Spring Boot

Created by :
Sangeeta Joshi

Agenda

- What is Spring Boot
- Why Spring Boot
- How Spring Boot

What is Spring?

- It is an application framework : unlike single tire framework like hibernate, struts.
- It's the ***only framework to address all architectural tiers*** of typical j2ee application
- It also offers a comprehensive range of service as well as lightweight container

Why Spring Boot

Spring :

- a very popular framework for building Java web and enterprise applications.
- It provides a wide variety of features addressing the modern business needs(via its portfolio projects).
- Unlike many other frameworks which focus on only one area

What is Spring Boot

- Spring Boot is a project created by Spring Team to build production ready spring applications.
- Spring Boot favours convention over configuration
- It is designed to get you up and running as quickly as possible.

Introducing Spring Boot

- Spring Boot makes it easy to create stand-alone, production-grade Spring-based Applications that we can run.
- We have **an opinionated view** of the Spring platform and third-party libraries, so that we can get started with minimum fuss.
- Most Spring Boot applications need very little Spring configuration.
- We can use Spring Boot to create Java applications that can be started by using java -jar or more traditional war deployments.
- Spring provides a command line tool that runs “spring scripts”.

Introducing Spring Boot

Primary goals are:

- Provide a radically faster and widely accessible getting-started experience for all Spring development.
- Be opinionated out of the box but get out of the way quickly as requirements start to diverge from the defaults.
- Provide a range of non-functional features that are common to large classes of projects (such as embedded servers, security, metrics, health checks, and externalized configuration).
- Absolutely no code generation and no requirement for XML configuration.

Why Spring Boot

Spring :

- provides flexibility to configure beans in multiple ways such as: **XML**, **Annotations,** and **JavaConfig**.
- With number of features increased complexity also gets increased
- configuring Spring applications becomes tedious and error-prone.

Spring Boot :

Spring Boot is created to address complexity of configuration.

Why Spring Boot

Usecase :

We want to build a Web Application with:

Spring MVC , JPA(Hibernate) and MySql DB

Various configurations-steps needed:

- Maven Dependencies
- Service/DAO layer dependencies
- Web Layer MVC dependencies
- Log4j

Why Spring Boot

Problems while doing all those configurations:

- So many configurations so can not get up and run quickly
- If we want to develop another spring web app with similar technology stack ? (copy and tweak?)
- hunt for all the **compatible libraries** for the specific Spring version and configure
- 95% of the times we configure **DataSource, EntitymanagerFactory**,**TransactionManager** etc beans in the same way
- Also configure SpringMVC beans like **ViewResolver**, **MessageSource** etc in the same way most of the times.

Solution : ***an automated way to do it ALL***

Why Spring Boot

Solution : ***an automated way to do it ALL***

(If Spring can automatically do it for me? : that would be awesome!!!.)

- what if Spring is capable of configuring beans automatically?
- What if we can customize automatic configuration using simple customizable properties?

So basically we want Spring to do things automatically but provide flexibility to override default configuration in a simpler way? So that is :

SPRING BOOT

BUILD ANYTHING WITH SPRING BOOT

- Spring Boot is starting point for building all Spring-based applications.
- It is designed to get you up and running as quickly as possible, with minimal upfront configuration of Spring.
    - Get started in seconds using Spring Initializr
    - Build anything - REST API, WebSocket, Web, Streaming, Tasks, and more
    - Simplified Security
    - Rich support for SQL and NoSQL

BUILD ANYTHING WITH SPRING BOOT

- Embedded runtime support - Tomcat, Jetty, and Undertow
- Developer productivity tools such as live reload and auto restart
- Curated dependencies that just work
- Production-ready features such as tracing, metrics and health status
- Works in any IDE - Spring Tool Suite, IntelliJ IDEA and NetBeans

Spring Configuration ways (*without Spring Boot*)

- Spring Framework provide three ways to configure beans :
    1. ***XML Based Configuration*** – By creating Spring Configuration XML file to configure the beans. In Spring MVC, the xml based configuration can be loaded automatically by writing some boiler plate code in web.xml file.

    2. Annotation Based Configuration – By using @Service or @Component annotations. Scope details can be provided with @Scope annotation.

    3. Java Based Configuration – Starting from Spring 3.0, we can configure Spring beans using java programs. Some important annotations used for java based configuration are ***@Configuration, @ComponentScan and @Bean.:***

System Requirements

- Spring Boot 2.0.1.BUILD-SNAPSHOT requires:
  - Java 8 or 9
  - Spring Framework 5.0.5.BUILD-SNAPSHOT or above.
  - Explicit build support is provided for Maven 3.2+ and Gradle 4.

System Requirements

Spring Boot supports following embedded servlet containers:

Name	Servlet Version
Tomcat 8.5	3.1
Jetty 9.4	3.1
Undertow 1.4	3.1

We can also deploy Spring Boot applications to any Servlet 3.1+ compatible container.

Installing Spring Boot

- Spring Boot can be used with “classic” Java development tools

OR

- installed as a command line tool. (Spring Boot Cli)

- Either way, we need Java SDK v1.8 or higher.
  (check your current Java installation by using following command:
  $ java –version)

- To experiment with Spring Boot, we can try the ***Spring Boot CLI*** (Command Line Interface) first.

or

- “classic” installation instructions : Next slide

Installation Instructions for Java Developer

- We can use Spring Boot in same way as any standard Java library.
- Just include appropriate spring-boot-*.jar files on classpath.
- Spring Boot does not require any special tools integration, so we can use any IDE or text editor.

- We can run and debug a Spring Boot application as we would any other Java program.

Using Maven

- Although we could copy Spring Boot jars, it is recommended to use a build tool that supports dependency management

  such as Maven

- It can consume artifacts published to the "***Maven Central***" repository.
- It is possible to get Spring Boot to work with other build systems (Ant, for example), but they are not particularly well supported.

Maven Installation

- Maven can be installed with a package manager.
- If we use OSX Homebrew then:
  - ***brew install maven.***
- Ubuntu users can run:
  - ***sudo apt-get install maven.***
- Windows users with Chocolatey can run
  - ***choco install maven*** *(from administrator prompt)*

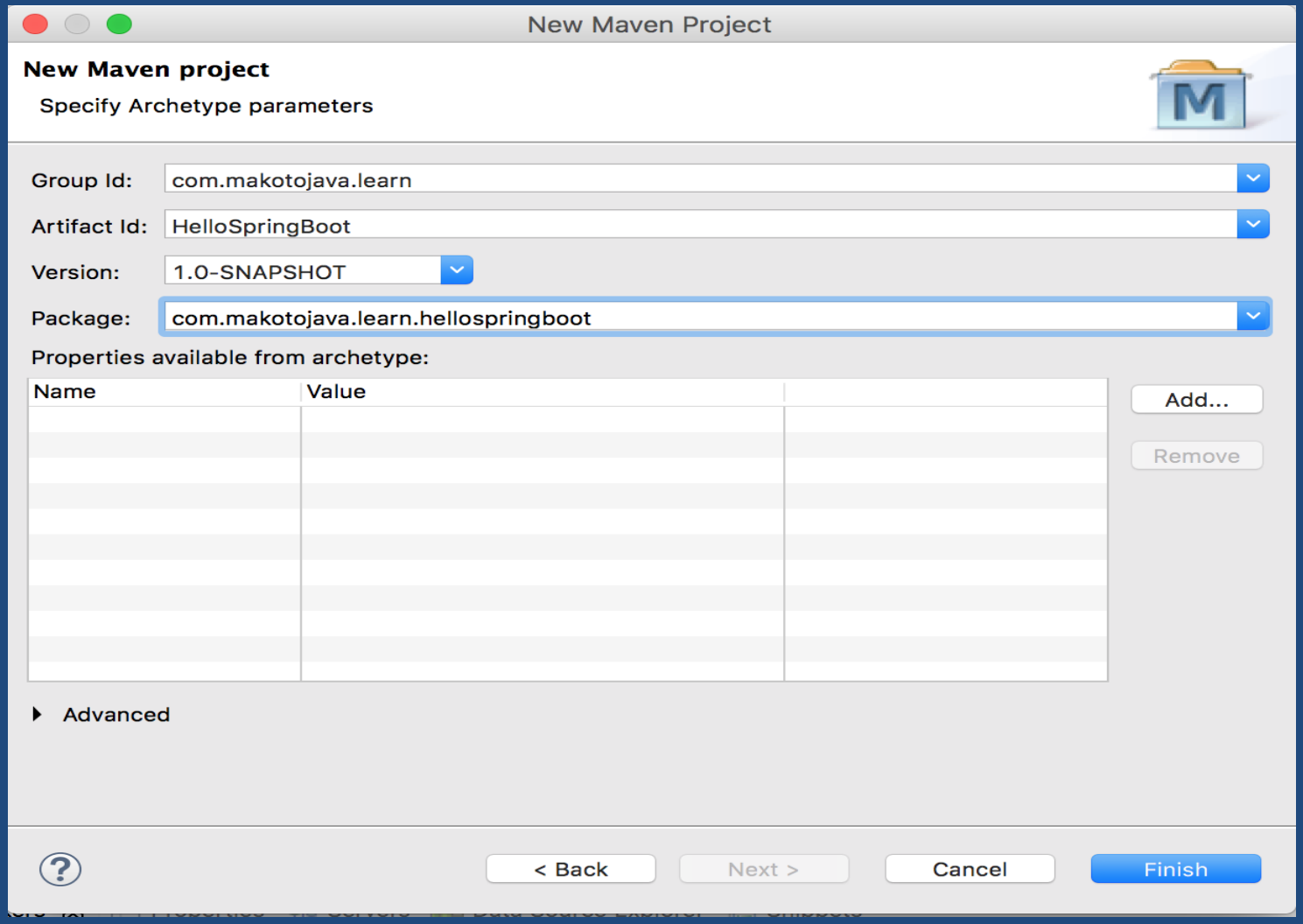
Maven Installation

- Spring Boot is compatible with :
  - ***Apache Maven 3.2 or above.***
- Spring Boot dependencies use :
  - ***org.springframework.boot groupId.***
- Maven POM file inherits from the spring-boot-starter-parent project
- and declares dependencies to one or more “Starters”.
- Spring Boot also provides an optional Maven plugin to create executable jars.

Creating Maven based project

- 1. Create the Maven project
- 2. Select **maven-archetype-quickstart**,(You'll be asked to choose the archetype of your new Maven project.)
- 3. Click Next.

Creating Maven based project



Creating Maven based project

- Group Id: com.makotojava.learn
- Artifact Id: HelloSpringBoot
- Version: 1.0-SNAPSHOT
- Package: com.makotojava.learn.hellospringboot
- Click Finish to create the project.

Creating Maven based project

- Now open App.java in Eclipse and replace its entire contents with the following:

Creating Maven based project

```
package com.makotojava.learn.hellospringboot;

import java.util.Arrays;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class App {
```

Creating Maven based project

```
private static final Logger log = LoggerFactory.getLogger(App.class);

  public static void main(String[] args) {
    SpringApplication.run(App.class, args);
  }

  @Bean
  public CommandLineRunner commandLineRunner(ApplicationContext ctx) {
    return args -> {

      log.debug("Let's inspect the beans provided by Spring Boot:");

      String[] beanNames = ctx.getBeanDefinitionNames();
      Arrays.sort(beanNames);
      for (String beanName : beanNames) {
        log.debug(beanName);
      }

    };
  }
}
```

Creating Maven based project

Then create a new class called HelloRestController in the same package as App that looks like this:

```
package com.makotojava.learn.hellospringboot;

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

@RestController
public class HelloRestController {

  @RequestMapping("/hello")
  public String hello() {
    return "Hello. All your base are belong to us.";
  }
}
```

Creating Maven based project

- Modify the POM created by the New Project wizard so it looks like Listing 1.

- The POM file for HelloSpringBoot

Creating Maven based project

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.makotojava.learn</groupId>
   <artifactId>HelloSpringBoot</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>jar</packaging>
   <name>HelloSpringBoot</name>
   <url>http://maven.apache.org</url>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>1.5.2.RELEASE</version>
   </parent>
```

Creating Maven based project

```
<dependencies>
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
      </dependency>
   </dependencies>
   <properties>
      <java.version>1.8</java.version>
   </properties>
   <build>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
      </plugins>
   </build>
</project>
```