ploinFaces

Robert Reiz

A presentation is no documentation!
A presentation just should support the speaker!

Table of Contents

Problem

- A flow should be realized. A flow can have N sites. For example, a registration can have 4 sites. Base-data (site 1), survey-data (site 2), contracts (site 3) and the Liability (site 4).

- In the end of the flow, after the customer have accept the contracts and the liability, the input-data should be written in the database.

- You have to shift the input-data from site 1 and 2 to the end of the flow.

The Request-Solution

- The input-data will be pass from one site to the other in hidden fields.

- The input-data from site 1 will be validate if you are going from site 1 to site 2 and on site 2 they are saved in input hidden fields.

- For a Hacker it is very easy to change the data in the input hidden fields.

The Request-Solution

- To protect your data in the hidden fields you have to validate it every time you are going to the next site.

- If there is an error by going from one site to another site, than all data in the input hidden fields are lost and the customer have to start the flow by site one.

The Session-Solution

- The input data will be stored in the HTTP-Session on the server.
- A Hacker have no access to the data in the HTTP-Session.
- In the end of the flow the input data will be stored in the database an it will be removed from the HTTP-Session.

The Session-Solution

- If the customer leave the flow before the “last site”/”end of the flow”, the input data stay in the HTTP-Session until the session will be end in a timeout.
- Often this is the reason for memory and performace problems.
- If there are to many not used objects in the HTTP-Session and there are to many parallel HTTP-Session you can get an OutOfMemmoryException and the whole Application is down.

More Scopes

- Seam, offers a Conversation Scope.
- Orchestra, offers a Conversation Scope.
- SpringWebFlow, offers Conversation Scope.
- Trinidad, offer a PageFlowScope.

Disadvantages of no-standard-scopes

- Every no-standard-scope have a very strong binding to a special container. You need always a container who are offering you the no-standard-scope.

- Example. You can **not** use the conversation scope from SpringWebFlow without the Spring IoC Container, because just this container offers you this special scope.

- The lifetime of the PageFlowScope from Trinidad is bind to the lifetime of the window. If your Application is running in just one window, the PageFlowScope is the same like the HTTP-Session-Scope.

ploinFaces need no more Scopes to define a flow.
Here are most important Features.

- Flow-Definition threw SessionHandling
- No more Scope
- No technology/container binding
- ploinFaces is very lightweight (40 KB)
- It is customized for JSF

The Framework is:

- OpenSource
- Apache 2.0 Lizenz
- Published on http://www.ploinfaces.org
- Download: http://sourceforge.net/projects/ploinfaces/

Configuration

faces-config.xml

```
<application>
   <navigation-handler>
      org.ploin.web.faces.core.BackNavigationHandler
   </navigation-handler>
</application>
```

web.xml

```
<listener>
    <listener-class>org.ploin.web.faces.core.PloinFacesListener</listener-class>
</listener>
```

Thats it. Your are ready with the configuration! Its a breeze!

Flow

- The managedBean you are using in a flow, should be in the session scope. PloinFaces is not responsible for the cration of the beans. For the creation and managing of your managedBeans you can use the standard JSF-Bean-Container configured in the faces-config.xml.

```
<managed-bean>
        <managed-bean-name>logInOutBean</managed-bean-name>
        <managed-bean-class>org.company.project.java.gui.model.LogInOutBean</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Flow

- The framework expect to find a flow-definition in the file ploinFlows.xml, in the root-ClassLoader. The root-ClassLoader is the directory where normaly your *.java files are placed and your log4j.properties.

Flow

```
<flows xmlns="http://www.ploinfaces.org/schema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.ploinfaces.org/schema http://www.ploinfaces.org/schema/ploinFlows_1.4.xml.xsd">

    <flow id=”loginFlow”>
        <views>
            <view>/login.xhtml</view>
            <view>/page/help/agbLogin.xhtml</view>
            <view>/page/help/haftungLogin.xhtml</view>
        </views>
        <attributes>
            <attribute>logInOutBean</attribute>
        </attributes>
    </flow>

</flows>
```

Flow

You can use java-regex for the views.
The Useradministration can have N sites.

```
<flows  xmlns="http://www.ploinfaces.org/schema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.ploinfaces.org/schema http://www.ploinfaces.org/schema/ploinFlows.xml_1.4.xsd" >

    <flow id=”useradminFlow”>
      <views>
         <view>/page/useradmin/useradministration.*</view>
      </views>
      <attributes>
         <attribute>useradminstrationBean</attribute>
      </attributes>
    </flow>

</flows>
```

Flow

- A flow can be entered with every site and it can leaved from every site.

- If you leave a flow, every attributes who are defined for the flow will be removed from the HTTP-Session.

Flow

- With the tag <viewsForAllFlows> you can define views should be in all your flows.

- This is really a nice feature if you have custom error pages

Flow

```
<flows xmlns="http://www.ploinfaces.org/schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ploinfaces.org/schema http://www.ploinfaces.org/schema/ploinFlows.xml_1.4.xsd">

  <viewsForAllFlows>
     <view>/page/error/.*</view>
  </viewsForAllFlows>

  <flow id=”useradminFlow”>

    <views>
      <view>/page/useradmin.*</view>
    </views>

    <attributes>
      <attribute>useradminBean</attribute>
      <attribute>secureBean</attribute>
    </attributes>

  </flow>

</flows>
```

Security

```
<flows xmlns="http://www.ploinfaces.org/schema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.ploinfaces.org/schema http://www.ploinfaces.org/schema/ploinFlows.xml_1.4.xsd" >

   <authoritySource>#{sessionBean.loginUserRole}</authoritySource>

   <accessDeniedPage>/page/accessdenied.xhtml</accessDeniedPage>

   <flow id=”useradminFlow”>
       <views>
           <view>/page/useradmin/useradministration.*</view>
       </views>
       <attributes>
           <attribute>useradminstrationBean</attribute>
       </attributes>
       <includeAuthoritys>
           <authority>Administrator</authority>
           <authority>GIS-Mitarbeiter</authority>
       </includeAuthoritys>
   </flow>

</flows>
```

Validatores

faces-config.xml

```
<validator>
    <validator-id>htmlKicker</validator-id>
    <validator-class>org.ploin.web.faces.validator.HtmlKicker</validator-class>
</validator>
```

View with xhtml or JSP

```
<ice:inputText id="firstName"
               value="#{profileBean.user.firstName}"
               required="true"
               validator="htmlKicker"/>
```

Customized Message from the standard ResourceBundle

```
ploin.htmlKicker
```

Validatores

In the same fashion you can use all PloinValidatores.

- HtmlKicker (protect for HTML injection)
- EmailValidator
- NumberOnlyValidator
- PlzValidator (just 5 Numbers, there is no checking for the plausibility.)

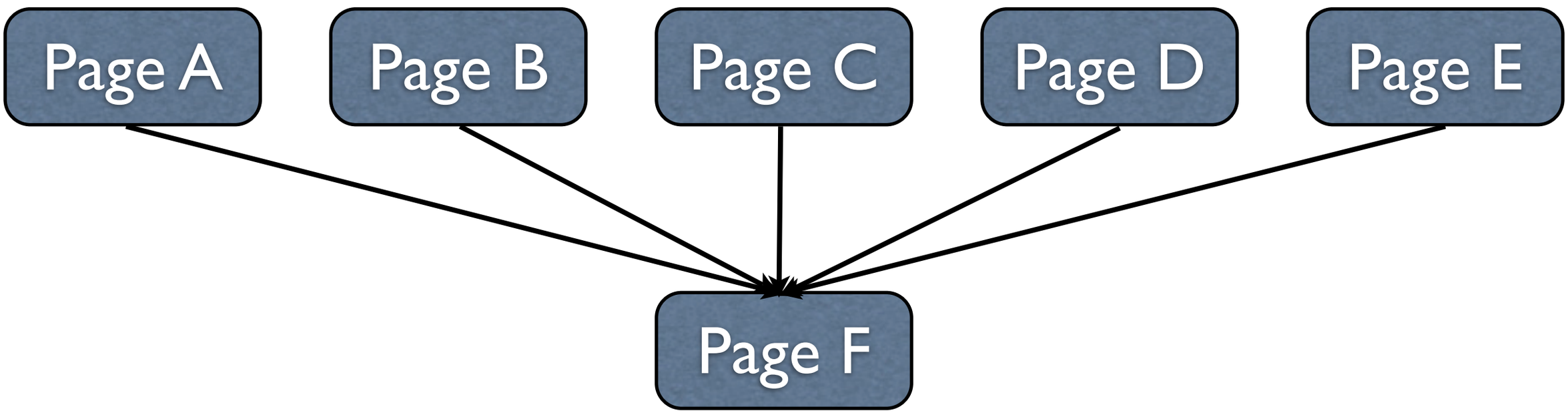
Problem

- The Browser-Back-Button don`t works correct in many JSF and AJAX Applications.
- Your Boss want many times a back-button on the site.

Problem

- The Developer have to look in the faces-config.xml witch outcome he have to use to navigate back.

- But what do you do if you have this construction?



Problem

- You have to store the viewId in your application, from where the customer is coming.
- This CR is coming again and again, and it sucks!

BackNavigation

The PLOIN Solution

```
<ice:commandButton action="back"
                   id="backButton"
                  value="#{labels['back']}" />
```

Just type in “back” in the action, the Framework navigates back to the last view.

BackNavigation

You can navigate two steps back with the outcome “backback”

```
<ice:commandButton action="backback"
                   id="backButton"
                  value="#{labels['back']}" />
```

And three steps back with the outcome “backbackback”

```
<ice:commandButton action="backbackback"
                   id="backButton"
                  value="#{labels['back']}" />
```

Navigation

In JSF you have to define your navigation-rules in the faces-config.xml. If your managedBean returns an outcome, JSF look for a navigation-rule in the faces-config.xml and navigate to the site.

With the new NavigationHandler you don't need navigation-rules in your faces-config.xml. If your outcome starts with "->". ploinFaces look for a viewId named like your outcome without "->". For example, your outcome is "->/page/welcome.xhtml" than ploinFaces will navigate to /page/welcome.xhtml". You can directly navigate from your ManagedBean to your xhtml site, without navigation rules.

```
public String finishSomething(){
   return “->/page/welcome.xhtml”;
}
```

The BaseBean from ploinFaces

```
public FacesContext getFacesContext();

public Application getApplication();

public HttpServletRequest getRequest();

public HttpSession getSession();

public Map getApplicationMap();

public Object getValueFromApplicationMap(Object key);

public Map<String, String> getRequestMap();

public ResourceBundle getResourceBundle();

public String getStringFromResourceBundle(final String key);

public String getStringFromResourceBundle(final String key, String[] params);

public String getStringFromResourceBundleDetail(String key);

public static String getStringFromResourceBundleDetail(String key, final String[] params);

public String getStringFromResourceBundleSummary(String key);
```

```
public void addHTMLMessageFromBundle(FacesMessage.Severity severity, String key);

public void addHTMLMessageFromBundle(String compId, FacesMessage.Severity severity, String key, String[] params)

public void addHTMLMessageFromBundle(String compId, FacesMessage.Severity severity, String key)

public void addHTMLMessageFromBundle(FacesMessage.Severity severity, final String destination, final Sring key);

public void addHTMLMessage(FacesMessage.Severity severity, String summary, String detail);

public void addHTMLMessageFromBundle(FacesMessage.Severity severity, String key, String[] params)

public Lifecycle getCurrentDefaultLifecycleInstance();

public Long getLifecycleId()

public Integer getCurrentPhaseId();

public String getFromViewId();
```

Maven2

If you are using maven2, you can add the ploinFaces dependency to your pom.xml

```
<dependency>
    <groupId>org.ploin.web</groupId>
    <artifactId>ploinFaces</artifactId>
    <version>1.4.1</version>
    <scope>compile</scope>
</dependency>
```

Maven2

To resolve the ploinFaces dependency you have to add the ploinFaces mirror to your ~/.m2/settings.xml

```
<mirror>
  <id>ploinRep</id>
  <name>PLOIN Repository</name>
  <url>http://www.ploin.de:8081/nexus/content/groups/ploinRep/</url>
  <mirrorOf>ploinRep</mirrorOf>
</mirror>
```

Any Questions?