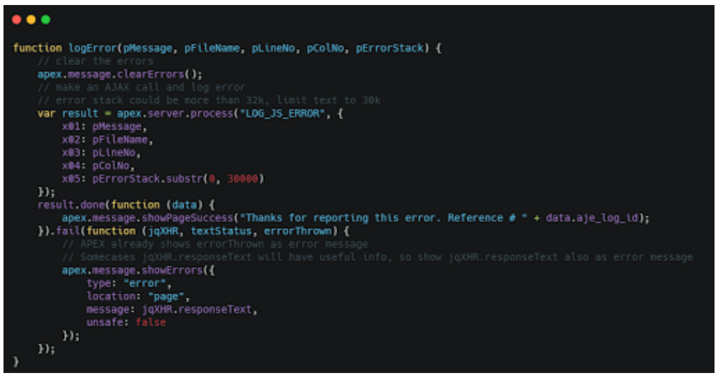
Handling JavaScript Errors - Part 2

January 23, 2021

In my previous blogpost, I have explained about JavaScript errors and how we can handle them in general and how we can handle them at application level. If you have not read it yet, you can read it here. In this blogpost, I will explain an approach to log JavaScript errors to a database table.



```
function logError(pMessage, pFileName, pLineNo, pColNo, pErrorStack) {
    // clear the errors
    apex.message.clearErrors();
    // make an AJAX call and log error
    // error stack could be more than 32k, limit text to 30k
    var result = apex.server.process("LOG_JS_ERROR", {
        x01: pMessage,
        x02: pFileName,
        x03: pLineNo,
        x04: pColNo,
        x05: pErrorStack.substr(0, 30000)
    });
    result.done(function (data) {
        apex.message.showPageSuccess("Thanks for reporting this error. Reference # " + data.aje_log_id);
    }).fail(function (jqXHR, textStatus, errorThrown) {
        // APEX already shows errorThrown as error message
        // Somecases jqXHR.responseText will have useful info, so show jqXHR.responseText also as error message
        apex.message.showErrors({
            type: "error",
            location: "page",
            message: jqXHR.responseText,
            unsafe: false
        });
    });
}
```

SHARE

Labels

Error Handling
Error Logging
JavaScript

`error` event on `window` object provides lot of information. I error event on window object provides lot of information. I found below information is useful for debugging and worth logging into DB.

- `event.message`: JavaScript error message
- `event.filename`: If source of the error is from a JavaScript file, then this will give full path of JavaScript file. If error is from inline JavaScript code written in APEX page, then this will give full APEX page URL including APEX session.
- `event.lineno`: Line number where error has occurred. Depending on event.filename, it could refer to line number from JavaScript file or from generated HTML page.
- `event.colno`: Column number where error has occurred. It should be read along with `event.lineno`.
- `event.error.stack`: Call stack information. Similar to `DBMS_UTILITY.FORMAT_ERROR_STACK` in PL/SQL.

Now, let's create a table to store these JavaScript errors. Please find sample table creation script below which I have used for the demo.

```
-- create tables
create table apex_js_error_log (
    aje_log_id                         number generated
                                       constraint apex_j
    message                            varchar2(4000 cha
    filename                           varchar2(255 char
    lineno                             number,
    colno                              number,
    error_stack                        clob,
    app_id                             number,
    page_id                            number,
    session_id                         number,
    app_user                           varchar2(255 char
    created_on                         date not null,
    created_by                         varchar2(255 char
);

-- triggers
create or replace trigger apex_js_error_log_bi
    before insert
    on apex_js_error_log
    for each row
begin
    :new.created_on := sysdate;
    :new.created_by := coalesce(sys_context('APEX$SE
end apex_js_error_log_bi;
/
```

Let's create a PL/SQL function to log errors

Let's create a PL/SQL function to log errors.

```
CREATE OR REPLACE FUNCTION log_js_error (
    p_message      IN  VARCHAR2,
    p_filename     IN  VARCHAR2,
    p_lineno       IN  NUMBER,
    p_colno        IN  NUMBER,
    p_error_stack  IN  VARCHAR2,
    p_session_id   IN  NUMBER,
    p_app_id       IN  NUMBER DEFAULT apex_applicati
    p_page_id      IN  NUMBER DEFAULT apex_applicati
    p_app_user     IN  VARCHAR2 DEFAULT apex_applica
) RETURN NUMBER IS
    l_aje_log_id apex_js_error_log.aje_log_id%TYPE;
BEGIN
    INSERT INTO apex_js_error_log (
        message,
        filename,
        lineno,
        colno,
        error_stack,
        app_id,
        page_id,
        session_id,
        app_user
    ) VALUES (
        p_message,
        p_filename,
        p_lineno,
        p_colno,
        p_error_stack,
        p_app_id,
        p_page_id,
        p_session_id,
        p_app_user
    ) RETURNING aje_log_id INTO l_aje_log_id;

    RETURN l_aje_log_id;
END log_js_error;
```

Let's create an application process LOG_JS_ERROR to call and log errors.

- Name: **LOG_JS_ERROR**
- Process Point: **Ajax Callback: Run this application process when requested by a page process.**
- Source:
    - Code: *As shown below*

```
DECLARE
    l_aje_log_id   apex_js_error_log.aje_log_id%TYPE
    l_json_output  json_object_t;
BEGIN
    l_aje_log_id := log_js_error(p_message => apex_a
                                 p_filename => apex_a
                                 p_lineno => apex_app
                                 p_colno => apex_appl
                                 p_error_stack => ape
```

```
                                    p_session_id => :app_
    -- Send error log id as JSON object to JavaScrip
    -- If you are using older version of DB (< 12.2)
    l_json_output := NEW json_object_t;
    l_json_output.put('aje_log_id', l_aje_log_id);
    htp.p(l_json_output.to_string);
END;
```

And finally, let's create a JavaScript function to make an AJAX call which logs errors into DB table.

```
function logError(pMessage, pFileName, pLineNo, pColN
    // clear the errors
    apex.message.clearErrors();
    // make an AJAX call and log error
    // error stack could be more than 32k, limit tex
    var result = apex.server.process("LOG_JS_ERROR",
        x01: pMessage,
        x02: pFileName,
        x03: pLineNo,
        x04: pColNo,
        x05: pErrorStack.substr(0, 30000)
    });
    result.done(function (data) {
        apex.message.showPageSuccess("Thanks for rep
    }).fail(function (jqXHR, textStatus, errorThrown
        // APEX already shows errorThrown as error m
        // Somecases jqXHR.responseText will have us
        apex.message.showErrors({
            type: "error",
            location: "page",
            message: jqXHR.responseText,
            unsafe: false
        });
    });
}
```

In the demo application, I have added this function to js_error_logging.js file and then I have referred the JS file in `Application > User Interfaces > JavaScript > File URLs` section. If you are not sure what's best way to include JavaScript code to APEX applications, then please do read Adding JavaScript to an Application Express application from APEX documentation.

Now, let's create Dynamic Action (DA) in Page-0 as below.

- Name: **Handle Errors or any proper name**
- Execution Options:
    - Sequence: **0**
- When:
    - Event: **Page Load**

- Event: Page Load

- True Action:
  - Action: **Execute JavaScript Code**
  - Code: *As shown below*

```
window.addEventListener('error', function (event) {
    // show APEX error message
    // provide option for users to log/report the er
    apex.message.showErrors({
        type: "error",
        location: "page",
        message: "Unhandled JavaScript Error. Please
        unsafe: false
    });
    // log error if user clicks on the link
    $("#log_js_error").click(function () {
        logError(event.message, event.filename, even
    });
});
```

If you want to enable automatic error handling only for few APEX pages, then you can achieve it by defining "Server Side Condition" for this DA.

If you see above DA code, we are using `apex.message.showErrors` to inform users about JavaScript errors. And then, we are providing option for users to log/report the error, instead of automatically logging the error. If we automatically log errors, then in some cases it could be overwhelming and in some cases it could cause additional problems. For e.g. if you have scheduled to execute some code repeatedly using `setInterval`, and if the code throws any JavaScript errors, and if we automatically log errors to DB, then it can cause additional traffic to server and in some cases can consume all DB connections at ORDS level.

Above approach, where we are just showing error message could

Above approach, where we are just showing error message could be equally annoying with `setInterval`. If JavaScript throws any errors, then users will see error messages continuously. So, in such cases, we can use "Nested setTimeout" technique to simulate `setInterval`.

Example code for "Nested setTimeout" below.

```
let timerId = setTimeout(function doSomeTask() {
  // throws error
  console.log(pValue);
  // go for next iteration only if current iteration
  // below code will be executed only if all of the
  timerId = setTimeout(doSomeTask, 5000);
}, 5000);
```

You can read more about Scheduling with `setTimeout` vs `setInterval` here.

That's it and you can see the demo here.

Caution: This approach can be misused to perform Denial-of-service attack. So, I suggest to use this approach with caution. It's best to use this approach in UAT/Staging environments where users test your applications. This will help to detect any JavaScript errors and subsequently to make your applications error free.

If you are not already aware, when you are running application as Developer, APEX gives you little red color warning icon in developer toolbar, so we can find out most of JS errors during development phase itself.

Thank you.

LABELS: ERROR HANDLING, ERROR LOGGING, JAVASCRIPT SHARE

Comments

Post a Comment

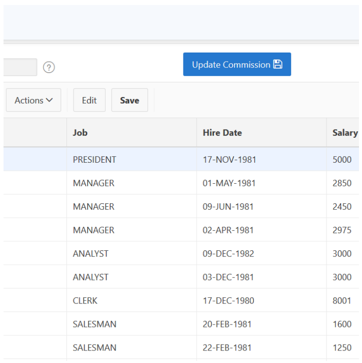
Popular posts from this blog

Interactive Grid - Bulk Operation on Selected Rows

February 09, 2019

What's the Problem? Let's say you have an IG on employee table and you want to update selected employees commission based on user's input. Ideally, it should be very simple, where you can write UPDATE …

SHARE 31 COMMENTS READ MORE



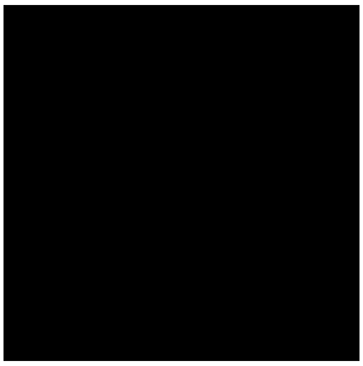
Job	Hire Date	Salary
PRESIDENT	17-NOV-1981	5000
MANAGER	01-MAY-1981	2850
MANAGER	09-JUN-1981	2450
MANAGER	02-APR-1981	2975
ANALYST	09-DEC-1982	3000
ANALYST	03-DEC-1981	3000
CLERK	17-DEC-1980	8001
SALESMAN	20-FEB-1981	1600
SALESMAN	22-FEB-1981	1250

Interactive Grid - Conditional Enable/Disable

July 05, 2020

In this blogpost, I am going to discuss few approaches using which we can conditionally enable/disable Interactive Grid (IG) column(s) based on other column(s) values. Note: There is a bug 30801170 in APEX 19.2/2 …

SHARE 4 COMMENTS READ MORE



Oracle APEX Stylish Radio Buttons

May 31, 2020

Do you know you can convert your radio buttons into

Do you know you can convert your radio buttons into something stylish and fancy almost declaratively in Oracle APEX? If you are not aware about this, then you may find this blog post useful. Pill Buttons You can easi …

[redacted]

SHARE   POST A COMMENT   READ MORE

→

About Me

Hari

I'm a passionate web application developer and technical leader with 14+ years of experience, primarily in Oracle APEX and Oracle Database technologies.

VISIT PROFILE

Follow @hari_639

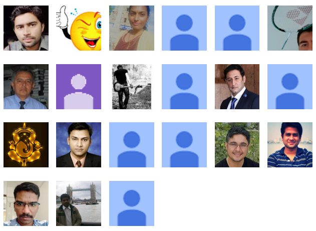
Archive

Labels

Report Abuse

Followers

Seguidores (25) Siguiente



Seguir

Subscribe To

Posts

Comments

Contact Form

Name

Email *

Message *

Send