

Scopes

JSF 1.0

request, session, application

- JSF 2.0 and 2.1 scopes added
view, none, custom

- JSF 2.2 scopes
added flow-scope

Request Scope: Interpretation

Meaning

- A new instance of the bean is created for every HTTP request, regardless of whether it is the same user or the same page. This is the most commonly used scope in all of JSF (session scope is second-most common).

- Annotation

- @RequestScoped
- Request scope is default, so developers may omit the annotation

Request Scope: Example

- Java

```
@Named(value = "bankingBean")
```

```
@RequestScoped
```

```
public class BankingBean
```

```
{
```

```
...
```

```
...
```

```
}
```

Facelets(JSF page)

```
<h:inputText value="#{bankBean.customerId}"/>
```

- Behavior

Bean is **instantiated twice** for each submission: once when form is displayed (and `getCustomerId` is called) and again when form is submitted (and textfield value is passed to `setCustomerId`).

- If same bean name appears on a different page, **different instances** are used.

Application Scope: Interpretation

- **Meaning**

- The bean is instantiated the first time any page with that bean name is accessed. From then on, the same bean instance is used, even if it is different user or different page. However, different Web apps are independent.

- Never use application scope for beans that contain user data! Testing on your local machine with a single user might show correct results, but with multiple simultaneous users, you have race conditions with one user's data overwriting another's.

- **Annotation**

```
@Named(value = "colorOptions")
@ApplicationScoped // Slightly better than making data static
public class ColorOptions
{
    private String[] colorNames =
    {
        "red", "orange", "yellow", "green", "blue", "purple"
    };
}
```

Facelets (JSF page)

```
#{colorOptions .colorNames }
```

Behavior

The first time this page (or any page with that bean name) is accessed, ColorOptions bean is instantiated. From then on, the same bean instance is used for all users and on all pages that use that bean name.

Session Scope: Interpretation

- **Meaning**

- The bean is instantiated the first time any page with that bean name is accessed by a particular user. From then on, the same bean instance is used if it is same bean name and same user, even if it is different page.

Different users get different instances. User determined by JSESSIONID -- a cookie(usually) or by jsessionid URL parameter (sometimes).

- Second-most common scope, after request scope.
- Used for accumulating data over time (shopping carts, questions on exams).
- The bean should be Serializable so that session data can live across server restarts and so that on server, session data can be shared between nodes.

Session Scope:

Example

```
@Named(value = "quizBean")
```

```
@SessionScoped
```

```
public class QuizBean implements Serializable
```

```
{
```

```
    private ArrayList<Problem> problems = new ArrayList<Problem>();
```

```
    private int currentIndexProblem;
```

```
    private int score;
```

```
    ...
```

```
}
```

- Facelets

```
#{quizbean.score}
```

- Behavior

- Bean is instantiated first time a particular user accesses any page that refers to that bean name.

- Same instance is used for that user on all pages that use that bean name.

- Annotation

- @SessionScoped

quizbean.score : The score is RETAINED throughout the quiz.

Annotations

@RequestScoped

Default. Make a new instance for every HTTP request. Since beans are also used for initial values in input form, this means bean is generally instantiated twice (once when form is displayed, once when form is submitted).

@SessionScoped

- If same user with same cookie (JSESSIONID) returns before session timeout, same bean instance is used. You should make bean Serializable.

@ApplicationScoped

Shared by all users. Bean either should have no mutable state or you must carefully synchronize access. Usually immutable.

@ViewScoped

Same bean instance is used as long as same user is on same page . Bean should implement Serializable

@FlowScoped

Same bean instance is used as long as it is same user on same set of pages

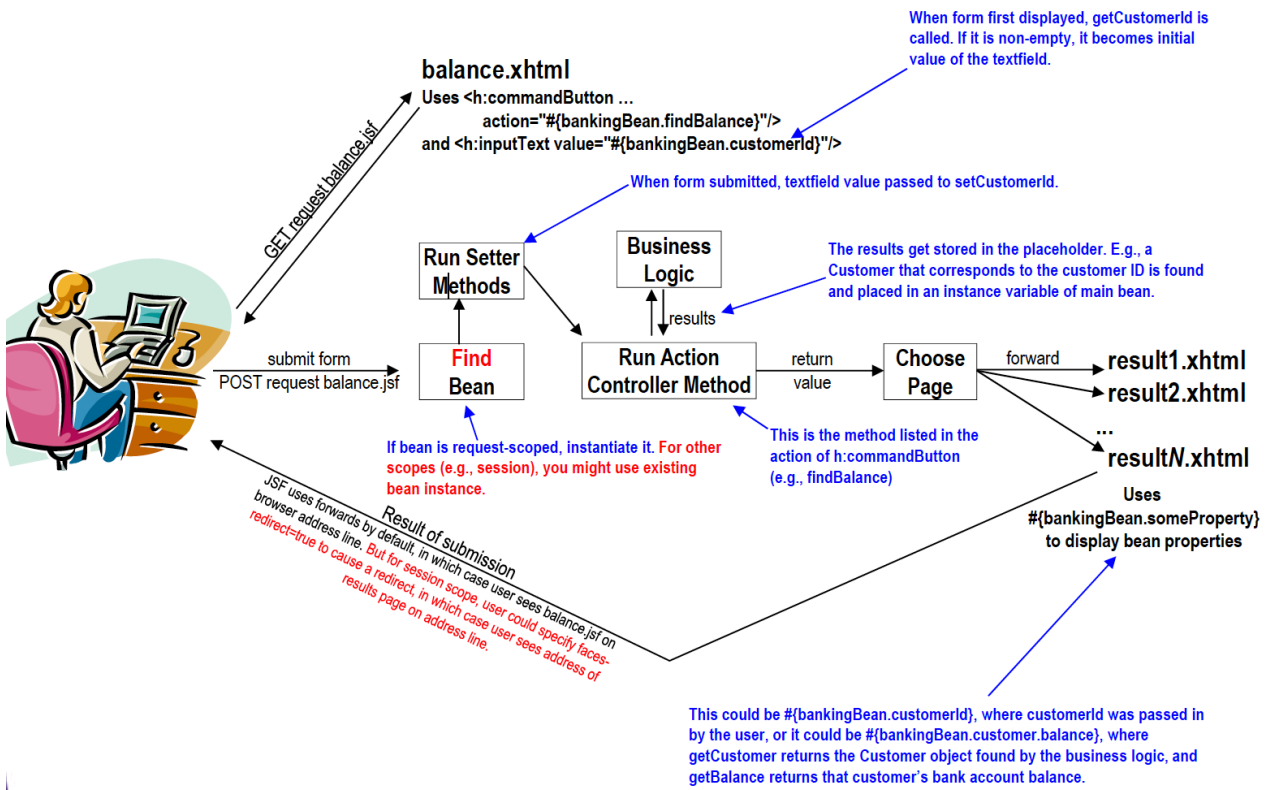
@CustomScoped(value="#{someMap}")

Bean is stored in Map; programmer can control lifecycle

@NoneScoped

Bean is not placed in a scope. Useful for beans that are referenced by other beans that are in scopes

JSF Flow of Control



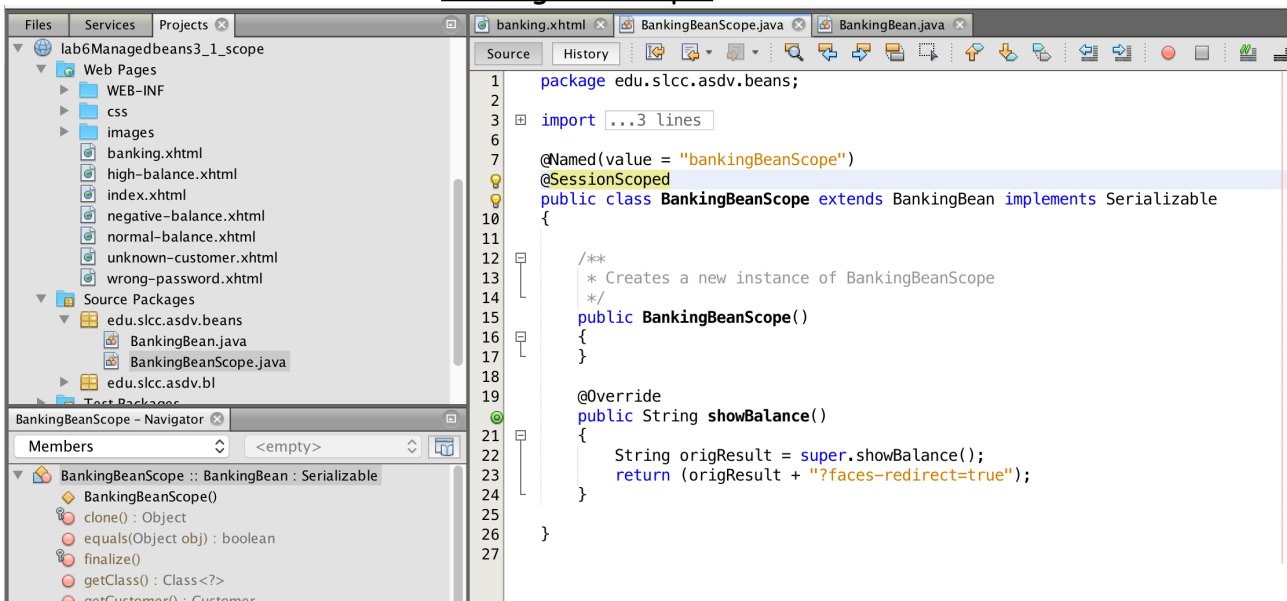
Session Scope: Example

- A variation of session-scoped banking example using redirects instead of forwards
- names of results pages are exposed to end users, who can bookmark them and navigate to them directly.
- This is extra work, because you must consider situation where user follows bookmark in new session, when there is no stored data. However, point is that this is possible with session data, but not with request data.

• What we need

- Add `faces-redirect=true` to end of return values, to tell JSF to redirect (instead of forward) to results pages
- Allows users to access results pages directly

1. Copy and paste the Banking Lab (lab3) and rename it: `lab6Managedbeans3_1_scope`
2. Create a new JSF Bean `BankingBeanScope`:



```
1 package edu.slcc.asdv.beans;
2
3 import ...3 lines
4
5
6
7 @Named(value = "bankingBeanScope")
8 @SessionScoped
9 public class BankingBeanScope extends BankingBean implements Serializable
10 {
11
12     /**
13      * Creates a new instance of BankingBeanScope
14      */
15     public BankingBeanScope()
16     {
17     }
18
19     @Override
20     public String showBalance()
21     {
22         String origResult = super.showBalance();
23         return (origResult + "?faces-redirect=true");
24     }
25
26 }
27
```


3. Modify the JSF banking.xhtml to access the scoped bean.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Banking</title>
  </h:head>
  <h:body>
    <h:form>
      Legal ids are id001, id002, and id003. Password is "secret".<br/><br/>
      Customer ID: <h:inputText value="#{bankingBeanScope.customerId}"/><br/>
      Password: <h:inputSecret value="#{bankingBeanScope.password}"/><br/>
      <h:commandButton value="Show Current Balance"
        action="#{bankingBeanScope.showBalance}"/>
    </h:form>
  </h:body>
</html>
```

4. Modify the high-balance.xhtml to access the bankingBeanScope

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Banking</title>
  </h:head>
  <h:body>
    <h:form>
      Legal ids are id001, id002, and id003. Password is "secret".<br/><br/>
      Customer ID: <h:inputText value="#{bankingBeanScope.customerId}"/><br/>
      Password: <h:inputSecret value="#{bankingBeanScope.password}"/><br/>
      <h:commandButton value="Show Current Balance"
        action="#{bankingBeanScope.showBalance}"/>
    </h:form>
  </h:body>
</html>
```

5. Clean build and run. You will see the redirection in your browser.