Extending Oracle EBS using APEX: Quick, Secure and Easy

Yves Chassein & Johannes Michler
PROMATIS Group, Ettlingen (Karlsruhe TechnologyRegion)

Oracle APEX (Oracle Application Express) makes it possible to rapidly develop and publish complex, database-driven desktop and mobile applications by use of the web browser only. Thanks to its simple, declarative approach, Oracle APEX is especially suited for the rapid, professional development of database-centered web applications. With this tool, the functionality of the Oracle E-Business Suite can be easily extended. An attractive and intuitive UI/UX as well as a short and steep learning curve make Oracle APEX the ideal development tool for database administrators, PL/SQL developers and even functional users with some technical background. There are several possibilities to create an integration between Oracle E-Business Suite and APEX. This paper will highlight these possibilities and will additionally show typical use cases for extended E-Business Suite using Oracle APEX.

What is APEX?

Oracle Application Express (APEX) is a development platform which uses SQL, PL/SQL, JavaScript and HTML to create applications on an Oracle database. The development in APEX only requires a browser. As a low-code platform, it enables the programming of simple web applications – quickly, with little effort and without any required prior experience. Of course, highly complex web applications such as customer portals can be realized as well. APEX applications are easily scalable. Since APEX - contrary to Oracle Application Framework (OAF) - uses SQL and PL/SQL as its main programming language, traditional Oracle Forms, Reports and (PL)-SQL developers will find it easier to learn.

The main characteristics / use cases of APEX are:

- Creation of reports on database tables which can be edited by the user (interactive report) or exported as CSV- or XLS-files (see figure 1, 2 and 3).
- Creation of forms for entering, editing or deleting data in database tables.

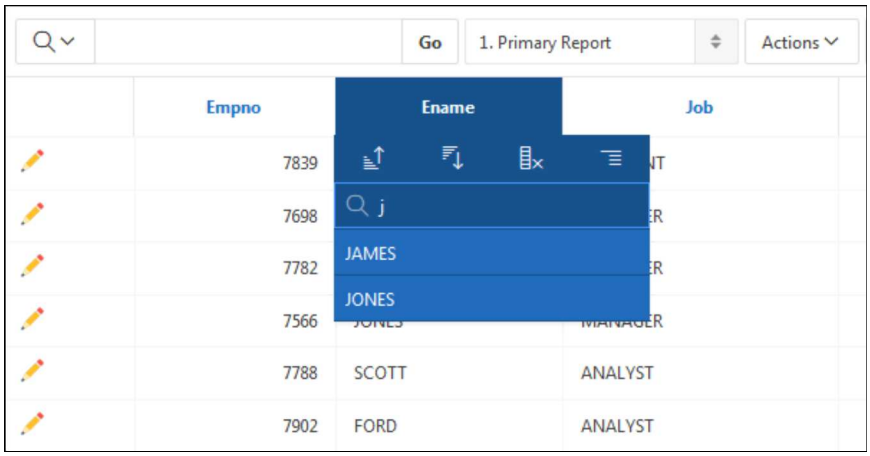

Figure 1: Sorting and searching in interactive reports (Source: www.oracle.com)

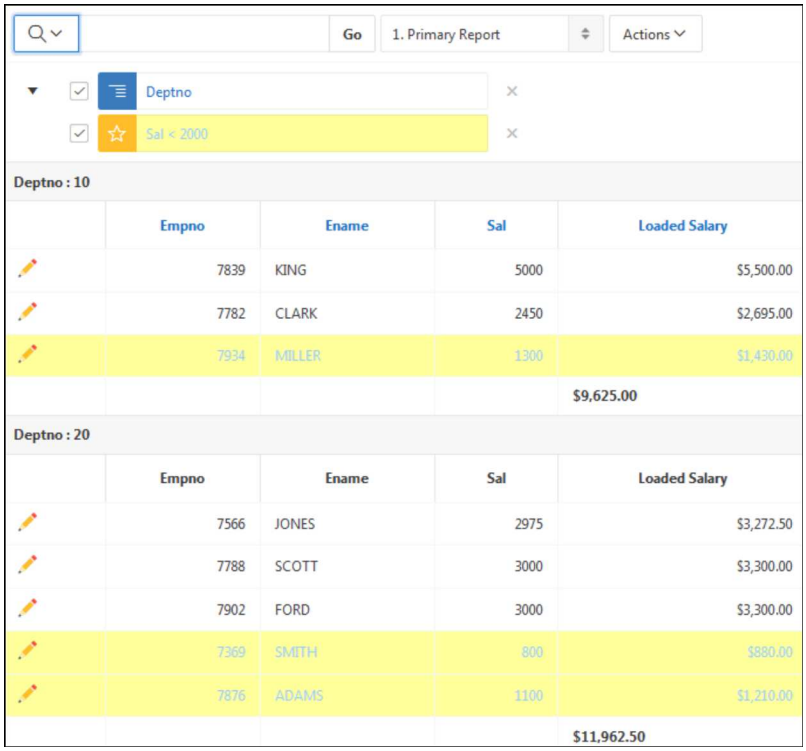

Figure 2: Highlighting rows (Source: www.oracle.com)

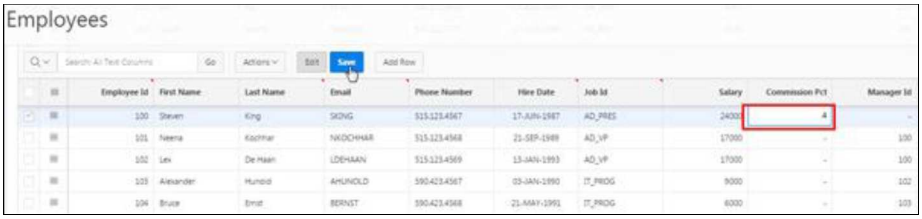

Figure 3: Interactive grid for data editing (Source: www.oracle.com)

The history of APEX

The first version of APEX was published in 2004, under its initial name HTML DB. With Version 2.1 in 2006, it was renamed and called APEX for the first time.

One year later with Version 3.1, interactive reports were introduced. With their help, end users can perform complex filter, sort and aggregation operations directly on the data and hence can further analyze the provided data.

In 2010, Version 4.0 introduced dynamic actions and plug-ins and thus enhanced the development tools enormously. With the release of APEX 5.0 in the year 2015, the entire IDE was reworked. The development was now conducted on only one page – by use of the Page Designer – and no longer on several ones (each component used to have its own page). The introduction of Universal Theme now enabled developers to make adjustments to applications and create visually appealing user interfaces.

With Version 5.1, the “interactive grid” which combines the functionalities of forms and interactive reports was introduced. Since May 2018, the name APEX 5.1(.4) has been changed into 18.1 and thus refers to the new global version numbering scheme at Oracle.

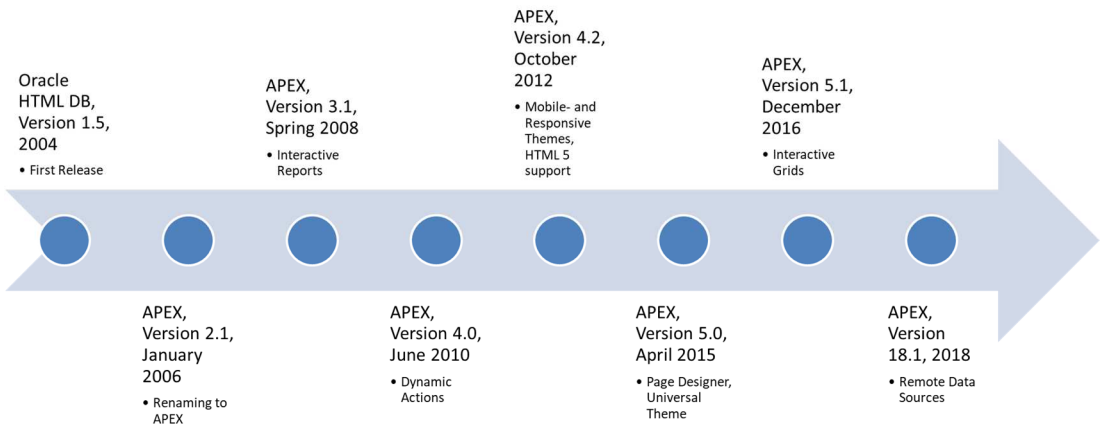

Figure 4: Temporal development of Oracle Application Express

APEX architecture

APEX uses a three-tier architecture. From the browser queries are sent to the database over a webserver. The entire business logic is executed on the database. This way, an Oracle Rest Data Service (ORDS) can be used as the webserver which accepts the request from the browser and forwards it to the database where the request is then actually processed. On completion of the processing, the result is returned to the browser via ORDS. Instead of the ORDS, it is also possible to use the webserver integrated into database - however Oracle does not recommend doing so on a production instance. Another but outdated possibility is the use of the Oracle HTTP server along with mod_plsql - but in later 12c releases of Oracle HTTP Server this is deprecated or even removed. The use of ORDS is therefore the best-practice and proven way of providing the middletier nowadays - for small development environments and scaling up to huge installations with multiple ORDS instances accessed through an up-front load balancer. It is possible to use ORDS in standalone mode (especially for test instances) or on an application server like Apache Tomcat or Oracle Weblogic.

When used for extending the Oracle E-Business Suite, APEX can be installed into the same database. Furthermore ORDS can be installed on the same physical or virtual server as the E-Business Suite application or database server. As an alternative, APEX can be installed on a separate database and/or application server. For access to the EBS data, in this case a database connection between the two instances is required. In most cases, it makes sense to install an additional Tomcat or Weblogic server on the physical EBS application server to host ORDS.

Figure 5: Architecture of Oracle ORDS and APEX (Source: apex.oracle.com)

Integration between Oracle E-Business Suite and Oracle APEX

In general, APEX has several alternatives to implement user authentication and authorization. In simple scenarios the login is performed with the APEX login credentials or via a database user. It is possible to use one of these login mechanisms to allow the user access to the application also in an EBS extension use case. However this is not really comfortable and good maintainable. Here, the same login information of the EBS login should be used. In APEX, this can be accomplished by using a custom authentication function. For an improved usability, however, a one-time login instead signing in twice (even with the same credentials) makes more sense. In addition to that, the following requirements should be considered:

- Login/SSO
  - When navigating from EBS screens to APEX, no (additional) password should be needed for the login.
  - EBS session ID, responsibility, user, and organization information should be transferred to APEX in a safe, unmodifiable way.
  - The login mechanism should function without using the session cookie of the EBS (in case the host of the ORDS server is not identical with the host of the EBS application server).
  - The login URL should only be valid for a limited time (no copy-paste of the URL e.g. by e-Mail so that other users can sign in).
- Session management
  - Every call of an APEX page should tickle and “keep alive” the EBS session.
  - Once the session (due to inactivity or total-session-length) has expired, the user should be redirected to the EBS login form.
  - The logout button in APEX and in the EBS should end the according session in the other application as well.
  - Users should optionally be able to log in into APEX without needing to sign into the EBS first.

- Menu in APEX
    - The menu the user can see in APEX should be controlled by the security functions within the EBS, which means every page in APEX needs a connection to an EBS function.
- Authorization
    - Every page should be connected with a function in the EBS.
    - After calling up the page, the authorization function checks if the user is allowed to access the according EBS function.
- Context initialization
    - This should be performed by every APEX page in order to allow access to tables such as e.g. OE_ORDER_HEADERS (without _ALL).

There are multiple published approaches that cover several, but not all of these requirements. An easy procedure is the transfer of the EBS username to APEX as a parameter. The authentication function only checks if there is an active session for the user. If this is the case, the "user" receives access to the application. If no active session can be found, the access is denied. This solution is very unsecure obviously since without any access to the EBS, usernames can be guessed and pasted into the URL. Even if instead of the username, the user ID in the URL is transmitted, it can be changed by guessing or by trial and error.

Thus, a secure way without the transfer of variables in the plain text of the URL is needed. To fulfill this requirement, an encrypted token (using DBMS_CRYPTO) for the URL can be created in the EBS which is then sent to APEX and is then subsequently decoded in order to sign on the user. The token consists of the username and the responsibility, EBS-Session ID and organizational information. Passing the Session ID makes it possible to additionally end the session in the EBS after logging out of APEX. An EBS session remains unaffected if a parallel login takes place directly using APEX login dialog. In order to prevent sessions initiated directly through APEX lasting “forever”, we suggest building a session timeout into APEX that corresponds to the session timeout implemented by Oracle in the EBS. As soon as the APEX session has expired, the EBS session is ended as well. The same functionality is also implemented for the opposite direction. This way, a timeout is triggered as well if the session has been started directly in APEX.

After opening the application, some kind of authorization is needed, or else all users have full access to all pages even though the according responsibility is not assigned to the user in the EBS. In APEX, it is possible to create authorization schemes in PL/SQL so that on every page loading, EBS functions can be queried. With this functionality, user access for certain pages in APEX can be restricted. As described in the requirements, every APEX function should be connected with an EBS function. This authorization can then be maintained directly in the EBS.

To access EBS APIs and tables, often the correct session context is needed. In an EBS session (OAF or Forms), this is usually taken care of automatically. In APEX, this needs to be set manually on every page in order to ensure a correct functionality. To do this, the information from the token can be used. By using user, responsibility and organization ID, it is possible to initialize the Oracle EBS and the organizational context.

With the PROMATIS approach, all requirements are fulfilled and the Oracle EBS can be integrated securely into APEX.

The benefits at a glance

By using APEX to extend Oracle EBS, a framework is provided which can be easily learned by Oracle Forms, Reports an (PL)-SQL developers and that allows creating powerful UIs quickly.

The APEX application is accessible via a browser so that for the use of APEX UIs, no more additional tools - especially no Java or Java Web Start - are needed on the client side. This benefit is of high significance since more and more browsers exclude Java (Plugin) from their core functionalities.

Since APEX contains a number of performant reporting functions such as JET diagrams, interactive reports and grids, we can easily review and print data in several formats such as PDF or CSV. The reports are even adjustable, and with the help of additional tools such as the BI Publisher, printing DOC or XLS files poses no more problems.

Furthermore, all applications are suited for mobile devices due to the responsive APEX interface, thus increasing availability and accessibility.

Due to the vast and helpful community and even though there is no official Oracle support, problems arising during development are usually solved rapidly.

Possible application

With the help of APEX, Oracle EBS can be extended with reports and forms. Such forms can for example be used to maintain custom data extending information held in EBS. APEX can even be used to provide the complete frontend for at least some user groups interacting with the E-Business Suite. Those users then never need to sign in into the “real” EBS backend applications. This is especially handy if users are unable to install Java to access standard Forms user interfaces or if those UIs are too complex for the end users.

The ORDS serving as the backend webserver for APEX can be used as well for providing REST Web Services to third party applications integrating with E-Business Suite. Another - last but not least - use case seen in many project consists of using APEX to provide access (reading and validated modifying) to seeded or custom staging tables: This provides a convenient way to fix all kinds of human or technical errors.