1. 어노테이션 기반 DI

Spring5 에서 어노테이션 기반으로 DI를 하는 방법은 3가지이다. (Constructor Injection, Setter Injection, Field Injection)
XML 기반인 경우에는 Constructor Injection, Setter Injection 만 가능하다.(Bean 요소의 constructor-arg, property 태그를 통해)

2. DI 어떤것을 사용할까?

Field Injection은 의존성 주입이 쉽기 때문에 의존관계가 복잡해질 우려가 있으며, 의존성이 숨어있기에, 즉 생성자, 세터주입과 다르게 의존 관계가 보이지 않아 검증이 어렵다. 또한 필요한 의존성을 가진 클래스를 곧바로 인스턴스화 시킬 수 없다. 그러므로 테스트 코드를 작성할 때 불편하다는 단점이 있다.
Field Injection은 final을 선언할 수 없다. 그래서 객체가 변할 수 있다.
Setter Inject역시 객체를 주입받은 이후 객체가 변경될 우려가 있다.
Constructor Injection과 달리 Setter Injection에서는 객체가 먼저 생성 된 다음 종속성이 주입된다.

```
@Controller
public class EmpController {
  @Autowired  //필드주입
  private EmpService empService;
  …
}
```

```
@Controller
public class EmpController {
  private EmpService empService;

  @Autowired  //세터주입
  public void) {
    this.empService = empService; setEmpService(EmpService empService
  }
  …
}
```

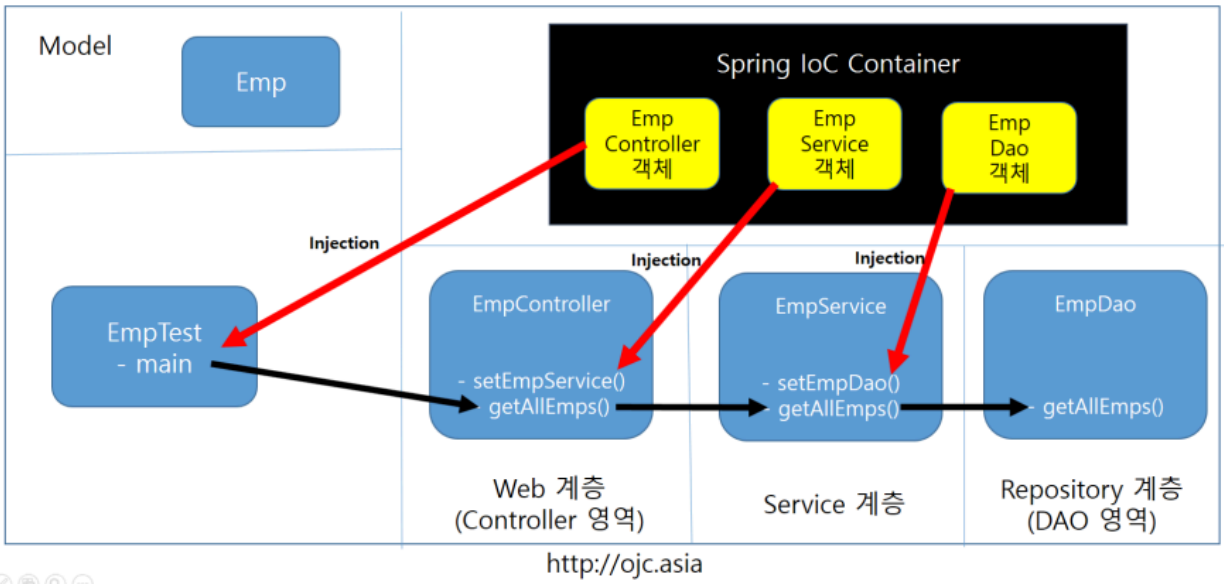
스프링 3.x 다큐멘테이션에서는 Setter Injection을 추천, 4.x에서는 더이상 Setter Injection이 아닌 Constructor Injection을 권장한다
Construnctor Inject을 통해 스프링 4.3버전부터는 클래스를 완벽하게 DI 프레임워크로부터 분리할 수 있다. 생성자가 하나인 경우 @Autowired를 붙이지 않아도 된다. 또한 final을 선언할 수 있으므로 객체가 변하지 않도록 할 수 있다. 또한 생성되는 시점에 객체를 주입받으므로 객체들의 주입 순서가 정해져 있어서 순환 의존성도 문제도 알 수 있다. 잘못 설계된 패턴인지 확인할 수 있다.
필수적인 프로퍼티의 경우 Constructor Inject을 사용하고 선택적 프로퍼티의 경우 Setter Injection을 사용한다. 대체로 생성자 주입을 사용하라. 그러나 생성자에 파라미터가 많아지고 복잡해지면 결국 리팩토링 대상이므로 세터 주입도 고려할 만 하다.(생성될 때 받아서 생성 할것도 많은데 객체 주입까지 받으면 복잡해 질 수있다)

```
@Controller
public class EmpController {
  private EmpService empService;
  @Autowired   //생성자주입
  public void EmpController(EmpService empService)
  {
    this.empService = empService;
  }
  …
}
```

3. 세터주입 이란?

Spring5 에서 어노테이션 기반으로 DI를 하는 방법은 3가지이다. (Constructor Injection, Setter Injection, Field Injection)
Spring Setter Injection이란? 객체의 Setter 메소드를 사용하여 Bean 종속성을 주입(필요로 하는 객체를 주입)하는 것입니다.
Spring Constructor Injection과 달리 Setter Injection에서는 객체가 먼저 생성 된 다음 종속성이 주입됩니다.

실습 (어노테이션 기반 세터주입)



스프링프레임워크, 세터주입, Spring Injection, 스프링DI

---------- pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>setterinjection</groupId>

<artifactId>setterinjection</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>setterinjection</name>

<description>setterinjection</description>

<properties>

<java-version>1.8</java-version>

<org.springframework-version>5.2.9.RELEASE</org.springframework-version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework-version}</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

```
</project>
```

---------- MODEL

```
package setterinjection;

/*
 * Model
 */
public class Emp {
private String empno;
private String ename;
public Emp(String empno, String ename) {
super();
this.empno = empno;
this.ename = ename;
}
@Override
public String toString() {
return "Emp [empno=" + empno + ", ename=" + ename + "]";
}


public String getEmpno() {
return empno;
}
public void setEmpno(String empno) {
this.empno = empno;
}
public String getEname() {
return ename;
}
public void setEname(String ename) {
this.ename = ename;
}
}
```

---------- DAO

```
package setterinjection;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Repository;

@Repository
public class EmpDao {
public List<Emp> getAllEmps() {
List<Emp> emps = new ArrayList<Emp>();
emps.add(new Emp("1", "이종철"));
emps.add(new Emp("2", "오라클자바커뮤니티"));
return emps;
}
}
```

---------- SERVICE

```
package setterinjection;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class EmpService {
private EmpDao empDao;
@Autowired
public void setEmpDao(EmpDao empDao) {
this.empDao = empDao;
}
public List<Emp> getAllEmps() {
return empDao.getAllEmps();
}
}
```

---------- Controller

```
package setterinjection;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

@Controller
public class EmpController {
private EmpService empService;
@Autowired
public void setEmpService(EmpService empService) {
this.empService = empService;
}
public void getAllEmps() {
List<Emp> emps = empService.getAllEmps();
emps.forEach( emp -> System.out.println(emp) );  //jdk 1.8이상
}
}
```

---------- TEST MAIN

```
package setterinjection;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class EmpTest {
public static void main(String[] args) {
ApplicationContext factory = new ClassPathXmlApplicationContext("classpath:emp.xml");
EmpController empController = factory.getBean(EmpController.class);
empController.getAllEmps();
}
}
```