Highlights

- Java 13 + Functional & Immutable style
- Maven Project
- Spring Test non Vintage style + H2DB
- REST API & Unit Test
- Protobuf & Avro

DIVERSITY VERSION

!Beginning Spring Boot 2

ไทยคำอังกฤษคำ

BeID
Enabling your digital transformation

Peerapat Asoktummarungsri
Chief Product & Technology, Co-Founder
c: Be ID Corporation
e: peerapat@beid.io | **w:** www.beid.io
m: 093-494-4493

CONTENT

CHAP 0 Introduction 1
Spring Boot 1
Features 1
Machine Setup 2
สำหรับคนใช้ OS X 2
สำหรับคนใช้ Windows 2
CHAP 1 Initial Project 3
Maven Project 3
Sample pom.xml 5
Main Class 7
Unit Test 8
เกี่ยวกับ Bean Scope ของ Spring จะมีทั้งหมดตามนี้ 11
Other Bean Scope (ที่เรายังไม่สนใจสำหรับคนสนใจ) 11

CHAP 0 Introduction

copy มาจาก https://spring.io

From configuration to security, web apps to big data – whatever the infrastructure needs of your application may be, there is a Spring Project to help you build it. Start small and use just what you need – Spring is modular by design.

Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration

- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

You can also join the Spring Boot community on Gitter [https://gitter.im/spring-projects/spring-boot]!

Machine Setup

สำหรับคนใช้ OS X

```
> brew cask install java
> brew cask install intellij-idea-ce
> brew install maven
> brew install protobuf
```

สำหรับคนใช้ Windows

```
> Set-ExecutionPolicy RemoteSigned -scope CurrentUser
> Invoke-Expression (New-Object
System.Net.WebClient).DownloadString('https://
get.scoop.sh/')
> scoop bucket add java
> scoop install openjdk
> scoop install maven
> scoop bucket add extras
> scoop install protobuf
```

CHAP 1 Initial Project

Maven Project

เราจะสร้าง Spring Boot จากเริ่มอย่างไร โดยทั่วไป Java Spring Boot Project จะสร้างผ่าน Build Tool ชื่อ Maven เป็น Project ที่มี Folder Structure ที่ชัดเจขน โดยเราจะเริ่มจากสร้าง home folder ของ project กันก่อนซึ่งจะมี โครงสร้างดังนี้

```
└──────maven-project
      ├──────pom.xml
      ├──────README.txt
      ├──────NOTICE.txt
      ├──────LICENSE.txt
      └──────src
            ├──────main
            │     ├──────java
            │     ├──────resources
            │     ├──────filters
            │     └──────webapp
            ├──────test
            │     ├──────java
            │     ├──────resources
            │     └──────filters
            ├──────it
            ├──────site
            └──────assembly
```

จากนั้นสร้างไฟล์ pom.xml โดยสำหรับ Spring Boot Project เราจะแบ่งไฟล์ pom.xml เป็นส่วนๆ ดังนี้

```
<project>
   <modelVersion>4.0.0</modelVersion>
  <groupId /> สำหรับกำหนด package ของตัว project
  <artifactId /> สำหรับกำหนด project name
  <version /> สำหรับกำหนด version ของตัว artifact
  <packaging /> สำหรับกำหนด output ของ project
  <parent /> [optional] สำหรับอ้างอิงไปถึง based config ของอีก
artifact
  <properties /> [optional] สำหรับกำหนด properties ต่างๆ
สำหรับ project
  <build /> [optional] สำหรับกำหนดค่าสำหรับการ build ต่างๆ
  <dependencies /> [optional] สำหรับกำหนด dependencies
ต่างๆ สำหรับ project
  <repositories /> [optional] สำหรับอ้างอิงถึง repository
สำหรับ dependencies พิเศษต่างๆ ที่ไม่ได้อยู่ใน default
maven repository
</project>
```

Sample pom.xml

```
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://maven.apache.org/POM/4.0.0"
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>link.colon</groupId>
    <artifactId>springboot-book</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.0.RELEASE</version>
    </parent>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>13</source>
                    <target>13</target>
                    <showDeprecation>true</showDeprecation>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <includeSystemScope>true</includeSystemScope>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <properties>
        <project.build.sourceEncoding>UTF-8</
project.build.sourceEncoding>
    </properties>


     <!-- continue ------------------------------------ -->
```

```
    <!-- continue ------------------------------------ -->

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <repositories>
        <repository>
            <id>spring-milestone</id>
            <url>https://repo.spring.io/libs-release</url>
        </repository>
    </repositories>

    <pluginRepositories>
        <pluginRepository>
            <id>spring-milestone</id>
            <url>https://repo.spring.io/libs-release</url>
        </pluginRepository>
    </pluginRepositories>

</project>
```

Main Class

สำหรับคนที่มีประสบการณ์กับ Spring มาก่อนจะจำได้ว่า Spring จะมีไฟล์ config หลักคือ applicationContext.xml แต่ใน Spring Boot นั้น ถูกออกแบบมาให้เป็น Annotation & Convention แทน โดยเรายจะสร้างเป็นPOJO Class พร้อมกับ Annotation แทน ซึ่งในที่นี้เราจะสร้าง Application.java โดยจะไปสร้างไว้ที่ $PROJ_HOME/src/main/java/link/colon/ สำหรับการ binding bean ต่างๆ นั้น SpringBoot จะทำการ auto scan แล้ว binding ให้เราโดยอัตโนมัติเพื่อให้สะดวกในการพัฒนา

```
package link.colon;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.servlet.ServletComponentScan;

@ServletComponentScan
@SpringBootApplication
public class Application {

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

}
```

หลังจากได้แล้วลอง shell ไปที่ $PROJ_HOME แล้วสั่งรัน command

“mvn clean compile”

Unit Test

ถ้าสามาระสร้าง Project ได้แล้ว ต่อไปคือมาลองเพิ่มในส่วนของ UnitTest กันดูต่อ โดยเบื้องต้นเพิ่ม dependency : spring-boot-starter-test เข้าไปที่ pom.xml ใน node <dependencies /> จากนั้นทดลอง run clean compile อีกรอบ

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
        <exclusion>
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

เนื่องด้วยเราต้องการที่จะทำ UnitTest ในส่วนของ Spring Component ดังนั้นต่อไปเราจะไปสร้าง HelloWorld compoment ไว้ที่ src/main/java/link/colon/component/HelloWorld.java

```
package link.colon.component;

import org.springframework.stereotype.Component;

@Component
public class HelloWorld {
    public String sayHi() { return "Hello World”; }
}
```

```
package link.colon.component;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class HelloWorldTest {

    @Autowired
    private HelloWorld h;

    @Test
    void testSayHi() {
        assert "Hello World".equals(h.sayHi());
    }
}
```

จากนั้นเราจะไปสร้าง HelloWorldTest.java ที่ src/test/java/link/colon/component/HelloWorldTest.java

“mvn clean test”

ตัวอย่างหน้าจอแสดงผลการรัน unit test

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.0.RELEASE)

2020-03-03 19:51:40.292  INFO 34000 --- [           main] link.colon.component.HelloWorldTest      : Starting
HelloWorldTest on Skywalker.local with PID 34000 (started by nuboat in /Users/nuboat/Github/springboot-book)
2020-03-03 19:51:40.293  INFO 34000 --- [           main] link.colon.component.HelloWorldTest      : No active
profile set, falling back to default profiles: default
2020-03-03 19:51:41.303  INFO 34000 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  :
Initializing ExecutorService 'applicationTaskExecutor'
2020-03-03 19:51:41.638  INFO 34000 --- [           main] link.colon.component.HelloWorldTest      : Started
HelloWorldTest in 1.767 seconds (JVM running for 2.909)
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.49 s - in
link.colon.component.HelloWorldTest
```

สำหรับ Component ต่างๆ ของ Spring นั้นเราจะไม่เรียก new instance ตรงๆ แต่จะให้ Spring เป็นคน initiate ให้ โดยเราจะเรียกใช้ผ่าน annotation @Autowired ทั้งในส่วนของ main code และ test code

เกี่ยวกับ Bean Scope ของ Spring จะมีทั้งหมดตามนี้

singleton - (Default) Scopes a single bean definition to a single object instance per Spring IoC container.

prototype - Scopes a single bean definition to any number of object instances.

ซึ่งเราจะได้ใช้งาน bean scope แบบ prototype ต่อในทีหลัง

Other Bean Scope (ที่เรายังไม่สนใจสำหรับคนสนใจ)

request - Scopes a single bean definition to the lifecycle of a single HTTP request; that is, each HTTP request has its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.

session - Scopes a single bean definition to the lifecycle of an HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.

global session - Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware SpringApplicationContext.

application - Scopes a single bean definition to the lifecycle of a ServletContext. Only valid in the context of a web-aware Spring ApplicationContext.

ก่อนจะไป DB Unit Test นั้นเราจะไปเริ่มโดยการเพิ่ม application config ที่ test กันก่อนโดยไปที่ src/test/resource จากนั้น new file application.properties

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
```

จากนั้นเปิดไปที่ pom.xml ตรง <dependencies> แล้วเพิ่ม dependency ที่เกี่ยวข้องตามนี้

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>

        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
        </dependency>
```

สุดท้ายทสร้าง “H2JdbcTest.java” ไว้ที่ที่ folder src/test/java/link/colon/jdbc

```
package link.colon.jdbc;

import lombok.extern.slf4j.Slf4j;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.jdbc.core.JdbcTemplate;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

@Slf4j
@SpringBootTest
class H2JdbcTest {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    private static boolean setUpIsDone = false;

    @BeforeEach
    public void beforeEach() { initialDB(); }

    @Test void testOne() { log.info("testOne"); }

    @Test void testTwo() { log.info("testTwo"); }


    <!-- continue ------------------------------------ -->
```

```
    <!-- continue ------------------------------------ -->

    private void initialDB() {
        if (setUpIsDone) return;

        jdbcTemplate.execute("DROP TABLE IF EXISTS customers");
        jdbcTemplate.execute("CREATE TABLE customers(id INT,
first_name VARCHAR(16), last_name VARCHAR(16))");

        val splitUpNames = Stream
           .of("1 John Woo", "2 Jeff Dean", "3 Josh Bloch")
           .map(name -> name.split(" "))
           .collect(Collectors.<Object[]>toList());

        splitUpNames.forEach(name -> log.info("Inserting customer
record for {} {} {}", name[0], name[1], name[2]));

        jdbcTemplate.batchUpdate("INSERT INTO customers(id,
first_name, last_name) VALUES (?, ?, ?)”, splitUpNames);

        setUpIsDone = true;
    }

}
```

จากนั้นลองทดสอบการรัน H2 Database ผ่าน “mvn clean test”