Spring framework

Y-A Dominica Kim

1. 프레임워크란?

[프레임워크]
프로그램을 만드는 과정에서 기본이 되는 **골격 코드**

예) 건물지을 때 철골로 뼈대를 세우는 것, 큰 배를 만들 때

골격을 보면 완성품이 무엇이 될지 예상 가능하다. → 반제품이라 할 수 있다.

[프로그래밍 기술에서의 프레임워크]

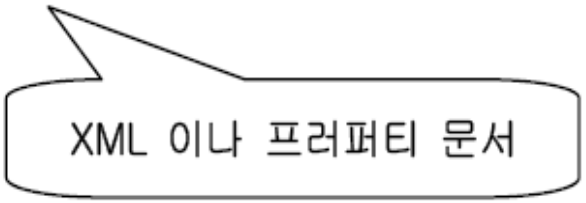
자체로는 완전한 어플리케이션 소프트웨어가 아니며, 어떠한 문제 영역을 해결하기 위한 잘 설계된 일반적인 그리고 재사용 가능한 모듈이라고 할 수 있다. 개발자는 프레임워크를 확장하여 비즈니스 요구사항을 만족하는 완전한 어플리케이션 소프트웨어를 만들게 된다.

1. 프레임워크란?

- 프레임워크는 Open-Closed 원칙을 따르고 있다.
  - 재사용되는 공통되는 부분은 프레임워크로 구현되어 그 내부를 가공하지 않고 이용하도록 한다.(Closed)
  - 확장이 필요한 부분은 사용자의 요구 사항에 맞게 정의하여 확장하므로 문제영역에 최적화된 애플리케이션 설계가 완성된다.(Open)
- 훅 포인트 : 확장 모듈을 연결하는 확장점

  추상 클래스나 인터페이스로 설계된다.

  메타데이터를 이용하여 훅 포인트로의 연결, 정책 설정등을 정의한다.



선언적인 방법

[자바의 주요 프레임워크]

웹 어플리케이션 프레임워크 – Spring MVC, Struts, WebWork
데이터베이스 관련 프레임워크 – iBATIS, Hibernate, Spring DAO
비즈니스 로직 관련 프레임워크 – Spring, EJB

1. Spring의 개요

[Spring 소개]

EJB Enterprise JavaBeans

엔터프라이즈 어플리케이션 개발을 단순화 한다.
선언적 프로그래밍 모델을 통해서 트랜잭션이나 보안과 같은 개발의 기반 구조에 해당하는 여러 측면을 단순화시키기는 하지만 배치 기술자, 홈/리모트 인터페이스 구현등과 같은 과도한 코드를 강제함으로써 다른 방식으로 복잡해졌다. EJB 는 분산 객체나 원격 트랜잭션 등의 복잡한 문제를 해결하기 위해 만들어졌기 때문에 복잡하다.

→ 개발자들은 좀 더 쉬운 개발 방법을 찾고 있다.

스프링은 로드 존슨이 만든 오픈 소스(2003년 2월부터) 프레임워크로서 저서인 Expert One-on-One: J2EE Design and Development 에서 처음 소개되었다.

1. Spring의 개요

스프링 프레임워크는 엔터프라이즈급 애플리케이션을 만들기 위한 경량솔루션이며 많은 기능을 제공하고 있으며 필요한 부분만 가져다 사용할 수 있도록 모듈화되어 있다. 스프링을 사용하면 "Pain Old Java objects"(POJOs)로 어플리케이션을 만들어 엔터프라이즈 서비스를 구축할 수 있다

1. Spring의 개요

평범한 JavaBeans(POJO)를 사용하여 EJB 에서만 가능했던 복잡한 엔터프라이즈 애플리케이션 개발을 가능하게 한다. 또한 자바 애플리케이션에서 단순성, 테스트 용이성, 느슨한 결합성 측면에서 이점을 제공한다.

스프링(Spring)은 간단히 말하면 엔터프라이즈 어플리케이션에서 필요로 하는 기능을 제공하는 프레임워크이다. 스프링은 J2EE가 제공하는 다수의 기능을 지원하고 있기 때문에, J2EE를 대체하는 프레임워크로 자리 잡고 있다.

스프링이 추구하는 바는 크게 두 가지이다.

[1] 복잡하고 무거운 J2EE 기술의 사용을 쉽고 가볍게 만들어주고, 자연스럽게 검증된 최상의 실천 사례들을 구현하도록 함으로써 좋은 프로그램이 작성될 수 있도록 유도한다.

[2] 기존의 잘 알려진 기술들을 프레임워크 내에서 일관된 방법으로 쉽게 사용할 수 있도록 돕는다.

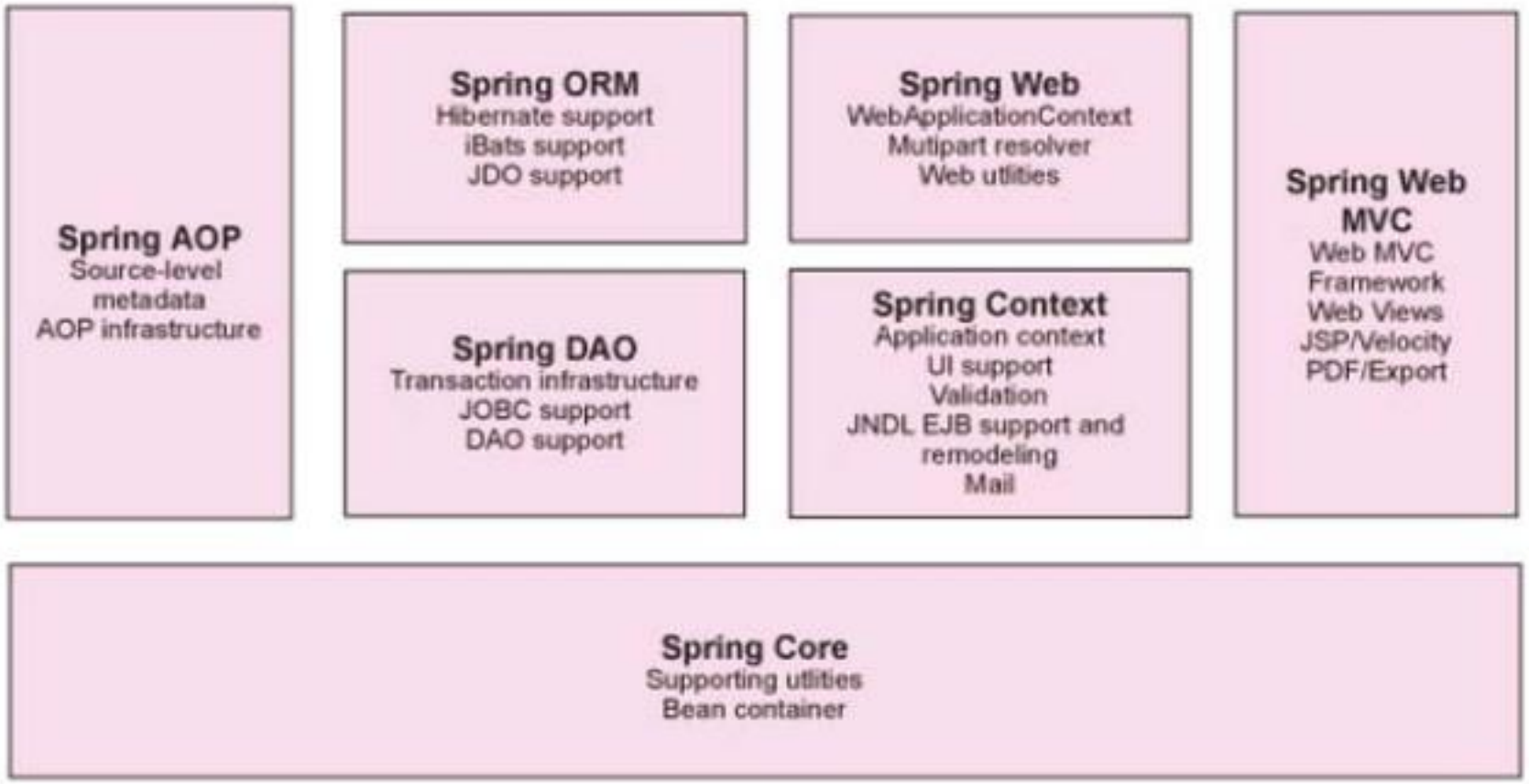
(1)스프링의 특징

1. 어플리케이션 프레임워크로 불리며,웹 어플리케이션은 물론,
   콘솔어플리케이션이나 스윙과 같은 **GUI** 어플리케이션등
   어떤 어플리케이션에도 적용 가능한 프레임워크이다

2. 스프링은 **EJB**와 같이 복잡한 순서를 거치지 않아도 간단하게
   이용할수 있기 때문에 '경량(**Lightweight**)컨테이너'라고도 부른다

3. **Dependency Injection(DI)**과 **Aspect Oreited Programming(AOP)**
   을 가장 중점적인 기술로 사용되지만, 이외에도 여러가지 기능을
   제공하고 있다.

(2)스프링의 구성요소(7개의 모듈)



Enterprise Application 개발을 겨냥해 만든 경량형 IoC 와 AOP 컨테이너 프레임워크이다.

1. Spring Core
   - 스프링의 근간이 되는 Ioc(또는 DI)기능을 지원하고 영역을 담당한다
   - BeanFactory를 기반으로 Bean클래스들을 제어할수 있는 기능을 지원한다
2. Spring AOP
   - Aspect Oreiented Programming(관점지향프로그램)을 지원한다
3. Spring ORM
   - ORM(Object/Relation Mapping)기능을 제공한다
   - ORM프레임워크와 Hibernate,iBatis와 JDO를 지원한다
4. Spring DAO
   - DAO(Data Access Object)기능을 제공한다
   - JDBC에 의한 데이터베이스 액세스를 지원하고 트렌젝션 관리의 기반이 된다
5. Spring Web
   - 웹 어플리케이션 개발에 필요한 Web Appliction Context와 MultiPart Request등의 기능을 지원한다
6. Spring Context
   - Spring Core 바로위에 있으면서 Spring Core에서 지원하는 기능 외에 추가적인 기능들과 좀 더 쉬운 개발이 가능하도록 지원하고 있다.
   - 유저 인터페이스 및 타당성 검증이라는 어플리케이션의 기반 성능,JNDI 및 EJB의 지원, 메일 송.수신 기능 등을 지원한다.
7. Spring Web MVC
   - 웹어플리케이션의 MVC프레임워크기능을 제공한다

2. DI(Dependency Injection)

: 객체간의 결합을 느슨하게 하는 스프링의 핵심 기술이다

※의존관계를 관리하기위한 방법

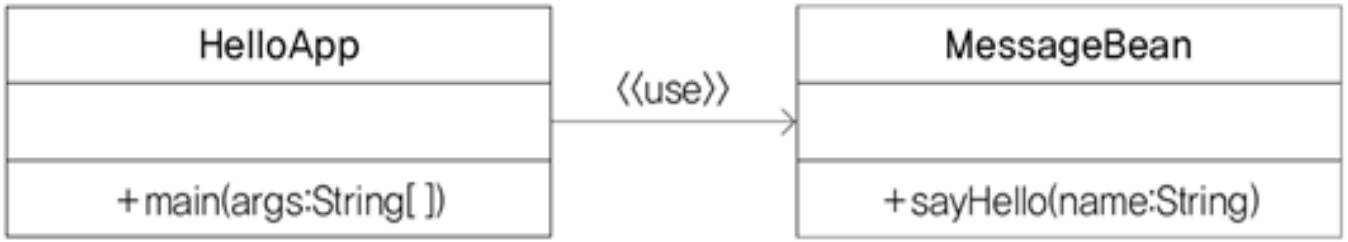
1. Construction Injection :
   생성자를 통해서 의존관계를 연결시키는것을 말함

2. Seter Injection :
   클래스 사이의 의존관계를 연결시키기 위해서 setter메소드를
   이용하는 방법을 말함

2. DI(Dependency Injection)

객체간 결합도가 강한 프로그램

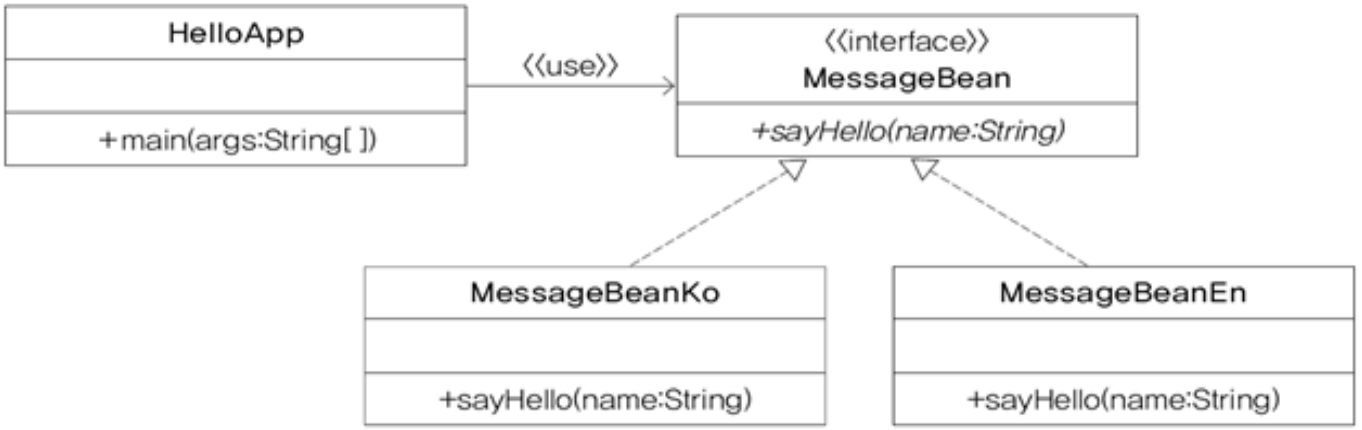
- HelloApp 에서 MessageBean 을 직접 객체 생성하여 사용하고 있다.
- MessageBean 클래스를 다른 클래스로 변경할 경우 HelloApp 의 소스를 같이 수정해 주어야 한다.



2. DI(Dependency Injection)

인터페이스를 사용하여 객체간 결합도를 낮춘 프로그램

- HelloApp 는 MessageBean 이라는 인터페이스를 통해서 객체를 사용한다
- 일반적으로 팩토리 메서드를 활용하여 사용할 객체(MessageBeanKo 또는 MessageBeanEn)를 생성한다. MessageBean 이라는 MessageBeanKo 의 객체가 생성되든 MessageBeanEn의 객체가 생성되든 HelloApp 는 수정될 사항이 없다.

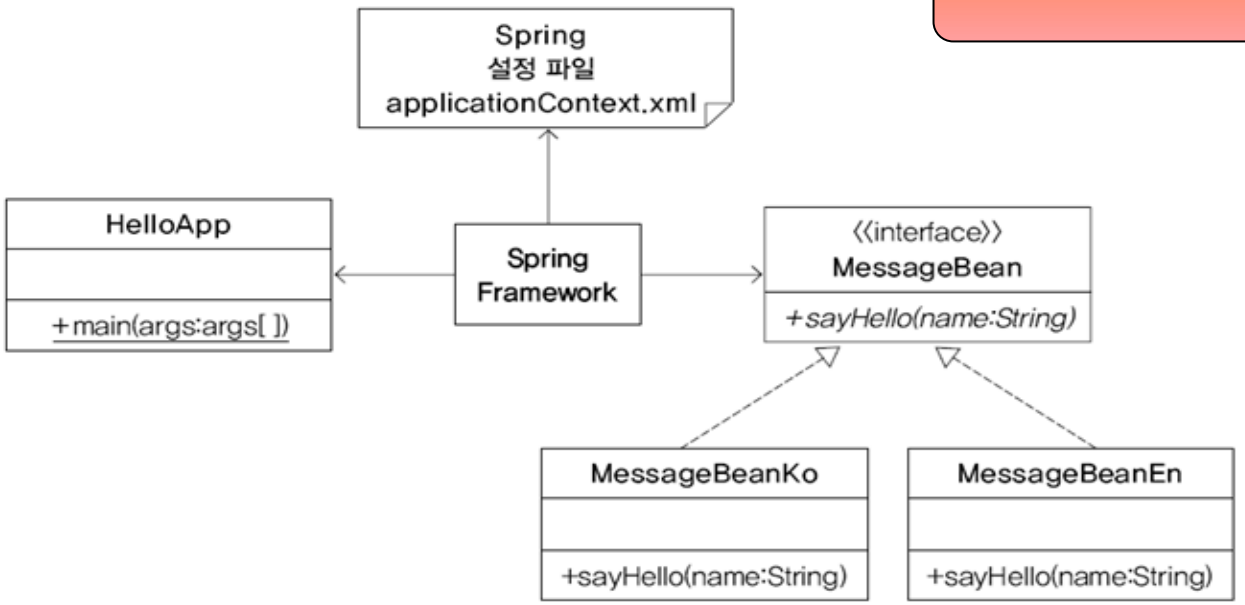


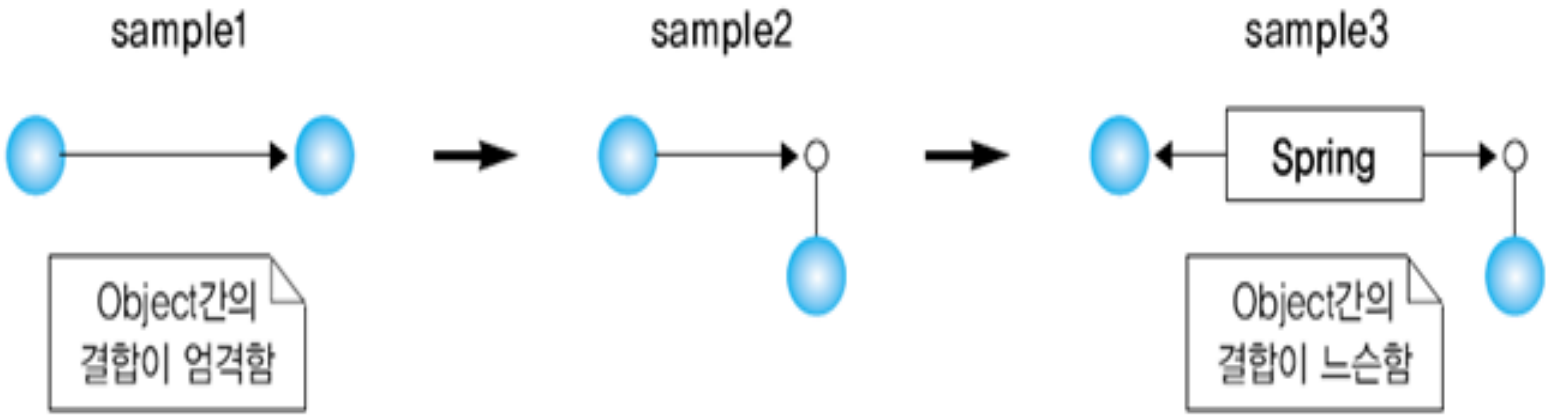
2. DI(Dependency Injection)

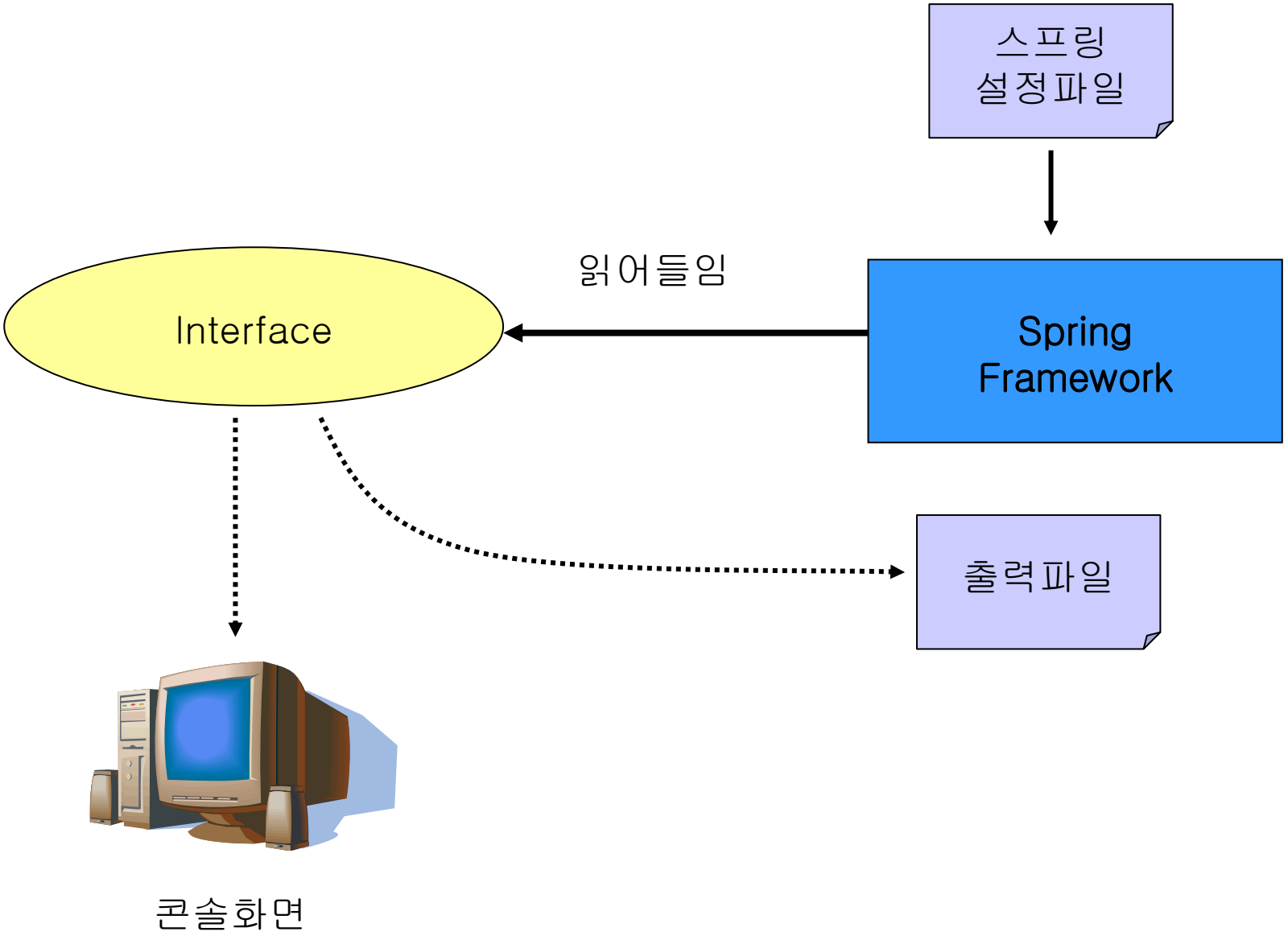
스프링을 사용한 객체간 결합도를 낮춘 프로그램

- 프로그램에서 필요한 객체를 스프링컨테이너가 미리 생성하여 이 객체를 필요로 하는 프로그램에 생성자 또는 Setter 메서드를 통해서 전달(주입)한다.
- 어떠한 객체를 생성하여 전달할지는 디스크립터 파일(XML로 작성)을 사용한다

선언적인 프로그래밍




sample1
sample2
sample3
Spring
Object간의
결합이 엄격함
Object간의
결합이 느슨함


스프링
설정파일
읽어들임
Interface
Spring
Framework
출력파일
콘솔화면

3. AOP(Aspect Oriented Programming)

관점지향프로그램(AOP)은 객체지향프로그램의 뒤를 이은 또하나의

프로그래밍언어구조이다

관점지향의 중요한 개념은 '횡단 관점의분리(Separation of Cross-Cutting Concern)'이다

AOP

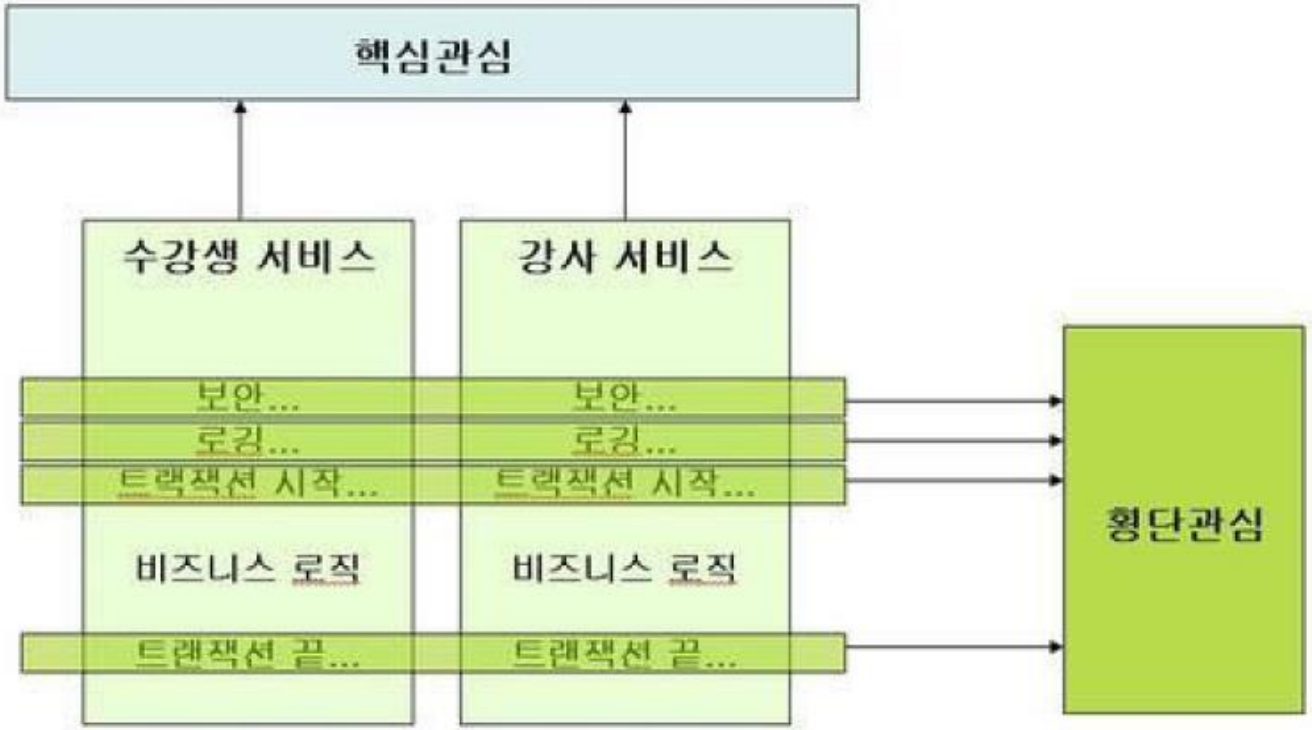
문제를 바라보는 관점을 기준으로 프로그래밍 하는 기법

문제를 해결하기 위한 핵심관심사항과 전체에 적용되는 공통관심사항을 기준으로 프로그래밍함으로써 공통모듈을 여러 코드에 쉽게 적용할 수 있도록 지원하는 기술이 AOP 이다.

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

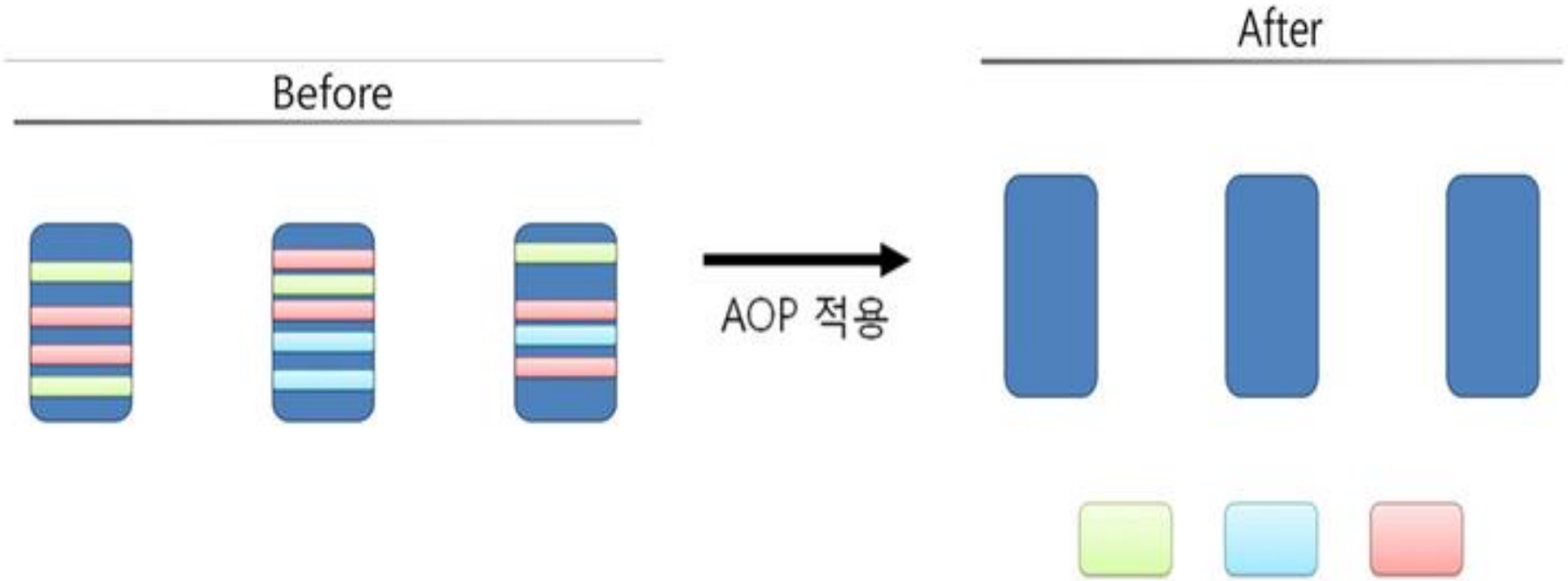
AOP란

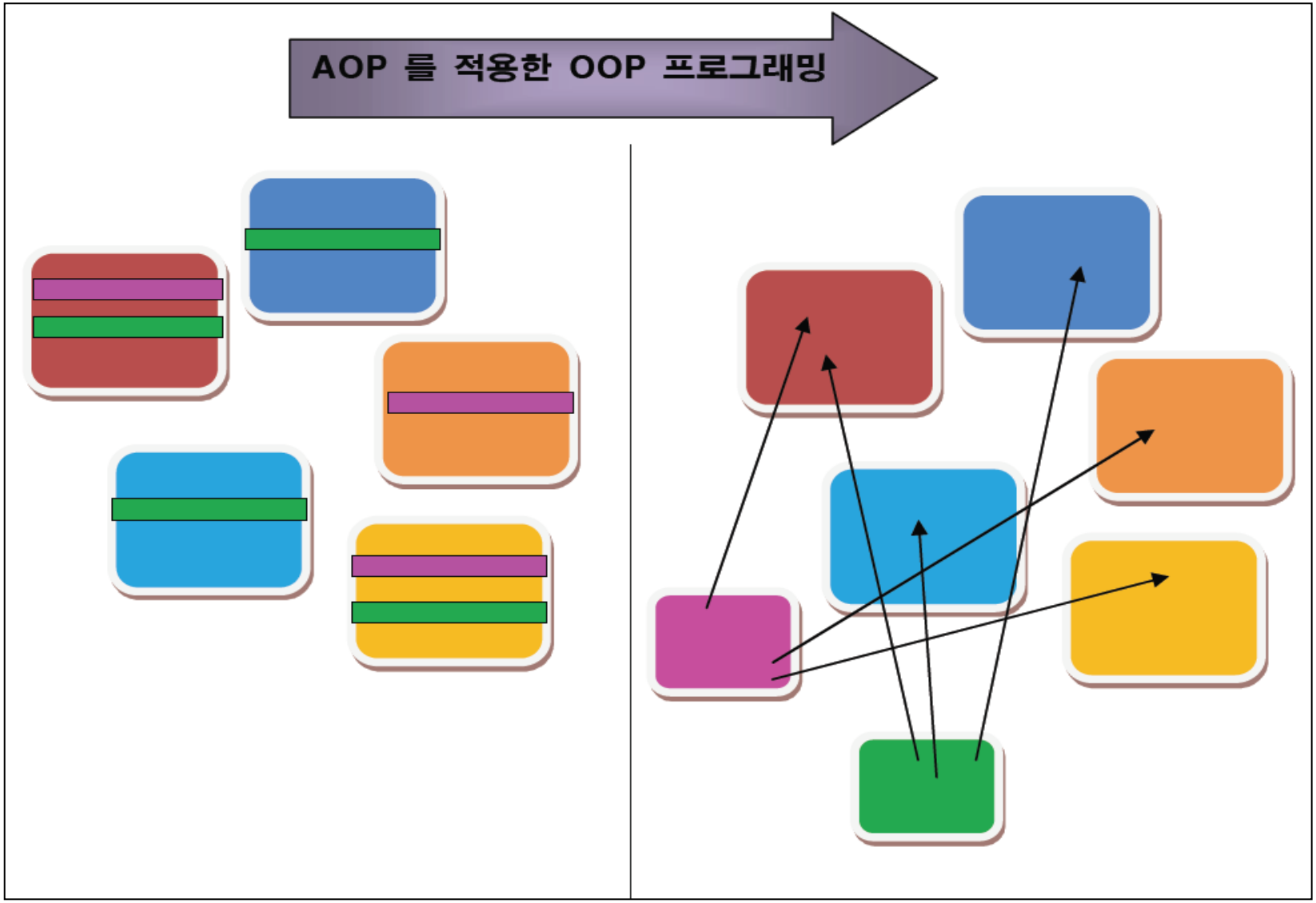
- 관심의 산재

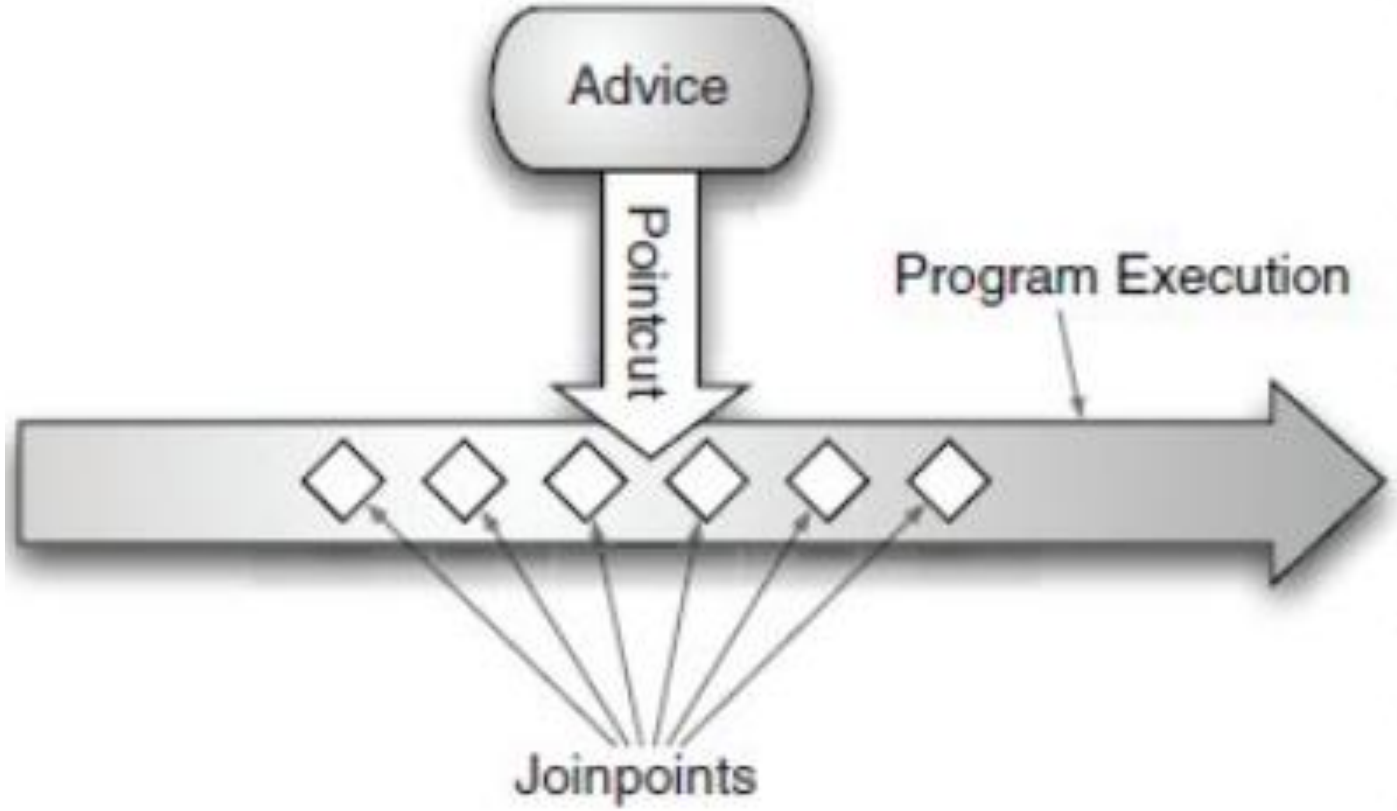

핵심관심
수강생 서비스
강사 서비스
보안...
로깅...
트랙잭션 시작...
보안...
로깅...
트랙잭션 시작...
비즈니스 로직
비즈니스 로직
트랜잭션 끝...
트랜잭션 끝...
횡단관심

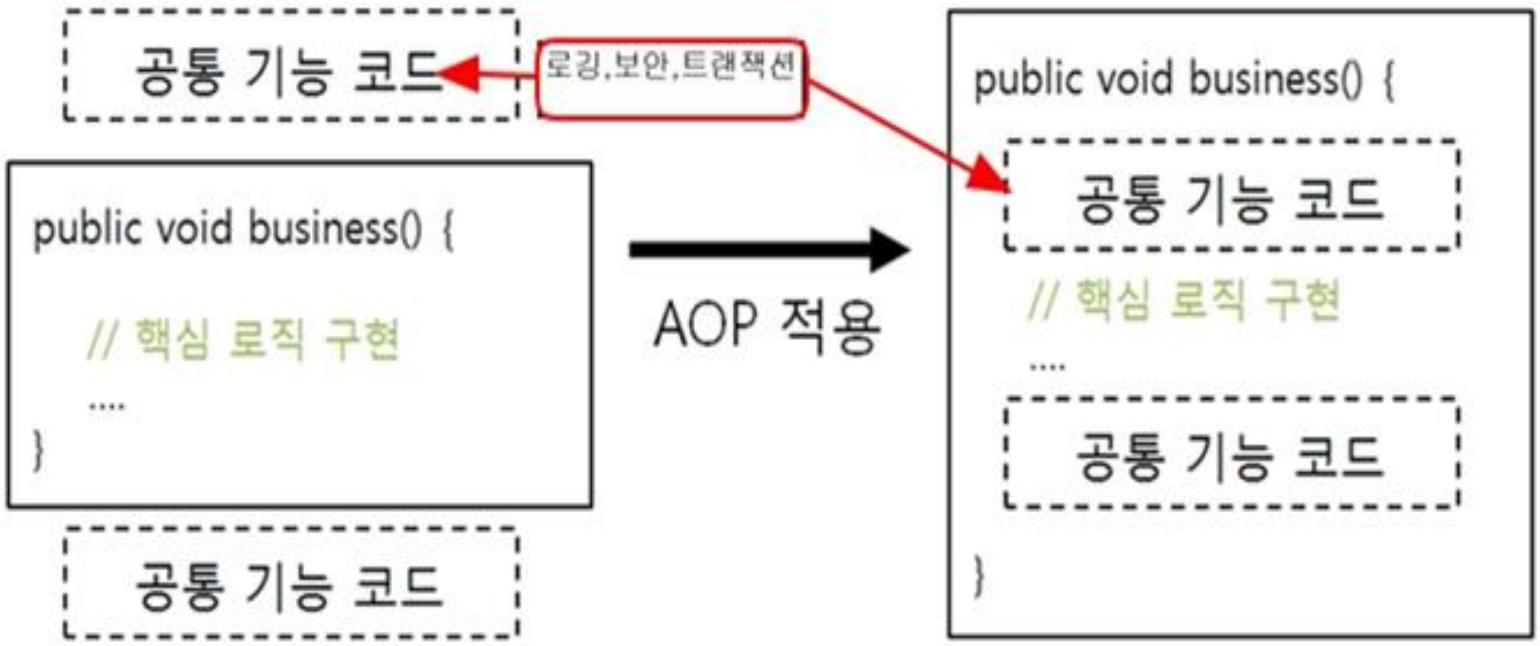
Before : AOP기법 적용전

Affer : AOP기법 적용후




AOP 를 적용한 OOP 프로그래밍


Advice
Pointcut
Program Execution
Joinpoints

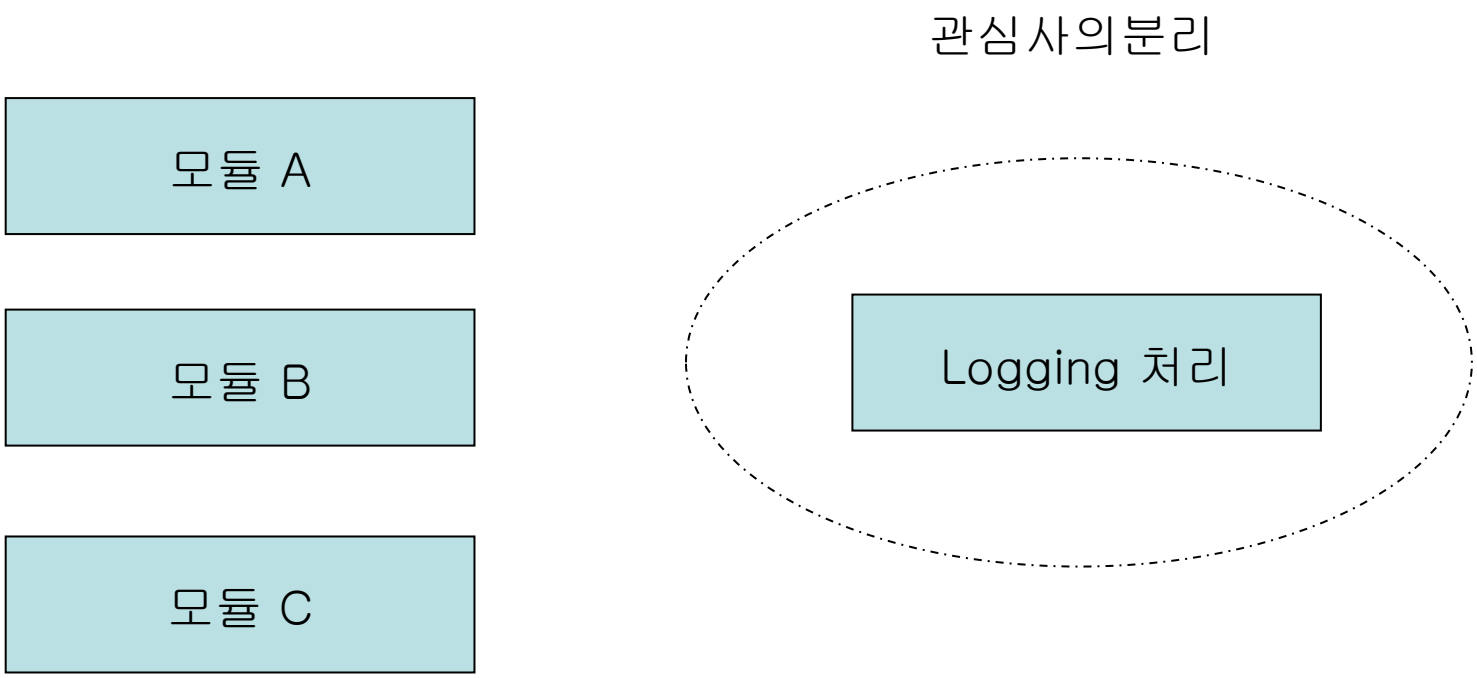


이렇게 공통 관심 사항을 구현한 코드를 AOP기법으로 적용을하면 위그림처럼 핵심로직안에 삽입이됩니다

*로깅: (일지에 기록하는 것) 어플리케이션에서 로깅이란 어플리케이션이 실행될 때 어느 부분의 어떤 메소드가 실행되고 있는지, 어떤 데이터를 가지고 오는지에 대해 일지 대신에 시스템의 내부에 기록을 하는 것
*트랜잭션 : 하나의 트랜잭션은 하나의 커다란 작업을 처리하는 것처럼 보이는 일련의 작업

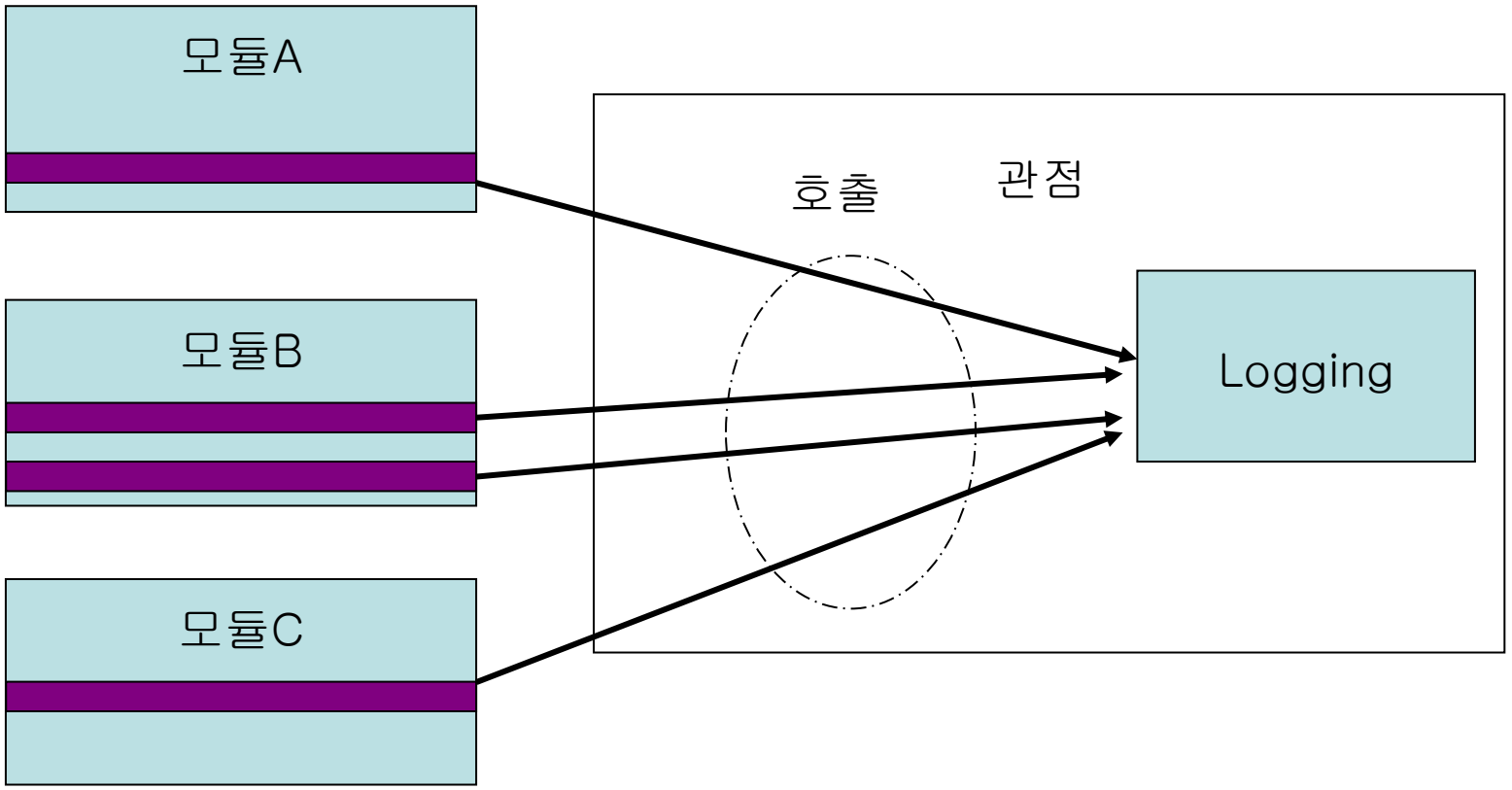
AOP 용어

용어	설명
1.결합점 (Join Point)	인스턴스의 생성시점,메소드를 호출하는 시점, Exception이 발생하는 시점과 같이 애플리케이션이 실행될 때 특정작업이 실행되는 시점을 의미한다.(Aspect를 플러그인 할 수 있는 애플리케이션의 실행 지점)
2.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)
3.충고 (Advice)	충고는 교차점에서 지정한 결합점에서 실행(삽입)되어야하는 코드이다. Aspect의 실제 구현체
4.에스팩트 (Aspect)	에스팩트는 AOP의 중심단위, Advice와 pointcut을 합친 것이다. 구현 하고자 하는 횡단 관심사의 기능, 애플리케이션의 모듈화 하고자 하는 부분
5.엮기 (Weaving)	에스팩트를 대상 객체에 적용하여 새로운 프록시 객체를 생성하는 과정을 말한다. Aspect는 대상 객체의 지정된 결합점에 엮인다.

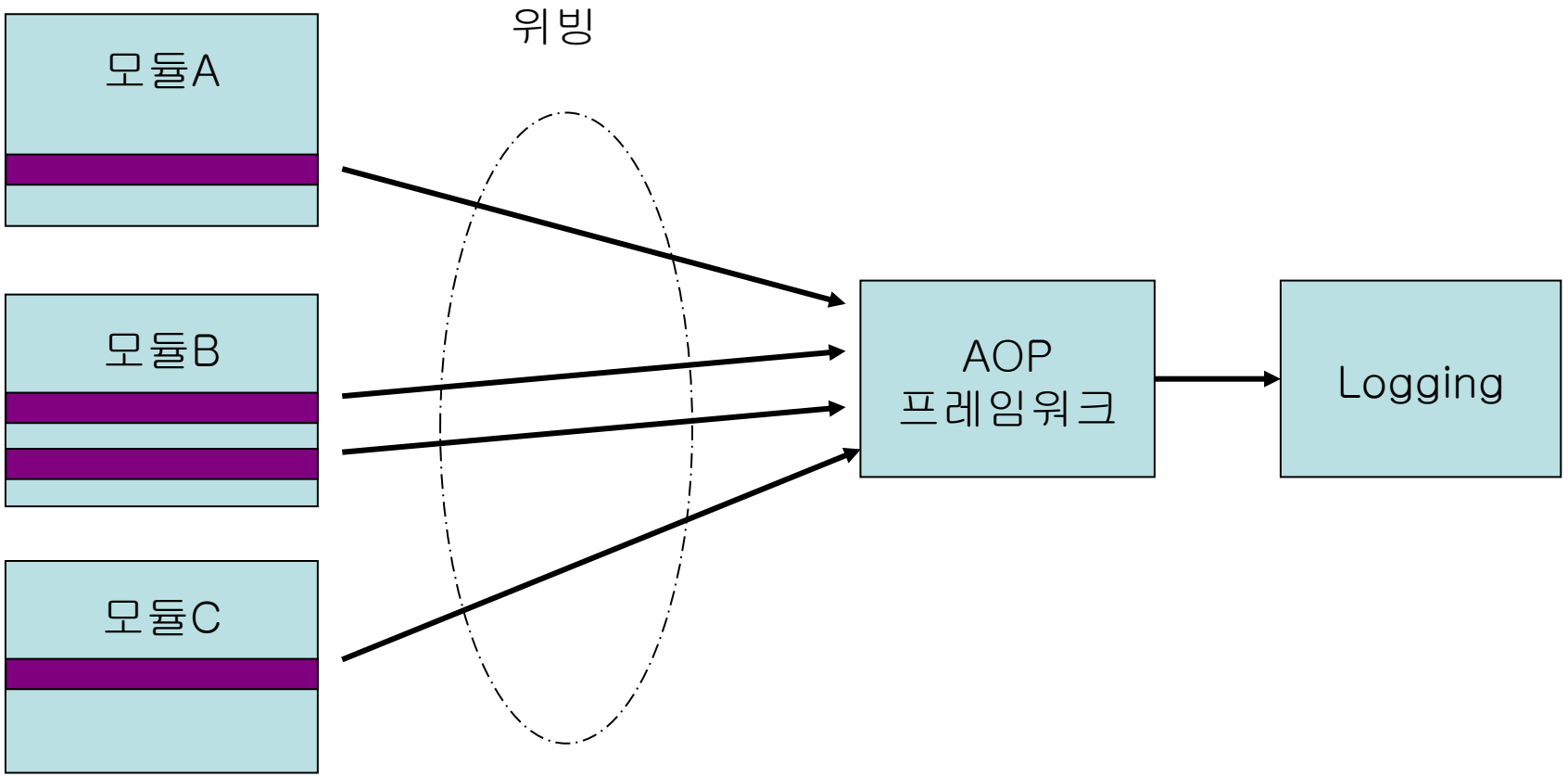


기존의 객체지향에서 관점의분리

횡단적 산재



횡단적으로 산재하는 '기능의 호출'



AOP의 횡단 관점의 분리와 위빙

AOP에서는 분리된 관접이 AOP프레임워크에 의해관리되고, 필요한 순간에 각 모듈로 삽입된다. 분리한 관점을 여러 차례 모듈에 삽입하는 것을 위빙(Weaving : 엮기)라고 부른다

AOP에서는 각 모듈에 분리된기능을 이용하기 위한 코드를 기술할 필요가 없다.즉 AOP에서는 각 모듈의 개발자들이 횡단관점에 대해 전혀 알필요가 없도록 하는데 목적을 둔다

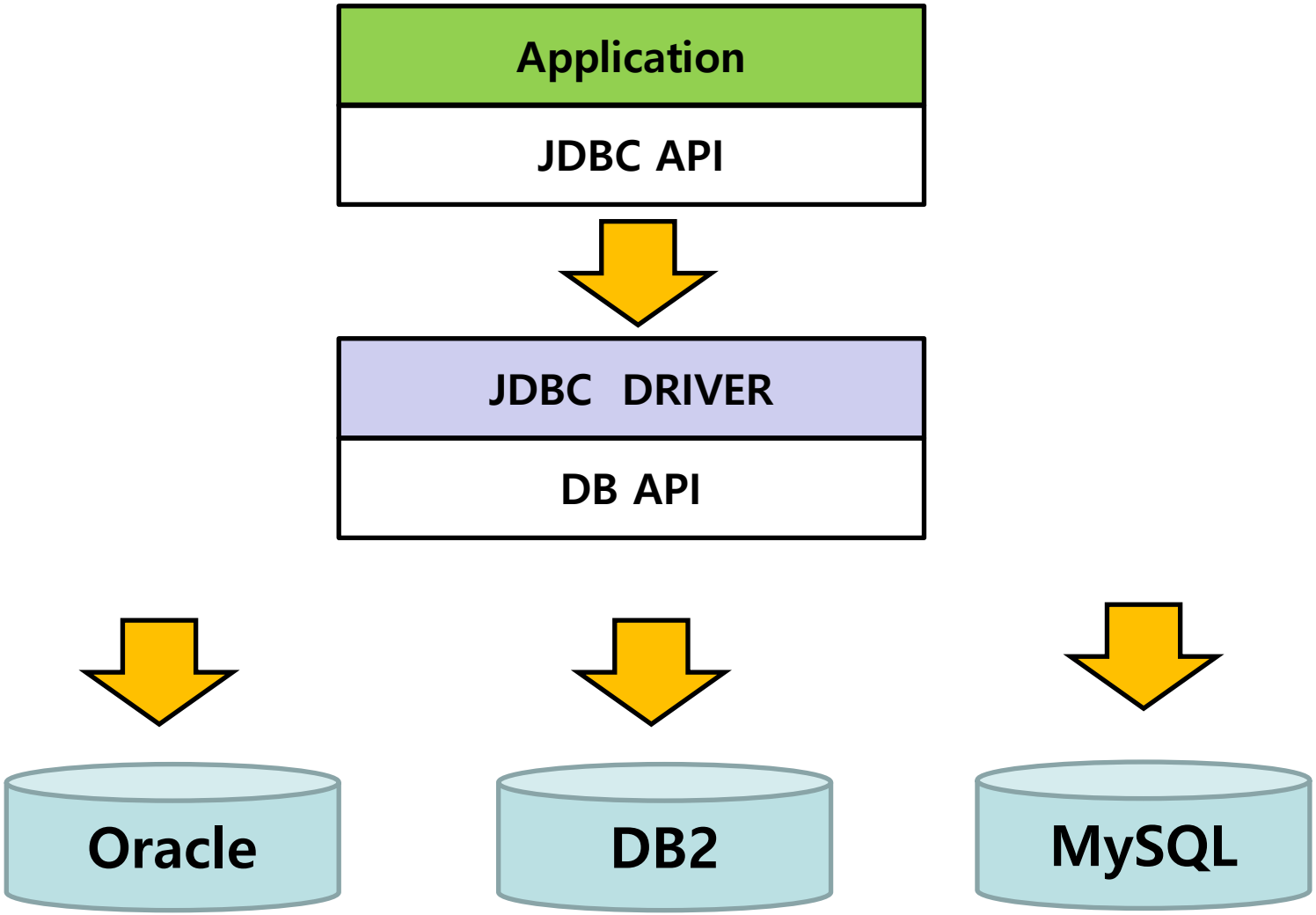
분리된 관점은 객체지향보다 독립성이 훨씬 높으며,재활용을 하거나 유지보수를 하기에 편리하다. 기존 소프트웨어의 코드에 손을 대지 않고도 새로운 기능을 추가하는것도 가능하다

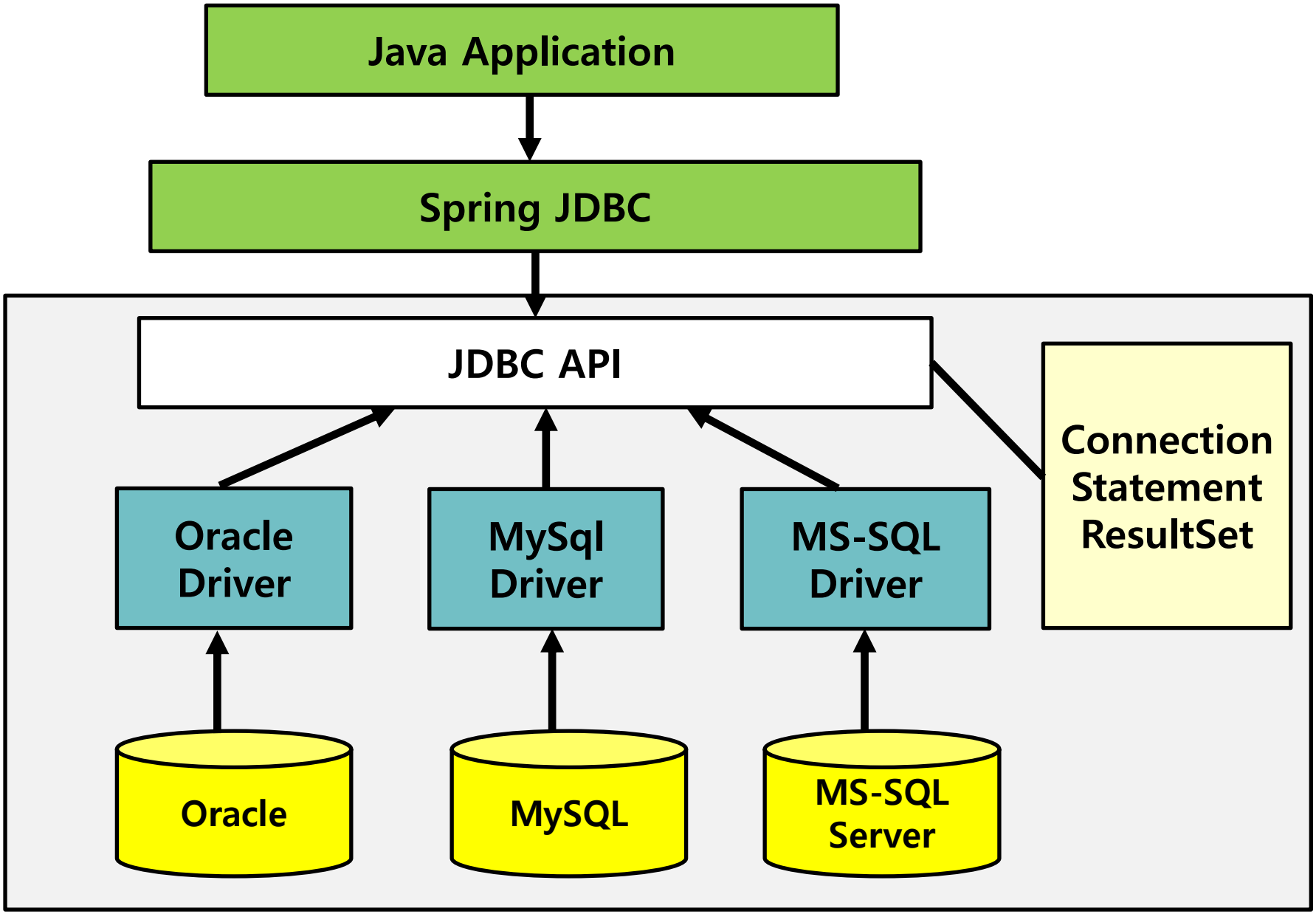
AOP 설정구조

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

3. Spring JDBC

JDBC를 이용하여 Database에 비 종속적인 DB연동을 구현 할 수 있다.
이는 vender들이 제공하는 JDBC드라이버를 이용하여 연동이 가능하며 DB2를 연동하고자 하면 DB2 드라이버, Oracle과 연동하려면 Oracle드라이버가 필요하다


Application
JDBC API
JDBC DRIVER
DB API
Oracle
DB2
MySQL


Java Application
Spring JDBC
JDBC API
Connection
Statement
ResultSet
Oracle
Driver
MySql
Driver
MS-SQL
Driver
Oracle
MySQL
MS-SQL
Server

5. MVC

1) 스프링MVC프레임워크는 스프링 기반으로 사용할수 있다

2) 스프링이 제공하는 트렌젝션처리가 DI및 AOP적용등을 손쉽게 사용할수 있다

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

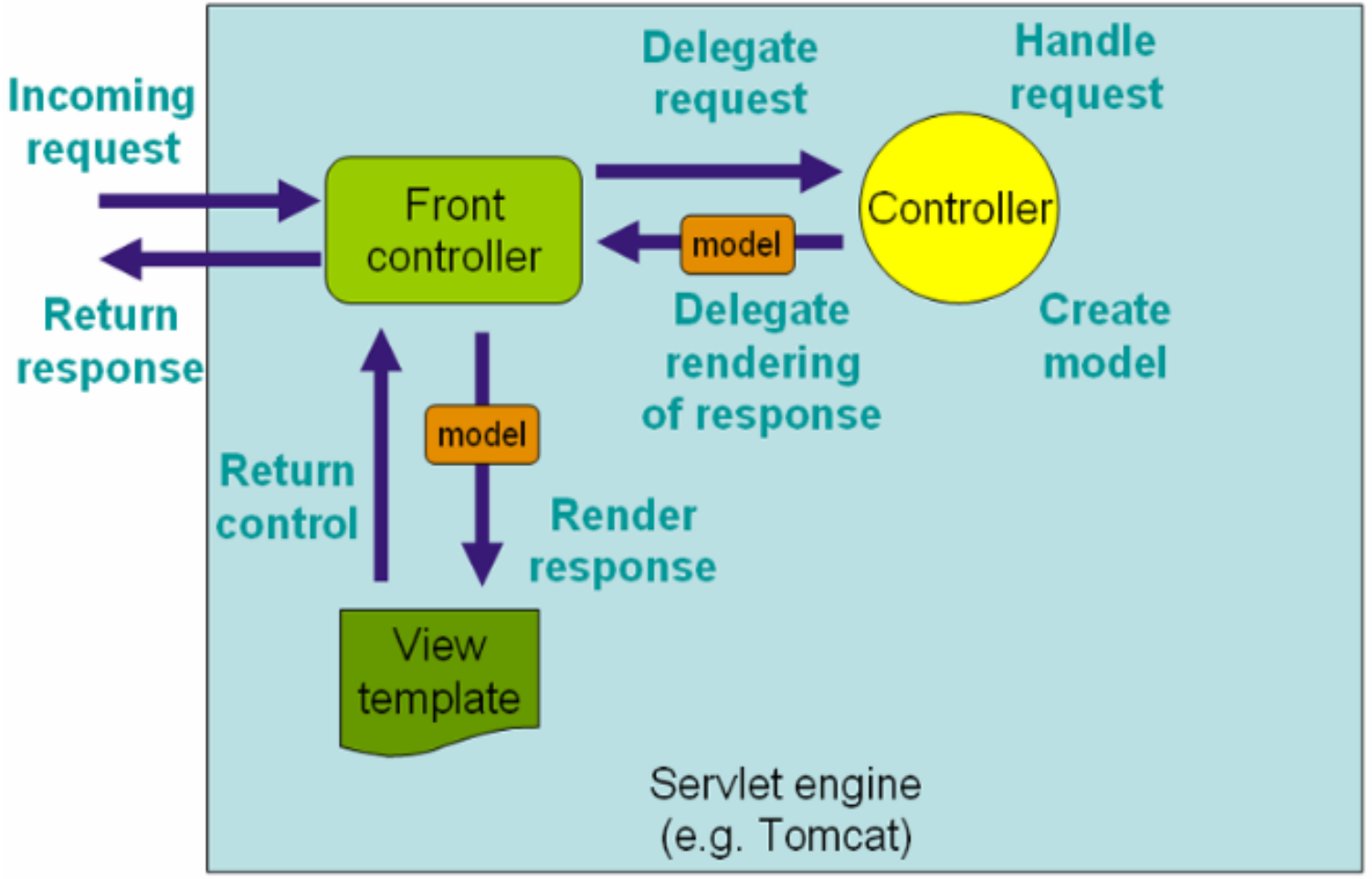
스프링 프레임워크에서 지원하는 Spring MVC는 **모델-뷰-컨트롤러 (MVC)** 구현을 포함하여 도메인 모델코드와 웹 폼을 깔끔하게 분리할 수 있도록 하고 스프링 프레임워크의 다른 모든 기능과 통합할 수 있게 하며 DI 와 선언적인 방식으로 MVC 기반의 웹 프로그램 개발을 효율적으로 할 수 있도록 지원한다.

Spring MVC (Model View Controller)의 대표적인 특징

- Model-And-View의 디자인 패턴(Model 2)의 아키텍처를 지원.
- Spring Framework의 다른 모듈과의 연계가 용이.
- 컨트롤러, command 객체, 모델 객체, Validator 등 각각의 역할에 대한 명확한 분리.
- Form 객체가 필요하지 않은 사용자 지정 가능한 데이터 바인딩과 유효성 체크 지원.
- 어떠한 View 기술과도 연계가 용이.
- 태그 라이브러리를 통한 Message 출력, Theme 적용 등과 입력폼을 보다 쉽게 구현.

1. spring MVC 처리 흐름

스프링 MVC는 프론트 컨트롤러 패턴을 적용한다. 프론트 컨트롤러 패턴이란, 하나의 핸들러 객체를 통해서 요청을 할당하고, 일관된 처리를 작성할 수 있게 하는 디자인 패턴이다.



브라우저로부터 받은 요청은 스프링 MVC가 제공하는
DispatcherServlet클래스 (FrontController)가 모두 관리한다.

web.xml파일에 다음과 같이 설정 추가

```
<web-app>

   <servlet>
      <servlet-name>mvc</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
       <servlet-name>mvc</servlet-name>
       <url-pattern>/*</url-pattern>
   </servlet-mapping>

</web-app>
```

위와 같이 설정을 추가하는 경우 스프링 MVC용 설정파일의 이름은 "mvc-servlet.xml"이 된다.

(스프링 MVC용 설정파일의 이름은 따로 설정하지 않으면 디폴트로 DispatcherServlet클래스의 servlet-name + "-servlet.xml"이 된다.)

※ 스프링 MVC를 사용하여 개발을 하는경우 스프링 설정파일은 두개 이상이 된다.

기존의 설정파일(ex: applicationContext.xml)과 MVC용 설정 파일(서블릿네임-servlet.xml)의 관계는 기존의 설정파일이 MVC설정파일의 부모(상속관계)가 된다)

```
<web-app>
  <listener>
     <listener-class>
            org.springframework.web.context.ContextLoaderListener</listener-class>
   </listener>
   <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/applicationContext.xml</param-value>
   </context-param>

   <servlet>
      <servlet-name>mvc</servlet-name>
      <servlet-class>
              org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>mvc</servlet-name>
      <url-pattern>/*</url-pattern>
   </servlet-mapping>

</web-app>
```

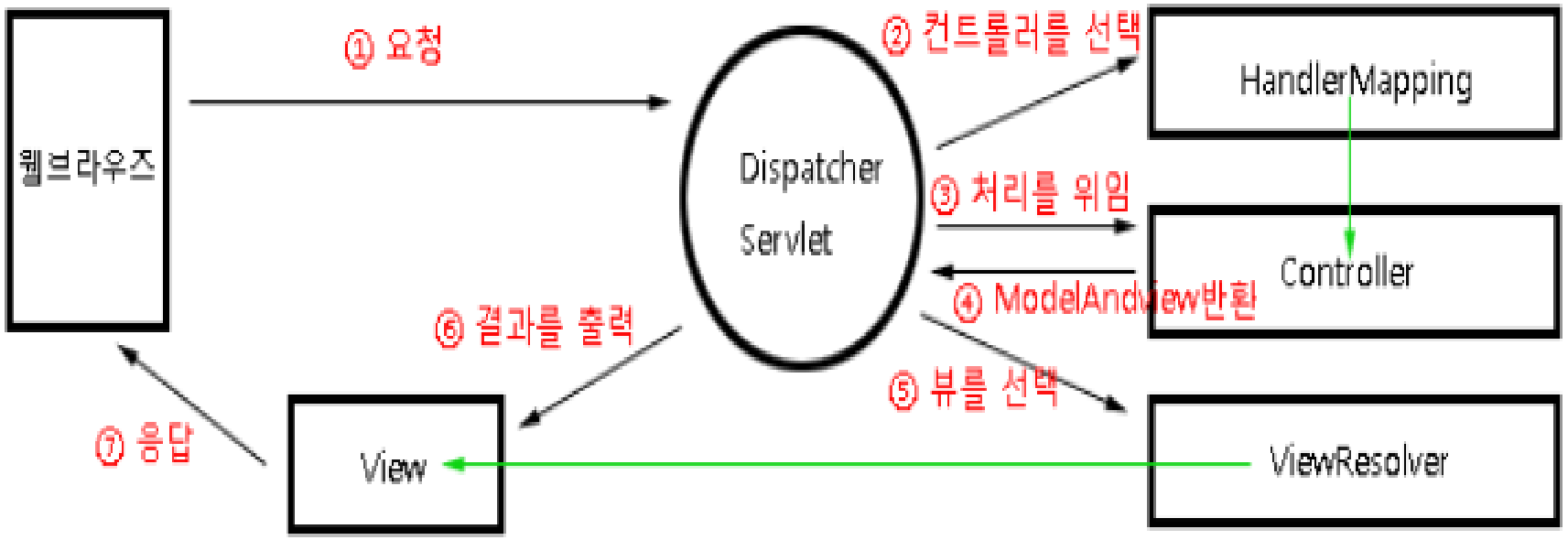
2. 스프링 MVC의 주요 구성요소

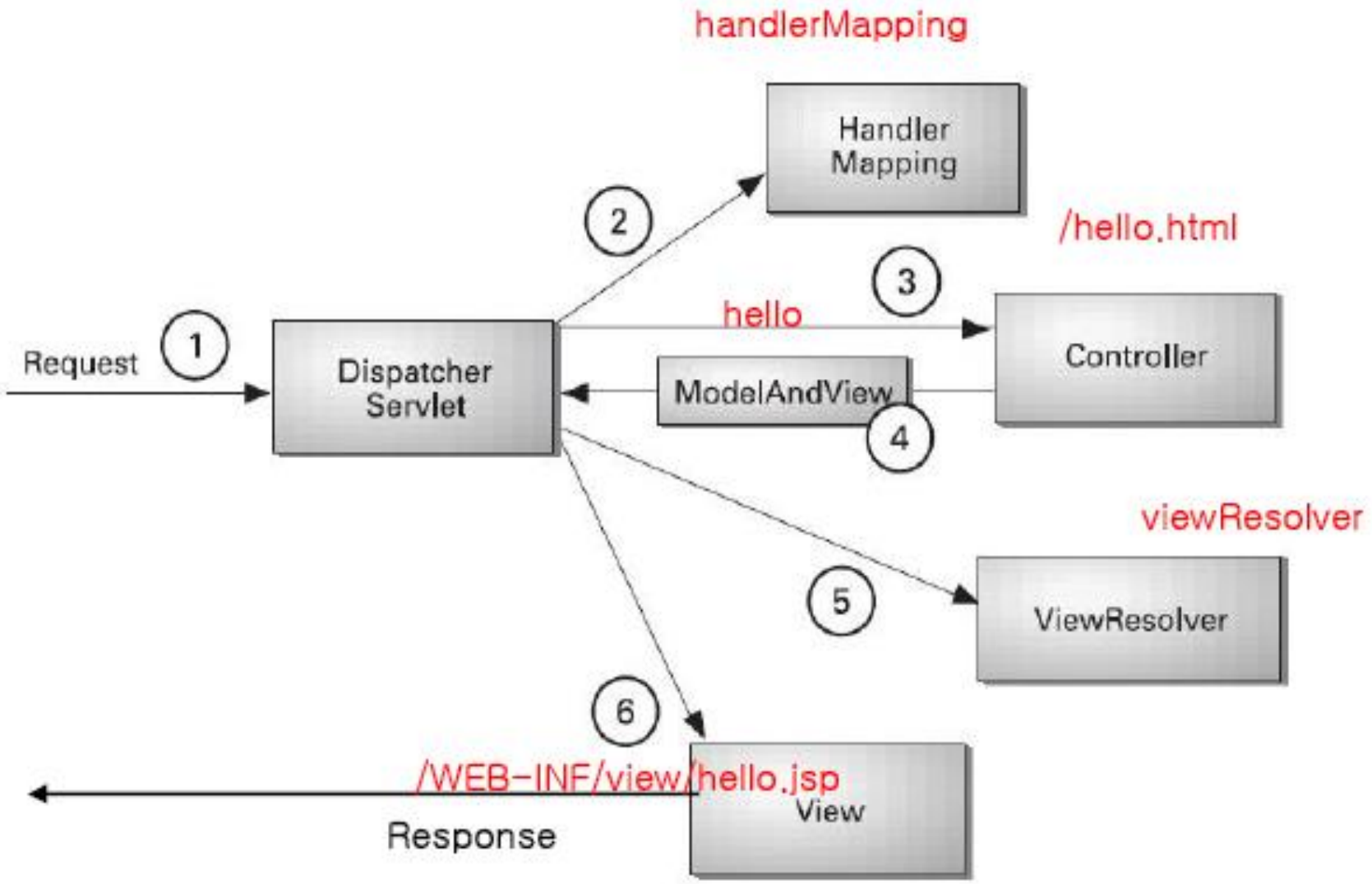
* spring-mvc 처리 흐름 요약

웹브라우저의 요청이 들어오면 DispatcherServlet의 객체가 이를 받는다. DispatcherServlet의 객체는 다시 HandlerMapping객체를 통해(참조하여) 어떤 Controller에게 처리를 위임해야할지를 통보받아서 그 Controller객체에게 처리를 위임한다.

Controller객체는 모델(Service-DAO)단과 통신을 하여 비지니스 로직을 호출하고 그 결과를 ModelAndView객체로 다시 DispatcherServlet 객체로 반환한다.

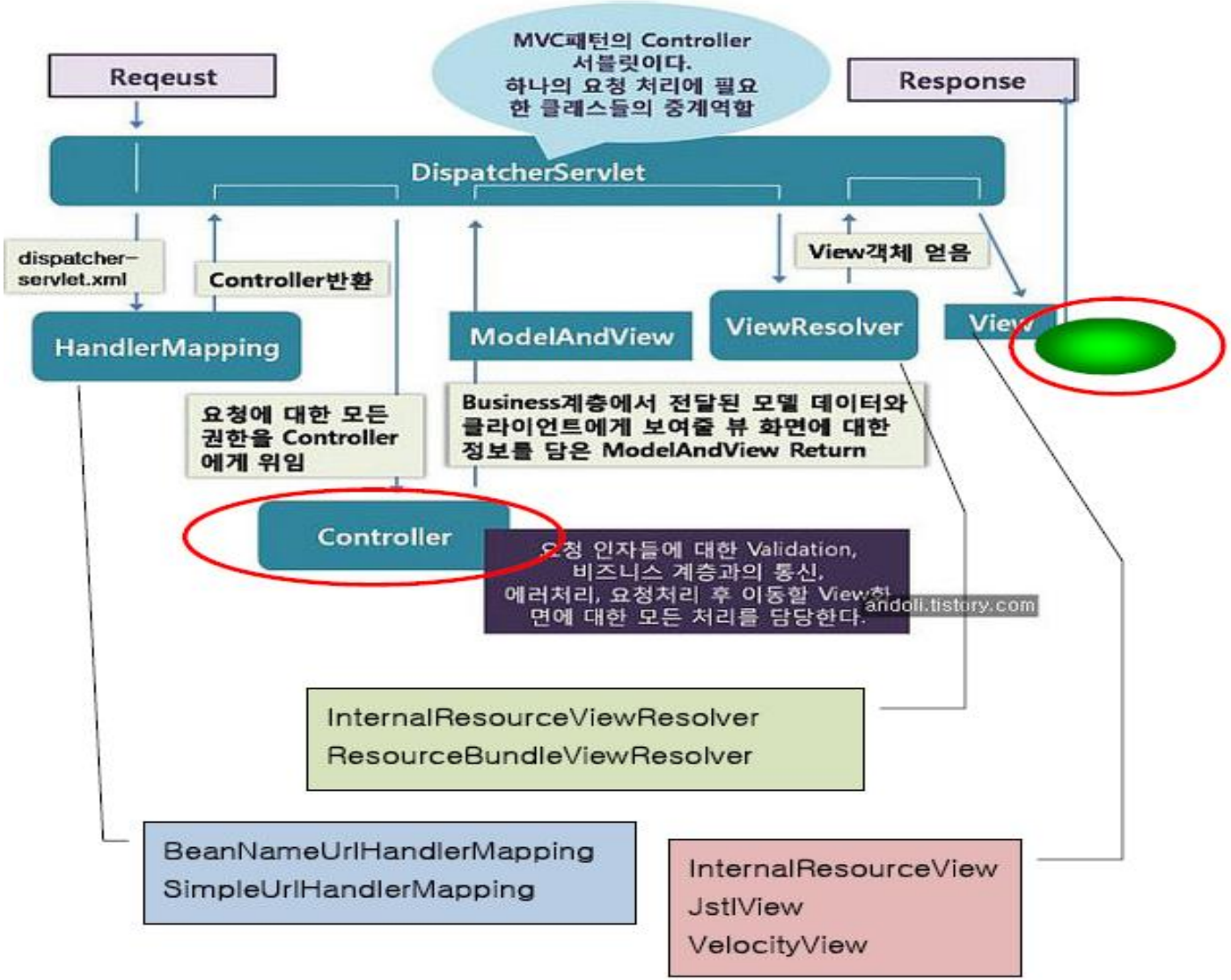
마지막으로 DispatcherServlet객체는 ViewResolver객체에게 사용할 View객체를 반환받아 그 View객체에 Controller객체가 반환한 정보(ModelAndView)를 포함시켜 그 결과를 브라우즈에 반환한다.


웹브라우즈
① 요청
Dispatcher
Servlet
② 컨트롤러를 선택
HandlerMapping
③ 처리를 위임
Controller
④ ModelAndView반환
⑤ 뷰를 선택
ViewResolver
⑥ 결과를 출력
View
⑦ 응답

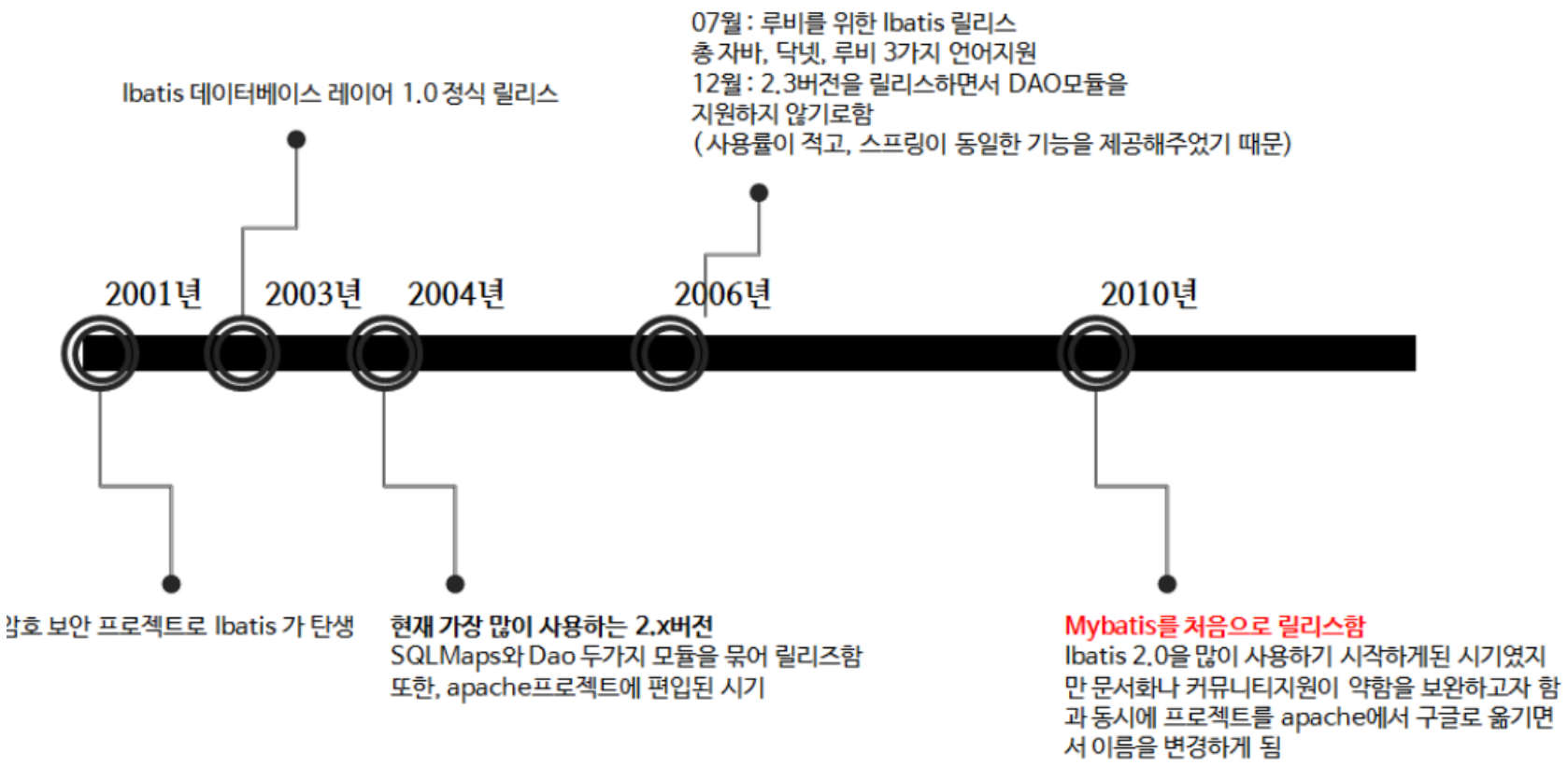

handlerMapping
Handler Mapping
/hello.html
2
3
hello
Request
1
Dispatcher Servlet
ModelAndView
Controller
4
viewResolver
5
ViewResolver
6
/WEB-INF/view/hello.jsp
View
Response

[Spring MVC 주요 클래스]

클래스명	기능
DispatcherServlet	클라이언트의 요청을 받는 프론트컨트롤러. Controller 에게 클라이언트 요청을 전달하고, 컨트롤러가 리턴한 결과 값을 View에 전달.
HandlerMapping	클라이언트 요청 URL을 어떤 컨트롤러가 처리할지 결정.
Controller	클라이언트 요청을 처리한 뒤, 그 결과를 DispatcherServlet에게 알려줌. (스트럿츠의 Action과 동일한 역할 수행)
ModelAndView	컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담는다.
ViewResolver	컨트롤러의 처리 결과를 생성할 뷰 결정.
View	컨트롤러의 처리 결과를 화면에 표시.


MVC패턴의 Controller 서블릿이다.
하나의 요청 처리에 필요한 클래스들의 중계역할
Reqeust
Response
DispatcherServlet
dispatcher-servlet.xml
Controller반환
View객체 얻음
HandlerMapping
ModelAndView
ViewResolver
View
요청에 대한 모든 권한을 Controller에게 위임
Business계층에서 전달된 모델 데이터와 클라이언트에게 보여줄 뷰 화면에 대한 정보를 담은 ModelAndView Return
Controller
요청 인자들에 대한 Validation, 비즈니스 계층과의 통신, 에러처리, 요청처리 후 이동할 View화면에 대한 모든 처리를 담당한다.
andoli.tistory.com
InternalResourceViewResolver
ResourceBundleViewResolver
BeanNameUrlHandlerMapping
SimpleUrlHandlerMapping
InternalResourceView
JstlView
VelocityView

5. MyBatis 개요


07월 : 루비를 위한 Ibatis 릴리스
총 자바, 닥넷, 루비 3가지 언어지원
12월 : 2.3버전을 릴리스하면서 DAO모듈을
지원하지 않기로함
(사용률이 적고, 스프링이 동일한 기능을 제공해주었기 때문)
Ibatis 데이터베이스 레이어 1.0 정식 릴리스
2001년
2003년
2004년
2006년
2010년
암호 보안 프로젝트로 Ibatis 가 탄생
현재 가장 많이 사용하는 2.x버전
SQLMaps와 Dao 두가지 모듈을 묶어 릴리즈함
또한, apache프로젝트에 편입된 시기
Mybatis를 처음으로 릴리스함
Ibatis 2.0을 많이 사용하기 시작하게된 시기였지
만 문서화나 커뮤니티지원이 약함을 보완하고자 함
과 동시에 프로젝트를 apache에서 구글로 옮기면
서 이름을 변경하게 됨

현재 Mybatis에서 제공하는 최신버전은 3.2.3버전

(1) 다운로드 : 라이브러리와 영문 PDF
http://blog.mybatis.org/p/products.html

(2) 한글 PDF
http://mybatis.github.com/mybatis-/ko/index.html

1. Mybatis와 ibatis의 차이점

구분	Ibatis(아이바티스)	Mybatis(마이바티스)
네임스페이스	선택사항	필수사항
매핑구문정의	xml만 사용	xml과 에노테이션을 사용
동적 SQL	XML 엘리먼트만 사용 동적 SQL을 위한 XML 엘리먼트는 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	향후 아이바티스에 대한 공식적인 릴리스는 없음	향후 계속 릴리스 될 예정

2.1 XML과 에노테이션

1. XML
사용방법은 기존과 동일하나 용어들이 변경되었다.

이전용어	변경된 용어
SqlMapConfig	Configuration
sqlMap	mapper

그 외에도 XML 엘리먼트가 축소되었다.

if	조건문
choose(when otherwise)	반복문
trim(where)	공백제거
foreach	반복문(IN절 사용시 유용)
set	update에서 마지막으로 명시된 칼럼 표기에서 쉼표제거

2. 인터페이스 (에노테이션 이용방법)

인터페이스를 이용하여 작성할 경우 에노테이션을 이용하여 작성이 가능하다.

XML에서 에노테이션으로 변경하는 방법은 다음과 같다.

매핑 구분의 네임스페이스	인퍼페이스의 패키지 명과 인터페이스 명
매핑 구분 아이디	에노테이션을 가진 메소드명
매핑 구분의 결과 데이터 타입(resultType 속성)	애노테이션을 가진 메소드의 반환 타입
매핑 구분의 파라미터 타입 (parameter Type 속성)	애노테이션을 가진 메소드의 파라미터 타입

※ Mapper를 정의하는 방법 (총 3가지)

정의방법	장점	단점
xml만 사용	mapper(과거 sqlMap)의 모든 기능을 이용할 수 있음	매핑구분의 아이디를 문자열 형태로 선언해야하기 때문에 버그 발생 빈도 높음 범용적인 API 특성상 타입 변환이 필요하고 타입변화 오류가 날 가능성이 있음
인터페이스만 사용	매핑 구문을 사용할 때 인터페이스의 메소드를 그대로 사용하기 때문에 매핑구문을 잘못적는 경우가 없음 타입변환이 필요없음	애노테이션 특성상 동적 SQL을 작성하기 위해 별도 클래스를 작성해야함 결과 매핑에 제약이 있음
XML과 애노테이션 같이 사용	매핑구문을 사용할 때 인터페이스의 메소드를 그대로 사용하기 때문에 매핑구문을 잘못 적는경우가 없음 mapper(과거 sqlMap)의 모든 기능을 사용할 수 있음	인터페이스와 매퍼 xml을 모두 작성해야 하기 때문에 코드 작성에 시간이 더 듬