* **스프링 설정**

스프링 Xml 네임스페이스 종류

|  |  |
| --- | --- |
| 네임스페이스 | 목적 |
| Aop | 애스펙트 선언을 위한 엘리먼트와 @AspectJ 애너테이션이 적용된 클래스를 자동으로 스프링 애스펙트로 프록시하는 엘리먼트를 제공한다. |
| Beans | 핵심 원시 스프링 네임스페이스로, 빈의 선언과 연결 방법을 정의할 수 있도록 한다. |
| Context | 스프링에 의해 직접 관리되지 않는 객체의 주입과 빈을 오토디텍트하고 오토와이어링하는 기능을 포함하여 스프링 애플리케이션 컨텍스트를 설정하기 위한 엘리먼트를 제공한다. |
| Jee | JNDI와 EJB 등의 Java EE API와의 통합을 제공한다. |
| Jms | 메시지 드리븐 POJO를 선언하기 위한 설정 엘리먼트를 제공한다. |
| Lang | Groovy, JRuby, 또는 BeanShell 스크립트로 구현되는 빈의 선언을 가능하게 한다. |
| Mvc | 애너테이션 지향 컨트롤러, 뷰 컨트롤러, 그리고 인터셉터 등의 스프링 MVC 기능을 가능하게 한다. |
| Oxm | 스프링의 객체 대 XML 매핑 구조의 설정을 지원한다. |
| Tx | 선언적 트랜젝션 설정을 제공한다. |
| util | 유틸리티 엘리먼트의 다양한 선택. 컬렉션을 빈으로 선언하는 기능과 프로퍼티 대치 엘리먼트에 대한 지원을 포함한다. |

* **생성자를 이용한 주입**

**[constructor-arg]**

<bean id=”빈 아이디” class=”클래스 경로”

constructor-arg value=”String Type 파라미터 값”

constructor-arg value=”객체 Type 파라미터 값” />

value와 ref에 생성자 파라미터를 입력하여 빈 인스턴스를 생성한다.

* **팩토리 메소드를 이용한 빈 생성**

**[factory-method]**

<bean id=”빈 아이디” class=”클래스 경로”

factory-method=”인스턴스를 생성하는 메소드명” />

하나의 인스턴스를 사용하고 싶을 때(스프링의 모든 빈은 기본적으로 싱글톤이다) 해당 빈을 싱글톤 클래스로 구현하고 factory-method 애트리뷰트를 사용하여 bean 생성 시 인스턴스 생성에 사용할 정적 메소드를 지정할 수 있다.

* **유효범위 지정**

**[scope]**

<bean id=”빈 아이디” class=”클래스 경로” scope=”prototype” />

빈을 생성할 때 마다 새로운 인스턴스를 요청하는 경우 scope를 prototype으로 설정한다.

|  |  |
| --- | --- |
| 유효범위 | 의미 |
| Singleton | 스프링 컨테이너당 하나의 인스턴스(기본값) |
| Prototype | 사용할 때마다 새 인스턴스 생성 |
| Request | HTTP 요청당 하나의 인스턴스. Spring MVC 같이 웹 환경에서 사용할 때만 유효함 |
| Session | HTTP 세션당 하나의 인스턴스. Spring MVC 같이 웹 환경에서 사용할 때만 유효함 |
| Global-session | 전역 HTTP 세션(global HTTP session)당 하나의 인스턴스. 포틀릿 환경에서 사용할 때만 유효함 |

* **빈의 초기화와 정리 작업**

**[init-method, destroy-method]**

<bean id=”빈 아이디” class=”클래스 경로”

init-method=”빈의 인스턴스화 직후에 호출될 메소드명”

destroy-method=”빈이 소멸되기 직전에 호출될 메소드명” />

한 컨텍스트 정의 파일에서 모든 빈이 같은 이름의 init-method와 destroy-method를 호출하는 경우에는 <beans> 엘리먼트에 default-init-method와 default-destroy-method를 지정해주면된다.

* **빈 프로퍼티에 주입하기**

<bean id=”빈 아이디” class=”클래스 경로” >

<property name=”변수명” value=”값”/>

<property name=”변수명” ref=”빈 명”/>

</bean>

해당 빈의 전역변수에 초기값을 세팅할 수 있다. value 값의 type은 스프링이 알아서 변환해준다. ref는 참조할 빈을 지정할 수 있다.

**내부 빈 주입**

<bean id=”빈 아이디” class=”클래스 경로” >

<property name=”변수명”>

<bean class=”클래스 경로”/>

</property>

</bean>

내부 빈으로 인스턴스를 생성하여 다른 빈들과 공유하지 않는 빈을 프로퍼티나 생성자에 주입할 수 있다. 이때 내부 빈 엘리먼트에는 id를 지정하지않는다.

**P 네임스페이스를 이용한 프로퍼티 와이어링**

<bean id=”빈 아이디” class=”클래스 경로” p:변수명=”값” p:변수명(객체)-ref=”빈 명”/>

P 네임스페이스를 사용하기 위해서는 스프링 xml 설정에 아래 선언을 추가한다.

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8”?>  <beans xmlns=<http://www.springframework.org/schema/beans>  xmlns:p=<http://www.springframework.org/schema/p>  xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>  xsi:schemalocation=”http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd”> |

* **컬렉션 와이어링**

복수형의 프로퍼티 값을 세팅해줄 때 사용하는 엘리먼트이다.

|  |  |
| --- | --- |
| 컬렉션용 엘리먼트 | 용도 |
| <list> | 값의 리스트에 사용. 중복허용  배열이나 java.util.Collection |
| <set> | 값의 집합에 사용. 중복 없음 보장  배열이나 java.util.Collection |
| <map> | 키-값 쌍의 컬렉션에 사용. 키와 값의 타입에 제한이 없음  java.util.Map |
| <props> | 키-값 쌍의 컬렉션에 사용. 키와 값이 모두 String  java.util.Properties |

**[list, set]**

<bean id=”빈 아이디” class=”클래스 경로”>

<property name=”컬렉션형태의 변수명(A)”

<list(또는 set)>

<ref bean=”A 컬렉션의 클래스 형태의 빈 명(B1)”/>

<ref bean=”A 컬렉션의 클래스 형태의 빈 명(B2)”/>

<ref bean=”A 컬렉션의 클래스 형태의 빈 명(B3)”/>

</list(또는 set)>

</property>

</bean>

**[map]**

<bean id=”빈 아이디” class=”클래스 경로”>

<property name=”맵형태의 변수명(A)”

<map>

<entry key=”키 이름” value-ref=”A 맵의 value 클래스 형태의 빈 명(B1)”/>

<entry key=”키 이름” value-ref=”A 맵의 value 클래스 형태의 빈 명(B2)”/>

<entry key=”키 이름” value-ref=”A 맵의 value 클래스 형태의 빈 명(B3)”/>

</map>

</property>

</bean>

map 엘리먼트에서는 key, key-ref, value, value-ref 모두 사용할 수 있으며 String 타입이 아닌 경우 사용할 수 있는 유일한 프로퍼티 값 주입 방법이다.

**[props]**

<bean id=”빈 아이디” class=”클래스 경로”>

<property name=”Property 형태의 변수 명”

<props>

<prop key=”키 이름”>값1<prop/>

<prop key=”키 이름”>값2<prop/>

<prop key=”키 이름”>값3<prop/>

</props>

</property>

</bean>

prop 엘리먼트는 값에 String만 주입할 수 있다.

* **프로퍼티에 널 와이어링**

<property name=”변수 명”><null/></property>

명시적으로 null을 사용해야하는 경우 사용.

* **표현식 와이어링**

런타임까지 알지못하는 값을 프로퍼티에 할당하기위해 스프링 표현식 언어(SpEL)를 사용한다.

SpEL은 다음과 같은 기능을 표현할 수 있다.

- ID로 빈을 참조하는 기능

- 메소드 호출과 객체의 프로퍼티 액세스

- 값의 수학, 관계, 그리고 논리연산

- 정규 표현식 매칭

- 컬렉션 처리

**리터럴 값**

<property name=”count” value=”#{5}”/>

<property name=”count” value=”블라블라 #{5}”/>

<property name=”count” value=”#{89.7}”/>

<property name=”count” value=”#{2e4}”/>

<property name=”count” value=”#{‘Hello’}”/> <property name=’count’ value=’#{“Hello”}’/>

<property name=”count” value=”#{false}”/>

**빈, 프로퍼티, 메소드 참조**

<property name=”변수명(빈)” value=”#{빈이름}”/>

같은 표현으로 value대신 ref를 사용하여 빈을 주입할 수 있다. (ex. ref=”빈 명”)

<property name=”변수 명” value=”#{빈이름.변수명}”/>

참조할 빈의 애트리뷰트 값을 가져와 주입한다.

<property name=”변수 명” value=”#{빈이름.메소드명()?.toUpperCase()}”/>

변수명 대신 메소드를 사용하여 값을 가져올 수도 있다. 이때 메소드에서 가져온 값의 null체크를 하기위해 ?.를 사용한다.

**타입을 이용한 작업**

<property name=”변수명” value=”#{T(java.lang.Math).PI}”/>

<property name=”변수명” value=”#{T(java.lang.Math).random()}”/>

**컬렉션을 통한 선별**

객체 목록 구성

<util:list id=”컬렉션 이름”>

<bean class=”클래스 경로” p:변수명1=”값1-1” 변수명2=”값2-1” 변수명3=”값3-1”/>

<bean class=”클래스 경로” p:변수명1=”값1-2” 변수명2=”값2-2” 변수명3=”값3-2”/>

<bean class=”클래스 경로” p:변수명1=”값1-3” 변수명2=”값2-3” 변수명3=”값3-3”/>

</util:list>

**컬렉션 멤버 액세스**

<property name=”세팅할 빈의 변수 명” value=”#{컬렉션이름[2]}”/>

객체 목록에서 3번째를 선택하여 프로퍼티에 할당한다.

<property name=”세팅할 빈의 변수 명” value=”#{컬렉션이름[‘변수명1’]}”/>

Map 컬렉션의 멤버 조회가 가능하다.

**컬렉션 멤버 선택**

<property name=”세팅할 빈의 변수 명” value=”#{컬렉션이름.?[변수명1 gt 1000]}”/>

조건에 해당하는 목록만 추려내어 할당하고자 할 때 .?를 사용한다.

<property name=”세팅할 빈의 변수 명” value=”#{컬렉션이름.^[변수명1 gt 1000]}”/>

조건에 해당하는 목록 중 첫 번째에 일치하는 항목을 할당한다.

<property name=”세팅할 빈의 변수 명” value=”#{컬렉션이름.$[변수명1 gt 1000]}”/>

조건에 해당하는 목록 중 마지막 번째에 일치하는 항목을 할당한다.

**컬렉션 프로젝션**

<property name=”세팅할 빈의 변수 명” value=”#{컬렉션이름.![컬렉션의 변수명]}”/>

컬렉션 각 멤버의 특정 프로퍼티를 새로운 컬렉션으로 모은다.