**Spring Boot 게시판**

**Spring Boot 동작 원리**

*Issue Date: 2018/10/15*

*KCC정보통신*

*인턴 김 지 윤*

**목차**

1. **[개요 3](#개요)**
2. [**Spring Boot란? 3**](#SpringBoot란)

**2-1.** [**개념 및 배경 3**](#개념및배경)

**2-2.** [**특징 3**](#특징)

1. [**Maven 빌드환경 4**](#Maven빌드환경)
   1. [**Maven이란? 4**](#Maven이란)
   2. [**Maven Lifecycle 5**](#MavenLifecycle)
   3. [**의존성 6**](#의존성)
2. **[게시판](#프로젝트기본구조)****[프로젝트 기본 구조 6](#프로젝트기본구조)**

**4-1.** [**src/main/wepapp 6**](#webapp)

**4-2.** [**src/main/java 6**](#java)

**4-3.** [**src/main/resources 10**](#resources)

**4-4.** [**pom.xml 10**](#pom)

1. [**참고자료 13**](#참고자료)
2. **개요**

이 문서에서는 Spring Boot의 개념과, Spring Boot 프레임워크로 만든 게시판 프로젝트의 구조와 동작 원리에 대하여 정리한다.

프로젝트의 빌드 툴로는 gradle과 maven 두 가지가 있는데, 이 중 maven을 사용하였고, 후에 gradle 빌드 툴을 사용하여 만든 프로젝트와의 장단점을 비교하기 쉽게 정리한다.

1. **Spring Boot란?**

**2-1. 개념 및 배경**

Spring Boot란, Spring 프레임워크를 기반으로 기존에 개발자가 직접 수작업 해야 했던 복잡한 설정들을 자동화, 간소화 시킬 목적으로 2014년에 개발된 프레임워크이다.

기존의 Spring의 경우, 코드 구현을 시작하기 전 개발 환경 구축을 위한 다양한 설정들을 개발자들이 직접 건드려야 하는 부분이 많았다. 때문에 본인이 잘 모르는 영역은 의미를 잘 모른 채 그대로 긁어와 붙이는 등의 이유로 코드에 불필요한 설정들이 들어가게 되고 결국 매우 비효율적이고, 공유 및 최적화가 어려운 코드가 되어버린다는 단점이 있었다.

이에 따라 Spring의 설정 부분을 대폭 줄여 개발자들이 실제 구현에 앞서 톰캣 WAS, 데이터베이스 등의 설정들을 별도의 설치와 복잡한 설정 없이 연동시킬 수 있게 나온 것이 Spring Boot 프레임워크이다.

**2-2. 특징**



- 단독 실행(stand-alone)이 가능한 수준의 스프링 어플리케이션 개발 가능

- 내장된 톰켓, Jetty, UnderTow등의 서버를 이용해 별도 서버 설치가 필요 없음

- 자동화 된 설정 방식을 제공하여 코드 생산성이 높음

- 스프링환경에서 설정했던 XML없이 단순 자바 수준의 설정 방식 제공

- maven, gradle의 빌드 환경 제공

- NoSQL, AWS등의 환경도 자동 제공(AWS : 아마존 웹 서비스, ‘인프라’를 제공. 웹 사이트 구축 시 필요한 서버 및 스토리지를 대여해줌)

* 이와 같이 Spring Boot는 기존 Spring 프레임워크에 서버 및 데이터베이스 등과 같은 복잡한 설정 부분을 추가해 간소화 시켜준 프레임워크이다.

1. **Maven 빌드 환경**
   1. **Maven이란?**

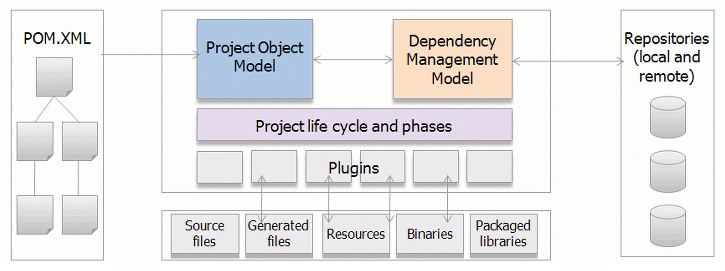
Maven은 프로젝트에 사용되는 라이브러리들과 전체적인 라이프 사이클을 관리하는 도구이며, 기존에는 Ant가 많이 사용되었지만 점차 Maven으로, Maven에서 Gradle로 흐름이 변해가는 추세이다.

* Ant와의 차이점
* Ant가 비교적 자유도가 높음(전처리, 컴파일, 패키징, 테스팅, 배포 가능)
* Maven은 정해진 Lifecycle에 의하여 작업 수행하며, 전반적인 프로젝트 관리 기능까지 포함하고 있음(Build Tool + Project Management)
* Gradle과의 차이점
* Gradle은 XML 대신 groovy 스크립트를 사용하여 동적인 빌드 가능

\* groovy : 스크립트(일반적으로 프로토타이핑이나 테스팅 시)에 자바를 중심으로 하는 구문을 사용하고자 하는 자바 개발자들을 위한 언어, JVM에서만 특성화 되어 구동되고 Java 오브젝트, APIs, tool들과 함께 작동되는 스크립트 언어이다.

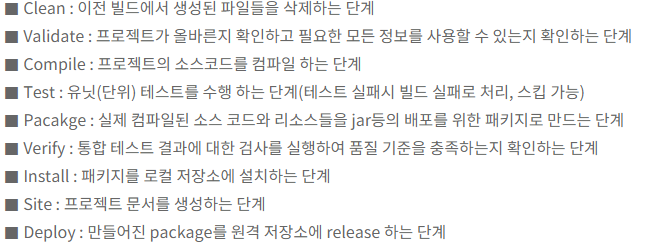
* Maven은 멀티프로젝트에서 상속 구조이고, gradle은 주입 방식. 멀티프로젝트에서 gradle이 더 적합함

Maven의 특징으로는 빌드 절차 간소화, 동일한 빌드 시스템 제공, 프로젝트 정보 제공 등이 있으며 구조는 아래와 같다.



* 1. **Maven Lifecycle**

Maven에서는 미리 정의하고 있는 빌드 순서가 있으며, 이 순서를 Lifecycle이라고 한다. Lifecycle의 각 빌드 단계를 Phase(실행단위로써 goal과 바인딩 됨)라고 하며, 이런 각 Phase들은 서로 의존 관계를 가지고 있다. 의존 관계란, 해당 Phase가 수행되려면 이전 단계의 Phase가 모두 수행되어야 한다는 뜻이다.



주로 많이 쓰이는 빌드 단계는 위와 같으며, 이를 크게 Clean, Default, Site 세가지로 표준 정의한다.

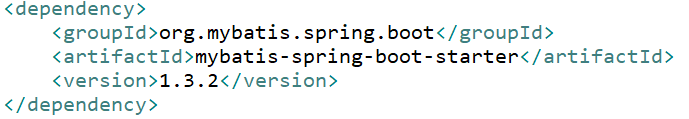
* Clean : 빌드 시 생성되었던 산출물을 삭제
* Default : 프로젝트 배포절차, 패키지 타입별로 다르게 정의 됨
* Site : 프로젝트 문서화 절차

Lifecycle 각 단계를 모두 수행하는 것이 아니라 원하는 단계 까지만 수행할 수도 있으며, test 단계는 큰 프로젝트의 경우 몇 시간 소요 될 수도 있어 수행하지 않도록 스킵 또한 가능하다.

* 1. **의존성**

개발자는 프로젝트에 사용할 라이브러리를 pom.xml에 dependency로 정의만 해 두면 Maven이 repository에서 검색해 자동으로 추가해준다. 심지어 참조하고 있는 library까지 모두 찾아서 추가해 주며, 이를 ‘의존성 전이’라 한다.

예시)

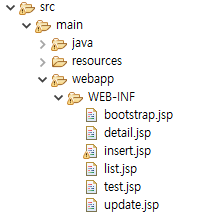


\* pom.xml에 대한 설명은 아래 ‘프로젝트 기본 구조’에서 이어진다.

1. **게시판 프로젝트 기본 구조**

**4-1. src/main/webapp**

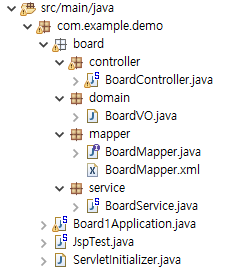
* /WEP-INF/~.jsp



* 처음 생성 시에는 src하위로 webapp까지만 존재한다. 따라서 임베디드된 톰캣이 인식하는 web루트를 생성 해주기 위해 기존 Spring 환경과 동일하게 WEB-INF을 만들어주고, 그 하위에 views등의 jsp파일들을 담을 파일을 생성해야 한다. jsp파일에는 사용자에게 보여질 view 정보를 담는다.

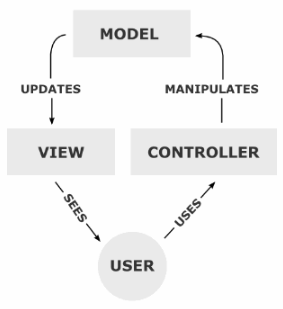
**4-2. src/main/java**

* /com/example/demo



* java부분에는 MVC관련 패키지를 정의한다.

\* **MVC** : Model-View-controller, 컴퓨터 소프트웨어 개발의 구조적 패턴

Model : 데이터 구조 정의(updates application to reflect added item - 응용프로그램에서 데이터를 처리하는 부분, 데이터베이스에 연결하고 데이터를 추출하거나 변환 등의 작업을 수행하는 역할)

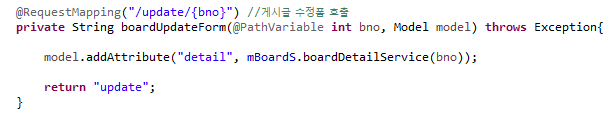
View : UI정의(user clicks ‘add to cart’ - 사용자와의 상호작용을 위한 인터페이스를 표시하는 영역)

Controller : 조작 로직을 포함(receives update form view then notifies model to ‘add item’ - 어플리케이션에서 각 요소들의 연결 관계 설정, 데이터와 시각적 부분의 연결 등을 관리. 대개 url로부터 입력되는 정보로부터 어떤 데이터와 뷰를 연결할지 등을 제어)

* 해당 demo패키지(com.example.demo)에는 스프링을 단독 실행할 수 있는 클래스 파일이 있다.
* Demo.config파일 : 기존 xml에서 설정했던 여러 정보들을 자바 코드로 설정하는 클래스들 정의한 패키지
* Demo.board 패키지 : MVC 환경구성에 필요한 패키지(controller, service, domain, mapper)

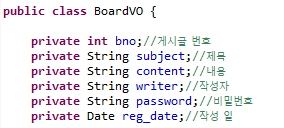
1. /board/controller/BoardContorller.java (@Controller, @RestController 어노테이션 사용)

Controller 클래스는 웹 어플리캐이션의 진입점(entry point) 역할을 한다. (예전의 HttpServlet 클래스와 같은 역할)



URL 호출 시 Request 메소드의 반환형을 String으로 지정했기 때문에, MVC의 ViewResolver는 반환문자인 “update” + .jsp(application.properties에서 설정한 suffix)를 붙여 update.jsp를 찾아 브라우저에 전달한다.

1. )/board/domain/BoardVO.java (@Entity 어노테이션을 주로 사용)

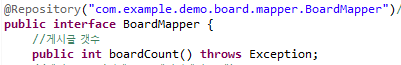


BoardVO.java에서는 VO(Value Object), 데이터를 담는 객체를 정의해준다. 주로 DB에서 한 테이블에 대한 컬럼들과 매칭하여 사용하며, MyBatis 등에서도 DAO(Data Access Object)에서 테이블에 대한 결과를 VO에 담아 처리하는 방식을 많이 사용한다. Getter와 setter 메서드를 통해 값을 접근한다.

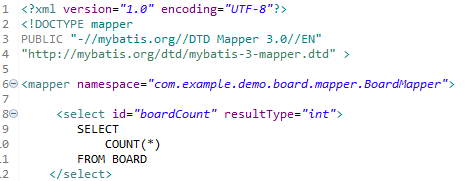
VO와 비슷한 개념으로 DTO(Data Transfer Object)가 있다. VO는 read only의 속성을 지니고, DTO는 데이터가 포함되어 있는 객체를 다른 시스템으로 전달하는 역할을 한다. 때문에 VO는 특정한 값만 담아 보관하는 역할의 객체, DTO는 통신 용도로 교환하는 VO의 객체라고 볼 수 있다.

1. /board/mapper/BoardMapper.xml, BoardMapper.java

BoardMapper.java : Model과 연결 될 mapper의 interface를 작성한다. @Repository 어노테이션을 사용하는데, @Repository는 해당 클래스가 데이터베이스에 접근하는 클래스임을 명시한다.

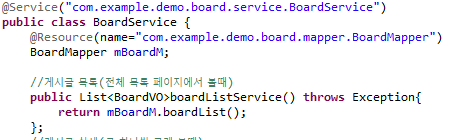


BoardMapper.xml : Mybatis와 연결시켜 실제 SQL문을 갖는 핵심 XML파일로, mapper의 interface와 동일한 패키지 경로에 넣는다. Mybatis 이외에도 mapper annotation을 이용해 SQL문을 지정할 수 있지만, Mybatis가 XML을 기반으로 SQL을 직관적이고 깔끔하게 관리할 수 있다는 장점을 갖는다.



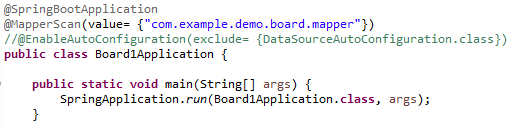
1. /board/service/BoardService.java (@Service 어노테이션 사용)

@Service 어노테이션은 Repository를 통해 데이터베이스에서 데이터를 가져온 후 Controller에게 전달해 주는 클래스임을 명시한다.



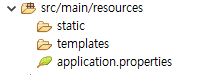
1. Application.java(Board1Application.java)

스프링 부트를 실행시키기 위한 핵심 코드이며, 프로젝트 생성 시 자동으로 만들어지는 클래스이다. 내장 톰켓 덕분에 서버를 따로 돌릴 필요 없이 자바 어플리케이션 실행처럼 App으로 실행을 돌리면 톰켓도 내부적으로 함께 실행되는 구조로, 실행 기능을 run()메소드가 해준다.



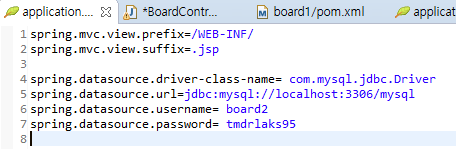
@SpringBootApplication : 설정과 환경을 담당하는 @Configuration, @EnableAutoConfiguration, @ComponentScan 3개의 어노테이션을 디폴트 속성으로 함께 가져간다.

**4-3. src/main/resources**



* application.properties

Spring Boot의 환경설정을 해 주는 곳으로, 데이터베이스 연결 설정, security 설정, MVC 속성 등을 해준다.



전통적으로 tomcat기반 자바 웹 어플리케이션에서는 보안 상 jsp의 위치를 URL로 접근할 수 없는 WEB-INF폴더 아래에 위치시킨다.

* Static

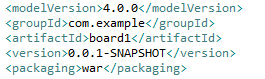
css, js, image등의 정적파일들이 저장되는 곳이다.

**4-4. pom.xml**

* POM(Project Object Model) : Maven의 빌드 파일로, 빌드 정보들을 기술하는 파일이다.

Maven의 강점은 다양한 라이브러리와 프레임워크 등을 이용하는 경우도 이를 모두 Maven이 관리 해준다는 점인데, pom.xml파일에서 이 라이브러리들의 의존성 관계를 모두 정의해준다.

* 프로젝트 정보



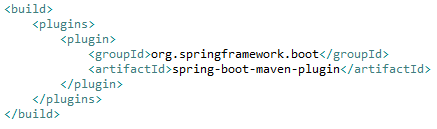
groupId : 프로젝트 속하는 그룹 식별 값. 회사, 본부, 단체를 의미하는 값이 오며, 패키지 형식으로 계층을 표현한다.

artifactId : 프로젝트 결과물의 식별 값. 프로젝트나 모듈을 의미하는 값이 온다.

Version : 결과물의 버전을 나타낸다.

Packaging : 기본적으로 생성할 패키지를 입력한다. 별도로 입력하지 않으면 groupId와 동일한 구조의 패키지를 생성한다.

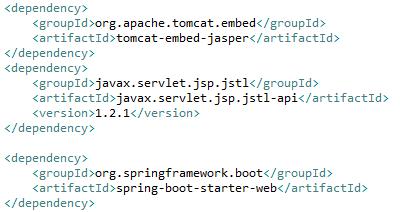
* 빌드 정보



Project 태그 바로 하위에 위치하며, <plugins>는 빌드에서 사용할 플러그인을 뜻한다.

* Dependencies

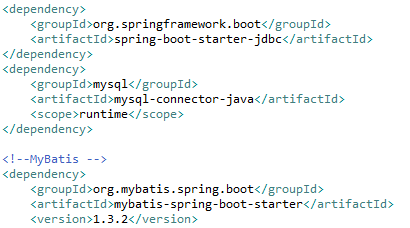
의존성 라이브러리 정보를 포함하며, 최소 groupId, artifactId, (version) 정보가 필요하다. <dependencies>섹션 아래에 세부 <dependency>가 기술된다.



임베디드 톰캣 사용과, JSTL 사용을 위한 정의이다.

JSTL(Jsp Standard Tag Library)은 JSP파일에 자바 형식의 코드를 사용하면 불편한 점을 해결해준다. 나만의 태그를 추가할 수 있는 기능을 제공하는데, 연산이나 조건문, 반목문, 데이터베이스 등을 편하게 처리할 수 있게 도와준다.

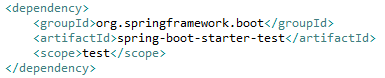
jsp파일에서 <%@taglib prefix=”c” uri=”http://java.sun.com/jsp/jstl/core”%>와 같이 정의하고 필요한 JSTL 태그는 prefix와 uri만 바꿔 정의한다. 태그의 종류는 5가지이다.



Mybatis와 JDBC, mysql연결 설정을 위한 정의이다.

Mybatis : SQL, 저장프로시저, 몇 가지 고급 매핑을 지원하는 퍼시스턴스 프레임워크(Persistence Framework)이다. Mybatis는 JDBC로 처리하는 상당부분의 코드와 파라미터 설정 및 결과 매핑을 대신 해주며, XML파일에서 자유롭게 쿼리 작성이 가능해진다.

JDBC(Java Database Connectivity) : 자바에서 데이터베이스에 접속할 수 있도록 하는 자바 API로, 데이터베이스에서 자료를 쿼리하거나 업데이트하는 방법을 제공한다.



어플리케이션을 테스트하기 위한 정의이다. Spring-boot-starter-test는 Junit, AssertJ, Hamcrest를 포함한 여러 유용한 라이브러리를 포함하고 있으며, spring-boot-test는 @SpringBootTest라는 어노테이션을 제공한다.



Restful API형식으로 게시판을 변환시킬 때 추가한 라이브러리로, json데이터를 java object로, java object를 json 형식으로 변환시켜준다. Jackson은 자바용 json라이브러리로 알려져 있지만, json 뿐만 아니라 XML, YAML, CSV 등 다양한 형식의 데이터를 지원하는 data-processing 툴이다.

1. **참고자료**

<https://m.blog.naver.com/PostView.nhn?blogId=scw0531&logNo=221066404723&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F> -SpringBoot 개념 및 동작 원리

<http://sjh836.tistory.com/131> : Maven 개념, Lifecycle

<http://boxfoxs.tistory.com/324> : Maven 개념

<http://yonguri.tistory.com/entry/%EC%8A%A4%ED%94%84%EB%A7%81%EB%B6%80%ED%8A%B8-SpringBoot-%EA%B0%9C%EB%B0%9C%ED%99%98%EA%B2%BD-%EA%B5%AC%EC%84%B1-2-MVC-%ED%99%98%EA%B2%BD%EA%B5%AC%EC%84%B1> : controller, MVC환경구성

<http://opennote46.tistory.com/124> : SpringBootApplication 어노테이션

<http://hyper-cube.io/2017/08/06/spring-boot-test-1/> : spring-boot-starter-test

<https://hunit.tistory.com/203> : JSTL 정의 및 사용법

<https://wikidocs.net/18340> : pom.xml 개념

<http://yellowh.tistory.com/109> : 어노테이션 정리

<http://webskills.kr/archives/479> : MVC 정의

<https://elfinlas.github.io/2018/02/15/dtovo/> : VO개념