Deploying and Developing Application Express with Oracle Database 12c

David Peake
Product Manager
Oracle Application Express
Oracle USA

Key Words
Oracle Application Express, Oracle Database 12c, Multitenant Architecture, CDB, PDB

Introduction

Oracle Application Express (APEX) is the native Web application development framework for the Oracle Database. Since its inception in 2004 as a feature of Oracle Database 10g, Oracle Application Express has seen tremendous uptake, with hundreds of thousands of downloads, and thousands of customers with secure, scalable web applications deployed into production. Oracle Application Express is a standard component of the Oracle Database. Oracle Application Express Release 4.2.0.00.08 is installed by default in Oracle Database 12c.

Oracle Database 12c Release 1 (12.1) introduces the Multitenant Architecture. This database architecture has a multitenant container database (CDB) that includes a root container, `CDB$ROOT`, a seed database, `PDB$SEED`, and multiple pluggable databases (PDBs). Each pluggable database is equivalent to a separate database instance in Oracle Database release 11g. The root container, `CDB$ROOT`, holds common objects that are accessible to every PDB utilizing metadata links or object links. The seed database, `PDB$SEED`, is used when creating a new PDB to seed the new database. The key benefit of the Oracle Database 12c multitenant architecture is that the database resources, such as CPU and memory, can be shared across all of the PDBs. This architecture also enables many databases to be treated as one for tasks such as upgrades or patches, and backups.

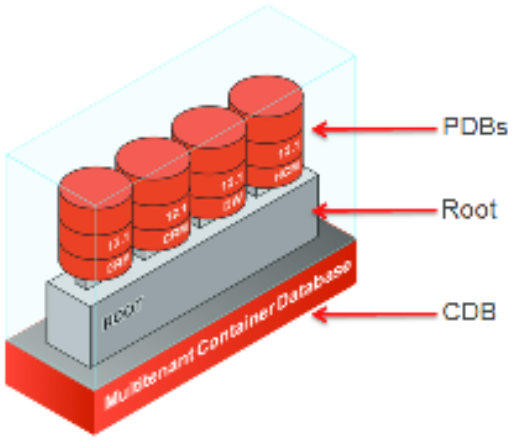

Figure 1 - Oracle Database 12c Multitenant

Oracle Application Express Release 4.2 is the earliest release that can be configured with Oracle Database 12c. When choosing to install Oracle Database 12c using the Oracle Database 12c multitenant architecture, Oracle Application Express 4.2.0 is installed as a common database option by default. You may also choose to uninstall Application Express from the root container and then install a local Application Express, Release 4.2.0 or later, individually into one or more PDBs.

Default Installation

When configuring multitenant architecture, Oracle Application Express 4.2.0 is installed in the root container database by default. In such an installation the root container, `CDB$ROOT`, includes the `APEX_040200` schema to store the common database objects for the Application Express engine such as packages, functions, procedures and views.

The seed database, `PDB$SEED`, and each PDB also includes the `APEX_040200` schema to store the tables that are part of the Application Express engine. As such there are multiple copies of the Application Express engine tables and only single copies of the Application Express engine packages, functions, procedures and views. Each PDB will have the `APEX_040200` schema and have its own copy of the Application Express engine's tables so that it can hold the meta-data for the Application Express applications defined within that PDB. The common database objects are accessed using meta-data links.

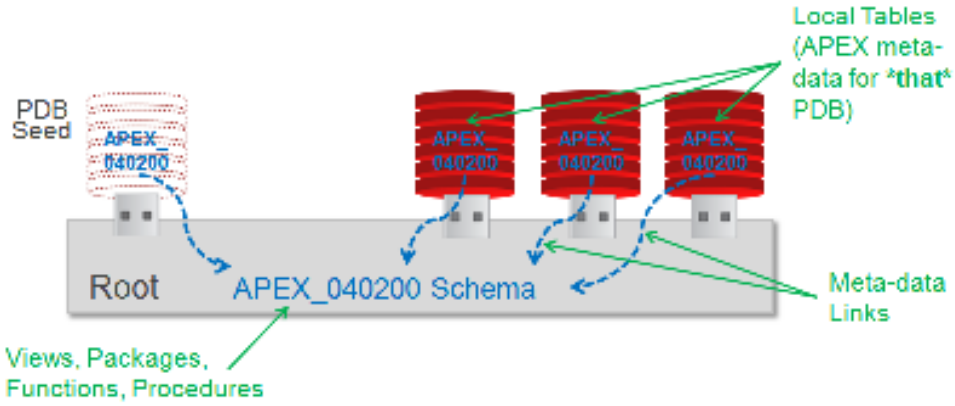

Figure 2 - Default Application Express Installation

To access Application Express you must configure a Web Listener for ***each*** PDB. When configuring the Web Listener(s) a unique port must be specified for each PDB. If using the APEX Listener, Release 2.0 or above, you can utilize a single installation and define access for multiple PDBs. Similarly, you can define multiple Database Access Descriptors (DADs) if using the Oracle HTTP Server with mod_plsql. If utilizing the Embedded PL/SQL Gateway (EPG) as a Web Listener for Application Express you will need to configure EPG in the root container and also within each PDB. Application Express ***can not*** be configured against the root container, `CDB$ROOT`, or the seed database, `PDB$SEED`.

Creating a New PDB from Seed

You can create a new PDB by copying `PDB$SEED`. Given that `PDB$SEED` includes the `APEX_040200` schema the new PDB will also have this schema which contains the Application Express tables. When

the PDB is created the meta-data links back to the common database objects will also be created. To utilize Application Express within this new PDB you must configure a Web Listener with a unique port.

Copying a PDB to another CDB

If both the source CDB database and target CDB database utilize the default installation, whereby Application Express is installed in the root container, then you can readily copy or move a PDB between the CDBs. When the PDB is installed into the target CDB then the meta-data links will automatically be re-created. The only configuration step required is to define the unique port within the Web Listener.

Upgrading from an Earlier Database Release

There are no special steps required when upgrading Oracle Database 11gR2, or earlier, to Oracle Database 12c Multitenant when Application Express is utilized in the original database. Part of the standard upgrade procedure is to run the *noncdb_to_pdb.sql* script once the new PDB has been plugged into a CDB. This script will remove the common objects from the `APEX_040200` schema within the new PDB and create meta-data links.

Note that when the original database is upgraded to Oracle Database 12.1 then Application Express will be upgraded to Release 4.2.0 unless that version or later of Application Express is already installed.

Incompatible Versions when Copying / Moving

When copying or moving a PDB from one CDB to another CDB then it is important to review the release of Application Express in both the source CDB and target CDB. If the Application Express release is higher in the source CDB then you will need to patch or upgrade the target CDB to the same release before copying the PDB to that CDB. Otherwise, you will not be able to run Application Express within the new PDB as the underlying common objects will be incompatible.

If the Application Express release is lower in the source CDB than that installed in the target CDB, then you have two options. The first option is to patch or upgrade the release in the source CDB to that running in the target CDB. All of the PDBs in the source CDB will now be running this later release of Application Express. Alternatively, you can copy or move the PDB to the target CDB and then log into the new PDB and run *catcon.pl* commands as specified in the Application Express Installation Guide.

Installation Options

Oracle Application Express is unique amongst Oracle Database 12c options in that it can be uninstalled from the root container, `CDB$ROOT`, unlike other options . This capability allows you to re-install "runtime only" Application Express into the root container. Alternatively, once Application Express is removed from the root container you can manually install different versions of Application Express, Release 4.2.0 or above, into each PDB.

Having a common Oracle Application Express installation in the root container ensures every PDB is running the same version and patch level of Oracle Application Express. As such supporting multiple PDBs is simplified. This also offers ease of administration by enabling centralized management of Oracle Application Express infrastructure tasks, such as upgrading and patching.

The other alternative is to uninstall Application Express from `CDB$ROOT`, `PDB$SEED`, and all other PDBs, and then install Application Express locally into each PDB. The advantage of a local Oracle Application Express is that you can run different versions of Application Express in each PDB. However, the disadvantage of choosing this option is that each local Oracle Application Express installation must be maintained separately. To perform an upgrade or patch for Application Express you would need to log into each PDB and perform the necessary steps and then log into the next PDB and repeat the same steps. As the number of PDBs grows this will add significantly to the management overhead compared to a common installation of Application Express.

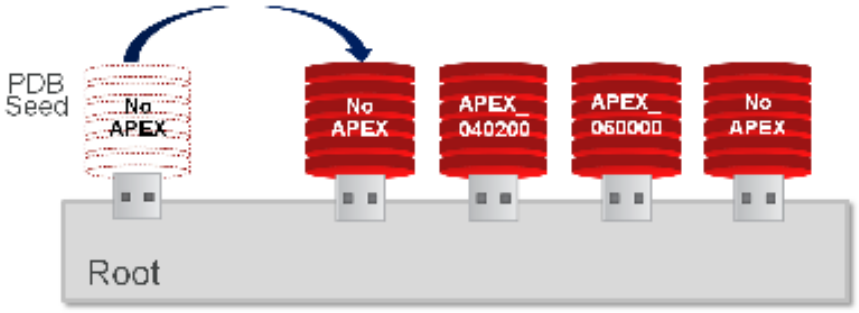

Figure 3 - Uninstalling from Root and Installing Different Versions

Container Scripts

Oracle Application Express 4.2.0.00.08, and later, include a number of new container scripts (**_con.sql*). These scripts are specifically engineered to run within the root container, `CDB$ROOT`. When you need to perform actions on the common installation of Application Express, such as uninstalling, installing, patching, and so forth, you will need to log into `CDB$ROOT` using SYS as SYSDBA and utilize the appropriate container script, for example *apexremov_con.sql*. If you try and run container scripts from within a PDB the script will fail.

If you have uninstalled Application Express from the root container then you can log into a specific PDB and utilize the standard scripts, such as *apexins.sql*. However, if you have not removed Application Express from the root container and try to run a standard script from within a PDB then the script will fail. Similarly if you try and run standard scripts from within the root container they will also fail.

In an Oracle Database 12c non-CDB (stand-alone) configuration and in Oracle Database 11gR2, or earlier, you will still use the standard scripts, for example *apexremov.sql*.

Uninstalling Common Application Express

Post-installation of Oracle Database 12c Multitenant you can log into the root container and run *apxremov_con.sql*. This will remove Application Express from the root container, the PDB Seed and all PDBs within the CDB. It is very important to understand that this will also remove all of the Application Express meta-data (application definitions) stored within all of the PDBs. Therefore, Oracle strongly recommends that this script only be run when initially configuring the CDB, and before starting any Application Express development, to ensure applications are not inadvertently deleted.

In production and test environments Oracle recommends that you install "runtime only" Application Express to harden the environment and minimize the attack surface. In order to achieve this within Oracle Database 12c Multitenant you are recommended to remove Application Express from the root container and then re-install runtime only Application Express, by logging into `CDB$ROOT` using SYS as SYSDBA and running *apxremov_con.sql* and then *apxrtins_con.sql*. These steps should be performed before installing applications into any of the PDBs.

As of the writing of this paper Database Bug # 16946990 prevents you from installing Application Express directly into a PDB. Once a patch for this bug is available and has been applied, and assuming Application Express has been removed from the root container, you can log into any PDB and run *apexins.sql* or *apxrtins.sql* to install Application Express directly into that specific PDB. A Web Listener will need to be configured for this PDB, specifying a unique port.

Copying / Moving PDBs between Non-Default Installations

A major complication introduced by having the different Application Express installation alternatives, in the root container or individually in PDBs, is manifested when copying or moving PDBs between different CDBs with different configurations. Different steps are required depending on where Application Express is installed in the source and target CDBs. With all of the different scenarios you will need to configure the Web Listener once the PDB is in the target CDB.

If Application Express is not installed in the root container of the source and target CDBs then you can simply copy or move the PDB between the CDBs. However, if Application Express is installed directly into a PDB that is being copied or moved to a CDB where Application Express is installed in the root container then after moving the PDB you will need to log into that PDB in the target CDB and run *apex_to_common.sql*. This script will remove the common objects within the PDB's `APEX_040200` schema and create the meta-data links to the `APEX_040200` schema within the root container.

If you removed Application Express from the source CDB and have not installed Application Express directly into the PDB, and you plan to copy or move this PDB to a target CDB which has Application Express in the root container then you will need to perform additional steps. First you will need to log into the PDB, while it is still in the source CDB, and install Application Express directly by running *apexins.sql* or *apxrtins.sql* as appropriate. After copying or moving the PDB log into the PDB, within the target CDB, and run the *apex_to_common.sql* script.

In the scenario where you wish to copy or move a PDB from a source CDB with Application Express installed in the root container to a target CDB where Application Express is not in the root container you will need to contact Oracle Support. Once the PDB is copied or moved to the target CDB the meta-data links can not be created as there is no corresponding Application Express within the root container. Oracle Support will supply you with a version specific *apex_to_local.sql* script to run in the PDB, within the target CDB, which will remove the meta-data links and create the necessary database objects locally.

Oracle Database 12c New Features for Application Express Developers

Apart from the new installation scenarios related to Oracle Database 12c Multitenant there are a number of other new features in Oracle Database 12c that are important to Application Express

developers. The features listed below are those which will have the greatest impact on Application Express developers, some of which necessitated special handling within Application Express.

Varchar2 32K

Oracle Database 12c now allows the maximum size of Varchar2 columns to be specified as 32,767 bytes/characters. To enable this feature the DBA needs to set the max_string_size parameter to EXTENDED and bounce the database. Once specified the DBA can not set this parameter back to the default STANDARD.

Once set developer's can define larger columns in their tables and then they simply set the maximum size for the corresponding items, generally text areas, in their applications. If you are using Application Express collections then you should run the *collection_member_resize.sql* script which will change the c001 to c050 columns within the collection tables to Varchar2(32676).

Default | Default On Null

When you specify this new column definition clause then it will alter how values are inserted into that column. If a column with DEFAULT specified is not included in an insert statement then the default clause will be utilized. If you specify a value explicitly then the DEFAULT clause will not be utilized. However, if you specify DEFAULT ON NULL then the default clause will also be utilized when the column is specified in the insert statement but evaluates to null.

The key value for developers is that it avoids the need for triggers and allows defaults to be added to columns even if the column itself is nullable. There are no specific changes within Application Express for this feature.

Identity Column

Instead of defining a sequence and using statements within triggers or processes to retrieve the next value prior to inserting a record, you can now define an IDENTITY clause directly on a column. This clause will automatically invoke the sequence generator when required. There are additional parameters: ALWAYS [Default] – which will always use the sequence generator; DEFAULT – which uses the sequence generator if the column is not specified; and, DEFAULT ON NULL – which will also use the sequence generator if the explicitly defined value evaluates to null.

Within Application Express the SQL Workshop has been enhanced to handle IDENTITY columns. Developers can specify the IDENTITY clause when creating columns, and the definition is shown in the Object Browser. Additionally, the Create Form and Create Report wizards have been enhanced to incorporate the IDENTITY column. For example, when a form page is generated columns of type IDENTITY will be rendered as 'Display Only'.

Invisible Column

You can include an INVISBLE clause when defining columns. This clause does not prevent access to a column but removes it from various SQL statements. For example, running SELECT * FROM … will not display the INVISIBLE column(s). Similarly, using INSERT INTO x VALUES … will not insert values into the

INVISIBLE column(s). However, such columns can be included in SQL statements by explicitly specifying the column name.

INVISIBLE columns are not listed in SQL Workshop > Object Browser or Query Builder. When using create wizards on a table with INVISIBLE columns then the invisible columns will not be included in the generated pages. After generating reports, developers can alter the report source and manually specify the columns if they want them displayed. Developers can also add such columns as items with a source of Database Column to generated forms.

Data Redaction

In previous Oracle Database releases it was possible to define 'Data Masking' policies. These were used when copying data between databases, such as from production to test or development. This allowed data to be sanitized so that testers and developers could get the benefits of working with production data without exposing sensitive or Personally Identifiable Information(PII) data.

Oracle Advanced Security Data Redaction is very different in that it operates directly on the production database. The main purpose of Data Redaction is to limit access to specific columns based on defined policies as the data is retrieved from the database. It provides selective, on-the-fly redaction of sensitive data in SQL query results prior to display by applications so that unauthorized users cannot view sensitive data. For example, in a call center application people within payroll processing need to be able access all of the employee columns. However, call center operators only need to see the last four digits of the Social Security Number (SSN), should not be able to read an employee's date of birth or PIN.

Data Redaction policies can be defined for all users or limited using an 'expression' clause. Within the Data Redaction expression clause you can incorporate conditions relating to the current Application Express user by specifying v('APP_USER') or nv('APP_USER'). These were included by the Advanced Security development team specifically to enhance the redaction capabilities within Application Express applications.

About the Author

David Peake is Oracle Application Express Product Manager at Oracle Corporation based out of Denver. David has been with Oracle over 19 years. He started within Oracle Consulting in Australia, New Zealand, and then the United States. Within Oracle Consulting his primary focus was on large custom development projects utilizing tools such as Oracle Forms, Oracle Designer, and later Project Marvel, which is now known as Application Express. In 2006, David joined the Server Technologies – Database Tools division within the Product Development team as the Oracle Application Express (APEX) Product Manager. He presents at many conferences around the world including Oracle Openworld, Oracle Develop(s), ODTUG, OGh APEX World and UKOUG.

Contact Information:

David Peake

Oracle Corporation

8704 Technology Way

Denver, CO 80237

USA

Phone: +1 303 334 4371

E-Mail david.peake@oracle.com

Website(s): http://otn.oracle.com/apex

http://apex.oracle.com

Blog: http://dpeake.blogspot.com