Orientation in Objects

Shale
APACHE PROJECT

JSF auf dem Highway

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

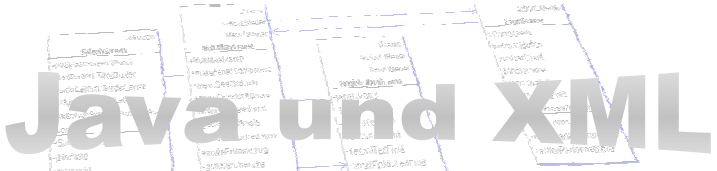
www.oio.de
info@oio.de

Oliver Wolff
<wolff@oio.de>

Version: 0.3

Java, XML und Open Source seit 1998

Orientation in Objects



) Projekte)

- **Schlüsselfertige Realisierung** von Software
- **Unterstützung laufender Projekte**
- **Pilot- und Migrationsprojekte**

) Beratung)

- **Methoden**, **Standards** und **Tools** für die Entwicklung von **offenen, unternehmens-weiten Systemen**

) Akademie)

- **Schulungen**, **Coaching**, **Weiterbildungsberatung**, **Train & Solve**-Programme

Der Sprecher

Orientation in Objects

Ajax mit Java
JavaServer Faces
QS mit Open Source
XML mit Java

Trainer
Berater
Entwickler

Gliederung

Orientation in Objects

- **Einführung**
- Klein, aber Fein
- Große Dinge
- Highway
- Diskussion & Ausblick

Shale
APACHE PROJECT

Herkunft

- Lange Geschichte
- Maßgeblich beteiligt: Craig McClanahan
- Ursprünglich Teil / Erweiterung von Struts:
  - Struts Action Framework / Struts Shale
- Angedacht für Struts 2.0, jedoch verworfen
- Transformation:
  - JSF als Basis
  - Apache Top-Level-Projekt (2006)

Warum Shale?

- „JSF ist ein UI-Framework für Webanwendungen“
- JSF Design:
  - Fokus auf Komponenten-APIs
  - Lebenszyklus nicht perfekt
  - Aber: Alle Framework-Aspekte leicht erweiterbar
  - → Grundlage für Anwendungs-Frameworks

→ JSF = **Komponenten-Framework**
JSF **+ Shale** = **Anwendungs-Framework**

Apache Shale

- Anwendungs-Framework basierend auf JSF
- Menge von Diensten, die nach Bedarf kombiniert werden können
- Dienste sind als dünne Schichten implementiert
- Begriff stammt aus Geologie: Schiefer
- Im Vortrag werden die Dienste vorgestellt

Gliederung

- Einführung
- **Klein, aber Fein**
- Große Dinge
- Highway
- Diskussion & Ausblick

Shale Core

- Basis-Funktionalität
- Einbindung: `shale-core.jar`

- **JNDI Integration**: Zugriff auf JNDI-Context über entsprechende Resolver → Zugriff über Binding

- **Shale Core Taglib**
  - `<s:token />` Hidden Input mit Transaktions-Token
  - `<s:subview>`, `<s:validatorScript>` wurden verlagert

- Utility Classes: Nützliche Helferlein für JavaBeans, Bundles & Co

Shale Tiger Extensions

- Unterstützung für Annotationen

- Ab Java 1.5

- Einbindung: `shale-tiger.jar`

- Vereinfachte Konfiguration von:
  - ManagedBeans
  - ViewController
  - JSF-Komponenten:
    - **Konverter**
    - **Validatoren**
    - **Renderer**

Shale Validator

- Einbinden der Apache Commons Validator
- Einbindung: `shale-validator.jar` + Abhängigkeiten
- Zugriff auf erweiterte Validatoren: Kreditkarten, Email, ISBN , ...
- Wahlweise Server- / Client-Side Validation (Vorsicht)
- Über XML-Konfiguration erweiter- / anpassbar

commons validator

Shale Test Framework

- Hilfsklassen für das Testen von JSF-Anwendungen
- Einbindung: `shale-test.jar`
- Beinhaltet **Mock-Objekte** für:
  - Shale: ViewController
  - Servlet-API
  - JavaServer Faces
- Basisklassen:
  - AbstractJsfTestCase: Bereitstellen von JSF und Servlet-Basisklassen
  - AbstractViewControllerTestCase: Zusätzliche ViewController Helfer

Shale Application Controller - Übersicht

- JSF: **Seiten-orientiert**
  - ManagedBean wird mit einer Seite verknüpft
  - Kein Application-Controller
- Andere Frameworks arbeiten **Action-orientiert**
  - Ein zentraler Controller, durch den **alle** Anfragen laufen
  - Beispiel Struts: RequestProcessor, Commons Chain Pipeline
  - JSF: Nachbildung durch PhaseListener, FrontController
- Einbindung: `shale-application.jar`
- Einbinden als **Servlet Filter**

Shale Application Controller - Verwendung

- Die Prozessierung erfolgt mit Commons Chain
- Hooks für Pre- und Postprocessing
- Setzt eigenen ShaleWebContext, auch für andere Komponenten
- Mitgelieferte Filter:
  - ContextRelativePathFilter: Schutz von Ressourcen
  - RemoteAddrFilter
  - RemoteHostFilter

Gliederung

- Einführung
- Klein, aber Fein
- **Große Dinge**
- Highway
- Diskussion & Ausblick

- **Clay**
- ViewController
- Shale Remoting

Clay - Übersicht

- Alternativer View-Handler für JSF
- HTML ersetzt JSPs
- Tapestry trifft Tiles
  - „Clay unabashedly copies Tapestry‘ s innovations“
- Sehr mächtig, viele Bestandteile und Möglichkeiten
- Einbindung: `shale-clay.jar` + Abhängigkeiten

Clay - View Composition - Übersicht

- Tiles-Style XML-Konfiguration
  - Einzelne Views werden in zentraler XML-Datei definiert
  - Tiles Benutzer fühlen sich schnell zu Hause
- Dynamische Komposition
  - Teilbäume werden dynamisch erzeugt
  - Enge Anbindung an ViewController
- Zusätzliche ManagedBean für dynamische Ausgaben
  `clayOut, clayImport`

Clay - View Composition - Tapestry Style I

- Teilbäume werden mit HTML beschrieben
- HTML-Werkzeuge werden unterstützt
- Prototyp kann in Anwendung transformiert werden
- **jsfid**-Attribut: Dynamische Übersetzung HTML → JSF
- Implizites Mapping von speziellen Elementen
  z.B: `<form>`

Clay - View Composition - Tapestry Style II

- Steuerung des Parsers über Schlüsselworte, z.B: `clay:remove`
- Partial Html-Views
  - JSP ist eigentliche View
  - Teilbaum wird clay-Element eingebunden
- Full Html-View
  - Html-Seite ist Ausgangspunkt
- Erweiterung zu Full Html-View: XML-View

Clay - Component I

- Top-Level-Element, repräsentiert JSF-Komponente

```
<component jsfid="outputText"
  componentType="javax.faces.HtmlOutputText"/>
```

- Vererbung von Eigenschaften analog zu Tiles:

```
<component jsfid="addOut" extends="outputText">
```

- Attribute können ererbt oder überschrieben werden

```
<attributes>
    <set name="style" value="color:blue"/>
</attributes>
```

Clay - Component II

- Element-Knoten
  - Dient dem Zusammenstecken mehrerer Komponenten
  - Attribut: `renderID` gibt die Reihenfolge der Elemente an
  - Attribut: `jsfid` gibt die entsprechende Komponente an

```
<component jsfid="addressPanel"
    extends="panelGrid">
    <attributes>
        <set name="columns" value="2" />
    </attributes>
    <element renderId="1" jsfid="nameLabel"/>
    <element renderId="2" jsfid="nameInput"/>
</component>
```


Clay - Symbols

- Mappen von ManagedBeans auf Symbole / Aliase
- Analog zu MyFaces: t:AliasBean

→ Eine Komponente / View kann wiederverwendet werden

```
<component jsfid="storeCommand" extends="commandButton">
  <attributes>
    <set name="value" value="Save" />
    <set name="action" value="#{@managed-bean-name.save}"/>
  </attributes>
</component>

<clay:clay id="saveAddress" managedBeanName="selected"
  jsfid="storeCommand"/>
```

Gliederung

- Einführung
- Klein, aber Fein
- **Große Dinge**
- Highway
- Diskussion & Ausblick

- Clay
- **ViewController**
- Shale Remoting

ViewController - Übersicht

- **Schön**: ManagedBeans: Keine Basisklassen bei JSF
- **Unschön**: Kein einfacher Zugriff auf Services & Lebenszyklus

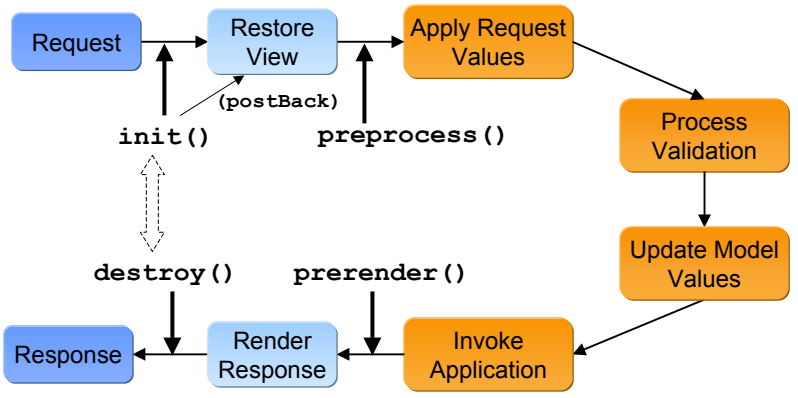
- Probleme:
  - Wo im Lebenszyklus bin ich?
  - Wie und wann greife ich auf Ressourcen zu?
  - Wann räume ich auf?

- Was darf es denn sein?
  - Signal, daß die Bean das erste Mal verwendet wird
  - Signal zum Aufräumen
  - Signale zu gewissen Phasen des Lebenszyklus
  - Signal ob Postback

ViewController - Funktionsweise

- **Hollywood-Prinzip**: „Don‘t call us, we call you“
- JSF-Standard: Eine Request-Scope-Bean / View
- Interface: `org.apache.shale.view.ViewController`
- Einbindung: `shale-view.jar` + Abhängigkeiten

ViewController - Lebenszyklus - Postback



ViewController - AbstractViewController

- Bequeme Basisklasse für ViewController
- boolean postBack & Leer-Implementierungen

- Einige weitere Funktionen von AbstractFacesBean
  - Messages: `info(), error(), warn(), fatal()`
  - Zugriff auf Anwendung, wie `getFacesContext(), getLifecycle()`
  - Zugriff auf Beans, Bindings und andere Daten

- Alternativ: Annotationen
  - @View, @Prerender, @Preprocess
  - @Init, @Destroy

Gliederung

- Einführung
- Klein, aber Fein
- **Große Dinge**
- Highway
- Diskussion & Ausblick

- Clay
- ViewController
- **Shale Remoting**

Shale Remoting - Übersicht I

Orientation in Objects

- Mappen von Server-seitigen Ressourcen
- Zugriff über URLs
- Verarbeitung mit Hilfe von Prozessoren
- Findet in anderen Projekten Verwendung
  - DynaFaces
  - Sun Java Studio Creator
- Grundlage für Ajax

Shale Remoting - Übersicht II

Orientation in Objects

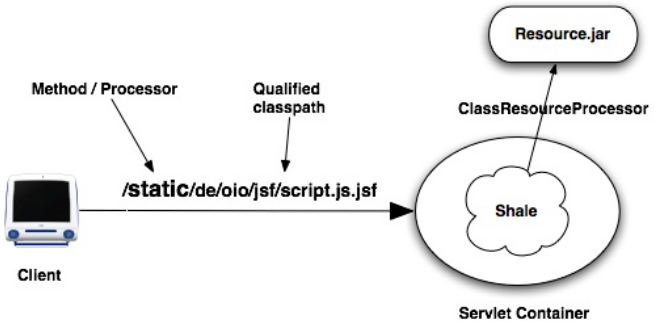
- Einbindung: `shale-remoting.jar`
- Nur JSF als Abhängigkeit, relativ klein (ca. 52k)
- Einbindung als PhaseListener (nach Restore View)
- Über context-Parameter genau konfigurierbar
  - excludes / includes für Ressourcen
  - Pfade
  - Ausführende Prozessoren

Shale Remoting - Processor

- Ziel für Remote-Aufrufe
- Einfaches Interface: `org.apache.shale.remoting.Processor`
- `void process(FacesContext cont,String resId)`
- **Wichtig**: URL muss als Faces-Request verarbeitet werden
- Mitgelieferte Implementierungen
  - ClassResourceProcessor, WebResourceProcessor
  - ChainProcessor, MethodBindingProcessor

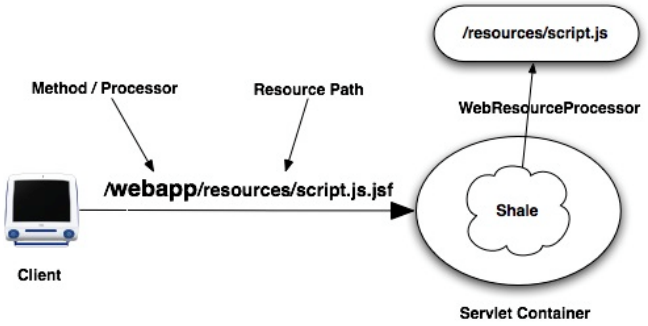
Shale Remoting - Classpath Ressourcen

- `org.apache.shale.remoting.impl.ClassResourceProcessor`
- Zugriff auf Stylesheets, javascript-Dateien,.. aus **Klassenpfad**
- Standard-Ort: `/WEB-INF/lib/, /WEB-INF/classes/`



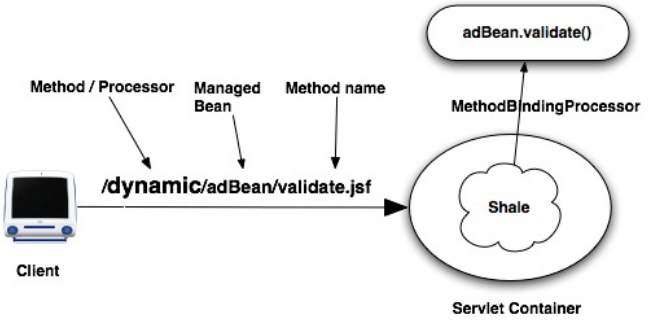
Shale Remoting - Datei-System Ressourcen

- `org.apache.shale.remoting.impl.WebResourceProcessor`
- Zugriff auf Stylesheets, etc,.. aus dem **Datei-System**
- Standard-Ort: `WebAppRoot/`



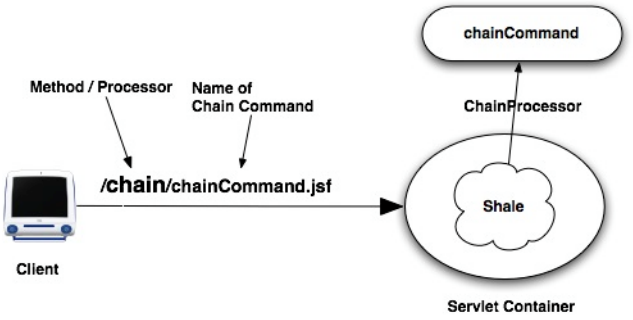
Shale Remoting - MethodBindingProcessor

- `org.apache.shale.remoting.impl.MethodBindingProcessor`
- Ausführen von Funktionen von ManagedBeans
- Die Funktionen müssen void und Parameter-los sein
- Zugriff auf Parameter und Rückgaben erfolgt programmatisch



Shale Remoting - ChainProcessor

- `org.apache.shale.remoting.impl.ChainProcessor`
- Ausführen einer konfigurierten Commons-Chain
- Identifizierung anhand des Namens



Gliederung

- Einführung
- Klein, aber Fein
- Große Dinge
- **Highway**
- Diskussion & Ausblick

Was bisher geschah - Klein, aber Fein

- Shale Core
- Shale Tiger
- Shale Validator
- Shale Test
- Shale Application Controller

Was bisher geschah - Große Dinge

- Clay
  - Mächtiger Ersatz für JSP als View-Handler
  - Ernsthafte Alternative zu Facelets
  - Vor allem für ehemalige Tiles / Tapestry Entwickler
- ViewController
  - Hollywood Prinzip
  - Mehr Kontrolle über den Lebenszyklus
- Shale Remoting
  - Mappen von Server-seitigen Ressourcen
  - Konzept der Prozessoren
  - Grundlage für Ajax

Eine Sache noch ....

Dialoge

Dialoge - JSF

- JSF-Standard: Eine Request-Scope-Bean / View
- Möglichkeiten innerhalb JSF
  - Beschränkung durch Servlet-API: Session - Request
  - Viel in die Session → Session Pollution
  - State in hidden fields? Unhandlich!
- Eine Bean für mehrere Views hintereinander?
- Wiederverwendung von Views?

Shale Dialog - Übersicht

- DialogManager-API: Abstrakte „Execution Engine“
  - Verschiedene Implementierungen verfügbar
- Dialoge werden als Zustände und Übergänge modelliert
- Bestandteile
  - **DialogContext**: Eigener Kontext für Dialoge
  - **DialogContextManager**: Verwalten / Zugriff von DialogContexten
  - **DialogListener**: Listener für Übergänge
- „...heavily inspired by the implementation of Spring Webflow “

Shale Dialog - DialogContext

- Eigener Kontext für Dialoge
  - In der Session, verwaltet durch DialogContextManager
  - Eine Instanz pro Fenster / Frame mit einem Dialog
- Funktionen
  - Start: Anfang eines Dialoges
  - Advance: Zustandsübergang, anhand des String-Rückgabewertes
  - Stop: Ende eines Dialoges, entfernen aus der Session
- Properties
  - active: Dialog wurde gestartet, noch nicht gestoppt
  - data: Ort für Zustandsspeicherung, inital eine Map
  - id, name, parent: Informationen zur Zuordnung

Shale Dialog - Basic Implementation

- Einfache Implementierung mit vier Zuständen

- **Action**: Ausführen einer beliebigen Action-Methode
  - String-Rückgabewert wird für den nächsten Übergang verwendet
- **Subdialog**: Einen Kind-Dialog aufrufen (BlackBox)
  - Der aktuelle Dialog wird auf einen Stack gelegt
  - Wenn der Kind-Dialog zu Ende ist, geht der Übergeordnete weiter
- **View**: Eine View anzeigen (Navigation)
  - Warten auf nächsten String-Rückgabewert
- **Exit**: Dialog beenden

- Einfache DTD für XML-Konfiguration

Shale Dialog - Basic Implementation - Beispiel I

```
<dialog name="LogOn" start="CheckCookie">

<action name="CheckCookie" method="#{profile.check}">
   <transition outcome="authenticated" target="Exit"/>
   <transition outcome="unauthentiC" target="LogonForm"/>
</action>

<view name="LogonForm" viewId="/logon.jsf">
   <transition outcome="authenticated" target="Exit"/>
   <transition outcome="create" target="CreateProfile"/>
   <transition outcome="cancel" target="Exit"/>
</view>
...
```

Shale Dialog - Basic Implementation - Beispiel II

```
<subdialog name="CreateProfile" dialogName="EditProfile">
   <transition outcome="cancel" target="Exit"/>
   <transition outcome="success" target="Exit"/>
</subdialog>
....
<end name="Exit" viewId="/content.jsf"/>
</dialog>
```

- Aufruf
  - Navigation: **dialog:LogOn**, Präfix ist konfigurierbar
  - Programmatisch
  - Für PopUps: URL Parameter

Shale Dialog - SCXML

- Zustände und Übergänge → UML StateChart
- W3C: SCXML StateChart XML
- SCXML-Implementation wertet das Ergebnis direkt aus
- Beispiel ArgoUML, siehe Links
  - Erstellen eines StateChart
  - XSLT-Transformation
  - Benutzen mit SCXML-Implementation
- Einfacher: UML-Werkzeug mit SCXML-Export

Gliederung

- Einführung
- Klein, aber Fein
- Große Dinge
- Highway
- **Diskussion & Ausblick**

Shale im Ganzen

- JSF **+ Shale = Anwendungs-Framework !**
- Erleichterung der Entwicklung
- Beschleunigung
- Verbesserung der Qualität
- Erweiterung der Möglichkeiten
- Implementierungs-Neutral

Alternativen

- Viele Erweiterungen mit ähnlicher Funktionalität
- Facelets
- JBoss Seam
- MyFaces Orchestra
- Spezielle Implementierungen

Ist Shale Tod?

- Erstaunliche Diskussion
- Rückgang der Aktivität
- Politische Implikationen
- Zitat (Shale Mailing List, Greg Reddin 02.08)
  „*We're not like an infant whose arms and legs are always in motion unless he's asleep. We're more like old guys who sit on the porch and don't move unless our beer glass is empty*“

Warum also Shale?

- Elegantes Design
- Getrennte Schichten, spielen jedoch sehr gut zusammen
- Hervorragende Dokumentation
- Gereifte Implementierung, auf dem Höhepunkt
- **Es funktioniert**, v.a. mit JSF 1.2

Ausblick

- Aktuelle Version: 1.04, funktioniert mit JSF 1.2
  - 1.05, 1.10 auf dem (gemächlichen) Weg
- Eventuell mit JSF 2 weniger relevant:
  - Transformation zu Anwendungs-Framework
  - Viel in der Diskussion
  - Jahr 201x?
  - Für aktuelle Projekte keine Relevanz
- **Viel Spaß mit Shale!!!**

Links

- Shale Homepage
  - http://shale.apache.org
- McClanahan über Shale & Co
  - http://www.jsfcentral.com/articles/mcclanahan-05-05.html
- All Hail Shale: Shale isn't Struts
  - http://www.ibm.com/developerworks/library/j-shale0228/index.html
- Visual Editing of Shale Dialogs
  - http://www.kotikone.fi/00/SCXML/

Bilder


Waterhouse-Crystal Ball
http://commons.wikimedia.org/wiki/Image:John_William_Waterhouse_-_The_Crystal_Ball.JPG

Tarot-Karte 13-XIII-arcane
http://commons.wikimedia.org/wiki/Image:13-XIII-arcane_sans_nom.jpg 


Raffael: Die Schule von Athen
http://en.wikipedia.org/wiki/Image:Sanzio_01.jpg 

Puvis de Chavannes, Pierre-Cécile
http://commons.wikimedia.org/wiki/Image:Chavannes_Poor_Fisherman.jpg 


IvanGrohar-Portrait of France Preseren
http://commons.wikimedia.org/wiki/Image:Ivan_Grohar_-_Portrait_of_France_Preseren.jpg

Cassatt Mary Children on the Beach 1884,
http://commons.wikimedia.org/wiki/Image:Cassatt_Mary_Children_on_the_Beach_1884_.jpg 


Sergeant at arms 14. Cnetury, St Catherine du Val des Ecoliers, Paris, France
http://www.gutenberg.org/files/10940/10940-h/images/fig310.png

19. Jh „: A painting of an animal trainer, a lion, a tiger, and 3 hybrid offspring from the 19th century“, unbekannt:
http://commons.wikimedia.org/wiki/Image:Lion-tiger2.jpg 

Alle Bilder sind public domain (commons.wikimedia, Projekt Gutenberg)

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 0.3

Orientation in Objects

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 0.3