[Spring] Spring Framework란? 기본 개념 핵심 정리

Spring Framework란?

자바 플랫폼을 위한 오픈소스 애플리케이션 프레임워크로서 엔터프라이즈급 애플리케이션을 개발하기 위한 모든 기능을 종합적으로 제공하는 경량화된 솔루션입니다.

엔터프라이즈급 개발이란 뜻대로만 풀이하면 기업을 대상으로 하는 개발이라는 말입니다. 즉, 대규모 데이터 처리와 트랜잭션이 동시에 여러 사용자로 부터 행해지는 매우 큰 규모의 환경을 엔터프라이즈 환경이라 일컫습니다.

Spirng Framework는 경량 컨테이너로 자바 객체를 담고 직접 관리합니다. 객체의 생성 및 소멸 그리고 라이프 사이클을관리하며 언제든 Spring 컨테이너로 부터 필요한 객체를 가져와 사용할 수 있습니다.

이는 Spirng이 IOC 기반의 Framework임을 의미합니다.

Spring Framework는 IOC기반이다. IOC란?

IOC는 Inversion of Control의 약자로 말 그대로 제어의 역전입니다. 그럼 제어의 역전이란 무엇일까요?

일반적으로 지금까지 프로그램은

객체 결정 및 생성 -> 의존성 객체 생성 -> 객채 내의 메소드 호출 하는 작업을 반복했습니다.

이는 각 객체들이 프로그램의 흐름을 결정하고 각 객체를 구성하는 작업에 직접적으로 참여한 것입니다.

즉, 모든 작업을 사용자가 제어하는 구조인 것입니다.

하지만 IOC에서는 이 흐름의 구조를 바꿉니다. IOC에서의 객체는 자기가 사용할 객체를 선택하거나 생성하지 않는다. 또한 자신이 어디서 만들어지고 어떻게 사용되는지 또한 모릅니다. 자신의 모든 권한을 다른 대상에 위임함으로 써 제어권한을 위임받은 특별한 객체에 의해 결정되고 만들어집니다.

즉, 제어의 흐름을 사용자가 컨트롤 하지 않고 위임한 특별한 객체에 모든 것을 맡기는 것입니다.

IOC란 기존 사용자가 모든 작업을 제어하던 것을 특별한 객체에 모든 것을 위임하여 객체의 생성부터 생명주기 등 모든 객체에 대한 제어권이 넘어 간 것을 IOC, 제어의 역전 이라고 합니다.

IOC의 구성요소 DI와 DL

IOC는 DI와 DL의 의해 구현됩니다.

DL(Dependency Lookup) - 의존성 검색

컨테이너에서는 객체들을 관리하기 위해 별도의 저장소에 빈을 저장하는데 저장소에 저장되어 있는 개발자들이 컨테이너에서 제공하는 API 를 이용하여 사용하고자 하는 빈 을 검색하는 방법입니다.

DI(Dependency Injection) - 의존성 주입

의존성 주입이란 객체가 서로 의존하는 관계가 되게 의존성을 주입하는 것입니다. 객체지향 프로그램에서 의존성 이란 하나의 객체가 어떠한 다른 객체를 사용하고 있음을 의미합니다. 그렇다면 IOC에서의 DI는 무엇일까요?

바로 **각 클래스 사이에 필요로 하는 의존관계를 빈 설정 정보를 바탕으로 컨테이너가 자동으로 연결**해 주는 것입니다.

Spring Framework의 특징 POJO

POJO(Plain Old Java Object) 란 말 그대로 평범한 자바 오브젝트입니다.

이전 EJB(Enterprise JavaBeans)는 확장 가능한 재사용이 가능한 로직을 개발하기 위해 사용 되었었는데 EJB는 한가지 기능을 위해 불필요한 복잡한 로직이 과도하게 들어가는 단점이 있었습니다. 그래서 다시 조명을 받은게 POJO입니다. POJO는 gettet/setter를 가진

단순 자바 오브젝트로 정의를 하고 있습니다. 이러한 단순 오브젝트는 의존성이 없고 추후 테스트 및 유지보수가 편리한 유연성의 장점을 가집니다. 이러한 장점들로 인해 객체지향적인 다양한 설계와 구현이 가능해지고 POJO의 기반의 Framework가 조명을 받고 있습니다.

Spring Framework 에서는 이러한 POJO을 지원하고 Spirng 홈페이지에는 이러한 글도 있습니다.

"POJO를 사용함으로써, 당신의 코드는 더욱 심플해졌고, 그로인해 테스트 하기에 더 좋으며, 유연하고, 요구사항에 따라 기술적 선택을 바꿀수 있도록 바뀌었다."

Spring Framework의 특징 AOP

AOP(Aspect Oriented Programming)란 말 그대로 관점 지향 프로그래밍입니다.

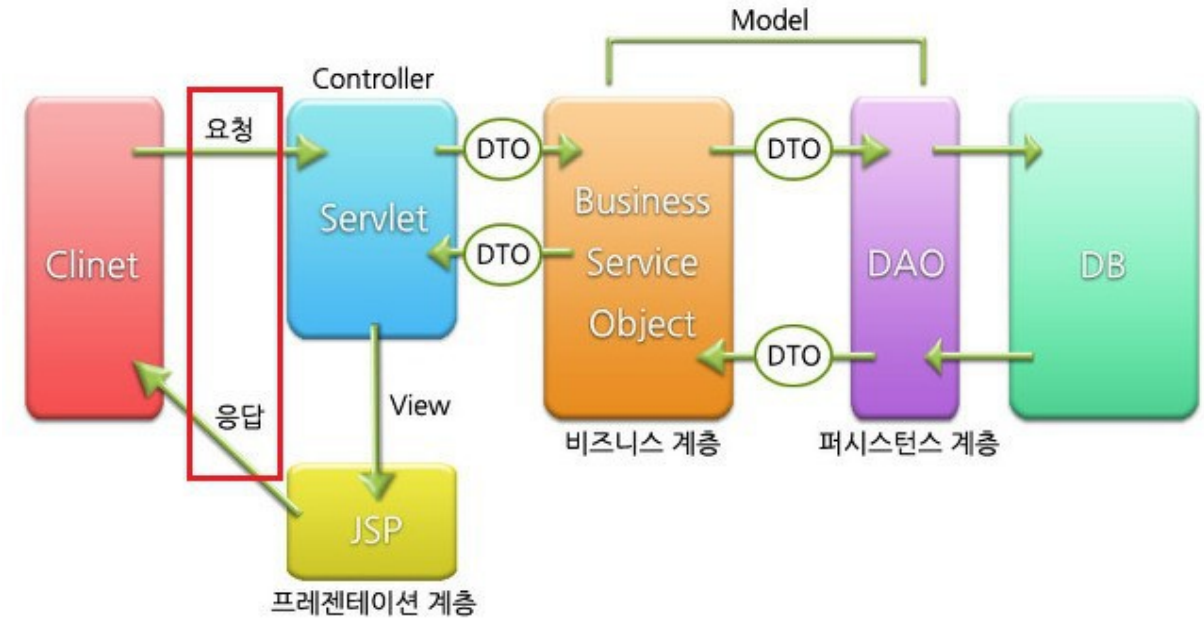
대부분 소프트웨어 개발 프로세스에서 사용하는 방법은 OOP(Object Oriented Programming) 입니다.

OOP는 객체지향 원칙에 따라 관심사가 같은 데이터를 한곳에 모아 분리하고 낮은 결합도를 갖게하여 독립적이고 유연한 모듈로 캡슐화를 하는 것을 일컫습니다. 하지만 이러한 과정 중 중복된 코드들이 많아지고 가독성, 확장성, 유지보수성을 떨어 뜨립니다. 이러한 문제를 보완하기 위해 나온 것이 AOP입니다.

AOP에서는 핵심기능과 공통기능을 분리시켜 핵심 로직에 영향을 끼치지 않게 공통기능을 끼워 넣는 개발 형태 이며 이렇게 개발함에 따라 **무분별하게 중복되는 코드를 한 곳에 모아 중복 되는 코드를 제거 할 수 있어지고 공통기능을 한 곳에 보관함으로써 공통 기능 하나의 수정으로 모든 핵심기능들의 공통기능을 수정 할 수 있어 효율적인 유지보수가 가능하며 재활용성이 극대화**됩니다.

물론 AOP로 만들 수 있는 기능은 OOP로 구현 할 수 있는 기능이지만 Spring에서는 AOP를 편리하게 사용 할 수 있도록 이를 지원하고 있습니다.

Spring Framework의 특징 MVC (Model2)



MVC란 (Model View Controller) 구조로 사용자 인터페이스와 비지니스 로직을 분리하여 개발 하는 것 입니다. MVC에서는 Model1과 Model2로 나누어져 있으며 일반적인 MVC는 Model2를 지칭합니다.

Model

Model에서는 데이터처리를 담당하는 부분입니다. Model부분은 Serivce영역과 DAO영역으로 나누어지게 되고 여기서 중요한 것은 Service 부분은 불필요하게 HTTP통신을 하지 않아야하고 request나 response와 같은 객체를 매개변수로 받아선 안된다. 또한 Model 부분의 Service는 view에 종속적인 코드가 없어야 하고 View 부분이 변경되더라도 Service 부분은 그대로 재사용 할 수 있어야 한다.

Model에서는 View와 Controller 어떠한 정보도 가지고 있어서는 안된다.

View

View는 사용자 Interface를 담당하며 사용자에게 보여지는 부분입니다. View는 Controller를 통해 모델에 데이터에 대한 시각화를 담당하며 View는 자신이 요청을 보낼 Controller의 정보만 알고 있어야 하는 것이 핵심이다.

Model이 가지고 있는 정보를 저장해서는 안되며 Model, Controller에 구성 요소를 알아서는 안된다.

Controller

Controller에서는 View에 받은 요청을 가공하여 Model(Service 영역)에 이를 전달한다. 또한 Model로 부터 받은 결과를 View로 넘겨주는 역할을 합니다. Controller에서는 모든 요청 에러와 모델 에러를 처리하며 View와 Controller에 정보를 알고 있어야한다.

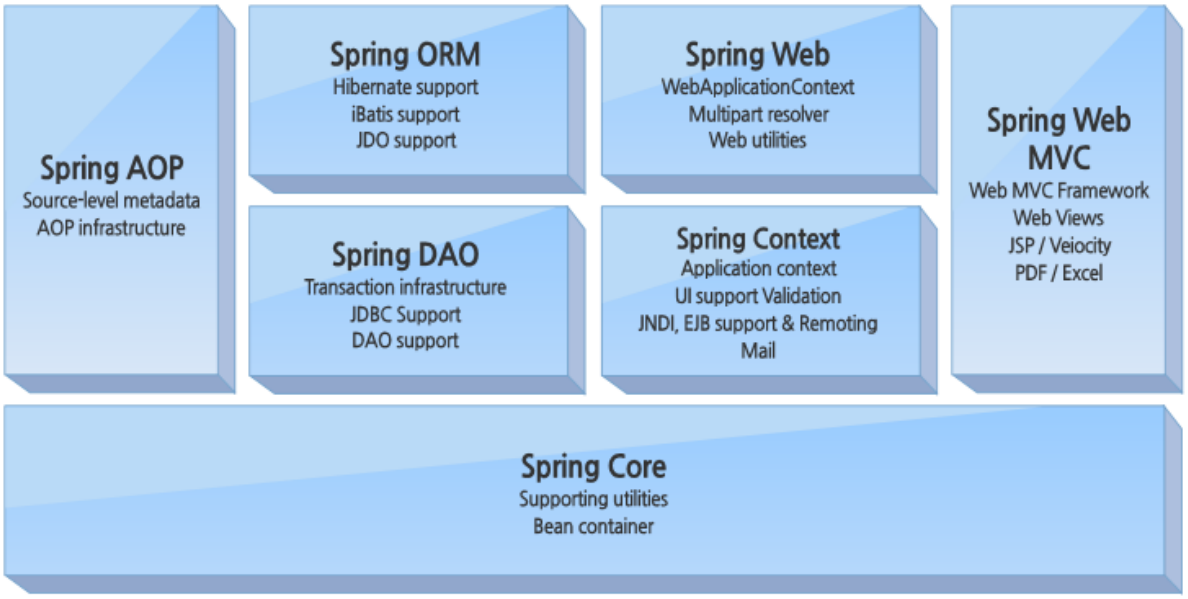
Model과 View의 정보에 대해 알고 있어야한다.

이렇게 Model, View, Controller를 나누는 이유는 소스를 분리함으로서 각 소스의 목적이 명확해 지고 유지보수하는데 있어서 용이하기 때문이다. Model의 Service영역은 자신을 어떠한 Controller가 호출하든 상관없이 정해진 매개변수만 받는다면 자신의 비즈니스 로직을 처리할 수 있어야한다.

즉, 모듈화를 통해 어디서든 재사용이 가능하여야 한다는 뜻이다.

이말은 View의 정보가 달라지더라도 Controller에서 Service에 넘겨줄 매개변수 데이터 가공만 처리하면 되기 때문에 유지보수 비용을 절감 할 수 있는 효과가 있다. 또한 Service 영역의 재사용이 용이하기 때문에 확장성 부분에서도 큰 효과를 볼 수 있는 장점이있다.

Spring Framework의 구조



Spring Core

Spring Core는 Spring Container을 의미합니다. core라는 말 그대로 Container는 Spring Framework의 핵심이며 그중 핵심은 Bean Factory Container입니다. 그 이유는 바로 Bean Factory는 IOC패턴을 적용하여 객체 구성 부터 의존성 처리까지 모든 일을 처리하는 역할을 하고 있기 때문입니다.

Spring Context

Spring context는 Spring Framework의 context 정보들을 제공하는 설정 파일입니다. Spring Context에는 JNDI, EJB, Validation, Scheduiling, Internaliztaion 등 엔터프라이즈 서비스들을 포함하고 있습니다.

Spring AOP

Spring AOP module은 Spring Framework에서 관점지향 프로그래밍을 할 수 있고 AOP를 적용 할수 있게 도와주는 Module입니다. 해당 AOP에 대한 내용은 위에서 설명 했기 때문에 넘어 가도록 하겠습니다.

Spring DAO

DAO란 Data Access Object의 약자로 Database Data에 접근하는 객체입니다. Spring JDBC DAO는 추상 레이어를 지원함으로써 코딩이나 예외처리 하는 부분을 간편화 시켜 일관된 방법으로 코드를 짤 수 있게 도와줍니다.

Spring ORM

ORM이란 Object relational mapping의 약자로 간단하게 객체와의 관계 설정을 하는 것입니다. Spring에서는 Ibatis, Hibernate, JDO 등 인기있는 객체 관계형 도구(OR도구)를 사용 할 수 있도록 지원합니다.

Spring Web

Spirng에서 Web context module은 Application module에 내장되어 있고 Web기반의 응용프로그램에 대한 Context를 제공하여 일반적인 Web Application 개발에 필요한 기본적인 기능을 지원합니다. 그로인해 Jakarta Structs 와의 통합을 지원하고 있습니다.

Spring MVC

Spring에서는 MVC에서는 Model2 구조로 Apllication을 만들 수 있도록 지원합니다. MVC (Model-View-Controller) 프레임 워크는 웹 응용 프로그램을 작성하기위한 완전한 기능을 갖춘 MVC를 구현합니다. MVC 프레임 워크는 전략 인터페이스를 통해 고급 구성 가능하며 JSP, Velocity, Tiles, iText 및 POI를 포함한 수많은 뷰 기술을 지원하고 있습니다.