Goal: collect screenshots for each page of an Oracle APEX application in order to prepare a user documentation.

I considered a custom developement for automating screenshots but I realized that it was not trivial and I had a look to an online service instead. There are plenty of these kinds of service and I tested the following subset, considering only ones providing a rest API:

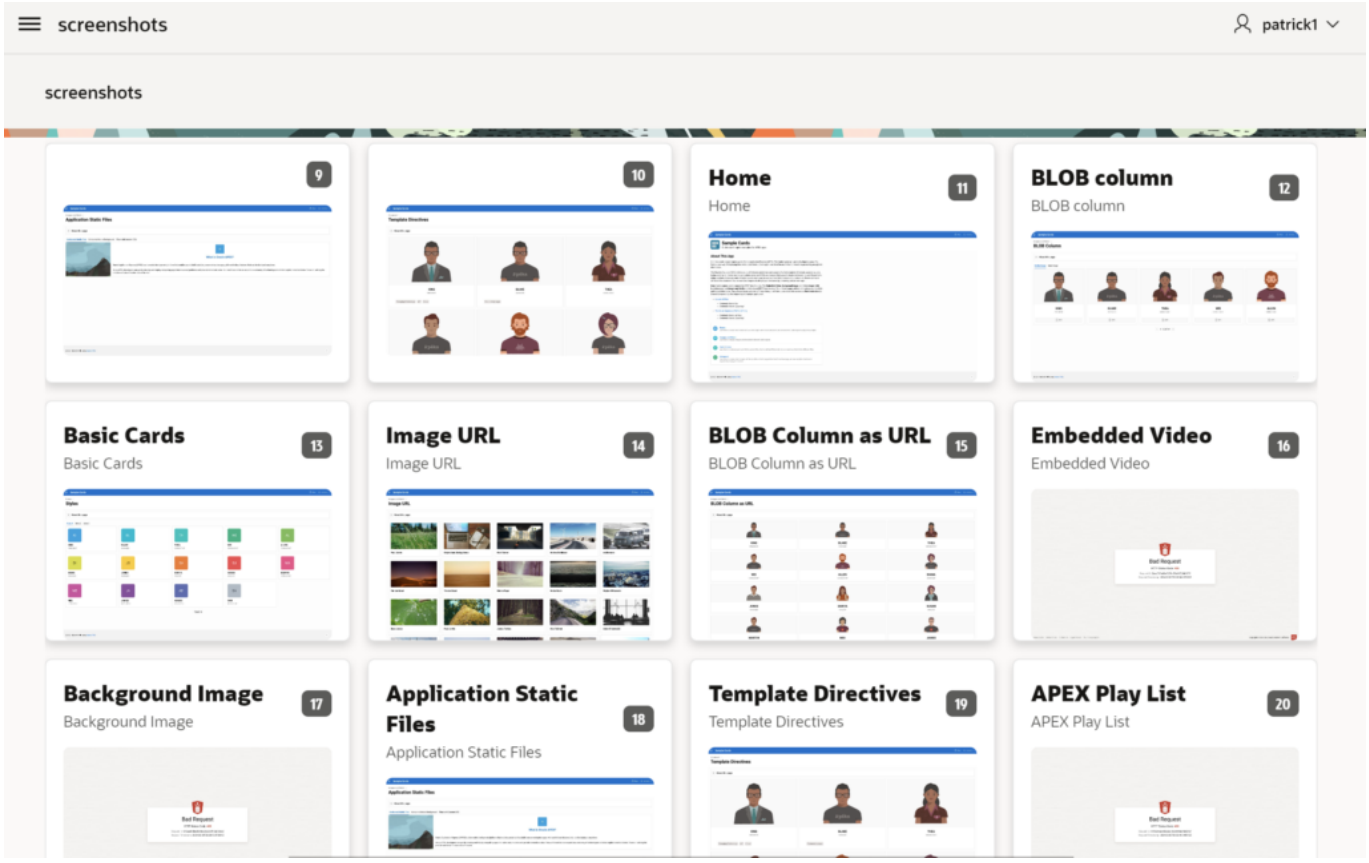
On line service	Basic Plan	Evaluation	Drawbacks
pikwy	3€/month + 0,003/snapshot	Very good	ok with apex.oracle.com but Impossible to access a free tier instance (timeout). Possible to access protected pages.
apiflash	Free with a limit of 100 snapshots/month (upgrade at 7€/month)	Very good	not possible to access protected pages
screenshotmachine	Free with a limit of 100 screenshots/month (upgrade at 9€/month)	Very good	not possible to access protected pages
site-shot	5€/mont for 2000 snapshots	Very good	not possible to access protected pages
url2png	29€/month for 5000 screenshots	not tested	
getscreenshotapi	5€Month for 2500 screenshots	not tested	

Finally, I realized my prototype with apiflash and I choosed the « Sample Cards » application as a demo case.

It’s just a matter of calling the screenshot API for each page (https://api.apiflash.com/v1/urltoimage) and put the result in a companion able (DEMO_BLOB). cf procedure code after and a sample display app.

Important: The API doesn't deals with authentication, so I set a new scheme at « No authentication », for the duration of the test, at least, then I switch back to the regular auth scheme. *Pickwy* allows providing credentials, but suffers a specific anomaly in may case (cf above)

Limitations: the modal pages can't be managed this way, except if API gives opportunity to send keystrokes before, or navigating possibilities.



Sample procedure

```
procedure buildscreens (pid number)
is
```

```
    turl varchar2(2000) :=
'https://api.apiflash.com/v1/urltoimage?access_key=
<ACCESS_KEY>&url=<APEX_ENCODED_URL>';
    l_blob    BLOB;
    tbody    CLOB;
    target varchar2(2000);

begin

    for c in (select page_alias, page_name
              from APEX_APPLICATION_PAGES
              where application_id=pid and PAGE_ALIAS IS NOT NULL
              ) loop
        target := turl || lower(c.PAGE_ALIAS);
        l_blob := APEX_WEB_SERVICE.make_rest_request_b(
            p_url         => target ,
            p_http_method => 'GET'
        );
        insert into demo_blob(image, page_name) values (l_blob,
c.page_name);
        commit;
        if apex_web_service.g_status_code = 404 then
          null;
          -- return '1';
        end if;
    end loop;
end;
```

Author

Patrick

GPM Factory