

PART 2. 아주 작은 빈 이야기

Section 2~3 스프링 핵심 원리 이해

Covenant Ko

소개

- Name. Covenant.
- Company. 11번가
- Github 용감한친구들 Organization Founder & Maintainer. <https://github.com/brave-people>
- Github. <https://github.com/KoEonYack>
- Tech Blog. <https://covenant.tistory.com/> (누적 방문 41만)

What is BEAN?

In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application. Beans, and the dependencies among them, are reflected in the configuration metadata used by a container.

spring.io

1.1. Introduction to the Spring IoC Container and Beans

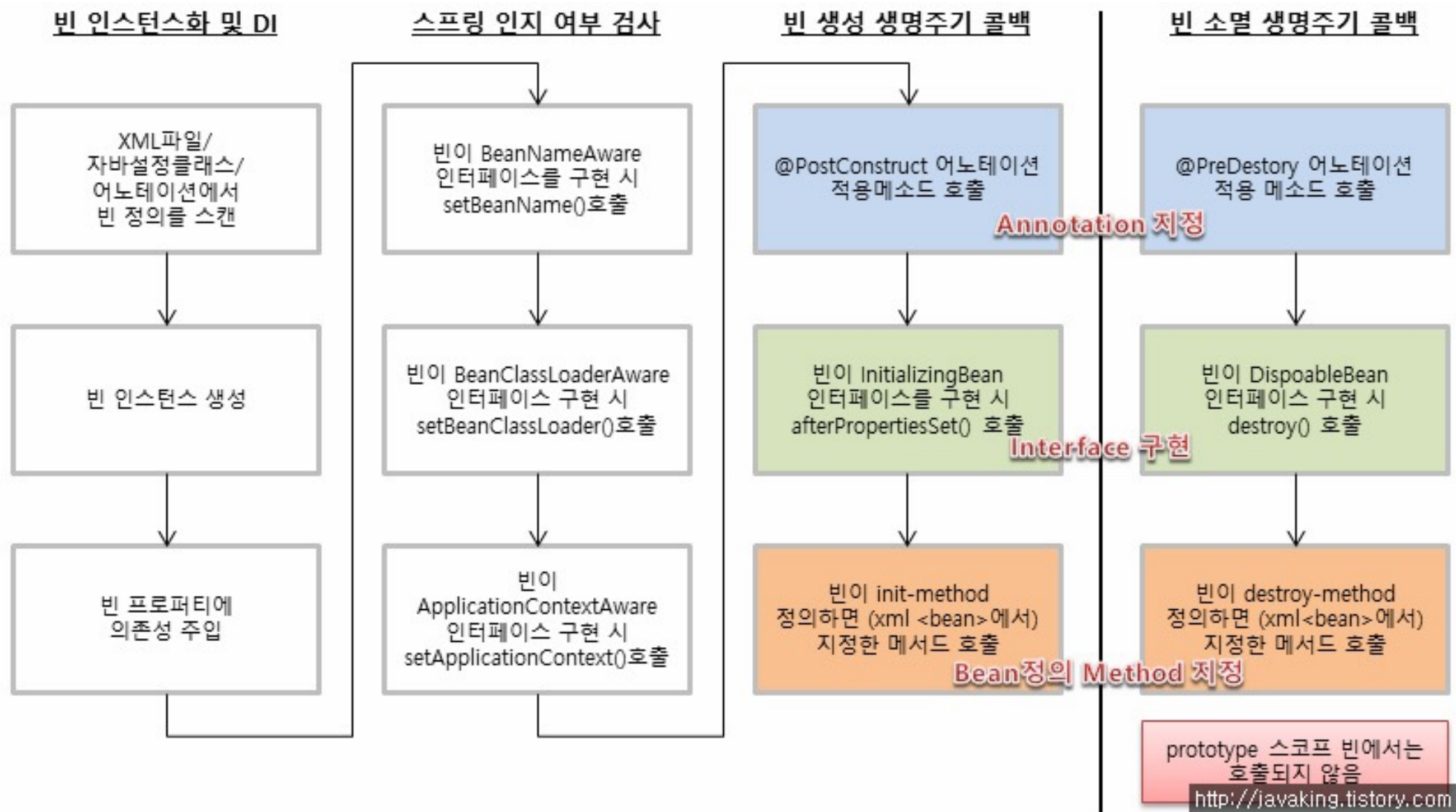
What is BEAN?

In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application. Beans, and the dependencies among them, are reflected in the configuration metadata used by a container.

spring.io

1.1. Introduction to the Spring IoC Container and Beans

Spring Bean Life Cycle





이 기능을 구현할 때 이 메서드를 쓰는게 가장 적합하다고 생각해? 대충하지 말고 가장 어울리는 메서드가 뭔지 고민해서 적용하라구!

난 개발자의 최고 덕목이 이해력이나 센스나 빨리 만드는 능력이 아니라 꼼꼼함이라고 생각해. 코딩을 할 때 가장 적합하고 잘 맞는 메서드와 기법을 사용하려고 애쓰란 말야.

넌 iOS SDK 코드 읽어봤어? 난 다 읽어봤어. 얼마나 아름다운지 알아?

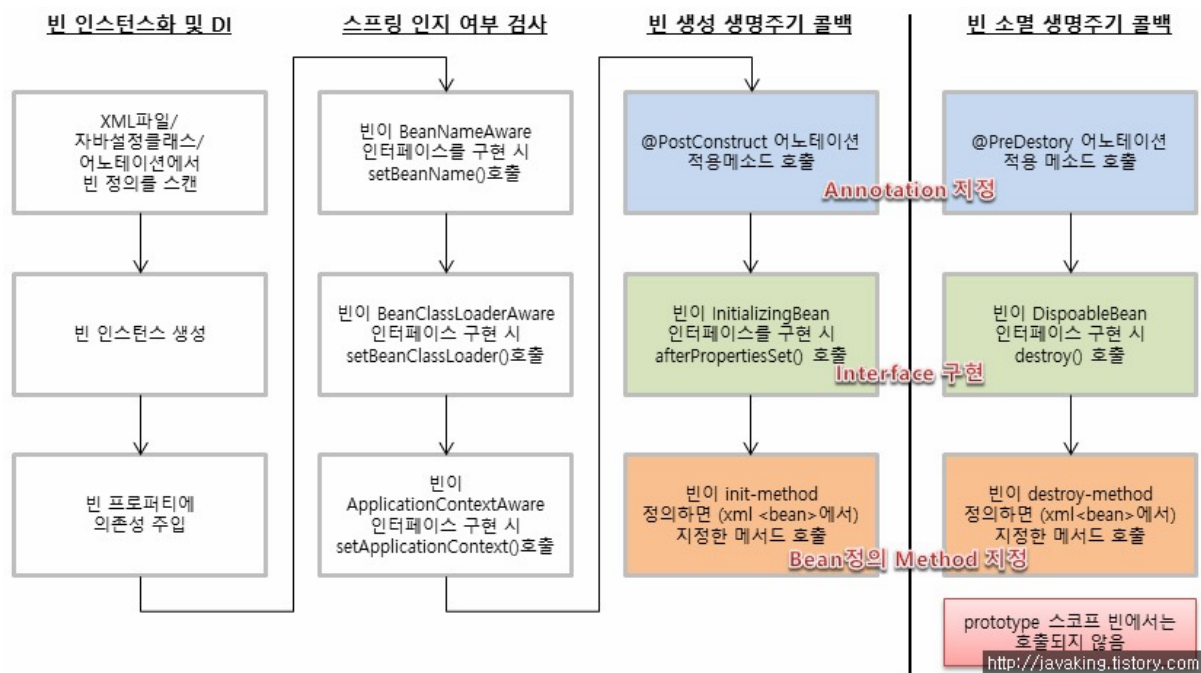
라이브러리나 프레임워크 등을 가져다 써보기전에 반드시 선행되어야하는 것은, 내가 다루어야 하는 것들에 대해 정확히 이해하는 것이다.

2010년의 카카오에서 있었던 일 1편

레거시

의외로 회사 와서 크게 배울 수 있었던 이유 중 하나가 바로 레거시 코드 때문이었다. 입사 전에는 레거시 코드를 짰 사람도 무능력한 사람이고, 그런 오래된 레거시를 가지고 개발하는 회사도 가면 안된다고 생각했다. 아무것도 모르는 상태에서 여기저기서 들은 이야기 때문에 그렇게 세뇌된 듯하다. 하지만 11번가에 오자마자 처음 마주했던 건 오래된 레거시 코드의 유지보수 티켓이었고 나는 코드를 보면서 많이 실망했다. 처음 보는 패키지 구조, 낡은 프레임워크, 낮은 자바 버전, 축약형 네이밍, 긴 메서드 길이, 테스트 코드 부재 등등, 테크캠프에서 이쁜 코드만 배우다가 말로만 듣던 레거시 코드를 접했는데 눈 앞이 깜깜해졌다. 그래도 업무를 진행하려고 레거시 프로젝트의 구조와 코드들을 천천히 살펴보았는데, 스프링 부트로 어노테이션 찍칠하면서 원리도 제대로 모르는 채 개발했던 때와는 달리, 처음 보는 이 프레임워크가 어떻게 동작하는지 원리를 알아야 업무를 할 수 있었고, 원리를 알아가는 과정에서 특정 기술들의 역사와 기본도 모르는 나를 발견할 수 있었다. 그리고 코드를 짜면서 느꼈다. 나도 결국 레거시에 레거시를 쌓아가고 있구나 라는 것을. 이걸 개발했을 당시에는 일정, 상황 등을 고려했을 때 이게 최선이었겠구나 라는 생각도 들었다.

Spring Bean Life Cycle



```

@Slf4j
public class BeanLifeCycle implements InitializingBean, DisposableBean {

    public BeanLifeCycle() {
        log.info("BeanLifeCycle 생성자 호출!");
    }

    @PostConstruct
    public void postConstruct() {
        log.info("postConstruct");
    }

    @Override
    public void afterPropertiesSet() throws Exception {
        log.info("afterPropertiesSet");
    }

    public void initMethod() {
        log.info("initMethod");
    }

    @Override
    public void destroy() throws Exception {
        log.info("destroy");
    }

    @PreDestroy
    public void preDestroy() {
        log.info("preDestroy");
    }

    public void destroyMethod() {
        log.info("destroyMethod");
    }
}
  
```


Bean Life Cycle Aware

AWARE INTERFACE	METHOD TO OVERRIDE	PURPOSE
<code>ApplicationContextAware</code>	<code>void setApplicationContext (ApplicationContext applicationContext) throws BeansException;</code>	Interface to be implemented by any object that wishes to be notified of the <code>ApplicationContext</code> that it runs in.
<code>ApplicationEventPublisherAware</code>	<code>void setApplicationEventPublisher (ApplicationEventPublisher applicationEventPublisher);</code>	Set the <code>ApplicationEventPublisher</code> that this object runs in.
<code>BeanClassLoaderAware</code>	<code>void setBeanClassLoader (ClassLoader classLoader);</code>	Callback that supplies the bean class loader to a bean instance.
<code>BeanFactoryAware</code>	<code>void setBeanFactory (BeanFactory beanFactory) throws BeansException;</code>	Callback that supplies the owning factory to a bean instance.
<code>BeanNameAware</code>	<code>void setBeanName (String name);</code>	Set the name of the bean in the bean factory that created this bean.
<code>BootstrapContextAware</code>	<code>void setBootstrapContext (BootstrapContext bootstrapContext);</code>	Set the <code>BootstrapContext</code> that this object runs in.
<code>LoadTimeWeaverAware</code>	<code>void setLoadTimeWeaver (LoadTimeWeaver loadTimeWeaver);</code>	Set the <code>LoadTimeWeaver</code> of this object's containing <code>ApplicationContext</code> .
<code>MessageSourceAware</code>	<code>void setMessageSource (MessageSource messageSource);</code>	Set the <code>MessageSource</code> that this object runs in.
<code>NotificationPublisherAware</code>	<code>void setNotificationPublisher (NotificationPublisher notificationPublisher);</code>	Set the <code>NotificationPublisher</code> instance for the current managed resource instance.
<code>PortletConfigAware</code>	<code>void setPortletConfig (PortletConfig portletConfig);</code>	Set the <code>PortletConfig</code> this object runs in.
<code>PortletContextAware</code>	<code>void setPortletContext (PortletContext portletContext);</code>	Set the <code>PortletContext</code> that this object runs in.
<code>ResourceLoaderAware</code>	<code>void setResourceLoader (ResourceLoader resourceLoader);</code>	Set the <code>ResourceLoader</code> that this object runs in.
<code>ServletConfigAware</code>	<code>void setServletConfig (ServletConfig servletConfig);</code>	Set the <code>ServletConfig</code> that this object runs in.
<code>ServletContextAware</code>	<code>void setServletContext (ServletContext servletContext);</code>	Set the <code>ServletContext</code> that this object runs in.

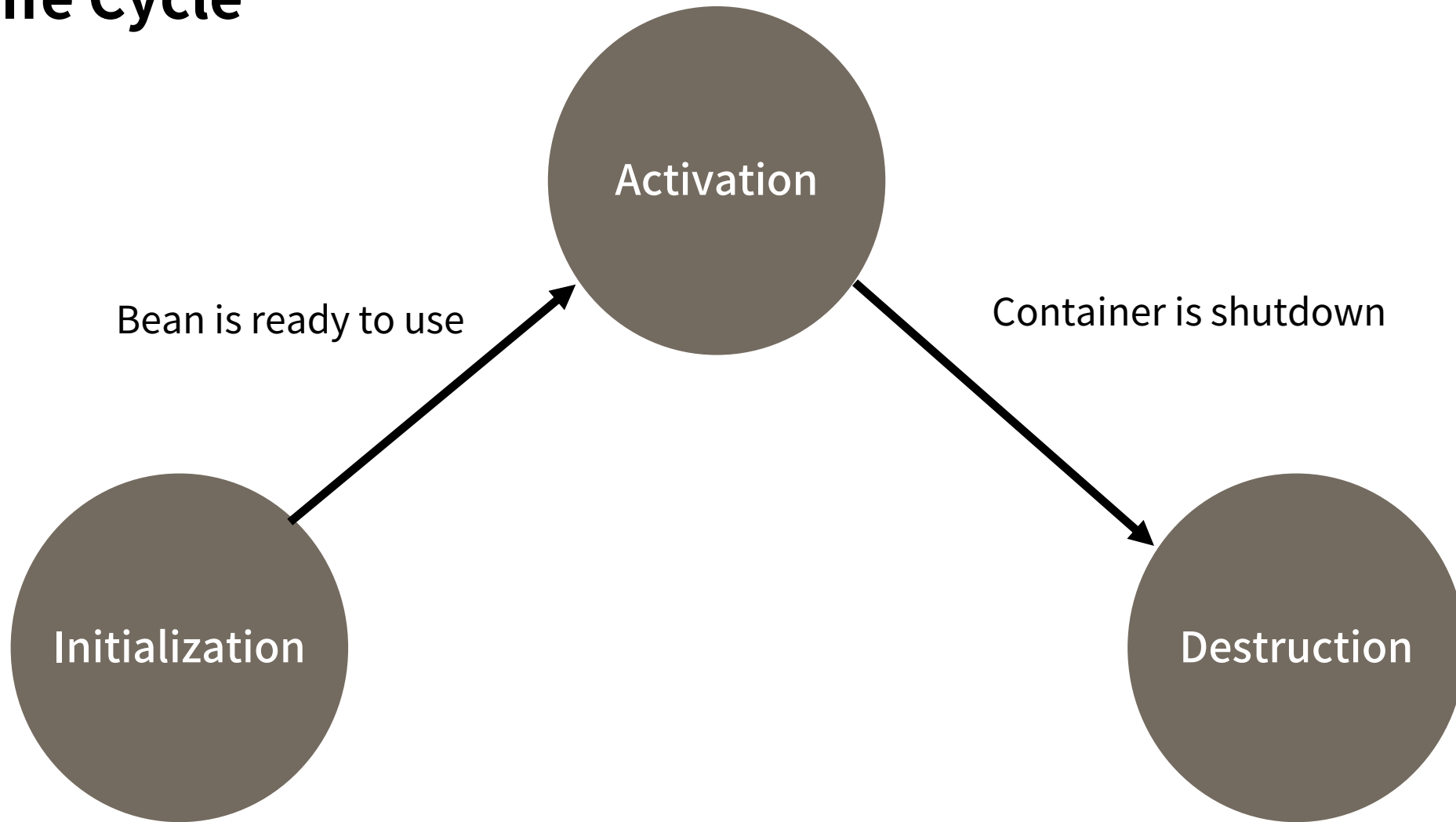
Bean Life Cycle Aware Example

```
public interface BeanNameAware extends Aware {  
    void setBeanName(String name);  
}
```

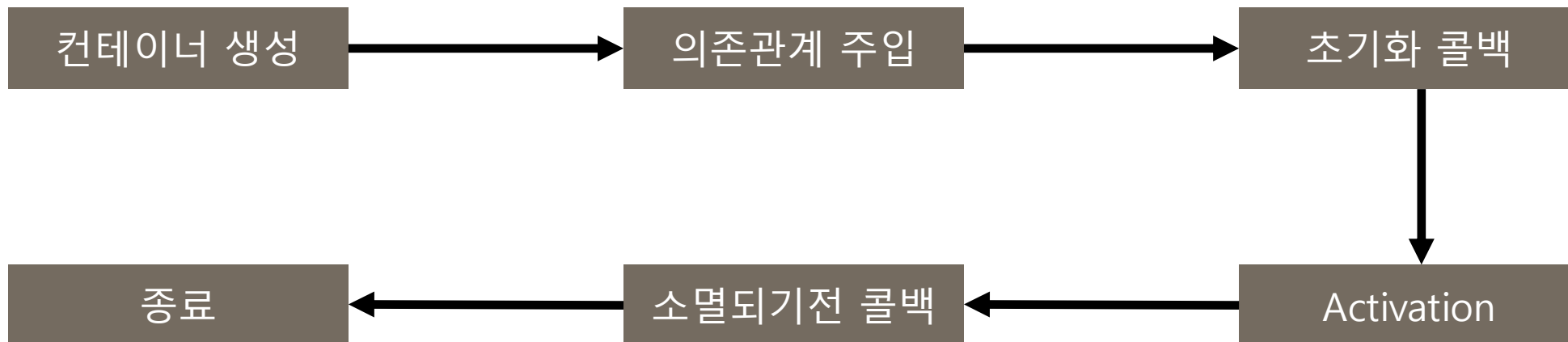
```
public class WorkRunner implements BeanNameAware {  
  
    private String beanId;  
  
    @Override  
    public void setBeanName(String name){  
        this.beanId = name;  
    }  
  
    public void execute(Work work){  
        logger.debug(  
            String.format("WorkRunner[%s] execute Work[%d]",  
                           beanId, work.getOrder()));  
        work.run();  
    }  
    ...  
}
```

BeanNameAware 인터페이스를 상속받은 빈 객체는 초기화 과정에서 빈 이름을 전달 받음


Bean Life Cycle



Bean Life Cycle



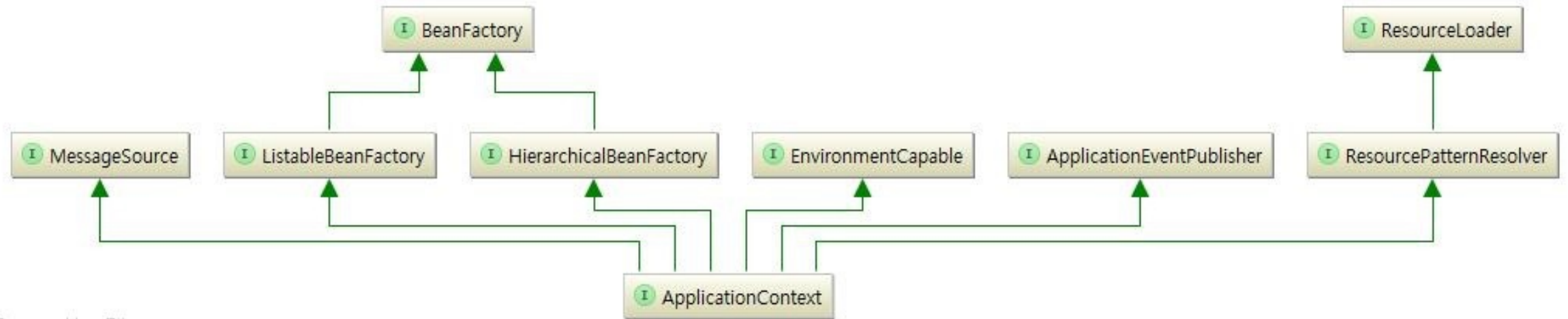
IoC(Inversion of Controller)



```
public class Car {  
  
    private Tire tire;  
  
    public Car(Tire tire) {  
        this.tire = tire;  
    }  
  
}
```

직접 new가 아닌 주입받아서 사용

ApplicationContext



Powered by yFiles.

의존성 주입의 필요성

```
public class Theater {  
  
    private TicketSeller ticketSeller;  
  
    public Theater(TicketSeller ticketSeller) {  
        this.ticketSeller = ticketSeller;  
    }  
  
}
```

- TicketSeller를 Theater에서 생성하면 강결합
- 외부에서 주입하면 느슨한 결합
- IoC컨테이너가 의존성 주입을 해주는 것

의존성 주입의 필요성

사족

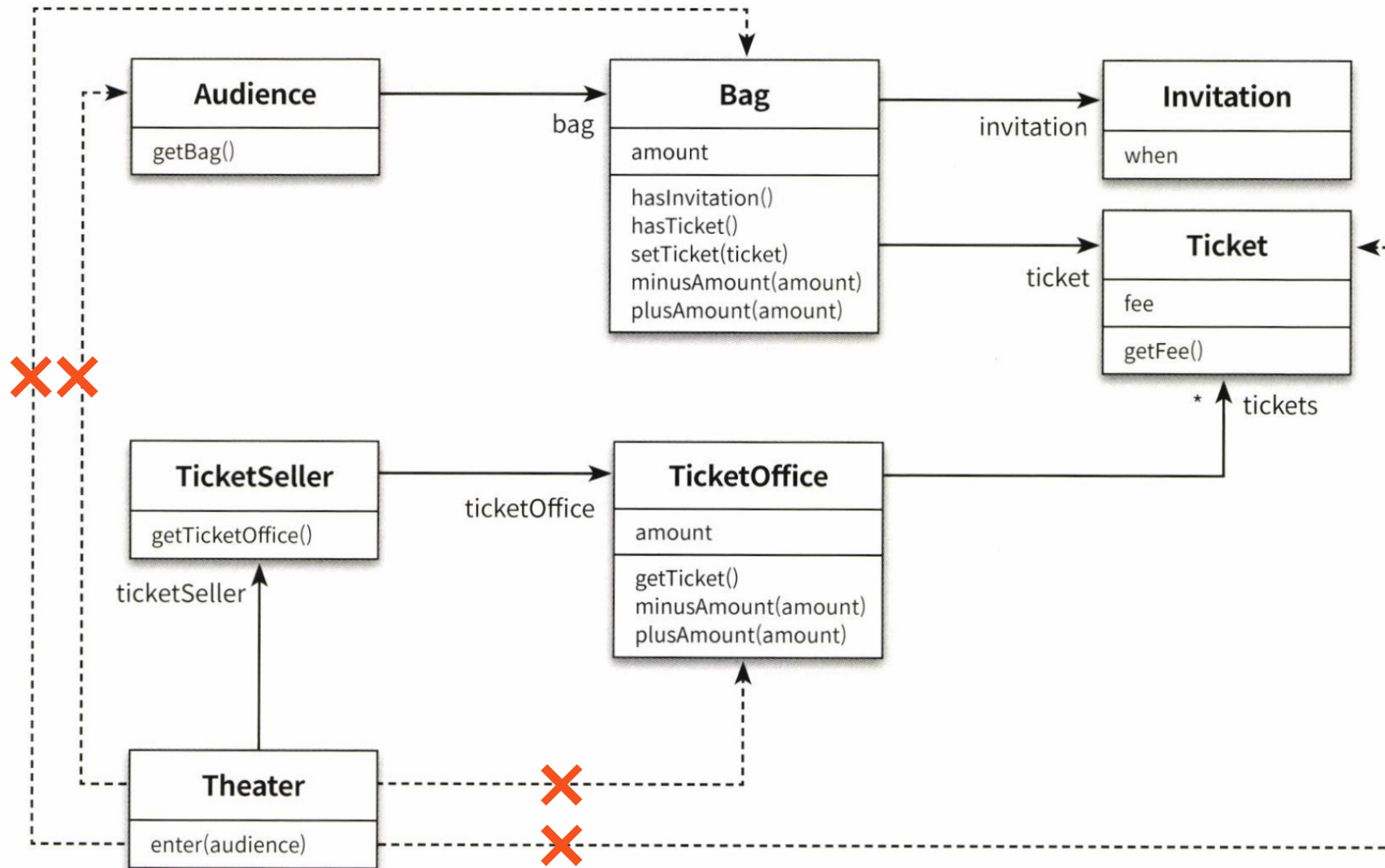
```
public class Theater {  
  
    private TicketSeller ticketSeller;  
  
    public Theater(TicketSeller ticketSeller) {  
        this.ticketSeller = ticketSeller;  
    }  
  
    public void enter(Audience audience) {  
        if (audience.getBag().hasInvitation()) {  
            Ticket ticket = ticketSeller.getTicketOffice().getTicket();  
            audience.getBag().setTicket(ticket);  
        } else {  
            Ticket ticket = ticketSeller.getTicketOffice().getTicket();  
            audience.getBag().minusAmount(ticket.getFee());  
            ticketSeller.getTicketOffice().plusAmount(ticket.getFee());  
            audience.getBag().setTicket(ticket);  
        }  
    }  
}
```

- TicketSeller를 Theater에서 생성하면 강결합
- 외부에서 주입하면 느슨한 결합
- IoC컨테이너가 의존성 주입을 해주는 것

1. Audience가 Bag을 소유
2. Bag안에는 현금과 티켓
3. TicketSeller가 TicketOffice에서 티켓 판매
4. TicketOffice안에 돈과 Ticket존재

의존성 주입의 필요성

사족



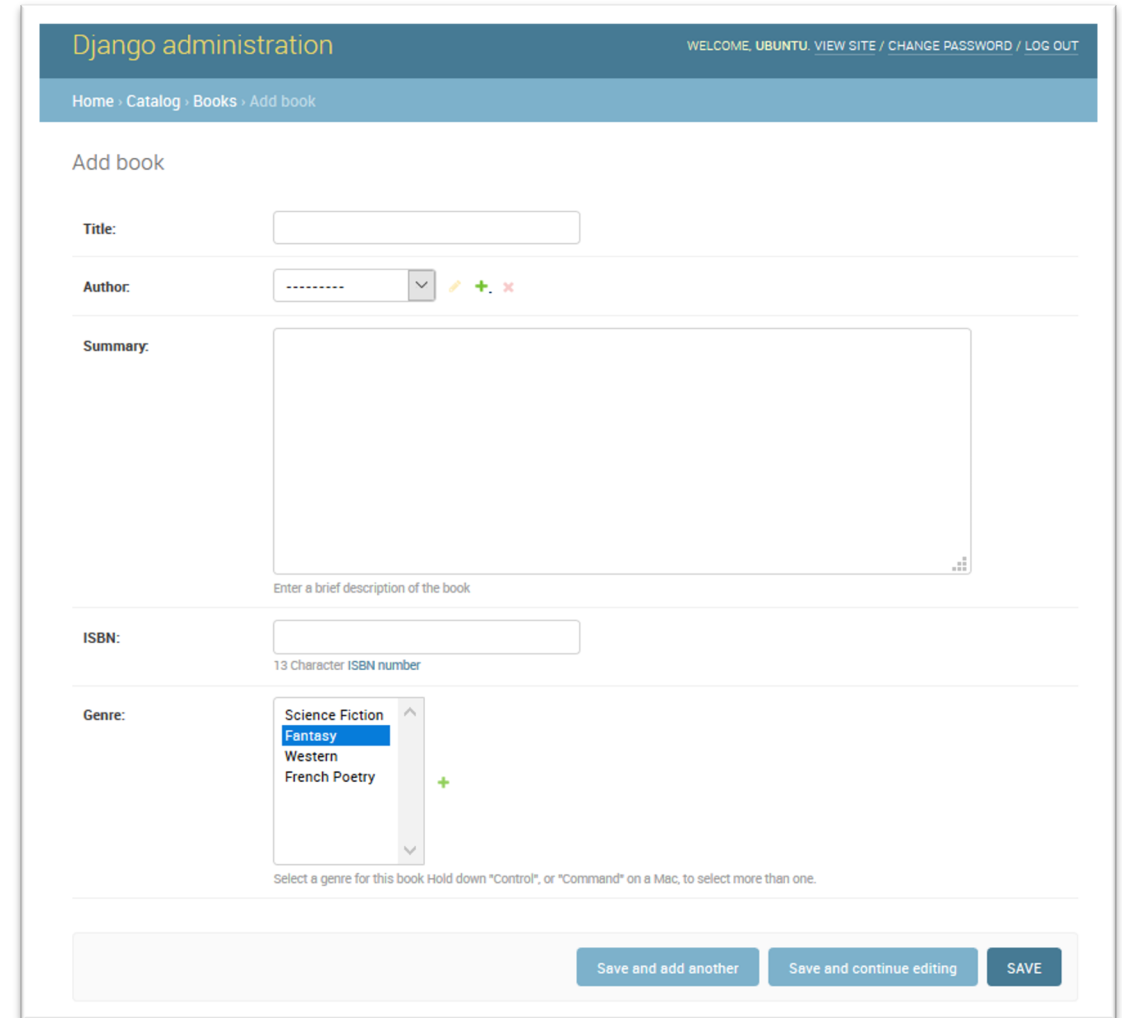
주입의 페러다임

사족

```
from django.contrib import admin
from .models import Profile

@admin.register(Profile)
class ProfileAdmin(admin.ModelAdmin):
    list_display = ['user', 'book']
```

django admin setup



The screenshot shows the Django administration interface for adding a new book. The page title is "Django administration" and the user is logged in as "WELCOME, UBUNTU". The breadcrumb trail is "Home > Catalog > Books > Add book". The form fields are:

- Title:** A text input field.
- Author:** A dropdown menu with a search icon and a plus sign.
- Summary:** A large text area with a placeholder "Enter a brief description of the book".
- ISBN:** A text input field with a placeholder "13 Character ISBN number".
- Genre:** A dropdown menu with a plus sign. The selected genre is "Fantasy". The list of genres includes "Science Fiction", "Fantasy", "Western", and "French Poetry".

At the bottom of the form, there are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

django admin page

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML
2. Using annotations

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations



Indicates that an annotated class is a "Controller" (e.g. a web controller).

This annotation serves as a specialization of `@Component`, allowing for implementation classes to be autodetected through classpath scanning. It is typically used in combination with annotated handler methods based on the `org.springframework.web.bind.annotation.RequestMapping` annotation.

Since: 2.5

See Also: `Component`, `org.springframework.web.bind.annotation.RequestMapping`, `org.springframework.context.annotation.ClassPathBeanDefinitionScanner`

Author: Arjen Poutsma, Juergen Hoeller

```
@Target({ElementType.TYPE})  
@Retention(RetentionPolicy.RUNTIME)  
@Documented  
@Component  
public @interface Controller {
```

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations

```
<beans>
  <bean
    id="studentBean"
    class="com.azhwani.beans.Student" />
  <bean
    id="employeeBean"
    class="com.azhwani.beans.Employee" />
</beans>
```

<https://devwithus.com/what-is-spring-bean/>

Bean 객체를 Spring IoC 컨테이너 등록

```
<bean id="userFactory" class="org.mangkyu.user.UserFactory" />
```

```
public class UserFactory {  
    // code  
}
```

1. Using XML

2. Using annotations

1. 어노테이션에 비해 더 많은 정보를 얻을 수 없다.
2. 변경의 어려움
3. 컴파일러

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML
2. Using “stereotype” annotations

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using “stereotype” annotations

스프링 컨테이너가 스프링 관리 컴포넌트로 식별하게 해주는 단순한 마커
scan-auto-detection과 dependency injection을 사용하기 위해서 사용되는 가장 기본 어노테이션

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ApplicationConfig {

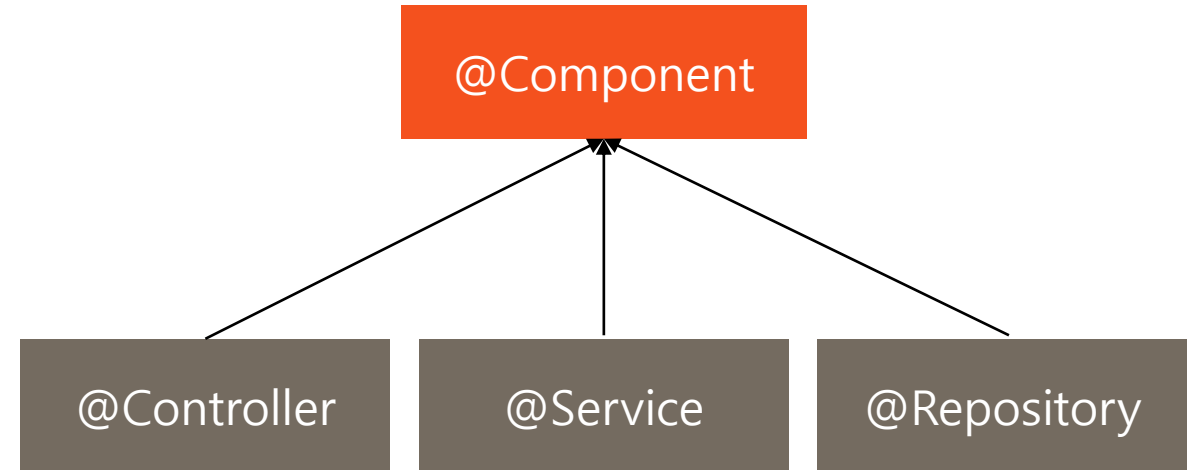
    @Bean
    public BookRepository bookRepository() {
        return new BookRepository();
    }

    @Bean
    public BookService bookService() {
        BookService bookService = new BookService();
        bookService.setBookRepository(bookRepository());
        return bookService;
    }
}
```

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations

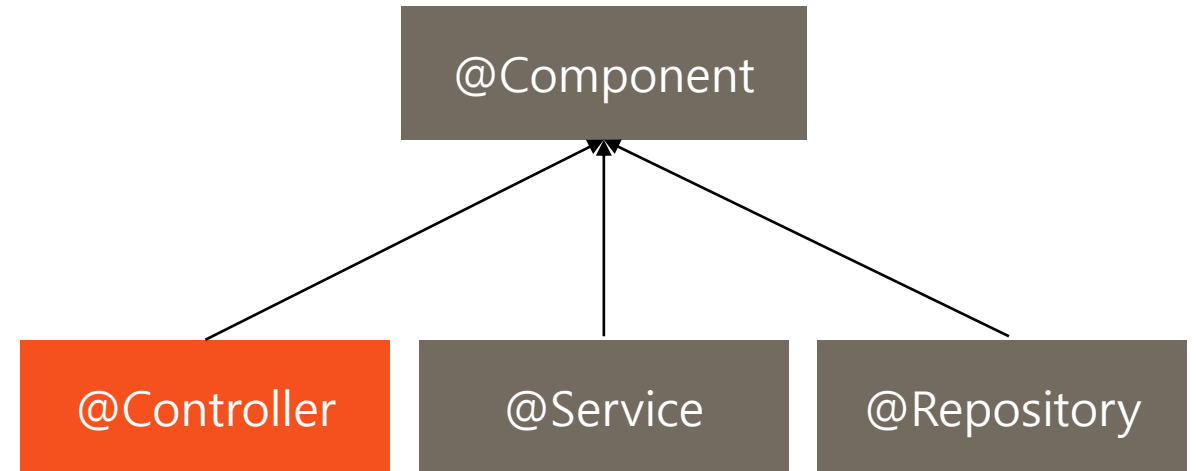


Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations



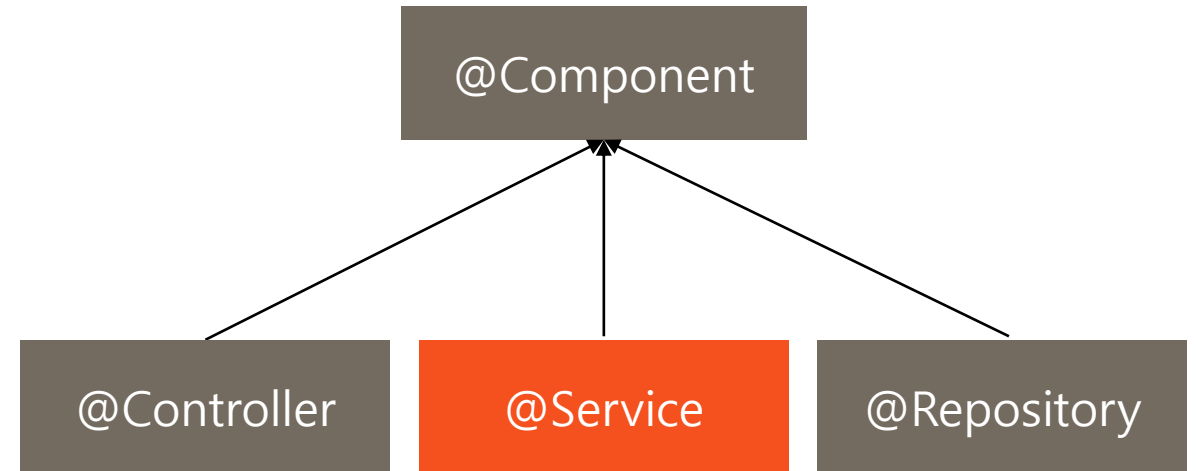
A stereotype used at class level in spring MVC.

It marks a class as a spring web controller, responsible to handle HTTP request!

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations

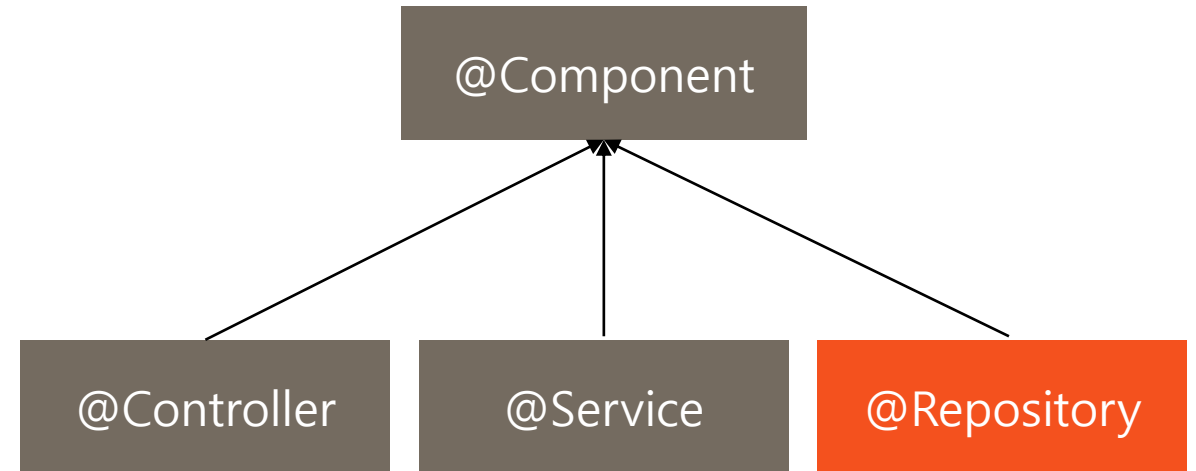


Service annotation is basically used for classes which hold business logic in the service layer.
It gives a logical sense that a class is a service

Bean 객체를 Spring IoC 컨테이너 등록

1. Using XML

2. Using annotations



Repository annotation is used to indicate that a class is a repository
Provides exception translation support for persistence operations

작지만 중요한 빈 이야기 끝

Covenant Ko