Spring Boot Tutorial

Spring Boot is a Spring module which provides RAD (Rapid Application Development) feature to Spring framework.

## **Prerequisite**

Before learning Spring Boot, you must have the basic knowledge of Spring Framework.

# Spring Boot

1. It is a Spring module which provides RAD (Rapid Application Development) feature to Spring framework.
2. It is used to create stand alone spring based application that you can just run because it needs very little spring configuration.
3. Spring Boot does not generate code and there is absolutely no requirement for XML configuration.

## **Prerequisite of Spring Boot**

* Java 1.8
* Gradle 2.3+ or Maven 3.0+
* Spring Framework 5.0.0.BUILD-SNAPSHOT
* An IDE (Spring Tool Suit) is recommended. No problem if we use Eclipse

# Spring Boot Features

## **Web Development**

It is well suited Spring module for web application development. We can easily create a self-contained HTTP server using embedded Tomcat, Jetty or Undertow. We can use the spring-boot- starter-web module to start and running application quickly.

## **SpringApplication**

It is a class which provides the convenient way to bootstrap a spring application which can be started from main method. You can call start your application just by calling a static run() method.

1. **public** **static** **void** main(String[] args){
2. SpringApplication.run(className.**class**, args);
3. }

# Steps to create Spring Boot Project

# -1- Go to Eclipse and create Maven Project and select QuickStart

# -2- Now go to <https://projects.spring.io/spring-boot/> and copy the dependencies

<parent>

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

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

<version>1.5.7.RELEASE</version>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

# </dependencies>

# If we want to change the version of java from pom.xml we can do like ths

<properties>

<java.version>1.8</java.version>

</properties>

**Now What is Spring Boot?**

There are two words Spring and Boot, Spring we are already familiar with and Boot means something like BookStrap the project (RAD Rapid Development)

**Official Defination** :: Spring Boot makes it easy to create stand-alone, production-grade Spring based Application that you can “JUST RUN”

Standalone means its does not depend on any type server to be downloaded, Production grade means the Spring Boot App is not just like Hello World App we use previously its a production or Enterprice level app

**To UnderStand Spring Boot we need to underStand what is Spring?**

If we answer this Question its something like IOC or DI thats the perfect answer as of 2006 now at this point thats no longer just a case Spring is much much more then IOC or DI

**1)Its a whole Application FrameWork** (Means we can build fully working Enterprise java Application)

**2)Programming and Configuration model** :: Means we are just focusing to build our Business Component and lets Spring to handle those comman problems like logging and transactions

Like we are just simply creating a class and just Anotate with that class with @Controller thats all

Now thats Spring will look out how to make that class as a Controller

**3)InfraStructure Support** :: Spring provide support like RDBS,DBMS,MongoDB using that we can directly comuincate with those database

**Problems with Spring ?**

**1)Huge Framework ::** Bcuz its not just a one thing we have a lots of thing in it

Why because there is a lots of way to build EnterPrise Java App Spring providing support almost all

**2)MultiPle SetUp Step ::** It can connect to MongoDB, Can connect to RDBMS and many more and all of them having Different Diffferent Steps

Since its does a whole lot for that it need to do a lots of Configuration for it to do exactly what you want it to do

For that above reason its a have **3)MultiPle Configuration Steps**

**4)Multiple build and deploy steps ::** Since its a have a lots of combination for that we have a diff diff deployment process

**So now Can we abstract these Steps?** Means i will just do 20% of my app remaing all will take care by Spring....

So for that we have Something Called **Spring BooT**

**Enter Spring BooT...**

1)Opinionated :: Means its a good starting point its make a certain configuration automatically means its say just start with this remaing i will handle

2)Convention over Configuration :: Suppose you want to do 100 things for to achive something but what it 80 will be the default and you just need to do 20 of it.... so its have a predifined configuration which you can use and reduce boiler plate code

3)Stand Alone :: Typically when we build a Spring App we are building a war file which we can deploy on server for that we need to find Servlet, Tomcat or some other container but in SB we will get Stand Alone app like some thing we can just run we no need to deploy on server no need to start even no need to download the server it has Embedded Tomcat Server in it....

## **Prerequisite of Spring Boot**

* Java 1.8
* Gradle 2.3+ or Maven 3.0+
* Spring Framework 5.0.0.BUILD-SNAPSHOT
* An IDE (Spring Tool Suit) is recommended. No problem if we use Eclipse

***Now Whenever we are doing Spring Project we need to manage Dependencies(Downloading Jars) inside the classPath so right now we are not gonna doing this we are gonna use Tool Like Maven for our Spring Boot Project***

**Creating a First Project**

1.Go to File and create Maven Project this time check to skip arctype

2.Fill all necessary details and click finish

Now as you see Maven has downloaded some predifined template if we go in details Maven has download the java library which is 1.5 we will see about this in later

3.Now Open POM.XML file of your project

Now if we see in POM.XML file there is nothing on it

Now to convert Our project into SpringBoot we need to define some SpringBoot Related dependencies on it

***Now first step we need to define this***

<parent>

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

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

<version>1.5.7.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

In our pom.xml file inside the project tab

The meaning of that is we telling to Maven that our project is child project to this parent

**For Ex ::** Maven will downloading all jar file for our project but how its come to know which file it need to be download for that it will take the help of that parent project

So in Parent project we just have Configuration but not JAR FILE and we are inheriting that config in our project

We have something called Convention over Configuration we are just configuring 20% and 80% done by Spring becuz in background we are inheriting Most of config from parent project

**Step 2**

Now we are creating Application which is gonna run on WEB so for that we required some set of JAR files

So for that we need to define

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

Now save the File and it till download all jar file which we required in web project

We Might get error which say Maven project is Up-to-date so for that Right Click on Project>>Go to Maven >> and click update project

Now if we see JRE its 1.5 and latest is 1.8 or 9 so i want JRE which is 1.8 so for that

<properties>

<java.version>1.8</java.version>

</properties>

And re Update the Project

Now we are all set with pom.xml now lets move to the project

Now go to src/main/java and create simple class with package p1 and class name is CourseApiApp.java and put main method in it

Now why we requied main becuz as we know SpringBoot App does not required container or server we can start our app simply running a main method

**public** **class** CourseApiApp {

**public** **static** **void** main(String[] args) {

***//now we need to put some code here which will start the server and deploy our project into the server***

Now its a simple java class with main method now we tell to Spring treat its as a SpringBoot App so for that

**@SpringBootApplication** we have to put annotation above class name which avail in package **org.springframework.boot.autoconfigure.SpringBootApplication;**

Now this will tell SpringBoot that this is the starting point my App

Now we need to define some content in a main method which tell SpringBoot to start this App and create a servlet container and start the server and host our app on it

SpringApplication.*run*(CourseApiApp.**class**, args);

Its take two args 1st the class name in which we have main meth and 2nd is the command line argument value

That’s all this is our Spring Boot App we have created

Now after running this if we see in cansole we will get

Tomcat started on port(s): 8080

Started CourseApiApp in 7.137 seconds (JVM running for 7.999)

To check go to browser and check localhost:8080 we will get an error page

Now what SpringApplication.run(CourseApiApp.class, args); will internally do

1. Sets up default configuration :: Already Scene above
2. Start Application Context :: its a container like we have lots of services Business, Controller etc Spring act as container for all that means that all services is gonna run inside a Spring
3. Performs class Path scan :: we are annotating our class with lots of annotation like @Cantroller, @Services its will scan for all and treat it is as respective way
4. Starts Tomcat Server

**So** finally we have not downloaded any server not created any container we have did nothing that s why SpringBoot is called a STANDALONE Application

**Why SpringBoot has Decided to have Embedded Tomcat Server**

**1)Convenience ::** means at run time we don’t need to go tomcat website and perform download staff and all

**2)Servlet container config is now application config ::** previously we have to config Tomcat as well as Application also now in that you can config that all in one piece

**3)Standalone App**

**Now we will see another way to create SpringBoot project**

***With Spring Initializer***

1. Go to this link <https://start.spring.io/>
2. Below Generate Project Button there is option called SWITCH TO FULL VERSION
3. Now after filling all required details don’t click on Generate button
4. Go to WEB section and select 1st option for Tomcat
5. Go to SQL and select JPA
6. Then Click on Generate project
7. Extract the file and import in eclipse with Maven Import

**Spring *Actuator Start Here***

Spring-Boot devTools?

If we add Spring-Boot devTools in our project we no need to start server it will be automatically started if we make any changes in our project.

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

</dependency>

Spring-Boot Actuator?

It is very useful when we want to monitor our production running application and we want to debug something in under development phase

Actuator exposes certain rest API endpoints

Pom.xml

<dependency>

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

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

</dependency>

That’s all we need to do for starting with Actuator

Browser

http://localhost:8080/actuator

*/actuator* :: It’s not real metrics API you can see it does not showing any info related to our app

After hitting above URL Spring Boot doesn’t expose all API, if we want to see all we can add this

application.properties

management.endpoints.web.exposure.include=\*

***ByDefault Actuator JSON response in unformatted for formatting***

spring.jackson.serialization.indent\_output=true

*/actuator/beans* :: It will show us all beans configured in our project

*/actuator/health* :: It like a ping service will Info about our App is UP or not

*/actuator/configprops* :: It will show all configuration we have done for our project, to check port in ***configprops*** configureitin ***application.properties*** filethen it will be visible over there

Server.port=8081

*/actuator/env* :: Will display info like **Java.Runtime.Environment**, **Java.class.path**, **Java.home, application.properties ETC**

*/actuator/info* :: It use to put our custom info

*/actuator/httptrace ::* will give info latest hit to the server

*/actuator/mappings ::* If suppose we are putting **@RequestMapping(“**/products**”)** now I want to see /product is going to handle by which class

**Display Custom info for** */actuator/info***?**

application.properties

info.current.user.count=20

info is mandatory, above approach is **static** info for **Dynamic** Info

InfoService.java

@Component

**public** **class** InfoService **implements** InfoContributor{

@Override

**public** **void** contribute(Builder builder) {

HashMap<String, Object> hashMap = **new** HashMap<String, Object>();

hashMap.put("Msg", "Hello World");

hashMap.put("Current Time", LocalDateTime.*now*());

builder.withDetail("key", hashMap);

}

}