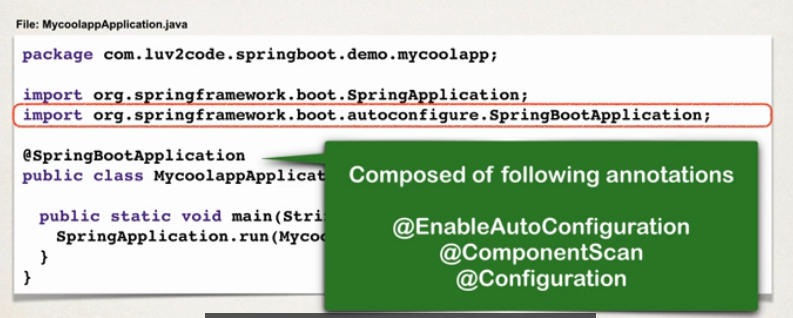
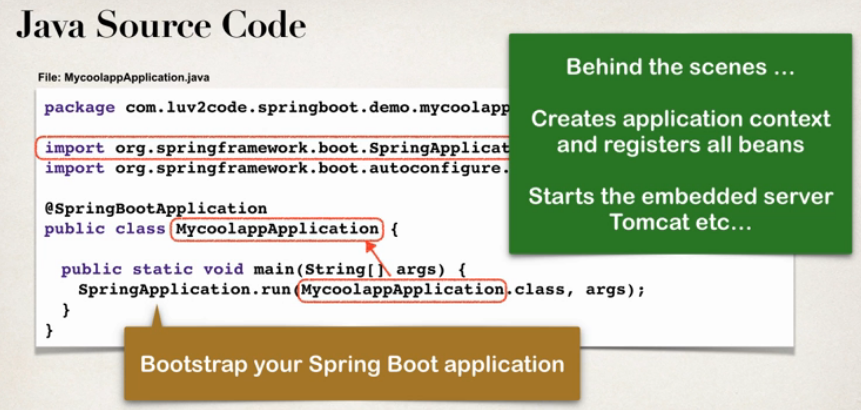
**Spring Boot**

1. **@SpringBootApplication:**  
   Spring Boot @SpringBootApplication annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning. It’s the same as declaring a class with **@Configuration**, **@EnableAutoConfiguration**, and **@ComponentScan** annotations.  
   
2. **SpringApplication Class:**  
   Spring Boot **SpringApplication** class is used to bootstrap and launch a Spring application from a Java main method. This class automatically creates the **ApplicationContext** from the classpath, scan the configuration classes and launches the application. This class is very helpful in launching Spring MVC or Spring REST applications using Spring Boot.  
   
3. **Simple Spring Boot POM file:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.0.1</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.skg.springbootdemo</groupId>

<artifactId>SpringBootDemo\_one</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringBootDemo\_one</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- ADD SUPPORT FOR AUTOMATIC RELOADING -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

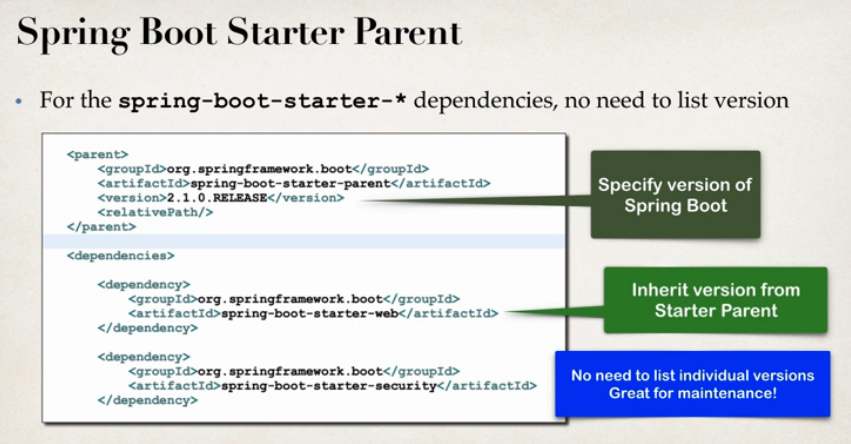
</build>

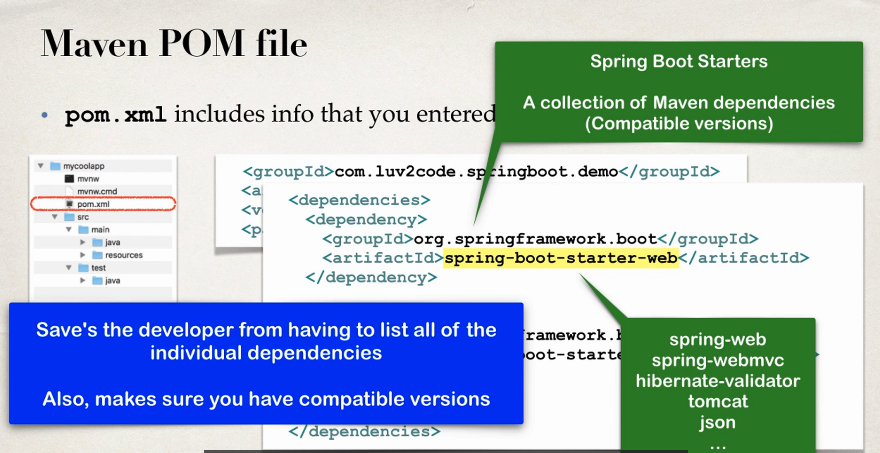
</project>

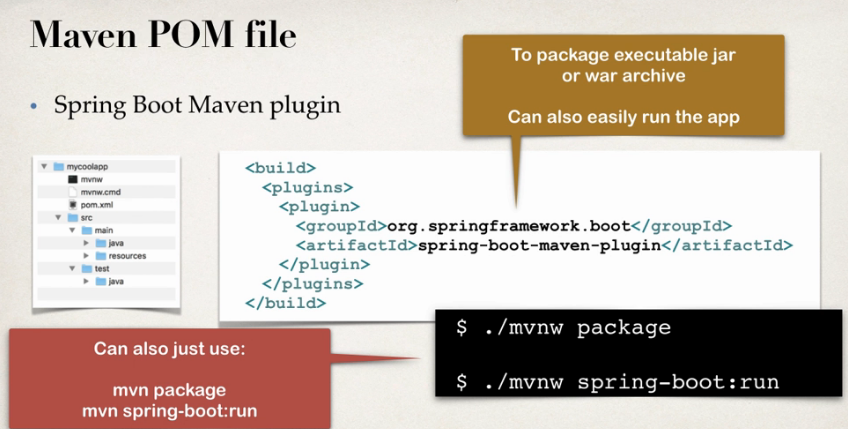
Spring Boot provides a number of "Starter POMs" that make it easy to add jars to our **classpath**. Our sample application has already used **spring-boot-starter-parent** in the parent section of the POM. The spring-boot-starter-parent is a special starter that provides useful Maven defaults. It also provides a dependency-management section so that we can omit version tags for dependencies.

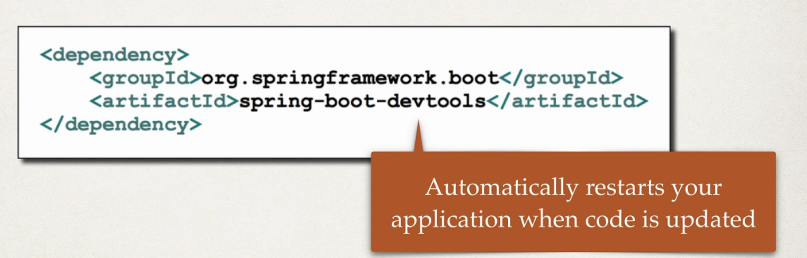
Note that **spring-boot-starter-parent** provides NO dependencies by itself.

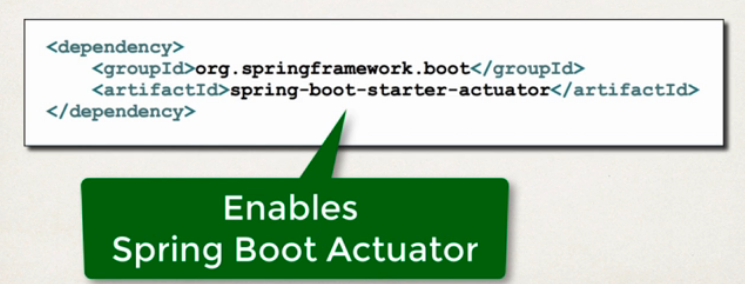
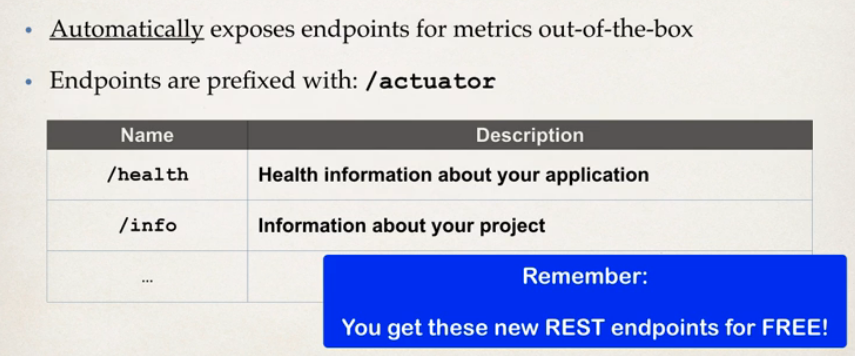
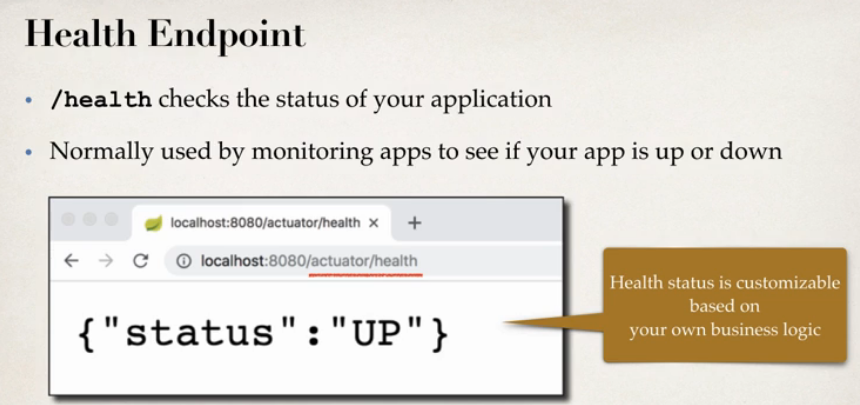
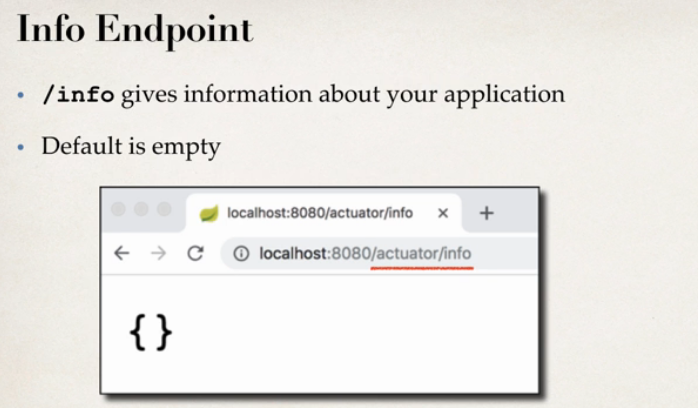
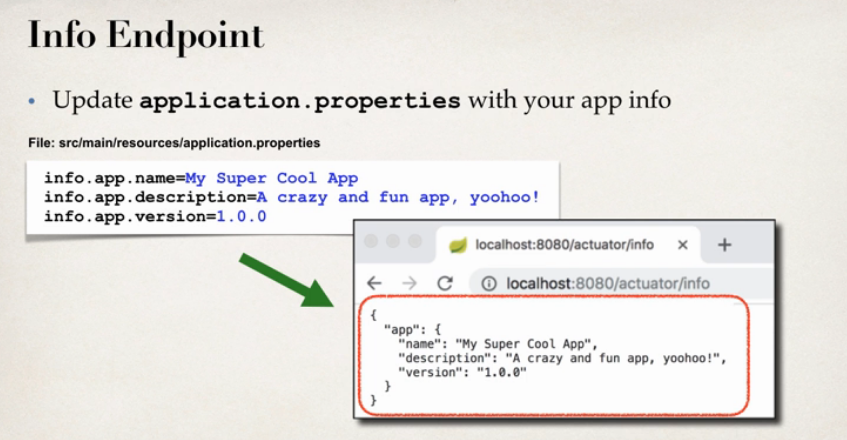
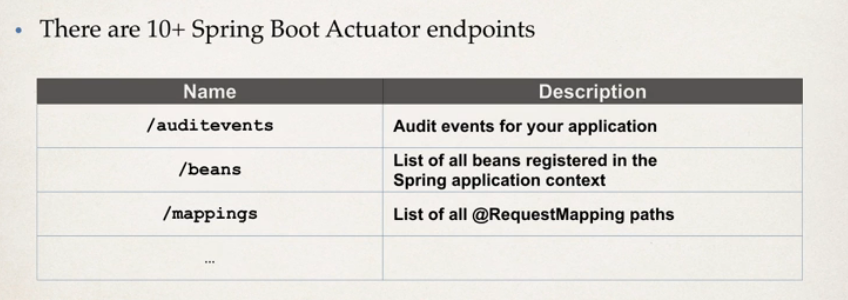
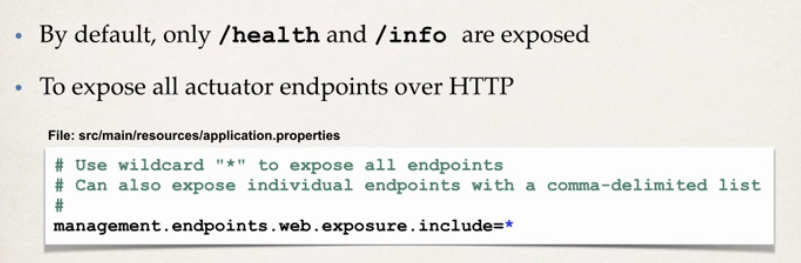
Other "Starter POMs" simply provide dependencies that we are likely to need when developing a specific type of application. Since we are developing a web application, we will add a **spring-boot-starter-web** dependency.

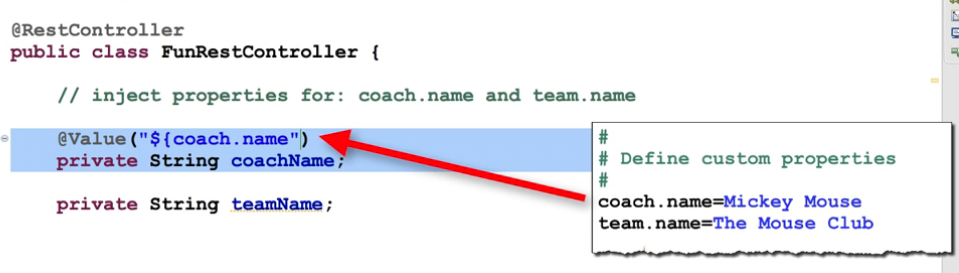








1. Spring Boot Actuator:  
   So the Spring Boot Actuator actually exposes endpoints for you to monitor and manage your application. So, you easily get the DevOps functionality out of the box. You simply add the dependency to your Maven POM file and then these rest endpoints are automatically added to your application.  
     
     
     
     
     
     
     
     
   
2. **Spring application.properties**  
     
   

**  
Example:**import java.time.LocalDateTime;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class FunRestController {

@Value("${coach.name}")

private String coachName;

@Value("${team.name}")

private String teamName;

// expose new endpoint for "teaminfo"

@GetMapping("/teaminfo")

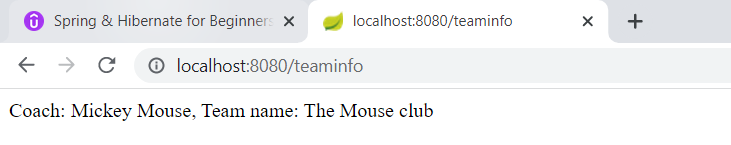
public String getTeamInfo() {

return "Coach: " + coachName + ", Team name: " + teamName;

}

}

**Output**

****

Some of the spring boot properties that we can use in **application.properties** are server port, context path, actuator related details, security etc..

Spring Boot has 1000+ properties.

