2023년 2월 14일
Spring
보고서

Spring

임원호

-목차-

[참고 문헌]

Ⅰ. Spring

1. Spring의 정의

Java의 웹 프레임워크로 Java 언어를 기반으로 작성한다. Java를 이용해 다양한 웹 어플리케이션을 만드는 데 사용되는 프로그래밍의 틀이며, Java의 활용도가 높아지며, JSP, Mybatis, JPA 등의 기술들이 생겨났다. Spring은 다른 사람의 코드를 참조하기 쉽고 편리한 구조로 되어있으며, 앞에서 말한 기술들을 더 쉽고 간편하게 사용하게 해주는 오픈소스 프레임 워크이다. Spring 프레임 워크는 경량 컨테이너로 자바 객체를 담고 직접 관리를 합니다. 객체의 생성 및 소멸 그리고 라이프 사이클을 관리하며 언제든지 Spring

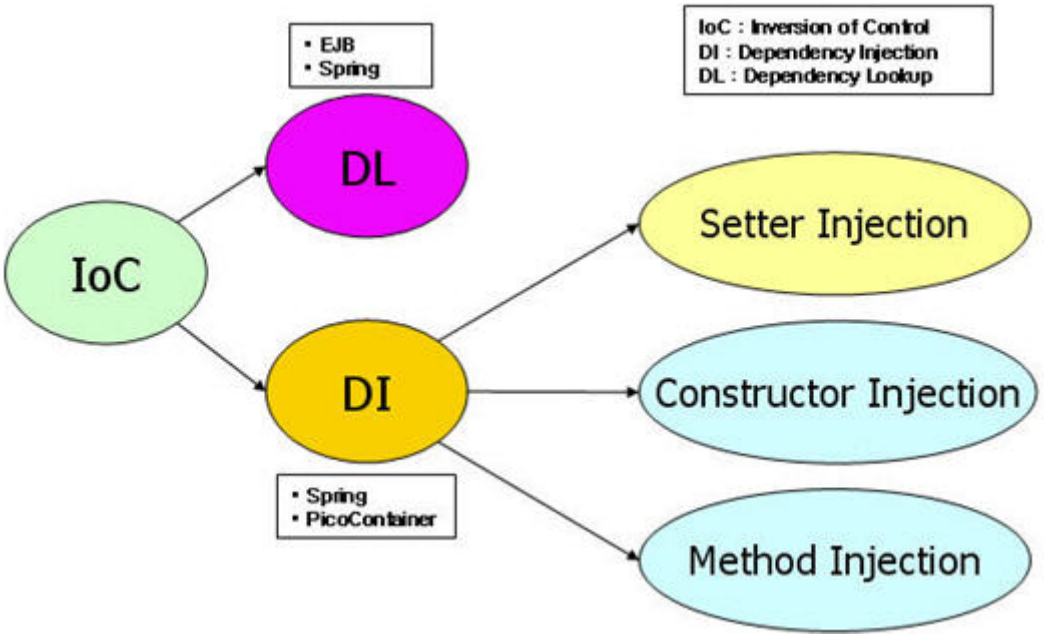
컨테이너로부터 필요한 객체를 가져와 사용할 수 있습니다. “로드 존슨이 2002년에 출판한 저서 Expert One-on-One J2EE Design and Development에 선보인 코드를 기반으로 시작하여, 점점 발전하게 되었다.”[1]

2. Spring은 왜 사용 할까?

Spring은 동적 웹 개발을 위해 사용되는 도구들의 집합체라고 생각한다. 거기서 Spring 프레임 워크등 가볍고 필수적인 것만 담은 프레임 워크도 탄생하게 되었다. 프레임 워크란 어떠한 목적을 달성하기 위해, 복잡하게 얽힌 문제를 쉽게 해결하기 위해 사용하는 약속이나 도구이며, 개발할 때의 뼈대 역할이다. 또한, 자주 쓰고 있는 유틸들의 모음이다. 우리가 일반적으로 PS를 할 때, 정렬, 스택, 큐, 덱과 같이 모든 것을 구현하고 문제를 풀지는 않는다. 이처럼 프레임 워크는 PS와 동일하게 필요한 로직을 짜고 나머지는 모두 지원해준다.

Ⅱ. Spring IOC

1. IOC 란?



Inversion of Control의 약자로 제어의 역전이다. 지금까지 프로그램을 하나 만들 때 객체를 결정 및 생성 -> 의존성 객체 생성 -> 객체 내의 메소드 호출하는 작업을 반복했다. 각 객체가 프로그램의 흐름을 결정하고 각 객체를 구성하는 작업에 직접적으로 참여한 것이다. 모든 작업을 사용자가 직접 제어하는 구조였다면, IOC를 통해 흐름의 구조를 반전시킨다. IOC에서는 객체는 자기가 사용할 객체를 선택하거나 생성하지 않는다. 또한 자신이 어디서 사용되는지 모르게 된다. 자신의 모든 권한을 다른 대상에 위임해 제어 권한을 가진 특별한 객체에 의해 결정되고 만들어지게 된다. 즉 제어의 흐름을 사용자가 컨트롤하지 않고, 이를

[1] 위키피디아, “스프링 프레임 워크”, https://ko.wikipedia.org/wiki/스프링_프레임워크, (2023.02.14.최종검색).

제어하는 특별한 객체를 만들어 특별한 객체가 객체의 생성 수정 삭제, 라이프 사이클을 관리하는 것을 말한다. IOC는 DI와 DL에 의해 구현이 된다.

Ⅲ. Spring DI

1. DI 란?

DI는 Dependency Injection의 줄임말로 의존 관계 주입(의존성 주입)이라 한다. A가 B를 의존한다는 말은.. B가 변하게 되면, A에 영향을 끼친다는 말이다. 정말 대표적인 예시를 보면, 햄버거 가게 요리사는 햄버거 레시피에 의존한다. 라고 했을 때, 햄버거의 레시피가 변화하게 되면, 변화된 레시피에 요리사는 햄버거 만드는 방법을 수정해야 한다. 레시피의 변화가 요리사의 행위에 영향을 미쳤기 때문에 요리사는 햄버거 레시피에 의존한다고 말할 수 있다. 의존관계를 인터페이스로 추상화하게 된다면, 요리사가 다양한 레시피에 의존 관계를 맺을 수 있고, 실제 구현 클래스와 관계가 느슨해짐과 결합도가 낮아진다. 그렇다면 의존성 주입은 무엇인가? '바로 제 3자를 추가해, 요리사가 의존하고 있는 햄버거 레시피를 사장이 보고 결정하고 주입하는 것이다. 이처럼 외부에서 결정하고 주입하는 것이 DI이다.'[2]
토비의 스프링에서는 다음의 세 가지 조건을 충족하는 작업을 의존관계 주입이라고 한다.
클래스 모델이나 코드에는 런타임 시점의 의존관계가 드러나지 않는다. 그러기 위해서는 인터페이스만 의존하고 있어야 한다. 런타임 시점의 의존관계는 컨테이너나 팩토리 같은 제3의 존재가 결정한다. 의존관계는 사용할 오브젝트에 대한 레퍼런스를 외부에서 주입해줌으로써 만들어진다.

2. DI를 통한 의존 처리

DI를 구현하는 방법은 생성자를 이용, 메소드를 이용, Setter를 이용하는 것이다.

3. DI와 의존 객체 변경의 유연함

의존성이 줄어든다. 의존성이 강하다는 것은 의존 대상이 변경될 경우 끼칠 영향에 대해 취약하다는 것을 의미한다. 즉 대상이 바뀌면, 이에 맞춰 영향을 받는 객체도 수정을 해줘야 한다. 하지만 DI로 구현하게 되면, 대상이 변하더라도 구현 자체를 수정할 일이 없거나 줄어들게 된다.
재사용성이 높은 코드가 된다. A가 B를 의존할 때 B를 A의 내부에서만 사용되었던 게, B를 별도로 구분하여 인터페이스화 시켜 다른 클래스 내에서도 사용이 가능해진다.
테스트하기 좋은 코드가 된다. A와 B를 분리하여 테스팅이 가능해진다. A가 B의 영향을 받을 때 테스트를 할 때마다 A를 변경하게 되면, 테스트하는데 아주 난감해질 것이다. 이런 경우를 없애준다.
가독성이 높아진다. 각각의 기능을 나누어 클래스로 분리해두면, 자연스럽게 가독성이 높아지게 된다.

---

[2] Teoble, "DI란" https://tecoble.techcourse.co.kr/post/2021-04-27-dependency-injection/, (2023.02.14.최종검색).

Ⅳ. Spring DL

1. DL 이란?

모든 IOC 컨테이너가 각 컨테이너에서 관리해야 하는 객체들을 관리하기 위한 별도의 저장소를 가지게 된다. Bean에 직접 접근하기 위해서, 컨테이너에서 제공하는 API를 이용하여 사용하고자 하는 Bean을 LookUp하는 것으로 컨테이너 API와 의존관계를 많이 가지면 가질수록 어플리케이션에 종속되는 단점이 있다.

Ⅴ. Spring POJO

1. POJO 란?

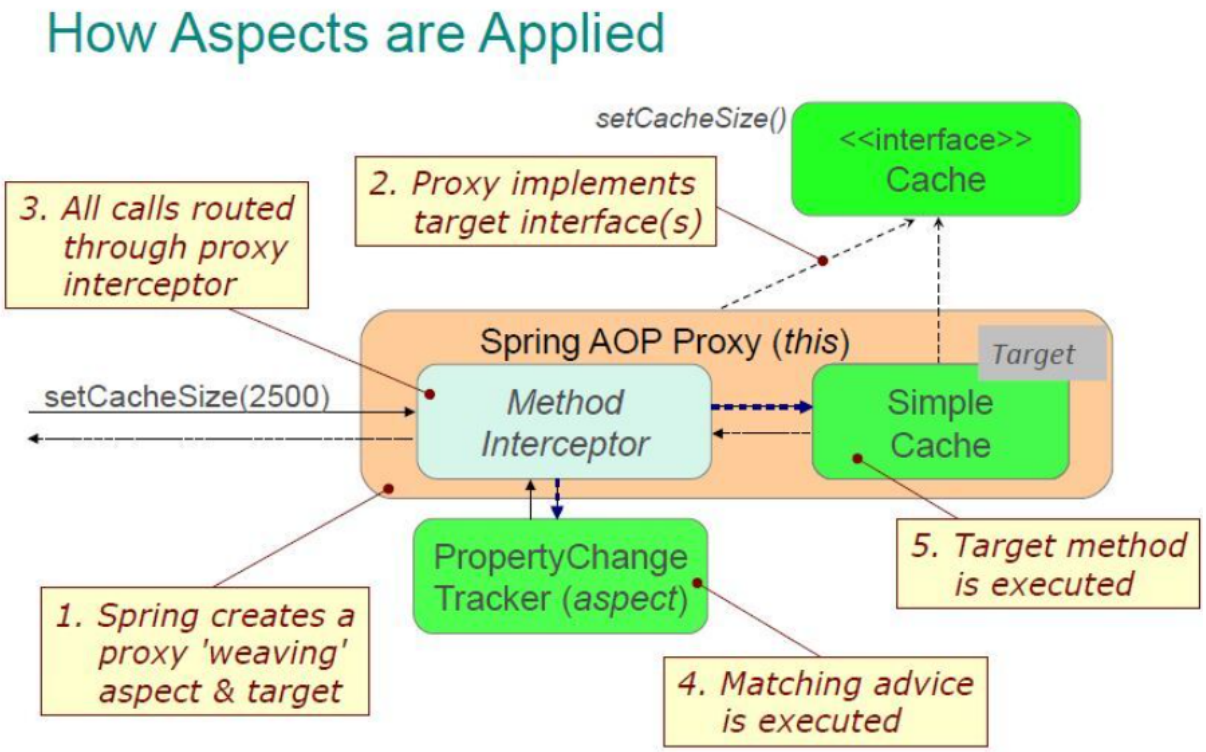
(그림 : https://incheol-jung.gitbook.io/docs/q-and-a/spring/spring-1)

POJO는 Plain Old Java Object로 순수한 자바 오브젝트를 뜻한다. 이전까지는 Java Beans를 통해 재사용이 가능한 로직을 개발하였지만, 하나의 기능을 위해 불필요한 복잡한 로직이 과도하게 들어가는 단점이 있다. 그래서 다시 조명을 받는게 POJO인데, 이 개념은 자바를 처음 시작하고 프로젝트를 만들 때 사용해 보았다. 바로 Getter/Setter를 가지는 단순 객체를 말한다. 이 객체는 의존성이 없고 추후 테스트 및 유지보수가 편리한 유연성의 장점을 가지게 된다. POJO의 기반한 프레임 워크가 조명을 받고 있고, Spring 프레임워크에서는 이러한 POJO를 지원하고 Spring 홈페이지에는 POJO를 사용함으로써, 당신의 코드는 더욱 간단해졌고, 그로 인해 테스트하기 좋으며, 유연하고, 요구사항에 따라

기술적 선택을 바꿀 수 있도록 바뀌었다. 라고 되어있다. POJO는 가장 단순화하게 추상화한 객체로 보면 될 것 같다.

Ⅵ. AOP

1. AOP 란?



AOP는 Aspect Oriented Programming의 약자로 관점 지향 프로그래밍이다. 대부분의 소프트웨어 개발 프로세스에서 사용하는 방법은 OOP 즉 객체 지향 원칙에 따라 관심사가 같은 데이터를 한곳에 모아 분리하고, 낮은 결합도를 갖게 하여 독립적이며, 유연한 모듈로 캡슐화를 하는 것을 말한다. 하지만 이런 과정 중 중복된 코드들의 문제와 가독성이 떨어지고 확장성, 유지보수성이 모두 떨어지게 된다. 이런 문제를 보완하기 위해 나온 것이 AOP이다. 'AOP는 핵심 기능과 공통 기능을 분리시켜 핵심 로직에 영향을 끼치지 않게 공통 기능을 끼워 넣는 개발 형태이며 이렇게 개발함에 따라 무분별하게 중복되는 코드를 한곳에 모아 코드를 제거할 수 있고, 공통 기능을 한곳에 보관함으로 공통 기능 하나의 수정으로 모든 핵심 기능들의 공통 기능을 수정할 수 있어야 효율적인 유지보수가 가능하며, 재활용성이 극대화된다.'[3] Spring에서는 AOP를 편리하게 사용할 수 있도록 이를 지원하고 있다.

[3] khj93, "AOP 특징", https://khj93.tistory.com/entry/Spring-Spring-Framework란-기본-개념-핵심-정리, (2023.02.14.최종검색).

[참고 문헌]

위키피디아, “스프링 프레임 워크”, https://ko.wikipedia.org/wiki/스프링_프레임워크, (2023.02.14.최종검색).

Teoble, “DI란” https://tecoble.techcourse.co.kr/post/2021-04-27-dependency-injection/, (2023.02.14.최종검색).

khj93, “AOP 특징”, https://khj93.tistory.com/entry/Spring-Spring-Framework란-기본-개념-핵심-정리, (2023.02.14.최종검색).