**1. Define the Custom Scope Annotation**

First, you need to define an annotation to represent your custom scope:

java

Copy code

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

import jakarta.enterprise.context.NormalScope;

@NormalScope

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.TYPE, ElementType.METHOD, ElementType.FIELD})

public @interface TabScoped {

}

**2. Implement the Custom Scope Context**

Next, you implement the Context interface for your custom scope. This class will manage the lifecycle of beans in the TabScope.

java

Copy code

import jakarta.enterprise.context.spi.Context;

import jakarta.enterprise.context.spi.Contextual;

import jakarta.enterprise.context.spi.CreationalContext;

import jakarta.faces.context.FacesContext;

import java.util.HashMap;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

public class TabScopeContext implements Context {

private final Map<String, Map<Contextual<?>, Object>> tabScopedBeanStorage = new ConcurrentHashMap<>();

private final Map<String, Map<Contextual<?>, CreationalContext<?>>> tabScopedCreationalContextStorage = new ConcurrentHashMap<>();

@Override

public Class<? extends Annotation> getScope() {

return TabScoped.class;

}

@Override

public <T> T get(Contextual<T> contextual) {

FacesContext facesContext = FacesContext.getCurrentInstance();

String tabId = getTabId(facesContext);

if (tabId == null) {

return null;

}

Map<Contextual<?>, Object> tabMap = tabScopedBeanStorage.get(tabId);

return tabMap != null ? (T) tabMap.get(contextual) : null;

}

@Override

public <T> T get(Contextual<T> contextual, CreationalContext<T> creationalContext) {

FacesContext facesContext = FacesContext.getCurrentInstance();

String tabId = getTabId(facesContext);

if (tabId == null) {

return null;

}

Map<Contextual<?>, Object> tabMap = tabScopedBeanStorage.computeIfAbsent(tabId, k -> new HashMap<>());

Map<Contextual<?>, CreationalContext<?>> tabCreationalMap = tabScopedCreationalContextStorage.computeIfAbsent(tabId, k -> new HashMap<>());

T bean = (T) tabMap.get(contextual);

if (bean == null) {

bean = contextual.create(creationalContext);

tabMap.put(contextual, bean);

tabCreationalMap.put(contextual, creationalContext);

}

return bean;

}

@Override

public boolean isActive() {

return FacesContext.getCurrentInstance() != null;

}

private String getTabId(FacesContext facesContext) {

// Retrieve the tab ID from the FacesContext or some other context

// For example, from a request parameter:

return facesContext.getExternalContext().getRequestParameterMap().get("tabId");

}

}

**3. Register the Custom Scope Context**

Register the custom scope context with the application by creating an extension class:

java

Copy code

import jakarta.enterprise.context.spi.AlterableContext;

import jakarta.enterprise.context.spi.Context;

import jakarta.enterprise.inject.spi.AfterBeanDiscovery;

import jakarta.enterprise.inject.spi.BeanManager;

import jakarta.enterprise.inject.spi.Extension;

public class TabScopeExtension implements Extension {

public void afterBeanDiscovery(AfterBeanDiscovery event, BeanManager beanManager) {

Context tabScopeContext = new TabScopeContext();

event.addContext(tabScopeContext);

}

}

**4. Define a Beans XML File**

Ensure your CDI container scans and registers the custom scope by defining a beans.xml file:

xml

Copy code

<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/beans\_2\_0.xsd"

version="2.0">

<scan>

<exclude name="com.example.yourpackage.TabScopeExtension" />

</scan>

</beans>

**5. Use the Custom Scope in Your Beans**

Finally, you can use the @TabScoped annotation in your beans:

java

Copy code

import jakarta.inject.Named;

@Named

@TabScoped

public class TabBean {

private String data;

// Getters and setters

}

**6. Pass the Tab ID in Your JSF Pages**

Ensure the tab ID is passed in the request, for example through a URL parameter:

html

Copy code

<h:form>

<h:commandLink value="Open Tab 1" action="#{tabBean.doSomething}">

<f:param name="tabId" value="tab1" />

</h:commandLink>

<h:commandLink value="Open Tab 2" action="#{tabBean.doSomething}">

<f:param name="tabId" value="tab2" />

</h:commandLink>

</h:form>

By following these steps, you can create and use a custom TabScope in your JSF application, enabling beans to be scoped to a specific tab identified by a unique tab ID.