ADF Naming Conventions

Motivation

During ADF applications development we may encompass many development challenges. One of these challenges is about implementing a convention of namings to be used by all involved personnel during implementation.

Each developer have his own background and his own ideas about how things should be implemented. We want them to have freedom of speech in order to get the best approaches to reach the end, but what we really don't want is to have multiple ways of doing the same thing otherwise we might face really difficult challenges in the future.

Developers usually change during project development. For the new ones will enter we need to provide training. If we can follow conventions we will have less training periods and they will be more easily familiar with the entire application.

After application being on production we face a new challenge, **Maintenance** and **Support**. Some big headaches usually appear right here, and they can be even bigger if we don't follow these important naming conventions in our applications' code.

I have found some information here about this topic but I needed more, and I needed to instance it to my project, so I decided to defined my own ADF Naming Conventions before starting my project.

In this document I will share my experience and my ADF Naming conventions regarding the following topics:

- **Application & Project Namings**
- **Packages Namings**
- **Business Components Namings**
- **Model & View Controller Projects**
- **Templates**
- **JSF, JSFF, JSPX**
- **JAVA Events**
- **JAR, WAR, EAR files**

Author	Pedro Gabriel	Twitter	@PedrohnGabriel

1. Abbreviations

Consider the following terms used in these document:

Abreviation	Definition	Example
PROJECT_NAME	Customers or projects short name. Try to use only the necessary characters that are capable to describe your customer/project in order that everyone can understand it.	**RMK** (Red Mavericks)
MODULE_NAME	Provide a name that describes the target of the project (module).	**MyAdfLib**
DOMAIN_NAME	Internet domain name.	**red.mavericks**
BUSINESS_AREA	Feature Business area (retail, financial, etc.)	**retail/financial**

2. Application & Project Namings

For Applications and Projects we have defined the following namings:

Application	Project
<PROJECT_NAME> + **<MODULE_NAME>** + **App** Example: RMKMyAdfLibApp	**<PROJECT_NAME>** + **<MODULE_NAME>** Example: RMKMyAdfLib

IMPORTANT NOTE: For both previous namings decide if they should be in upper case, lower case or camel case.

3. Packages Namings

In the wizard for creating a new application you are prompted to set up the "**Application Package Prefix**". This package prefix will define the root package for the projects contained in this new application.

The default "**Application Package Prefix**" for new applications should be:

<DOMAIN_NAME> + . + **<PROJECT_NAME>**

Example: red.mavericks.rmk

For each new project you will inherit application's root package structure. Nevertheless you should configure it to have a distinct name from the other projects. With this you are able to identify your modules/libraries. The package structure for your project should follow the next pattern:

<DOMAIN_NAME> + . + **<PROJECT_NAME>** + . + **<MODULE_NAME>**

Example: red.mavericks.rmk.my.adf.lib

4. Business Components Namings

In this section we will provide packages structure and file namings for Business Components.

IMPORTANT NOTE: Files should be named in camel case.

5.1. Packages Structure

For Business Components projects we should gather **Entity Objects**, **View Objects**, **View Links** and **Associations** under the following packages:

Type	Package	Description
Entity Associations	<DOMAIN_NAME> + . + <PROJECT_NAME> + . + <MODULE_NAME> + **adfc.entity.associations**	Contain entity associations
Entity Objects	<DOMAIN_NAME> + . + <PROJECT_NAME> + . + <MODULE_NAME> + **adfc.entity.objects**	Contain entity objects

View Links	<DOMAIN_NAME> + . + <PROJECT_NAME> + . + <MODULE_NAME> + **adfc.view.links**	Contain view links
View Objects	<DOMAIN_NAME> + . + <PROJECT_NAME> + . + <MODULE_NAME> + **adfc.view.objects**	Contain view objects
Model JPX File	<DOMAIN_NAME> + . + <PROJECT_NAME> + . + <MODULE_NAME> + **adfc**	Contain model JPX file

Example:

Entity Associations:	red.mavericks.rmk.my.adf.lib.**adfc.entity.associations**
Entity Objects:	red.mavericks.rmk.my.adf.lib.**adfc.entity.objects**
View Links:	red.mavericks.rmk.my.adf.lib.**adfc.view.links**
View Objects:	red.mavericks.rmk.my.adf.lib.**adfc.view.objects**
Model's JPX File:	red.mavericks.rmk.my.adf.lib.**adfc**

5.2. File Namings

To each Business Component type you should provide the following prefixes:

File Type	Suffix	Example
Entity Associations	Assoc	SomeNameAssoc
Entity Objects	**EO**	SomeNameEO
View Links	**VL**	SomeNameVL
View Objects	**VO**	SomeNameVO
List Of Values	**LOV**	SomeNameLOV
Model JPX File	**Equal to project's name**	

5.3. View Link Files

View Links names should be self-explanatory so we can easily identify their purpose. Based on this we defined the following pattern:

<ViewObjectSourceName> + <ViewObjectDestinationName> + VL

Example: EmployeeEmployeeBranchVL

5.4. Application Modules

You may have multiple "**Application Modules**" in your in application. For this case we should be able to identify their purpose as well as their business area target. Based on this we followed the next pattern:

<PROJECT_NAME>+ <MODULE_NAME> + AM

Example: RMKMyAdfLibAM

6. Model & View Controller Namings

In this section we will provide the namings we used inside **Model** and **ViewController** projects for ADF applications.

6.1. Model

For new ADF applications we are requested to set the names for **Model** and **ViewController** projects. In what concerns the **Model** it should following naming:

<PROJECT_NAME> + <**MODULE_NAME**> + **Model**

Example: RMKMyAdfLibModel

For the package structure it should be configured as follows:

<DOMAIN_NAME> + . + **<PROJECT_NAME>** + . + **<MODULE_NAME>** + . + **model**

Example: red.mavericks.rmk.my.adf.lib.model

As you can see, package structure follows the same structure defined in my previous post plus "**model**".

6.2. View Controller

In **ViewController** projects we have a wide range of namings we can follow since we have multiple features we can take advantage of. For that reason we divided it in sub topics.

6.2.1. Project Name

The name for the project should be defined as follows:

<PROJECT_NAME> + <**MODULE_NAME**> + Controller

Example: RMKMyAdflibController

By using "**Controller**" prefix we are able to automatically identify projects type and purpose.

6.2.2. Project Default Package Structure

Package structure for View Controller project should be defined as follows:

<DOMAIN_NAME> + . + **<PROJECT_NAME>** + . + **<MODULE_NAME>** + . + **view**

Example: red.mavericks.rmk.my.adf.lib.view

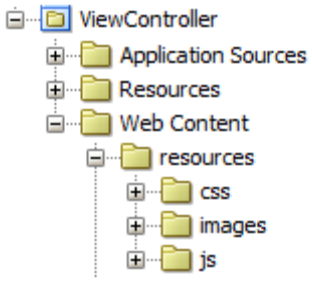
As you can see, package structure follows the same structure defined in my previous post plus "**view**".

6.2.3. Images, CSS and JavaScript Directories

Images, CSS and JavaScript directories should be defined right under "**Web Content**" folder. The folders should have the following names:

Folder Type	Folder Path
To contain images	resources/images
To contain CSS files	resources/css
To contain javascript files	resources/js

Example:



Inside "**web.xml**" file in ViewController project set the following mappings:

```
<servlet-mapping>
    <servlet-name>resources</servlet-name>
    <url-pattern>resources/images/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>resources</servlet-name>
    <url-pattern>resources/css/*</url-pattern>
```

```
</servlet-mapping>
<servlet-mapping>
    <servlet-name>resources</servlet-name>
     <url-pattern>resources/js/*</url-pattern>
</servlet-mapping>
```

6.2.4. Bounded Task Flows, JSFF, JSPX Directories

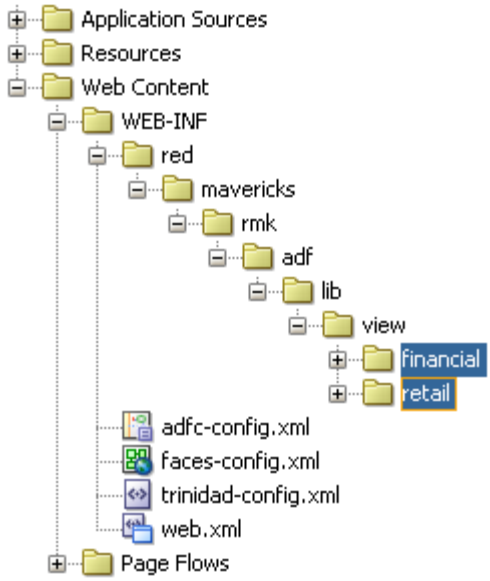
Reusable bounded task flows can be published in ADF Libraries and consumed by other ADF applications. For this use case is important, and a must, that all bound task flows have a unique name. If this isn't accomplished ADF has no way to distinguish between task flows with the same name.

Before creating bounded task flows you should set a folder structure under "**WEB-INF**" folder. After this folder structure is created you can start creating your bounded task flows.

Folder structure should be created according to the following convention:

<DOMAIN_NAME> + <PROJECT_NAME> + <MODULE_NAME> + view + <BUSINESS_AREA>

Example:



JSF, JSFF and JSPX files must be saved on the same folder of the task flow.

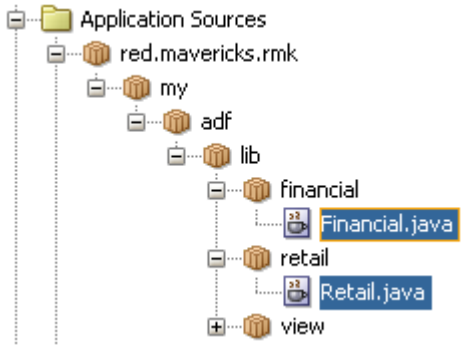
Page definitions will be automatically generated with same package structure under Application Sources folder.

Example: red.mavericks.rmk.my.adf.lib.view.financial
red.mavericks.rmk.my.adf.lib.view.retail

Managed Beans for Task Flows should be created under Application Sources folder with the following package structure:

<DOMAIN_NAME> + **<PROJECT_NAME>** + **<MODULE_NAME>** + **<BUSINESS_AREA>**

Example:



We decided to create Managed Beans outside "view" package in order to not have a mixture between page definitions and java classes.

6.2.5. Task Flow Namings

The name for Task Flows should be defined as follows:

<TASK_FLOW_CAMEL_CASE> + TF

Example: myTaskFlowTF

We may have different task flows types, each one built to address one purpose. For these cases we added a constant to the name in order to easily recognize their purpose/target. Next table presents the Types and Target Names:

Type	Task Flow Target Name
Task Flow to filter data	<TASK_FLOW_CAMEL_CASE> + **Filter** + TF
Task Flow to perform actions on data	<TASK_FLOW_CAMEL_CASE> + **Action** + TF
Task Flow to list data	<TASK_FLOW_CAMEL_CASE> + **List** + TF
Task Flow to detail data	<TASK_FLOW_CAMEL_CASE> + **Detail + TF**
Task Flow to combine multiple task flows	<TASK_FLOW_CAMEL_CASE> + **Container** + TF

6.2.6. Task Flow Managed Beans

Task Flows' managed beans are responsible for managing data. Multiple managed beans can be created for a single Task Flow. Nevertheless you usually have one main managed bean. For these cases managed beans should have a similar name as the task flow.

As you may have already noticed, you are free to provide any name you want to the Java Class you assign to your managed bean in "Managed Beans" task flow's tab. For this scenario we recommend to use the same name of the Java Class.

By following this two advises you will be able to find what you are looking more easily without losing time and effort to understand the mappings made by each developer. This is very

important for developers during development and maintenance phases. Based on these assumptions take a look on the following example:

Task Flow Name:	**myTaskFlowTF**
Java Class Name:	**MyTaskFlow**
Managed Bean Name:	**MyTaskFlow**

6.2.7. Task Flow Input & Return Parameters

Task flow's input parameters should be prefixed with "**in**":

in + <CAMEL_CASE>

Example: inMyParameter

Task flow's return parameters should be prefixed with "**rtn**":

rtn + <CAMEL_CASE>

Example: rtnMyParameter

7. Templates Namings

ADF lets us to create different types of templates to abstract common functionalities used in our projects, for example: **Page Templates** and **Task Flow Templates**. This templates should be created using the following pattern:

<CAMEL_CASE> + Template

Example: myTaskFlowTemplate, myPageTemplate

8. JSF, JSFF, JSPX Namings

Pages names should have a self-explanatory name in order to be easily identified and recognized with its purpose. The default pattern we followed was:

<CAMEL_CASE>

In Task Flows you may have multiple pages depending on the route it takes, but for the main page (if we have one) we should provide it with the same name as the task flow but without 'TF' suffix. Example:

Task Flow Name:	**myFinantialTasksListTF**
Page Name:	**myFinantialTasksList**

9. Java Events Namings

For our controls events we set suffixes for them. This will help you to understand the event type you are taking care without the need to go directly to the page, only if you need to understand it in more detail. The list of suffixes for each type of event are listed in the following table:

Event Type	Suffix
Action Event	**Action**
Value Change Listener	**VCL**
Selection Event Listener	**SEL**
Client Event Listener	**CEL**
Return Event Listener	**REL**
Action Event Listener	**AEL**

10. JAR, WAR, EAR Files

Building and deploying projects lead us to create deployment profiles. In this deployments profiles you should use the same namings. The next table addresses this topic by providing the namings for each type of deployment profile.

File Type	Prefix
JAR	**jar** + <PROJECT_NAME> + <MODULE_NAME>
ADF JAR Library	**adflib** + <PROJECT_NAME> + <MODULE_NAME>
Shared Library	**sharedlib** + <PROJECT_NAME> + <MODULE_NAME>
WAR	**war** + <PROJECT_NAME> + <MODULE_NAME>
EAR	**ear** + <PROJECT_NAME> + <MODULE_NAME>