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
- Writing Unit tests
- Deploy to Microsoft Azure

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 Initializer

- 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

- 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 Initializer website
- Download the zip file
- Unzip the file
- Import the project into our IDE

Spring Framework Overview

- Why **Spring**?
  - Simplify Java Enterprise Development

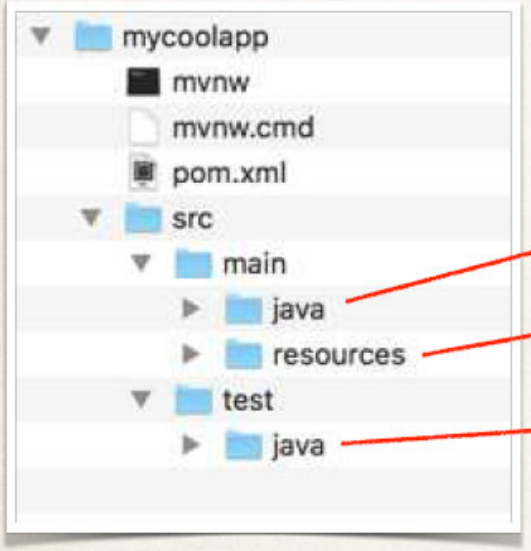
Goals of Spring

- Lightweight development with Java POJOs (Plain-Old-Java-Objects)
- Dependency injection to promote loose coupling
- 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



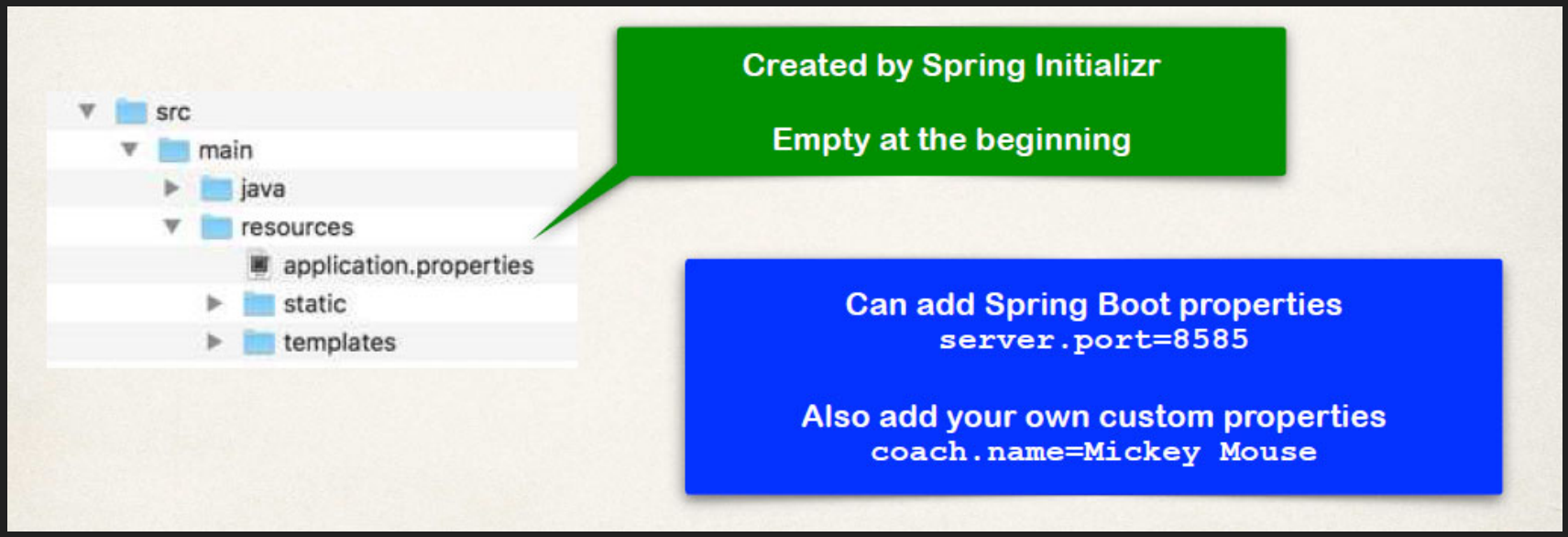
Directory	Description
src/main/java	Your Java source code
src/main/resources	Properties / config files used by your app
src/test/java	Unit testing source code

Maven POM file

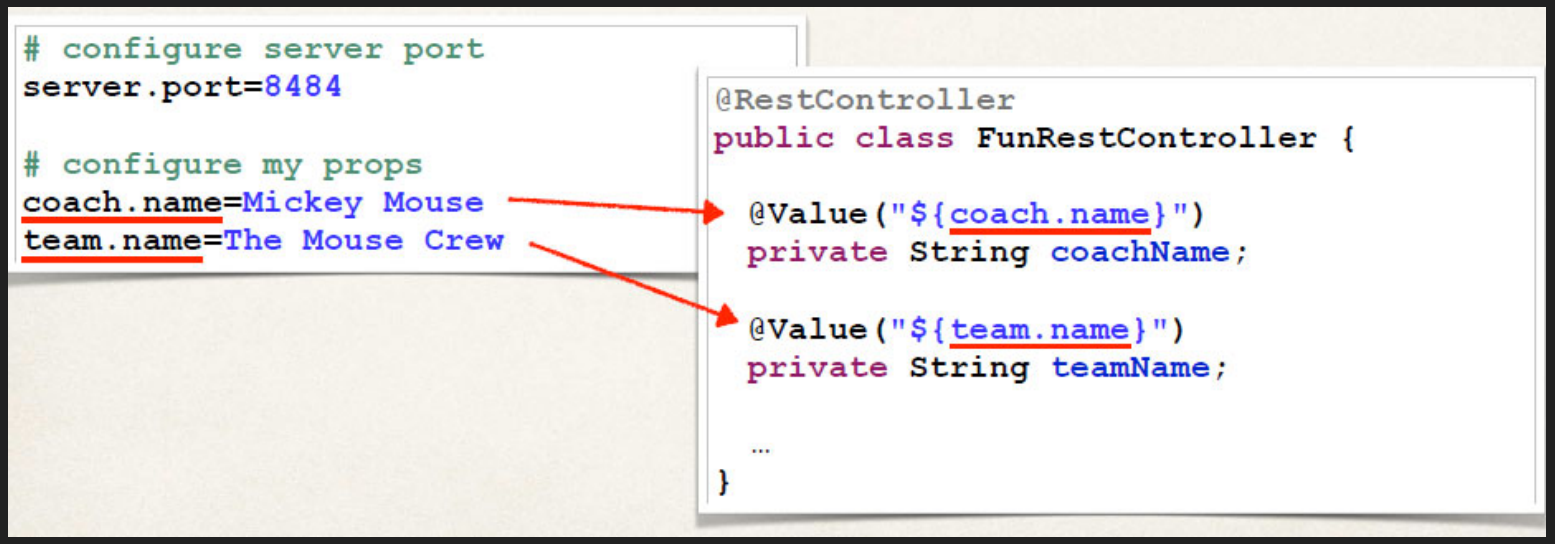
Name	Description
Group ID	Name of company, group, or organization. Convention is to use reverse domain name: com.jac
Artifact ID	Name for this project: myapp
Version	A specific release version like: **1.0, 1.6, 2.0** … If project is under active development then: **1.0-SNAPSHOT**

Application Properties

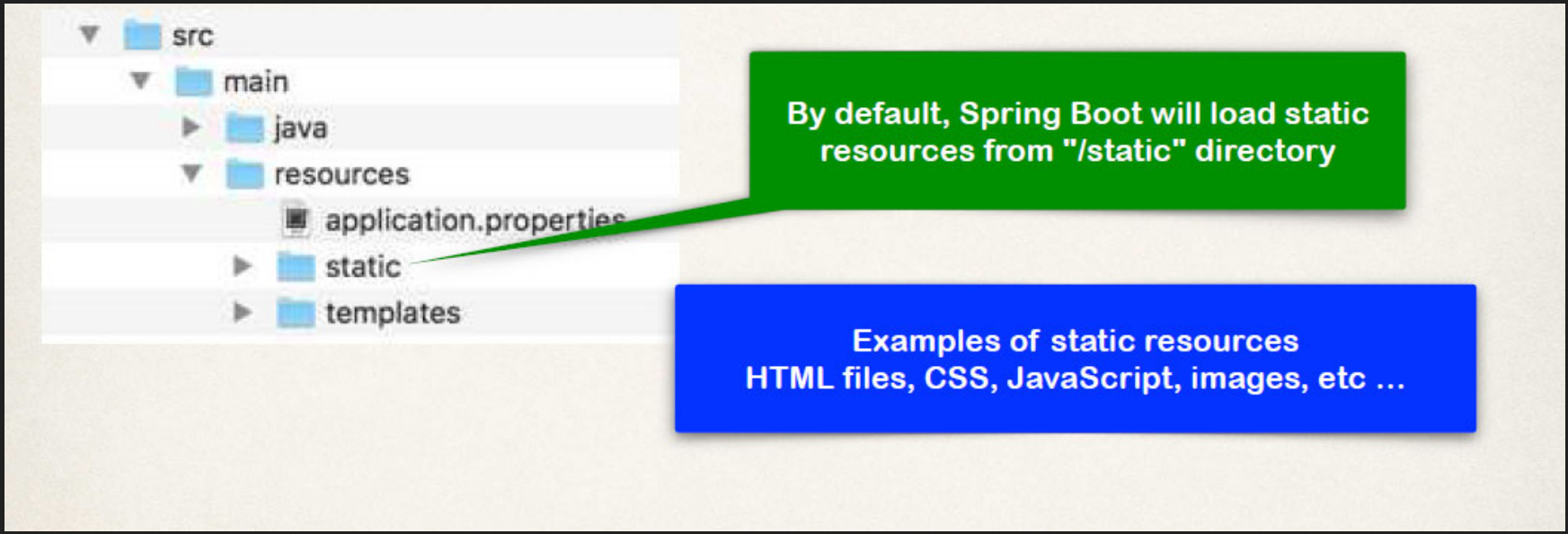
- By default, Spring Boot will load properties from: **application.properties**



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

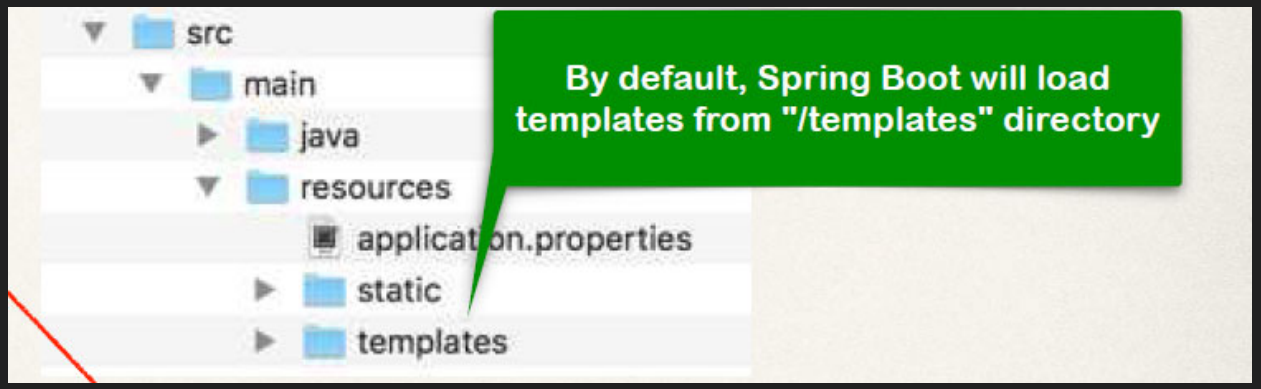


Static Content



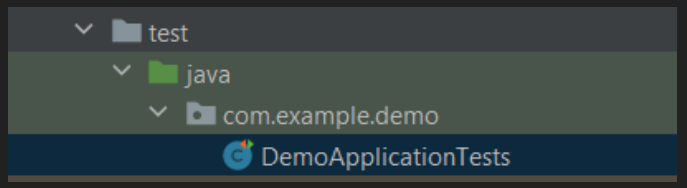
Templates

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



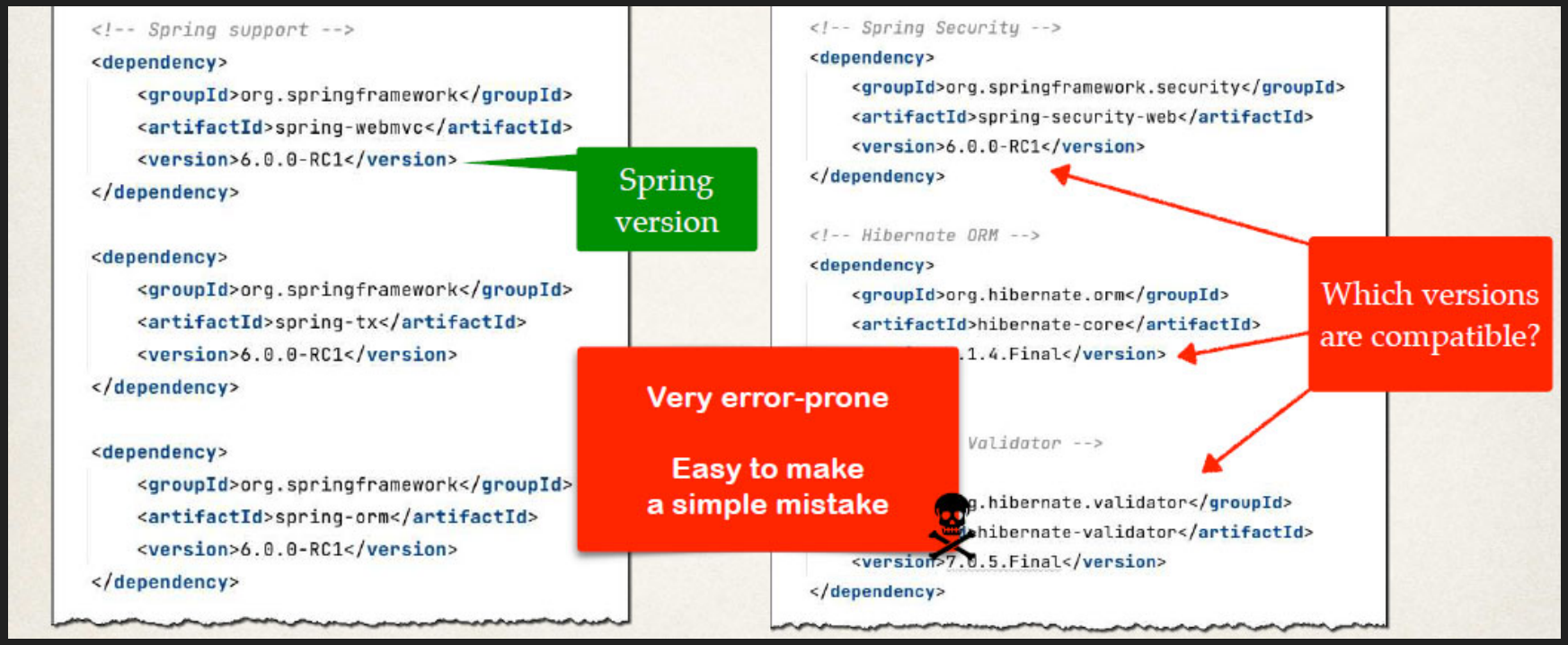
Unit Tests

- 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?


<!-- 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>
Spring version
<!-- 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>
Which versions are compatible?
Very error-prone
Easy to make a simple mistake
g.hibernate.validator</groupId>
hibernate-validator</artifactId>
<version>7.0.5.Final</version>
</dependency>

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

```xml
<!-- 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)

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

Also, makes sure you have compatible versions

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

Spring Initializr

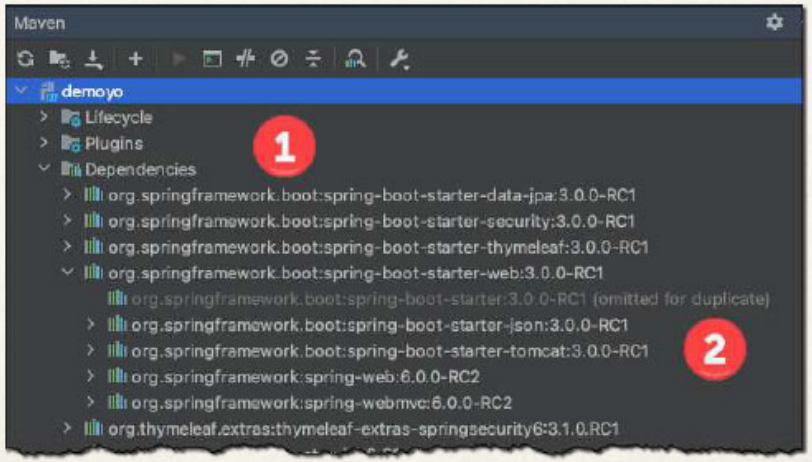
Spring Boot Starters

- 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**



Spring Boot Starter Parent

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

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

Included in pom.xml when using Spring Initializr

- 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>
```

Specify your Java version

Spring Boot Starter Parent

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

```
<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

```
<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!