Property PlaceHolder Kabiliyeti

Placeholder Kabiliyeti

- Bean tanımlarında ortama veya platforma göre **değişiklik gösteren değerleri** konfigürasyon **metadata'nın dışında bir yer**de yönetmeyi sağlar
- Örneğin JDBC bağlantı ayarlarını doğrudan bean tanımı içerisine koymak yerine bu değerler **bir properties dosyasından** yüklenebilir
- Ya da Properties dosyaları yanında ilaveten Java **system property**'leri de kullanılabilir

Placeholder Kabiliyeti

```
<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${jdbc.driverClassName}"/>
    <property name="url" value="${jdbc.url}"/>
    <property name="username" value="${jdbc.username}"/>
    <property name="password" value="${jdbc.password}"/>
</bean>
```

${} ile belirtilmiş placeholder değerleri properties dosyalarından veya JVM sistem property'lerinden resolve edilir

Placeholder Kabiliyetinin Aktivasyonu

Aşağıdaki bean tanımının Spring Container içerisinde yapılması gerekir

```
<bean
class="org.springframework.beans.factory.config.PropertySourcesPlaceholderConfigurer">
    <property name="locations"
value="classpath:application.properties"/>
</bean>
```

Placeholder değişkenleri locations ile belirtilen properties dosya(ları)dan çözümlenecektir

```
jdbc.driverClassName=org.hsqldb.jdbcDriver
jdbc.url=jdbc:hsqldb:hsql://production:9002
jdbc.username=sa
jdbc.password=secret
```

application.properties

Namespace Kabiliyeti ile Placeholder Aktivasyonu

```
<context:property-placeholder
location="classpath:application.properties"/>
```

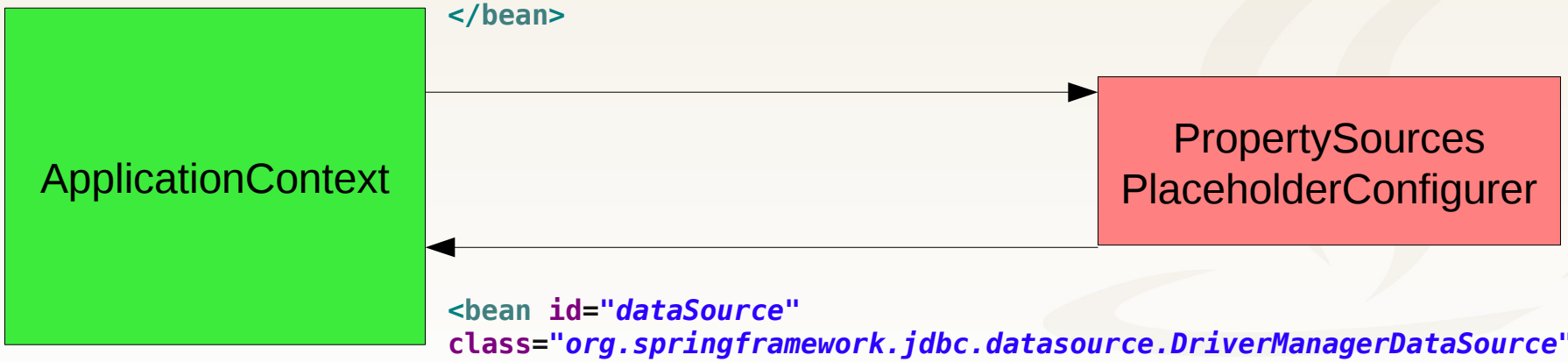
Bir önceki bean tanımına karşılık gelir. Context namespace'inin sağladığı bir kolaylıktır

Spring 3 ile birlikte önerilen ve doğru tanımlama yöntemi budur

Location attribute'una virgülle ayrılarak birden fazla properties dosyasının path'i yazılabilir

Placeholder Kabiliyeti Nasıl Çalışır?

```
<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${jdbc.driverClassName}"/>
    <property name="url" value="${jdbc.url}"/>
    <property name="username" value="${jdbc.username}"/>
    <property name="password" value="${jdbc.password}"/>
</bean>
```




```
<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="org.hsqldb.jdbcDriver"/>
    <property name="url" value="jdbc:hsqldb:hsql://production:9002"/>
    <property name="username" value="sa"/>
    <property name="password" value="secret"/>
</bean>
```

Placeholder Kabiliyeti Ne Zaman Devreye Girer?

Konfigürasyon metadata bilgisi yüklendikten sonra bean'lar yaratılmadan evvel devreye girer

```
<bean id="foo" class="${foo.class}"/>
```

Dolayısı ile bean tanımlarındaki sınıf attribute'ları da properties'den yüklenebilir

```
foo.class=x.y.FooImpl
```

application.properties

Değişkenlerin Ortama Göre Yüklenmesi

```
<context:property-placeholder
location="classpath:application.properties,
          classpath:application-${targetPlatform}.properties"/>
```

targetPlatform bizim tanımladığımız bir değişkendir, değerleri de bizim tarafımızdan tanımlanmıştır

-DtargetPlatform=dev|test|prod gibi JVM system property ile uygulamaya verilebilir, yada **OS env değişkeni** olarak tanımlanabilir

Spring runtime'da **application-dev.properties**, **application-test.properties** veya **application-prod.properties** dosyalarından uygun olanını yükler

İletişim

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