Building a REST API to Deploy APEX Apps

By

Md. Ashiqul Islam Shajal (ASC)

- Building the PL/SQL Package:

Such a simplified API could look as in the following **APEX_APPS_REST** package - to install the code into your workspace schema, simply copy and paste and run it using either SQL*Plus, SQLcl, SQL Developer or with APEX SQL Workshop, SQL Scripts.

```
--=========================
-- Wrapper package for ORDS Export / Import API
--
-- Contains procedures to be called by ORDS handlers for exporting or
-- importing an application. This package encapsulates all logic to map
-- the invocation of the ORDS handler to APEX_EXPORT or APEX_APPLICATION_INSTALL
-- package invocations.
--=========================
create or replace package apex_apps_rest
is
--
-- This is the name of the ORDS REST Module
--
c_ords_module_name constant varchar2(16) := 'apex.apps.expimp';

--=========================
-- exports an application or application components, as SQL or ZIP file.
--
-- Parameters:
-- * p_application_file   Application ID to be exported; append ".zip" or ".sql"
--                        to determine the file type.
-- * p_components         Only export the specified components; use syntax
--                        of APEX_EXPORT.GET_APPLICATION procedure; components
--                        separated by comma.
-- * p_mimetype           mimetype of the expected target file. Supports .sql or
.zip
--                        and .json in the future. Overrides the suffix specified
--                        in p_application_file.
--=========================
procedure export(
    p_application_file in varchar2,
    p_components       in varchar2,
    p_mimetype         in varchar2 );

--=========================
-- imports an application or application components, as SQL or ZIP file.
--
-- Parameters:
```

```
-- * p_export_file        Export file
-- * p_mimetype           Mime Type of the export file, to determine whether
--                        this is ZIP or SQL
-- * p_application_id     Import file as this application ID
-- * p_to_workspace       if provided, import into this workspace
--=======================
procedure import(
    p_export_file    in blob,
    p_mimetype       in varchar2,
    p_to_workspace   in varchar2 default null,
    p_application_id in number   default null );

--=======================
-- deletes an application.
--
-- Parameters:
-- * p_in_workspace       if provided, delete application in this workspace
-- * p_application_id     Application ID to be deleted; extension will be
ignored.
--====================
procedure delete(
    p_in_workspace   in varchar2 default null,
    p_application_id in number );

end apex_apps_rest;
/
```

The **EXPORT**, **IMPORT** and **DELETE** procedures of this package are simple enough to be exposed as a REST API.

- **EXPORT:**
  Based on the **P_MIMETYPE** parameter, the export is provided as SQL or as a ZIP file. If **P_COMPONENTS** is not passed, the procedure exports the whole application, otherwise it returns application components.
- **IMPORT:**
  Imports the BLOB which is passed in. The **P_MIMETYPE** argument indicates whether a ZIP or a SQL file was passed in. Since the database schema where this code runs in might be mapped to multiple APEX workspaces, the procedure allows to optionally pass in the workspace name also (**P_TO_WORKSPACE**).
- **DELETE:**
  This one is simple - it just deletes the specified application. As for the **IMPORT** procedure, there is a parameter to pass in the workspace, if required (**P_IN_WORKSPACE**).

The following code contains the implementation of the package.The logic is pretty simple: it does not do much more than preparing parameters and invoking **APEX_EXPORT** or **APEX_APPLICATION_INSTALL**.

```
--============================
-- Package implementation
-- (scroll down within the code window to walk through)
--============================
create or replace package body apex_apps_rest
is

LF constant varchar2(1) := chr( 10 );

--==========================
-- Helper Function: Convert a CLOB to a BLOB
--==========================
function clob_to_blob(
    p_clob in clob )
    return blob
is
    l_blob      blob;
    l_dstoff    pls_integer := 1;
    l_srcoff    pls_integer := 1;
    l_lngctx    pls_integer := 0;
    l_warn      pls_integer;
begin
    sys.dbms_lob.createtemporary(
        lob_loc     => l_blob,
        cache       => true,
        dur         => sys.dbms_lob.call );

    sys.dbms_lob.converttoblob(
        dest_lob     => l_blob,
        src_clob     => p_clob,
        amount       => sys.dbms_lob.lobmaxsize,
        dest_offset  => l_dstoff,
        src_offset   => l_srcoff,
        blob_csid    => nls_charset_id( 'AL32UTF8' ),
        lang_context => l_lngctx,
        warning      => l_warn );

    return l_blob;
end clob_to_blob;
```

```
--============================
-- Helper Function: Convert a BLOB to a CLOB
--============================
function blob_to_clob(
    p_blob in blob )
    return clob
is
    l_clob      clob;
    l_dstoff    pls_integer := 1;
    l_srcoff    pls_integer := 1;
    l_lngctx    pls_integer := 0;
    l_warn      pls_integer;
begin
    sys.dbms_lob.createtemporary(
        lob_loc     => l_clob,
        cache       => true,
        dur         => sys.dbms_lob.call );

    sys.dbms_lob.converttoclob(
        dest_lob     => l_clob,
        src_blob     => p_blob,
        amount       => sys.dbms_lob.lobmaxsize,
        dest_offset  => l_dstoff,
        src_offset   => l_srcoff,
        blob_csid    => nls_charset_id( 'AL32UTF8' ),
        lang_context => l_lngctx,
        warning      => l_warn );

    return l_clob;
end blob_to_clob;

--============================
-- split filename to file name and extension
--============================
procedure split_filename(
    p_full_filename  in varchar2,
    p_filename      out varchar2,
    p_extension     out varchar2 )
is
begin
    if instr( p_full_filename, '.' ) > 0 then
        p_filename  := substr( p_full_filename, 1, instr( p_full_filename, '.' )
- 1 );
```

```
        p_extension := lower( substr( p_full_filename, instr( p_full_filename,
'.' ) + 1 ) );
    else
        p_filename := p_full_filename;
    end if;
end split_filename;

--============================
-- sets workspace to specified workspace, or to first workspace assigned to
-- current schema
--============================
procedure set_workspace( p_workspace in varchar2 )
is
begin
    if p_workspace is not null then
        apex_util.set_workspace( p_workspace );
    else
        for w in (
            select workspace
              from apex_workspaces
             where rownum = 1 )
        loop
            apex_util.set_workspace( w.workspace );
        end loop;
    end if;
end set_workspace;

--=========================
-- Public API, see specification
--=========================
procedure delete(
    p_in_workspace   in varchar2 default null,
    p_application_id in number )
is
begin
    set_workspace( p_workspace => p_in_workspace );
    apex_application_install.remove_application( p_application_id =>
p_application_id );
end delete;

--=======================
-- Public API, see specification
--=======================
procedure export(
    p_application_file in varchar2,
```

```
    p_components       in varchar2,
    p_mimetype         in varchar2 )
is
    l_files       apex_t_export_files;
    l_filename    varchar2(255);
    l_extension   varchar2(255);

    l_components  apex_t_varchar2;
    l_blob        blob;

    l_as_zip      boolean;
begin
    split_filename(
        p_full_filename => p_application_file,
        p_filename      => l_filename,
        p_extension     => l_extension );

    l_as_zip := case when p_mimetype is null
                    then coalesce( l_extension = 'zip', false )
                    else coalesce( lower( p_mimetype ) = 'application/zip', false
)
                end;

    if p_components is not null then
        l_components := apex_string.split( ltrim(rtrim( p_components ) ) , ',' );
    end if;

    l_files := apex_export.get_application(
                    p_application_id => to_number( l_filename ),
                    p_components     => l_components,
                    p_split          => l_as_zip );

    sys.dbms_lob.createtemporary(
        lob_loc     => l_blob,
        cache       => true,
        dur         => sys.dbms_lob.call );

    if l_as_zip then
        for i in 1 .. l_files.count loop
            apex_zip.add_file (
                p_zipped_blob => l_blob,
                p_file_name   => l_files(i).name,
                p_content     => clob_to_blob( l_files(i).contents ) );
        end loop;
        apex_zip.finish( l_blob );
```

```
        sys.owa_util.mime_header( 'application/zip', false );
    else
        l_blob := clob_to_blob( l_files(1).contents );
        sys.owa_util.mime_header( 'application/sql', false );
    end if;

    sys.htp.p( 'Content-Length: ' || sys.dbms_lob.getlength( l_blob ) );
    sys.htp.p( 'Content-Disposition: attachment; filename=' || l_filename || '.'
|| case when l_as_zip then 'zip' else 'sql' end );
    sys.owa_util.http_header_close;
    sys.wpg_docload.download_file( l_blob );

end export;

--============================
-- Public API, see specification
--============================
procedure import(
    p_export_file    in blob,
    p_mimetype       in varchar2,
    p_to_workspace   in varchar2 default null,
    p_application_id in number   default null )
is
    l_files         apex_t_export_files := apex_t_export_files();
    l_zip_files     apex_zip.t_files;
    --
    l_dstoff        pls_integer := 1;
    l_srcoff        pls_integer := 1;
    l_lngctx        pls_integer := 0;
    l_warn          pls_integer;
begin

    set_workspace( p_workspace => p_to_workspace );

    if lower( p_mimetype ) = 'application/zip' then
        l_zip_files := apex_zip.get_files(
                           p_zipped_blob => p_export_file,
                           p_only_files  => true );

        l_files.extend( l_zip_files.count );
        for i in 1 .. l_zip_files.count loop
            l_files( i ) := apex_t_export_file(
                                l_zip_files( i ),
                                blob_to_clob(
                                    apex_zip.get_file_content(
```

```
                                        p_zipped_blob => p_export_file,
                                        p_file_name   => l_zip_files( i ) ) ) );
        end loop;
    else
        l_files.extend(1);
        l_files( 1 ) := apex_t_export_file( 'import-data.sql', blob_to_clob(
p_export_file ) );
    end if;

    apex_application_install.set_application_id(
        p_application_id => p_application_id );

    apex_application_install.install(
        p_source             => l_files,
        p_overwrite_existing => true );

end import;

end apex_apps_rest;
/
```

- Creating the ORDS REST Module:

With **SQLcl** or **SQL*Plus**, we could start using this package right now. However, that is not the goal of the exercise - instead we'll now use the ORDS package in order to build a *REST Module* with *REST Handlers* to import, export and delete applications. These ORDS handlers will just call into the new **APEX_APPS_REST** package.

```
--================================
-- Set up the ORDS REST Module and its handlers
-- (scroll down within the code window to walk through)
--================================
begin
    ords.enable_schema;
end;
/
sho err

--
-- delete the module if it already exists, to make this script re-runnable.
--
```

```
begin
    ords.delete_module(
        p_module_name => apex_apps_rest.c_ords_module_name );
exception
    -- ignore errors ...
    when others then null;
end;
/
sho err

begin
    ords.define_module(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_base_path      => 'deploy/app/' );

    --------------------------------
    -- Export Handler for the full application
    --
    -- Parameters:
    -- * app_id (URL)               ID of the application to export
    -- * Accept (Request Header)    format in which to return the export file
    --
    -- Example:
    --
    -- curl -X GET
    --      -H "Accept: application/sql
    --      http://localhost:8080/ords/schema/deploy/app/102
    --------------------------------
    ords.define_template(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_pattern        => ':app_file' );

    ords.define_handler(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_pattern        => ':app_file',
        p_method         => 'GET',
        p_source_type    => ords.source_type_plsql,
        p_source         =>
q'~begin
    apex_apps_rest.export(
        p_application_file => :app_file,
        p_components       => null,
        p_mimetype         => null );
end;~' );
```

```
    ords.define_parameter(
        p_module_name        => apex_apps_rest.c_ords_module_name,
        p_pattern            => ':app_file',
        p_method             => 'GET',
        p_name               => 'Accept',
        p_bind_variable_name => 'ACCEPT',
        p_source_type        => 'HEADER' );

    --------------------------------
    -- Export Handler for application components
    --
    -- Parameters:
    -- * app_id (URL)               ID of the application to export
    -- * Accept (Request Header)    format in which to return the export file
    -- *        (Request Body)      components to export, as outlined in the
documentation
    --                              for APEX_EXPORT.GET_APPLICATION. Components
separated
    --                              by comma.
    --
    -- Example:
    --
    -- curl -X POST
    --      -H "Accept: application/sql
    --      -d 'PAGE:1,PAGE:2'
    --      http://localhost:8080/ords/schema/deploy/app/102/components
    --------------------------------
    ords.define_template(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_pattern        => ':app_id/components' );

    ords.define_handler(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_pattern        => ':app_id/components',
        p_method         => 'POST',
        p_source_type    => ords.source_type_plsql,
        p_source         =>
q'~begin
    apex_apps_rest.export(
        p_application_file => :app_id,
        p_components       => :body_text,
        p_mimetype         => :accept );
end;~' );

    ords.define_parameter(
```

```
        p_module_name        => apex_apps_rest.c_ords_module_name,
        p_pattern            => ':app_id/components',
        p_method             => 'POST',
        p_name               => 'Accept',
        p_bind_variable_name => 'ACCEPT',
        p_source_type        => 'HEADER' );

    ----------------------------------
    -- Import Handler
    -- curl -X POST
    --      -H "Content-Type: tapplication/octet-stream"
    --      --data-binary @f101.sql
    --      http://localhost:8080/ords/schema/deploy/app/102
    --
    -- Parameters:
    -- X-Target-Workspace - HTTP Header
    ----------------------------------
    ords.define_template(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_pattern        => ':app_id/' );

    ords.define_handler(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_pattern        => ':app_id/',
        p_method         => 'POST',
        p_source_type    => ords.source_type_plsql,
        p_source         =>
q'~begin
    apex_apps_rest.import(
        p_application_id => :app_id,
        p_mimetype       => :content_type,
        p_to_workspace   => :workspace,
        p_export_file    => :body );
end;~' );

    ords.define_parameter(
        p_module_name        => apex_apps_rest.c_ords_module_name,
        p_pattern            => ':app_id/',
        p_method             => 'POST',
        p_name               => 'X-Target-Workspace',
        p_bind_variable_name => 'WORKSPACE',
        p_source_type        => 'HEADER' );

    ----------------------------------
    -- Delete Handler
```

```
    -- curl -X DELETE
    --      http://localhost:8080/ords/schema/deploy/app/102
    --
    -- Parameters:
    -- X-Target-Workspace - HTTP Header
    -----------------------------------
    ords.define_handler(
        p_module_name    => apex_apps_rest.c_ords_module_name,
        p_pattern        => ':app_file',
        p_method         => 'DELETE',
        p_source_type    => ords.source_type_plsql,
        p_source         =>
q'~begin
    apex_apps_rest.delete(
        p_application_id => :app_file,
        p_in_workspace   => :workspace );
end;~' );

    ords.define_parameter(
        p_module_name        => apex_apps_rest.c_ords_module_name,
        p_pattern            => ':app_file',
        p_method             => 'DELETE',
        p_name               => 'X-Target-Workspace',
        p_bind_variable_name => 'WORKSPACE',
        p_source_type        => 'HEADER' );

end;
/
--
-- the COMMIT is important.
commit
/
```

- Testing the new REST API:

Now we have installed the PL/SQL package, as well as the ORDS REST Handlers. We can now do a first test by calling the REST Handler to *export* an application. The following example assumes that application **101** exists in the APEX workspace, which is mapped to the database schema where the package and REST API are installed.

➔ Follow Provided Postman API Collection

E:\ImpExpModuleORDS\ImpExpAPI.json

- Full API Document:

Url: https://blogs.oracle.com/apex/post/building-a-rest-api-to-deploy-apex-apps

- Full API Document: Save this json with .json extension for find the API collection then import in postman.

```
{
    "info": {
        "_postman_id": "97b1c83f-1866-4b27-9b93-62c89ca41a40",
        "name": "ImpExpAPI",
        "schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
    },
    "item": [
        {
            "name": "API",
            "item": [
                {
                    "name": "Export Application",
                    "request": {
                        "auth": {
                            "type": "oauth2",
                            "oauth2": [
                                {
                                    "key": "grant_type",
                                    "value": "client_credentials",
                                    "type": "string"
                                },
                                {
```

```
                                    "key": "authUrl",
                                    "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
                                    "type": "string"
                                },
                                {
                                    "key": "accessTokenUrl",
                                    "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
                                    "type": "string"
                                },
                                {
                                    "key": "clientSecret",
                                    "value": "YoDeS7vvALgSnRYIlbaLpg..",
                                    "type": "string"
                                },
                                {
                                    "key": "clientId",
                                    "value": "kkBhk0rdp8SCBhyV3k9NNQ..",
                                    "type": "string"
                                },
                                {
                                    "key": "addTokenTo",
                                    "value": "header",
                                    "type": "string"
                                }
                            ]
                        },
                        "method": "GET",
                        "header": [],
                        "url": {
                            "raw":
"http://10.11.201.82:8080/cblagent/services/deploy/app/110",
                            "protocol": "http",
                            "host": [
                                "10",
                                "11",
                                "201",
                                "82"
                            ],
                            "port": "8080",
                            "path": [
                                "cblagent",
                                "services",
                                "deploy",
```

```
                                "app",
                                "110"
                            ]
                        }
                    },
                    "response": []
                },
                {
                    "name": "Export Application Page",
                    "request": {
                        "auth": {
                            "type": "oauth2",
                            "oauth2": [
                                {
                                    "key": "grant_type",
                                    "value": "client_credentials",
                                    "type": "string"
                                },
                                {
                                    "key": "authUrl",
                                    "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
                                    "type": "string"
                                },
                                {
                                    "key": "accessTokenUrl",
                                    "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
                                    "type": "string"
                                },
                                {
                                    "key": "clientSecret",
                                    "value": "YoDeS7vvALgSnRYIlbaLpg..",
                                    "type": "string"
                                },
                                {
                                    "key": "clientId",
                                    "value": "kkBhk0rdp8SCBhyV3k9NNQ..",
                                    "type": "string"
                                },
                                {
                                    "key": "addTokenTo",
                                    "value": "header",
                                    "type": "string"
                                }
```

```
                        ]
                    },
                    "method": "POST",
                    "header": [],
                    "body": {
                        "mode": "raw",
                        "raw": "PAGE:2"
                    },
                    "url": {
                        "raw":
"http://10.11.201.82:8080/cblagent/services/deploy/app/110/components",
                        "protocol": "http",
                        "host": [
                            "10",
                            "11",
                            "201",
                            "82"
                        ],
                        "port": "8080",
                        "path": [
                            "cblagent",
                            "services",
                            "deploy",
                            "app",
                            "110",
                            "components"
                        ]
                    }
                },
                "response": []
            },
            {
                "name": "Import Application",
                "request": {
                    "auth": {
                        "type": "oauth2",
                        "oauth2": [
                            {
                                "key": "grant_type",
                                "value": "client_credentials",
                                "type": "string"
                            },
                            {
                                "key": "authUrl",
```

```
                                    "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
                                    "type": "string"
                                },
                                {
                                    "key": "accessTokenUrl",
                                    "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
                                    "type": "string"
                                },
                                {
                                    "key": "clientSecret",
                                    "value": "YoDeS7vvALgSnRYIlbaLpg..",
                                    "type": "string"
                                },
                                {
                                    "key": "clientId",
                                    "value": "kkBhk0rdp8SCBhyV3k9NNQ..",
                                    "type": "string"
                                },
                                {
                                    "key": "addTokenTo",
                                    "value": "header",
                                    "type": "string"
                                }
                            ]
                        },
                        "method": "POST",
                        "header": [
                            {
                                "key": "Content-Type",
                                "value": "tapplication/octet-stream",
                                "type": "default"
                            },
                            {
                                "key": "X-Target-Workspace",
                                "value": "1400484024213076",
                                "type": "default"
                            }
                        ],
                        "body": {
                            "mode": "file",
                            "file": {
                                "src": "110.sql"
                            }
```

```
                    },
                    "url": {
                        "raw":
"http://10.11.201.82:8080/cblagent/services/deploy/app/111/",
                        "protocol": "http",
                        "host": [
                            "10",
                            "11",
                            "201",
                            "82"
                        ],
                        "port": "8080",
                        "path": [
                            "cblagent",
                            "services",
                            "deploy",
                            "app",
                            "111",
                            ""
                        ]
                    }
                },
                "response": []
            },
            {
                "name": "Import Application Page or Other Component",
                "request": {
                    "auth": {
                        "type": "oauth2",
                        "oauth2": [
                            {
                                "key": "grant_type",
                                "value": "client_credentials",
                                "type": "string"
                            },
                            {
                                "key": "authUrl",
                                "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
                                "type": "string"
                            },
                            {
                                "key": "accessTokenUrl",
                                "value":
"http://10.11.201.82:8080/cblagent/services/oauth/token",
```

```
                                "type": "string"
                            },
                            {
                                "key": "clientSecret",
                                "value": "YoDeS7vvALgSnRYIlbaLpg..",
                                "type": "string"
                            },
                            {
                                "key": "clientId",
                                "value": "kkBhk0rdp8SCBhyV3k9NNQ..",
                                "type": "string"
                            },
                            {
                                "key": "addTokenTo",
                                "value": "header",
                                "type": "string"
                            }
                        ]
                    },
                    "method": "POST",
                    "header": [
                        {
                            "key": "Content-Type",
                            "value": "tapplication/octet-stream",
                            "type": "default"
                        },
                        {
                            "key": "X-Target-Workspace",
                            "value": "1400484024213076",
                            "type": "default"
                        }
                    ],
                    "body": {
                        "mode": "file",
                        "file": {
                            "src": "110_page_2.sql"
                        }
                    },
                    "url": {
                        "raw":
"http://10.11.201.82:8080/cblagent/services/deploy/app/111/",
                        "protocol": "http",
                        "host": [
                            "10",
                            "11",
```

```
                                "201",
                                "82"
                            ],
                            "port": "8080",
                            "path": [
                                "cblagent",
                                "services",
                                "deploy",
                                "app",
                                "111",
                                ""
                            ]
                        }
                    },
                    "response": []
                }
            ]
        }
    ]
}
```