**Spring Boot CLI installation and Hello World Example**

Welcome for Spring Boot tutorial. Here we are going to discuss more detail about Spring Boot CLI (Command Line Interface). Previously we have already discussed [**Introduction of Spring Boot**](http://www.dineshonjava.com/2016/06/introduction-to-spring-boot-a-spring-boot-complete-guide.html#.V2vAcvl97IU) and [**Essentials Key Components of Spring Boot**](http://www.dineshonjava.com/2016/06/essentials-key-components-and-internals-of-spring-boot-framework.html). The quickest way to get started with Spring Boot is to install the Spring Boot CLI so that you can start writing code.

**What is Spring Boot CLI?**

The Spring Boot CLI is a command line tool that can be used if we want to quickly develop with Spring. It allows us to run Groovy scripts, which means that we have a familiar Java-like syntax, without so much boilerplate code. We can also bootstrap a new project or write your own command for it.

It is Spring Boot software to run and test Spring Boot applications from command prompt. When we run Spring Boot applications using CLI, then it internally uses Spring Boot Starter and Spring Boot AutoConfigurate components to resolve all dependencies and execute the application.

It internally contains Groovy and Grape (JAR Dependency Manager) to add Spring Boot Defaults and resolve all dependencies automatically.

**Installing the Spring Boot CLI**

There are several ways to install the Spring Boot CLI:

* From a downloaded distribution
* Using the Groovy Environment Manager
* With OS X Homebrew
* As a port using MacPorts

**MANUALLY INSTALLING THE SPRING BOOT CLI**

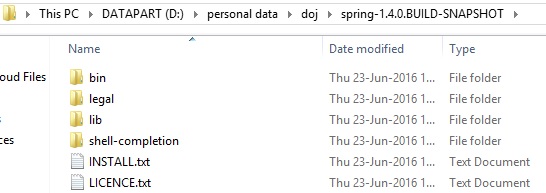
We can install Spring Boot CLI software using either Windows Installer or Zip file. Both approaches are easy to install and will give us same Spring Boot CLI software.

**Step 1:**You can download the Spring CLI distribution from the Spring software repository:

* [**spring-boot-cli-1.4.0.BUILD-SNAPSHOT-bin.zip**](http://repo.spring.io/snapshot/org/springframework/boot/spring-boot-cli/1.4.0.BUILD-SNAPSHOT/spring-boot-cli-1.4.0.BUILD-SNAPSHOT-bin.zip)
* [**spring-boot-cli-1.4.0.BUILD-SNAPSHOT-bin.tar.gz**](http://repo.spring.io/snapshot/org/springframework/boot/spring-boot-cli/1.4.0.BUILD-SNAPSHOT/spring-boot-cli-1.4.0.BUILD-SNAPSHOT-bin.tar.gz)

Once downloaded, follow the [**INSTALL.txt**](https://raw.github.com/spring-projects/spring-boot/master/spring-boot-cli/src/main/content/INSTALL.txt) instructions from the unpacked archive. In summary: there is a spring script (spring.bat for Windows) in a bin/ directory in the .zip file, or alternatively you can use java -jar with the .jar file (the script helps you to be sure that the classpath is set correctly).

**Step 2:** Extract **spring-boot-cli-1.4.0.BUILD-SNAPSHOT-bin.zip** file into our local FileSystem.



**Step 3:**Set Spring Boot CLI Environment Variables in Windows System as shown below.

1. **set** PATH= D:\personal data\doj\spring-1.4.0.BUILD-SNAPSHOT\bin

**Step 4:** Execute the below command to verify our installation process.

**Using the CLI**

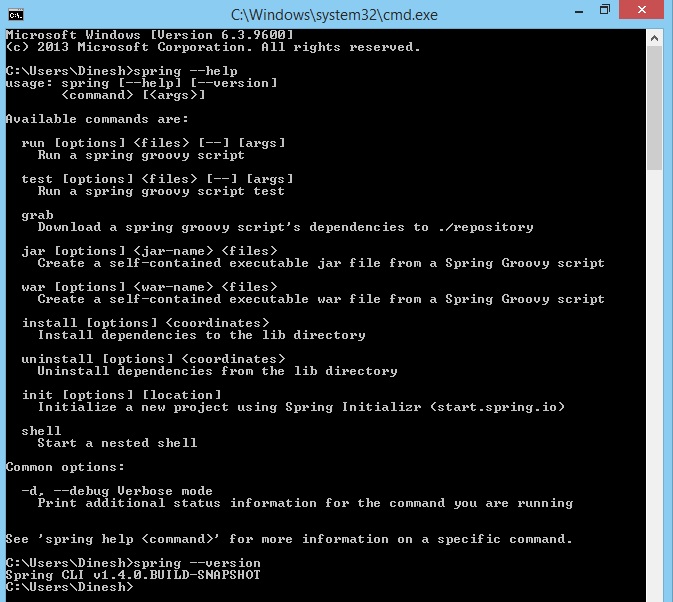
Once you have installed the CLI you can run it by typing spring.

We can use “spring –version” to know the Spring Boot CLI Version as shown below.

1. C:\Users\Dinesh>spring --version
2. Spring CLI v1.4.0.BUILD-SNAPSHOT

We can use “spring –help” to know the Spring Boot CLI Version as shown below.

1. C:\Users\Dinesh>spring --help
2. usage: spring [--help][--version]
3. <command>[<args>]
4. Available commands are:
5. run [options]<files>[--][args]
6. Run a spring groovy script
7. test [options]<files>[--][args]
8. Run a spring groovy script test
9. ……..more command help **is** shown here



Now our Spring Boot CLI Installation process is done successfully.

**Quick start Spring Boot CLI Hello World Example**

Here we are using same example provided in Spring Boot Documentation.

Please follow the following steps to develop a Spring Boot HelloWorld Example:

**Step 1:** Create an “app” Folder in our Local FileSystem to place our java file.

**Step 2:** Create a file called app.java

1. @RestController
2. **class** app{
3. @RequestMapping("/")
4. String home(){
5. "Hello World!"
6. }
7. }

**To compile and run the application type:**

*$ spring run app.java*

To pass command line arguments to the application, you need to use a -- to separate them from the “spring” command arguments, e.g.

*$ spring run app.java -- --server.port=9000*

To set JVM command line arguments you can use the JAVA\_OPTS environment variable, e.g.

*$ JAVA\_OPTS=-Xmx1024m spring run app.java*

**Observation on Code:**

If we observe our **app.java**, we can find the following important points.

* No imports
* No other XML configuration to define Spring MVC Components like Views,ViewResolver etc.
* No web.xml and No DispatcherServlet declaration
* No build scripts to create our Application war file
* No need to build war file to deploy this application

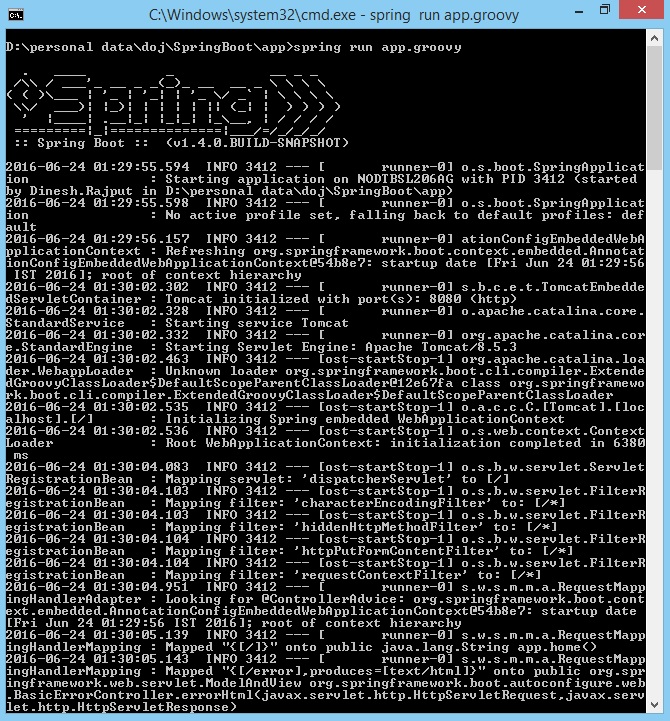
**Now for running the application**

Open command prompt at “**app**” Folder in our Local FileSystem.

**Execute the following command**

*spring run app.java*

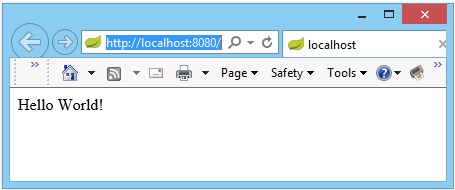
Observe the output at “**spring run**” command console.



If we observe here, when we execute “***spring run app.java***”, it starts Embedded Tomcat server at Default port number: **8080**.( inthis example 9000)

Now our Spring Boot **HelloWorld**Example application is up and running. It’s time to test it now.

* Open browser and access the following link.
* Access this URL: [***http://localhost:8080/***](http://localhost:8080/)



In the above example we can observe following points.

* There is no Import
* There is no XML or JAVA configuration
* There is no DispatcherServlet and web.xml
* There is no build file for creating war like Maven or Gradle

It is the responsibility of Spring Boot Core Components, Groovy Compiler (groovyc) and Groovy Grape (Groovy’s JAR Dependency Manager).

Spring Boot Components uses Groovy Compiler and Groovy Grape to provide some Defaults lime adding required imports, providing required configuration, resolving jar dependencies, adding main() method etc. As a Spring Boot Developer, We don’t need to worry all these things. Spring Boot Framework will take care of all these things for us.

That’s the beauty of Spring Boot Framework.

**Default import statements**

To help reduce the size of your Java code, several import statements are automatically included. Notice how the example above refers to @Component,@RestController and @RequestMapping without needing to use fully-qualified names or import statements.

**Automatic main method**

you do not need to include a public static void main(String[] args) method .ASpringApplication is automatically created, with your compiled code acting as the source.

**Summary**

The Spring Boot CLI takes the simplicity offered by Spring Boot auto-configuration and starter dependencies and turns it up a notch. Using the elegance of the Groovy language, the CLI makes it possible to develop Spring applications with minimal code noise.