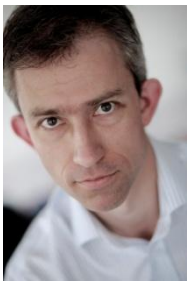
# AMIS on Fusion Middleware



*Column for the ODTUG Technical Journal – Q1 - 2010*

*This edition by Lucas Jellema (Oracle ACE Director and CTO of AMIS from Nieuwegein, The Netherlands)*

## Follow the leader or is ADF really Advanced Dog Food?

"No one builds an application on the scale of Fusion Applications. Well, except Oracle of course – as they are building Fusion Apps. So what should I care about it? By the time I need all the scalability, robustness and functional richness that Fusion Applications require for my own application, I will come back and ask for FMW (Fusion Middleware). In the mean time, all that Fusion Apps' involvement with FMW can buy me is a lot of overhead that will slow down my application and my developers." A not uncommon point of view. And one worth paying attention to, as things have indeed changed at Oracle.

In the past, Oracle made money on development tools. The distant past I would almost say. Back then, Oracle created development tools for *our* sake, even according to our desires – us being custom application developers. At some point in time, Oracle instructed its own EBS development team to use those tools as well to test and prove them – 'we eat our own dogfood' was the tagline that went with it.

Then the world and Oracle changed. Most development tools on their own don't sell well enough anymore to even break even. Companies like Oracle need development tools to be able to offer a full suite to their customers, which is important during product selection of database and application server. However, they do not make money from those tools by themselves.

At the same time, Oracle has embarked on a project as challenging as Fusion Applications. Oracle needs tools that support their own development teams overcome technical and functional challenges in a productive and accessible way, with good integration with database and middleware. And those tools should be geared first and foremost to their own teams and challenges – they cannot be the left-overs from whatever we as customers would like to have. So Oracle created Fusion Middleware. Based on over a hundred open industry standards and with an application with the size and complexity of Fusion Applications in mind.

Now instead of them using the dogfood they are trying to sell to us, we now get to eat the dogfood they prepared for themselves. Pretty Advanced Dog Food one could say. By the time we get our

hands on the FMW tools and technology, it has been tried and tested in the many development teams at Oracle.

Returning to the opening statement in this article, we can probably conclude that the functional richness, the strategic and long term viability of the stack, the robustness and scalability of FMW are enough for our purposes. So what about that overhead that might be in the way of custom applications at a much smaller scale than Fusion Apps? Well, fortunately that is easy to find out. While it normally is quite hard to prove the aspects mentioned in the beginning of this paragraph, the overhead we may fear is quite easy to establish. Just develop a small application and see whether the many features in FMW that you do not need are bothering you, to an extent where you cannot be productive. Create a few pages with relatively mild functional requirements and determine if the visual and declarative development style, the reuse mechanisms and the Fusion Apps specific elements really prevent lean and mean application development.

It only takes a relatively small proof of concept project to gauge JDeveloper, ADF and other FMW components in its own right or against other development stacks. I would suggest that the fairly steep learning curve – which comes with most standards based web development technology stacks – is probably the only thing between you and rich, slick custom application. And even that has been flattened a lot by the wizard driven, declarative style of development. Don't forget that Oracle has a lot of developers too that it needs to make productive – and by no means are all of them more experienced or savvy than you or me.

## How Fusion Applications benefits us?

Okay, perhaps the Fusion Apps focus is not necessarily in our ways when doing our own development effort. But does it buy us anything for developing smaller applications?

Many new features make it into the FMW products – and let's focus on ADF for now – because of specific requirements from the Fusion Apps teams. Sometimes very advanced features, that are not found in any other application development tool stack – as no other vendor or open source initiative perceives the need for such features.  Many of the Data Visualizations Tags have been influenced by Fusion Apps or were specifically created for a certain module – like the Hierarchy Viewer that was requested by the Human Capital Management team of Fusion Apps. Of course we can benefit and

use those components ourselves. If not, nothing is lost – however, if we do a lot can be gained.

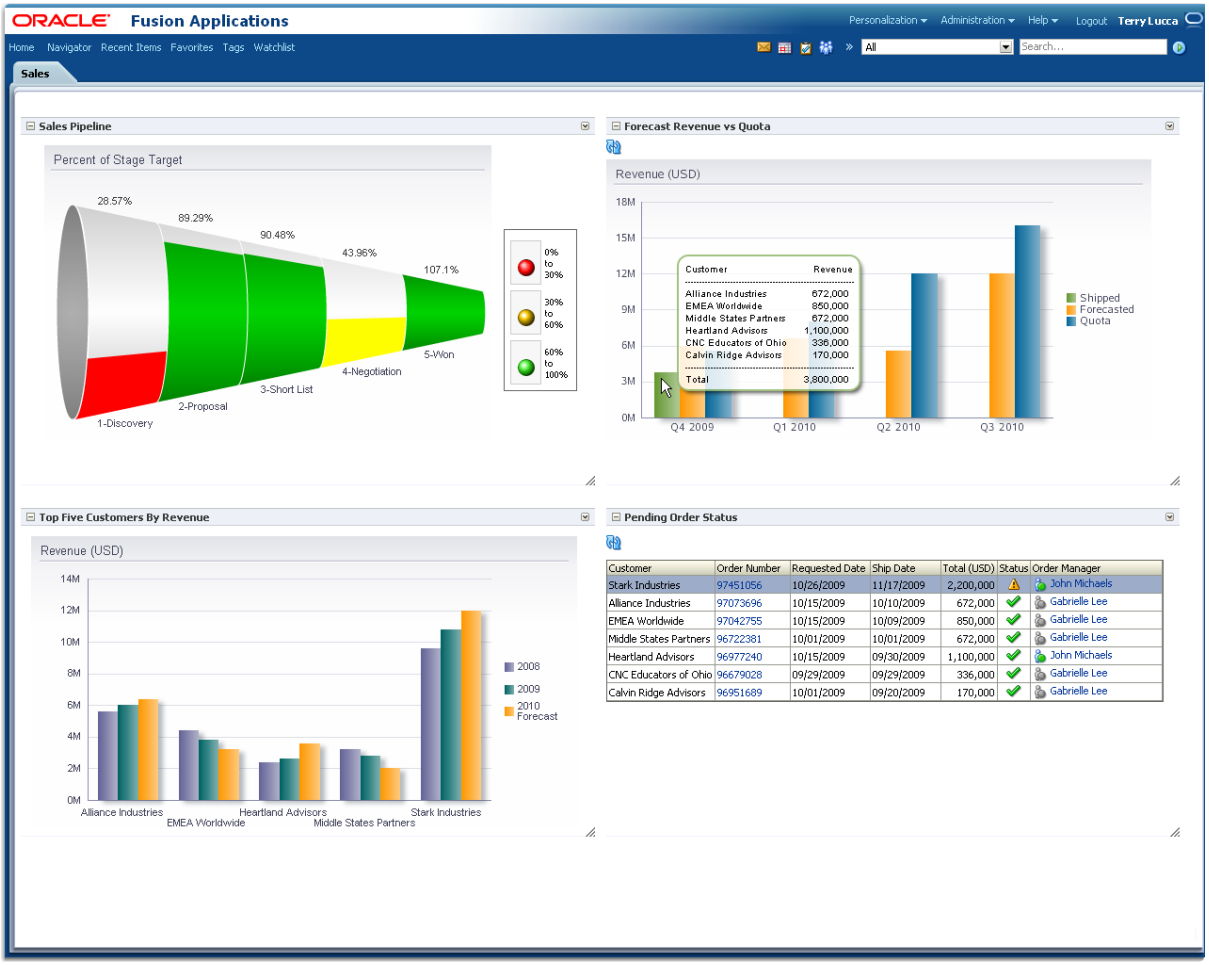

Other examples of fairly large features that were commissioned by Fusion Apps include the Calendar and Gantt Chart components, the BAM (Business Activity Monitoring) Data Control and the recent Fusion skin for modern, attractive styling of the UI (and even the fusion-projector skin for when we want to demonstrate our ADF application using table-top projectors). The carousel component – added in the PS1 release of last November – also seems primarily intended for demo purposes, as it looks pretty cool.

It is clear to see how the main themes for Fusion Applications – such as manage by exception, task and workflow driven, focus on collaboration, integrated communication and SaaS readiness - translate into components and features in ADF, WebCenter and other FMW products. And can be copied in our own applications.

Take for example at the concept of “Manage by Exception”. Instead of monitoring all data in the application, users focus on *what* they need to know *when* they need to know it: Business Status, Alerts/Notifications and system derived Recommendations. Subsequently it is about what they need to do – To Do Lists and Task oriented pages – and how they should do it - Process and Workflows – and with whom - Contacts, Communication, Collaboration.

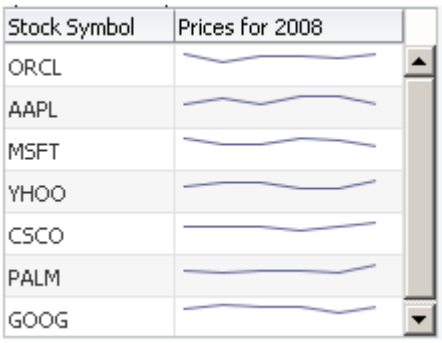
Embedding various forms of Data Visualizations – from bar chart to pivot table based heat map – presenting data in much more intuitive formats than plain text or tables, that support drill down from high level aggregate to detail cause, that can be active (server push automatically refreshes

them, occasionally driven by the Business Activity Monitor engine) is one way of striving for exception based management. One that we can easily mimic in our own applications – given the ease with which ADF DVTs can be used to create rich graphs.

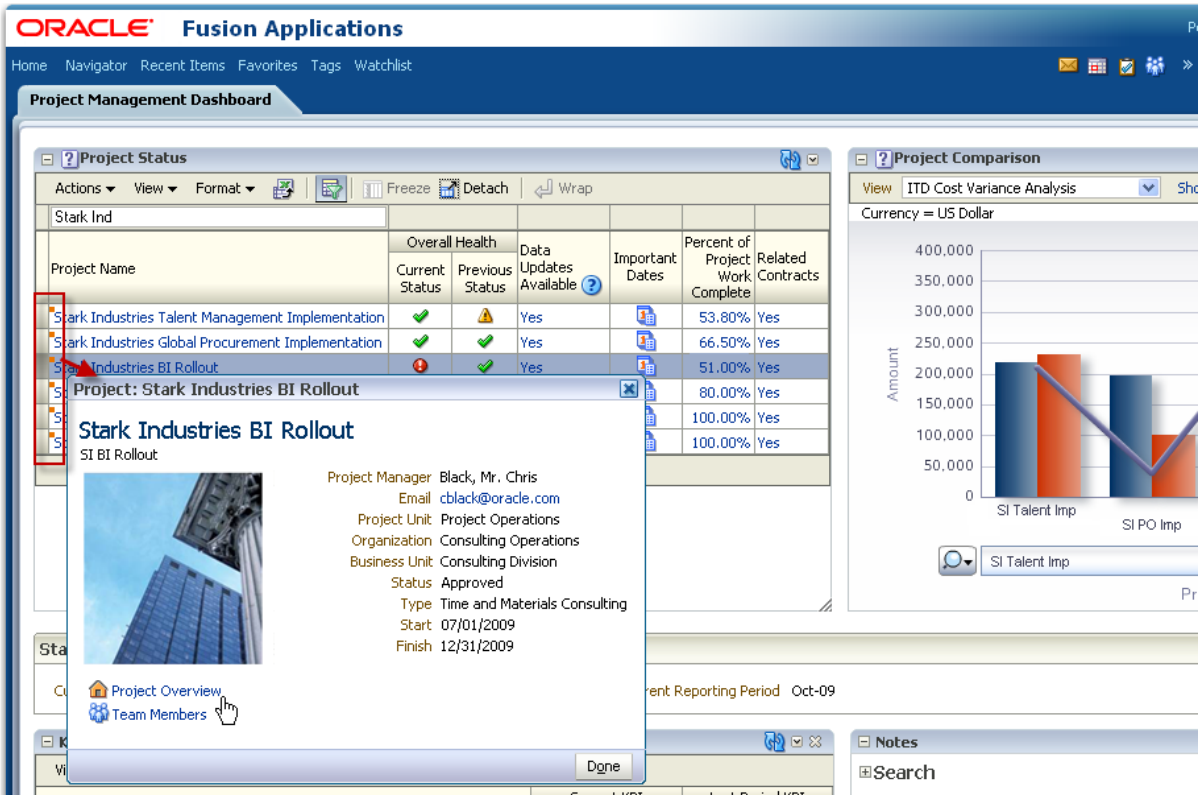



## Smaller features driven by Fusion Applications

Sometimes Fusion features are fairly small. The Sparkchart and the ContextInfo components, both add in PS1 (November 2009) are examples of this. Sparkcharts are simple, condensed graphs that display trends or variations, often in-line in the column of a table.



However, even small features can be important for providing a consistent way of implementing application wide facilities and end user interaction. The ContextInfo component for example is used all over Fusion Applications – to indicate the availability of additional information in the context of a certain record or field. This component manifests itself through a little orange square – though it can be skinned with a different appearance - in the upper left-hand corner of the component it is associated with.




When the context info marker is clicked on, a popup can be shown with the context information. The popup you see for Stark Industries BI Rollout is the effect of activating the context info component. And it demonstrates the purpose rather nicely and effectively: whenever the user sees a context Info marker she knows that additional information associated with that field is available. The user can click on the marker and the context information is presented in whatever way the developer feels is most appropriate, though usually a popup will be used.

This context info facility can easily be implemented in our own applications. The basics are: add an af:contextInfo child component to the component that should have the visual marker. Then add a showPopupBehavior to the contextInfo, referring to the popup that should be displayed.

The code used for the example in the picture:

```
<af:inputText label="First Name"
              value="#{item.firstName}" columns="15"
              id="it7">
  <f:facet name="context">
    <af:contextInfo id="contextInfoFirstName">
      <af:showPopupBehavior align="beforeStart"
                            popupId="::FirstNameContextPopup"
                            triggerType="contextInfo"/>
```

```
    </af:contextInfo>
  </f:facet>
</af:inputText>
```

For details on how to implement context info in your own ADF application, see: http://technology.amis.nl/blog/6583/adf-11g-contextinfo-to-implement-a-common-fusion-applications-pattern.


## Customization and Personalization

Rich and interactive components such as the accordion, (panel)dashboard, panel boxes, tables and splitters introduce new challenges in addition to new functionality. All these components allow the end user to manipulate the appearance of the user interface. The page will first render with the settings made by the developer. And then the user can start collapsing boxes, opening others, rearranging the dashboard, changing the location of the panel splitters and hiding some table columns and changing the position of others. Which results in a wonderfully personalized page. Now when the user navigates away from the page and then returns to it: will and should it render as the developer designed it or the way the user personalized it? And when user exits the application altogether and returns to it the next day, what should happen them?

This personalization challenge is covered by the user customization or personalization change persistence framework in ADF – undoubtedly created on the vehement urging by the Fusion Apps teams in their quest for 'design time at run time' – the ability for application managers and users to tailor the application to their personal needs and mood swings.

Related to this personalization is customization – perhaps the biggest productivity booster in all of FMW for the Fusion Applications development teams. Let me explain: suppose you have to build an application for managing your general ledger. That is one challenge. However, then you are told that your application is to be used by local and federal governments and in various industries – and that it should have specific facilities for each category of users. And to make it even more interesting, the application will be used globally, in up to 200 countries and across languages and cultural boundaries. And it would be nice if the application could be rolled out in a light and an advanced edition next to the standard edition.

A first approach to this challenge could be something like: let's build the application – for the most general use case. Then copy it for each specific flavor – such as the light variant for local governments in rural Indonesia – and make the specific modifications required for that flavor. There are two obvious flaws with this approach. First of all, it would not work for large organizations that operate globally – as they would not want to install several slightly different copies of what is basically the same application. Even worse, the maintenance effort for this approach would be mind

boggling: every simple change in the base application would have to be applied to every copy. Before too long, the burden of keeping all versions in synch would become unbearable.

The customization features of Oracle Metadata Services (MDS) framework allows us to create customizable applications. These are applications that consist of a base application along sets of customizations for each "flavor" supported by the application. Our general ledger application would have customization sets for all supported values in each customization layer. Customization layers are the dimensions along which the application varies – such as industry, region and edition of the application. Customizations are the deltas that need to be applied to base artifacts like JSF pages, ADF Task Flows or Resource Bundles to construct the specialized flavor of that particular artifact.

Needless to say that customization is a life saver for the Fusion Applications teams that face exactly the multi-flavor challenge described earlier on. The same mechanism is available to us – and especially when our custom built applications need to support various user groups across our enterprise or when we offer a SaaS style application will the customization facilities in FMW be a boon.

Note: WebCenter Composer provides a different style of run time modification for ADF web applications that allows content editors and application managers to customize a running application by adding, editing, rearranging or hiding components and content.

## Conclusion

Oracle has created JDeveloper, ADF and WebCenter primarily for Oracle itself. Oracle needs it for developing Fusion Applications. We have been granted the right to use their tool stack too. As a result, we inherit the functionality, reuse features and development facilities required for Fusion Apps as well as the robustness, scalability and real-world-hardened-ness of the FMW platform. Whether the ability to create Fusion Applications with this technology stack introduces unbearable overhead in developing small scale applications is something you need to determine yourself – that should not take too long.

*AMIS is an Oracle Certified Advantage Partner, founded in 1991 and located in The Netherlands. AMIS are friends of the Oracle community, for example through frequent presentations at the ODTUG Kaleidoscope and Oracle Open World conferences and its popular technology weblog: http://technology.amis.nl/blog. AMIS staff were awarded two Oracle ACE Director and two Oracle ACE nominations in recent years. This edition of the column AMIS on Fusion was written by Lucas Jellema, Oracle ACE Director and CTO of AMIS and author of the upcoming Oracle SOA Suite 11g Handbook (Oracle Press). He can be contacted at lucas.jellema@amis.nl*