



J-Fall

11 oktober 2007 Spant! - Bussum



Controllers

The Next Generation

Duncan Mills

ORACLE®

.nl.
jug



Topics

- Who am I?
- Where JSF let us down
- A look at the players
- Wrap up



About Me...

- Duncan Mills @ Oracle
 - Director of Product Management for JDeveloper and the ADF Framework
 - Previously
 - Heavily involved in the design of ADF Task Flow
 - Responsible for the Struts / JSF Navigation tooling in JDeveloper
 - Still passionate about controller technology
 - (For some strange reason)
 - <http://groundside.com/blog/DuncanMills>



In the Beginning...

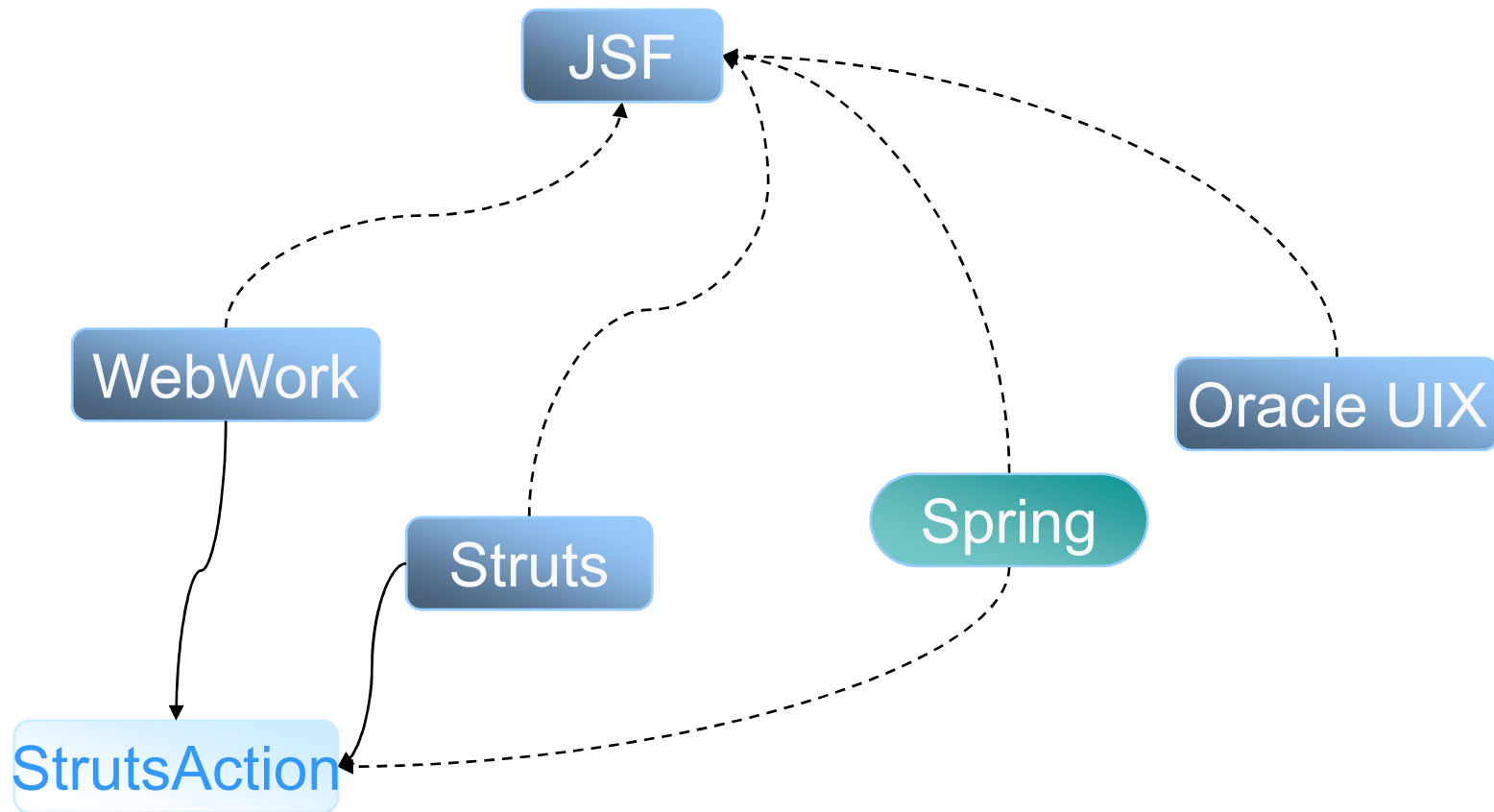


Craig McClanahan

There was Struts...



Along Came JavaServer Faces





JSF's Ugly Secrets

- JSF beats Struts hands-down – right?
 - Yes!
 - Standard
 - Component based
 - Modern patterns
 - Implementation independence
 - Hold on though...?
 - Security
 - Reuse
 - Where do I do stuff?



Enter Shale



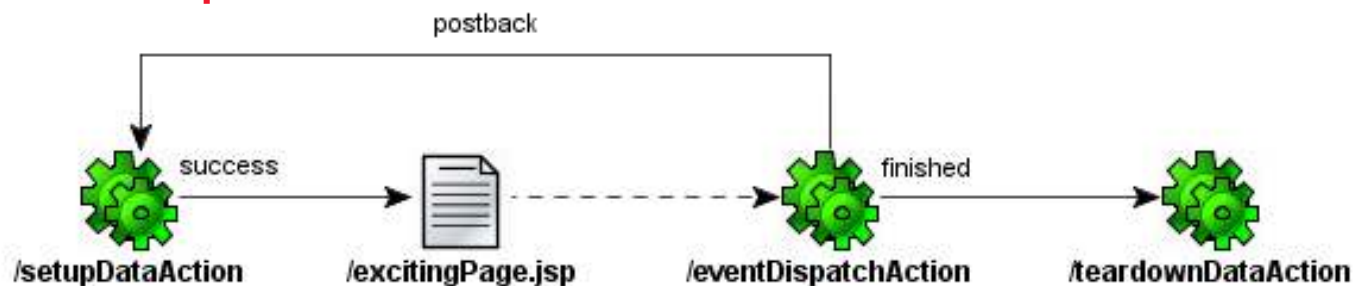
Shale
APACHE PROJECT

- Fairly immediate reaction to JSF limitations
- Built on top of JSF
 - Using extension points, not fighting it
- Addressed two key use cases
 - A home for my code
 - Dialogs



A Home for My Code

- Struts Supported in-line execution
 - Setup and teardown



- Not having this in JSF seemed to be a problem...
 - More a lack of understanding rather than a real problem



Shale Added Convenience

- The View Controller
 - Implements code behind / front controller pattern
 - Page controllers as well as event listeners
 - Relies on naming conventions
- Key usecases: initialization and cleanup
- Methods
 - `init()`, `preprocess()`, `prerender()`, `destroy()`
- But – back to some of the old habits
 - `org.apache.shale.view.ViewController` interface



Shale Dialogs

- Unlike Struts, JSF is not strictly tied to the HttpServlet lifecycle
 - Room for innovation
- Dialog
 - A scope between Request and Session
 - Easy to access due to JSF's EL extension mechanism
 - Awkward to instantiate because Shale does not own the XML.



Is Shale Enough?

- For simple apps JSF navigation is OK
- Shale can add some value but:
 - Shale is a toolkit not a controller
 - Enterprise applications need more
- Let's take a step back



Key Controller Features

- Abstraction of flow ✓
- Execute code and display views ✓
- Enhanced state management ✓
- Reuse views ✗
- Secure activities ✗
- Nest / reuse flows ✗



So What is Beyond Shale?

- Three key frameworks (in age order)
 - Spring WebFlow
 - JBoss Seam
 - Oracle ADF Task Flow
- All convergent as we will see
- But first an important question...



What is a Flow?

- If everything is an Object
- Likewise everything is a Process...?
 - Question: *Is a web-flow controller the same as a Business Process Engine or Workflow Engine?*
- *Should we have a single meta-language to describe all possible processes*
 - *Something like BPEL?*
 - *Seam attempts this (jBPM)*



No, Let's Stay Focused

- Only a subset of possible BPM / Workflow models
- No need for
 - Approval hierarchies
 - State dehydration
 - Parallel execution
 - Task lists
 - Compensation

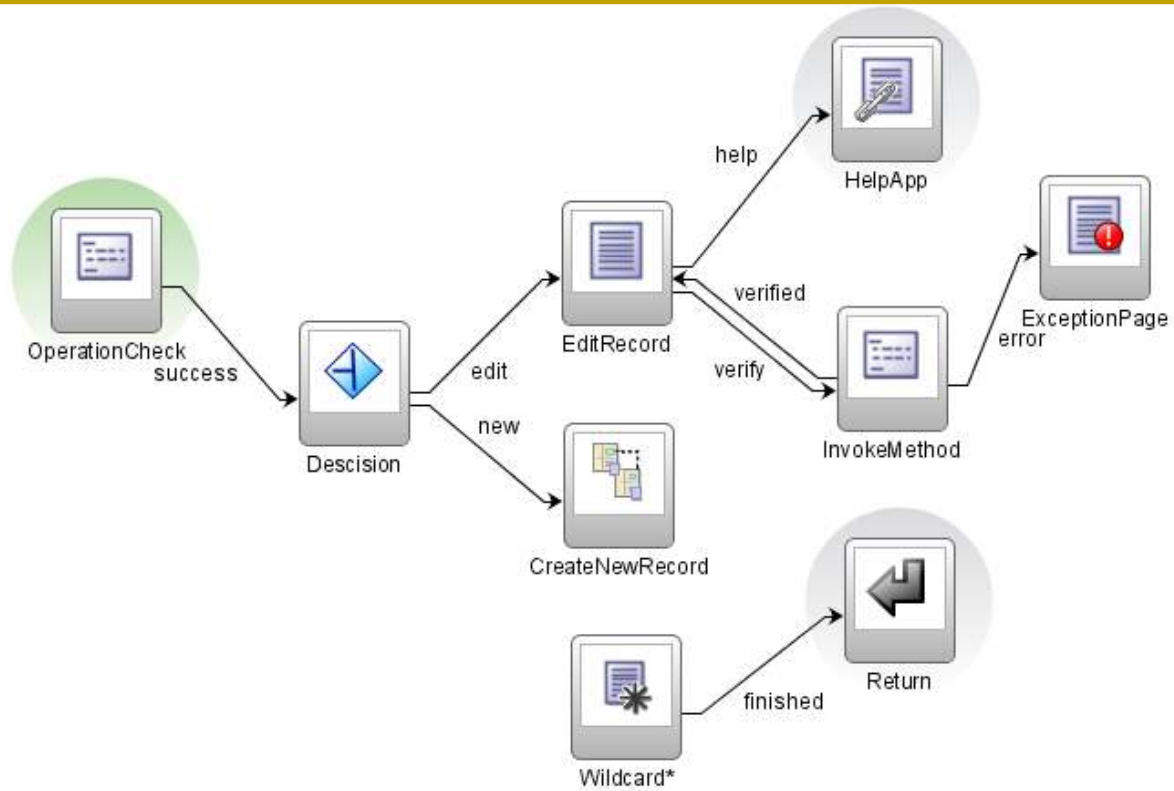


Convergence in the Market

- Key features all exhibited by:
 - Spring Web Flow
 - Seam (jPDL – *not* Seam navigation)
 - ADF Task Flow
- Although not in terminology
 - Task flow, web flow, page flow?
 - Dialog, process, conversation?



Common Features





Common Concepts

- View Activity
- Method Execution
- Router / Decision Activity
- Transition / Navigation cases
- Exception handling



ShowView



InvokeMethod



Decision



The View / Page Activity



- Display a named view
 - Allows page reuse in more than one context
 - Forward or redirect
 - Security / bookmarks
 - Valid transitions
 - The frameworks differ here in syntax
 - Seam / Spring within the View definition
 - ADF as separate navigation cases



Transition Definition – Spring

```
<view-state id="EditRecord" view="edit">  
  <transition on="help" to="appHelp"/>  
</view-state>
```



Transition Definitions – Seam

- Seam Navigation

```
<page view-id=">/edit.jspx">  
  <navigation if-outcome="help">  
    <redirect view-id="/edit.jspx" />  
  </navigation>  
</page>
```

- jPDL

```
<page name="EditRecord" view-id="/edit.jspx">  
  <redirect/>  
  <transition name="help" to="appHelp"/>  
</page>
```



Transition Definition – ADF

```
<view id="EditRecord">  
  <redirect/>  
  <page>/edit.jsp</page>  
</view>
```

```
<control-flow-rule>  
  <from-activity-id>EditRecord</from-activity-id>  
  <control-flow-case>  
    <from-outcome>help</from-outcome>  
    <to-activity-id>appHelp</to-activity-id>  
  </control-flow-case>  
</control-flow-rule>
```



Global Navigation Rules

- Want a "Logout" transition on every page?
 - Wildcards allow the definition of Global rules
 - As implemented by both Struts and JSF
- Spring - `<global-transitions>`
 - No wildcard matching though
- Seam – overload the `<page>` element
 - `<page view-id="*">`
- ADF – just a navigation case
 - Same model as JSF



Calling Methods

- Here there is some diversity

	Spring Web Flow	Seam	ADF Task Flow
Action Activities	✓		✓
Prerender on Views	✓	✓	
Action in PostBack	✓	✓	



Relative Merits?

- Different syntaxes do not block any use cases
- Having multiple options may be confusing to novice users - but not for long
- An area ripe for standardization
 - There is no need for this to be complex
 - Flexibility can hinder re-use



Why I like the ADF Approach

- Less is more
 - Only one concept
 - Just another activity type
 - Same parameter mapping as a view or process
- Different execution paths possible *inbound* and *outbound*
- Simpler to represent on a diagram



Decisions in a Flow

- Routing the flow based on EL
 - More lightweight than calling a method
- Essentially identical in each framework



if {evaluate expression} then
elseif {evaluate different expression} ...
else ...



Exception Handling

- Again common to all
 - Various levels of granularity
- Spring – a type of transition
 - `<transition on-exception="...">`
- Seam – `<exception>` element
 - Granular yet global to flow
- ADF – just mark an activity as an exception handler
 - Only one per flow



ExceptionPage



A Flow must be Reusable

- Therefore:
 - Parameters
 - End states
 - Private scope
 - Can be nested



And So To Scope

- Shale
 - Dialog
- Spring
 - Flash, Flow, Conversation
- Seam
 - Conversation
- ADF
 - PageFlow



So Everyone Agrees

- We need some kind of intermediate scope
 - How "managed" is that?
 - Can scope boundaries be tied to transaction boundaries?
 - What do we call the thing?
- Managed by a finite state machine
 - Can state changes really be "undone"?

.nl.
jug



K.I.S.S.

- Spring is perhaps overly complex
 - Why have 3 scopes when flows can be nested?
 - Flash == flow with a single page
 - Conversation == parent flow to multiple flows

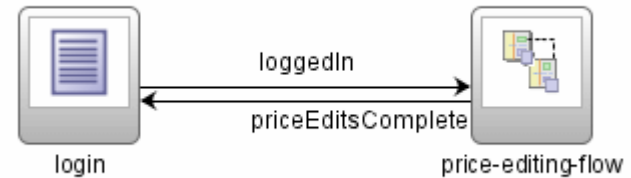


The Flow as a Component

- ADF takes this approach
- Supported by
 - Initializers & Finalizers or Listeners or Events
 - Parameters by reference or value
 - Control over entry points / Reentrancy
 - Transaction management
- Implies certain larger infrastructure
 - How do you package such a thing?



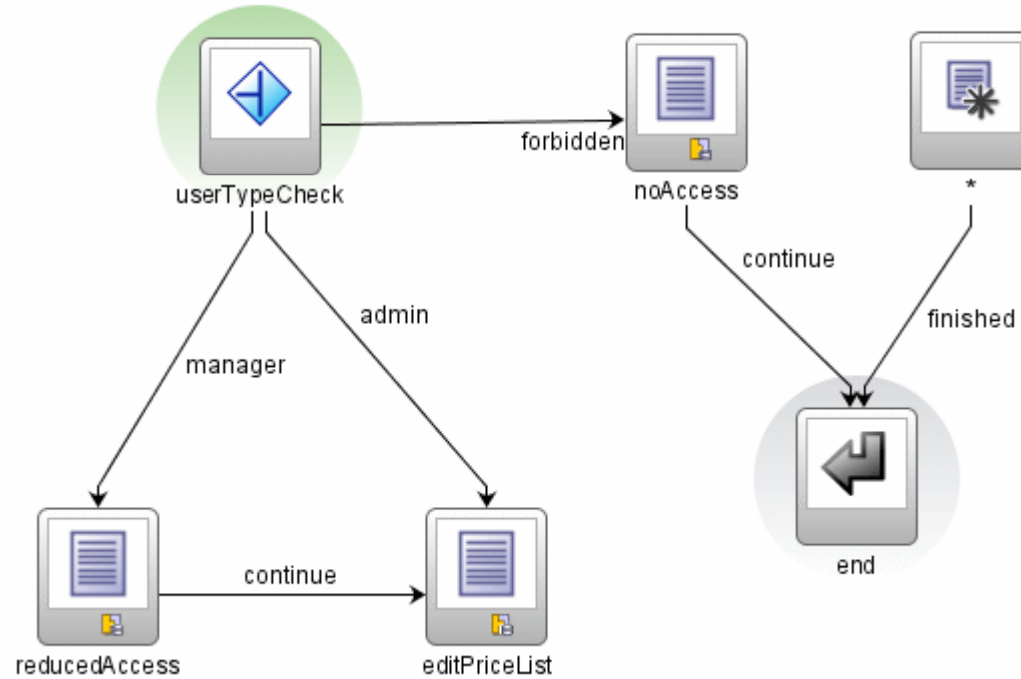
ADF Task Flow Call



```
<task-flow-call id="price-editing-flow">
  <task-flow-reference>
    <document>/WEB-INF/price-editing-flow.xml</document>
    <id>price-editing-flow</id>
  </task-flow-reference>
  <input-parameter>
    <name>uiState</name>
    <value>
      #{!empty cookie.uistate?cookie.uistate.value:""}
    </value>
    <pass-by-value/>
  </input-parameter>
</task-flow-call>
```



ADF Task Flow Definition



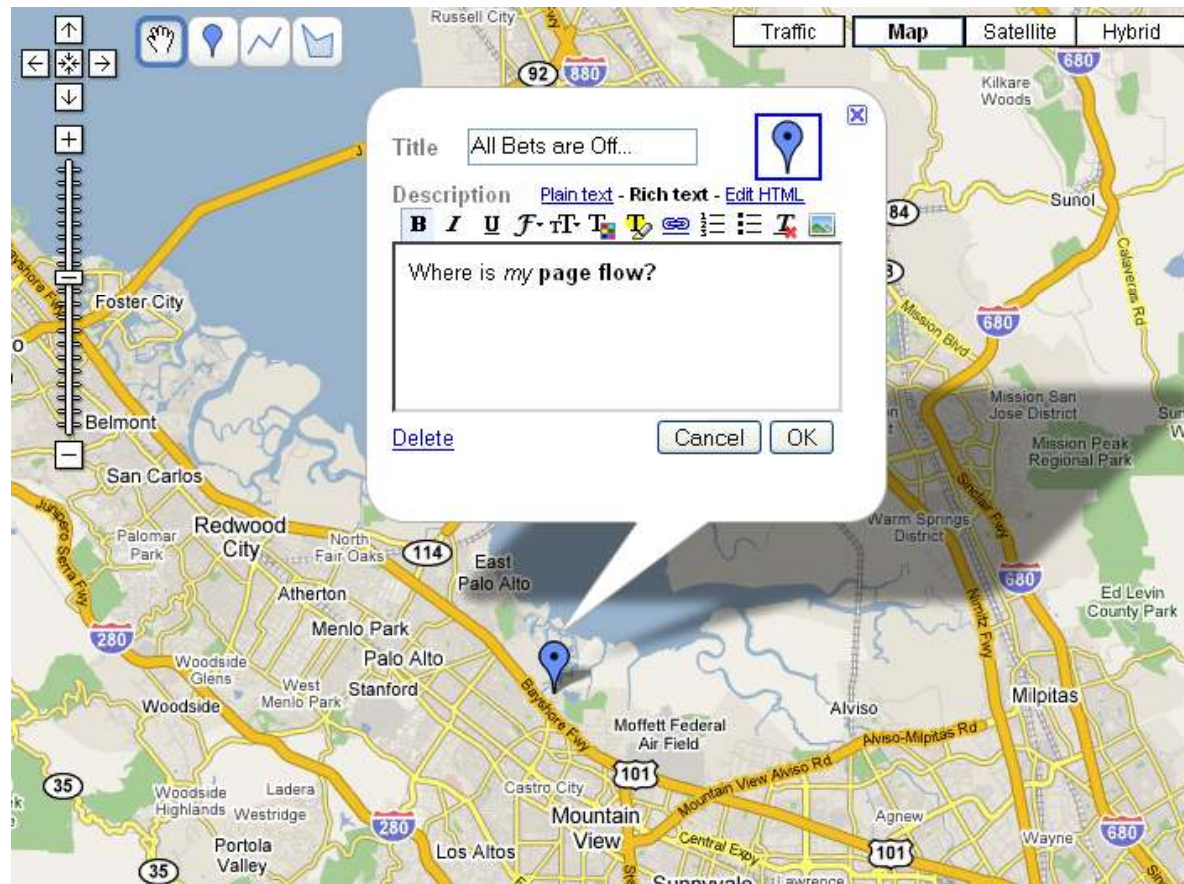


Not All Flows are Components

- Unbounded Flow
 - Any activity is reachable via URL
 - Generally the top level flow
 - Parent for other flows
- Bounded / Inline Flow
 - Encapsulated
 - Control over entry point
 - Control over transaction



However – What About?





Ajax Poses New Challenges

- Fewer page transitions
 - More pop-ups
 - More creative UIs
 - Tabs, splitters, accordions
- Back to the Desktop paradigm
 - Controller on a per component basis?



The Ajax Environment

- Actually similar to the portal environment
 - Multiple parallel lifecycles / conversations
- ADF Task Flow specifically handles this
 - A task flow can be made up of Page fragments
 - Task flows embedded into page regions
 - The logical flow still happens "in-place"



Controller Challenges

- Code vs XML definition
 - This is one place where XML makes sense (I think)
 - Problem: debugging metadata based flows
 - Problem: Unit testing
- Security!
 - Still an issue
- Standardization
 - JSF EG is not focused on this issue
 - Informal discussions @events
 - A home in JSR 299?