Spring Boot

It is a module in the spring framework which needs minimum configurations i.e, no XML file required at all.

It performs auto-configurations based on the libraries you have in your classpath

Spring boot released some of the starter libraries to perform these auto-configurations, some of them are:

1. spring-boot-starter-web
2. spring-boot-starter-aop
3. spring-boot-starter-jpa
4. spring-boot-starter-actuator
5. spring-boot-starter-devtools

These starters takes care of auto-configurations for your applications.

i.e., starter-web will configure a server, dispatcher servlet, view resolver, annotation based configurations, component scanning

starter-jpa will configure all the dependencies required for the databases and scans all the entities, however you need to provide datasource informations in the application.properties

Resolving Version mismatch

Spring boot has a *spring-boot-starter-parent* POM that has all the list of compatible starters according to the spring boot version

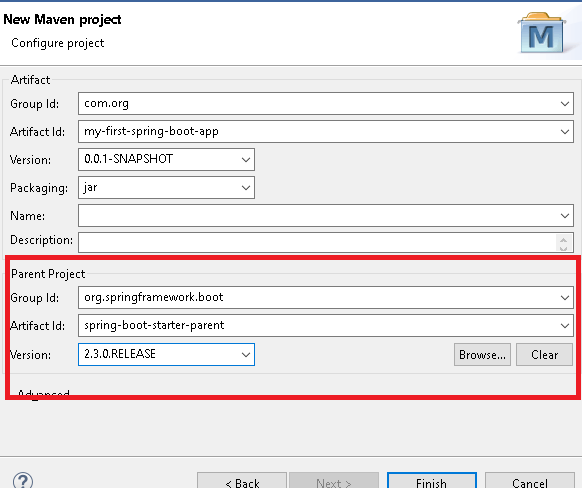
i.e., if you add spring-boot-starter-parent version 2.3 then any starters of spring boot like web, or jpa doesn’t need to mention the version number, because spring boot pulls the compatible version of web or jpa as per the starter-parent.

Note: Spring boot can be done only through either Maven or Gradle

There are two ways you can create project to use spring-boot

1. Maven project with spring boot parent starter
2. Maven project downloading from spring initializr (better)

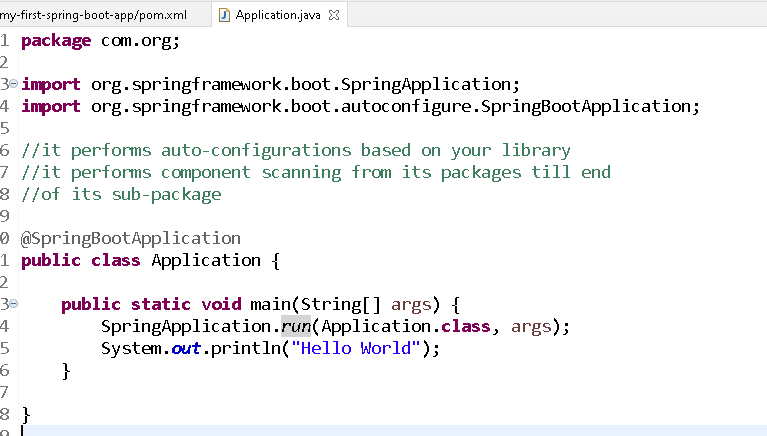
Maven project by manually selecting the spring boot starter parent and mentioning some dependencies



pom.xml



Application.java



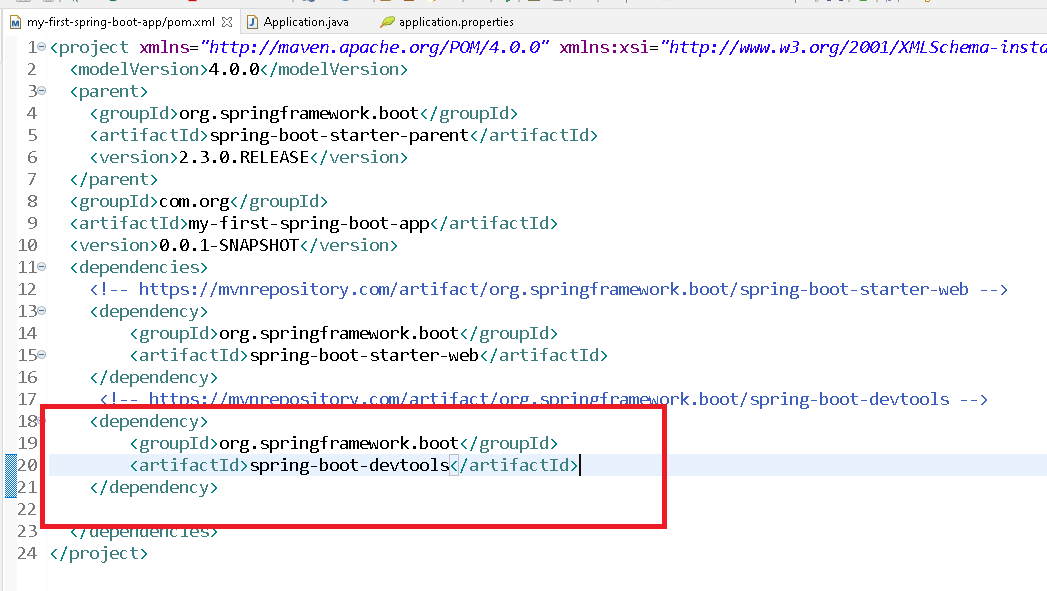
@SpringBootApplication will do auto-configurations like

* stating a tomcat server
* component scanning from its root package
* configuring dispatcher servlet
* reading application.properties & configuring the application

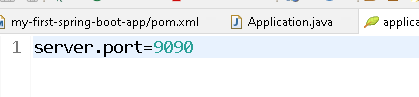
application.properties file should be created inside src/main/resources folder, however you can have yml file instead of properties i.e, application.yml

You can add spring boot devtools to automatically reload your application on server

pom.xml



create application.properties inside the src/main/resources



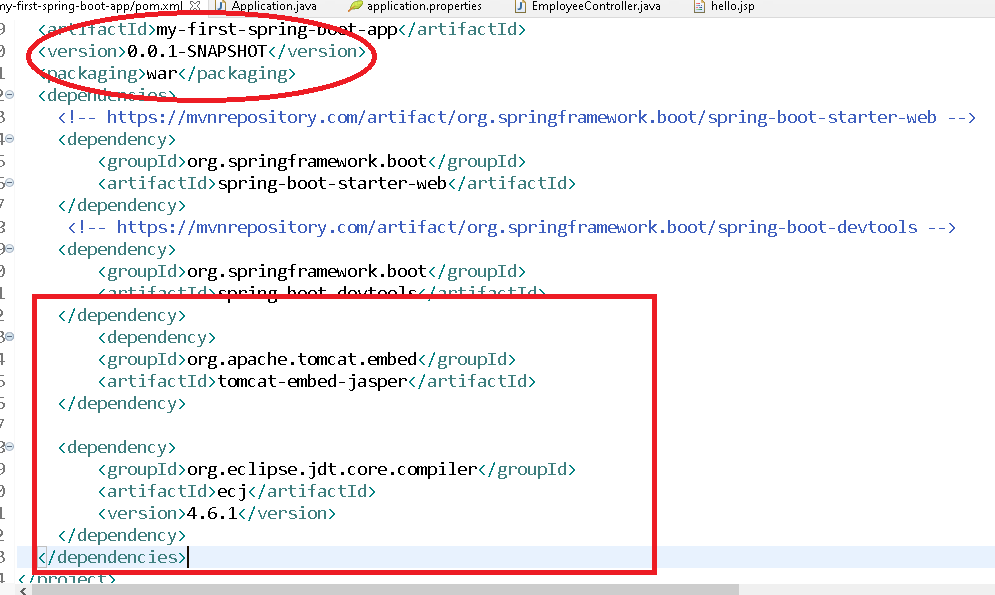
You can create a controller using @Controller because of auto-configuration feature DispatcherServlet will be configured and you can use controller directly.

Adding JSP files to your spring boot project needs lot of configurations.

In pom.xml you add two dependencies and also change the <package> to war

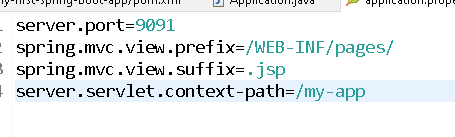
1. tomcat embedded jasper
2. ecj

pom.xml



Add view resolver properties in pom.xml

application.properties



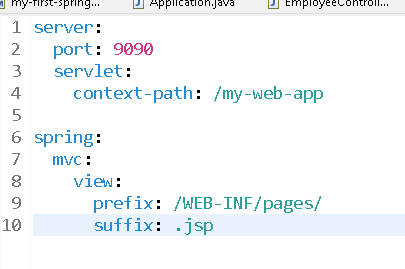
Create a controller



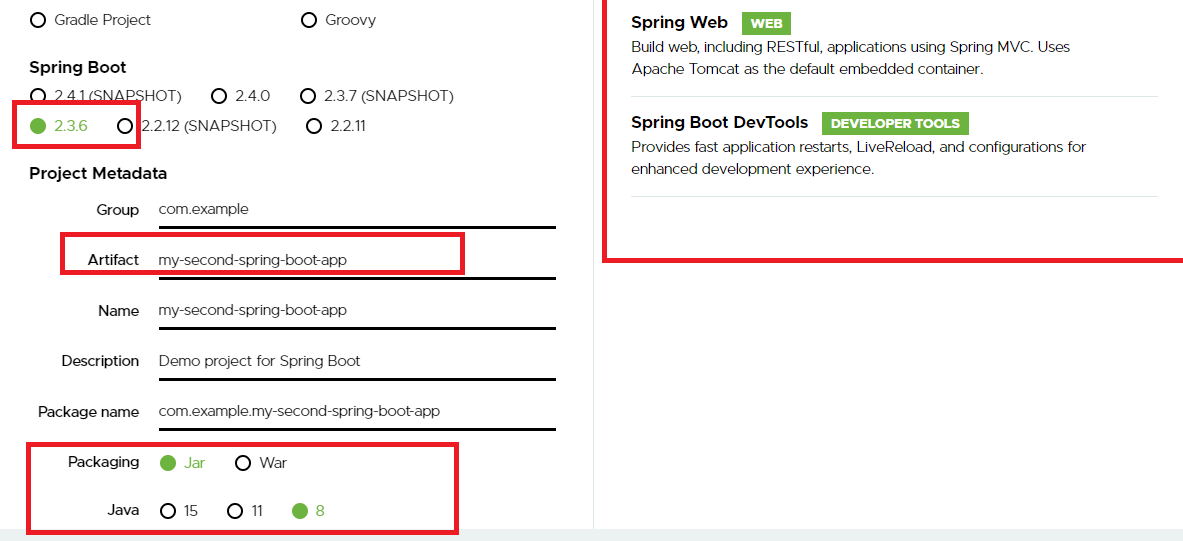
You can use yml files instead of properties because it avoids you using repeatable properties.

Note: yml files use indentation to recognize the nested properties

Delete the application.properties & create application.yml



Creating the project using spring initializr



When you use spring initializr you will get a Java file that launches the spring boot application and also an application.properties where you can configure the application configurations like port, server address, datasource, context-paths, actuator informations, you can create yml file by deleting the properties file.

How to run/launch the application in the production

Inorder to launch the application in the production you need a deployable war/jar, since you have created project through spring initlializr it gives you deployable jar/war.

You need to use mvn commands to build a jar and run the jar

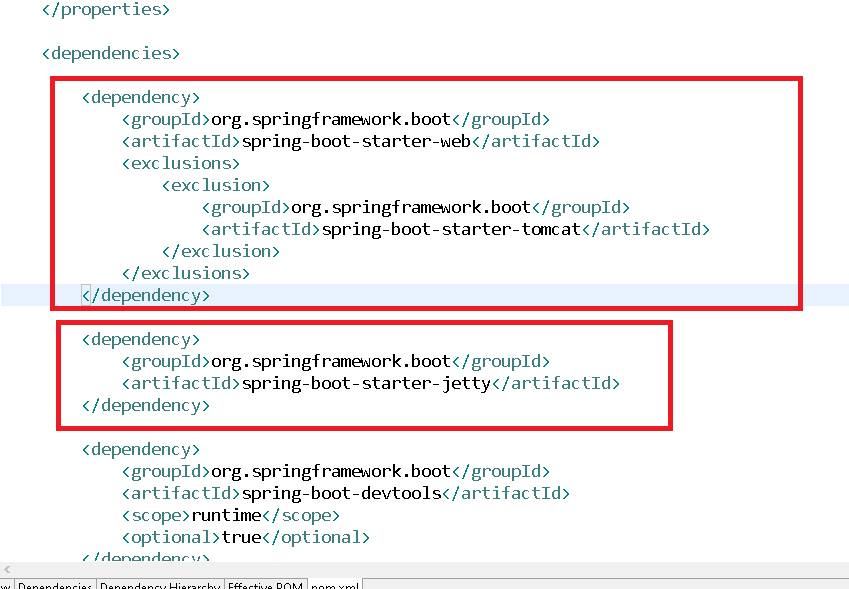
mvn clean: deletes the target directory

mvn package: compiles and creates the executable file

java -jar target/file.jar --server.port=9093

Note: All these commands you need to enter from the project locations

You can add different servers other than tomcat like jetty server, but you must exclude the default tomcat server in the web dependency



Output:

