MUNIQSOFT

APEX Security

Marco Patzwahl

DOAG APEX CONNECT 2018

Muniqsoft GmbH

- **Tätigkeitsbereiche:**
  - **Oracle IT-Consulting & Services**
  - **Oracle Remote Support und Rufbereitschaft**
  - **Oracle Schulungen (SQL, PL/SQL, DBA, APEX, … - gerne auch Inhouse)**
  - **Software-Lösungen**
  - **Oracle Lizenzen**

Muniqsoft GmbH
Schulungszentrum
Grünwalder Weg 13a
82008 Unterhaching
Tel.: 089 / 679090 40

Muniqsoft GmbH
IT-Consulting & Support
Witneystr. 1
82008 Unterhaching
Tel.: 089 / 6228 6789 0

Die größten Hackerangriffe

Firma	Wann	Betroffen
Ashley Madison	2015	37 Millionen Accounts
Home Depot Hack	2014	50 Mio. Kreditkartenaccounts
Ebay	2014	145 Mio. Benutzeraccounts
JPMorgan Chase	2014	Ca. 80. Mio Kundendaten
LinkedIn	2016	164 Mio Benutzeraccounts
Anthem Health Care	2015	78 Mio. Kundendaten
UBER	2017	57 Mio. Kunden und Fahrerdaten
MyFitnessPal	2018	150 Mio. Benutzeraccounts
Swisscom	2018	0.8 Mio Kundendaten
T-Mobile (US)	2017	Ca 70 Mio. Kundendaten
Paypal Tochter	2017	1.6 Mio Kundendaten

OWASP

- **Open Web Application Security Project**
  - **gemeinnützige, unabhängige Organisation**
  - **Ziel: Unterstützung von Unternehmen bei Entwicklung, Kauf und Wartung von sicheren Anwendungen**
  - **Angebot:**
    - **Werkzeuge und Standards für Anwendungssicherheit**
    - **Sicherheitsforschung**
    - **Publikationen, Sicherheitsmaßnahmen und –bibliotheken**
  - **Top 10: die zehn kritischsten Sicherheitsrisiken für Organisationen**

OWASP

Rang	OWASP Top 10 2017	2013
1	Injection	1
2	Broken Authentication	2
3	Sensitive Data Exposure	6
4	XML External Entities (XXE)	(Neuzugang)
5	Broken Access Control	4 & 7
6	Security Misconfiguration	5
7	Cross-Site Scripting (XSS)	3
8	Insecure Deserialization	(Neuzugang)
9	Using Components with Known Vulnerabilities	9
10	Insufficient Logging & Monitoring	(Neuzugang)

SQL Injection

SQL Injection

- SQL Injection ist der Versuch zusätzliche Kommandos/Filter in ein SQL Befehl einzubauen
- bei Oracle nicht möglich: mit Semikolon weiteres Kommando anzuhängen! Der DELETE hier wird nicht ausgeführt
  - ```
    SELECT sal FROM emp
    WHERE empno=7839; delete from emp;
    ```
- Angriffsmöglichkeiten der Hacker
  - Hinzufügen von Filtern: OR 1=1
  - Einbauen von SQL Funktionen, die als autonome Transaktionen in der DB Änderungen durchführt
    ```

SQL Injection - Beispiele

- **Report Query:**

  ```
  SELECT * FROM scott.emp WHERE ename='&P1_ENAME.';
  ```

  - Filter A' or 'a'='a → zeigt alle Zeilen der Tabelle an

- **UNION Klausel an SQL-Befehl hängen:**

  - `SELECT * FROM emp UNION SELECT … FROM all_tables;`

- **Unterabfrage anhängen**
- **Aufruf von gefährlichen Packages/Funktionen**
- **WHERE Klausel auskommentieren:** `--WHERE`

SQL Injection - Beispiele

- **Anzeige der installierten Benutzer:**
  - `select * from dual`
    `where 1 = sys.dbms_metadata.openw(null, (select`
    `listagg(username, ':') within group (order by username)`
    `from all_users))`

    Textlänge: 113!
  - `ORA-31600: Ungültiger Eingabewert`
    `ANONYMOUS:APEX_050000:APEX_PUBLIC_USER:APPQOSSYS:AUDSYS:BI`
    `:CTXSYS:DBSFWUSER:DBSNMP:DIP:DVF:DVSYS:FLOWS_FILES:GGSYS:G`
    `SMADMIN_INTERNAL:GSMCATUSER:GSMUSER:HR:IX:LBACSYS:MDDATA:M`
    `DSYS:OE:OJVMSYS:OLAPSYS:ORACLE_OCM:ORDDATA:ORDPLUGINS:ORDS`
    `YS:OUTLN:PM:REMOTE_SCHEDULER_AGENT:SCOTT:SH:SI_INFORMTN_SC`
    `HEMA:SPATIAL_CSW_ADMIN_USR:SPATIAL_WFS_ADMIN_USR:SYS:SYS$U`
    `MF:SYSBACKUP:SYSDG:SYSKM:SYSRAC:SYSTEM:WMSYS:XDB:XS$NULL`
  - `ORA-06512: in "SYS.DBMS_SYS_ERROR", Zeile 105`

SQL Injection - Schutzmaßnahmen

- **Bindvariablen statt Austauschvariabeln!**
  - `SELECT * FROM emp WHERE ename=:P1_ENAME;`
  - **Bindvariablen nur in WHERE-Klausel verwenden (nicht bei Spalten-/Tabellennamen, Gruppenfunktionen, ORDER BY)**
- **Aktivierung der Session State Protection**
- **Einsatz von VPD (Virtual Private Database)**
  - **nur in EE verfügbar**

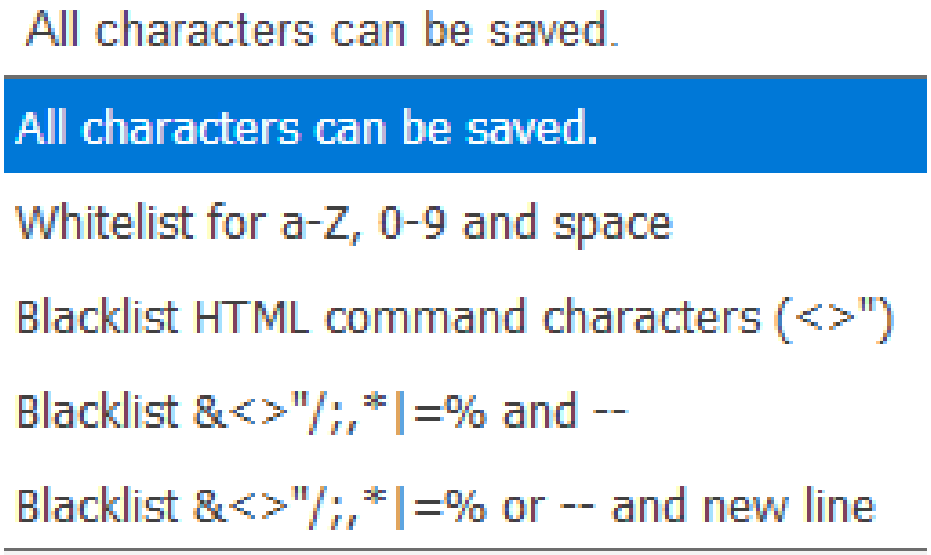
SQL Injection - Schutzmaßnahmen

- **werden Bindvariablen z.B. für Tabellennamen genutzt, verwenden Sie folgende Syntax:**

```
DECLARE
s CLOB;
BEGIN
s := 'select ENAME from EMP
      where ename= '|| nvl( chr(39) || replace(
          :P17_SUCHE, chr(39), chr(39)||chr(39) )
      || chr(39), 0);
htp.p(s);
RETURN s;
END;
```

Items gegenüber Sonderzeichen schützen

- **Ab Version 5.1 kann bei Textitems im Bereich Security /Restricted Characters ein zusätzlicher Schutz aktiviert werden:**



Items gegenüber Sonderzeichen schützen (f)

- **Prüfung, welche Items diesen Schutz besitzen:**
- ```
  SELECT f.flow_step_id,
  f.name,f.csize,f.cmaxlength,
  f.restricted_characters
  FROM apex_050100.wwv_flow_step_items f
  WHERE f.flow_id=:APP_ID
  AND f.flow_step_id=:APP_PAGE_ID;
  ```
  ```

gefährliche Module für SQL Injection

- APEX Variablen (&MY_VAR.)
- EXECUTE IMMEDIATE
- dynamische Cursor
  - REF Cursor
  - DBMS_SQL

- Bindvariablen (:MY_VAR)
- PL/SQL Cursor
  - CURSOR LOOP
  - OPEN FETCH CLOSE
- in anderen Programmier-sprachen (C, Java):
  - Feldlängen begrenzen
  - Filter einbauen
  - Werte maskieren

DBMS_ASSERT

- **Benutzereingaben mittels DBMS_ASSERT filtern**
- **DBMS_ASSERT.SIMPLE_SQL_NAME**
  - **Prüfung auf nutzbare SQL-Objektnamen**
  - **Beispiele:**
    - `SELECT DBMS_ASSERT.simple_sql_name('007Bond') FROM dual;` ✗
    - `SELECT DBMS_ASSERT.simple_sql_name('Bond007#') FROM dual;` ✓
    - `SELECT DBMS_ASSERT.simple_sql_name('"$Bond007"') FROM dual;` ✓

DBMS_ASSERT

- **DBMS_ASSERT.ENQUOTE_LITERAL**
  - **Prüfung, ob einfache Hochkommata falsch verwendet werden**
  - **Beispiele:**

```
SELECT dbms_assert.enquote_literal(
   q'#O''Neill#')
FROM dual;
```

```
SELECT dbms_assert.enquote_literal(
   'KING''or 1=1')
FROM dual;
```

DBMS_ASSERT

- **Beispiel** 

```
create or replace procedure doit(
        p_table in varchar2,
        p_valuecol in varchar2,
        p_wherecol in varchar2,
        p_param in varchar2 )
is
        v_value varchar2(32767);
        v_sql varchar2(32767);
begin
v_sql := 'select '||p_valuecol||' from '||p_table||
' where '||p_wherecol||' = '''||p_param||'''';
execute immediate v_sql into v_value;
end;
```

DBMS_ASSERT

- **Beispiel** ✓

```
create or replace procedure doit(
       p_table in varchar2,
       p_valuecol in varchar2,
       p_wherecol in varchar2,
       p_param in varchar2 )
is
       v_value varchar2(32767);
       v_sql varchar2(32767);
begin
v_sql := 'select
'||dbms_assert.enquote_name(p_valuecol)||' '|| 'from
'||dbms_assert.enquote_name(p_table)||' '|| 'where
'||dbms_assert.enquote_name(p_wherecol)||' =
'||dbms_assert.enquote_literal(p_param);
execute immediate v_sql into v_value;
end;
```

Reguläre Ausdrücke

- **alle Zeichen, die keine Buchstaben, Zahlen oder Leerzeichen sind, werden durch ein Leerzeichen ersetzt:**

```
SELECT regexp_replace(
'Hallo ''x'' ','[^[:alnum:][:blank:]]','')
FROM dual;
```

→ **Hallo x**

```
SELECT regexp_replace(
'Hallo "x" ','[^[:alnum:][:blank:]]','')
FROM dual;
```

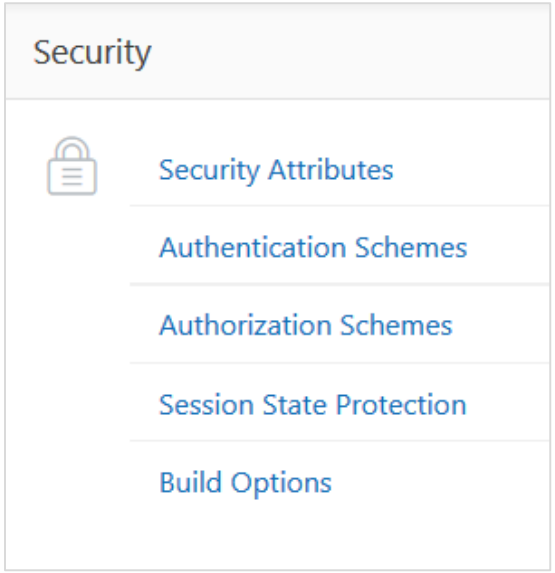
→ **Hallo x**

- **weitere Sonderzeichen müssen nach Bedarf freigegeben werden, wie z.B. - ; : .**

Fehler in Authentifizierung und Session Management

Übersicht des Security Bereichs

- **Shared Komponents:**
  - **Sicherheitseinstellungen**
  - **Möglichkeiten der Authentifizierung**
  - **Authorisierung für Seiten und Komponenten**
  - **Prüfsummen für Parameterübergabe**
  - **Aktivieren bzw. Deaktivieren von Elementen**



Benutzer und Passwörter

- für jeden Workspace einen Administrator mit einem Namen (nicht ADMIN) und einem langen Passwort anlegen
- strenge Passwortvorgaben einhalten und Passwörter regelmäßig ändern lassen
- Accounts sperren, wenn mehrmals ein falsches Passwort verwendet wurde

Neuer Admin für Internal Workspace

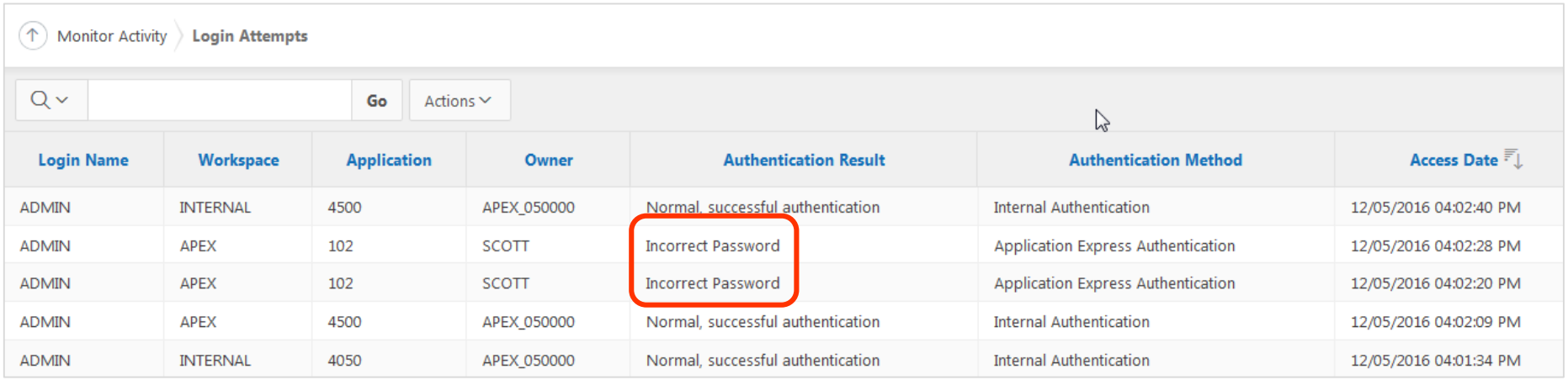
- neuen Benutzer für den Internal Workspace anlegen
- alternativ: Skript apxchpwd.sql
  - ADMIN Name
  - ADMIN Email
  - ADMIN Passwort
- Name sollte nicht ADMIN beinhalten (Besser MARCO_A)
- Passwort sollte schwer zu knacken sein (z. B. KuhliefumdenTeich17)

Überwachung von Logins

- **Überprüfen Sie regelmäßig die fehlerhaften Logins**
  - **alternativ mit dem SELECT:**

```
SELECT * FROM APEX_050100.APEX_WORKSPACE_ACCESS_LOG
WHERE authentication_result<>'AUTH_SUCCESS';
```

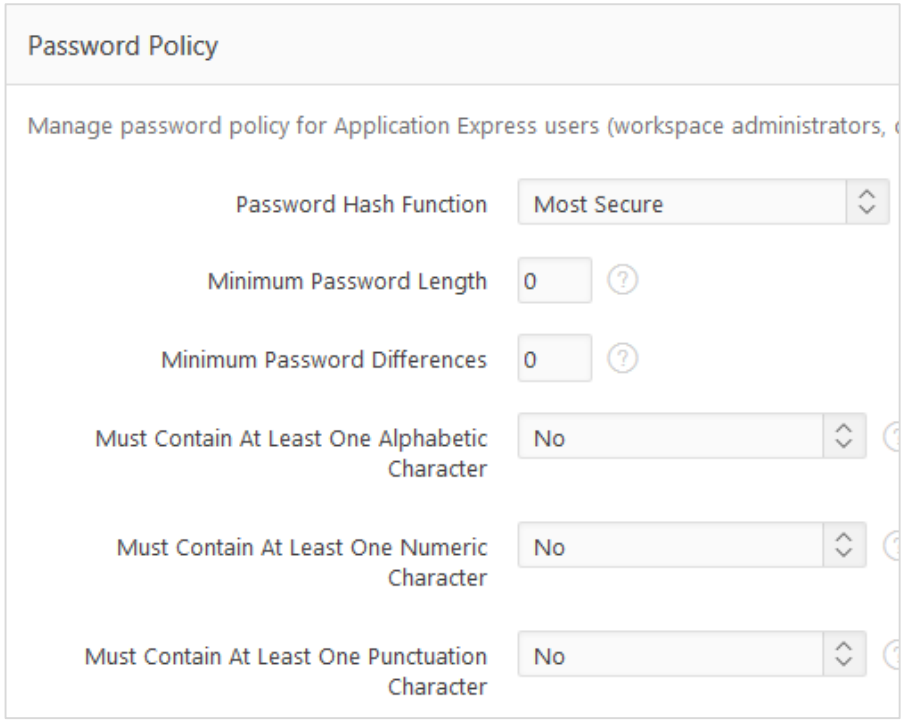
→ ggf. Email schicken



Login Name	Workspace	Application	Owner	Authentication Result	Authentication Method	Access Date
ADMIN	INTERNAL	4500	APEX_050000	Normal, successful authentication	Internal Authentication	12/05/2016 04:02:40 PM
ADMIN	APEX	102	SCOTT	Incorrect Password	Application Express Authentication	12/05/2016 04:02:28 PM
ADMIN	APEX	102	SCOTT	Incorrect Password	Application Express Authentication	12/05/2016 04:02:20 PM
ADMIN	APEX	4500	APEX_050000	Normal, successful authentication	Internal Authentication	12/05/2016 04:02:09 PM
ADMIN	INTERNAL	4050	APEX_050000	Normal, successful authentication	Internal Authentication	12/05/2016 04:01:34 PM

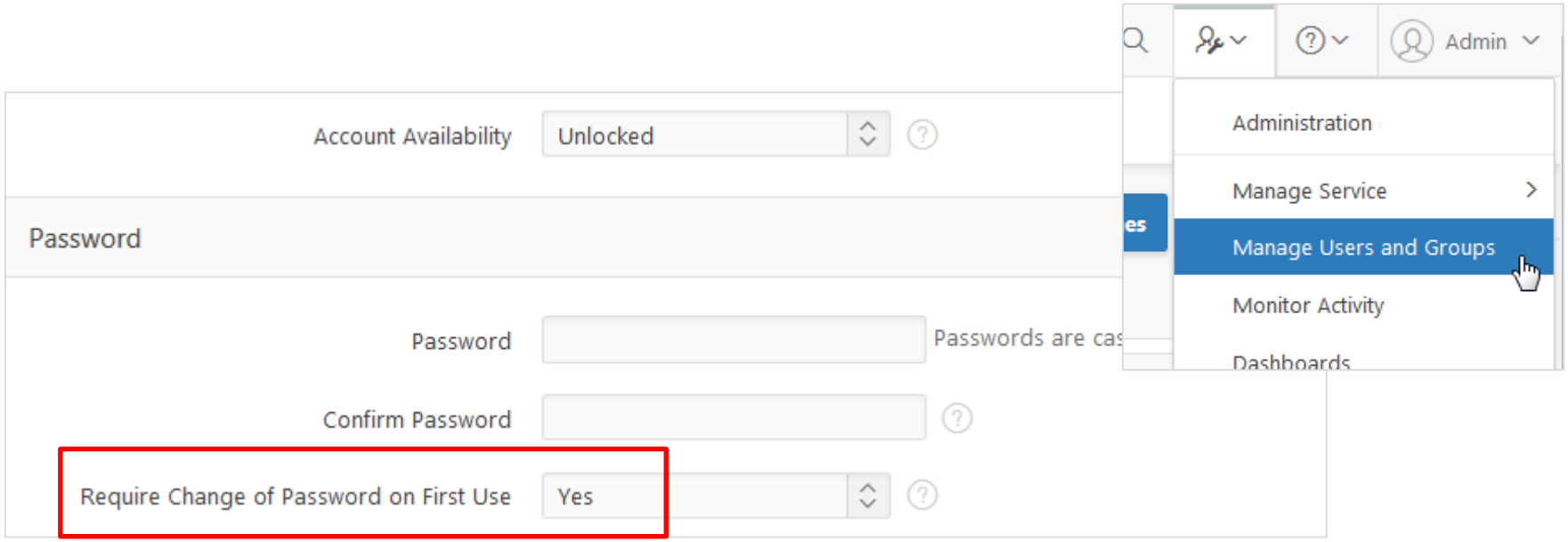
Password Policy

- **INTERNAL WS → Manage Instance → Security → Password Policy**



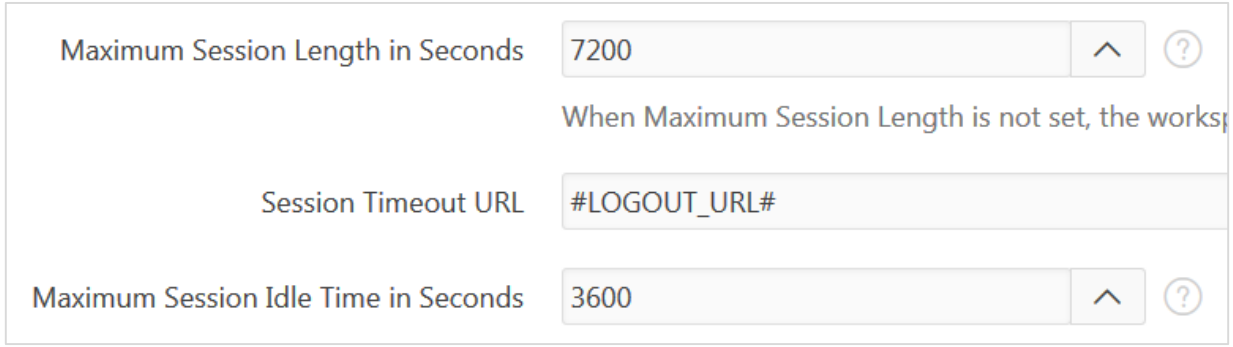
Accountverwaltung

- **Administration → Manage Users and Groups → Create | Edit User**
- **u.a. Passwortänderung beim ersten Anmelden**



Session Timeout

- **Festlegung der Session Lebensdauer (max. 12 Stunden)**
- **unter Shared Components → Security Attributes → Session Management**
  - **Tipp: erstellen Sie eine öffentliche Seite mit der Info, dass die Session abgelaufen ist**



Session Timeout

- **Festlegung der Session Lebensdauer per APEX_UTIL (ab 12c):**

```
BEGIN
APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(
      p_seconds => 7200);
END;
```

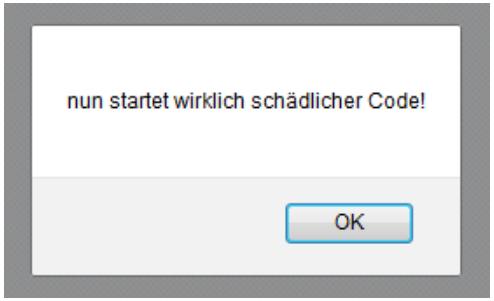
```
BEGIN
APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(
      p_seconds => 1200);
END;
```

- **Beispiel: Application Prozess (After Authentication)**

Cross Site Scripting

Cross Site Scripting

- **Problem: Scriptcode wird im Browser des Benutzers ausgeführt**
  - **Übernahme der Session**
  - **Änderung von Seiteninhalten**
  - **Umleitung auf bösartige Seiten**



Cross Site Scripting

- **XSS Beispiel:**
  - **Tabellen Kommentare anlegen (mit HTML Tags zur Formatierung, siehe Notizen)**
  - **Report erstellen:**
    ```
    select * from kommentare
    ```
  - **Anpassung der Kommentar-Spalte:**
    - **Escape Special Characters: NO**
    - **Sicherheitslücke!!**

APEX_ESCAPE (ab 5.0)

- **Lösung: APEX_ESCAPE Package (Groß und Kleinschreibung wird unterschieden!)**
  - **Report Query:**

```
select
   datum,
   username,
   apex_escape.html_whitelist(
          kommentar,
          '<h1>,</h1>,<h2>,</h2>,<h3>,
           </h3>,<h4>,</h4>,<p>,</p>,
           <b>,</b>,<strong>,</strong>,<i>,</i>,
           <ul>,</ul>, <ol>,</ol>,<li>,</li>,
           <br/>,<hr/>, <em>,</em>'
   ) as kommentar
from kommentare
```

APEX_ESCAPE (ab 5.0)

- **APEX_ESCAPE.HTML:**
  - **Escaping mithilfe eines PL/SQL Aufrufs**
  - **Analog zur Standard-Einstellung in APEX**

    → alles wird maskiert!

- **APEX_ESCAPE.HTML_WHITELIST**
  - **lässt die in der Whitelist enthaltenen HTML-Tags intakt**
  - **kann auch selbst definiert werden**
  - **standardmäßig in der Whitelist:**
    **<h1>, </h1>, <h2>, </h2>, <h3>, </h3>, <h4>, </h4>, , , , , , , , , <ul>, , <ol>, , <li>, ,
, <hr/>**

APEX_ESCAPE

- `SELECT apex_escape.html('<B>Hello</B>') FROM dual;`
  - `<B>Hello</B>`
- `SELECT apex_escape.html_whitelist ('<B>Hello</B>','<B>,</B>') FROM dual;`
  - `<B>Hello</B>`
- `SELECT apex_escape.json(q'!O'Brien!') FROM dual;`
  - `O\u0027Brien`

APEX_ESCAPE

- `SELECT apex_escape.js_literal('<script>alert(1);</script>') FROM dual;`
  - `\u003Cscript\u003Ealert(1);\u003C\u002Fscript\u003E'`
- `SELECT apex_escape.ldap_dn('Hans+Dieter') FROM dual;`
  - `Hans\+Dieter`
- `APEX_ESCAPE.SET_HTML_ESCAPING_MODE ( p_mode IN VARCHAR2);`
  - `p_mode = E(xtended) oder B(asic)`

Substitution Syntax (ab 5.0)

- **Vermeidung von Cross Site Scripting durch folgende Item-Referenzierung:**
  - `&P3_TEXT!JS.` → escape JavaScript (`apex_escape.js_literal`)
  - `&P3_TEXT!HTML.` (`apex_escape.html`) → escape HTML
  - `&P3_TEXT!ATTR.` → escape HTML Attribute (`apex_escape.html_attribute`)
  - `&P3_TEXT!RAW.` → VORSICHT: kein Escape!

Prüfung der Textlänge

- **Problem: Hacker könnten die Länge von Textfeldern im HTML-Code beliebig erhöhen**

- **Lösung #1: Prüfung durch Validation, ob die Textlänge den internen maximalen Wert überschreitet (und manipuliert ist)**

  - **Validation (Typ: No Rows returned)**

```
SELECT 1 FROM apex_050100.wwv_flow_step_items
WHERE flow_id=:APP_ID
AND name='P3_NAME'
AND length(:P3_NAME)>CMAXLENGTH
```

Prüfung der Textlänge

- **Lösung #2: Applikations-Prozess**

```
DECLARE
stmt VARCHAR2(32000);
BEGIN
FOR c IN (SELECT PAGE_ID,PAGE_NAME,ITEM_NAME,
ITEM_ELEMENT_WIDTH,ITEM_ELEMENT_MAX_LENGTH
FROM APEX_050100.APEX_APPLICATION_PAGE_ITEMS
WHERE application_id=:APP_ID AND page_id=:APP_PAGE_ID
AND ITEM_ELEMENT_MAX_LENGTH IS NOT NULL)
LOOP
stmt:='BEGIN IF length(v('''||c.item_name||''')) >
'||c.ITEM_ELEMENT_MAX_LENGTH||q'!  THEN
RAISE_APPLICATION_ERROR(-20500,'Hacker Angriff
entdeckt!!!'); END IF; END; !';
EXECUTE IMMEDIATE stmt;
END LOOP;
END;
```

Fehlerhafte Zugriffskontrolle

Hidden Items

- **Arten von Hidden Items:**
  - **Hidden:**
    - **wird auf Seite nicht angezeigt, kann aber Werte aufnehmen**
  - **Hidden & Protected:**
    - **Prüfung bei Submit, ob der Inhalt, der beim Seitenaufbau existierte, nachträglich geändert wurde**
- **Alternative: Application Items**
  - **liegen in der Applikation**
  - **werden auf keiner Seite angezeigt (Auslesen der Werte nicht möglich)**

Gefälschte Formularwerte

- **Manipulation von Itemwerten direkt in der Webseite (HTML-Code) möglich**
- **auch Prüfsummen verhindern dies nicht!**
  - **Fehlermeldung kommt bei der Weiterverarbeitung**
- **Lösungsansatz #1:**
  - **After Submit Prozess**
  - **Prüfung, ob Werte des Items zugelassen sind, wenn nicht:**

```
apex_authentication.logout(
p_app_id=>:APP_ID,p_session_id=>0);
```

Gefälschte Formularwerte

- **Lösungsansatz #2: Select List mit Hash Funktion**

- ```
  SELECT distinct job,
  job||':'||dbms_crypto.hash(utl_raw.cast_to_raw(job),4) r
  FROM emp
  ```

- **Validation:**
  - **Function Returning Boolean:**

```
IF
dbms_crypto.hash(utl_raw.cast_to_raw(substr(:P29_JOB_SEC,1,instr(
:P29_JOB_SEC,':')-1),4))=
substr(:P29_JOB_SEC,instr(:P29_JOB_SEC,':')+1) then
RETURN true;
ELSE
RETURN false;
END IF;
```

```
<select id="P29_JOB_SEC" class="selectlist" onchange="apex.submit('P29
B_SEC');" size="1" name="p_t01">
    <option value=""></option>
    <option value="MANAGER:3A2449232D248F19B8B5CC136B0AA712">MANAGER<
    /option>
```
```

Gefälschte Formularwerte

- **VORSICHT!**
  Validation und Prozesse (mit RAISE_APPLICATION_ERROR) brechen die weitere Verarbeitung der Seite ab, laden sie aber mit dem gefälschten Wert erneut!!
- **Verbesserung bringt 5.1 mit der Seiteneinstellung Reload on Submit: Only for success**

- **Lösungsansatz #3:**
  - **Werte bereits in der Report Query ausschließen:**

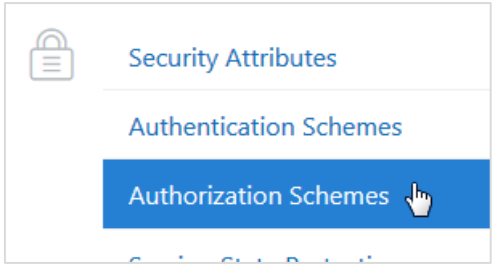
```
select ename, job, sal, deptno
from emp where deptno= :P8_DEPTNO
and deptno in (20,30)
```

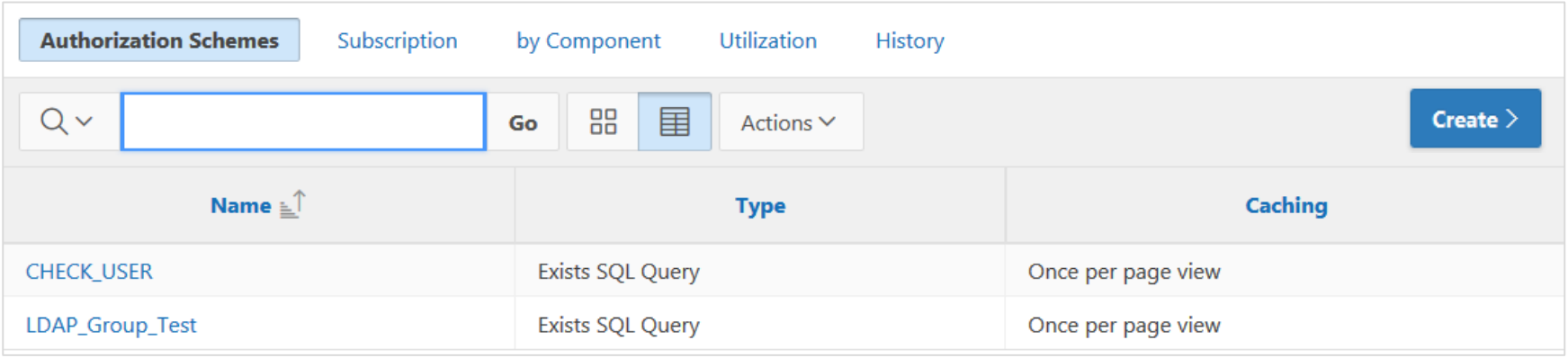
Authorization Scheme

- **Definition von Zugriffsberechtigungen für Seiten und Komponenten**
  - **u. a. Regionen, Reportspalten, Formulare, Navigationsmenü, Items**
- **erstellen Sie ein Authorization Scheme für die jeweiligen Objekte**
- **Beispiel: nur der Benutzer "Admin" soll besondere Rechte bekommen**

Authorization Scheme

- unter Shared Components → Authorization Schemes
- neue Prüfung anlegen



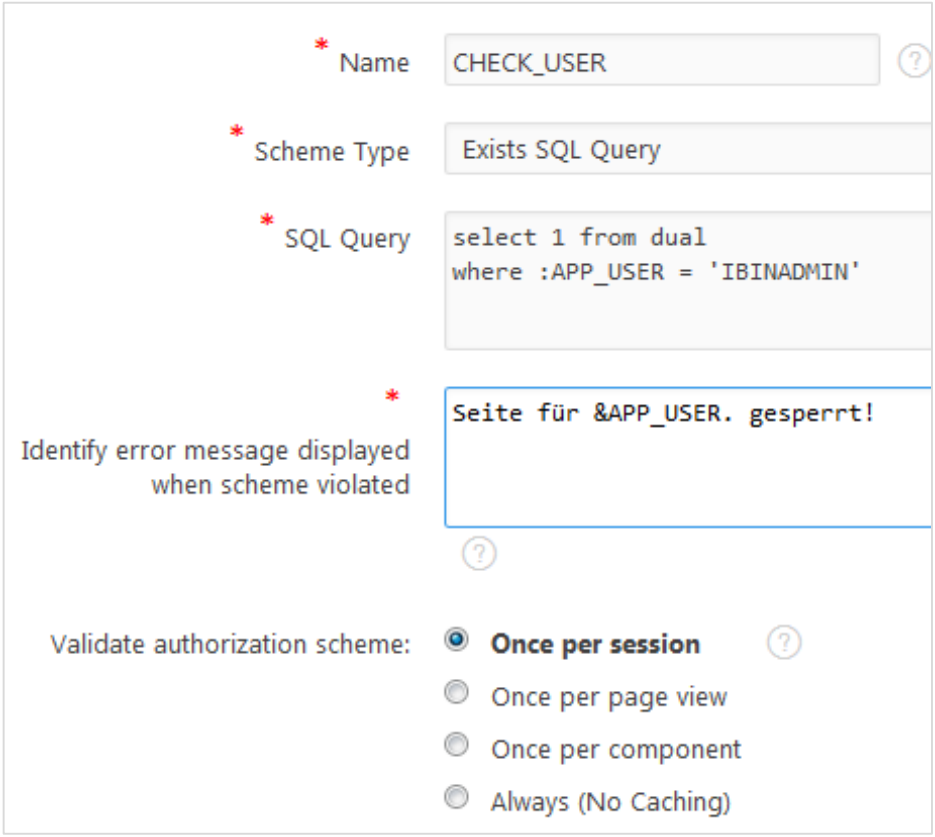


Authorization Scheme

- **Schema Type:**
  "Exists SQL Query"

- **SQL Query:**

```
SELECT 1 FROM dual
WHERE 'ADMIN'
= :APP_USER
```



Authorization Scheme

- **Benutzer:** `WHERE :APP_USER = 'OTTO'`
- **Zeit:**
  - `WHERE substr(to_char(sysdate,'DY', q'!nls_date_language='AMERICAN'!'),1,1) <>'S'`
- **Werte in einer Tabelle:** `WHERE value=1`
- **Server (Production/Test/Abnahme)**

Authorization Scheme

- **Beispiel #1: eigene Tabelle mit Benutzer und Berechtigungen**
  - ```
    CREATE TABLE scott.user_a_u_t_h (
    username VARCHAR2(30),
    r_name VARCHAR2(30));

    INSERT INTO scott.user_a_u_t_h VALUES
    ('MARCO','ADMIN');
    ```
  - **Schema Type: Exists SQL Query**

    ```
    SELECT 1 FROM scott.user_a_u_t_h
    WHERE username=:APP_USER
    AND r_name='ADMIN';
    ```
    ```

Authorization Scheme

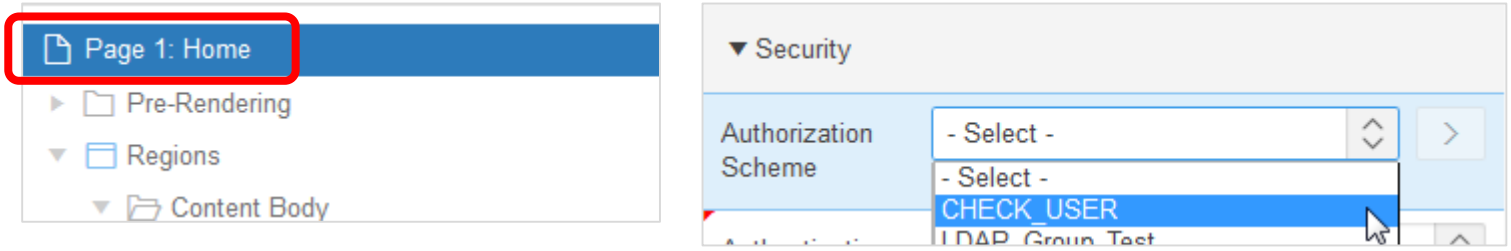
- **Beispiel #2: Tabelle mit Definition für Seitenzugriffe**
  - **Benutzer Marco darf die Seiten 1, 2, 4, 6 sehen**
  - **die Prüfung dazu lautet (SQL Exists):**

```
SELECT 1 FROM rechte
WHERE lower(name)=lower(lower(:APP_USER))
AND instr(seiten,':'||:APP_PAGE_ID||':') > 0;
```

ID	NAME	RECHT_NAME	SEITEN
1	Hans	Recht1	:0:
2	Marco	Recht2	:1:2:4:6:
3	Petra	Recht3	:2:3:4:
4	Sophia	Recht4	:1:4:10:20:

Authorization Scheme

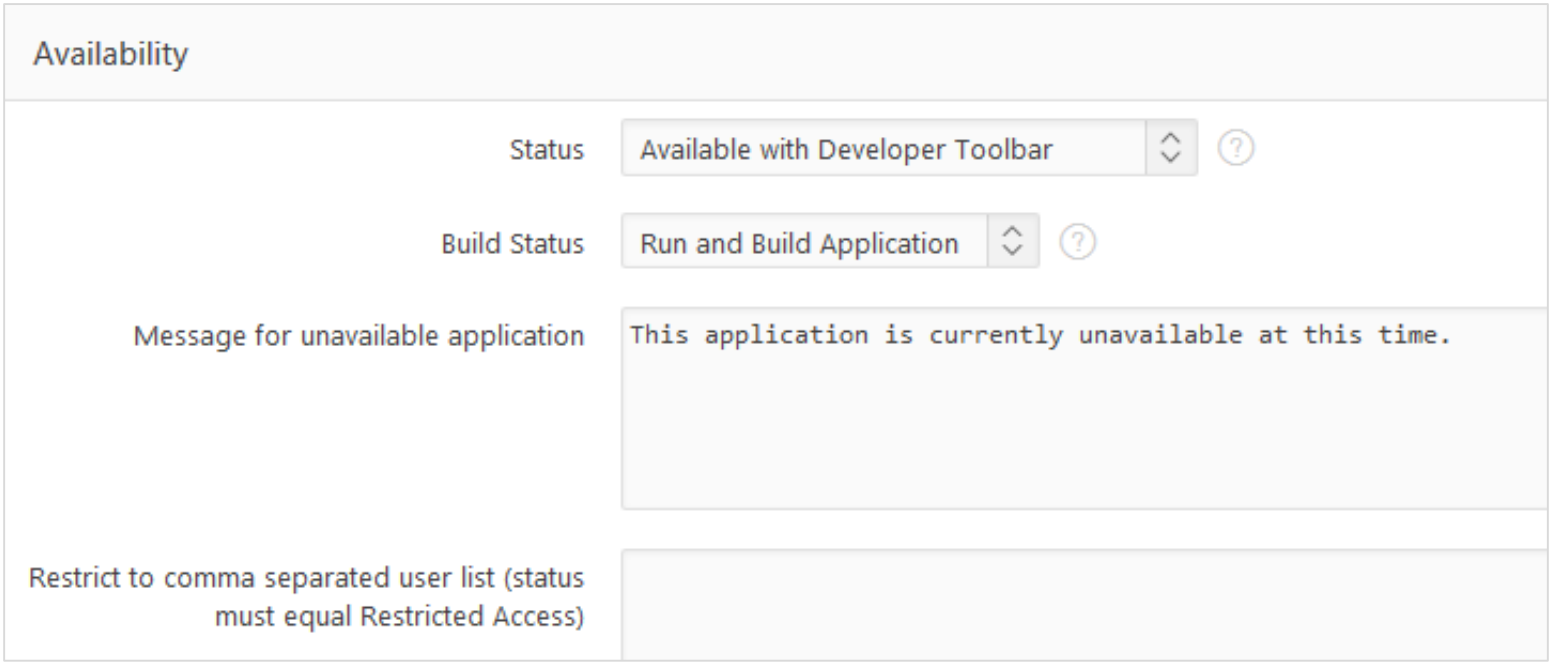
- **verknüpfen Sie das Authorization Scheme mit der gewünschten Komponente**
  - **Eigenschaften → Security → Authorization Scheme**



Sicherheitsrelevante Fehlkonfiguration

Verfügbarkeit der Applikation

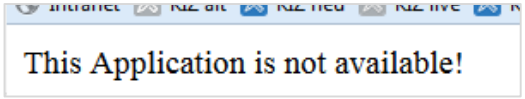
- **Shared Components → Definition**
- **Regelung der Applikations-Verfügbarkeit im Bereich Availability**



Verfügbarkeit der Applikation

- **ab APEX 5.1: APEX_UTIL.SET_APPLICATION_STATUS**
  - **Festlegen des Applikations-Status über SQL*PLUS**
  - **Beispiel:**

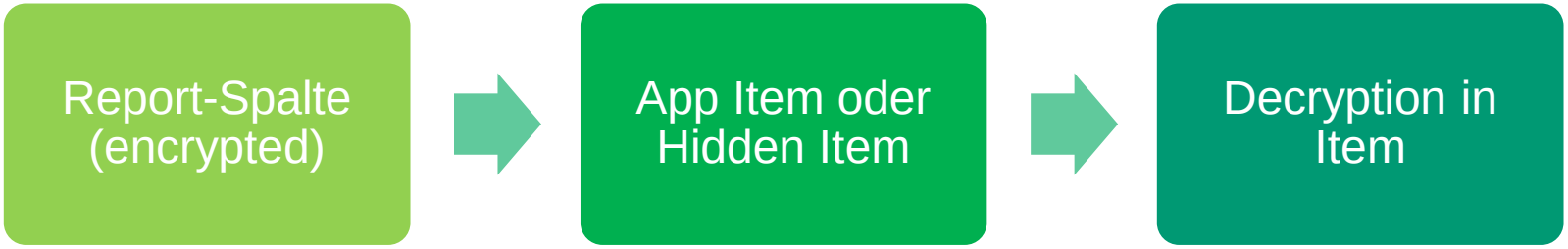
```
begin
    apex_util.set_workspace('APEX');
    apex_util.set_application_status(
      p_application_id     => 101,
      p_application_status => 'UNAVAILABLE',
      p_unavailable_value  => 'This Application
                               is not available!');
end;
/
commit;
```



Verlust der Vertraulichkeit sensibler Daten

Verschlüsselung von Items

- **Verschlüsselungsfunktion in der DB anlegen**
  - **in allen DB-Versionen ohne Zusatzoptionen verfügbar**




```
SELECT crypt(empno)
as empno
FROM emp;
```

```
:P1_EMPNO :=
RETURN crypt(
 :P1_EMPNO_ENC,'D'
);
```

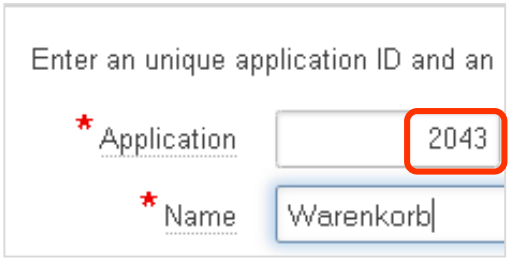
Verschlüsselung von Items

- **Beispiel einer einfachen Crypt-Funktion:**

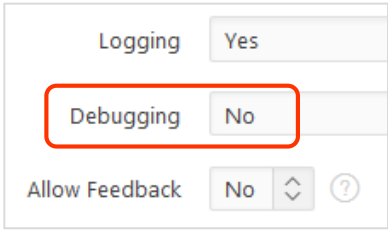
```
CREATE OR REPLACE FUNCTION crypt (
text           IN VARCHAR2,
key            IN VARCHAR2 DEFAULT 'MuniQSoft_Key',
cryptmode      IN VARCHAR2 DEFAULT 'E') RETURN VARCHAR2 IS
  p_key_raw   RAW(2048) := utl_raw.cast_to_raw(key);
BEGIN
  IF substr(upper(cryptmode),1,1)='E' THEN
    RETURN (sys.dbms_crypto.encrypt(
     src => sys.utl_raw.cast_to_raw(text),
     typ => sys.dbms_crypto.des_cbc_pkcs5,
     key => p_key_raw));
  ELSE
   RETURN (utl_raw.cast_to_varchar2(sys.dbms_crypto.decrypt(
     src => text,
     typ => sys.dbms_crypto.des_cbc_pkcs5,
     key => p_key_raw)));
  END IF;
END;
/
```

APEX Security Tipps

- sicherheitskritischen PL/SQL Code in Package auslagern und wrappen
- eigene Tabelle (BLOB/CLOB Spalten) für Bilder und Dokumente verwenden
  - statt der "öffentlichen Tabelle" wwv_flow_file_objects$
- für neue Applikationen eine Applikationsnummer > 1000 vergeben (4000-9000 werden von Oracle genutzt)



APEX Security Tipps



- **Debugging ausschalten**
  - **Shared Components → Definition → Properties**
  - **Prüfung, ob Debugging aktiviert ist:**

```
SELECT name,owner, decode( f.APPLICATION_TAB_SET,
1,'Allowed',0,'Not Allowed','Allowed') debugging
FROM wwv_flows f
WHERE security_group_id<>10
```

APEX in einem eigenen Tablespace installieren!

- **bei fertiger Applikation: Editiermöglichkeit abschalten**
  - **Shared Components → Definition → Availability → Build Status → Run Application Only**

APEX Runtime Installation

- in reiner Produktions- oder Testumgebung: nur Runtime Installation verwenden

- diese kann nachträglich installiert werden:
  - @?/apex/apxdevrm.sql

Muniqsoft GmbH

- **Tätigkeitsbereiche:**
  - **Oracle IT-Consulting & Services**
  - **Oracle Remote Support und Rufbereitschaft**
  - **Oracle Schulungen (SQL, PL/SQL, DBA, APEX, … - gerne auch Inhouse)**
  - **Software-Lösungen**
  - **Oracle Lizenzen**

Muniqsoft GmbH
Schulungszentrum
Grünwalder Weg 13a
82008 Unterhaching
Tel.: 089 / 679090 40

Muniqsoft GmbH
IT-Consulting & Support
Witneystr. 1
82008 Unterhaching
Tel.: 089 / 6228 6789 0