ADF Code Corner

Oracle JDeveloper OTN Harvest 06 / 2012

Abstract:

The Oracle JDeveloper forum is in the Top 5 of the most active forums on the Oracle Technology Network (OTN). The number of questions and answers published on the forum is steadily increasing with the growing interest in and adoption of the Oracle Application Development Framework (ADF).

The ADF Code Corner "Oracle JDeveloper OTN Harvest" series is a monthly summary of selected topics posted on the OTN Oracle JDeveloper forum. It is an effort to turn knowledge exchange into an interesting read for developers who enjoy harvesting little nuggets of wisdom.

twitter.com/adfcodecorner

http://blogs.oracle.com/jdevotnharvest/

Author:

Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
30-JUNE-2012

Oracle ADF Code Corner OTN Harvest is a monthly blog series that publishes how-to tips and information around Oracle JDeveloper and Oracle ADF.

Disclaimer: ADF Code Corner OTN Harvest is a blogging effort according to the Oracle blogging policies. It is not an official Oracle publication. All samples and code snippets are provided "as is" with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

If you have questions, please post them to the Oracle OTN JDeveloper forum:
http://forums.oracle.com/forums/forum.jspa?forumID=83

June 2012 Issue – Table of Contents

Free ADF online learning course Part 2 available

A second part of the advanced Oracle ADF online training is available for free. Lynn Munsinger and her team worked hard to deliver this second part of the training as close to the release of the first installment as possible. This two part advanced ADF training provides you with a wealth of advanced ADF know-how and insight for you to adopt in a self-paced manner.

Part 1:

http://tinyurl.com/advadf-part1

Part 2:
http://tinyurl.com/advadf-part2

Quoting Lynn:

"This second installment provides you with all you need to know about regions, including best practices for implementing contextual events and other region communication patterns.

It also covers the nitty-gritty details of building great looking user interfaces, such as how to work with (not against!) the ADF Faces layout components, how to build page templates and declarative components, and how to skin the application to your organization's needs. It wraps up with an in-depth look at layout components, and a second helping of additional region considerations if you just can't get enough.

Like the first installment, the content for this course comes from Product Management. This 2nd eCourse compilation is a bit of a "Swan Song" for Patrice Daux, a long-time JDeveloper and ADF curriculum developer, who is retiring the end of this month. Thanks for your efforts, Patrice, and bon voyage!"

Indeed: Great job Patrice! (+ all others involved)

Oracle Fusion Applications Design Patterns Online

The Oracle User Experience (UX) Team has published the design patterns used by Oracle Fusion Applications for building consistent layout and user interfaces. This for sure is a huge step ahead for customers adopting Oracle ADF that like some guidance when designing their application user interfaces.

Below is the slightly edited announcement text of the UX team:

"Oracle Fusion Applications user experience design patterns are published!

http://www.oracle.com/technetwork/topics/ux/applications/gps-1601227.html

These new, reusable usability solutions and best-practices, which join Oracle dashboard patterns and guidelines (http://www.oracle.com/webfolder/ux/applications/DPG/index.html) that are already available online

Now, the Oracle applications development community can benefit from the science behind the Oracle Fusion Applications user experience, too.

These Oracle Fusion Applications UX Design Patterns, or blueprints, enable Oracle applications developers and system implementers everywhere to leverage professional usability insight when [...] designing exciting, new, highly usable applications -- in the cloud or on-premise.

Based on the Oracle Application Development Framework (ADF) components, the Oracle Fusion Applications patterns and guidelines are proven with real users and in the Applications UX usability labs, so you can get right to work coding productivity-enhancing designs that provide an advantage for your entire business.

What's the best way to get started? We've made that easy, too. The Design Filter Tool (DeFT) selects the best pattern for your user type and task. Simply adapt your selection for your own task flow and content, and you're on your way to a really great applications user experience.

More Oracle applications design patterns and training are coming your way in the future. To provide feedback on the sets that are currently available, let us know in the comments section or use the contact form provided."

OEPE with ADF binding support available: Total Eclipse

The current release of Oracle Enterprise Pack for Eclipse, though in technology preview, brings Oracle ADF binding to the Eclipse IDE. You can download the Software from the link below:

*"**Oracle Enterprise Pack for Eclipse (12.1.1.1.0) Technical Preview** **New June 2012***
Certified on Windows 7/XP/Vista, MacOS, and Linux. Supported on JDK 6."

For many Eclipse users, ADF is new and therefore I expect them to need guidance and help in case they run into issues they don't know how to recover from. Similar, ADF users familiar with Oracle JDeveloper that want to give OEPE a try, will find things different in Eclipse and thus may have questions.

For both audiences I suggest to post issues to the OEPE forum on the Oracle Technology Network: I'll extend my OTN monitoring to include the OEPE forum on a daily basis to learn about developer needs, requirements and - of course - to catch bugs that need to be filed. From my side this is a part-time involvement, which means that the more ADF questions show on the forum, the more help I could need in answering them. The OTN forum for JDeveloper in my opnion wouldn't be the right place to go to unless the question is a generic ADF question that is not dependent on the integration in Eclipse.

For a start, here the **OEPE forum** link

https://forums.oracle.com/forums/forum.jspa?forumID=578

ADF EMG Sample Project (Wiki)

Steven Davelaar from Oracle (and active member of the EMG) started a project for the ADF EMG to categorize existing samples on a single "go-to" Wiki

"This project stores, lists and categorizes ADF samples created by various members of the ADF EMG. The samples may be undocumented, with only a small description in the page, or might be created as part of an article or blog post. In the latter case, a link to this documentation is provided with the sample.

For easy reference all samples have a unique identifier consisting of the author initials and a sequence number. The idea is that the ADF EMG community as a whole owns these sample pages, so if you know of samples that should be added, please register for this project and update the wiki page. The source code of the samples can be hosted in this project as well, using the SVN repository, so feel free to upload your own samples in this project!"

Bookmark this URL if you are interested in this little almanac of developer help

http://java.net/projects/smuenchadf/pages/ADFSamples#Categories

To contribute, providing your own sample, use this link:

http://java.net/projects/smuenchadf/pages/Home

Which JDeveloper version works with which version of WLS

Agreed, it is not easy to keep on top of this version-with-version matter and recent forum posts showed that some help is needed.

Timo Hahn blogged an overview of JDeveloper versions and their WLS version, plus – if required in the case of JDeveloper 11g R2 – the patches that need to be applied on top of WLS

http://tompeez.wordpress.com/2011/09/14/jdeveloper-versions-vs-weblogic-server-versions/

There is more in his blog post than the snapshot below, but for a quick overview, I copied the following table from Timo's blog

JDeveloper (ADF Runtime)	**Weblogic Server**	**Info**
11.1.1.2.0	10.3.2	
11.1.1.3.0	10.3.3	
11.1.1.4.0	10.3.4	
11.1.1.5.0	10.3.5	
11.1.1.6.0	10.3.5, 10.3.6	Integrated WLS is 10.3.5, stand alone WLS can be 10.3.5 or 10.3.6 (see Chris Muir on adf runtime 11.1.1.6.0)
11.1.2.0.0	10.3.5 + Sherman patch	Only available via MOS: patch #12611176 and patch #12556632
11.1.2.1.0	10.3.5 + Sherman patch UPDATE1	Only available via MOS: patch #12979653 and patch #12917525
11.1.2.2.0	10.3.6 + Sherman patch UPDATE2	Only available via MOS: patch #13656274 and patch #13656372 (see Patch Numbers for ADF Runtime Libraries Update to 11.1.2.2.0 for more info)
N/A	12c	No JDev or ADF runtime available (waiting for JDev 12c)

Source: Timo Hahn, http://tompeez.wordpress.com/

When JDeveloper IDE doesn't render the visual editor

Though with Oracle JDeveloper 11g the problem of the IDE not rendering JSF pages properly in the visual editor has become rare, there always is a way for the creative to break IDE functionality. A possible reason for the visual editor in JDeveloper to break is a failed dependency reference, which often is in a custom JSF `PhaseListener` configured in the faces-config.xml file. To avoid this from happening, surround the code in your `PhaseListener` class with the following statement (for example in the `afterPhase` method)

```
public void afterPhase(PhaseEvent phaseEvent) {
  if(!ADFContext.getCurrent().isDesigntime()){

    ... listener code here ...
  }
}
```

The reason why the visual editor in Oracle JDeveloper fails rendering the WYSIWYG view has to do with how the live preview is created. To produce the visual display of a view, JDeveloper actually runs the ADF Faces view in JSF, which then also invokes defined PhaseListeners. With the code above, you check whether the PhaseListener code is executed at runtime or design time.If it is executed in design time, you ignore all calls to external resources that are not available at design time.

Pretty URL in JDeveloper 11g R2

Many features planned for Oracle JDeveloper 12c find their way into current releases of Oracle JDeveloper 11g R1 and JDeveloper 11g R2. One example of such a feature is "pretty URL" - or "clean URL" as the Oracle JDeveloper 11g R2 (11.1.2.2) documentation puts it.

"A.2.3.24 Clean URLs

Historically, ADF Faces has used URL parameters to hold information, such as window IDs and state. However, URL parameters can prevent search engines from recognizing when URLs are actually the same, and therefore interfere with analytics. URL parameters can also interfere with bookmarking.

By default, ADF Faces removes URL parameters using the HTML5 History Management API. If that API is unavailable, then session cookies are used.

You can also manually configure how URL parameters are removed using the context parameter oracle.adf.view.rich.prettyURL.OPTIONS. Set the parameter to off so that no parameters are removed. Set the parameter to useHistoryApi to only use the HTML5 History Management API. If a browser does not support this API, then no parameters will be removed. Set the parameter to useCookies to use session cookies to remove parameters. If the browser does not support cookies, then no parameters will be removed."

See: http://docs.oracle.com/cd/E26098_01/web.1112/e16181/ap_config.htm#ADFUI12856

So basically, **what this part in the documentation says is**:

In JDeveloper 11g R2 (11.1.2.2), Oracle ADF Faces automatically removes its internally used dynamic parameters from the URL

You can influence the setting with the "prettyURL.OPTIONS" context option, which however is not recommended you to do because the default behavior is able to detect if the browser client supports HTML 5 History management or not.

In the latter case it the uses a session cookie and if this doesn't work, falls back to the "old" URL parameter adding.

The information that is not so explicit and clearly mentioned in the documentation is that this is only for ADF Faces parameters (such as _afrLoop, Adf-Window-Id, etc.), but not the ADF controller token (_adf.ctrl-state)! Removing the ADF controller token is an enhancement request that will be implemented in Oracle JDeveloper 12c

ADF Security Sample for Oracle JDeveloper 11g R2

ADF Security is an area not well understood by developers beginning with ADF. Recently I've seen more questions about this topic on OTN, especially in context of Oracle JDeveloper 11g R2. In January I did write an Oracle Magazine article titles "Security for Everyone" in which I summarize the ADF Security functionality. Providing a JDeveloper 11g R2 sample for download that has all the common requirements on board:

Article:

http://www.oracle.com/technetwork/issue-archive/2012/12-jan/o12adf-1364748.html

Sample:

http://www.oracle.com/technetwork/issue-archive/2012/12-jan/o12adf-524995.zip

ADF Insider Recording

http://www.oracle.com/technetwork/issue-archive/2012/12-jan/o12adf-524995.zip

Runtime Deployment of ADF Security protected applications

http://download.oracle.com/otn_hosted_doc/jdeveloper/11gdemos/adf_security1/adf_security1.html

http://download.oracle.com/otn_hosted_doc/jdeveloper/11gdemos/adf_security2/adf_security2.html

Hope this helps solving the security puzzle. Another common problem developers have when starting with ADF security is explained in this blog entry by Chad Thompson

http://e20labs.org/2012/06/07/notes-on-adf-security-for-11-1-2-2/

Solving the context menu problem with drag and drop in trees

The following drag-and-drop problem has been reported on OTN: An ADF Faces tree component is configured with a af:collectionDropTarget tag to handle drop events. The same tree component also has a context menu defined that is shown when users select the tree with the right mouse button. The problem now was - and I could reproduce this - that the context menu stopped working after the first time the tree handled a drop event. The drag and drop use case is to associate employees from a table to a department in the tree using drag and drop.

The drop handler code in the managed bean looked up the tree node that received the drop event to determine the department ID to assign to the employee. For this code similar to the one shown below was used

```
List dropRowKey = (List) dropEvent.getDropSite();
//if no dropsite then drop area was not a data area
if(dropRowKey == null){
   return DnDAction.NONE;
}

tree.setRowKey(dropRowKey);
JUCtrlHierNodeBinding dropNode = (JUCtrlHierNodeBinding)
tree.getRowData();
```

So what happens in this code? The drop event contains the *dropSite* reference, which is the row key of the tree node that received the drop event. The code then sets the key to the tree in a call to getRowDate() returns the node information for the drop target (the department). This however causes the tree state to go out of synch with its model (ADF tree binding), which is known to cause issues.

In this use case the issue caused by this is that the context menu no longer shows up. To fix the problem, the code needs to be changes to read the current row key from the key, then perform the drop operation and at the end set the origin (or model) row key back

```
//memorize current row key
Object currentRowKey = tree.getRowKey();
List dropRowKey = (List) dropEvent.getDropSite();
//if no dropsite then drop area was not a data area
  if(dropRowKey == null){
    return DnDAction.NONE;
  }
tree.setRowKey(dropRowKey);
JUCtrlHierNodeBinding dropNode = (JUCtrlHierNodeBinding)
tree.getRowData();
... do your stuff here ....
//set current row key back
tree.setRowKey(currentRowKey);
AdfFacesContext.getCurrentInstance().addPartialTarget(tree);
```

Note the code line that sets the row key back to its original value.

Programmatically disclosing tree and treeTable nodes

A common developer requirement when working with af:tree or af:treeTable components is to programmatically disclose (expand) a specific node in the tree.

If the node to disclose is not a top level node, like a location in a LocationsView -> DepartmentsView -> EmployeesView hierarchy, you need to also disclose the node's parent node hierarchy for application users to see the fully expanded tree node structure. Working on ADF Code Corner sample #101, I wrote the following code lines that show a generic option for disclosing a tree node starting from a handle to the node to disclose.

The use case in ADF Coder Corner sample #101 is a drag and drop operation from a table component to a tree to relocate employees to a new department. The tree node that receives the drop is a department node contained in a location. In theory the location could be part of a country and so on to indicate the depth the tree may have. Based on this structure, the code below provides a generic solution to parse the current node parent nodes and its child nodes.

The drop event provided a rowKey for the tree node that received the drop. Like in af:table, the tree row key is not of type oracle.jbo.domain.Key but an implementation of java.util.List that contains the row keys. The `JUCtrlHierBinding` class in the ADF Binding layer that represents the ADF tree binding at runtime provides a method named `findNodeByKeyPath` that allows you to get a handle to the `JUCtrlHierNodeBinding` instance that represents a tree node in the binding layer.

```
CollectionModel model = (CollectionModel)
your_af_tree_reference.getValue();
JUCtrlHierBinding treeBinding = (JUCtrlHierBinding )
model.getWrappedData();
JUCtrlHierNodeBinding treeDropNode =
treeBinding.findNodeByKeyPath(dropRowKey);
```

To disclose the tree node, you need to create a `RowKeySet`, which you do using the `RowKeySetImpl` class. Because the `RowKeySet` replaces any existing row key set in the tree, all other nodes are automatically closed.

```
RowKeySetImpl rksImpl = new RowKeySetImpl();
//the first key to add is the node that received the drop
//operation (departments).
rksImpl.add(dropRowKey);
```

Similar, from the tree binding, the root node can be obtained. The root node is the end of all parent node iteration and therefore important.

```
JUCtrlHierNodeBinding rootNode = treeBinding.getRootNodeBinding();
```

The following code obtains a reference to the hierarchy of parent nodes until the root node is found.

```
JUCtrlHierNodeBinding dropNodeParent = treeDropNode.getParent();
//walk up the tree to expand all parent nodes
while(dropNodeParent != null && dropNodeParent != rootNode){
   //add the node's keyPath (remember its a List) to the row key set
   rksImpl.add(dropNodeParent.getKeyPath());
   dropNodeParent = dropNodeParent.getParent();
}
```

Next, you disclose the drop node immediate child nodes as otherwise all you see is the department node. It is not quite exactly "dinner for one", but the procedure is very similar to the one handling the parent node keys

```
ArrayList<JUCtrlHierNodeBinding> childList =
                    (ArrayList<JUCtrlHierNodeBinding>)
treeDropNode.getChildren();
for(JUCtrlHierNodeBinding nb : childList){
  rksImpl.add(nb.getKeyPath());
}
```

Next, the row key set is defined as the disclosed row keys on the tree so when you refresh (PPR) the tree, the new disclosed state shows

```
tree.setDisclosedRowKeys(rksImpl);
AdfFacesContext adfCtx = adfFacesContext.getCurrentInstance();
adfCtx.addPartialTarget(tree.getParent());
```

The refresh in my use case is on the tree parent component (a layout container), which usually shows the best effect for refreshing the tree component.

How-to dynamically filter model-driven LOV

Often developers need to filter a LOV query with information obtained from an ADF Faces form or from other where. The sample below shows how to define a launch popup listener configured on the **launchPopupListener** property of the af:inputListOfValues component to filter a list of values.

```
<af:inputListOfValues id="departmentIdId"
    value="#{bindings.DepartmentId.inputValue}"
    model="#{bindings.DepartmentId.listOfValuesModel}"
    launchPopupListener="#{PopupLauncher.onPopupLaunch}" … >
    …
</af:inputListOfValues>
```

A list of values is queried using a search binding that gets created in the PageDef file of a view when a lis of value component gets added. The managed bean code below looks this search binding up to then add a view criteria that filters the query.

Note: There is no public API yet available for the `FacesCtrlLOVBinding` class, which is why I use the internal package class it in the example.

```
public void onPopupLaunch(LaunchPopupEvent launchPopupEvent) {
  BindingContext bctx = BindingContext.getCurrent();
  BindingContainer bindings = bctx.getCurrentBindingsEntry();
  FacesCtrlLOVBinding lov =
           (FacesCtrlLOVBinding)bindings.get("DepartmentId");
  ViewCriteriaManager vcm =
     lov.getListIterBinding().getViewObject().getViewCriteriaManager();

  //make sure the view criteria is cleared
  vcm.removeViewCriteria(vcm.DFLT_VIEW_CRITERIA_NAME);
  //create a new view criteria
  ViewCriteria vc =
         new ViewCriteria(lov.getListIterBinding().getViewObject());
  //use the default view criteria name
  //"__DefaultViewCriteria__"
  vc.setName(vcm.DFLT_VIEW_CRITERIA_NAME);
  //create a view criteria row for all queryable attributes
  ViewCriteriaRow vcr = new ViewCriteriaRow(vc);
  //for this sample I set the query filter to DepartmentId 60.
  //You may determine it at runtime by reading it from a managed bean
  //or binding layer
  vcr.setAttribute("DepartmentId", 60);
  //also note that the view criteria row consists of all attributes
  //that belong to the LOV list view object, which means that you can
  //filter on multiple attributes
  vc.addRow(vcr);

  lov.getListIterBinding().getViewObject().applyViewCriteria(vc);
}
```

Note: Instead of using the `vcm.DFLT_VIEW_CRITERIA_NAME` name you can also define a custom name for the view criteria.

Partial Submit vs. Auto Submit

Partial Submit

ADF Faces adds the concept of partial form submit to JavaServer Faces 1.2 and beyond. A partial submit actually is a form submit that does not require a page refresh and only updates components in the view that are referenced from the command component PartialTriggers property. Another option for refreshing a component in response to a partial submit is call `AdfContext.getCurrentInstance.addPartialTarget(component_instance_handle_goes_here)` in a managed bean. If a form contains required fields that the user left empty invoking the partial submit, then errors are shown for each of the field as the full form gets submitted.

Autosubmit

An input component that has its autosubmit property set to true also performs a partial submit of the form. However, this time it doesn't submit the entire form but only the component that triggers the submit plus components referenced it in their PartialTriggers property.

For example, consider a form that has three input fields **inpA**, **inpB** and **inpC** with *autosubmit=true* set on inpA and *required=true* set on inpB and inpC.

use case 1: Running the view, entering data into inpA and then tabbing out of the field will submit the content for inpA but not for inpB and inpC. Furthermore, none of the required field settings on inpB and inpC causes an error.

use case 2: You change the configuration of inpC and set its PartialTriggers property to point to the ID of component inpA. When rerunning the sample, entering a value into inpA and tabbing out of the field will now submit the inpA and inpC fields and thus show an error for the missing required value on inpC.

Internally, using *autosubmit=true* on an input component sets the event root to just this field, which good to have in case of dependent field validation or behavior. The event root can extended to include other components by using the Partial Triggers property on these components to point to the input field that has *autosubmit=true* defined.

PartialSubmit vs. AutoSubmit

Partial submit set on a command component submits the whole form and leaves it to the developer to decide which UI component is refreshed in response. Client side required field validation (as well as the server side equivalent) is not disabled by executed in this scenario. Setting immediate=true on the command item to skip validation doesn't help as it would also skip the model update.

Auto submit is a functionality on the input components and also performs a partial form submit. However, in addition an event root is defined that narrows the scope for the submitted data and thus the components that are validated on the request.

To read more about this topic, see:
http://docs.oracle.com/cd/E23943_01/web.1111/b31973/af_lifecycle.htm#CIAHCFJF

ADF Code Corner

OTN Harvest Spotlight

- Navaneetha Krishnan

Blog: http://navaneeth-on-fusion-middleware.blogspot.com/

Twitter: https://twitter.com/navan1983

ACC: What is your current role?

I am an ADF/Web center Consultant working for Keste (an Oracle Platinum Partner) based out of Dallas, Texas. My current responsibilities are ADF/Web center Software Development, Architecture Design & gathering Business requirements from the client. I am currently working for C&S Wholesale Grocers – developing an ADF/Web center based portal.

ACC: What is your IT background?

After obtaining my master's degree in computer science in 2005, I worked for Intel as a Software Intern building OLAP/OLTP solution using Oracle Database and generating reports using C#.

I moved on to SAP Labs for my first job, where I worked on integrating SAP with Outlook to bring CRM functionality in outlook using J2EE, Java & .Net technologies.

After a year & half with SAP, I moved on to my dream company – Oracle, where I started work on ADF, Web Center & SOA Technologies working with multiple Fusion Applications – TCM & CRM. I also worked on building an ADF based mobile solution for CRM Sales Module.

I moved to US in late 2010 and I have been working as an ADF/Web center Consultant building solutions based on Fusion Middleware Platform for multiple clients (Emerson Network Power, Qualcomm & CSWG).

For the last 5 years, I have been spending lot of time building my knowledge & developing solutions based on ADF, Web center, SOA and other Fusion Middleware technologies.

ACC: How do you currently use Oracle JDeveloper and ADF?

I use Oracle JDeveloper for building ADF & Web center Portal applications – ADF Business Components, ADF Task flows, Web services, Web center Portal & Web center Portlets.

As I have mentioned earlier, I am currently working for CSWG developing ADF/Web center based Portal & also integrating OBIEE reports in the Portal.

ACC: So far, what has been your biggest challenge in building Java EE application with Oracle ADF?

During the initial days, the challenge was about the best practice. I could achieve multiple solutions for a problem but I was not sure which one was the standard & recommended solution.

It took lot of time understanding best practices/paradigms. In the last couple of years, with lot of niche UI technologies coming up, the clients want similar UI components & look and feel in ADF/Web center. With powerful ADF Rich faces components, I have been successful to some extent.

ACC: Which feature of ADF was the greatest benefit to your project?

Declarative Software Development, Bounded Task flows, Rich set of ADF Faces Components & skinning.

ACC: Apart from the on line help, what have been your most valuable sources of ADF knowledge?

JDeveloper Forums, ADF Code Corner & ADF EMG discussions. Lot of experts have been actively blogging about various use-cases that they encounter in their projects have been really useful.

ACC: Are you in any way actively involved in the ADF Community?

I am very active in JDeveloper forums and provide suggestions & sample solutions for the various posts. I also blog at. http://navaneeth-on-fusion-middleware.blogspot.com . I am also a member of ADF EMG (Enterprise Methodology Group).

ACC: What are your top 3 ADF best-practices for ADF developers?

1. Use framework wisely. You could achieve a solution for a problem but always think/check/verify if it is based on the recommend standard/practice.
2. Use bounded task flows whenever necessary as it provides lot of features such as reusability, easy plug & play, transaction support.
3. Follow proper naming conventions for the various artifacts created in JDeveloper. This ensures better readability for other team members & easier understanding for debugging.

ACC: What is your recommendation for developers new to Oracle ADF. How and where should they start?

If the developer has prior understanding on Java, J2EE then attending trainings or taking up couple of tutorials will make them productive in a project. Fusion Developer's guide & Web interface guide are bible for any developer. Although these guides are huge, but there is no materials that could provide better information.

The following books are also really useful:

1. Oracle Fusion Developer Guide: Building Rich Internet Applications with Oracle ADF Business Components and Oracle ADF Faces (Frank & Lynn).

2. Quick Start Guide to Oracle Fusion Development (Grant Ronald).

ACC: By your experience and estimation, how long does it take for developer new to ADF to learn it?

Depends on developer & current skill sets. It takes a short amount of time to be productive and a long time to master.

After some time, the experience in working on a complex project(s) leads a developer to learn several concepts and master the technology.

ACC: You are very active in answering questions on the Oracle JDeveloper forum. Is there anything people can do to make your life on the forum easier?

Yes.
The users should mention the post clearly along with JDeveloper Version, what they tried & what is not working along with any other additional information.

The JDeveloper forum is a very active forum with lot of active participants and posting a forum post with above guidelines, I can bet you will get the better solution with shorter time.

If the forum poster finds solution in other means, please post your solution. It will be definitely useful for some other user on another day.

ACC: ADF Genie grants you a wish, what would you ask for?

I would love to have these features.

- Bookmarking support for Bounded Task flows with Pages.
- In-built Pagination Support for ADF Tables.
- Better export to excel, export to PDF support, better printable pages.
- Better solution for Mobile version of ADF.

RELATED DOCOMENTATION

☒	
☒	
☒	