Oracle ADF 11g Primer

Introduction to the building blocks of a Fusion Web application

An Oracle White Paper
April 2007

ORACLE

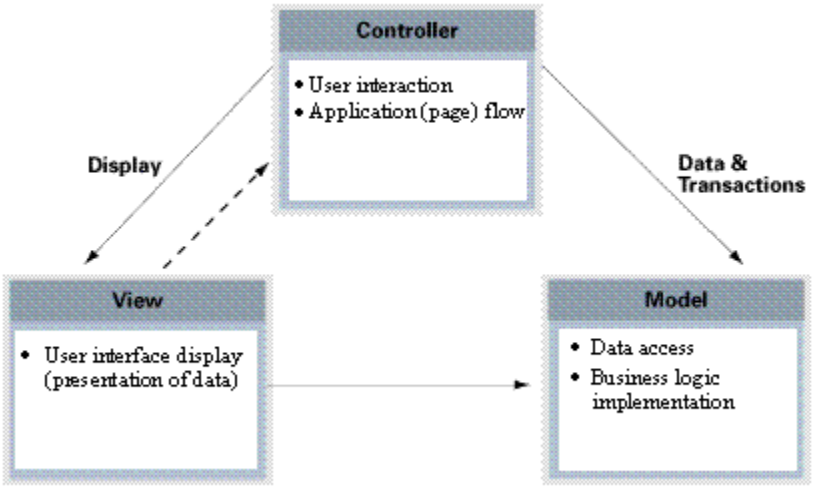
Developer guides are shipped with JDeveloper for easy access. These guides can be found from the JDeveloper Start Page or online on the Oracle Technology Network.

UNDERSTANDING THE BUILDING BLOCKS OF AN APPLICATION

To understand the building blocks of a Fusion application, we can first look at the basic tasks that an application is responsible for undertaking.

- Data access
- Business logic implementation
- User interface display (presentation of data)
- User interaction
- Application (page) flow

In Fusion, these basic tasks can be grouped into three distinct layers; the model, view and controller.



The **View Layer** provides the user interface display to the application and raises events to the controller layer. The most basic and important component of the view layer is the page. The page is built in JDeveloper and is the full web page that is shown to the user. There are many other components of the view layer that are expanded upon below.

Want hands on experience building a Fusion Web Application?

Use cue cards to guide you through the steps of building a Fusion Web application. Cue cards can be found in JDeveloper both in the Start Page or under the Help Menu.

The **Controller Layer** manages the application flow. The ADF Controller processes user input, handles errors and decides what page the user should see. The key definition that the controller layer uses to manage the application is the task flow. The task flow is built in JDeveloper and is where the developer defines an application task (e.g. Create Expense Report, Enter Budget). The definition of the task includes the pages and logic that interact to allow a user to complete the task. For Create Expense Report, the Task Flow may contain 3 pages (Enter Expense Lines, Submit Expense Report and Confirm) and the logic that navigates the user between the pages.

The **Model Layer** is responsible for data access and business logic implementation. In Fusion the model layer is comprised of two important sub components. The first is the databinding facility that connects the view layer to the data and business services layer. This databinding facility is called the data control and is the main component of the ADF Model. The second is called by the data control and is business services. Application Module and ADF Business Components are the key components of the business services layer that are connected to the view layer by the data control. This separation allows for other business services to be called or for the data access and business services to be implemented after creation of the view layer.

VIEW LAYER

As highlighted earlier, the view layer in its simplest form could be a page with user interface components (e.g. table, input text, output text, and button) added to the page and bound to the model layer. The user interface components of the page can invoke the controller to navigate to other pages. However, ADF provides other components of the view layer that allow for the creation of rich and reusable user interfaces. These components are described in this section.

Page

The page is built in JDeveloper and is the full web page that is shown to the user. For our Create Expense Report example, the developer may create three pages: Enter Expense Lines, Submit Expense Report and Confirmation.

As will be seen later in this document, application pages are likely to be more sophisticated than this and will use more of the ADF features described below to create rich and reusable user interfaces.

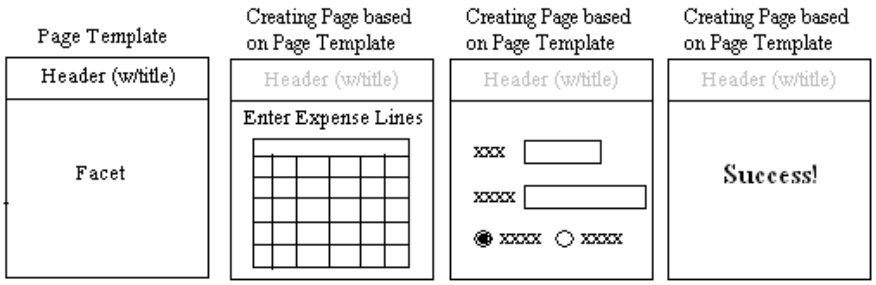
Page Fragment

Page fragments are created and used very much like pages. In fact, to a developer there is very little difference between how they would build a page in JDeveloper versus a page fragment. In our Create Expense Report example, the same three pages can be created as page fragment definitions instead of pages. The one major difference between pages and page fragments is that page fragments are not defined as a full web page. This allows page fragments to be used as a smaller part of a larger web page. For example, if you want the Create Expense Report task to be displayed within a page that has other content displayed, you would create the pages as page fragment definitions. Because of this capability, it is likely most of application tasks will be built with page fragments.

Page Template

Page templates are reusable, data-bound templates that can be used as a basis for building pages and page fragments. If there are portions of a page that are common across multiple pages, developers can create a page template for those portions of the page and then use that page template as a basis for creating the pages or page fragments that use the common functionality. Page templates can have parameters that allow developers to configure how the page template is used (e.g. setting the title that appears in the header). Page templates also define areas on the page (facets) where content can be added by the developers when they create their pages. These facets can also contain default content.

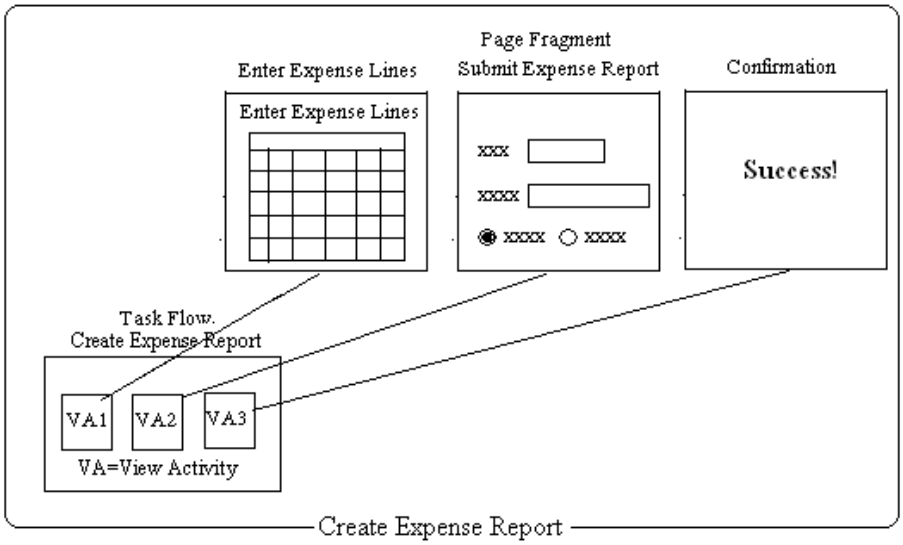
For instance, if all application pages were to share a similar header (with title) at the top of the page, then a developer could create a page template that contains the header and make the page template available to other developers building pages. When other developers create their pages, they will base their pages on the header template.



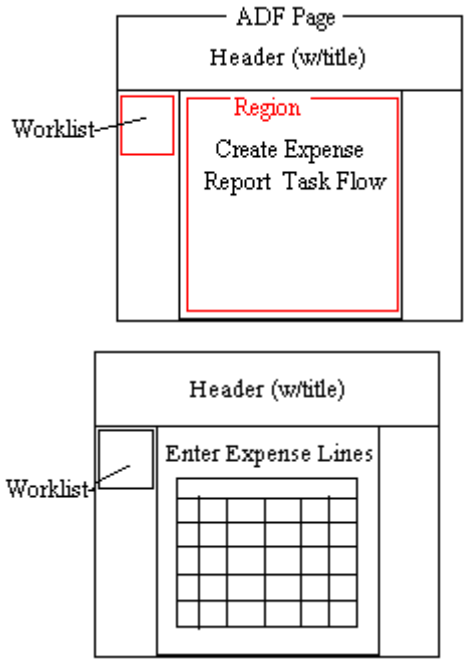
Region

A region is a powerful new feature in Fusion that can be used within a page or page fragment. A region can be considered synonymous with a portlet in regards to functionality and user behavior. However, regions will be used when you are including the application functionality in a local context and portlets will be used when exposing the application functionality for external usage.

As described earlier, a developer would create a task flow for the Create Expense Report task. This definition would contain three pages, Enter Expense Lines, Submit Expense Report and Confirmation.



If developers want this task to be displayed to the user within a page that has other content (e.g. a worklist and a header), they would use a region to wrap this task flow within their pages. Or rather, in JDeveloper, they would simply drop the task flow onto a page as a region.



There are special requirements to use a task flow as a region within a page. For instance the task flow must be made up of page fragments instead of pages and the task flow must be a bounded task flow (see task flow section for a description of the differences between bounded and unbounded task flows.).

Components within regions can communicate with other regions on a page through a publish-and-subscribe mechanism called the contextual event framework.

Contextual Event Framework

Many of application pages will be constructed in such a way that multiple pieces of application content can be combined to create rich, modular, contextualized applications. For instance, if the user is entering expense lines in our Create Expense Report example, it may be important to display the company's expense policy and guidelines in another region of the page.

When a page contains multiple regions that need to work together to display relevant information to the user, the need to communicate between regions becomes very important. When the regions on a page share an ADF BC application module their context will be synchronized automatically. However, when the sources of data are not in the same application module, this communication can be handled through an event framework called the Contextual Event Framework. The Contextual Event Framework provides the page with the ability to map events that will be produced and consumed by the page's various regions.

ADF Rich Client Components

ADF provides over 100 components for developers to use to create rich user interface pages and page fragments in Fusion. These components will be available in a common palette within JDeveloper. These components are classified as follows.

- Layout components
- Table and tree components
- LOV components
- Input components
- Navigation components
- Output components
- Query components
- Data visualization components

Declarative Component

Declarative components combine the functionality of multiple ADF components. Declarative components have no business logic and are not data bound. An important trait of a declarative component is that the developer of the component can hide attributes so they cannot be changed by the consumers. For example, a declarative component could be created to group "Okay" and "Cancel" buttons, and the developer can hide the label attribute so the consuming developer cannot change "Cancel" to "Exit".

CONTROLLER LAYER

The ADF Controller is a critical and necessary component to all of our web applications. The controller manages the user interaction and application flow that has been defined by the developer in the task flow.

Task Flow

The task flow is built in JDeveloper and is where the developer defines an application task (e.g. Create Expense Report, Enter Budget). The definition of the task includes the pages and logic that interact to allow a user to complete the task.

At design time, the pages and page fragments that make up a task are added to a task flow as activities, in the case of pages and page fragments they are view activities. So, when a user navigates from one page to another within a task flow, they are transitioning from one activity to another. Activities can be considered the building blocks of the Task Flows. There are other activities besides view activities that will be critical to developers in creating task flows. For instance, if a developer wanted to have both Enter Expense Line page and Update Expense Line page as

view activities in their task flow and have an initial activity that would make the decision as to what view activity to display to the user, they would add a different type of activity to their task flow called a router activity. A router activity evaluates a declarative case statement and determines where to route the control flow next. There are two types of task flows in Fusion, bounded and unbounded task flows. Both are described in more detail below.

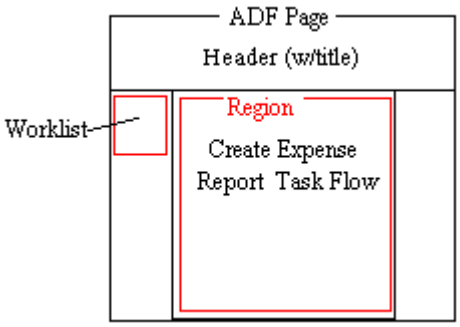
Bounded Task Flow

Bounded task flows are one of two types of task flows and have specific functionality that has been added to support transactions and reuse. Bounded task flows are the only type of task flow that can be used as a region in a page. Most application task flows will be bounded task flows because of the additional features to support transactions. Some of those features include:

- Well-defined boundary with a transaction beginning and end
- Single entry point and zero or more well-defined exit points
- Memory scope variables, called pageFlowScope variables, for passing data between activities within the task flow
- Declarative support for transaction management
- Able to begin a new transaction upon task flow entry
- Able to commit or rollback upon task flow exit
- Declarative support for back button navigation
- Able to pass input parameters from the task flow caller to store within the pageFlow scope
- Able to return values back to task flow caller upon exit

Unbounded Task Flow

An unbounded task flow contains activities similar to a bounded task flow, but does not contain the well-defined boundary of a bounded task flow nor a single entry point. Unbounded task flows cannot be parameterized or define transactional and reentry characteristics, hence they are not able to be used within a page as a region. Unbounded task flows are not callable from BPEL and are not securable objects. Therefore, the unbounded task flow will be used in a different way by applications than the bounded task flow. For example, the Create Expense Report task would be implemented as a bounded task flow so that it could be used within a region of a page. The page that contains the entire user interface with the header, worklist and expense report task also needs to reside in a task flow. This is likely where the unbounded task flow will fit in. It is likely that this application page will be added to the unbounded task flow as a view activity and be callable from the menu navigation via this unbounded task flow.



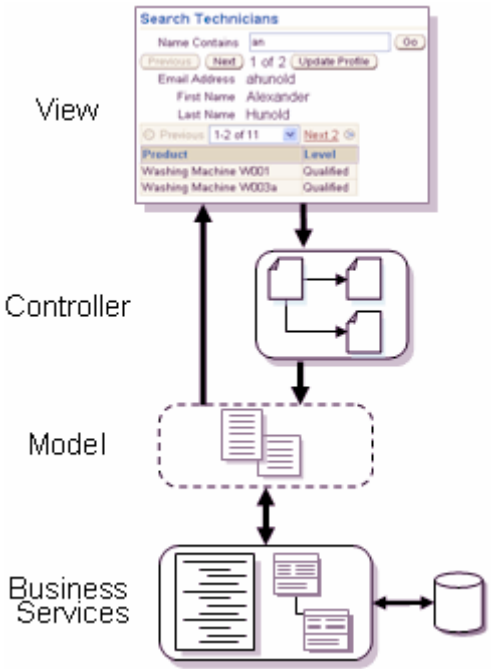
Note, each web application will only have one unbounded task flow. These may be defined at design time across one or more individual physical files. However, at runtime these files will be merged and loaded as one unbounded task flow. Also, unlike the bounded task flow that has a single entry point, the view activities of an unbounded task flow can be set to bookmarkable so that you may have a bookmarkable URL to the application page referenced by the view activity.

Task Flow Template

Task flow templates provide the ability to capture common task flows and behavior for reuse across many different bounded task flows. Two mechanisms will be provided: reuse by copy and reuse by reference. Exception handling is an area where developers may make use of task flow templates.

MODEL LAYER

The model layer provides access to data as well as business logic implementation. In Fusion the model layer is comprised of two important sub components. The first is the databinding facility that connects the view layer to the data and business services layer. This databinding facility is called the data control and is the main component of ADF Model. The second is the business services layer. Application Module and ADF Business Components are the key components of the business services layer that are connected to the view layer by the data control. This separation allows for other business services to be called or for the data access and business services to be implemented after creation of the view layer.



Model

Data Control

Data controls provide for the ability to create data bound pages and page fragments. These data controls can be backed by many business services such as XML files, Web Services or ADF Business Components. Data controls are created in JDeveloper just like other common definitions. However, to ease implementation, when you build an application module in JDeveloper a corresponding data control is automatically created for you for use within your pages. To build your data bound pages, you simply drag and drop your data control from the data control pallette onto your page.

Placeholder Data Control

Placeholder data controls can be used to create pages and page fragments in a UI first development approach , where the data sources are connected after the pages have been built.

Business Services

ADF-Business Components will be the main source of data access and business logic in Fusion applications. ADF Business Components, or ADF BC, includes the following key objects:

Application Module

An Application Module is the transactional component that UI clients use to work with application data. It defines an updateable data model and top-level procedures

and functions (called service methods) related to a logical unit of work related to an end-user task. Application Modules contain View Objects.

To simplify application integration, you can enable a service interface on any Application Module. The service interface is a programmatic API allowing third-party applications to find, create, update, and delete business information using standard web services protocols. All of the business validation rules encapsulated in your Entity Objects are automatically enforced through this programmatic integration API as well.

View object

A View Object represents a SQL query and simplifies working with its results. SQL is used to join, project, filter, sort, and aggregate data into exactly the "shape" required by the end-user task at hand. This includes the ability to link a View Object with others to create master/detail hierarchies. When end users modify data in the user interface, View Objects collaborate with Entity Objects to consistently validate and save the changes.

For the UI layer, new View Objects and Application Modules will be created to support the UI and the task flows.

Entity object

An Entity Object represents a row in a database table and simplifies modifying its data by handling all DML operations. It can encapsulate business logic for the row to ensure business rules are consistently enforced. Entity Objects can be associated with others to reflect relationships in the underlying database schema to create a layer of business domain objects to reuse in multiple applications.

ADF 11*g* Primer
April 2007
Author: Laura Akel
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com