

NS4307

Network Programming

Lecture 7: Java Web Programming I

Disclaimer

- Some parts of the notes is based on the online course provided in Treehouse (teamtreehouse.com).
 - **Course:** Spring
 - **Teacher:** Chris Ramacciotti

Overview

- I am sure that you have accessed many websites:
 - For Example: Facebook, Twitter and Instagram.
- All of these are examples of web applications.

Requirement

- Before we start, please make sure you have the following installed in your computer.
 1. Java SE Development Kit (JDK 8)
 - <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
 2. Java Runtime Environment (JRE)
 - Usually included with JDK.
 3. Spring Tool Suite
 - It is an Eclipse-based development environment that is customized for developing Spring applications.
 - <http://spring.io/tools/sts>
 - You may download a standalone Spring Tool Suite or if you're using eclipse IDE you may install Spring Tool Suite directly from the Eclipse Marketplace.

Web Application

- A web application is a client/server application where user communicates with a web server through a web browser which the web server send back a response.
- To make it easier to create your own web application, there are many frameworks available.

Web Application (cont.)

- Usually there are frameworks available to most of the popular programming languages. Example:
 - **Java:** Spring Framework (<http://spring.io>), Play Framework (<https://www.playframework.com/>), JavaEE (<http://www.oracle.com/technetwork/java/javaee/overview/index.html>)
 - **PHP:** Laravel (<https://laravel.com/>), Symfony (<https://symfony.com/>)
 - **Ruby:** Ruby on Rails (<http://rubyonrails.org/>)
 - **JavaScript:** Angular JS (<https://angular.io/>), Node JS (<https://nodejs.org/en/>)
 - **Python:** Django (<https://www.djangoproject.com/>)
- There are more frameworks available if you want to search for it. Since you are familiar with Java languages. For this module, we will be using Spring Framework.

Spring Framework

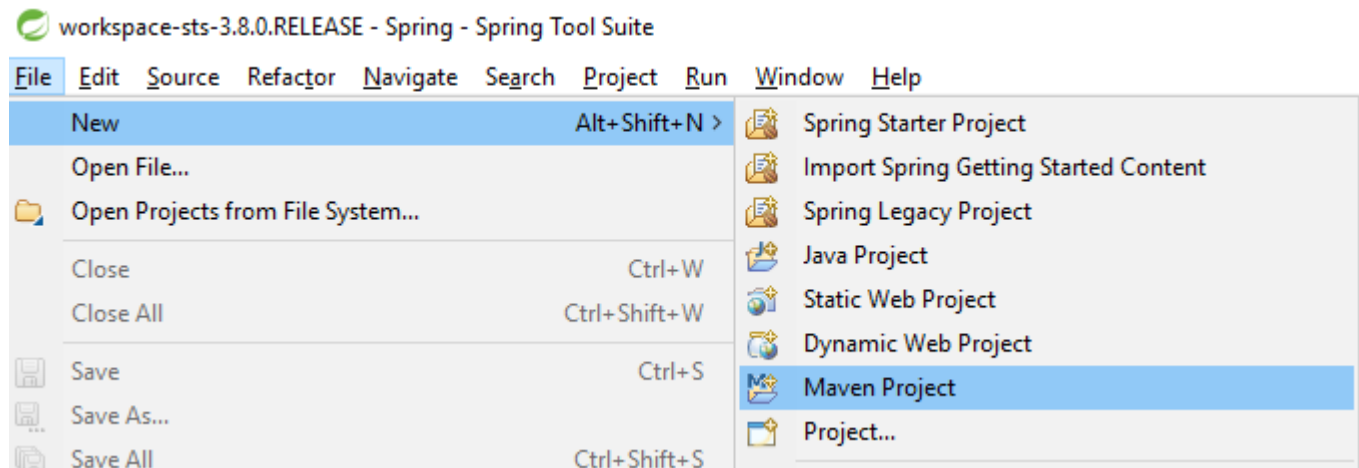
- Spring Framework is a tool to help you write a Java Application that can run on a web server.
- Spring Framework consists of many different components, but we will cover only the basic components that allows you to create a basic web application.

Maven

- Before we start creating a web application using Java, I would to introduce to you a build automation tool called Maven.
- Maven provides a lot of features but the feature that we are going to use is its dependency management.
 - This allows you to download any JARs required for building your project from a central JAR repository (<http://search.maven.org/>).
 - This means that you don't need to manually download each third party libraries and import it into your project. Maven will do it for you.
- It is included with Spring Tool Suite, so you don't need to worry about installing it.

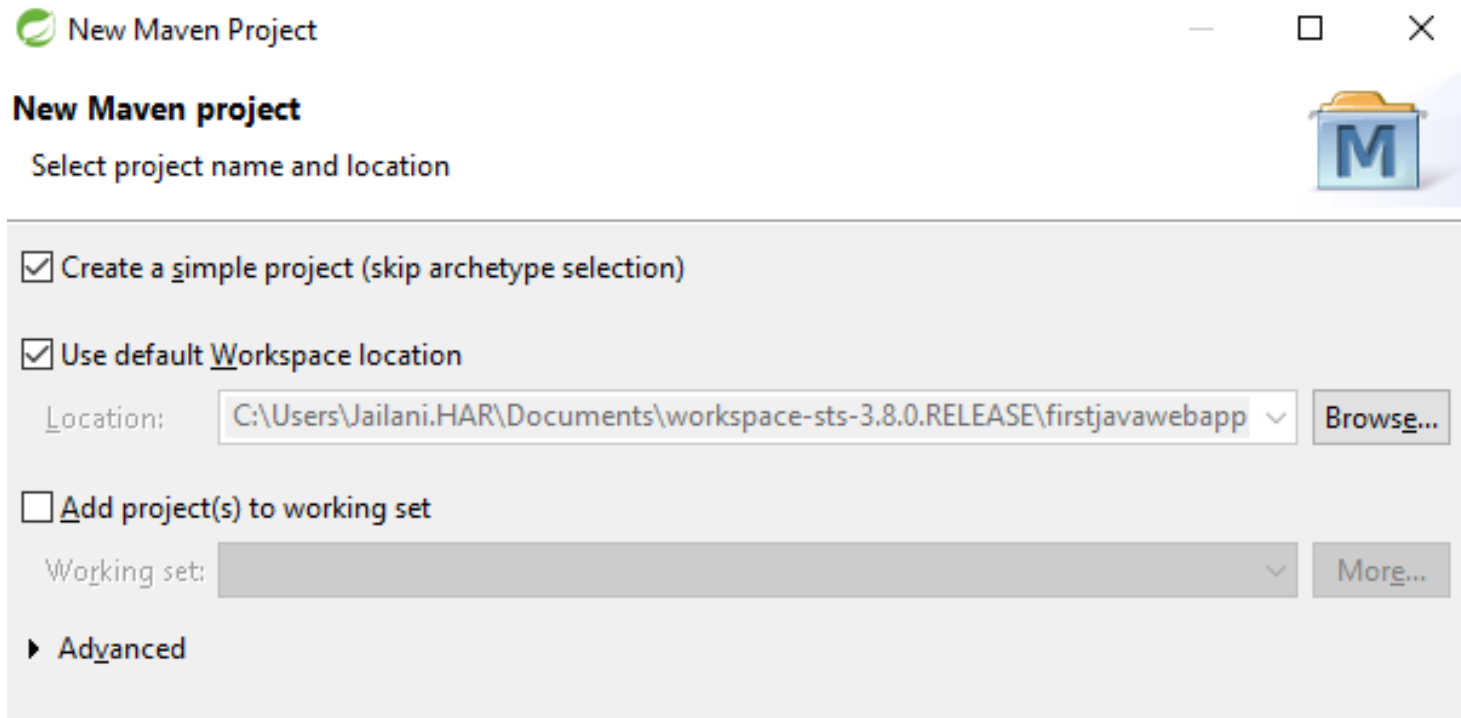
Create Maven Project

- So to make a Maven Project in Spring Tool Suite:
 - File > New > Maven Project



Create Maven Project (cont.)

- Lets create a simple project, follow the setting as below and click Next.



New Maven Project

New Maven project

Select project name and location

☒ Create a simple project (skip archetype selection)

☒ Use default Workspace location

Location: C:\Users\Jailani.HAR\Documents\workspace-sts-3.8.0.RELEASE\firstjavawebapp

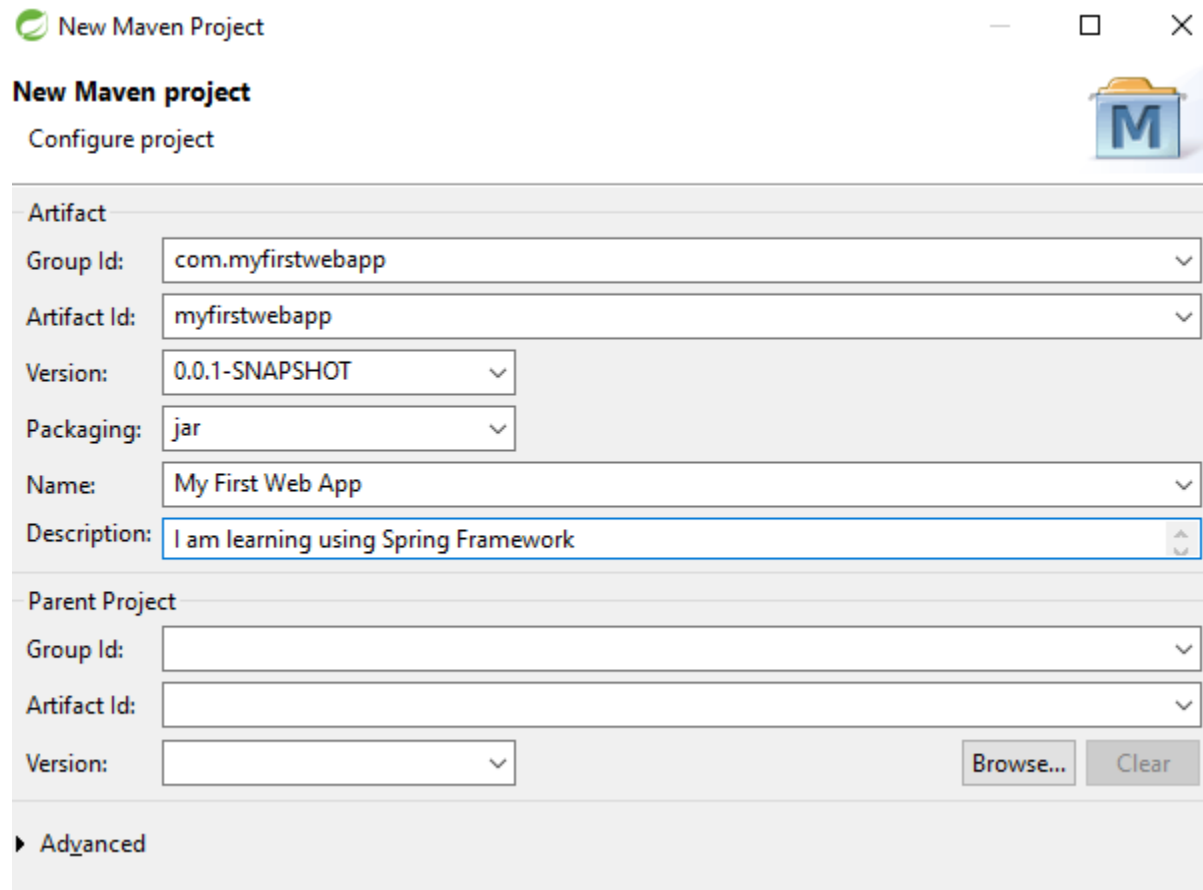
☐ Add project(s) to working set

Working set:

► Advanced

Create Maven Project (cont.)

- The values is up to you and please leave out Parent Project blank. Then click Finish.



New Maven Project

New Maven project
Configure project

Artifact

Group Id: com.myfirstwebapp

Artifact Id: myfirstwebapp

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name: My First Web App

Description: I am learning using Spring Framework

Parent Project

Group Id:

Artifact Id:

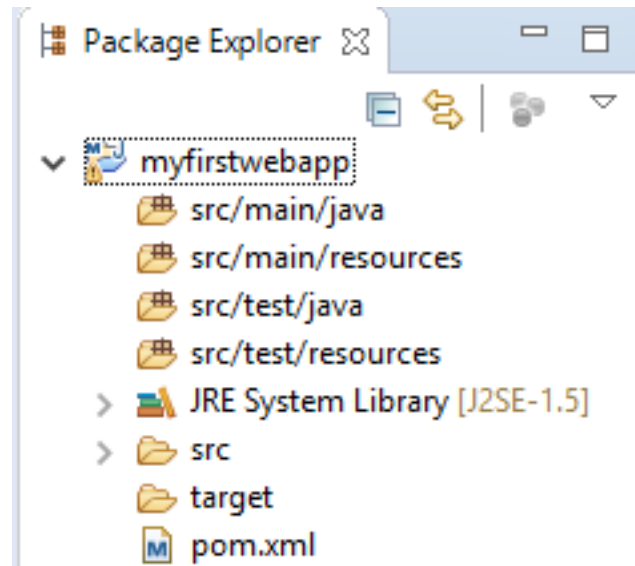
Version:

Browse... Clear

Advanced

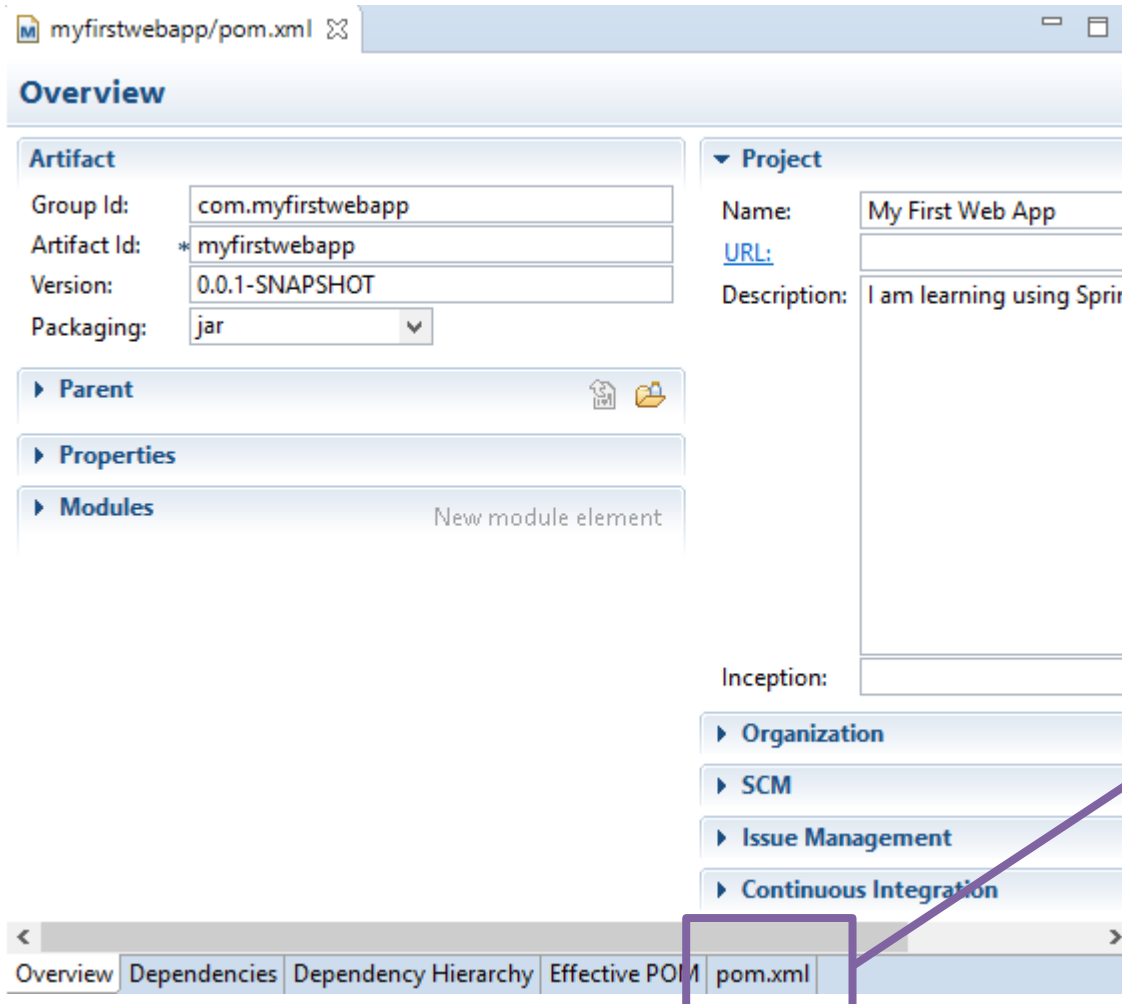
Create Maven Project (cont.)

- So you have your project created as followed.



- Then open up pom.xml file.

Create Maven Project (cont.)



myfirstwebapp/pom.xml

Overview

Artifact

Group Id: com.myfirstwebapp

Artifact Id: *myfirstwebapp

Version: 0.0.1-SNAPSHOT

Packaging: jar

Project

Name: My First Web App

URL:

Description: I am learning using Spring

Parent

Properties

Modules New module element

Inception:

Organization

SCM

Issue Management

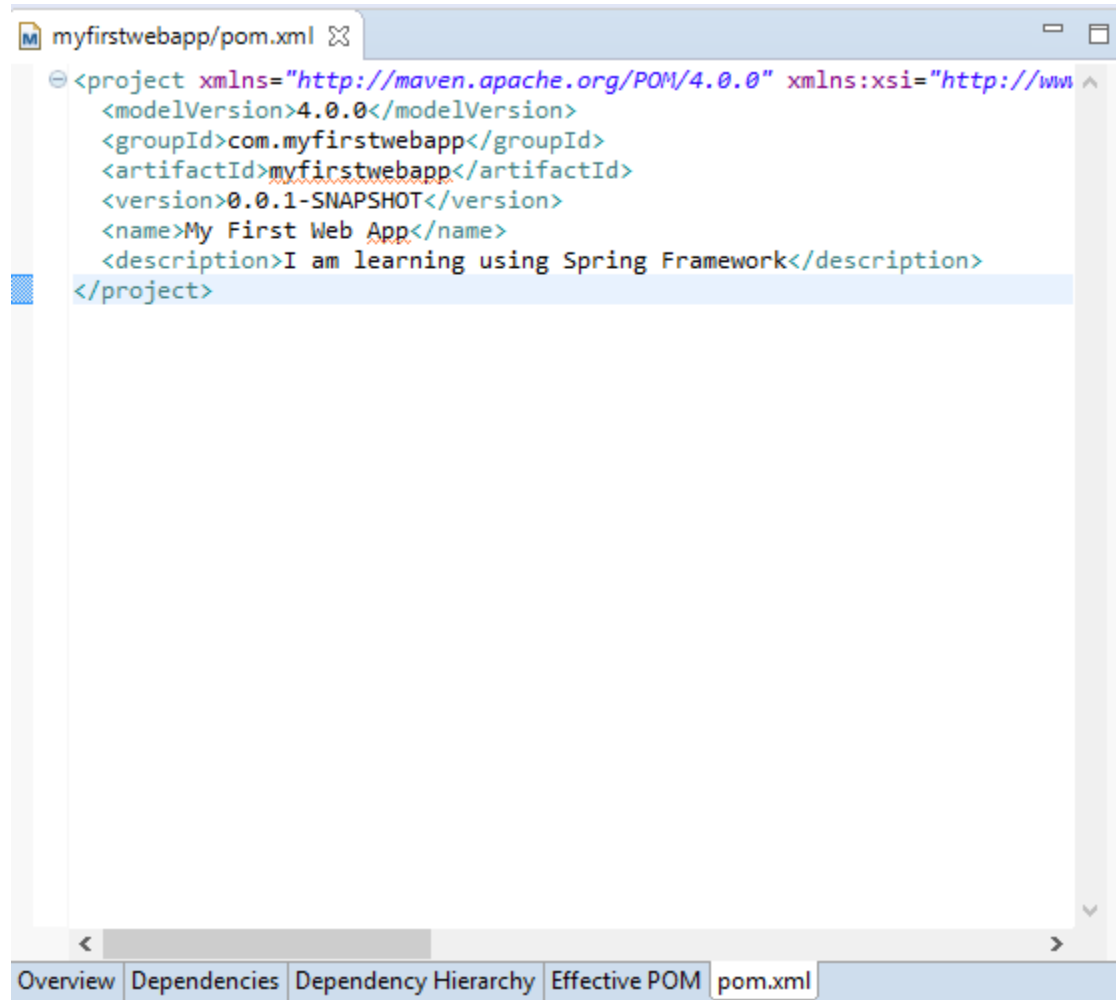
Continuous Integration

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

Click here to open pom.xml to edit in XML format.

Create Maven Project (cont.)

- We are going to edit this in the future to set up Spring Framework.



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.myfirstwebapp</groupId>
    <artifactId>myfirstwebapp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>My First Web App</name>
    <description>I am learning using Spring Framework</description>
</project>
```

The screenshot shows an IDE window titled 'myfirstwebapp/pom.xml'. The XML content is displayed in a code editor with syntax highlighting. The bottom of the window features a tabbed interface with 'Overview', 'Dependencies', 'Dependency Hierarchy', 'Effective POM', and 'pom.xml' (the active tab).

Spring Boot

- Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can “just run”.
- Features:
 - Create stand-alone Spring applications.
 - Embed Tomcat.
 - Automatically configure Spring whenever possible.
 - and more (<http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/>).

Configure Spring Boot to your Maven Project

- To get you up and running without much configuration to your current project, all you need to do is to define the parent in the pom.xml file.
- Between the <project> </project> tag, you need to add this lines of code.

<parent>

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

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

<version>1.4.0.RELEASE</version>

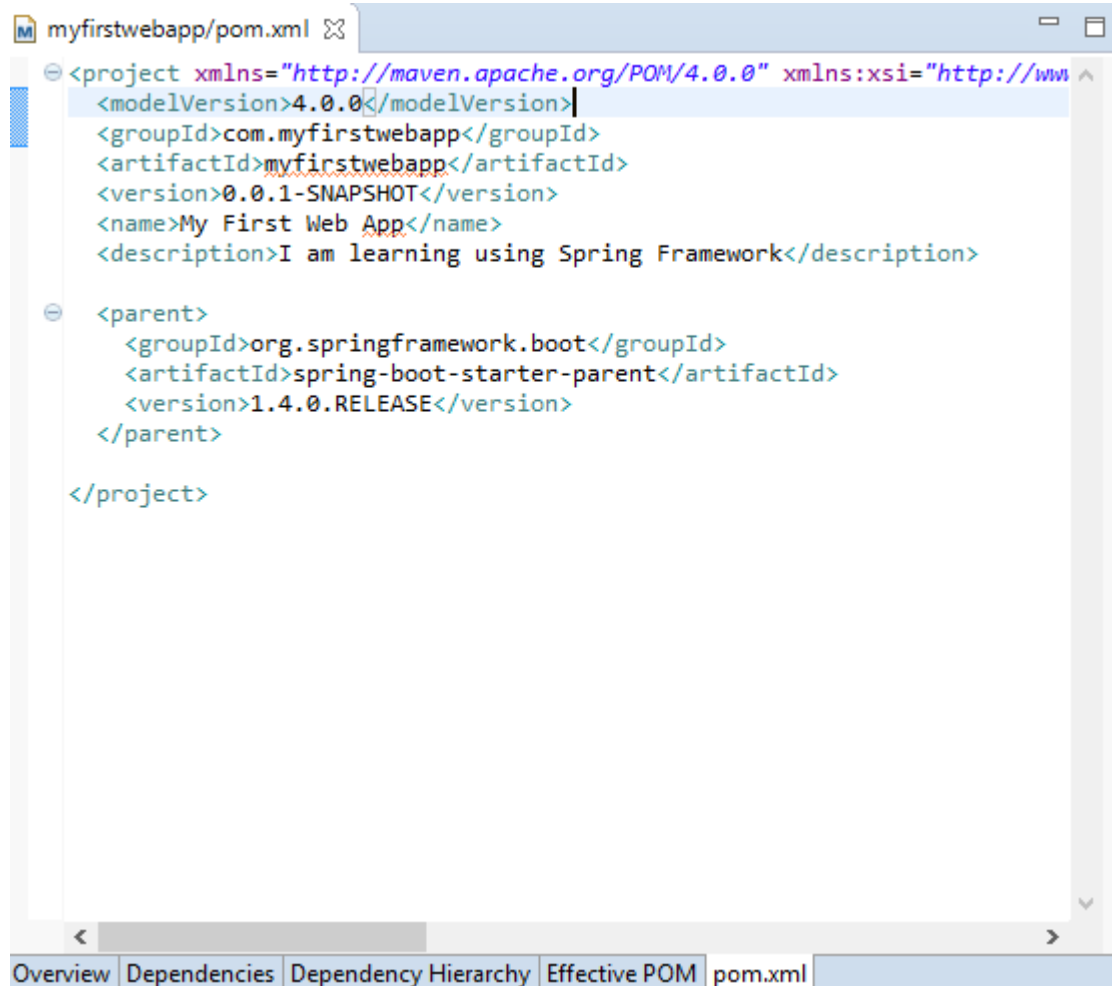
</parent>

Configure Spring Boot to your Maven Project

- Tag `<parent></parent>` is to state the project pom.xml has a parent.
- Tag `<groupId></groupId>` is the package name
- Tag `<artifactId></artifactId>` is the file name.
- Tag `<version></version>` is the version of the parent.

Configure Spring Boot to your Maven Project (cont.)

- This means that your Project will inherit the pom file provided by Spring Boot Starter Parent.
 - So that means all dependencies needed for Spring Boot will be included into your own project.



```
myfirstwebapp/pom.xml
<?xml version="1.0" encoding="UTF-8" ?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.myfirstwebapp</groupId>
    <artifactId>myfirstwebapp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>My First Web App</name>
    <description>I am learning using Spring Framework</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.4.0.RELEASE</version>
    </parent>

</project>
```

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

Dependencies

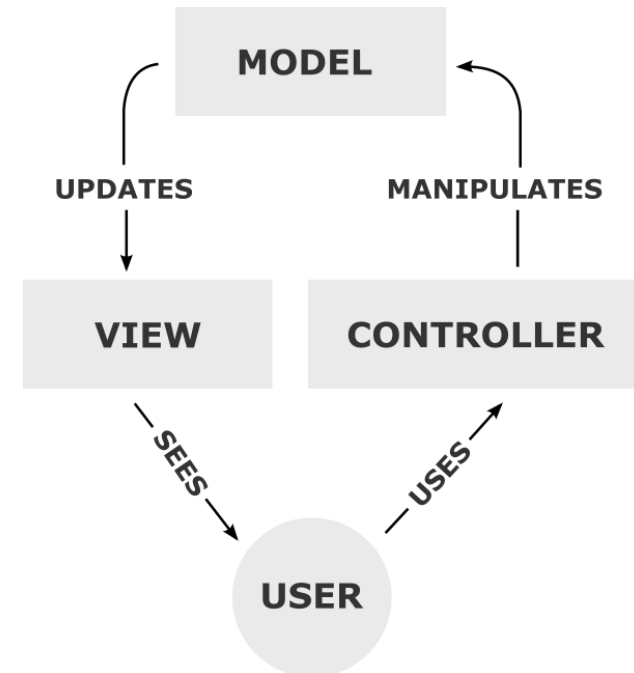
- Now we need to add the required tools for you to make a Spring Web Application. The dependencies that we are going to add are:
 - Spring Boot Starter Web
 - Spring Boot Starter Thymeleaf
 - Spring Boot Starter Test
 - NekoHTML
- We will add further dependencies when we are creating Web Application that handle Database in future lecture.

Spring Boot Starter Web

- Since we are going to develop a web application, we need to add the tools to your project to help you create a Spring Web Application.
- It is called the **Spring Boot Starter Web** which is for building web applications using Spring MVC. Uses Tomcat as the default embedded container.
- Model View Controller (MVC) is basically a design pattern for implementing user interfaces on computers.

Model View Controller

- It is using the **Model View Controller (MVC)** perspective.
 - The **HTML file** that contains the description of the user interface **is the view**.
 - The **controller is a Java class**, implementing components that handle the URI, POST and GET Request.
 - The **model consists of domain objects, defined on the Java side**, that you connect to the view through the controller.



Configure Spring Boot Starter Web

- To configure Spring Boot Starter Web into your current project. You need to add Spring Boot Starter Web into your dependencies list.
- Between the `<project>` `</project>` tag, you need to add this tag `<dependencies>` `</dependencies>`
 - This tag `<dependencies>` `</dependencies>` is where you will list all dependencies that you need for your project.
 - For each dependency, you need to add a tag `<dependency>` `</dependency>`

Configure Spring Boot Starter Web (cont.)

- In the tag `<dependency>` `</dependency>`, you need to identify the `groupId`, `artifactId` and version of the dependency that you want to include to your project.

`<dependencies>`

`<dependency>`

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

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

`</dependency>`

`</dependencies>`

Spring Boot Starter Thymeleaf

- Thymeleaf is a modern server-side Java template engine for both web and standalone environments. It is a starter for building MVC web applications.
- Add the following as another dependency.

<dependency>

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

<artifactId>spring-boot-starter-thymeleaf**</artifactId>**

</dependency>

Spring Boot Test Starter

- Spring Boot Test Starter is a starter for testing Spring Boot applications with libraries including Junit, Hamcrest and Mockito.
- Add the following as another dependency.

<dependency>

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

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

<scope>test</scope>

</dependency>

NekoHTML

- NekoHTML is a simple HTML scanner and tag balancer. This tool will help when we try to use templates that uses HTML5 tags.
- Add the following as another dependency.

`<dependency>`

`<groupId>net.sourceforge.nekohtml</groupId>`

`<artifactId>nekohtml</artifactId>`

`</dependency>`

Setting up Build

- Since we are making a Spring Web Application, we want to compile and deploy our application in an embedded web server.
- So you don't need to install tomcat in our local or remote server.
- Between the `<project>` `</project>` tag, you need to add this lines of code.

Setting up Build

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

Final pom.xml

```

myfirstwebapp/pom.xml portfolio/pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.xsi.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.myfirstwebapp</groupId>
  <artifactId>myfirstwebapp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>My First Web App</name>
  <description>I am learning using Spring Framework</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.4.0.RELEASE</version>
  </parent>

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

Final pom.xml (cont.)

```
    </dependency>
    <dependency>
      <groupId>net.sourceforge.nekohtml</groupId>
      <artifactId>nekohtml</artifactId>
    </dependency>
  </dependencies>

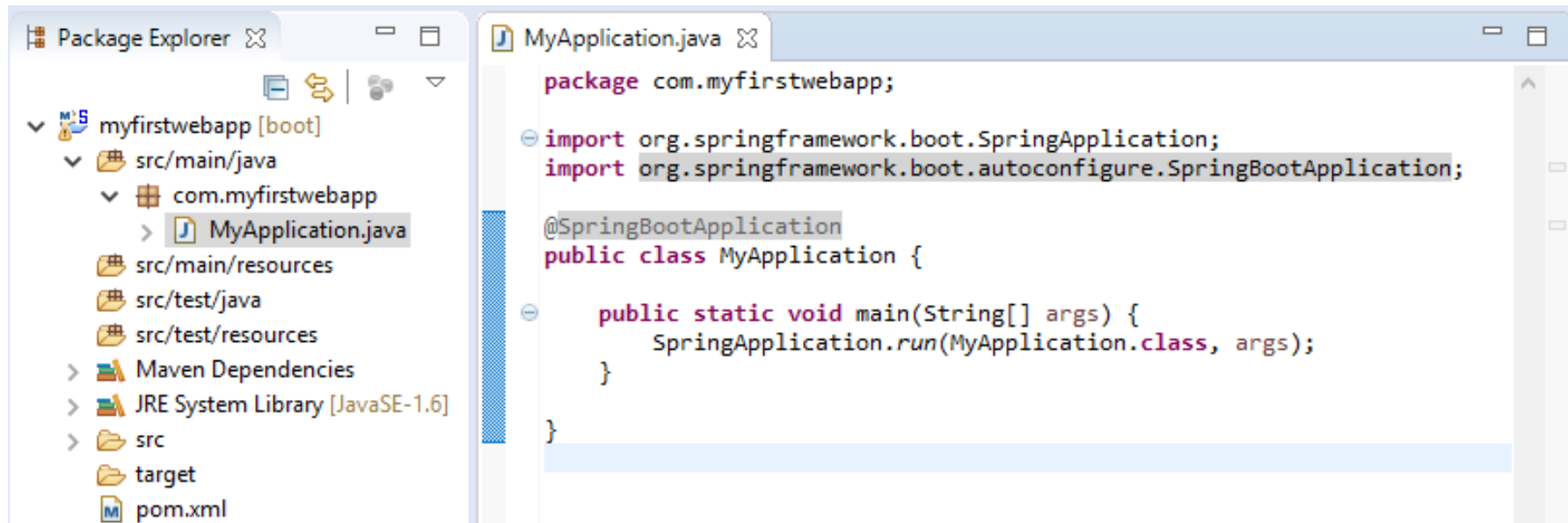
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <configuration>
          <executable>true</executable>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

Main Class

- To be able to run your Spring Web Application, you need a java class with a main method.
 - Create a package and a java class in your src/main/java folder.
 - In the Java class, add a main method.
 - In the main method, add the following statement:
 - `SpringApplication.run(ClassName.class, arg);`
 - Then we want the application to be auto configured, we add annotation `@SpringBootApplication` on top of the class.
- Note: Don't forget to import all necessary classes.

Main Class (cont.)



The screenshot shows an IDE window with two panes. The left pane is the Package Explorer, showing a project named 'myfirstwebapp [boot]'. Under 'src/main/java', there is a package 'com.myfirstwebapp' containing a file 'MyApplication.java'. The right pane shows the code for 'MyApplication.java'. The code is as follows:

```
package com.myfirstwebapp;

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

@SpringBootApplication
public class MyApplication {

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


Faster Way to Create Maven Project

- On the previous slides, I showed you how to create and set up your Maven Project.
- But Spring Framework does provide easier way to generate Maven Project with Spring Boot immediately.
- It is called Spring Initializr (<https://start.spring.io/>).

Spring Initializr

- In there you can type in your Group and Artifact ID.
- And add dependencies:
 - Web
 - Thymeleaf
- But you need to add NekoHTML manually.
- Then click Generate Project.

SPRING INITIALZR bootstrap your application now

Generate a Maven Project with Spring Boot 1.4.0

Project Metadata

Artifact coordinates

Group

Artifact

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

Web Thymeleaf

Generate Project alt + ↵

Don't know what to look for? Want more options? [Switch to the full version.](#)

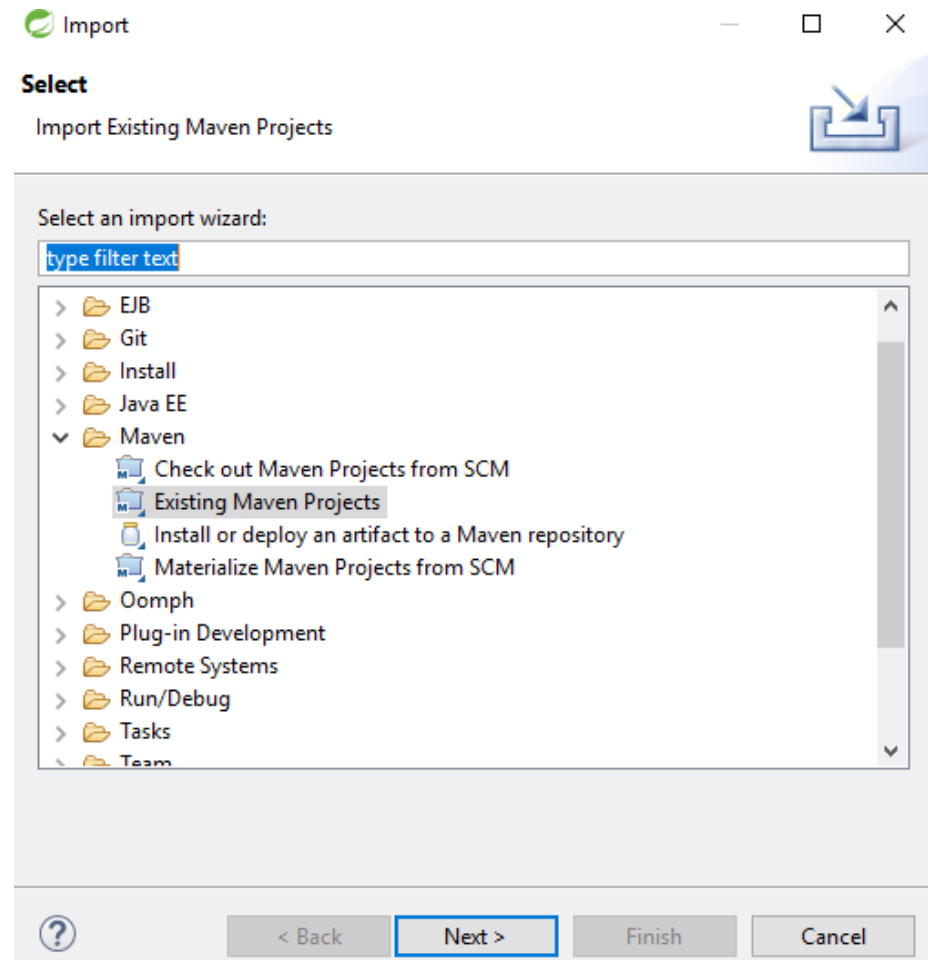
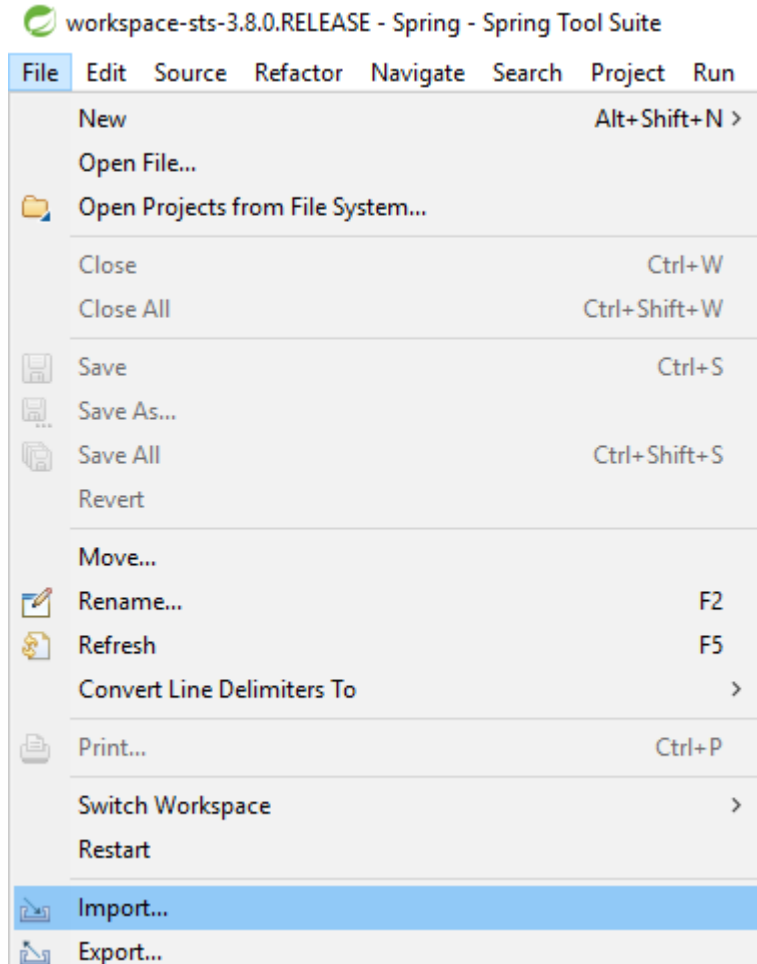
Spring Initializr (cont.)

- Your browser will automatically download the zip file for the project.
- You need to unzip the file and then copy it into your workspace.
- Then all you need to do is to import the project into your Spring Tool Suite.

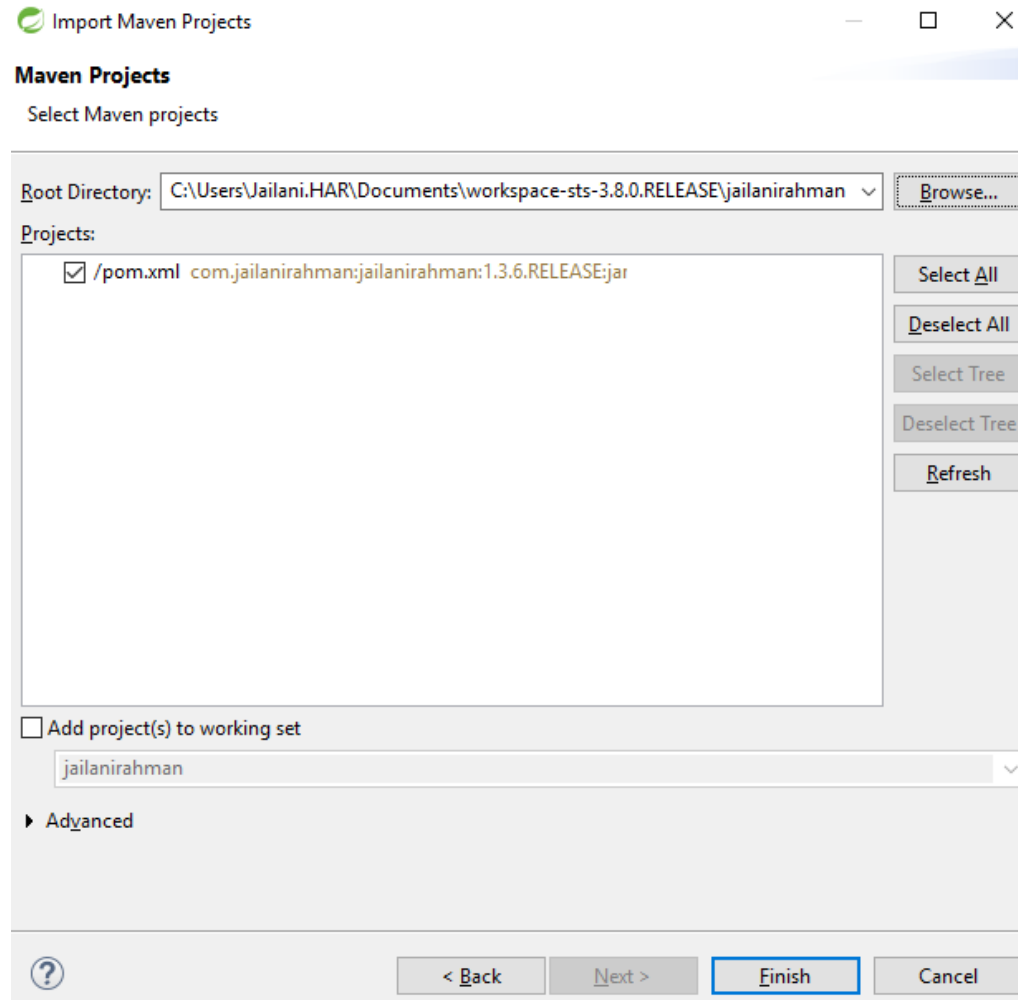
Import Existing Maven Project

- To import Existing Maven Project.
 - File > Import
 - In the import window:
 - Click Maven > Then choose Existing Maven Project
 - After that click Next.
 - On Root Directory, specify the location of the unzipped project that you generated using Spring Initializr or click on Browse to navigate through your files.
 - Finally click Finish.
- Note: It will take a while for your Maven project to load.

Import Existing Maven Project (cont.)

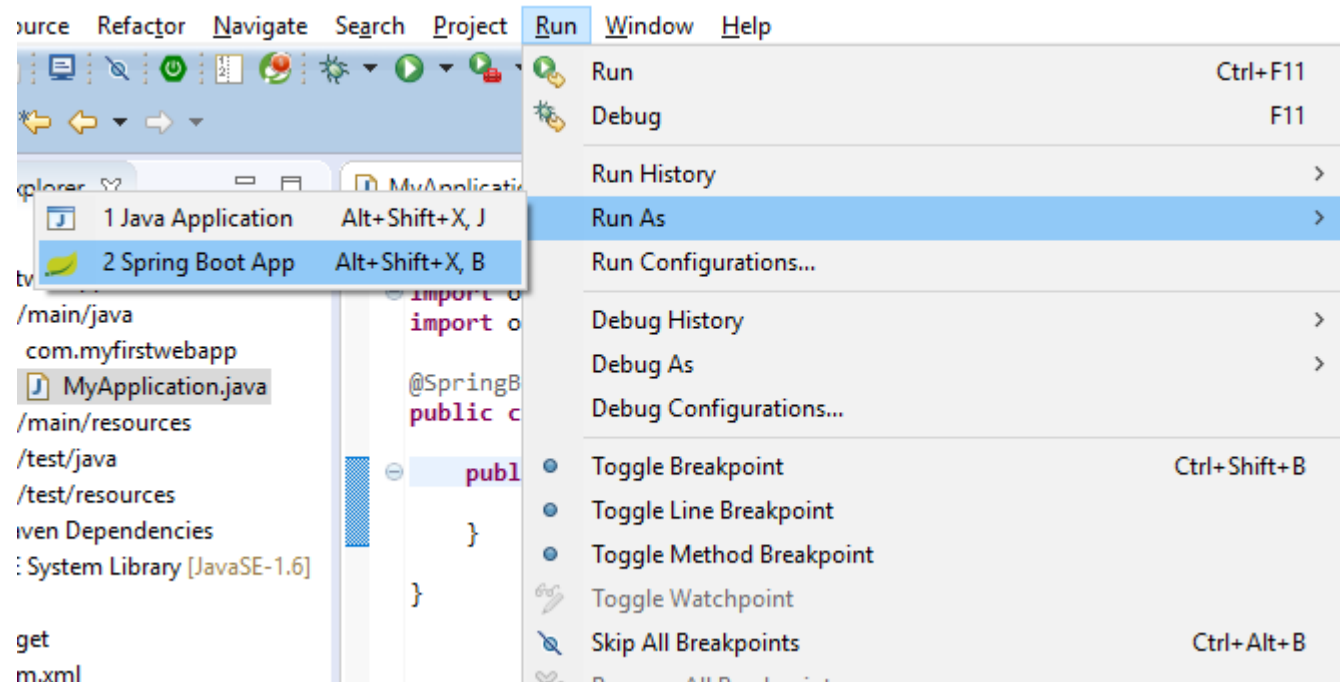


Import Existing Maven Project (cont.)



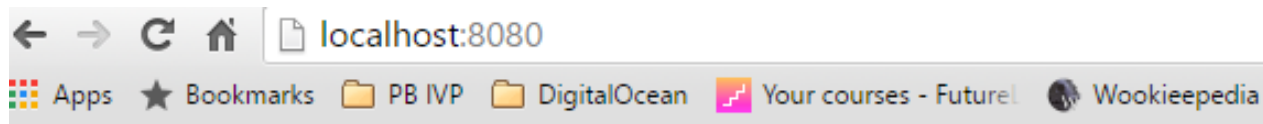
Running your main class

- To run your main class, it is the same as when you are trying to run a Java Application in eclipse but instead of running as Java Application you need to run it as Spring Boot App.



Running your main class (cont.)

- You can access your web application using a web browser.
 - localhost:8080
- Currently, we don't have any web pages to show. So when you run your web application, it shows you an error page.



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Aug 03 16:35:19 SGT 2016

There was an unexpected error (type=Not Found, status=404).

No message available

Why only error page

- Please make sure you can access your web application from your web browser, localhost:8080 .
- The reason why you can only view an error page is because the web application does not handle requests to any URIs.
- We need another component which is Spring Controller.

Spring Controller

- First thing first, we need to write a Spring controller to handle requests to certain URIs.
 - URL: Uniform Resource Locator
 - In general a URL is an address that includes a method for locating a resource, with a protocol like HTTP or FTP.
 - Example: **http://www.pb.edu.bn/app**
 - URI: Uniform Resource Identifier
 - URI is the resource location where no host name and protocol are included.
 - Example: **/app**

Spring Controller (cont.)

- Lets create a new package called controller in src/main/java directory and create a Java class into the package.
- To indicate that the Java class is a spring controller, we add an annotation on the class itself called @Controller, on top of the class.
- Then create a method that return String data type to response to URIs request.

Example Spring Controller

To indicate the
class is Spring
Controller →

```
package controller;  
  
import org.springframework.stereotype.Controller;  
  
@Controller  
public class ControllerClass {  
  
}
```

Request Mapping

- For now, we are going to use this single controller to handle URI requests.
- Add a method to handle all requests to our home page.
- To indicate which method to run for certain URI request, you need to add `@RequestMapping` annotation.
- You can add several elements with the `@RequestMapping` annotation. But for this module, we will just use one of them.

Request Mapping (cont.)

- The element is value. So the annotation will look like this.

`@RequestMapping(value = "/")`

- The forward slash ("/") will indicate that the method with this `@RequestMapping` value will be handling request to our application root or home page.
- Since we still have yet implemented a view component, we will add additional annotation called `@ResponseBody`.
 - This will indicate that the String we return in the method be used as the response without any further processing.

Example Request Mapping

```
package controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class ControllerClass {

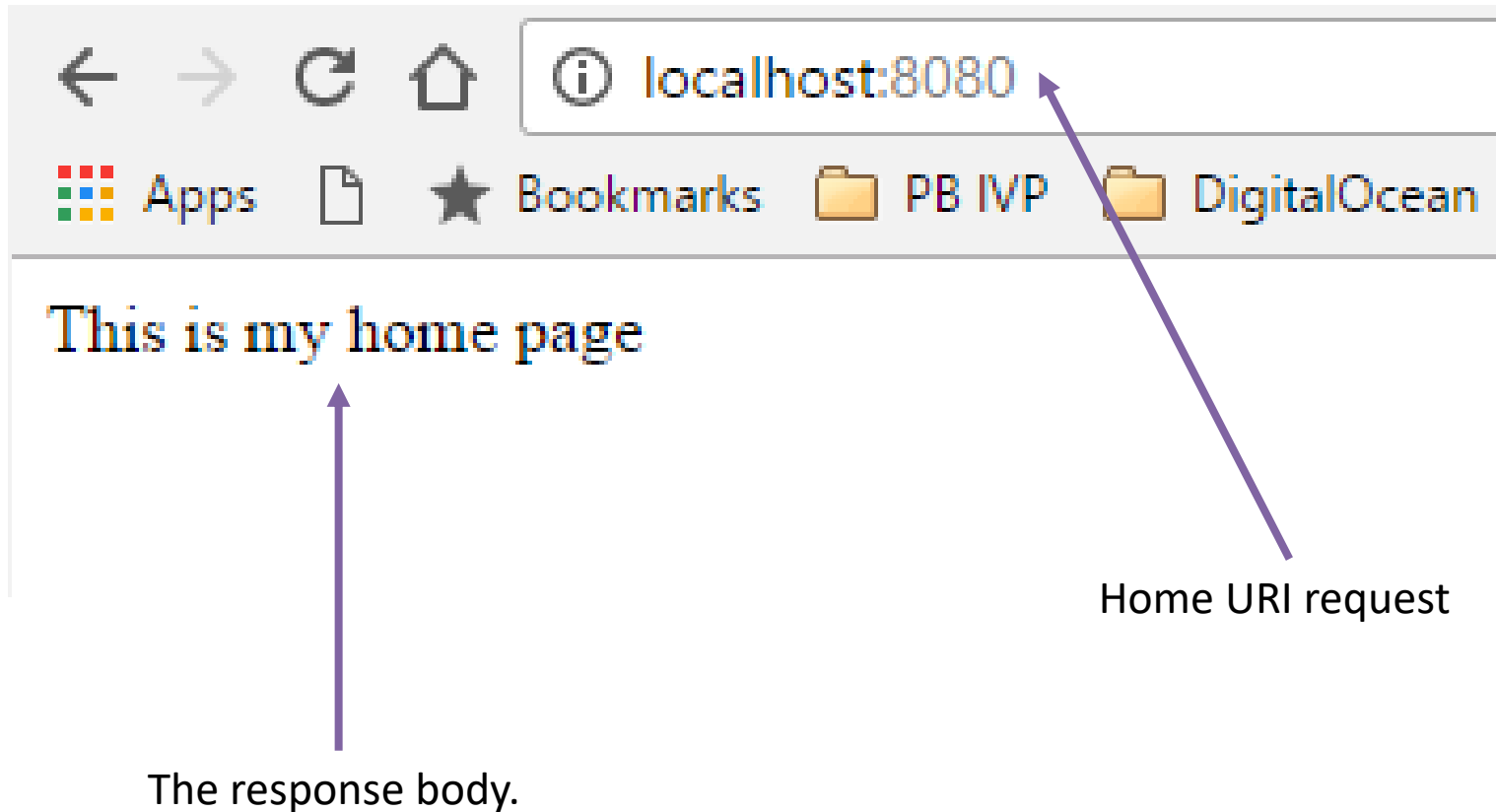
    @RequestMapping(value="/")
    @ResponseBody
    public String home() {
        return "This is my home page";
    }
}
```

To handle the
URI request.
In this case
home page.

This is to
indicate the
return value is to
be response
body.

The response body.

Example Request Mapping (cont.)



Example Request Mapping (cont.)

- If you put the value for @RequestMapping as “/app”

```
@RequestMapping(value="/app")  
@ResponseBody  
public String app() {  
    return "Welcome to my app page";  
}
```



Welcome to my app page