

Spring-boot시작 및 JSP사용

강병준

스프링부트의 기능

스프링부트는 다음과 같은 특성을 갖고 있다.

단독으로 스프링 어플리케이션 제작 가능

내장된 Tomcat, Jetty, Undertow를 바로 실행가능 (WAR 파일배포없이 가능)

최대한 자동화된 설정을 제공 : POM이 제공되어 Maven 구성이 단순

Spring 환경을 자동으로 설정 : XML설정과 코드 생성이 필요 없음

스프링부트를 시작하기 위해 **Spring Tool Suite (STS)** 를 설치해야 한다.

여기서 STS는 이클립스기반 스프링 개발 환경을 말한다. 이클립스 (Eclipse)를 사용해도 되지만, 이클립스의 경우 스프링 환경설정을 위해 설정해야 할 것들이 많으니 스프링 개발에 적합하게 제작된 STS를 사용하였다.

스프링 부트는 단독 실행되는 상용화 가능한 수준의 스프링 기반 애플리케이션을 쉽게 만들어낼 수 있으며 최소한의 설정으로 스프링 플랫폼과 3rd Party 라이브러리들을 사용할 수 있도록 함

스프링 부트는 단독 실행되는 상용화 가능한 수준의 스프링 기반 애플리케이션을 쉽게 만들어낼 수 있으며 최소한의 설정으로 스프링 플랫폼과 3rd Party 라이브러리들을 사용할 수 있도록 함

Spring Boot 장단점

✓ 장점

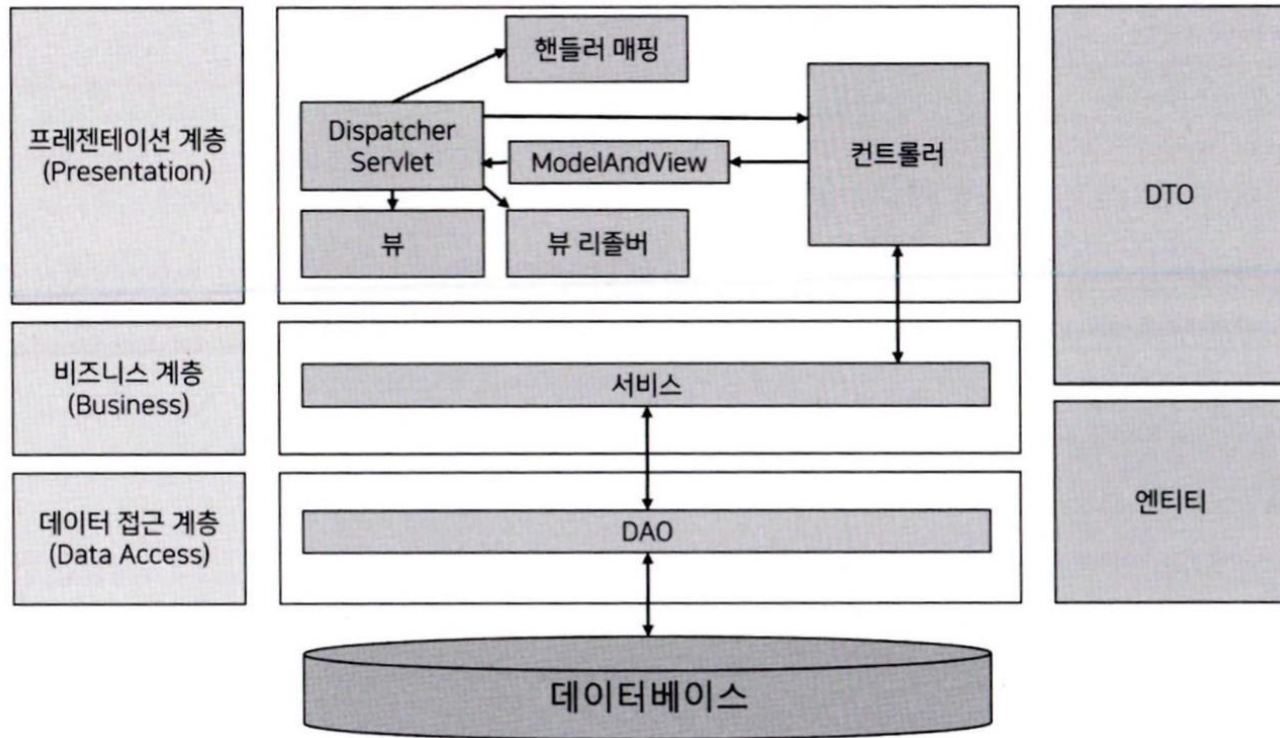
- 자동 설정 - 환경 설정을 최소화
- WAS를 내장해서 독립 실행이 가능한 스프링 애플리케이션 개발이 가능
- Spring Boot Starter 의존성을 제공하여 Maven/Gradle 의 설정을 간소화 하고 자동화된 설정 제공
- XML 설정 없이 자바 수준의 설정 방식 제공
- JAR을 사용하여 자바 옵션 만으로 배포 가능
- 애플리케이션의 모니터링과 관리를 위한 Spring Actuator 제공

✓ 단점

- 설정을 자동화하지 않고 별도로 설정하면 버전을 올릴 때 기존 스프링 프레임워크를 사용하는 것과 동일한 불편함
- 특정 설정을 별도로 하거나 설정 자체를 변경하고 싶다면 내부의 설정 코드를 살펴봐야 하는 불편함

Spring Boot

- ❖ Spring Boot
 - ✓ Layerd Architecture



Spring Boot

❖ Spring Boot

✓ Layerd Architecture

- 프레젠테이션 계층
 - ◆ 유저 인터페이스(UI; User Interface) 계층
 - ◆ 클라이언트 와 의 접점
 - ◆ 클라이언트로부터 데이터와 함께 요청을 받고 처리 결과를 응답으로 전달하는 역할
- 비즈니스 계층
 - ◆ 서비스(Service) 계층
 - ◆ 핵심 비즈니스 로직을 구현하는 영역
 - ◆ 트랜잭션 처리나 유효성 검사 등의 작업도 수행
- 데이터 접근 계층
 - ◆ 영속(Persistence) 계층
 - ◆ 데이터베이스에 접근해야 하는 작업을 수행
 - ◆ DAO라는 컴포넌트로 표현했지만 Spring Data JPA에서는 DAO 역할을 리포지토리가 수행하기 때문에 리포지토리로 대체할 수 있음

Spring Boot

❖ Spring Boot Starter

- ✓ 의존 관계를 개발자가 간편하게 설정할 수 있도록 도와주는 특정 목적을 달성하기 위한 의존성 그룹
- ✓ 스프링과 JPA가 필요하다면 pom.xml 파일이나 build.gradle 파일에 spring-boot-starter-data-jpa 만 추가하면 됨
- ✓ 명명 규칙
 - spring-boot-starter-*: *에 해당 starter 이름을 명시
 - 스프링에서 웹 관련 프로젝트를 진행하는 경우: Spring-boot-starter-web

Ant, Maven, Gradle

웹/안드로이드 등등 프로그래밍 개발이 발전하며 프로젝트 생성에 필요한 라이브러리도 점점 많아지게 되었다. 이렇게 많아진 라이브러리를 직접 사이트에서 다운로드 받아 시스템에 추가하는 방법이 있지만 번거로움이 뒤따른다. 여기서 빌드도구(Build Tool)가 등장한다. 빌드 도구란, 소스코드를 컴파일, 테스트, 정적분석을 통하여 실행 가능한 애플리케이션으로 자동 생성하는 프로그램으로 1)빠른 기간 동안에 계속해서 늘어나는 라이브러리의 추가와 2)프로젝트를 진행하며 라이브러리의 버전을 동기화하기 어렵기 때문에 등장하였다.

1. Apache ANT

Ant는 Java 기반의 빌드 도구로 다른 빌드 도구보다 역사가 오래되었다. Ant는 개발자가 원하는 것을 개발할 수 있다는 유연성에 큰 장점이 있다.

[ANT의 특징]

각 프로젝트에 대한 XML기반 빌드 스크립트 개발

형식적인 규칙이 없음 : 결과물을 넣을 위치를 정확히 알려줘야 하며, 프로젝트에 특화된 Target과 Dependency를 이용해 모델링

절차적 : 명확한 빌드 절차 정의가 필요

생명주기를 갖지 않기 때문에 각각의 target에 대한 의존관계와 일련의 작업을 정의해 주어야 함

[ANT의 단점]

유연성이 높으나 프로젝트가 복잡해질 경우 각각의 Build과정을 이해하기 어려움

XML, Remote Repository를 가져올 수 없었음 (IVY 도입)

스크립트의 재사용이 어려움

Ant, Maven, Gradle

2. Apache Maven

Maven은 프로젝트에 필요한 모든 'Dependency (종속성)'를 리스트의 형태로 Maven에게 알려 관리 할 수 있도록 돕는 방식을 말한다.

Dependency를 관리하고, 표준화된 프로젝트(Standardized project)를 제공

XML, remote repository를 가져 올 수 있음 : 개발에 필요한 종속되는 'jar', 'class path'를 다운로드 할 필요 없이 선언만으로 사용 가능

상속형 : 하위 XML이 필요 없는 속성도 모두 표기

과 같은 기능을 한다. 즉, 'POM.xml' 이라는 Maven 파일에 필요한 'Jar', 'Class Path'를 선언해 주면 직접 다운로드 할 필요 없이 Maven은 Repository에서 필요한 모든 파일들을 해당 프로젝트로 불러와 준다. 이러한 장점에도 불구하고, Maven은 몇가지 단점이 있는데 그것은 바로 아래와 같다.

라이브러리가 서로 종속할 경우 XML이 복잡해짐

계층적인 데이터를 표현하기에 좋지만, 플로우나 조건부 상황을 표현하기엔 어려움
편리하나 맞춤화된 로직 실행이 어려움

Ant, Maven, Gradle

3. Apache Gradle

최근 소프트웨어개발 범위의 변화에 따라 빌드의 자동화에 대한 요구도 증가하게 되었다. Gradle은 JVM 기반의 빌드 도구로 기존의 Ant와 Maven을 보완하였다. 따라서 JAVA 혹은 Groovy를 이용해 logic을 개발자의 의도에 따라 설계할 수 있다.

오픈소스기반의 build 자동화 시스템으로 Groovy 기반 DSL(Domain-Specific Language)로 작성

Build-by-convention을 바탕으로함: 스크립트 규모가 작고 읽기 쉬움

Multi 프로젝트의 빌드를 지원하기 위해 설계됨

설정 주입 방식 (Configuration Injection)

따라서 초기 프로젝트 설정에 드는 시간을 절약할 수 있으며 기존의 Maven이나 Ivy등과 같은 빌드 도구들 과도 호환이 가능하다는 점이다.

Build Tool

❖ Maven

✓ Maven

- 아파치 메이븐은 자바 기반의 프로젝트를 빌드하고 관리하는 도구
- <https://maven.apache.org>
- 초창기 자바 프로젝트의 대표적 관리 도구였던 Ant를 대체하기 위해 개발
- pom.xml 파일에 필요한 라이브러리를 추가하면 해당 라이브러리에 필요한 라이브러리까지 함께 내려받아 관리
- 기능
 - ◆ 프로젝트 버전과 아티팩트를 관리
 - ◆ 의존성을 관리하고 설정된 패키지 형식으로 빌드를 수행
 - ◆ 빌드를 수행하기 전에 단위 테스트를 통해 작성된 애플리케이션 코드의 정상 동작 여부를 확인
 - ◆ 배포 빌드가 완료된 패키지를 원격 저장소에 배포

Build Tool

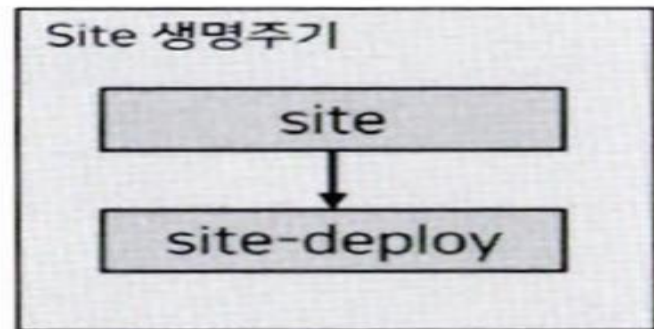
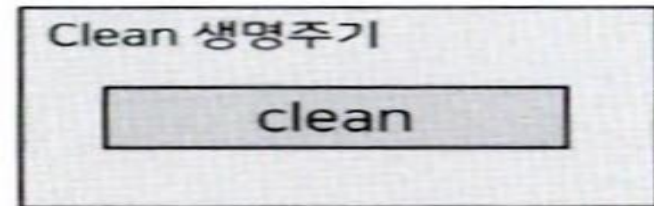
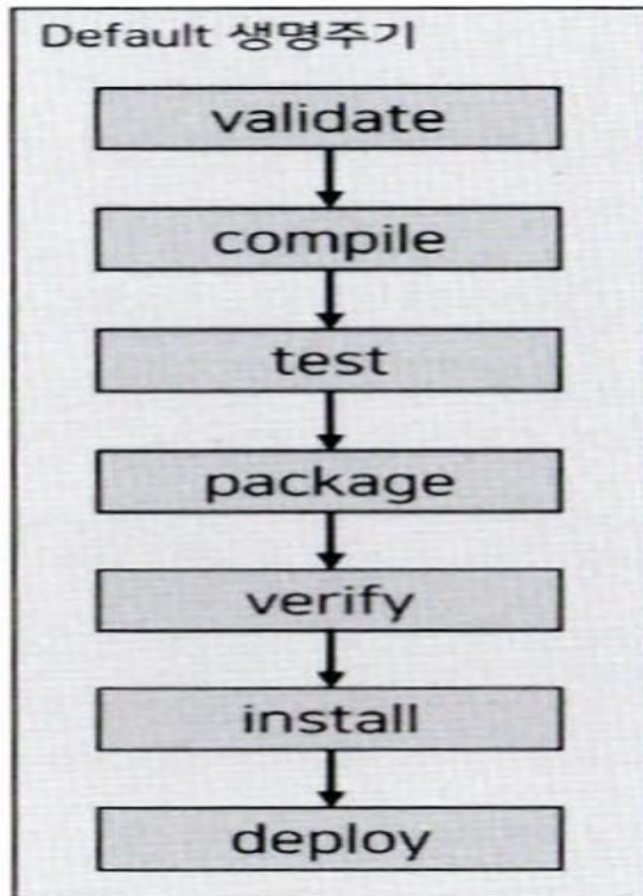
✓ Maven

● 수명 주기

- ◆ clean: 이전 빌드가 생성한 모든 파일을 제거
- ◆ validate: 프로젝트를 빌드하는 데 필요한 모든 정보를 사용할 수 있는지 검토
- ◆ compile: 프로젝트의 소스 코드를 컴파일
- ◆ test: 단위 테스트 프레임워크를 사용해 테스트를 실행
- ◆ package: 컴파일한 코드를 가져와서 JAR 등의 형식으로 패키징을 수행
- ◆ verify: 패키지가 유효하며 일정 기준을 충족하는지 확인
- ◆ install: 프로젝트를 사용하는 데 필요한 패키지를 로컬 저장소에 설치
- ◆ deploy: 프로젝트를 통합 또는 릴리스 환경에서 다른 곳에 공유하기 위해 원격 저장소에 패키지를 복사
- ◆ site: 메이븐의 설정 파일 정보를 기반으로 프로젝트의 문서 사이트를 생성
- ◆ site-deploy: 생성된 사이트 문서를 웹 서버에 배포

Build Tool

- ✓ Maven
 - 수명 주기



Build Tool

❖ gradle

- ✓ Groovy 기반의 빌드 도구
- ✓ Ant 로부터 기본적인 빌드 도구의 기능을 가져오고 Maven 으로 부터 의존 라이브러리 관리 기능을 가져와서 사용
- ✓ Multi Project 구성 시에는 Maven 처럼 상속 구조가 아닌 설정 주입 방식을 사용하여 훨씬 유연하게 빌드 환경을 구성
- ✓ gradle 설치: <https://gradle.org/install/>
 - Mac - \$ brew install gradle
 - Windows는 다운로드 받아서 설치
- ✓ gradle 설치 이유
 - gradle wrapper 와 관련된 설정을 해주는데 첫 설정 시 gradle 관련 빌드 설정을 자동으로 해주며 이를 이용해서 Git 과 같은 Version Control System 에서 관리하면 공동 작업자들이 gradle 설치 및 버전 관리를 편리하게 할 수 있음

Build Tool

❖ gradle

✓ gradle 설정 관련 기본 구조

```
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradlew
└── gradlew.bat
```

- gradlew: 리눅스 및 Mac OS 등 Shell Script
- gradlew.bat: 윈도우즈 용 Batch Script
- gradle-wrapper.jar: Wrapper JAR
- gradle-wrapper.properties: gradle 설정 정보 프로퍼티 파일(버전 정보 등)

실행 환경 속성 설정

❖ application.properties

- ✓ 스프링 부트 애플리케이션 실행 시 사용하는 여러 가지 설정 값들을 정의하는 파일
- ✓ src/main/resources 디렉토리 아래에 자동으로 생성되며 자동으로 생성되지 않았다면 직접 생성해서 사용 가능
- ✓ 개발 환경, 테스트 환경, 운영 환경에 따라서 연결해야 할 데이터베이스, port, debug level 등을 나눠야 한다면 다음 명명 규칙으로 설정 파일을 생성

application-{profile}.properties

- 개발 환경의 설정 파일은 application-dev.properties로 만들고 운영 환경의 설정 파일은 application-prod.properties로 생성
 - 실행되는 환경에 따라서 어떤 설정 파일을 사용할지를 jar 파일 실행 시 VM 옵션 등을 통해 지정할 수 있음
- ✓ application.properties에 설정해 둔 값을 자바 코드에서 사용해야 한다면 @Value 어노테이션을 통해서 읽어올 수 있음

개발환경

1. STS란?

- 이클립스에서 스프링을 편하게 쓰기 위해 스프링 IDE 프로젝트를 만들었고 로드 존슨이 설립한 스프링 지원회사인 스프링소스(SpringSource)에서 스프링소스툴 스위트(STS, SpringSource Tools Suite)를 만들었다. 한때 유료제품이었지만 지금은 무료.

- Download : google에서 sts download조회

<https://spring.io/tools/>

<http://docs.spring.io/spring/docs/current/javadoc-api/>

spring®

Why Spring ▾

Learn ▾

Projects ▾

Training

Support

Community ▾



Spring Tools | 4

Spring Tools 4 is the next generation of Spring tooling for your favorite coding environment. Largely rebuilt from scratch, it provides world-class support for developing Spring-based enterprise applications, whether you prefer Eclipse, Visual Studio Code, or Thelia IDE.

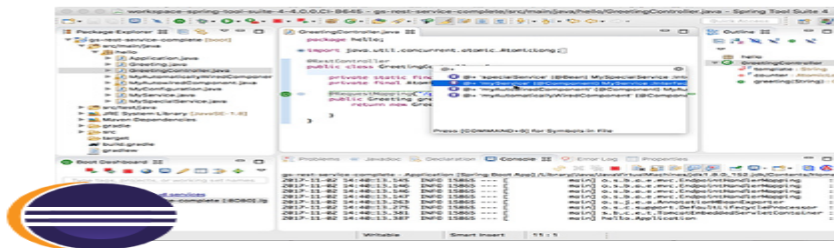
Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4.
Free. Open source.

LINUX 64-BIT

MACOS 64-BIT

WINDOWS 64-BIT



Gradle

Eclipse Marketplace


Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.




Search Recent Popular Favorites Installed Giving IoT an Edge

Find: All Markets All Categories Go

Buildship Gradle Integration 3.0


 Extend your Eclipse IDE to support building software using Gradle. This solution is provided by the Eclipse Foundation. [more info](#)

by [Eclipse Buildship Project](#), EPL
[fileExtension_gradle](#)


★ 1026  Installs: **451K** (3,146 last month) Installed

Grace tool
Buildship로 조회

Minimalist Gradle Editor 1.0.1

 <https://github.com/Nodeclipse/GradleEditor> Minimalist Gradle Editor for build.gradle files with highlight for keywords, strings and matching brackets and android... [more info](#)

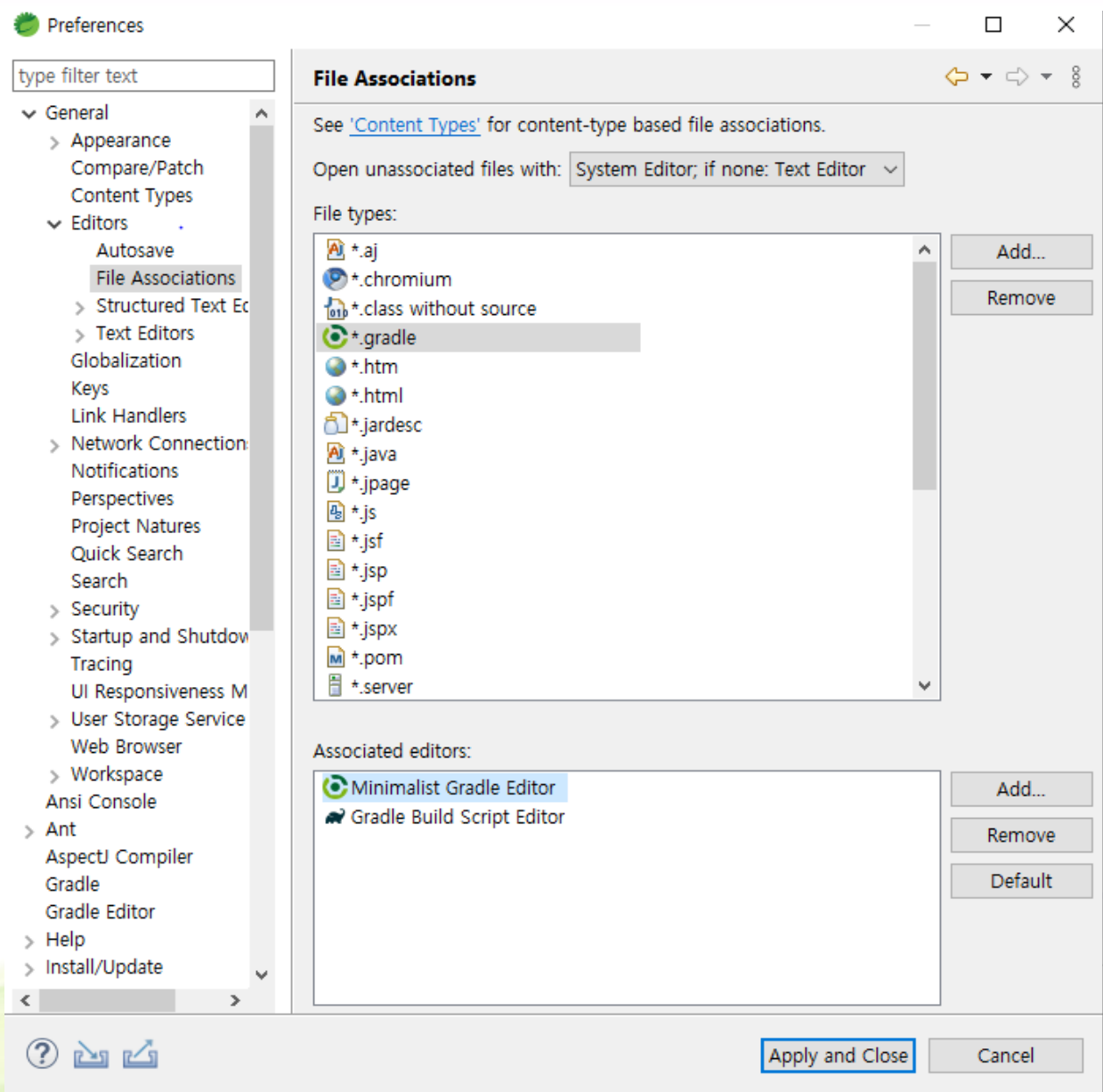
by [Nodeclipse/Enide](#), GPL
[gradle editor highlight build android ...](#)

★ 91  Installs: **77.6K** (523 last month) Installed

Grace tool editor
Minimalist 로 조회

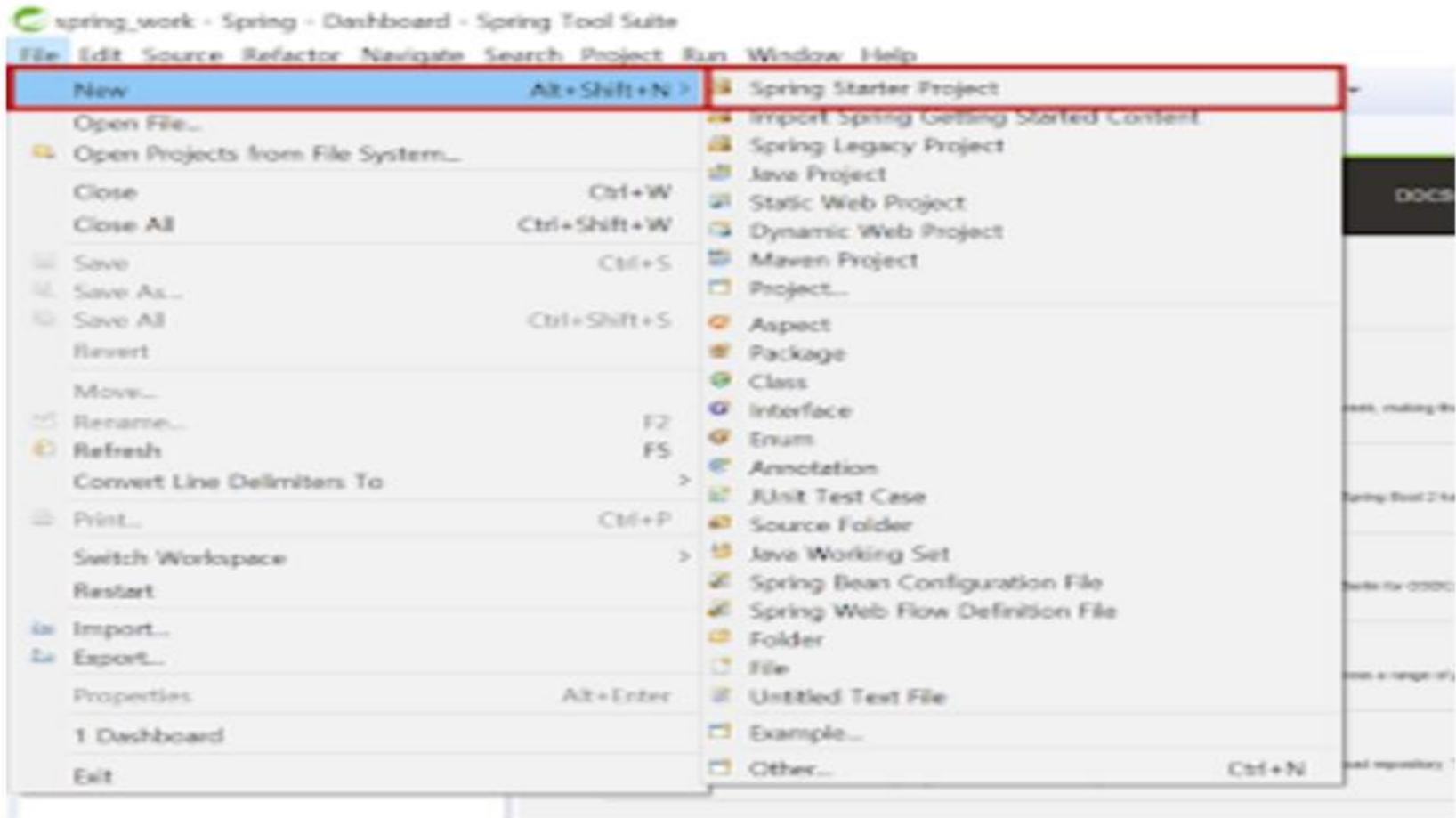
Gradle

그레이스 툴 파일에 연결 Window – Preferences General – Editor - File Associations Minimalist Grace Editor - Default



Spring Boot 프로젝트 생성하기

스프링부트 프로젝트를 생성하기 위해 STS를 실행해 'Spring Starter Project'로 스프링부트 프로젝트를 만든다



File > New > Spring Start Project 메뉴를 선택

Spring Boot 프로젝트 생성하기

1. New Spring Starter Project창이 열리며, 이 창을 통해 새 프로젝트를 생성할 수 있다. Name, Type, Group, Description, Package 등을 다음과 같이 수정해 주었다. 다음 요소들은 개발자 개발 목적과 원하는 이름에 따라 수정해 주면 된다.
2. Name 은 Project의 이름이 된다. 예를 들어 lauraSpring 이라고 이름을 지을 경우, lauraSpring 이름의 프로젝트가 생성되는 것이다.
3. Type은 build 방식을 설정하는 것으로 STS 에는 Maven과 Gradle 방식이 있다. 여기선 Gradle 을 선택하였다. 원하는 Build 방식에 따라 설정해 주면 된다.
4. Maven과 Gradle에 대한 설명은 다음페이지를 참조
5. Group, Description은 임의로 설정하였다.
6. Package는 보통 Group을 따라간다.
7. 설명

Group : 프로젝트를 생성하는 생성자의 소속을 입력해 준다. 보통 default로 com.example 가 입력되어 있다.

Artifact : 프로젝트에서 만들어질 제품/서비스의 이름을 입력하면 된다. 프로젝트 명과 동일할 수도 있다.

Name : 프로젝트의 이름을 설정해 줄 수 있다.

Description : 프로젝트의 설명을 입력할 수 있다.

Package Name : 프로젝트 내에 주로 활용할 코드 경로와 이름을 지정해 줄 수 있다. default일 경우, group 뒤에 프로젝트의 이름이 붙는다.

Spring Boot 프로젝트 생성하기

Service URL	<input type="text" value="https://start.spring.io"/>		
Name	<input type="text" value="sp_boot1-1"/>		
<input checked="" type="checkbox"/> Use default location			
Location	<input type="text" value="E:\spring\springSrc\sp_boot1-1"/>	<input type="button" value="Browse"/>	
Type:	<input type="text" value="Gradle (Buildship 3.x)"/>	Packaging:	<input type="text" value="Jar"/>
Java Version:	<input type="text" value="8"/>	Language:	<input type="text" value="Java"/>
Group	<input type="text" value="com.example"/>		
Artifact	<input type="text" value="sp_boot1-1"/>		
Version	<input type="text" value="0.0.1-SNAPSHOT"/>		
Description	<input type="text" value="Demo project for Spring Boot"/>		
Package	<input type="text" value="com.example.demo"/>		
Working sets			
<input type="checkbox"/> Add project to working sets	<input type="button" value="New..."/>		
Working sets:	<input type="text" value=""/>	<input type="button" value="Select..."/>	

옵션 설정

- Type 콤보 박스는 Build 방법을 선택
- Packaging 콤보 박스는 원하는 애플리케이션 형태를 선택
- Java Version 콤보 박스는 자바 버전을 선택
- Language 콤보 박스는 프로그래밍 언어를 선택
- Group 은 패키지의 상위 경로를 적는데 보통 도메인 주소의 역순으로 작성하는데 예를 들어 도메인이 company.com 일 경우 패키지 상위 경로는 com.company
- Artifact 에는 프로젝트 약칭을 적는데 라이브러리가 됐을 때 jar에 들어가는 이름
- Version 은 버전 값
- Description 은 프로젝트 설명
- Package 에는 패키지 경로를 적는데 패키지 상위 경로 뒤에 프로젝트명을 추가하는 형태로 작성

Spring Boot 프로젝트 생성하기

1. 모두 작성하였다면 Next> 를 선택해 [New Spring Starter Project Dependencies] 창에서 Gradle Build의 Dependency를 설정한다.
2. Gradle Build의 Dependency란 앞으로 Spring Boot를 사용하며 활용할 SQL, I/O, Web, Social 등 다양한 기능을 사전에 설정해 줄 수 있다.
3. Dependency를 이 단계에서 설정해 주지 않더라도 향후에 프로그래밍하며 추가해 줄 수 있다.

The screenshot shows the 'New Spring Starter Project Dependencies' window. At the top, 'Spring Boot Version:' is set to '2.2.5'. Below this, the 'Available:' section contains a search bar with the placeholder text 'Type to search dependencies'. A list of dependency categories is shown below the search bar, each preceded by a right-pointing triangle icon. The categories are: Alibaba, Amazon Web Services, Developer Tools, Google Cloud Platform, I/O, Messaging, Microsoft Azure, NoSQL, Ops, Pivotal Cloud Foundry, SQL, Security, Spring Cloud, Spring Cloud Circuit Breaker, Spring Cloud Config, Spring Cloud Discovery, Spring Cloud Messaging, Spring Cloud Routing, Spring Cloud Security, and Spring Cloud Tools. To the right of the 'Available:' section is the 'Selected:' section, which is currently empty. At the bottom right of the 'Available:' section are two buttons: 'Make Default' and 'Clear Selection'. At the bottom of the window are four buttons: a help icon (question mark), '< Back', 'Next >', and 'Finish' (which is highlighted with a blue border), and 'Cancel'.

Spring Boot 프로젝트 생성하기

Spring Boot Version: 2.2.5

Available:

Type to search dependencies

- ▶ Alibaba
- ▶ Amazon Web Services
- ▼ Developer Tools
 - ☒ Spring Boot DevTools
 - ☐ Lombok
 - ☐ Spring Configuration Processor
- ▶ Google Cloud Platform
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL
- ▶ Ops
- ▶ Pivotal Cloud Foundry
- ▶ SQL
- ▶ Security
- ▶ Spring Cloud
- ▶ Spring Cloud Circuit Breaker
- ▶ Spring Cloud Config
- ▶ Spring Cloud Discovery
- ▶ Spring Cloud Messaging
- ▶ Spring Cloud Routing
- ▶ Spring Cloud Security
- ▶ Spring Cloud Tools
- ▶ Spring Cloud Tracing
- ▶ Template Engines
- ▶ Testing
- ▼ Web
 - ☒ Spring Web

Selected:

- X Spring Boot DevTools
- X Spring Web

스프링부트 버전 및 의존성 설정
웹 그룹에 spring web과
devtools선택

Site Info

Base Url

<https://start.spring.io/starter.zip>

Full Url

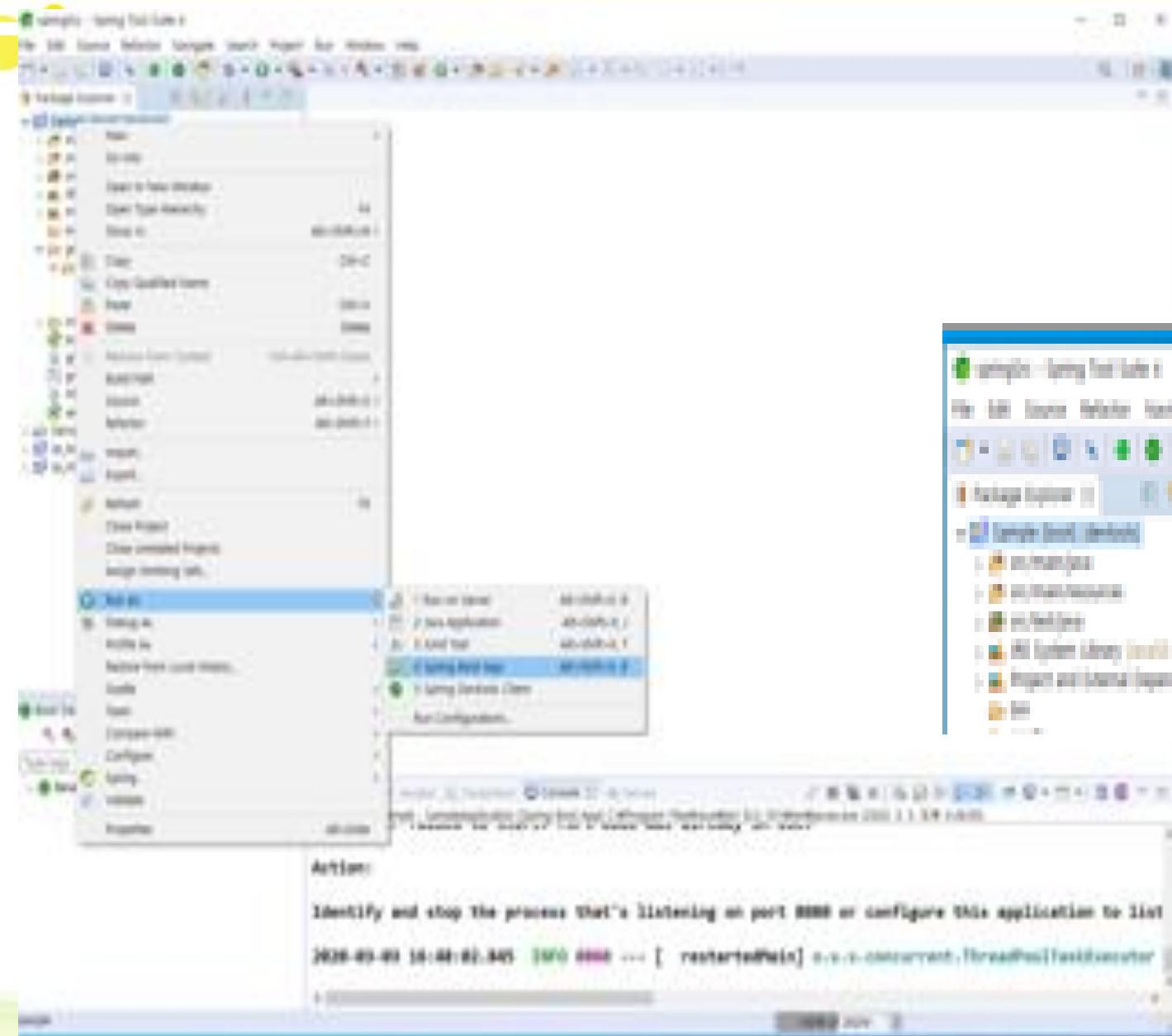
[https://start.spring.io/starter.zip?
name=Sample&groupId=com.example&artifactId=Sample&version=
0.0.1-SNAPSHOT&description=Demo+project+for+Spring
+Boot&packageName=com.example.demo&type=gradle-
project&packaging=jar&javaVersion=1.8&language=java&bootVersio](https://start.spring.io/starter.zip?name=Sample&groupId=com.example&artifactId=Sample&version=0.0.1-SNAPSHOT&description=Demo+project+for+Spring+Boot&packageName=com.example.demo&type=gradle-project&packaging=jar&javaVersion=1.8&language=java&bootVersion=2.2.5)

Make Default

Clear Selection

Spring Boot 프로젝트 실행

프로젝트 우클릭 –
Run As –
Spring Boot App
두번째는 바로 실행



스프링부트는 톰캣을
내장하고 있기 때문에
설치할 필요 없음

Tomcat 충돌 방지위해서 Oracle port 변경

시작+R 을 눌러 실행창에서 cmd 입력
>sqlplus / as sysdba
입력 후 암호 잠겨 있으면 암호입력
SQL> 로 바뀌면

SQL>select dbms_xdb.gethttpport() from dual;

위 명령은 현재 오라클 포트번호 확인

DBMS_XDB.GETHTTPPORT()

8080

SQL>exec dbms_xdb.sethttpport(9090);

오라클 포트번호를 9090으로 변경하겠다는 명령 (원하는 포트번호 입력)

SQL>select dbms_xdb.gethttpport() from dual;

다시 확인해보면 바뀐 포트번호를 확인할 수 있음

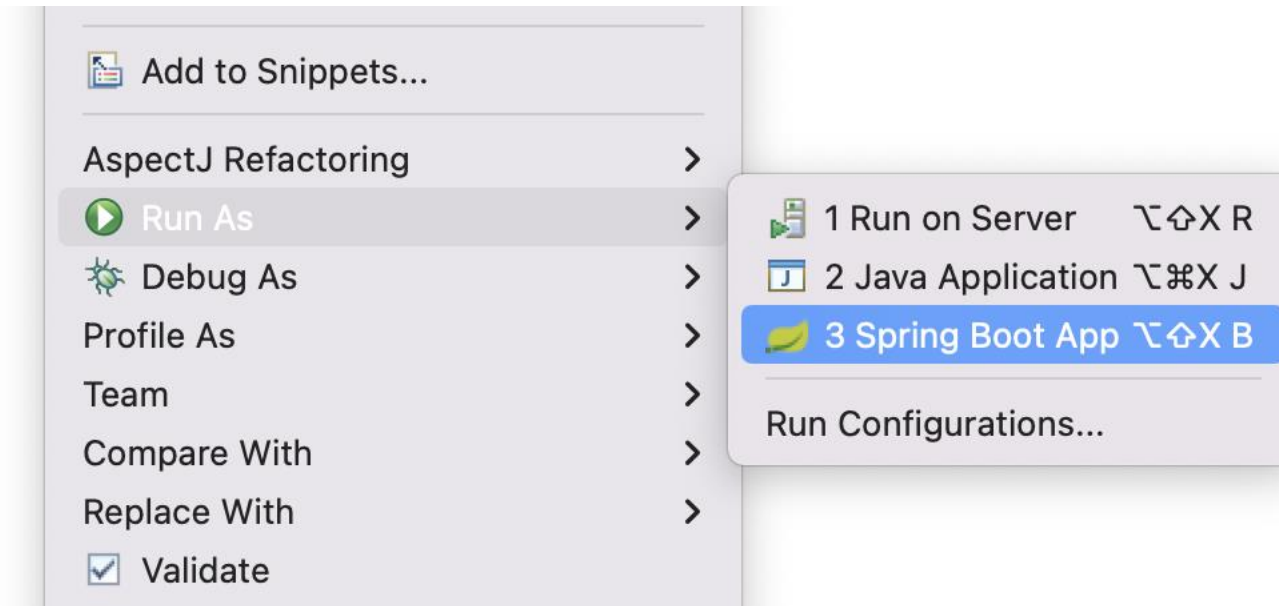
DBMS_XDB.GETHTTPPORT()

9090

프로젝트 생성 및 실행

❖ STS


- ✓ SpringBootApplication.java 파일을 선택한 후 마우스 오른쪽을 클릭 한 후 실행



프로젝트 생성 및 실행

❖ STS

✓ 마우스 오른쪽쪽을 클릭 한 후 실행



```
sts_springboot - StsSpringbootApplication [Spring Boot App] /Library/Java/JavaVirtualMachines/jdk-11.0.10.jdk/Contents/Home/bin/java (2022. 1. 4. 오전 8:51:11)

:: Spring Boot :: (v2.6.2)

2022-01-04 08:51:14.069 INFO 963 --- [main] k.c.a.s.StsSpringbootApplication : Starting StsSpringbootApplication using Java 11.0.10 on ADAMui-MacBookPro
2022-01-04 08:51:14.071 INFO 963 --- [main] k.c.a.s.StsSpringbootApplication : No active profile set, falling back to default profiles: default
2022-01-04 08:51:14.875 INFO 963 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-01-04 08:51:14.886 INFO 963 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-01-04 08:51:14.886 INFO 963 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.56]
2022-01-04 08:51:14.950 INFO 963 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-01-04 08:51:14.950 INFO 963 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 841 ms
2022-01-04 08:51:15.240 INFO 963 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-01-04 08:51:15.251 INFO 963 --- [main] k.c.a.s.StsSpringbootApplication : Started StsSpringbootApplication in 1.511 seconds (JVM running for 2.012s)
2022-01-04 08:51:26.189 INFO 963 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-01-04 08:51:26.189 INFO 963 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-01-04 08:51:26.190 INFO 963 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
```

프로젝트 생성 및 실행

❖ STS

- ✓ 브라우저를 실행시켜 <http://localhost:8080> 을 입력



브라우저에서 확인

← → ↻ ⓘ localhost:8080

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Mar 03 17:10:04 KST 2020

There was an unexpected error (type=Not Found, status=404).

No message available

스프링 부트로 프로젝트를 생성하면 프로젝트의 실행에 대한 기능이 자동으로 설정됨
화면에 보이는 부분이 자동으로 설정되지 않기 때문에 에러 화면 출력

Hello World작성

Class 작성

```
package com.example.demo.controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
@RestController  
public class HelloController {  
    // @GetMapping("/")  
    @RequestMapping("/")  
    public String hello() {  
        return "Hello World";  
    }  
}
```

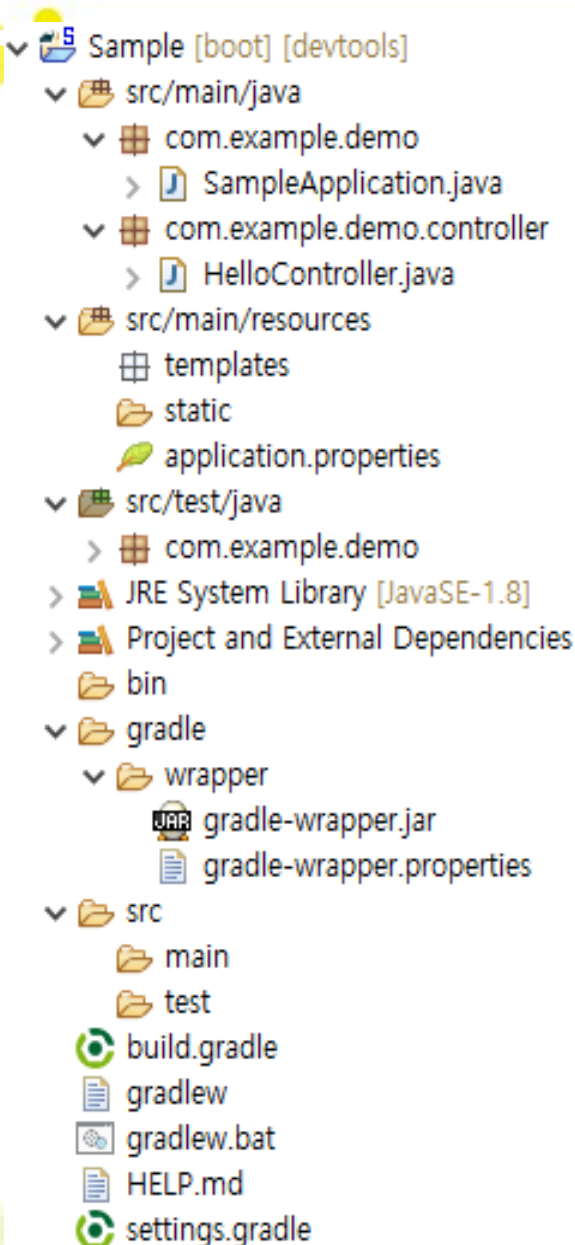
다시 웹을 실행하면



localhost:8080

Hello World

스프링 프로젝트



src/main/java : java source directory

SampleApplication : 애플리케이션을 시작할 수 있는 main메소드가 존재하는 스프링 구성 메인 클래스

templates : 스프링부트에서 사용 가능한 여러 뷰 템플릿 파일 유지

static : 스타일 시트, 자바스크립트, 이미지 등 정적 리소스

application, properties : 애플리케이션 및 스프링의 설정 등에서 사용할 여러 프로퍼티 정의

Project and External Dependencies : 그레이스들에 명시한 프로젝트의 필수 라이브러리 모음

src : JSP 등 리소스 디렉토리

build.gradle : 그레이들 빌드 명세, 프로젝트에 필요한 라이브러리 관리, 빌드 및 배포 설정

SampleApplication

```
package com.example.demo;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

@SpringBootApplication

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

@SpringBootApplication

- 1) **@EnableAutoConfiguration** : 설정 자동 완료
- 2) **@ComponentScan** : 자동으로 여러 컴퍼넌트 검색하고 등록
- 3) **@Configuration** : 자바기반 설정 파일

main 메소드는 스프링 부트의 `Application.run()` 메소드를 사용하여 스프링 부트 애플리케이션을 실행할 수 있게 함

@SpringBootApplication

@EnableAutoConfiguration 어노테이션에는 2가지 속성이 있다. exclude와 excludeName이라는 속성이다. 이것이 실제로 Spring boot의 핵심적인 어노테이션이며 자동 설정을 담당한다. 만약 자동 설정을 하고 싶지 않는 클래스가 있다면 해당 속성을 이용하여 자동설정에서 제외 시킬 수 있다.

@EnableAutoConfiguration 어노테이션의 일부분이다. exclude는 클래스 형태이며 excludeName은 String 형태이다. 만약 JacksonAutoConfiguration 클래스의 자동 설정을 제외 시키고 싶다면 다음과 같이 하면 된다.

@EnableAutoConfiguration(exclude = JacksonAutoConfiguration.class)
위 의 방법은 exclude를 이용해서 JacksonAutoConfiguration 설정을 제외 시킨 방법이다.

@EnableAutoConfiguration(excludeName =
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConfiguration")
다음은 excludeName을 이용해서 JacksonAutoConfiguration 설정을 제외 시킨 방법이다.

bundle.gradle

```
plugins {  
    id 'org.springframework.boot' version '2.2.5.RELEASE' 버전  
    id 'io.spring.dependency-management' version '1.0.9.RELEASE'  
    id 'java'  
}  
group = 'com.example'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '1.8' 자바버전  
configurations {  
    developmentOnly  
    runtimeClasspath {  
        extendsFrom developmentOnly  
    }  
}  
repositories {  
    mavenCentral()  
}  
dependencies { 스프링 부트 플러그인 적용  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    testImplementation('org.springframework.boot:spring-boot-starter-test') {  
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'  
    }  
}
```

Lombok download

이클립스에서 VO(Value Object) 를 생성하고, getter, setter, toString 을 추가하기 위해서는 메뉴에서 source -> Generate Getters and Setters... 와 source -> toString()... 를 선택하여 간단히 생성을 할 수 있습니다.

생성은 간단하게 할 수 있지만 코드가 상당히 길어 보기가 좋지 않고, 나중에 멤버가 추가/삭제 되면 다시 생성을 하여야 하는 불편함이 있습니다.

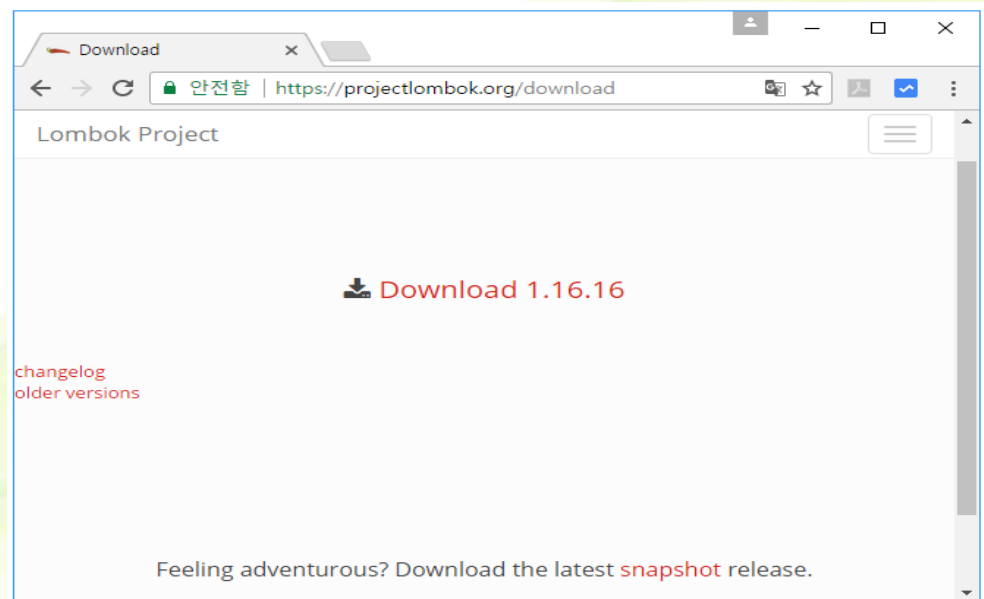
lombok 라이브러리는 아노테이션을 통해서 자동으로 생성이 가능합니다. 실제로 Getter와 Setter 코드는 보이지 않아서 코드가 길지 않아 보기 좋고, 멤버가 추가/삭제 되더라도 생성과 제거가 자동으로 처리됩니다.

lombok 라이브러리는 따로 사용할 수도 있지만 이클립스와 같은 IDE 에서 사용하는 것이 제일 편리한것 같습니다. 이클립스에 설치를 해보겠습니다.

1. 다음 사이트에서 다운로드 합니다.

<https://projectlombok.org/download>

다운로드한 파일은 lombok.jar 입니다.



Lombok실행

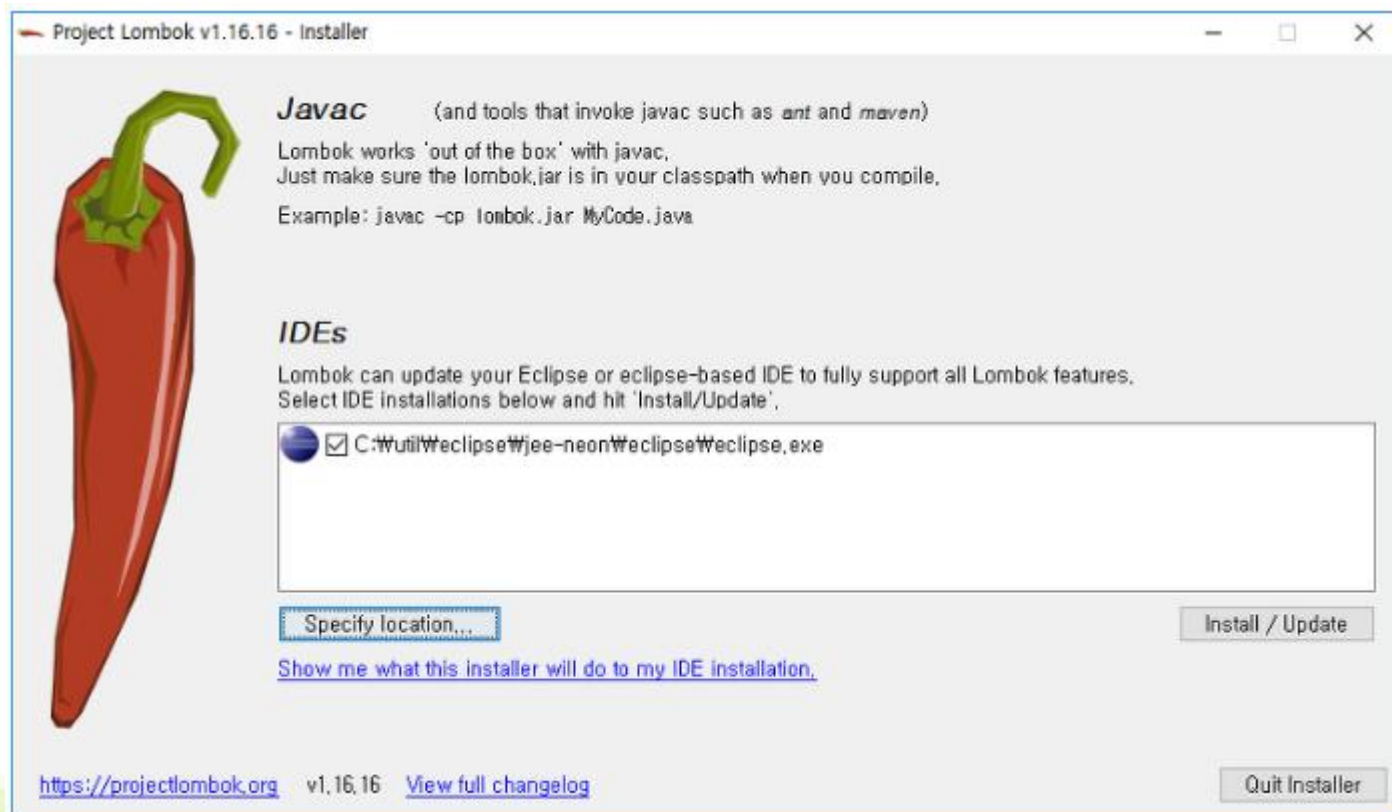
2. lombok.jar 파일을 더블 클릭하여 실행합니다.

또는 java -jar 폴더/Lombok.jar로 실행

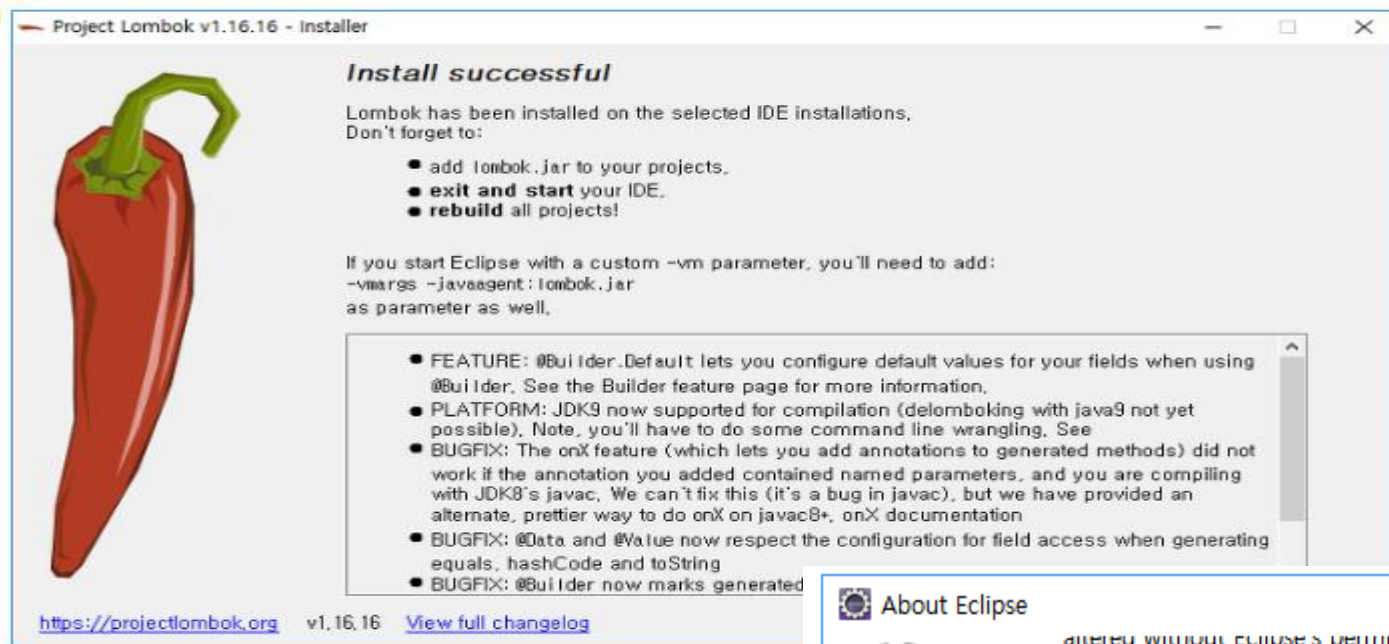
실행이 되면 자동으로 설치된 IDE 를 찾습니다. 자동으로 찾지 못할 경우에는 "Specify location..." 버튼을 눌러 직접 IDE가 설치된 폴더를 지정하면 됩니다.

eclipse.exe 가 있는 폴더를 지정하면 됩니다

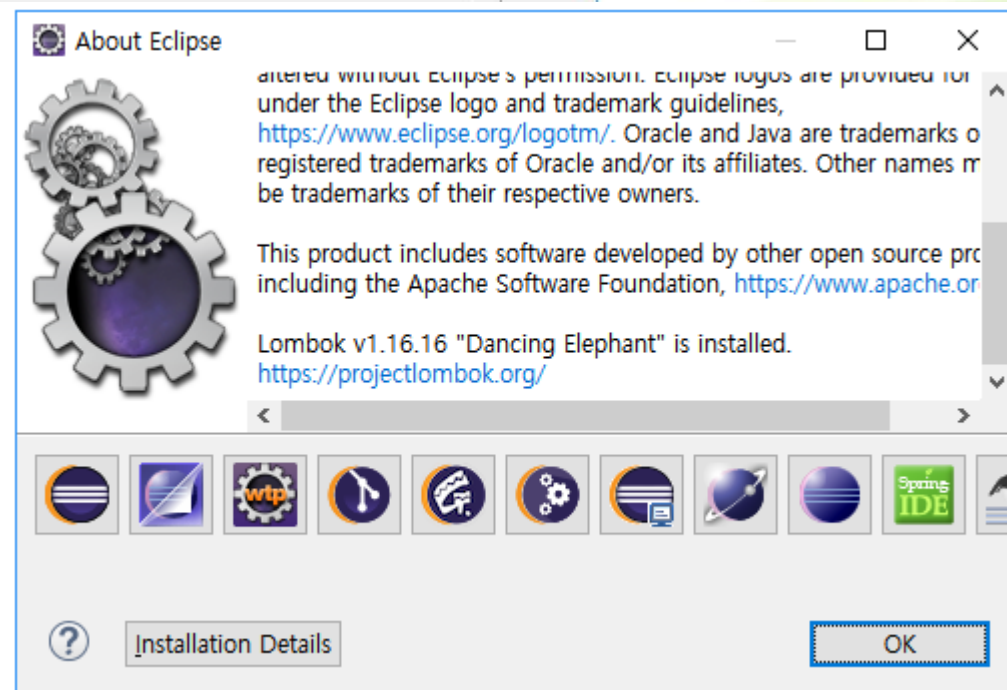
Install/Update버튼을 클릭하여 롬복설치



Lombok 완료 화면



설치완료 화면



lombok.jar 라이브러리 추가

설치만으로 lombok를 사용할 수 있는 것이 아닙니다. 사용하고자 하는 프로젝트에 lombok.jar 를 라이브러리로 등록해야 합니다. 일반 프로젝트라면 build path 에 추가를 하고, web 프로젝트라면 /WEB-INF/lib 폴더에 복사를 합니다. Maven 프로젝트라면 pom.xml 파일에 의존성을 추가를 하면 됩니다.

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.12</version>
</dependency>
```

@Data 아노테이션은 Getter, Setter, hashCode, equals, toString 메소드를 모두 자동으로 추가합니다

```
public @Data class ChagyebuV0 {
    private I
    private S
    private S
    private I
    private I
    private I
    private S
    private S
    private S
    private I
    private I
    private I
}
```

Press 'Ctrl+Space' to show Template Proposals

추가된 메소드 확인

outline 뷰를 확인해보면 추가된 메소드들을 확인할 수 있습니다.

The image shows a screenshot of an IDE with two panels. The left panel displays the source code of `ChagyebuVO.java`, and the right panel displays the Outline view.

Source Code (ChagyebuVO.java):

```
1 package com.tistory.pentode.vo;
2
3 import lombok.Data;
4
5 public @Data class ChagyebuVO {
6
7     private Integer num;
8     private String date;
9     private String type;
10    private Integer amount;
11    private Integer mileage;
12    private Integer price;
13    private String repair;
14    private String complete;
15    private String note;
16
17    private Integer page;
18    private Integer first;
19    private Integer last;
20
21 }
22
```

Outline View:

- com.tistory.pentode.vo
 - ChagyebuVO
 - getNum() : Integer
 - getDate() : String
 - getType() : String
 - getAmount() : Integer
 - getMileage() : Integer
 - getPrice() : Integer
 - getRepair() : String
 - getComplete() : String
 - getNote() : String
 - getPage() : Integer
 - getFirst() : Integer
 - getLast() : Integer
 - setNum(Integer) : void
 - setDate(String) : void
 - setType(String) : void
 - setAmount(Integer) : void
 - setMileage(Integer) : void
 - setPrice(Integer) : void
 - setRepair(String) : void
 - setComplete(String) : void
 - setNote(String) : void
 - setPage(Integer) : void
 - setFirst(Integer) : void
 - setLast(Integer) : void
 - equals(Object) : boolean
 - canEqual(Object) : boolean
 - hashCode() : int

MVC DB연동(mysql Board)

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:

Spring Boot Version:

Frequently Used:

<input checked="" type="checkbox"/> Lombok	<input checked="" type="checkbox"/> MyBatis Framework	<input checked="" type="checkbox"/> MySQL Driver
<input type="checkbox"/> Oracle Driver	<input checked="" type="checkbox"/> Spring Boot DevTools	<input checked="" type="checkbox"/> Spring Configuration Processor
<input checked="" type="checkbox"/> Spring Data JPA	<input checked="" type="checkbox"/> Spring Web	<input checked="" type="checkbox"/> Thymeleaf

Available:

Type to search dependencies

- ▶ Alibaba
- ▶ Amazon Web Services
- ▶ Developer Tools
- ▶ Google Cloud Platform
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL
- ▶ Observability
- ▶ Ops
- ▶ Pivotal Cloud Foundry
- ▶ SQL
- ▶ Security
- ▶ Spring Cloud
- ▶ Spring Cloud Circuit Breaker

Selected:

- X Spring Boot DevTools
- X Lombok
- X Spring Configuration Processor
- X Spring Data JPA
- X MyBatis Framework
- X MySQL Driver
- X Thymeleaf
- X Spring Web

build.gradle(체크한 라이브러리 추가)

```
plugins {  
    id 'org.springframework.boot' version '2.2.5.RELEASE'  
    id 'io.spring.dependency-management' version '1.0.9.RELEASE'  
    id 'java'  
}  
group = 'com.board'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '1.8'  
configurations {  
    developmentOnly  
    runtimeClasspath {  
        extendsFrom developmentOnly  
    }  
    compileOnly {  
        extendsFrom annotationProcessor  
    }  
}  
repositories {  
    mavenCentral()  
}
```


build.gradle(체크한 라이브러리 추가)

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.1.2'  
    compileOnly 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    runtimeOnly 'mysql:mysql-connector-java'  
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-  
processor'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation('org.springframework.boot:spring-boot-starter-test') {  
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'  
    }  
}  
test {  
    useJUnitPlatform()  
}
```


DB연결(resources/applocation.properties)

```
spring.datasource.hikari.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.jdbc-
url=jdbc:mysql://localhost:3306/test?useSSL=true&serverTimezone=UTC
spring.datasource.hikari.username=root
spring.datasource.hikari.password=mysql
```

```
mybatis.configuration.map-underscore-to-camel-case=true
// jsp, js, css변경할 때 바로적용
spring.devtools.livereload.enabled=true
spring.thymeleaf.cache=false
spring.devtools.restart.enabled=false
```

Hikari CP

스프링부트 2.01부터 톰캣에서 hikari로 변경

camelCase : 단어와 단어의 합성어로 이루어진 변수의 경우 합성되는 단어의 첫 부분을 대문자로 표기하고 나머지는 소문자로 표기하는 방식입니다. 이 모양이 흡사 낙타의 혹과 같다고 해서 지어진 이름이지요.

게시판

- board [boot] [devtools]
 - > Spring Elements
 - src/main/java
 - board
 - > BoardApplication.java
 - board.board.controller
 - > BoardController.java
 - board.board.dto
 - > BoardDto.java
 - board.board.mapper
 - > BoardMapper.java
 - board.board.service
 - > BoardService.java
 - > BoardServiceImpl.java
 - board.configuration
 - > DatabaseConfiguration.java
 - src/main/resources
 - mapper
 - sql-board.xml
 - templates.board
 - boardDetail.html
 - boardList.html
 - boardWrite.html
 - static
 - css
 - style.css
 - application.properties

- src/test/java
- JRE System Library [JavaSE-1.8]
- Project and External Dependencies
- bin
- gradle
 - > wrapper
- src
- build.gradle
- gradlew
- gradlew.bat
- HELP.md
- settings.gradle

게시판(mysql)

Table 생성

```
create table t_board (  
    board_idx int not null primary key auto_increment ,  
    title varchar(300) not null ,  
    contents text not null ,  
    hit_cnt int not null default 0 ,  
    creator_id varchar(50) not null ,  
    created_date date not null ,  
    deleted_yn char(1) not null default 'n'  
);
```

BoardApplication

```
package board;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
@SpringBootApplication  
public class BoardApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(BoardApplication.class, args);  
    }  
}
```

DatabaseConfiguration

package board.configuration; Spring Boot는 Java-based configuration을 선호한다
import javax.sql.DataSource;
import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
@Configuration
@PropertySource("classpath:/application.properties")
public class DatabaseConfiguration {
 @Autowired
 private ApplicationContext applicationContext;
 @Bean
 @ConfigurationProperties(prefix="spring.datasource.hikari")
 public HikariConfig hikariConfig() {
 return new HikariConfig();
 }
}

DatabaseConfiguration

@Bean

```
@ConfigurationProperties(prefix="mybatis.configuration")
public org.apache.ibatis.session.Configuration mybatisConfig(){
    return new org.apache.ibatis.session.Configuration();
}
```

@Bean

```
public DataSource dataSource() throws Exception{
    DataSource dataSource = new HikariDataSource(hikariConfig());
    return dataSource;
}
```

@Bean

```
public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception{
    SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();
    sqlSessionFactoryBean.setDataSource(dataSource);
    sqlSessionFactoryBean.setMapperLocations(applicationContext.getResources("class
path:/mapper/**/*.xml"));
    sqlSessionFactoryBean.setConfiguration(mybatisConfig());
    return sqlSessionFactoryBean.getObject();
}
```

@Bean

```
public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory){
    return new SqlSessionTemplate(sqlSessionFactory);
}
```

```
}
```



BoardDto

```
package com.ch.mysql.dto;

import java.sql.Date;
import lombok.Data;

@Data

public class BoardDto {

    private int boardIdx;
    private String title;
    private String contents;
    private int hitCnt;
    private String creatorId;
    private Date createdAt;
    private String deletedYn;

}
```


BoardMapper

```
package com.ch.mysql.mapper;  
import java.util.List;  
import org.apache.ibatis.annotations.Mapper;  
import com.ch.mysql.dto.BoardDto;  
@Mapper  
public interface BoardMapper {  
    List<BoardDto> selectBoardList();  
    int insert(BoardDto board);  
    BoardDto select(int board_idx);  
}
```

BoardService/ BoardServiceImpl

```
package com.ch.mysql.service;
import java.util.List;
import com.ch.mysql.dto.BoardDto;
public interface BoardService {
    List<BoardDto> selectBoardList();
    int insert(BoardDto board);
    BoardDto select(int board_idx);
}
```

BoardServiceImpl

```
package com.ch.mysql.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.ch.mysql.dto.BoardDto;
import com.ch.mysql.mapper.BoardMapper;
@Service
public class BoardServiceImpl implements BoardService {
    @Autowired
    private BoardMapper boardMapper;
    public List<BoardDto> selectBoardList() {
        return boardMapper.selectBoardList();
    }
    public int insert(BoardDto board) {
        return boardMapper.insert(board);
    }
    public BoardDto select(int board_idx) {
        return boardMapper.select(board_idx);
    }
}
```

BoardController

```
package com.ch.mysql.controller;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import com.ch.mysql.dto.BoardDto;
import com.ch.mysql.service.BoardService;
@Controller
public class BoardController {
    @Autowired
    private BoardService bs;
    @RequestMapping("/board/boardList.do")
    public String boardList(Model model) {
        List<BoardDto> list = bs.selectBoardList();
        model.addAttribute("list", list);
        return "/board/boardList";
    }
}
```

BoardController

```
@RequestMapping("/board/boardWriteForm.do")
public String insert() {
    return "/board/boardWriteForm";
}
@RequestMapping("/board/boardWrite.do")
public String insertBoard(BoardDto board, Model model) {
    int result = bs.insert(board);
    model.addAttribute("result", result);
    return "/board/insertBoard";
}
@RequestMapping("/board/boardDetail.do")
public String boardDetail(int boardIdx, Model model) {
    BoardDto board = bs.select(boardIdx);
    model.addAttribute("board", board);
    return "/board/boardDetail";
}
}
```

sql-board.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.ch.mysql.mapper.BoardMapper">
  <select id="selectBoardList" resultType="com.ch.mysql.dto.BoardDto">
    select * from t_board
      where deleted_yn='n' order by board_idx desc
  </select>
  <insert id="insert">
    insert into t_board (title, contents, created_date,
      creator_id) values (#{title},#{contents},now(),'admin')
  </insert>
  <select id="select" parameterType="integer"
resultType="com.ch.mysql.dto.BoardDto">
    select * from t_board where board_idx=#{board_idx}
      and deleted_yn='n'
  </select>
</mapper>
```

boardList.html

```
<!DOCTYPE html>
<!-- thymeleaf : jstl와 유사한 역할 -->
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link rel="stylesheet" th:href="@{/css/style.css}">
</head><body>
<div class="container">
    <h2>게시글 목록</h2>
<table class="board_list">
    <tr><th>글번호</th><th>제목</th><th>조회수</th>
        <th>작성일</th></tr>
    <tr th:if="${#lists.size(list)} > 0" th:each="board:${list}">
        <td th:text="${board.boardIdx}"></td>
        <td class="title"><a href="/board/boardDetail.do?boardIdx="
            th:attrappend="href=${board.boardIdx}"
            th:text="${board.title}"></a></td>
        <td th:text="${board.hitCnt}"></td>
        <td th:text="${board.createdDate}"></td></tr>
    <tr th:unless="${#lists.size(list)} > 0">
        <td colspan="4">데이터가 없습니다</td></tr>
</table>
<a href="/board/boardWriteForm.do" class="btn">글쓰기</a>
</div>
</body></html>
```


boardDetail.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link rel="stylesheet" th:href="@{/css/style.css}">
</head><body>
<div class="container">
<table class="board_detail">
  <caption>게시글 상세</caption>
  <tr><td>글번호</td><td th:text="${board.boardIdx}"></td>
    <td>작성자</td><td th:text="${board.creatorId}"></td></tr>
  <tr><td>조회수</td><td th:text="${board.hitCnt}"></td>
    <td>작성일</td><td th:text="${board.createdDate}"></td></tr>
  <tr><td>제목</td><td colspan="3" th:text="${board.title}"></td></tr>
  <tr><td>내용</td><td colspan="3" th:text="${board.contents}"></td></tr>
</table>
</div>
</body>
</html>
```

boardWriteForm.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link rel="stylesheet" th:href="@{/css/style.css}">
</head><body>
<div class="container">
<h2>게시판 작성</h2>
<form action="/boardboardWrite.do" method="post">
<table class="board_detail">
    <tr><td>제 목</td><td><input type="text" name="title"
        required="required" autofocus="autofocus"></td></tr>
    <tr><td>내 용</td><td><textarea rows="5" cols="40"
        required="required" name="contents"></textarea></td></tr>
    <tr><th colspan="2"><input type="submit"></th></tr>
</table>
</form>
</div>
</body>
</html>
```

boardWrite.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
</head><body>
<span th:if="${result} > 0">
    <script type="text/javascript">
        alert("입력 되었습니다");
        location.href="/board/boardList.do";
    </script>
</span>
<span th:if="${result} == 0">
    <script type="text/javascript">
        alert("입력 실패");
        history.go(-1);
    </script>
</span>
</body>
</html>
```

Alias사용

```
import org.apache.ibatis.type.Alias;
```

```
import lombok.Data;
```

```
@Data
```

```
@Alias("boardDto")
```

```
public class BoardDto {
```

```
    private int boardIdx;
```

```
    private String title;
```

```
<select id="selectBoardList" resultType="boardDto">
```

```
    select * from t_board where deleted_yn='n' order by board_idx desc
```

```
</select>
```

```
public class DatabaseConfiguration {
```

```
    @Bean
```

```
    public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {  
        SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();  
        sqlSessionFactoryBean.setDataSource(dataSource);
```

```
        sqlSessionFactoryBean.setMapperLocations(applicationContext.getResources("classpath:/mapper/*  
*/sql-*.xml"));
```

```
        sqlSessionFactoryBean.setConfiguration(mybatisConfig());
```

```
        sqlSessionFactoryBean.setTypeAliasesPackage("com.ch.mysql.dto");
```

```
        return sqlSessionFactoryBean.getObject();
```

```
    }
```

Stringboot에서 jsp사용

1. sts에 jsp사용

help - Eclipse marketplace를 통해 설치

Eclipse Web Developer Tools

Eclipse Enterprise Java and Web Developer Tools

2. build.gradle에 추가

```
dependencies {  
    implementation 'javax.servlet:jstl'  
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'  
}
```

implementation 'org.springframework.boot:spring-boot-starter-thymeleaf' 제거

3. application.properties에 추가

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

. webapp, resources, WEB-INF폴더를 똑같이 만든다

web.xml, classes, spring폴더 제거

폴더 생성

Spring의 webapp형식 생성

- src/main/java
 - com.ch.spring1
 - com.ch.spring1.configuration
 - com.ch.spring1.controller
 - com.ch.spring1.mapper
 - com.ch.spring1.model
 - com.ch.spring1.service
- src/main/resources
- src/test/java
- JRE System Library [JavaSE-17]
- Project and External Dependencies
- bin
- gradle
- src
 - main
 - webapp
 - resources
 - bootstrap
 - WEB-INF
 - views
 - dept
 - emp
 - header.jsp
 - test
- build.gradle
- gradlew
- gradlew.bat

application.properties

```
spring.datasource.hikari.driver-class-  
name=oracle.jdbc.OracleDriver  
spring.datasource.hikari.jdbc-  
url=jdbc:oracle:thin:@127.0.0.1:1521:xe  
spring.datasource.hikari.username=scott  
spring.datasource.hikari.password=tiger  
mybatis.configuration.map-underscore-to-camel-case=true
```

```
spring.mvc.view.prefix=/WEB-INF/views/  
spring.mvc.view.suffix=.jsp  
spring.devtools.livereload.enabled=true  
spring.thymeleaf.cache=false  
spring.devtools.restart.enabled=false  
spring.jpa.open-in-view=false
```


build.gradle

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'  
  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-  
starter:2.2.2'  
  
    implementation 'javax.servlet:jstl'  
        implementation "org.apache.tomcat.embed:tomcat-embed-jasper"  
  
    compileOnly 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    runtimeOnly 'com.oracle.database.jdbc:ojdbc8'  
    annotationProcessor 'org.springframework.boot:spring-boot-  
configuration-processor'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
}
```

DataBaseConfiguration

```
package com.ch.spring1.configuration;
import javax.sql.DataSource;
import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

@Configuration
@PropertySource("classpath:/application.properties")
public class DataBaseConfiguration {
    @Autowired
    private ApplicationContext applicationContext;
    @Bean
    @ConfigurationProperties(prefix = "spring.datasource.hikari")
    public HikariConfig hikariConfig() {
        return new HikariConfig();
    }
}
```

DataBaseConfiguration

@Bean

@ConfigurationProperties(prefix = "mybatis.configuration")

```
public org.apache.ibatis.session.Configuration mybatisConfig() {  
    return new org.apache.ibatis.session.Configuration();  
}
```

@Bean

```
public DataSource dataSource() {  
    DataSource dataSource = new HikariDataSource(hikariConfig());  
    return dataSource;  
}
```

@Bean

```
public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {  
    SqlSessionFactoryBean sqlSessionFactoryBean =  
        new SqlSessionFactoryBean();  
    sqlSessionFactoryBean.setDataSource(dataSource);  
    sqlSessionFactoryBean.setMapperLocations(  
        applicationContext.getResources("classpath:/mapper/**/*.xml"));  
    sqlSessionFactoryBean.setConfiguration(mybatisConfig());  
    sqlSessionFactoryBean.setTypeAliasesPackage("com.ch.spring1.model");  
    return sqlSessionFactoryBean.getObject();  
}
```

@Bean

```
public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory) {  
    return new SqlSessionTemplate(sqlSessionFactory);  
}
```

}

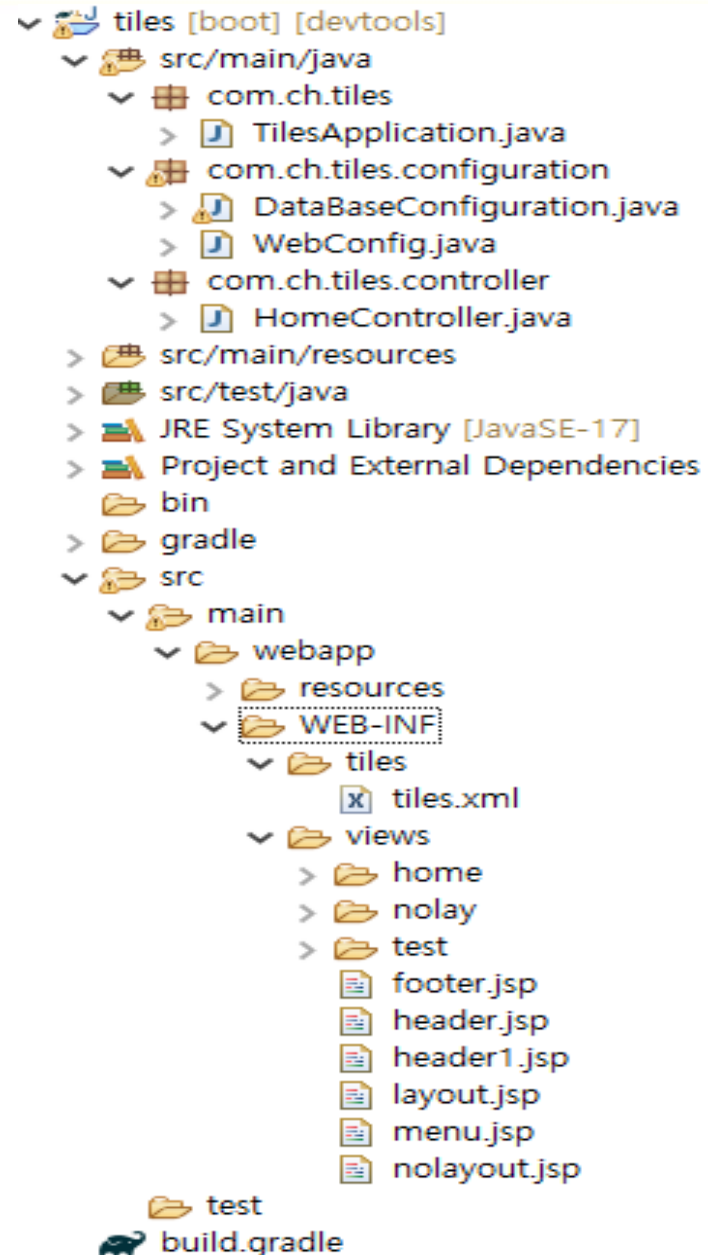
Spring boot에서 tiles사용

타일즈란 화면의 레이아웃을 구성해주는 라이브러리이다.

- 화면을 구성하면서 헤더, 푸터, 레프트 메뉴 등이 동일하게 구성되는 경우 레이아웃 설정을 통해서 반복되는 코딩을 최소화 할 수 있다.

적용순서

타일즈 라이브러리 추가(gradle) > 스프링부트 타일즈 @Configuration 설정 > tilex.xml 설정 -> 레이아웃 화면 구성



Spring boot에서 tiles사용

- build.gradle 에 타일관련 라이브러리 추가

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.2.2'  
  
    implementation 'javax.servlet:jstl'  
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'  
  
    implementation 'org.apache.tiles:tiles-jsp:3.0.8'  
    implementation 'org.apache.tiles:tiles-core:3.0.8'  
    implementation 'org.apache.tiles:tiles-servlet:3.0.8'  
  
    compileOnly 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    runtimeOnly 'com.oracle.database.jdbc:ojdbc8'  
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-processor'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'}
```

Spring boot에서 tiles사용

```
<!DOCTYPE tiles-definitions PUBLIC
"-//Apache Software Foundation//DTD Tiles Configuration 3.0//EN"
"http://tiles.apache.org/dtds/tiles-config_3_0.dtd">
<tiles-definitions>
<!-- 레이아웃 적용 -->
<definition name="tilesbase" template="/WEB-INF/views/layout.jsp">
<put-attribute name="header" value="/WEB-INF/views/header1.jsp" />
<put-attribute name="menu" value="/WEB-INF/views/menu.jsp" />
<put-attribute name="body" value="" />
<put-attribute name="footer" value="/WEB-INF/views/footer.jsp" />
</definition>
<!-- 레이아웃 적용 안함 -->
<definition name="nolay" template="/WEB-INF/views/nolayout.jsp">
<put-attribute name="header" value="" />
<put-attribute name="menu" value="" />
<put-attribute name="footer" value="" />
</definition>
<!-- (1) nolay폴더 안에 {1}에 jsp이름 -->
<definition name="nolay/*" extends="nolay">
<put-attribute name="body" value="/WEB-INF/views/nolay/{1}.jsp" />
</definition>
<!-- (1) {1}폴더 안에 {2}에 jsp이름 -->
<definition name="*/*" extends="tilesbase">
<put-attribute name="body" value="/WEB-INF/views/{1}/{2}.jsp" />
</definition>
</tiles-definitions>
```

Spring boot에서 tiles사용

```
package com.ch.tiles.configuration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.view.tiles3.TilesConfigurer;
import org.springframework.web.servlet.view.tiles3.TilesView;
import org.springframework.web.servlet.view.tiles3.TilesViewResolver;
@Configuration
public class WebConfig {
    @Bean
    public TilesConfigurer tilesConfigurer() {
        final TilesConfigurer configurer = new TilesConfigurer();
        configurer.setDefinitions(new String[] {
            "/WEB-INF/tiles/tiles.xml"
        });
        configurer.setCheckRefresh(true);
        return configurer;
    }
    @Bean
    public TilesViewResolver tilesViewResolver() {
        final TilesViewResolver tilesViewResolver = new TilesViewResolver();
        tilesViewResolver.setViewClass(TilesView.class);
        tilesViewResolver.setOrder(1);
        return tilesViewResolver;
    }
}
```

Spring boot에서 tiles사용

@Configuration

@PropertySource("classpath:/application.properties")

```
public class DataBaseConfiguration {  
    @Autowired  
    private ApplicationContext applicationContext;  
    @Bean  
    @ConfigurationProperties(prefix="spring.datasource.hikari")  
    public HikariConfig hikariConfig() {  
        return new HikariConfig();  
    }  
    @Bean  
    @ConfigurationProperties(prefix = "mybatis.configuration")  
    public org.apache.ibatis.session.Configuration mybatisConfig() {  
        return new org.apache.ibatis.session.Configuration();  
    }  
    @Bean  
    public DataSource dataSource() {  
        DataSource ds = new HikariDataSource(hikariConfig());  
        return ds;  
    }  
}
```


Spring boot에서 tiles사용

@Bean

public SqlSessionFactory sqlSessionFactory(DataSource ds) throws

Exception {

 SqlSessionFactoryBean ssfb = new SqlSessionFactoryBean();

 ssfb.setDataSource(ds);

 // ssfb.setMapperLocations(

 //

 applicationContext.getResources("classpath:/mapper/sql-*.xml"));

 // ssfb.setConfiguration(mybatisConfig());

 // ssfb.setTypeAliasesPackage("com.ch.spring1.dto");

 return ssfb.getObject();

}

@Bean

public SqlSessionTemplate sst(SqlSessionFactory ssf) {

 return new SqlSessionTemplate(ssf);

}

}

Spring boot에서 tiles사용

```
package com.ch.tiles.controller;import org.springframework.stereotype.Controller;import org.springframework.web.bind.annotation.RequestMapping;@Controllerpublic class HomeController {    @RequestMapping("home/home")    public String home() {        return "home/home";    }    @RequestMapping("home/first")    public String first() {        return "home/first";    }    @RequestMapping("test/second")    public String second() {        return "test/second";    }    @RequestMapping("test/third")    public String third() {        return "test/third";    }    @RequestMapping("nolay/test1")    public String test1() {        return "nolay/test1";    }    }
```