Basic bean wiring

백기선
whiteship2000@gmail.com
http://whiteship.tistory.com

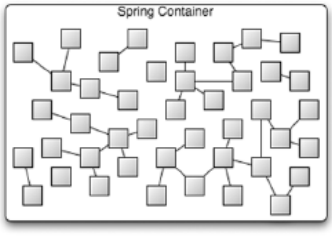
목표

- Spring Container 이해하기
- Bean의 Life Cycle 이해하기
- Bean 선언하기
- Constructor/Setter Injection 사용하기
- Bean 엮기(wiring)

Containing your bean

Containing your beans

- 모든 객체는 Spring Container가 관리
  - 객체(bean) 생성
  - 객체(bean) 설정
  - 객체들(beans) 엮기
  - 객체(bean)의 모든 라이프사이클
- Bean
  - POJO
  - JavaBean



Spring Container

- BeanFactory
- ApplicationContext

BeanFactory

- bean의 생성과 소멸을 담당한다.
- bean을 생성할 때 필요한 속성을 설정할 수 있다.
- bean의 Life Cycle에 관련된 메소드를 호출한다.
- 다수의 BeanFactory 인터페이스 구현체를 제공하고 있다.

BeanFactory 인터페이스

Method Summary	
boolean	**containsBean**(String name) Does this bean factory contain a bean with the given name?
String[]	**getAliases**(String name) Return the aliases for the given bean name, if any.
Object	**getBean**(String name) Return an instance, which may be shared or independent, of the specified bean.
Object	**getBean**(String name, Class requiredType) Return an instance, which may be shared or independent, of the specified bean.
Class	**getType**(String name) Determine the type of the bean with the given name.
boolean	**isPrototype**(String name) Is this bean a prototype?
boolean	**isSingleton**(String name) Is this bean a shared singleton?
boolean	**isTypeMatch**(String name, Class targetType) Check whether the bean with the given name matches the specified type.

XmlBeanFactory

- 생성자에 org.springframework.core.io.Resource 타입의 객체를 주어야 함.

```
BeanFactory factory =
    new XmlBeanFactory(new FileSystemResource("c:/beans.xml"));
```

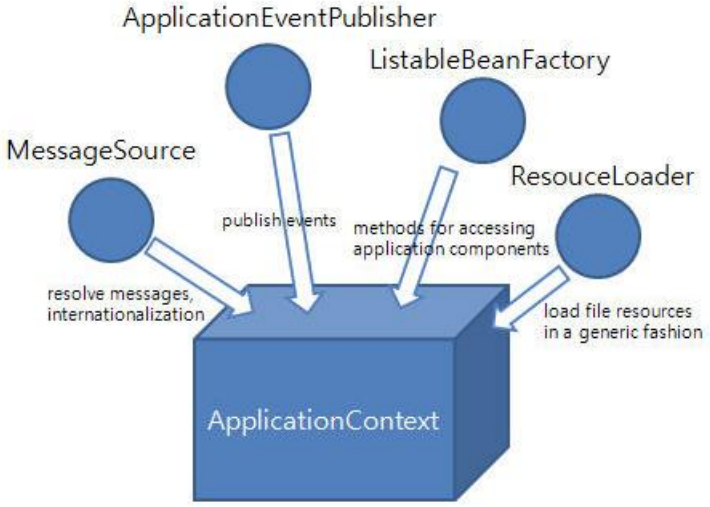
Resource 인터페이스 구현체

Resource implementation	Purpose
`org.springframework.core.io.ByteArrayResource`	Defines a resource whose content is given by an array of bytes
`org.springframework.core.io.ClassPathResource`	Defines a resource that is to be retrieved from the classpath
`org.springframework.core.io.DescriptiveResource`	Defines a resource that holds a resource description but no actual readable resource
`org.springframework.core.io.FileSystemResource`	Defines a resource that is to be retrieved from the file system
`org.springframework.core.io.InputStreamResource`	Defines a resource that is to be retrieved from an input stream
`org.springframework.web.portlet.context.PortletContextResource`	Defines a resource that is available in a portlet context
`org.springframework.web.context.support.ServletContextResource`	Defines a resource that is available in a servlet context
`org.springframework.core.io.UrlResource`	Defines a resource that is to be retrieved from a given URL

ApplicationContext

- BeanFactory의 모든 기능을 가지고 있슴.
- 텍스트 메시지를 다룸으로써 국제화를 지원한다.
- 이미지 같은 파일 리소스를 다룰 수 있다.
- 리스너로 등록된 빈이 반응하도록 이벤트를 발생시킬 수 있다.
- 역시 다수의 ApplicationContext 구현체를 제공한다.

ApplicationContext 인터페이스



자주 사용하는 ApplicationContext 구현체

- XmlWebApplicationContext
  - 웹 기반의 Spring 애플리케이션을 작성할 때 내부적으로 사용하게 됨.
- FileSystemXmlApplicationContext
  - 파일 시스템에 위치한 XML 설정 파일을 읽어 들이는 ApplicationContext
- ClassPathXmlApplicationContext
  - 클래스 패스에 위치한 XML 설정 파일을 읽어 들이는 ApplicationContext

ApplicationContext 사용 예제 코드

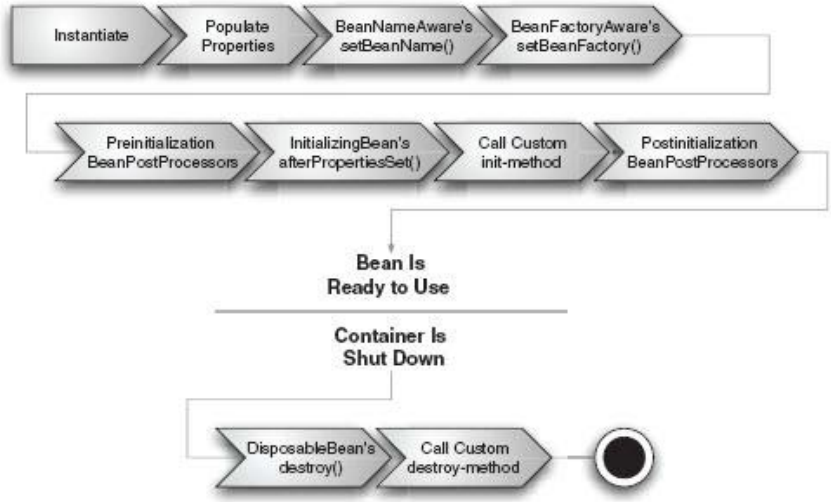
```
ApplicationContext context =
    new FileSystemXmlApplicationContext("c:/foo.xml");

ApplicationContext context =
    new ClassPathXmlApplicationContext('foo.xml");
```


BeanFactory와 ApplicationContext의 차이

- 보다 많은 기능을 제공하는 쪽은?
- Singleton bean을 preloading 하는 쪽은?
- 정답은 모두 ApplicationContext

Bean's Life Cycle (BeanFactory)



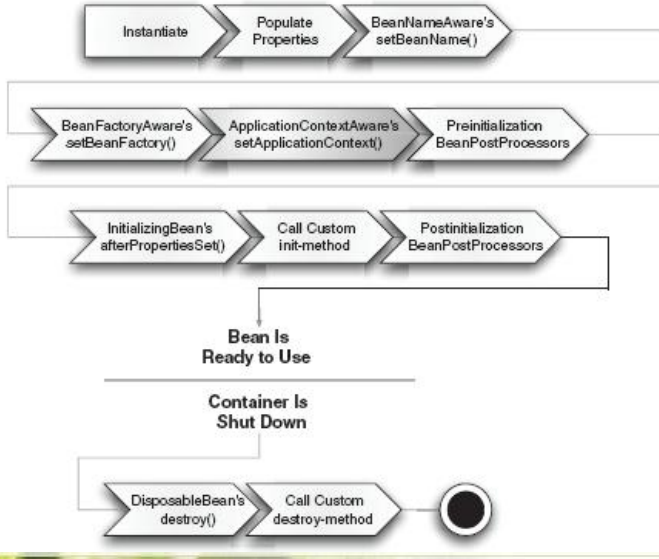
Bean's Life Cycle (BeanFactory)

Step	Description
1. Instantiate.	Spring instantiates the bean.
2. Populate properties.	Spring injects the bean's properties.
3. Set bean name.	If the bean implements `BeanNameAware`, Spring passes the bean's ID to `setBeanName()`.
4. Set bean factory.	If the bean implements `BeanFactoryAware`, Spring passes the bean factory to `setBeanFactory()`.
5. Postprocess (before initialization).	If there are any `BeanPostProcessors`, Spring calls their `postProcessBeforeInitialization()` method.
6. Initialize beans.	If the bean implements `InitializingBean`, its `afterPropertiesSet()` method will be called. If the bean has a custom init method declared, the specified initialization method will be called.

Bean's Life Cycle (BeanFactory)

Step	Description
7. Postprocess (after initialization).	If there are any `BeanPostProcessors`, Spring calls their `postProcessAfterInitialization()` method.
8. Bean is ready to use.	At this point the bean is ready to be used by the application and will remain in the bean factory until it is no longer needed.
9. Destroy bean.	If the bean implements `DisposableBean`, its `destroy()` method will be called. If the bean has a custom `destroy-method` declared, the specified method will be called.

Bean's Life Cycle (ApplicationContext)



http://whiteship.tistory.com/791

- Bean's Life Cycle

Creating beans

Creating beans (소스 코드 참조)

- 인터페이스 만들기
- 구현체 만들기
- bean 설정하기
  - <bean> 엘리먼트만 사용
  - <bean> 내부에 <constructor-arg> 엘리먼트 사용하기
  - <bean> 내부에 <property> 엘리먼트 사용하기

Setter Injection VS Construct Injection

- Setter Injection
  - 생성자 파라미터가 길어 지는 것을 막을 수 있다.
  - 생성자가 많아 지는 것을 막을 수 있다.
  - **Circular dependencies**를 피할 수 있다.
- Construct Injection
  - 종속성을 더욱 강제 한다.
  - 세터 메소드 과다 사용을 억제할 수 있다.
  - 세터 메소드가 줄어들기 때문에 실수로 속성 값을 변경하는 일을 사전에 방지할 수 있다.

Circular dependencies

- Constructor Injection 사용시 주의 사항
- 예제 코드 보기
  - test/chapter2/CircularDependenciesTest.java

Injecting into bean properties

Injecting into bean properties

- simple value
- referencing other object
- wiring collection
- wiring null

Simple value

- <property> 엘리먼트의 value 속성을 사용합니다.
- 예제 코드 보기
  - test/chapter2/SimpleValueSettingTest.java

Referencing other beans

- <property> 엘리먼트의 ref 속성을 사용합니다.
- 예제 코드 보기
  - test/chapter2/ReferencingOtherBeanTest.java

Injecting inner beans

- <property> 엘리먼트 내부에 <bean> 엘리먼트를 사용합니다. id 나 name 속성을 설정할 필요 없습니다.
- 예제 코드 보기
  - test/chapter2/InjectingInnerBeanTest.java

Wiring collections

Collection element	Useful for...
`<list>`	Wiring a list of values, allowing duplicates.
`<set>`	Wiring a set of values, ensuring no duplicates
`<map>`	Wiring a collection of name-value pairs where name and value can be of any type
`<props>`	Wiring a collection of name-value pairs where the name and value are both `Strings`

- Spring이 실제 사용하는 Collection 클래스
  - http://younghoe.info/492

Lists and arrays

- <list> 엘리먼트 내부에
  - 객체일 경우
    - <ref> 엘리먼트와 bean 속성 사용
  - Primitive 타입 값일 경우
    - <value> 엘리먼트 사용
  - Collection에 포함될 아이탬이 하나 일 경우
    - <list>의 value 속성 사용
- 예제 코드 보기
  - test/chapter2/ListAndArraysTest.java

Sets

- <set> 엘리먼트를 사용하며 <list>와 동일하게 사용합니다.
- 예제 코드 보기
  - test/chapter2/SetsTest.java

Maps

- <map> 엘리먼트 내부에
  - 키가 Primitive 타입이고 값도 Primitive 타입일 때
    - <entry> 엘리먼트의 key와 value 속성 사용
  - 키가 Primitive 타입이고 값은 Reference 타입일 때
    - <entry> 엘리먼트의 key와 value-ref 속성 사용
  - 키가 Reference 타입이고 값이 Primitice 타입일 때
    - <entry> 엘리먼트의 key-ref와 value 속성 사용
  - 키가 Reference 타입이고 값도 Reference 타입일 때
    - <entry> 엘리먼트의 key-ref와 value-ref 속성 사용
- 예제 코드 보기
  - test/chapter2/MapsTest.java

Properties

- <props> 엘리먼트 내부에
  - <prop> 엘리먼트로 값을 둘러싸고, 키는 <prop>의 key 속성을 설정.
- 예제 코드 보기
  - test/chapter2/PropertiesTest.java

Wiring nothing (null)

- <null/> 사용.
- 예제 코드 보기
  - test/chapter2/WiringNothingTest.java

Autowiring

Autowiring

- byName
- byType
- constructor
- autodetect
- no

Autowiring by name

- <bean> 엘리먼트의 autowiring 속성을 byName으로 설정.
- 예제 코드 보기
  - test/chapter2/AutowiringByNameTest.java

Autowiring by type

- <bean> 엘리먼트의 autowiring 속성을 byType으로 설정.
- 예제 코드 보기
  - test/chapter2/AutowiringByTypeTest.java

Using constructor autowiring

- <bean> 엘리먼트의 autowiring 속성을 constructor로 설정.
- 예제 코드 보기
  - test/chapter2/UsingConstructorAutowiringTest.java

Autodetect autowiring

- <bean> 엘리먼트의 autowiring 속성을 autodetect로 설정.
- 예제 코드 보기
  - test/chapter2/AutodetectAutowiringTest.java

Autowiring by default

- <beans> 엘리먼트의 default-autowire 속성에 원하는 autowiring 형태를 설정.
- 예제 코드 보기
  - test/chapter2/AutowiringByDefaultTest.java

Autowiring 단점.

- Great power comes great responsibility
  - http://whiteship.tistory.com/1121

Controlling bean creation

Bean scoping

<bean> 태그의 scope 속성에 설정

Scope	What it does
singleton	Scopes the bean definition to a single instance per Spring container (default).
prototype	Allows a bean to be instantiated any number of times (once per use).
request	Scopes a bean definition to an HTTP request. Only valid when used with a web-capable Spring context (such as with Spring MVC).
session	Scopes a bean definition to an HTTP session. Only valid when used with a web-capable Spring context (such as with Spring MVC).
global-session	Scopes a bean definition to a global HTTP session. Only valid when used in a portlet context.

Bean Scope 예제

- singleton
- prototype

Creating beans from factory methods

- <bean> 엘리먼트의 factory-method 속성 사용
- 예제코드
  - test/chapter2/UsingFactoryMethodTest.java

Initializing and destroying beans

- 초기화: <bean>의 init-method 속성
- 소멸자: <bean>의 destroy-method 속성
- 기본 초기화: <beans>의 default-init-method 속성
- 기본 소멸자: <bean>의 default-destroy-mathod 속성
- 예제코드
  - test/chapter2/InitializingAndDestroyingBeansTest.java

Summary

- Spring Container
  - BeanFactory
  - ApplicationContext
- Bean's Life Cycle
  - Whiteship's Note/791
- Property Setting
  - Value, Bean, Collection, Null
- Bean Scope
- Using Factory Method
- Using Initializing/Destroying Method

모두 한 마디씩 해주세요.

- 유치수: 샘플 코드 미리 올려주세요. 어떤 경우에 적용할 수 있는지...
- 이대엽: 잘 들었습니다. Autowiring이랑 Bean Life Cycle이 잘 안 와 닿지만 다시 알게 되어 좋다. me too.
- 최혜영: 기억에 남는게 서큘러 디펜던시스. 숙제 내주세요.
- 최한수: 집중하기가 힘들다. 방법 마련.
- 이윤걸: 테스트코드 있어서 좋았나. 테스트 너무 많아서 지루함.
- 양철근: Autowiring 좋은 것 같다. 쌩유.

감사합니다.

- 수고하셨어요~