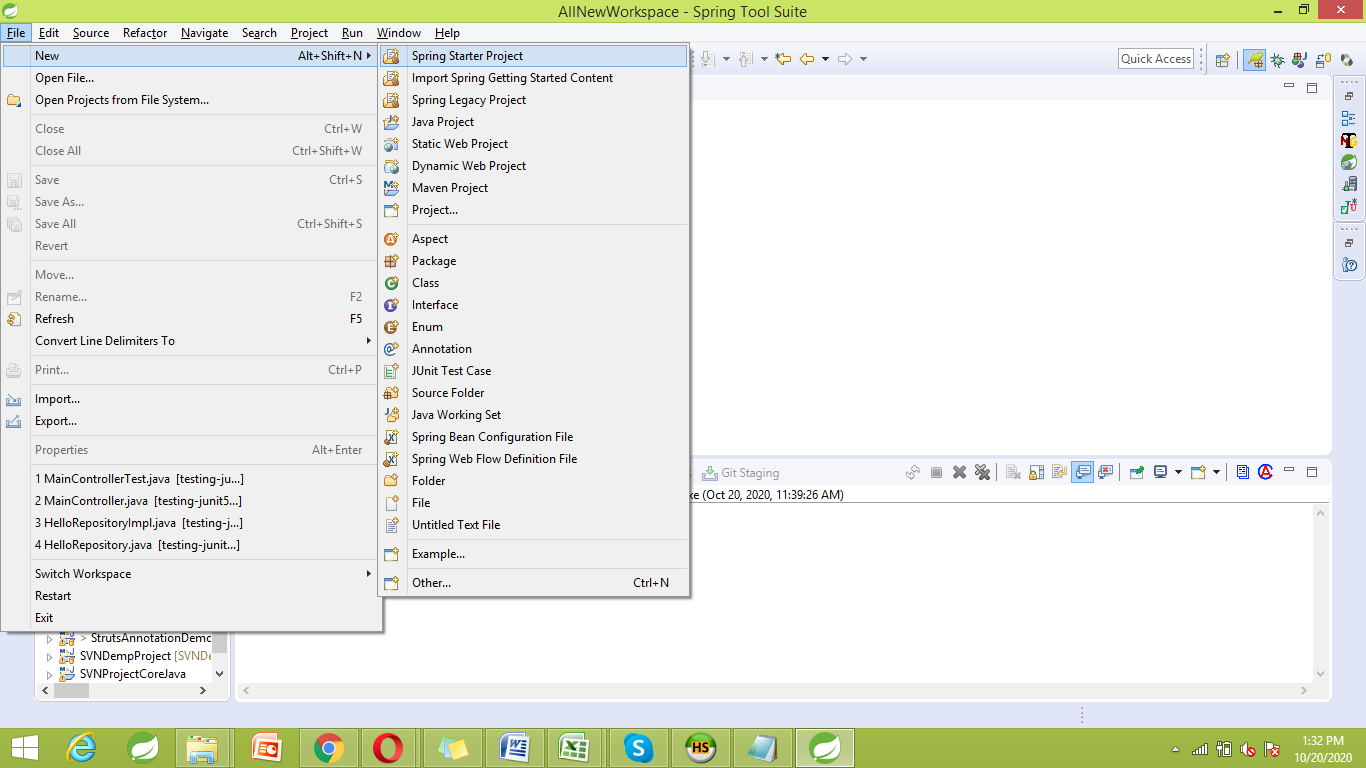
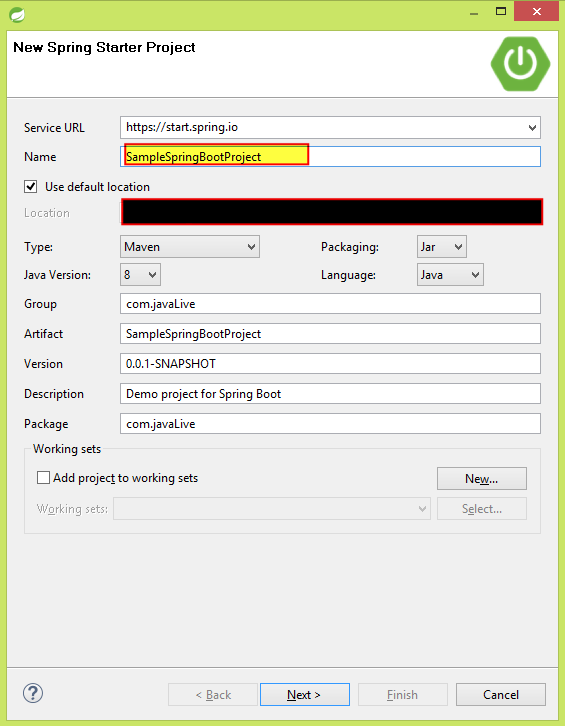
File->New->Spring Starter Project

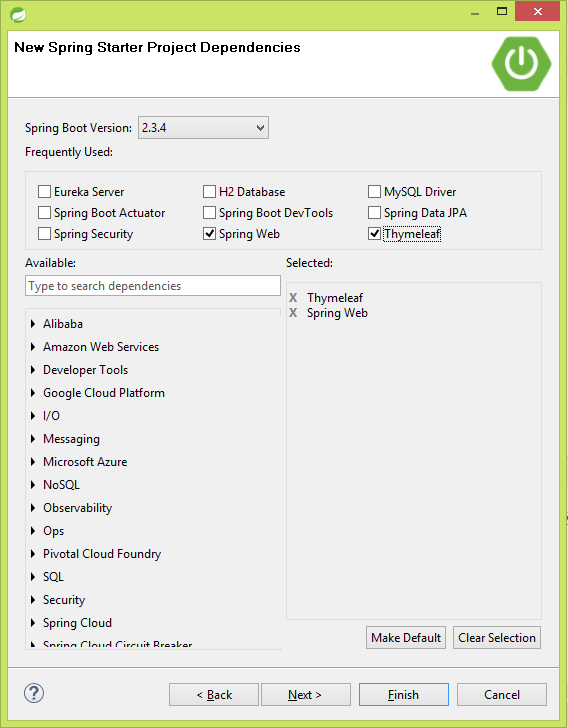


Provide a name to the project as follows:



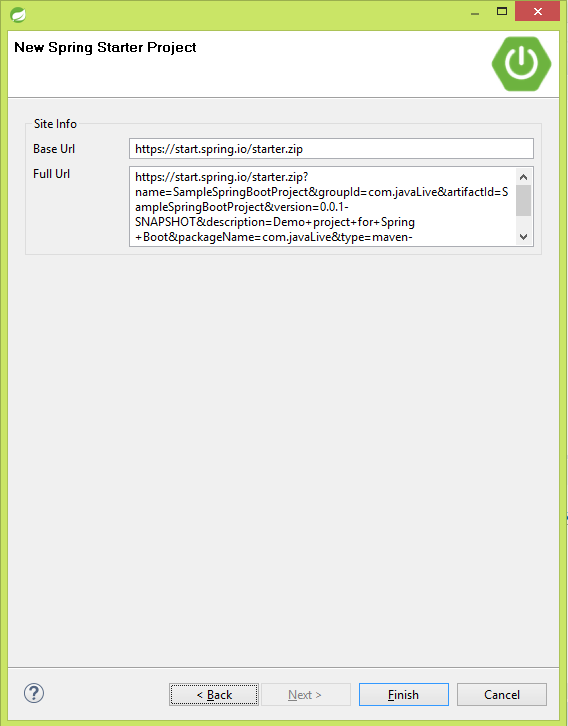
Now select the required dependencies as follows:

And then click 'Next'



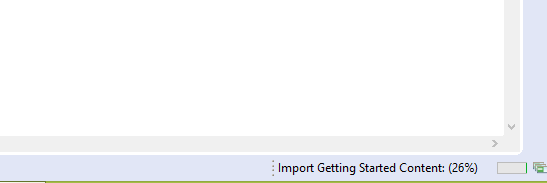
\*\*\*\*\*\*\*Note that above process is fulfills following feature of Spring Boot

Dependency Management - The different versions of commonly used libraries are pre-selected and grouped in different starter POMs that we can include in your project. By selecting one Spring Boot version we are implicitly selecting dozens of dependencies that we would have to otherwise select and harmonize our self.

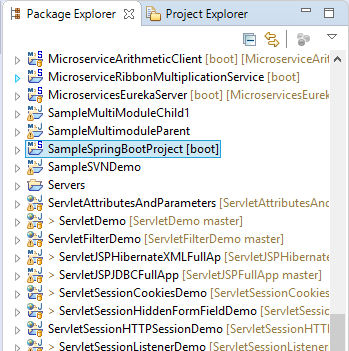


Cliclk 'Finish' button.

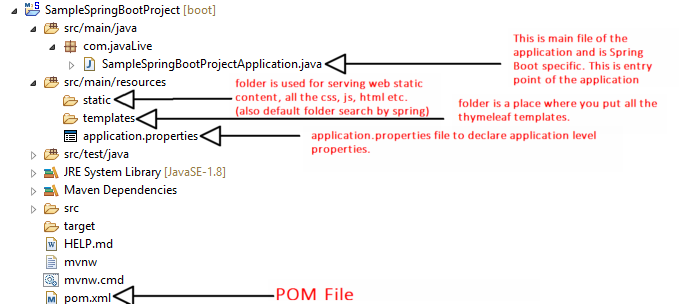
Now process to create the project get initiated which's progress we can see in bottom right corner.



After its completion, we can observe the created project in Package Explorer' as follows:

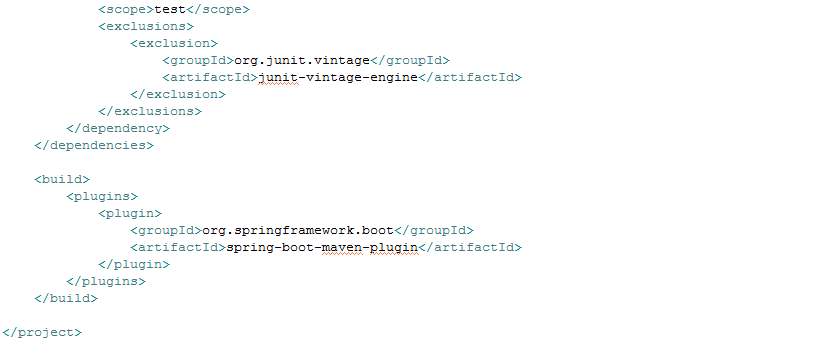
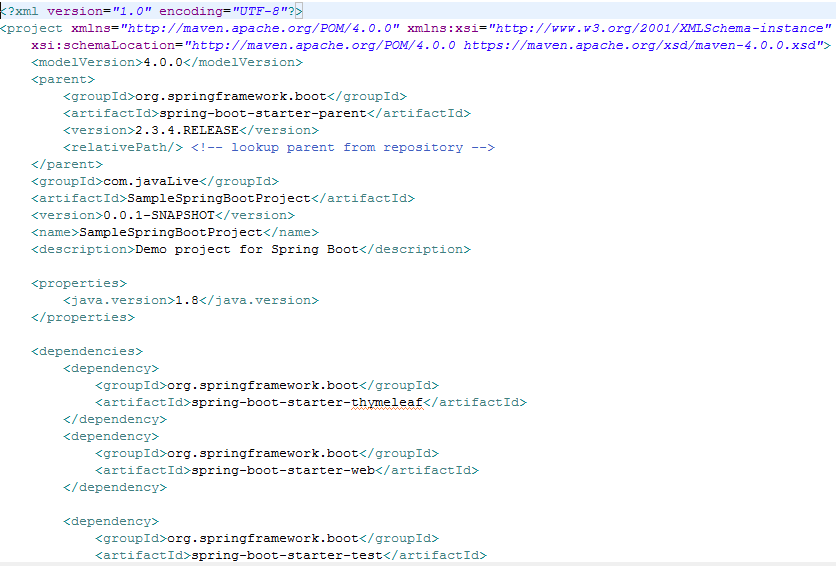


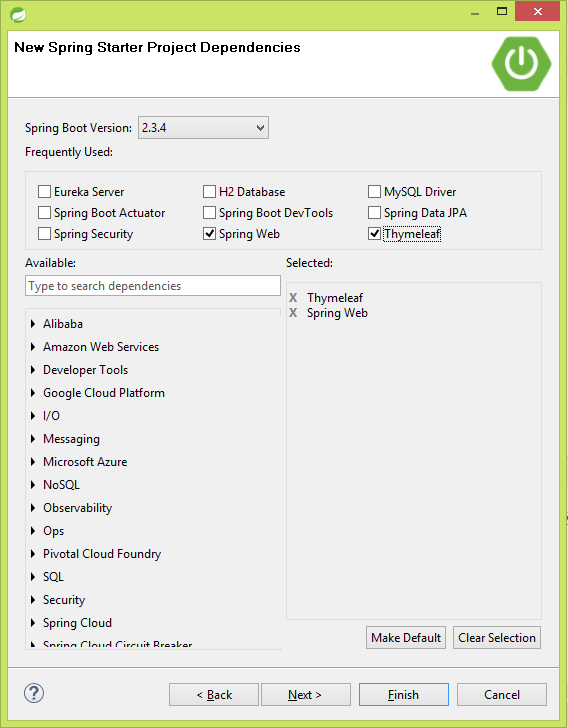
When we expand above project, it looks like below:



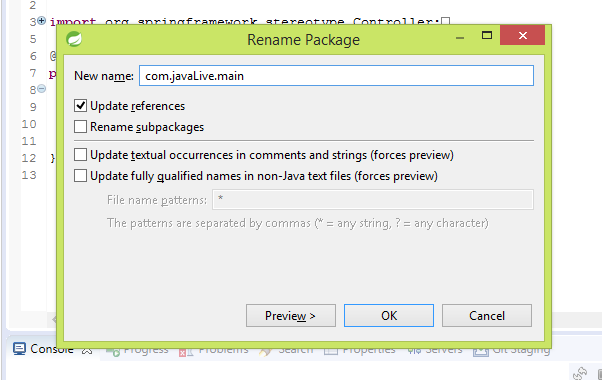
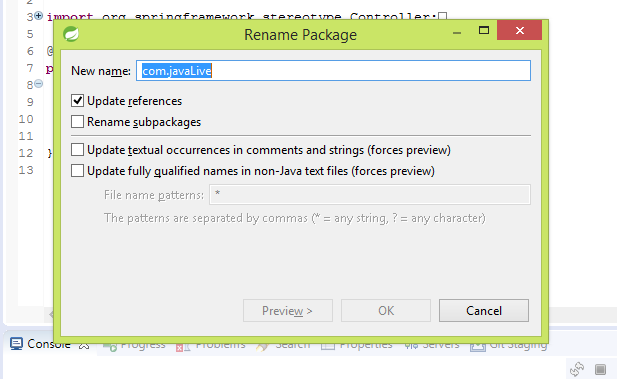
Please go through above file structure along with its usage very carefully.

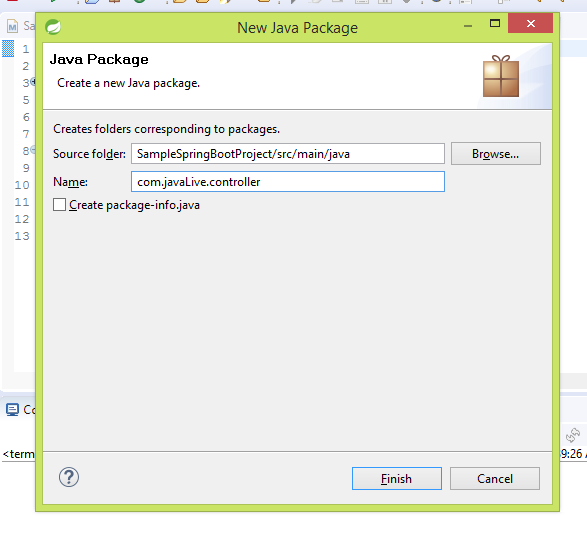
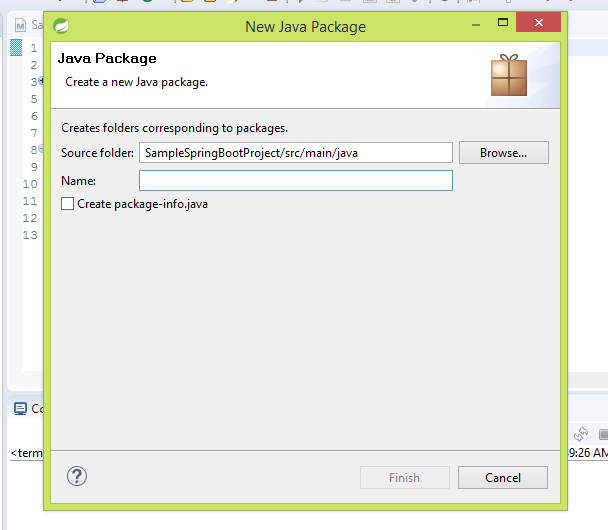
Now below is the screen shot of 'POM.xml' file.

Compare it with following selected dependencies during creation of the project.

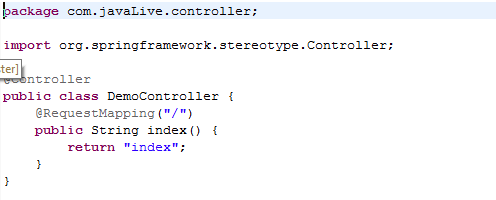


Now we rename package com.javaLive to com.javaLive.main as follows:

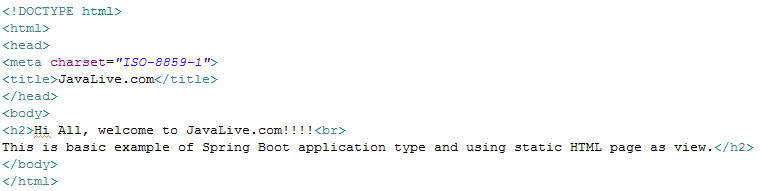
Now create one more package for controllers as follows:



Now add a controller file viz. DemoController.java in package com.javaLive.controller as follows:



Add index.html file in src/main/resources's static folder as follows:

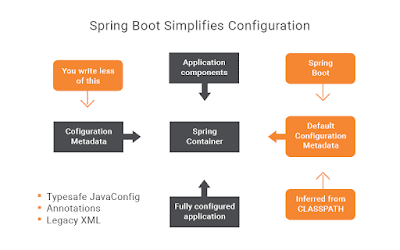


SampleBootProjectApplication.java file will look like below:

Please note the annotation @SpringBootApplication.

The Main class serves two purposes in a Spring Boot application: **configuration** and **bootstrapping**. First, it's the main Spring configuration class and second, it enables the auto-configuration feature of a Spring Boot application.

If you are interested in learning more about essential Spring Boot features, e.g. auto-configuration and the Starter dependency, then [Spring Boot Essentials](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-boot-essentials%2F) is a good place to learn them quickly.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-boot-essentials%2F)

**@SpringBootApplication = @Configuration + @ComponentScan + @EnableAutoConfiguration**

The @SpringBootApplication annotation is a combination of following three Spring annotations and provides the functionality of all three with just one line of code.

**@Configuration**

This annotation marks a class as a Configuration class for Java-based configuration. This is particularly important if you favor Java-based configuration over XML configuration.

**@ComponentScan**

This annotation enables component-scanning so that the web controller classes and other components you create will be automatically discovered and registered as beans in Spring's Application Context. All the @Controller classes you write are discovered by this annotation.

**@EnableAutoConfiguration**

This annotation enables the magical auto-configuration feature of Spring Boot, which can automatically configure a lot of stuff for you.

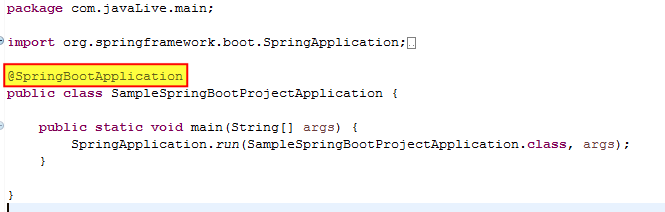
For example, if you are writing a Spring MVC application and you have Thymeleaf JAR files on the application classpath, then Spring Boot auto-configuration can automatically configure the Thymeleaf template resolver, view resolver, and other settings automatically.

So, you can say that @SpringBootApplication is a 3-in-1 annotation that combines the functionality of @Configuration, @ComponentScan, and @EnableAutoConfiguration.

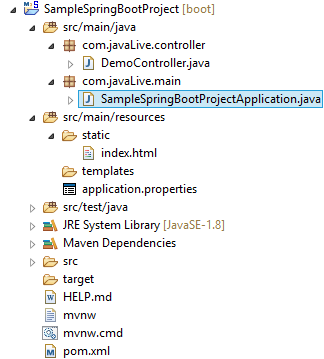
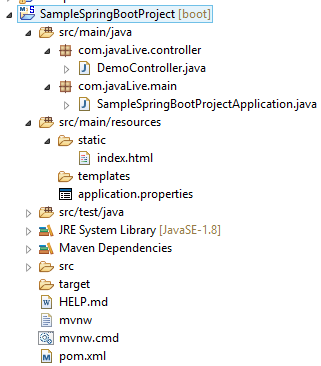
It also marks the class as a BootStrap class, which means you can runt it as a normal Java class, e.g. by running its JAR file from the command prompt as shown here, or just right-clicking and running it as a Java program in Eclipse IDE.

This will start the embedded server that comes along with Spring Boot and runs your web application inside it. Once you see the log without any errors, you can go to the browser and open the localhost with the server port to access your Spring Boot application.

That's all about the @SpringBootApplication annotation and a simple application to demonstrate how to use it. As I said, this nice little annotation packs quite a lot of punch. You can just write this one line of code to enable Java-based configuration, component scanning, and to enable the auto-configuration feature of Spring Boot. It makes your code more readable.

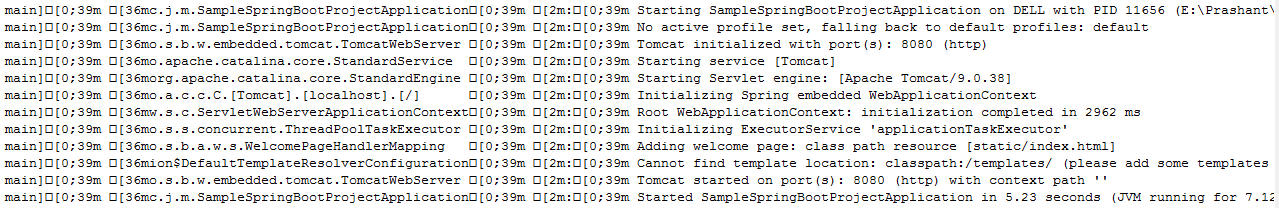


Now project structure will look like as below:

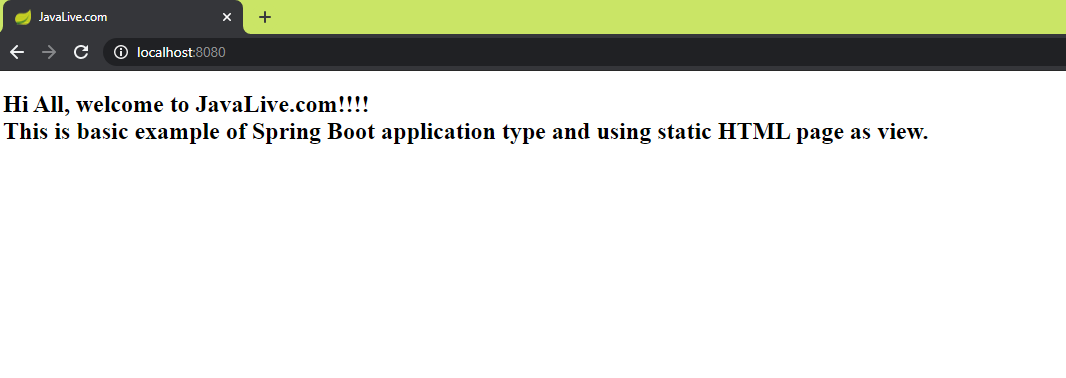


To run the application right click on SampleSpringBootApplication.java file(as highlighted above) and Run As->Spring Boot App

It shows below console output which confirms that application is up and running.



Go to Chrome and type localhost:8080 it shows below output(as we have mentioned in the controller).



Please pay attention to following feature of Spring Boot:

**Auto-Configuration** - No need to manually configure dispatcher servlet, static resource mappings, property source loader, message converters etc.

Thus here we have studied how to create and run a simple Spring Boot Application.

**Please go through the procedure during this project creation for better understanding of each upcoming Spring Boot project and practice it very well so that you can create and run Spring Boot Project on your own without any help.**

One of the most important feature of Spring Boot is

**Advanced Externalized Configuration** - There is a large list of bean properties that can be configured through application.properties file without touching java or xml config.

This feature is accomplished with the help of application.properties file, so we have given a separate file viz. 'Introduction To Spring Boot application\_properties File' in this project folder. Please go through it very carefully as it is very important for day to ays operations regarding Spring Boot application.