IR Vertical Headers Plug-in

I. Preface

The plug-in allows you to position IR headers vertically.
This is free cross browser and easy in adjustment extension to Interactive Report.

See plugin page for examples:

http://apex-plugin.com/oracle-apex-plugins/dynamic-action-plugin/ir-vertical-headers_433.html

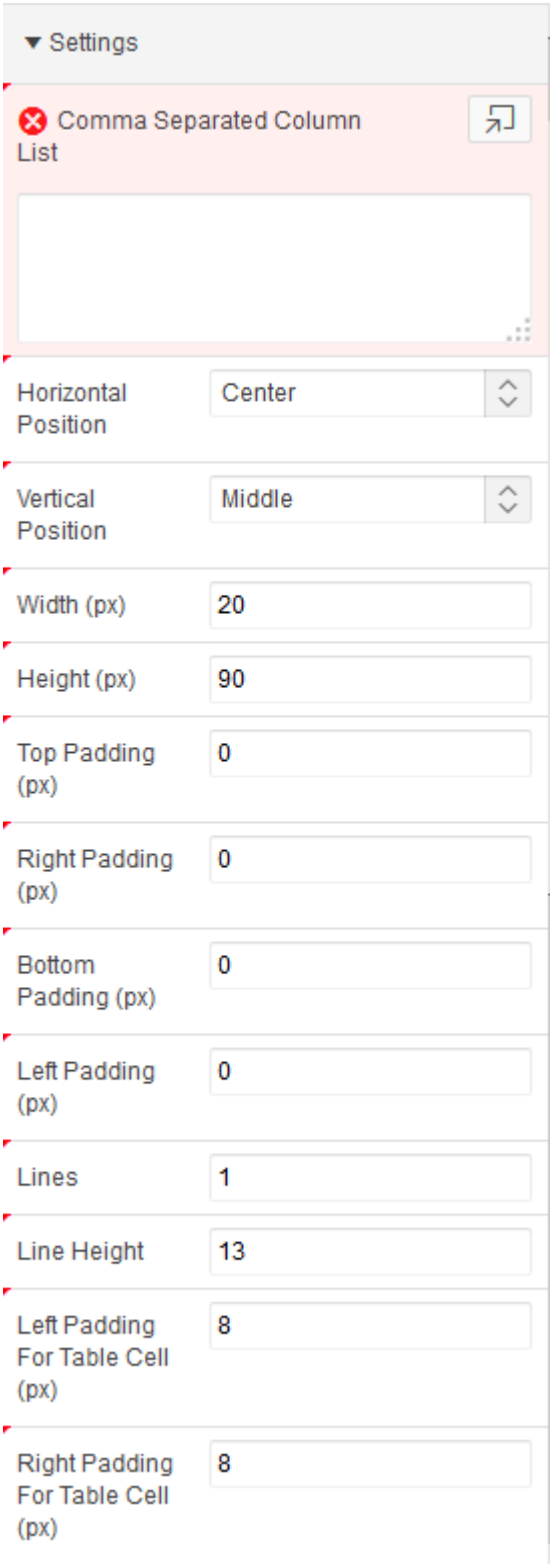
II. Installation

a. Select the application where you want to import the plug-in.
b. Go on page **Shared Components** > **Plug-Ins** then click **Import** button.
c. Select sql file located in archive then select **File Type**: **Plug-in**, click **Next**.
d. On final page click **Install Plug-in**.

III. How to use

Go to the page where you want to use the plug-in, create new dynamic action:

a. Enter any dynamic action name.
b. Enter Event: **Page Load**
c. Enter (True) Action: **IR Vertical Headers [Plug-In]**
d. Enter Affected Elements
   Selection Type: **Region**
   Region: Select your Interactive Report.
e. Enter plugin settings:



Comma Separated Column List
The List of the headers you want to be vertical
Horizontal Position,
Vertical Position
control the position and alignment of the text
Width (px),
Height (px)
The width and the height for the vertical headers. The value should be adjusted to fit the text and should be larger than the width of cells in each row.
Top Padding (px),
Right Padding (px),
Bottom Padding (px),
Left Padding (px)
The Padding for vertical headers.
Default: 0 (px)
Lines:
The Number of Lines in Vertical Headers.
You can use
 tag to make several lines.
Line Height (px)
This attribute is used when computing horizontal position
Left Padding for Table Cell (px),
Right Padding for Table Cell (px)
This is padding for Table cells that are under Vertical Headers. Here you can reduce the summary width for these cells.

f. Save Changes.

Also see help for more information about these attributes.
You can add multiple dynamic actions on one page.

IV. Using dynamic SQL for vertical columns

This technique is commonly used if the report has dynamic number of columns and this number depends on data stored in oracle tables.

In this case report data is commonly stored in temporary apex collection that is used as the source for IR report and the column header names are stored in page items.

Step 1. Creating page items

Determine the maximum number of dynamic columns in your report and then create hidden page item for each dynamic column, e.g.:

```
P5_HEADER_NAME_1 .. P5_HEADER_NAME_50
```

Limitations:

The maximum number of items on page is 200.
The maximum number for string columns in apex collection is 50.

So 50 is an optimal choice. By using Page Designer view or Create Multiple Items option in component view you can create 50 items quickly.

If you need more than 50 dynamic columns then you probably need to create a table to store the data later on instead of using apex collection or use pipelined function.

Step 2. Creating before header process

Create before header process where select dynamic header names into the page items you created and then execute your dynamic sql and put the report data to apex collection.

E.g. we have a task to display information about first 5 tables from *all_tables* oracle dictionary in *sys* scheme. There are many way to do this, below is the example of code:

```
select max(decode(T.rn,1,T.table_name,null)) HEADER_NAME_1
     , max(decode(T.rn,2,T.table_name,null)) HEADER_NAME_2
     , max(decode(T.rn,3,T.table_name,null)) HEADER_NAME_3
     , max(decode(T.rn,4,T.table_name,null)) HEADER_NAME_4
     , max(decode(T.rn,5,T.table_name,null)) HEADER_NAME_5
 into :P5_HEADER_NAME_1
    , :P5_HEADER_NAME_2
    , :P5_HEADER_NAME_3
    , :P5_HEADER_NAME_4
    , :P5_HEADER_NAME_5
from (
  select t.table_name
       , row_number() over (order by t.table_name) rn
  from all_tables t
  where t.owner = 'SYS'
) T;

if apex_collection.collection_exists( p_collection_name => 'IRVH' ) then

  apex_collection.delete_collection( p_collection_name => 'IRVH' );
end if;

apex_collection.create_collection_from_query( p_collection_name => 'IRVH'
                                            , p_query => '
select T.cat Category
     , max(decode(T.rn,1,''x'',null)) HEADER_NAME_1
     , max(decode(T.rn,2,''x'',null)) HEADER_NAME_2
     , max(decode(T.rn,3,''x'',null)) HEADER_NAME_3
     , max(decode(T.rn,4,''x'',null)) HEADER_NAME_4
     , max(decode(T.rn,5,''x'',null)) HEADER_NAME_5
from (
  select t.table_name
       , t.num_rows

       , case when nvl(t.num_rows,0) = 0 then ''Empty''
              when nvl(t.num_rows,0) between 1 and 10 then ''1-10 Rows''
              when nvl(t.num_rows,0) between 11 and 30 then ''11-30 Rows''
              when nvl(t.num_rows,0) between 31 and 50 then ''31-50 Rows''
              when nvl(t.num_rows,0) between 51 and 100 then ''51-100 Rows''
              when nvl(t.num_rows,0) between 101 and 300 then ''101-300 Rows''
              when nvl(t.num_rows,0) between 301 and 1000 then ''301-1000 Rows''
              when nvl(t.num_rows,0) > 1001 then ''1001 Rows or More''
         end cat

       , row_number() over (order by t.table_name) rn
  from all_tables t
  where t.owner = ''SYS''
) T
group by T.cat
order by max(t.num_rows)');
```

Step 3. Creating IR report

Create IR report, use *apex_collections* view as source, e.g.:

```
select t.c001
     , t.c002
     , t.c003
     , t.c004
     , t.c005
     , t.c006
from apex_collections t
where t.collection_name = 'IRVH'
order by t.seq_id
```

Then go to report attributes and fill **Heading** for each column, e.g.:

```
Category /<br>Table Name
&P5_HEADER_NAME_1.
…
&P5_HEADER_NAME_5.
```

Then go to column attributes of each dynamic column (columns 2-6 in this example) and place pl/sql condition:

```
:P5_HEADER_NAME_1 is not null
…
:P5_HEADER_NAME_5 is not null
```

You can do this quickly using page designer view in apex 5.0.

The last step is to create dynamic action plug-in action and enter comma separated column alias list: `C002,C003,C004,C005,C006`

V. Known issues and limitations

- Plug-in can only make vertical headers rotated at 270 degree. Vertical headers rotated at 90 degree are not supported.

VI. Notes

You can send feedbacks and bug reports to my email address: svdev74@gmail.com
For bug reports, please include all kind of technical information: full apex version, browser version, application theme name, plugin options, detailed description of the problem, screenshots or examples on apex.oracle.com.

Note: English is not my native language so if you see errors in English text somewhere you can help me to fix them just send me the correct version.