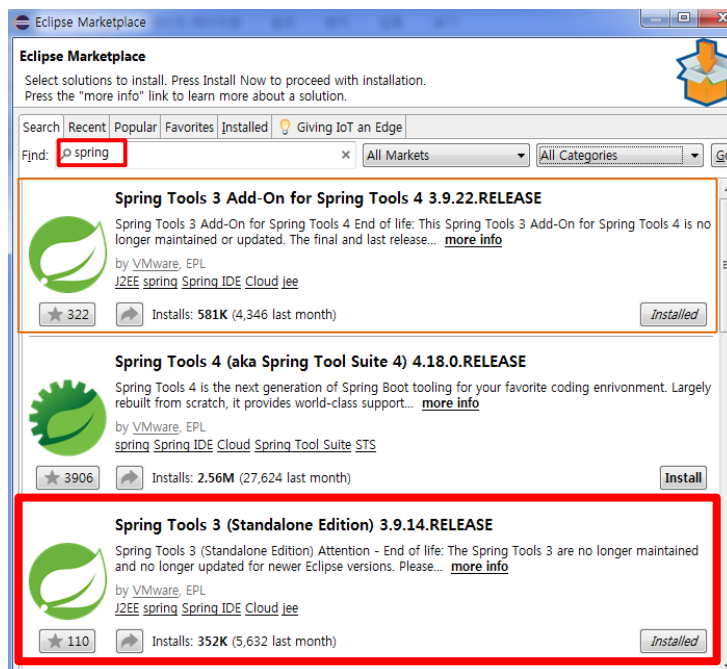
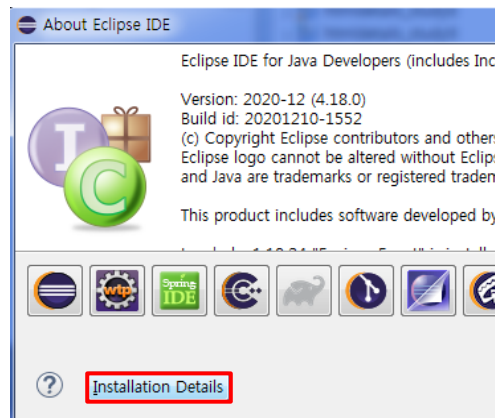
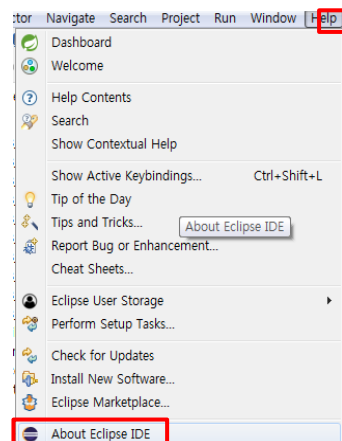


## 0. 환경 구축

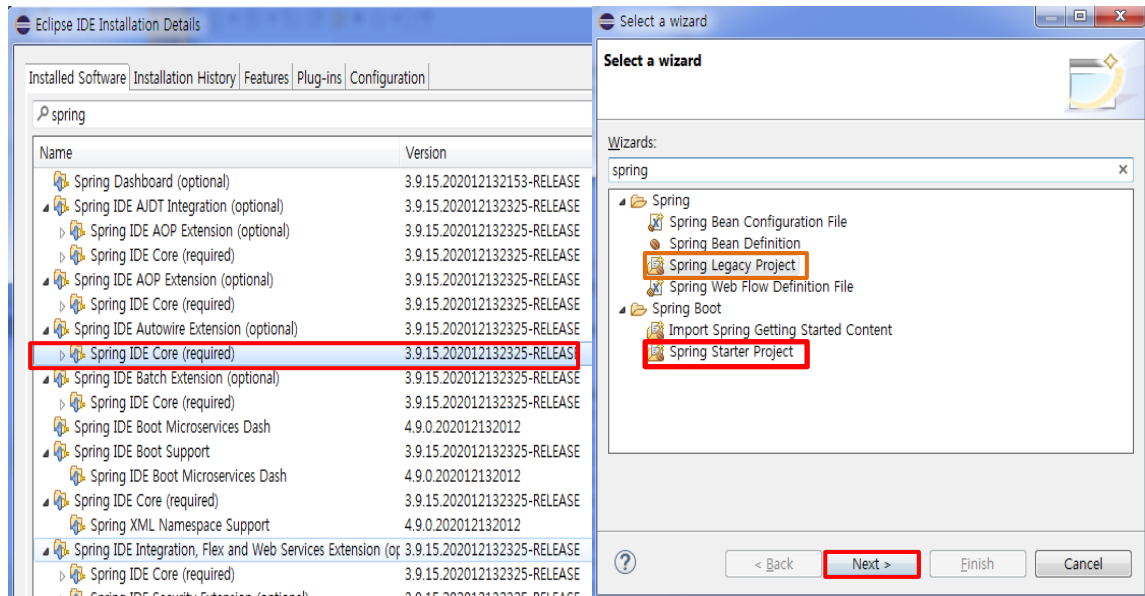
### 가. Spring Tools 설치



### 나. 설치 여부 확인



## 다. 설치 모듈 확인 및 프로젝트 마법사



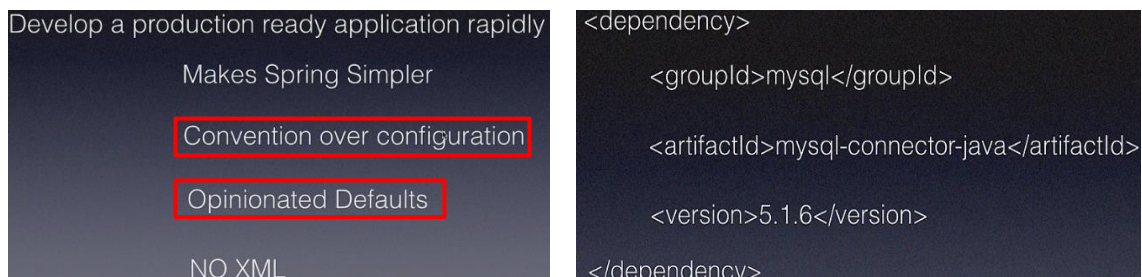
### 1. 소개

- 어플리케이션 개발 속도를 높여주는 스프링 모듈
- 모든 다른 스프링 모듈을 래핑해 쉽게 설정하고 사용할 수 있게 해줌
- 구성에 대한 규칙(convention over configuration) 사용
- 구성에 대한 규칙 의미: 프로젝트에 대한 특정 폴더 구조를 따르라고 하는 요청
- **Convention over configuration**은 클래스, 테스트, jar 또는 war 파일을 자동적으로 생성
- 스프링은 당신이 프로젝트를 만들 때 특정 Convention을 기대하며, 임의의 XML 파일을 만들지 않고, 자동적으로 필요한 모든 설정파일을 추가
- **Opinionated Defaults**: 기본 속성으로 추후 변경이 가능
- 예: 스프링 Web MVC

디스패처 서블릿을 설정: web.xml

뷰 리졸버 설정: dispatcher-servlet.xml

XML없이 런타임에 자동적으로 디스패처 서블릿 설정 및 변경 시 프로퍼티 파일 통해 변경



가정 A-1: 클래스 패스에서 MySQL에 대한 JDBC 드라이버 발견

가정 A-2: 클래스 패스에서 JPA관련 jar 파일 발견

결론 A: JPA의 데이터베이스 접속을 자동적으로 설정

가정B: 클래스 패스에서 스프링MVC 의존 발견

결론B: 디스패처 서블릿과 뷰 리졸버를 자동으로 설정

```
spring-mvc @EnableWebMvc
```

- 스프링 부트의 큰 이점: 많은 애노테이션을 받아들여 단일 애노테이션으로 합치기
- @SpringBootApplication = @Configuration + @EnableAutoConfiguration + @ComponentScan

```
@Configuration
@SpringBootApplication @EnableAutoConfiguration
@ComponentScan
```

- @SpringBootApplication: 스프링부트 애플리케이션의 시작점

## 2. 스프링부트 시작 프로젝트들(Starter Projects)

가. 프로젝트 유형

- spring-boot-**starter** dependency: 스탠드얼론 스프링 애플리케이션 빌드
- spring-boot-**starter-web** dependency
- spring-boot-**starter-data-jpa**
- spring-boot-**starter-data-rest**

```
spring-boot-starter
spring-boot-starter-web
spring-boot-starter-data-jpa
spring-boot-starter-data-rest
```

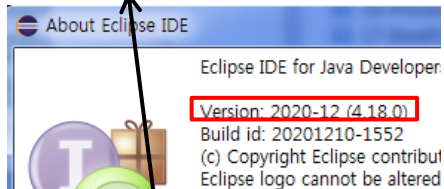
## 나. 프로젝트 생성

### - pom 설정 시 유의 사항

```
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-starter-parent</artifactId>
```

```
<version>...</version> ---> 이클립스버전 년도를 확인해
```



<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-parent> 접속해

적절한 Spring Boot Starter Parent 버전 확인 필요



```
<dependency>
```

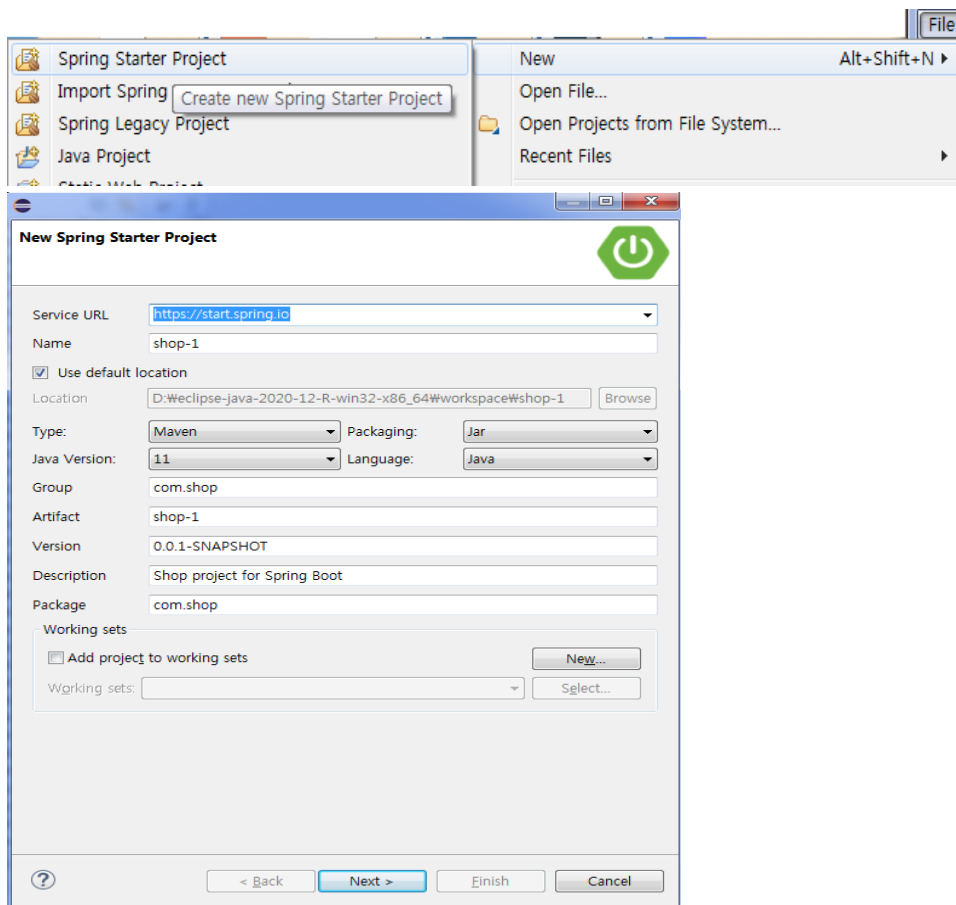
```
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-starter-parent</artifactId>
```

```
<version>2.4.1</version>
```

```
<type>pom</type>
```

```
</dependency>
```



### 3. 스프링 부트 작동 방식

- 즉각적으로 코드를 생성하지도 XML 설정을 사용하지도 않음
- (기존 자바 기반 설정에서 했던) 프로그래밍에 의한 설정(Programmatic configuration)
- 개발자가 자바 설정(web.xml, dispatcher-servlet.xml, hibernate 설정 등)을 만들어내 대신 스프링이 배포되는 jar 파일안에 모든 설정 클래스를 가지고 있어, 특정 조건에 활성화 됨

No Code Generation

No XML

Starter POMs

Add Jars

META-INF/spring.factories

프로젝트 유형 체크 @Condition

스프링Web 이면 @Configuration

Web.xml을 대체하는 자바기반 설정 클래스 생성

HibernateJpaAutoConfiguration

### 4. 스프링 부트 프로젝트를 생성하기 위한 다른 방법

Four Ways

Create a maven project and add the starter dependencies

User the Spring\_INITIALIZER <https://start.spring.io>

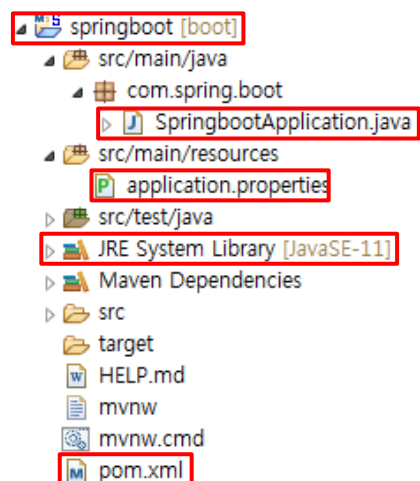
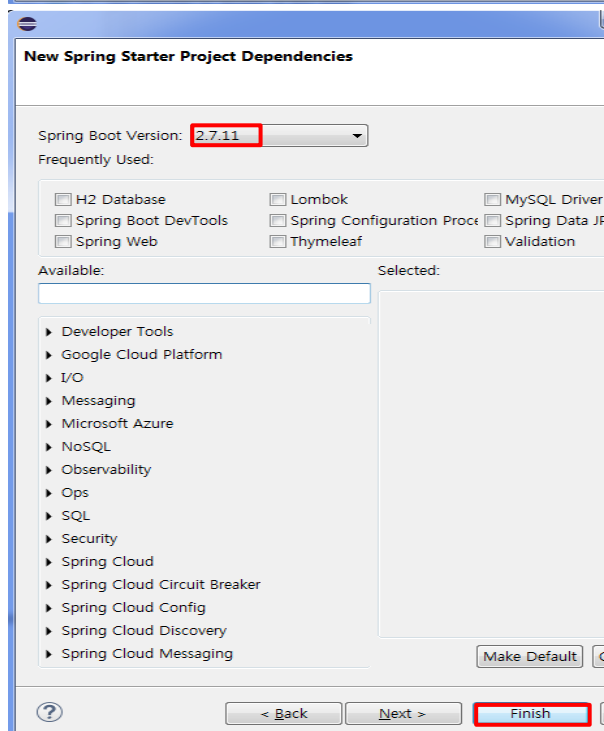
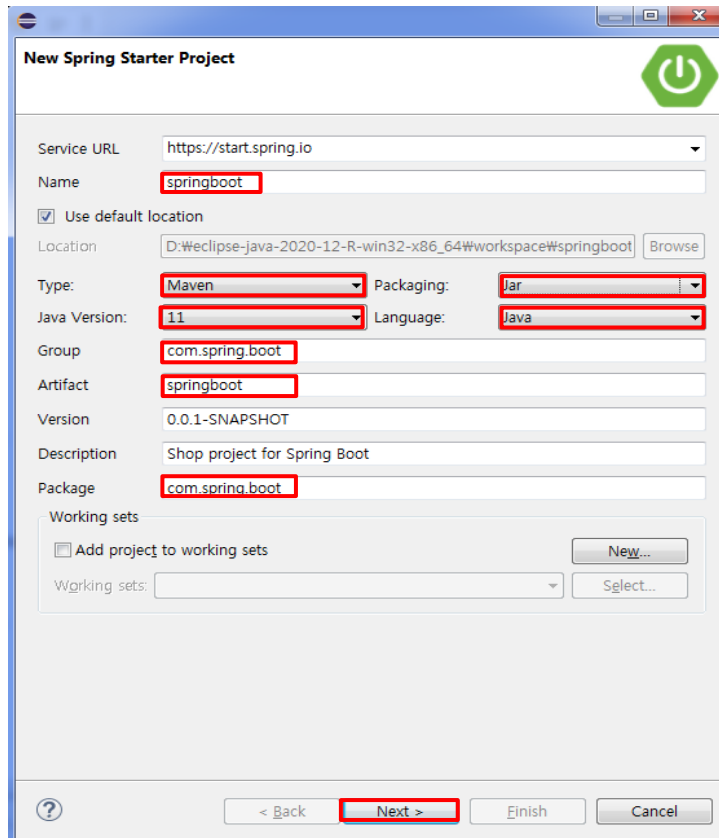
Using and IDE

Using Spring Boot CLI

## 5. 스프링 부트 애플리케이션 만들기

### 가. 스프링 부트 버전 선택

- <https://medium.com/sjk5766/spring-boot-%EB%B2%84%EC%A0%84%EC%97%90-%EB%94%B0%EB%A5%B8-java-%EB%B2%84%EC%A0%84-ff15c5ba7ecb>
- 스프링 부트 3.x 버전에선 Java 17 이상 버전을, 2.x 버전에선 자바 11을 쓰면 될 것 같다.



## 6. DAO와 서비스 클래스 만들기

springboot [boot]

- src/main/java
  - com.spring.boot
    - SpringBootApplication.java
  - com.spring.boot.dao
    - Dao.java
- src/main/resources
- src/test/java
- JRE System Library [JavaSE-11]
- Maven Dependencies
- src
- target
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml

SpringBootApplication.java    Dao.java

```
1 package com.spring.boot.dao;
2
3 public class Dao {
4
5     public void create()
6     {
7         System.out.println("Created");
8     }
9
10 }
```

Dao.java    Service.java

```
1 package com.spring.boot.service;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Component;
5
6 import com.spring.boot.dao.Dao;
7
8 @Component
9 public class Service {
10
11     Dao dao;
12
13     @Autowired
14     public Service(Dao dao) ⇒ 생성자 주입
15     {
16         this.dao = dao;
17     }
18
19     public void save()
20     {
21         dao.create();
22     }
23
24 }
```

- springboot [boot]
- src/main/java
  - com.spring.boot
    - SpringBootApplication.java
  - com.spring.boot.dao
    - Dao.java
  - com.spring.boot.service
    - Service.java

Dao.java    Service.java

```
1 package com.spring.boot.dao;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class Dao {
7
8     public void create()
9     {
10         System.out.println("Created");
11     }
12
13 }
```

## 7. 스프링 테스트 소개

The screenshot illustrates the setup and execution of a Spring Boot application with tests. It shows the following components:

- Dao.java:** A Spring component with a constructor that prints "Dao bean created" and a `create()` method. The constructor is highlighted with a red box.
- Service.java:** A Spring component that is `@Autowired` and uses the `Dao` to implement a `save()` method. The `@Autowired` annotation and the constructor are highlighted with a red box.
- SpringbootApplicationTests.java:** A test class with `@SpringBootTest` and a `@Test` method `contextLoads()`. Both annotations are highlighted with red boxes.
- Project Structure:** The IDE's project explorer shows the package structure, with `SpringbootApplicationTests.java` highlighted in red.
- JUnit Runner:** The bottom panel shows the JUnit runner interface. The "Errors" count is 0, and the "Run As" button is highlighted with a red box.
- Console Output:** The console shows the output of the test run, including "Spring Boot :: (v2.7.11)" and the log messages "Dao bean created" and "Service Bean Created.", which are highlighted with a red box.

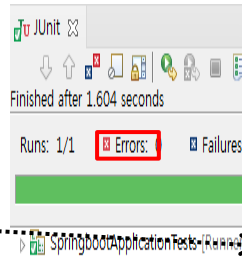
## 8. 애플리케이션 테스트



```

SpringbootApplicationTests.java Dao.java Service.java
1 package com.spring.boot;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.context.SpringBootTest;
6 import org.springframework.context.ApplicationContext;
7
8 import com.spring.boot.service.Service;
9
10 @SpringBootTest
11 class SpringbootApplicationTests {
12
13     // @Test
14     // void contextLoads() {
15     // }
16
17     @Autowired
18     ApplicationContext context;
19
20     @Test
21     void testService() {
22         Service service = context.getBean(Service.class);
23         service.save();
24     }
25 }

```



```

:: Spring Boot ::
(v2.7.11)

```

```

2023-04-26 04:55:51.034 INFO 6596 --- [
2023-04-26 04:55:51.035 INFO 6596 --- [
Dao bean created
Service Bean Created.
2023-04-26 04:55:51.575 INFO 6596 --- [
Created

```