Oracle Applications Express

The Fast Way to Extend the Oracle E-Business Suite

By Rod West, Cabot Consulting

Oracle Applications Express (APEX) is becoming established as an excellent tool for developing web-based applications, with many essential build-in features such as session state management, page templates and navigational controls. With APEX you can use the form and report wizards to quickly create a fully functional web application and the spreadsheet wizard lets you upload spreadsheet data to the database. Finally, you can extend the functionality of your APEX application in almost any way by incorporating PL/SQL and JavaScript.

This article describes how an APEX application can be integrated with the E-Business Suite 11i and R12. There are many tools that Applications developers can use to extend E-Business Suite but often developers are unaware that APEX can be closely integrated with the E-Business Suite. APEX forms and reports can be added to E-Business Suite menus and spreadsheet data can be uploaded into the E-Business Suite application.

Developing APEX Applications for the E-Business Suite

APEX is ideal for developing small applications that can be used alongside or integrated into Oracle E-Business Suite. APEX provides a more robust and scalable solution than using spreadsheets and faster development than with forms and reports. Some examples of the APEX applications that can be developed for the E-Business Suite are:

- Defining alerts and exceptions.
- Auditing and management reports and charts.
- Uploading personnel data.
- Creating and maintaining flexfield hierarchies.
- Managing workflow responses.

Not all APEX applications will need to be tightly integrated applications where the user can move seamlessly from Oracle Applications to APEX, but once integrated some or all of the following tasks are possible:

- Use the APEX wizards to develop forms and reports based on E-Business Suite tables.
- Use the E-Business Suite security to allow users to logon to APEX using their E-Business Suite username and password.
- Use E-Business Suite menus to link directly to APEX reports and forms.
- Use APEX to call E-Business Suite APIs directly.
- Add links in APEX forms and reports to E-Business Suite forms.
- Use E-Business Suite responsibilities with APEX so that different APEX functionality is available and different data is returned to the user depending on the responsibility.

Sharing the E-Business Suite Infrastructure

APEX resides completely in the Oracle database and the easiest option is for APEX to share the database with the E-Business Suite. If APEX is not already installed in the database then APEX can be downloaded from apex.oracle.com and installed in the database. APEX is free, fully supported so no additional license is required and the Oracle database installed with the E-Business Suite has all the prerequisites for APEX.

APEX is accessed from a web browser through an Oracle HTTP server (Apache) and mod_plsql. If APEX is being used with E-Business Suite 11i then the E-Business Suite Application Server can be used to run APEX applications. However, some minor changes are required to the Apache configuration. These are creating a separate DAD to access the APEX applications and creating an alias to map to the APEX image directory.

If APEX is being used with E-Business Suite R12 then a separate Oracle HTTP server must be used because mod_plsql is not supported in the R12 Application Server. The Oracle HTTP Server distributed with Oracle Database 11g or Oracle Application Server 10g can be downloaded and installed in a separate Oracle Home as described in the APEX installation guide.

Getting Started

All APEX applications are held in a workspace, so before you start developing your APEX application you will need to log in as the APEX administrator and create a workspace for your applications.

The workspace should be associated with the E-Business Suite APPS database schema. This does not change anything in the APPS schema as the APEX application is held within the APEX tables but gives the APEX application developer full access to the E-Business Suite tables, views and APIs.

Creating your application

The forms and reports wizards are one of the most useful features of APEX and allow the developer to rapidly create APEX pages from Oracle database tables. However, the APEX forms and reports should never be created directly onto the E-Business Suite tables. Your E-Business Suite installation will not be supported if you update the tables directly.

Figure 1
```
CREATE OR REPLACE VIEW apex_fnd_flex_values
    (flex_value_set_id
   , flex_value_id
   , flex_value
   , flex_value_set_name
   , description
   , CONSTRAINT apex_fnd_flex_values_pk
     PRIMARY KEY (flex_value_id)
     RELY DISABLE NOVALIDATE
    ) AS
   SELECT  ffvs.flex_value_set_id
         , ffv.flex_value_id
         , ffvs.flex_value_set_name
         , ffv.flex_value
         , ffv.description
      FROM fnd_flex_value_sets ffvs
         , fnd_flex_values_vl ffv
     WHERE ffv.flex_value_set_id =
           ffvs.flex_value_set_id
   WITH READ ONLY
```

Your first step should be to define views that APEX will use to access the E-Business Suite tables. (See Figure 1) These views can include joins and security conditions to hide the complexity of the E-Business Suite table structures from the APEX application. The query restriction WITH READ ONLY can be used in the view definition to stop APEX inadvertently changing the data in the underlying tables.

Some of the APEX wizards expect primary and foreign keys to be defined, so that, for example, a form can be opened for a record that the user has selected in a report. Therefore it is useful to define primary and foreign key constraints in the APEX views so that the wizards understand the metadata in the underlying tables.

With these views the forms and reports created by the wizards will work correctly when selecting data from the E-Business Suite tables. However, for inserts, updates and deletes you must use the E-Business Suite APIs rather than try to change the tables directly with APEX.

The APEX row processing can be modified to call the required E-Business Suite APIs update the database. Where the data load is complex APEX can load data directly into a staging tables and then submit an E-Business Suite concurrent request to transfer the data from the staging tables into the E-Business Suite.

For simple data loads an easier option is to create INSTEAD OF triggers for your APEX views that call these E-Business Suite APIs. With this approach the APEX automatic row processing does not require any modification. When APEX inserts, updates and deletes from the view the INSTEAD OF trigger is fired calling the API to change the data in the tables. (See Figure 2.)

Figure 2
```
CREATE OR REPLACE TRIGGER apex_fnd_flex_values_tr
INSTEAD OF INSERT OR UPDATE ON apex_fnd_flex_values
DECLARE
v_storage_value VARCHAR2(32000);
BEGIN
IF INSERTING THEN
    fnd_flex_val_api.create_independent_vset_value
 (p_flex_value_set_name=>:NEW.flex_value_set_name,
  p_flex_value=>:NEW.flex_value,
  p_description=>:NEW.description,
  x_storage_value => v_storage_value);
ELSIF UPDATING THEN
    fnd_flex_val_api.update_independent_vset_value
  (p_flex_value_set_name=>:NEW.flex_value_set_name,
   p_flex_value=>:NEW.flex_value,
   p_description=>:NEW.description,
     x_storage_value => v_storage_value);
END IF;
END;
```

Many E-Business Suite APIs require the session to be initialised for the E-Business Suite user and responsibility. For this you will need to make APEX aware of the Application 11i users and responsibilities. This will be covered later in the document.

Using the E-Business Suite Username and Password to logon

APEX uses an authentication scheme to logon users. A Single Sign-On (SSO) authentication scheme is supplied with APEX so if your E-Business Suite system is configured to use SSO you can use SSO with APEX as well.

For E-Business Suite systems that are not using SSO, you will have to create a custom authentication scheme to validate the APEX user. A custom authentication scheme calls an external function to validate the username and password. You can simply use your own function that calls the E-Business Suite login validation function as shown in Figure 3.

Figure 3
```
FUNCTION ebs_authenticate (
    p_username    IN    VARCHAR2
  , p_password    IN    VARCHAR2) RETURN BOOLEAN
AS
BEGIN
RETURN (FND_WEB_SEC.validate_login
         (p_username, p_password) = 'Y');
END ebs_authenticate;
END IF;
END;
```

When defining your custom authentication scheme only the following needs to be entered:

- The invalid session target page should be set to the login page (page 101) not the built-in login page.
- The authentication function should be set to return your ebs_authenticate function.
- The logout url should call the APEX logout procedure and redirect to the logon page. (See Figure 4.)

Figure 4
```
wwv_flow_custom_auth_std.logout?
p_this_flow=&APP_ID.&p_next_flow_page_sess=&APP_ID.:101:&SESSION.:LOGOUT
```

You will now be able to log in to your APEX application using any valid E-Business Suite username and password.

Linking directly to APEX Forms and Reports from E-Business Suite Menus

To connect directly to APEX from E-Business Suite a secure mechanism is required to hand over control from E-Business Suite to the APEX Application. There are many different mechanisms that can be used, but the first problem to be overcome is that the APEX application requires a valid password for the user and the database only contains the encrypted E-Business Suite passwords.

This can be overcome by creating an alternative password for the user using the DBMS_OBFUSCATION_TOOLKIT md5 function. (See Figure 5 below) The password is generated by hashing the username, a time component and a key held within the database. This key can be set in the package initialisation by, for example, selecting the encrypted password for the SYSADMIN user from the FND_USER table.

Figure 5

```
CREATE OR REPLACE PACKAGE APPS.XXAPX_SECURITY_PKG AUTHID DEFINER AS
FUNCTION generate_hash (
   p_string     IN   VARCHAR2
 , p_offset     IN   NUMBER DEFAULT 0) RETURN VARCHAR2;

FUNCTION validate_hash (
   p_string     IN   VARCHAR2
 , p_hash       IN   VARCHAR2
 , p_delay      IN   NUMBER DEFAULT 5) RETURN BOOLEAN;

END XXAPX_SECURITY_PKG;
/
CREATE OR REPLACE PACKAGE BODY APPS.XXAPX_SECURITY_PKG AS

g_key       VARCHAR2(100);

FUNCTION generate_hash (
   p_string     IN   VARCHAR2
 , p_offset     IN   NUMBER DEFAULT 0) RETURN VARCHAR2
IS
BEGIN
IF p_string IS NULL THEN RETURN NULL; END IF;
RETURN RAWTOHEX(UTL_RAW.cast_to_raw(
     DBMS_OBFUSCATION_TOOLKIT.MD5(
        input_string=>p_string||':'||
          TO_CHAR(SYSDATE-(p_offset/24*60*60), 'YYYYMMDD HH24MISS')||g_key)));
END generate_hash;

FUNCTION validate_hash (
   p_string     IN   VARCHAR2
 , p_hash       IN   VARCHAR2
 , p_delay      IN   NUMBER DEFAULT 5) RETURN BOOLEAN
IS
BEGIN
FOR i IN 0..p_delay LOOP
    IF p_hash = generate_hash (p_string, i) THEN RETURN TRUE; END IF;
END LOOP;
RETURN FALSE;
END validate_hash;

BEGIN
SELECT encrypted_user_password INTO g_key
FROM FND_USER WHERE user_name = 'SYSADMIN';
END XXAPX_SECURITY_PKG;
/
```

Checking for this password can be added to the ebs_authenticate function. (See Figure 6.)

Figure 6

```
FUNCTION ebs_authenticate (
   p_username   IN   VARCHAR2
 , p_password   IN   VARCHAR2) RETURN BOOLEAN
AS
BEGIN
IF validate_hash (p_username, p_password) THEN RETURN TRUE; END IF;
RETURN (FND_WEB_SEC.validate_login (p_username, p_password) = 'Y');
END ebs_authenticate;
```

In principle, a user could log onto the APEX Application using this hash password but as the password is continually changing and can only be generated from within the database then this is considered unlikely.

The next problem to overcome is the automatic logon to the APEX application. Some approaches that could be used are:

1. Create an APEX public logon page where you can supply the username and password as parameters to the page;
2. Pass the username and password from E-Business Suite to APEX in a custom cookie.
3. Use the E-Business Suite ICX cookie to obtain details of the user.

If you are connecting to APEX using an E-Business Suite form function then using the ICX cookie is the easier more secure option. The form function then just needs to redirect to the APEX URL to launch the APEX application.

With the E-Business Suite 11i this is achieved by creating a web enabled mod_plsql procedure as shown below. (See Figure 7.) The procedure assumes that /pls/apex is the APEX DAD. The request field in the APEX URL contains APPS to tell APEX that the user is connecting directly from the E-Business Suite.

Figure 7
```
PROCEDURE apex_launch (
   application IN VARCHAR2
 , page        IN VARCHAR2 DEFAULT '1'
 , item_names  IN VARCHAR2 DEFAULT NULL
 , item_values IN VARCHAR2 DEFAULT NULL)
AS
BEGIN
OWA_UTIL.mime_header('text/html', false);
OWA_UTIL.redirect_url(
    FND_PROFILE.value('APPS_FRAMEWORK_AGENT')||'/pls/apex/f?p='||
    application||':'||page||'::APPS:::'||
    item_names||':'||item_values);
END apex_launch;
```

The E-Business Suite plsql form function is then created with the following attributes:

- Properties/Type – Must be a SSWA plsql function.
- Web HTML/HTML Call – Enter your apex_launch procedure.
- Form/Parameters – Enter the parameters to apex_launch procedure.

With the E-Business Suite R12 this is achieved by creating a custom jsp in the $OA_HTML directory on the server. An example is shown in the LaunchApex.jsp shown in Figure 9. The jsp assumes that /pls/apex is the APEX DAD and that there is a custom system profile (APEX_HTTP_SERVER) containing details of the Oracle HTTP Server used by APEX.

The jsp can be compiled using the command shown in Figure 8 (See metalink notes 215268.1 and 458338.1).

Figure 8
```
export PATH=$PATH:$FND_TOP/patch/115/bin
ojspCompile.pl --compile -s 'LaunchApex.jsp ' --retry
```

The E-Business Suite jsp form function is then created with the following attributes:

- Properties/Type – Must be a SSWA jsp function.
- Web HTML/HTML Call – Enter your LaunchApex.jsp.
- Form/Parameters – Enter the parameters for the APEX URL.

Figure 9

```
<%@ page contentType="text/html;charset=UTF-8" %><%--
/* LaunchApex.jsp */
--%><%@ page
import="java.io.*"
import="java.net.*"
import="java.sql.*"
import="javax.servlet.http.*"
import="java.util.Enumeration"
import="java.util.Properties"
import="java.lang.Math"

import="oracle.apps.fnd.common.VersionInfo"
import="oracle.apps.fnd.functionSecurity.Function"
import="oracle.apps.fnd.functionSecurity.RunFunction"
import="oracle.apps.fnd.common.WebAppsContext"
import="oracle.apps.fnd.common.AppsEnvironmentStore"
import="oracle.apps.fnd.common.WebRequestUtil"
import="oracle.apps.fnd.common.ResourceStore"
import="oracle.apps.fnd.security.CSS"
import="oracle.apps.fnd.common.Message"

import="oracle.jdbc.OracleConnection"
import="oracle.jdbc.OraclePreparedStatement"
import="oracle.jdbc.OracleResultSet" %><%!

// Session has to be validated first
  WebAppsContext ctx = WebRequestUtil.validateContext(request, response);

  if (ctx==null) {
      return;  }
  String cookieName = ctx.getSessionCookieName();

  boolean validSession = ctx.validateSession(cookieName);
  WebRequestUtil.setClientEncoding(response, ctx);

%>
<html>
<head>
<title>Launch Apex</title>
<!-- LaunchApex.jsp  -->
</head>
<body>
<%
    String p_application = request.getParameter("application");
    p_application = ( p_application==null ? "NONE" : p_application);

    String p_page = request.getParameter("page");
    p_page = ( p_page==null ? "1" : p_page);

    String p_item_names = request.getParameter("item_names");
    p_item_names = ( p_item_names==null ? "" : p_item_names);

    String p_item_values = request.getParameter("item_values");
    p_item_values = ( p_item_values==null ? "" : p_item_values);

    AppsEnvironmentStore m_env = (AppsEnvironmentStore) ctx.getEnvStore();

    try {
      String l_launcher = ctx.getProfileStore().getProfile("APEX_HTTP_SERVER");
      l_launcher = l_launcher + "/pls/apex/f?p=" + p_application + ":" + p_page;
      l_launcher = l_launcher + "::APPS:::" + p_item_names + ":" + p_item_values;

      if (ctx!=null) ctx.freeWebAppsContext();

      response.sendRedirect(l_launcher);
      }  //try
      catch (Exception e) {
        out.println("Exception found : <pre>");
        e.printStackTrace(new PrintWriter(out));
        out.println("</pre>");
        } //catch
%>
</body>
</html>
```

The form function can then be added to the E-Business Suite menu to link the item in the menu to an APEX application and page, and if necessary pre-load the page with some item values. In the APEX application a 'before header' process for the logon page (page 101) can be used to read the ICX cookie and log the user in. This process should be conditional; only be run if the login page is processing the APPS request entered into the URL by the form function.

The PL/SQL used to process the login is shown in Figure 10. The process calls the get_session procedure to obtain the username and then generates the password as described earlier in this document. The process can then log on the user with this username and password.

Figure 10
```
BEGIN
wfa_sec.getsession(:P101_USERNAME);
:P101_PASSWORD :=
  XXAPX_SECURITY_PKG.generate_hash
            (FND_GLOBAL.user_name);

IF :P101_PASSWORD IS NOT NULL THEN
    APEX_CUSTOM_AUTH.login(
        P_UNAME        => :P101_USERNAME,
        P_PASSWORD     => :P101_PASSWORD,
        P_SESSION_ID   => v('APP_SESSION'),
        P_APP_PAGE     => :APP_ID||':1'
        );
END IF;
EXCEPTION WHEN OTHERS THEN NULL;
END;
```

Making APEX aware of Users' Responsibilities

Every user connects to an E-Business Suite application using a responsibility that defines which processes the user can access from within the application. You can build responsibilities into your APEX application so that different features and options are available to the user depending on the responsibility that was used to connect to APEX. The responsibility can also be used to control what data is displayed to the user.

Details of the user's responsibility can be stored in APEX application-level items so that they are available throughout the APEX application. You will probably also want to modify the APEX page template to add the responsibility name held in the application-level item into the template so that the current responsibility is shown on every page. If you are using the ICX cookie then adding the code in Figure 11 to the 'before header' process saves the user's security details into the application-level items.

Figure 11
```
:FND_GLOBAL_USER_ID := FND_GLOBAL.user_id;
:FND_GLOBAL_RESP_ID := FND_GLOBAL.resp_id;
:FND_GLOBAL_RESP_APPL_ID := FND_GLOBAL.resp_appl_id;
:FND_GLOBAL_RESP_NAME := FND_GLOBAL.resp_name;
```

For users logging on directly to APEX, a responsibility field can be added onto the standard login page (page 101) so that the users can enter a responsibility as well as the username and password. The responsibility entered can then be validated by adding to the start of the logon process the code (shown in Figure 12) which checks the responsibility entered and loads the user's security details into the application-level items.

Figure 12
```
BEGIN
SELECT fu.user_id, fr.responsibility_id,
       fr.application_id, fr.responsibility_name
INTO :FND_GLOBAL_USER_ID, :FND_GLOBAL_RESP_ID,
     :FND_GLOBAL_RESP_APPL_ID, :FND_GLOBAL_RESP_NAME
FROM fnd_user_resp_groups furg
   , fnd_responsibility_vl fr
   , fnd_user fu
WHERE furg.responsibility_id = fr.responsibility_id
AND furg.responsibility_application_id = fr.application_id
AND furg.user_id = fu.user_id
AND (TRUNC(SYSDATE) <= fr.end_date OR fr.end_date IS NULL)
AND (TRUNC(SYSDATE) >= fr.start_date OR fr.start_date IS NULL)
AND fu.user_name = UPPER(:P101_USERNAME)
and fr.responsibility_name = :P101_RESPONSIBILITY;
EXCEPTION WHEN NO_DATA_FOUND THEN :P101_PASSWORD := NULL;
END;
```

If the APEX application requires every session to be initialised for the E-Business Suite user and responsibility then this can be achieved by entering a call to the FND_GLOBAL.apps_initialize procedure into Virtual Private Database (VPD) security attribute for the APEX application as shown in Figure 13.

Figure 13

```
IF :FND_GLOBAL_USER_ID IS NOT NULL THEN
  FND_GLOBAL.apps_initialize(
     :FND_GLOBAL_USER_ID,
     :FND_GLOBAL_RESP_ID,
     :FND_GLOBAL_RESP_APPL__ID);
END IF;
```

An APEX Authorization Scheme is the mechanism that APEX uses to control access to different components within an application. You can create authorization schemes based on the E-Business Suite responsibilities and then attach the authorization schemes to different components within APEX. For example, you can make some buttons or pages only visible to users that have connected with manager responsibilities.

Conclusion

Oracle supplies many different tools that can be used to extend and customize Oracle E-Business Suite. APEX offers a low cost alternative to Oracle Forms, Reports and ADI. A fully functional web application can be built using APEX in a few days and there are also many packaged applications that can be downloaded from OTN (http://www.oracle.com/technology/products/database/application_express/ packaged_apps/packaged_apps.html).

APEX will run on an existing E-Business Suite infrastructure and use existing load sharing mechanisms. User administration is minimal because users can logon to APEX using their existing username, password and responsibilities.

With APEX you can rapidly build applications that integrate with any E-Business Suite module or implement standalone APEX applications to which E-Business Suite users can connect. The APEX wizards let you quickly develop reports, forms, charts and spreadsheet uploads that integrate with E-Business Suite 11i and R12.

About the Author

Rod West has been using Oracle databases since 1985 and is principal consultant at Cabot Consulting. He specialises in Oracle E-Business Suite and Discoverer. Rod can be contacted at rodwest@cabotconsulting.co.uk.