# Spring Boot Introduction

# You will learn how to …

# You will learn how to …

- Quickly develop Spring Boot applications

# You will learn how to …

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot

# You will learn how to …

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot
- Create a Spring MVC app with Spring Boot

# You will learn how to …

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot
- Create a Spring MVC app with Spring Boot
- Connect Spring Boot apps to a Database for CRUD development

# You will learn how to …

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot
- Create a Spring MVC app with Spring Boot
- Connect Spring Boot apps to a Database for CRUD development
- Leverage all Java configuration (no xml) and Maven

# Practical Results

# Practical Results

- Introduction to Spring Boot development

# Practical Results

- Introduction to Spring Boot development
- Not an A to Z reference

# Practical Results

- Introduction to Spring Boot development
- Not an A to Z reference
- For complete reference, see Spring Boot Reference Manual

# Practical Results

- Introduction to Spring Boot development
- Not an A to Z reference
- For complete reference, see Spring Boot Reference Manual

**https://spring.io/projects/spring-boot**

# The Problem

# The Problem

- Building a Spring application is really HARD!!!

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

Q: Which Maven dependencies do I need?

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

Q: Which Maven dependencies do I need?

Q: How do I set up configuration (xml or Java)?

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

Q: Which Maven dependencies do I need?

Q: How do I set up configuration (xml or Java)?

Q: How do I install the server? (Tomcat, JBoss etc…)

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

Q: Which Maven dependencies do I need?

Q: How do I set up configuration (xml or Java)?

Q: How do I install the server? (Tomcat, JBoss etc…)

And that's
JUST the basics
for getting started

# The Problem

# The Problem

- Tons of configuration

# The Problem

- Tons of configuration

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</s
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Servlet -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

# The Problem

- Tons of configuration

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</s
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Servlet -->
<servlet-
    <serv
    <url-
</servlet
```

```
<!-- Step 3: Add support for component scanning -->
<context:component-scan base-package="com.luv2code.springdemo" />

<!-- Step 4: Add support for conversion, formatting and validation support -->
<mvc:annotation-driven/>

<!-- Step 5: Define Spring MVC view resolver -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>
```

# The Problem

- Tons of configuration

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherS
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</par
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Serv
<servlet-
    <serv
    <url-
</servlet
```

```
<!-- Step 3: Add support for component scannin
<context:component-scan base-package="com.luv2
<!-- Step 4: Add support for conversion, forma
<mvc:annotation-driven/>

<!-- Step 5: Define Spring MVC view resolver -
<bean
    class="org.springframework.web.servlet.vie
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>
```

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {

    // define a bean for ViewResolver

    @Bean
    public ViewResolver viewResolver() {

        InternalResourceViewResolver viewResolver = new InternalResour

        viewResolver.setPrefix("/WEB-INF/view/");
        viewResolver.setSuffix(".jsp");

        return viewResolver;
    }
```

# The Problem

- Tons of configuration

**Very error-prone**

**Easy to make a simple mistake**

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherS
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</par
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Serv
<servlet-
    <serv
    <url-
</servlet
```

```
<!-- Step 3: Add support for component scanning
<context:component-scan base-package="com.luv2
<!-- Step 4: Add support for conversion, forma
<mvc:annotation-driven/>
<!-- Step 5: Define Spring MVC view resolver -
<bean
    class="org.springframework.web.servlet.vie
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>
```

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {

    // define a bean for ViewResolver

    @Bean
    public ViewResolver viewResolver() {

        InternalResourceViewResolver viewResolver = new InternalResour

        viewResolver.setPrefix("/WEB-INF/view/");
        viewResolver.setSuffix(".jsp");

        return viewResolver;
    }
```

# The Problem

- Tons of configuration

**Very error-prone**

**Easy to make a simple mistake**

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherS
```

```
@Configuration
@EnableWebMvc
```

```
    class="org.springframework.web.servlet.vie
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>
```

**There should be an easier solution**

# Spring Boot Solution

# Spring Boot Solution

- Make it easier to get started with Spring development

# Spring Boot Solution

- Make it easier to get started with Spring development
- Minimize the amount of manual configuration

# 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 and JAR classpath

# 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 and JAR classpath
- Help to resolve dependency conflicts (Maven or Gradle)

# 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 and JAR classpath
- Help to resolve dependency conflicts (Maven or Gradle)
- Provide an embedded HTTP server so you can get started quickly

# 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 and JAR classpath
- Help to resolve dependency conflicts (Maven or Gradle)
- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, …

# Spring Initializr

# Spring Initializr

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project
- Import the project into your IDE

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project
- Import the project into your IDE
  - Eclipse, IntelliJ, NetBeans etc …

http://start.spring.io

# Spring Boot Embedded Server

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, …

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, …
- No need to install a server separately

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, …
- No need to install a server separately

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, …
- No need to install a server separately

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, …
- No need to install a server separately

**mycoolapp.jar**

Self-contained unit
Nothing else to install

JAR file
includes your application code
AND
includes the server

# Running Spring Boot Apps

# Running Spring Boot Apps

- Spring Boot apps can be run standalone (includes embedded server)

# 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

# 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

# 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
```

Name of our JAR file

# Deploying Spring Boot Apps

# Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way

# Deploying Spring Boot Apps

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

# Deploying Spring Boot Apps

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

# Deploying Spring Boot Apps

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

# Deploying Spring Boot Apps

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

# Deploying Spring Boot Apps

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

# Spring Boot FAQ #1

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc …?

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc …?

- No. Instead, Spring Boot actually uses those technologies

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc …?

- No. Instead, Spring Boot actually uses those technologies

Spring Core | Spring AOP | Spring …

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc …?

- No. Instead, Spring Boot actually uses those technologies

| | | |
|---|---|---|
| Spring MVC | Spring REST | Spring … |
| Spring Core | Spring AOP | Spring … |

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc …?

- No. Instead, Spring Boot actually uses those technologies

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc …?

- No. Instead, Spring Boot actually uses those technologies

Once you do Spring Boot configs then you make use of regular Spring coding

@Component
@Controller
@Autowired
etc…

# Spring Boot FAQ #2

# Spring Boot FAQ #2

Q: Does Spring Boot run code faster than regular Spring code?

# Spring Boot FAQ #2

Q: Does Spring Boot run code faster than regular Spring code?

- No.
- Behind the scenes, Spring Boot uses same code of Spring Framework

# Spring Boot FAQ #2

Q: 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

# Spring Boot FAQ #2

Q: 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 …

# Spring Boot FAQ #3

# Spring Boot FAQ #3

Q: Do I need a special IDE for Spring Boot?

# Spring Boot FAQ #3

**Q: Do I need a special IDE for Spring Boot?**

- No.
- You can use any IDE for Spring Boot apps … even use plain text editor

# Spring Boot FAQ #3

**Q: 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]

# Spring Boot FAQ #3

**Q: 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

# Spring Boot FAQ #3

**Q: 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
- Not a requirement. Feel free to use the IDE that works best for you 😄

# Spring Boot Initializr Demo

# Spring Initializr

# Spring Initializr

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project
- Import the project into your IDE

http://start.spring.io

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project
- Import the project into your IDE
  - Eclipse, IntelliJ, NetBeans etc …

http://start.spring.io

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Configure our project at Spring Initializr website

http://start.spring.io

# Development Process

Step-By-Step

1. Configure our project at Spring Initializr website

2. Download the zip file

http://start.spring.io

# Development Process

Step-By-Step

1. Configure our project at Spring Initializr website

2. Download the zip file

3. Unzip the file

http://start.spring.io

# Development Process

Step-By-Step

1. Configure our project at Spring Initializr website

2. Download the zip file

3. Unzip the file

4. Import Maven project into our IDE

http://start.spring.io

# Spring Boot - Create REST Controller

# REST Controller

# REST Controller

- Let's create a very simple REST Controller

# REST Controller

- Let's create a very simple REST Controller

# Create REST Controller

# Create REST Controller

# Create REST Controller

```
@RestController
public class FunRestController {
```

# Create REST Controller

```
@RestController
public class FunRestController {

  // expose "/" that return "Hello World"
```

# Create REST Controller

```
@RestController
public class FunRestController {

  // expose "/" that return "Hello World"

  @GetMapping("/")
  public String sayHello() {
    return "Hello World! Time on server is " + LocalDateTime.now();
  }

}
```

Current time stamp

# Spring Boot Project Structure

# Spring Initializr

# Spring Initializr

- Spring Initializr created a Maven project for us

**http://start.spring.io**

# Spring Initializr

- Spring Initializr created a Maven project for us

- Let's explore the project structure

**http://start.spring.io**

# Maven Standard Directory Structure

# Maven Standard Directory Structure

| Directory | Description |
| --- | --- |
| src/main/java | Your Java source code |

# Maven Standard Directory Structure

| Directory | Description |
| --- | --- |
| src/main/java | Your Java source code |
| src/main/resources | Properties / config files used by your app |

# Maven Standard Directory 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 Wrapper files

# Maven Wrapper files

# Maven Wrapper files

# Maven Wrapper files

- **`mvnw`** allows you to run a Maven project

# Maven Wrapper files

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path

# Maven Wrapper files

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer

# Maven Wrapper files

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct version and runs Maven

# Maven Wrapper files

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct version and runs Maven
- Two files are provided

# Maven Wrapper files

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct ve
      and runs Maven
- Two files are provided
  - **`mvnw.cmd`** for MS Windows

```
> mvnw clean compile test
```

# Maven Wrapper files

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct ve
      and runs Maven
- Two files are provided
  - **`mvnw.cmd`** for MS Windows
  - **`mvnw.sh`** for Linux/Mac

```
> mvnw clean compile test
```

```
$ ./mvnw clean compile test
```

# Maven Wrapper files

# Maven Wrapper files

- If you already have Maven installed previously

# Maven Wrapper files

- If you already have Maven installed previously
  - Then you can ignore / delete the **`mvnw`** files

# Maven Wrapper files

- If you already have Maven installed previously
  - Then you can ignore / delete the **`mvnw`** files

- Just use Maven as you normally would

```
$ mvn clean compile test
```

# Maven POM file

# Maven POM file

- **`pom.xml`** includes info that you entered at Spring Initializr website

# Maven POM file

- **`pom.xml`** includes info that you entered at Spring Initializr website

```
<groupId>com.luv2code.springboot.demo</groupId>
<artifactId>mycoolapp</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
```

# Maven POM file

- **`pom.xml`** includes info that you entered at Spring Initializr website

```
<groupId>com.luv2code.springboot.demo</groupId>
<a
<v
<p
```

```
<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>
  …
</dependencies>
```

# Maven POM file

- **`pom.xml`** includes info that you entered

**Spring Boot Starters**

**A collection of Maven dependencies (Compatible versions)**

```
<groupId>com.luv2code.springboot.demo</groupId>
<a
<v
<p
```

```
<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>
  …
</dependencies>
```

# Maven POM file

- **`pom.xml`** includes info that you entered

```
<groupId>com.luv2code.springboot.demo</groupId>
<a
<v
<p
```

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

  <dependency>
    <groupId>org.springframework.b
    <artifactId>spring-boot-starte
    <scope>test</scope>
  </dependency>
  …
</dependencies>
```

**Spring Boot Starters**

**A collection of Maven dependencies (Compatible versions)**

**spring-web**
**spring-webmvc**
**hibernate-validator**
**tomcat**
**json**
**…**

# Maven POM file

- **`pom.xml`** includes info that you entered

```
<groupId>com.luv2code.springboot.demo</groupId>
<a
<v
<p
```

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

framework.b
oot-starte

</dependencies>
```

**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**

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

# Maven POM file

# Maven POM file

- Spring Boot Maven plugin

# Maven POM file

- Spring Boot Maven plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

# Maven POM file

- Spring Boot Maven plugin

To package executable jar
or war archive

Can also easily run the app

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

# Maven POM file

- Spring Boot Maven plugin

To package executable jar
or war archive

Can also easily run the app

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

```
$ ./mvnw package

$ ./mvnw spring-boot:run
```

# Maven POM file

- Spring Boot Maven plugin

To package executable jar
or war archive

Can also easily run the app

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

```
$ ./mvnw package

$ ./mvnw spring-boot:run
```

Can also just use:

mvn package
mvn spring-boot:run

# Java Source Code

# Java Source Code

**Main Spring Boot application class**

**Created by Spring Initializr**

# Java Source Code

Main Spring Boot application class

Created by Spring Initializr

RestController that we created in an earlier video

# Java Source Code

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplicat

  public static void main(Stri
    SpringApplication.run(Mycoo
  }
}
```

Enables

Auto configuration
Component scanning
Additional configuration

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplicat

  public static void main(Stri
    SpringApplication.run(Mycoo
  }
}
```

Composed of following annotations

@EnableAutoConfiguration
@ComponentScan
@Configuration

# Annotations

# Annotations

- `@SpringBootApplication` is composed of the following annotations:

| Annotation | Description |
|---|---|
| | |
| | |
| | |

# Annotations

- **`@SpringBootApplication`** is composed of the following annotations:

| Annotation | Description |
|---|---|
| `@EnableAutoConfiguration` | **Enables Spring Boot's auto-configuration support** |
| | |
| | |

# Annotations

- **`@SpringBootApplication`** is composed of the following annotations:

| Annotation | Description |
|---|---|
| `@EnableAutoConfiguration` | **Enables Spring Boot's auto-configuration support** |
| `@ComponentScan` | **Enables component scanning of current package<br>Also recursively scans sub-packages** |
| | |

# Annotations

- **`@SpringBootApplication`** is composed of the following annotations:

| Annotation | Description |
|---|---|
| `@EnableAutoConfiguration` | **Enables Spring Boot's auto-configuration support** |
| `@ComponentScan` | **Enables component scanning of current package**<br>**Also recursively scans sub-packages** |
| `@Configuration` | **Able to register extra beans with `@Bean`**<br>**or import other configuration classes** |

# Annotations

- **`@SpringBootApplication`** is composed of the following annotations:

| Annotation | Description |
|---|---|
| `@EnableAutoConfiguration` | **Enables Spring Boot's auto-configuration support** |
| `@ComponentScan` | **En… kage Al…** |
| `@Configuration` | **Al… or import other configuration classes** |

**Same annotations that we've used before with traditional Spring apps**

# Annotations

- **`@SpringBootApplication`** is composed of the following annotations:

| Annotation | Description |
|---|---|
| `@EnableAutoConfiguration` | **Enables Spring Boot's auto-configuration support** |
| `@ComponentScan` | **Enables component scanning of current package**<br>**Also recursively scans sub-packages** |
| `@Configuration` | **Able to register extra beans with `@Bean`**<br>**or import other configuration classes** |

# Java Source Code

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

Bootstrap your Spring Boot application

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

Bootstrap your Spring Boot application

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

Bootstrap your Spring Boot application

# Java Source Code

File: MycoolappApplication.java

```
package com.luv2code.springboot.demo.mycoolapp

import org.springframework.boot.SpringApplicat
import org.springframework.boot.autoconfigure.

@SpringBootApplication
public class MycoolappApplication {

  public static void main(String[] args) {
    SpringApplication.run(MycoolappApplication.class, args);
  }
}
```

Behind the scenes …

Creates application context and registers all beans

Starts the embedded server Tomcat etc…

Bootstrap your Spring Boot application

# More on Component Scanning

Best Practice

# More on Component Scanning

Best Practice

Place your main application class in
the root package above your other packages

# More on Component Scanning

Best Practice

**Place your main application class in the root package above your other packages**

- This implicitly defines a base search package
  - Allows you to leverage default component scanning
  - No need to explicitly reference the base package name

# More on Component Scanning

Best Practice

**Place your main application class in the root package above your other packages**

- This implicitly defines a base search package
  - Allows you to leverage default component scanning
  - No need to explicitly reference the base package name

**Common pitfall in traditional Spring apps**

***Anyone use the wrong package name before in traditional Spring apps???***

# More on Component Scanning

# More on Component Scanning

**Main Spring Boot application class**

**Automatically component scans sub-packages**

# More on Component Scanning

Main Spring Boot application class

Automatically component scans sub-packages

Includes "rest" sub-package

# More on Component Scanning

Main Spring Boot application class

Automatically component scans sub-packages

Includes "rest" sub-package

Any other sub-packages you create
Can give them any name

# More on Component Scanning

# More on Component Scanning

- Default scanning is fine if everything is under
  - `com.luv2code.springboot.demo.mycoolapp`

# More on Component Scanning

- Default scanning is fine if everything is under
  - `com.luv2code.springboot.demo.mycoolapp`

- But what about my other packages?

# More on Component Scanning

- Default scanning is fine if everything is under
  - **`com.luv2code.springboot.demo.mycoolapp`**

- But what about my other packages?
  - **`org.acme.iot.utils`**

# More on Component Scanning

- Default scanning is fine if everything is under
  - `com.luv2code.springboot.demo.mycoolapp`

- But what about my other packages?
  - `org.acme.iot.utils`
  - `edu.cmu.wean`

# More on Component Scanning

- Default scanning is fine if everything is under
  - **`com.luv2code.springboot.demo.mycoolapp`**

- But what about my other packages?
  - **`org.acme.iot.utils`**
  - **`edu.cmu.wean`**

```
package com.luv2code.springboot.demo.mycoolapp;
…
@SpringBootApplication(
    scanBasePackages={"com.luv2code.springboot.demo.mycoolapp",
                      "org.acme.iot.utils",
                      "edu.cmu.wean"})
public class MycoolappApplication {
  …
}
```

# More on Component Scanning

- Default scanning is fine if everything is under
  - **`com.luv2code.springboot.demo.mycoolapp`**
- But what about my other packages?
  - **`org.acme.iot.utils`**
  - **`edu.cmu.wean`**

**Explicitly list base packages to scan**

```
package com.luv2code.springboot.demo.mycoolapp;
…
@SpringBootApplication(
    scanBasePackages={"com.luv2code.springboot.demo.mycoolapp",
                      "org.acme.iot.utils",
                      "edu.cmu.wean"})
public class MycoolappApplication {
  …
}
```

# Application Properties

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

# Application Properties

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

**Created by Spring Initializr**

**Empty at the beginning**

# 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`**

# 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`

# Application Properties

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

# Application Properties

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

# Application Properties

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

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

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

# Application Properties

- 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 {
```

# Application Properties

- 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;
```

# Application Properties

- 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;
```

# Application Properties

- 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;
```

# Application Properties

- 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

# Static Content

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

# Static Content

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

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

# Static Content

# Static Content

**WARNING:**

**Do not use the `src/main/webapp` directory if your application is packaged as a JAR.**

**Although this is a standard Maven directory, it works only with WAR packaging.**

**It is silently ignored by most build tools if you generate a JAR.**

# Templates

# Templates

- Spring Boot includes auto-configuration for following template engines

# Templates

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

# Templates

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

# Templates

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

# Templates

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

# Templates

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

# Templates

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

By default, Spring Boot will load templates from "/templates" directory

Thymeleaf is a popular template engine
We will use it later in the course

# Unit Tests

# Unit Tests

src
main
java
resources
test
java
com
luv2code
springboot
demo
mycoolapp
MycoolappApplicationTests.java

# Unit Tests

**Spring Boot unit test class**

**Created by Spring Initializr**

# Unit Tests

Spring Boot unit test class

Created by Spring Initializr

You can add unit tests to the file

# Spring Boot Starters

# The Problem

# The Problem

- Building a Spring application is really HARD!!!

# The Problem

- Building a Spring application is really HARD!!!

FAQ: Which Maven dependencies do I need?

# The Problem

- Building a Spring application is really HARD!!!

FAQ: Which Maven dependencies do I need?

**Hi, I want to know the minimum required Maven Dependencies for Spring MVC + Hibernate connection.**
· Lecture 314 · 3 days ago

**if we use pom . What are the exact dependencies**
· Lecture 268 · 19 days ago

# The Problem

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
```

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframewo
        <artifactId>spring-tx</art
        <version>5.1.2.RELEASE</ve
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
```

Spring version

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframewo
        <artifactId>spring-tx</art
        <version>5.1.2.RELEASE</ve
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
```

Spring version

```
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.1.1.RELEASE</version>
</dependency>

<!-- Hibernate ORM -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.16.Final</version>
</dependency>

<!-- Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
```

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframewo
        <artifactId>spring-tx</art
        <version>5.1.2.RELEASE</ve
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
```

Spring version

```
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.1.1.RELEASE</version>
</dependency>

<!-- Hibernate ORM -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.16.Final</version>
</dependency>

<!-- Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
```

Which versions are compatible?

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframewo
        <artifactId>spring-tx</art
        <version>5.1.2.RELEASE</ve
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
```

Spring version

```
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.1.1.RELEASE</version>
</dependency>

<!-- Hibernate ORM -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.16.Final</version>
</dependency>

<!-- Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
```

Which versions are compatible?

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframewo
        <artifactId>spring-tx</art
        <version>5.1.2.RELEASE</ve
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
```

Spring version

```
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.1.1.RELEASE</version>
</dependency>

<!-- Hibernate ORM -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.16.Final</version>
</dependency>

<!-- Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
```

Which versions are compatible?

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframewo
        <artifactId>spring-tx</art
        <version>5.1.2.RELEASE</ve
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
```

Spring version

```
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.1.1.RELEASE</version>
</dependency>

<!-- Hibernate ORM -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.16.Final</version>
</dependency>

<!-- Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
```

Which versions are compatible?

# The Problem

```
<dependencies>

    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.2.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframewo
        <artifactId>spring-tx</art
        <version>5.1.2.RELEASE</ve
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifac
        <version>5.1.2.RELEASE</version
    </dependency>
```

Spring version

```
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.1.1.RELEASE</version>
</dependency>

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

alidator -->

.hibernate</groupId>
ibernate-validator</artifactId>
13.Final</version>
</dependency>
```

Which versions are compatible?

**Very error-prone**

**Easy to make a simple mistake**

# Why Is It So Hard?

# Why Is It So Hard?

- It would be great if there was a simple list of Maven dependencies

# 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

# 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

# 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

**There should be an easier solution**

# The Solution - Spring Boot Starters

# The Solution - Spring Boot Starters

- **Spring Boot Starters**

# The Solution - Spring Boot Starters

- **Spring Boot Starters**
- A curated list of Maven dependencies

# The Solution - Spring Boot Starters

- **Spring Boot Starters**
- A curated list of Maven dependencies
- A collection of dependencies grouped together

# The Solution - Spring Boot Starters

- **Spring Boot Starters**
- A curated list of Maven dependencies
- A collection of dependencies grouped together
- Tested and verified by the Spring Development team

# The Solution - Spring Boot Starters

- **Spring Boot Starters**
- A curated list of Maven dependencies
- 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

# The Solution - Spring Boot Starters

- **Spring Boot Starters**
- A curated list of Maven dependencies
- 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

# Spring MVC

# Spring MVC

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

```
<!-- Spring MVC -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.1.2.RELEASE</version>
</dependency>

<!-- Form validation: Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>

<!-- Web template: JSP or Thymeleaf etc -->
<dependency>
    <!-- ... -->
</dependency>
```

# Solution: Spring Boot Starter - Web

# Solution: Spring Boot Starter - Web

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

# 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>
```

# 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)**

# 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
...

# 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 Initializr

- In Spring Initializr, simply select **Web** dependency

# Spring Initializr

- In Spring Initializr, simply select **Web** dependency
- You automatically get **`spring-boot-starter-web`** in **`pom.xml`**

# Spring Initializr

- In Spring Initializr, simply select **Web** dependency
- You automatically get **`spring-boot-starter-web`** in **`pom.xml`**

# Spring Initializr

- In Spring Initializr, simply select **Web** dependency
- You automatically get **`spring-boot-starter-web`** in **`pom.xml`**

# Spring Initializr

- In Spring Initializr, simply select **Web** dependency
- You automatically get **`spring-boot-starter-web`** in **`pom.xml`**

# Spring Initializr

- In Spring Initializr, simply select **Web** dependency
- You automatically get **`spring-boot-starter-web`** in **`pom.xml`**

# Spring Initializr

# Spring Initializr

- If we are building a Spring app that needs: Web, Security, …

# Spring Initializr

- If we are building a Spring app that needs: Web, Security, …
- Simply select the dependencies in the Spring Initializr

# Spring Initializr

- If we are building a Spring app that needs: Web, Security, …
- Simply select the dependencies in the Spring Initializr

# Spring Initializr

- If we are building a Spring app that needs: Web, Security, …
- Simply select the dependencies in the Spring Initializr

# Spring Initializr

- If we are building a Spring app that needs: Web, Security, …
- Simply select the dependencies in the Spring Initializr
- It will add the appropriate Spring Boot starters to your **`pom.xml`**

**`pom.xml`**

# Spring Initializr

# Spring Initializr

# Spring Initializr

**File: pom.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 Initializr

**File: pom.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 Initializr

**File: pom.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 Initializr

**File: pom.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 Initializr

**File: pom.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

# Spring Boot Starters

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

| Name | Description |
| --- | --- |
| | |
| | |
| | |
| | |

# 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 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 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** |
| ... | |

# Spring Boot Starters

# Spring Boot Starters

**Full list**

**www.luv2code.com/spring-boot-starters**

| Name | Description |
| --- | --- |
| `spring-boot-starter` | Core starter, including a |
| `spring-boot-starter-activemq` | Starter for JMS messag |
| `spring-boot-starter-amqp` | Starter for using Spring |
| `spring-boot-starter-aop` | Starter for aspect-orient |
| `spring-boot-starter-artemis` | Starter for JMS messag |
| `spring-boot-starter-batch` | Starter for using Spring |
| `spring-boot-starter-cache` | Starter for using Spring |

# What Is In the Starter?

# What Is In the Starter?

- FAQ: What is in **`spring-boot-starter-xyz`**?

# What Is In the Starter?

- FAQ: What is in **`spring-boot-starter-xyz`**?

- View the starter's POM file

**www.luv2code.com/spring-boot-starters**

# What Is In the Starter?

- FAQ: What is in **`spring-boot-starter-xyz`**?

- View the starter's POM file

  **www.luv2code.com/spring-boot-starters**

  - Normally references other starters … so you will need to dig a bit

# What Is In the Starter?

- FAQ: What is in **`spring-boot-starter-xyz`**?

- View the starter's POM file

  **www.luv2code.com/spring-boot-starters**

  - Normally references other starters … so you will need to dig a bit
  - Somewhat cumbersome …

# What Is In the Starter?

# What Is In the Starter?

- Most IDEs have a Dependency Management / View feature

# What Is In the Starter?

- Most IDEs have a Dependency Management / View feature
  - Much easier to navigate

# What Is In the Starter?

- Most IDEs have a Dependency Management / View feature
  - Much easier to navigate
  - I'll show you how to do this with Eclipse and IntelliJ

# What Is In the Starter?

# What Is In the Starter?

- For Eclipse Users

# What Is In the Starter?

- For Eclipse Users
- Open the pom.xml

# What Is In the Starter?

- For Eclipse Users
- Open the pom.xml

# What Is In the Starter?

- For Eclipse Users
- Open the pom.xml
- Select the tab **Dependency Hierarchy**

# What Is In the Starter?

- For Eclipse Users
- Open the pom.xml
- Select the tab **Dependency Hierarchy**
- Expand the desired starter

# What Is In the Starter?

# What Is In the Starter?

- For IntelliJ users

# What Is In the Starter?

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

# Spring Boot Starter Parent

# Spring Boot Starter Parent

# Spring Boot Starter Parent

- Spring Boot provides a "Starter Parent"

# Spring Boot Starter Parent

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

# 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>2.1.0.RELEASE</version>
    <relativePath/>
</parent>
```

# 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>2.1.0.RELEASE</version>
    <relativePath/>
</parent>
```

Included in pom.xml when using Spring Initializr

# Spring Boot Starter Parent

# Spring Boot Starter Parent

- Maven defaults defined in the Starter Parent

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.0.RELEASE</version>
    <relativePath/>
</parent>
```

# Spring Boot Starter Parent

- Maven defaults defined in the Starter Parent
  - Default compiler level *(currently Java 8)*

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.0.RELEASE</version>
    <relativePath/>
</parent>
```

# Spring Boot Starter Parent

- Maven defaults defined in the Starter Parent
  - Default compiler level *(currently Java 8)*
  - UTF-8 source encoding

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.0.RELEASE</version>
    <relativePath/>
</parent>
```

# Spring Boot Starter Parent

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

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.0.RELEASE</version>
    <relativePath/>
</parent>
```

# Spring Boot Starter Parent

# Spring Boot Starter Parent

- To override a default, set as a property

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.0.RELEASE</version>
    <relativePath/>
</parent>

<properties>
    <java.version>12</java.version>
</properties>
```

# Spring Boot Starter Parent

- To override a default, set as a property

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.0.RELEASE</version>
    <relativePath/>
</parent>

<properties>
    <java.version>12</java.version>
</properties>
```

Specify your Java version

# Spring Boot Starter Parent

# Spring Boot Starter Parent

- For the **`spring-boot-starter-*`** dependencies, no need to list 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>2.1.0.RELEASE</version>
    <relativePath/>
</parent>

<dependencies>

    <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>
```

# 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>2.1.0.RELEASE</version>
    <relativePath/>
</parent>

<dependencies>

    <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>
```

Specify version of Spring Boot

# 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>2.1.0.RELEASE</version>
    <relativePath/>
</parent>

<dependencies>

    <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>
```

Specify version of Spring Boot

Inherit version from Starter Parent

# 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>2.1.0.RELEASE</version>
    <relativePath/>
</parent>

<dependencies>

    <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>
```

Specify version of Spring Boot

Inherit version from Starter Parent

No need to list individual versions
Great for maintenance!

# Spring Boot Starter Parent

# Spring Boot Starter Parent

- Default configuration of Spring Boot plugin

# Spring Boot Starter Parent

- Default configuration of Spring Boot plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

# Spring Boot Starter Parent

- Default configuration of Spring Boot plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

```
> mvn spring-boot:run
```

# Benefits of the Spring Boot Starter Parent

# Benefits of the Spring Boot Starter Parent

- Default Maven configuration: Java version, UTF-encoding etc

# Benefits of the Spring Boot Starter Parent

- Default Maven configuration: Java version, UTF-encoding etc
- Dependency management

# Benefits of the Spring Boot Starter Parent

- Default Maven configuration: Java version, UTF-encoding etc
- Dependency management
  - Use version on parent only

# Benefits of the Spring Boot Starter Parent

- Default Maven configuration: Java version, UTF-encoding etc
- Dependency management
  - Use version on parent only
  - **`spring-boot-starter-*`** dependencies inherit version from parent

# Benefits of the Spring Boot Starter Parent

- Default Maven configuration: Java version, UTF-encoding etc
- Dependency management
  - Use version on parent only
  - **`spring-boot-starter-*`** dependencies inherit version from parent
- Default configuration of Spring Boot plugin

# Spring Boot Dev Tools

# The Problem

# The Problem

- When running Spring Boot applications

# The Problem

- When running Spring Boot applications
  - If you make changes to your source code

# The Problem

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

# Solution: Spring Boot Dev Tools

# Solution: Spring Boot Dev Tools

- **`spring-boot-devtools`** to the rescue!

# Solution: Spring Boot Dev Tools

- **`spring-boot-devtools`** to the rescue!
- Automatically restarts your application when code is updated

# Solution: Spring Boot Dev Tools

- **`spring-boot-devtools`** to the rescue!
- Automatically restarts your application when code is updated
- Simply add the dependency to your POM file

# Solution: Spring Boot Dev Tools

- **`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

# Spring Boot Dev Tools

- Adding the dependency to your POM file

# Spring Boot Dev Tools

- Adding the dependency to your POM file

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

# 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

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-devtools`**

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-devtools`**

2. Add new REST endpoint to our app

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-devtools`**

2. Add new REST endpoint to our app

3. Verify the app is automatically reloaded

# Spring Boot DevTools Documentation

# Spring Boot DevTools Documentation

**For more details**

**www.luv2code.com/devtools-docs**

# Spring Boot Actuator

# Problem

# Problem

- How can I monitor and manage my application?

# Problem

- How can I monitor and manage my application?
- How can I check the application health?

# 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

# Solution: Spring Boot Actuator

- Exposes endpoints to monitor and manage your application

# Solution: Spring Boot Actuator

- Exposes endpoints to monitor and manage your application
- You easily get DevOps functionality out-of-the-box

# 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

# 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 <u>automatically</u> added to your application

# 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

# Spring Boot Actuator

- Adding the dependency to your POM file

# 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

# Spring Boot Actuator

- Automatically exposes endpoints for metrics out-of-the-box

# Spring Boot Actuator

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

| Name | Description |
|---|---|
| | |
| | |
| | |

# 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** |
| | |
| | |

# 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** |
| `/info` | **Information about your project** |
| | |

# 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** |
| `/info` | **Information about your project** |
| ... | |

# Recent Updates in Spring Boot 2.6
# 19 November 2021

/info endpoint is not exposed by default.

You have to manually expose it and also enable custom application info properties.

Here are the steps:

1. Edit application.properties
2. Add following entries

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

# Health Endpoint

# Health Endpoint

- **`/health`** checks the status of 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

# Health Endpoint

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

# Health Endpoint

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

Health status is customizable based on your own business logic

# Info Endpoint

# Info Endpoint

- `/info` gives information about your application

# Info Endpoint

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

# Info Endpoint

# Info Endpoint

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

# Info Endpoint

- 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
```

# Info Endpoint

- 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
```

# Spring Boot Actuator Endpoints

# Spring Boot Actuator Endpoints

- There are 10+ Spring Boot Actuator endpoints

| Name | Description |
| --- | --- |
| | |
| | |
| | |
| | |

# Spring Boot Actuator Endpoints

- There are 10+ Spring Boot Actuator endpoints

| Name | Description |
|---|---|
| `/auditevents` | **Audit events for your application** |
| | |
| | |
| | |

# Spring Boot Actuator Endpoints

- There are 10+ Spring Boot Actuator endpoints

| Name | Description |
|---|---|
| `/auditevents` | **Audit events for your application** |
| `/beans` | **List of all beans registered in the Spring application context** |
| | |
| | |

# Spring Boot Actuator Endpoints

- There are 10+ Spring Boot Actuator endpoints

| Name | Description |
|---|---|
| `/auditevents` | **Audit events for your application** |
| `/beans` | **List of all beans registered in the Spring application context** |
| `/mappings` | **List of all @RequestMapping paths** |
| ... | |

# Spring Boot Actuator Endpoints

# Spring Boot Actuator Endpoints

**Full list**

**www.luv2code.com/actuator-endpoints**

| ID | Description |
|---|---|
| `auditevents` | Exposes audit events information for the current applic |
| `beans` | Displays a complete list of all the Spring beans in your |
| `caches` | Exposes available caches. |
| `conditions` | Shows the conditions that were evaluated on configura or did not match. |
| `configprops` | Displays a collated list of all `@ConfigurationPrope` |
| `env` | Exposes properties from Spring's `ConfigurableEnv` |

# Exposing Endpoints

# Exposing Endpoints

- By default, only **`/health`** and **`/info`** are exposed

# Exposing Endpoints

- By default, only **`/health`** and **`/info`** are exposed
- To expose all actuator endpoints over HTTP

# Exposing Endpoints

- By default, only **`/health`** and **`/info`** are exposed
- To expose all actuator endpoints over HTTP

**File: src/main/resources/application.properties**

```
# Use wildcard "*" to expose all endpoints
# Can also expose individual endpoints with a comma-delimited list
#
management.endpoints.web.exposure.include=*
```

# Get A List of Beans

# Get A List of Beans

- Access `http://localhost:8080/actuator/beans`

# Get A List of Beans

- Access **`http://localhost:8080/actuator/beans`**


```
{
  "contexts": {
    "application": {
      "beans": {
        "endpointCachingOperationInvokerAdvisor": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.endpoint.invoker.cache.Cach
          "resource": "class path resource [org/springframework/boot/actuate/au
          "dependencies": [
            "environment"
          ]
        },
        "defaultServletHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.web.servlet.HandlerMapping",
          "resource": "class path resource [org/springframework/boot/autoconfig
          "dependencies": []
        },
        "org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfigura
          "aliases": [],
          "scope": "singleton",
          "type":
```

# Get A List of Beans

- Access **`http://localhost:8080/actuator/beans`**


```
{
  "contexts": {
    "application": {
      "beans": {
        "endpointCachingOperationInvokerAdvisor": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.endpoint.invoker.cache.Cach
          "resource": "class path resource [org/springframework/boot/actuate/au
          "dependencies": [
            "environment"
          ]
        },
        "defaultServletHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.web.servlet.HandlerMapping",
          "resource": "class path resource [org/springframework/boot/autoconfig
          "dependencies": []
        },
        "org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfigura
          "aliases": [],
          "scope": "singleton",
          "type":
```


**What about security??**

**We'll add security in later videos**

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-starter-acuator`**

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-starter-acuator`**

2. View actuator endpoints for: **`/health`** and **`/info`**

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-starter-acuator`**

2. View actuator endpoints for: **`/health`** and **`/info`**

3. Edit **`application.properties`** to customize **`/info`**

# Spring Boot Actuator - Security

# What about Security?

# What about Security?

- You may **NOT** want to expose all of this information


```
{
  "contexts": {
    "application": {
      "beans": {
        "endpointCachingOperationInvokerAdvisor": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.endpoint.invoker.cache.Cach
          "resource": "class path resource [org/springframework/boot/actuate/au
          "dependencies": [
            "environment"
          ]
        },
        "defaultServletHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.web.servlet.HandlerMapping",
          "resource": "class path resource [org/springframework/boot/autoconfig
          "dependencies": []
        },
        "org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfigura
          "aliases": [],
          "scope": "singleton",
          "type":
```

# What about Security?

- You may **NOT** want to expose all of this information
- Add Spring Security to project and endpoints are secured :-)

# What about Security?

- You may **NOT** want to expose all of this information
- Add Spring Security to project and endpoints are secured :-)

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

# What about Security?

- You may **NOT** want to expose all of this information
- Add Spring Security to project and endpoints are secured :-)

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

/health and /info
are still available
You can disable them
if you want

# Secured Endpoints

# Secured Endpoints

- Now when you access: **`/actuator/beans`**

# Secured Endpoints

- Now when you access: **`/actuator/beans`**
- Spring Security will prompt for login

# Secured Endpoints

- Now when you access: **`/actuator/beans`**
- Spring Security will prompt for login

# Secured Endpoints

- Now when you access: **`/actuator/beans`**
- Spring Security will prompt for login

Default user name: **`user`**

# Secured Endpoints

- Now when you access: **`/actuator/beans`**
- Spring Security will prompt for login

Default user name: **`user`**

Check console logs for password

```
11-02 21:05:57.074  INFO 24986 --- [          main] .s.s.UserDetailsSe
Using generated security password: 78fd68a6-c190-421d-934b-df7852fc7dc2
```

# Spring Security configuration

# Spring Security configuration

- You can override default user name and generated password

# Spring Security configuration

- You can override default user name and generated password

**File: src/main/resources/application.properties**

```
spring.security.user.name=scott
spring.security.user.password=tiger
```

# Customizing Spring Security

# Customizing Spring Security

- You can customize Spring Security for Spring Boot Actuator

# Customizing Spring Security

- You can customize Spring Security for Spring Boot Actuator
  - Use a database for roles, encrypted passwords etc …

# Customizing Spring Security

- You can customize Spring Security for Spring Boot Actuator
  - Use a database for roles, encrypted passwords etc …
- Follow the same techniques as regular Spring Security

```
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource);
   }
   …
}
```

# Customizing Spring Security

- You can customize Spring Security for Spring Boot Actuator
  - Use a database for roles, encrypted passwords etc …
- Follow the same techniques Security

Read security info from database

```
public class DemoSecurityConfig extends WebSecur

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource);
   }
   …
}
```

# Customizing Spring Security

- You can customize Spring Security for Spring Boot Actuator
  - Use a database for roles, encrypted passwords etc …
- Follow the same techniques as regular Spring Security

```
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerB

      auth.jdbcAuthentication().dataSource(security
   }
   …
}
```

```
   @Override
   protected void configure(HttpSecurity http) throws Exception {

     http.authorizeRequests()
      .antMatchers("/actuator/**").hasRole("ADMIN")
       …
   }
```

Only authorize ADMIN users

# Excluding Endpoints

# Excluding Endpoints

- To exclude **`/health`** and **`/info`**

# Excluding Endpoints

- To exclude **`/health`** and **`/info`**

**File: src/main/resources/application.properties**

```
# Exclude individual endpoints with a comma-delimited list
#
management.endpoints.web.exposure.exclude=health,info
```

# Spring Boot Actuator Documentation

# Spring Boot Actuator Documentation

**For more details**

**www.luv2code.com/actuator-docs**

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-starter-security`**

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-starter-security`**

2. Verify security on actuator endpoints for: **`/beans`** etc

# Development Process

Step-By-Step

1. Edit **`pom.xml`** and add **`spring-boot-starter-security`**

2. Verify security on actuator endpoints for: **`/beans`** etc

3. Disable endpoints for **`/health`** and **`/info`**

# Spring Boot: Run from Command-Line

# Running from the Command-Line

# Running from the Command-Line

- During development we spend most of our time in the IDE

# Running from the Command-Line

- During development we spend most of our time in the IDE

# Running from the Command-Line

- During development we spend most of our time in the IDE

- However, we may want to run our Spring Boot app outside of the IDE

# Running from the Command-Line

- During development we spend most of our time in the IDE

- However, we may want to run our Spring Boot app outside of the IDE

- One approach is running from the command-line

# Running from the Command-Line

# Running from the Command-Line

- When running from the command-line

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

- Since we using Spring Boot, the server is embedded in our JAR file

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

- Since we using Spring Boot, the server is embedded in our JAR file
  - No need to have separate server installed/running

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

- Since we using Spring Boot, the server is embedded in our JAR file
  - No need to have separate server installed/running

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

- Since we using Spring Boot, the server is embedded in our JAR file
  - No need to have separate server installed/running

# Running from the Command-Line

- When running from the command-line
  - No need to have IDE open/running

- Since we using Spring Boot, the server is embedded in our JAR file
  - No need to have separate server installed/running

- Spring Boot apps are self-contained

# Running from the Command-Line

# Running from the Command-Line

- Spring Boot apps are self-contained

# Running from the Command-Line

- Spring Boot apps are self-contained

# Running from the Command-Line

- Spring Boot apps are self-contained

**mycoolapp.jar**

JAR file
includes your application code
AND
includes the server

# Running from the Command-Line

- Spring Boot apps are self-contained

mycoolapp.jar

Self-contained unit
Nothing else to install

JAR file
includes your application code
AND
includes the server

# Running from the Command-Line

# Running from the Command-Line

- Two options for running the app

# Running from the Command-Line

- Two options for running the app

- Option 1: Use **`java -jar`**

# Running from the Command-Line

- Two options for running the app

- Option 1: Use **`java -jar`**

- Option 2: Use Spring Boot Maven plugin
  - **`mvnw spring-boot:run`**

# Option 1: Use `java -jar`

# Option 1: Use `java -jar`

# Option 1: Use `java -jar`

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

# Option 1: Use `java -jar`

mycoolapp.jar

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

Name of our JAR file

# Option 2: Use Spring Boot Maven plugin

# Option 2: Use Spring Boot Maven plugin

# Option 2: Use Spring Boot Maven plugin

# Option 2: Use Spring Boot Maven plugin

- **`mvnw`** allows you to run a Maven project

# Option 2: Use Spring Boot Maven plugin

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path

# Option 2: Use Spring Boot Maven plugin

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer

# Option 2: Use Spring Boot Maven plugin

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct version and runs Maven

# Option 2: Use Spring Boot Maven plugin

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct version and runs Maven
- Two files are provided

# Option 2: Use Spring Boot Maven plugin

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct ve
      and runs Maven
- Two files are provided
  - **`mvnw.cmd`** for MS Windows

```
> mvnw clean compile test
```

# Option 2: Use Spring Boot Maven plugin

- **`mvnw`** allows you to run a Maven project
  - No need to have Maven installed or present on your path
  - If correct version of Maven is NOT found on your computer
    - **Automatically downloads** correct ve
      and runs Maven
- Two files are provided
  - **`mvnw.cmd`** for MS Windows
  - **`mvnw.sh`** for Linux/Mac

```
> mvnw clean compile test
```

```
$ ./mvnw clean compile test
```

# Maven Wrapper files

# Maven Wrapper files

- If you already have Maven installed previously

# Maven Wrapper files

- If you already have Maven installed previously
  - Then you can ignore / delete the **`mvnw`** files

# Maven Wrapper files

- If you already have Maven installed previously
  - Then you can ignore / delete the **`mvnw`** files

- Just use Maven as you normally would

```
$ mvn clean compile test
```

# Option 2: Use Spring Boot Maven plugin

# Option 2: Use Spring Boot Maven plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

# Option 2: Use Spring Boot Maven plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

To package executable jar or war archive

Can also easily run the app

# Option 2: Use Spring Boot Maven plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

To package executable jar or war archive

Can also easily run the app

```
$ ./mvnw package

$ ./mvnw spring-boot:run
```

# Option 2: Use Spring Boot Maven plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

To package executable jar or war archive

Can also easily run the app

```
$ ./mvnw package

$ ./mvnw spring-boot:run
```

Can also just use:

mvn package
mvn spring-boot:run

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Exit the IDE

# Development Process

Step-By-Step

1. Exit the IDE

2. Package the app using **`mvnw package`**

# Development Process

Step-By-Step

1. Exit the IDE

2. Package the app using **`mvnw package`**

3. Run app using **`java -jar`**

# Development Process

Step-By-Step

1. Exit the IDE

2. Package the app using **`mvnw package`**

3. Run app using **`java -jar`**

4. Run app using Spring Boot Maven plugin, **`mvnw spring-boot:run`**

# Spring Boot - Custom Application Properties

# Problem

# Problem

- You need for your app to be configurable … no hard-coding of values

# Problem

- You need for your app to be configurable … no hard-coding of values
- You need to read app configuration from a properties file

# Solution: Application Properties file

# Solution: Application Properties file

- By default, Spring Boot reads information from a standard properties file

# Solution: Application Properties file

- By default, Spring Boot reads information from a standard properties file
  - Located at: **`src/main/resources/application.properties`**

# Solution: Application Properties file

- By default, Spring Boot reads information from a standard properties file
  - Located at: **`src/main/resources/application.properties`**

**Standard Spring Boot file name**

# Solution: Application Properties file

- By default, Spring Boot reads information from a standard properties file
  - Located at: **`src/main/resources/application.properties`**

Standard Spring Boot file name

- You can define ANY custom properties in this file

# Solution: Application Properties file

- By default, Spring Boot reads information from a standard properties file
  - Located at: **`src/main/resources/application.properties`**

Standard Spring Boot file name

- You can define ANY custom properties in this file
- Your Spring Boot app can access properties using **`@Value`**

# Solution: Application Properties file

- By default, Spring Boot reads information from a standard properties file
  - Located at: **`src/main/resources/application.properties`**
- You can define ANY custom properties in this file
- Your Spring Boot app can access properties using **`@Value`**

**Standard Spring Boot file name**

**No additional coding or configuration required**

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Define custom properties in **`application.properties`**

# Development Process

Step-By-Step

1. Define custom properties in **`application.properties`**

2. Inject properties into Spring Boot application using **`@Value`**

# Step 1: Define custom application properties

# Step 1: Define custom application properties

**File: src/main/resources/application.properties**

```
#
# Define custom properties
#
```

# Step 1: Define custom application properties

**File: src/main/resources/application.properties**

```
#
# Define custom properties
#
coach.name=Mickey Mouse
```

# Step 1: Define custom application properties

**File: src/main/resources/application.properties**

```
#
# Define custom properties
#
coach.name=Mickey Mouse
team.name=The Mouse Club
```

# Step 1: Define custom application properties

File: src/main/resources/application.properties

```
#
# Define custom properties
#
coach.name=Mickey Mouse
team.name=The Mouse Club
```

**You can give ANY custom property names**

# Step 2: Inject Properties into Spring Boot app

# Step 2: Inject Properties into Spring Boot app

```
@RestController
public class FunRestController {
```

# Step 2: Inject Properties into Spring Boot app

```
@RestController
public class FunRestController {

  // inject properties for: coach.name and team.name
```

# Step 2: Inject Properties into Spring Boot app

```
@RestController
public class FunRestController {

  // inject properties for: coach.name and team.name

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

# Step 2: Inject Properties into Spring Boot app

```
@RestController
public class FunRestController {

  // inject properties for: coach.name and team.name

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

File: src/main/resources/application.properties

```
#
# Define custom properties
#
coach.name=Mickey Mouse
team.name=The Mouse Club
```

# Step 2: Inject Properties into Spring Boot app

```
@RestController
public class FunRestController {

  // inject properties for: coach.name and team.name

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

File: src/main/resources/application.properties

```
#
# Define custom properties
#
coach.name=Mickey Mouse
team.name=The Mouse Club
```

# Step 2: Inject Properties into Spring Boot app

```
@RestController
public class FunRestController {

  // inject properties for: coach.name and team.name

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

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

  …
}
```

File: src/main/resources/application.properties

```
#
# Define custom properties
#
coach.name=Mickey Mouse
team.name=The Mouse Club
```

# Step 2: Inject Properties into Spring Boot app

```
@RestController
public class FunRestController {

  // inject properties for: coach.name and team.name

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

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

  …
}
```

File: src/main/resources/application.properties

```
#
# Define custom properties
#
coach.name=Mickey Mouse
team.name=The Mouse Club
```

# Step 2: Inject Properties into Spring Boot app

**No additional coding or configuration required**

```
@RestController
public class FunRestController {

  // inject properties for: coach.name and team.name

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

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

  …
}
```

File: src/main/resources/application.properties

```
#
# Define custom properties
#
coach.name=Mickey Mouse
team.name=The Mouse Club
```

# Spring Boot Properties

# Spring Boot Properties

# Spring Boot Properties

- Spring Boot can be configured in the **`application.properties`** file

# Spring Boot Properties

- Spring Boot can be configured in the **`application.properties`** file

- Server port, context path, actuator, security etc …

# Spring Boot Properties

- Spring Boot can be configured in the **`application.properties`** file
- Server port, context path, actuator, security etc …
- Spring Boot has 1,000+ properties … wowzers!

# Spring Boot Properties

# Spring Boot Properties

**List of Common Properties**

**www.luv2code.com/spring-boot-props**

# Spring Boot Properties

# Spring Boot Properties

- Don’t let the 1,000+ properties overwhelm you

# Spring Boot Properties

- Don’t let the 1,000+ properties overwhelm you
- The properties are roughly grouped into the following categories

# Spring Boot Properties

- Don't let the 1,000+ properties overwhelm you
- The properties are roughly grouped into the following categories

Core | Web | Security | Data

# Spring Boot Properties

- Don't let the 1,000+ properties overwhelm you
- The properties are roughly grouped into the following categories

| | | | |
|---|---|---|---|
| Core | Web | Security | Data |
| Actuator | Integration | DevTools | Testing |

# Spring Boot Properties

# Spring Boot Properties

**We'll review some of the properties …**

# Core Properties

Core

# Core Properties

Core

File: src/main/resources/application.properties

```
# Log levels severity mapping
logging.level.org.springframework=DEBUG
logging.level.org.hibernate=TRACE
logging.level.com.luv2code=INFO
```

# Core Properties

Core

File: src/main/resources/application.properties

```
# Log levels severity mapping
logging.level.org.springframework=DEBUG
logging.level.org.hibernate=TRACE
logging.level.com.luv2code=INFO
```

**Logging Levels**

TRACE
DEBUG
INFO
WARN
ERROR
FATAL
OFF

# Core Properties

Core

File: src/main/resources/application.properties

```
# Log levels severity mapping
logging.level.org.springframework=DEBUG
logging.level.org.hibernate=TRACE
logging.level.com.luv2code=INFO

# Log file name
logging.file=my-crazy-stuff.log
...
```

**Logging Levels**

TRACE
DEBUG
INFO
WARN
ERROR
FATAL
OFF

# Core Properties

Core

File: src/main/resources/application.properties

```
# Log levels severity mapping
logging.level.org.springframework=DEBUG
logging.level.org.hibernate=TRACE
logging.level.com.luv2code=INFO

# Log file name
logging.file=my-crazy-stuff.log
...
```

**Logging Levels**

TRACE
DEBUG
INFO
WARN
ERROR
FATAL
OFF

**Spring Boot Logging**
**www.luv2code.com/spring-boot-logging**

# Web Properties

Web

# Web Properties

Web

File: src/main/resources/application.properties

```
# HTTP server port
server.port=7070
```

# Web Properties

Web

File: src/main/resources/application.properties

```
# HTTP server port
server.port=7070

# Context path of the application
server.servlet.context-path=/my-silly-app
```

# Web Properties

Web

http://localhost:7070/my-silly-app/fortune

File: src/main/resources/application.properties

```
# HTTP server port
server.port=7070

# Context path of the application
server.servlet.context-path=/my-silly-app
```

# Web Properties

Web

http://localhost:7070/my-silly-app/fortune

File: src/main/resources/application.properties

```
# HTTP server port
server.port=7070

# Context path of the application
server.servlet.context-path=/my-silly-app

# Default HTTP session time out
server.servlet.session.timeout=15m
...
```

15 minutes

# Actuator Properties

Actuator

# Actuator Properties

Actuator

File: src/main/resources/application.properties

```
# Endpoints to include by name or wildcard
management.endpoints.web.exposure.include=*
```

# Actuator Properties

Actuator

File: src/main/resources/application.properties

```
# Endpoints to include by name or wildcard
management.endpoints.web.exposure.include=*

# Endpoints to exclude by name or wildcard
management.endpoints.web.exposure.exclude=beans,mapping
```

# Actuator Properties

Actuator

File: src/main/resources/application.properties

```
# Endpoints to include by name or wildcard
management.endpoints.web.exposure.include=*

# Endpoints to exclude by name or wildcard
management.endpoints.web.exposure.exclude=beans,mapping

# Base path for actuator endpoints
management.endpoints.web.base-path=/actuator
...
```

# Actuator Properties

Actuator

File: src/main/resources/application.properties

```
# Endpoints to include by name or wildcard
management.endpoints.web.exposure.include=*

# Endpoints to exclude by name or wildcard
management.endpoints.web.exposure.exclude=beans,mapping

# Base path for actuator endpoints
management.endpoints.web.base-path=/actuator
...
```

http://localhost:7070/actuator/health

# Security Properties

# Security Properties

Security

File: src/main/resources/application.properties

```
# Default user name
spring.security.user.name=admin
```

# Security Properties

Security

File: src/main/resources/application.properties

```
# Default user name
spring.security.user.name=admin

# Password for default user
spring.security.user.password=topsecret
...
```

# Data Properties

Data

# Data Properties

Data

File: src/main/resources/application.properties

```
# JDBC URL of the database
spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce
```

# Data Properties

Data

File: src/main/resources/application.properties

```
# JDBC URL of the database
spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce

# Login username of the database
spring.datasource.username=scott
```

# Data Properties

Data

File: src/main/resources/application.properties

```
# JDBC URL of the database
spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce

# Login username of the database
spring.datasource.username=scott

# Login password of the database
spring.datasource.password=tiger
...
```

# Data Properties

Data

File: src/main/resources/application.properties

```
# JDBC URL of the database
spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce

# Login username of the database
spring.datasource.username=scott

# Login password of the database
spring.datasource.password=tiger
...
```

More on this
in later videos

# Spring Boot Properties

# Spring Boot Properties

**List of Common Properties**

**www.luv2code.com/spring-boot-props**

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Configure the server port

# Development Process

Step-By-Step

1. Configure the server port

2. Configure the application context path