**< Basic Concept of Framework and Spring >**

**Spring** : 자바 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 어플리케이션 프레임워크

1. 일반적으로 Framework란 특정 계층이나, 기술 업무에 국한된 목표를 가지고 만들어짐.

예를 들어, 관계형 DB와 자바 Object를 매핑해주는 ORM 기술을 제공하는 것과 같다. 하지만, Spring은 Application Framework라는 특성을 가진다. 이는 Applicatioin 개발의 전 과정을 빠르고 효율적으로 처리하게 해준다. 스프링 MVC Framework 또는 JDBC/ORM 지원의 Framework라는 것은 일부 영역, IoC/DI 프레임워크나 AOP 툴이라고 보는 이유는 스프링 제공의 핵심 영역만 본 것.

1. Tomcat, Jetty에서도 작동이 가능하다. EJB나 WAS를 갖추지 않아도 가능하다.
2. Based on Open Source

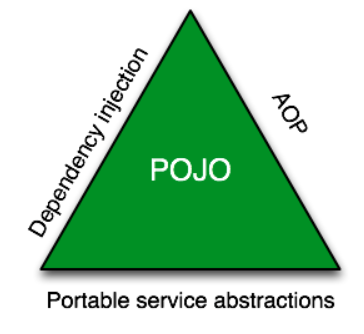
**OO와 DI (Dependency injection)** : 레고의 사람을 생각하면 바지를 막 바꿔 낄 수 있다. 머리통도 바꿔낄 수 있다. 팔에 막 무기도 쥐어줄수 있고 빼서 다른걸 낄 수 있다. 이게 DI이다. DB랑 연동부분 (DAO)에서 JDBC를 사용하는 사람, iBatis, JPA 등의 프레임 워크를 이용해서 짜는 사람이 있다. 이때 Spring을 이용하면 그런 부분을 필요시에만 뺐다 끼웠다가(similar to injection) 할 수 있다. 때문에 DI는 자연스럽게 OO적 설계와 개발로 좋은 동반자이다.

**IOC** : object의 생성부터 소멸까지의 life-cycle 관리를 개발자 대신 spring이 대신하게 하는 것.

**POJO 프로그래밍** : 스프링의 핵심 개발자들이 함께 쓴 Professional Spring Framework

스프링의 정수는 엔터프라이즈 서비스 기능을 POJO에 제공하는 것이다.

스프링 소스의 CTO가 스프링의 핵심개념을 설명하기 위해 만든 그림이다.

 스프링은 POJO를 이용해서 만든 어플리케이션 코드와 POJO가 어떻게 관계를 맺고 동작하는지를 정의해 놓은 설계정보로 구분된다.

**POJO (Plain Old Java Object)**

* EJB처럼 복잡하고 제한이 많은 기술을 사용하는 것 대신에 자바의 단순한 오브젝트를 이용해 비즈니스 로직을 구현하는 것

POJO의 조건

1 . 특정 규약에 종속되지 않는다.

* Java를 제외한 꼭 필요한 API 외에는 종속되지 않는다.
* 이는 규약이 적용된 환경은 종속적이기 때문이라고 함.

2 . 특정 환경에 종속되지 않는다.

POJO의 Framework

* Spring은 POJO를 이용한 Enterprise application 개발을 목표로 하는 Framework
* POJO programming이 가능하도록 기술적인 기반을 제공하는 Framework를 POJO Framework라고 한다. 대표적으로 Spring Framework와 Hibernate가 대표적이다.
* 주로 DB 이용 기술에 POJO를 적용하는 것을 목표로 하는 Hibernate
* 스프링은 모든 영역과 계층에서 POJO 방식 구현이 목표

**< what kind of DB-plug using for connect to Spring? >**

**[JPA와 ORM]**

JPA란 자바 ORM 기술에 대한 API 표준 명세.

JPA는 ORM을 사용하기 위한 Interface를 모아둔 것. JPA를 사용하기 위해서는 JPA를 구현한 Hibernate, EclipseLink, DataNucleus와 같은 ORM Framework를 사용해야 한다.

ORM을 사용하면 SQL Query가 아닌 Method를 이용해 데이터 조작이 가능하다.

SELECT \* FROM user가 아니라, user.findAll()이라는 method call로 대체 가능하다.

쿼리 작성이 아니라, method call이므로 생산성은 높다. 그러나 쿼리가 복잡해지면 ORM으로 표현하는데 한계가 있고, 성능이 raw query보다 느리다.

**1 . Hibernate**

* 한국에서는 MyBatis가 주를 이루는데, 외국은 높은 생산성으로 인해 Hibernate를 많이 사용한다.
* Hibernate를 쓰지 않는 이유 – 단점

Hibernate는 자체적으로 쿼리를 생성하고, OR Mapper로써, Object를 Db에서 로딩할 때, referenced object를 로딩하는 등 특성을 제대로 이해하지 않고 사용하면 장애를 일으킴.

* Hibernate는 자바 기반의 ORM (Object Relationship Mapper)이다.

즉 자바 객체를 RDBMS의 하나의 row로 맵핑해준다. 이 개념은 오래되었는데 EJB Entity Beans가 그 시초이다. 넘 어려워서 근래는 잘 쓰지 않는다. Hibernate 개발 후, java 스펙에도 ORM의 표준 스펙을 정의했고 이것이 JPA(Java Persistence API)이다. ORM API를 JPA를 통해서 추상화 시켜놓음으로써, 다른 ORM과 쉽게 교체가 가능하지만, 실제로 ORM은 Hibernate만 사용되기 때문에 JPA를 사용하는게 큰 의미가 없다.

**2 . Mybatis**

* DAO 개발이 매우 반복되는 작업이다.
* 테이블에 칼럼이 하나 추가된다면 이와 관련된 모든 DAO의 SQL문을 수정해야한다.
* 즉 DAO와 테이블은 강한 의존성을 가지고 있다.
* SQL을 자동으로 생성해주는 툴인데, 반복성은 여전히 가지고 있다.
* 객체 모델링보단 테이블설계를 우선시 했음. (즉 OO적이지 못함)

동북아 제외하고는 다 hibernate임.

JDBC를 직접 사용하는 Mybatis보다 JDBC를 노출하지 않고 ORM 기술을 사용하는 JPA가 좋은 방식일 수 있다는 말이다.

서비스에 따라 결정해야 한다.

**< 자신에게 맞는 툴 고르는 기준 >**

* SI, 금융시장 🡪 비즈니스가 복잡하고, 안정성 중시 🡺 JPA보다 SQL 작성이 좋다.
* JPA는 통계 쿼리처럼 복잡한 SQL을 수행하기 힘들기 때문에, 서비스에 따라 고른다.
* Hibernate가 SQL 직접 사용하지 않는다해서, JDBC API를 사용하지 않는 것도 아님.
* Hibernate를 지원하는 매서드 내부 JDBC API동작, 개발자가 SQL 작성 안 하는 것 뿐.

\*\* Reference

<http://a07274.tistory.com/200>

<https://12bme.tistory.com/157>

<http://bcho.tistory.com/906>

<http://victorydntmd.tistory.com/195>