의존성 주입(Dependency Injection)과 IOC 컨테이너

# ▣ 프로젝트명 : B03DI

## ◈ 준비사항

* 의존설정 : Spring Web
* JSP 사용을 위한 설정을 한다.
* Refresh Gradle Project 를 눌러 적용한다.

## ◈ DI란..??

* 스프링의 3대 요소 중 하나
* IOC(Inversion Of Control : 역제어 혹은 제어의 역전)의 한 형태
  + Java에서는 객체가 필요한 경우 개발자가 new를 통해 직접 생성
  + 하지만 스프링에서는 IoC 컨테이너가 미리 Bean(빈)을 생성
  + 즉 개발자가 아닌 IoC 컨테이너가 제어 흐름에 대한 권한을 가지고 있으므로 제어의 역전이라 한다.
* '의존성 주입' 또는 '종속개체 주입'이라고 한다.
* 일반적인 객체 생성
  + new 연산자를 이용해서 생성한다.
  + 이 경우 두 개체간에 결합도가 높아지고 독립성이 떨어지게 된다.
* 의존성 주입
  + IoC 컨테이너가 미리 생성해둔 객체를 외부에서 주입받아 사용한다.
  + 결합도가 낮아지고 독립성이 높아지게 된다.

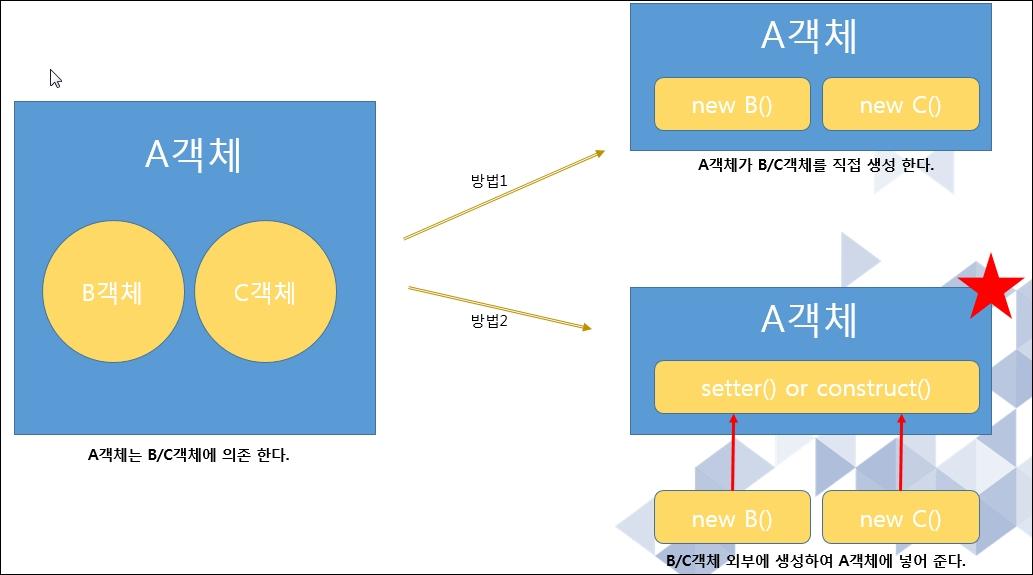
## ◈ 의존성이란..?? 결합도란..?? 독립성이란..??



* 위 2개의 장난감은 배터리가 있어야 사용할 수 있다. 즉 배터리에 의존하고 있다.
* 배터리 일체형
  + 배터리가 소모되면 새롭게 구매해야한다.
  + 장난감과 배터리는 결합도가 높다.
  + 비효율적이다.
* 배터리 분리형
  + 배터리가 소모되면 배터리만 새것으로 교체하면 된다.
  + 장난감과 배터리를 독립성이 높다.
  + 효율적이고 유연하다.

## ◈ DI(Dependency Injection)의 목적

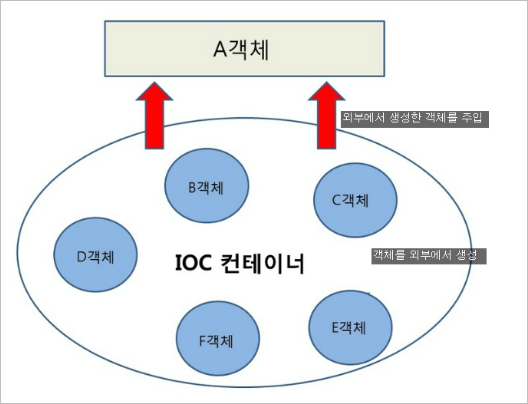
* new를 통해 필요한 객체를 생성하지 않고, 스프링 컨테이너와 같은 외부에서 주입을 받아 객체간의 낮은 결합도를 유지하는게 목적이다.
* 개발자가 직접 객체를 생성하지 않으므로 IOC(제어의 역전)이라고 표현한다.



A 객체가 B와 C객체에 의존하고 있다고 했을때

* 방법1은 일반적인 객체생성을 보여주고 있다.
* 방법2는 DI를 보여주고 있다.

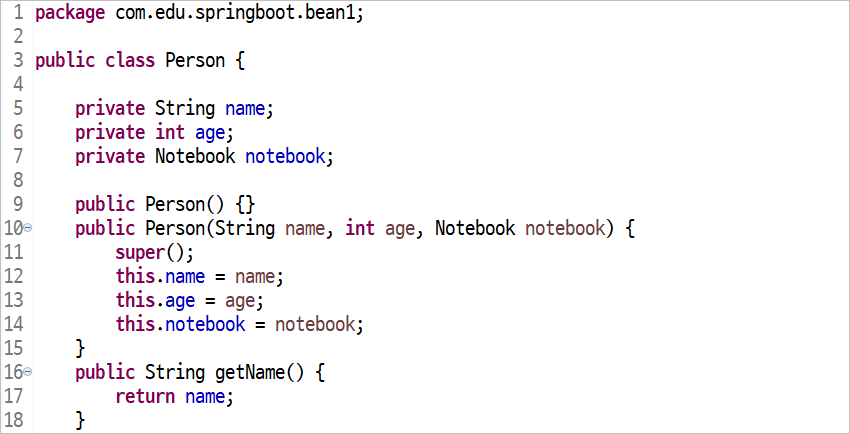
## ◈ 그렇다면 DI에서 말하는 외부란 어디인가??



위 그림에서 보듯이 IoC컨테이너 즉 스프링 컨테이너를 말한다.

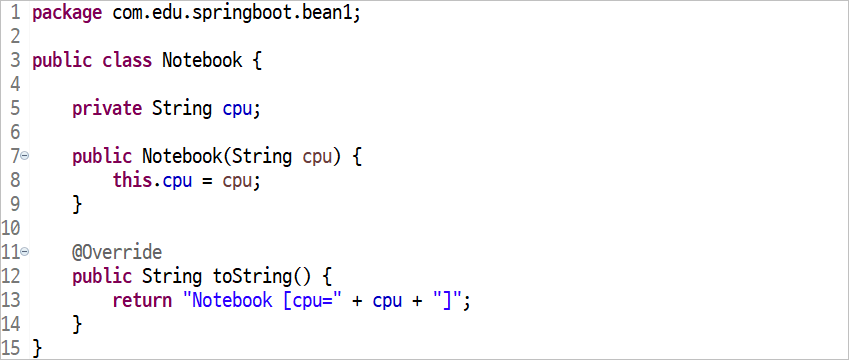
# ▣ 의존주입1 - Java클래스를 통한 빈 생성VO객체 : com.edu.springboot.bean1.Person.java

기본생성자, 인수생성자, getter, setter, toString() 메서드를 메뉴를 통해 생성하세요.

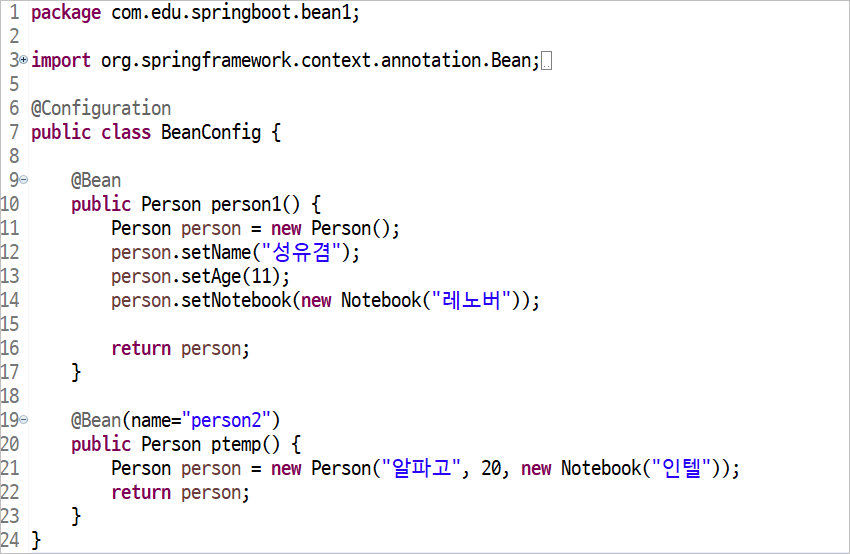


### VO객체 : com.edu.springboot.bean1.Notebook.java

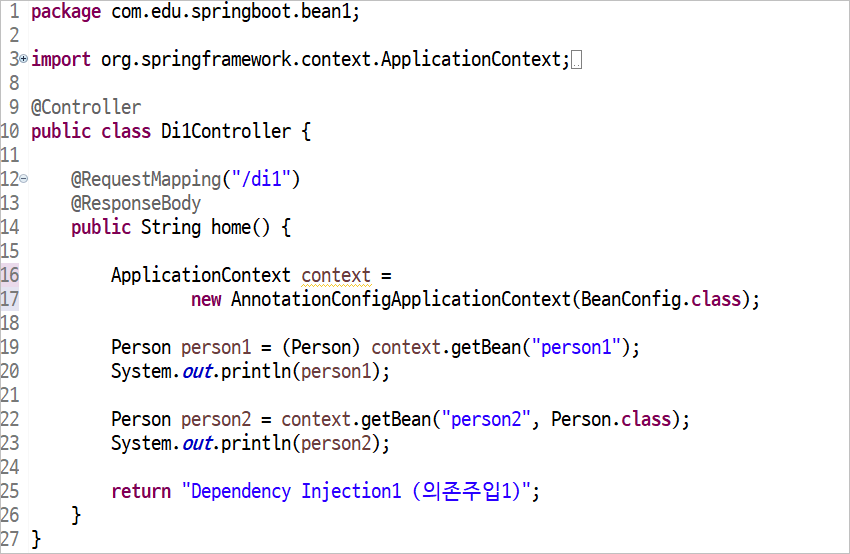
인수생성자, toString() 메서드를 메뉴를 통해 생성하세요.



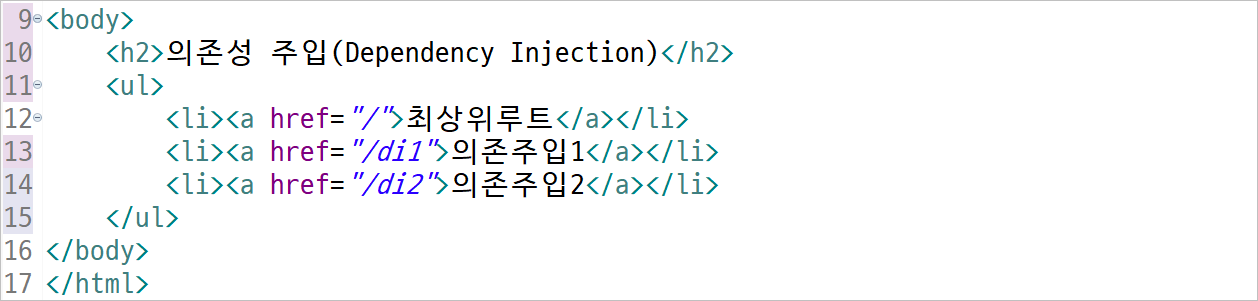
### 설정파일 : com.edu.springboot.bean1.BeanConfig.java



### 컨트롤러 : com.edu.springboot.bean1.Di1Controller.java



### View : /B04DI/src/main/webapp/WEB-INF/views/home.jsp

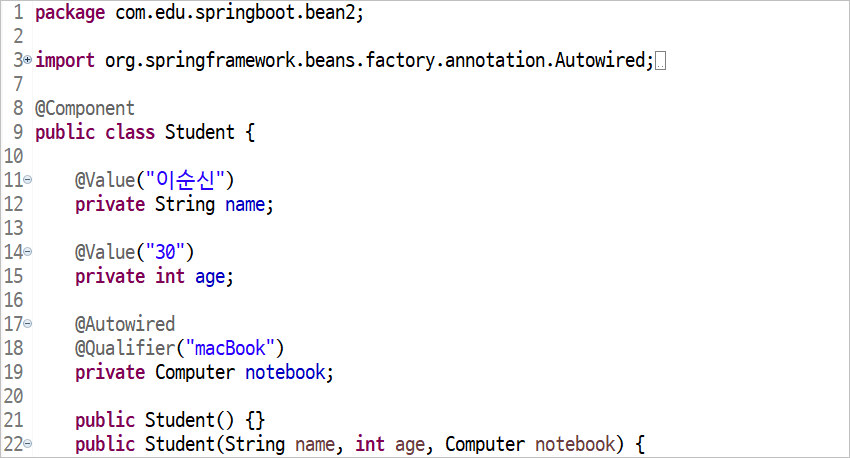


여기까지 작성하세요

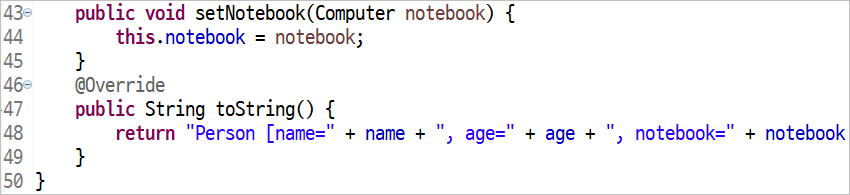
# ▣ 의존주입2 - 어노테이션을 통해 빈 생성

### VO객체 : com.edu.springboot.bean2.Student.java

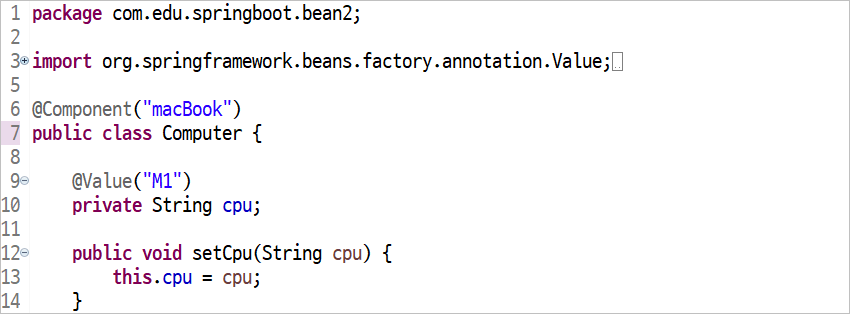
기본생성자, 인수생성자, getter, setter, toString() 메서드를 메뉴를 통해 생성하세요.

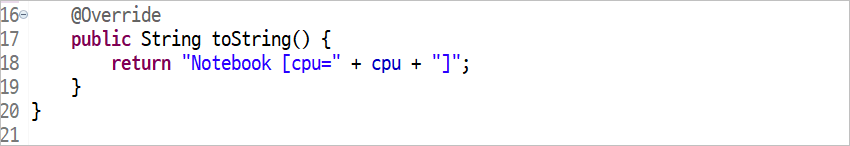


중간생략..



### VO객체 : com.edu.springboot.bean2.Computer.java





### 컨트롤러 : com.edu.springboot.bean2.Di2Controller.java

