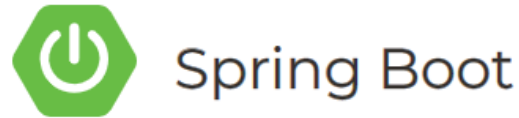


# Spring Boot

# 스프링부트

<https://spring.io/projects/spring-boot>

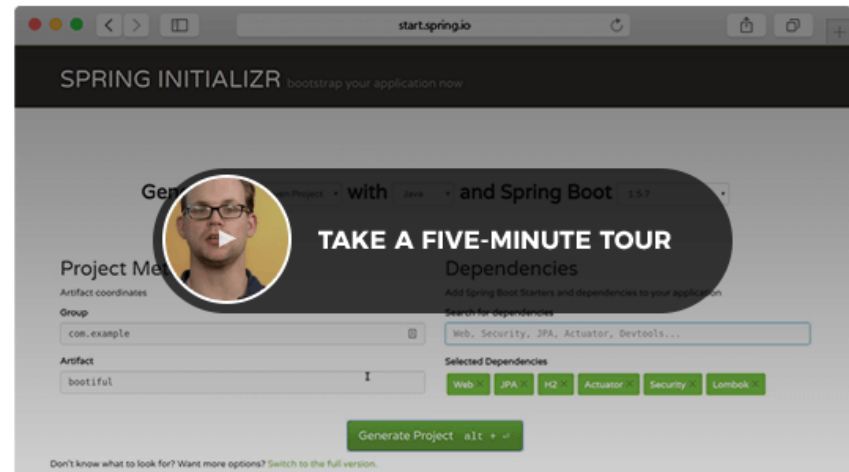
# Spring Boot



## BUILD ANYTHING WITH SPRING BOOT

Spring Boot is the starting point for building all Spring-based applications. Spring Boot 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
- Embedded runtime support: Tomcat, Jetty, and Undertow
- Developer productivity tools such as LiveReload and Auto Restart
- Curated dependencies that just work
- Production-ready features such as tracing, metrics, and health status
- Works in your favorite IDE: Spring Tool Suite, IntelliJ IDEA, and NetBeans



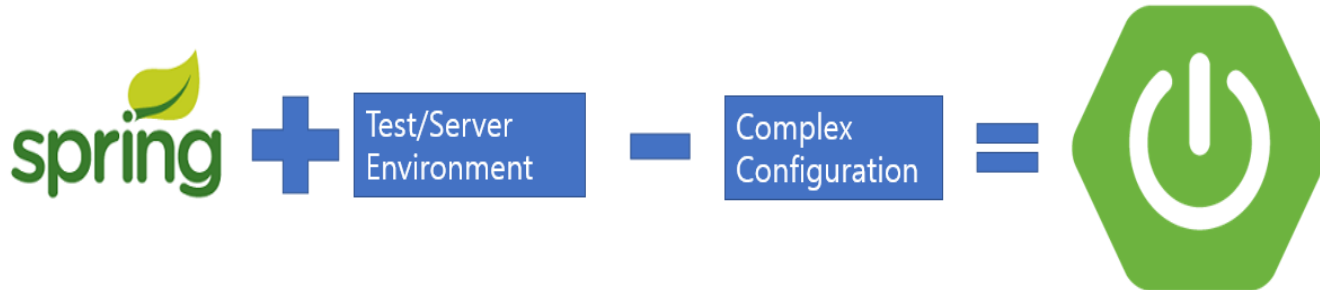
<https://spring.io/>

# Spring Boot

- 단독으로 실행 가능하고 제품 수준의 스프링기반 어플리케이션을 제작하는 것을 목표로 진행된 프로젝트
- 주요기능
  - 단독실행 가능한 수준의 스프링 애플리케이션 개발 가능
  - 내장된 WAS(예. Tomcat)를 이용해서 별도의 서버 설치 없이 실행 가능
  - 최대한 자동화된 설정
  - XML 설정 없이 단순한 설정 방식 제공

# Spring Boot

- Spring Boot가 해결하는 것들
  - 자동화된 라이브러리 관리
  - Spring Boot Auto Configuration
  - 라이브러리의 자동 결정과 XML없는 환경
  - 테스트 환경과 내장 Tomcat



# Spring Boot

- Spring Boot는 표준프레임워크가 제공하는 기본적인 공통기반 영역(Foundation Layer ; Spring Core, Batch Spring Web 등)을 기반 위에 모듈 단위의 배포 및 설정 최소화를 지원하는 실행(execution) 환경의 기반을 제공한다.
  - 스프링부트는 단독 실행되는, 실행하기만 하면 되는 상용화 가능한 수준의 스프링 기반 애플리케이션을 쉽게 만들어 낼 수 있다.
  - 최소한의 설정으로 스프링 플랫폼과 third-party 라이브러리들을 사용할 수 있도록 하고 있다.



# Spring Boot

- 이는 표준프레임워크 활용 측면에서 보다 손 쉽게 설정을 구성하거나 개발 및 배포를 빠르게 지원할 수 있다. 또한, embedded 방식의 container를 사용하여 web server를 통한 배포가 아닌, 독립적으로 실행 가능한 웹 애플리케이션을 구성할 수 있다.
- 제공 기능
  - 단독 실행이 가능한 스프링 애플리케이션을 생성
  - 내장형 Tomcat, Jetty 또는 Undertow를 지원 (WAR파일로 배포 시 불필요)
  - 기본으로 설정되어 있는 'starter' 컴포넌트들을 쉽게 환경 설정 (dependency, build 등)
  - Library 인식을 통한 자동 환경 구성 지원
  - 상용화 수준의 통계(metrics), 상태 점검(health check) 및 외부 설정 제공
  - 설정을 위한 XML 코드 불필요

# Spring Boot

- **Spring Boot Starters**

- 스타터(Starters)는 응용 프로그램에 포함 할 수 있는 편리한 종속성 관리의 집합이다. 샘플 코드와 복사-붙여넣기의 의존성 관리를 거치지 않고도 필요한 모든 Spring 및 관련 기술을 한 번에 관리를 할 수 있다. 예를 들어, 데이터베이스 액세스를 위한 Spring 및 JPA를 사용하려면 “spring-boot-starter-data-jpa” 프로젝트를 종속성에 포함 시켜 사용할 수 있다.
- 스타터에는 프로젝트를 신속하게 시작하고 실행하는데 필요한 많은 종속성이 포함되어 있으며 일관되게 지원 관리되는 종속성 세트를 제공한다.
- 스프링 부트에는 3가지 카테고리로 분류하여 스타터를 제공한다.
  - 스프링 부트 애플리케이션 스타터
  - 스프링 부트 프로덕션 스타터
  - 스프링 부트 테크니컬 스타터
  - <https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#using.build-systems.starters>



# Spring Boot

- Spring Boot Starters : 스프링 부트 애플리케이션 스타터

이름	설명
<b>spring-boot-starter</b>	자동 구성 지원, 로깅 및 YAML을 포함한 핵심 스타터
<b>spring-boot-starter-activemq</b>	Apache ActiveMQ를 사용한 JMS 메시징 스타터
<b>spring-boot-starter-amqp</b>	Spring AMQP 및 Rabbit MQ 사용을 위한 스타터
<b>spring-boot-starter-aop</b>	Spring AOP 및 AspectJ를 이용한 Aspect 지향 프로그래밍 스타터
<b>spring-boot-starter-artemis</b>	Apache Artemis를 사용한 JMS 메시징 스타터
<b>spring-boot-starter-batch</b>	스프링 배치 사용을 위한 스타터
<b>spring-boot-starter-cache</b>	Spring Framework의 캐싱 지원 사용을 위한 스타터
<b>spring-boot-starter-data-cassandra</b>	Cassandra 분산 데이터베이스 및 Spring Data Cassandra 사용을 위한 스타터
<b>spring-boot-starter-data-cassandra-reactive</b>	Cassandra 분산 데이터베이스 및 Spring Data Cassandra Reactive 사용을 위한 스타터
<b>spring-boot-starter-data-couchbase</b>	Couchbase 문서 지향 데이터베이스 및 Spring Data Couchbase 사용을 위한 스타터
<b>spring-boot-starter-data-jdbc</b>	스프링 데이터 JDBC 사용을 위한 스타터
<b>spring-boot-starter-data-jpa</b>	Hibernate와 함께 Spring Data JPA를 사용하기 위한 스타터
<b>spring-boot-starter-data-ldap</b>	스프링 데이터 LDAP 사용을 위한 스타터
<b>spring-boot-starter-data-mongodb</b>	MongoDB 문서 지향 데이터베이스 및 Spring Data MongoDB 사용을 위한 스타터
<b>spring-boot-starter-data-mongodb-reactive</b>	MongoDB 문서 지향 데이터베이스 및 Spring Data MongoDB Reactive 사용을 위한 스타터

# Spring Boot

- Spring Boot Starters : 스프링 부트 애플리케이션 스타터

이름	설명
<b>spring-boot-starter-data-neo4j</b>	Neo4j 그래프 데이터베이스 및 Spring Data Neo4j 사용을 위한 스타터
<b>spring-boot-starter-data-r2dbc</b>	Spring Data R2DBC 사용을 위한 스타터
<b>spring-boot-starter-data-redis</b>	Spring Data Redis 및 Lettuce 클라이언트와 함께 Redis 키-값 데이터 저장소를 사용하기 위한 스타터
<b>spring-boot-starter-data-redis-reactive</b>	Spring Data Redis 반응 형 및 Lettuce 클라이언트와 함께 Redis 키-값 데이터 저장소를 사용하기 위한 스타터
<b>spring-boot-starter-data-rest</b>	Spring Data REST를 사용하여 REST를 통해 Spring Data 저장소를 노출하기 위한 스타터
<b>spring-boot-starter-data-solr</b>	Spring Data Solr과 함께 Apache Solr 검색 플랫폼을 사용하기 위한 스타터
<b>spring-boot-starter-freemarker</b>	FreeMarker보기를 사용하여 MVC 웹 애플리케이션 빌드를 위한 스타터
<b>spring-boot-starter-groovy-templates</b>	Groovy 템플릿 뷰를 사용하여 MVC 웹 애플리케이션 구축을 위한 스타터
<b>spring-boot-starter-hateoas</b>	Spring MVC 및 Spring HATEOAS로 하이퍼 미디어 기반 RESTful 웹 애플리케이션 구축을 위한 스타터
<b>spring-boot-starter-integration</b>	스프링 통합 사용을 위한 스타터
<b>spring-boot-starter-jdbc</b>	DB 연결 풀에서 JDBC를 사용하기 위한 스타터
<b>spring-boot-starter-jersey</b>	JAX-RS 및 Jersey를 사용하여 RESTful 웹 애플리케이션을 빌드하기 위한 스타터. 대안 spring-boot-starter-web

# Spring Boot

- Spring Boot Starters : 스프링 부트 애플리케이션 스타터

이름	설명
spring-boot-starter-jooq	jOOQ를 사용하여 SQL 데이터베이스에 액세스하기 위한 스타터. spring-boot-starter-data-jpa또는에 대한 대안 spring-boot-starter-jdbc
spring-boot-starter-json	JSON을 읽고 쓰는 스타터
spring-boot-starter-jta-atomikos	Atomikos를 사용한 JTA 트랜잭션 스타터
spring-boot-starter-jta-bitronix	Bitronix를 사용한 JTA 트랜잭션 스타터. 2.3.0부터 사용되지 않음
spring-boot-starter-mail	Java Mail 및 Spring Framework의 이메일 전송 지원을 위한 스타터
spring-boot-starter-mustache	콧수염보기를 사용하여 웹 애플리케이션을 빌드하기 위한 스타터
spring-boot-starter-oauth2-client	Spring Security의 OAuth2 / OpenID Connect 클라이언트 기능을 사용하기 위한 스타터
spring-boot-starter-oauth2-resource-server	Spring Security의 OAuth2 리소스 서버 기능을 사용하기 위한 스타터
spring-boot-starter-quartz	Quartz 스케줄러 사용을 위한 스타터
spring-boot-starter-rsocket	RSocket 클라이언트 및 서버 구축을 위한 스타터
spring-boot-starter-security	스프링 시큐리티 사용을 위한 스타터
spring-boot-starter-test	JUnit, Hamcrest 및 Mockito를 포함한 라이브러리로 Spring Boot 애플리케이션을 테스트하기 위한 스타터
spring-boot-starter-thymeleaf	Thymeleaf보기를 사용하여 MVC 웹 애플리케이션 빌드를 위한 스타터

# Spring Boot

- **Spring Boot Starters : 스프링 부트 애플리케이션 스타터**

이름	설명
<b>spring-boot-starter-validation</b>	Hibernate Validator와 함께 Java Bean Validation을 사용하기 위한 스타터
<b>spring-boot-starter-web</b>	Spring MVC를 사용하는 RESTful 애플리케이션을 포함한 웹 구축을 위한 스타터. Tomcat을 기본 내장 컨테이너로 사용
<b>spring-boot-starter-web-services</b>	스프링 웹 서비스 사용을 위한 스타터
<b>spring-boot-starter-webflux</b>	Spring Framework의 Reactive Web 지원을 사용하여 WebFlux 애플리케이션 구축을 위한 스타터
<b>spring-boot-starter-websocket</b>	Spring Framework의 WebSocket 지원을 사용하여 WebSocket 애플리케이션 구축을 위한 스타터

# Spring Boot

- Spring Boot Starters : 스프링 부트 프로덕션 스타터

이름	설명
<b>spring-boot-starter-actuator</b>	애플리케이션을 모니터링하고 관리 할 수 있는 프로덕션 준비 기능을 제공하는 Spring Boot Actuator 사용을 위한 스타터

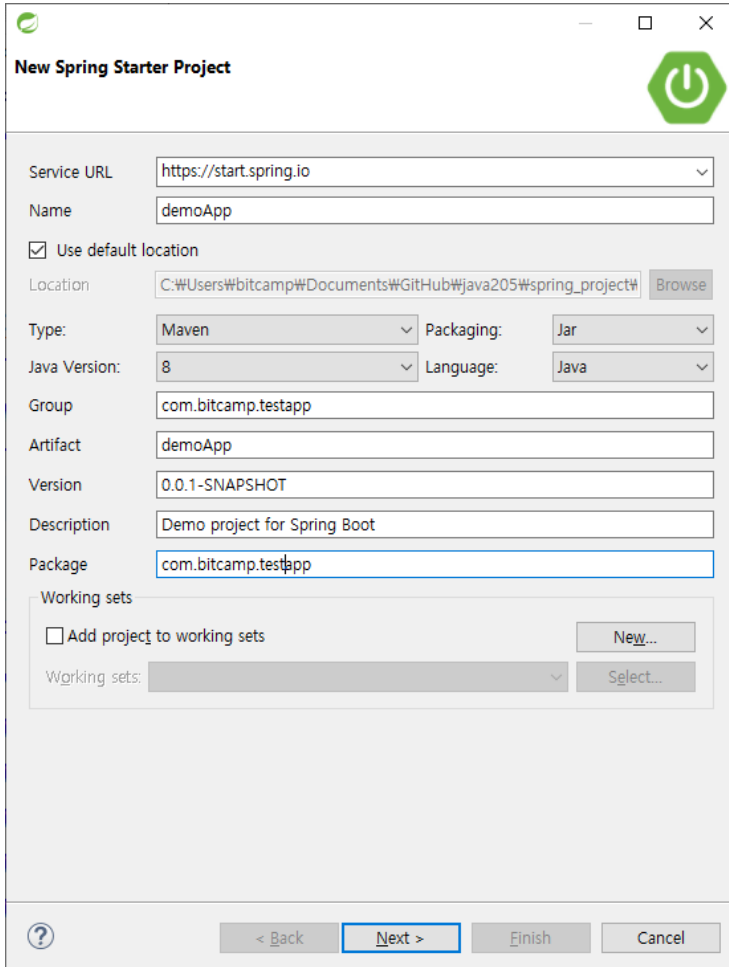
# Spring Boot

- Spring Boot Starters : 스프링 부트 테크니컬 스타터

이름	설명
<b>spring-boot-starter-jetty</b>	내장 서블릿 컨테이너로 Jetty를 사용하기 위한 스타터. 대안spring-boot-starter-tomcat
<b>spring-boot-starter-log4j2</b>	로깅을 위해 Log4j2를 사용하기 위한 스타터. 대안spring-boot-starter-logging
<b>spring-boot-starter-logging</b>	Logback을 이용한 로깅 스타터. 기본 로깅 스타터
<b>spring-boot-starter-reactor-netty</b>	임베디드 Reactive HTTP 서버로 Reactor Netty를 사용하기 위한 스타터.
<b>spring-boot-starter-tomcat</b>	임베디드 서블릿 컨테이너로 Tomcat을 사용하기 위한 스타터. 에 의해 사용되는 기본 서블릿 컨테이너 스타터spring-boot-starter-web
<b>spring-boot-starter-undertow</b>	Undertow를 임베디드 서블릿 컨테이너로 사용하기 위한 스타터. 대안spring-boot-starter-tomcat

# Spring Boot

- 프로젝트 생성



**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:

Service URL : <https://start.spring.io>

Use default location : 체크 (기본 프로젝트 경로 변경을 원하면 해제 후 지정)

Type : Maven

Packaging : Jar

Java Version : 8

Language : Java

Group : egovframework.msa.sample

Artifact : Catalogs

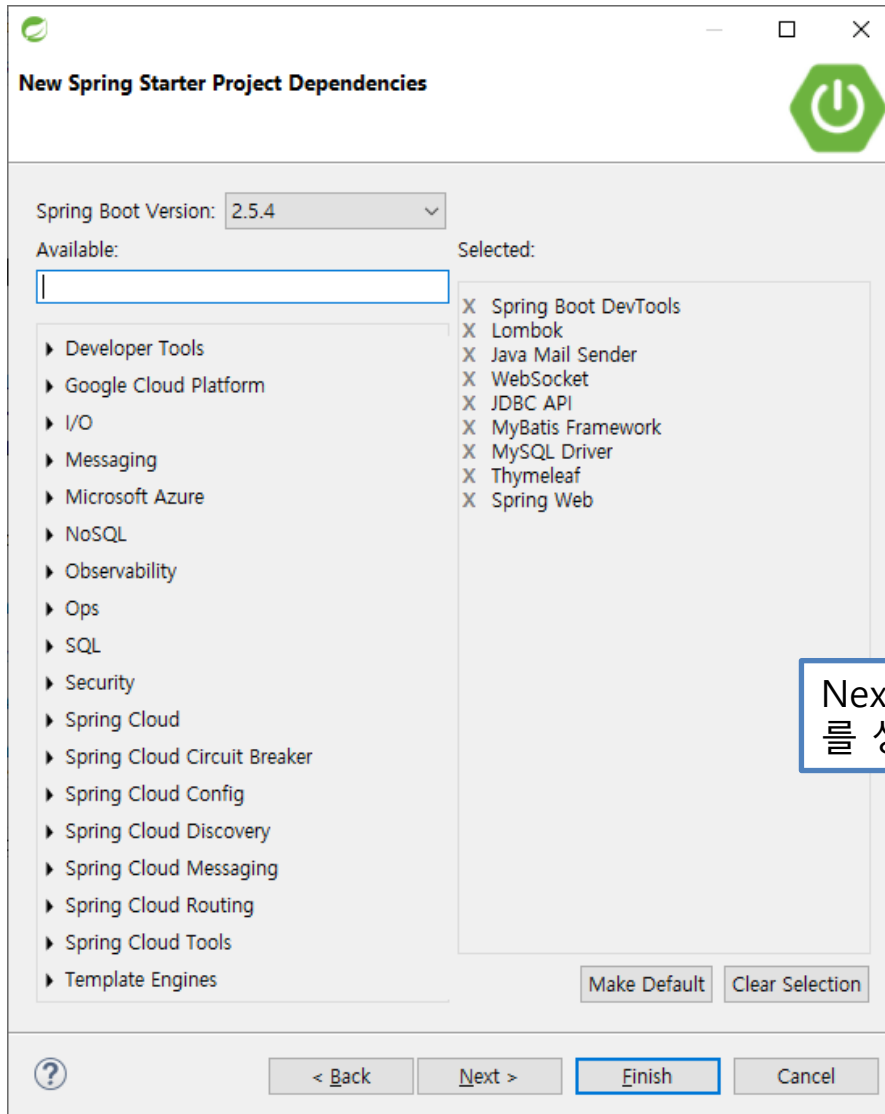
Version : 1.0.0

Description : MSA Sample Project

Group Id : egovframework.msa.sample

# Spring Boot

- 프로젝트 생성



**New Spring Starter Project Dependencies**

Spring Boot Version: 2.5.4

Available:

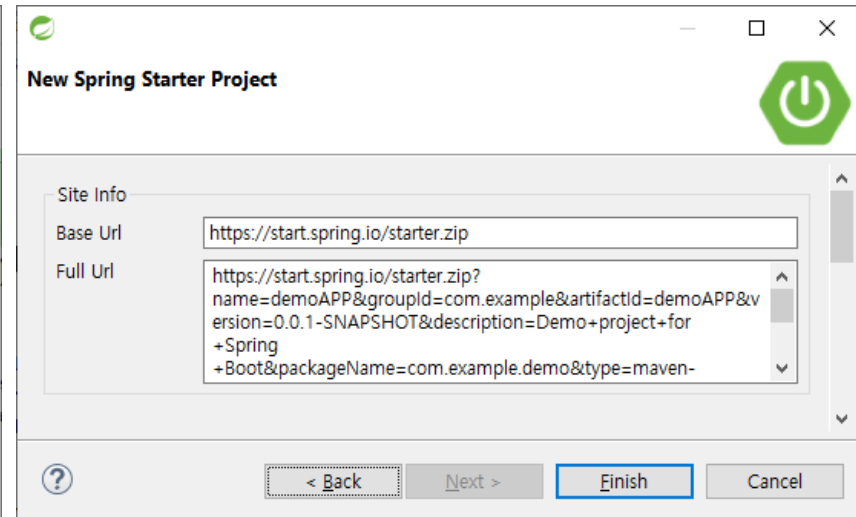
- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud
- Spring Cloud Circuit Breaker
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Messaging
- Spring Cloud Routing
- Spring Cloud Tools
- Template Engines

Selected:

- X Spring Boot DevTools
- X Lombok
- X Java Mail Sender
- X WebSocket
- X JDBC API
- X MyBatis Framework
- X MySQL Driver
- X Thymeleaf
- X Spring Web

Make Default Clear Selection

< Back Next > Finish Cancel



**New Spring Starter Project**

Site Info

Base Url:

Full Url:

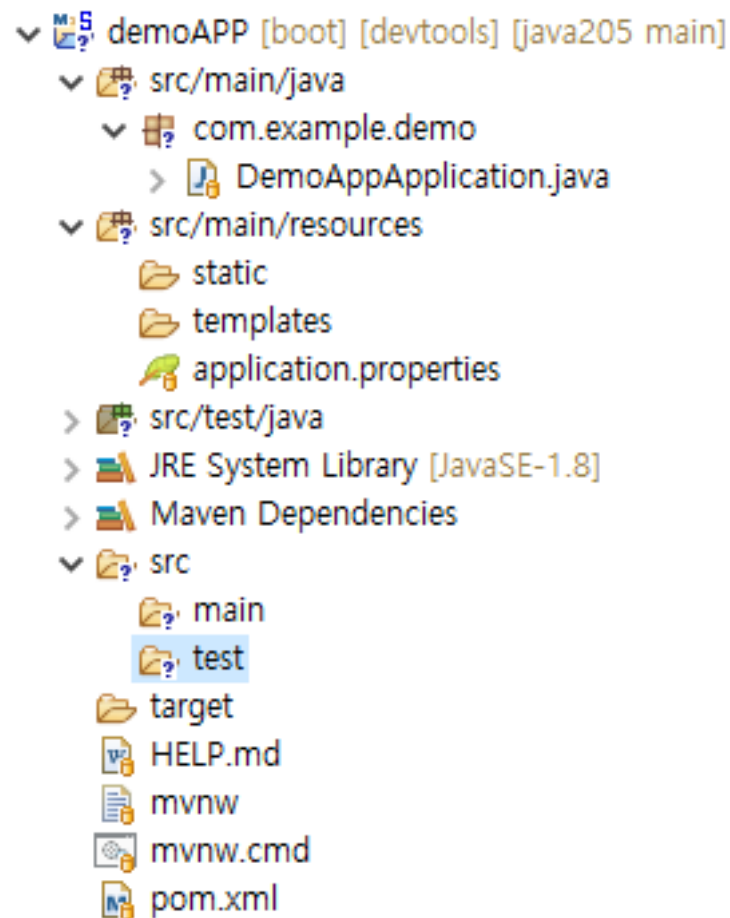
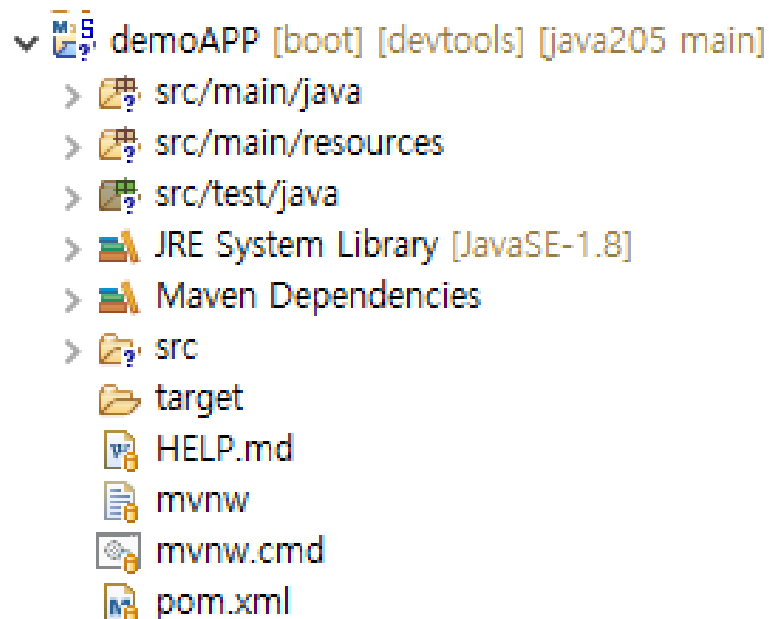
? < Back Next > Finish Cancel

Next > Finish 또는 Finish 를 바로 선택하여 프로젝트를 생성한다.



# Spring Boot

- 생성된 프로젝트



# Spring Boot

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>demoAPP</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demoAPP</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
```

# Spring Boot

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-websocket</artifactId>
  </dependency>
  <dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.2.0</version>
  </dependency>
</dependencies>
```

# Spring Boot

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>
```

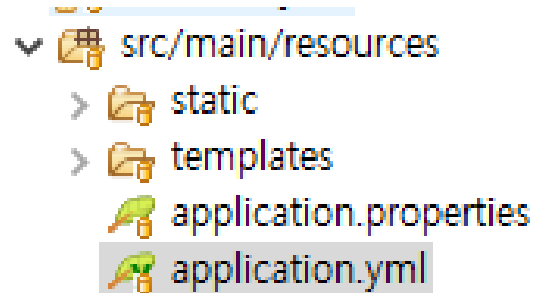
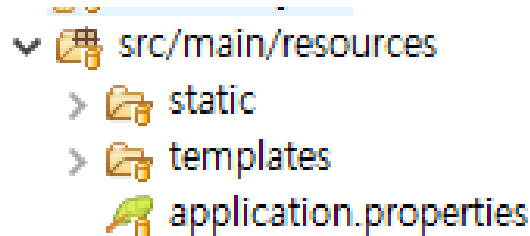
# Spring Boot

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>
```

# Spring Boot

- **static** : HTML, CSS, JavaScript, 이미지 파일들을 보관하는 경로
- **templates** : Thymeleaf와 같은 템플릿 경로
- **application.properties** : 애플리케이션 내의 설정파일



- **yaml 파일 사용하기**
  - new → File → application.yml

# Spring Boot

- application.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://[RDS 주소]:3306/project?servertimezone=UTC
    username: bit
    password: bit

  thymeleaf:
    prefix: classpath:templates/
    check-template-location: true
    suffix: .html
    mode: HTML5
    cache: false
    template-resolver-order: 0

  mybatis:
    mapper-locations: com/bitcamp/demo/member/mapper/**/*.xml

  server:
    port: 8081
```

# Spring Boot

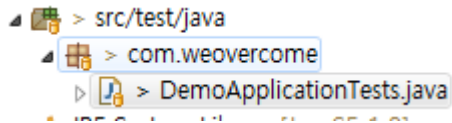
- DataSource 설정
  - application.properties 를 이용한 데이터 설정

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/project?
serverTimezone=UTC
spring.datasource.username=bit
spring.datasource.password=bit
```



# Spring Boot

- datasource 테스트



```
package com.weovercome;

import java.sql.Connection;
import java.sql.SQLException;

import javax.sql.DataSource;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class DemoApplicationTests {

    @Autowired
    private DataSource datasource;

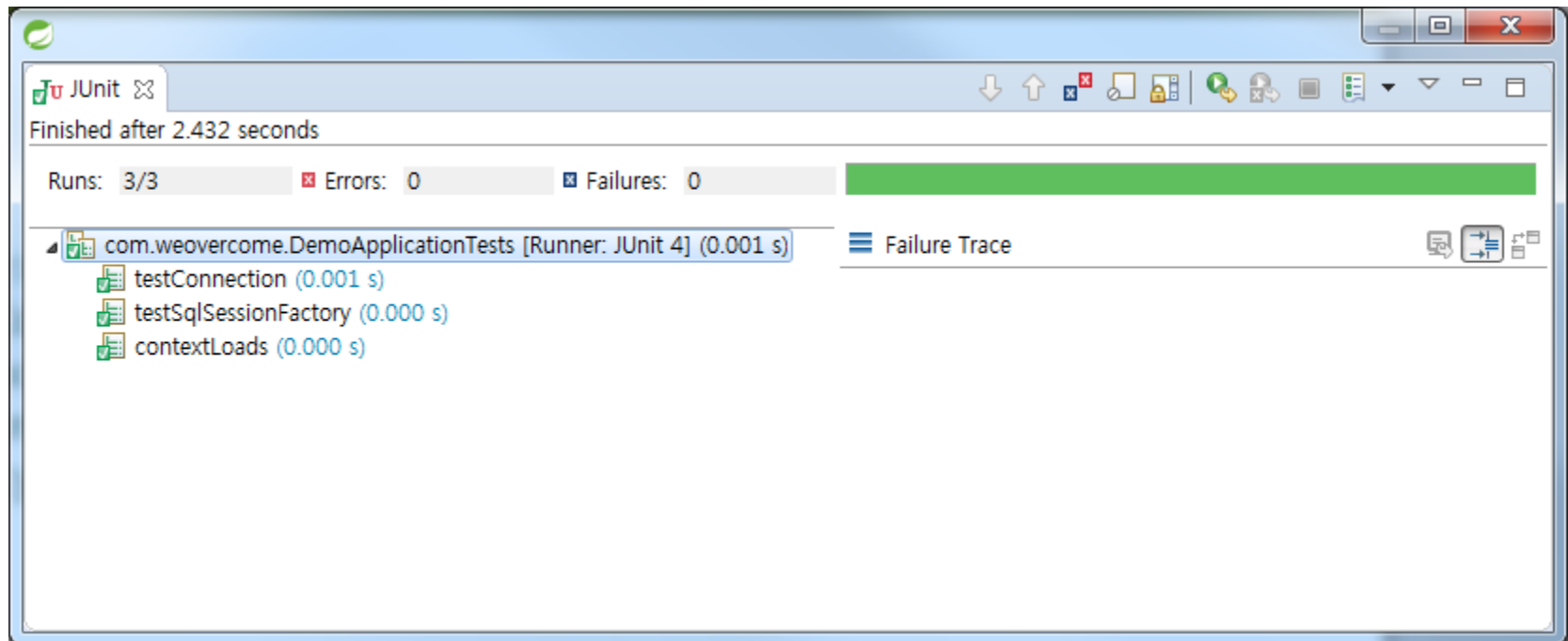
    @Test
    public void contextLoads() {
    }

    @Test
    public void testConnection() throws SQLException {
        System.out.println(datasource);
        Connection conn = datasource.getConnection();
        System.out.println(conn);
        conn.close();
    }
}
```

# Spring Boot

- Mybatis 설정

```
<dependency>  
  <groupId>org.mybatis.spring.boot</groupId>  
  <artifactId>mybatis-spring-boot-starter</artifactId>  
  <version>2.1.0</version>  
</dependency>
```

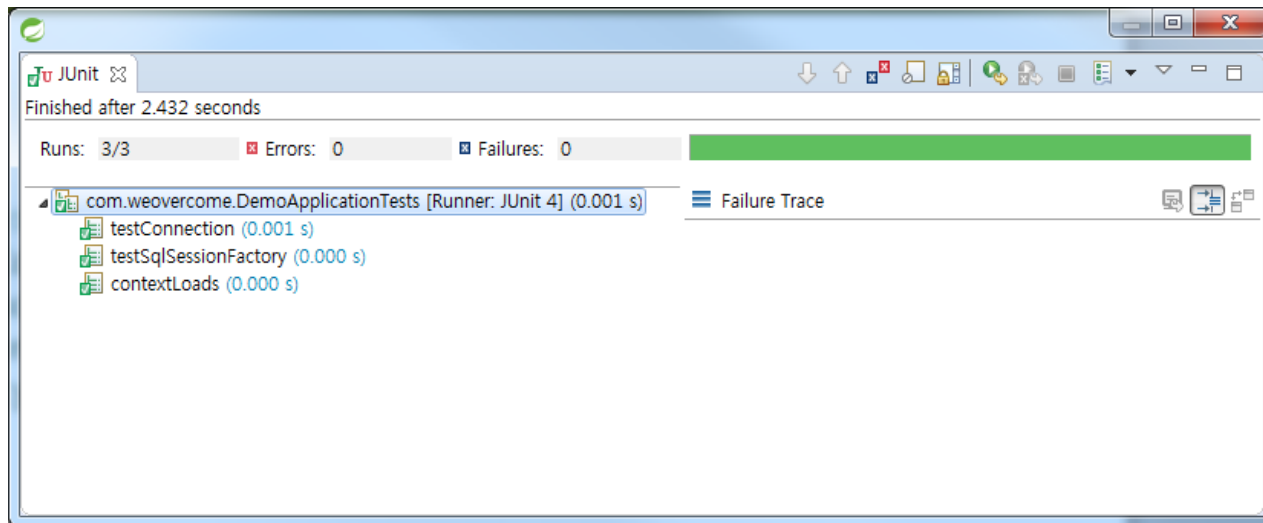


# Spring Boot

- Mybatis 설정

```
@Autowired
private SqlSessionFactory sessionFactory;

@Test
public void testSqlSessionFactory() {
    System.out.println(sessionFactory);
}
```



# Spring Boot

- **MemberInfo.java**

com.weovercome.domain  
MemberInfo.java

```
package com.weovercome.domain;

public class MemberInfo {

    private int idx;
    private String uid;
    private String uPW;
    private String uName;
    private String uPhoto;
    private Date date;
    private char verify;
    private String code;

    // default 생성자 필수
    public MemberInfo() {
        this.date = new Date();
        getRandomString();
    }

    // 변수들의 Getter/Setter 시작
}
```

# Spring Boot

- Mapper 생성

com.weovercome.mapper  
MemberMapper.java

```
package com.weovercome.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.springframework.web.bind.annotation.RequestParam;

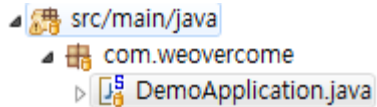
import com.weovercome.domain.MemberInfo;

public interface MemberMapper {

    @Select("SELECT * FROM member order by idx")
    public List<MemberInfo> getMemberList();
    public MemberInfo selectMemberById(
        @RequestParam("uId") String uId);
}
```

# Spring Boot

- Mapper 설정



```
package com.weovercome;

import org.mybatis.spring.annotation.MapperScan;

@SpringBootApplication
@MapperScan("com.weovercome.mapper")
public class DemoApplication {

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

}
```

# Spring Boot

- 외부 XML mapper 설정

src/main/resources

- mappers
  - memberMapper.xml
- static
- templates
- application.properties

```
1 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
2 spring.datasource.url=jdbc:mysql://localhost:3306/project?a
3 spring.datasource.username=bit
4 spring.datasource.password=bit
5
6 mybatis.mapper-locations=classpath:/mappers/*.xml
```

# Spring Boot

- View 처리 방법
  - JSP 사용방법
    - application.properties 설정 변경
    - Tomcat 서버의 JSP 설정 변경
  - Thymeleaf 사용방법



# Spring Boot

- JSP사용 설정
  - application.properties 설정 변경

```
1 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
2 spring.datasource.url=jdbc:mysql://localhost:3306/project?autoReco
3 spring.datasource.username=bit
4 spring.datasource.password=bit
5
6 mybatis.mapper-locations=classpath:/mappers/*.xml
7
8 spring.mvc.view.prefix=/WEB-INF/views/
9 spring.mvc.view.suffix=.jsp|
```



경로에 맞게 폴더 생성해야 한다.  
main > webapp > WEB-INF > views

# Spring Boot

- JSP사용 설정
  - Tomcat 서버의 JSP 설정 변경

```
<dependency>  
<groupId>org.apache.tomcat.embed</groupId>  
<artifactId>tomcat-embed-jasper</artifactId>  
</dependency>
```

```
<dependency>  
<groupId>javax.servlet</groupId>  
<artifactId>jstl</artifactId>  
</dependency>
```

Spring boot

# SPRING BOOT THYMELEAF 기본 문법

<https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#standard-expression-syntax>

# 1. Thymeleaf 기본 표현

- **Thymeleaf Document**

- 간단한 표현
- 리터럴
- 텍스트 작업
- 산술 연산
- 부울 연산
- 비교 및 평등
- 조건 연산자
- [링크](#)

## Simple expressions

- Variable Expressions: `${...}`
- Selection Variable Expressions: `*{...}`
- Message Expressions: `#{...}`
- Link URL Expressions: `@{...}`
- Fragment Expressions: `~{...}`

## Literals

- Text literals: 'one text', 'Another one!',...
- Number literals: 0, 34, 3.0, 12.3,...
- Boolean literals: true, false
- Null literal: null
- Literal tokens: one, sometext, main,...

## Text operations

- String concatenation: `+`
- Literal substitutions: `|The name is ${name}|`

## Arithmetic operations

- Binary operators: `+`, `-`, `*`, `/`, `%`
- Minus sign (unary operator): `-`

## Boolean operations

- Binary operators: `and`, `or`
- Boolean negation (unary operator): `!`, `not`

## Comparisons and equality

- Comparators: `>`, `<`, `>=`, `<=` ( `gt`, `lt`, `ge`, `le` )
- Equality operators: `==`, `!=` ( `eq`, `ne` )

## Conditional operators

- If-then: `(if) ? (then)`
- If-then-else: `(if) ? (then) : (else)`
- Default: `(value) ?: (defaultvalue)`

## 2. 자주 사용하는 문법 및 표현식 예제

- 표현식 : `th:[속성]="서버 전달 받은 값 또는 조건식"`
  - Tag 안에 삽입되면 됨. Thymeleaf 3.x 에서는 inline 표현식이 추가되어 html 태그 없이 표현이 가능. `[[${member.name}]]`

Title	Description
th:text	텍스트 내용 <code>&lt;span th:text="\${user.name}"&gt;&lt;/span&gt;</code>
th:utext	html 내용 <code>&lt;div th:utext="\${content}"&gt;&lt;/div&gt;</code>
th:value	element value값, checkbox, input 등 <code>&lt;input type="text" th:value="\${title}" /&gt;</code>
th:with	변수값 지정 <code>&lt;p th:with="authType = \${user.authType}+' Type'" th:text="\${authType}"&gt;&lt;/p&gt;</code>
th:switch th:case	switch-case문 <code>&lt;div th:switch="\${user.role}"&gt;   &lt;p th:case="admin"&gt;User is an administrator   &lt;p th:case="#{roles.manager}"&gt;User is a manager &lt;/div&gt;</code>
th:if	조건문 <code>&lt;p th:if="\${user.authType}=='web'" th:text="\${user.authType}"&gt;&lt;/p&gt;</code>
th:unless	else 표현 <code>&lt;p th:unless="\${user.authType}=='facebook'" th:text="not ' + \${user.authType}"&gt;&lt;/p&gt;</code>
th:each	반복문 <code>&lt;p th:each="user : \${users}" th:text="\${user.name}"&gt;&lt;/p&gt;</code>















- <https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#appendix-b-expression-utility-objects>

## 2. 자주 사용하는 문법 및 표현식 예제

th:abbr	th:accept	th:accept-charset	th:ondrop	th:ondurationchange	th:onemptied
th:accesskey	th:action	th:align	th:onended	th:onerror	th:onfocus
th:alt	th:archive	th:audio	th:onformchange	th:onforminput	th:onhashchange
th:autocomplete	th:axis	th:background	th:oninput	th:oninvalid	th:onkeydown
th:bgcolor	th:border	th:cellpadding	th:onkeypress	th:onkeyup	th:onload
th:cellspacing	th:challenge	th:charset	th:onloadeddata	th:onloadedmetadata	th:onloadstart
th:cite	th:class	th:classid	th:onmessage	th:onmousedown	th:onmousemove
th:codebase	th:codetype	th:cols	th:onmouseout	th:onmouseover	th:onmouseup
th:colspan	th:compact	th:content	th:onmousewheel	th:onoffline	th:ononline
th:contenteditable	th:contextmenu	th:data	th:onpause	th:onplay	th:onplaying
th:datetime	th:dir	th:draggable	th:onpopstate	th:onprogress	th:onratechange
th:dropzone	th:enctype	th:for	th:onreadystatechange	th:onredo	th:onreset
th:form	th:formaction	th:formenctype	th:onresize	th:onscroll	th:onseeked
th:formmethod	th:formtarget	th:fragment	th:onseeking	th:onselect	th:onshow
th:frame	th:frameborder	th:headers	th:onstalled	th:onstorage	th:onsubmit
th:height	th:high	th:href	th:onsuspend	th:ontimeupdate	th:onundo
th:hreflang	th:hspace	th:http-equiv	th:onunload	th:onvolumechange	th:onwaiting
th:icon	th:id	th:inline	th:optimum	th:pattern	th:placeholder
th:keytype	th:kind	th:label	th:poster	th:preload	th:radiogroup
th:lang	th:list	th:longdesc	th:rel	th:rev	th:rows
th:low	th:manifest	th:marginheight	th:rowspan	th:rules	th:sandbox
th:marginwidth	th:max	th:maxlength	th:scheme	th:scope	th:scrolling
th:media	th:method	th:min	th:size	th:sizes	th:span
th:name	th:onabort	th:onafterprint	th:spellcheck	th:src	th:srclang
th:onbeforeprint	th:onbeforeunload	th:onblur	th:standby	th:start	th:step
th:oncanplay	th:oncanplaythrough	th:onchange	th:style	th:summary	th:tabindex
th:onclick	th:oncontextmenu	th:ondblclick	th:target	th:title	th:type
th:ondrag	th:ondragend	th:ondragenter	th:usemap	th:value	th:valuetype
th:ondragleave	th:ondragover	th:ondragstart	th:vspace	th:width	th:wrap
			th:xmlbase	th:xmllang	th:xmlspace

### 3. 프로젝트 파일 구조

- 정적 파일과 Thymeleaf

- ▼  > src/main/resources
  - ▼  static
    - >  css
    - >  images
    - >  js
  - ▼  > templates
    - ▼  frame
      -  bootstrap\_header.html
      -  footer.html
      -  metaheader.html
    - ▼  > member
      -  list.html
  -  application.properties
  -  application.yml

### 3. 프로젝트 파일 구조

- Thymeleaf : list.html

```
<!DOCTYPE html>
<html xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
<meta charset="UTF-8">
<title>Open Project : 로그인</title>

<th:block th:replace="frame/metaheader"></th:block>

</head>
<body class="bg-light">

    <th:block th:replace="frame/bootstrap_header :: bootstrap_header"></th:block>
```



### 3. 프로젝트 파일 구조

- Thymeleaf : list.html

```
<main role="main" class="container">
  <div class="my-3 p-3 bg-white rounded shadow-sm">
    <h2>회원 리스트</h2><hr>
    <form class="form-inline">
      <label class="mr-2">검색타입</label> <select name="searchType"
        class="form-control mr-2">
        <option value="id">아이디</option>
        <option value="name">이름</option>
        <option value="both">아이디+이름</option>
      </select> <label class="mr-2">검색 키워드</label> <input type="text"
        class="form-control mr-2" name="keyword"> <input
        type="submit" class="form-control btn btn-success mr-2" value="검색">
    </form>
    <hr>
  </div>
  <table class="table table-hover">
    <tr>
      <th>IDX</th>
      <th>아이디</th>
      <th>비밀번호</th>
      <th>이름</th>
      <th>사진</th>
      <th>가입일</th>
      <th>관리</th>
    </tr>
```

### 3. 프로젝트 파일 구조

- Thymeleaf : list.html

```
<tr th:each="member : ${memberList}">
  <td th:text="${member.idx}"></td>
  <td th:text="${member.memberid}"></td>
  <td th:text="${member.password}"></td>
  <td th:text="${member.membername}"></td>
  <td th:text="${member.memberphoto}"></td>
  <td th:text="${#calendars.format(member.date, 'yyyy.MM.dd. HH:mm')}"></td>
  <td>
    <a class="btn btn-primary" href="edit?idx=[[${member.idx}]]">수정</a>
    <a class="btn btn-danger" href="delete?idx=[[${member.idx}]]">삭제</a>
  </td>
</tr>
</table>

</main>

<th:block th:include="frame/footer"></th:block>

</body>
</html>
```

### 3. 프로젝트 파일 구조

- Thymeleaf : bootstrap\_header.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<nav class="navbar navbar-expand-lg fixed-top navbar-dark bg_red" th:fragment="bootstrap_header">
    <!-- LOGO -->
    <a class="navbar-brand mr-auto mr-lg-0" href="#">Offcanvas navbar</a>
    <!-- 햄버거 토글 버튼 : offcanvas -->
    <button class="navbar-toggler p-0 border-0" type="button" data-toggle="offcanvas">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="navbar-collapse offcanvas-collapse" id="navbarsExampleDefault">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item "><a class="nav-link" href="@{/member/memberReg}">회원가입</a></li>
            <li class="nav-item"><a class="nav-link" href="@{/member/login}">로그인</a></li>
            <li class="nav-item"><a class="nav-link" href="@{/member/logout}">로그아웃</a></li>
            <li class="nav-item"><a class="nav-link" href="@{/member/list}">회원리스트</a></li>
            <li class="nav-item"><a class="nav-link" href="#">방명록</a></li>
        </ul>
    </div>
</nav>
</html>
```

### 3. 프로젝트 파일 구조

- Thymeleaf : footer.html

```
<!-- <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
      crossorigin="anonymous"></script> -->

<script>
    window.jQuery
        || document
            .write('<script src="@{/js/vendor/jquery.slim.min.js}"></script>')
</script>

<script th:src="@{/js/bootstrap.bundle.min.js}"></script>
<script th:src="@{/js/offcanvas.js}"></script>
```

### 3. 프로젝트 파일 구조

- Thymeleaf : bootstrap\_header.html

```
<!-- <link rel="stylesheet" href="/op/css/default.css"> -->
<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
<link rel="canonical" href="https://getbootstrap.com/docs/4.6/examples/offcanvas/">
  <!-- Bootstrap core CSS -->
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<style>
.bd-placeholder-img {
    font-size: 1.125rem; text-align: middle;
    -webkit-user-select: none; -moz-user-select: none;
    -ms-user-select: none; user-select: none;
}
@media ( min-width : 768px) {
    .bd-placeholder-img-lg { font-size: 3.5rem; }
}
</style>

<!-- Custom styles for this template -->
<link th:href="@{/css/offcanvas.css}" rel="stylesheet">

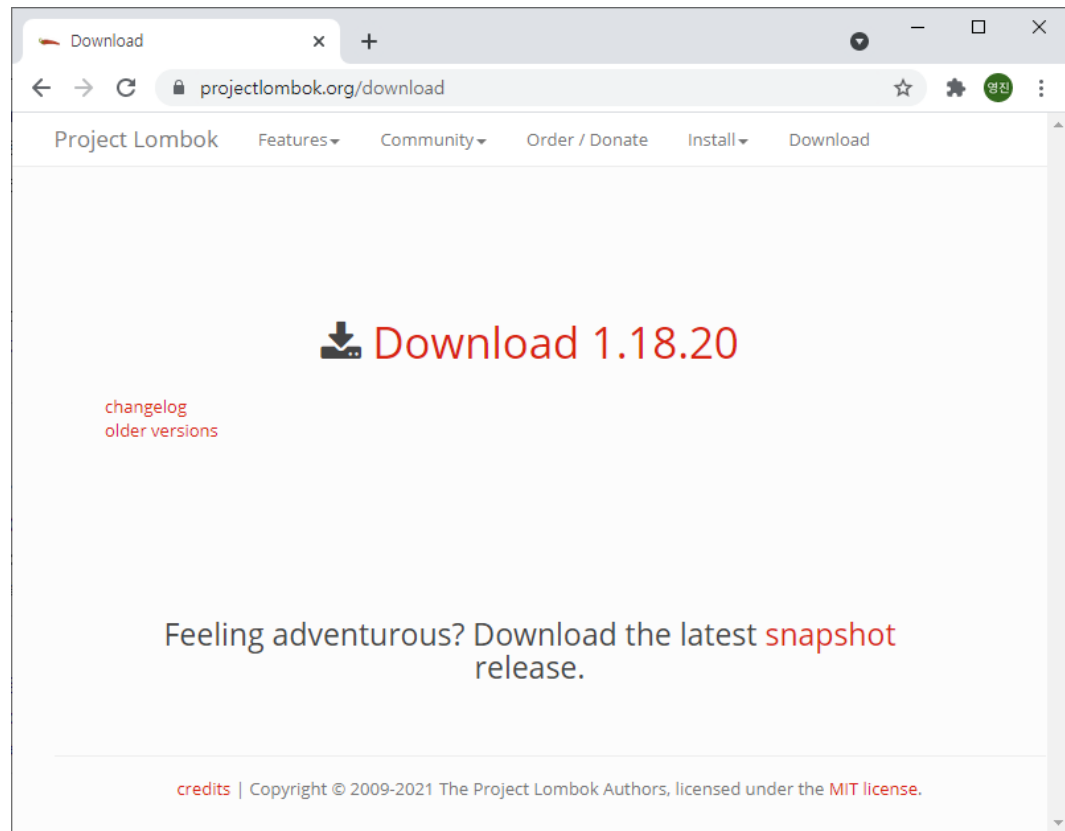
<style>
    .bg_ornage{ background-color: orange; }
    .bg_red{ background-color: red; }
    .nav-link{ color: white!important; }
</style>
```

**Spring boot**

**LOMBOK**

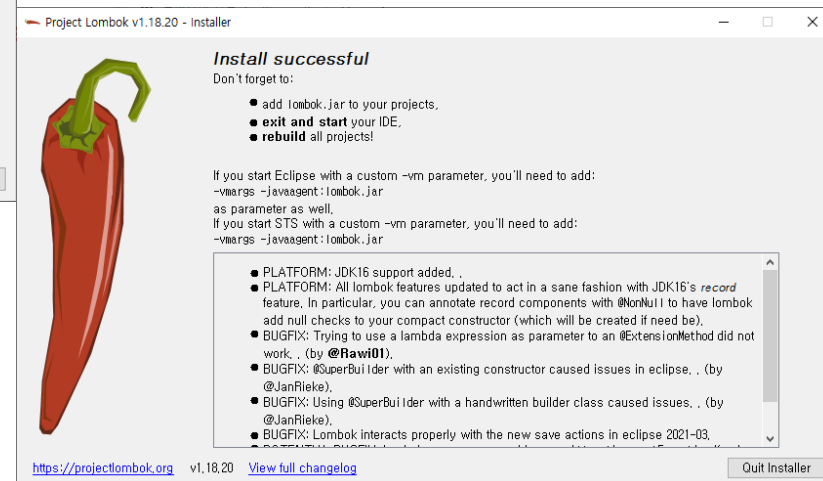
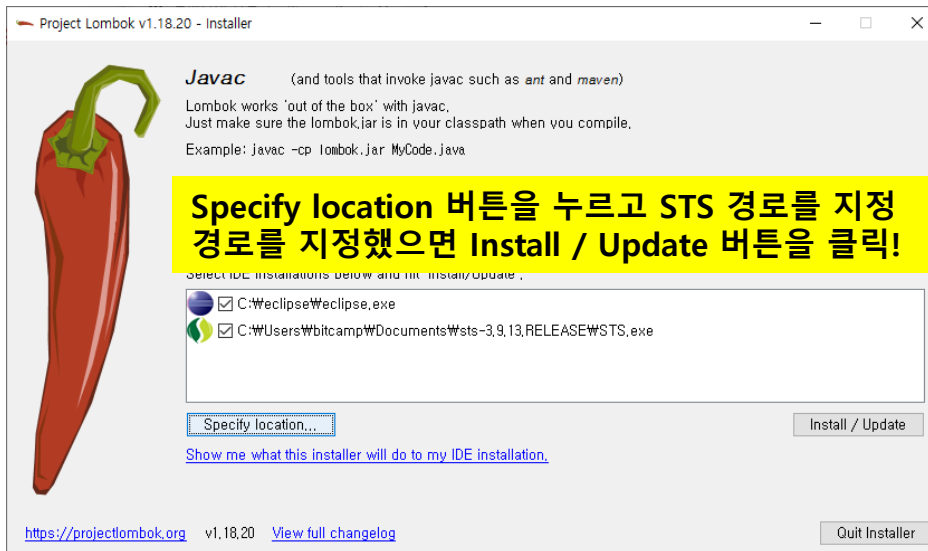
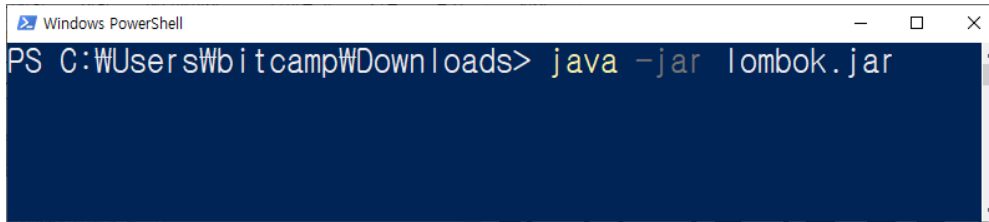
# Lombok

- Lombok 설정
  - lombok은 스프링에서 VO나 DTO에서 getter / setter를 자동으로 설정해줍니다.
- lombok 다운로드
  - <https://projectlombok.org/download>
  - lombok.jar을 다운로드



# Lombok

- **Lombok 설치**





- 자주 쓰는 Lombok Annotations

Annotation	설명
@NonNull	Null 값이 될 수 없다는 것을 명시합니다. NullPointerException에 대한 대비책이 될 수 있습니다.
@Cleanup	자동으로 close() 메소드를 호출합니다.
@Getter/@Setter	코드가 컴파일 될 때, Getter/Setter 메소드를 생성합니다.
@ToString	toString() 메소드를 생성합니다. @ToString(exclude={"제외할 값"}) 처럼 원하지 않는 속성은 제외할 수 있습니다.
@EqualsAndHashCode	해당 객체의 equals()와 hashCode() 메소드를 생성합니다.
@NoArgsConstructor	파라미터를 받지 않는 생성자를 만들어 줍니다.
@RequiredArgsConstructor	지정된 속성들에 대해서만 생성자를 만듭니다.
@AllArgsConstructor	모든 속성에 대해서 생성자를 만들어 냅니다.
@Data	@ToString, @EqualsAndHashCode, @Getter, @Setter, @RequiredArgsConstructor를 합쳐 둔 어노테이션입니다.
@Value	불변 클래스를 생성할 때 사용합니다.
@Builder	빌더 패턴을 사용할 수 있도록 코드를 생성합니다.
@SneakyThrows	예외 발생 시 Throwable 타입으로 반환합니다.
@Synchronized	메소드에서 동기화를 설정합니다.
@Getter(lazy=true)	동기화를 이용해서 최초 한번만 getter를 호출합니다.

Spring boot

**SPRING BOOT**

**빌드-배포-실행**

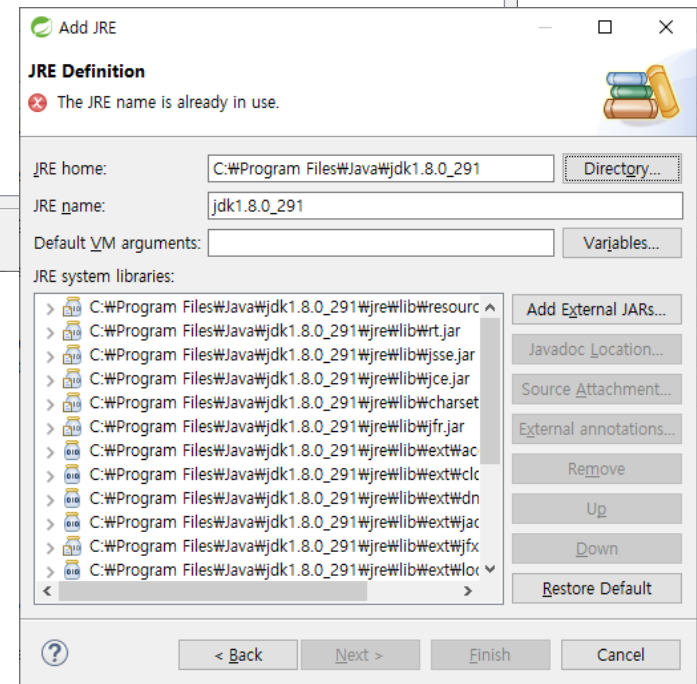
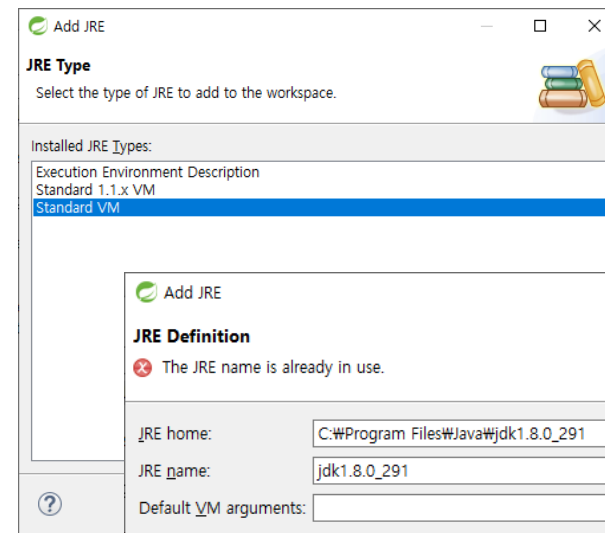
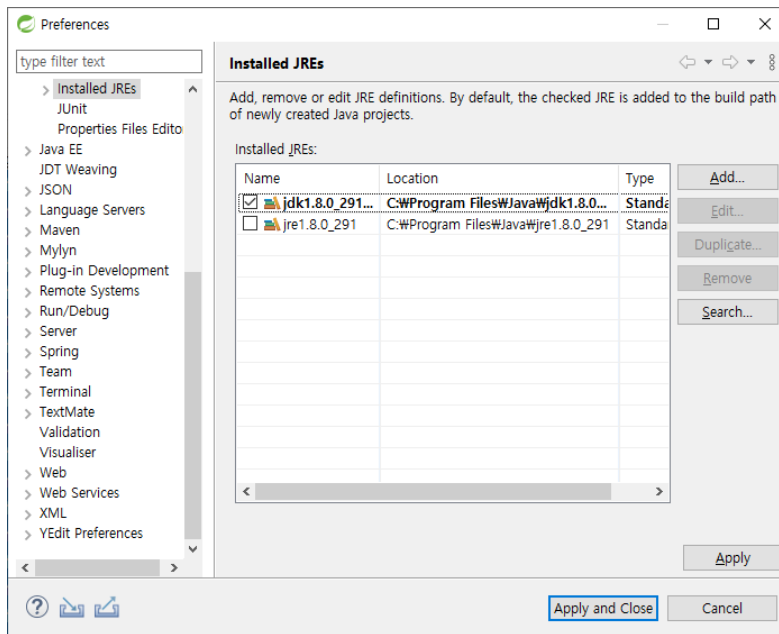
- **Project + Run As + Maven Build**

The screenshot shows the 'Edit Configuration' dialog box for a Maven build. The title bar says 'Edit Configuration' with standard window controls. Below the title bar, it says 'Edit configuration and launch.' and has a green play button icon.

The dialog is divided into several sections:

- Name:** A text field containing 'Demo'.
- Tabs:** A row of tabs: 'Main' (selected), 'JRE', 'Refresh', 'Source', 'Launch Extensions', 'Environment', and 'Common'.
- Base directory:** A text field containing '\${project\_loc:Demo}'. Below it are buttons for 'Workspace...', 'File System...', and 'Variables...'.
- Goals:** A text field containing 'package'.
- Profiles:** An empty text field.
- User settings:** A text field containing 'C:\Users\bitcamp\m2\settings.xml'. Below it are buttons for 'Workspace...', 'File System...', and 'Variables...'.
- Options:** A group of checkboxes: 'Offline', 'Update Snapshots', 'Debug Output', 'Skip Tests', 'Non-recursive', and 'Resolve Workspace artifacts'. All are currently unchecked.
- Threads:** A dropdown menu showing '1' and a label 'Threads'.
- Parameter Table:** A table with two columns: 'Parameter Name' and 'Value'. It is currently empty. To the right of the table are buttons for 'Add...', 'Edit...', and 'Remove'.
- Maven Runtime:** A dropdown menu showing 'EMBEDDED (3.6.3/1.16.0.20200610-1735)'. To the right is a 'Configure...' button.
- Buttons:** At the bottom right are 'Revert' and 'Apply' buttons. At the very bottom are a help icon (?), a 'Run' button (highlighted with a blue border), and a 'Close' button.

- 컴파일 오류가 난다면?!!
- 컴파일러가 없다고 나온다면
- Preference → Java → Installed JREs → add → JDK 경로 설정



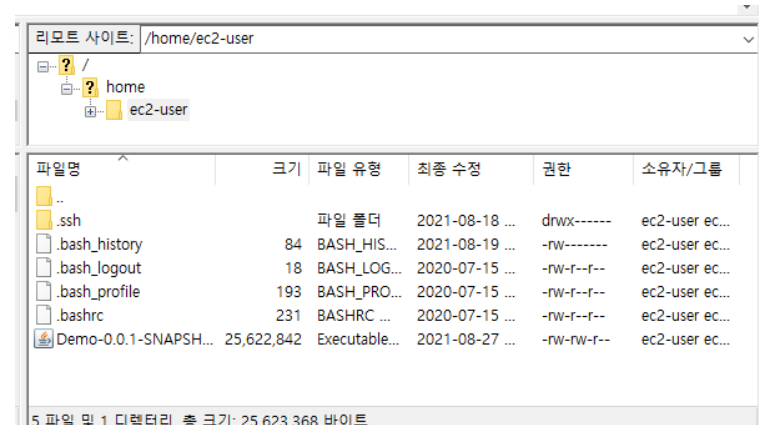
- 빌드하면 target 폴더에 jar 파일 생성
- Demo.jar → ec2에 파일 업로드
- Putty SSH 접속

```
[root@ip-172-31-15-189 ~]# cd /home/ec2-user
```

```
[root@ip-172-31-15-189 ec2-user]# ls
```

```
Demo-0.0.1-SNAPSHOT.jar
```

```
[root@ip-172-31-15-189 ec2-user]# java -jar Demo-0.0.1-SNAPSHOT.jar
```



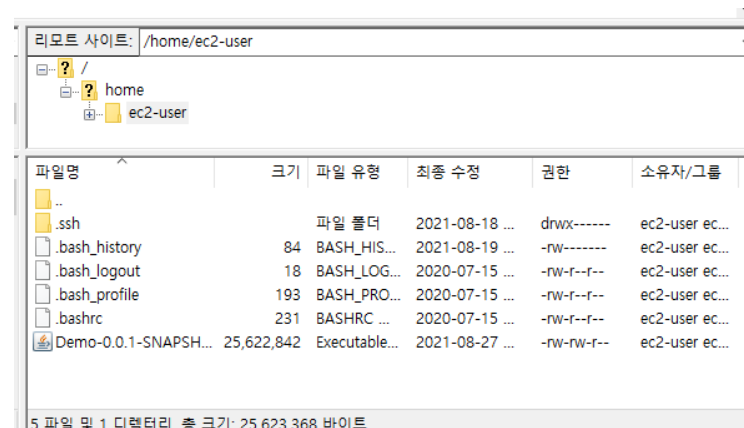
- 빌드하면 target 폴더에 jar 파일 생성
- Demo.jar → ec2에 파일 업로드
- Putty SSH 접속

```
[root@ip-172-31-15-189 ~]# cd /home/ec2-user
```

```
[root@ip-172-31-15-189 ec2-user]# ls
```

```
Demo-0.0.1-SNAPSHOT.jar
```

```
[root@ip-172-31-15-189 ec2-user]# java -jar Demo-0.0.1-SNAPSHOT.jar
```



- AWS 보안설정

인바운드 규칙 [정보](#)

Security group rule ID	유형 <a href="#">정보</a>	프로토콜 <a href="#">정보</a>	포트 범위 <a href="#">정보</a>	소스 <a href="#">정보</a>	설명 - 선택 사항 <a href="#">정보</a>		
sgr-0b42bcbf648f2cf55	<div>사용자 지정 TCP ▼</div>	TCP	8080	<div>사용자 지정 ▼</div>	<div>Q</div>	<div></div>	<div>삭제</div>
					<div>::/0 ✕</div>		
sgr-064169e4ce0b90301	<div>SSH ▼</div>	TCP	22	<div>사용자 지정 ▼</div>	<div>Q</div>	<div></div>	<div>삭제</div>
					<div>::/0 ✕</div>		
sgr-0c392317566cd8f84	<div>사용자 지정 TCP ▼</div>	TCP	8080	<div>사용자 지정 ▼</div>	<div>Q</div>	<div></div>	<div>삭제</div>
					<div>0.0.0.0/0 ✕</div>		
sgr-01aea811ecd10e4ad	<div>사용자 지정 TCP ▼</div>	TCP	8081	<div>사용자 지정 ▼</div>	<div>Q</div>	<div></div>	<div>삭제</div>
					<div>0.0.0.0/0 ✕</div>		
sgr-0fda0f82ba7937158	<div>SSH ▼</div>	TCP	22	<div>사용자 지정 ▼</div>	<div>Q</div>	<div></div>	<div>삭제</div>
					<div>0.0.0.0/0 ✕</div>		

규칙 추가