SPRING FRAMEWORK

백성애

1. 스프링을 위한 환경설정

[1] 스프링 프레임웤 다운로드

http://www.springsource.org/download

[2] 이클립스에서 Eclipse Marketplace 에서 STS 를 검색하여 설치

[3]또는 스프링 전용 IDE인 **STS를 설치하여 개발**하자.

https://spring.io/tools/sts/all

운영체제에 맞는 파일을 다운로드하여 압축을 푼다.

⇒ **(우리는 sts로 한다)**

[4] spring framework api

http://docs.spring.io/spring/docs/3.2.18.RELEASE/javadoc-api/

2. SPRING FRAMEWORK이란?

- Enterprise Application에서 필요로하는 기능을 제공하는 프레임워크.
- Enterprise 애플리케이션 개발을 쉽고, 편리하게 할 수 있도록 지원해주는 경량급 오픈 소스 프레임워크

- Spring 탄생 배경
- 창시자 : Road Johnson
- 2003년 오픈 소스로 시작된 프로젝트
- Spring의 모태 : “Expert One-on-One J2EE Development without EJB”라는 책을 통해 제시 했던 예제 중
- EJB를 사용하지 않고 Enterprise Application 전 계층에 등장하는 기술과 애플리케이션의 전 영역에 대한 효과적인 설계와 개발 기법 소개

3. SPRING FRAMEWORK 특징

[1] 스프링은 경량컨테이너다

. -자바 객체의 생성,소멸 등 라이프사이클을 관리함

[2] DI(Dependency Injection)를 지원

-설정파일이나 어노테이션을 통해 객체 간의 의존관계를 설정하도록 함.

3. SPRING FRAMEWORK 특징

[3] AOP(Aspect Oriented Programming)를 지원

- 트랜잭션이나 로깅, 보안과 같이 여러 모듈에서 공통으로 필요하지만, 실제 모듈의 핵심은 아닌 기능들을 분리해서 각 모듈에 적용 가능

[4] POJO(Plain Old Java Object)를 지원

- **- POJO 란? -**특정 규약 및 환경에 종속적이지 않은 평범한 일반 자바 클래스 의미
- 스프링 컨테이너에 저장되는 자바 객체는 특정한 인터페이스를 구현하거나 클래스를 상속받지 않아도 된다.
- 기존 코드 수정 없이 사용 가능하며 특정 구현 기술에 종속적이지 않음.

3. SPRING FRAMEWORK 특징

[5] 트랜잭션 처리를 위한 일관된 방법을 제공

- -JDBC API 및 JTA를 사용하거나 컨테이너가 제공하는 트랜잭션을 사용하든
- 설정 파일을 통해 트랜잭션 정보를 관리하기 때문에
- 특정 트랜잭션 구현 방법에 상관없이 동일한 코드를 여러 환경에서 사용 가능

[6] 영속성과 관련된 다양한 API를 지원

- - iBatis, MyBatis, Hibernate 등 데이터베이스 처리를 위해 널리 사용되는 library들과의 연동을 지원

[7] 그 외의 다양한 API를 지원.

- JMS,스케줄링 등 많은 API를 설정파일,어노테이션 통해 쉽게 사용하도록 지원

4. SPRING FRAMEWORK 장점

- 기술에 대한 접근 방식이 일관성이 있고, 특정 환경에 종속적이지 않아
- **개발자들이 애플리케이션 로직 개발에만 집중할 수 있음**
- 서버 등의 실행 환경이 바뀌고 적용되는 조건이 바뀐다 해도 코드까지 수정할 필요가 없음

- **개발이 단순해짐**
- Spring의 의존 관계, 트랜잭션등의 설정 방법에 대한 지식을 습득한 후에는 설정 적용 기술만으로도 Enterprise 개발의 기술적인 복잡함과 그에 따른 수고를 덜 수 있다.

- **POJO 방식의 기술 사용 가능**
- 특정 규약 및 환경에 종속되지 않은 일반 자바 클래스를 지원하므로 컨테이너에 의존적인 코드를 추가하지 않아도 애플리케이션을 개발 할 수 있다. 개발 후의 테스트도 쉽고 빠르게 할 수 있음

5. SPRING FRAMEWORK의 모듈

- Spring3.0은 20여 개의 모듈로 구성

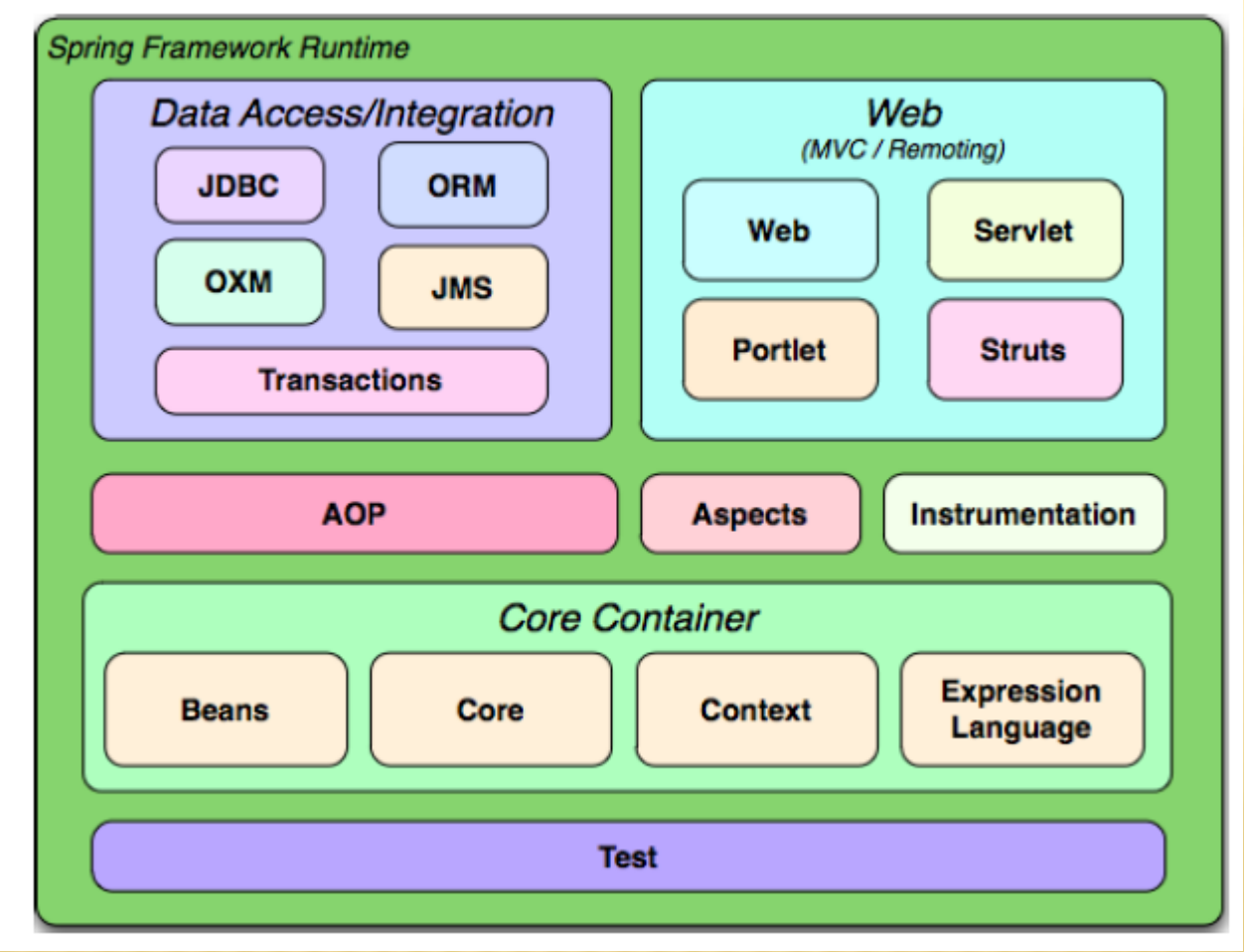


그림 출처

http://www.springsource.org

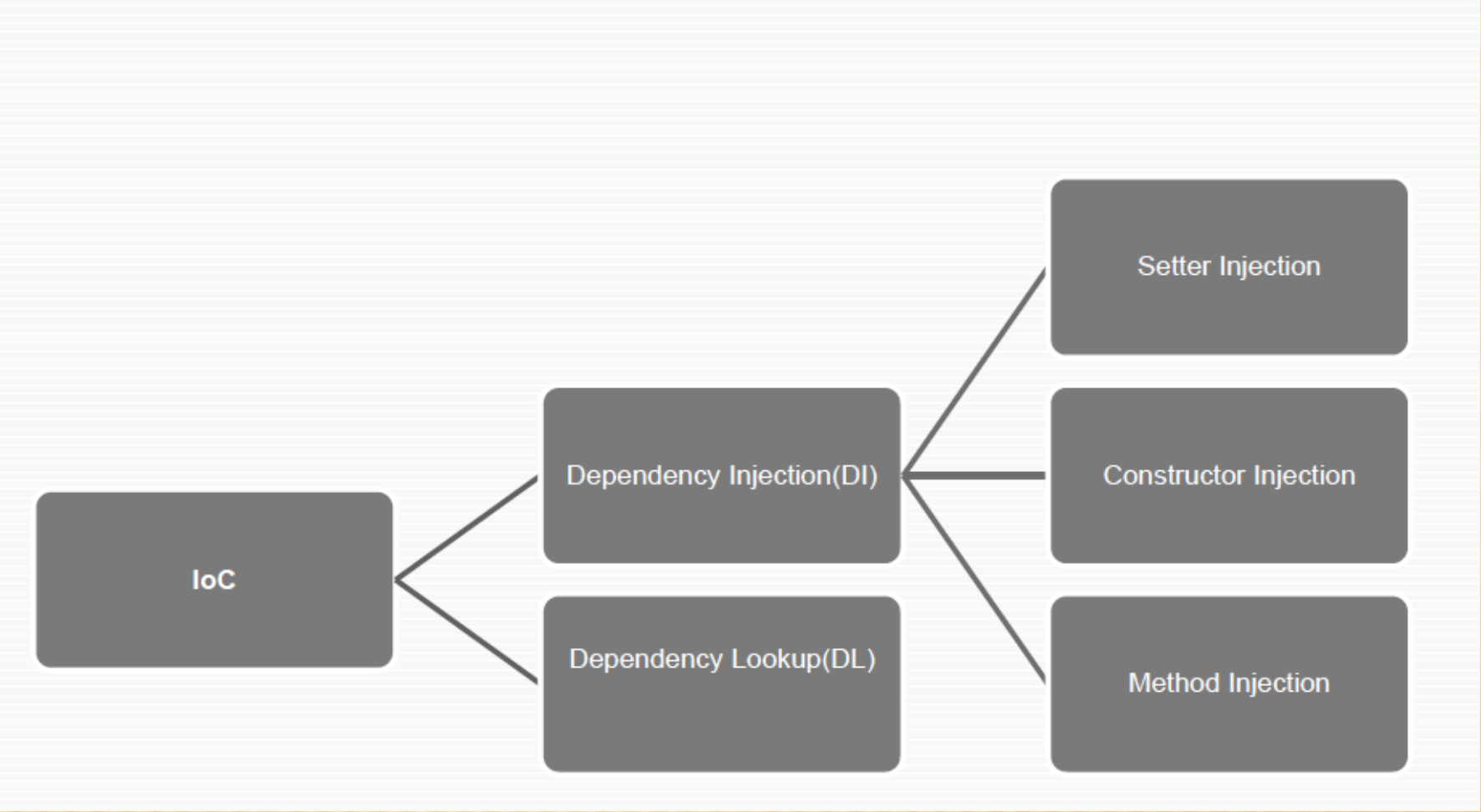
5. 스프링을 구성하는 모듈

- Spring Core : IoC 기능을 지원하는 영역을 담당. BeanFactory를 기반으로 Bean클래스들을 제어할 수 있는 기능 지원
- Spring AOP : Aspect Oriented Programming
- Spring ORM : MyBatis, Hibernate 등의 ORM프레임워크 지원
- Spring DAO : DAO 기능 지원
- Spring Web : 스트럿, 웹워크 같은 프레임웍의 통합을 지원.
  Web Application Context와 Multipart Request를 지원
- Spring Context: JNDI 및 EJB 지원, 메일 송.수신 기능 제공
- Spring Web MVC : MVC 프레임웍 기능을 제공.

6. 스프링에서 제공하는 기능 중 IOC란?

- IoC (Inversion of Control)- 역제어
- 예전에는 객체 생성 및 서로 간의 의존관계 연결 등에 대한 제어권을 개발자들이 가졌던 반면,
  스프링 프레임웍 에서는 객체 생성에서부터 생명 주기의 관리까지 모든 객체에 대한 제어권을 스프링 컨테이너가 제공하고 있는데,
  이를 역제어 (IoC)라고 한다.

4_1. IOC 컨테이너 분류 체계



4_1. IOC 컨테이너 분류 체계

- [1] DL (Dependency Lookup)
- [2] DI (Dependency Injection)
- <1> Setter Injection
- <2> Constructor Injection
- <3> Method Injection

[1] DL (DEPENDENCY LOOKUP)

- 컨테이너 저장소에서 관리되고 있는 빈을 개발자들이 직접 LOOKUP하여 사용하는 것을 Dependency Lookup이라고 한다.

- DL을 사용할 경우 빈을 룩업하기 위해 컨테이너에서 제공하는 API를 사용해야 한다.
  - ▶ 이 경우 의존관계가 발생함
  - ▶ 종속성 문제 발생

따라서 이를 줄이는 방법으로 DI를 사용

[2] DI (DEPENDENCY INJECTION)

- 각 클래스 사이의 의존관계를 빈 설정-xml파일 정보를 바탕으로 컨테이너가 자동으로 연결해 주는 것.

- 컨테이너가 의존관계를 자동으로 연결해주기 때문에 개발자들이 컨테이너 API에 종속될 필요가 없다.

[2] DI (DEPENDENCY INJECTION)

- <1>Constructor Injection

  : 생성자를 이용하여 클래스 사이의 의존관계를 연결.

  # constructor-arg 요소 속성

- index : 생성자 몇 번째 인수에 값을 전달할 것인지 지정
- type : 생성자의 어떤 자료형 인수에 값을 전달할 것인지 지정
- ref : 주입할 참조 객체를 지정
- value : 주입할 값을 지정

[예제]

```
<bean id="messageBean" name="mb"
   class="my.com.MessageBeanImpl">
   <constructor-arg>
      <value>스프링</value>
   </constructor-arg>
</bean>
```

MessageBean mb
=new MessageBeanImple(“스프링”)’;

[2] DI (DEPENDENCY INJECTION)

- <2>Setter Injection

  .. setter메소드를 이용하여 클래스 간의 의존관계를 연결하는 방법

  # property 요소의 속성

  - ref : 주입할 참조 객체를 지정
  - value: 주입할 값을 지정

[예제]

```
<bean id="messageBean" name="mb"
   class="my.com.MessageBeanImpl">
<property name="greeting" value="안녕~~"/>
<property name="outputter">
    <ref local="outputter"/>
</property>
</bean>
<bean id="outputter"
   class="my.com.ConsoleOutputter">
```

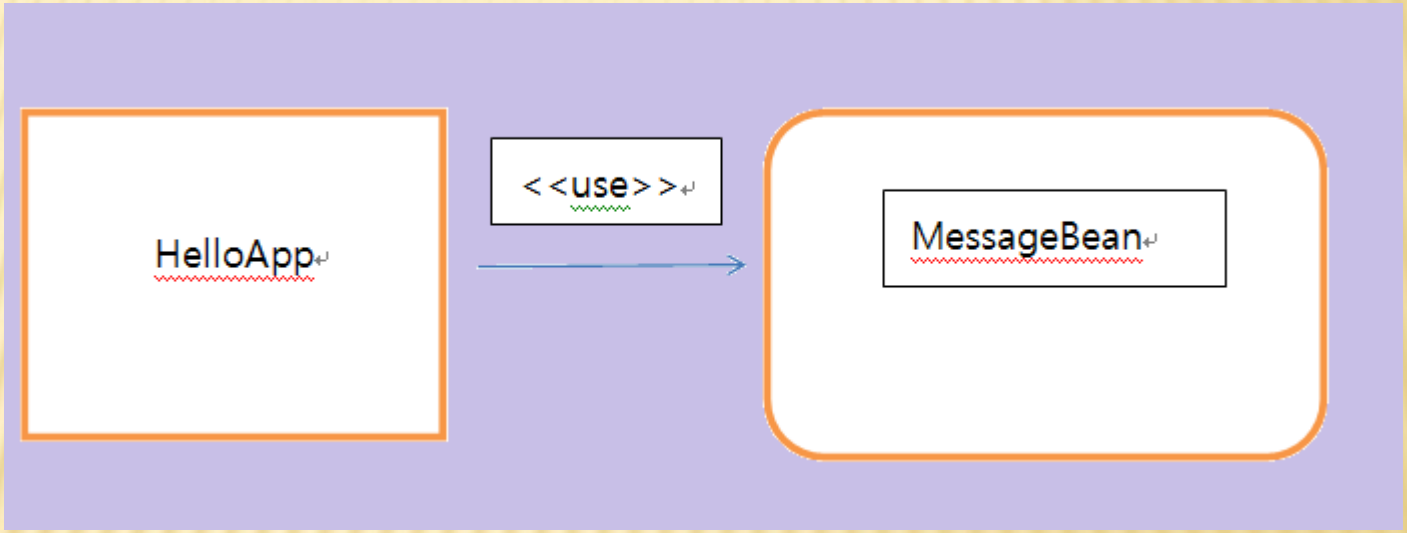
```
mb.setGreeting("안녕");

Outputter outputter=
        ConsoleOutputter();

mb.setOutputter(outputter);
```

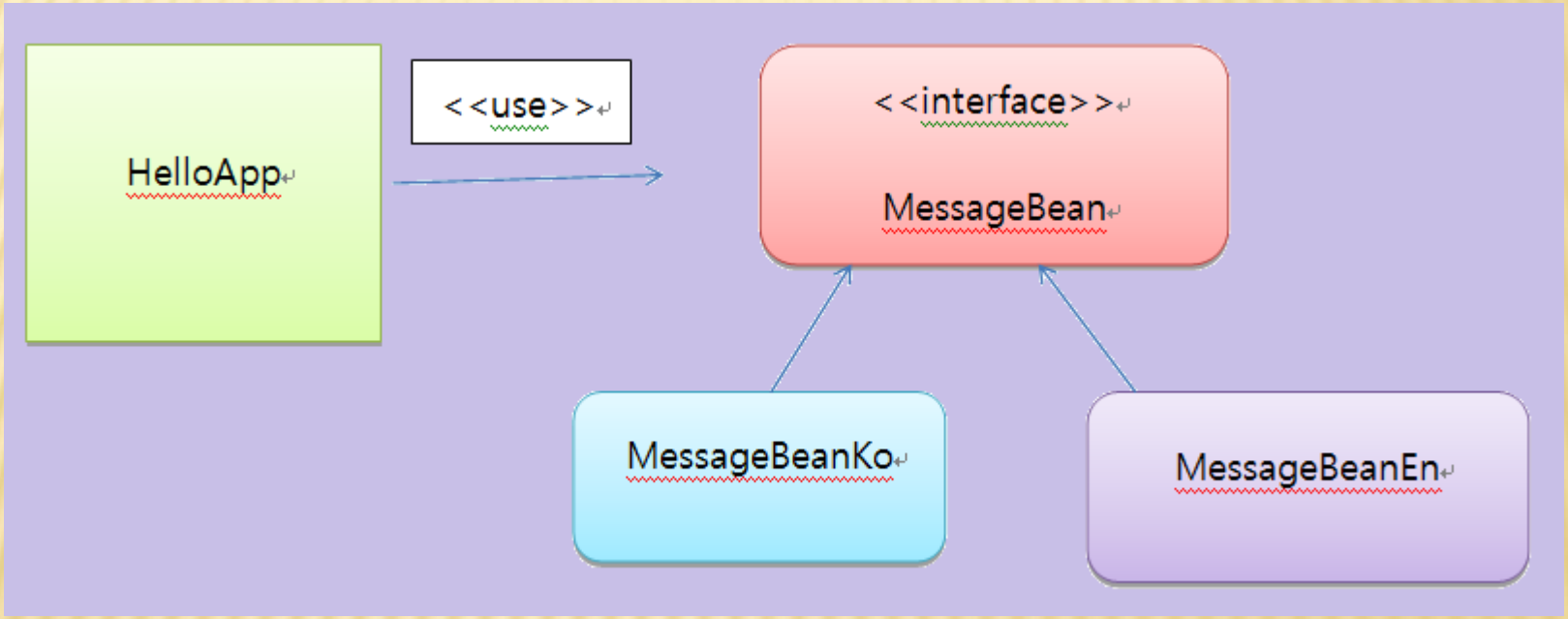
[실습] 스프링의 결합도

- [1] 첫번째 예제 : Object 결합이 엄격함



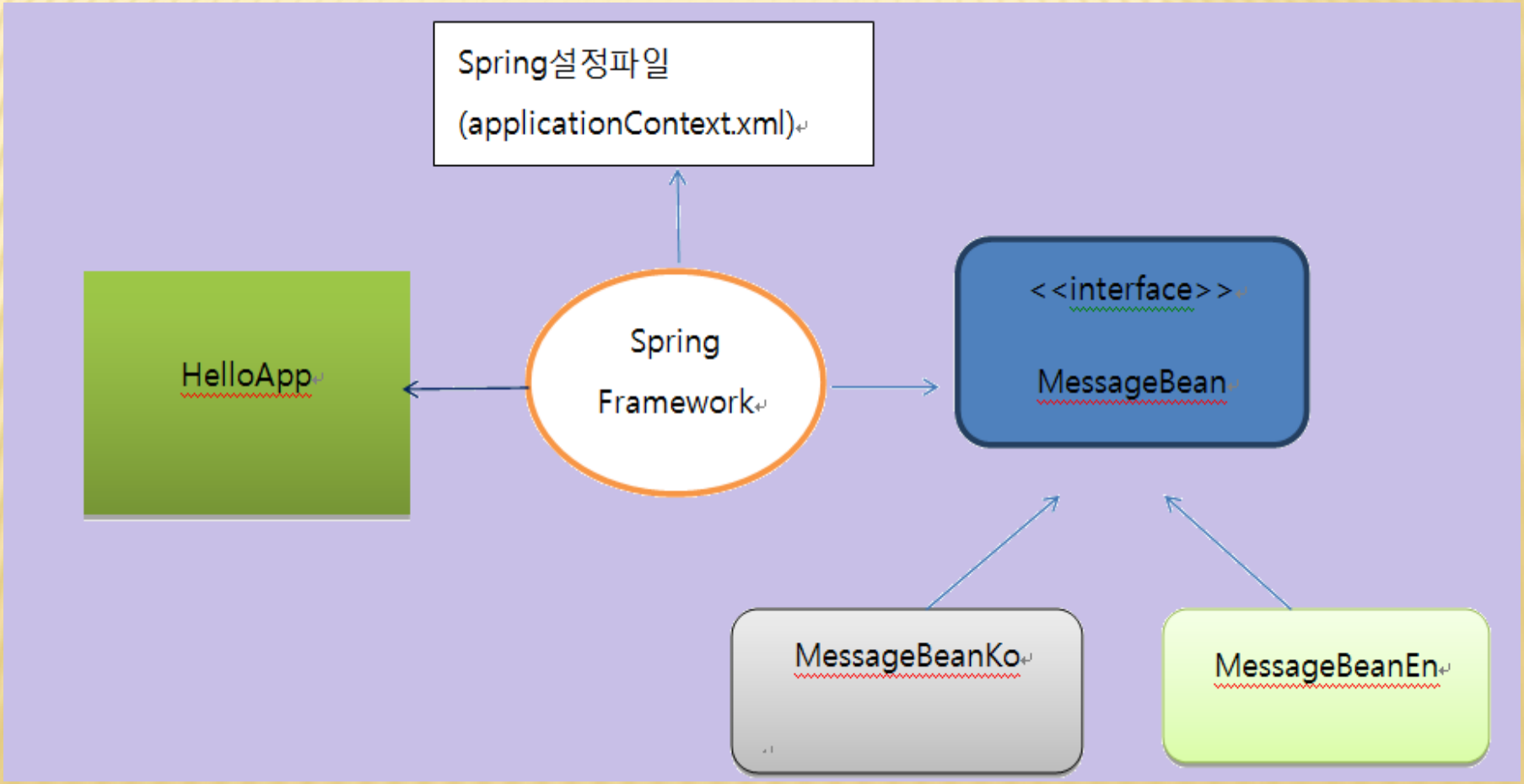
[실습] 스프링의 결합도

- [2] 두 번째 예제: 결합도를 다소 완화 시킴



[실습] 스프링의 결합도

- [3] 세번째 예제: Object결합이 느슨함



스프링의 결합도

- 결합도가 높은 코드의 경우, 애플리케이션이 복잡해지면 단점이 나타난다.
- 계층 사이에 결합이 강하기 때문에 한 쪽을 수정하면 연관된 코드도 다 함께 수정해야 한다.
- 코드의 중복은 많아지고 결합이 강해 그만큼 테스트 하기도 힘들다.
- 따라서 의존관계가 약한 클래스들을 작성하기 위해 스프링은 인터페이스를 설계하여 클래스 사이의 의존관계를 약하게 설정한다.

[실습 예제] DI실습 예제

- SpringApp_02DI프로젝트
- 1. 생성자 DI : ex01패키지
- 2. setter DI : ex02패키지