ploinFaces

Robert Reiz

Eine Presentation ist keine Dokumentation!
Eine Presentation soll lediglich den Speaker bei seinem Vortrag unterstützen!

Übersicht

- Problembeschreibung
- ploinFaces
- Konfiguration
- Flows
- Security
- Validatoren
- BackNavigation
- BaseBean

Problem

- Es soll ein Flow realisiert werden. Ein Flow besteht aus N Seiten. Eine Registrierung kann z.B. aus 4 Seiten bestehen. Stammdaten (Seite 1), Umfragedaten (Seite 2), Agbs (Seite 3) und Haftung (Seite 4).

- Erst am Ende des Flows, wenn der Kunde Agbs und Haftung akzeptiert hat, sollen die Daten in die DB geschrieben werden.

- Die Formulardaten von Seite 1 und 2 müssen bis zum Ende des Flows mitgeführt werden.

Die Request-Lösung

- Die Daten werden von Request zu Request in HiddenFields mitgeführt.

- Die Stammdaten werden beim Übergang zur zweiten Seite validiert und dann in HiddenFields auf der zweiten Seite zwischengespeichert.

- Ein Hacker kann die bereits validierten Daten in den HiddenFields ändern.

Die Request-Lösung

- Die Daten in den HiddenFields müssten eigentlich bei jedem Seitenübergang nochmals validiert werden.

- Wenn bei einem Seitenübergang ein Fehler auftritt dann sind alle Daten in den HiddenFields verloren und der Kunde muss den Flow erneut von Seite 1 an ausfüllen.

Die Session-Lösung

- Die Daten aus dem Formular werden in der HTTP-Session auf dem Server geparkt.
- Hacker haben auf einmal validierte Daten keinen Zugriff mehr.
- Am Ende des Flows werden die Daten in die DB geschrieben und aus der HTTP-Session entfernt.

Die Session-Lösung

- Wenn der Kunde den Flow vor dem Ende verlässt bleiben die Daten in der HTTP-Session liegen bis die Session zu ende läuft.
- Dies kann zu massiven Speicher und Performanceproblemen führen.
- Wenn zuviele nicht mehr benötigte Objekte in der Session liegen kann es bei parallelen Zugriffen zu einer OutOfMemmoryException kommen und dadurch ist die ganze App. nicht mehr verfügbar.

Weitere Scopes

- Seam, bietet eine Conversation Scope.
- Orchestra, bietet eine Conversation Scope.
- SpringWebFlow, bietet eine Conversation Scope.
- Trinidad, PageFlowScope.

Nachteile von weiteren Scopes

- Jeder nicht standard-Scope bringt eine sehr starke Technologiebindung mit sich. Es wird ein Container benötigt der den Scope zur Verfügung stellt.

- So kann z.B. der Scope von SpringWebFlow **nicht** ohne den Spring IoC Container benutzt werden, weil nur dieser Container diesen speziellen Scope anbietet.

- Der PageFlowScope von Trinidad lebt solange wie das aktuelle Fenster offen ist. Bei einer App. die komplett in einem Fenster ablaeuft ist der PageFlowScope gleich dem SessionScope.

Aus der eben gezeigten Problematik heraus ist ploinFaces entstanden.
Hier sind die wichtigsten Features.

- Flow-Definition durch SessionHandling
- Kein weiterer Scope
- Technologieunabhängig
- ploinFaces ist ein Leichgewicht (40 KB)
- Masgeschneidert für JSF

Das Framework ist:

- OpenSource
- Apache 2.0 Lizenz
- Veröffentlicht auf SourceForge
- Download: http://sourceforge.net/projects/ploinfaces/

Konfiguration

faces-config.xml

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

Konfiguration

faces-config.xml

```
<lifecycle>
    <phase-listener>
        org.ploin.web.faces.phaselistener.JsfPhaseListener01
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.JsfPhaseListener02
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.JsfPhaseListener03
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.JsfPhaseListener04
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.JsfPhaseListener05
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.JsfPhaseListener06
    </phase-listener>
</lifecycle>
```

Das war es. Die Konfiguration ist abgeschlossen!

Konfiguration

ODER eine alternative config für spring-IoC-Benutzer

faces-config.xml

```
<lifecycle>
    <phase-listener>
        org.ploin.web.faces.phaselistener.SpringPhaseListener01
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.SpringPhaseListener02
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.SpringPhaseListener03
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.SpringPhaseListener04
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.SpringPhaseListener05
    </phase-listener>
    <phase-listener>
        org.ploin.web.faces.phaselistener.SpringPhaseListener06
    </phase-listener>
</lifecycle>
```

Konfiguration

spring config

```
<bean id="JsfPhaseListener01"
      scope="request"
      class="org.ploin.web.faces.phaselistener.JsfPhaseListener01">
</bean>

<bean id="JsfPhaseListener02"
      scope="request"
      class="org.ploin.web.faces.phaselistener.JsfPhaseListener02">
</bean>

<bean id="JsfPhaseListener03"
      scope="request"
      class="org.ploin.web.faces.phaselistener.JsfPhaseListener03">
</bean>

<bean id="JsfPhaseListener04"
      scope="request"
      class="org.ploin.web.faces.phaselistener.JsfPhaseListener04">
</bean>

<bean id="JsfPhaseListener05"
      scope="request"
      class="org.ploin.web.faces.phaselistener.JsfPhaseListener05">
</bean>

<bean id="JsfPhaseListener06"
      scope="request"
      class="org.ploin.web.faces.phaselistener.JsfPhaseListener06">
</bean>
```

Flow

- Die Flow-Definition wird in der Datei ploinFlows.xml eingetragen, die direkt im Root-ClassLoader liegen muss.

Flow

```
<flows>

   <flow>
      <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

Die Views können auch mit java-regex angegeben werden. Wie hier gezeigt.
Die Useradministration kann aus N Seiten bestehen.

```
<flows>

   <flow>
      <views>
         <view>/page/useradmin/useradministration.*</view>
      </views>
      <attributes>
         <attribute>useradminstrationBean</attribute>
      </attributes>
   </flow>

</flows>
```

- Ein Flow kann mit jeder Seite die zum Flow gehört bereten werden und sie kann an jedem beliebigen Punkt wieder verlassen werden.

- Beim verlassen eines Flows werden alle Attribute die zum Flow gehören aus der Session entfernt.

Security

```
<flows>

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

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

</flows>
```

Validatoren

faces-config.xml

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

View mit xhtml oder JSP

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

Customized Message aus dem standard ResourceBundle

```
ploin.htmlKicker
```

Validatoren

Nach dem gleichen Schema können alle PloinValidatoren benutzt werden.

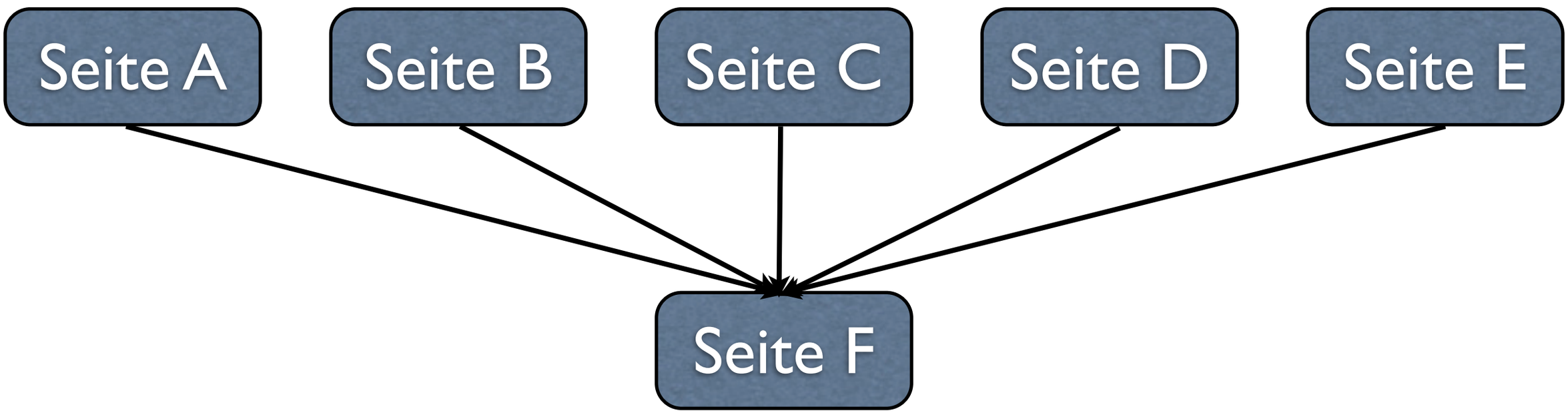
- HtmlKicker (schützt vor HTML injection)
- EmailValidator
- NumberOnlyValidator
- PlzValidator (nur 5 Zahlen, prüft nicht ab ob es die PLZ wirklich gibt. Die Plausibiltaetspruefung kann bei der PLOIN GmbH nachgekauft werden.)

Problem

- Der Browser-Back-Button funktioniert in vielen JSF und AJAX Applikation nicht richtig.
- Der Auftraggeber will deshalb oft einen Zurückbutton auf der Seite haben.

Problem

- Der Entwickler muss in der faces-config.xml nachschauen mit welchem outcome er zurück zur alten view gelangt.
- Was ist aber bei folgender Konstallation?



Problem

- Irgendwo muss in der Applikation vorgehalten werden von wo der Kunde auf die Seite kommt.
- Diese Anforderung tritt immer wieder auf und sie ist lästig!

BackNavigation

Die PLOIN Lösung

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

Als action muss nur “back” eingetragen werden, das Framework leitet automatisch auf die letzte View zurück.

Die BaseBean in 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 getStringFromResourceBundleDetail(String key);

public String getStringFromResourceBundleSummary(String key);

public void addHTMLMessageFromBundle(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 Lifecycle getCurrentDefaultLifecycleInstance();

public Integer getCurrentPhaseId();

public String getFromViewId();
```

Noch Fragen?