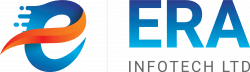
# Building a REST API to Deploy APEX Apps

****

By



* 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](https://docs.oracle.com/en/database/oracle/oracle-rest-data-services/20.4/aelig/ORDS-reference.html#GUID-E4476C14-01B1-4EA4-94D3-73B92C8C9AB3) 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](file:///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": []

                }

            ]

        }

    ]

}