

Struts Shale

Von Bernhard Slominski

Struts Shale - Hintergrund

- Aus der Shale Homepage:

„It is time to base a web tier framework on top of the new standard API in this space (JavaServer Faces), and eliminate the need to implement redundant features, instead of just treating JSF as a UI component technology. It is time to answer the question 'if we knew then what we know now, what would Struts have looked like?'

Thus, Shale is a proposal for a modern web application framework, fundamentally based on JavaServer Faces, and focused on improving ease of use for developers adopting JSF as a foundational technology in their own development environments.“

Hintergrund

- "Shale" bedeutet "Schiefer" und damit soll ein Framework bezeichnet werden, das aus mehreren Schichten besteht, die für sich aber unabhängig sein können.
- => Folgerungen
 - Struts classic neigt sich dem Ende zu!
 - Die Zukunft baut auf JSF als Basistechnologie auf!

Status

- Basis: Baut auf JSF 1.1, JSP 2.0, JDK 1.4 auf (alle Technologien sind bereits stabil)
- Lediglich nightly builds
- Kein offizieller Release Termin
- Aber: Die Funktionalität des aktuellen Standes wird wohl weitgehend der Version 1.0 entsprechen
- Craig McClanahan am 2. November (Struts Mailinglist):

„As far as I'm concerned, the feature work I wanted to see for a 1.0.0 release is complete -- it's primarily a matter of being in-country long enough to go through the release process.“

Überblick

- Bestandteile (? = keine Doku)
 - **View Controller**: Lifecycle für managed beans
 - **Dialog Manager**: Ein Dialog wird über verschiedene Stati beschrieben
 - **Application Manager**: ?
 - **Validation**: Integration des "Jakarta Commons Validator Framework", unterstützt Client und serverseitige Validierung
 - **Remoting**: Ajax Unterstützung ?
 - **Spring Integration**: ?
 - **Reusable Views**: Clay Component, um einfach Subviews anzulegen
 - **Test Framework**: Ermöglichen von Unittesting mit Hilfe von Mockobjekten

Beispiele

- Es sollen als Beispiele folgende Komponenten vorgestellt werden:
 - View Controller
 - Dialog Manager
 - Validierung

View Controller

- Interface, das "Lifecycle Events" für managed beans ermöglicht
 - unterstützt ein binäres "postback" Attribut, mit dem festgestellt werden kann, ob die Seite zum ersten Mal aufgerufen wird, oder erneut z.B. durch einen Validierungsfehler
 - **init()** nachdem der zugeordnete View kreiert worden ist
 - **preprocess()** vor dem Setzen der Request Werte (d.h. Füllen des Formulars)
 - **prerender()** direkt vor dem Rendern der Response
 - **destroy()** nach Beendigung des Views

View Controller

- Vorteil
 - Fehlt bisher in der JSF Spezifikation, löst viele Workarounds
- Nachteile:
 - Wird in Lifecycle Annotations in JSF 1.2 enthalten sein (@PostConstruct and @PreDestroy)
 - Durch die Implementierung des (ViewController) Interfaces sind die Managed Beans KEINE POJOs mehr!

Dialog Manager

- In JSF wird der Kontrollfluß durch Navigations-Regeln festgelegt

- JSF-page

```
<h:commandButton label="Login"  
    action="#{loginController.verifyUser}"
```

- faces-config.xml

```
<navigation-rule>  
    <from-view-id>/login.jsp</from-view-id>  
    <navigation-case>  
        <from-outcome>success</from-outcome>  
        <to-view-id>/success.jsp</to-view-id>  
    </navigation-case>  
    <navigation-case>  
        <from-outcome>failure</from-outcome>  
        <to-view-id>/goodbye.jsp</to-view-id>  
    </navigation-case>  
</navigation-rule>
```

Dialog Manager

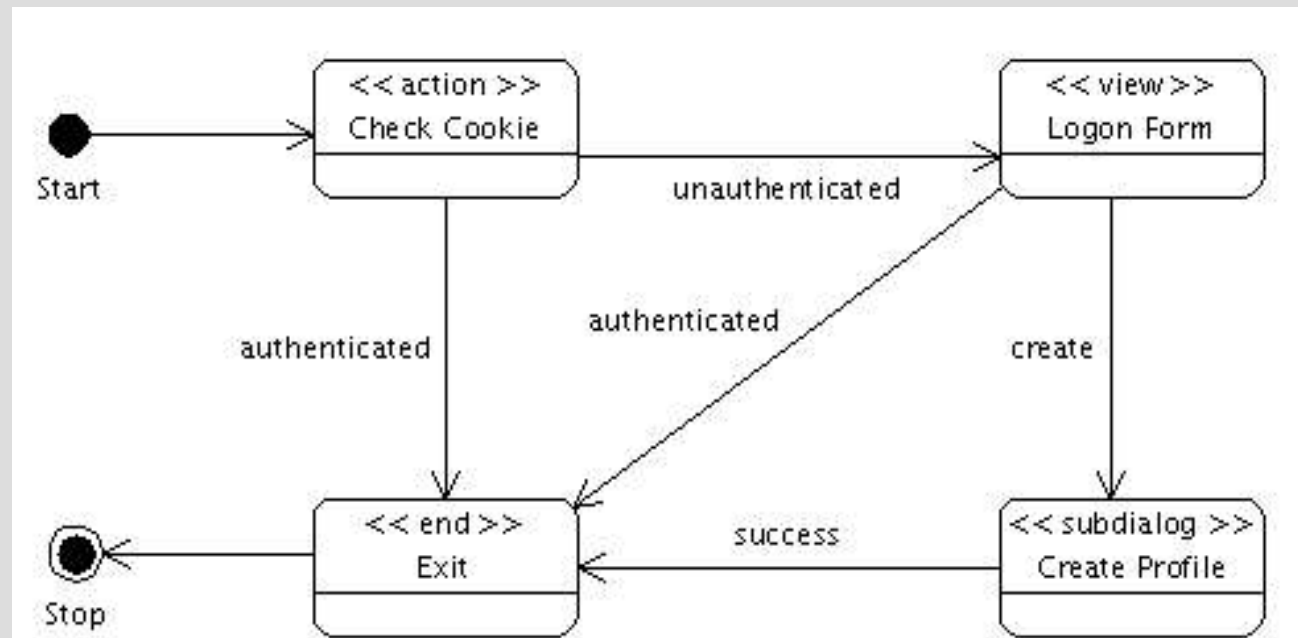
- Nachteile
 - Diese Struktur ist seitenorientiert und nicht dialogorientiert
 - Sie wird dadurch schnell unübersichtlich und schlecht wiederverwendbar
- Lösung: Dialog Manager mit Dialogorientierung
 - Die Lösung wurde stark an "Spring Webflow" angelehnt
 - Ein Dialog wird als Ganzes definiert, er hat eine Reihe von definierten Stati die durch definierte Übergänge erreicht werden
 - Wird typischerweise als UML Status Diagramm visualisiert
 - Die Definition erfolgt (normalerweise) in einem XML File mit Namen dialog-config.xml
 - Nachteil: Visuelle Darstellung des Seitenflusses nicht mehr gewährleistet

Dialog Manager

- Die Stati
 - **Action Status**: Es wird eine Methode aufgerufen, die den Übergang zum nächsten Status steuert
 - **View Status**: Zeigt einen JSF View an und wartet auf einen submit
 - **Subdialog Status**: Startet einen neuen Unterdialog, nach Beendigung des Unterdialogs kehrt die Kontrolle zum aufrufenden Dialog zurück
 - **End Status**: Beendet den aktuellen Dialog

Dialog Manager

- Beispiel



- Aufruf

```
<h:commandLink id="logon" action="dialog:Log On" />
```

Dialog Manager

- **dialog-config.xml**

```
<dialog name="Log On" start="Check Cookie">
  <action name="Check Cookie" method="#{profile$logon.check}">
    <transition outcome="authenticated" target="Exit"/>
    <transition outcome="unauthenticated" target="Logon Form"/>
  </action>

  <view name="Logon Form" viewId="/profile/logon.jsp">
    <transition outcome="authenticated" target="Exit"/>
    <transition outcome="create" target="Create Profile"/>
  </view>

  <subdialog name="Create Profile" dialogName="Edit Profile">
    <transition outcome="success" target="Exit"/>
  </subdialog>

  <end name="Exit" viewId="/usecases.jsp"/>
</dialog>
```

Validierung

- JSF 1.x unterstützt keine explizite clientseitige Validierung, außerdem nur wenige serverseitige Basis-Validatoren
- "Apache Common Validator" hat folgende zusätzliche Validatoren:
 - Kreditkarte
 - Datum
 - EMail
 - ISBN
 - URL

Validierung

- Die Validatoren können Server- oder *Clientseitig* (oder auf beiden Seiten) aufgerufen werden
- Ermöglicht eine (gewisse) Wiederverwendbarkeit der Validatoren
- Aber: JSF 2.0 wird Clientseitige Validatoren unterstützen, auch in MyFaces sind zusätzliche Validatoren eingebaut (allerdings nicht clientseitig)

Validierung

- Benutzung ist einfach:
 - `validator-rules.xml` muß in WEB-INF vorhanden sein
 - Ein `onsubmit` Attribute des `h:form` tags muß hinzugefügt werden
 - Die Validatoren werden mit `s:commonsValidation` zu den Eingabefeldern hinzugefügt
 - Ein Javascript muß mit `s:validatorScript` am Ende des Formulars hinzugefügt werden