Creating a simple image application in APEX

This is the first part of a series of 3 to be followed in order.

In SQL Developer, create a table to hold your images

CREATE TABLE images

(image\_id NUMBER CONSTRAINT images\_image\_id\_pk PRIMARY KEY,

filename VARCHAR2(100),

image ORDSYS.ORDImage,

thumbnail BLOB);

Note the data type of the thumbnail. This is to ensure that APEX is capable of handling the thumbnail in a report.

Set up a sequence to use to generate a primary key.

CREATE SEQUENCE seq\_image\_id

START WITH 1

INCREMENT BY 1

NOCACHE;

Note that a trigger to populate the primary key is not used in this example.

Create a procedure to load some test images located on the server into the table[[1]](#footnote-1).

CREATE OR REPLACE PROCEDURE load\_image\_from\_file

(p\_filename IN VARCHAR2) AS

l\_image\_id INTEGER;

l\_image ORDSYS.ORDImage;

ctx RAW (64) := NULL;

BEGIN

INSERT INTO images

(

image\_id,

filename,

image

)

VALUES

(

seq\_image\_id.nextval,

p\_filename,

ORDSYS.ORDImage

('FILE',

'ISAD353\_IMAGES',

p\_filename,

1 -- setProperties, default is 0

)

)

RETURNING image\_id, image INTO l\_image\_id, l\_image;

-- useful INSERT variant

l\_image.import(ctx);

-- explicit property setting if needed

-- l\_image.setProperties();

UPDATE images

SET image = l\_image

WHERE image\_id = l\_image\_id;

COMMIT;

EXCEPTION

WHEN others THEN

BEGIN

ROLLBACK;

dbms\_output.put\_line(sqlerrm);

END;

END;

And then call it to load two test images:

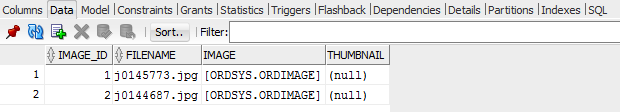
BEGIN

load\_image\_from\_file('j0145773.jpg');

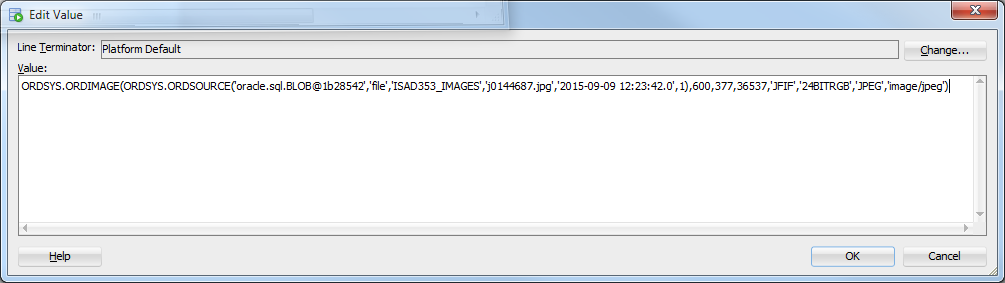
load\_image\_from\_file('j0144687.jpg');

END;

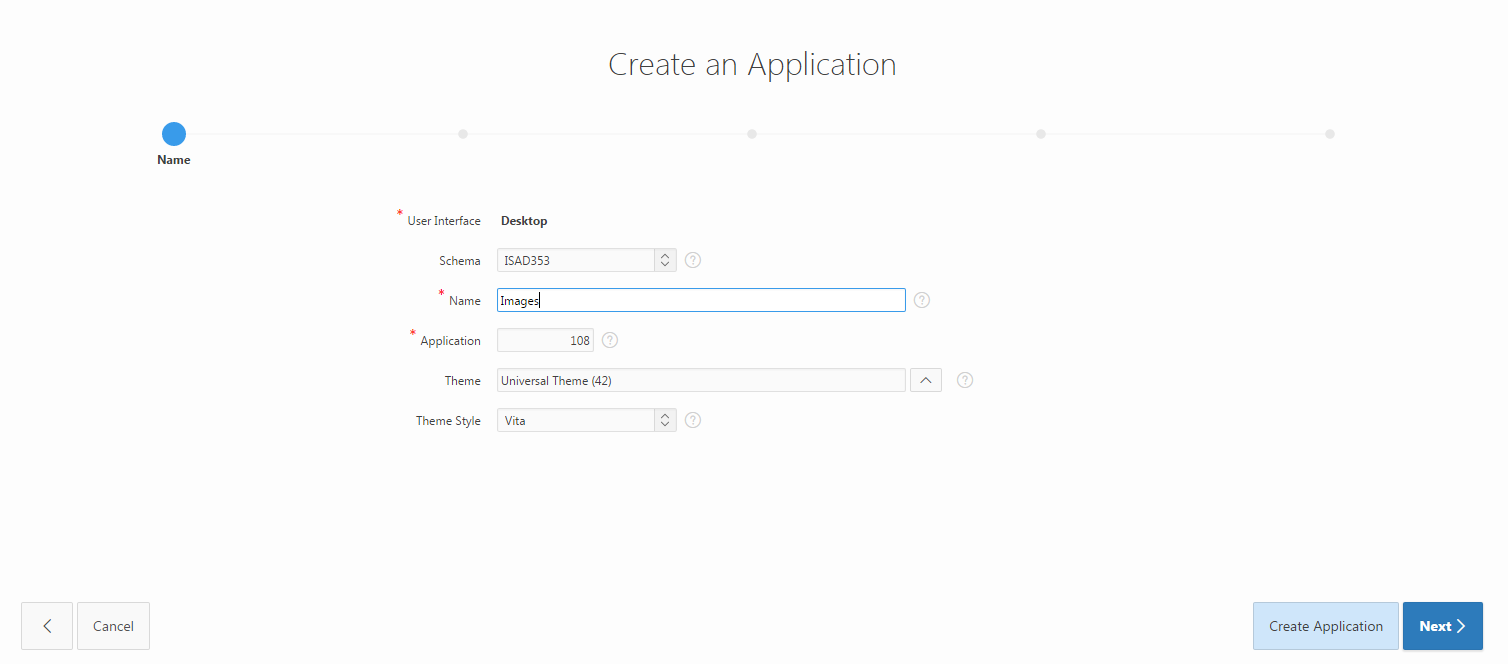
Navigate to the images table and select the data tab – you should see that the two rows have been inserted



Double click on [ORDSYS.ORDIMAGE] and click the pencil icon to reveal the ORDIMAGE object type attribute values.

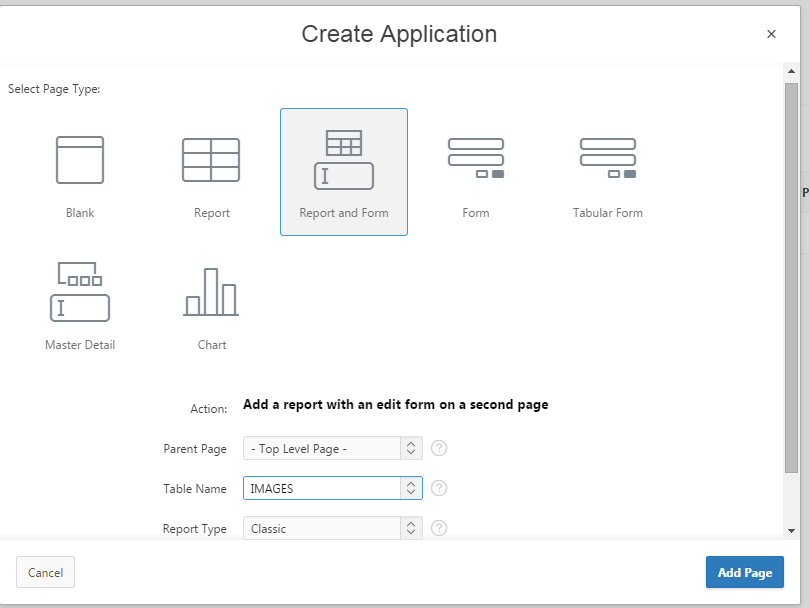


In APEX, create a new Desktop application called **Images**.



Add a Report and Form Page based on your Images table.

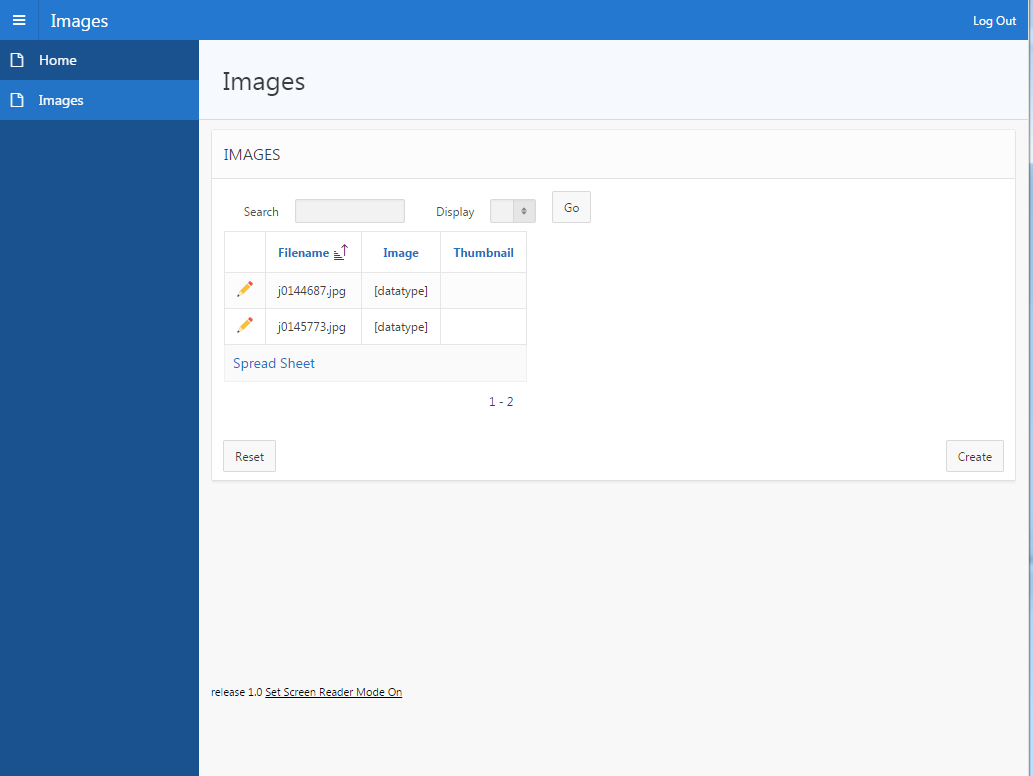
Ensure that you select a **Classic** Report (i.e. not Interactive, which is the default).



As expected, two pages are created.

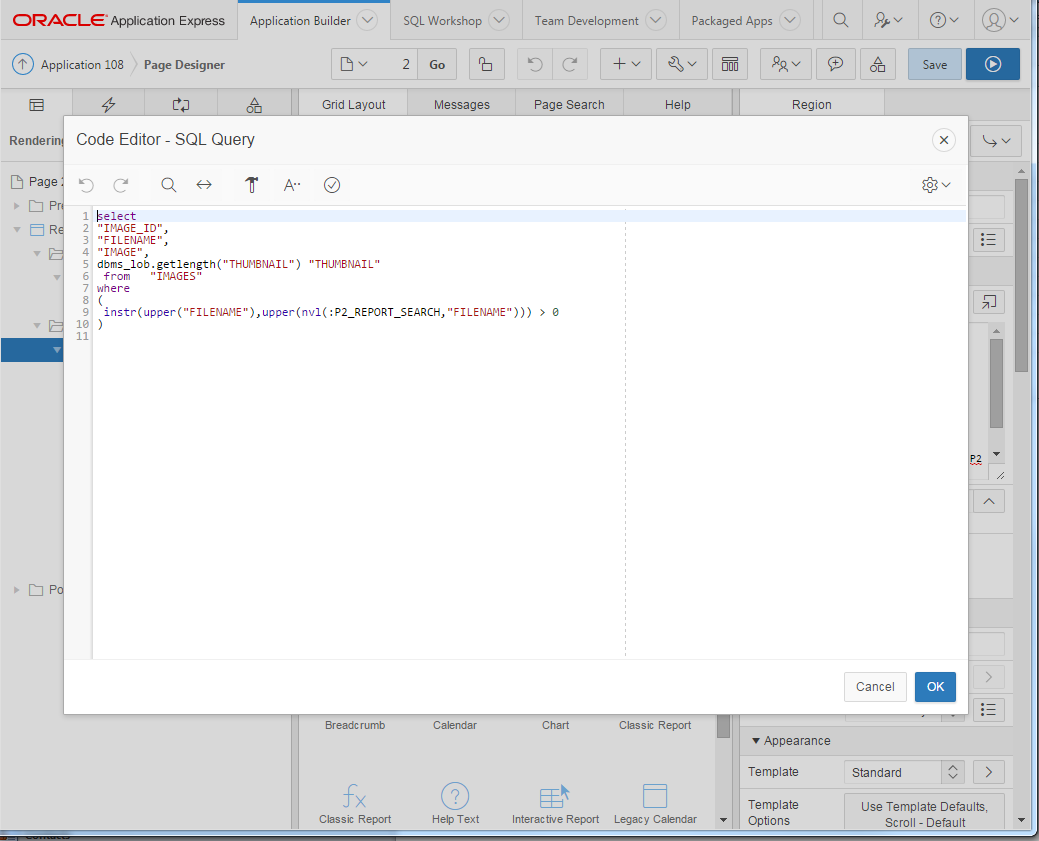
Continue to create the application and then run it.

After the login page, navigate to the Images page. Note there is a [datatype] entry for image but the thumbnail is empty. This is expected as the image column is an object type and the thumbnail has not been created yet.



Edit this page (Page 2?) and open the Images Region

The Source SQL query viewed in the Code Editor is



As the full size image is not needed, change the query to remove the IMAGE column

select

"IMAGE\_ID",

"FILENAME",

dbms\_lob.getlength("THUMBNAIL") "THUMBNAIL"

from "IMAGES"

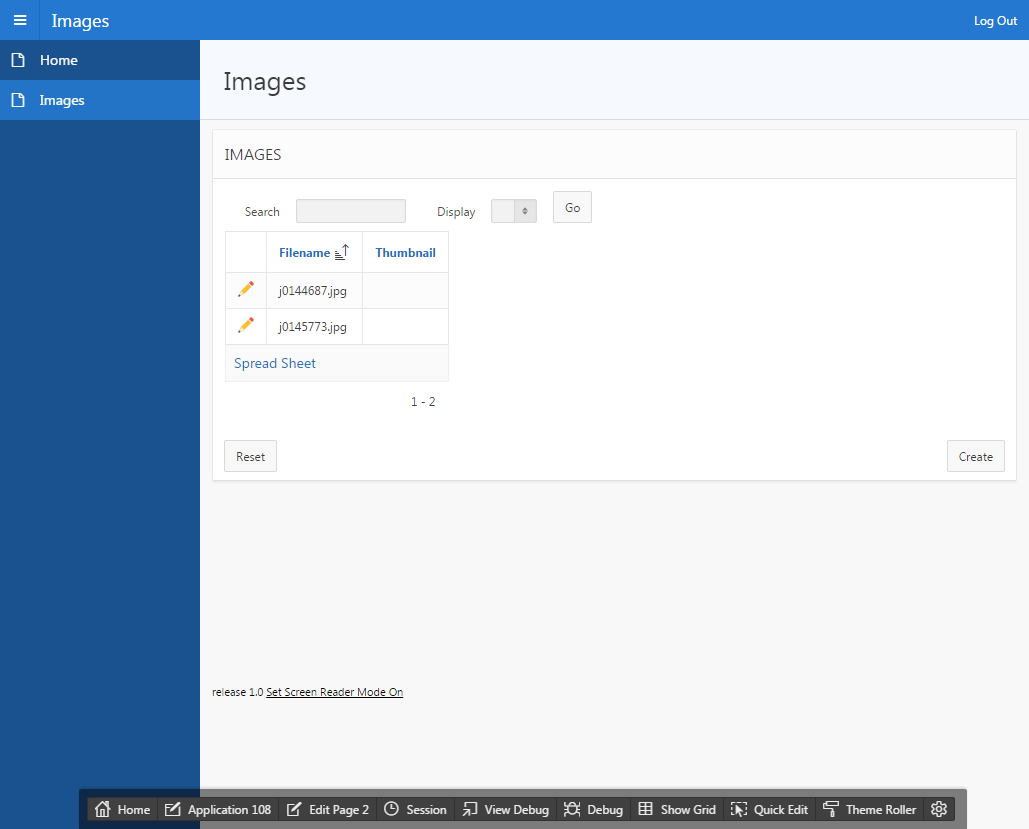
where

(

instr(upper("FILENAME"),upper(nvl(:P1\_REPORT\_SEARCH,"FILENAME"))) > 0

)

and **Save** the changes. Re-run the page to ensure that the image column is not displayed.



In SQL Developer, create a procedure to generate a BLOB thumbnail

CREATE OR REPLACE PROCEDURE create\_blob\_thumbnail (p\_image\_id IN INTEGER) IS

l\_orig ORDSYS.ORDImage;

l\_thumb ORDSYS.ORDImage;

l\_blob\_thumb BLOB;

BEGIN

-- lock row

SELECT image

INTO l\_orig

FROM images

WHERE image\_id = p\_image\_id FOR UPDATE;

l\_thumb := ORDSYS.ORDImage.Init();

dbms\_lob.createTemporary(l\_thumb.source.localData, true);

ORDSYS.ORDImage.processCopy(l\_orig,

'maxscale=128 128',

l\_thumb);

-- extract BLOB from OrdImage

UPDATE images

SET thumbnail = l\_thumb.source.localData

WHERE image\_id = p\_image\_id;

dbms\_lob.freeTemporary(l\_thumb.source.localData);

COMMIT;

END;

Run the procedure for the test images

BEGIN

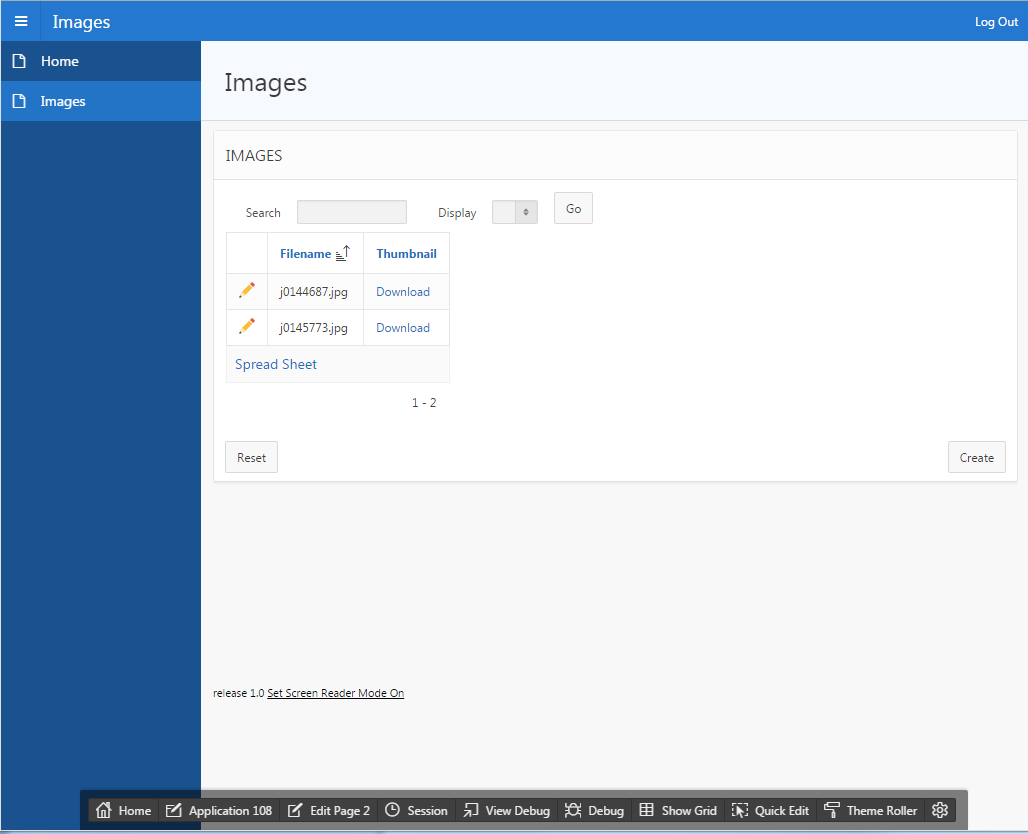
create\_blob\_thumbnail(1);

create\_blob\_thumbnail(2);

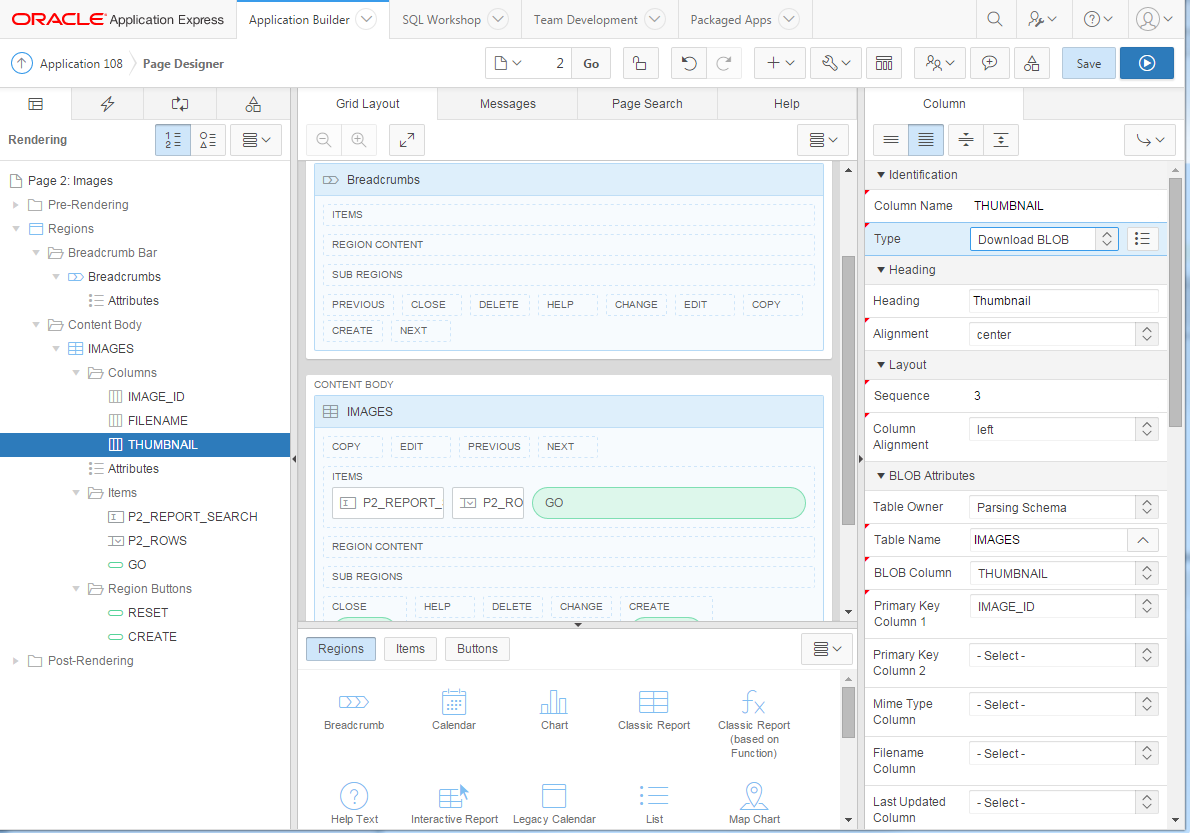
END;

You may need to adjust the parameters to match your image\_ids.

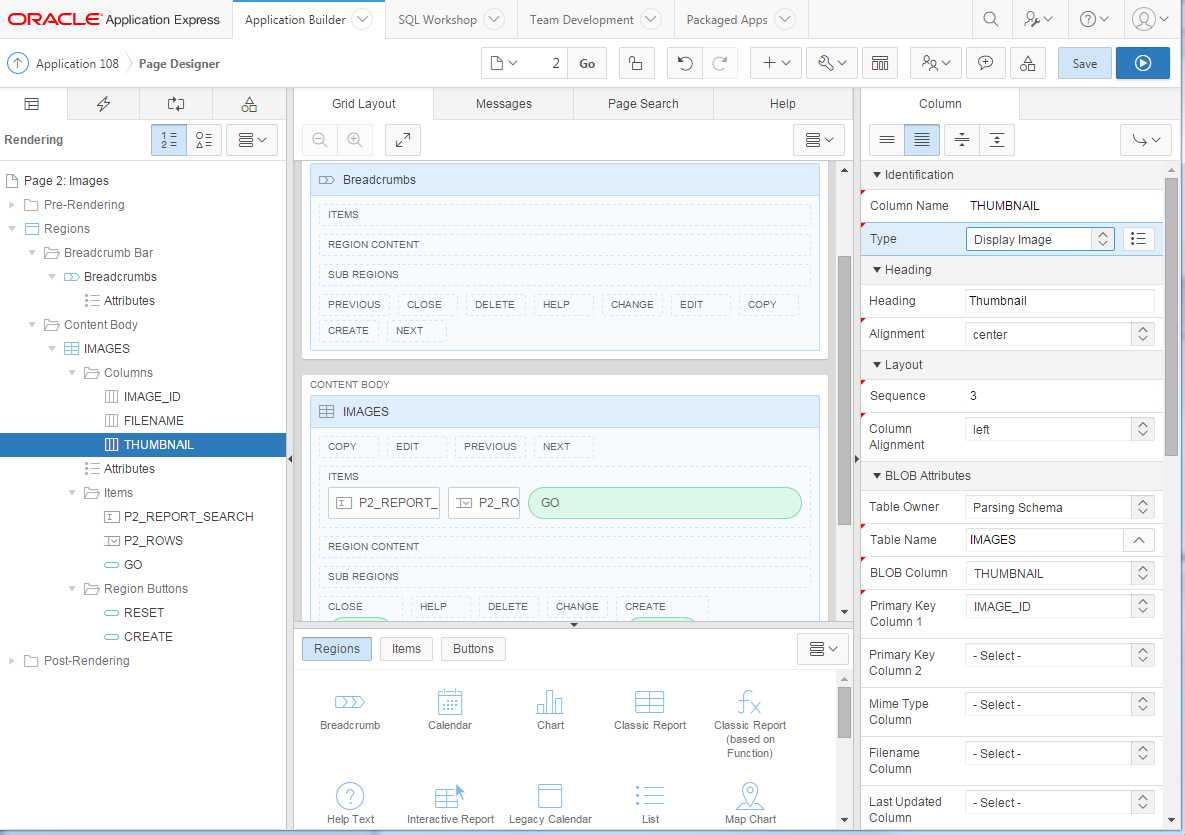
Re-run the application again. A thumbnail can be downloaded and subsequently viewed but there is no thumbnail image being displayed in the application.



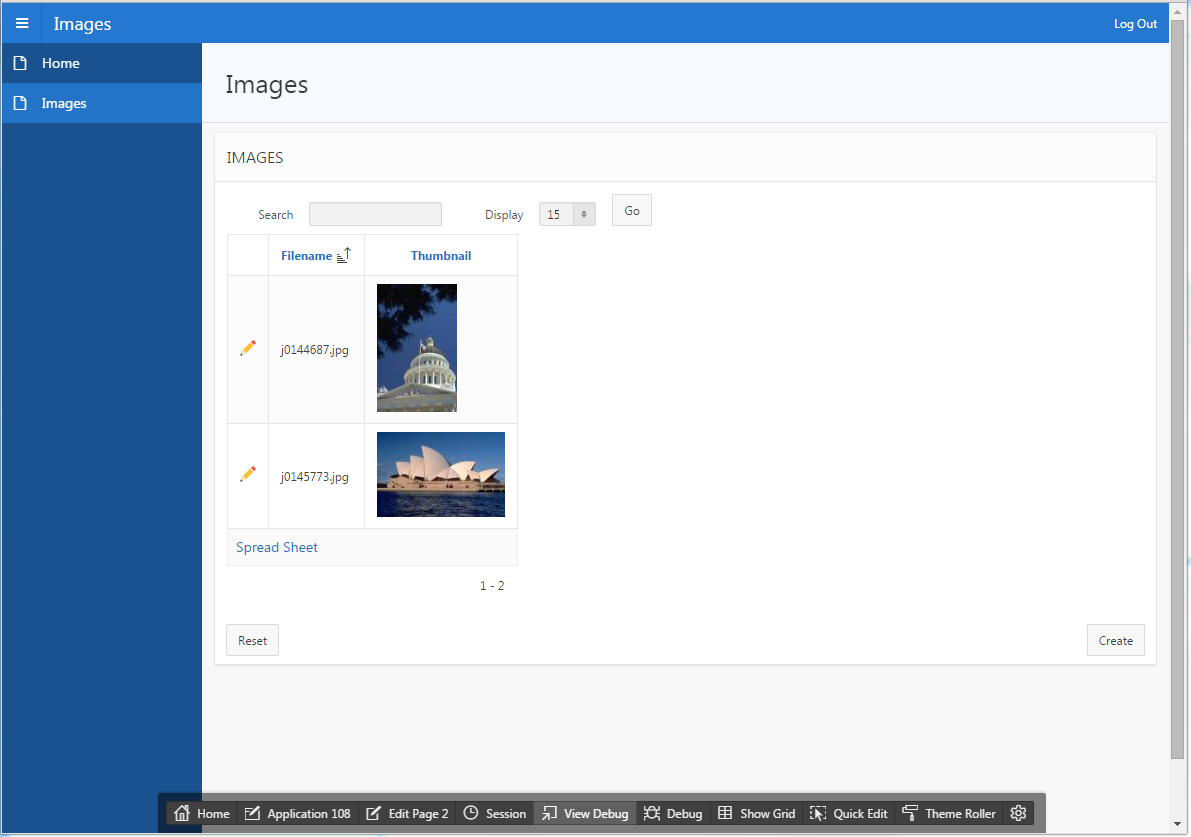
To rectify this, edit the attributes for the thumbnail column



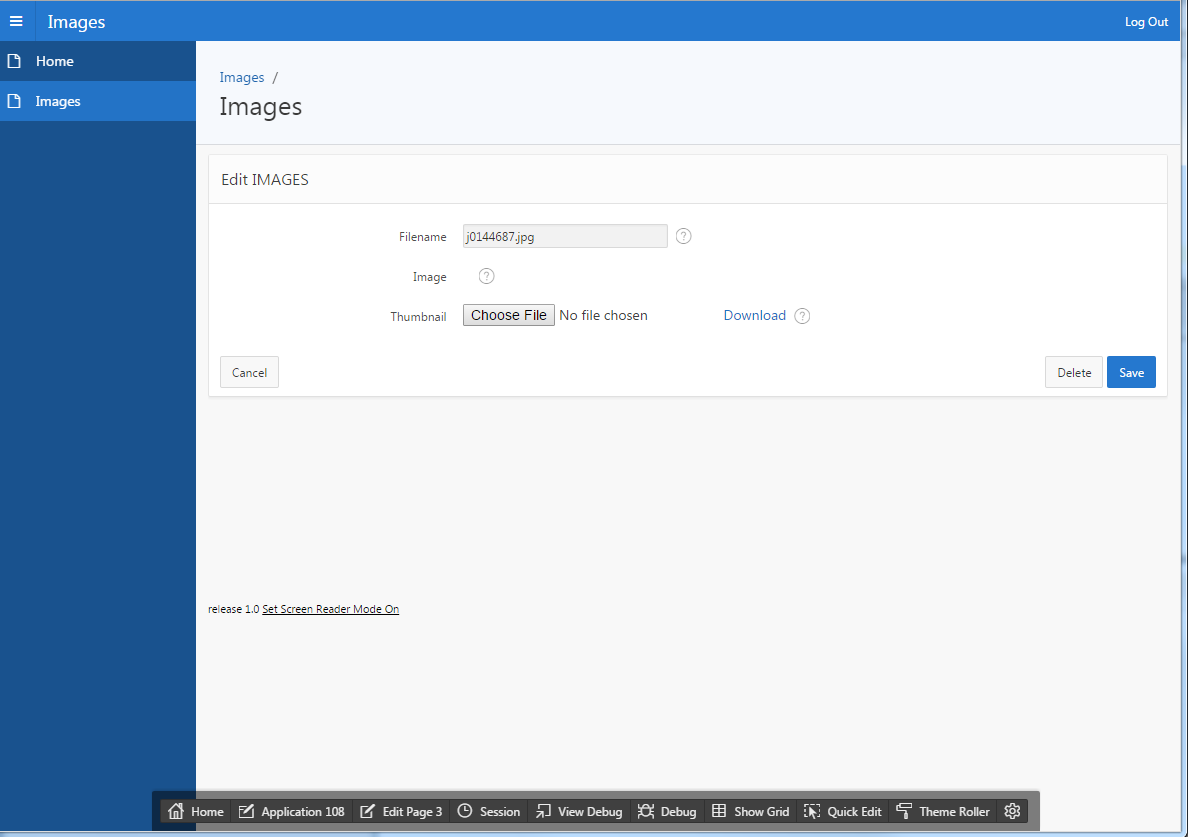
And change the Type from Download BLOB to Display Image



Save the change and re-run the application to view the thumbnails



Click on the pencil icon to navigate to the Form page (Page 3?)



The page is rather sparse so display the full-size image here

Open the P3\_IMAGE item.

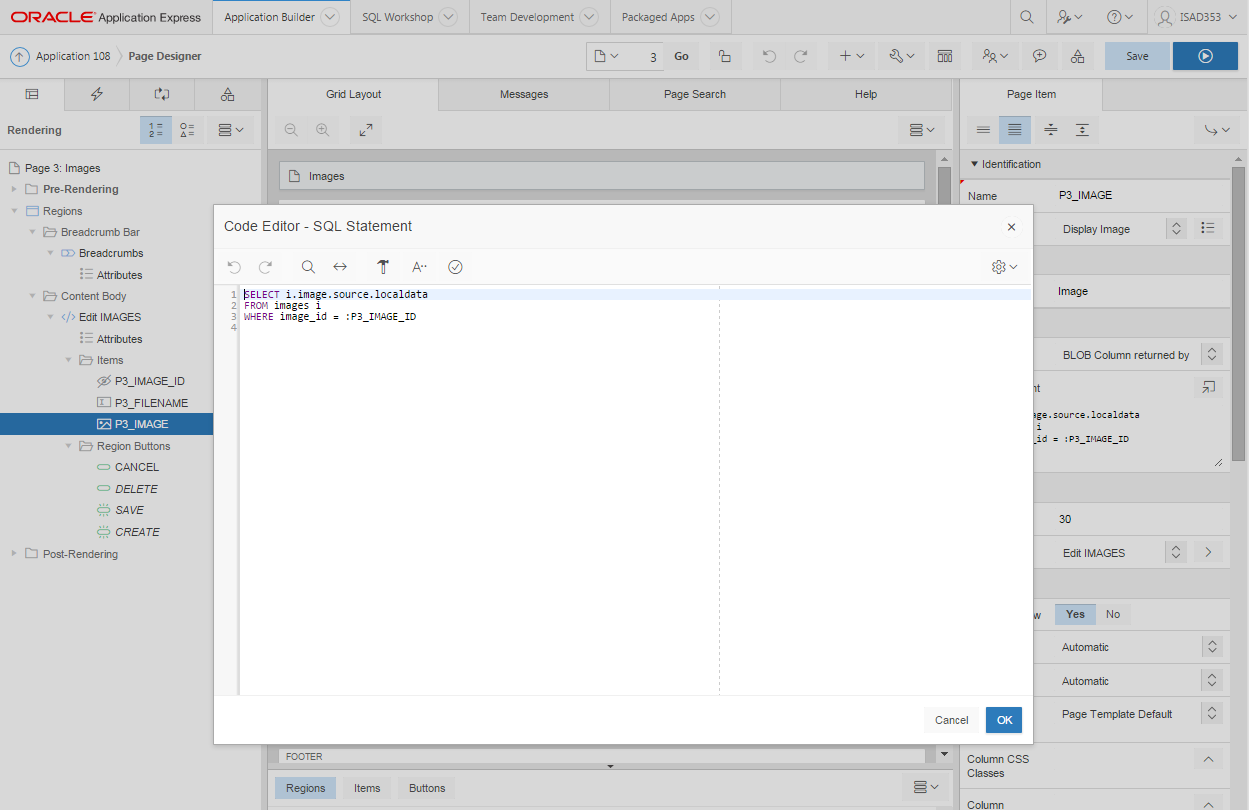
Change the Type entry to **Display Image**

and change the Based On entry to **BLOB Column returned by SQL Statement** and enter[[2]](#footnote-2)

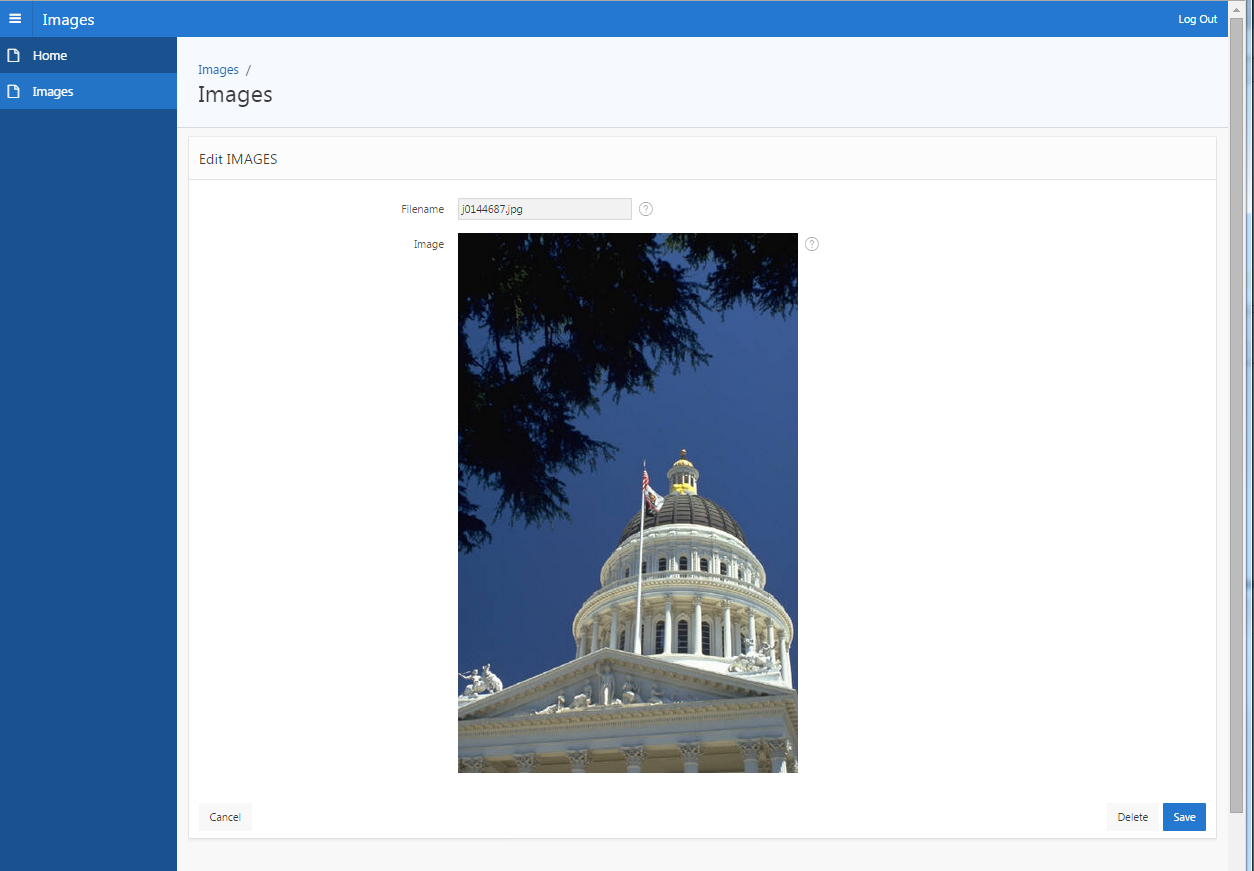
SELECT i.image.getContent()

FROM images i

WHERE i.image\_id = :P3\_image\_id

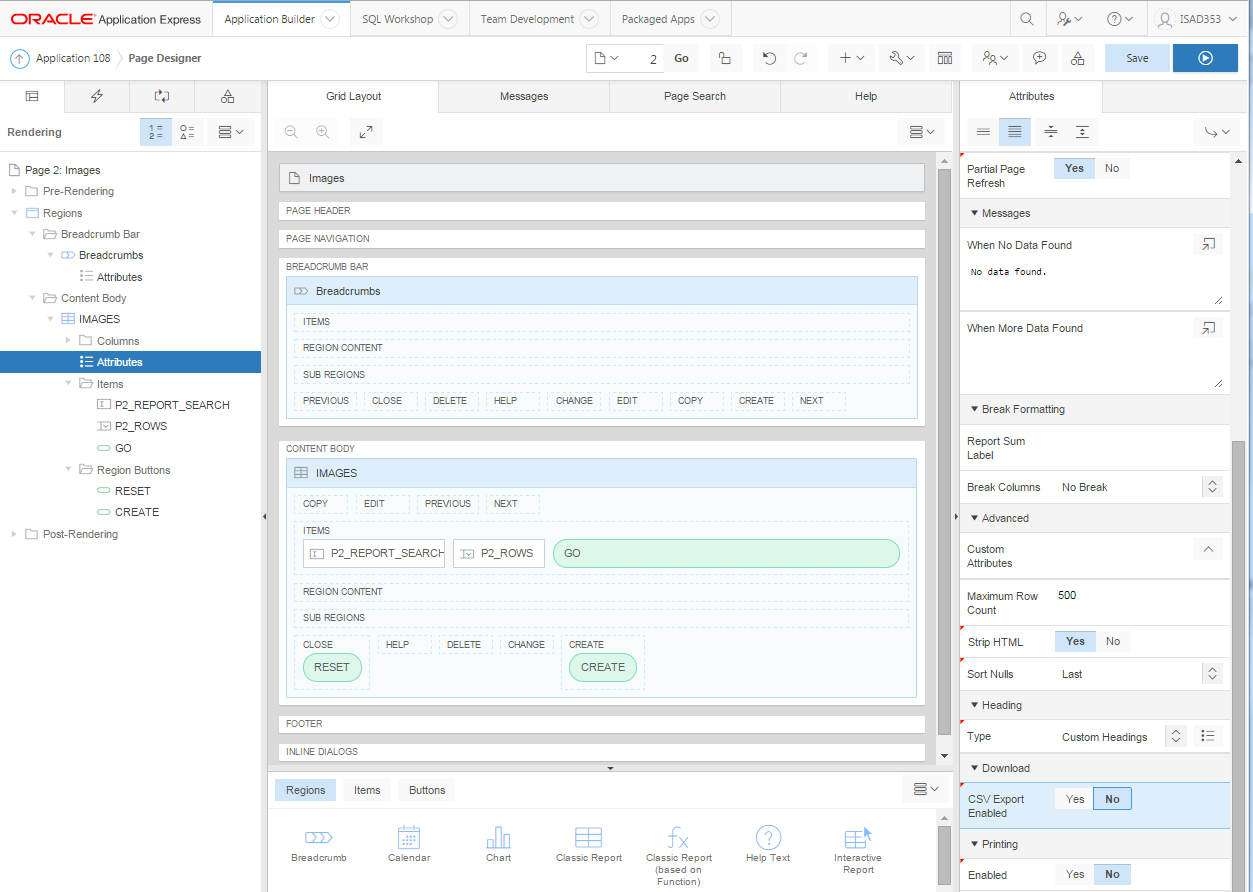


Save the changes and re-run the application – the full size image is displayed



Delete the P3\_THUMBNAIL item to tidy up the display

Finally to remove the Spread Sheet link on Page 2, set CSV Export Enabled to **No**



1. Note that the ISAD353\_IMAGES directory has been created and populated for you as below. You should not execute this code!

   CREATE OR REPLACE DIRECTORY ISAD353\_images AS 'D:\Files\Images\ISAD353\_images';

   GRANT READ ON DIRECTORY ISAD353\_images TO PUBLIC; [↑](#footnote-ref-1)
2. This is equivalent to

   SELECT i.image.source.localdata

   FROM images i

   WHERE image\_id = :P3\_IMAGE\_ID [↑](#footnote-ref-2)