4. 스프링 부트 스타터와 라이브러리 관리

#2.인강/9. 스프링부트/강의#

라이브러리 관리의 어려움

프로젝트를 처음 시작하면 어떤 라이브러리들을 사용할지 고민하고 선택해야 한다. 예를 들어서 스프링 WEB, 내장 톰캣, JSON 처리기, 로거 등등 수 많은 라이브러리를 선택해야 한다. 여기에 추가로 각 라이브러리의 버전까지 고민해야 한다. 더 심각한 문제는 각 라이브러리들끼리 호환이 잘 되는 버전도 있지만 잘 안되는 버전들도 있다. 과거에는 이런 문제들 때문에 처음 프로젝트를 세팅하는데 상당히 많은 시간을 소비했다.

스프링 부트는 개발자가 라이브러리들을 편리하게 사용할 수 있는 다양한 기능들을 제공한다.

- 외부 라이브러리 버전 관리
- 스프링 부트 스타터 제공

라이브러리 직접 관리

- 스프링 부트가 제공하는 편리한 라이브러리 관리 기능을 사용해보기 전에, 잠깐 과거로 돌아가서 직접 라이브러리를 하나하나 고르고 설정하는 방법을 알아보자.
- 다음 예는 스프링 웹과 내장 톰캣을 사용하는 웹 애플리케이션이다.

프로젝트 설정 순서

1. `lib-start`의 폴더 이름을 `lib`로 변경하자.
2. **프로젝트 임포트**

   File ➡ Open ➡ 해당 프로젝트의 `build.gradle`을 선택하자. 그 다음에 선택창이 뜨는데, Open as Project를 선택하자.

build.gradle 확인

```
plugins {
    id 'org.springframework.boot' version '3.0.2'
    id 'java'
```

```
}

group = 'hello'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}



dependencies {

    //1. 라이브러리 직접 지정
    //스프링 웹 MVC
    implementation 'org.springframework:spring-webmvc:6.0.4'
    //내장 톰캣
    implementation 'org.apache.tomcat.embed:tomcat-embed-core:10.1.5'
    //JSON 처리
    implementation 'com.fasterxml.jackson.core:jackson-databind:2.14.1'
    //스프링 부트 관련
    implementation 'org.springframework.boot:spring-boot:3.0.2'
    implementation 'org.springframework.boot:spring-boot-autoconfigure:3.0.2'
    //LOG 관련
    implementation 'ch.qos.logback:logback-classic:1.4.5'
    implementation 'org.apache.logging.log4j:log4j-to-slf4j:2.19.0'
    implementation 'org.slf4j:jul-to-slf4j:2.0.6'
    //YML 관련
    implementation 'org.yaml:snakeyaml:1.33'

}

tasks.named('test') {
```

```
    useJUnitPlatform()
}
```

- 스프링으로 웹 애플리케이션을 실행하려면 생각보다 수 많은 라이브러리가 필요하다.
- 스프링 웹 MVC, 내장 톰캣, JSON 처리, 스프링 부트 관련, LOG, YML 등등 다양한 라이브러리가 사용된다.

참고

`io.spring.dependency-management` 플러그인은 일부로 적용하지 않았다. 자세한 내용은 뒤에서 설명한다.

- 동작 확인
  - 기본 메인 클래스 실행(`LibApplication.main()`)
  - http://localhost:8080 호출해서 Whitelabel Error Page가 나오면 정상 동작

HelloController

```
package hello.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/hello-spring")
    public String hello() {
        return "hello spring!";
    }
}
```

- http://localhost:8080/hello-spring 호출해서 정상 동작 확인

라이브러리 직접 선택시 발생하는 문제

웹 프로젝트를 하나 설정하기 위해서는 수 많은 라이브러리를 알아야 한다. 여기에 추가로 각각의 라이브러리의 버전까지 골라서 선택해야 한다. 여기서 눈에 보이지 않는 가장 어려운 문제는 각 라이브러리들 간에 서로 호환이 잘 되는 버전도 있지만 호환이 잘 안되는 버전도 있다는 점이다. 개발자가 라이브러리의 버전을 선택할 때 이런 부분까지 고려하는 것은 매우 어렵다.

스프링 부트 라이브러리 버전 관리

스프링 부트는 개발자 대신에 수 많은 라이브러리의 버전을 직접 관리해준다.
이제 개발자는 원하는 라이브러리만 고르고 라이브러리의 버전은 생략해도 된다. 그러면 스프링 부트가 부트 버전에 맞춘 최적화된 라이브러리 버전을 선택해준다.

버전 관리 기능을 사용하려면 `io.spring.dependency-management` 플러그인을 사용해야 한다.
다음을 참고해서 추가하자.

build.gradle - plugins 수정

```
plugins {
    id 'org.springframework.boot' version '3.0.2'
    id 'io.spring.dependency-management' version '1.1.0' //추가
    id 'java'
}
```

build.gradle - dependencies 수정

```
dependencies {
    //2. 스프링 부트 라이브러리 버전 관리
    //스프링 웹, MVC
    implementation 'org.springframework:spring-webmvc'
    //내장 톰캣
    implementation 'org.apache.tomcat.embed:tomcat-embed-core'
    //JSON 처리
    implementation 'com.fasterxml.jackson.core:jackson-databind'
    //스프링 부트 관련
    implementation 'org.springframework.boot:spring-boot'
    implementation 'org.springframework.boot:spring-boot-autoconfigure'
    //LOG 관련
    implementation 'ch.qos.logback:logback-classic'
    implementation 'org.apache.logging.log4j:log4j-to-slf4j'
    implementation 'org.slf4j:jul-to-slf4j'
    //YML 관련
```

```
    implementation 'org.yaml:snakeyaml'
}
```

- 라이브러리를 보면 버전 정보가 모두 제거되었다.

잘 동작하는지 확인해보자.

- http://localhost:8080/hello-spring 호출해서 정상 동작 확인

dependency-management 버전 관리

`io.spring.dependency-management` 플러그인을 사용하면 `spring-boot-dependencies`에 있는 다음 bom 정보를 참고한다.
참고로 `spring-boot-dependencies`는 스프링 부트 gradle 플러그인에서 사용하기 때문에 개발자의 눈에 의존관계로 보이지는 않는다.

버전 정보 bom

- https://github.com/spring-projects/spring-boot/blob/main/spring-boot-project/spring-boot-dependencies/build.gradle
- 해당 `build.gradle` 문서안에 보면 `bom`이라는 항목이 있다.
- 각각의 라이브러리에 대한 버전이 명시되어 있는 것을 확인할 수 있다.
- 물론 현재 프로젝트에서 지정한 스프링 부트 버전을 참고한다.
  - `id 'org.springframework.boot' version '3.0.2'` : 여기에 저징된 스프링 부트 버전 참고
  - 스프링 부트의 버전을 변경해보면 나머지 라이브러리들의 버전도 변하는 것을 확인할 수 있다.

> **참고 - BOM(Bill of materials)**
> 자재 명세서(Bill of materials)란 제품구성하는 모든 부품들에 대한 목록이다.
> 부품이 복잡한 요소들로 구성된 조립품인 경우는 계층적인 구조로 작성될 수 있다. - 위키백과

> **참고 - 스프링 부트가 관리하는 외부 라이브러리 버전을 확인하는 방법**
> https://docs.spring.io/spring-boot/docs/current/reference/html/dependency-versions.html#appendix.dependency-versions.coordinates

> **참고 - 스프링 부트가 관리하지 않는 라이브러리**
> 스프링 부트가 관리하지 않는 외부 라이브러리도 있다. 특히 아직 잘 알려지지 않거나 대중적이지 않은 경우가 그러한데, 이때는 다음과 같이 라이브러리의 버전을 직접 적어주어야 한다.
> `implementation 'org.yaml:snakeyaml:1.30'`

정리

스프링 부트가 제공하는 버전 관리는 스프링 자신을 포함해서 수 많은 외부 라이브러리의 버전을 최적화

해서 관리해준다. 이제 개발자는 스프링 부트 자체의 버전만 지정하면 된다.
그리고 스프링 부트가 해당 스프링 부트 버전에 맞는 각 라이브러리의 호환성을 테스트 했기 때문에 안전하게 사용할 수 있다.
(물론 항상 100% 완벽할 수는 없다.)

다음에는 라이브러리를 자체를 고르는 고민을 줄여주는 스프링 부트 스타터에 대해서 알아보자.

스프링 부트 스타터

- 앞서 보았듯이 웹 프로젝트를 하나 실행하려면 생각보다 수 많은 라이브러리가 필요하다.
- 스프링 웹 MVC, 내장 톰캣, JSON 처리, 스프링 부트 관련, LOG, YML 등등 다양한 라이브러리가 사용된다.
- 개발자 입장에서는 그냥 웹 프로젝트를 하나 시작하고 싶은 것이고, 일반적으로 많이 사용하는 대중적인 라이브러리들을 포함해서 간단하게 시작하고 싶을 것이다.
- 스프링 부트는 이런 문제를 해결하기 위해 프로젝트를 시작하는데 필요한 관련 라이브러리를 모아둔 스프링 부트 스타터를 제공한다.
- 스프링 부트 스타터 덕분에 누구나 쉽고 편리하게 프로젝트를 시작할 수 있다.

build.gradle - dependencies 수정

```
dependencies {
    //3. 스프링 부트 스타터
    implementation 'org.springframework.boot:spring-boot-starter-web'
}
```

- `spring-boot-starter-web` 이 라이브러리 하나로 지금까지 우리가 직접 넣어주었던 모든 라이브러리가 포함된다.
- 이것은 사용하기 편리하게 의존성을 모아둔 세트이다.
  - 이것을 하나 포함하면 관련 의존성 세트가 한번에 들어온다.
  - 스타터도 스타터를 가질 수 있다.
- 스프링과 웹을 사용하고 싶으면 `spring-boot-starter-web`
  - 스프링 웹 MVC, 내장 톰캣, JSON 처리, 스프링 부트 관련, LOG, YML 등등
- 스프링과 JPA를 사용하고 싶으면 `spring-boot-starter-data-jpa`
  - 스프링 데이터 JPA, 하이버네이트 등등

스프링 부트 스타터 - 이름 패턴

- `spring-boot-starter-*`
- 쉽게 찾게 도와줌
- 공식: `spring-boot-starter-*`
- 비공식: `thirdpartyproject-spring-boot-starter`
  - ex) `mybatis-spring-boot-starter`

스프링 부트 스타터 - 자주 사용하는 것 위주

- `spring-boot-starter` : 핵심 스타터, 자동 구성, 로깅, YAML
- `spring-boot-starter-jdbc` : JDBC, HikariCP 커넥션풀
- `spring-boot-starter-data-jpa` : 스프링 데이터 JPA, 하이버네이트
- `spring-boot-starter-data-mongodb` : 스프링 데이터 몽고
- `spring-boot-starter-data-redis` : 스프링 데이터 Redis, Lettuce 클라이언트
- `spring-boot-starter-thymeleaf` : 타임리프 뷰와 웹 MVC
- `spring-boot-starter-web` : 웹 구축을 위한 스타터, RESTful, 스프링 MVC, 내장 톰캣
- `spring-boot-starter-validation` : 자바 빈 검증기(하이버네이트 Validator)
- `spring-boot-starter-batch` : 스프링 배치를 위한 스타터

스프링 부트 스타터의 전체 목록은 다음 공식 메뉴얼을 참고하자.

- https://docs.spring.io/spring-boot/docs/current/reference/html/using.html#using.build-systems.starters

라이브러리 버전 변경

외부 라이브러리의 버전을 변경하고 싶을 때 다음과 같은 형식으로 편리하게 변경할 수 있다.

```
ext['tomcat.version'] = '10.1.4'
```

> 스프링 부트가 관리하는 외부 라이브러리 버전 변경에 필요한 속성 값
> https://docs.spring.io/spring-boot/docs/current/reference/html/dependency-versions.html#appendix.dependency-versions.properties

- 스프링 부트가 관리하는 외부 라이브러리의 버전을 변경하는 일은 거의 없다. 하지만 아주 가끔 문제가 발생하기도 하므로 알아두자.

정리