Spring Bean Profil Kabiliyeti

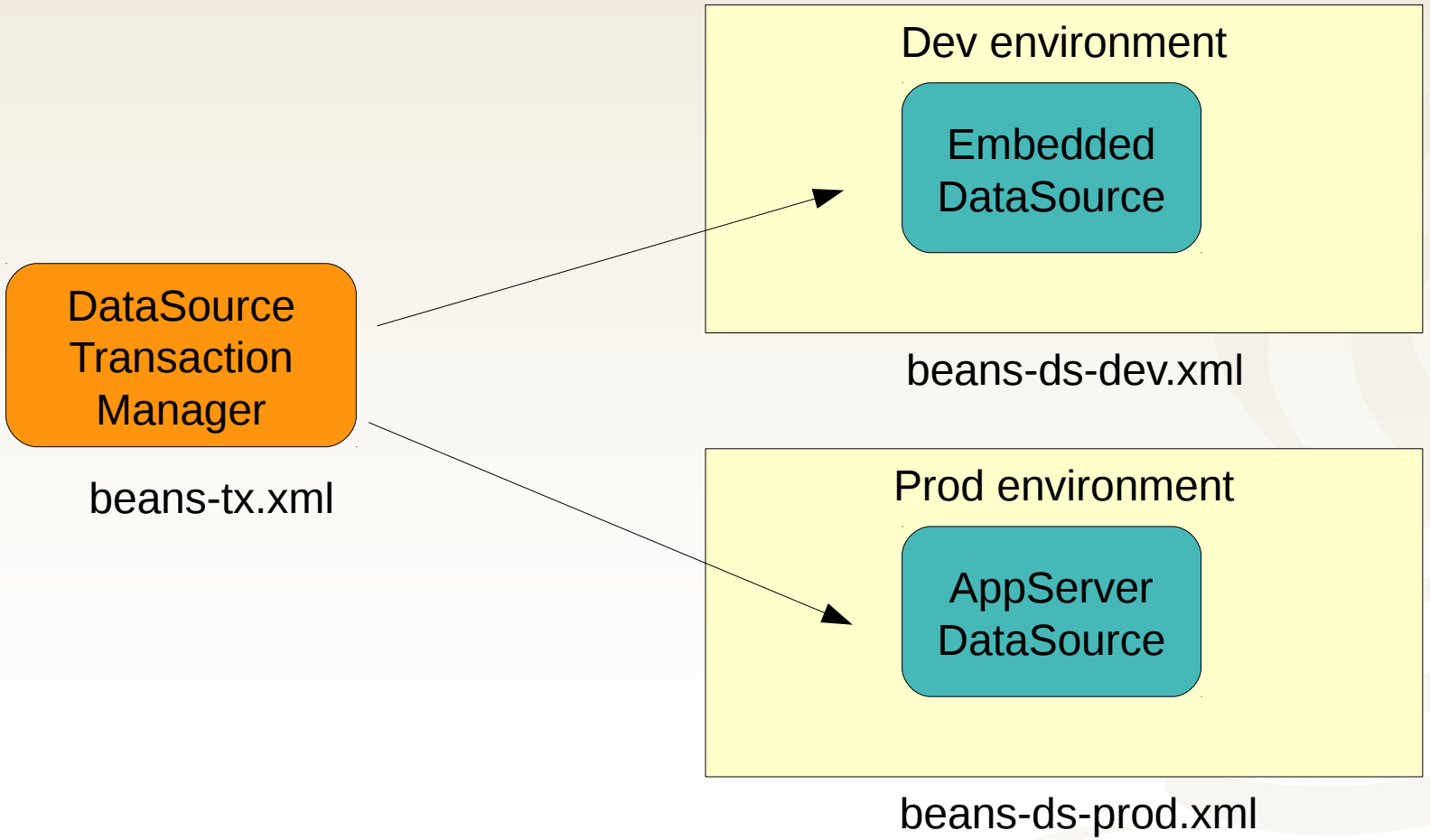
Platforma Özgü Bean Tanımlama İhtiyacı

- Bazı durumlarda **farklı platformlara veya ortamlara farklı bean tanımları** yapmak gerekebilir
- Örneğin **dev ortamında** geliştirme sürecini kolaylaştıracak test amaçlı **embedded bir dataSource** bean'i tanımlanabilir
- **Prod ortamında** ise uygulama sonucusunda tanımlamış **connection pool kabiliyetine sahip dataSource** instance'ının kullanılması istenebilir

Spring 3 Öncesinde Platforma Özgü Bean Tanımları

- **Spring 3 öncesinde** böyle bir ihtiyaç bean tanımlarını farklı konfigürasyon dosyalarında yapıp, **platforma uygun konfigürasyon dosyalarını** placeholder yardımı ile **import ederek** çözülmekteydi

Spring 3 Öncesinde Platforma Özgü Bean Tanımları



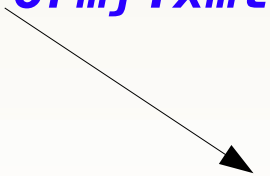
Spring 3 Öncesinde Platforma Özgü Bean Tanımları

```
<beans...>

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

   <import resource="classpath:/appcontext/beans-ds-$
{targetPlatform}.xml"/>

</beans>
```



-DtargetPlatform=dev

Spring 3 ve Bean Profile Kabiliyeti

- Spring 3 ile birlikte **bean profile kabiliyeti** sayesinde aynı bean konfigürasyonu içerisinde **bean tanımlarını platforma veya ortama göre gruplayarak yapmak** mümkün hale gelmiştir

Spring 3 ve Bean Profile Kabiliyeti

JAVA Eğitimleri

```
<beans...>

    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"/>
    </bean>

    <beans profile="dev">
        <jdbc:embedded-database type="H2" id="dataSource">
            <jdbc:script location="classpath:/schema.sql"/>
            <jdbc:script location="classpath:/data.sql"/>
        </jdbc:embedded-database>
    </beans>

    <beans profile="prod">
        <jee:jndi-lookup jndi-name="java:comp/env/jdbc/DS" id="dataSource"/>
    </beans>

</beans>
```

Spring 3 ve Bean Profile Kabiliyeti

- Child `<beans profile="...">` tanımları **diğer bean tanımlarından sonra** yapılmalıdır
- Kendilerinden sonra **başka tanım olmamalı**dır
- Profil tanımları dekleratif veya programatik olarak aktive edilebilir
- Programatik olarak **ApplicationContextInitializer** arayüzü implement edilerek **Environment** üzerinde set edilebilir

Spring 3 ve Bean Profile Kabiliyeti

- Dekleratif olarak JVM sistem parametresi olarak belirtilebilir
  - -Dspring.profiles.active=dev,oracle
- Ya da web.xml'de **spring.profiles.active** context paramteresi ile set edilebilir

```
<context-param>
    <param-name>spring.profiles.active</param-name>
    <param-value>dev,oracle</param-value>
</context-param>
```

Bean Profile Kabiliyeti ve Testler

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/appcontext/beans.xml")
@ActiveProfiles({"prod"})
public class FooTests {

    @Test
    public void testFoo() {
    ...
    }

}
```

@ActiveProfiles anotasyonu ile testlerin çalışması sırasında aktif olması istenen profiller listelenir

İletişim

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