Chap01. SpringFramework 개요 및 개발환경 구축

1. SpringFramework

1-1. SpringFramework 개요

1-1-1. SpringFramework란?

스프링 프레임워크는 자바 플랫폼을 위한 오픈 소스 애플리케이션 프레임워크로서 간단히 스프링이라고도 한다.
동적인 웹 사이트를 개발하기 위한 여러 가지 서비스를 제공하고 있다.
또한 대한민국 공공기관의 웹 서비스 개발 시 사용을 권장하고 있는 전자정부 표준프레임워크의 기반 기술로서 쓰이고 있다.

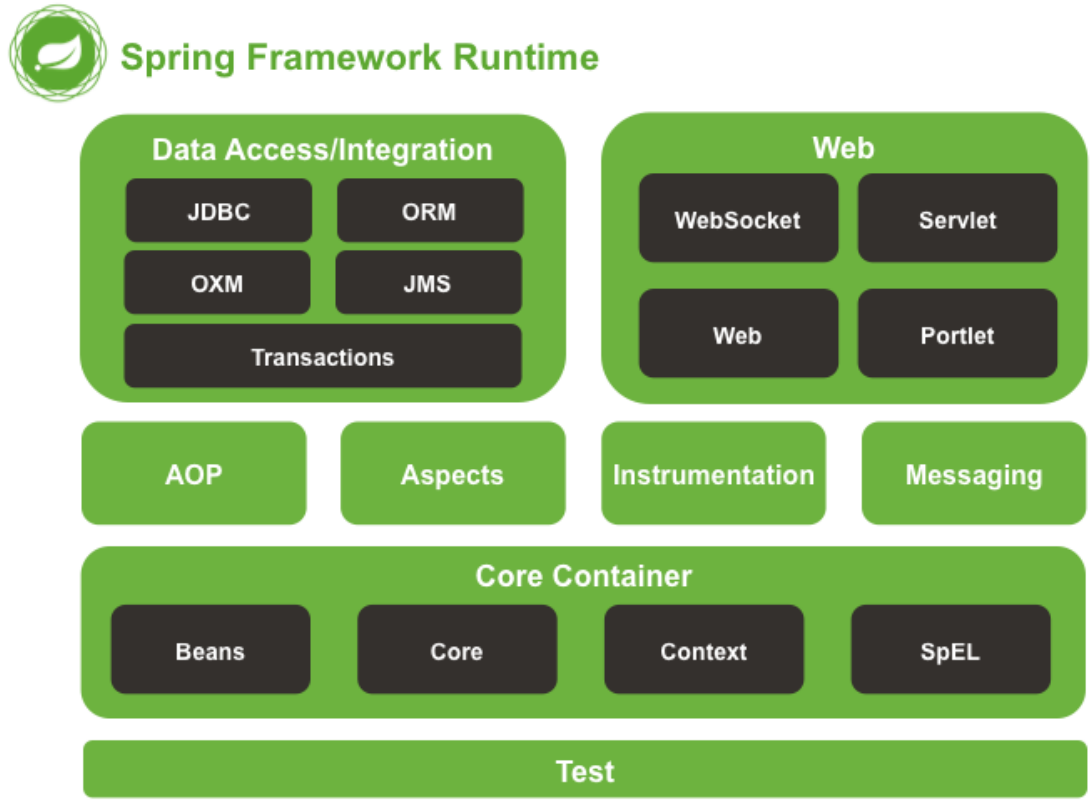
> Spring 공식 사이트
>
> Home
> Level up your Java code and explore what Spring can do for you.
> https://spring.io/

1-2. SpringFramework의 특징

- 오픈 소스이다.
  - 다양한 실제 사용 사례를 기반으로 지속적인 피드백을 제공하는 크고 활동적인 커뮤니티가 있어 오랜 시간에 걸쳐 성공적으로 진화하는 데 도움이 되었다.
- Java 엔터프라이즈 애플리케이션을 쉽게 만들 수 있다.
  - 애플리케이션의 요구 사항에 따라 다양한 종류의 아키텍처를 생성할 수 있는 유연성과 함께 엔터프라이 환경에서 Java 언어를 수용하는 데 필요한 모든 것을 제공한다.
- 프레임워크의 기본 원칙에 충실하다.
  - 강력한 이전 버전과의 호환성을 유지한다.
  - 유연성을 수용하며 작업 수행 방법에 대해 독선적이지 않다. 다양한 관점에서 다양한 애플리케이션 요구 사항을 지원한다.

- 모든 수준에서 선택권을 제공한다. Spring을 사용하면 가능한 한 늦게 디자인 결정을 연기할 수 있다. 예를 들어 코드를 변경하지 않고 구성을 통해 지속성 공급자를 전환할 수 있다.
- 의미 있고 최신이며 정확한 javadoc을 강조한다. 패키지 간의 순환 종속성이 없는 깨끗한 코드 구조를 주장할 수 있는 극소수의 프로젝트 중 하나이다.

2. SpringFramework 구성 모듈



https://docs.spring.io/spring-framework/docs/5.0.0.M5/spring-framework-reference/html/overview.html#overview-modules

2-1. Spring Core Container

2-1-1. Spring Core Container

`Spring Core Container` 는 스프링에서 가장 기본적이며 중요한 모듈 중 하나이다. 이 모듈은 스프링에서 객체의 생성과 관리를 담당한다. 스프링의 `DI(Dependency Injection)` 과 `IoC(Inversion of Control)` 개념이 구현되어 있다. 이를 이용하여 코드의 `재사용성` 과 `유지보수성` 을 높일 수 있다.

2-2. AOP(Aspect Oriented Programming) / Aspects / Instrumentation / Messaging

2-2-1. AOP(Aspect Oriented Programming)

`AOP`는 객체지향 프로그래밍에서의 코드 중복을 제거하고 코드를 재사용하기 위한 프로그래밍 기법이다. 기존의 코드를 변경하지 않고 새로운 기능을 추가하는 방식으로 사용한다. 스프링에서는 AOP를 이용해 로깅, 트랜잭션, 보안 등의 기능을 제공한다.

2-2-2. Aspects

- 별도의 spring-aspects 모듈은 `AspectJ`와의 통합을 제공한다.

2-2-3. Instrumentation

- `Instrumentation`은 자바 프로그램의 동작을 관찰하고 조작하는 기술이다. 스프링에서는 Instrumentation을 이용해 클래스 로딩, 메소드 실행 등의 작업을 추적하고 변경할 수 있다.

2-2-4. Messaging

- Apache Kafka 및 RabbitMQ와 같은 메시지 브로커의 통합을 통해 메시징 아키텍처를 지원한다. 메시지 채널, 메시지 처리기 및 메시징 기반 응용 프로그램을 생성할 수 있는 메시지 끝점에 대한 추상화가 포함된다. 또한 이 모듈을 사용하면 메시지 채널에서 메시지를 수신하고 처리할 수 있는 메시지 구동 POJO(Plain Old Java Objects)를 생성할 수 있다.

2-3. Data Access / Integration

2-3-1. JDBC(JAVA DataBase Connectivity)

`JDBC`는 Java에서 데이터베이스와 연결할 수 있는 API이다. JDBC는 데이터베이스에 대한 쿼리를 실행하고 결과를 처리할 수 있는 메서드와 클래스를 제공한다. 이를 통해 Java 애플리케이션에서 데이터베이스를 쉽게 사용할 수 있다.

2-3-2. ORM(Object Relation Mapping)

`ORM`은 객체 지향 프로그래밍에서 관계형 데이터베이스와의 상호작용을 추상화하는 프로그래밍 기술이다. 이를 통해 개발자는 객체를 직접 다루면서 데이터베이스를 조작할 수 있다. ORM 프레임워크는 객체와 데이터베이스 간의 매핑을 담당하며, 데이터베이스 쿼리를 생성하고 실행하는 등의 작업을 대신 수행한다.

2-3-3. Transaction Management

💡 `트랜잭션`은 데이터베이스 상태를 변경하는 작업의 단위이다. 여러 개의 쿼리가 트랜잭션으로 묶이면, 모든 쿼리가 성공하면 변경 내용이 데이터베이스에 반영되고, 하나라도 실패하면 변경 내용이 취소된다. 이런 트랜잭션을 보다 쉽게 처리하기 위해 스프링에서는 `트랜잭션 관리 기능`을 제공한다. 이를 이용하면 선언적 트랜잭션 처리, 프로그램적 트랜잭션 처리 등을 할 수 있으며, 트랜잭션 속성을 XML이나 어노테이션을 이용해 설정할 수 있다.

2-3-4. OXM(Object XML Mapping)

- `OXM(Object XML Mapping)`은 자바 객체와 XML 문서 간의 매핑을 지원하는 모듈이다. 개발자는 XML 문서를 직접 작성할 필요 없이, 자바 객체를 이용하여 간단하게 XML 문서를 생성하고 읽을 수 있다.

2-3-5. Java Messaging Service(JMS)

- `Java Messaging Service(JMS)`는 자바에서 메시지를 주고 받을 수 있는 API이다. 이 API를 사용하면 분산 시스템에서 메시지를 교환할 수 있다. JMS는 메시지 전송과 수신을 위한 스펙과 메시지의 속성을 정의하기 위한 API를 제공한다. 메시지를 보내는 쪽에서는 메시지를 생성하고 목적지를 지정한 다음, JMS Provider를 통해 메시지를 보낸다. 메시지를 받는 쪽에서는 메시지를 수신하고 처리한다. JMS는 대부분의 메시지 지향 미들웨어(MOM)에서 사용된다.

2-4. Web

2-4-1. Servlet

💡 `Servlet`은 웹 애플리케이션 서버에서 실행되는 자바 클래스이다. 클라이언트의 요청을 처리하고, 응답을 생성하는데 사용된다. Servlet은 JSP(JavaServer Pages)와 함께 Java EE(Java Platform, Enterprise Edition)의 Web API를 구성한다. 스프링에서는 Servlet API를 이용하여 웹 어플리케이션을 개발할 수 있다.

2-4-2. Web

💡 `Spring Web MVC`는 스프링에서 제공하는 웹 어플리케이션 개발을 위한 MVC 패턴 구현 모듈이다. 웹 어플리케이션을 개발할 때, 요청과 응답을 처리하며 페이지 이동을 관리하는 컨트롤러, 데이터를 처리하고 화면에 출력하는 뷰, 비즈니스 로직을 수행하는 모델로 구분하여 개발하는 것이 일반적이다. 이러한 구조를 MVC 패턴이라고 한다. 대부분의 웹 프레임워크에서 제공하는 MVC 패턴의 구현체이며, 스프링에서도 이를 지원한다. Spring Web MVC는 이러한 컨트롤러, 뷰, 모델을 개발하는 데 도움을 주며, 특히 유연한 URL 매핑 기능을 제공하여 URL과 컨트롤러를 매핑시키는 작업을 쉽게 수행할 수 있다.

- `Spring WebFlux`는 스프링 5에서 추가된 모듈로, 비동기 웹 어플리케이션 개발을 위해 Reactive Streams를 지원하는 모듈이다. 기존의 Spring Web MVC와는 달리, Servlet API에 의존하지 않

고, Netty, Undertow, Servlet 3.1+ 컨테이너와 같은 네이티브 리액티브 서버를 사용하여 개발할 수 있다. Spring WebFlux는 Reactor를 이용하여 비동기 프로그래밍을 지원하며, 이를 통해 높은 처리량과 확장성을 제공한다.

2-4-3. WebSocket

- `WebSocket`은 HTTP와 달리 양방향 통신을 지원하는 프로토콜이다. 이를 이용하면 서버와 클라이언트 간의 실시간 양방향 통신이 가능하다. 스프링에서는 웹 소켓을 지원하기 위해 `Spring Websocket` 모듈을 제공한다. Spring Websocket 모듈은 STOMP(Subscribable-Text-Oriented Messaging Protocol) 프로토콜을 이용하여 WebSocket을 구현한다. STOMP는 WebSocket을 이용하여 메시지를 주고 받을 때 사용되는 메시지 프로토콜이다. Spring Websocket 모듈은 메시지 브로커와 연동하여 구현할 수 있으며, 브로커를 이용하여 메시지를 보내고 받을 수 있다. 이를 이용하면 다수의 사용자 간의 실시간 채팅, 게임 등의 기능을 구현할 수 있다.

2-4-4. Portlet

- `Portlet`은 웹 페이지를 구성하는 데 사용되는 모듈이며, 클라이언트의 요청에 따라 동적으로 로드된다. Portlet은 일반적인 웹 애플리케이션과 달리, 특정 Portlet 컨테이너에서 실행되어야 하며, Portlet 컨테이너는 웹 서버에서 분리되어 실행된다. Spring Portlet MVC는 Spring MVC와 비슷한 방식으로 동작하며, Portlet 컨테이너에서 실행되는 Portlet 컨트롤러, 뷰, 모델을 개발할 수 있다.

2-5. Test

2-5-1. Test

- spring-test 모듈은 `JUnit` 또는 TestNG를 사용하여 Spring 구성 요소의 단위 테스트 및 통합 테스트를 지원한다.
  - Spring ApplicationContexts의 일관된 로딩과 해당 컨텍스트의 캐싱을 제공한다.
  - 코드를 별도로 테스트하는 데 사용할 수 있는 모의 개체를 제공한다.

2-6. Languages

2-6-1. Groovy 지원

- `Groovy`는 JVM에서 실행되는 스크립트 언어이다. Java와 완벽하게 호환되며, Java와 같은 문법을 사용하면서도 보다 간결한 코드를 작성할 수 있다. Groovy는 Gradle 등의 빌드 툴에서 널리 사용되어, Java 개발 프로세스에서 자주 사용된다.

2-6-2. Kotlin 지원

- `Kotlin`은 JVM에서 실행되는 프로그래밍 언어로, 자바와 완벽하게 상호운용이 가능하다. 자바보다 간결한 문법과 람다 표현식, 널 안전성 등의 기능을 제공하여 개발 생산성을 높일 수 있다. 스프링에서는 Kotlin을 공식적으로 지원하며, 스프링 부트에서도 Kotlin을 이용한 개발을 쉽게 할 수 있다.

3. SpringFramework 버전

3-1-1. SpringFramwork의 등장 배경

`J2EE(Java 2 Platform, Enterprise Edition)`은 1999년에 발표된 자바 기반의 엔터프라이즈 애플리케이션 개발 표준이다. J2EE는 다양한 기술과 API를 포함하고 있으며, 이들을 조합하여 웹 애플리케이션, 엔터프라이즈 애플리케이션, 분산 컴퓨팅 등을 개발할 수 있다. J2EE는 다양한 기능을 제공하며, EJB(Enterprise JavaBeans)와 같은 기술을 이용하여 비즈니스 로직을 구현할 수 있다. 하지만 J2EE는 매우 복잡하고, 서버 환경에서 실행되는 애플리케이션에서는 높은 성능을 발휘하지 못하는 문제점이 있다. 이에 대한 대안으로 등장한 것이 `Spring`으로, Spring은 J2EE의 복잡성을 줄이고 보다 간결하게 개발할 수 있도록 지원한다.

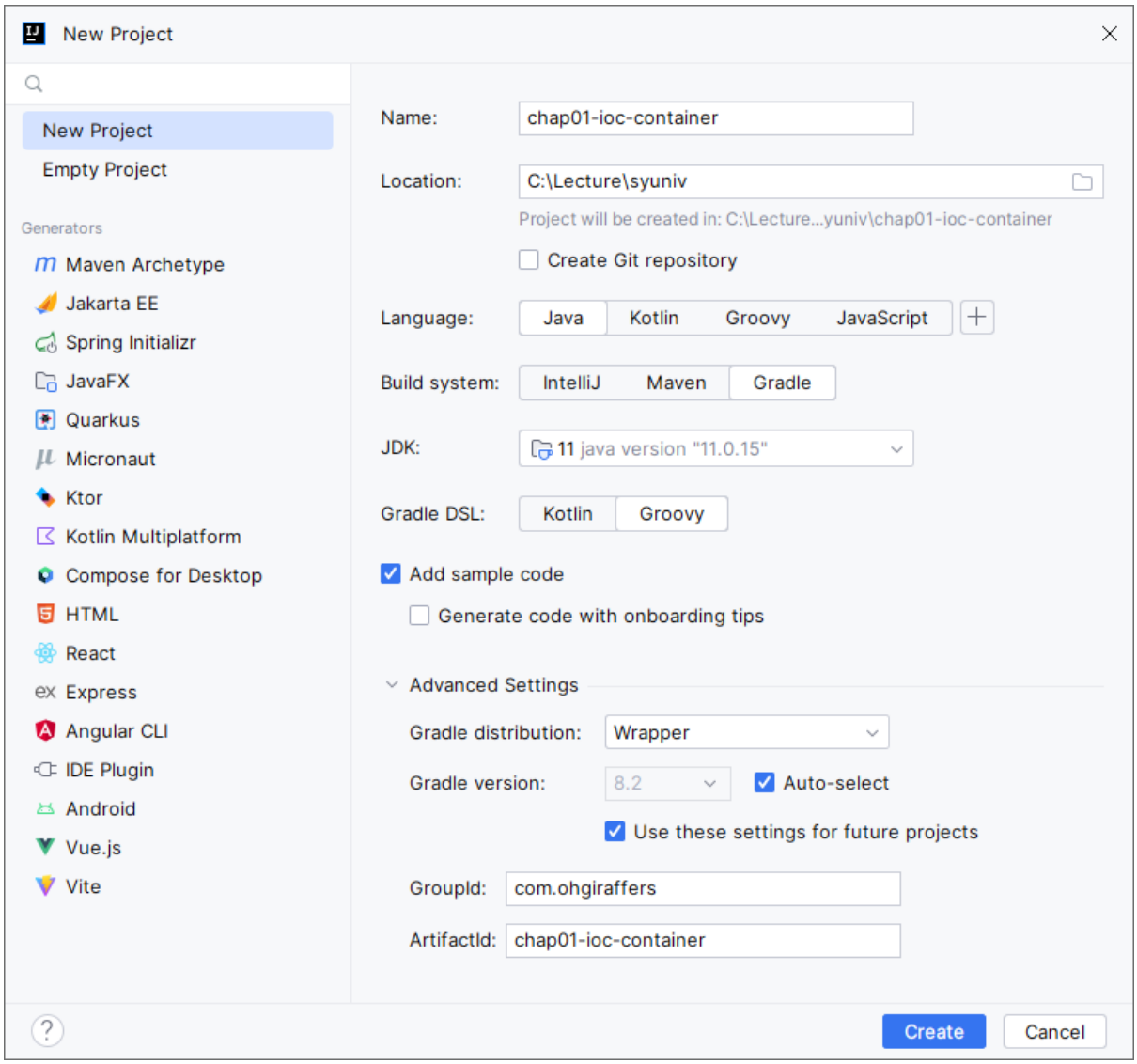
3-1-2. SpringFramwork 버전

SpringFramework 버전	출시일	특징
1.0	2004년	경량 컨테이너, AOP, JDBC 추상화, 트랜잭션 관리 등의 기능 제공
2.0	2006년	AspectJ 통합, XML 네임스페이스, 자바 5 지원 등의 기능 추가
2.5	2007년	어노테이션 기반의 구성요소 스캔, OXM(Object-XML Mapping) 모듈, JPA(Java Persistence API) 지원 등의 기능 추가
3.0	2009년	Java SE 5 이상 지원, RESTful 웹 서비스 지원, 자바 구성 어노테이션 등의 기능 추가
3.1	2011년	자바 7 지원, 캐시 추상화, 스프링 MVC 테스트 프레임워크 등의 기능 추가
3.2	2012년	Java SE 6 이상 지원, 자바 기반의 설정 클래스 지원, 테스트 컨텍스트 프레임워크 등의 기능 추가
4.0	2013년	자바 8 지원, WebSocket, JMS 2.0, HTML5 지원 등의 기능 추가
4.1	2014년	자바 8의 람다 표현식 지원, 캐시 추상화 개선, Groovy 스크립트 작성을 위한 스크립트 빈 등의 기능 추가
4.2	2015년	자바 8의 Date-Time API 지원, HTTP Streaming, WebSocket 개선 등의 기능 추가
4.3	2016년	애플리케이션 컨텍스트 개선, 캐시 추상화 개선, 람다 지원 개선 등의 기능 추가
5.0	2017년	자바 8 이상 지원, 리액티브 프로그래밍 지원, Kotlin 지원 등의 기능 추가
5.1	2018년	리액티브 웹 클라이언트, Kotlin 코루틴 지원, Spring MVC 개선 등의 기능 추가
5.2	2019년	리액티브 테스트 지원, 로깅 시스템 개선, OAuth 2.0 클라이언트 등의 기능 추가
6.0	2022년	자바 17 이상 지원, XML 설정 지양

4. 수업 환경 설정

4-1. Project 생성

4-1-1. Intelli J에서 New Project 실행



- `Name` : 프로젝트명
- `Location` : 저장 경로
- `Language` : Java
- `Build system` : Gradle
- `JDK` : 11
- `Gradle DSL` : Groovy

4-2. 라이브러리 의존성 추가

4-2-1. Build System이란?

`Build System`은 소프트웨어를 빌드하고 패키징하는 데 사용되는 도구이다. 소프트웨어 개발의 생산성과 효율성을 향상시키기 위해 자동화 작업을 수행한다. 소스 코드와 라이브러리의 의존성을 처리하고, 컴파일러와 링커를 호출하며, 테스트 및 배포를 수행한다. 대표적인 Build System으로는 Maven, Gradle 등이 있다.

4-2-2. Maven, Gradle이란?

`Maven` : Java용 Build System으로, XML 기반으로 빌드 작업을 수행한다. Maven은 의존성 관리와 빌드 과정의 표준화가 잘 이루어져 있다. 또한, 중앙 저장소에서 필요한 라이브러리를 자동으로 다운로드 받아 사용할 수 있다. 그러나, 빌드 시간이 오래 걸리는 단점이 있다.

`Gradle` : Java용 Build System으로, Groovy 기반으로 빌드 작업을 수행한다. Gradle은 Maven보다 더 빠르고, 유연성이 높다. 스크립트 언어로 작성되어 있어, 빌드 스크립트의 가독성이 높고 유지보수가 용이하다. 빌드 과정에서 필요한 의존성 라이브러리를 미리 캐시해두어 빌드 시간을 단축할 수 있다.

- `Groovy`는 객체 지향 프로그래밍 언어인 Java를 기반으로 만들어진 스크립트 언어이다. Java와 마찬가지로 JVM(Java Virtual Machine)에서 동작하며, Java 라이브러리를 사용할 수 있다. Groovy는 Java와 달리 좀 더 간결하고 유연한 문법을 가지고 있어, Java보다 코드 작성이 간편하다.
- `Kotlin`은 Java보다 코드 작성이 간결하고 생산성이 높으며, Null 안전성이 높다는 장점이 있다. Gradle `Kotlin`은 Groovy나 Java로 작성된 스크립트보다 가독성이 높고 유지보수가 용이하다.

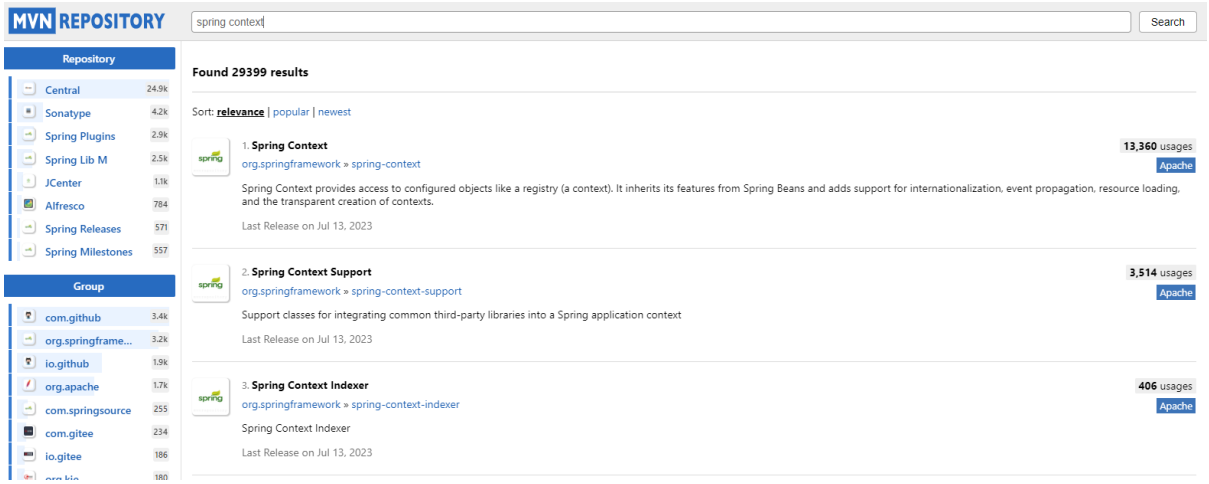
4-2-3. Maven Repository 사용하기

`Maven Repository`는 Maven Build System에서 사용하는 라이브러리 저장소이다. Maven은 이 저장소에서 필요한 라이브러리를 자동으로 다운로드 받아 사용한다. Maven Central Repository는 가장 대표적인 Maven Repository 중 하나이다. 이 저장소에는 수많은 Java 라이브러리들이 등록되어 있어, 개발자들은 필요한 라이브러리를 Maven Central Repository에서 검색하여 사용할 수 있다. Gradle도 Maven과 마찬가지로 라이브러리 관리를 위해 Maven Repository를 사용한다.

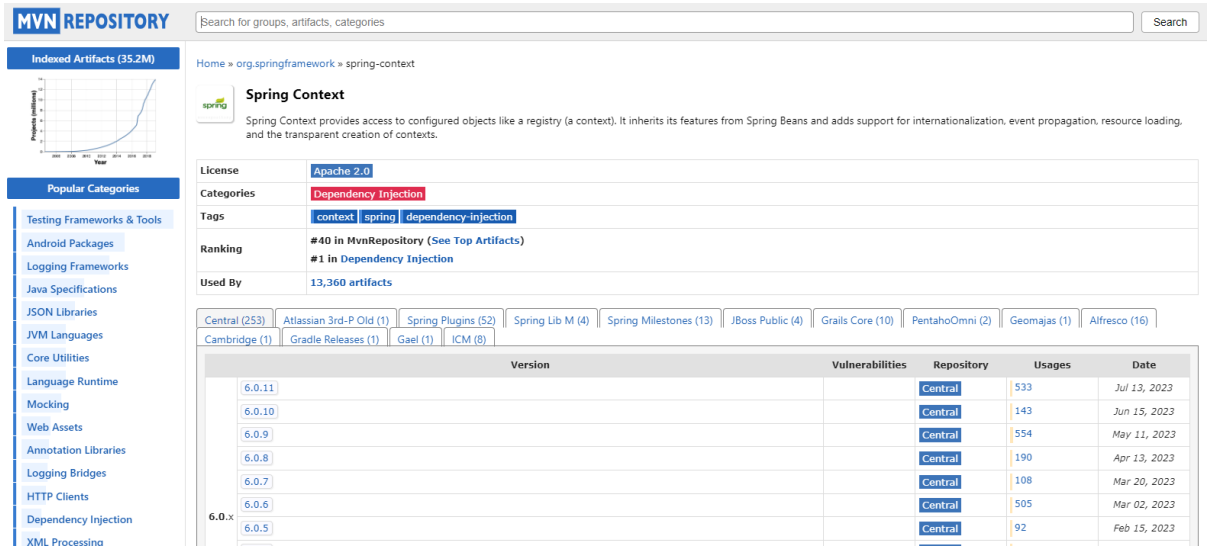
- 아래 링크로 `Maven Repository` 에 접속할 수 있다.

  https://mvnrepository.com/

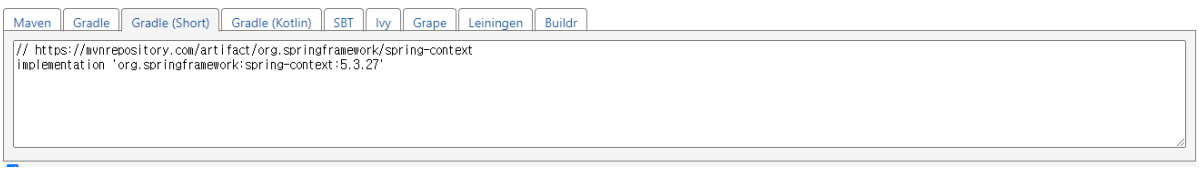
- 검색 창에 사용하고자 하는 라이브러리를 검색한 뒤 검색 결과 중 알맞은 라이브러리를 클릭한다.



- 해당 라이브러리의 다양한 저장소와 버전을 확인할 수 있다. 원하는 저장소의 버전을 클릭한다.



- 하단의 Maven, Gradle 등의 Tab을 클릭하여 원하는 구문을 복사한다. 여기서는 프로젝트 생성 시 `Gradle DSL` : Gradle을 선택했기 때문에 `Gradle(Short)` 탭을 선택한다.



4-2-4. build.gradle 설정

- Project의 directory중 `build.gradle` 파일을 열어 `Maven Repository` 에서 복사한 구문을 붙여 넣어 아래와 같이 작성한다.

```
dependencies {

    implementation 'org.springframework:spring-context:5.3.27'

    ...생략
}
```

- 우측 상단의 `Load Gradle Change` 버튼을 눌러 `dependencies` 의 변경 사항을 반영시킨다.

build.gradle (chap01-ioc-container)

```
plugins {
    id 'java'
}

group = 'com.ohgiraffers'
version = '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {

    implementation 'org.springframework:spring-context:5.3.27'

    testImplementation platform('org.junit:junit-bom:5.9.1')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}

test {
    useJUnitPlatform()
}
```

⇒ 해당 프로젝트에서 사용 될 라이브러리가 자동으로 다운로드 되고 환경에 등록 된다.

4-3. Lombok 사용

4-3-1. Lombok이란?

💡 Lombok은 자바 클래스에서 반복적으로 작성되는 getter, setter, toString, 생성자 코드 등의 소스들을, 어노테이션을 사용하여 생략할 수 있도록 컴파일 시점에 자동으로 생성해주는 라이브러리이다.

4-3-2. Lombok Annotation 종류

- `@Getter` : 해당 멤버 변수에 대한 getter를 생성한다.
- `@Setter` : 해당 멤버 변수에 대한 setter를 생성한다.
- `@ToString` : 해당 클래스의 toString()을 자동으로 생성한다.
- `@EqualsAndHashCode` : 해당 클래스의 equals()와 hashCode()를 자동으로 생성한다.
- `@NoArgsConstructor` : 파라미터가 없는 생성자를 생성한다.
- `@AllArgsConstructor` : 모든 멤버 변수를 파라미터로 받는 생성자를 생성한다.
- `@RequiredArgsConstructor` : final이나 @NonNull인 멤버 변수만 파라미터로 받는 생성자를 생성한다.
- `@Data` : @ToString, @EqualsAndHashCode, @Getter, @Setter, @RequiredArgsConstructor을 모두 적용한 것과 같다.

4-3-3. Lombok 사용 방법

Lombok 라이브러리를 사용하려면 다음과 같은 단계를 진행해야한다.

1. IntelliJ에 Lombok 플러그인 설치 (2020.03 버전 이후에는 설치 되어 있어 별도의 설치가 필요 없다.)
   a. IntelliJ 실행
   b. `File` > `Settings` 클릭
   c. `Plugins` 클릭
   d. `Marketplace` 탭에서 `Lombok` 검색
   e. `Lombok Plugin`을 설치하고 IntelliJ를 다시 시작
2. 프로젝트에 Lombok 의존성 추가
   a. `build.gradle` 파일에 다음 코드를 추가

```
dependencies {

    implementation 'org.projectlombok:lombok:1.18.26'
    annotationProcessor 'org.projectlombok:lombok:1.18.26'

    ...생략
}
```

b. `build.gradle` 파일을 저장하고, IntelliJ에서 `Reload Changes` 버튼을 누른다.

3. 사용할 Lombok 어노테이션 적용

a. 사용할 Lombok 어노테이션을 클래스에 import 한다.

```
import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@ToString
public class MyClass {
    private String name;
    private int age;
}
```

b. Lombok 어노테이션을 적용한 클래스를 사용하면, getter, setter, toString 등의 메서드를 별도로 작성하지 않아도 된다.

```
MyClass myClass = new MyClass();
myClass.setName("John");
myClass.setAge(30);

System.out.println(myClass.getName()); // "John" 출력
System.out.println(myClass.getAge()); // 30 출력
System.out.println(myClass); // "MyClass(name=John, age=30)" 출력
```