professional it software & services

From Oracle Forms to Oracle ADF and JEE

Modernizing Oracle Forms applications to Oracle Application Development Framework and the JEE Architecture

PITSS.CON 8.0.3

White Paper, November 2010

Introduction

Oracle offers nowadays an extensive spectrum of development solutions. The technical capabilities for creating new database applications are almost unlimited. But what about existing, legacy Oracle Forms applications? Can we integrate them into the new JEE architecture? To what extent can we take advantage of the new technologies while avoiding an expensive and risky re-write? This document's purpose is to dig deeper into these questions, take a look at both the opportunities and the challenges brought by such a change of the software architecture, and of course, present the perspective and solutions offered by PITSS.CON products like ADF Assistant, Application Engineering, Business Logic Assistant or Web Services Assistant.

But first of all:

Why JEE? Why ADF? Why Change?

The Web is changing our business processes and, along with them, users have different requirements for the software they use. Among the applications we have developed with Forms over the last years, most can be Web deployed by migrating to the latest supported version of Oracle Forms. However, there are business scenarios where the software could be further optimized to meet the requirements of today's Web based architectures even better: richer user interfaces, middle tier save-points… For such business scenarios Oracle advises a re-write to the JEE architecture and integration with the rest of the modules on the application server.

"[...] Oracle recommends customers to web deploy their existing Forms and Reports applications, consider the opportunities of new development in J2EE using JDeveloper and ADF and integrate these applications together on the application server, sharing common services and business logic."[1]

JEE promises long term easier maintenance for the applications through open standards, simplified connectivity with external products. However, for all the benefits it brings, JEE appears to database developers as being extremely complex and difficult. Oracle Forms developers (more generally, 4GL developers) are used to very high productivity, which cannot be equaled in the JEE world: it is difficult to offer clients the same quick development time they were used to. This is why Oracle recommends the Oracle Application Development Framework (ADF) as the solution to ease the challenge of implementing JEE design patterns (Model View Controller, MVC) with its visual, declarative environment that minimizes the need to write code. But are the things really as easy as they seem?

The Re-Write Challenge

The recipe for a successful Forms- to-ADF software re-engineering project:

1) **Thoroughly understand the Forms application that needs to be re-written, in all its complexity** – legacy applications were developed time ago and often evolved to a notable size and complexity. In most cases the Oracle Forms developers are no longer with the company and the documentation is not sufficient or not available at all to answer critical

[1] Oracle Forms – Oracle Reports – Oracle Designer: Statement of Direction, Oracle, October 2008

questions. The Java developers assigned with rewriting the applications find it difficult to fully understand the various inter-connected components or may require training on the Forms runtime architecture.

2) **Have a deep knowledge of the new technologies, and transpose all application functionality into the new architecture, while even improving the application** – this is a challenging task because Forms and JEE are fundamentally different. Plus, the application change is only successful when the new version offers a better user experience and superior technological capabilities, like a 'self service' style of user interface. After all, when we invest money in a modernization process we may not want to end up with a clone of the initial application.

3) **Establish project management processes to schedule and monitor progress during the transition** – Software projects based on new technology often fail caused by unforeseen implementation challenges, lack of experience, and missing implementation standards. Project management needs to monitor continuously the schedule, budget and quality constraints applied to the effort to ensure a successful initial project.

Behind the three above simple ingredients we can see the real challenges:

Skills

A mixed set of skills: both PL/SQL, Forms AND Java, JavaScript, XML, JEE expertise is vital to the project success. Java skills may be easy to find on the market, but the key is an understanding of both Oracle Forms complexity and the capabilities of the new technology and software architecture.

User Interface

Adjusting to the Web model strengths and weaknesses: for the increased accessibility the new user interfaces offer, browser based applications also have their limitations. Oracle Forms offers a very data-efficient architecture, able to query and display in an applet thousands of records at once, hundreds of fields on a single page, access the client computer and perform host commands or file access. This is why a typical Forms application needs to be redesigned to be able to run in a simple browser, by displaying sets of 20 records at a time, distributing the hundred fields on several pages or reducing the degree to which they affect the client computer. For those applications where such adjustments are not possible, staying with the latest supported version of Forms is still the best option until the new technology may offer better suited alternative solutions in the coming years.

Business Logic

Forms applications contain mixed business logic performing in the same unit of code actions like: user interface actions, data validation, data manipulation, etc. When taking this code to ADF, we need to **separate the business logic into its components**, and redefine it according to the MVC design pattern, in order to create a real ADF architecture.

```
if p_dept_id < 0 or p_dept_id > 99 then
  message('Invalid department number');
else
  if pac_util.check_dept(p_dept_id, p_dept_name, p_location_id) then
    insert into dept (id, name, location_id)
              values (p_dept_id, p_dept_name, p_location_id);
  end if;
  go_block('dept');
end if;
```

Figure 1: Example of mixed business logic in a Forms application trigger

The possible solutions for managing business logic are:

1) **Translate entire code to Java** – such an approach could succeed only when a 100% manual re-write would be performed. The entire investment in the Forms business logic would be lost in such a case. There are tools on the market that promise to automatically convert the whole Oracle Forms code into Java. The resulted code is sequential and procedural, like PL/SQL, but not truly object-oriented, like a Java code is supposed to be. In addition, it contains the same mix of data-handling business logic with user interface code, like the initial code did and includes artificial constructs. This code is difficult to understand and in most cases not recognized as Java standard code which leads to a higher maintenance effort in case the application will be further enhanced. But the main reason we believe this is not the solution is: We are talking about database applications - the business logic in Forms manages mostly data, so the majority of the logic belongs closer to the Oracle database than to the client tier.

2) **Leave the code in PL/SQL and place it in the database** – the data access or data manipulation code is typically faster in the database, where it minimizes the network traffic. PL/SQL in Oracle Forms applications as well as in the database has managed all business needs over years and will do it in the future as well. That's risk reduction. However, if we choose to move to JEE, some code artifacts are not intended for usage in the database: user interface code, like navigation, field validations belong closer to the client tier.

3) **A mixed approach:** user interface code in Java, data manipulation in the database. This is the solution we recommend and will be explained in detail in the following section.

"These days, you have a number of choices when it comes to writing software to run against the Oracle database. You can use Java and JDBC; you can use Visual Basic and ODBC; you can go with Delphi, C++, and so on. You will find, however, that it is easier to write highly efficient code to access the Oracle database in PL/SQL than it is in any other language."[2]

"A PL/SQL program with effective cursor handling can execute many times faster than a Java program written to perform the same task running on an application server."[3]

2 Oracle PL/SQL Programming, Fourth Edition, Steven Feuerstein with Bill Pribyl, 2005
3 Oracle PL/SQL for Dummies, Michael Rosenblum, Paul Dorsey, 2006

The PITSS.CON Solution

Our solution combines the deep understanding of Oracle Forms and Reports applications architecture and analysis capabilities of the PITSS.CON suite with the powerful re-engineering capability of the “Application Analysis”, “Application Engineering” and “Application Modernization” modules.

PITSS.CON loads the Forms and Reports application into its repository, parses all of the code and then constructs the whole structure of object inter-dependencies. This in depth knowledge of the entire application allows the PITSS.CON solution to maximize the amount of Forms components that can be naturally transposed into the ADF world.

Architecture

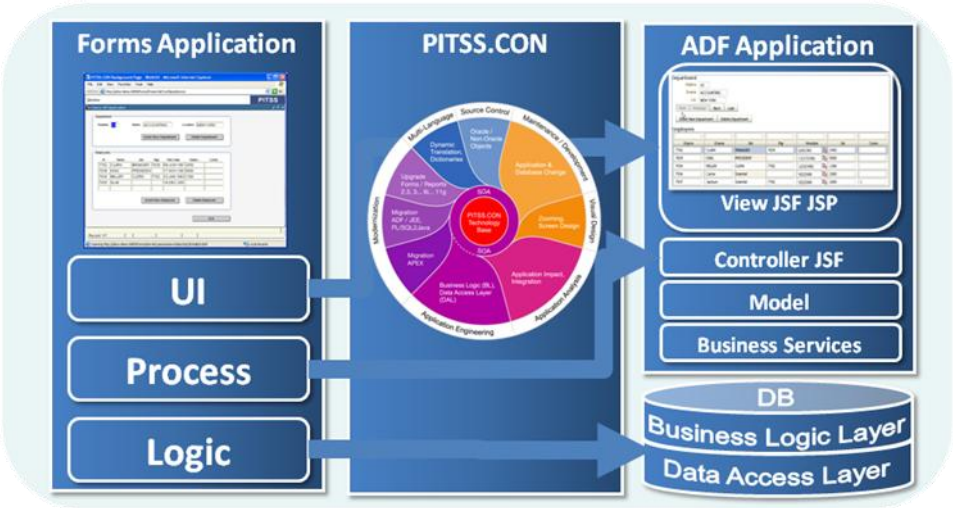

Figure 2: PITSS.CON Forms-to-ADF Migration Architecture

The most part of the Forms business logic is handling data. Such logic naturally belongs to the Oracle database. PITSS.CON assists the migration of the data intensive business logic to the database, where it creates a Business Logic Layer. After the code is migrated from the Forms and Reports to the database, this layer can be accessed not only from the Oracle Forms and Reports, but also now from Oracle ADF, Oracle APEX, or could even be exposed as Web Services to be accessible by any other User Interface environment. That's investment protection: no matter how the User Interface environment changes in the future, we will be able to modernize the application or individual components a lot easier.

Keeping the business logic in the database is also an excellent solution for the frequent situation where the code is accessed from multiple applications, and we would need to keep some of these applications in Forms, while redefining others within ADF. In this case, the business logic needs to be available to multiple interface worlds, with the common denominator being the Oracle database. This way the application becomes Service-Oriented Architecture (SOA) enabled, because the whole SOA concept is about components reuse and integration.

The rest of the application contains mostly Forms components that can be replicated within the ADF MVC pattern, using ADF Faces, JSF and ADF Business Components.

Process

1) **Application Analysis** – the first step is loading the entire application into the PITSS.CON repository and performing a detailed analysis. Here PITSS.CON will not ask for some disparate XML versions of your Forms modules, but will load the entire application, with all its components: FMB, MMB, PLL, OLB, RDF, SQL, etc. and also the database schema definitions. All these 'pieces of puzzle' are important in order to perform a correct Application Analysis, where all objects inter-dependencies are analyzed and properly documented.

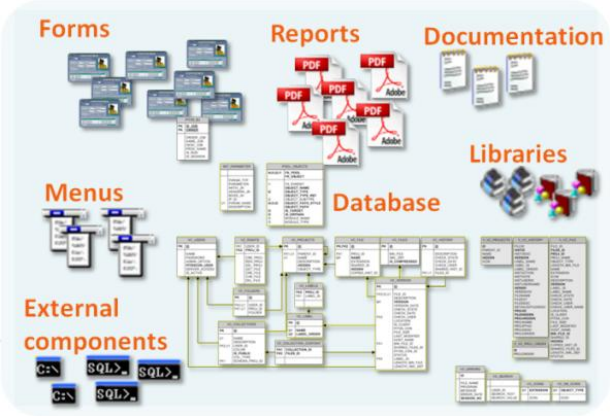

Figure 3: Typical Forms application components

At this phase we plan the strategy based on the application specific, complexity, user interface challenges and the degree to which the application needs to be further modified to adapt to the Web model.

2) **Eliminating Redundancies, Unused Objects Analysis** – the next step is to clean up the application and reduce it to its pure core functionality. This is an ideal starting point to gain back control of the existing application and to create essential documentations before the journey of modernization begins. From our experience, this step identifies at least 20-30% of the application objects as being redundant: unused tables, libraries, duplicated or unused code units, or even full obsolete functionalities, like calendar functionality for a date field, for instance, not necessary any more in the new JEE world.

Figure 4: PITSS.CON Unused Objects Analysis Report

3) **Migrating Business Logic to the Database** – this step is laying the foundations of a successful modernization to a Service-Oriented Architecture. We can significantly save time and budget by using tools like:

- **BL Assistant** – extracts business logic from the Forms and Reports applications. After analyzing and visually presenting the degree to which the entire code can be migrated to the database, it performs the migration, re-mapping the generated database packages to the initial application. The created database code is not only available to the Forms applications, but also to ADF applications or can be further exposed as Web services.

- **Web Service Wizard** – exposes any stored functionality as a Web service, opening the application to the outside world.

- **Application Analysis and Application Impact** – lets you quickly and precisely analyze the impact of proposed changes, identify obsolete or problematic code, and streamline the transition to a Service-Oriented Architecture.

4) **Tool-assisted application re-engineering** to ADF - After freeing the application of the most of its business logic, the user interface re-creation is almost as easy as a press of a button. The ADF Assistant will automatically create the corresponding ADF objects:

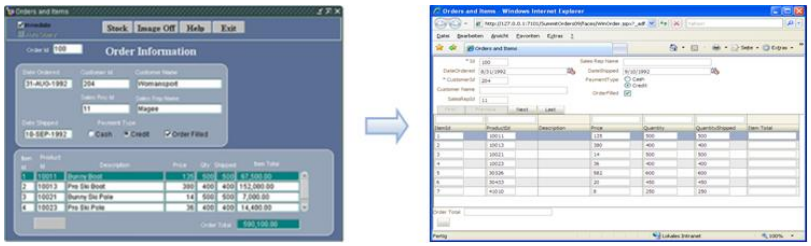

Figure 5 Example of a page automatically generated with PITSS.CON ADF Assistant

- **Business Services** with ADF Business Components;

- **Data access objects** that ease the process of performing database logic calls from the ADF application;

- **Finishing the model, view and controller layers**, analyzing each component of the Forms application – windows, canvasses, blocks, items, etc and creating corresponding ADF objects; taking into consideration each individual object property and the extent to which it can be reproduced into the ADF world;

- Including **user-interface Java methods** by translating the PL/SQL business logic into Java, providing automated support for variable re-mapping and calling of stored business logic via the data access objects.

5) **Post-Generation Changes Phase -** The result of the above process is thoroughly documented, which provides a great resource for developers working on the converted application and for an estimation of the remaining fine tuning work. Typically, manual post generation changes are necessary for:

   - fine-tuning the web layout – if desired, the tool-generated pages can be further refined using Oracle JDeveloper
   - re-defining the page navigation using pre-built components
   - refining and enabling the tool-translated Java methods
   - fully defining the components for which a 100% tool-supported re-creation is not yet possible or finding workarounds to the web model limitations (like client file system or registry access, for instance)

Business Benefits

From decades of experience in offering modernization services and products for Oracle Forms and Reports applications, we know the real complexity of changing from Oracle Forms to the JEE. We are confident that our approach is the most complete one, the only viable, long-term solution, bringing real savings in time and money.

- **Productivity –** much higher efficiency compared to manual re-creation from scratch by bulk-performing similar operations and maximising the degree in which Forms components are naturally integrated within the JEE architecture and re-used.
- **Effectiveness –** very high percentage of functionality automatically recreated within ADF – we offer not only a solution for the user interface, but PITSS.CON also offers excellent tools for business logic analysis and support in migrating it to the database.
- **Consistency:** applying consistent development practices leads to consistent look and feel, naming conventions and coding practices, easier maintainability.
- **Natural architecture –** we strive to offer automated solutions that lead to applications complying with the JEE architecture, which are easy to understand and further develop, having a natural design, identical to one that would be obtained by manual rewrite.
- **No proprietary components –** the resulted ADF application is pure source code, free of proprietary software, DLL's, etc, and can be further shared or distributed.

- **Developer support**
  - **for the Java developer:** help in thoroughly understanding the Forms application complexity, through detailed reports and analysis
  - **for the Forms developer:** an easier transition to JEE with a step-by-step approach, guidelines and coaching from our consultants

- **Risk reduction –** by choosing a tested approach, opening the application to a wide palette of technologies and future possibilities. Extracting the data-specific business logic and storing it as a database layer makes the application less dependent on the user interface technological changes, more flexible and robust.

Conclusion

We know the decision of changing towards JEE and JDeveloper ADF is an important and challenging one. PITSS guides you through these strategic decisions and provides a solution to minimize the typical risks involved:

- consolidating the business logic in the database - whichever is the approach you choose (staying with Forms or going to any other framework), the no-risk 1st step is consolidating the business logic and moving it to the database

- preparing for the new technology by acquiring the necessary skills

- starting pilot projects - moving to ADF the externally-used applications, the ones that need to be accessed from the browser; and defining the applications that need to stay in Forms - the internally-used ones, especially if they are data intensive. We can use for this selection process the PITSS.CON ADF application assessment report

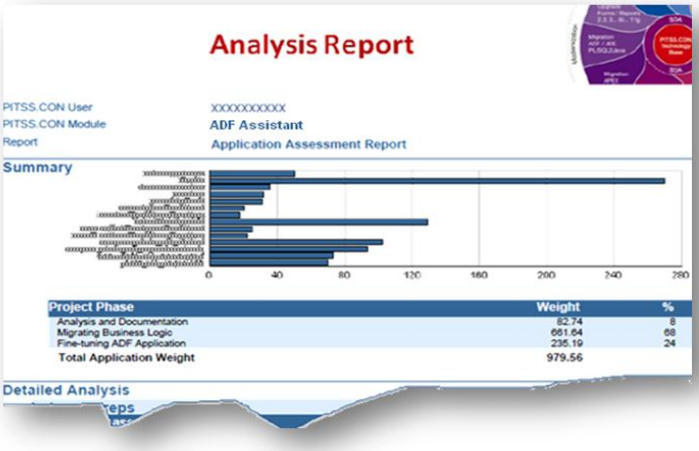

Figure 6: PITSS.CON Application Assessment Report

- estimating the amount of work – the PITSS.CON suite can significantly save time and make the difference; we've seen countless customers coming to us after failed attempts to manually modernize legacy applications, after spending man-years of work and huge budgets with disappointing result

So call or visit us on Oracle development conferences close to you, register for our free webinars. We are looking forward to analyze your existing application and provide you with a free assessment of your situation.

About PITSS

PITSS is the leading supplier of fully integrated solutions for effective management of Oracle Forms applications. The innovative PITSS.CON software helps its customers to analyze, migrate, upgrade and maintain their Oracle Forms applications in its entirety. PITSS thus opens an evolutionary path for the migration of Oracle Forms applications to a Service Oriented Architecture (SOA). PITSS.CON has earned a reputation through its high level of automation and performance. Migration and development projects are run rapidly, economically and reliably within shortest possible time frames. With PITSS.CON, companies achieve an average cost saving of 30% for regular development projects and up to 90% for upgrade projects. PITSS is an Oracle Certified Advantage Partner and has customers in Europe, USA and Asia.

From Oracle Forms to Oracle ADF and JEE

November 2010

Authors: Magdalena Serban, Bahar Us
Contributing Authors: Andreas Gaede, Martin Disterheft
Reviewer: Chris Baker

PITSS in Europe
Germany
+49-711-728.752.00
info@pitss.com
www.pitss.com

PITSS in Americas
USA
248.740.0935
info@pitssamerica.com
www.pitssamerica.com