Spring Framework

0.Spring Framework

Framework?

- 프로그램의 골격이 되는 기본 코드

- 소프트웨어 개발을 간소화하기 위해 개발됨

- 개발자는 프레임워크를 기반으로 소스코드를 작성하여
  소프트웨어를 완성시키면 된다.

0.Spring Framework

참고

- Solution ?

  특정한 상황에 대한 해결 방안.

  사용자의 요구에 대응되는 H/W , S/W, Skill 등

  (ex. ERP, DBMS, POS 등..)

- Library?

  특정 목적을 위해 사용하는 함수들을 모듈화 시킨 것.

  (*.jar)

0.Spring Framework

Spring?

- EJB(Enterprise JavaBean) 기반 개발 에서
  POJO(Plain Old Java Object) 기반 개발 으로

- Spring framework는 엔터프라이즈 애플리케이션 개발을
  복잡한 EJB가 아닌 POJO를 통해서 개발 할 수 있도록 돕는다.

1.Spring 개요

기본 설명

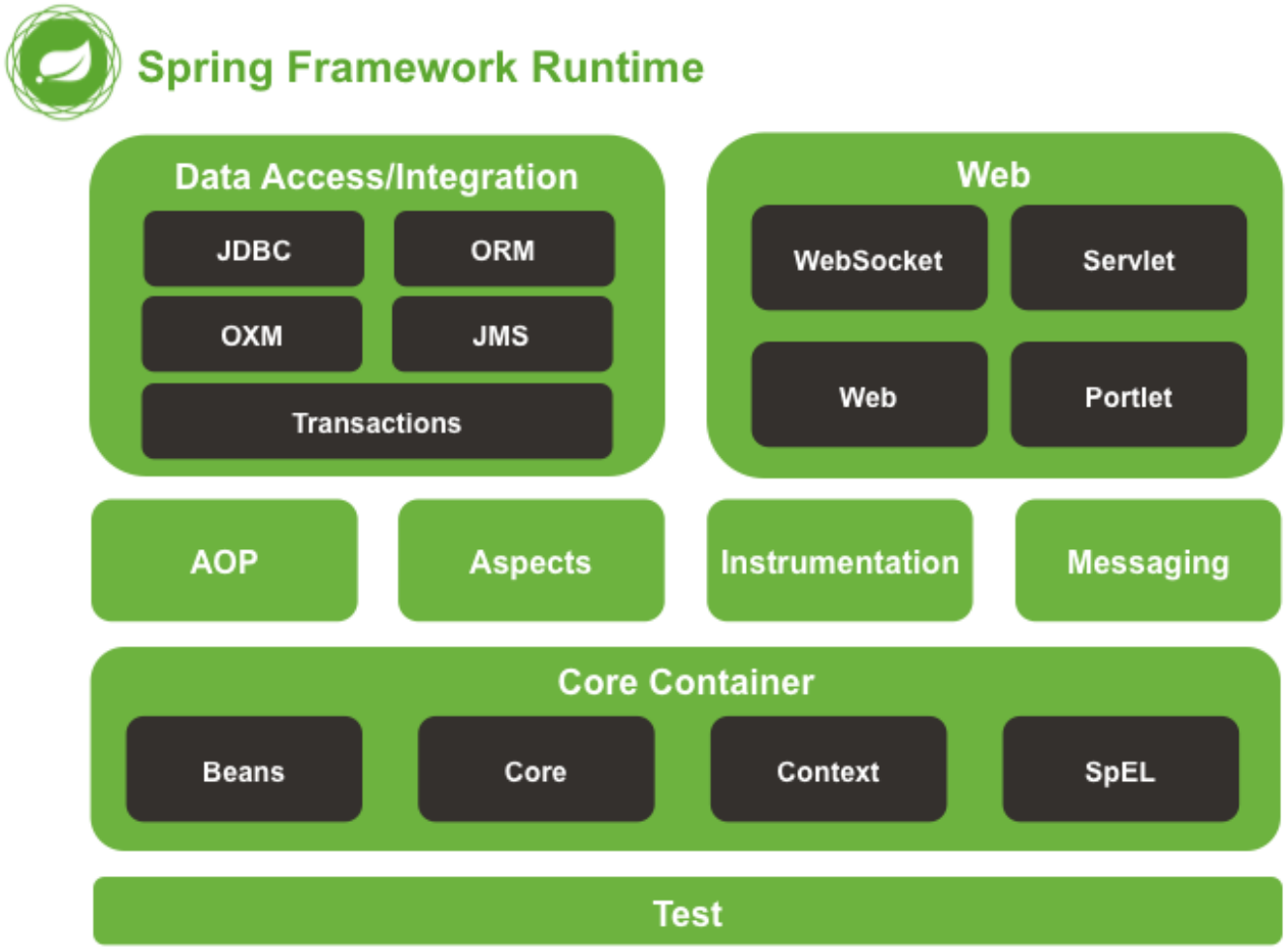
1) 어플리케이션 프레임워크로 불리며, 웹 어플리케이션은 물론 콘솔 어플리케이션이나 스윙과 같은 GUI 어플리케이션 등 어떤 어플리케이션에도 적용 가능한 프레임워크 이다.

2) 스프링은 EJB와 같이 복잡한 순서를 거치지 않아도 간단하게 이용할 수 있기 때문에 '경량 컨테이너' 라고도 부른다.

3) DI(Dependency Injection) 과 AOP(Aspect Oriented Programming), OCP(Open-Closed Principle)을 중점 기술로 사용하고 있지만, 이 외에도 여러가지 기능을 제공한다.

1.Spring 개요

구성 요소(module)



https://spring.io/

1.Spring 개요

구성 요소(module)

Core Container : IOC, DI 기능을 포함하여 프레임워크의 기본 부분 제공
AOP : Aspect 지향 프로그래밍 구현 제공
Aspects : AspectJ와의 통합 제공
Instrumentation : 애플리케이션 서버에서 사용되는 클래스 제공
Messaging : message 기반 애플리케이션의 토대가 되는 모듈
DataAccess/Intergration : JDBC 추상화 계층 제공, 트랜잭션 관리 등.
Web : 웹 어플리케이션 개발에 필요한 기능 제공
Test : 스프링 구성 요소에 대한 유닛 테스트 및 통합 테스트 지원.

1.Spring 특징

3대 용어

DI(Dependency Injection) / IoC(Inversion of Control)

AOP (Asperct Oriented Programming)

OCP(Open Closed Principle)

1.Spring 특징

DI/IoC?

DI(Dependency Injection)

- 객체간의 결합을 느슨하게 하는 스프링의 핵심 기술

※ 의존관계를 관리하는 방법

1. Construction Injection
2. Setter Injection
3. Method Injection

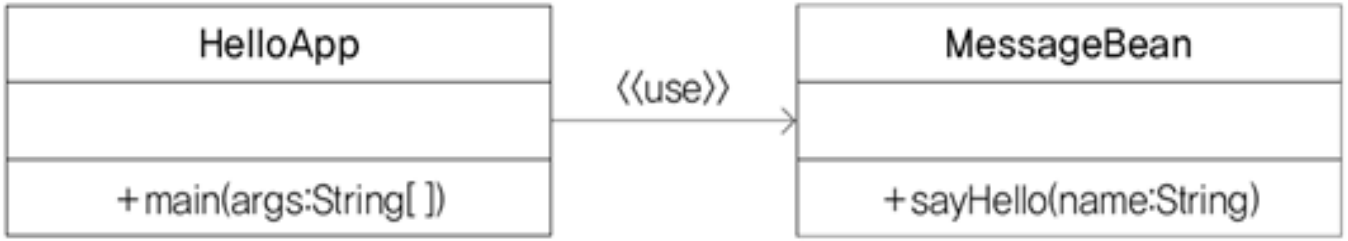
1.Spring 특징

DI/IoC?

강결합 : 객체 간 결합도가 강한 프로그램

- HelloApp 에서 MessageBean을 직접 객체 생성하여 사용하고 있다.

- MessageBean 클래스를 다른 클래스로 변경할 경우,
  HelloApp의 소스를 같이 수정해 주어야 한다.



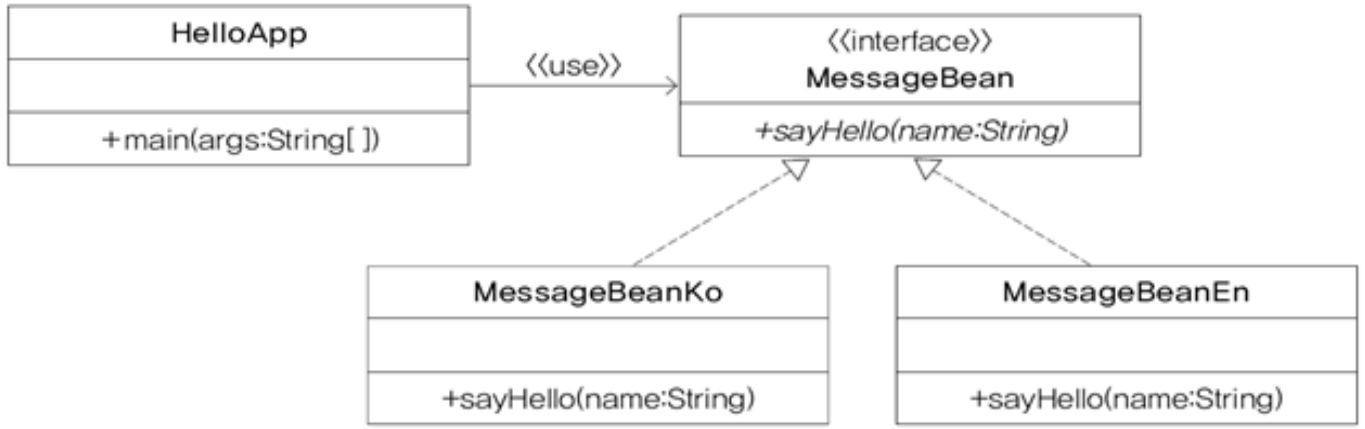
1.Spring 특징

DI/IoC?

약결합 : 인터페이스를 사용하여 객체 간 결합도를 낮춘 프로그램

- HelloApp 은 MessageBean 이라는 인터페이스를 통해 객체를 사용.

- 일반적으로 팩토리 메서드를 활용하여, 사용할 객체(MessageBeanKo or MessageBeanEn)를 생성한다. MessageBean 이라는 이름의 MessageBeanKo의 객체가 생성되든 MessageBeanEn의 객체가 생성되든 HelloApp은 수정될 사항이 없다.



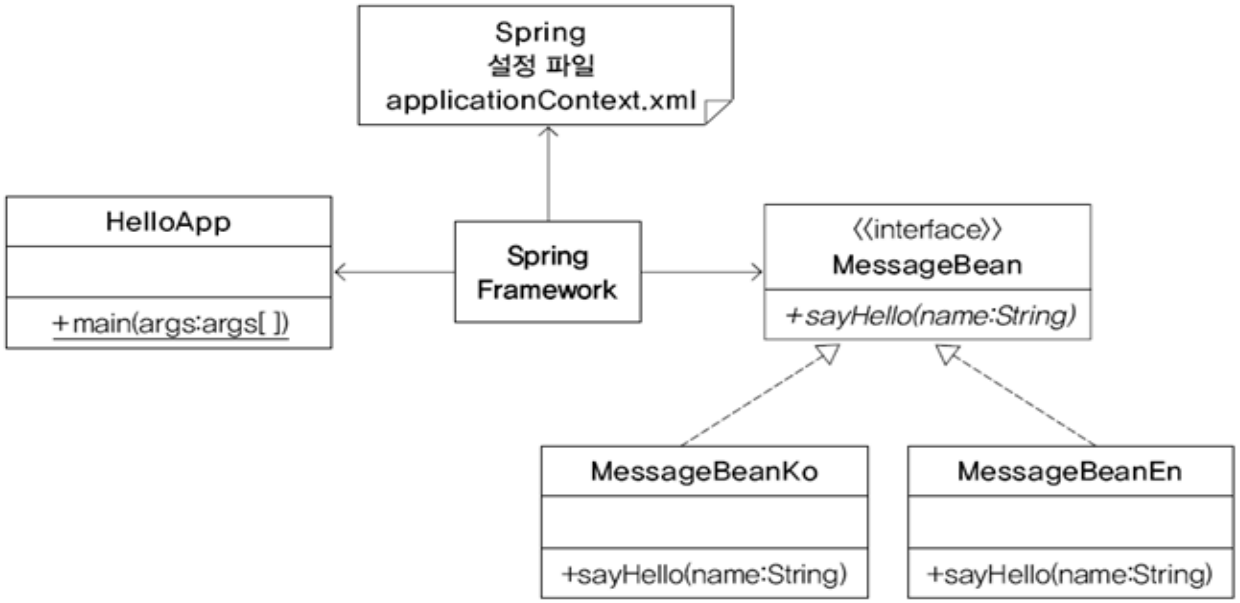
1.Spring 특징

DI/IoC?

약결합 : 스프링을 사용하여 객체 간 결합도를 낮춘 프로그램

- 프로그램에서 필요한 객체를 스프링컨테이너가 미리 생성하여 이 객체를 필요로 하는 프로그램에 생성자, setter 또는 메서드를 통해서 전달(주입)한다.

- 어떠한 객체를 생성하여 전달할지는 디스크립터 파일(XML로 작성)을 사용한다.



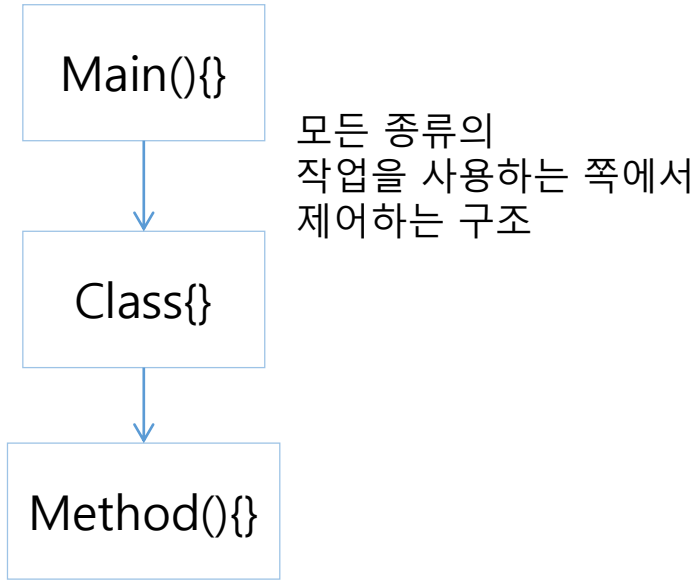
1.Spring 특징

DI/IoC?

IoC (Inversion of Control)

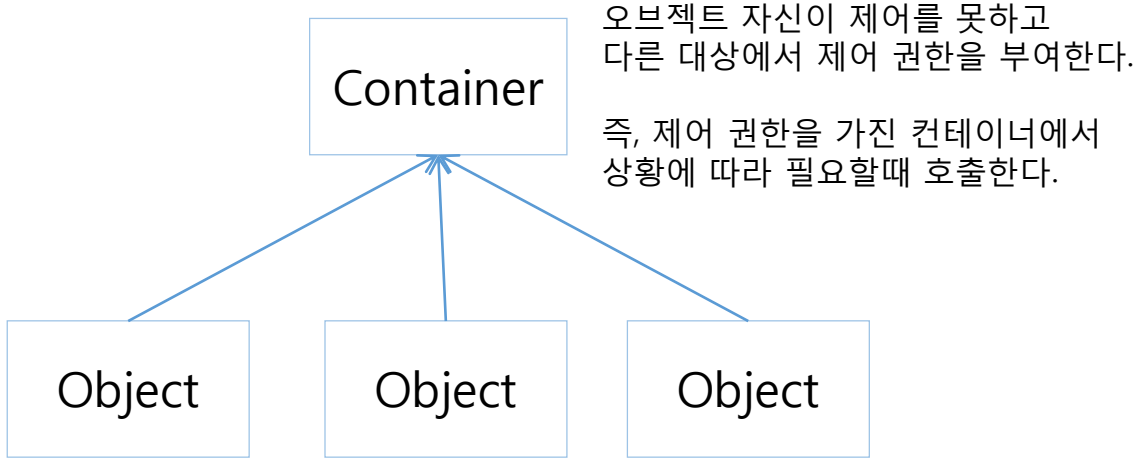
- 프로그램의 제어 흐름 구조가 뒤바뀌는 것

기존의 흐름



모든 종류의
작업을 사용하는 쪽에서
제어하는 구조

제어 역전



오브젝트 자신이 제어를 못하고
다른 대상에서 제어 권한을 부여한다.

즉, 제어 권한을 가진 컨테이너에서
상황에 따라 필요할때 호출한다.

1.Spring 특징

AOP

AOP(Aspect Oriented Programming)

- 관점지향 프로그래밍(AOP)은 객체지향 프로그래밍의 뒤를 이은 또 하나의 프로그래밍 언어구조이다.
- 관점지향의 중요한 개념은 '횡단 관점의 분리(Separation of Cross Cutting Concern)' 이다.

1.Spring 특징

AOP

CC (Core Concern)

- 주 관심사항.

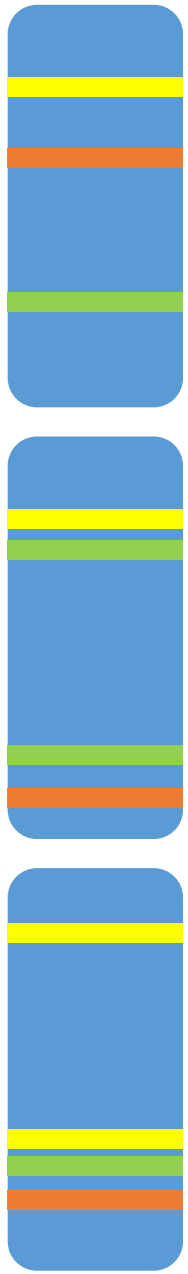
CCC (Cross Cutting Concern)

- 공통 관심사항. Logging, transaction 등.

1.Spring 특징

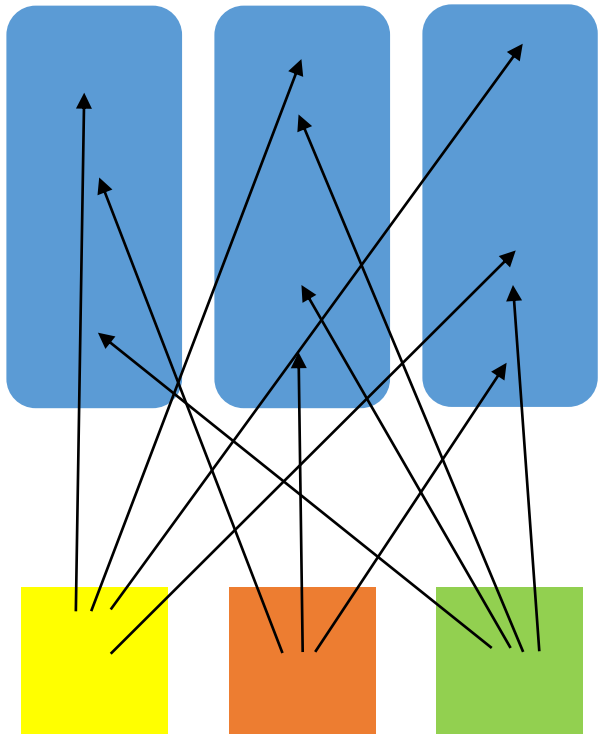
AOP

<Before>



AOP 적용

<After>

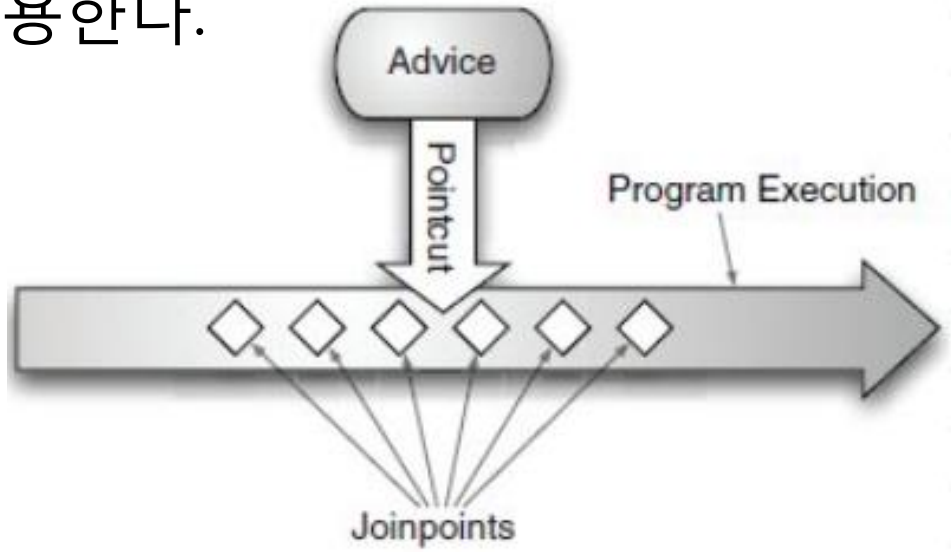


1.Spring 특징

AOP

- 문제를 해결하기 위한 핵심관심사항과 전체에 적용되는 공통관심사항을 기준으로 프로그래밍함으로써 공통모듈을 여러 코드에 쉽게 적용할 수 있도록 지원하는 기술

- 공통으로 사용하는 기능들을 모듈화하고 해당 기능을 프로그램 코드에서 직접 명시하지 않고 선언적으로 처리하여 필요한 컴포넌트에 계층적으로 다양한 기능들을 적용한다.



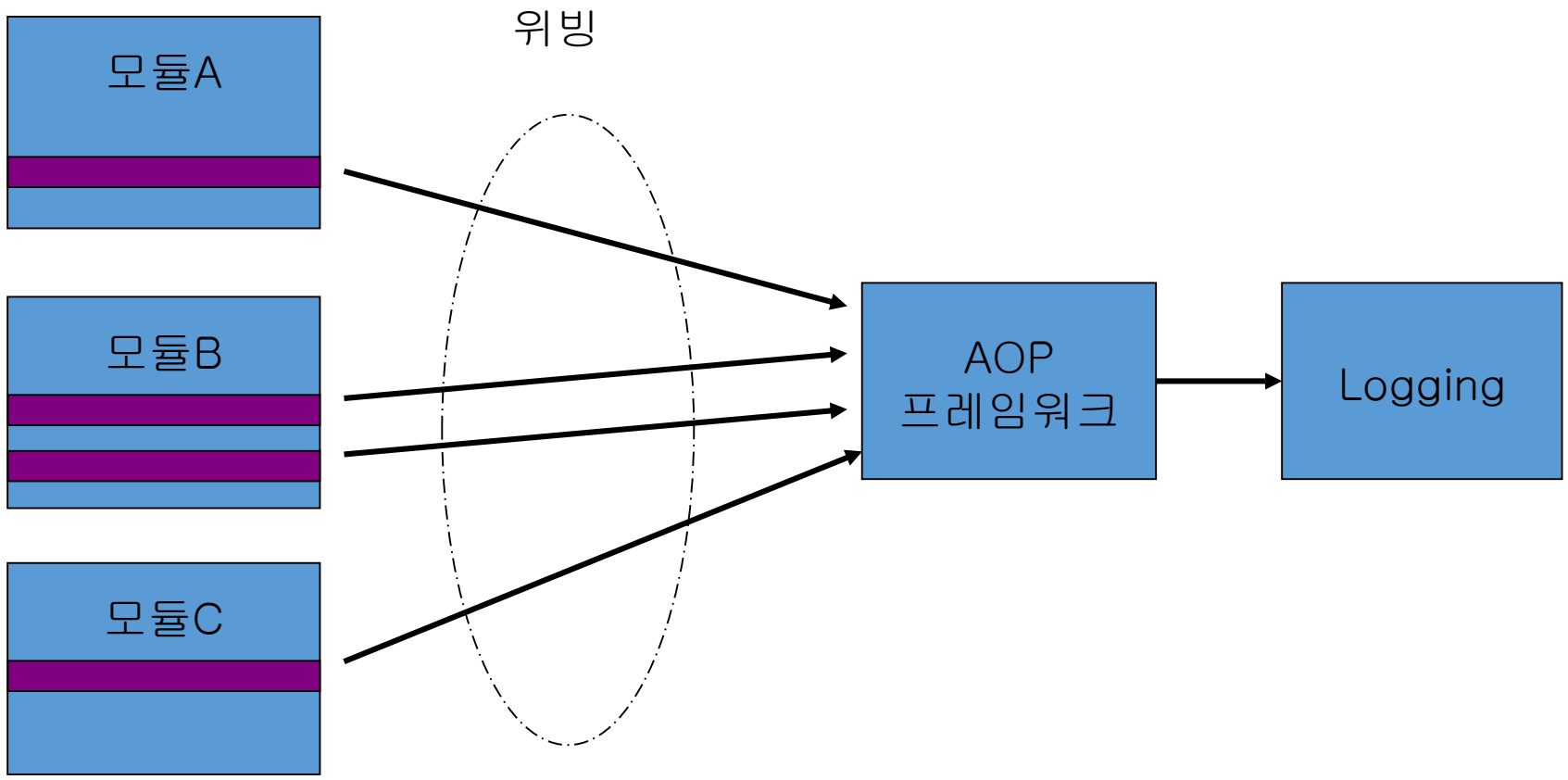
1.Spring 특징

AOP

용 어	설 명
결합점 **(Join Point)**	인스턴스의 생성시점. 메소드를 호출하는 시점. Exception이 발생하는 시점과 같이 애플리케이션이 실행될 때 특정 작업이 실해오디는 시점을 의미한다. (Aspect를 플러그인 할 수 있는 애플리케이션의 실행 시점)
교차점 **(Pointcut)**	충고가 어떤 결합점에 적용되어야 하는지 정의. 명시적인 클래스의 이름, 메소드의 이름이나 클래스나 메소드의 이름과 패턴이 일치하는 결합점을 지정 가능토록 해준다.(스프링 설정파일 안에 XML로 작성)
충고 **(Advice)**	교차점에서 지정한 결합점에서 실행(삽입)되어야 하는 코드. Aspect의 실제 구현체
에스펙트 **(Aspect)**	에스펙트는 AOP의 중심 단위. Advice와 pointcut을 합친 것이다. 구현하고자 하는 횡단 관심사의 기능, 애플리케이션의 모듈화 하고자 하는 부분
엮기 **(Weaving)**	에스펙트를 대상 객체에 적용하여 새로운 프록시 객체를 생성하는 과정을 말한다. Aspect는 대상 객체의 지정된 결합점에 엮인다.

1.Spring 특징

AOP



AOP의 횡단 관점의 분리와 위빙

1.Spring 특징

AOP

<설정 구조>

```
<aop:config>
    <aop:pointcut/>              : pointcut 설정
    <aop:aspect>                 : aspect 설정
        <aop:befor/>             : method 실행 전
        <aop:after-returning/>   : method 정상 실행 후
        <aop:after-throwing/>    : method 예외 발생 시
        <aop:after/>             : method 실행 후 (예외 발생 여부 상관 없음)
        <aop:around/>            : 모든 시점 적용 가능
    </aop:aspect>
</aop:config>
```

1.Spring 특징

OCP

OCP(Open-Closed Principle)

- 인터페이스를 통해 제공되는 확장 포인트는 확장을 위해 개방되어 있고, 인터페이스를 이용하는 클래스는 자신의 변화가 불필요하게 일어나지 않도록 굳게 폐쇄되어 있다.

- 즉, A가 B를 의존 한다면, B를 가져다 쓰는 A는 변화가 없게 폐쇄 시키고, 공개 되어 있는 B는 언제든 다른 기능으로 변화를 줄 수 있는 확장성이 좋게 개방시켜 두는 게 좋다.

2.SpringMVC

소개

- 스프링 MVC 프레임워크는 스프링 기반으로 사용할 수 있다.
- 스프링이 제공하는 트랜잭션처리가 DI 및 AOP 적용 등을 쉽게 사용할 수 있도록 돕는다.
- 스트럭츠와 같은 프레임워크와 스프링 프레임워크를 연동하기 위해 추가적인 설정을 하지 않아도 된다.

- 스프링 프레임워크에서 지원하는 Spring MVC는 모델-뷰-컨트롤러(MVC) 구현을 포함하여 도메인 모델코드와 웹 폼을 깔끔하게 분리할 수 있도록 하고 스프링 프레임워크의 다른 모든 기능과 통합할 수 있게 하며 DI와 선언적인 방식으로 MVC 기반의 웹 프로그램 개발을 효율적으로 할 수 있도록 지원한다.

2.SpringMVC

특징

- Spring Framework의 다른 모듈과의 연계 용이
- 컨트롤러, command 객체, 모델 객체, Validator 등 각각의
  역할에 대한 명확한 분리
- Form 객체 없이 사용자 지정 가능한 데이터 바인딩과 유효성 체크 지원
- 어떠한 View 기술과도 연계가 용이
- Tag lib통한 Message 출력, Theme 적용 등과 입력 폼을 보다 쉽게 구현

2.SpringMVC

장점

1) 스프링 MVC 프레임워크는 스프링 기반으로 사용할 수 있다.

2) 스프링이 제공하는 트랜젝션 처리가 DI 및 AOP 적용 등을 손쉽게 사용할 수 있다.

3) 스트럿츠와 같은 프레임워크와 스프링프레임워크를 연동하기 위해 추가적인 설정을 하지 않아도 된다.

2.SpringMVC

주요 구성요소

-DispatcherServlet :

클라이언트의 요청을 전달 받는다. 컨트롤러에게 클라이언트의 요청을 전달하고 컨트롤러가 리턴한 결과값을 View에 전달하여알맞은 응답을 생성한다.

-HandlerMapping :

클라이언트의 요청 URL을 어떤 컨트롤러가 처리할지를 결정한다.

RequestURL과 Controller 클래스의 맵핑을 관리한다.

-Controller :

클라이언트의 요청을 처리한 뒤, 그 결과를 DispatcherServlet에 알려준다.

비즈니스 로직을 호출하여 처리 결과 ModelAndView 인스턴스를 반환한다.

-ModelAndView :

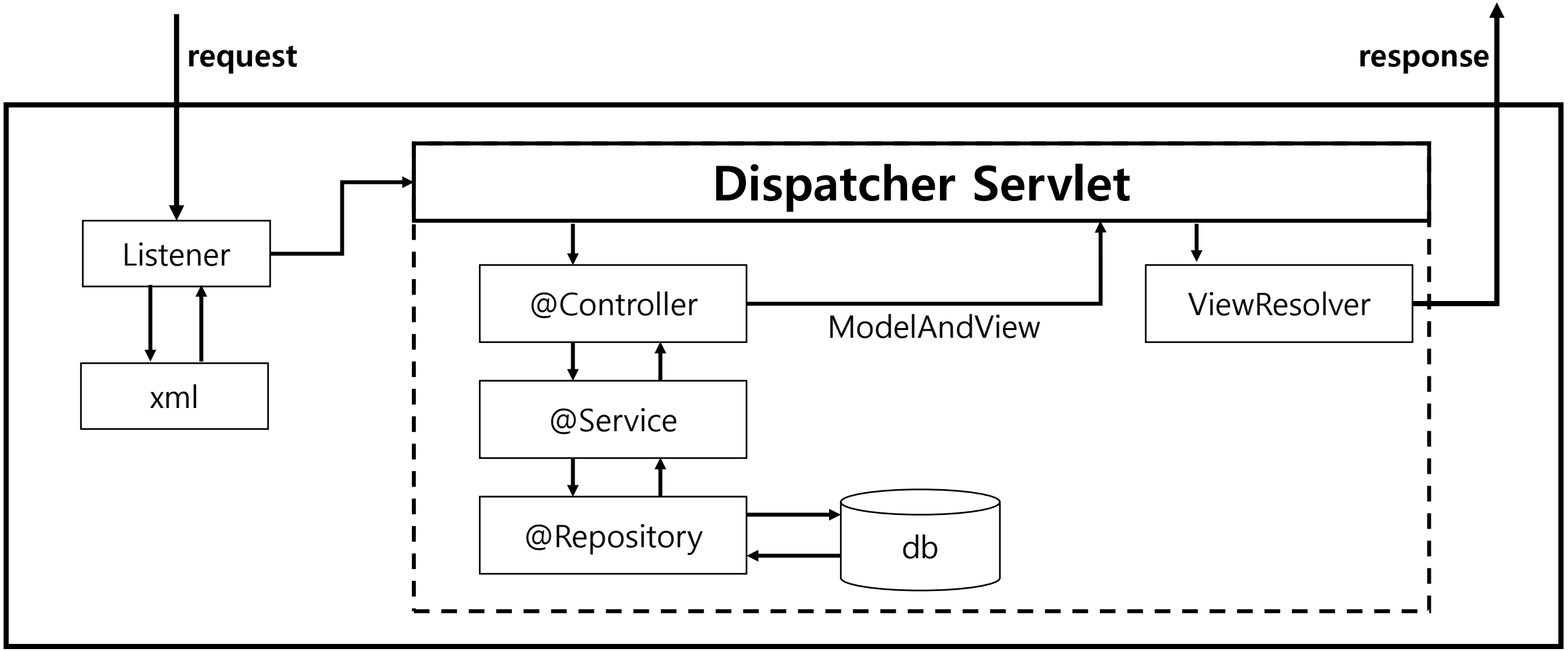
컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담는다.

-ViewResolver :

컨트롤러의 처리결과를 생성할 뷰를 결정한다.

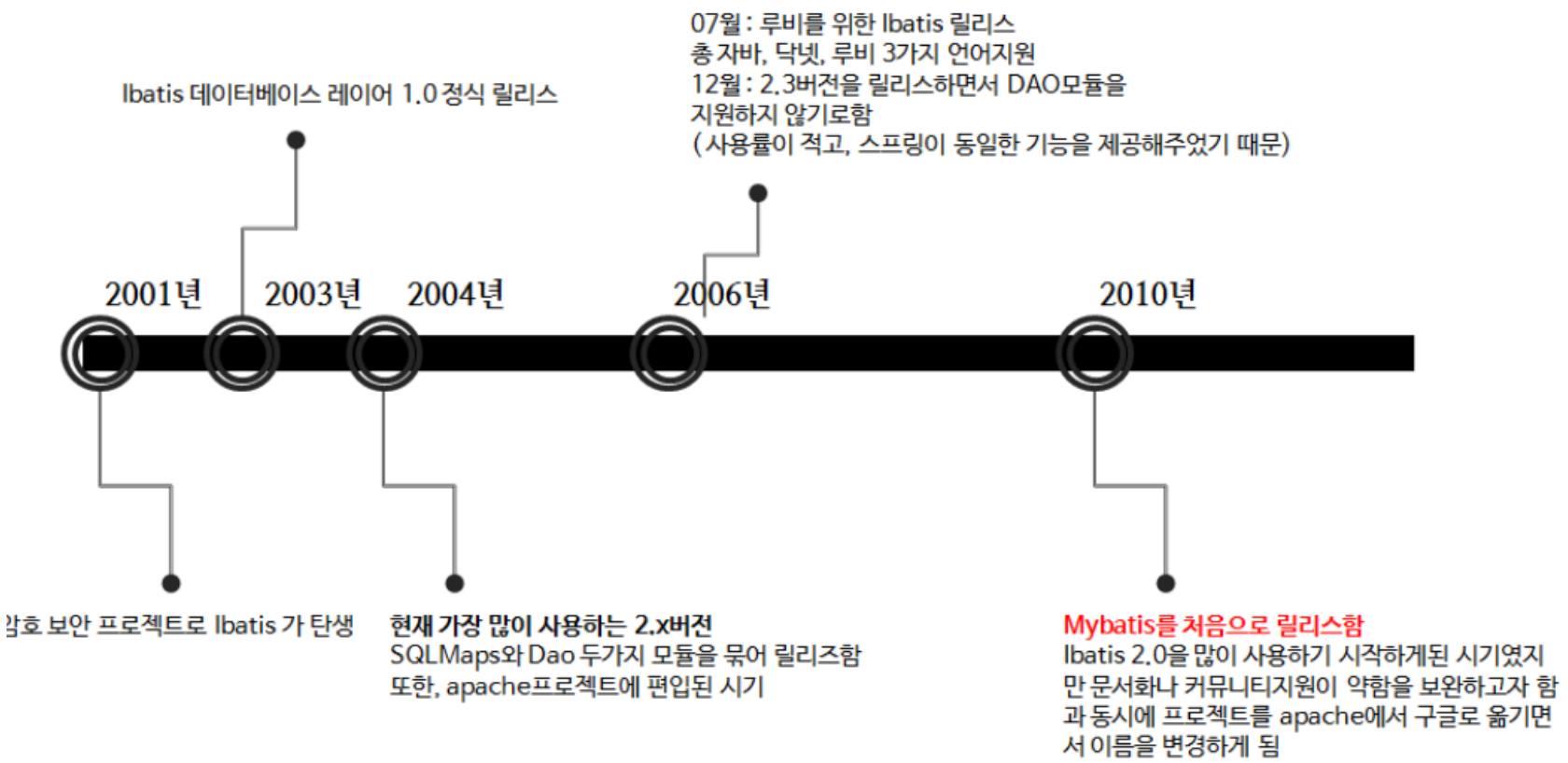
2.SpringMVC

흐름도



3.iBatis/MyBatis

개요



3.iBatis/MyBatis

iBatis와 MyBatis 차이점

구분	iBatis	MyBatis
네임스페이스	선택사항	필수사항
매핑구문정의	XML만 사용	XML과 어노테이션 사용
동적SQL	XML 엘리먼트만 사용 동적 SQL을 위한 XML엘리먼트는 16개 내외	XML 엘리먼트 및 구문 빌더 사용 동적SQL을 위한 XML엘리먼트는 4개 내외 (if, choose, trim, foreach, set)
스프링 연동	스프링 자체 구현체 사용	마이바티스 별도 모듈 사용
지원계획	공식적인 릴리스 없음	향후 계속 릴리스 될 예정

3.iBatis/MyBatis

XML

- XML

사용방법은 기존과 동일하나 용어들이 변경되었다.

이전 용어	변경된 용어
SqlMapConfig	Configuration
sqlMap	mapper

- 그 외에도 XML 엘리먼트가 축소되었다.

if	조건문
choose(when otherwise)	반복문
trim	공백제거
foreach	반복문
set	Update 마지막으로 명시된 칼럼표기에서 쉼표제거

3.iBatis/MyBatis

인터페이스

- 인터페이스를 이용하여 작성할 경우 어노테이션을 이용하여 작성이 가능하다.

- XML에서 어노테이션으로 변경하는 방법은 다음과 같다.

매핑 구분의 네임스페이스	인터페이스의 패키지 명과 인터페이스 명
매핑 구분 아이디	어노테이션을 가진 메소드명
매핑 구분의 결과 데이터타입 (resultType 속성)	어노테이션을 가진 메소드의 반환 타입
매핑 구분의 파라미터 타입 (parameterType 속성)	어노테이션을 가진 메소드의 파라미터 타입

3.iBatis/MyBatis

Mapper 정의 방법

정의방법	장 점	단 점
XML만 사용	Mapper(과거 sqlMap)의 모든 기능을 이용할 수 있음	매핑구분의 아이디를 문자열 형태로 선언해야 하기 때문에 버그 발생 빈도 높음 범용적인 API특성상 타입 변환이 필요하고 타입변화 오류가 날 가능성이 있음
인터페이스만 사용	매핑구문을 사용할 때 인터페이스의 메소드를 그대로 사용하기 때문에 매핑구문을 잘 못적는 경우가 없음 타입 변환이 필요없음	어노테이션 특성상 동적SQL을 작성하기 위해 별도 클래스를 작성해야 함 결과 매핑에 제약이 있음
XML과 어노테이션 함께 사용	매핑구문을 사용할 때 인터페이스의 메소드를 그대로 사용하기 때문에 매핑구문을 잘 못 적는 경우가 없음 Mapper(과거 sqlMap)의 모든 기능을 사용할 수 있음	인터페이스와 매퍼 XML을 모두 작성해야하기때문에 코드 작성에 시간이 더 듦

제목

소제목

내용