스프링 부트(Spring Boot)

1. 스프링 부트(Spring Boot)

1-1. Spring Boot 소개

- 스프링 응용프로그램을 독립적으로 작성, 빌드, 실행할 수 있으며 Embedded Tomcat, Jetty 사용이 가능하므로 WAR 파일로 묶어서 배포할 필요가 없다.
- Tomcat이나 Jetty가 내장되어 웹 프로젝트 띄우는 시간이 독립적인 Tomcat을 이용하는 경우에 비해 훨씬 줄어든다. 이렇게 서블릿 컨테이너가 내장되어 있으므로 프로젝트를 .jar 파일 형태로 간단히 만들어 배포할 수 있다.
- 스프링 부트를 사용하면 메이븐의 pom.xml에서 의존하는 라이브러리의 버전을 일일이 지정하지 않아도 된다. 즉 스프링 부트가 적절한 버전을 알아서 관리한다.
- 스프링에서는 XML 파일 또는 JavaConfig 등을 이용하여 설정을 하는데 스프링 부트에서는 자동으로 해준다. 모든 설정을 다 자동화 하지는 않지만 스프링 프레임워크를 시작할 수 있을 정도는 가능하다.(Application.java)
- Spring Tool Suit(STS)를 사용하지 않고도 간단히 스프링 프로젝트를 만들 수 있다.
- 파이썬(python)의 flask, 장고(django) 나 ruby on rails 처럼 빠르게 웹 프로젝트를 만들 수 있는 도구이다.

1-2. Spring Boot & Maven

- 메이븐 설정을 위해 "starter" POMS를 제공한다.
  (Spring Boot에서 미리 정의된 MAVEN Dependency)
- 메이븐에서 사용하려면 <parent>태그로 spring-boot-starter-parent를 상속받아야 하며 기본자바 컴파일러가 버전은 아래처럼 설정 할 수 있다.

```
<properties>
<java.version>1.8</java.version>
</properties>
```

- spring-boot-gradle-plugin은 실행 가능한 jar파일을 만들고 프로젝트를 실행할 수 있게하는 task를 제공한다.
- spring-boot-starter-web을 추가하면 Tomcat 및 Spring MVC 가 자동 추가된다.

```
<!-- Inherit defaults from Spring Boot -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
```

```
        <version>1.3.1.RELEASE</version>
    </parent>

    <!-- Add typical dependencies for a web application -->
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <!-- Package as an executable jar -->
    <build>
        <plugins>
            <plugin>
              <groupId>org.springframework.boot</groupId>
              <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

1-3. Spring Boot & Gradle

■ 메이븐 Gradle에서 사용하려면 "stater POMs"를 직접 import하면 된다..

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:1.3.1.RELEASE")
    }
}
apply plugin: 'java'
apply plugin: 'spring-boot'
repositories {
```

```
    jcenter()
}
dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    testCompile("org.springframework.boot:spring-boot-starter-test")
}
```

1-4. Writing the code(Spring Boot main)

■ 아래와 같이 main 함수를 만들면 된다.

```
//스프링부트 1.2 이상에서 @SringBootApplication 어노테이션은 아래 모든 어노테이션을 포함한다.

//Spring Boot에게 클래스패스 세팅, 다른 빈, 다양한 설정들에 의해 Bean을 추가하도록 지시하는데, 묵시적으로 패키지 탐색의 베이스를 지정하여, JPA 어플리케이션을 작성했다면 현재 패키지가 @Entity 아이템을 찾기 위한 디폴트 패키지가 된다. 일반적으로 기존 Spring MVC에서는 @EnableWebMvc 태그를 사용했지만 Spring Boot는 클래스패스에서 spring-webmvc를 발견할 경우 자동으로 추가한다.
@EnableAutoConfiguration
@ComponentScan  //현재 패키지의 빈,설정,서비스 자동스캔
@Configuration     //현재 파일이 컨텍스트의 설정 파일임을 의미
public class DemoApplication{
    // Spring Boot의 SpringApplication.run() 메소드를 실행, 시작점
    // 웹응용프로그램이면 자동 설정된 Tomcat 웹 서버를
    // 기동하면서 응용프로그램을 시작한다.
    public static void main(String[] args) throws Exception {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

[실행 방법]

1. (DOS에서)프로젝트 루트에서 spring-boot-starter-parent의 run을 이용하여 Maven으로 실행한다.
D:₩..>**mvn spring-boot:run**

2. 이클립스(STS) 프로젝트 루트 -> **Run as -> Maven Build 또는 Java Application(main이 있는 Application클래스 지정)**

2. 이클립스(STS) main이 있는 Application클래스에서 **Run –As -> Spring Boot Application** 또는 Java

Application으로 실행가능 하다.

1-5. Writing the code(CommandLineRunner)

■ 스프링 부트를 시작할 때 Command Line arguments를 주거나, 어떤 코드를 실행하려면 CommandLineRunner 인터페이스의 run(String...args)를 구현하면 된다. 만약 CommandLineRunner 인터페이스 구현이 여러 개 있는 경우 순서를 부여하기 위해서는 @Order 어노테이션을 사용하면 된다.

```
@SpringBootApplication
public class CommandLineRunnerExam {
//Spring Boot 메인
  public static void main(String... args) {
    SpringApplication.run(CommandLineRunner.class, args);
  }
}

class DefaultRunner implements CommandLineRunner {
  //CommandLineRunner의 run 메소드 구현
  //args는 메인 메소드의 아규먼트를 받아들인다.
  public void run(String... args) throws Exception {
    // joining(delimiter, prefix, suffix)
    System.out.println(Arrays.asList(args).stream().collect(Collectors.joining(",",
getClass().getSimpleName() + "[", "]"))); }
}

@Named
@Order(2)
class Runner1 extends DefaultRunner {
}
@Named
@Order(1)
class Runner2 extends DefaultRunner {
}
@Named
@Order(3)
```

```
class Runner3 extends DefaultRunner {
}

[실행]
$ java -jar build/libs/commandlinerunner-exam-1.0.0.jar  이순신 안중근 유관순

Runner2[이순신 안중근 유관순]
Runner1[이순신 안중근 유관순]
Runner3[이순신 안중근 유관순]
```

1-6. Spring Boot에서 property 파일 읽기

```
[src/main/resources/ojc.properties]
url=ojc.asia

[Application.java]
@SpringBootApplication
@PropertySource("ojc.properties")
public class Application {
	public static main(String[] args) {
		SpringApplication.run(Application.class, args);
	}
}

[OjcController.java]
@Controller
public class OjcController {
	@Value("${url}")  //ojc.properties 파일의 url을 값을 주입
	String url;

	@RequestMapping("/ojc")
	public void home() {
		System.out.println(url);  //ojc.asia 출력
	}
}
```