Fast, but not furious: ADF Task Flow in 60 Minutes

Frank Nimphius
Oracle Corporation

Schlüsselworte:

ADF, Ajax, Java, JDeveloper, Task Flow

Introduction

The best way to express the importance of Oracle ADF controller and task flow skills for Oracle ADF is by this quote taken from the presentation slide deck:

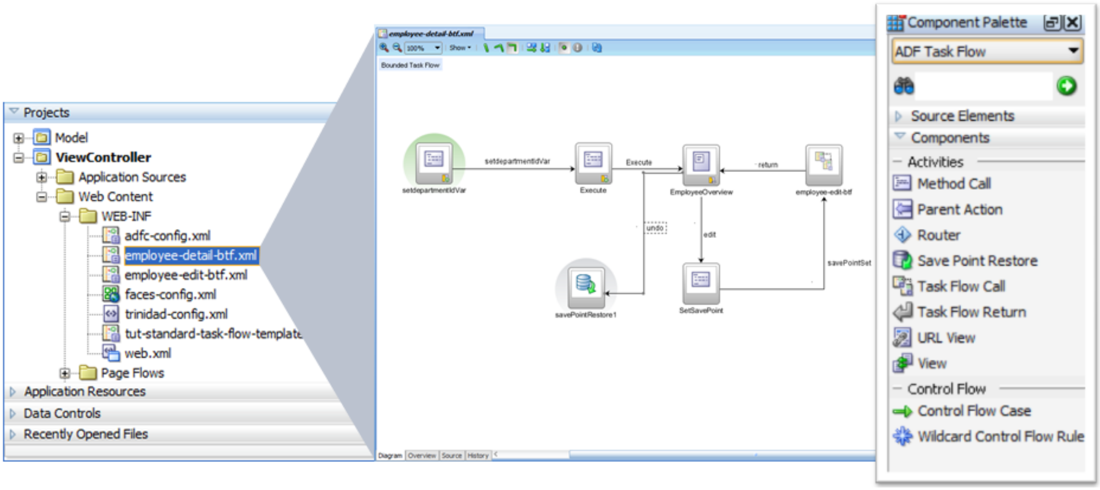
> "You can survive up to 10 days without water but you won't last
> more than a day without task flows "
> – Frank Nimphius, 2011

Though the wow-factor in rich enterprise application development with Oracle JDeveloper and Oracle Application Development Framework (ADF) is typically contributed to the ADF Faces generated rich user interface, in fact it is the ADF controller (ADFc) that deserves praise. ADF controller extends the JSF navigation model and provides functionality that "green-field" JavaServer Faces programmers can only dream of. Though there is a lot to say about ADF controller and its task flow concept, this session stays focused on the core knowledge and functionality developers must know about.

The following summary is an excerpt of the 60 minutes spot light this session put on what I think really is the wow-factor in Oracle ADF, despite of all the other magic that exists in the ADF stack.

Task Flow Overview

The key concept in ADF Controller is ADF task flow. A task flow is the blueprint of a reusable unit of navigation and control flow logic. Task flows are defined in XML metadata and edited in the Oracle JDeveloper task flow modeler and the Property Inspector. Rarely do you use the source code editor to directly manipulate XML metadata.



Using the Oracle JDeveloper visual task flow modeler (shown in the image above), navigation flows and processes are modeled declaratively. Task flows can be designed without implementing its logic and thus simplify communication between project teams in IT and non-technical groups.

Application developers use two types – or flavors – of task flow:

Unbounded task flow – The unbounded task flow is the equivalent of the default navigation model in JavaServer Faces. It usually only holds a minimum set of activities and views that present a landing pad for users to enter an application.

Bounded task flow – The concept of bounded task flow is unique in the JavaSever Faces industry and allows developers to partition application development into reusable control flow units. Bounded task flows have a well defined boundary and a single point of entry. They are functional building blocks that developers assemble to business applications.

For bounded task flows there are three use cases in which they can be used in an application

ADF region – The ADF region concept allows bounded task flows to execute processes in the context of a page without the page to refresh upon task flow navigation. ADF regions enable developers to build real desktop like Ajax web user interfaces in ADF, for which Oracle also provides a specific page template, the UI Shell template.

Remote task flow – Similar to portlets, though different, bounded task flows can be deployed as Java EE applications on a remote server and accessed from other ADF applications. The calling application navigates to the remote task flow instance for the user to continue his work. Once finished, upon task flow return, the calling application state is recovered and the outcome of the remote task flow becomes available for use in this context. Remote task flows can be looked at as services with a user interface.

Traditional navigation – Task flows can be called from task flow call activities, in which the called task flow becomes a sub flow of the calling task flow. Upon return, the calling task flow is resumed for the user to proceed. Traditional task flow navigation works for task flows displayed directly in the browser or in ADF regions.

Bounded Task Flow

Bounded task flows are defined in XML configuration files and usually consist of view - , method - router -, task flow call -, and return activities to describe a functional application use case. Views in bounded task flow can be page fragments (used in ADF regions) or top level browser pages (JSPX documents).

The following lists core characteristics of bounded task flows:

- Bounded task flows have a single entry point that ensures users enter a flow according to how the business use case requires it. In standard JavaServer Faces and unbounded task flows, for example, there exists no restriction like that on how users enter an application flow.

- Bounded task flows have defined boundaries that allow developers to run them within their own transaction, isolated from the transaction of the calling flow or application.

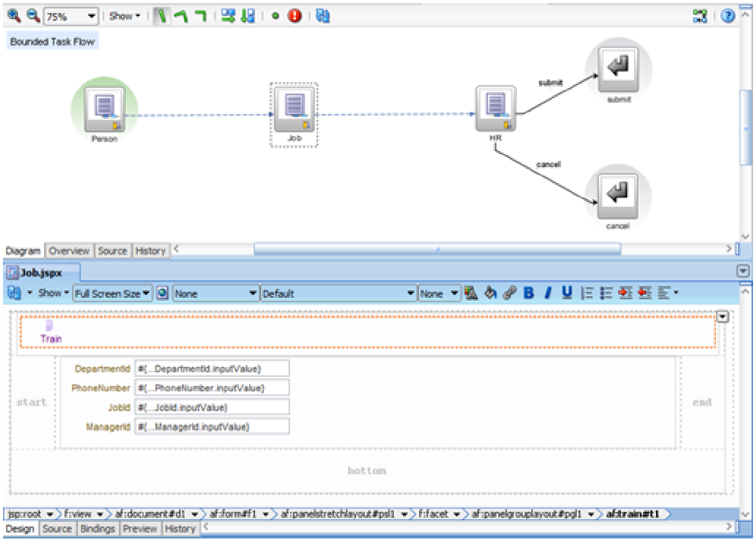
- Bounded task flows are well integrated with ADF Security in that the ADF framework automatically check access permissions for task flows executed in a security enabled ADF application.

- Bounded task flows have their own memory scope – pageFlowScope – that becomes available when users enter a control flow and that is automatically cleaned up when users leave the process.

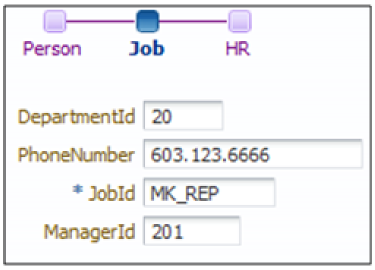
- Bounded task flows are reusable in ADF libraries for use in different applications or as a composite flow in a larger enterprise application.

Train Navigation

Navigation in bounded task flows usually is through named JSF control flow rules. Bounded task flows however can be configured to also expose a train navigation model for its contained views, which often is the preferred user option when working in long sequential processes, like data input wizards.



DepartmentId #{...DepartmentId.inputValue}
PhoneNumber #{...PhoneNumber.inputValue}
JobId #{...JobId.inputValue}
ManagerId #{...ManagerId.inputValue}



DepartmentId	20
PhoneNumber	603.123.6666
* JobId	MK_REP
ManagerId	201

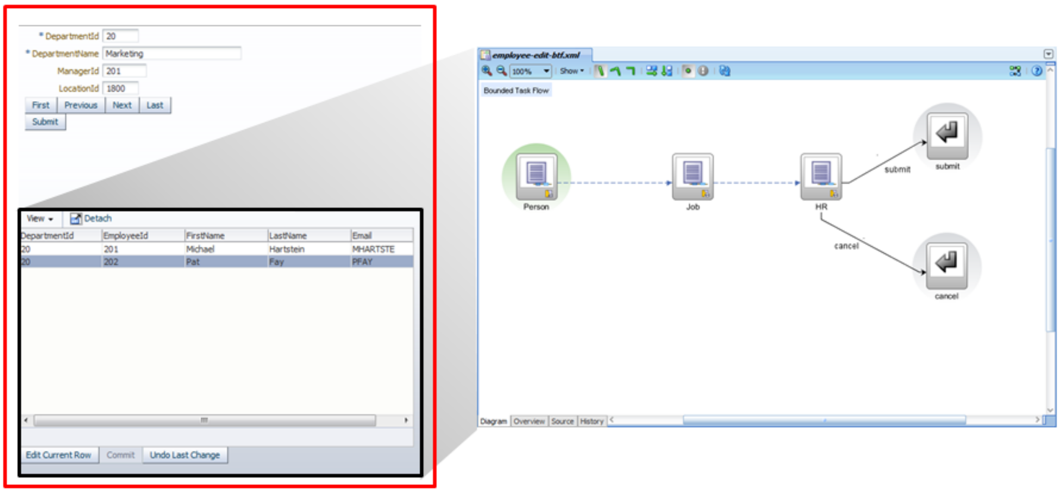
At runtime, the sequence of views is displayed as train stops on a page, as shown in the image above. Users navigate within the process flow by clicking on one of the enable train stops. Though trains are based on a generated train model, they can be customized and manipulated at runtime.

ADF Region

The ability to execute reusable control flows in a region is a unique capability of ADFc that does not exist in any other JavaServer Faces or Ajax framework. ADF region also is a key design pattern used in Oracle Fusion Applications.

ADF region display bounded task flows in independent areas of a page or a page fragment. Navigating a control flow within a region does not require the containing page or task flow (regions can be nested) to refresh, which gives the application the impression of instant application response: this really is how Ajax should be.

To communicate between bounded task flows displayed in regions, developers use known interaction patterns. For example, one way to establish communication is to use task flow input parameters and to partially refresh the region. Another, more sophisticate option, is contextual events, which is a publish-subscribe mechanism handled by the ADF binding layer.

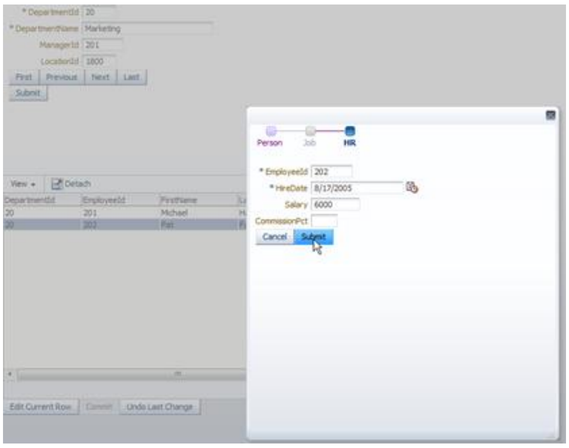


* DepartmentId 20
* DepartmentName Marketing
ManagerId 201
LocationId 1800

DepartmentId	EmployeeId	FirstName	LastName	Email
20	201	Michael	Hartstein	MHARTSTE
20	202	Pat	Fay	PFAY

ADF Region in Dialog

Often input forms require complex data lookup. Bounded task flows accessed from a task flow call activity can be declaratively configured to launch in a dialog. This way, for example, developers can use bounded task flows to build complex data lookup dialogs to show as-list-of-values, as shown in the image below.



* EmployeeId 202
* HireDate 8/17/2005
Salary 6000
CommissionPct

Conclusion

You can survive up to 10 days without water but you won't last more than a day without task flows. Oracle ADF task flow is a core skill every ADF application developer must master. It also is a major advantage of the ADF platform compared to other Ajax and JSF frameworks on the market. Every minute spent on learning task flow is a minute well spent for your success.

Kontaktadresse:

Frank Nimphius
Oracle Corporation
E-Mail frank.nimphius@oracle.com
Internet: www.oracle.com
www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html