Spring Bean

Spring Bean 개요

- 컨테이너(Container)
  - 애플리케이션 코드를 작성할 때, 특정 기능이 필요하면 라이브러리를 사용한다. 이때는 프로그램의 흐름을 제어하는 주체가 애플리케이션 코드이다.
  - 하지만, 프레임워크 기반의 개발에서는 프레임워크 자신이 흐름을 제어하는 주체가 되어, 필요할 때마다 애플리케이션 코드를 호출하여 사용한다.
  - 프레임워크에서 이 제어권을 가지는 것이 바로 **컨테이너**이다.

- IoC(Inversion of Control : 제어의 역전)
  - 컨테이너는 객체에 대한 생명주기 관리 등의 제어권을 개발자(애플리케이션 코드)대신 수행한다.
  - 이것을 일반적으로 제어권의 흐름이 바뀌었다고 하여 제어의 역전(IoC)이라고 한다.

- DI(Dependency Injection : 의존성 주입)
  - ✓ 자료조사 필요...

- Spring Bean
  - Spring Bean이란 자바 객체이다.
  - Spring IoC 컨테이너에 의해서 자바 객체가 만들어지면 해당 자바 객체는 Spring Bean이 되는 것이다.
  - Spring IoC 컨테이너에 의해 인스턴스화, 관리, 생성된다.
  - 스프링 공식 레퍼런스 메뉴얼 발췌

> In spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans.
> 스프링 IoC(Inversion of Control) 컨테이너에 의해서 관리되고 애플리케이션의 핵심을 이루는 객체들을 스프링에서는 빈즈(Beans)라고 부른다.
>
> A bean is an object that is instantiated, assembled, and otherwise managed by a Spring Ioc container.
> 빈은 스프링 Ioc 컨테이너에 의해서 인스턴스화되어 조립되거나 관리되는 객체를 말합니다.
>
> Otherwise, a bean is simply one of many objects in your application.
> 이같은 점을 제외하고 빈은 수많은 객체들중의 하나일 뿐입니다.
>
> Beans, and the dependencies among them, are reflected in the configuration metadata used by a container.
> 빈과 빈 사이의 의존성은 컨테이너가 사용하는 메타데이터 환경설정에 반영됩니다.

Spring Bean 정의

➢ 일반적으로 Spring Bean은 XML 파일에 정의된다.

➢ 주요속성

- class (필수) : 정규화된 자바 클래스 이름
- id : bean의 고유 식별자
- scope : 객체의 범위 (sigletone, prototype)
- constructor-arg : 생성 시 생성자에 전달할 인수
- property : 생성 시 bean setter에 전달할 인수
- init method와 destroy method

- Ex) Spring Bean 정의

```
<!-- A simple bean definition -->
<bean id="..." class="..."></bean>

<!-- A bean definition with scope-->
<bean id="..." class="..." scope="singleton"></bean>

<!-- A bean definition with property -->
<bean id="..." class="...">
    <property name="message" value="Hello World!"/>
</bean>

<!-- A bean definition with initialization method -->
<bean id="..." class="..." init-method="..."></bean>
```

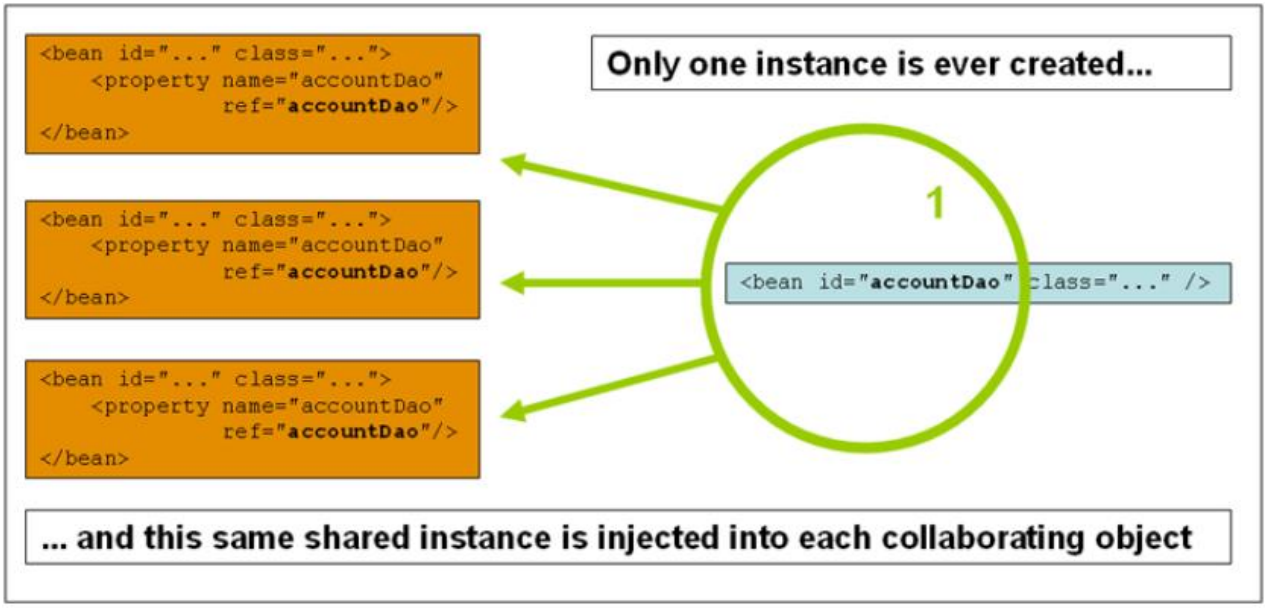
- Spring Bean Scope
  - 스프링은 default로 모든 bean을 **singleton**으로 생성하여 관리한다.
  - request, session, global session의 Scope는 Spring MVC Web Application에서만 사용된다.

Scope	Description
singleton	하나의 Bean 정의에 대해서 Spring IoC Container 내에 단 하나의 객체만 존재한다.
prototype	하나의 Bean 정의에 대해서 다수의 객체가 존재할 수 있다.
request	하나의 Bean 정의에 대해서 하나의 HTTP request의 생명주기 안에 단 하나의 객체만 존재한다; 즉, 각각의 HTTP request는 자신만의 객체를 가진다. Web-aware Spring ApplicationContext 안에서만 유효하다.
session	하나의 Bean 정의에 대해서 하나의 HTTP Session의 생명주기 안에 단 하나의 객체만 존재한다. Web-aware Spring ApplicationContext 안에서만 유효하다.
global session	하나의 Bean 정의에 대해서 하나의 global HTTP Session의 생명주기 안에 단 하나의 객체만 존재한다. 일반적으로 portlet context 안에서 유효하다. Web-aware Spring ApplicationContext 안에서만 유효하다.

➢ Bean Scope : Singleton

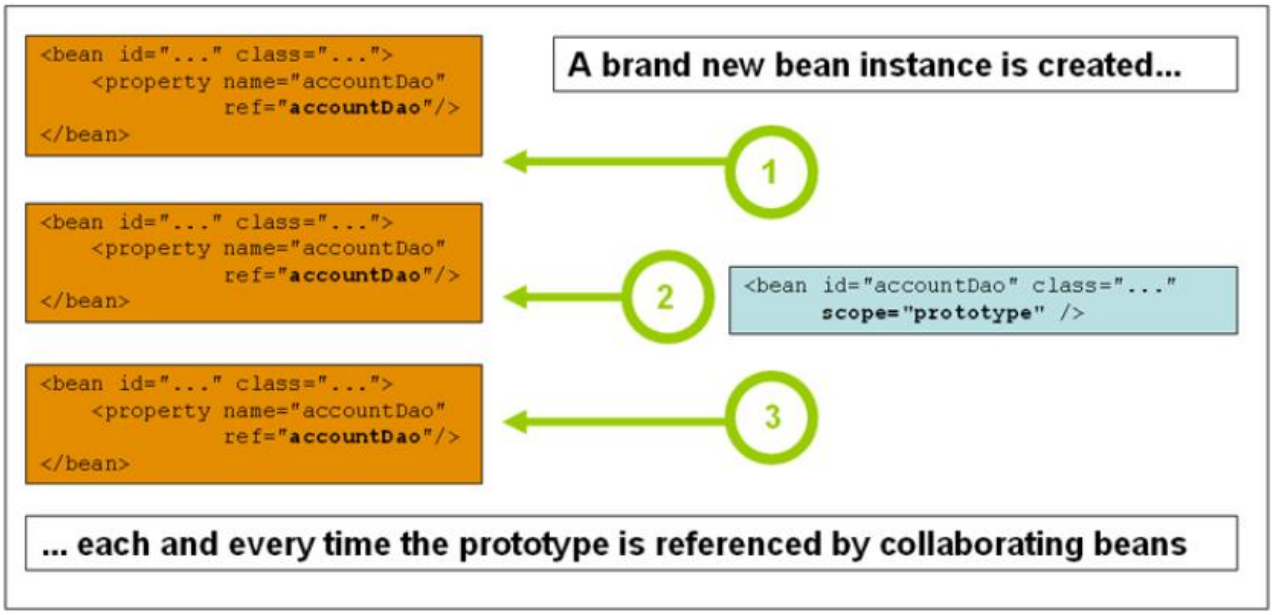
- **singleton bean**은 Spring 컨테이너에서 한 번만 생성된다.
- 생성된 하나의 인스턴스는 **single beans cache**에 저장되고, 해당 bean에 대한 요청과 참조가 있으면 캐시된 객체를 반환한다. (하나만 생성되기 때문에 동일한 곳을 참조)

- xml 설정
  : `<bean id="..." class="..." scope="singleton"></bean>`
- annotation 설정
  : 대상 클래스에 `@Scope("singletone")`
- singleton bean 참조 구조



➢ Bean Scope : Prototype

- **prototype bean**은 모든 요청에서 새로운 객체를 생성한다.
- 의존성 관계의 bean에 주입될 때 새로운 객체가 생성되어 주입된다.
- 정상적인 방식으로 GC(Garbage Collection)에 의해 prototype bean은 제거된다.
- xml 설정
  : `<bean id="..." class="..." scope="prototype"></bean>`
- annotation 설정
  : 대상 클래스에 `@Scope("prototype")`
- prototype bean 참조 구조



예시) Spring Bean Scope Example

- MainApp Class

```
package com.spring;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
	public static void main(String[] args) {
		// main함수에서 Container를 생성
		// 컨테이너 생성자의 인자로 bean설정 파일을 넣는다.
		ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("com/spring/beans/bean.xml");

		// getBean()에 bean의 id를 활용하여 bean을 불러온다.
		PetOwner person1 = (PerOwner) context.getBean("petOwner");
		person1.setUserName("Alice");
		person1.getUserName();

		PetOwner person2 = (PerOwner) context.getBean("petOwner");
		person2.getUserName();

		context.close();
	}
}
```

- PetOwner Class

```
package com.spring;

public class PetOwner {
	String userName;
	public AnimalType animal;

	public PetOwner(AnimalType animal) { this.animal = animal; }

	public String getUserName() {
		System.out.println("Person name is " + , userName);
		return userName;
	}
	public void setUserName(String userName) { this.userName = userName; }

	public void play() { animal.sound(); }
}
```

- bean.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd">

    <bean id="dog" class="com.spring.Dog">
        <property name="myName" value="poodle"></property>
    </bean>

    <bean id="cat" class="com.spring.Cat">
        <property name="myName" value="bella"></property>
    </bean>

    <bean id="petOwner" class="com.spring.PetOwner" scope="singleton">
        <constructor-arg name="animal" ref="dog"></constructor-arg>
    </bean>
</beans>
```

- MainApp 실행 결과
  - ➢ petOwner bean의 scope가 singletone인 경우
    : Person name is Alice
    : Person name is Alice
  - ➢ petOwner bean의 scope가 prototye인 경우
    : Person name is Alice
    : Person name is null

참고문헌

- 김종민 (2018). https://jongmin92.github.io/2018/02/11/Spring/spring-ioc-di/
- HeeJeong Kwon (2018). https://gmlwjd9405.github.io/2018/11/10/spring-beans.html