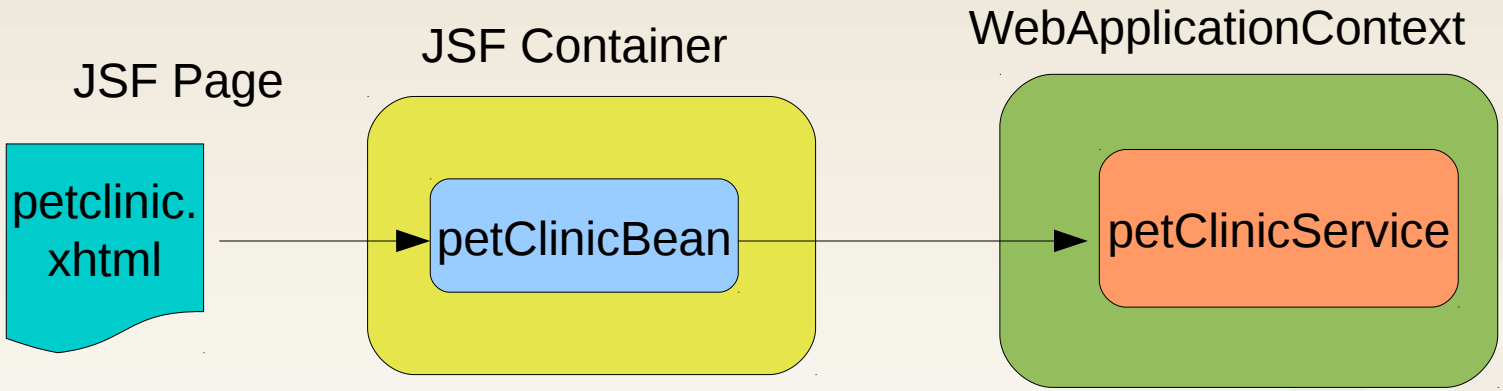
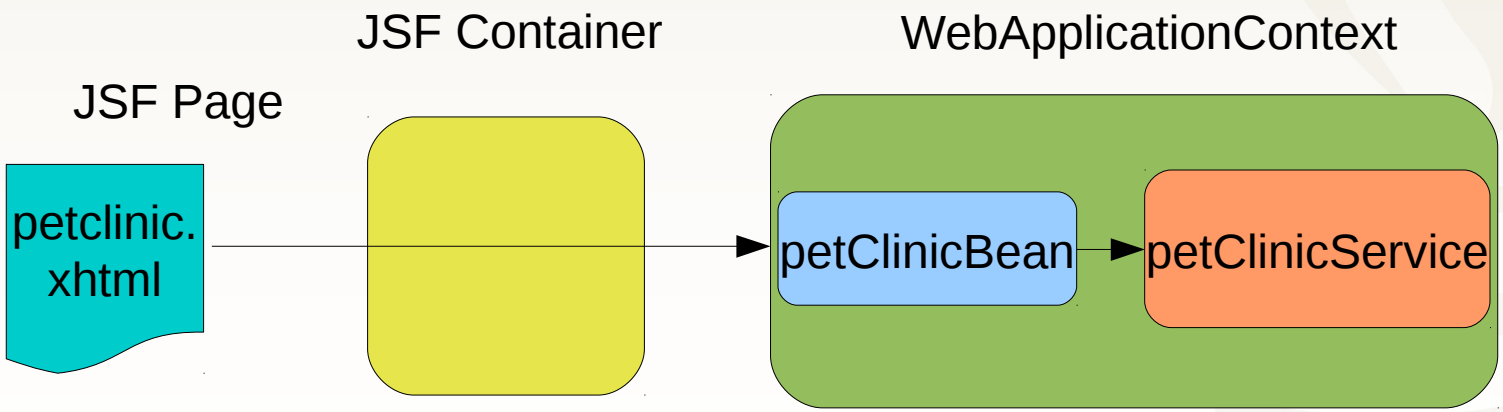
Spring JSF Entegrasyonu

Spring JSF Entegrasyonu

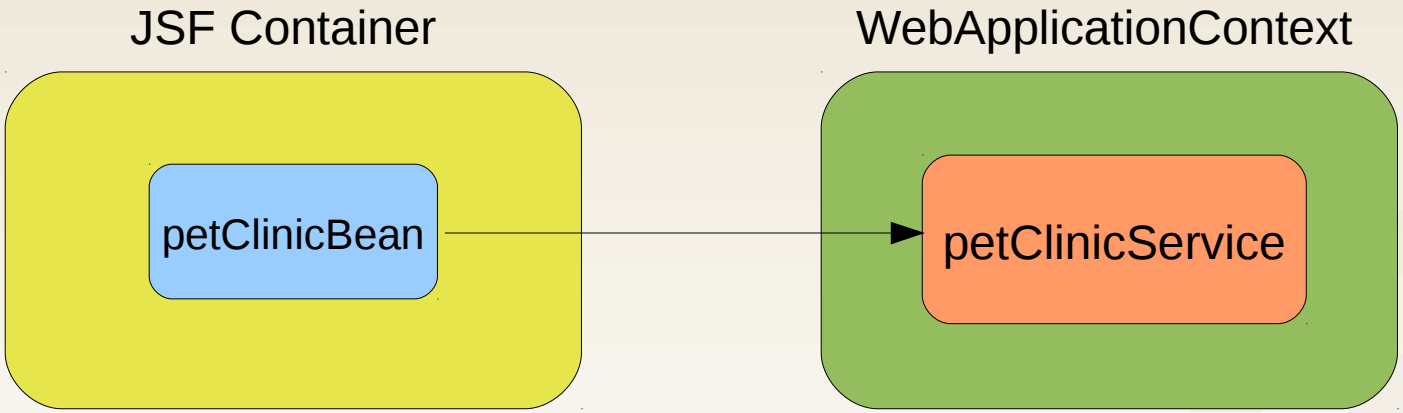
- Spring JSF entegrasyonu ile JSF tarafındaki backing (managed) bean'lerin **managed property'leri Spring Container'dan enjekte edilebilir**
- Ya da **backing (managed) bean'ler** JSF Container yerine **Spring tarafında yönetilebilir**
- Böylece JSF backing bean'lerin **Spring Container'ın sunduğu kabiliyetler**den yararlanması sağlanabilir

Spring JSF Entegrasyonu





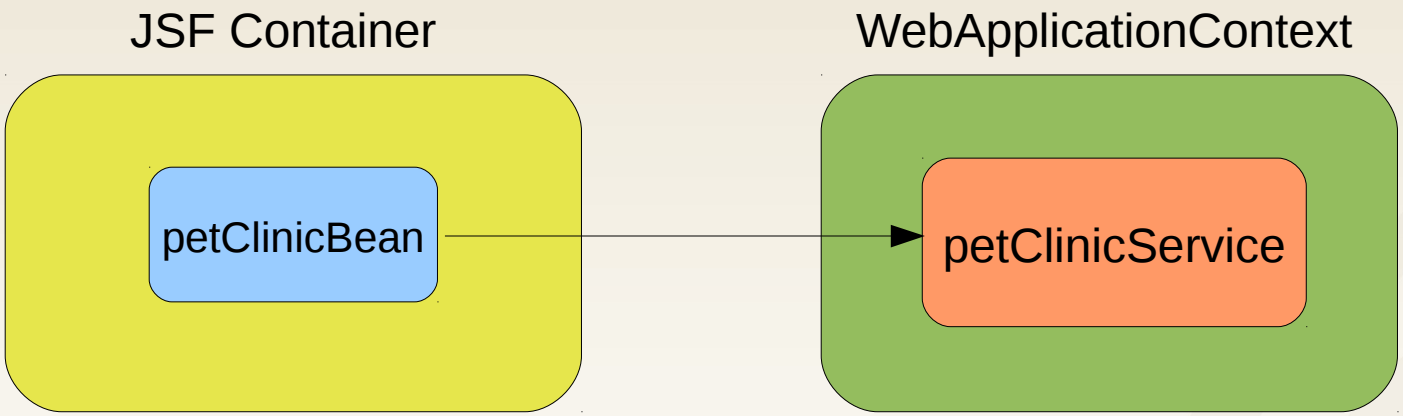
Spring JSF Entegrasyonu




```
<managed-bean>
  <managed-bean-name>
     petClinicBean
  </managed-bean-name>
  <managed-bean-class>
     x.y.z.PetClinicBean
  </managed-bean-class>
  <managed-property>
    <property-name>
     petClinicService
    </property-name>
    <value>#{petClinicService}</value>
  </managed-property>
</managed-bean>
```

```
<bean id="petClinicService"
class="x.y.z.PetClinicServiceImpl">
  <property name="petClinicDao"
ref="petClinicDao"/>
</bean>
```

Spring JSF Entegrasyonu




```
@ManagedBean
@ViewScoped
public class PetClinicBean {

    @ManagedProperty(value = "#{petClinicService}")
    private PetClinicService petClinicService;

    public void setPetClinicService(
              PetClinicService petClinicService) {
         this.petClinicService = petClinicService;
    }
    ...
}
```

```
<bean id="petClinicService"
class="x.y.z.PetClinicServiceImpl">
  <property name="petClinicDao"
ref="petClinicDao"/>
</bean>
```

ELResolver Konfigürasyonu

- JSF bean'lerin Spring tarafında çözümlenebilmesi için **faces-config.xml**'de bir **el-resolver** tanımlamak gerekir
- **SpringBeanFacesELResolver** JSF managed property ve beanlarını resolve etmek için **önce WebApplicationContext**'e **ardından JSF context**'e bakar

```
<faces-config>
  <application>
    <el-resolver>
org.springframework.web.jsf.el.SpringBeanFacesELResolver
    </el-resolver>
  </application>
</faces-config>
```

JSF ve Scoped Bean'lar

- JSF managed bean'ların Spring container içerisinde tanımlanması durumunda **JSF scope'larının Spring tarafından desteklenmesi** gerekir
- **Request**, **session** ve **application scope'ların** Spring tarafında bire bir **karşılıkları var**dır
- Scoped JSF backing bean'ların Spring scoped bean'lar olarak tanımlanabilmesi ve sağlıklı çalışabilmeleri için **web.xml'de RequestContextListener** konfigürasyonuna ihtiyaç vardır

Scoped Bean'lar için Konfigürasyon

```
<web-app>
   <listener>
      <listener-class>
org.springframework.web.context.request.RequestContextListener
      </listener-class>
   </listener>
</web-app>
```

Custom View Scope Konfigürasyonu

- **Request**, **session** ve **application scope'ların** Spring tarafında bire bir **karşılıkları var**dır
- Ancak JSF **flash** ve **view scope**'a karşılık gelen scope tanımları Spring tarafında **mevcut değil**dir
- JSF uygulamalarında "**view scope**" yaygın biçimde kullanılmaktadır
- Dolayısı ile **custom bir view scope tanımı**nın Spring tarafında yapılması gereklidir

Custom View Scope Konfigürasyonu

- Custom scope kabiliyeti sayesinde Spring tarafında da **view scope** tanımı yapılabilir
- Bu sayede view scope bean'ların da Spring Container'ın kabiliyetlerinden yararlanmaları mümkün hale gelir
- İnternet ortamında değişik **view scope implemantasyonları** mevcuttur
- http://www.harezmi.com.tr/blogpost411-Spring-View-Scope-For-JSF-2-Users adresinde örnek bir implemantasyonu bulabilirsiniz

Custom View Scope Konfigürasyonu

- Custom view scope sınıflarının çalışabilmesi için Spring ve JSF tarafında ayrı ayrı ayarlar yapılmalıdır
- **web.xml** içerisinde **RequestContextListener** tanımı yapılmalıdır
- Spring Container içerisinde de "**custom view scope**" tanımı yapılmalıdır

```
<beans>
    <bean class="org.springframework.beans.factory.config.CustomScopeConfigurer">
        <property name="scopes">
            <map>
                <entry key="view">
                    <bean class="tr.com.harezmi.jsf.ViewScope" />
                </entry>
            </map>
        </property>
    </bean>
</beans>
```

Custom View Scope Konfigürasyonu

- View scope sonlandığı vakit çalışması istenen callback'lerin register edildiği **viewMap**'in ilgili **UIViewRoot** instance içerisinde yönetilmesi için **faces-config.xml** içersinde de aşağıdaki tanımların yapılması önemlidir

```
<faces-config>
    <application>
        <system-event-listener>
            <system-event-listener-
class>tr.com.harezmi.jsf.ViewScopeCallbackRegistrar</system-event-listener-class>
            <system-event-class>javax.faces.event.PostConstructViewMapEvent</system-
event-class>
            <source-class>javax.faces.component.UIViewRoot</source-class>
        </system-event-listener>

        <system-event-listener>
            <system-event-listener-
class>tr.com.harezmi.jsf.ViewScopeCallbackRegistrar</system-event-listener-class>
            <system-event-class>javax.faces.event.PreDestroyViewMapEvent</system-
event-class>
            <source-class>javax.faces.component.UIViewRoot</source-class>
        </system-event-listener>
    </application>
</faces-config>
```

Custom View Scope Kullanımı

- Spring scoped bean'ler XML, Java veya anotasyon tabanlı tanımlanabilir
- XML tabanlı konfigürasyonda scoped bean tanımına `<aop:scoped-proxy/>` elemanının da eklenmesi önemlidir
- Anotasyon tabanlıda ise ya `<context:component-scan/>` düzeyinde yada **@Scope** anotasyonunda proxy modu tanımlanmalıdır

Custom View Scope Kullanımı (Anotasyon)

```
<context:component-scan base-package="com.javaegitimleri.petclinic"
    scoped-proxy="interfaces">
</context:component-scan>
```

Değeri “interfaces” ise JDK tabanlı proxy “targetClass” ise CGLIB sınıf tabanlı proxy üretilir

```
@Component
@Scope(scopeName="view",proxyMode=ScopedProxyMode.TARGET_CLASS)
public class CreateVetBackingBean {
    @Autowired
    private PetClinicService petClinicService;
    ...
}
```

component-scan elemanı düzeyindeki konfigürasyonu bean düzeyinde özelleştirmeye imkan tanır

Custom View Scope Kullanımı (XML)

```
<bean class="com.javaegitimleri.petclinic.web.jsf.CreateVetBackingBean"
scope="view">
    <property name="petClinicService" ref="petClinicService"/>
    <aop:scoped-proxy/>
</bean>
```

Tanımlanmadığı takdirde scoped bean için proxy yaratılmaz. Eğer scoped bean başka bir bean'e enjekte edilecek ise proxy'nin yaratılması şarttır

İletişim

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