

Dependency Injection Yöntemleri



Dependency Injection Yöntemleri

- DI'nın **iki farklı şekli** vardır
 - **Constructor** injection
 - **Setter** injection
- Constructor injectionda bağımlılıklar nesne yaratılırken **constructor parametreleri** ile enjekte edilirler
- Setter injectionda ise bağımlılıklar nesne yaratıldıktan sonra **setter metotlar** aracılığı ile enjekte edilirler

Constructor Injection

```
<bean id="bar" class="x.y.Bar"/> Bar bar = new Bar();
```

```
<bean id="baz" class="x.y.Baz"/> Baz baz = new Baz();
```

```
<bean id="foo" class="x.y.Foo"> Foo foo = new Foo(bar,baz);
    <constructor-arg ref="baz"/>
    <constructor-arg ref="bar"/>
</bean>
```

```
public class Foo {
    public Foo(Bar bar,Baz baz) {

    }
}
```

Bean oluşturulurken verilen **constructor parametrelerinin** her biri bir bağımlılığı karşılar. **Constructor argümanlarının tiplerine** bakılarak parametrelerin uyumluluğu kontrol edilir

Constructor Injection

- **<constructor-arg>** elemanlarının **sırası önemli değildir**
- Spring sınıf içerisinde **uygun constructor'ı** kendisi tespit edecektir
- Fakat bazı durumlarda **bu davranış yetersiz kalabilir**

```
public class Foo {
    public Foo(Baz baz, Bar bar) {
    }

    public Foo(Bar bar, Baz baz) {
    }
}
```

Böyle bir durumda Spring'e hangi constructor-arg hangi constructor argümanına karşılık geliyor belirtmek gerekir

Constructor Injection

```
<bean id="foo" class="x.y.Foo">
```

```
  <constructor-arg index="0" ref="baz"/>
```

```
  <constructor-arg index="1" ref="bar"/>
```

```
</bean>
```

index attribute ile constructor argümanının sırası belirtilebilir (index 0 tabanlıdır)

Constructor Injection

- Argüman tipleri primitive ise **hangi constructor-arg elemanının hangi parametreye karşılık geldiği** dışarıdan yardım almadan tespit edilemeyebilir

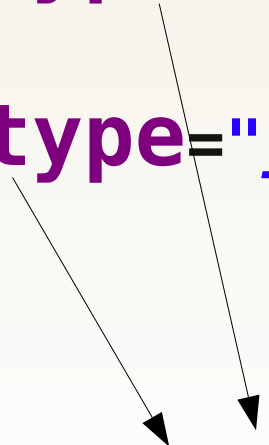
```
public class Foo {  
    public Foo(String s, int i) {  
        //...  
    }  
}
```

Constructor Injection

```
<bean id="foo" class="x.y.Foo">
  <constructor-arg type="int" value="750"/>

  <constructor-arg type="java.lang.String"
value="42"/>

</bean>
```



Type attribute'una verilen tip parametreleri bu belirsizliği çözmeye yardımcı olur

Setter Injection

`<bean id="bar" class="x.y.Bar"/>` —→ `Bar bar = new Bar();`

`<bean id="baz" class="x.y.Baz"/>` —→ `Baz baz = new Baz();`

`<bean id="foo" class="x.y.Foo">` —→ `Foo foo = new Foo();`

`<property name="bar" ref="bar"/>` —→ `foo.setBar(bar);`

`foo.setBaz(baz);`

`<property name="baz" ref="baz"/>`
`</bean>`

Bean'lerdeki **setter** metotları çağırılarak gerçekleştirilir
Öncelikle nesne **no-arg constructor** çağırılarak yaratılır.
Ardından property elemanları ile Bağımlılıklar enjekte edilir

Constructor ve Setter Injection

```
<bean id="bar" class="x.y.Bar"/> Bar bar = new Bar();
```

```
<bean id="baz" class="x.y.Baz"/> Baz baz = new Baz();
```

```
<bean id="foo" class="x.y.Foo"> Foo foo = new Foo(bar);  
    <constructor-arg ref="bar"/>  
    <property name="baz" ref="baz"/> foo.setBaz(baz);  
</bean>
```

Bean tanımında hem constructor, hem de setter injection birlikte kullanılabilir

Inner Bean Tanımları

Tek kullanımlık bean tanımlarıdır

```
<bean id="outer" class="x.y.Foo">
  <property name="bar">
```

<property/> veya <constructor-arg/> elemanları içerisinde tanımlanır

```
<bean class="x.y.Bar">
```

```
  <property name="name" value="My Bar"/>
```

```
  <property name="level" value="5"/>
```

```
</bean>
```

```
</property>
</bean>
```

Id veya name attribute **gerektirmez**, container bunları göz ardı eder

İletişim

- **Harezmi** Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- <http://www.java-egitimleri.com>
- info@java-egitimleri.com

