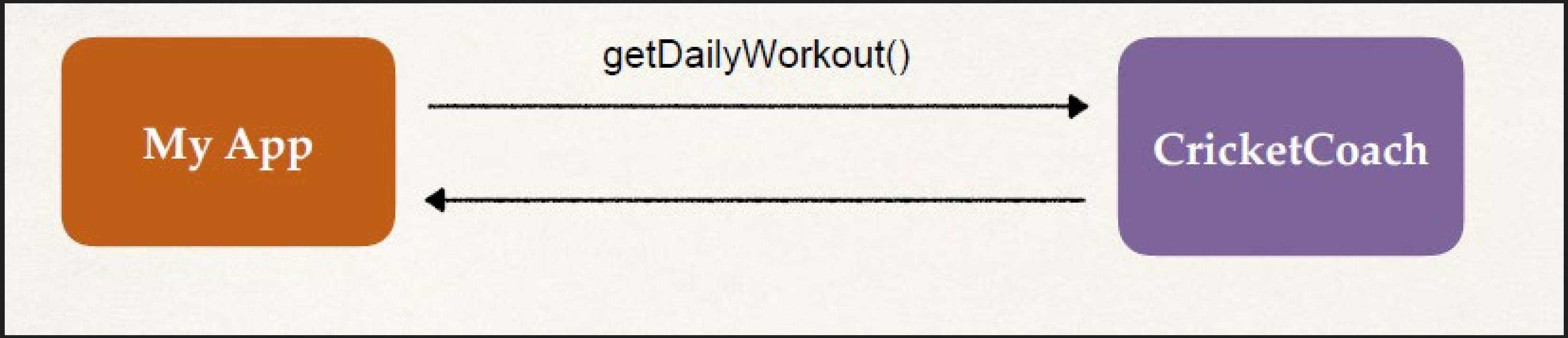
```
{
  "app": {
    "name": "My Super Cool App",
    "description": "A crazy and fun app, yoohoo!",
    "version": "1.0.0"
  }
}
```

Inversion of Control(IOC)

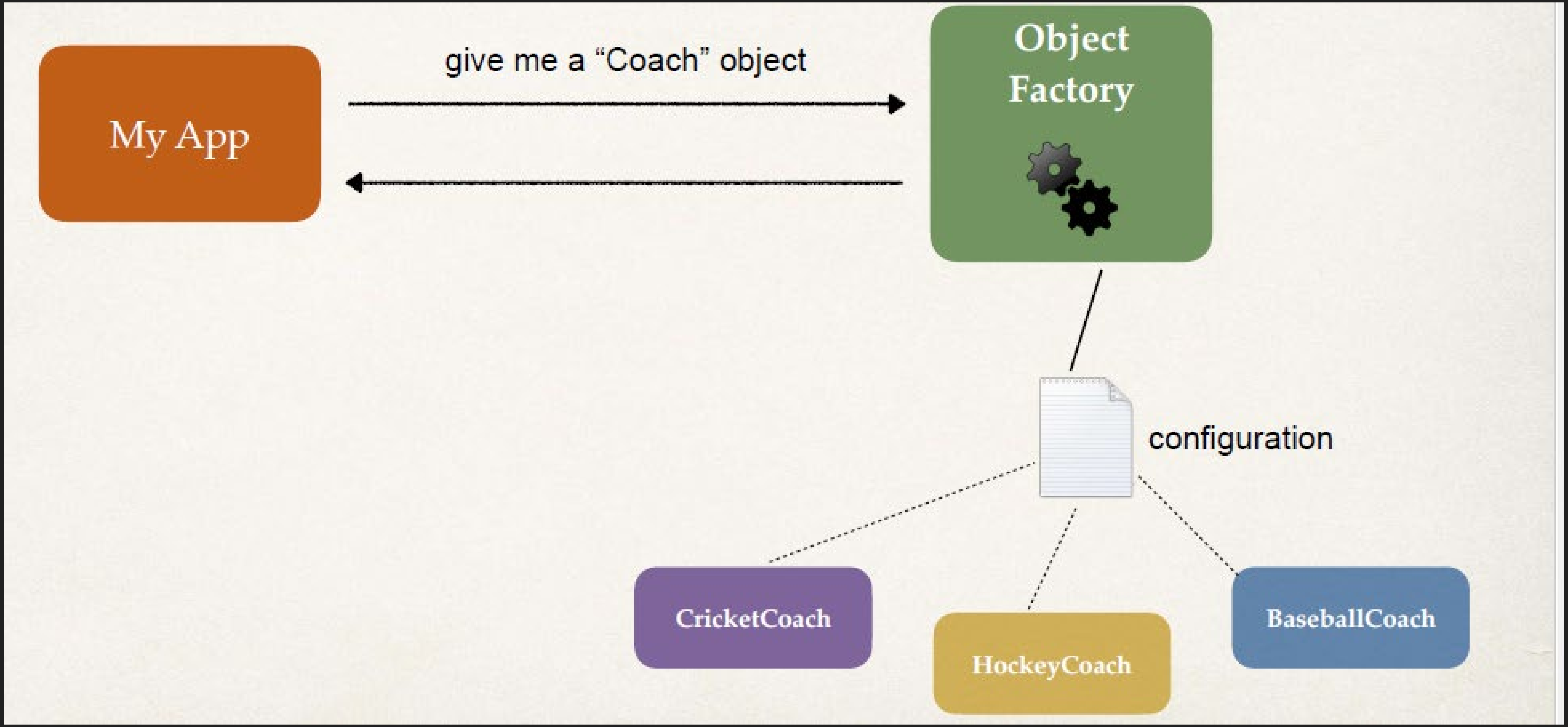
The approach of outsourcing the construction and management of objects.

Coding Scenario

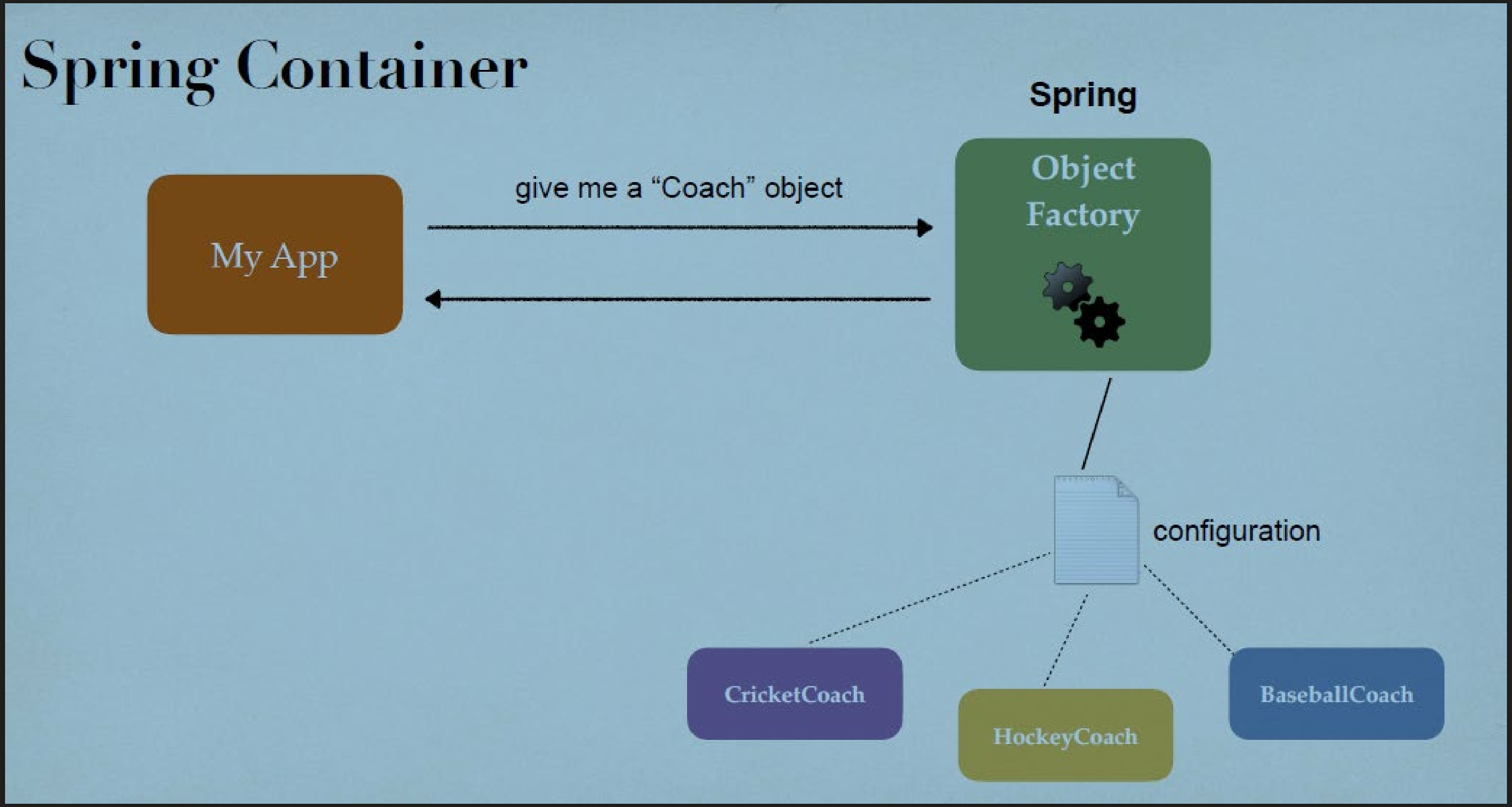
- App should be configurable
- Easily change the coach for another sport
  - Baseball, Hockey, Tennis, Gymnastics



Ideal Solution

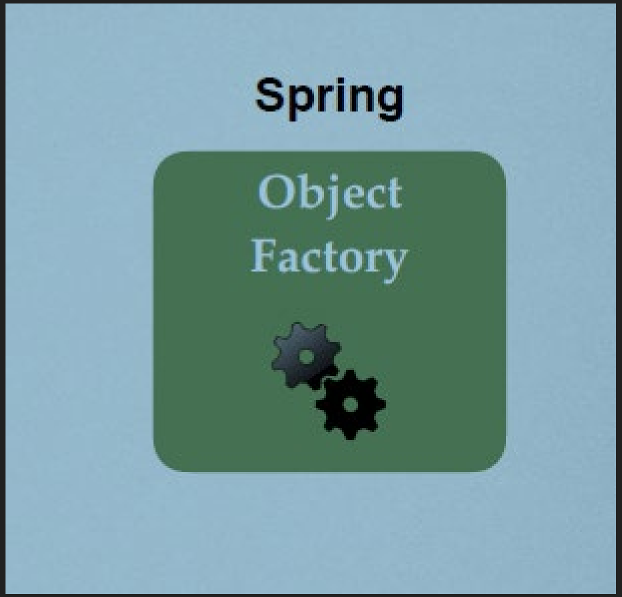


Spring Container

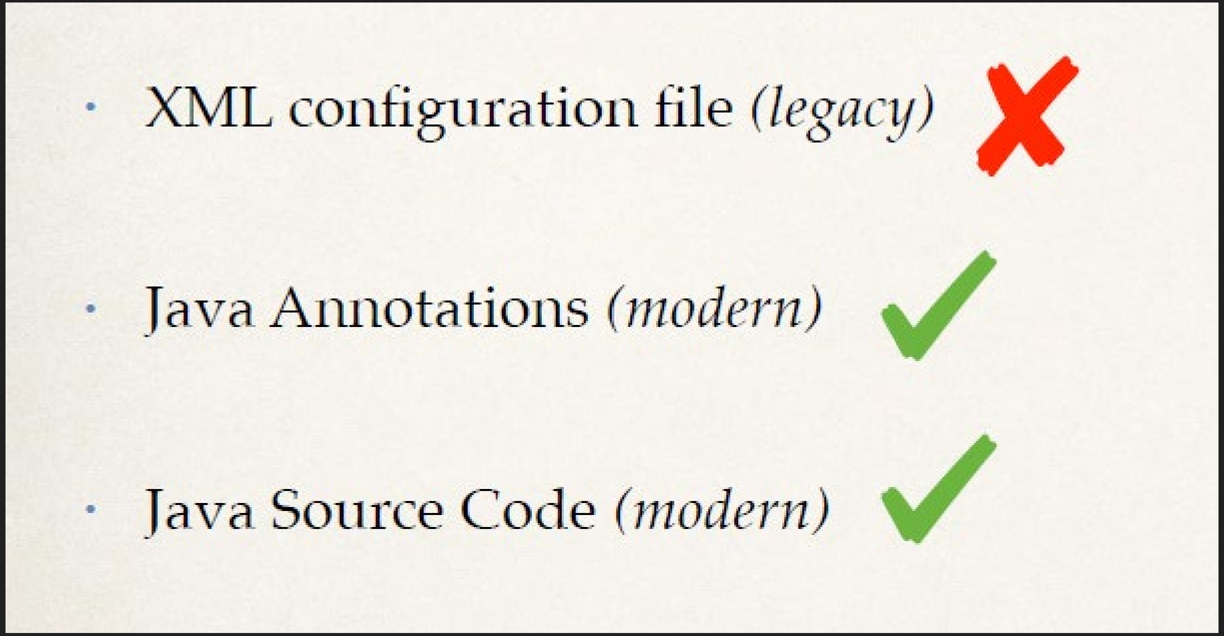


Spring Container

- Primary functions
  - Create and manage objects *(Inversion of Control)*
  - Inject object dependencies *(Dependency Injection)*



Configuring Spring Container



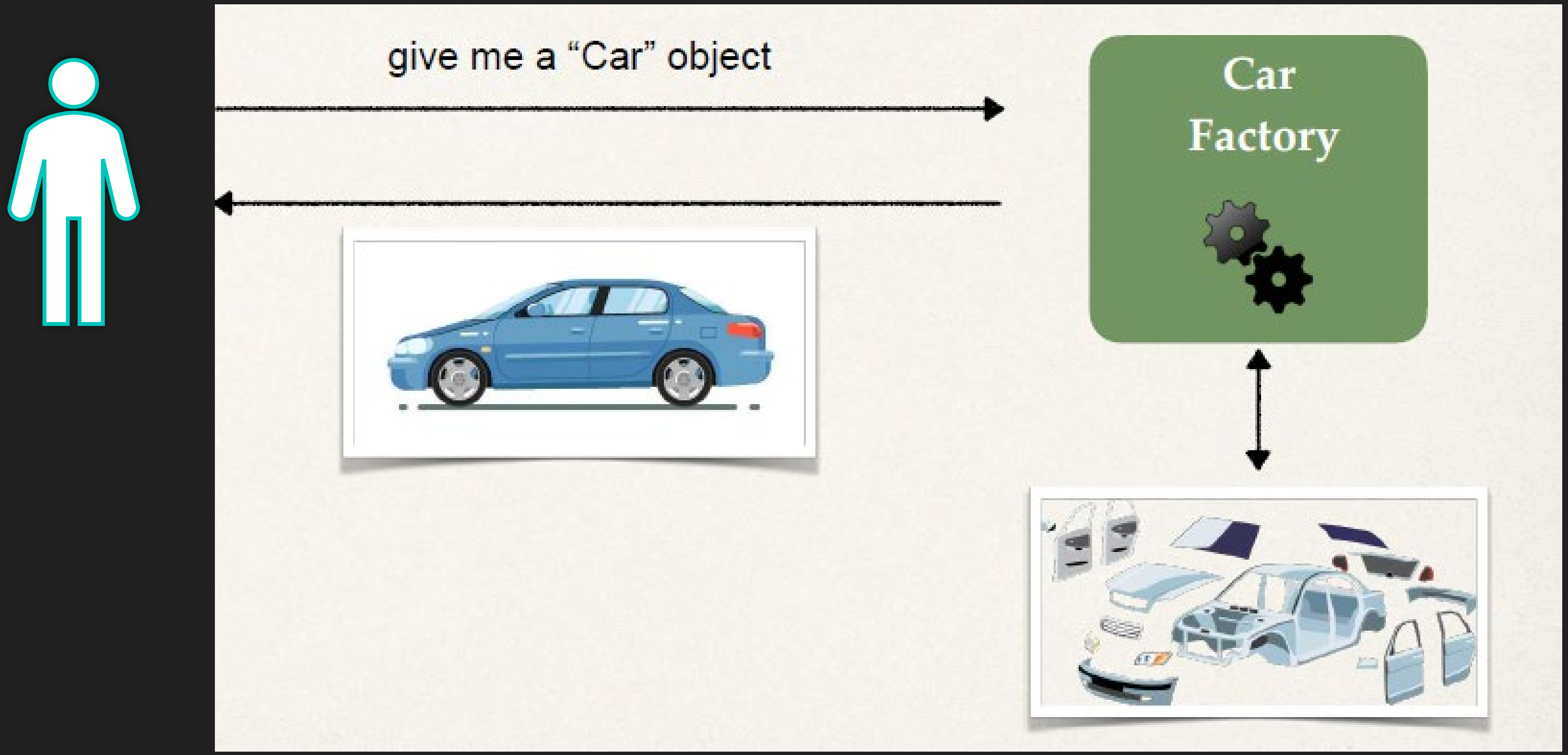
- XML configuration file *(legacy)* ✗
- Java Annotations *(modern)* ✓
- Java Source Code *(modern)* ✓

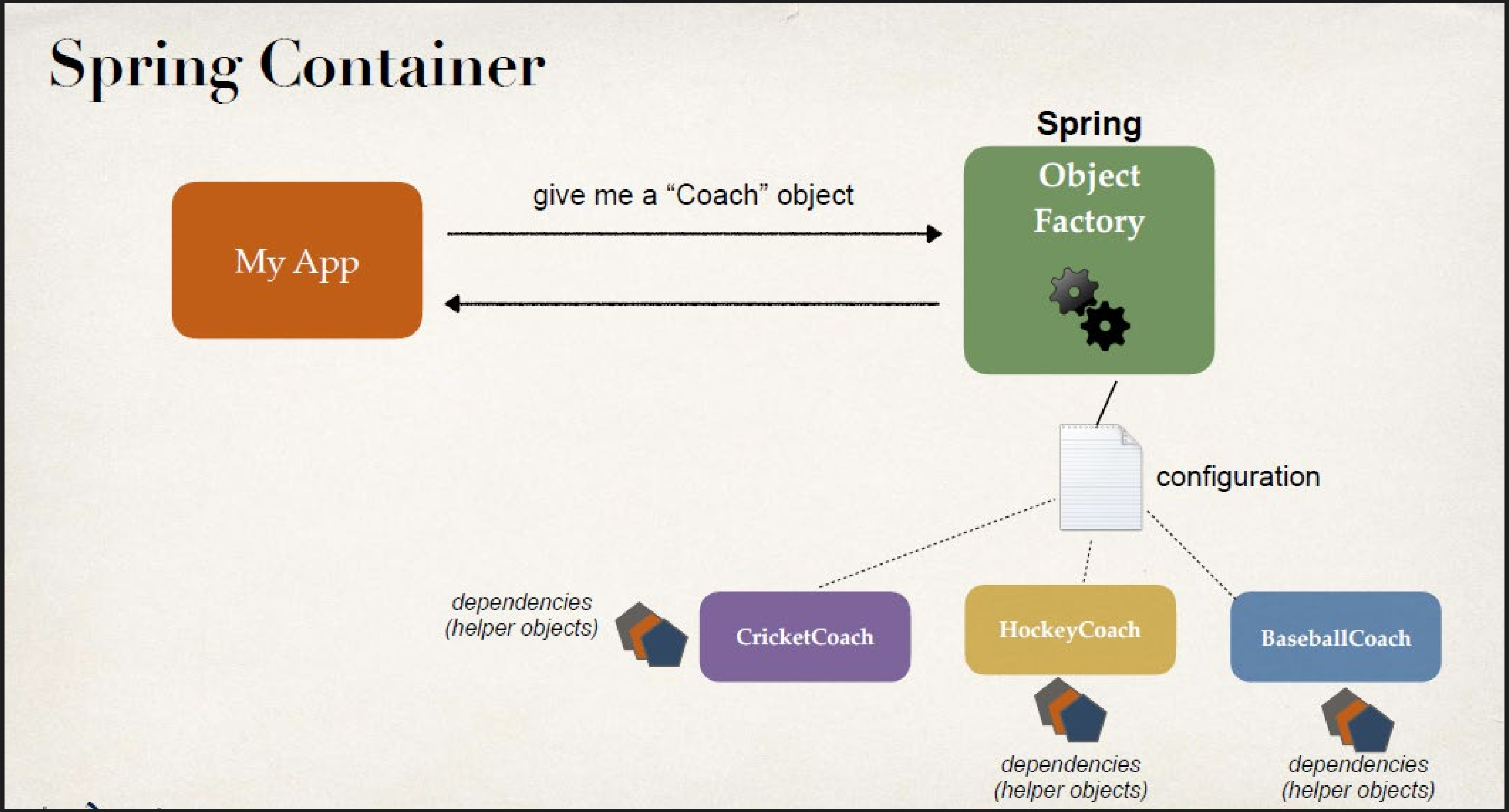
Spring Dependency Injection

The dependency inversion principle.

The client delegates to another object the responsibility of providing its dependencies.

Car Factory





Spring Container

DEMO

Injection Types

- There are multiple types of injection with Spring
- We will cover the two recommended types of injection
  - Constructor Injection
  - Setter Injection

Injection Types - Which one to use?

- Constructor Injection
  - Use this when you have required dependencies
  - Generally recommended by the spring.io development team as first choice
- Setter Injection
  - Use this when you have optional dependencies
  - If dependency is not provided, your app can provide reasonable default logic