Online Spreadsheet Editor - A Migration Solution from Oracle Forms to ADF

Pratima Bhat[1], Bushra Alam[1], Dr Shankaraiah[2], Karamjit Kaur[2]

[1,1]Post Graduate Student, EC Department, SJCE, Mysore, India
[1,2]Post Graduate Student, CSE Department, Thapar University, Patiala, India
[2,1]Professor, EC Department, SJCE, Mysore, India
[2,2]Lecturer, CSE Department, Thapar University, Patiala, India

***Abstract*— Oracle Forms have been used widely for building screens that interact with the database. In order to meet the challenging requirements of today's business, there is a rapid need to adopt Java/JEE technologies and shift towards service-oriented architecture (SOA). Oracle ADF is one such emerging web based technology for building enterprise applications. There are existing ideas to build an ADF screen for every single Oracle Forms screen. This paper proposes an idea to build a single ADF screen that can handle multiple screens. This approach would save a huge amount of time for manipulating bulk data and also the maintenance cost of multiple screens. This approach would reduce the time taken by the entire process to 1/10th of the time taken by Oracle Forms.**

***Keywords*— Business logic, CRUD operations, Model-View-Controller (MVC) architecture, Oracle Application Development Framework (ADF), Oracle Forms, Online Spreadsheet Editor.**

I. INTRODUCTION

Oracle Forms is a GUI based software product used to create the screens that interact with the Oracle database. Oracle Forms have been used for years. As more users move towards web based technologies as their IT backbone, there is a need to adopt Java/JEE technologies and shift towards service-oriented architecture (SOA). With Oracle JDeveloper and Oracle Application Development Framework (ADF), it is possible to build feature-rich JavaServer Faces (JSF) web applications. There is an emerging need to convert existing Oracle Forms' screens to ADF. The business is required to adopt leading edge, modern technologies. ADF is one such emerging technology which provides a Java Framework for building enterprise applications. This migration from Forms to ADF is challenging primarily because it would require an in-depth understanding of business logic and complexity involved in each screen. For each Oracle Forms' screen a corresponding ADF screen needs to be built. This, if at all feasible, is going to consume a lot of time. A simpler and more effective way would be to build a single screen and then change it dynamically to display the details of the screen that the user selects. This would eliminate the need to build multiple ADF screens.

PITSS.CON [1] discusses the need for ADF and the challenges involved in migrating from Forms to ADF. PITSS.CON maintains a repository in which Forms and Reports application is loaded. The code is then parsed to construct the whole structure of object inter-dependencies. It discusses the migration of the data intensive business logic to the database and creates a Business Logic Layer for each screen. After the migration to the database, this layer can be accessed from the Oracle Forms and Reports, and also from Oracle ADF, Oracle APEX. YASH Technologies [2] discusses a migration tool for converting Oracle Forms to Oracle ADF.

The approach involves creating ADF folder structures and generating configuration files like adfcconfig.xml, web.xml and connection.xml. It then generates all ADF business components and converts each Oracle Forms screen to an ADF UI pages.

This paper discusses a migration solution from Oracle Forms to Oracle ADF. The Online Spreadsheet Editor will dynamically display the data of the required screen thus removing the need for creating multiple screens. The approach provides a spreadsheet like UI wherein the user can perform CRUD operations on any screen. The application would be beneficial to users who use different screens to perform CRUD operations on database tables as the need for understanding a different screen for a new table would not be required. The application would also be beneficial to the developers as the designing of multiple screens would not be needed and the focus could only be on the business logic.

II. MOTIVATION

Oracle Forms have sustained for long. They are a boon to the industry with their numerous applications. But with the change in trend and requirements of the industry, their limitations have taken over. Oracle ADF, based on Java/JEE technology, is an emerging replacement.

Oracle Forms is based on transactional client/server applications. With the rapid growth of web based applications, the need arises to migrate Oracle Forms based applications to web based technology. ADF applications are light-weight and hence could easily be accessed through various electronic devices such as phone, tablets etc. In Oracle Forms, the UI and the business logic are closely coupled hence making development and maintenance processes difficult. On the other side, ADF is built upon MVC architecture which

separates the UI from the business services thus providing reusability. Oracle Forms build stateful dedicated connections whereas ADF connections are pooled stateless which gives the simplicity of stateful but performance of stateless. Oracle Forms is based on PL/SQL as a scripting language and is primarily focused on the Oracle database as a data source. Oracle ADF has many crossovers but is based around Java and much more extensive in terms of data sources, user interface technologies etc.

Oracle ADF is a Metadata driven framework offering declarative options for development, configured from XML metadata, while accommodating custom coding wherever necessary. This metadata driven approach allows the use of all or part of the framework in the applications you build, making the application components much more reusable and flexible. The use of metadata also enables rules for data bound fields to be specified at the model layer [3].

The migration approaches considered so far are focused on converting single Oracle Forms' screen to a single ADF screen which requires huge investments in terms of time and other resources. The single ADF screen built against a single Forms screen would have the same level of difficulty in customization such as adding a new field to incorporate the change in customer requirement. Such screens would also have a drawback of manipulating bulk data simultaneously.

III. Implementation

Online Spreadsheet Editor is an ADF application which would help users to feed data into the database tables through a spreadsheet like interface. The business logic would be validated instantly and invalid records would error out, displaying user friendly error messages thus ensuring that only the valid data is processed. The screen would change dynamically yet would retain the structure and hence eradicating the need to analyze different screens. The segregation of UI and the business logic helps to plug in various database tables and perform CRUD operations.

Some of the major benefits Online Spreadsheet Editor would provide are –

1. Replaces multiple screens with a single UI.
2. Performing CRUD operations on bulk data simultaneously.
3. Ease of use.
4. No need to traverse to locate a screen.
5. Viewing all the data corresponding to a screen on a single UI.
6. Ease of development.
7. Light weight because of ADF framework.
8. Customization flexibility.
9. Errors are displayed at runtime.
10. Familiar spreadsheet like user interface.

Oracle ADF is a JEE based framework providing a robust platform to build web based applications. It provides robust, maintainable and high performing applications. In addition, it also provides the best of breed infrastructure code to implement agile SOA based applications. Oracle ADF is based on MVC architecture. Its Model layer holds the implementation details in metadata. This helps to reuse business logic without modifying the user interface thus making the applications extremely agile.

There are challenges involved in migrating from Oracle Forms to ADF. Migrating Forms' screens to ADF requires thorough understanding of all the screens with all its complexity. It requires understanding all the business logic involved. The documentation for Forms' screens are not sufficient or are retired since the Forms' screens were built long back and have been modified time and over. Hence this makes understanding difficult. Oracle Forms have been developed using PL/SQL while ADF is a Java based framework. Thus this migration requires a mixed set of skills such as PL/SQL, Java, JavaScript, Forms and ADF. It also requires a good understanding of both Oracle Forms complexity and the capabilities of ADF.

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 [2].

Oracle ADF implements the Model View Controller design pattern. The model layer handles the data and the business logic. The view layer handles the user interface for the application and the controller is the interface between the model and the view layers and also manages the application flow.

To implement Online Spreadsheet Editor metadata tables need to be maintained. The metadata tables are created to hold the information regarding number of sheets, sheet names, number of columns, and column headers. Based on the metadata, entity objects and view objects are created. These objects reside in the model layer of the application. In order to provide a spreadsheet look-a-like interface, ADF Table is used. This ADF Table is dynamically populated with the data fetched from the base view object in accordance to the screen name provided by the user.

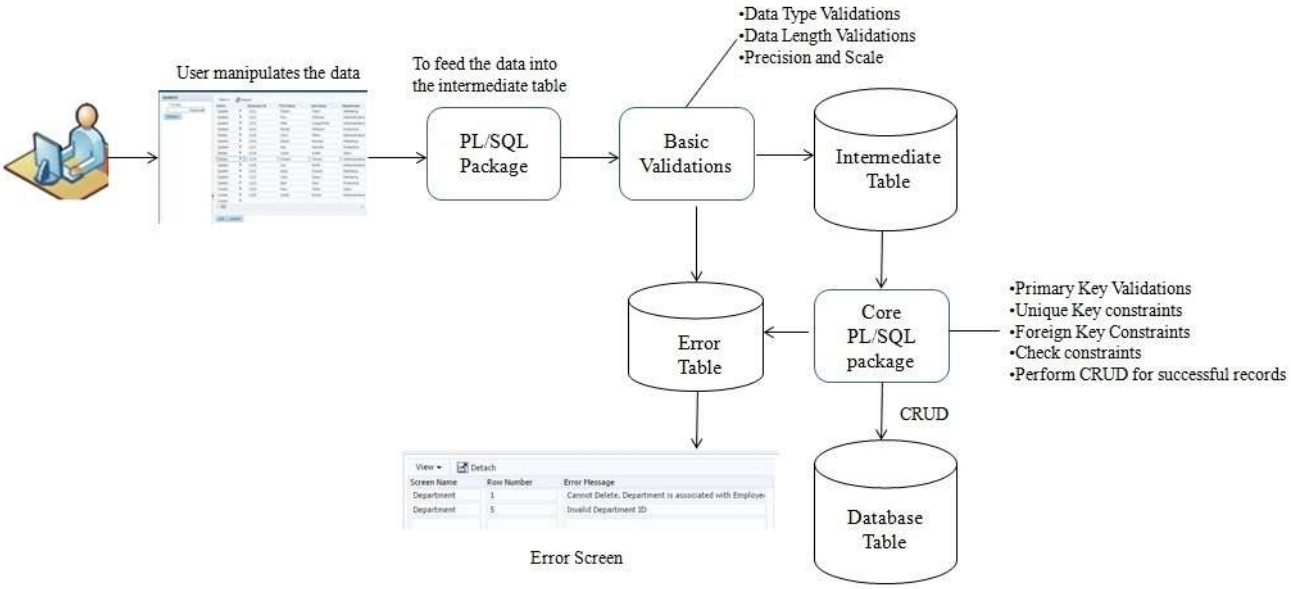

Fig 1: Overall process

The Online Spreadsheet Editor GUI consists of two parts. One section holds a search box enabling the user to provide the screen name to be manipulated. The other section is dynamically reloaded to display the data of the specified screen. Along with this data, an additional column is attached. This column provides an LOV against each record which allows the user to select the required action to be performed on each individual record. This would help in bulk manipulation of data. The number of columns to be displayed and the column headers are dynamically fetched.

For updating or deleting a record, the user directly selects the action from the dropdown against the required record. For creating a new record, a fresh row is added at the end of the existing data. Once the user submits the manipulated data, this data needs to be validated against the business logic. The application is designed assuming that the user can enter invalid records as well.

Oracle Forms are not based on MVC architecture and hence user interface actions, data validations and data manipulations are contained within the same unit of code. When migrating to ADF, the business logic needs to be segregated into its components in accordance with the MVC architecture. There are various approaches to implement business logic. The approach followed in this paper is to leave the entire business logic in PL/SQL. These PL/SQL packages are then called from the ADF model layer.

The Figure 1 shows the overall process involved in this approach. Once the data is submitted by the user, it is fed into an intermediate table. This intermediate table has the structure of main table except the constraints. The constraints are not implemented so as to allow the upload of incorrect data. Each record is validated against the database validations such as data type, length, precision and scale of the data. Only the records that satisfy these validations are fed into the intermediate table, others are errored out. The business logic validations are then applied to the records in the intermediate table. Along with the business logic validations, primary key validations, unique key constraints and foreign key constraints are also validated. If a record fails in any of the above listed validations, it is not processed and displayed to the user with an error message. Only the valid records are processed and specified CRUD operations are performed on the database table. As the application would be accessible to different users having different privileges, providing data level security is essential. All the security features of Oracle Forms' screen can be implemented in this application as well. This approach would save a huge amount of time for manipulating bulk data and also the maintenance cost of multiple screens. This would also reduce the time taken by the entire process to $1/10^{th}$ of the time taken by the Oracle Forms.

For PoC, a screen corresponding to a table with 1000 records was considered.
SQL queries took 4 seconds (in real-world-scenario) to fetch data from database to the ADF screen.
Now, assuming that 10 users concurrently submitted similar jobs and assuming that client has configured the maximum parallel processes to be 2, then

Wait-time-to-Fetch
(Concurrent Users/Max parallel configuration)*(Data Fetch Time)
(10/2)*4 = 20 seconds.
Assuming that processing 1000 rows in to the database (inserting/updating into the table or deleting from the table) takes another 6 seconds.
Wait-time-for-Processing

(Concurrent Users/Max parallel configuration)*(Process data in to database Time)
(10/2)*6 = 30 seconds

A PoC test with 1000 records took roughly 1/10th of the time taken by the Oracle Forms.

IV. Results and Discussion

As more users move towards the web and service oriented architecture as their IT backbone, standalone Oracle Forms applications can limit down the efficiency. To adapt to this technology shift, Oracle ADF is an emerging solution. The whole idea was to develop an approach which would provide a better migration solution and a flexible user interface by handling multiple Oracle Forms' screens along with its business logic and all the complexity.

The Figure 2 shows the Online Spreadsheet Editor screen having two parts. The first part contains a search box wherein the user gives the name of the screen on which he wants to perform CRUD operations. The second part of the screen has a spreadsheet like view of the required screen data. The user can select the required action using the dropdown provided. The user then submits the data. The Figure 3 shows a dynamically changed screen in accordance with the screen selected by the user. Once the data is submitted by the user, the records are validated. Only the records that satisfy the business logic and the database constraints are processed. The invalid records are displayed on the error screen along with the user

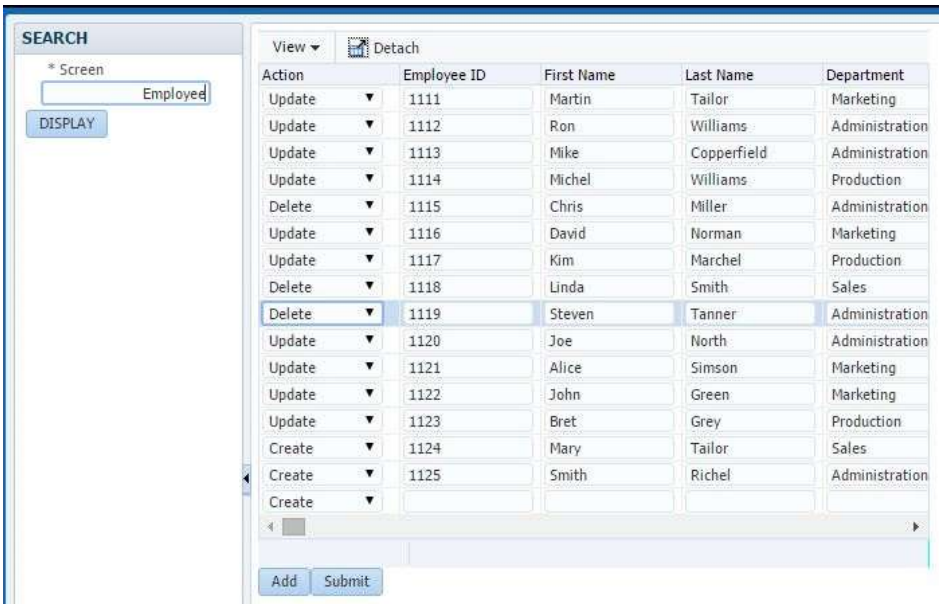

Fig. 2: ADF Screen displaying the data fetched.

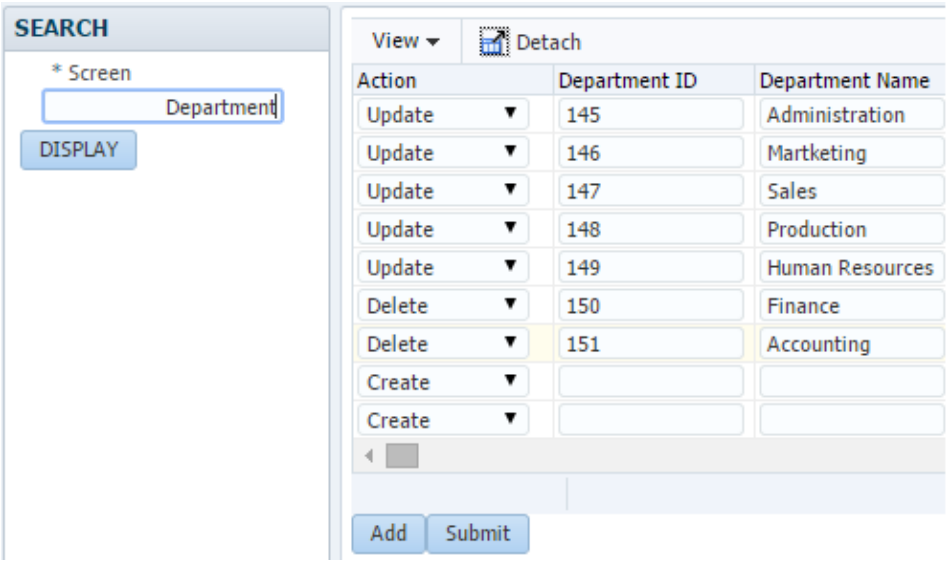

Fig. 3: Dynamically changed screen.

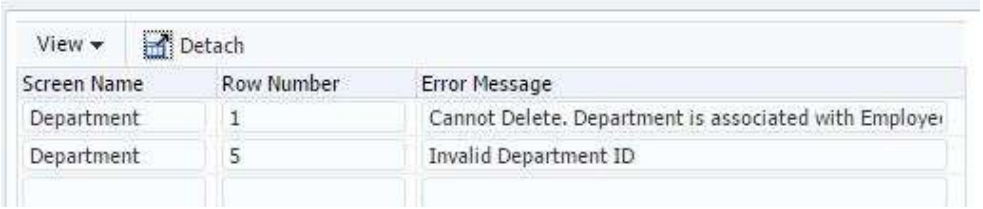

Fig. 4: Error Screen

understandable error messages as shown in the Figure 4.

V. Conclusion

With the growing need to migrate towards web based applications, the legacy applications based on Oracle Forms need to be upgraded. Oracle ADF is a JEE based framework providing a robust platform to build web based applications. It provides robust, maintainable and high performing applications. Online Spreadsheet Editor is an approach to migrate multiple Oracle Forms' screens to ADF. The existing approaches are based on the idea of converting each Forms screen to an ADF screen. This paper provides a more effective approach by building a single ADF screen that can handle multiple Forms' screens. Online Spreadsheet Editor provides a better interface and flexibility when compared to Oracle Forms for performing CRUD operations.

For end users to get comfortable with Oracle Forms, trainings are needed. The trainings are essential mainly because Forms screens vary greatly and could be nested and hence analysing different screens and navigations are difficult. Online Spreadsheet Editor is designed to have a simple, consistent look that resembles that of a spreadsheet and hence no user training would be required. Oracle Forms' screens do not allow bulk data upload or manipulation. Only a single record could be worked upon at any instance of time. Online Spreadsheet Editor would allow users to work on bulk data simultaneously. This approach would reduce the time taken by the entire process to $1/10^{th}$ of the time taken by Oracle Forms. This implementation can also be extended for interlinked screens.

References

[1] PITSS.CON 8.0.3, "From Oracle Forms to Oracle ADF and JEE, Modernizing Oracle Forms applications to Oracle Application Development Framework and the JEE Architecture", White Paper, November 2010.

[2] YASH Technologies , "Oracle Forms to Oracle ADF, A migration tool", White paper, 2014.

[3] "Oracle Application Development Framework Overview", an Oracle White Paper, June 2011.

[4] "From Oracle Forms to Oracle ADF and JEE", White Paper, November 2010.

[5] Simon Haslam, "Practical ADF application deployment for fusion middleware administrators", Veriton Limited, ODTUG Kaleidoscope, 2008.

[6] "A Case Study in an Oracle Forms Redevelopment Project to Oracle ADF", An Oracle White Paper July 2011 Version 1.0 Published 8th July 2011.

[7] "Case Study: Redeveloping an Oracle Forms application using Oracle JDeveloper 11g and Oracle ADF 11g", An Oracle White Paper, October 2008.

[8] Grant Ronalt, "Migrating Oracle Forms to Fusion: Myth or Magic Bullet?", ODTUG Technical Journal, Second Quarter 2009.

[9] Abhijeet Sartape, Prof. Vasgi B. "Data-Base Security Using Different Techniques: A Survey", International Journal of Computer Trends and Technology (IJCTT) - volume4Issue4 –April 2013.