* Spring Framework

✓ IoC Container = $\underline{\text{Bean}}$ Container
$$\underline{\text{Bean}} = \text{object} = \text{instance}$$

✓ WebMVC

AOP

⋮

* IoC Container

Inversion of Control → 제어의 역행

① 객체 생성 → 컨테이너가 객체를 생성해 줌.

↓

의존 객체 생성 → 컨테이너가 생성하여 주입해 줌.

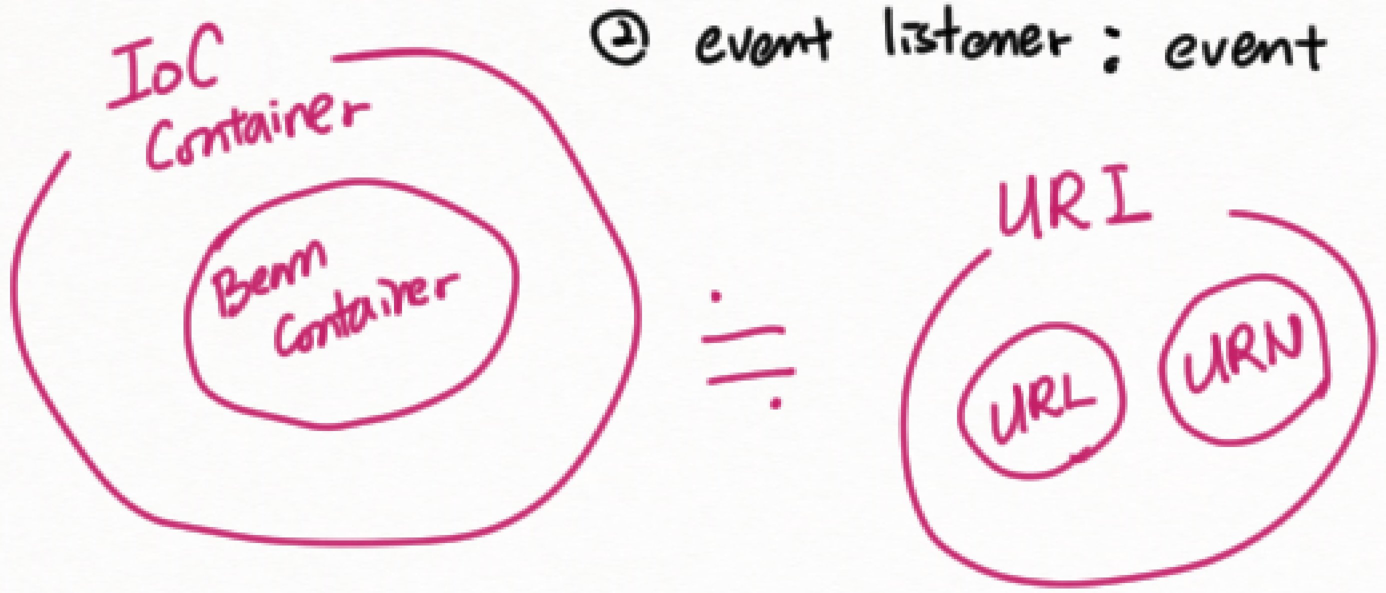
↳ Dependency Injection (DI)

⇒ Bean Container || DI Container

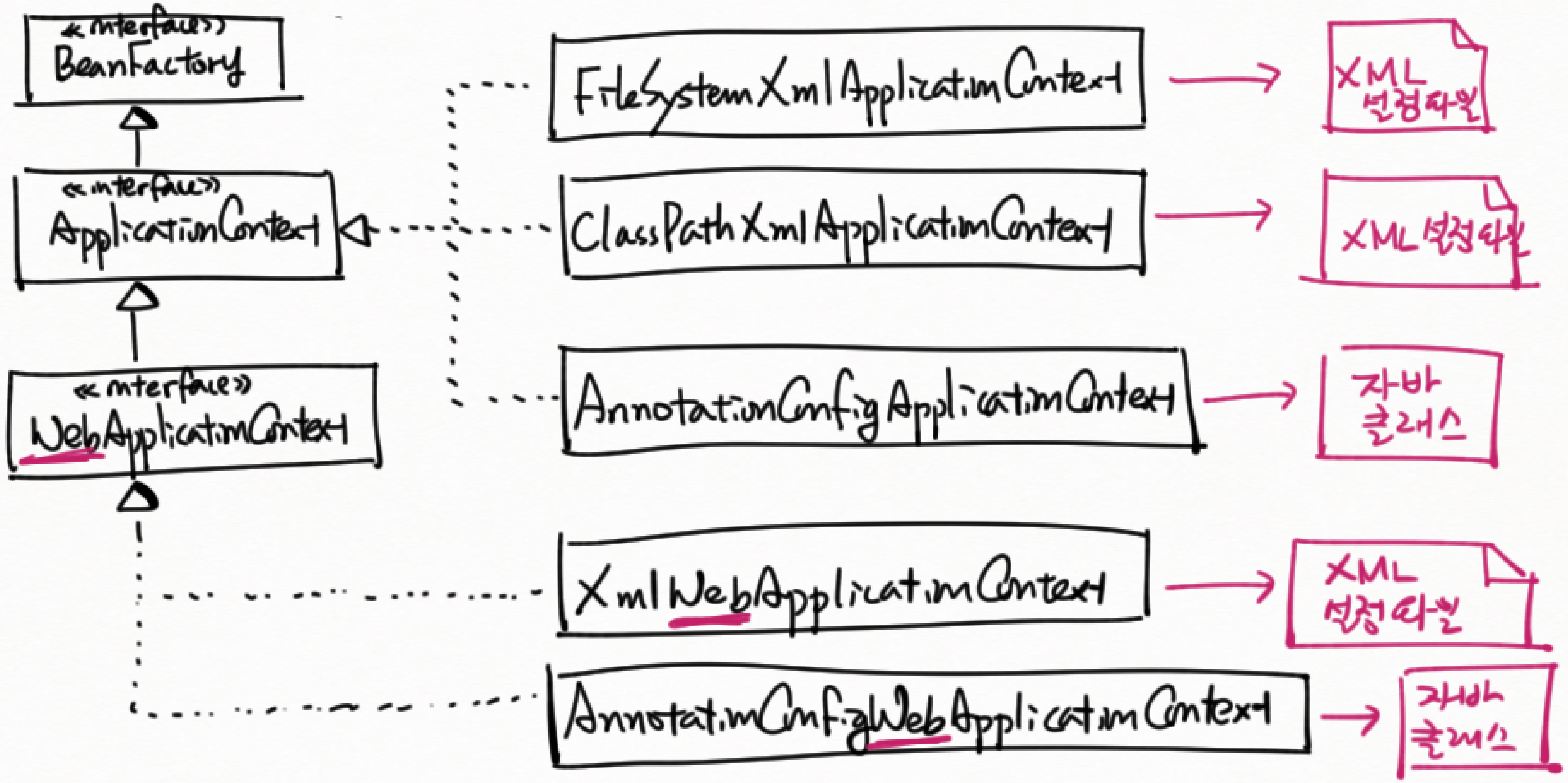
② event listener : event 발생 → ~~메서드를 호출~~

→ 개발자가 정의한 메서드가 호출 당함!

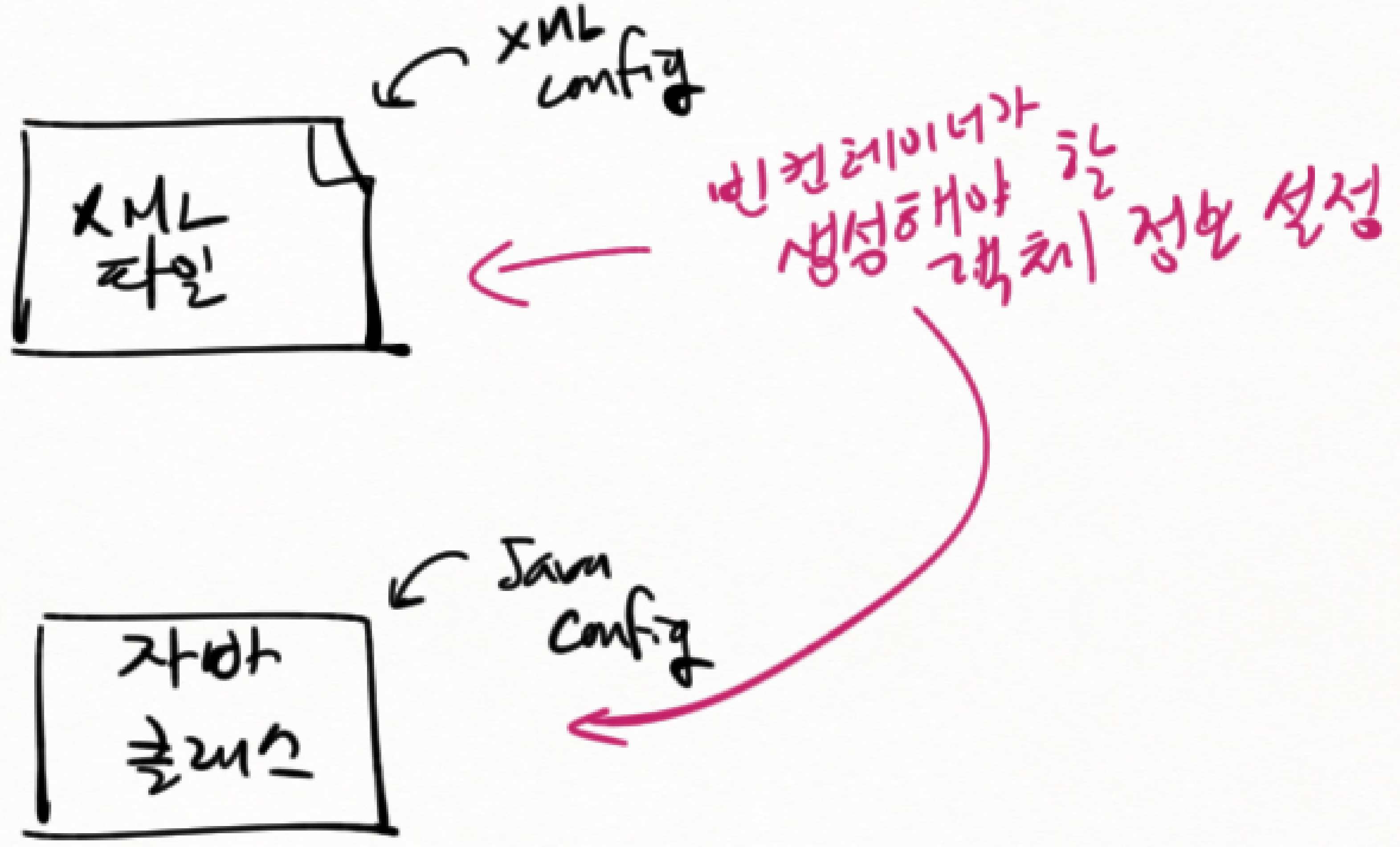
→ 순차적인 실행 흐름에서 벗어나서 특정 상황에서 임의적으로 호출됨.



* Spring IoC 컨테이너



* IoC 컨테이너 설정



* class path or filesystem path

com/eomcs/spring/ioc/ex01/a/application-context.xml

패키지명

file://c:/tools/a/application-context.xml

파일경로 (Window)

file:///Users/study/a/application-context.xml

파일경로 (Unix)

* <bean> 태그 와 @Bean

```
<bean id="c1" name="aaa" class="com.eomcs.spring.ioc.ex01.Car"/>
```

c1: 빈 컨테이너에 객체를 저장할 때 사용할 이름

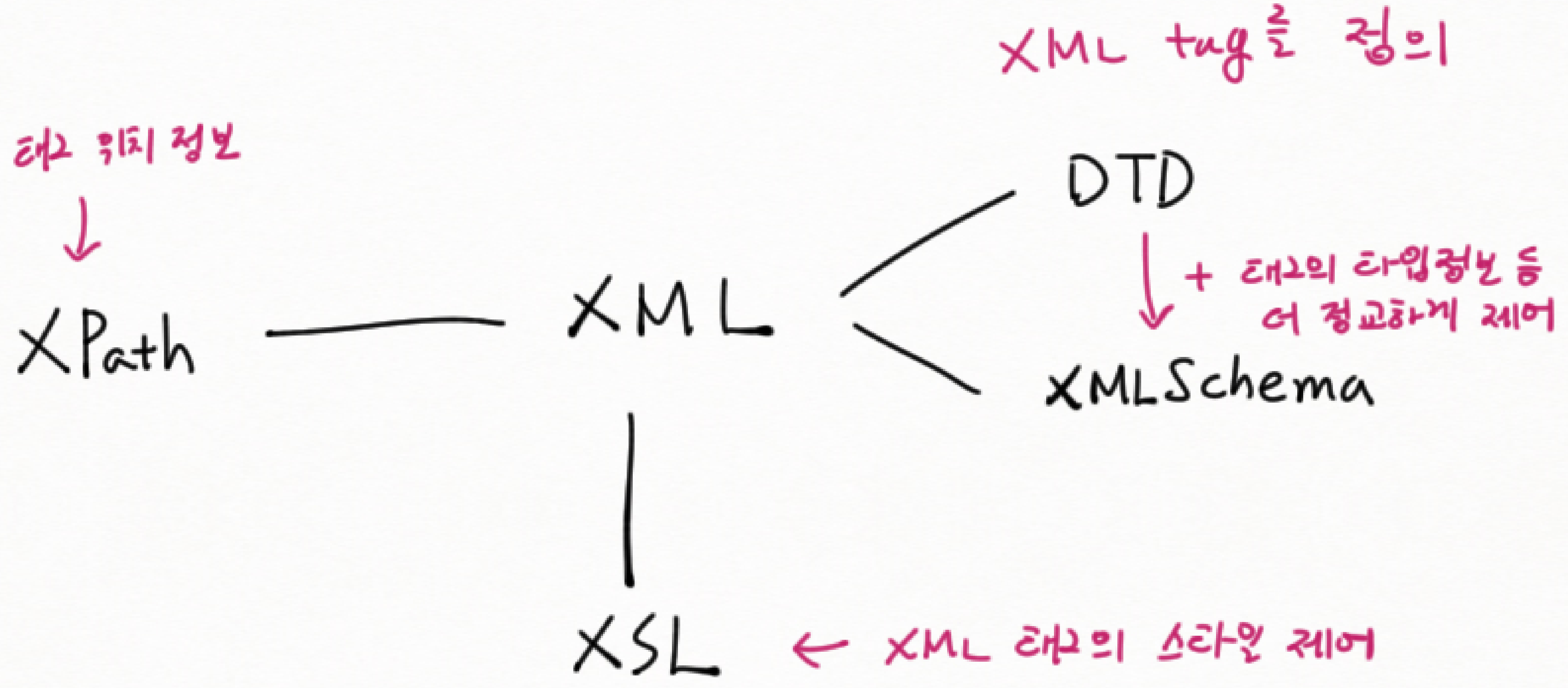
aaa: 객체의 별명

com.eomcs.spring.ioc.ex01.Car: FQName = QName = 패키지명을 포함한 클래스명

```
@Bean Car c1() { return new Car(); }
```

@Bean: 빈 컨테이너에게 이 메서드를 호출하여 리턴값을 받아서 보관하라고 명령

c1: 객체를 저장할 때 사용할 이름


XML tag를 정의
태그 위치 정보
XPath
XML
DTD
+ 태그의 타입정보 등
더 정교하게 제어
XMLSchema
XSL
XML 태그의 스타일 제어

* Spring 기본 property editor

```
<bean id="c1"
      class="———.Car">
  <property name="model" value="—"/>
  <property name="cc"    value="—"/>
  <property name="auto"  value="—"/>
  <property name="createdDate" value="2021-6-1"/>
</bean>
```

String → String model;
String → int cc;
String → boolean auto;
String ✗→ java.sql.Date createdDate;

```
class Car {
  String model;
  int cc;
  boolean auto;
  java.sql.Date createdDate;
  :
}
```

* Spring IoC 컨테이너는 String 값을 primitive type 으로는 자동 변환해 준다!

별도로 변환기를 장착해야 한다 ← 다른 타입