Graphics editor online. Using the potential of Oracle InterMedia

Andrzej Barczak, Dariusz Zacharczuk
Institute of Computer Science, University of Podlasie,
St. 3 Maja 54, 08-110 Siedlce, Poland

Abstract: Oracle database system is a powerful tool. Article shows how to use some of its capabilities for processing multimedia data. We will show how to build web pages, how to use CSS and consequently step by step will create online image editor.

Key words: InterMedia, Oracle, graphics editor, WWW, HTP, HTF

1 Introduction

At the begining it should be noted that the article is not a course of create pages, learning HTML[1] or CSS[2] cascading. To take full advantage of the information contained herein, readers should be even familiar with HTML, CSS styles, basics of PL/SQL[3], including, in particular with:

- ORDImage[4] methods and functions,
- access descriptors,
- methods and the use of HTP[5] and HTF[6] packages.

The working plan is as follows:

- presentation of the schema editor
- Overview of functionality and interface,
- implementation of the editor.

[1] HTML – HyperText Markup Language
[2] CSS - Cascading Style Sheets
[3] PL/SQL – procedural language extension to structured query language (SQL)
[4] ORDImage – Oracle interMedia object type, which supports the storage, management, and manipulation of image data
[5] HTP – hypertext procedures package to generate HTML tags
[6] HTF – hypertext functions package to generate HTML tags

2 Pattern of action

Editor as the name suggests is aimed to editing - in this case the graphics files. Since the target is available through the web interface, working principle will be a little different from the normal program running on your computer. The first basic difference:

1. application from a PC has direct access to files on disk,
2. Second - This application also has direct access to resources such as CPU, RAM, etc.

By contrast:

- Our editor must receive files that will be processed,
- and is reliant on the sharing of hardware resources of the server on which it is run, but unfortunately the server has to himself, that is usually not only we, and not only one application using it.

What does this mean? Well - the speed of operation of the system will left much to wish and will depend mainly on the speed of Internet connection, since the image must first be send to the server and then (after some processing time) downloaded to your local hard disk. Processing time is difficult to define, also it is hard to say whether it would be slower or faster compared to the PC on which we work. Everything depends on the performance of hardware and server configuration. But this is another issue in which we will not discuss further.

The diagram of operation is as follows (Fig. 2.1):

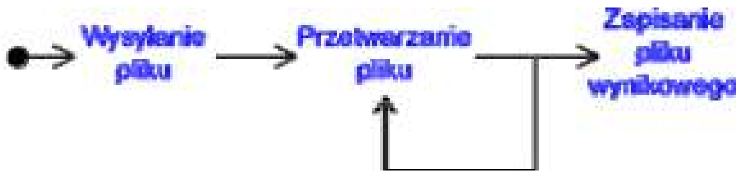


Figure 2.1 Schematic of on-line editor

3 Functionality and interface

Looking for stationary applications, we can mention a few features of a good editor, which could also implement its 'mobile' version:

- preview of the action of the process of editing,
- Gallery - common applications have access to files on disk, therefore can view their preview, here's a good idea to remain uploaded images, so if necessary it can be downloaded or further processed without re-loading the server.

Image processing may allow:

- update uploaded file of course
- Generate thumbnails
- resize
- compression
- change the contrast

- change file format
- rotation of the angle
- cutting area
- deleting a file from the server.

On the interface is not going to dwell specifically. Pure HTML does not look interesting and if we create a graphical editor that does not hurt to create some CSS, to provide an aesthetic appearance of the page. It was a bad idea to put the style in PL/SQL code, because any change would have to be accompanied by a compilation of code. On the other hand, Oracle (opposed to parties based on Apache + PHP) do not have the directory in which .css files can by place and include in your HTML code. So how do we solve this problem in order to influence the appearance of the editor without having to compile the source?

Create a file with the styles and put it in a directory such as:
\`C:\oracle\my_files\css\edytor.css\`. Create an alias to the folder in Oracle:

```
   CREATE DIRECTORY IMG_CSS_DIR AS
'C:\oracle\my_files\css\';
```

thanks to it we have access to the above directory from PL/SQL code. Then, create a function that will load the contents of the file and place it on your page:

```
   CREATE OR REPLACE PROCEDURE img_css IS
       buf VARCHAR2(100) := ''; css VARCHAR2(10000);
       F1 UTL_FILE.FILE_TYPE;

   BEGIN
```

Open a file with the definition of styles:

```
       F1 := UTL_FILE.fOpen('IMG_CSS_DIR', 'edytor.css',
'R', 100);
```

and we read its contents line by line:

```
       LOOP
             UTL_FILE.GET_LINE(F1, buf);
             EXIT WHEN LENGTH(buf)=0;
             css := css || buf || chr(13) || chr(10);
       END LOOP;
       UTL_FILE.fClose(F1);
```

Function \`htp.style\` places text between the tags parameter <style></style>:

```
       htp.style(css);
      END;
```

In this way, when the code for creating a web page to has a call to \`img_css()\`, defined style sheet will be read directly from the file each time it is called a site.

At this point we can go to meet the objectives of our functional editor.

4 Editor on-line

First what we need to be addressed is to send a file to the server. This will enable by Website form:

```
     htp.formOpen('img_f_upload', 'POST', '_self',
     'multipart/form-data');
     htp.formFile('new_file');
     htp.formSubmit;
     htp.formClose;
```

which sends data directly to a function `img_f_upload`:

```
     CREATE OR REPLACE PROCEDURE img_f_upload(new_file
VARCHAR2) IS
```

In her body, retrieve the data about the file loaded by the form as `new_file`:

```
     SELECT mime_type, doc_size, dad_charset,
   last_updated,
            content_type, blob_content
     INTO   v_mime_type, v_doc_size, v_dad_charset,
   v_last_updated,
            v_content_type, v_blob_content
     FROM img_upload_table
     WHERE name = new_file;
```

insert new record to the table that will hold editor pictures:

```
     INSERT INTO images ( img_id, img, img_min, tmp)
     VALUES ( images_id.nextval, ORDSYS.ORDIMAGE.INIT(),
ORDSYS.ORDIMAGE.INIT(), ORDSYS.ORDIMAGE.INIT() )
     RETURN img_id INTO new_id;
```

retrieve the newly inserted objects (still empty):

```
     SELECT img,img_min INTO new_img,new_imgm
     FROM images WHERE img_id = new_id FOR UPDATE;
```

Under the variable `img` will copy the image of uploaded file

```
     DBMS_LOB.COPY( new_img.source.localData,
v_blob_content, v_doc_size );
```

and under the `img_min` automatically created thumbnail to have a preview of the file without loading the original:

```
     new_img.processCopy( 'maxScale=64 64,
fileFormat=JFIF', new_imgm );
```

At the end we updating the target table and write data

```
     UPDATE images SET img = new_img, img_min = new_imgm
     WHERE img_id = new_id;
```

Bearing the file - or files – in the database you want to display them to the user to be able to choose which one wants to be treated. Given the fact that everything is done via the internet and the images typically have a large size we will display thumbnails created when uploading images. Again, we come to a similar problem as in the styles case. We know how to deal with non-existent directory, but this time the file is in the BD and will not export it to a directory and then download it from the source of the page. We'll display them directly from the database using the PL/SQL:
The parameters tell us: which record with the image is interest to us - we look after \`id\` and if you want to download thumbnail (thumb = \`1\`) or the original file size:

```
     CREATE OR REPLACE PROCEDURE img_get(id NUMBER, thumb
NUMBER := 0) IS
            v_img ORDImage;
     BEGIN

            IF thumb=1 then
              SELECT img_min INTO v_img FROM images WHERE
img_id=id
            ELSE
              SELECT img INTO v_img FROM images WHERE
img_id=id;
            END IF;
```

We need say to your browser's what type of data will be sent to her. For this purpose, before the binary data we serve MIME header

```
     owa_util.mime_header(v_img.mimeType, TRUE);
```

Now you can send the binary data

```
     wpg_docload.download_file(v_img.source.localData);
```

So constructed procedure, we can use as follows:

```
     htf.img('img_get?id=3&thumb=1', 'bottom', 'id=3')
```

which means that we want to display a thumbnail image of the record with id equal to 3.

With the ability to upload and view files and their thumbnails it is remain to make a major mechanism - the heart of the editor - that is, methods which do processing of images.
In fact, we need a pair of procedures:

- one displays a form where the user enters the necessary values of parameters by which change are made, such as \`45\` for rotate photos by 45 degrees,
- perform a second operation on the selected image and save the result.

Methods which displaying the form are similar to those presented earlier, used to send the file to the server. The only treatment that modifies will adjust the number and types of fields for the needs of functionality. For example, the form through which we obtain the values necessary for cropping photos will be like this:

```
     CREATE OR REPLACE PROCEDURE img_form_cut IS
     BEGIN
            htp.formOpen('img_f_cut', 'POST', '_self');

            htp.p('Type x: ');
            htp.formText('x', 3);
            htp.p('Type y: ');
            htp.formText('y', 3);
            htp.p('Type width: ');
            htp.formText('w', 3);
            htp.p('Type height: ');
            htp.formText('h', 3);
            htp.formHidden('id', '3');
            htp.formSubmit;
            htp.formClose;
     END;
```

Since the above form affected cropping now we present the method to realize this operation. Parameters are given by the form. It is very important that the names were identical - the order does not matter.

```
     CREATE OR REPLACE PROCEDURE img_f_cut ( id
NUMBER:=0,
     x VARCHAR2:='', y VARCHAR2:='', w VARCHAR2:='', h
VARCHAR2:='') IS
```

\`f\` will keep the action name to execute. Tthe variable \`process\` will serve to build a chain of command that will be transferred to the method \`process\`, and that depending on the command syntax, performs the final processing.

```
      f VARCHAR2(20) := 'cut';
      process VARCHAR2(100);

     BEGIN

       process := 'cut='||x||' '||y||' '||w||' '||h;
```

Function \`Edit\` has to display the appropriate form and a picture before or after making your modifications, together with comments returned by \`img_process\` (described below).

```
       edit(id, page, f, img_process(process, id, 'img',
'img') );

     END;
     /
```

The parameters adopted by the `img_process` mean respectively:

- command, which occurs by converting the image
- ID of record in a table with pictures
- the field name from the original image
- destination field name

```
     CREATE OR REPLACE FUNCTION img_process(process
VARCHAR2, id NUMBER,
                          src VARCHAR2, dest VARCHAR2)
     RETURN VARCHAR2 IS
            i ORDImage;
            i2 ORDImage;
            ctx RAW(64) := NULL;
            query VARCHAR2(200);
     BEGIN
```

If the source is also destination, so that in the event of an error not to lose the original data, the operations we perform on extra object `tmp`.

```
         if src=dest then
```

We are building a SQL query...

```
         query := 'SELECT '||src||', tmp FROM images
WHERE img_id='||id
                     ||' FOR UPDATE';
```

...then call them dynamically using the `EXECUTE IMMEDIATE` statement

```
        EXECUTE IMMEDIATE query INTO i,i2;
```

Calling a method `process` on ORDImage object types makes the image change in a specified way:

```
        query := process;
        i.processCopy(process, i2);
        i2.setProperties();
```

At the end save changes.

```
        query := 'UPDATE images SET '||src||'=:1 WHERE
img_id='||id;
        EXECUTE IMMEDIATE query USING i2;
```

We make similar instructions for the case where the source object and target are different.

```
    else
      query := 'SELECT '||src||', '||dest||' FROM images
WHERE img_id='
           ||id||' FOR UPDATE';
      EXECUTE IMMEDIATE query INTO i, i2;
      query := process;
      i.processCopy(process, i2);
    end if;

    COMMIT;
    RETURN ('OK!');

  END;
```

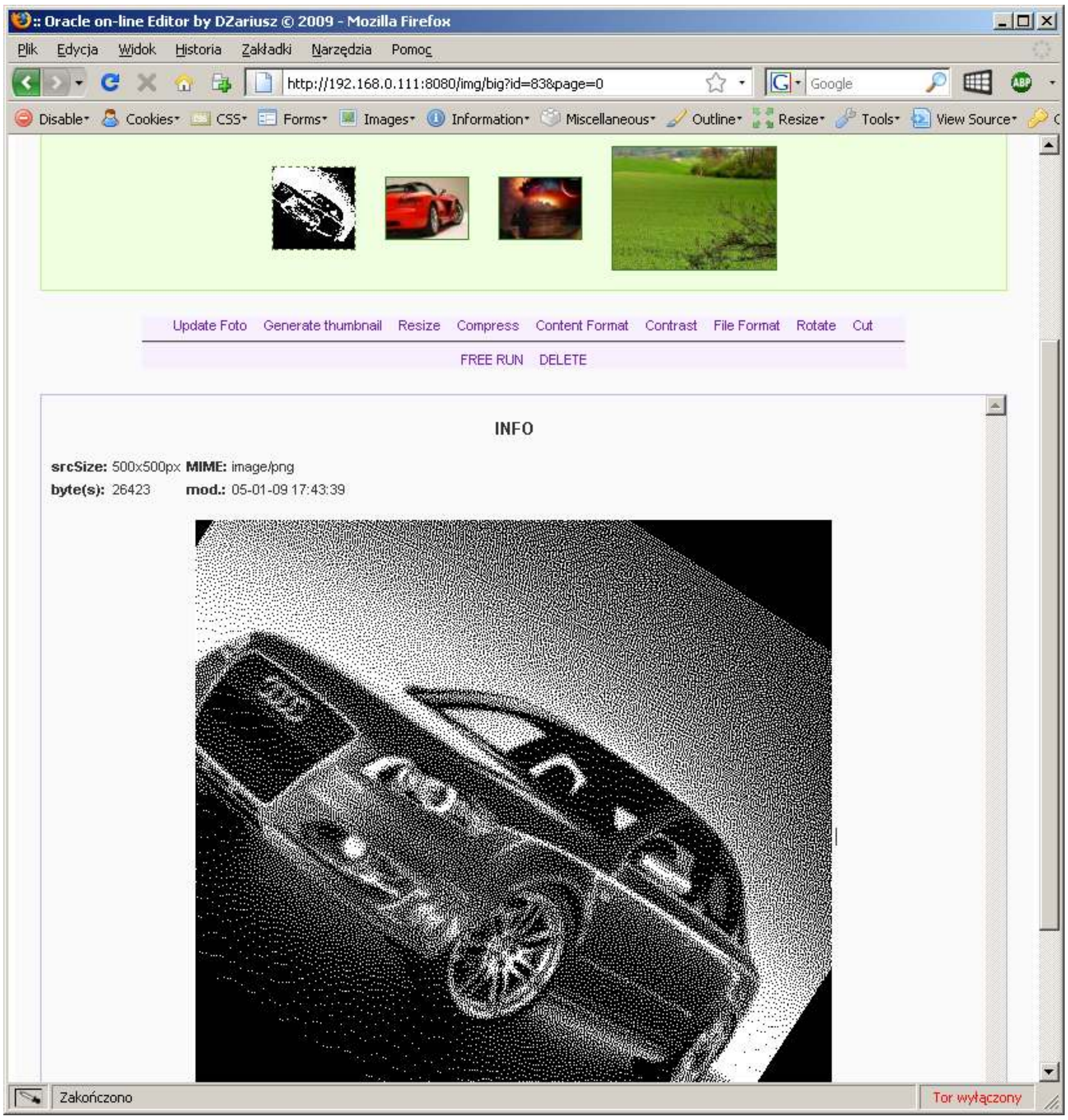
In fact, at this moment we have all the necessary knowlage to create a graphics editor online. Has been described as:

- use of cascading style sheets CSS
- upload files to the server,
- view files
- create forms for data entry
- perform operations on objects.

Since the functional assumptions are still the other features, will present below sequences of commands which, when transmitted to the `process` will fulfill the task:

- we update upload file in the same way that getting the new one except that it omitted to insert a new record,
- Generate thumbnails:
  `process := 'maxScale='||max_px||' '||max_px||', fileFormat=JFIF';`
  This method we used when creating thumbnails for new file after uploading to the database,
- resize: it can be done in two ways:
  - forcing for proportion:
    `process := 'maxScale='||w||' '||h;`
  - and without proportion:
    `process := 'fixedScale='||w||' '||h;`
- compression:
  `process := 'compressionFormat='||cf;`
  Supported compression formats: JPEG, LZW, DEFLATE, and ASCII or uncompressed NONE.
- change the contrast:
  `process := 'contrast='||r||' '||g||' '||b;`
  Values for r, g, b, we give in percentage from 0 to 100.
- change file format:
  `process := 'fileFormat='||ff;`
  `ff` can take values like: Jiff, GIF, PNGF, TIFF, WBMP
- rotating:
  `process := 'rotate='||r;`
- deleting a file from the server.: this risky issue will leave without comment.

Finally, our editor might look as follows (Figure 4.1):


:: Oracle on-line Editor by DZariusz © 2009 - Mozilla Firefox
Plik Edycja Widok Historia Zakładki Narzędzia Pomoc
http://192.168.0.111:8080/img/big?id=83&page=0
Update Foto Generate thumbnail Resize Compress Content Format Contrast File Format Rotate Cut
FREE RUN DELETE
INFO
Zakończono
Tor wyłączony

srcSize: 500x500px **MIME:** image/png
byte(s): 26423 **mod.:** 05-01-09 17:43:39

Figure 4.1 A screenshot of the final on-line editor

From the top (on a green background) are displayed thumbnail images on the server. Below links to relevant forms and functions, then preview of the selected image in its original size.

5 Summary

At the end it should be noted that in each of the examples of PL/SQL had to verify the accuracy of reported data. If we expect the new width of image – it should be given a value greater than zero, or whether at all is given? Likewise, when uploading photos:

- handling possible exceptions,
- the appropriate response to errors
- check file size before saving it in the target table - if the size is zero - write a message to the user of the failure.
- Etc.

Such treatments in an attempt to demonstrate how to build on-line editor unnecessarily put to confusion and obscured the examples presented in the SQL code, so have been omitted.
Full scripts with examples and CSS styles, taking the validation variables and error handling are available at: http://dzariusz.pl/oracle/editor_online/.

References

1. Oracle Database Online Documentation 10g Release 2 (10.2) http://www.mcs.csueastbay.edu/support/oracle/doc/10.2/nav/portal_5.htm
2. „Oracle Database 10g Express Edition. Tworzenie aplikacji internetowych w PHP”, Helion, 2007