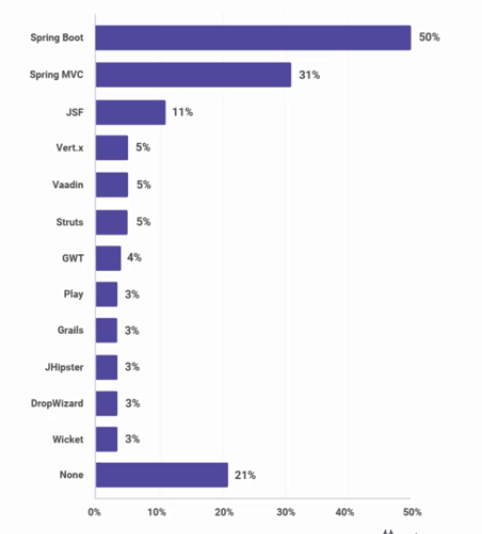
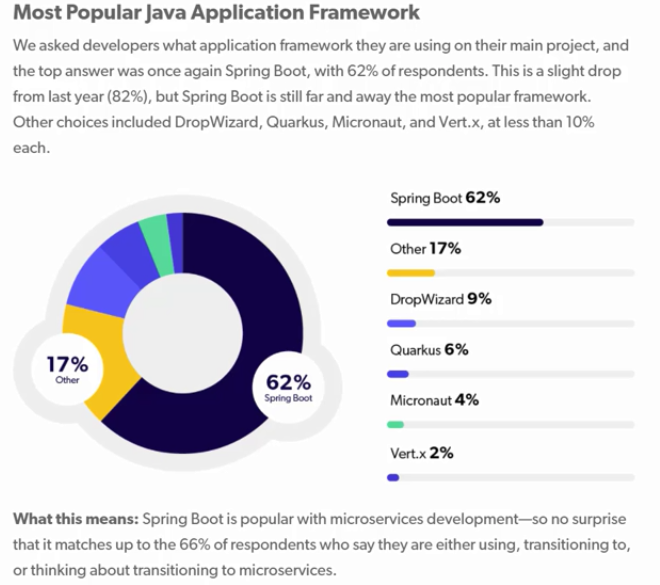
**SPRING BOOT NOTES BY VIJAY SIR**

**What is Spring?**

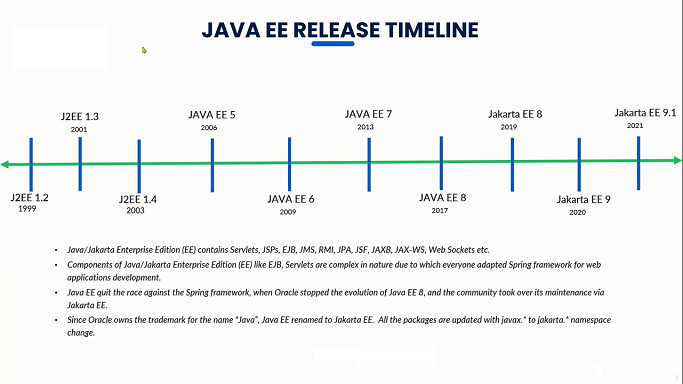
* **Spring is a very popular framework for building Java applications.**
* **Provides a large number of helper classes and annotations.**
* **The Spring Framework is a well-established, strong, and highly adaptable framework designed for creating web applications using Java.**
* **Spring makes programming Java quicker, easier, and safer. Focusing on speed, simplicity, and productivity has made it the world’s most popular Java framework.**
* **It doesn’t matter that you are intend to build secure, reactive, cloud-based microservices for the web or complex streaming data flows for the enterprise, spring has the tool to help.**
* **Spring is an alternative to EJB’s and struts which were used in the 2000’s offering more simplicity, more features, as well as offering third party integrations.**
* **Spring framework is an open-source.**
* **Spring website – official 🡺** [**www.spring.io**](http://www.spring.io)
* **Lightweight development with Java POJOs(Plain-Old-Java-Objects)**
* **Dependency injection to promote loose coupling(EJBs are heavy weight)**
* **Minimize boilerplate Java code.**

**Note: Java EE now called as Jakarta EE because of Oracle Renaming**

****

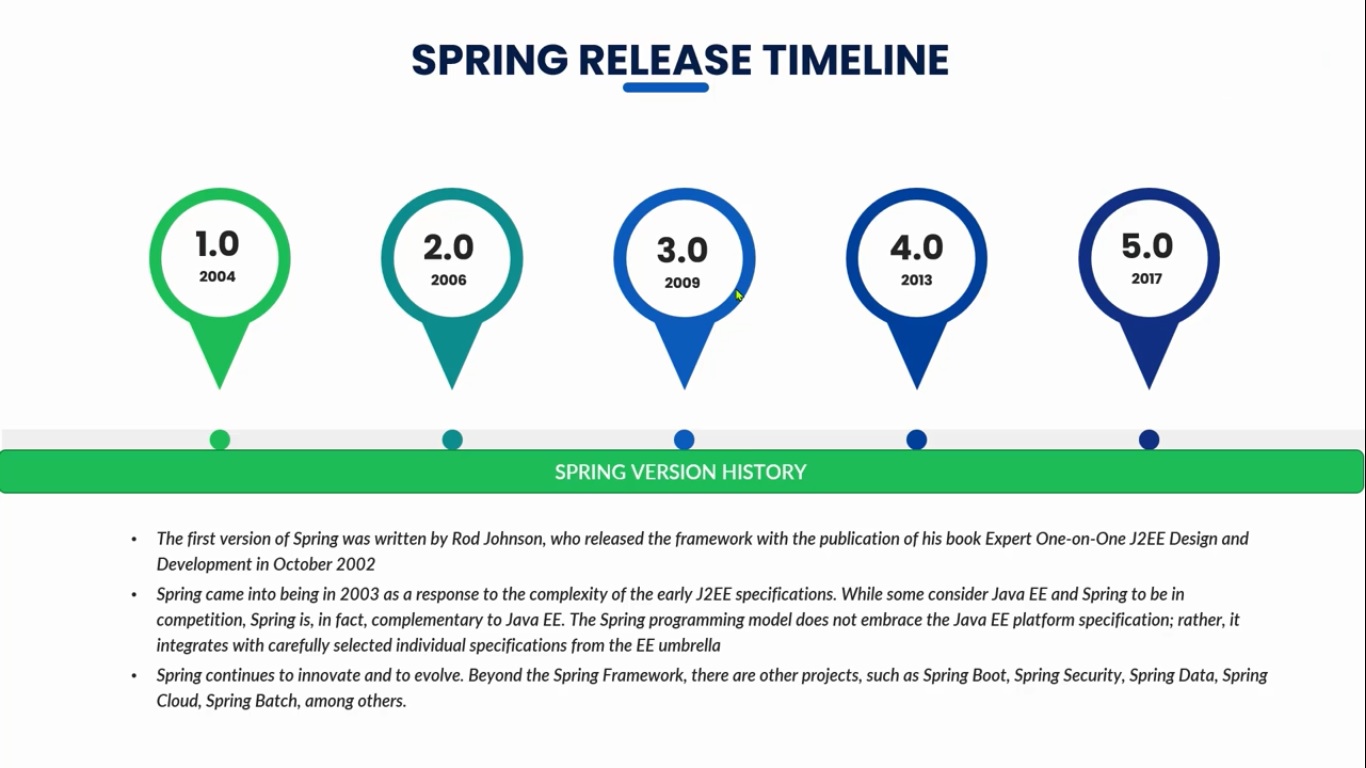


1. **Java EE Release Timeline**

****

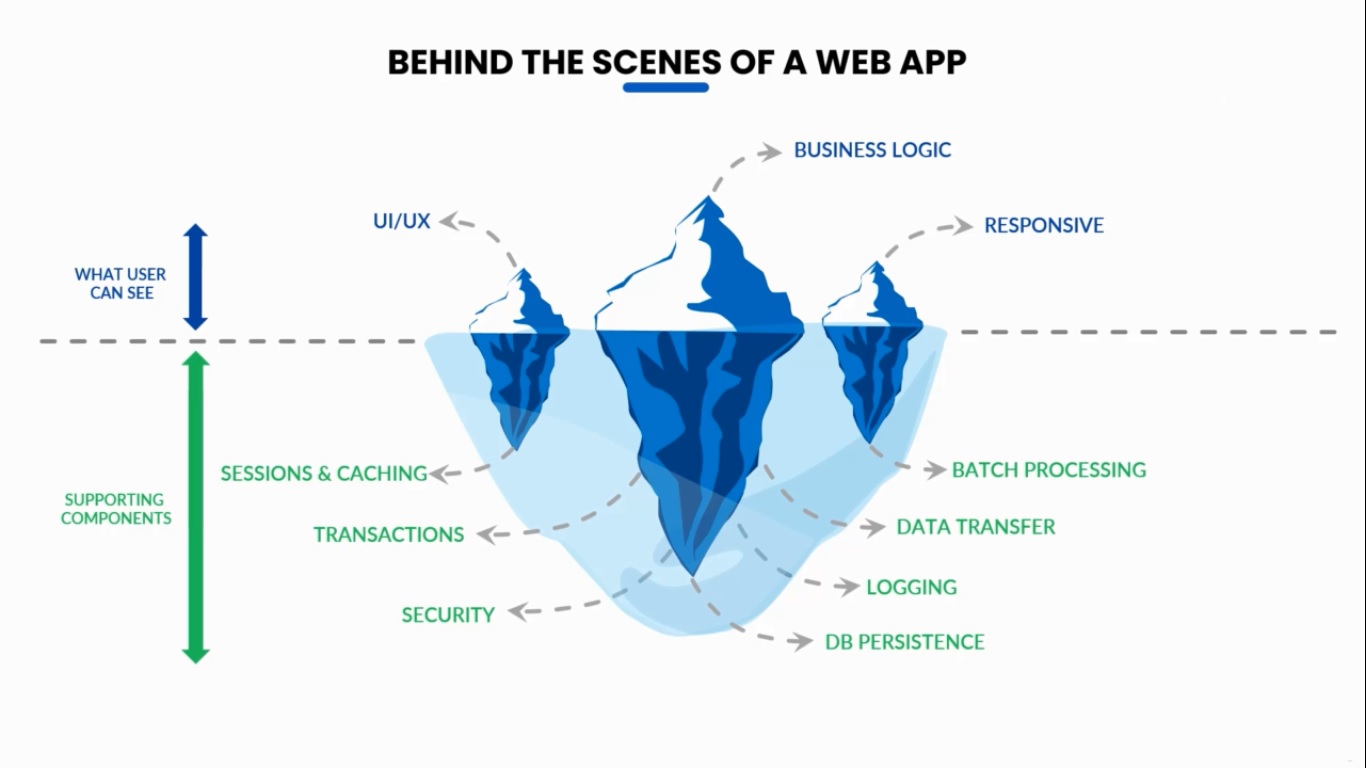
* **Spring framework developed by the person called Rod Johnson**
* **Roderick "Rod" Johnson is an Australian computer specialist who created the Spring Framework and co-founded SpringSource**
* **He released the framework with the publication of his book expert one-on-one J2EE design and development in October 2002.**

1. **Spring Release Timeline**

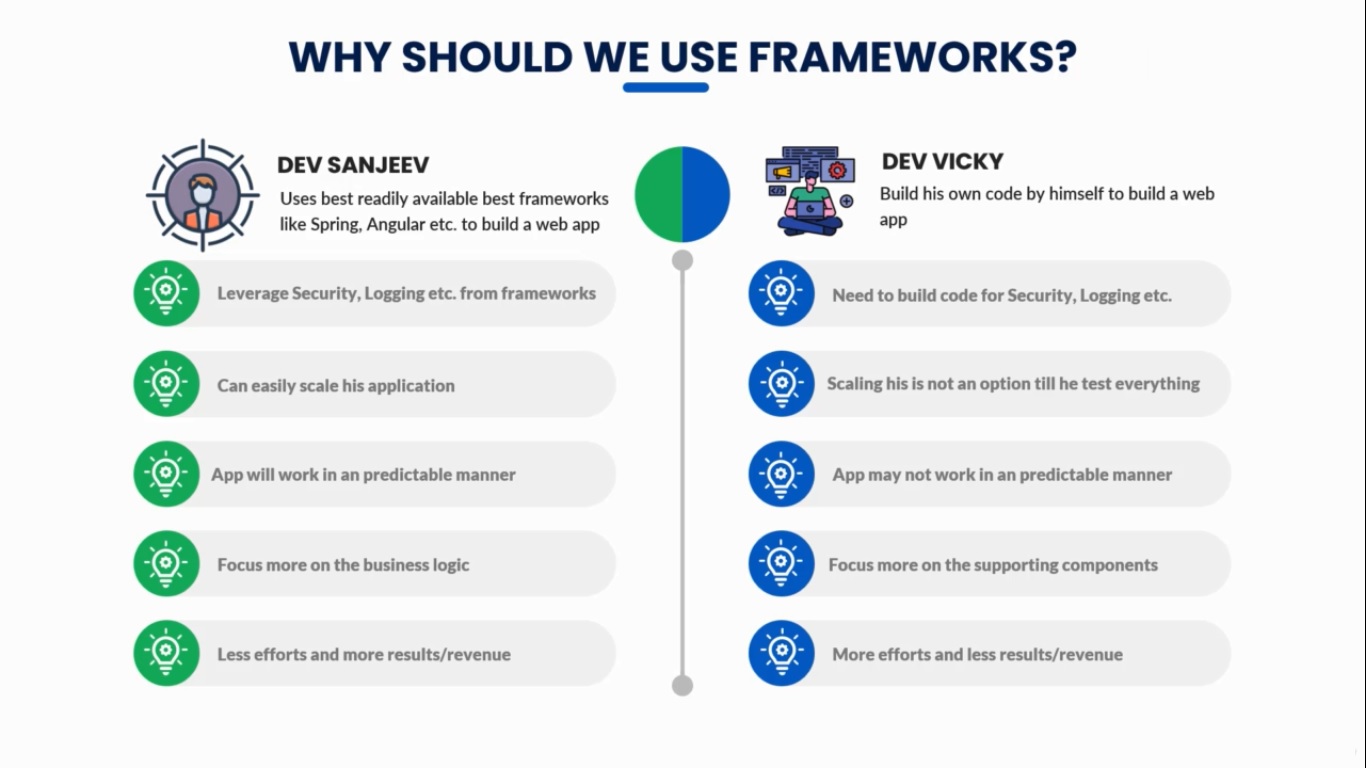


**Note:** Spring Framework 6.0 has been released on 16 November 2022 and came with a Java 17+ baseline and a move to Jakarta EE 9+ (in the jakarta namespace), with a focus on the recently released Jakarta EE 10 APIs such as Servlet 6.0 and JPA 3.1.

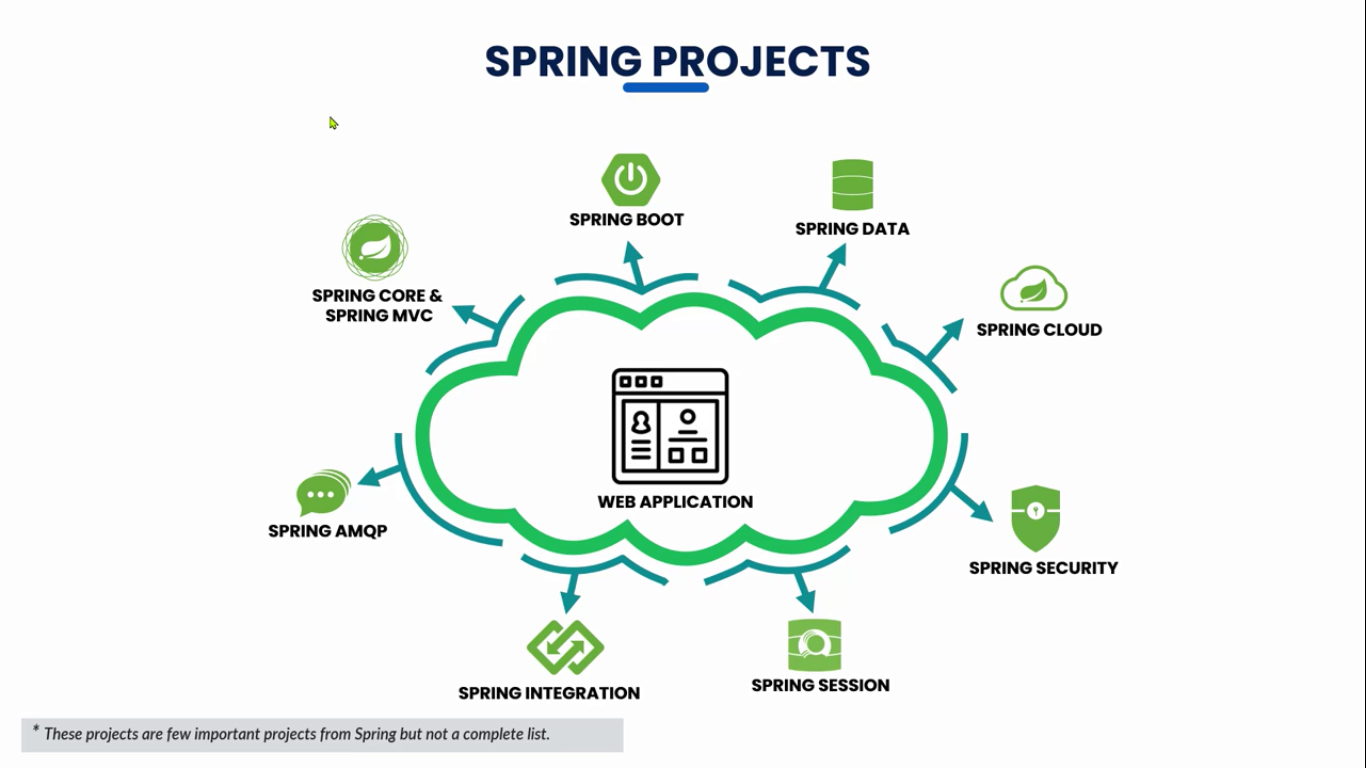
1. **Spring Release Timeline**

****

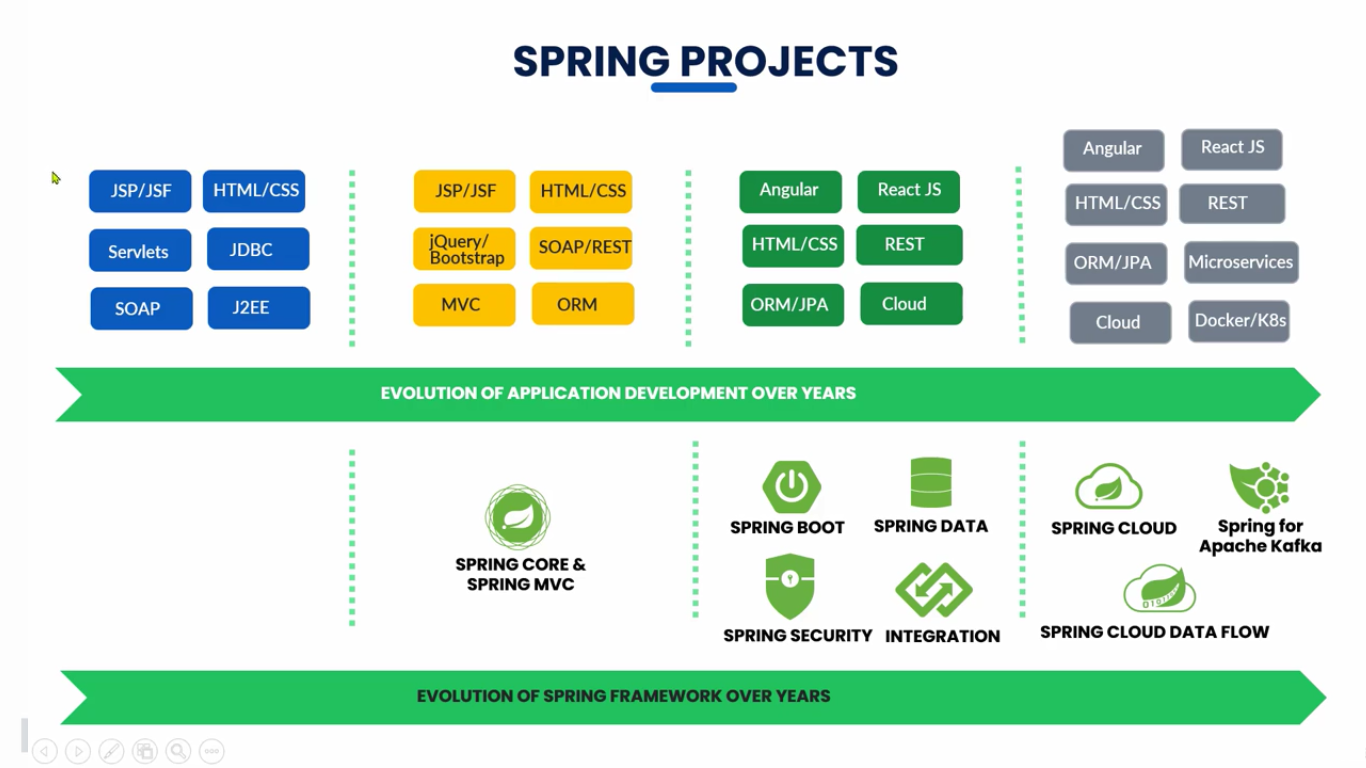
**5.Why The Frameworks?**

****

**6. Spring Projects**

****

**6. Evolution for Spring framework for web applications**

****

**The Problem:**

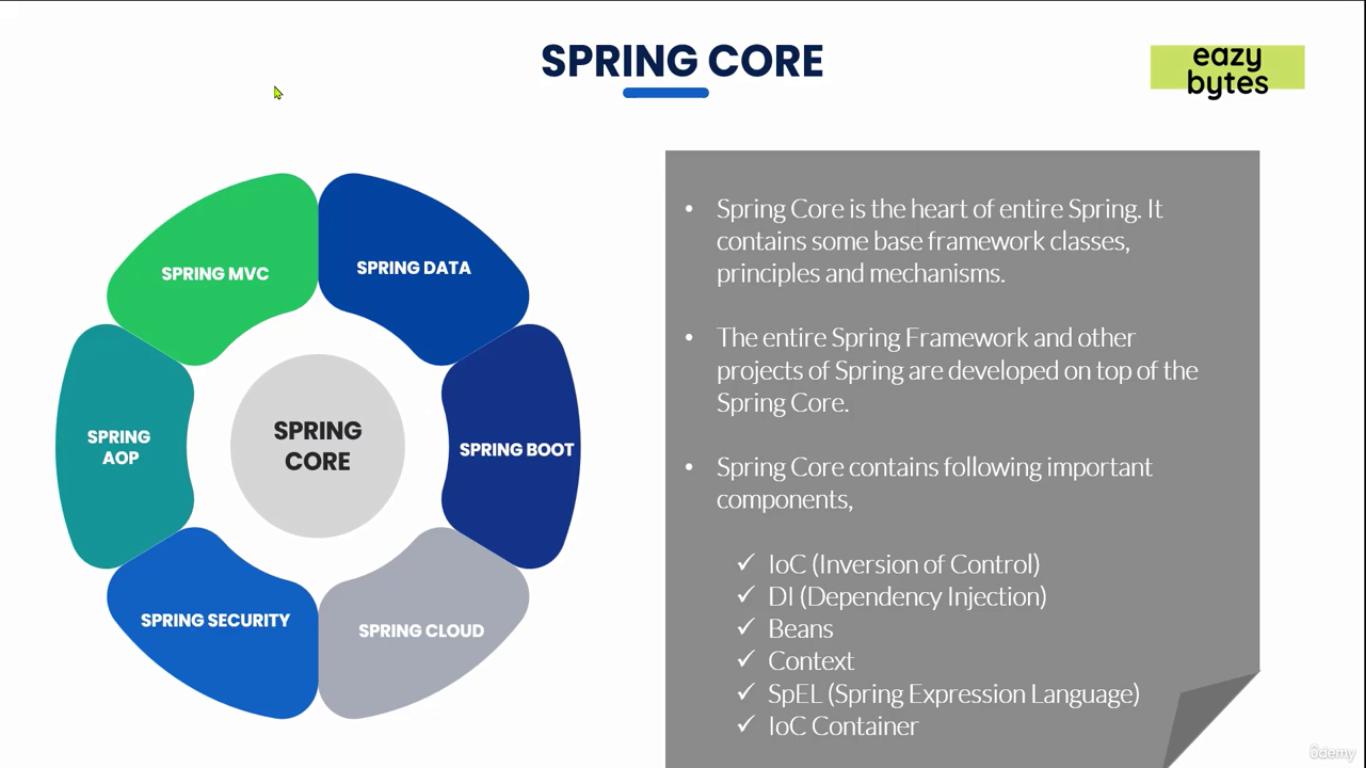
**Building a traditional spring application is really HARD!**

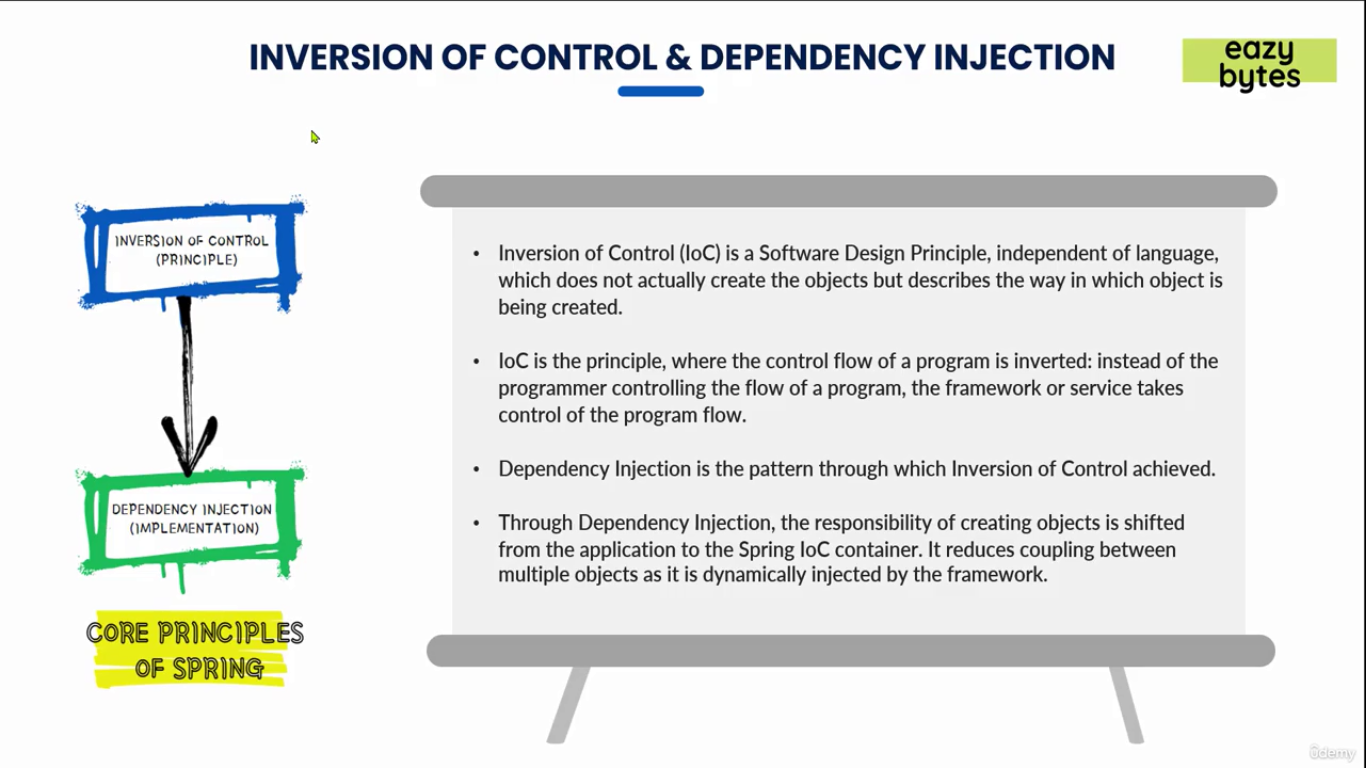
* **Which JAR dependencies do I need?**
* **How do I set up configuration(xml or Java)?**
* **How do I install server?(Tomcat, JBoss ect..)?**

****

**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)
* Helps to resolve dependency conflicts(Maven or Gradle)
* Provide an embedded HTTP server so you can get started quickly(Tomcat, Jetty, Undertow)

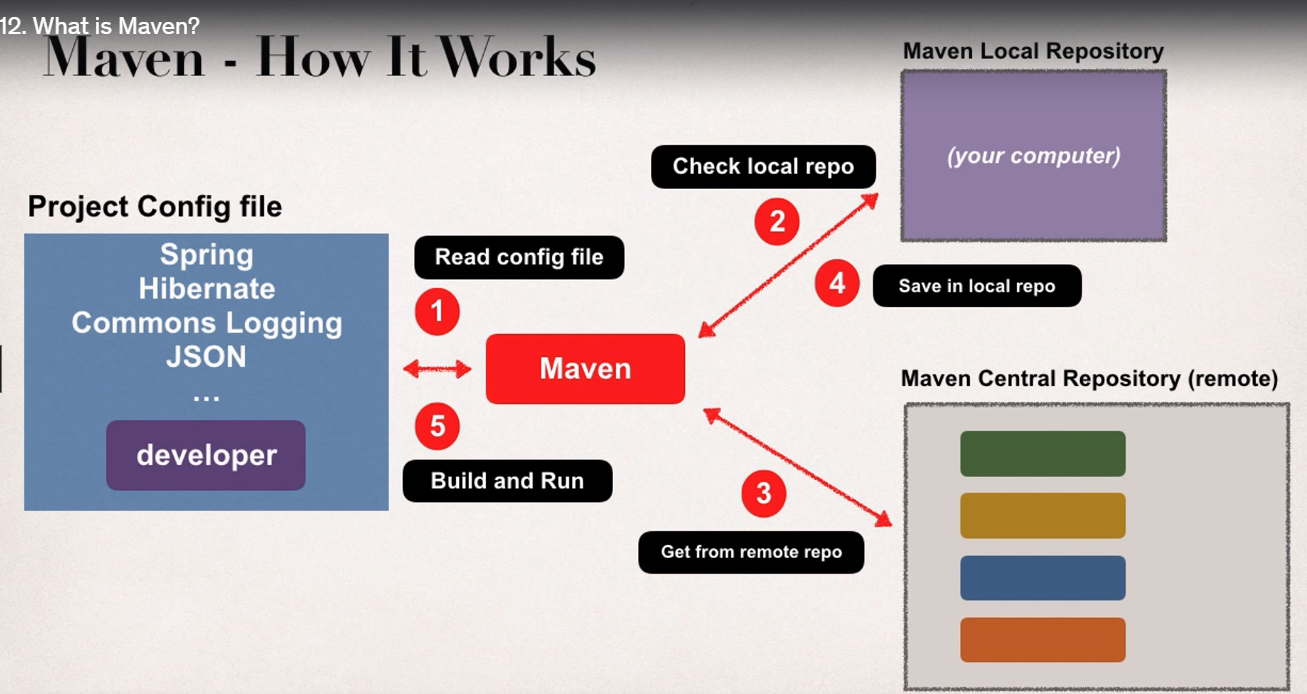




Note: Maven is a open source build tool.

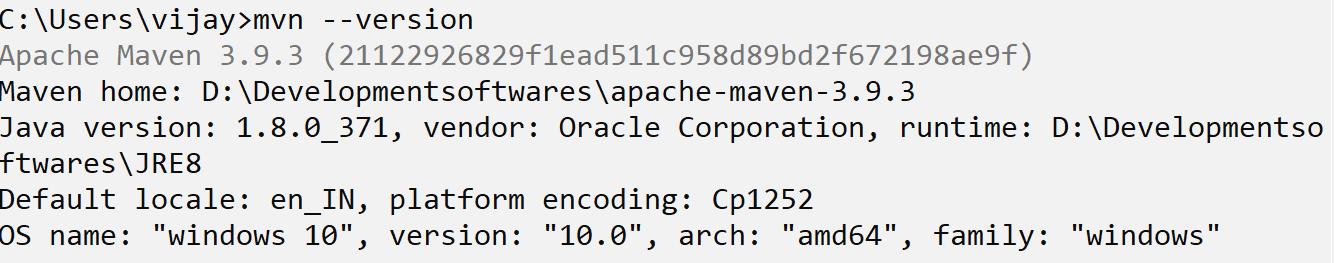
What is Maven?

* Maven is a open source build tool or Project Management Tool
* When you generate projects using spring initializr 🡺 start.spring.io it can generate a Maven project for us.
* Most popular use of Maven is for build management and dependencies.



Installing Maven:

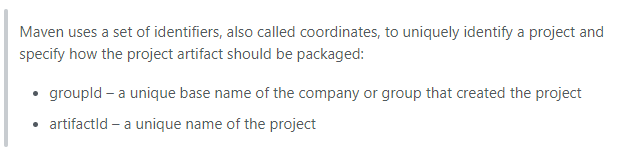
* Go to maven.apache.org
* Click on download
* Under Files select BinaryZipArchive - apache-maven-3.9.3-bin.zip
* Download and Extract the content and keep the folder inside DevelopmentSoftwares Folder
* Set the path of bin folder by Creating a new variable called MAVEN\_HOME 🡺 MyComputer 🡺 Properties 🡺 Select Environment Variables 🡺 Under user variables 🡺 create MAVEN\_HOME 🡺 as a value paste the bin folder location path.
* Set the path of bin folder location to already available path variable 🡺 MyComputer 🡺 Properties 🡺 Select Environment Variables 🡺 Under user variables 🡺 select path variable that is already available 🡺 as a value paste the bin folder location path.
* By default IDE’s may have already maven installed but it’s a good practice to install maven locally as well, so that you can execute the maven based commands locally by using command prompt as well.
* To find out the version of maven trigger

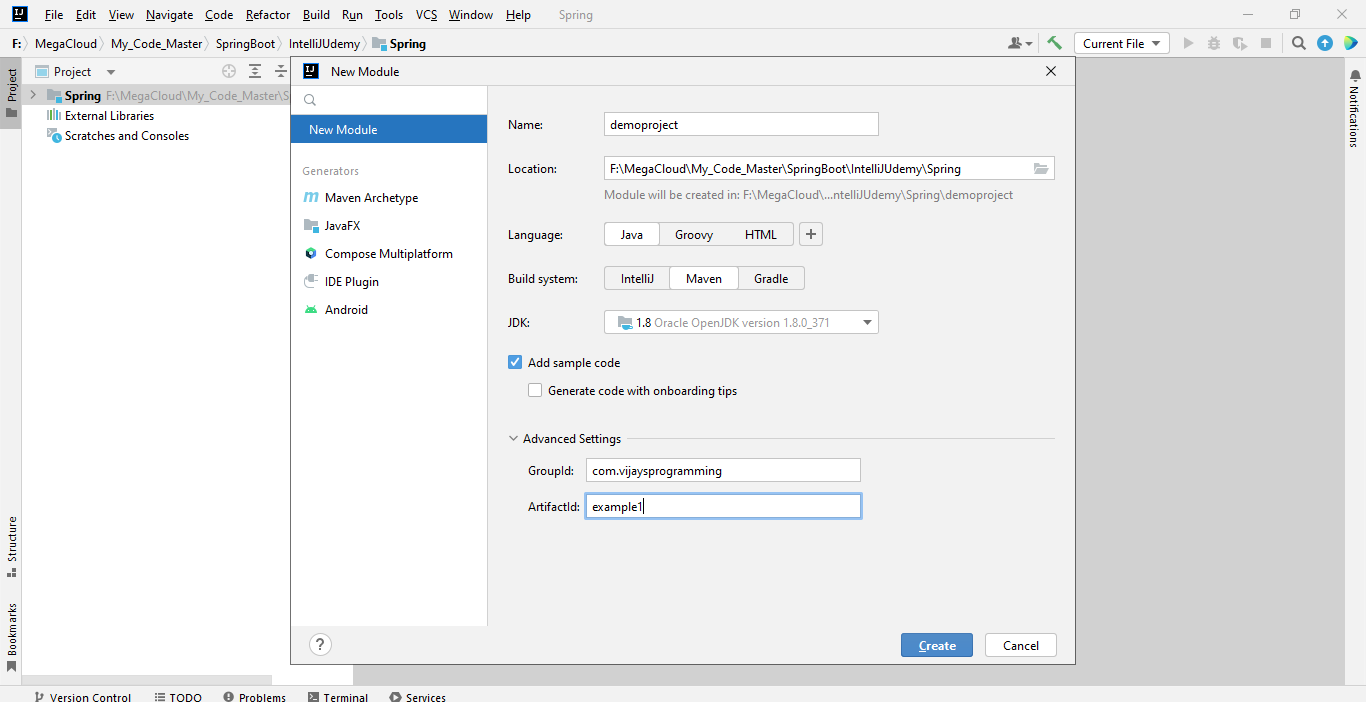


* Go to the intelliJ website and download community edition .exe or zip file

(.exe standalone installer and zip file for executable file)

* After installing intelliJ click on file 🡺 new 🡺 project 🡺 empty project 🡺 Give the name 🡺 Location 🡺 click on create
* Right click on the project 🡺 new 🡺 module 🡺 give the name as demoproject 🡺 language as Java and Build System as Maven 🡺 in Advanced settings create groupId ex: com.vijaysprogramming and create artifactID(by default the artifactID will be your module name) ex: example1
* artifactId is the name of the jar without version. If you created it then you can choose whatever name you want with lowercase letters and no strange symbols. If it's a third party jar you have to take the name of the jar as it's distributed. eg. maven, commons-math
* groupId will identify your project uniquely across all projects, so we need to enforce a naming schema. It has to follow the package name rules, what means that has to be at least as a domain name you control, and you can create as many subgroups as you want. Look at More information about package names. eg. org.apache.maven, org.apache.commons





**Quick Word on Maven**

**The steps to get the dependencies without using Maven**

* When building your Java Project, you may need additional JAR files, ex: 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 our build path / classpath

**The steps to get the dependencies using Maven**

* We can specify the projects that we are working with (dependencies) Spring, Hibernate etc…
* Maven will go out and download the JAR files for those projects for us and Maven will make those JAR files available during compile / run time.
* We can think Maven as our friendly helper.

**Spring Boot Initializr Demo**

* For Quickly creating the started Spring project go to <https://start.spring.io/>
* Select your dependencies.
* Create Maven/Gradle Project
* Import into IDE like Eclipse, IntelliJ, NetBeans etc..

**Development Process**

1. Configure out project at Spring Initializr website
2. Download the zip file
3. Unzip the file
4. Import the project into out IDE

**FIRST APPLICATION**

* Go to <https://start.spring.io/> under project select Maven
* Under Spring Boot select the latest version released. Avoid the “snapshot” versions since they are alpha/beta(testing stage)
* Under Project Metadata

Group - com.vijaysprogramming.demo

Artifact(project/application name) – myfirstapp

Name – myfirstapp

Description – Demo project for Spring Boot

Package name – com.vijaysprogramming.demo.myfirstapp

* Under packaging select Jar
* Under Java version select the Java version that is installed in your system for example 17
* Beside for the Dependencies click on add dependencies

(shortcut windows + B) select Spring Web and click on generate (shortcut windows + enter).

* Go to downloads and unzip the file.
* Copy the folder inside that zip file.
* Create a new folder in the Spring Boot folder name it as 1.dev-spring-boot and paste the folder that you have copied.
* Open the IntelliJ and select open and navigate to the folder where myfirstapp folder is there and select open.
* Click on check box and select trust the project.
* It will take some time to import the project.

Note: if you are getting Cannot resolve symbol 'String' then click on file 🡺 project structure 🡺 select the installed JDK path

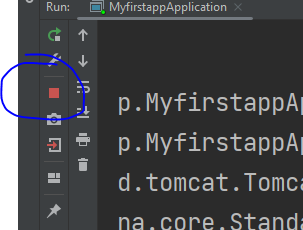
* Run it as a Java application.(cntr + shift + f10)
* Go to <http://localhost:8080/> and you should be getting this message



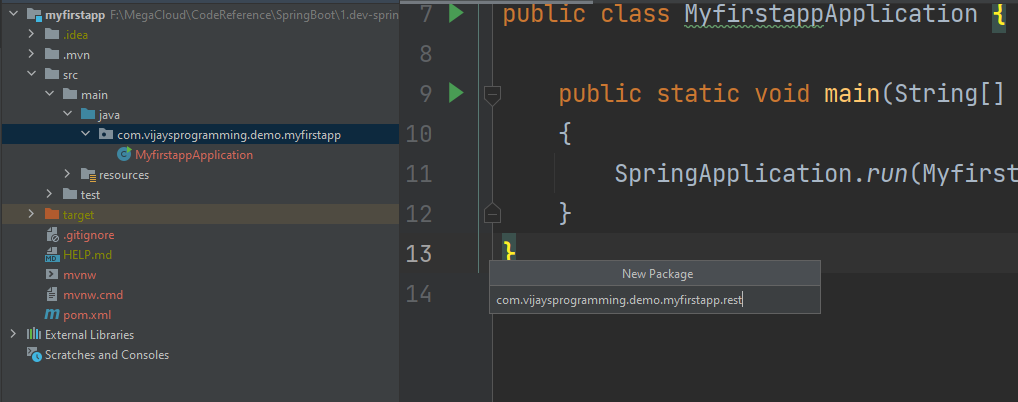
* Because we haven’t added any real code to our project yet no controllers, no view pages.

**Spring Boot – Create a REST Controller**

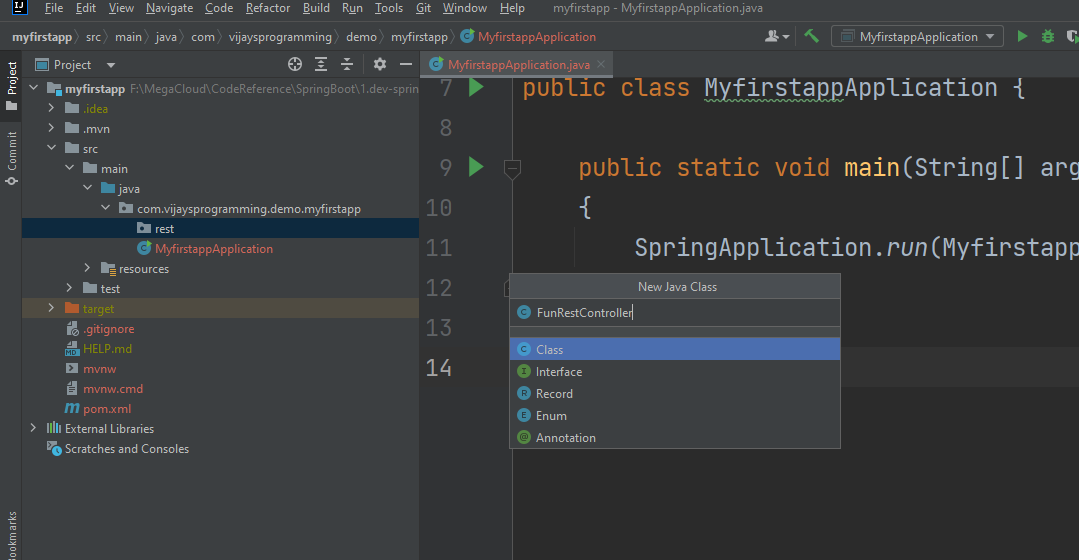
* Lets create a very simple REST Controller. Displaying Hello World on the web page.
* If the IDE is currently running then stop it by clicking on red button



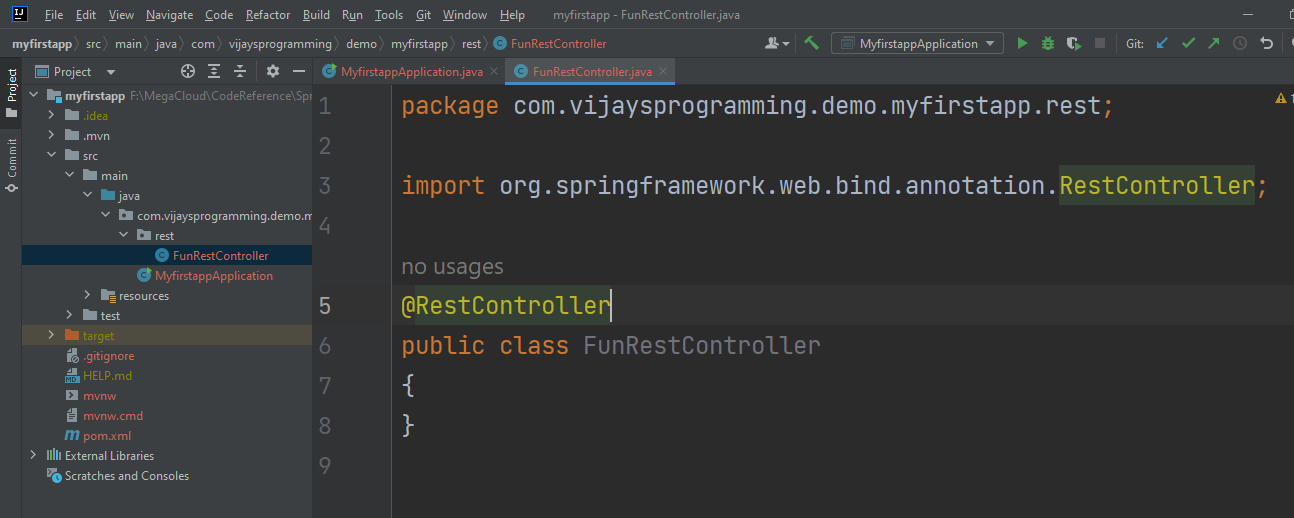
* Create a new package by right clicking on com.vijaysprogramming.demo.myfirstapp and select new package and type rest



* Create a new class called FunRestController inside the newly created package



* On top of public class FunRestController keep the annotation called @RestController



* Add this much of code

package com.vijaysprogramming.demo.myfirstapp.rest;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class FunRestController  
{  
 @GetMapping("/")  
 public String sayHello()  
 {  
 return "Hello World";  
 }  
}

* Go to myfirstapp and run as Java application.
* Go to the localhost 8080 and refresh you should see Hello World as the output

**SECOND APPLICATION**

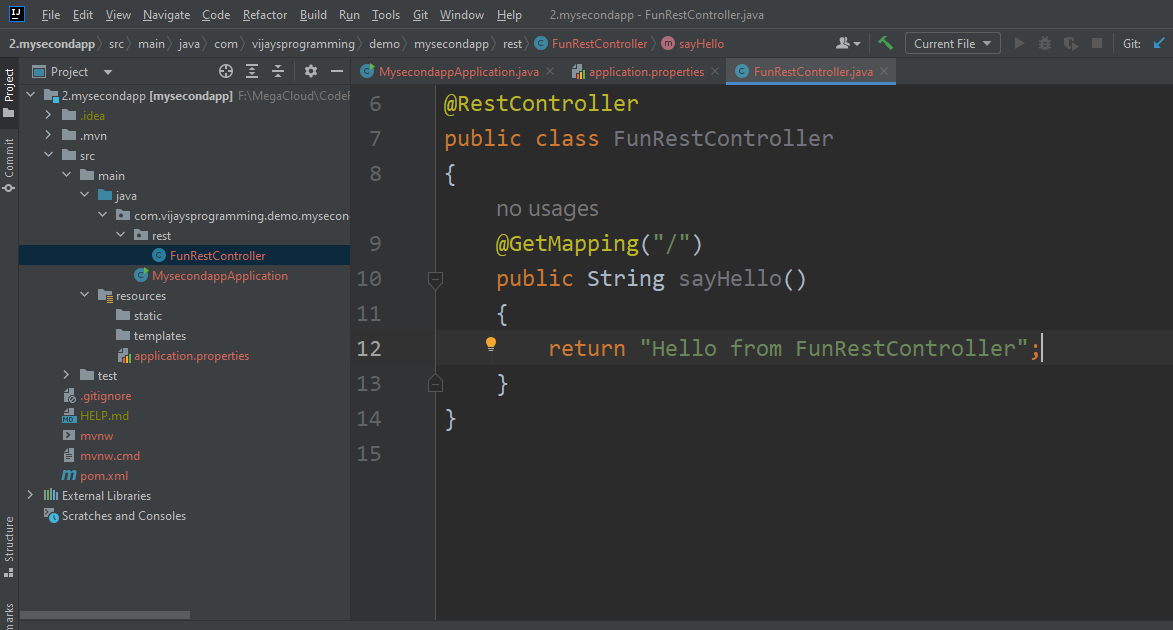
* Go to start.spring.io again and give the artifact name as mysecondapp and select dependency as spring web and click on generate.
* Extract the zip file and keep the mysecondapp folder inside the SpringBoot folder.
* Import that project inside the IDE
* Add one more print statement

package com.vijaysprogramming.demo.mysecondapp;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class MysecondappApplication  
{  
 public static void main(String[] args)  
 {  
 SpringApplication.*run*(MysecondappApplication.class, args);  
 System.*out*.println("My Second Spring Boot Application");  
 }  
}

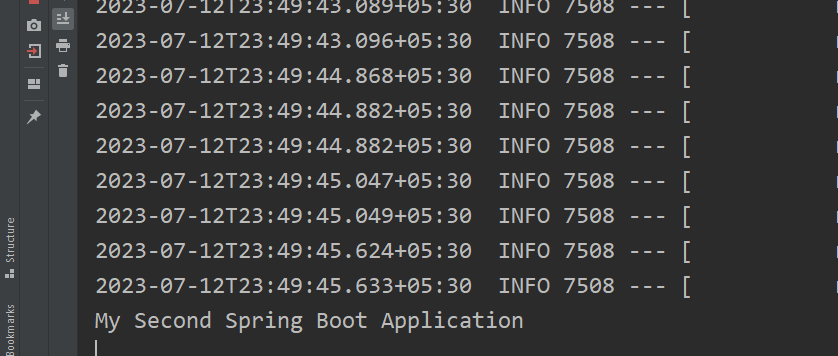
* Its recommended to change the port to other than 8080 because it might be occupied by some other application. To change the port number just expand resources folder and open application.properties and add this

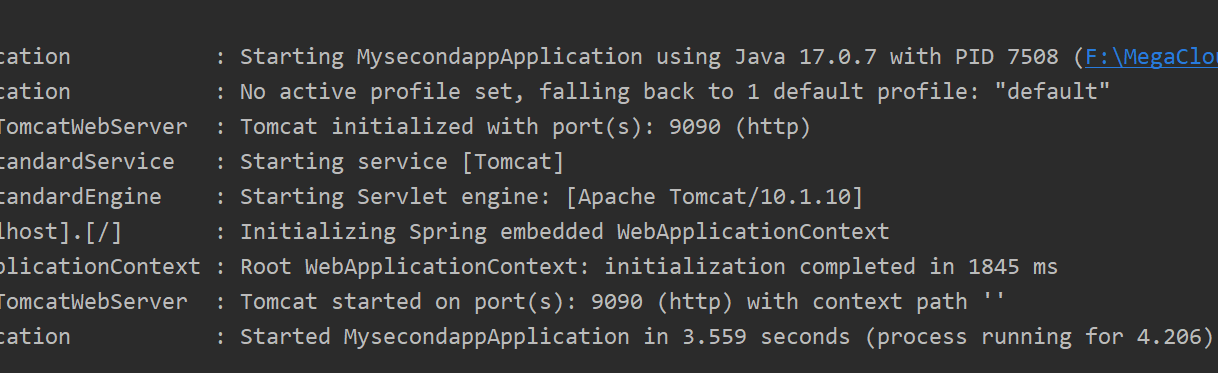
server.port=9090

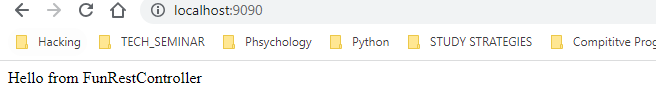
* Create another rest controller class like in the previous app and run the application and see the output in the console and also in the web browser by going to the address localhost:9090



* You should see this much of o/p







* Once the Spring Boot project is ready you can develop any number of controllers.

**THIRD APPLICATION**

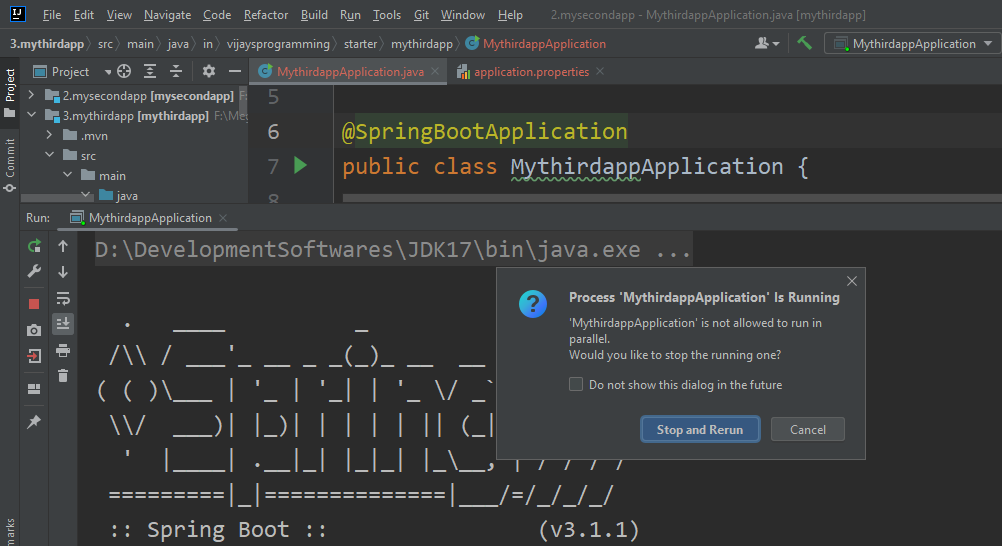
* You can choose same group name for the multiple projects. There is no issue.

Note: to import a new project to the same working directory just open file 🡺 project structure(cntrl + alt + shift + S) and select the project directory and import by selecting the second option(import project from external model) and select maven. Click on ok.

* Change the port number to 9090 and run the application.

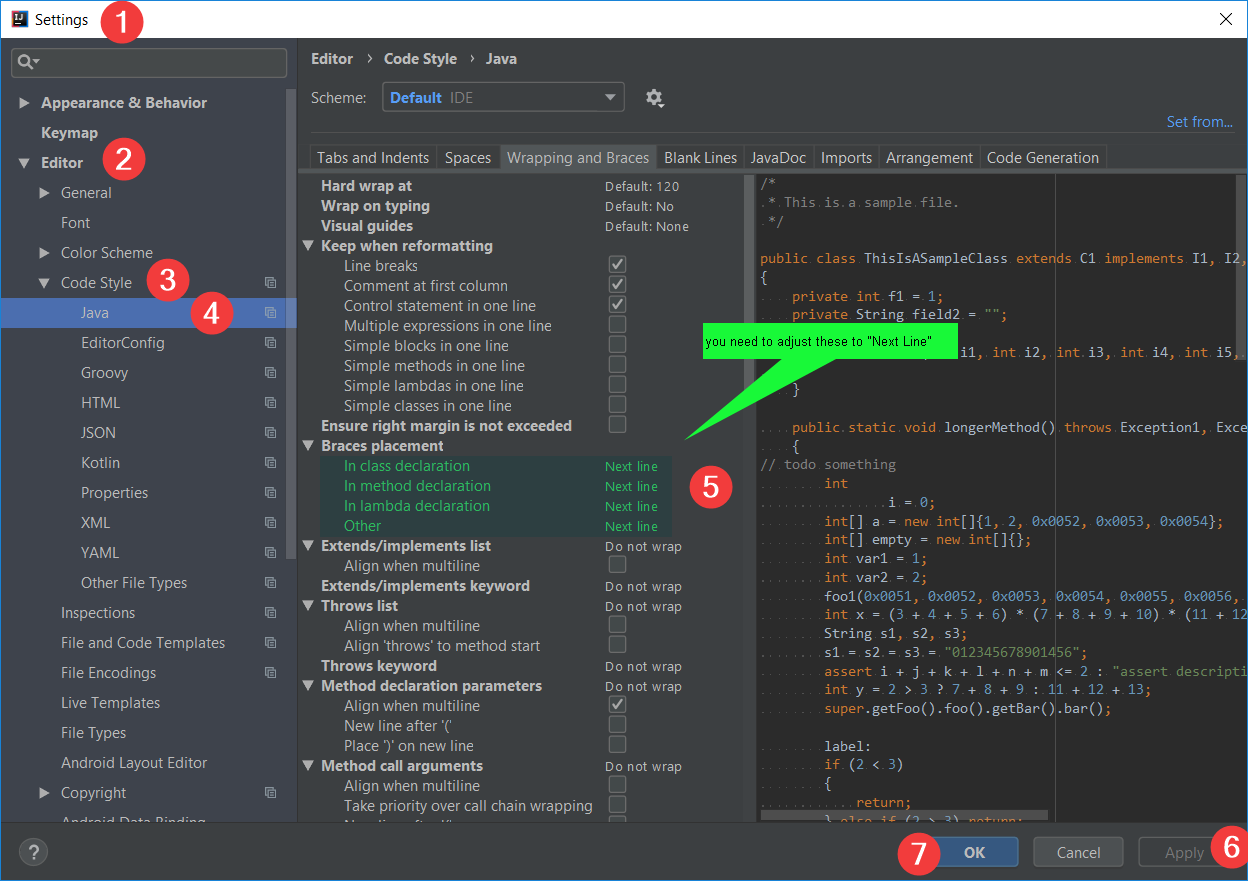
**Note:** if you are starting the application again without stopping the existing application then you will get an error.

**Note:** To avoid the error in IntelliJ you will be getting this notification if you try to run again while another application is already running. But in Eclipse you won’t get this.



* In eclipse you will be getting an error in the console called Web server failed to start. Port 9090 was already in use

**In IntelliJ to place the curly brace position to the next line**



Note: In IntelliJ IDEA, type sout and press the Enter or Tab button from your keyboard to generate System. out. println() automatically.

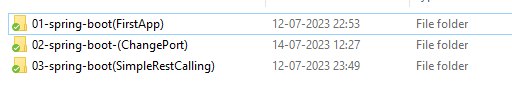
Note: if you are not setting the port number by default it will be considering port number 8080



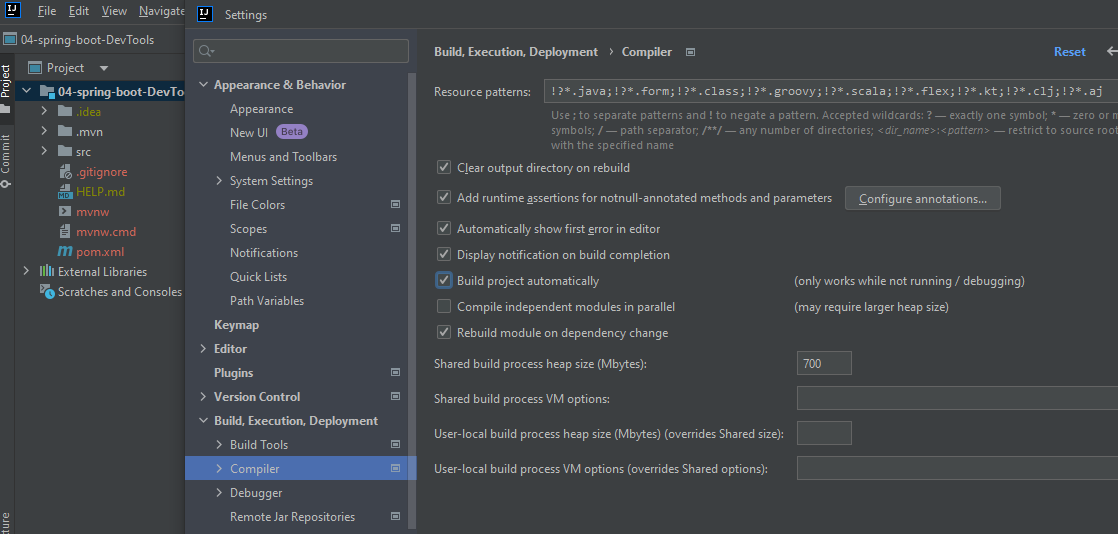
**Spring Boot Dev Tools**

* If you make some changes to your source code then you have to manually restart your application. To avoid this problem
* We have to use spring-boot-devtools
* After we configure spring-boot-devtools which will automatically restarts your application when code is updated
* In order to do this we need to add the dependency to our POM file.
* There is no need for writing additional code
* For IntelliJ we need some additional configurations. Because in IntelliJ community edition by default DevTools is supported.

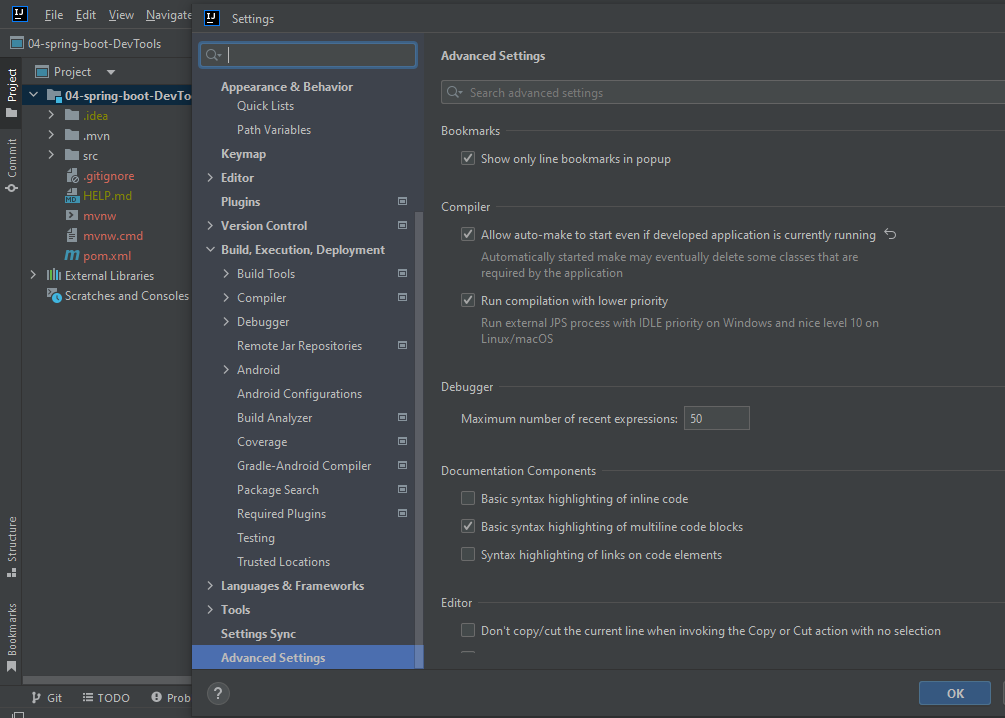
Note: make some changes in the folder names.



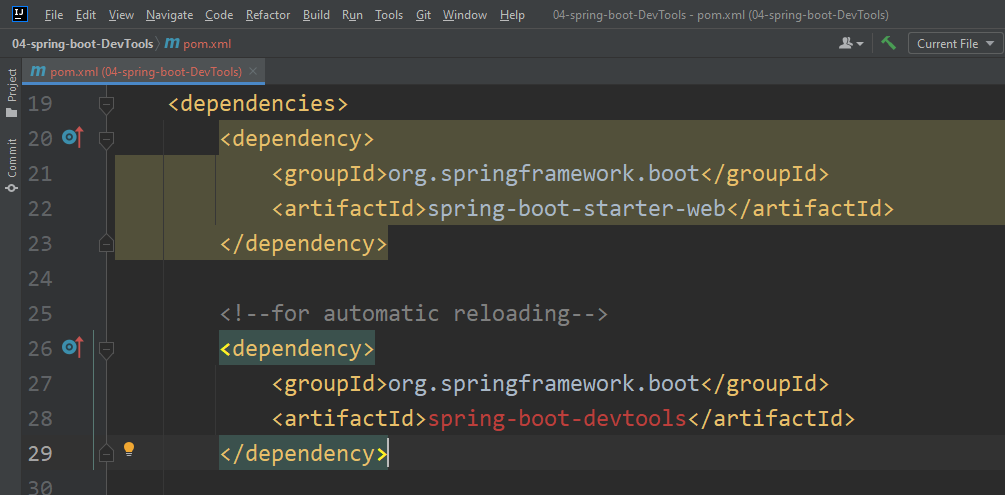
* Create another app 04-spring-boot-DevTools using start.spring.io
* Go to file 🡺 settings(cntrl + alt + S) 🡺 expand Build, Execution, Deployment 🡺 select compiler 🡺 select the checkbox build project automatically



* Go to Advanced Setting and Check Allow auto-make to start



* **Go to pom.xml file add spring DevTool dependency**
* **Before the closing of </dependencies> add this**



* **Be sure to reload the Maven changes by clicking on this icon**



* **Run the Application**
* **Create the rest controller like in the previous examples.**

**Create the package called rest, create the class named ApplicationRestController and create the method inside that called Authentication like this.**

package in.vijaysprogramming.springboot.springboot.DevTools.rest;

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

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

@RestController

public class ApplicationRestController

{

@GetMapping("/")

public String Authentication()

{

return "Authentication Succeded!";

}

}

* Expose a new endpoint for “workout”

**For this scenario you will be getting error**

package in.vijaysprogramming.springboot.springbootSimpleRESTController.rest;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class MyRestController  
{  
 @GetMapping("/")  
 public String Authentication()  
 {  
 return "Authentication is Granted!!";  
 }  
  
 @GetMapping("/")  
 public String APIAccess()  
 {  
 return "APIAccess Granted!!";  
 }  
}

Cause:

Caused by: java.lang.IllegalStateException: Ambiguous mapping. Cannot map 'myRestController' method

in.vijaysprogramming.springboot.springbootSimpleRESTController.rest.MyRestController#APIAccess()

to {GET [/]}: There is already 'myRestController' bean method