Chapter 01
스프링 프레임워크 소개

1. 스프링 프레임워크란

- 웹사이트가 점점 커지면서 엔터프라이즈급의 서비스가 필요하게 됨
- 자바진영에서는 EJB가 엔터프라이즈급 서비스로 각광을 받게 됨
  - 세션빈에서 Transaction 관리가 용이함
  - 로깅, 분산처리, 보안등
- 하지만 EJB는 개발시 여러 가지 제약이 존재함(배보다 배꼽이 더 큼)
  - EJB스펙에 정의된 인터페이스에 따라 코드를 작성하므로 기존에 작성된 POJO를 변경해야 함
  - 컨테이너에 배포를 해야만 테스트가 가능해 개발속도가 저하됨
  - 배우기 어렵고, 설정해야 할 부분이 많음

1. 스프링 프레임워크란

- Rod Johnson이 'Expert One-on-One J2EE Development without EJB' 라는 저서에서 EJB를 사용하지 않고 엔터프라이즈 어플리케이션을 개발하는 방법을 소개함(스프링의 모태가 됨)
  - AOP나 DI같은 새로운 프로그래밍 방법론으로 가능
  - POJO로 선언적인 프로그래밍 모델이 가능해 짐

1. 스프링 프레임워크란

- 스프링 프레임 워크의 목표
  - 엔터프라이즈 서비스를 쉽게 구축
  - 의존성 주입(Dependency Injection)을 통한 유연한 프레임워크 구현
  - 관점지향 프로그래밍(Aspect oriented Programming) 지원
  - Application의 완전한 이식성 제공
  - 반복적인 코드의 제거
  - 생산성 향상

1. 스프링 프레임워크란

- **스프링의 특징**
  - 경량의 컨테이너 프레임워크
  - 평범한 자바빈을 이용하여 EJB에서 가능한 서비스를 수행함
  - 의존성 주입을 통해 느슨한 결합을 도모함(xml문서를 통해 선언적으로)
  - 관점지향 프로그래밍을 통해 비즈니스 로직과 서비스를 분리하여 응집된 개발이 가능함
  - 다양한 API, 프레임워크와의 연동을 지원함

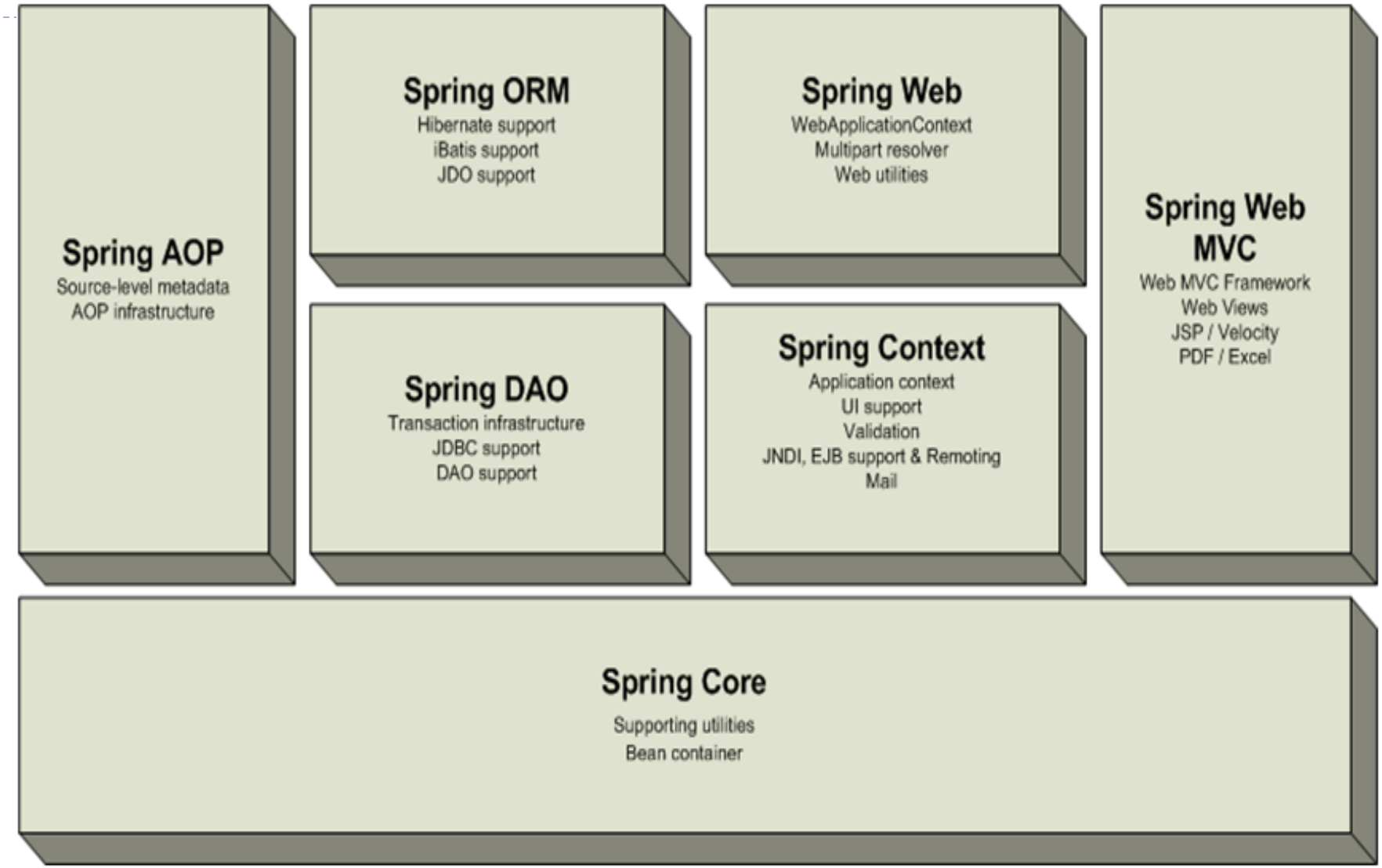
1. 스프링 프레임워크란

- 스프링의 장점
  - 필요한 인스턴스를 스프링에서 미리 생성해 준다.
  - 클래스 사이의 결합(loosely coupled)을 느슨하게 할 수 있어 클래스 간의 의존 관계가 약해진다.

2. 스프링 프레임워크 설치와 모듈 구성

- 스프링 예제를 작성하기 위해서는 JDK를 설치
- 개발 툴로 이클립스를 설치
- 개발 환경 세팅을 위해서 이클립스에 스프링 플러그인을 설치
- 스프링 프레임워크 인스톨

2. 스프링 프레임워크 설치와 모듈 구성



2. 스프링 프레임워크 설치와 모듈 구성

- Spring Core
  - 스프링의 가장 기본적인 기능 수행(DI)
  - 모든 스프링 애플리케이션 기반인 Bean Factory를 포함하고 있음
  - Bean Factory는 DI를 통해 설정과 의존성을 실제 코드와 분리하는 팩토리패턴 구현객체임
- Spring Context
  - 국제화메시지, 애플리케이션 생명주기, 이벤트, 유효성 검증등을 지원함
  - 이메일, jndi접근, ejb 연계, 스케쥴링 등의 엔터프라이즈 서비스를 지원

2. 스프링 프레임워크 설치와 모듈 구성

- Spring AOP
  - 관점지향 프로그래밍 지향
  - 메타데이터의 지원을 통해 Aspect를 적용할지 알려주는 annotation을 추가할 수 있음
  - 사용하거나 배우기가 어려움
- Spring ORM
  - jdbc 상위에서 ORM 프레임워크와 연동가능하게 함
  - 연동된 ORM 프레임워크의 트랜잭션도 관리가 가능

2. 스프링 프레임워크 설치와 모듈 구성

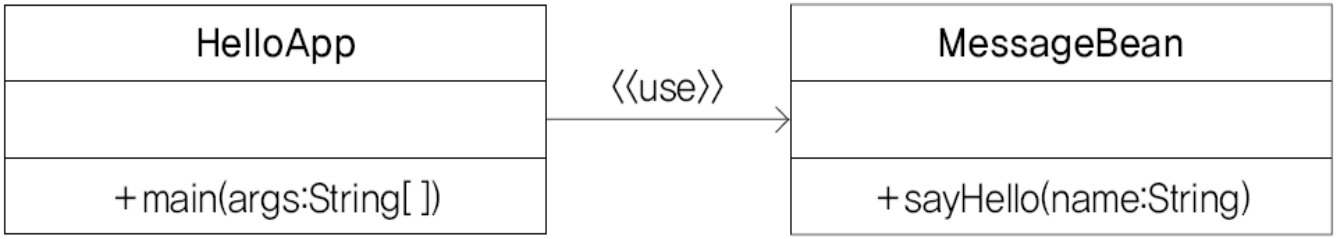
- Spring DAO
  - jdbc로 작업할 때 연결취득, 명령, 실행, 결과집합처리 연결 끊기 등의 반복적인 코드를 추상화함
  - 데이터베이스 접근 코드를 간결하게 함
  - 트랜잭션 서비스를 제공
  - 다른 ORM보다 코딩하기 불편하여 거의 사용하지 않음
- Spring Web
  - 웹 기반 어플리케이션에 적합한 Context를 제공함
  - 파일업로드 등의 multipart 요청처리나 요청파라미터를 빈즈에 바인딩하는 등의 웹 관련 작업을 지원
  - 스트럿츠, 스트럿츠2와 연동

2. 스프링 프레임워크 설치와 모듈 구성

- Spring Web MVC
  - 웹 어플리케이션 구축을 위한 모델-뷰-컨트롤러 프레임워크 제공
  - 사용시 요청 파라미터를 선언적인 방법으로 비즈니스 객체에 바인딩 가능
  - 스트럿츠의 성능이 우수하므로 사용빈도 낮음
- 그 외 JMX, JCA, JMS, portlet MVC, remoting 등
  - 팩토리패턴이란?
    - 객체를 생성하기 위한 인터페이스를 정의하는데, 어떤 클래스 인스턴스를 만들지는 서브클래스에서 결정하게 만든다. 구성 요소별로 '객체의 집합'을 생성해야 할 때 유용하다. 이 패턴을 사용하여 상황에 알맞은 객체를 생성할 수 있다.

세 개의 예제를 통한 스프링의 기본 기능과 구조

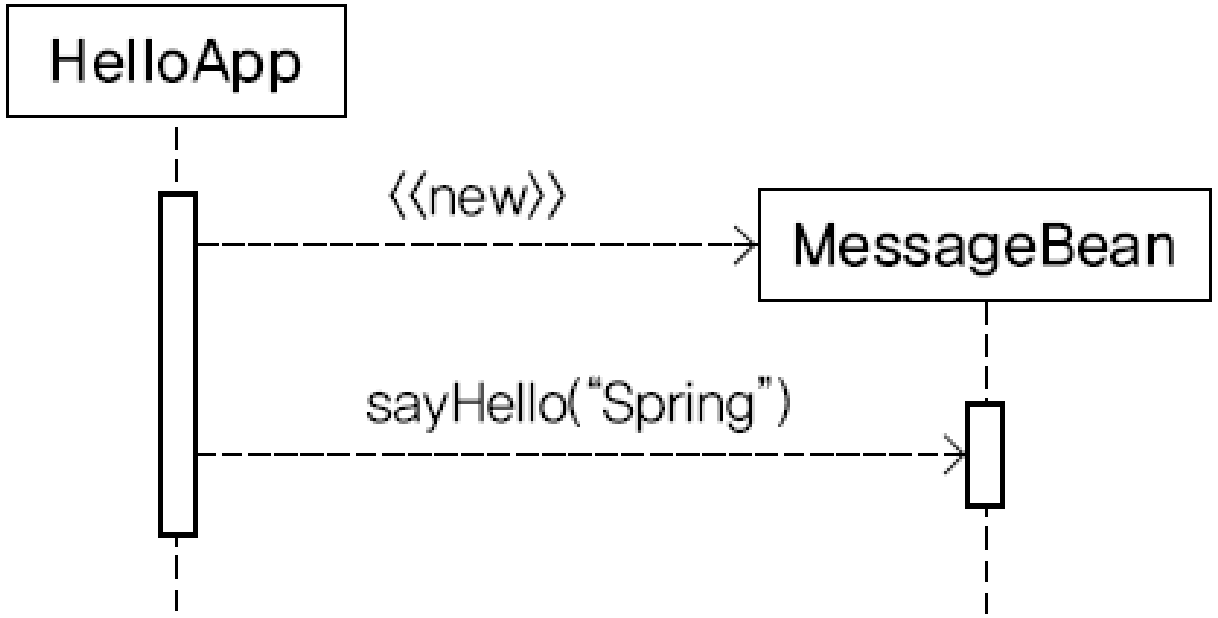
- 2.1 예제 설명
  - 단순 샘플 Application 1
    - MessageBean의 멤버인 sayHello(String) 메서드를 이용하여 HelloApp 에서 화면에 Hello,… 출력 하는 예제



▲ sample1 클래스 그림

세 개의 예제를 통한 스프링의 기본 기능과 구조

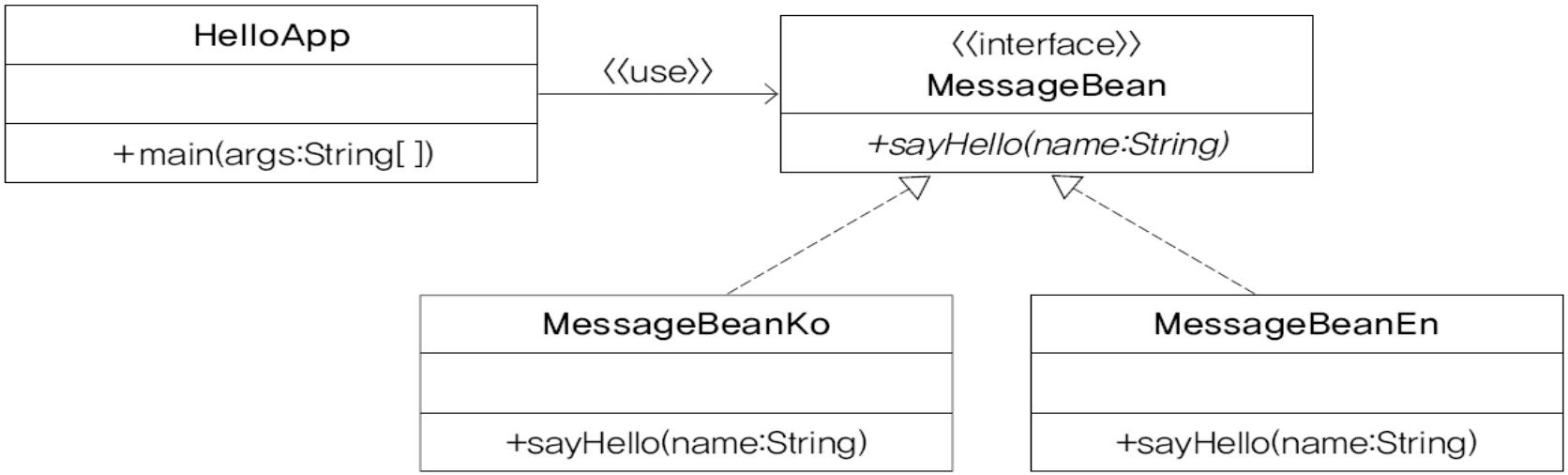
- 가장 단순한 샘플 어플리케이션 분석하기(sample1)



▲ Sample 샘플 어플리케이션 시퀀스 그림

세 개의 예제를 통한 스프링의 기본 기능과 구조

- 단순 샘플 Application 2
  - 인터페이스를 이용하여 클래스 사이의 의존 관계 낮추어 보자
  - MessageBean의 멤버인 sayHello(String) 메서드를 이용하여 HelloApp 에서 화면에 Hello,… 출력 하는 예제



세 개의 예제를 통한 스프링의 기본 기능과 구조

- 인터페이스 적용한 샘플 어플리케이션 분석하기 (sample2)

```
MessageBean bean = new MessageBeanEn( );
Bean.sayHello("Spring");
```

수정하지 않아도 되는 코드

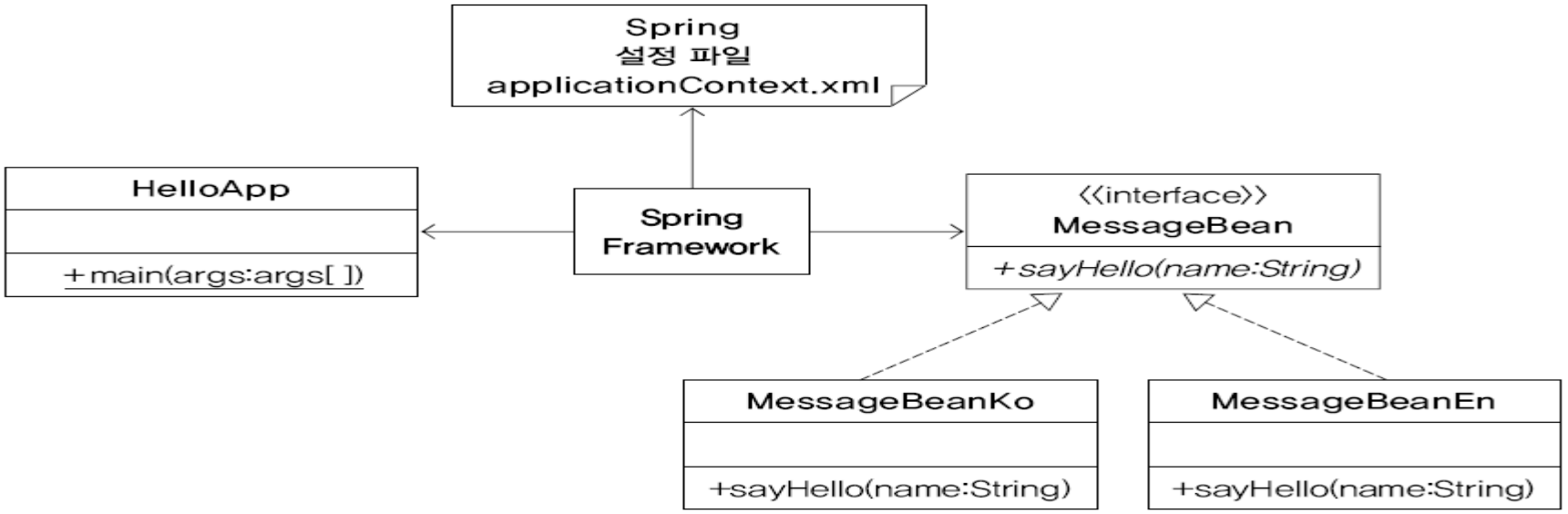
```
MessageBean bean = new MessageBeanKo( );
Bean.sayHello("Spring");
```

아직은 수정해야 하는 코드가 존재

수정하지 않아도 되는 코드

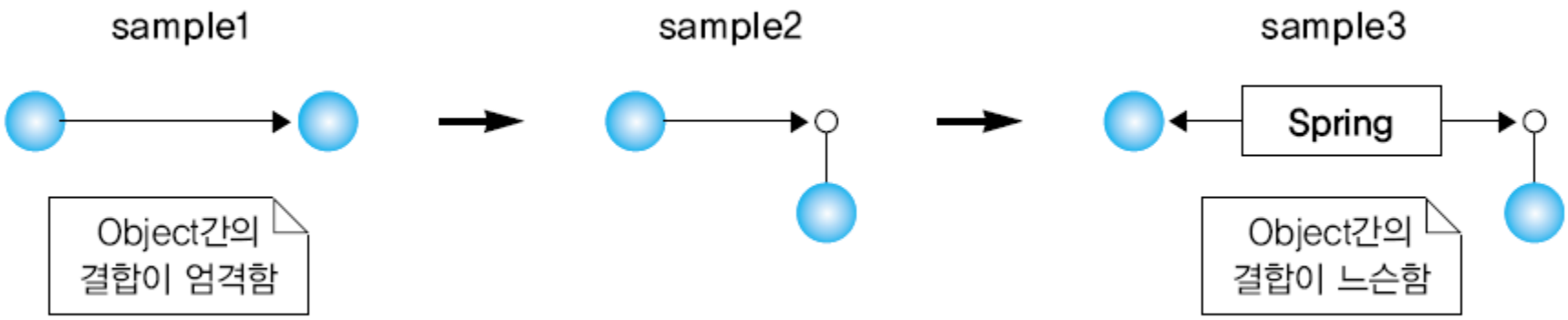
세 개의 예제를 통한 스프링의 기본 기능과 구조

- 단순 샘플 Application 3
  - 스프링인터페이스를 이용, 클래스 사이의 의존 관계 낮추어 보자
  - MessageBean의 멤버인 sayHello(String) 메서드를 이용하여 HelloApp 에서 화면에 Hello,… 출력 하는 예제



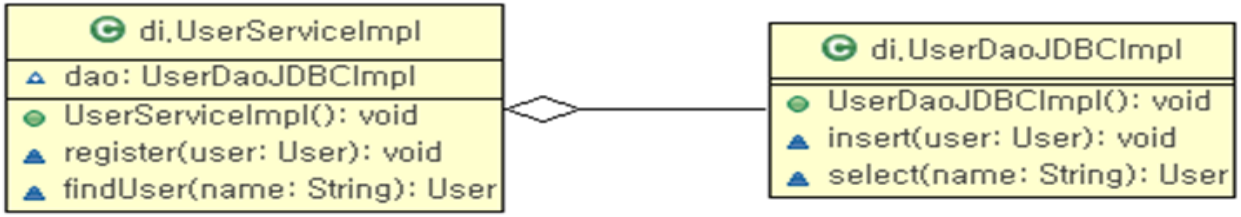
세 개의 예제를 통한 스프링의 기본 기능과 구조

- 2.2 스프링을 활용하기 위한 설계



3. Dependency Injection 과 스프링프레임워크

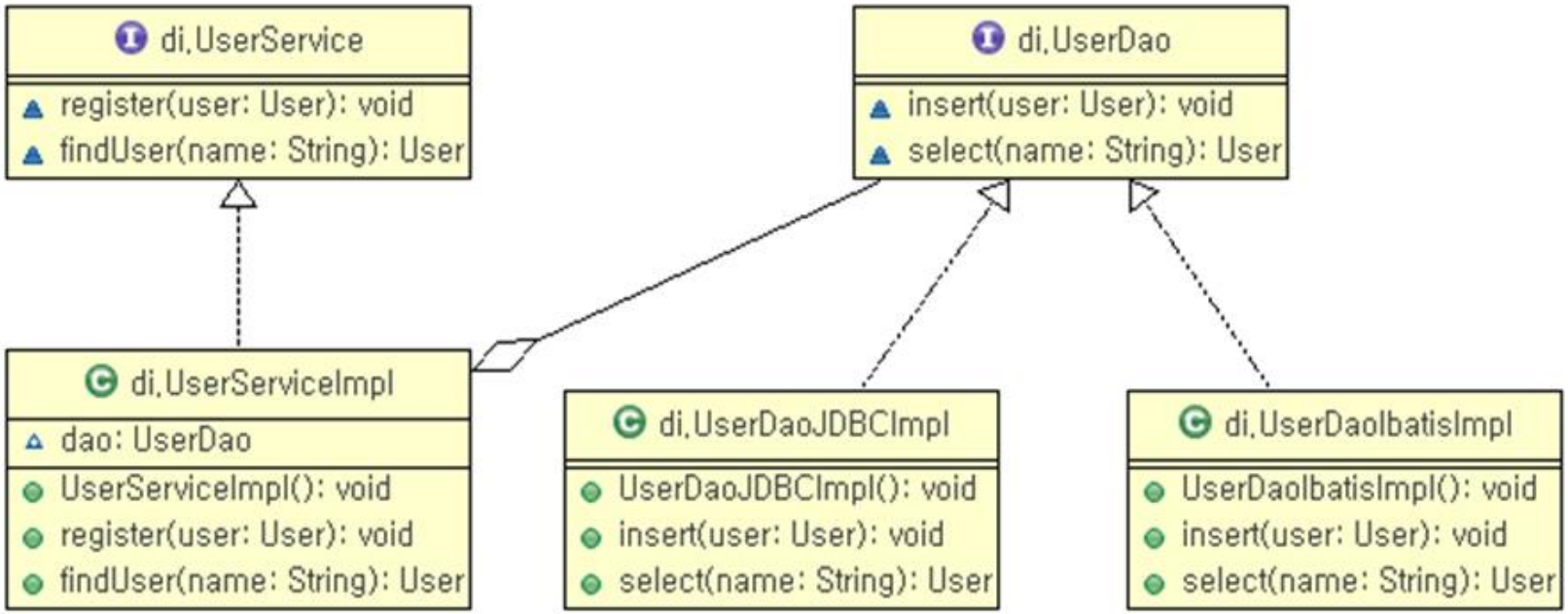
- 의존성이 높은 결합



  - 자신이 사용할 DAO객체를 직접 new라는 키워드를 사용하여 생성하여 사용할 경우 의존성이 높음
  - 프로젝트 스펙이 바뀌거나, 다른 DAO를 참조해야 할 때 코드의 수정이 불가피함

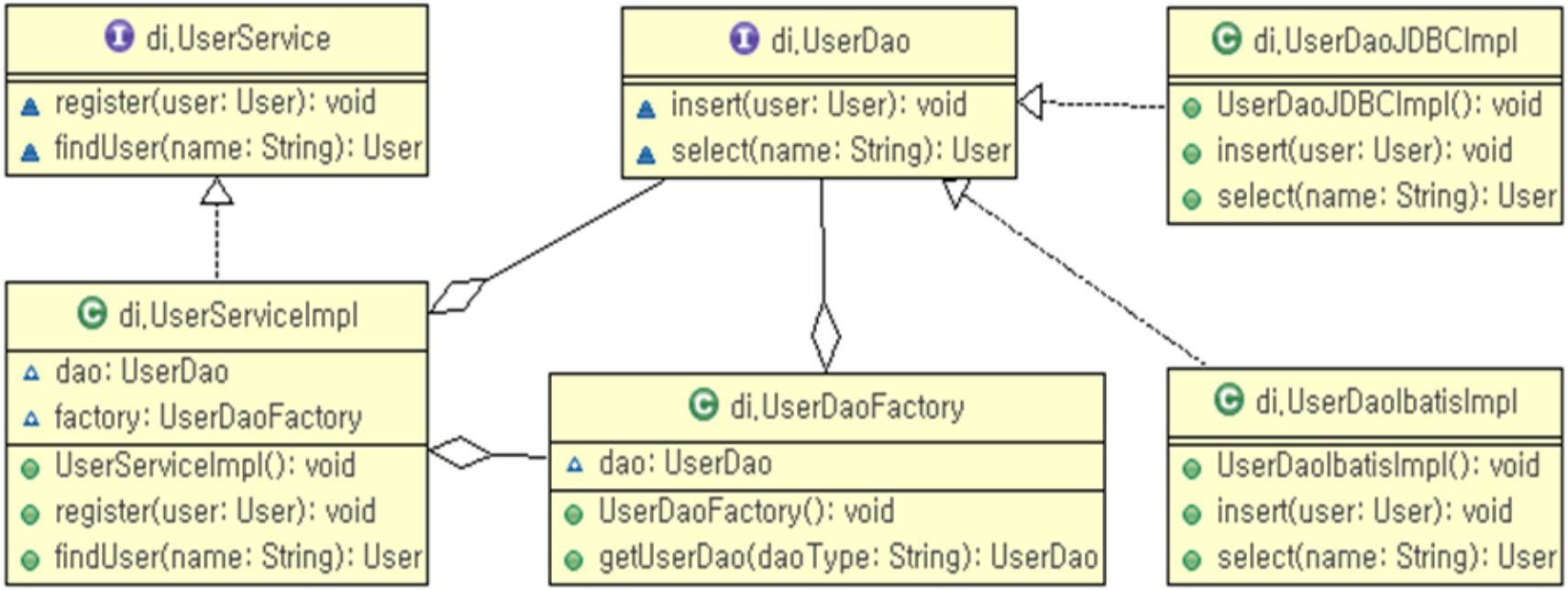
3. Dependency Injection 과 스프링프레임워크

- 인터페이스를 사용함



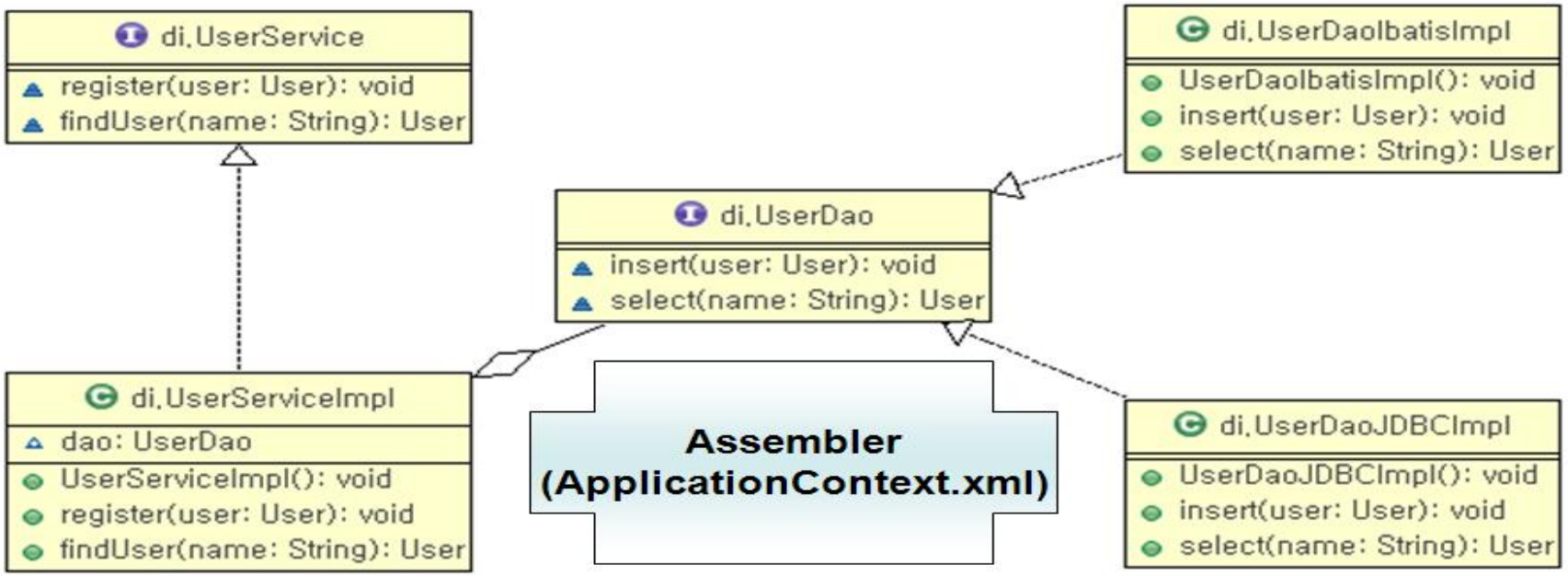
3. Dependency Injection 과 스프링프레임워크

- Factory패턴을 이용하여 의존성을 낮춤



3. Dependency Injection 과 스프링프레임워크

- 조립기(Assembler)를 통해 의존성 주입

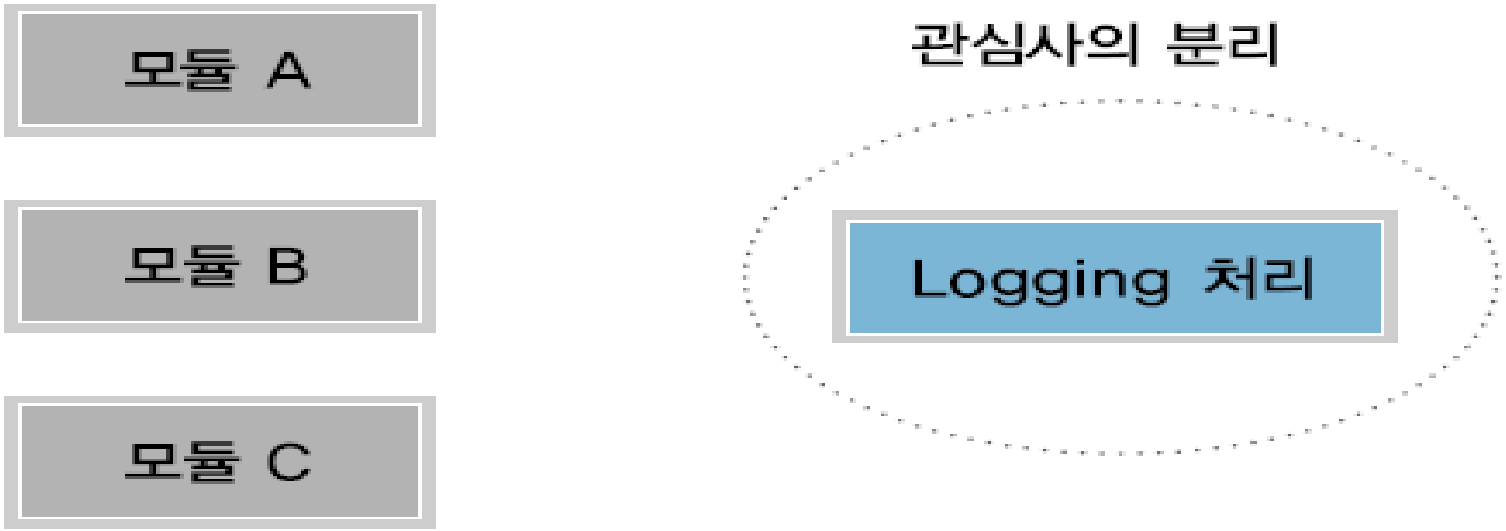


4. AOP 와 스프링

- AOP (Aspect oriented Programming)소개
  - 의존 관계의 복잡성과 코드 중복을 해소해 주는 프로그래밍 기법
- Aspect 지향에서 중요한 개념은 「횡단 관점의 분리 (Separation of Cross-Cutting Concern)」임
- 은행 업무를 처리하는 시스템
  - 은행 업무 중에서 계좌이체, 이자계산, 대출처리 등은 주된 업무(핵심 관점, 핵심 비즈니스 기능)로 볼 수 있다.
  - 로깅」, 「보안」, 「트랜잭션」등의 처리는 어플리케이션 전반에 걸쳐 필요한 기능으로 핵심 비즈니스 기능과는 구분하기 위해서 공통 관심 사항(Cross-Cutting Concern)이라고 표현한다.

4. AOP 와 스프링

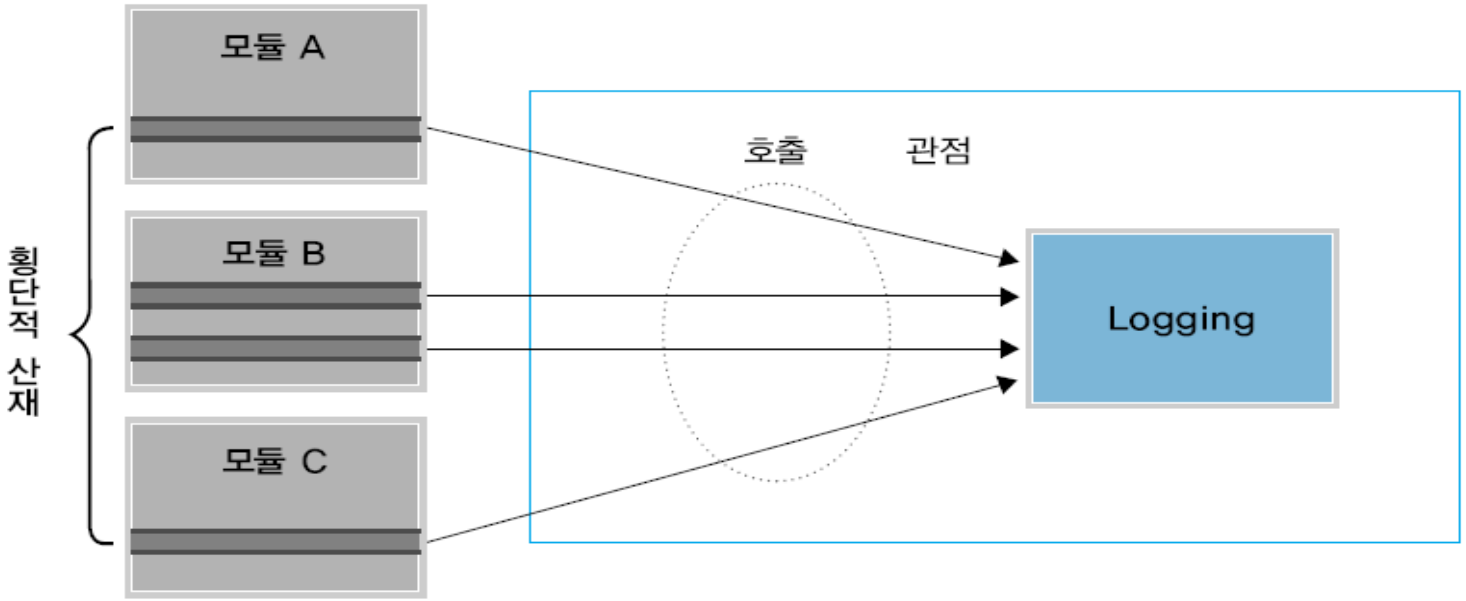
- 오브젝트 지향에서는 이들 업무들을 하나의 클래스라는 단위로 모으고 그것들을 모듈로부터 분리함으로써 재이용성과 보수성을 높이고 있다.



▲ 기존의 객체 지향에서 관점의 분리

4. AOP 와 스프링

- 오브젝트 지향에서는 로깅이라는 기능 및 관련하는 데이터 자체는 각 모듈로부터 분리하는 것으로 성공했지만 그 기능을 사용하기 위해서 코드까지는 각 모듈로부터 분리할 수 없다. 그렇기 때문에 분리한 기능을 이용하기 위해서 코드가 각 모듈에 횡단으로 산재하게 된다.



▲ 횡단적으로 산재하는 '기능의 호출'

1.1 Aspect 지향과 횡단 관점 분리

- AOP에서는 분리한 기능의 호출도 포함하여 「관점」으로 다룬다. 그리고 이러한 각 모듈로 산재한 관점을 「횡단 관점」라 부르고 있다.
- AOP에서는 이러한「횡단 관점」까지 분리함으로써 각 모듈로부터 관점에 관한 코드를 완전히 제거하는 것을 목표로 한다.