Gradle 사용하기

#01.자바 빌드 자동화 도구

1. 빌드

소스코드를 실행 가능한 형태로 만들기 위해서는 '코딩--> 컴파일 --> 테스트 -->패키징(jar)' 의 단계가 필요함.

- 빌드는 소스 코드를 실행 가능한 프로그램으로 변환하는 과정임.
- 컴파일, 테스트, 패키징 등의 단계를 포함함.
- 개발자가 작성한 코드를 **배포 가능한 형태**로 만드는 작업임.

원래는 컴파일과 패키징을 위한 명령어를 각각 CLI에 직접 입력해야 하고 이 명령어는 수많은 옵션을 포함하고 있다.

대부분의 IDE는 이러한 명령어를 버튼이나 단축키 하나로 일관 처리하는 기능을 포함하고 있다.

> ex : VSCODE에서의 'F5'

2. JAVA 빌드 자동화 도구

- Java 빌드 자동화 도구는 컴파일, 테스트, 배포 등을 자동화함.
- 반복 작업을 줄이고 일관된 빌드 환경 제공함.
- 대표 도구로는 **Maven**, **Geadle**, Ant 등이 있음.
  -Maven과 Gradle은 라이브러리의 다운로드 및 설정을 자동화하는 기능을 포함.

1) Maven (전통적인 도구)

- Maven은 Java용 빌드 자동화 및 프로젝트 관리 도구임.
- XML(작성할때쓰는 언어) 기반의 `pom.xml` 파일로 프로젝트 구성 정의함.
  -이 파일에 사용하고자 하는 라이브러리 정보를 기입하면 다운로드 및 설정 자동화됨 -> 의존성
- 의존성 관리, 컴파일, 테스트, 패키징 과정을 자동화함.
- 정형화된 디렉터리 구조와 표준화된 빌드 과정 제공함
- 중앙 저장소를 통해 라이브러리와 플러그인 관리함.

> 내가 작업중인 프로그램이 특정 라이브러리를 사용해야 한다면 그 **라이브러리에 의존한다**라고 표현.
>
> 의존성 관리란 사용하는 라이브러리에 대한 관리를 의미함.

2) Gradle (현대적인 도구)

- Groovy 또는 Kotlin으로 빌드 스크립트(설정 파일) 작성함.
- 의존성 관리, 빌드, 테스트, 배포를 자동으로 수행함. -->Maven과 기능이 동일함.
- 멀티 프로젝트 빌드에 강함
- **속도와 유연성이 Maven에 비해 뛰어남.**
- Maven에 비해 요구되는 메모리 사용이 높다.

> VSCode(순정) + Gradle 조합으로 개발할 경우 16G메모리 권장
>
> VSCode에 추가 익스텐을 설치하고, 그 외에 다른 프로그램을 동작시킨다면 32G 권장함 -> 거의 2배

맥북은 사지마라. 원도우에서도 자바 잘 쓰지도 못하면서 왜 복잡한 맥을 쓰려고하냐, 나중에 자바 익숙해지면 사라.

컴퓨터로 뭘 만드는 직업인데 일반인보다는 좋은 컴퓨터 써야하지않겠냐

윈도우 확정, 마이크로소프트 안좋음, 컴퓨터는 비쌀수록 좋은건 확실.

3. Gradle 설치

- 1. Gradle 공식 사이트(https://gradle.org/releases/)에서 최신 버전 다운로드함.
- 2. 압축 파일을 원하는 경로에 압축 해제함.
- 3. 환경 변수 설정:
  - 시스템 환경 변수 `GRADLE_HOME`에 Gradle 폴더 경로 설정함.
  - `Path` 변후에 `%GRADLE_HOME%\bin` 추가함.
- 4. 명령 프롬프트에서 `gradle -v` 입력해 설치 확인함.
- 5. 설치 후 명령프롬프트에서 `gradle -v` 버젼 확인 및 설치 확인

- **Gradle 8.13**

4. VSCode 익스텐션

Gradle Language Support 을 검색하여 설치

- > 설치 완료 후 VSCode 재시작 필요 (환경변수 인식을 위해 필요함)

#02. VSCode로 Gradle 프로젝트 생성하기

1. 명령 팔레트(`CTRL+SHIFT+P`)에서 **Gradle: Create a Gradle Java Project** 선택

- Advenced가 아님에 주의
- Advenced 명령밖에 표시되지 않는 경우 현재 열고 있는 폴더를 닫아야 함
  - (`파일--> 폴더 닫기`)

2. 프로젝트가 저장될 폴더를 직접 생성해야 한다.
   - JAVA 프로젝트는 프로젝트 생성 과정에서 폴더가 만들어지지만 Gradle은 폴더가 만들어지지 않는다.
3. 설정 파일 언어 종류를 Grooby로 선택
4. 프로그램 이름은 폴더 이름으로 자동으로 인식되므로 그냥 엔터

프로젝트 구조

폴더, 파일	설명
.gradle	gradle의 작동과 관련한 설정 정보가 저장되는 폴더(삭제안됨)
.vscode	VSCode의 자체 설정 폴더

폴더, 파일	설명
app	소스파일과 리소스(=소스를 제외한 재료 파일)가 저장됨
gradle	Gradle이 작업을 수행하는 폴더 (삭제안됨)
.git~~~	github 관련 파일 (삭제하세요~~~)
gradlew,gradlew.bat	Gradle 실행 파일 (절대 삭제 안됨)
settings.gradle	프로그램의 이름과 소스파일 저장 위치(app)를 설정하는 파일

2.프로젝트 생성 후 기본 작업

불필요한 파일 삭제

`app/src/test` 폴더 삭제

Spring에서는 사용하지만 지금은 불필요함

기본 패키지 변경

프로젝트가 생성된 폴더 이름으로 패키지가 기본 생성됨
이 패키지를 다른 이름으로 변경

> ex: kr.thjeong.프로그램이름
> 여기서는 `kr.yjh.gradle` 로 설정함.

```
//
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■■
//■ 1. 명령 팔레트(`CTRL+SHIFT+P`)에서 **Gradle: Create a Gradle Java Project** 선택
//■                                  `Gradle` 치면 나옴
//■ 2. 설정 파일 언어 종류를 Grooby로 선택
//■ 3. 프로그램 이름은 폴더 이름으로 자동으로 인식되므로 그냥 엔터(gradle 이라 나올 거임)
//■ 4. app(bin, src, build.gradle) 여기서 src에서 test 지우면 안에 것들이 밖으로 나옴
//■ 5. 거기서 java폴더 안에 app.java 클릭
//■ 6. package kr.yjh.gradle; 로 변경
//■ 7. 31라인에서 JAVA 버전을 `21`에서 `17`로 변경
//■  ( languageVersion = JavaLanguageVersion.of(17) )   <----버전을 의미함
//■ 8. 37라인에서 메인 클래스의 위치를 변경 ( 'kr.yjh.gradle.App' )
//■ 9. 이것이 '''기본 셋팅''' Ex) 호쌤: "이거 만드세요~" 하시면  !!이대로 하면됨!!
//
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■
```

`build.gradle` 파일 설정 변경

31라인에서 JAVA 버젼을 21에서 17로 변경

```
java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17) //< ---여기(21번째라
인)!!!!!
    }
}
```

37라인에서 메인 클래스의 위치를 변경

```
application {
    // Define the main class for the application.
    mainClass = 'kr.yjh.gradle.App'  //< ---------------여기(37번째라
인)!!!!!!!!!
}
```

#03. Maven Repository

https://mvnrepository.com/

Maven Repository 는 다양한 **Java 라이브러리(.jar 파일)와 메타데이터를 저장하는 저장소**

Gradle도 Maven Repository를 활용하여 **의존성(dependency)을 자동으로 다운로드** 함

Maven Repository

1. Lombok 이라는 키워드로 라이브러리 검색 (검색어는 사용하려는 라이브러리에 맞추어 결정해야함)
2. 2. 검색 결과 중에서 **Project Lombok**으로 이동
3. 가장 최신 버전의 버전 숫자를 클릭해서 페이지 이동
   -2025-04-17 기준 1.18.38이 최신 버젼임
4. 화면 중앙의 탭에서 **Gradle** 선택
   - Scope: Provied
   - Format: Groovy short
5. 화면 중앙에 나타나는 코드 복사

```
// https://mvnrepository.com/artifact/org.projectlombok/lombok
   compileOnly 'org.projectlombok:lombok:1.18.38'
```

6. **build.gradle** 파일에서 18라인 브근 **dependencies**의 괄호 안에 복사한 내용 삽입 (닫는 괄호 직전에 넣으세요.)
   - 파일 저장 후 동기화 관련 팝업창이 VSCode 우측 하단에 표시됨 ==>**YES** 클릭
   - 프로젝트가 갱신되는동안 VSCode 왼쪽 하단의 상태표시중에서 진행상황이 표시됨
     - 진행 상황이 **Java Ready**로 나타나기 전까지 프로그램 사용 금지!!!!

#04. Log

- 프로그램이 실행되는 동안 발생하는 **이벤트, 상태, 오류 등을 기록한 텍스트 정보**
- 프로그램의 실행 과정을 추적하는 용도로 활용됨

> 프로그램이 출력하는 모든 내용을 텍스트 파일로 기록하는 형태

로그의 주요 용도

- **디버깅**: 버그 발생 원인 추적
  - 에러 발생시 출력되는 내용을 별도 파일로 저장
- **모니터링**: 시스템 상태 실시간 확인
- **분석**: 사용자 행동, 성능, 트래픽 분석 등
- **보안**: 비정상 접근 탐지, 침해 분석

Logback

- Java 기반 애플리케이션에서 사용되는 **로깅 라이브러리**
- **SLF4J(Simple Logging Facade for Java)**의 공식 구현체
- 이전 로깅 시스템인 **Log4j**를 개선하여 설계됨

```
// https://mvnrepository.com/artifact/org.slf4j/slf4j-api
implementation 'org.slf4j:slf4j-api:2.0.17'

// https://mvnrepository.com/artifact/ch.qos.logback/logback-classic
implementation 'ch.qos.logback:logback-classic:1.5.18'
```

```
//━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
//★★★★★★★★  단원 이름 말하면    ★★★★★★★★★
//★★★★★★★★                ★★★★★★
//★★★★★★★★★★프로젝트 만들고★★★★★★★★★★
//★★★★★★롬복, SLF4J, 로그백 등 셋팅 넣고★★★★★
//★★★★★★                    ★★★★
//★★★★★★★★★★로그백 넣고
//★★★★★★                    ★★★★
//★★★★★★★★★5분만에 셋팅해야함★★★★★★★★★
//━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
```