百見不如一打
백견불여일타

이젠 프로젝트다!

# 스프링 부트
## 쇼핑몰 프로젝트
## with JPA

# 百見不如一打

코드를 한번 쳐보고 실행해보는 것이
프로그래밍을 익히는 으뜸 공부법이라는
철학을 담았습니다.

## 지은이의 말

처음으로 이커머스에 관심을 갖게 된 것은 취업 준비를 하던 때였습니다. 커머스 관련 회사의 공고를 보고 지원하여 백엔드 개발자 과제 준비를 한 것이 계기입니다. 스프링 부트로 회원 가입, 로그인, 상품 구매 기능 등 작은 쇼핑몰을 만들어서 제출하는 것이 과제였죠.

그 당시에는 웹 개발 자체에도 익숙하지 않았고, 스프링 부트를 사용해 본 것도 처음이었습니다. 결과는 물론 탈락이었죠. 하지만 과제를 준비하는 동안 MVC 패턴, 데이터베이스, HTML, CSS, 자바스크립트 등 웹 애플리케이션 개발을 위한 기본기를 많이 배울 수 있었습니다. 그러다 보니 이커머스 개발 자체에 흥미가 생겼고 정신을 차리고 보니 이커머스 업계에 발을 담그게 되었습니다.

주니어 개발자일 때, 스프링 프레임워크를 공부하기 위해서 몇 백 쪽이나 되는 스프링 프레임워크 책만 붙들고 공부했습니다. 이제 와서 생각해 보면 막 입문하는 초보 개발자가 이해도 못 하는 책을 붙잡고 씨름하던 것은 꽤 비효율적이었습니다. 프레임워크를 배우기 위해서 가장 좋은 방법은 일단 작은 서비스를 만들어 보는 것이 가장 빠르고 확실한 방법이었음을 깨닫게 됩니다.

서비스를 만들어서 직접 배포하고 사람들에게 홍보도 하다보면 개발의 즐거움을 느끼실 수 있을 것입니다. 그렇게 즐거움을 느끼다 보면 스스로 더 깊게 공부를 하고 싶다는 욕심이 생깁니다.

이 책은 제가 주니어 개발자일 때부터 지금까지 쌓아온 노하우들을 담았습니다. 기본 예제를 통해서 Thymeleaf, Spring Data JPA의 기본 사용법을 익히고 스프링 부트 위에서 상품, 주문, 장바구니 도메인 로직을 구현해보는 구성입니다. 끝까지 따라가 보신다면 앞으로 다른 프로젝트를 시작할 때 기반이 될 수 있으며, 이커머스에 관심이 많은 초보 개발자 여러분의 포트폴리오 제작에도 도움이 될 것이라고 생각합니다.

기회가 된다면 출판을 해봐야겠다는 생각을 실제로 이룰 수 있게 도와주시고, 작성한 원고를 직접 따라해 보시면서 꼼꼼하게 피드백 주신 임성춘 편집장님께 감사 인사를 드립니다. 그리고 원고를 만들면서 교정을 정말 많이 했는데 꼼꼼하게 반영해주신 조서희 편집자님께도 감사 인사를 드립니다.

오픈마켓 개발 프로젝트 와중에도 원고를 피드백해주신 김지영, 정선민 선임 개발자에게 고마운 마음을 전하고 싶습니다. 마다하지 않고 베타 리뷰에 참여해 주신 권샘찬 님에게도 감사 인사를 드립니다. 마지막으로, 책이 나올 때까지 격려와 응원을 해주신 가족들에게 감사를 전합니다.

## 베타테스터의 말

이커머스는 코로나를 등지고 자타공인 핫이슈로 떠올랐습니다. 이커머스에 도전하는 주니어 개발자들은 어디부터 이커머스를 공부해야 할지 갈피를 잡기가 어려울 것 같습니다. 이 책은 바로 그런 분들의 이정표입니다. 가장 대중적인 스프링 부트부터 강력한 데이터 관리를 위한 JPA, 어려운 로그인을 쉽게 구현할 수 있는 스프링 시큐리티, 쉽고 효과적이고 경제적으로 유지보수할 수 있는 Thymeleaf까지, 꼭 필요하고 대중적으로 기업에서 활용하는 최신 기술을 이커머스 업무 흐름과 함께 담아낸 것이 이 책의 장점입니다. 내용을 꼼꼼히 읽어보면서 예제를 꼭 '직접' 코딩하는 것이 이 책을 이해하는 핵심입니다. 끝까지 모두 학습하시고 이커머스의 세계로 들어오세요. 저희는 언제나 환영합니다!

**에스에스지닷컴_김지영**

실무에서 개발을 하다 보면 비즈니스 로직에만 집중해 내가 개발하고 있는 프로젝트의 구성 또는 설정을 잘 들여다 보지 않게 됩니다. 이 책을 읽다 보면 프로젝트 패키지 구성부터 메이븐을 통한 빌드 구성과 같이 기초부터 시작해 실무에서는 몰랐던 내용들을 배울 수 있습니다. JPA와 Thymeleaf의 기본적인 설명부터 실습까지 담겨 있어 초보자도 쉽게 따라 할 수 있습니다. 저자는 대형 온라인 쇼핑몰 서비스를 개발 · 운영하며 커머스에서 필요한 데이터 모델을 익히 알고 있습니다. 그 경험과 노하우가 이 책에 녹여 있다고 생각합니다. 실습을 모두 따라 하면 흔히 볼 수 있는 쇼핑몰의 기능을 모두 구현할 수 있어서 재미도 느낄 수 있을 겁니다. 스프링 부트를 사용하면서 재미있게 익히고 싶은 분들께 꼭 한번 읽어보시길 추천합니다.

**에스에스지닷컴_정선민**

스프링을 공부하려고 책을 고르다 보면 너무 깊게 들어가서 손대기를 포기하게 되거나 혹은 너무 간단해서 많은 도움이 되지 않는 책이 대부분입니다. 이 책은 꼭 필요한 개념을 적절히 학습하면서 쇼핑몰까지 구현해볼 수 있다는 점이 커다란 매력입니다. 현업에서 처음 개발을 해보는 주니어 개발자 분들은 개념은 알지만 실제로 어떻게 구현해야 할지 고민을 많이 하게 되는데, 스프링 부트의 기본 개념부터 JPA, Thymleaf, Test(Junit) 등 여러 가지 개념이 현업에서 실제로 어떻게 사용되는지 배울 수 있습니다. 기본적인 개념을 설명한 뒤 구현된 코드를 보여주고 그 코드에 개념이 어떻게 녹아 있는지 설명하는 방식이기 때문에 처음 프로젝트를 시작하시는 개발자 분들도 쉽게 학습할 수 있을 것입니다.

**키위스튜디오_권샘찬**

## 일러두기

### 프로젝트 개발 환경

1. 운영체제: Window 10
2. 통합개발환경(IDE): 인텔리제이(IntelliJ)
3. JDK 버전: JDK 11
4. 스프링 부트 버전: 2.5.2
5. 데이터베이스: MySQL
6. 빌드 툴: 메이븐

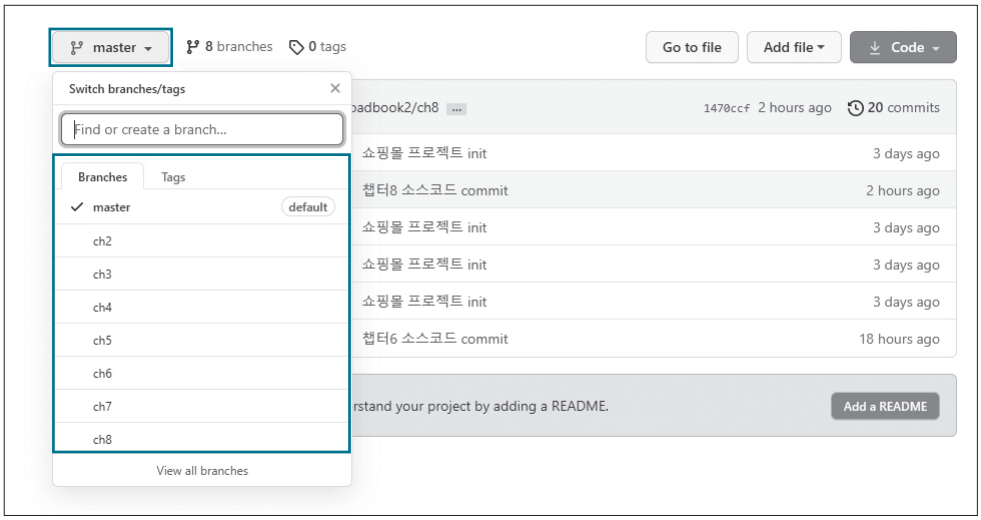
### 이 책에서 구현하는 최종 프로젝트 모습

상품등록, 상품 관리, 장바구니, 구매이력, 로그인/로그아웃 등 쇼핑몰 프로젝트를 스프링부트 & JPA 기반으로 만들어본다.


### 예제 다운로드

책에서 진행하는 예제의 소스코드는 다음 깃허브 주소에서 확인할 수 있습니다. 장별로 브랜치를 만들어 두었으니 해당 장의 소스코드를 이용하려면 해당 브랜치로 이동한 후 참고해 주십시오. 6, 7, 8장은 뷰를 만들기 위한 HTML 코드 내용이 길고, 지면에서 긴 코드를 확인하기에 가독성의 한계가 있습니다. 소스코드를 복사해서 인텔리제이 에디터를 통해 책의 설명을 함께 보시기를 권합니다.

- 깃허브 주소: https://github.com/roadbook2/shop




### 따라하다 잘 안 되면 깃허브 소스로 문의하기

장별로 소스코드를 깃허브에 올려두었으니 해당 소스와 먼저 비교해서 확인해 주십시오.

1단계: 자신의 깃허브 페이지를 만듭니다.

2단계: 인텔리제이에 자신의 깃허브 페이지를 등록합니다(VCS 〉 Get from Version Control 메뉴).

3단계: 프로젝트를 커밋한 후 푸시합니다.

4단계: 자신의 깃허브 URL을 문제점과 함께 저자에게 문의합니다.

(자세한 절차는 백견 카페를 확인합니다)

### 이 책의 주요 특징

[함께 해봐요] 예제는 단계별로 따라할 수 있게 순서대로 진행했습니다. 괄호 안에 어느 위치에 소스 파일을 만들어야 하는지 구체적으로 명시했고, 중요한 소스는 하단에 별도로 설명하였습니다.

[함께 해봐요 3-24] Thymeleaf 페이지 레이아웃 예제: thymeleaf 파일 만들기
resources/templates/thymeleafEx07.html

```
01 <!DOCTYPE html>
02 <html xmlns:th="http://www.thymeleaf.org"
03       xmlns:layout=http://www.ultraq.net.nz/thymeleaf/layout
04       layout:decorate="~{layouts/layout1}">  ❶
05
06     <div layout:fragment="content">  ❷
07         본문 영역 입니다.
08     </div>
09
10 </html>
```

❶ layouts 폴더 아래에 있는 layout1.html을 적용하기 위해서 네임스페이스를 추가합니다.
❷ layout1.html 파일의 〈div layout:fragment="content"〉 영역에 들어가는 영역입니다.

중요한 메소드는 따로 참고할 수 있도록 별도의 표로 정리를 해두었습니다만, 자세한 내용은 해당 도큐먼트를 참고하기를 바랍니다.

[표 2-5] JPAQuery 데이터 반환 메소드

| 메소드 | 기능 |
|---|---|
| List<T> fetch() | 조회 결과 리스트 반환 |
| T fetchOne | 조회 대상이 1건인 경우 제네릭으로 지정한 타입 반환 |
| T fetchFirst() | 조회 대상 중 1건만 반환 |
| Long fetchCount() | 조회 대상 개수 반환 |
| QueryResult<T> fetchResults() | 조회한 리스트와 전체 개수를 포함한 QueryResults 반환 |

각 단계마다 주요 내용과 주의사항을 언급하며 시작하여 독자가 학습하면서 실수를 줄일 수 있도록 배려하였습니다.

## 2.7 Spring DATA JPA Querydsl

@Query 어노테이션을 이용한 방법에도 단점이 있습니다. @Query 어노테이션 안에 JPQL 문법으로 문자열을 입력하기 때문에 잘못 입력하면 컴파일 시점에 에러를 발견할 수 없습니다. 에러는 가능한 빨리 발견하는 것이 가장 좋습니다. 이를 보완할 수 있는 방법으로 Querydsl를 알아보겠습니다.

### 백견불여일타 카페에서 함께 공부합시다

백견불여일타 시리즈는 "만들어 보는 것만이 학습의 가장 빠른 지름길"이라는 콘셉으로 만들어진 실습 위주의 책입니다. HTML5와 안드로이드 앱 개발에서 없어서는 안 될 파이어베이스, C#, Vue.js, 파이썬 생활밀착형 프로젝트 등의 다양한 주제로 많은 독자분들이 백견불여일타 카페에서 도움을 받고 있습니다. 외롭게 홀로 고군분투하며 어렵게 학습하는 입문자들에게 힘이 되는 공간으로 발전시켜 나가겠습니다.

백견불여일타 네이버 카페 주소 : **cafe.naver.com/codefirst**

# 목차

## 3장 Thymeleaf 학습하기

## 4장 스프링 시큐리티를 이용한 회원 가입 및 로그인



## 5장 연관 관계 매핑

## 6장 상품 등록 및 조회하기

## 7장 주문



## 8장 장바구니

# 1장

# 개발 환경 구축

## 학습목표

1. 웹 애플리케이션 개발을 위해서 사용하는 스프링부트의 특징을 알아본다.
2. 개발 환경 구축을 위해서 자바, 인텔리제이, MySQL을 설치한다.

# 1.1 스프링 부트의 특징

이 책에서는 스프링 부트를 이용하여 쇼핑몰 애플리케이션을 만들기 위해 여러가지 예제를 진행합니다. 이를 통해 웹 서비스 개발을 경험해 볼 수 있도록 구성했습니다. 실무에서도 신규 프로젝트는 모두 스프링 부트 기반으로 만들고 있습니다. 스프링 부트의 특징을 먼저 알아봅니다.

스프링 프레임워크를 사용하다가 스프링 부트를 사용하게 되면 정말 많은 것들이 편리합니다. 스프링 부트는 기존의 스프링 프레임워크보다 더 봄이 왔다는 뜻으로 '스프링 부트'라는 이름이 지어졌습니다. 스프링 프레임워크는 '설정이 절반'이라는 말이 있을 정도로 복잡합니다. 스프링 부트는 최소한의 설정으로, 실행 버튼을 누르면 바로 애플리케이션이 실행되는 독립 실행 애플리케이션을 지향합니다.

스프링 부트의 특징을 정리해보면 다음과 같습니다.

1. 내장 서버를 이용해 별도의 설정 없이 독립 실행이 가능한 스프링 애플리케이션
2. 톰캣, 제티 또는 언더토우와 같은 웹 애플리케이션서버(WAS) 자체 내장
3. 빌드 구성을 단순화하기 위한 'Spring Boot Starter' 의존성 제공
4. XML 설정 없이 단순 자바 수준의 설정 방식 제공
5. JAR를 이용해 자바 옵션만으로 배포 가능
6. 애플리케이션의 모니터링과 관리를 위한 스프링 액추에이터 제공

## 톰캣, 제티, 언더토우와 같은 웹 애플리케이션서버(WAS) 자체 내장

스프링 부트는 디폴트 내장 서버로 톰캣Tomcat을 사용하고 있습니다. 내장 웹 서버에 대한 설정을 자동으로 처리하기 때문에 스프링 부트 사용자는 웹 서버와 관련된 설정을 하지 않아도 프로젝트 내부에 포함하게 됩니다. 제티Jetty나 언더토우Undertow와 같은 내장 웹 서버를 사용하기 위해서는 pom.xml에 설정 값을 작성하는 것만으로 쉽게 변경이 가능합니다.

## 빌드 구성을 단순화하기 위한 '스프링 부트 스타터' 의존성 제공

스프링 부트에서 스타터stater란 설정을 자동화해주는 모듈을 의미합니다. 프로젝트에서 설정해야 하는 다양한 의존성을 사전에 미리 정의해서 제공합니다. 따라서 프로젝트에서 설정해야 하는 다수의 의존성들을 스타터가 이미 포함하고 있기 때문에 스타터에 대한 의존성만 추가하면 프로젝트를 쉽게 진행할 수 있습니다.

## XML 설정 없이 단순 자바 수준의 설정 방식 제공

스프링 부트는 XML에 설정을 작성할 필요 없이 자바 코드로 설정할 수 있습니다. XML은 문법이 틀리거나 선언이 선언을 잘못하면 원인을 찾기 힘듭니다. 자바 코드는 컴파일러의 도움을 받기 때문에 오타 등의 설정 정보 오류를 미리 알 수 있습니다. 또한 클래스 단위로 설정하기 때문에 쉽게 관리할 수 있습니다.

## 애플리케이션의 모니터링과 관리를 위한 스프링 액추에이터 제공

서비스를 운영하려면 서비스 개발뿐 아니라 서비스가 정상적으로 동작하고 있는지 상태를 모니터링해야 합니다. 스프링 액추에이터Spring Actuator는 스프링 부트 애플리케이션에서 제공하는 여러 가지 정보를 손쉽게 모니터링 할 수 있도록 도와주는 라이브러리입니다.  배포한 기능이 장애가 있는지 모르는 상태로 몇 달이 흐르고 그로인해 회사가 큰 손실을 얻는다면 그때 겪을 상실감은 상당합니다!

# 1.2 JDK 설치

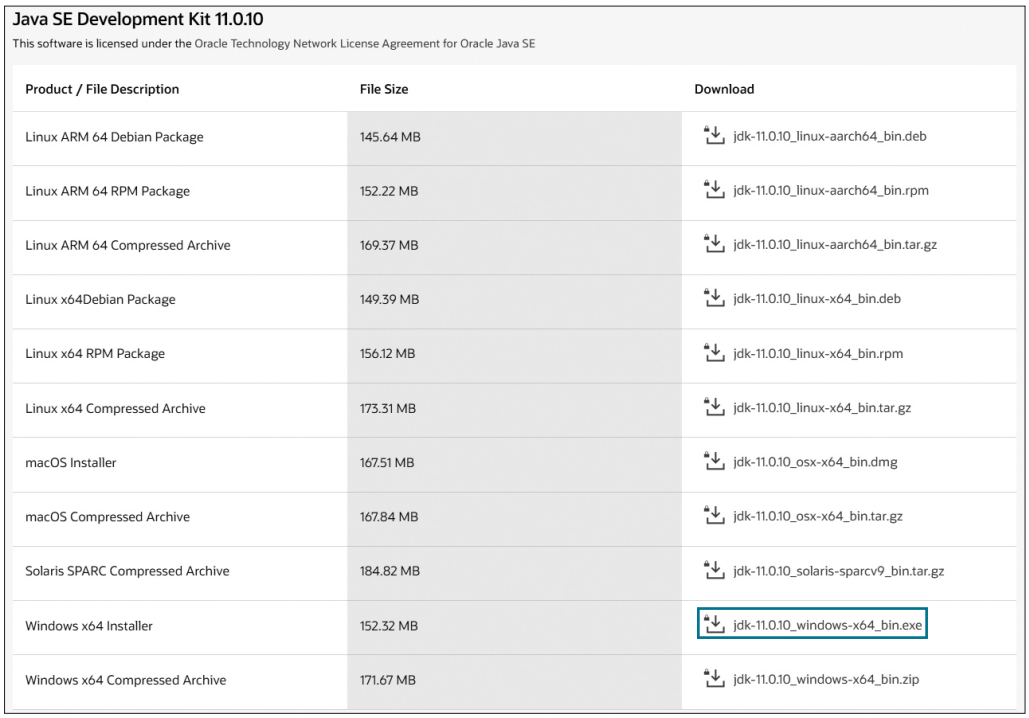
자바 애플리케이션을 만들기 위해서 JDK(Java Development Kit)를 설치하겠습니다. JDK는 자바 환경에서 돌아가는 프로그램을 개발하는 데 필요한 툴을 모아놓은 소프트웨어 패키지입니다.

우리가 진행할 쇼핑몰 프로젝트의 개발 환경을 소개합니다. 빌드 툴은 범용적으로 많이 사용하는 메이븐Maven으로 진행하겠습니다. JDK가 이미 깔려있다면 1.8 이상의 버전을 사용하시면 됩니다. 인텔리제이 설치 후 자바의 환경 변수 수정 및 인텔리제이에서 프로젝트의 SDK 버전을 변경하시면 됩니다.

프로젝트를 진행하면서 사용할 개발 환경입니다.

1. 운영체제: Window 10
2. 통합개발환경(IDE): 인텔리제이(IntelliJ)
3. JDK 버전: JDK 11
4. 스프링 부트 버전: 2.5.2
5. 데이터베이스: MySQL
6. 빌드 툴: 메이븐

자바 버전은 JDK 11을 사용하겠습니다. https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html 사이트로 접속해 자신의 운영체제에 맞는 JDK 설치 파일을 다운로드한 후 실행해 설치를 완료합니다. 해당 주소는 시간이 지나면 변경될 수 있습니다. 그때는 오라클 사이트에 접속하여 JDK를 내려받으면 됩니다.



**Java SE Development Kit 11.0.10**

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

| Product / File Description | File Size | Download |
|---|---|---|
| Linux ARM 64 Debian Package | 145.64 MB | jdk-11.0.10_linux-aarch64_bin.deb |
| Linux ARM 64 RPM Package | 152.22 MB | jdk-11.0.10_linux-aarch64_bin.rpm |
| Linux ARM 64 Compressed Archive | 169.37 MB | jdk-11.0.10_linux-aarch64_bin.tar.gz |
| Linux x64Debian Package | 149.39 MB | jdk-11.0.10_linux-x64_bin.deb |
| Linux x64 RPM Package | 156.12 MB | jdk-11.0.10_linux-x64_bin.rpm |
| Linux x64 Compressed Archive | 173.31 MB | jdk-11.0.10_linux-x64_bin.tar.gz |
| macOS Installer | 167.51 MB | jdk-11.0.10_osx-x64_bin.dmg |
| macOS Compressed Archive | 167.84 MB | jdk-11.0.10_osx-x64_bin.tar.gz |
| Solaris SPARC Compressed Archive | 184.82 MB | jdk-11.0.10_solaris-sparcv9_bin.tar.gz |
| Windows x64 Installer | 152.32 MB | jdk-11.0.10_windows-x64_bin.exe |
| Windows x64 Compressed Archive | 171.67 MB | jdk-11.0.10_windows-x64_bin.zip |

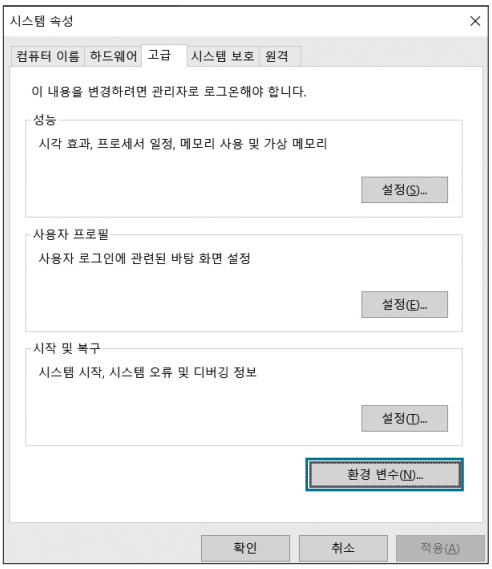
[그림 1-1] JDK 11 다운로드

설치가 끝나면 어떤 경로에서든 자바 명령어를 사용할 수 있도록 환경 변수인 Path에 JDK의 경로를 넣어줍니다.

[제어판] – [시스템 및 보안] – [시스템]에서 오른쪽의 고급 시스템 설정으로 들어갑니다.

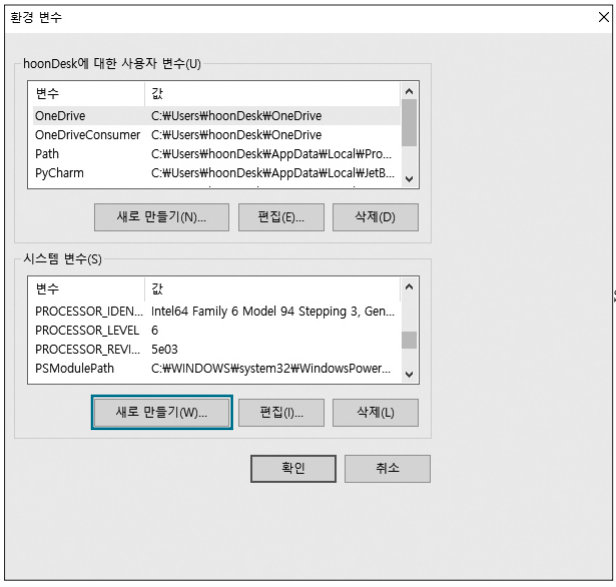

[그림 1-2] 자바 환경 변수 설정 1단계

시스템 속성창에서 〈환경 변수〉 버튼을 클릭합니다.



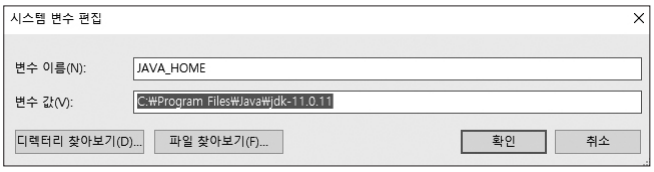

[그림 1–3] 자바 환경 변수 설정 2단계

시스템 변수창에서 〈새로 만들기〉 버튼을 누르면 시스템 변수를 추가할 수 있습니다.



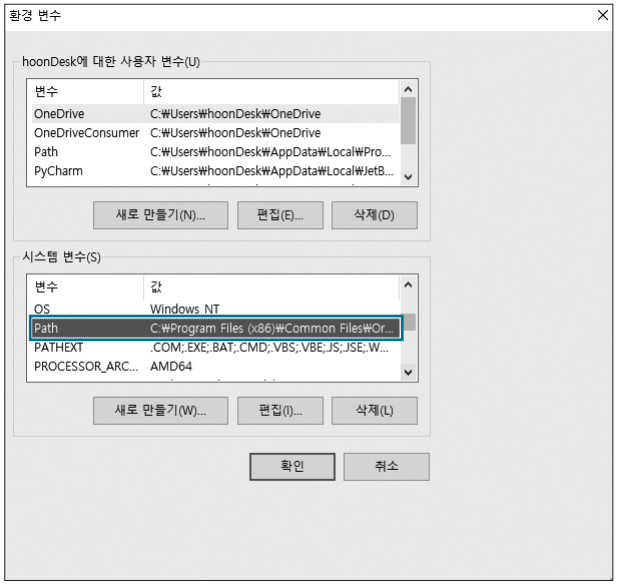

[그림 1–4] 자바 환경 변수 설정 3단계

설치한 JDK 11 디렉토리의 경로를 JAVA_HOME 환경 변수의 변수 값으로 설정합니다.



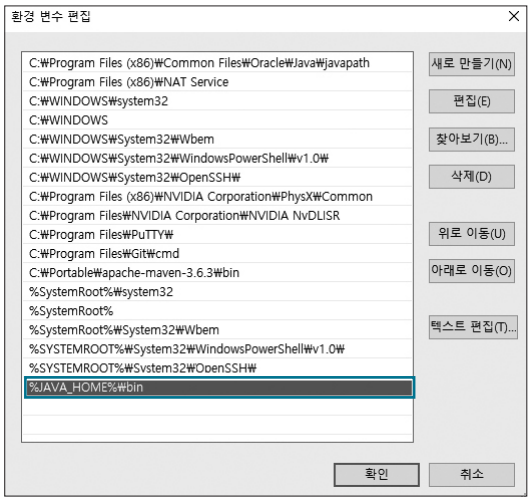

[그림 1–5] 자바 환경 변수 설정 4단계

JAVA_HOME 시스템 변수 추가 후, 시스템 변수 목록에 있는 Path를 더블 클릭합니다.




[그림 1–6] 자바 환경 변수 설정 5단계

오른쪽 상단의 〈새로 만들기(N)〉를 누른 후 [그림 1-7]과 같이 값을 추가합니다.




[그림 1-7] 자바 환경 변수 설정 6단계

명령 프롬프트(CMD)창을 열고 자바가 정상적으로 설치됐는지 버전을 확인합니다.

```
java -version
```

# 1.3 인텔리제이 설치

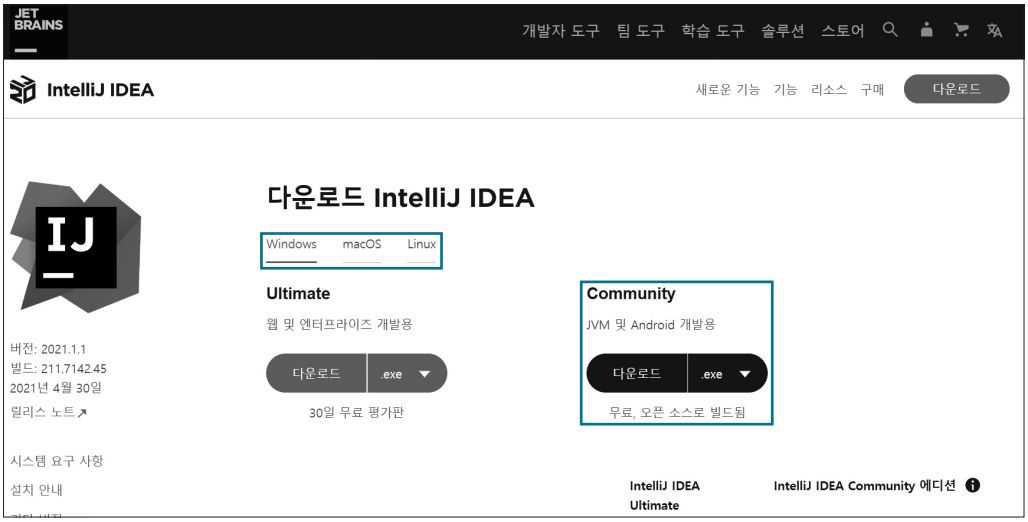
스프링 부트 프로젝트를 생성하기 위한 통합 개발 환경(IDE)으로 젯브레인스(JetBrains)에서 개발한 인텔리제이를 사용하겠습니다. 기존에는 이클립스로 많이 개발했지만, 최근에는 인텔리제이의 인기가 높은 추세입니다.

브라우저에서 https://www.jetbrains.com/ko-kr/idea/download로 접속하면 인텔리제이를 다운로드할 수 있는 화면이 나타납니다. Ultimate 버전과 Community 버전이 있는데, 무료 버전인 Community 버전으로 프로젝트를 진행합니다. Ultimate 버전은 Community 버전에 비해 더 많은 기능을 제공합니다.

| | IntelliJ IDEA Ultimate | IntelliJ IDEA Community 에디션 |
|---|---|---|
| Java, Kotlin, Groovy, Scala | ✓ | ✓ |
| Android | ✓ | ✓ |
| Maven, Gradle, sbt | ✓ | ✓ |
| Git, SVN, Mercurial | ✓ | ✓ |
| 디버거 | ✓ | ✓ |
| 프로파일링 도구 | ✓ | ✗ |
| Spring, Java EE, Micronaut, Quarkus, Helidon 및 그 외 다양한 지원 | ✓ | ✗ |
| Swagger, Open API 사양 | ✓ | ✗ |
| JavaScript, TypeScript | ✓ | ✗ |
| 데이터베이스 도구, SQL | ✓ | ✗ |

[그림 1-8] IntelliJ Community Ultimate 버전 비교

Ultaimate 버전은 30일까지 무료로 사용이 가능하며, 대학생이라면 이메일 인증을 통해 교육 라이선스를 받아 무료로 사용할 수 있습니다. [그림 1-9]와 같이 자신에게 맞는 운영체제를 선택한 후 〈Community 다운로드〉를 클릭합니다.

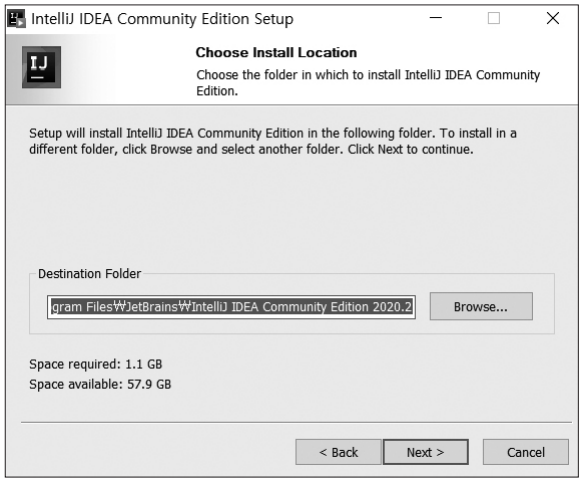



[그림 1-9] IntelliJ Community 버전 다운로드

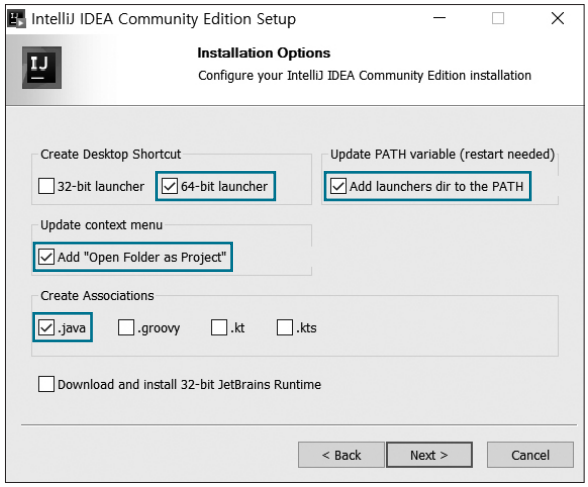
설치 파일을 다운로드하고 실행합니다. 〈Next〉를 클릭합니다.


[그림 1-10] IntelliJ Community 버전 설치 1단계

인텔리제이를 설치할 경로를 선택하고 〈Next〉, 4가지 항목을 선택 후 〈Next〉를 클릭합니다.

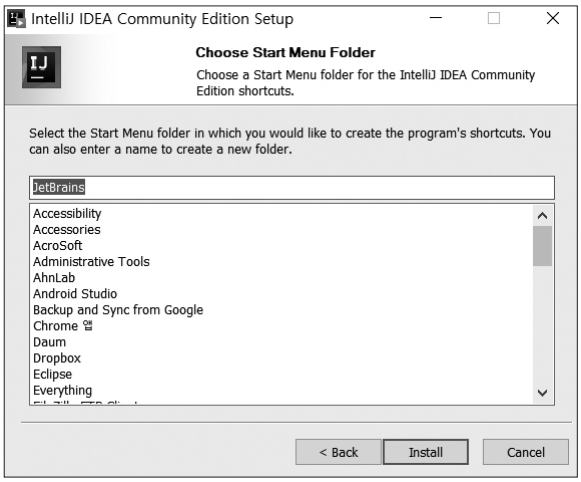



[그림 1-11] IntelliJ Community 버전 설치 2단계




[그림 1-12] IntelliJ Community 버전 설치 3단계

〈Install〉을 클릭한 후 설치를 완료하기 위해서 컴퓨터를 한 번 재부팅합니다.




[그림 1-13] IntelliJ Community 버전 설치 4단계


[그림 1-14] IntelliJ Community 버전 설치 5단계

# 1.4 애플리케이션 실행하기

프로그래밍 언어를 연습할 때 가장 먼저 해보는 "Hello World!"를 출력하는 애플리케이션을 만들어 봅니다.

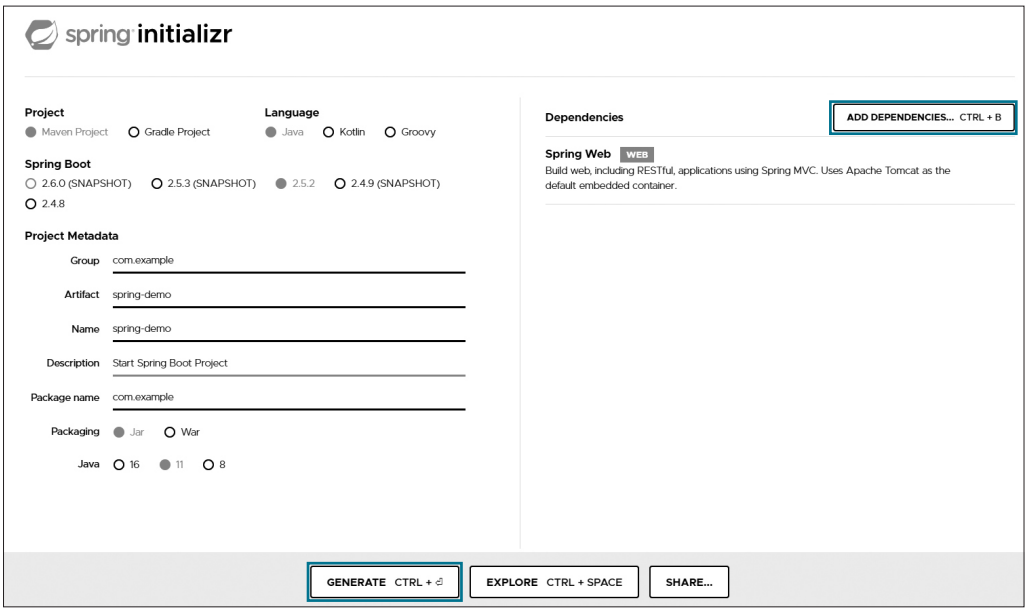
## 1.4.1 Spring Boot Project 생성하기

스프링부트 프로젝트를 쉽게 만들기 위해서 Spring Initializr 사이트에서 템플릿을 다운로드해 임포트하겠습니다. https://start.spring.io/에 접속한 후 프로젝트의 옵션들을 선택합니다. 이니셜라이저는 스프링 프로젝트를 생성할 때마다 여러분의 프로젝트 조건에 맞게 내려 받아서 사용합니다. 이니셜라이저는 애플리케이션에 필요한 의존성을 쉽게 추가할 수 있는 방법을 제공하며 많은 설정을 수행합니다.

앞으로 의존성이라는 말이 계속 등장할 텐데 메이븐Maven에서 의존성을 추가한다는 것은 다른 라이브러리를 사용하기 위해서 추가하는 것이라고 생각하시면 됩니다. 의존성 추가를 위해서 [그림 1-15]와 같이 오른쪽 상단에 〈ADD DEPENDENCIES〉 버튼을 눌러 'Spring Web'을 선택합니다. 스프링 웹에는 웹 애플리케이션을 만들기 위한 각종 라이브러리 정보가 담겨 있습니다. 아티팩트artifact는 일반적으로 소프트웨어 분야에서 개발 프로세스에 의해 생산되는 산출물을 뜻합니다. 아티팩트의 id로 spring-demo를 입력해줍니다. 패키지의 이름은 com.example로 정하겠습니다. 아티팩트의 이름을 수정하면 Package name 뒤에 spring-demo가 자동으로 입력되는데 삭제하겠습니다.

**프로젝트 설정**

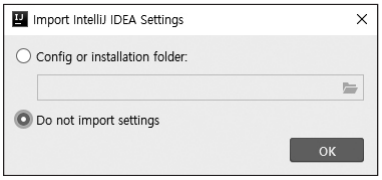
- 빌드 툴 - 메이븐
- 언어 - Java 11
  (기존에 사용하던 Java 1.8 이상의 버전이 있을 경우 해당 버전 선택 가능)
- 스프링 부트 버전: 2.5.2
- 패키징: Jar
- 의존성: Spring Web




[그림 1-15] 스프링부트 프로젝트 생성 1단계

설정 완료 후 〈GENERATE〉 버튼을 누르면 아티팩트의 이름과 같은 spring-demo.zip 파일이 다운로드됩니다. 자신이 사용하는 워크스페이스(작업 공간, 예를 들어 C 드라이브의 test 디렉토리를 프로젝트 디렉토리로 사용하겠다고 하면 C:₩test가 작업공간임)에 알집을 풀어줍니다. 제 작업공간은 C:₩Users₩dksek₩IdeaProjects 입니다. 그리고 이전에 설치했던 인텔리제이를 실행합니다. [그림 1-17]에서 〈Open or Import〉 버튼을 클릭해 spring initializr에서 다운로드한 프로젝트를 열도록 하겠습니다.

처음 인텔리제이를 실행하면 [그림 1-16] 화면이 나타날 수 있는데 'Do not import settings'를 선택하고 〈OK〉 버튼을 클릭합니다.

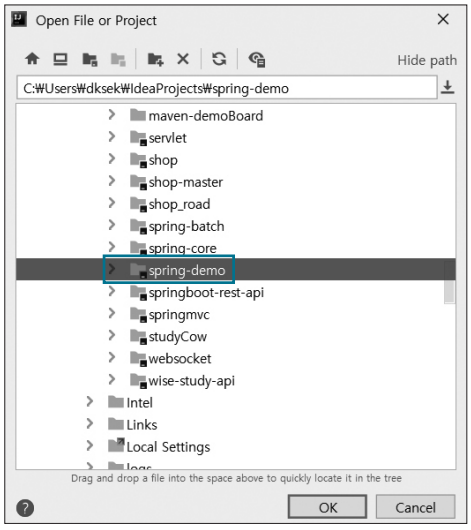



[그림 1-16] 스프링부트 프로젝트 생성 2단계

〈Open or Import〉를 클릭합니다.

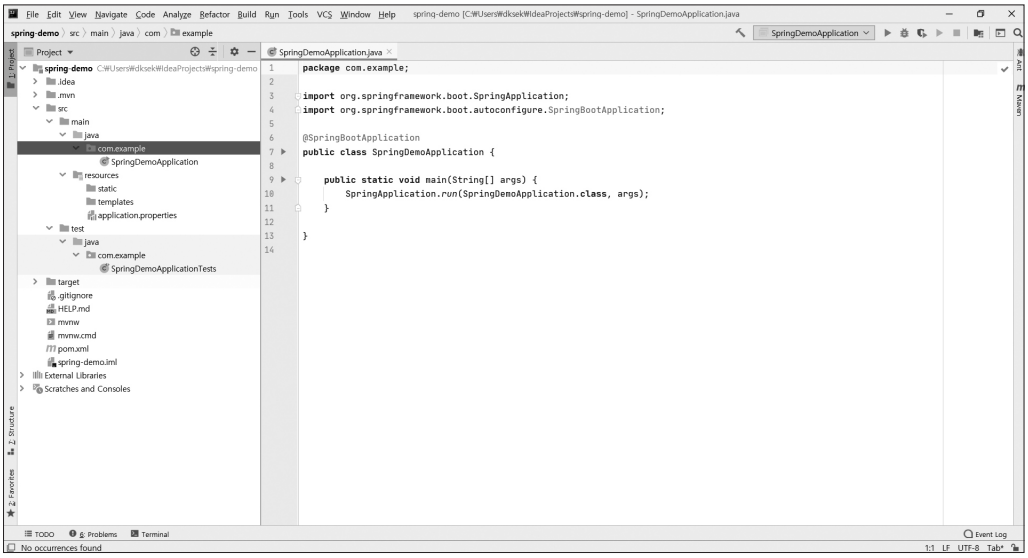

[그림 1-17] 스프링부트 프로젝트 생성 3단계

[그림 1-18]과 같이 알집을 풀어준 폴더를 선택하고 〈OK〉를 선택합니다.



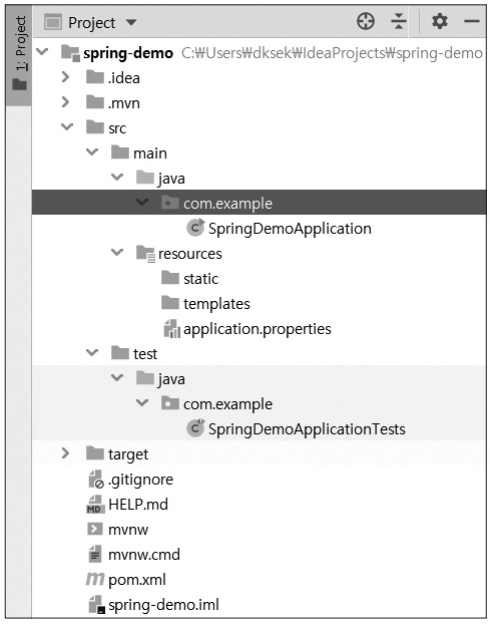

[그림 1-18] 스프링부트 프로젝트 생성 4단계

인텔리제이에서 spring-demo 프로젝트가 띄워진 모습을 확인할 수 있습니다.




[그림 1-19] 스프링부트 프로젝트 생성 완료

생성된 프로젝트의 기본 패키지 구조를 간단히 살펴보겠습니다.




[그림 1-20] 프로젝트 패키지 구조

1: src/main/java패키지 아래에는 자바 소스코드를 작성합니다.
2: src/main/resources 디렉토리 아래에는 HTML, CSS, JS, 이미지 파일 등의 정적 리소스를 저장합니다.
3: 쇼핑몰 제작 프로젝트를 진행하면서 사용할 템플릿 엔진인 thymeleaf는 기본적으로 뷰를 src/main/resources/templates에서 찾습니다. 해당 디렉토리 아래에 HTML 파일들을 작성하고, Controller Class에서 반환한 뷰와 동일한 이름의 html 파일을 찾아서 웹 브라우저에 띄워줍니다.
4: src/test/java 패키지 아래에는 테스트 코드를 작성합니다.

### 1.4.2 빌드 도구

메이븐이란 자바 프로젝트의 빌드를 자동화해주는 빌드 툴입니다. 개발 과정 중에 많은 라이브러리들이 필요한데 pom.xml 파일에 필요한 라이브러리를 적어주면 메이븐이 알아서 네트워크를 통해서 다운로드하고 경로까지 지정해 줍니다. 메이븐 같은 빌드 툴이 없었다면 필요한 jar 파일들을 일일이 받아서 직접 프로젝트에 넣어줬어야 했을 것입니다.

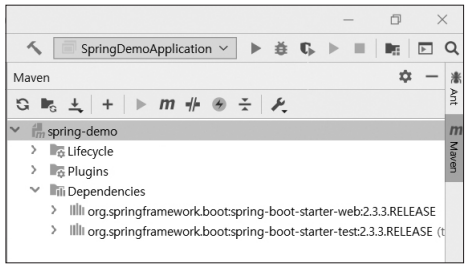
pom.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <project xmlns="http://maven.apache.org/POM/4.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                         https://maven.apache.org/xsd/maven-4.0.0.xsd">
04     <modelVersion>4.0.0</modelVersion>
05     <parent>                                                    1
06         <groupId>org.springframework.boot</groupId>
07         <artifactId>spring-boot-starter-parent</artifactId>
08         <version>2.5.2</version>
09         <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>com.example</groupId>
12     <artifactId>spring-demo</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>spring-demo</name>
15     <description>Start Spring Boot Project</description>
16     <properties>
17         <java.version>11</java.version>
18     </properties>
19     <dependencies>
20         <dependency>                                            2
21             <groupId>org.springframework.boot</groupId>
```

```
22              <artifactId>spring-boot-starter-web</artifactId>
23          </dependency>
24
25          <dependency>                                                    ❸
26              <groupId>org.springframework.boot</groupId>
27              <artifactId>spring-boot-starter-test</artifactId>
28              <scope>test</scope>
29          </dependency>
30      </dependencies>
31
32      <build>
33          <plugins>
34              <plugin>
35                  <groupId>org.springframework.boot</groupId>
36                  <artifactId>spring-boot-maven-plugin</artifactId>
37              </plugin>
38          </plugins>
39      </build>
40
41  </project>
```

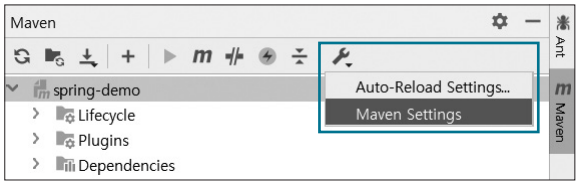
❶ 스프링부트 최상위 모듈로서 스프링부트에 필요한 의존성(dependency)를 자동으로 추가
❷ 웹 애플리케이션에 필요한 라이브러리
❸ Spring Test Framework 라이브러리

[그림 1-21]과 같이 인텔리제이 오른쪽에 [Maven]이라는 항목을 볼 수 있습니다. [Maven] 탭을 누른 후 'Dependencies'를 클릭하면 pom.xml에 추가한 'spring-boot-starter-web'과 'spring-boot-starter-test' 의존성이 들어와 있는 것을 확인할 수 있습니다.



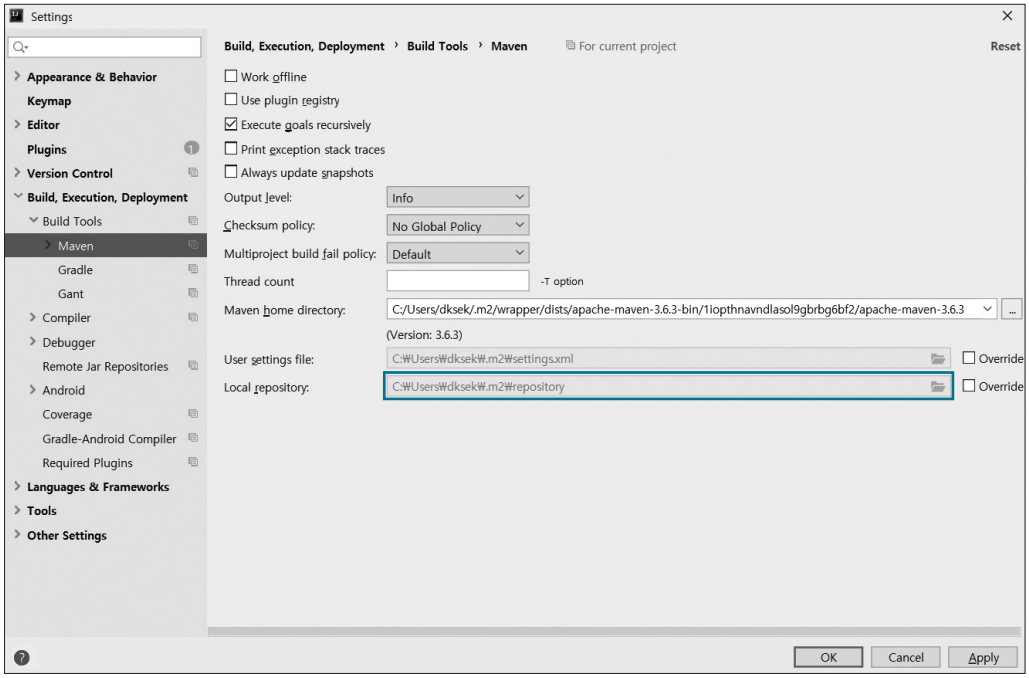

[그림 1-21] maven dependency 확인

이렇게 메이븐을 통해서 받은 파일들은 Local repository에 저장됩니다. 직접 경로를 확인해보기 위해서 'Build Tools Settings'를 클릭합니다. 그러면 2가지의 옵션이 나오는데 [Maven Settings]를 선택하겠습니다.



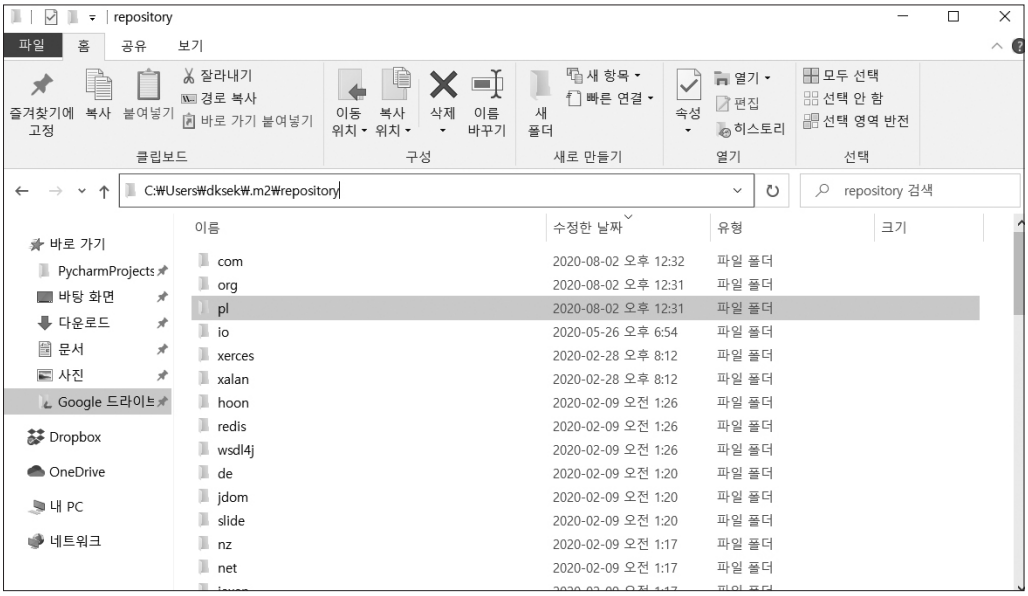

[그림 1-22] maven settings

[그림 1-23]과 같이 자신의 컴퓨터의 Maven Local repository 경로를 확인할 수 있습니다.




[그림 1-23] local maven repository 위치 확인

실제 저 경로로 들어가보면 메이븐을 통해 설치한 라이브러리들이 저장돼 있음을 볼 수 있습니다. 여러 가지의 프로젝트를 동시에 진행하면 메이븐의 의존성이 서로 꼬일 수 있으므로 프로젝트별로 다른 폴더를 Local repository를 지정하기를 권합니다.




[그림 1-24] local maven repository

### 1.4.3 설정 파일(application.properties)

스프링 부트 애플리케이션 실행 시 사용하는 여러 가지 설정값들을 정의하는 파일입니다.

'src/main/resources' 폴더 아래에 자동으로 생성되며 바로 설정 파일로 이용이 가능합니다. 만약 자동으로 생성되지 않았다면 직접 생성해주셔도 됩니다.

개발 환경, 테스트 환경, 운영 환경에 따라서 연결해야 할 데이터베이스, port, debug level 등을 나눠야 한다면 다음 명명 규칙으로 설정 파일을 만듭니다.

```
application-{profile}.properties
```

가령, 개발 환경의 설정 파일은 application-dev.properties로 만들고 운영 환경의 설정 파일은 application-prod.properties로 만듭니다. 실행되는 환경에 따라서 어떤 설정 파일을 사용할지를 jar 파일 실행 시 VM 옵션 등을 통해 지정할 수 있습니다.

또한 application.properties에 설정해 둔 값을 자바 코드에서 사용해야 한다면 @Value 어노테이션을 통해서 읽어올 수 있습니다.

### [함께 해봐요 1-1] application.properties 설정하기

```
01  server.port = 80                    ①
02  application.name = spring-demo      ②
```

① 애플리케이션의 실행할 포트를 설정합니다. 따로 설정하지 않으면 기본 포트는 8080입니다. 80포트는 url 뒤에 포트 번호를 생략할 수 있습니다.

② 애플리케이션의 이름을 'spring-demo'로 설정했습니다. 설정해둔 애플리케이션의 값을 읽어와서 자바 코드에서 사용해야 하면 @Value 어노테이션을 통해서 읽어올 수 있습니다.

**참고 | 어노테이션이란?**

어노테이션(Annotation)은 주석이라는 사전적 의미가 있습니다. JDK5부터 등장했으며 메타데이터(데이터를 위한 데이터)라고도 불립니다. 클래스나 메소드, 변수 등을 선언할 때 '@'를 붙여서 사용합니다. 어노테이션은 컴파일러에게 정보를 알려주거나, 실행할 때 별도의 처리가 필요할 때 등 매우 다양한 용도로 사용할 수 있습니다. 자바에서 가장 쉽게 볼 수 있는 어노테이션의 예시로 @Override가 있습니다. 해당 메소드가 부모 클래스에 있는 메소드를 오버라이드했다는 것을 컴파일러에게 알려줍니다. 만약 메소드를 제대로 오버라이드하지 않았다면 에러가 발생합니다.

애플리케이션 설정 파일을 만드는 다른 방법으로 application.yml 파일을 사용할 수 있습니다. application.properties 파일과 비교했을 때 들여쓰기를 통해 설정 값들을 계층 구조로 관리할 수 있기 때문에 가독성을 향상시킬 수 있다는 장점이 있습니다.

```
server:
  port: 9091
```

단점으로는 문법이 좀 더 엄격하다는 것입니다. 예를 들어서 콜론 다음에 값을 쓸 때 공백이 한 칸 있어야 해당 설정이 정상적으로 동작합니다. 띄어쓰기를 잘못했을 때는 예제가 정상적으로 동작하지 않을 수 있어서 원활한 예제 진행을 위해서 application.properties 파일로 진행하도록 하겠습니다.