Long Live Multimedia with APEX and REST!

(co-starring: Oracle Autonomous Cloud)

Presented on 18th October 2019

at

hroug 2019

in Rovinj, Croatia.

by

Niall Mc Phillips
Long Acre sàrl
niall.mcphillips@longacre.ch
@Niall_McP

long acre
solutions today

1

About me: Niall Mc Phillips

Owner + CEO, Long Acre (founded 2015).

Co-owner + Director, Stephenson and Associates (founded 1995).

Based in Geneva, Switzerland.

- Oracle ACE ♠
- Background in Computer Science
- Working with Oracle database as a Developer and DBA since 1989
- Developing database-centric web applications since 1995
- Developing with APEX since 2005 (HTML DB 1.6)
- Organiser of the Swiss APEX Meetup group

long acre
solutions today

2

Some of our clients

- UN – United Nations
- ILO - International Labour Organization*
- WHO – World Health Organisation*
- ISSA – International Social Services Association
- IEC - International Electrotechnical Commission*
- Royal Dutch Shell
- Addax Petroleum*
- APPA – African Petroleum Producers Association*
- DGH Gabon (government)*
- Nestlé
- Petronas
- ExxonMobil

* APEX

3

500+ Technical Experts Helping Peers Globally

ORACLE ACE Program

ORACLE ACE Director

ORACLE ACE

ORACLE ACE Associate

3 Membership Tiers
- Oracle ACE Director
- Oracle ACE
- Oracle ACE Associate

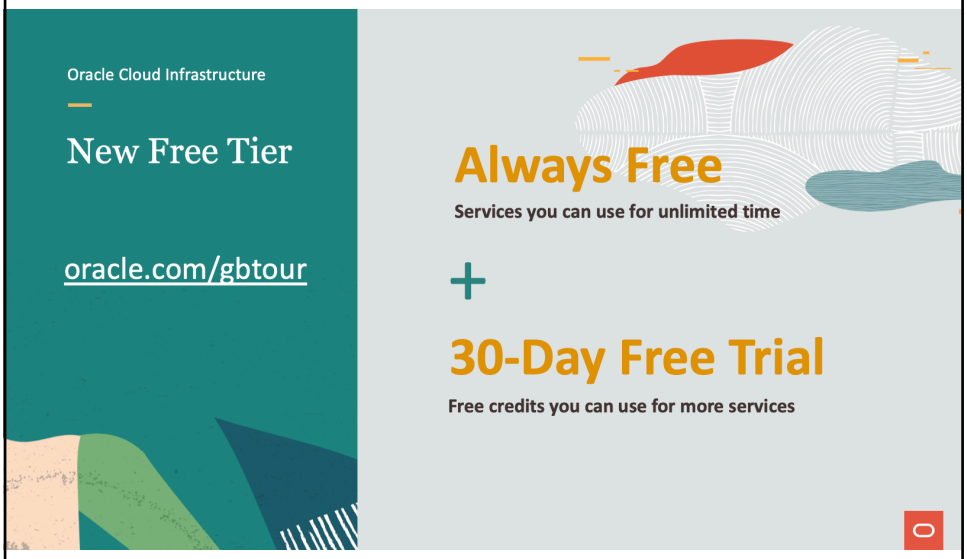
bit.ly/OracleACEProgram

Connect:
oracle-ace_ww@oracle.com
Facebook.com/oracleaces
@oracleace

Oracle Groundbreakers

Nominate yourself or someone you know: acenomination.oracle.com

4



5

Oracle Multimedia

- Available since Oracle8 (1997)
- Is an integral part of all Oracle databases up to 18c.
- It's a one-stop shop – everything is already there inside your Oracle DB.

long acre
solutions today

6

Oracle multimedia object types

- ORACLE Multimedia object types are implemented as types inside the database
- Included in Oracle SE and EE editions and in 18c XE
- Are part of the ORDSYS schema
- Can be used as native columns in tables or as variables in PL/SQL

Different Oracle Multimedia types

ORDSYS types

- ORDAudio
- ORDDoc
- ORDImage
- ORDVideo
- ORDDicom
- ORDSource

Examining the ORDImage object type

Type definition

```
-------------------
-- TYPE ATTRIBUTES
-------------------
source              ORDSource,
height              INTEGER,
width               INTEGER,
contentLength       INTEGER,
fileFormat          VARCHAR2(4000),
contentFormat       VARCHAR2(4000),
compressionFormat   VARCHAR2(4000),
mimeType            VARCHAR2(4000),
```

The Oracle Multimedia ORDImage object type

ORDImage Methods available

checkProperties( )
copy( )
getCompressionFormat( )
getContentFormat( )
getContentLength( )
getDicomMetadata( )
getFileFormat( )
getHeight( )
getMetadata( )
getWidth( )
import( )
importFrom( )
process( )
processCopy( )
putMetadata( )
setProperties( )

Oracle Multimedia is dead!

- Born in 1997
- Lived a happy life as an integral part of the Oracle database for almost 24 years
- De-supported and removed as of Oracle 19c
- † **R.I.P. Oracle Multimedia** †
  you will be sadly missed by many

*18c support still available until the end-of-life of 18c (June 2021, maybe longer with extended support?)

Oracle Multimedia is dead!

Long live Multimedia in Oracle with APEX and REST!

Processing Images without Oracle Multimedia

What is essential for us

- image resizing
- image overlay

What would be nice to have

- Metadata extraction (XMP / EXIF / IPTC)

The search for alternatives

- There are many non-Oracle alternatives using JavaScript, Python, Ruby, etc.
- However, we would like to stay within the Oracle ecosystem and are looking at how to do this with the minimum disruption

REST-enabled

- Our conclusion is that the platform that would fit best with our environment is a REST-enabled platform
- Image manipulation that is currently done within the database will be replaced by manipulation outside of the database
- This will allow us to keep our precious data and metadata within Oracle.

Photo Library history

- Originally developed in 2003 on Oracle9i using PL/SQL web toolkit and Oracle MultiMedia.
- New user interface in 2017 to prepare for the 2019 centenary
- Video added in 2018

Photo Library

- Leverage existing infrastructure (DB)
- Store approx. 40’000 photos
- Dates ranging from 1919 to present day
- Store high-res images without manipulation
- Automatically resize / transform (web, detail, watermark)
- Automatically extract metadata and insert into underlying database

Let’s take a look

- Multimedia Download Platform front-end
- Both front-end and back-end are developed in APEX

long acre
solutions today

Preparing for the end of Oracle Multimedia

- Replace the images in the database with BLOBs and process them in the cloud
- Change to data tables in our application

Cloud Processing

For our proof of concept, we will be using Cloudinary as a provider

https://cloudinary.com

There are many choices available for image processing in the cloud

See: Menno Hoogendijk's recent blog post (11 Oct 2019)
https://blogs.oracle.com/apex/alternatives-for-oracle-multimedia

Uploading the photos using APEX and the Oracle Autonomous Cloud

- Create an application in the Oracle Autonomous Cloud
- Create an image upload region
- Create a classic report region

Create an image table

```
drop table my_photos;
create table my_photos   (id number,
                          uploaded_on date,
                          mime_type varchar2(255),
                          name varchar2(400),
                          filename varchar2(400),
                          width  integer,
                          height  integer,
                          image_format  varchar2(30),
                          uploaded_url varchar2(4000),
                          image_properties  clob, -- json
                          thephoto blob,
                          thumbnail blob,
                          watermark blob);

alter table my_photos add constraint pk_my_photos primary key (id) enable;
```

Upload the images using APEX

- Create an simple upload region containing two items and a button

```
Px_PHOTO_UPLOAD - type file browse
Px_UPLOAD_ID – hidden
UPLOAD – button
```

- Create a process to get the uploaded file from the APEX_APPLICATION_TEMP_FILES table and move it to our own table along with a few basic metadata.

Upload the images – then move to the my_photos table

Initial process to execute when a photo is uploaded.

Get the image(s) from the APEX tables store it in our table.

```
declare
  v_id  my_photos.id%type;
begin

  select seq_everything.nextval into v_id from dual;

  insert into my_photos (id, thePhoto, mime_type, filename, uploaded_on)
  select v_id,
         blob_content,
         mime_type,
         filename,
         created_on
    from apex_application_temp_files
   where name = :P111_PHOTO_UPLOAD;

  delete from apex_application_temp_files where name = :P111_PHOTO_UPLOAD;
  :P111_UPLOADED_ID := to_char(v_id);  -- store the ID for later
  commit;
end;
```

Make a report to see what has been uploaded

New Region – Classic Report

```
select id,
       mime_type,
       filename,
       uploaded_on,
       dbms_lob.getlength(thephoto) as  photo_size
from my_photos;
```


Create a procedure to view the photo

This is one technique for viewing the photo directly

```
create or replace procedure showPhoto (p_id in my_photos.id%type)
is
   rec_photo  my_photos%rowtype;
begin
   select * into rec_photo
     from my_photos
    where id = p_id;

   owa_util.mime_header (rec_photo.mime_type, false);
   htp.p('Content-length: ' ||
   to_char(dbms_lob.getlength(rec_photo.thephoto)));
   owa_util.http_header_close;
   wpg_docload.download_file (rec_photo.thephoto);

end showPhoto;
/
grant execute on showPhoto to ords_public_user;
```

Modify the report to see the photo

New Region – Classic Report

```
select id,
       mime_type,
       filename,
       uploaded_on,
       dbms_lob.getlength(thephoto) as photo_size,
       '<img src="'||'niall'||'.showPhoto?p_id='
                   ||trim(to_char(id))||'" />' as photo
  from my_photos;
```

Result: giant photo, unusable for our purposes

Modify the report to reduce the displayed size

Add **`width="300"`** to display a smaller version

```
select id,
       …
       '<img src="'||'niall'||'.showPhoto?p_id='
                   ||trim(to_char(id))
                   ||'" width="300" />' as photo
  from my_photos;
```

Result: Looks ok, ***but*** *the entire full resolution photo has been passed via http to the browser and the resizing has been done by the browser when rendering*

Uploading the image to Cloudinary using APEX_WEB_SERVICE

- *Create a second page process to upload the photo to Cloudinary using APEX_WEB_SERVICE and REST API*
- *Oracle Autonomous Cloud already has the Cloudinary certificates installed.*

  For an on-premises database, you would need to
  - *a) import the certificate into an Oracle Wallet*
  - *b) ensure that the Network ACLs allow you to execute the Web Service call.*

Uploading the image to Cloudinary using APEX_WEB_SERVICE (1)

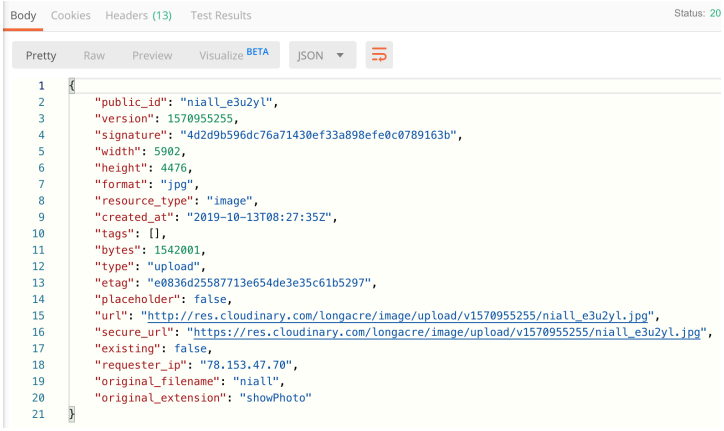
Process that passes the URL of the full resolution uploaded image to Cloudinary via REST

```
declare
  v_clob  clob;
begin
  v_clob := apex_web_service.make_rest_request
(p_url => 'https://api.cloudinary.com/v1_1/longacre/image/upload?upload_preset='
      ||:G_UPLOAD_PRESET||'&file='
      ||apex_util.url_encode('https://'||owa_util.get_cgi_env('SERVER_NAME')
                                       ||'/ords/niall.showPhoto?p_id='
                                       ||trim(:P111_UPLOADED_ID)),
 p_http_method => 'GET');
```

- ***upload_preset*** *is a sort of secret key that I have defined as an application item*
- ***v_clob*** *receives the json response*

Uploading the image to Cloudinary using APEX_WEB_SERVICE (2)

Typical JSON response after upload to Cloudinary



31

Uploading the image to Cloudinary using APEX_WEB_SERVICE (3)

```
... continued from previous code slide…

  -- parse the response and extract some interesting attributes
  apex_json.parse(p_source => v_clob);
  update my_photos  -- save the new metadata
       set uploaded_url     = apex_json.get_varchar2(p_path => 'url'),
           width            = apex_json.get_varchar2(p_path => 'width'),
           height           = apex_json.get_varchar2(p_path => 'height'),
           image_format     = apex_json.get_varchar2(p_path => 'format'),
           image_properties = v_clob
     where id = to_number(:P111_UPLOADED_ID);
end;
```

Verify by adding the upload_url column to the classic report

32

Resize the photo to 300px by calling a Cloudinary Web Service

- Add a process that will call the Cloudinary REST API and ask it to resize the photo.
- To create a 300px wide thumbnail while preserving the aspectratio, we add ***/w_300*** to the URL
- We then save the result in a column named **thumbNail**

Resize the photo to 300px by calling a Cloudinary API Web Service

```
declare
  v_url  varchar2(4000);
  v_clob clob;
  v_blob blob;
begin
  select uploaded_url
    into v_url
    from my_photos
   where id = to_number(:P111_UPLOADED_ID);
  apex_debug.message(v_url);

  -- add the width to the url
  v_url := 'https://api.cloudinary.com/v1_1/longacre/image/upload?upload_preset='
           ||:G_UPLOAD_PRESET || '&file=' || replace(v_url,'/upload/','/upload/w_300/');

  v_clob := apex_web_service.make_rest_request(p_url => v_url, p_http_method => 'GET');
  apex_debug.message(substr(v_clob,1,4000));

  -- parse and extract the secure url, we will then use this to craete a thumbnail
  apex_json.parse(p_source => v_clob);
  v_url := apex_json.get_varchar2(p_path => 'secure_url');

  -- now let's go and get the thumbnail
  v_blob := apex_web_service.make_rest_request_b(v_url,'GET');

  update my_photos
     set thumbnail = v_blob
   where id = to_number(:P111_UPLOADED_ID);
end;
```

Modify the Report to display the now-populated thumbnail column

- Create a new procedure to display the new thumbnail column – *showThumbnail*
- Replace the *showPhoto* reference in the reports by *showThumbnail*
- *Run the report, this time the amount of data being passed is exactly what is needed to show a 300px image.*

Add a watermark/overlay to an image

- Add a process that will call the Cloudinary REST API and ask it to add a watermark
- Add ***/w_500*** to the URL for a 500px wide image
- Add ***/l_text:Arial_40:watermarktext*** for the watermarked text
- Save the result in a column named **watermark**

Extracting metadata

- The photos contain metadata (date, title, description, keywords) placed by the photographers.

- 3 main metadata formats and photos

EXIF, IPTC and XMP

XMP metadata example

XMP metadata example

Getting attributes from the
<rdf:Description>

xmp:CreateDate
dc:title
dc:description

XMP metadata example

Getting the keywords from <dc:keywords>

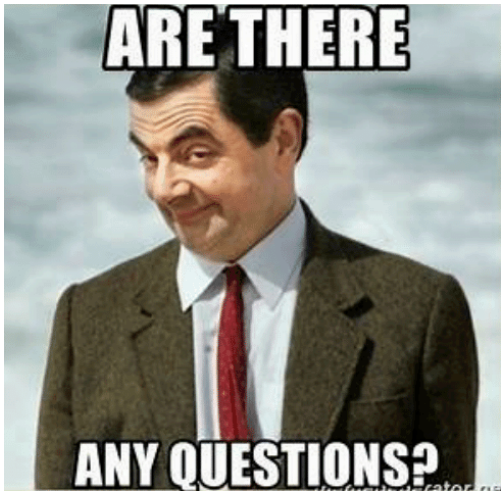
```
<dc:keywords>
    <rdf:Bag>
      <rdf:li>ILO</rdf:li>
      <rdf:li>ILC</rdf:li>
      <rdf:li>2014</rdf:li>
      <rdf:li>opening</rdf:li>
      <rdf:li>session</rdf:li>
      <rdf:li>OIT</rdf:li>
      <rdf:li>International Labour Organisation</rdf:li>
    </rdf:Bag>
  </dc:keywords>
```

XMP metadata

- After extracted the metadata are simply inserted into the appropriate columns of their tables (e.g. date_taken, legend, etc.)

Conclusion

- Oracle Multimedia is going away
- There are many alternatives both within and outside the Oracle Ecosystem
- The alternative that appears to give the most flexibility and allows us to keep substantial portions of the applications already developed is to use Cloud Services via REST and APEX.


ARE THERE
ANY QUESTIONS?