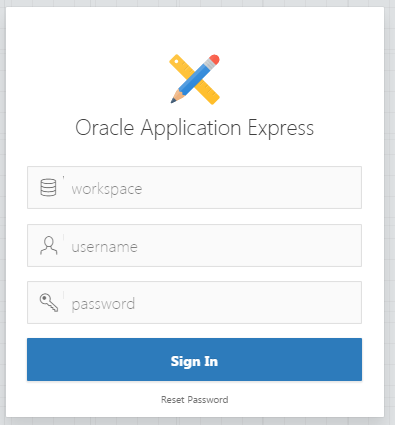
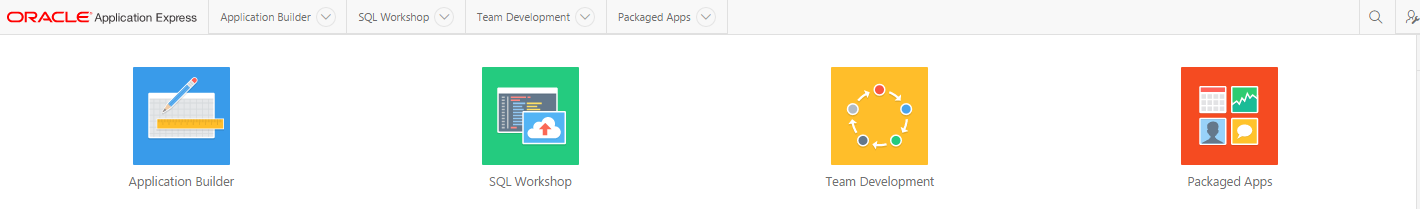
Creating a text search application in APEX 5.0

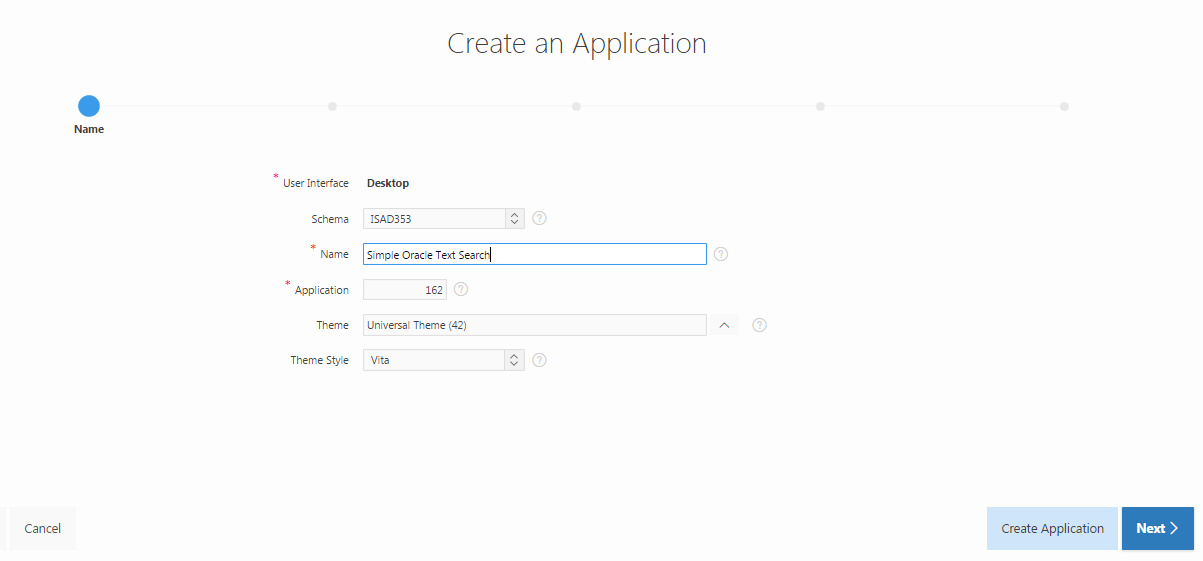
Navigate to [APEX on Larry](http://larry.uopnet.plymouth.ac.uk:8080/apex) and enter your APEX credentials



Select **Application Developer**

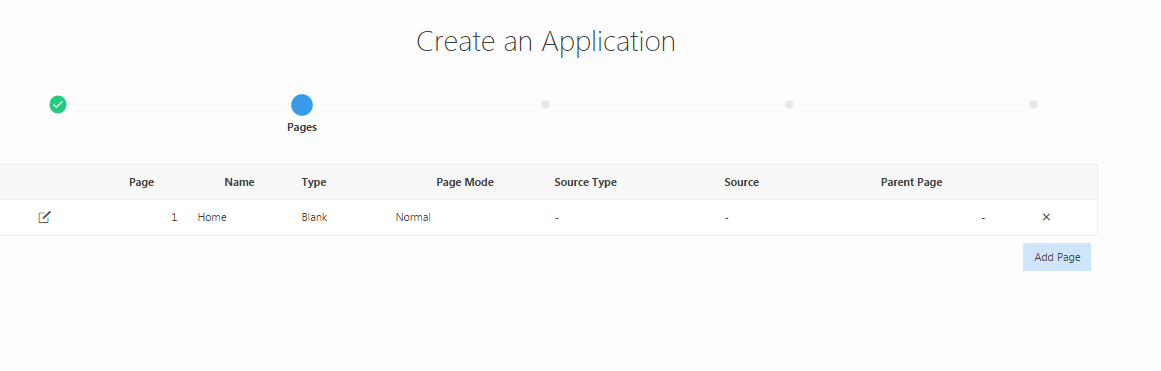


**Create** a new **Desktop** application

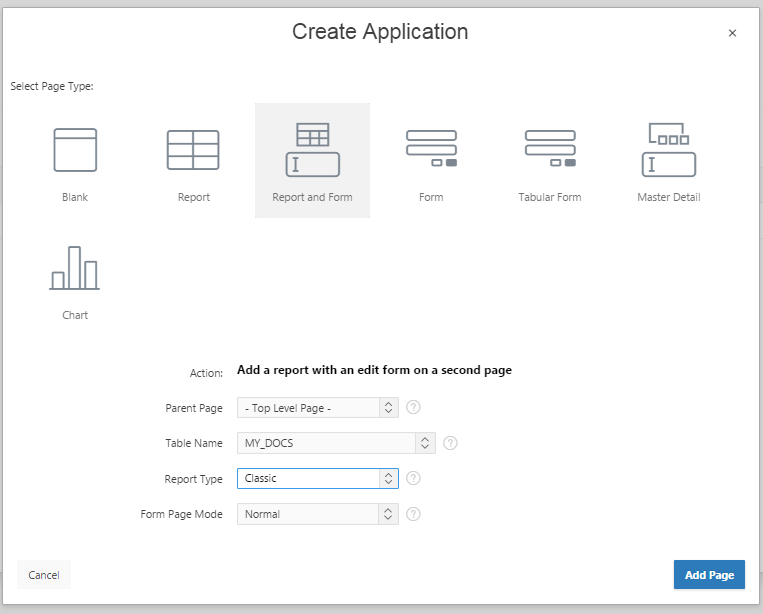


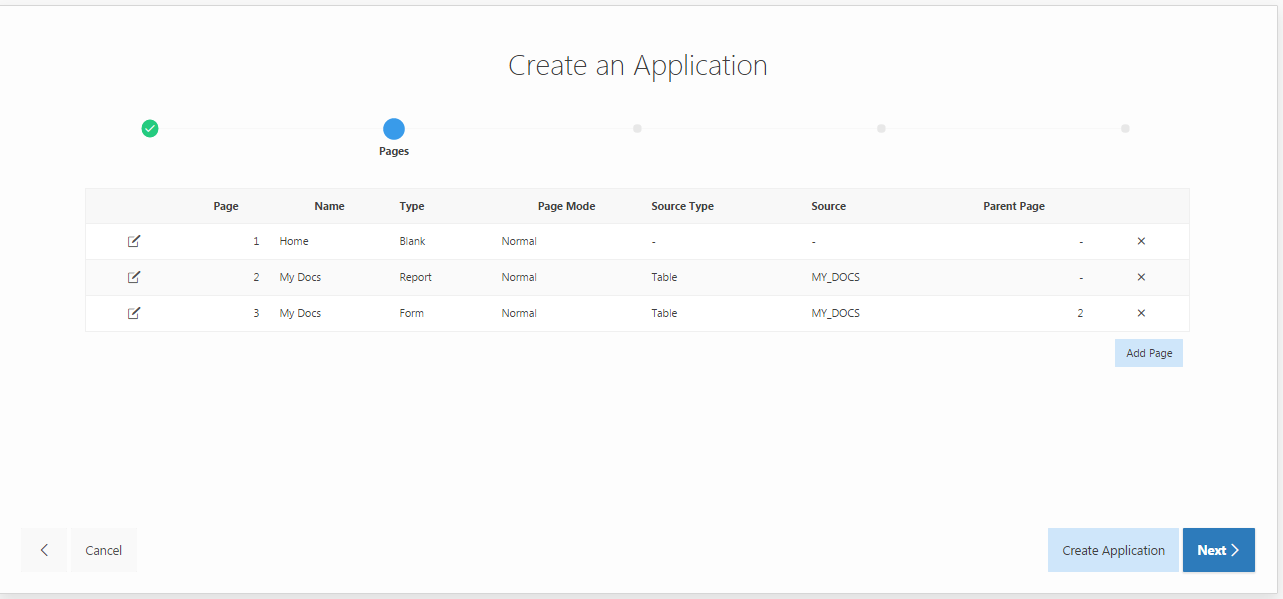
Supply a meaningful **Name** (the Application id is generated for you so leave it alone) and accept the defaults. Click **Next**

A Home Page is generated by default so click **Add Page**

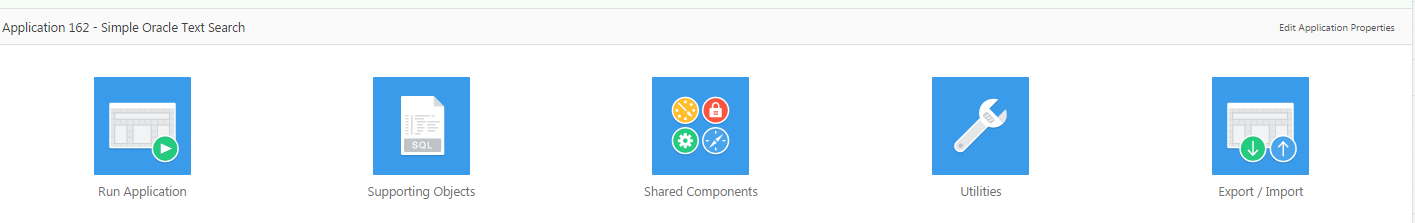


Check **Report and Form**, select **MY\_DOCS** from the drop-down menu for Table Name and select a **Classic** Report Type. Click **Add Page**.

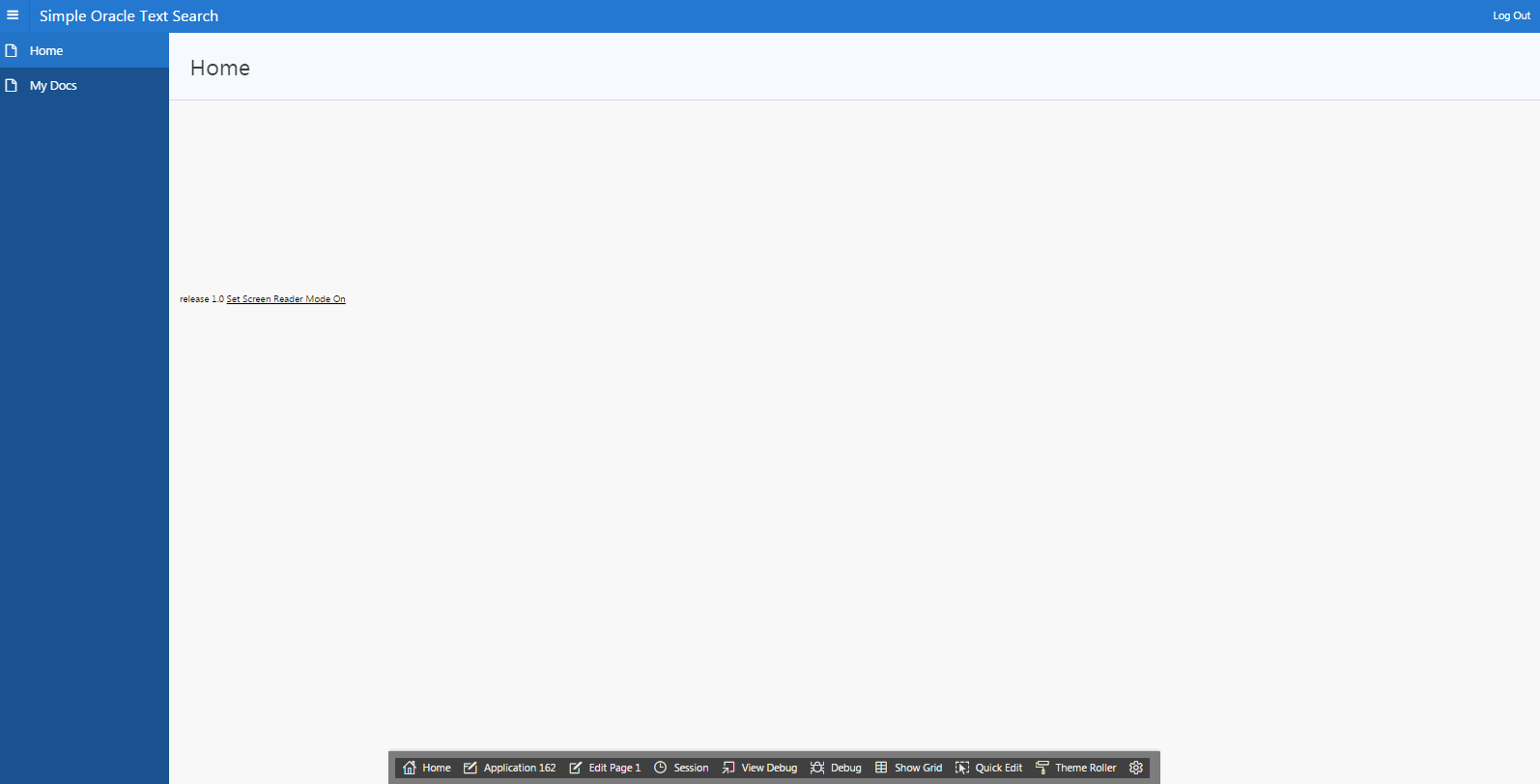


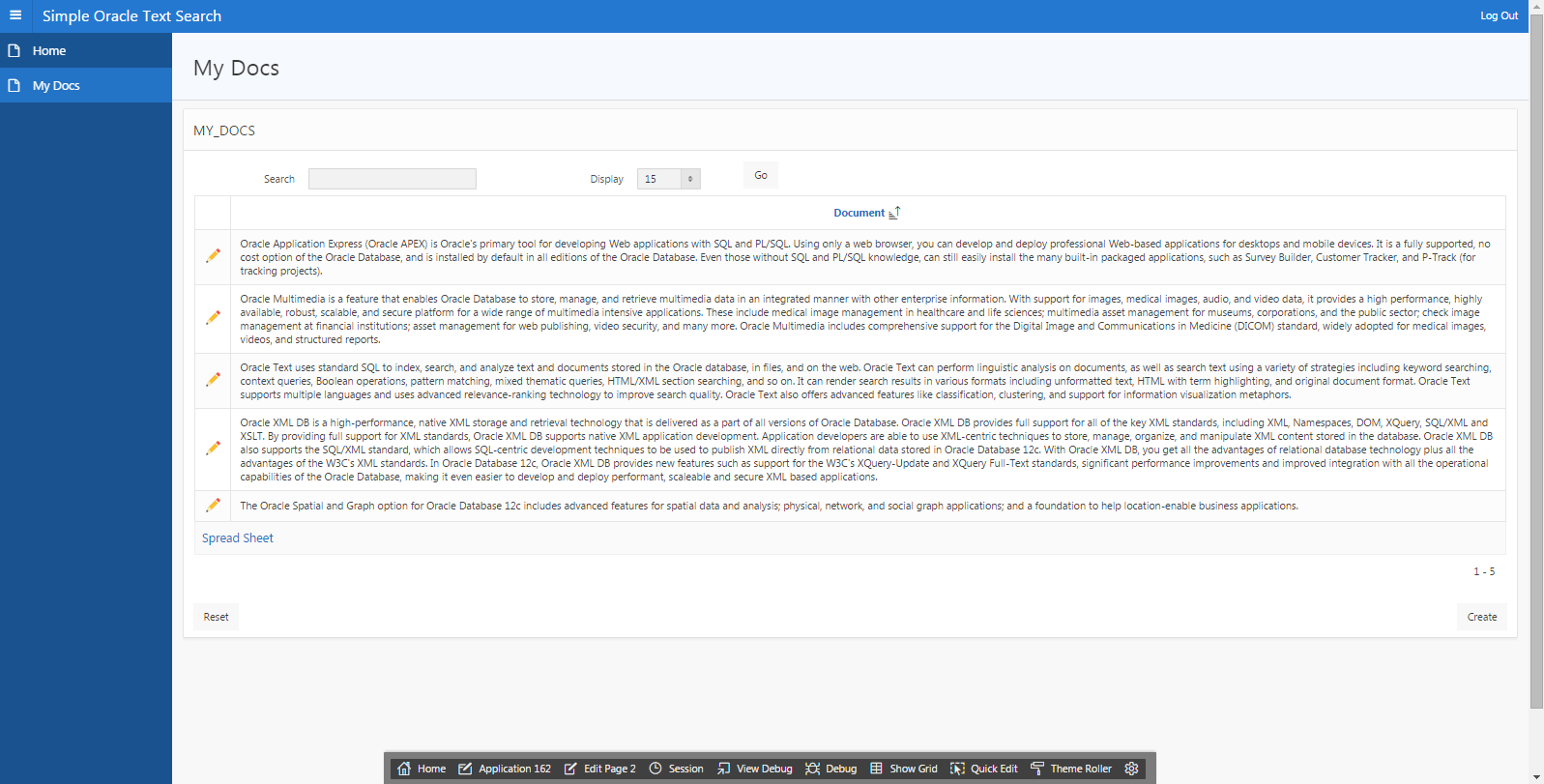


Two additional pages have been created (one report and one form). Click **Create Application** and then **Create Application** to confirm your selections.

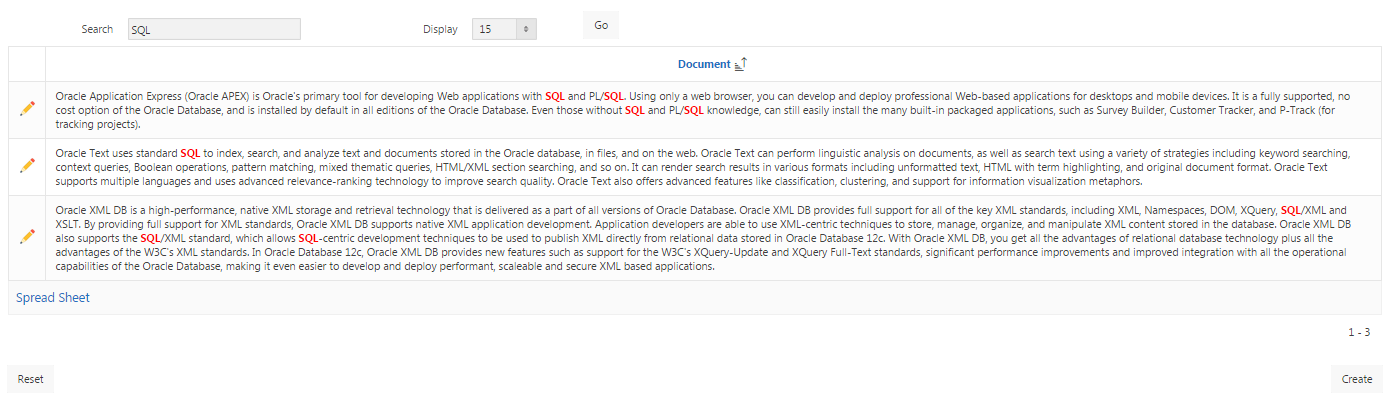


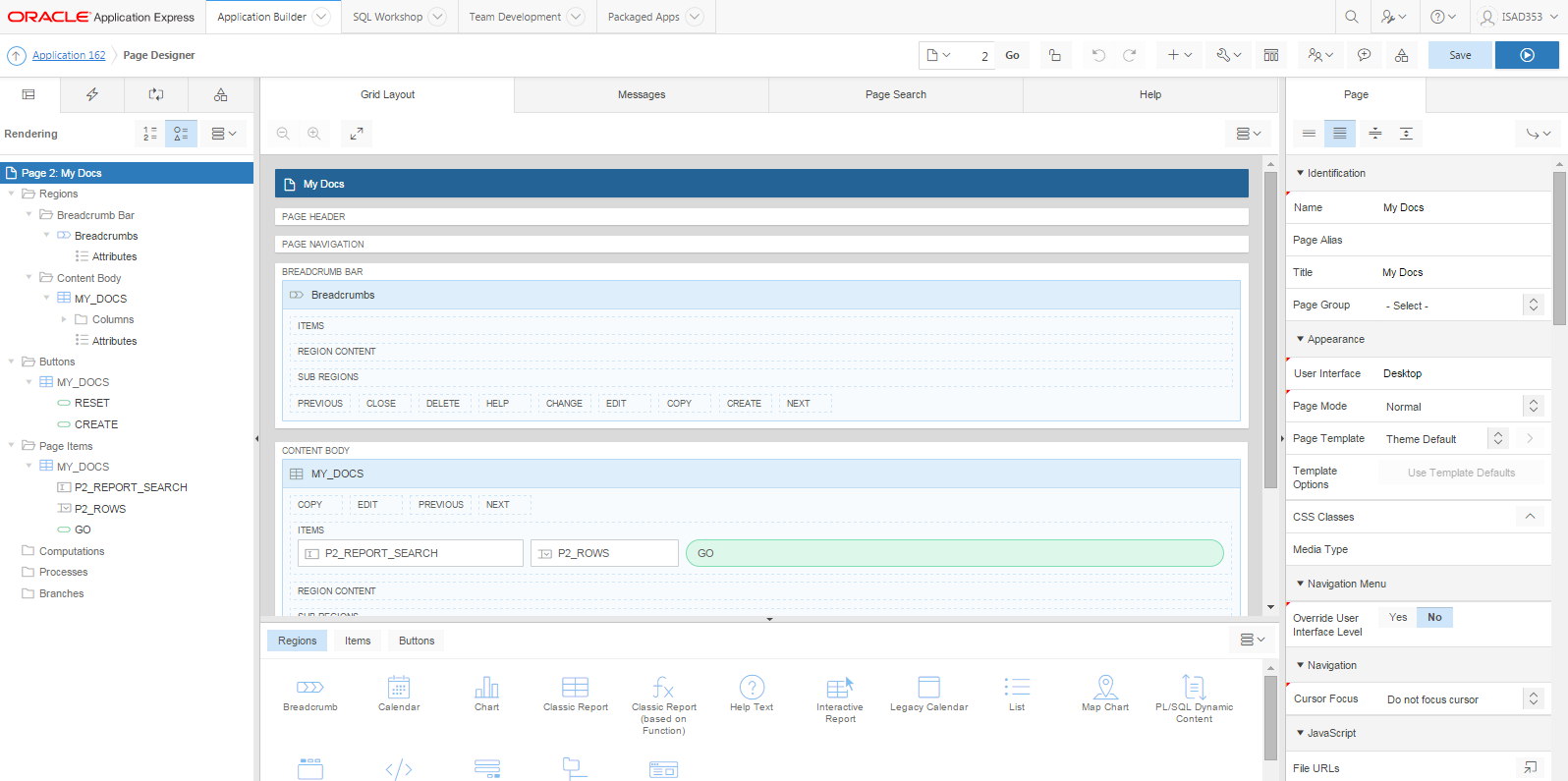
Click **Run Application** and login using your APEX username and password. You’ll see the default Home page. Note the Developer Toolbar at the bottom of the window – more later.

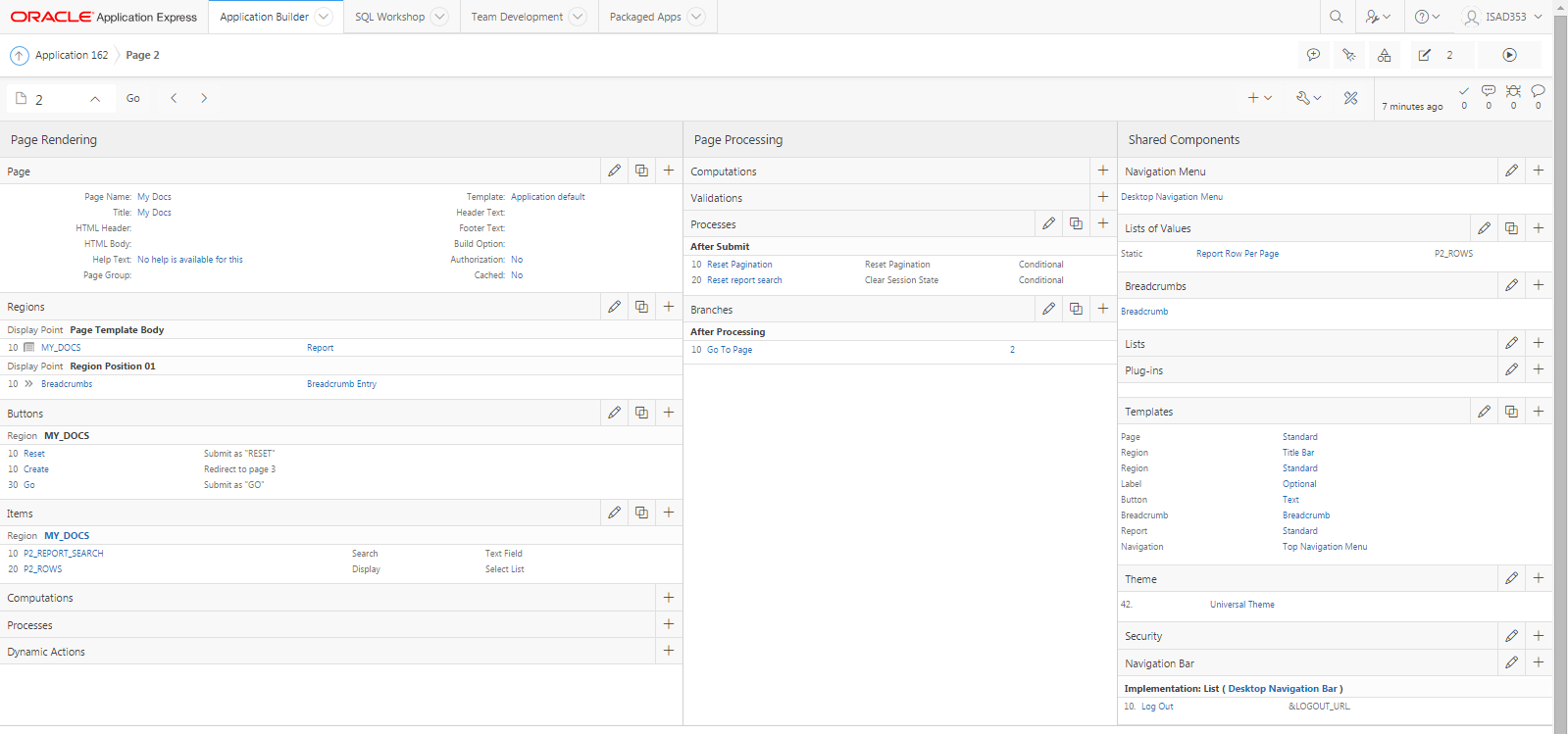


Go to the My Docs page using the Navigation List entry 

Enter a search string in the Search box and click **Go**. Note that the matches are highlighted in red.

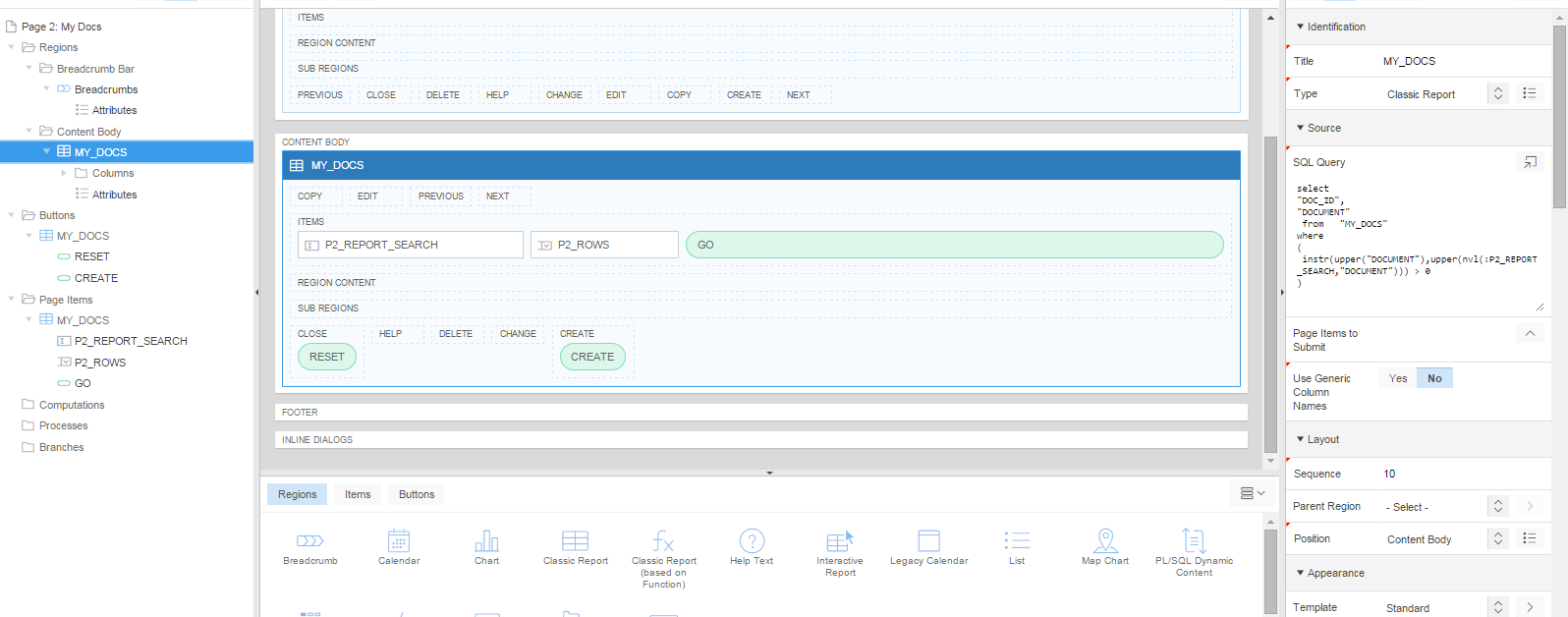


A report region is based on an SQL query. To view the query for this region, click **Edit Page 2** in the Developer Toolbar at the bottom of the window. You will either get a Page Designer view (new in APEX 5.0)

or a Component View It is down to personal preference which you use as both have (more or less) the same content but the display and navigation differ. You can toggle between the two by selecting or .

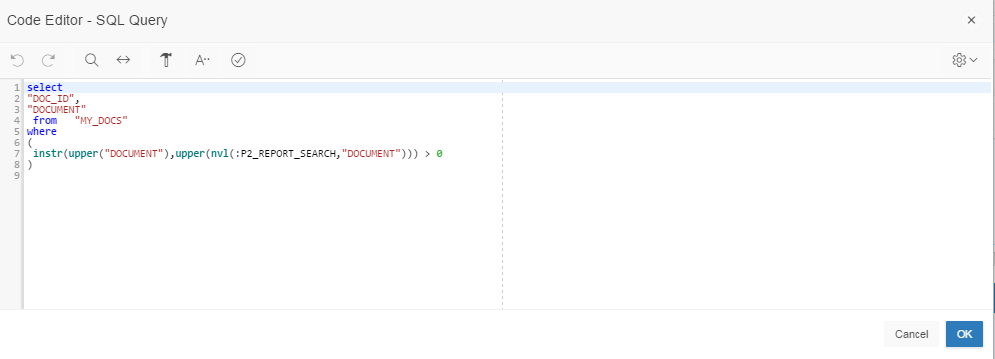
APEX generates *pages* made up of *regions* which contain *items* and *buttons*.

Explore the MY\_DOCS region by clicking the icon in the Content Body.



The important content is the SQL Query for the Source and in particular the WHERE clause.

Use the Code Editor to view the query more clearly



* :P2\_REPORT\_SEARCH refers to the search box item on the page and uses bind variable syntax with a leading colon. This helps to foil SQL injection attacks in APEX.
* INSTR returns the position of a string in a column so any match in a column will deliver a value greater than zero.
* UPPER converts a string or column to upper case so the search is case insensitive.
* NVL returns the second argument if the first argument is null so if the search box is null, all rows are returned.

So in essence, all this is achieving is a simple character match and not taking advantage of the Oracle Text CONTEXT index on the document column.

As a first step to remedying this, change the Source to (copy and paste from this document):

select

"DOC\_ID",

"DOCUMENT"

from "MY\_DOCS"

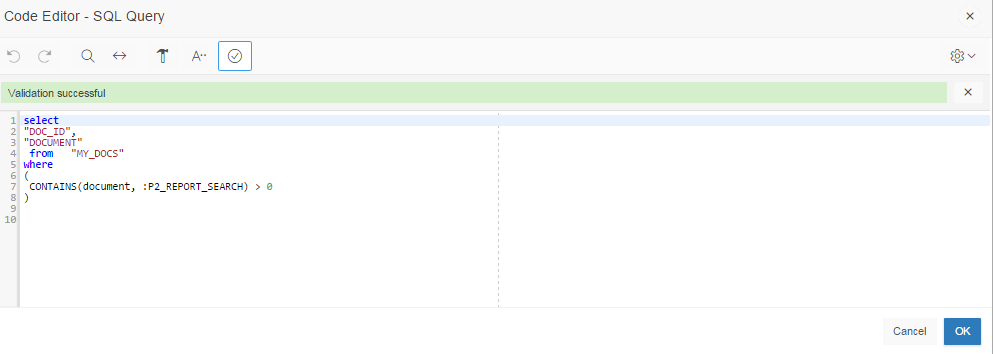
where

(

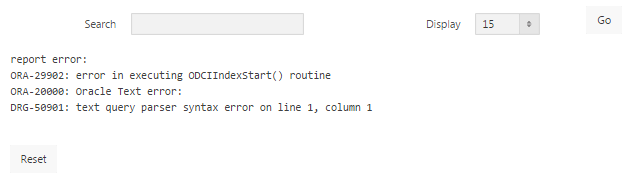
CONTAINS(document, :P2\_REPORT\_SEARCH) > 0

)

And Validate the query



Click OK, **Save** and then  to Run. Notice that there is an error message



but you can ignore this for the moment.

Try searches for [sql], [$metaphor] and [fuzzy(orcale)] to test case insensitivity, stemming and fuzzy matching respectively. Notice that matching terms are only highlighted when there is an exact match – don’t worry about this. Notice also that the error disappears.

Click **Reset** and the error returns. The problem is that the CONTAINS operator cannot handle a null search string. You could use the NVL function with a string that is unlikely to appear in some text, e.g.

NVL(:P2\_REPORT\_SEARCH, 'w21fevhds77rt3e15656999mnbuiwdgtyvgf')

but if you want to mimic what an APEX report normally produces with a null search, i.e. the complete contents without filtering rather than no results at all, the code for the report needs changing to handle a null search string.

To do this you need to use some PL/SQL to dynamically construct a query. As a first step,

change the **Type** to **Classic Report (based on Function)** and the **Source** to

DECLARE

l\_query VARCHAR2(4000);

BEGIN

l\_query:=

'select

"DOC\_ID",

"DOCUMENT"

from "MY\_DOCS"

where

(

CONTAINS(document, :P2\_REPORT\_SEARCH) > 0

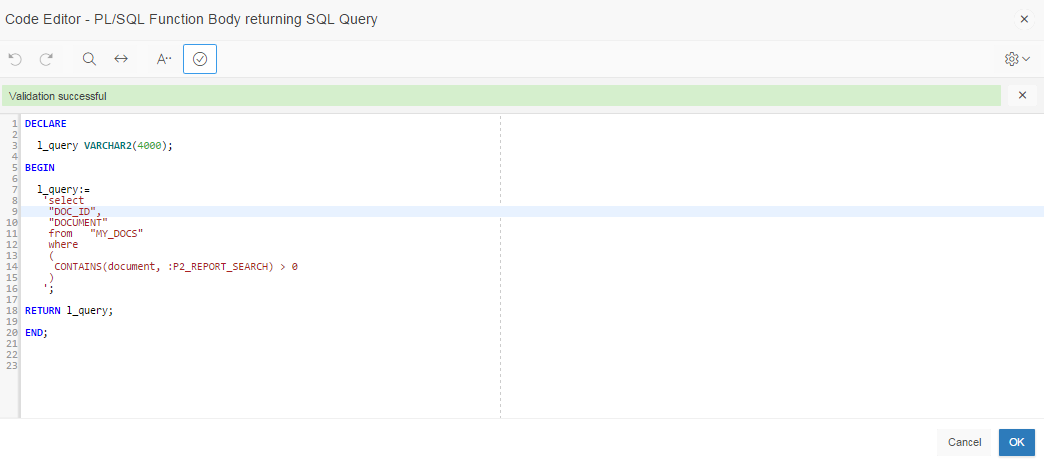
)

';

RETURN l\_query;

END;

Note that the Code Editor gives a more meaningful description of the report source - PL/SQL Function Body returning SQL Query



Click OK, Save and then run. Notice that the error message is still there.

By using this powerful feature you have been able to use PL/SQL to dynamically generate the same SQL query, store it as a local variable and then return it to be run. You can now go on to add some logic to handle the null search string and change the query returned dependent on the presence of a null value.

DECLARE

l\_query VARCHAR2(4000);

BEGIN

l\_query:=

'select

"DOC\_ID",

"DOCUMENT"

from "MY\_DOCS" ';

IF v('P2\_REPORT\_SEARCH') IS NOT NULL THEN

l\_query := l\_query||' '||'

where

(

CONTAINS(document, :P2\_REPORT\_SEARCH) > 0

)

';

END IF;

RETURN l\_query;

END;

N.B. || is the concatenation operator in Oracle SQL.



Note the use of the v('P2\_REPORT\_SEARCH') syntax which is the preferred syntax for PL/SQL. Sometimes the colon bind syntax, e.g., :P2\_REPORT\_SEARCH, will also work or you may get an *ORA-01008: not all variables bound* error.

Click **Save** and then run. Notice that the error message has gone and that all records are shown in the report. Try searching to get some results.

For testing and debugging, it is useful to be able to see what query you have generated. To do this, add the following just before the RETURN statement:

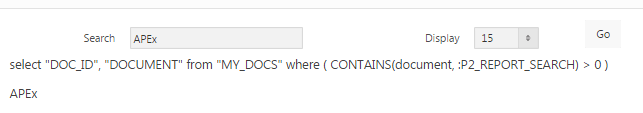
-- test and debug only

htp.p( l\_query );

htp.p( '<br>' );

htp.p( v('P2\_REPORT\_SEARCH') );

Your query code will be displayed in the region.





To provide automatic stemming, rather than require the user to use the $ operator, try

DECLARE

l\_query VARCHAR2(4000);

BEGIN

l\_query:=

'select

"DOC\_ID",

"DOCUMENT"

from "MY\_DOCS" ';

IF v('P2\_REPORT\_SEARCH') IS NOT NULL THEN

l\_query := l\_query||' '||'

where

(

CONTAINS(document, ''$'|| v('P2\_REPORT\_SEARCH') ||''') > 0

)

';

END IF;

RETURN l\_query;

END;

Note the need to switch to using the v('…') syntax.

As an example the following is generated:

select "DOC\_ID", "DOCUMENT"

from "MY\_DOCS"

where ( CONTAINS(document, '$develop') > 0 )

You could even copy and paste this into SQL Developer and run it!

Note also the need to “double up” single quotes in a quoted string. This can be confusing!

As an example, the following would generate a syntax error as the string ends with the quote in Joe’s

l\_string := 'Joe's happy that he's going to his twin sisters' party';

So the quotes need to be repeated:

l\_string := 'Joe''s happy that he''s going to his twin sisters'' party';

The alternative to repeated quotes is to use Oracle q syntax with any bracket symbol that will not be used in the string, {} here, e.g.

l\_string := q'{Joe's happy that he's going to his twin sisters' party}';