제 1차 스터디 모임

주제 : Spring 2.5 Reference
담당 : 3. IoC Contaner

발표자 : 박용권(Arawn)
발표일 : 2008년 08년 31일

The IoC container

- IoC Container 란?
- 의존성(Dependencies) 설정
- 빈 객체 범위
- 라이프 싸이클
- 어노테이션(Annotation) 기반 작성 및 설정
- 빈 객체 스캔

1. IoC Container 란?

- 제어 역행(Inversion of Control, IoC)
  - 의미

    대부분의 애플리케이션은 어떤 비즈니스 로직을 수행하기 위해 서로 협업하는 둘 또는 그 이상의 클래스들로 이루어져 있다. 각 객체는 협업할 객체의 참조를 취득해 임무를 수행하고 결과를 반환한다. 이것은 각 객체들이 서로 의존적인 관계에 있고 높은 결합도를 가지는 것을 증명한다.

    IoC 는 협업에 필요한 각 객체를 내부에서 직접 취득하지 않고 의존하는 객체를 역행적으로 취득한다. 이 말은 어떤 외부의 존재에의해 객체의 생성 시점에서 의존성을 객체로 주입(inject)받는다는 말이다. 따라서 IoC 는 한 객체가 협업해야 할 다른 객체의 참조를 취득하는 방법에 있어서 자신이 책임지고 취득하는 것이 아니라 외부의 존재에게 책임을 넘겨 역행적으로 취득함을 의미한다.

  - 의존성 주입(Dependency Injection, DI)
    - 마틴 파울러 : Inversion of Control Containers and the Dependency Injection pattern

1. IoC Container 란?

❖ Container

빈의 인스턴스 생성 방법이나 각 빈들의 연관성이 어떻게 이루어져 있는지 직접 설정 할 수 있다. 이러한 설정을 바탕으로 스프링은 애플리케이션 객체의 생명주기와 설정을 포함하고 관리한다는 점에서 일종의 컨테이너라고 볼 수 있다.

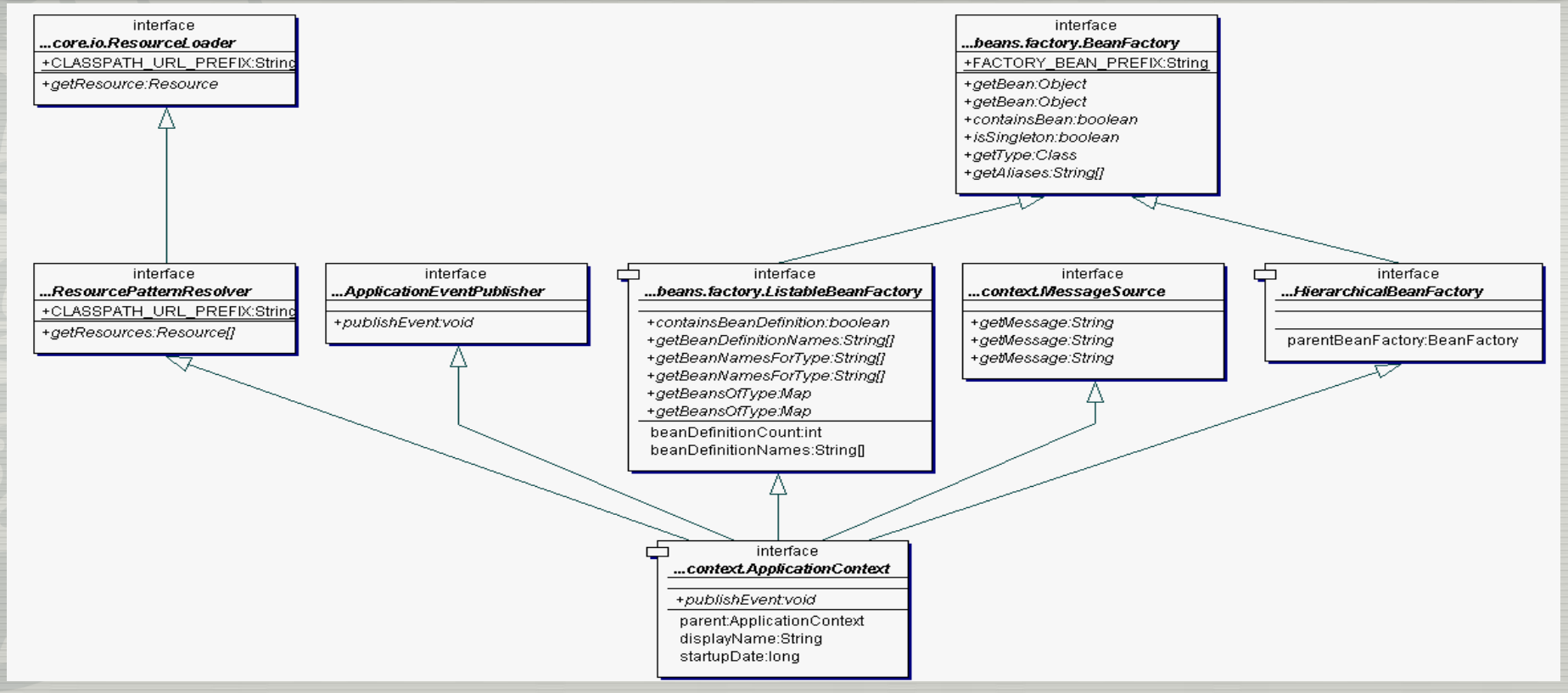
❖ IoC Container

- ◆ 애플리케이션의 객체를 관리한다.
- ◆ 각 객체들의 생명주기를 책임진다.
- ◆ 객체들간의 의존성을 관리한다.

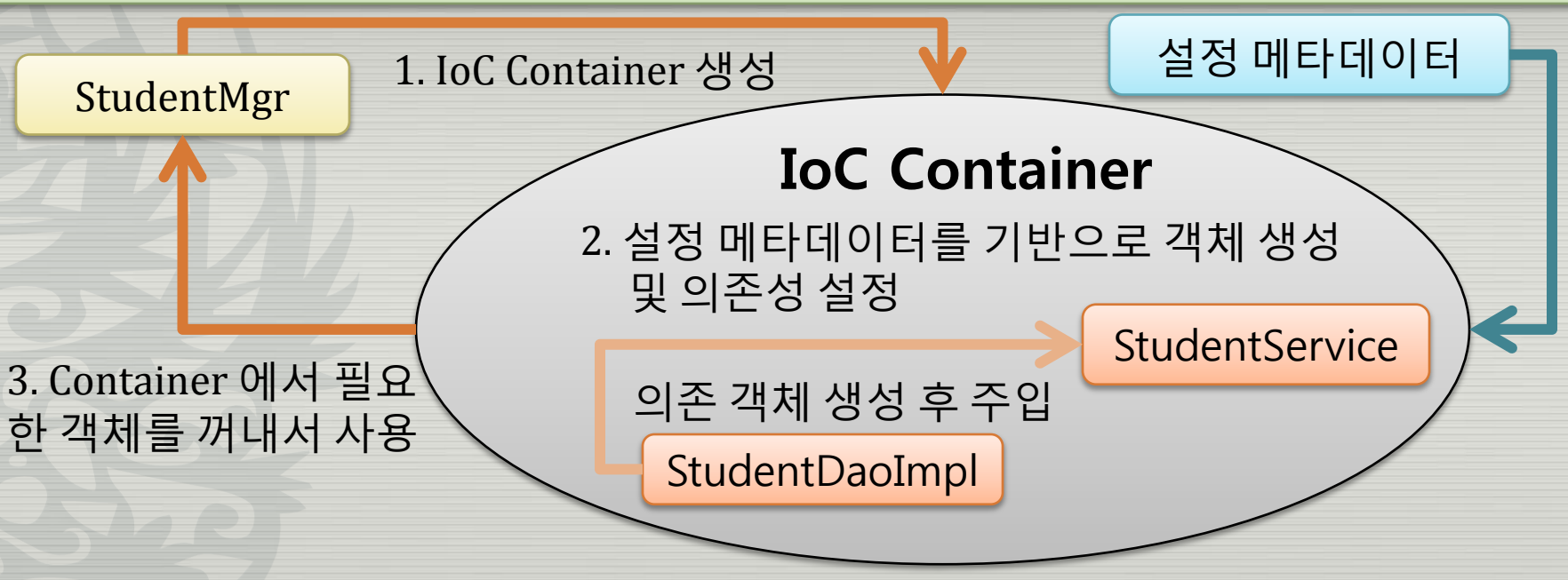
1. IoC Container 란?

- Spring IoC Container
  - BeanFactory Interface
    - 기본적인 의존성 주입을 지원하는 가장 간단한 형태의 컨테이너
    - org.springframework.beans.factory.xml.XmlBeanFactory
  - ApplicationContext Interface
    - BeanFactory 개념을 기반으로 구현된 것으로 파일과 같은 자원 처리 추상화, 메시지 지원 및 국제화 지원, 애플리케이션 이벤트 지원 등의 애플리케이션을 구현하기에 적합한 컨테이너
    - org.springframework.context.support.ClassPathXmlApplicationContext
    - org.springframework.context.support.FileSystemXmlApplicationContext
  - WebApplicationContext Interface
    - ApplicationContext 기반으로 웹 어플리케이션을 위한 추가적인 서비스를 제공
    - org.springframework.web.context.support.XmlWebApplicationContext
- 설정 메타데이터
  - 인스턴스화, 설정, 의존관계 조합 등을 정의하는 방법으로 간단하고 직관적인 XML 형태로 제공 (Java 프라퍼티 형태 또는 Spring public API 도 가능)
  - DTD 와 XML 스키마의 두 가지 방식을 지원

1. IoC Container 란?



1. IoC Container 란?



- 예제 springstudyclub.ioccontainer.general : 일반적인 객체 생성
- 예제 springstudyclub.ioccontainer.ioccontainer : IoC 를 사용한 객체 생성

2. 의존성(Dependencies) 설정

- 2-1. 의존성 주입 방식(Injecting dependencies)
- 2-2. 다양한 의존성 설정법(Dependencies and configuration in detail)
- 2-3. 빈의 조건부 초기화(Using depends-on)
- 2-4. 빈의 늦은 초기화(Lazily-instantiated beans)
- 2-5. 의존 관계 자동 설정(Autowiring collaborators)
- 2-6. 의존성 검사(Checking for dependencies)
- 2-7. 메소드 삽입(Method Injection)
- 2-8. 부모 빈을 통한 설정 재사용(Bean definition inheritance)

2-1. 의존성 주입 방식

- 생성자 방식(Constructor Injection)
  - 생성자를 통해 의존 객체나 값을 전달
  - <constructor-arg>
    - 객체 참조 : <ref bean="" /> or <constructor-arg ref="" />
    - 기본 데이터 타입 : <value></value> or <constructor-arg value="" />
  - 생성자의 인자 타입 또는 인덱스 대응
    - 전달 받는 값(또는 객체)의 타입에 가장 근접하는 생성자 선택
    - type 속성
    - index 속성
  - 예제 springstudyclub.dependencies.constructorinjection

- 프로퍼티 설정 방식(Setter Injection)
  - 자바 Beans 명명규약에 따른 setXXX() 형태의 설정 메서드를 통해서 의존 객체와 값을 전달
  - <property>
    - 객체 참조 : <ref bean="" /> or < property ref="" />
    - 기본 데이터 타입 : <value></value> or < property value="" />
    - XML 네임스페이스를 이용한 설정 (http://www.springframework.org/schema/p)
      - <bean ... p:xxx-ref=""> or <bean ... p:xxx="">
  - 예제 springstudyclub.dependencies.setterinjection

2-2. 다양한 의존성 설정법

❖ 순수값(원시변수, String 등 - Straight values (primitives, Strings, etc.))
- ◆ <value> 태그 또는 value 속성

❖ 빈 객체 참조(References to other beans (collaborators))
- ◆ <ref >

Ref 태그 속성	설명
bean	- 같은 컨테이너나 부모 컨테이너에서 bean 의 참조를 생성 - 속성 값은 대상 bean 의 id 나 name 속성 중 하나
local	- 같은 컨테이너 내에서 bean 의 참조를 생성 - 속성 값은 대상 bean 의 id 와 같아야 함
parent	- 부모 컨테이너에서 bean 의 참조를 생성 - 속성 값은 대상 bean 의 id 나 name 속성 중 하나

- ◆ 예제 springstudyclub.dependencies.referencestootherbeans

❖ 임의 빈 객체 참조(Inner beans)
- ◆ 식별 값을 가지지 않는 bean 객체 전달
- ◆ 생성자 방식, 프로퍼티 설정 방식 전부 가능
- ◆ 예제 springstudyclub.dependencies.innerbean

2-2. 다양한 의존성 설정법

- 컬렉션 타입 프로퍼티(Collections)
  - List 타입과 배열
    - <list>, <값태그> 표현식
    - type 속성 사용한 값의 타입 설정(제너릭 사용시 type 생략 가능)
  - Map 타입
    - <map> , <entry>, <key>, <값태그> 표현식
    - <entry key="" value="" /> 등의 간단한 표현식 지원
    - type 속성 사용한 값의 타입 설정(제너릭 사용시 type 생략 가능)
  - Set 타입
    - <set>, <값태그> 사용해 표현
    - type 속성 사용한 값의 타입 설정(제너릭 사용시 type 생략 가능)
  - Properties 타입
    - <props>, <prop key="">value</prop> 표현식

2-2. 다양한 의존성 설정법

- ◆ 컬렉션 타입 프로퍼티(Collections)
  - ■ <값태그>
    - • <ref> : 다른 빈 객체를 값으로 사용
    - • <bean> : 임의의 빈 객체를 생성해서 값으로 사용
    - • <value> : 순수값(원시, 래퍼 타입, String 등)을 값으로 사용
    - • <list>, <map>, <pros>, <set> : 컬렉션 객체를 값으로 사용
    - • <null> : null 값을 사용

❖ Null(Nulls)

- ◆ <null/>

❖ 다단계 프로퍼티 설정(Compound property names)

❖ 예제 springstudyclub.dependencies.configurationindetail

2-3. 빈의 조건부 초기화

❖ Using depends-on(Using depends-on)

◆ <bean id=“” depends-on=""/>

2-4. 빈의 늦은 초기화

❖ Lazily-instantiated beans(Lazily-instantiated beans)

◆ <bean id=“” lazy-init =“true"/>

❖ 예제 springstudyclub.dependencies.dependsonandlazyinit

2-5. 의존 관계 자동 설정

- 자동 설정을 위한 네 가지 방식
  - byName : 프로퍼티의 이름과 같은 이름을 갖는 빈 객체를 설정
  - byType : 프로퍼티의 타입과 같은 타입을 갖는 빈 객체를 설정
  - constructor : 생성자 파라미터 타입과 같은 타입을 갖는 빈 객체를 설정
  - autodetect : constructor 방식을 먼저 적용 후 byType 방식을 적용해 설정
- autowire 속성을 사용한 설정
  - <bean> 태그에 설정하며 하나의 bean 에만 적용
- default- autowire 속성을 사용한 설정
  - <beans> 태그에 설정하며 해당 설정 메타데이터(XML)의 모든 bean 에 적용
- 예제 springstudyclub.dependencies.autowiring

2-6. 의존성 검사

- dependency-check 속성
  - <bean name="" class="" dependency-check="none"/>

- 의존성 검사 모드
  - none : 검사를 하지 않는다.
  - simple : 원시 타입과 컬렉션 검사
  - object : 참조되는 bean 검사
  - all : simple, object 검사

2-7. 메소드 삽입

❖ 룩업 메소드 삽입(Lookup method injection)

- ◆ 컨테이너가 관리하는 빈의 내에서 임의의 빈을 반환하는 메소드를 오버라이드
- ◆ CGLIB 의 바이트코드 생성을 통해 메소드가 오버라이드된 클래스를 동적생성
- ◆ CGLIB 외부 라이브러리 필요
- ◆ <lookup-method>

속성	설명
name	룩업 대상 메소드 명
bean	반환할 bean 의 id 또는 name 속성

- ◆ 오버라이드 대상 메소드 시그니처
  - ■ <public|protected> [abstract] <return-type> theMethodName(no-arguments);

2-7. 메소드 삽입

- 임의 메소드 교체(Arbitrary method replacement)
  - 빈 내의 임의의 메소드를 교체
  - org.springframework.beans.factory.support.MethodReplacer 인터페이스 구현
  - <replaced-method>

속성	설명
name	교체 대상 메소드 명
replacer	MethodReplacer 인터페이스를 구현한 bean 의 id 또는 name 속성

  - <arg-type>
    - <replaced-method> 하위에 설정
    - 대상 메소드의 인자 형태

- 예제 springstudyclub.dependencies.methodinjection

2-8. 부모 빈을 통한 설정 재사용

- 동일한 설정 정보를 가지는 bean 들을 부모 빈을 정의한 뒤, 부모 빈 정보를 상속받아 재사용하도록 설정

- abstract 속성
  - <bean abstract="설정값">
  - 설정값 : true, false(기본)
  - true 설정이 되어 있으면 BeanFactory 는 해당 빈을 생성하지 않음

- 예제 springstudyclub.dependencies.beandefinitioninheritance

3. 빈 객체 범위

❖ 빈 범위 설정

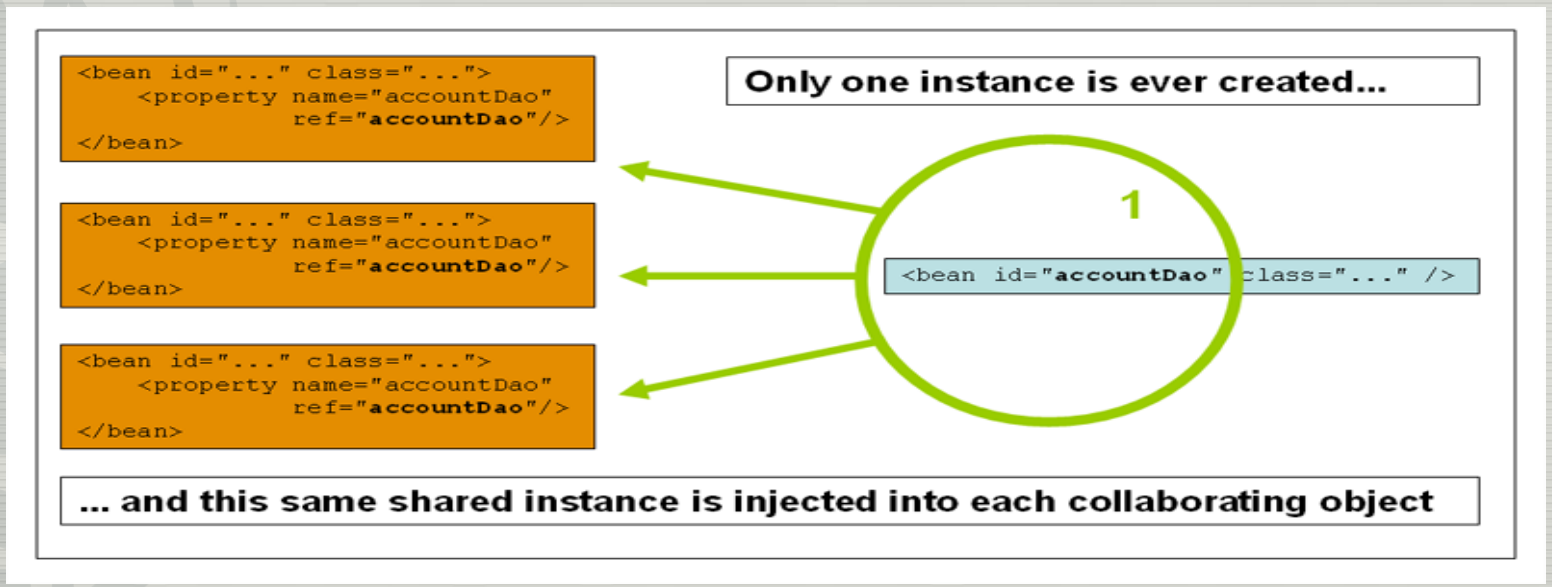
◆ scope 속성 : <bean name="" class="" scope=""/>

■ Spring 2.0 부터 지원

Scope 값	설명
Singleton	- 컨테이너에 한 개의 빈 객체만 존재(기본값)
Prototype	- 매번 새로운 객체를 생성
Request	- HTTP 요청마다 빈 객체를 생성 - WebApplicationContext 에서만 적용가능
Session	- HTTP 세션마다 빈 객체를 생성 - WebApplicationContext 에서만 적용가능
Global-session	- 글로벌 HTTP 세션에 대해 빈 객체를 생성 - 포틀릿을 지원하는 컨텍스트에 대해서만 적용가능

3. 빈 객체 범위

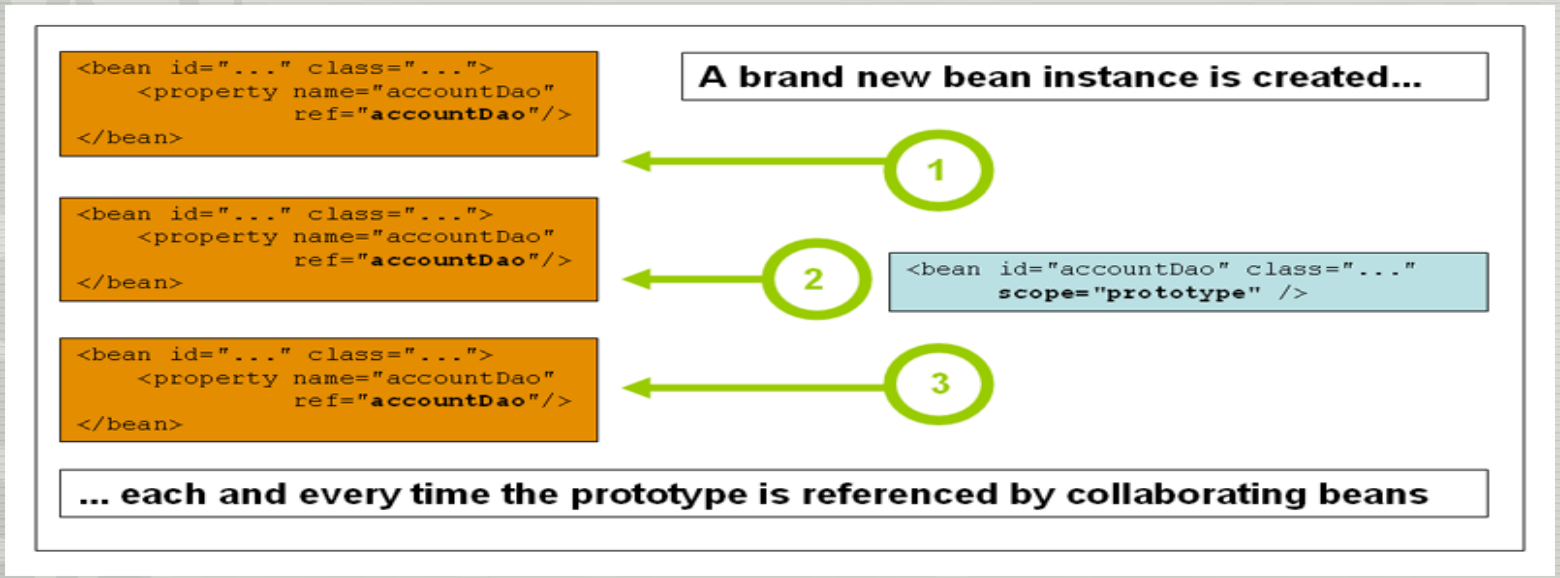
❖ Singleton



◆ Spring 1.x : <bean name="" singleton="true">

3. 빈 객체 범위

❖ Prototype



◆ Spring 1.x : <bean name="" singleton="false">

3. 빈 객체 범위

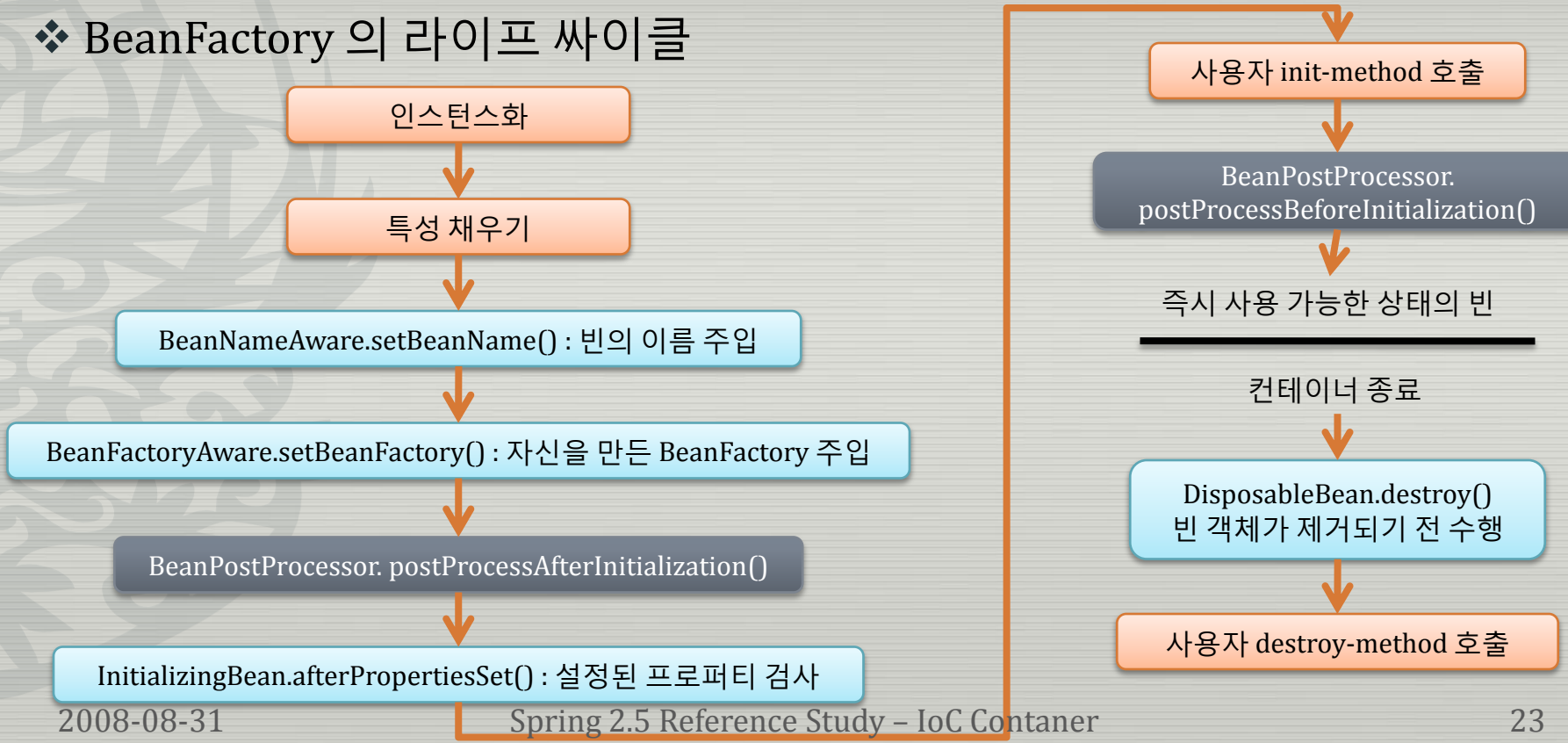
- Web-Scoped Beans(request, session, global session)
  - web container 설정
    - Servlet 2.4+ : ContextListener 등록
      - org.springframework.web.context.request.RequestContextListener
    - older web container (before Servlet 2.4) : javax.servlet.Filter 등록
      - org.springframework.web.filter.RequestContextFilter
    - ContextListener 등록
  - <aop:scoped-proxy/> : 각 빈들의 프록시 객체를 생성
    - proxy-target-class="true | false"
      - True : CGLib 를 사용하여 프록시 생성
      - False : 해당 빈이 인터페이스를 기반으로 작성되었다면, JDK API 를 사용해 프록시 생성
- springstudyclub.beanscope

3. 빈 객체 범위

- 사용자 빈 객체 범위(Custom scopes)
  - org.springframework.beans.factory.config.Scope 인터페이스 구현
  - beanFactory .registerScope("name", customScope) 메소드 또는 설정 메타데이터에서 org.springframework.beans.factory.config.CustomScopeConfigurer 를 통해 등록

4. 라이프 싸이클

❖ BeanFactory 의 라이프 싸이클



4. 라이프 싸이클

❖ ApplicationContext 의 라이프 싸이클

4. 라이프 싸이클

- 빈의 내부에서 생명주기에 관여하는 인터페이스
  - 라이프 싸이클 그림에서 xxxAware.setXXX() 와 비슷한 형태는 전부 해당 빈이 xxxAware 라는 인터페이스를 구현했을때 호출되는 메소드
- 빈의 외부에서 생명주기에 관여하는 인터페이스
  - BeanPostProcessor : BeanFactory 에서 Bean 을 생성할때 관여
  - BeanFactoryPostProcessor : BeanFactory 에서 Bean 을 생성하기 전에 관여
    - PropertyPlaceholderConfigurer
      - 외부 프로퍼티 파일을 사용해 Bean 의 정의를 설정
    - CustomEditorConfigurer
      - String 타입 설정값을 다른 타입의 설정값과 묶어준다.
- 예제 springstudyclub.lifecycle

5. 어노테이션(Annotation) 기반 작성 및 설정

❖ @Required
- ◆ 필수 프로퍼티 명시
- ◆ 스프링 2 부터 지원
- ◆ 프로퍼티의 설정 메서드에 어노테이션을 지정
- ◆ org.springframework.beans.factory.annotation.RequiredAnnotationBeanPostProcessor 를 BeanFactory 또는 ApplicationContext 등록해야 적용

❖ @Autowired
- ◆ 의존 관계 자동 설정
- ◆ 스프링 2.5 부터 지원
- ◆ 생성자, 필드, 설정 메서드의 세 곳에 적용 가능
- ◆ required 속성을 통한 필수여부 설정
- ◆ org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor 를 BeanFactory 또는 ApplicationContext 등록해야 적용

❖ @Qualifier
- ◆ 인자로 Bean 의 이름을 명시해 모호한 의존 관계 자동 설정 제한
- ◆ 스프링 2.5 부터 지원
- ◆ @Autowired 어노테이션과 함께 사용
- ◆ <qualifier> 태그와 value 속성을 사용해서 수식어 지정

5. 어노테이션(Annotation) 기반 작성 및 설정

- @Resource
  - 의존하는 빈 객체 전달(설정)
  - 자바 6 버전 및 JEE 5 버전에 추가된 어노테이션, 스프링 2.5 부터 지원
  - 필드와 설정 메서드(Setter Method)에 적용
  - name 속성에 연결할 빈 객체의 이름 지정
  - org.springframework.context.annotation.CommonAnnotationBeanPostProcessor 를 BeanFactory 또는 ApplicationContext 등록해야 적용

- @PostConstruct, @PreDestroy
  - 사용자 초기화 및 소멸 메소드와 동일한 효과(init-method, destroy-method)
  - 자바 6 버전 및 JEE 5 버전에 추가된 어노테이션, 스프링 2.5 부터 지원
  - org.springframework.context.annotation.CommonAnnotationBeanPostProcessor 를 BeanFactory 또는 ApplicationContext 등록해야 적용

- <context:annotation-config>
  - XML 스키마(http://www.springframework.org/schema/context)를 통한 필요 클래스 등록
    - RequiredAnnotationBeanPostProcessor
    - AutowiredAnnotationBeanPostProcessor
    - CommonAnnotationBeanPostProcessor

- 예제 springstudyclub.annotationbased

6. 빈 객체 스캔

❖스캔 대상이 되는 어노테이션 기반 POJO

Annotation	버전	설 명
@Repository	2.0	- @Component 기반 - DataBase Exception 을 DataAccessException 으로 변경(AOP 사용)
@Component	2.5	스캔 대상이 되는 기본 어노테이션
@Service	2.5	@Component 기반, 서비스 레이어에 위치
@Controller	2.5	- @Component 기반, WebApplicationContext 에서 사용 - org.springframework.web.servlet.mvc.Controller Interface 역할

❖ <context:component-scan>
- ◆ @Component 기반 모든 클래스를 자동 등록(ComponentScanBeanDefinitionParser)
- ◆AutowiredAnnotationBeanPostProcessor, CommonAnnotationBeanPostProcessor 등의 클래스도 빈으로 등록, 어노테이션을 이용한 설정도 함께 적용

❖ 자동 검색된 빈의 이름과 범위
- ◆ 기본적으로 검색된 클래스의 이름을 빈의 이름으로 등록(첫 글자는 소문자로 변경)
- ◆ @Scope(“”) 어노테이션을 통해서 범위 지정 가능(기본적으로 singleton 범위)

6. 빈 객체 스캔

❖ 스캔 대상 클래스 범위 지정하기
- ◆ <context:include-filter> : 대상 클래스 포함
- ◆ <context:exclude-filter> : 대상 클래스 미포함
- ◆ type 속성과 expression 속성을 사용해서 명시

Type 속성	설명 및 Expression 속성
annotation	- 클래스에 지정한 어노테이션이 적용됐는지의 여부. - expression = "org.springframework.stereotype.Controller"
assignable	- 클래스가 지정한 타입으로 할당 가능한지의 여부. - expression = "org.springstudyclub.StudyService"
regex	- 클래스 이름이 정규 표현식에 매칭되는지의 여부. - expression = "정규 표현식"
aspectj	- 클래스 이름이 AspectJ의 표현식에 매칭되는지의 여부. - expression = "AspectJ 표현식"

❖ 예제 springstudyclub.componentscan