Course Introduction

© luv2code LLC

Main Objective

Learn how to build Java applications with

Spring Boot and Hibernate

© luv2code LLC

You will learn how to …

- Develop Spring Boot applications
- Leverage Hibernate/JPA for database access
- 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
- Apply Spring Security to control application access
- Leverage all Java configuration (no xml) and Maven

© luv2code LLC

Source Code and PDFs

All source code is available for download

All PDFs of slides are available for download

© luv2code LLC

Questions / Help

If you have questions or need help…

Post the question in the classroom discussion forum

© luv2code LLC

Java Development Environment

© luv2code LLC

Java Development Environment

- We assume that you are already have experience with Java
  - OOP, classes, interfaces, inheritance, exception handling, collections

- You should have the following items already installed
  - Java Development Kit (JDK) - *Spring Boot 3 requires JDK 17 or higher*
  - Java IDE (we'll use IntelliJ in the videos, but any Java IDE will work)

© luv2code LLC

About IntelliJ

Super Amazing IDE!!!

- In this course, we will use the free version of IntelliJ
  - Known as **IntelliJ Community Edition**
  - Download from: **https://www.jetbrains.com/idea/download**
  - Select **Community Edition**

- You can also use the Ultimate Edition ($) ... free trial version is available

© luv2code LLC

Java Development Environment Checkpoint

- You should have the following items already installed
  - Java Development Kit (JDK) - *Spring Boot 3 requires JDK 17 or higher*
  - Java IDE (we'll use IntelliJ in the videos, but any Java IDE will work)

- Before continuing with the course
  - Make sure you can run a basic HelloWorld Java app in your Java IDE

© luv2code LLC

Spring Boot Introduction

© luv2code LLC

Spring in a Nutshell

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

© luv2code LLC

The Problem

- Building a traditional Spring application is really HARD!!!

Q: Which JAR 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

© luv2code LLC

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, …

© luv2code LLC

Spring Boot and Spring

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

© luv2code LLC

Spring Initializr

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

http://start.spring.io

© luv2code LLC

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

© luv2code LLC

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

© luv2code LLC

Deploying Spring Boot Apps

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

© luv2code LLC

Spring Boot FAQ #1

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

- No. Instead, Spring Boot actually uses those technologies

© luv2code LLC

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 …

© luv2code LLC

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 😄

© luv2code LLC

Spring Boot Initializr Demo

© luv2code LLC

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

© luv2code LLC

Quick Word on Maven

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

© luv2code LLC

Maven Solution

- Tell Maven the projects you are working with (dependencies)
  - Spring, Hibernate etc ….
- Maven will go out and download the JAR files for those projects for you
- And Maven will make those JAR files available during compile/run
- Think of Maven as your friendly helper / personal shopper :-)

© luv2code LLC

Development Process

Step-By-Step

1. Configure our project at Spring Initializr website

http://start.spring.io

2. Download the zip file

3. Unzip the file

4. Import the project into our IDE

© luv2code LLC

Spring Boot - Create REST Controller

© luv2code LLC

REST Controller

- Let's create a very simple REST Controller

© luv2code LLC

Create REST Controller

Set up rest controller

```
@RestController
public class FunRestController {

  // expose "/" that returns "Hello World"

  @GetMapping("/")
  public String sayHello() {
    return "Hello World!";
  }

}
```

Handle HTTP GET requests

© luv2code LLC

Spring Framework Overview

© luv2code LLC

Spring Website - Official

www.spring.io

© luv2code LLC

Why Spring?

Simplify Java Enterprise Development

© luv2code LLC

Goals of Spring

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

© luv2code LLC

"Boilerplate" Java kodu, birçok Java uygulamasında sıkça kullanılan, genel amaçlı ve tekrarlanan kod parçalarını ifade eder. Bu kod parçaları, Java dilinin belirli bir yapısını uygulamak veya genel işlevleri gerçekleştirmek için kullanılır.

"Lightweight development with Java POJOs" ifadesi, Java programlama dilinde "POJO" olarak adlandırılan basit ve hafif nesneler kullanarak geliştirme yapmayı ifade eder. POJO'lar, Java'nın temel özelliklerini kullanarak sadece veri tutan ve basit işlemleri gerçekleştiren nesnelerdir.

Factory for creating beans
Manage bean depedencies

Core Container

© luv2code LLC

Web Layer

All Web related classes

Home of the Spring MVC framework

© luv2code LLC

Data Access Layer

JDBC Helper classes

Reduce your JDBC code by 50%

Object to Relational Mapping

Integration with Hibernate and JPA

Add transaction support

Makes heavy use of AOP behind the scenes

Java Message Service

For sending async messages
to a Message Broker

Spring provides helper classes for JMS

© luv2code LLC

Infrastructure

AOP: Aspect Oriented Programming
Add functionality to objects declaratively
- Logging, security, transactions, etc...

Java agents to remotely monitor your app
with JMX (Java Management Extension)

© luv2code LLC

Test Layer

Supports Test-Driven-Development (TDD)

Mock objects and out-of-container testing

© luv2code LLC

Spring Framework Overview

© luv2code LLC

Spring Projects

© luv2code LLC

What Are Spring "Projects"

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

© luv2code LLC

Spring Projects

© luv2code LLC

Maven Crash Course

© luv2code LLC

Spring Boot and Maven

- When you generate projects using Spring Initializr: start.spring.io
  - It can generate a Maven project for you

- In this section, we will learn the basics of Maven
  - Viewing dependencies in the Maven pom.xml file
  - Spring Boot Starters for Maven

© luv2code LLC

What is Maven?

- Maven is a Project Management tool
- Most popular use of Maven is for build management and dependencies

© luv2code LLC

What Problems Does Maven Solve?

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

© luv2code LLC

My Project without Maven

© luv2code LLC

Maven Solution

- Tell Maven the projects you are working with (dependencies)
  - Spring, Hibernate etc ….
- Maven will go out and download the JAR files for those projects for you
- And Maven will make those JAR files available during compile/run
  ❤️ Standard Directory Structure
  ❤️ Maven solves this problem by providing a standard directory structure
- Think of Maven as your friendly helper / personal shopper :-)

© luv2code LLC

My Project with Maven

© luv2code LLC

Maven - How It Works

© luv2code LLC

Handling JAR Dependencies

- When Maven retrieves a project dependency
  - It will also download supporting dependencies
  - For example: Spring depends on commons-logging …
- Maven will handle this for us automagically

© luv2code LLC

Building and Running

- When you build and run your app …
- Maven will handle class / build path for you
- Based on config file, Maven will add JAR files accordingly

© luv2code LLC

Standard Directory Structure

- Normally when you join a new project
  - Each development team dreams up their own directory structure
  - Not ideal for new comers and not standardized

- Maven solves this problem by providing a standard directory structure

© luv2code LLC

Standard Directory Structure

Directory	Description
src/main/java	Your Java source code
src/main/resources	Properties / config files used by your app
src/main/webapp	JSP files and web config files other web assets (images, css, js, etc)
src/test	Unit testing code and properties
target	Destination directory for compiled code. Automatically created by Maven

© luv2code LLC

Standard Directory Structure

Place your Java source code here

© luv2code LLC

Standard Directory Structure

Place your Web assets here

© luv2code LLC

Standard Directory Structure Benefits

- For new developers joining a project
  - They can easily find code, properties files, unit tests, web files etc …

© luv2code LLC

Standard Directory Structure Benefits

- Most major IDEs have built-in support for Maven
  - Eclipse, IntelliJ, NetBeans etc
  - IDEs can easily read/import Maven projects

- Maven projects are portable
  - Developers can easily share projects between IDEs
  - No need to fight about which IDE is the best LOL!

© luv2code LLC

Advantages of Maven

- Dependency Management
  - Maven will find JAR files for you
  - No more missing JARs
- Building and Running your Project
  - No more build path / classpath issues
- Standard directory structure

© luv2code LLC

My Personal Best Maven Benefit(s)

- Once you learn Maven, you can join a new project and be productive
- You can build and run a project with minimal local configuration

© luv2code LLC

Maven Key Concepts

© luv2code LLC

Maven Key Concepts

- POM File - pom.xml
- Project Coordinates

POM File - pom.xml

- Project Object Model file: POM file
- Configuration file for your project
  - Basically your "shopping list" for Maven :-)
- Located in the root of your Maven project

POM File Structure

pom.xml

Project name, version etc
Output file type: JAR, WAR, …

List of projects we depend on
Spring, Hibernate, etc…

Additional custom tasks to run:
generate JUnit test reports etc…

Simple POM File

```
<project ...>

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.luv2code</groupId>
  <artifactId>mycoolapp</artifactId>
  <version>1.0.FINAL</version>
  <packaging>jar</packaging>

  <name>mycoolapp</name>

  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>5.9.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

    <!-- add plugins for customization -->

</project>
```

Project name, version etc
Output file type: JAR, WAR, …

List of projects we depend on
Spring, Hibernate, etc…

Additional custom tasks to run:
generate JUnit test reports etc…

Project Coordinates

- Project Coordinates uniquely identify a project
  - Similar to GPS coordinates for your house: latitude / longitude
  - Precise information for finding your house (city, street, house #)

```
<groupId>com.luv2code</groupId>
<artifactId>mycoolapp</artifactId>
<version>1.0.FINAL</version>
```

City

Street

House Number

Project Coordinates - Elements

Name	Description
Group ID	Name of company, group, or organization. Convention is to use reverse domain name: **com.luv2code**
Artifact ID	Name for this project: **mycoolapp**
Version	A specific release version like: **1.0, 1.6, 2.0** … If project is under active development then: **1.0-SNAPSHOT**

```
<groupId>com.luv2code</groupId>
<artifactId>mycoolapp</artifactId>
<version>1.0.FINAL</version>
```

Example of Project Coordinates

```
<groupId>com.luv2code</groupId>
<artifactId>mycoolapp</artifactId>
<version>1.0.RELEASE</version>
```

```
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>6.0.0</version>
```

```
<groupId>org.hibernate.orm</groupId>
<artifactId>hibernate-core</artifactId>
<version>6.1.4.Final</version>
```

Adding Dependencies

```
<project ...>
 ...
 <dependencies>

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

     <dependency>
        <groupId>org.hibernate.orm</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>6.1.4.Final</version>
     </dependency>
     ...

 </dependencies>

</project>
```

Dependency Coordinates

- To add given dependency project, we need
  - **Group ID**, **Artifact ID**
  - **Version** is optional …
    - Best practice is to include the version (repeatable builds)

DevOps

- May see this referred to as: **GAV**
  - <u>G</u>roup ID, <u>A</u>rtifact ID and <u>V</u>ersion

How to Find Dependency Coordinates

- Option 1: Visit the project page (spring.io, hibernate.org etc)
- Option 2: Visit http://search.maven.org (easiest approach)

Spring Boot Project Structure

© luv2code LLC

Spring Initializr

- Spring Initializr created a Maven project for us

- Let's explore the project structure

http://start.spring.io

© luv2code LLC

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

© luv2code LLC

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
```


© luv2code LLC

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
```

© luv2code LLC

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
boot-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
...

© luv2code LLC

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

© luv2code LLC

Java Source Code

Main Spring Boot application class

Created by Spring Initializr

RestController that we created in an earlier video

© luv2code LLC

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`

© luv2code LLC

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;

  ...
}
```


© luv2code LLC

Static Content

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

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

© luv2code LLC

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.

© luv2code LLC

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

© luv2code LLC

Unit Tests

Spring Boot unit test class

Created by Spring Initializr

You can add unit tests to the file

© luv2code LLC

Spring Boot Starters

© luv2code LLC

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

© luv2code LLC

The Problem

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

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

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

Spring version

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

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

Validator -->

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

Which versions are compatible?

Very error-prone

Easy to make a simple mistake

© luv2code LLC

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

© luv2code LLC

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

© luv2code LLC

Spring MVC

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

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

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

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


© luv2code LLC

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

© luv2code LLC

Spring Initializr

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

© luv2code LLC

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`**

© luv2code LLC

Spring Initializr

Dependencies ADD ... ⌘ + B

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

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

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

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

File: pom.xml

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

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

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

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


© luv2code LLC

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**
...	

© luv2code LLC

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

© luv2code LLC

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 …

© luv2code LLC

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

© luv2code LLC

What Is In the Starter?

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

© luv2code LLC

What Is In the Starter?

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

© luv2code LLC

Spring Boot Starter Parent

© luv2code LLC

Spring Boot Starter Parent

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

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

Included in pom.xml
when using
Spring Initializr

© luv2code LLC

Spring Boot Starter Parent

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

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


© luv2code LLC

Spring Boot Starter Parent

- To override a default, set as a property

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

Specify your Java version

© luv2code LLC

Spring Boot Starter Parent

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

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

Specify version of Spring Boot

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

Inherit version from Starter Parent

No need to list individual versions
Great for maintenance!

© luv2code LLC

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
```


© luv2code LLC

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

© luv2code LLC

Spring Boot Dev Tools

© luv2code LLC

The Problem

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

© luv2code LLC

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 :-)
- For IntelliJ, need to set additional configurations ... will discuss shortly

© luv2code LLC

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

© luv2code LLC

IntelliJ Community Edition - DevTools

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

© luv2code LLC

IntelliJ Community Edition

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

© luv2code LLC

Development Process

Step-By-Step

1. Apply IntelliJ configurations

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

3. Add new REST endpoint to our app

4. Verify the app is automatically reloaded

© luv2code LLC

Spring Boot DevTools Documentation

For more details

www.luv2code.com/devtools-docs

© luv2code LLC

Spring Boot Actuator

© luv2code LLC

Problem

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

© luv2code LLC

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!

© luv2code LLC

Spring Boot Actuator

- Adding the dependency to your POM file

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


© luv2code LLC

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**
...	

Get these endpoints for free!

No additional coding required

© luv2code LLC

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

© luv2code LLC

Exposing Endpoints

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

File: src/main/resources/application.properties

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


© luv2code LLC

Info Endpoint

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

© luv2code LLC

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
```

© luv2code LLC

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**
...	

© luv2code LLC

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`

© luv2code LLC

Exposing Endpoints

- By default, only **`/health`** is 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=*
```


© luv2code LLC

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

© luv2code LLC

Development Process

Step-By-Step

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

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

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

© luv2code LLC

Spring Boot Actuator - Security

© luv2code LLC

What about Security?

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

```
"beans": {
  "endpointCachingOperationInvokerAdvisor": {
    "aliases": [],
```

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

```
    "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":
```

/health
is still available
You can disable it
if you want

© luv2code LLC

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
```


© luv2code LLC

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
```


© luv2code LLC

Customizing Spring Security

- You can customize Spring Security for Spring Boot Actuator
  - Use a database for roles, encrypted passwords etc …
- We will cover details of Spring Security later in the course

© luv2code LLC

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
```

exclude

© luv2code LLC

Spring Boot Actuator Documentation

For more details

www.luv2code.com/actuator-docs

© luv2code LLC

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`**

© luv2code LLC

Spring Boot: Run from Command-Line

© luv2code LLC

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

© luv2code LLC

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

© luv2code LLC

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

© luv2code LLC

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`**

© luv2code LLC

Option 1: Use `java -jar`

Self-contained unit
Nothing else to install

mycoolapp.jar

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

Name of our JAR file

Start our app

Also start
embedded server
(Tomcat)

© luv2code LLC

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
```


© luv2code LLC

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
```

© luv2code LLC

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

© luv2code LLC

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`**

© luv2code LLC

Spring Boot - Custom Application Properties

© luv2code LLC

Problem

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

© luv2code LLC

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`**

No additional coding or configuration required

© luv2code LLC

Development Process

Step-By-Step

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

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

© luv2code LLC

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

© luv2code LLC

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
```


© luv2code LLC

Spring Boot Properties

© luv2code LLC

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!

© luv2code LLC

Spring Boot Properties

List of Common Properties

www.luv2code.com/spring-boot-props

© luv2code LLC

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

© luv2code LLC

Spring Boot Properties

We'll review some of the properties …

© luv2code LLC

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.name=my-crazy-stuff.log
logging.file.path=c:/myapps/demo
...
```

Logging Levels

TRACE
DEBUG
INFO
WARN
ERROR
FATAL
OFF

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

© luv2code LLC

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

Default timeout
30 minutes

© luv2code LLC

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

© luv2code LLC

Security Properties

File: src/main/resources/application.properties

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

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


© luv2code LLC

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

© luv2code LLC

Spring Boot Properties

List of Common Properties

www.luv2code.com/spring-boot-props

© luv2code LLC

Development Process

Step-By-Step

1. Configure the server port

2. Configure the application context path

© luv2code LLC