



# Installationsprozess für APEX Anwendungen

Oliver Lemm

Düsseldorf, 15.09.2015

# Agenda

**1. Import & Export der APEX Anwendung**

**2. Installation der Datenbank**

**3. Datapump**

**4. Statische Dateien**



# Import & Export der APEX Anwendung

# Export/Import - Einzelseiten vs. Gesamt

Einzelseiten	Gesamtexport
Änderungen einzeln exportieren & versionieren	Immer alle Änderungen zusammen versionieren
Problem bei Änderungen an zentralen Komponenten zusammen mit Seiten	Es können keine Änderungen vergessen werden
Import nur mit Manipulation der Security ID / Workspace ID in anderen Workspace möglich	Entwicklung innerhalb APEX auch nur an gesamter Anwendung möglich
Aufteilung nur mittels Kommandozeile möglich	Import über Oberfläche und Kommandozeile möglich
Aufwändiger Vergleich aller einzelnen Objekte	Ab APEX 5 Verbesserungen im Export

# Anwendungsexport

- über Browser
- mit SQL Developer
- mittels Kommandozeile
  - apex\utilities\readme.text -> Infos
  - apex\utilities\oracle\apex\APEXExport.class
- Weitere Informationen
  - <http://www.oracle.com/webfolder/technetwork/de/community/apex/tipps/export-script/index.html>

# Anwendungsexport

- export.bat

```
@echo off
set ORACLE_HOME=C:\oracle\product\11.2.0\dbhome_1
REM APEX HOME auf das Verzeichnis setzen, in das APEX ausgepackt wurde.
set APEX_HOME=D:\apex_5\apex

set CLASSPATH=%ORACLE_HOME%\oui\jlib\classes12.jar
set
CLASSPATH=%CLASSPATH%;%APEX_HOME%\utilities;%ORACLE_HOME%\jdb
c\lib\ojdbc6.jar
set CLASSPATH=%CLASSPATH%;.

java.exe oracle.apex.APEXExport %*
```

# Anwendungsexport

- Kapselung mittels Datei um das Format was nötig ist zu definieren

```
@echo off
Rem Pruefen ob alle Parameter gefuellte sind
if %1!==! goto usage
if %2!==! goto usage
if %3!==! goto usage
if %4!==! goto usage

call 14_export.bat -db %1 -user %2 -password %3 -applicationid %4

goto ende
:usage
echo "fehlerhafte Parameter! Korrekte Parameter sind: <SourceTNS> <SourceUser>
<SourcePw> <APP_ID>,"
echo "Beispiel: export.cmd host:port:sid dbuser dbpw 100"
:ende
```

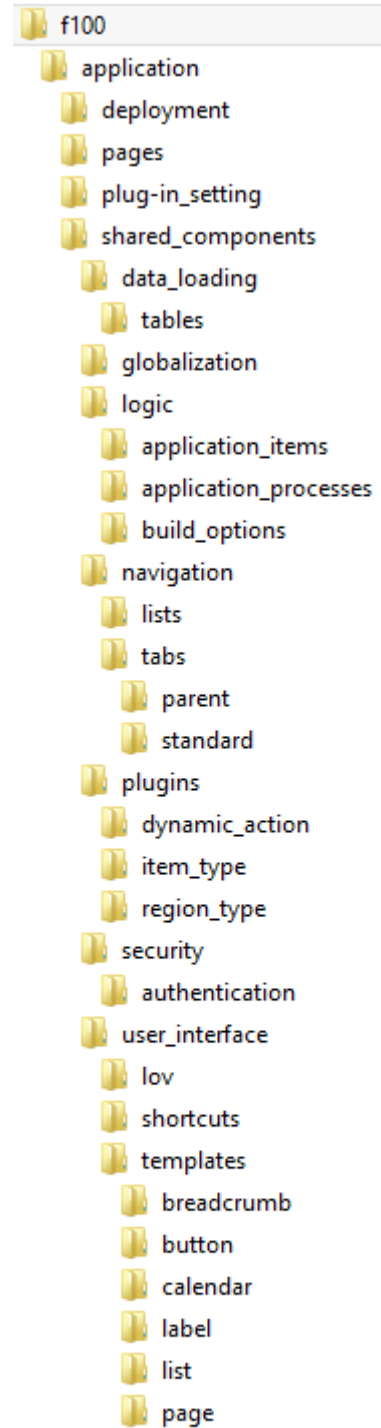
# Anwendungsexport Beispielaufruf

- `export.cmd localhost:1521:orcl expertenseminar_2015 expertenseminar_2015 100`
- Ergebnis:  
Exporting application 100  
Completed at Mon Aug 31 14:40:22 CEST 2015
- `f100.sql`



# Anwendungsexport Hinweise

- Version von installierter APEX Version muss gleich oder kleiner der Exportfunktion die aufgerufen wird sein.
- Splitter Aufruf gleich, nur anstatt „APEXExport“ „APEXExportSplitter“ aufrufen.
- splitter.bat f100.sql  
Processing:f100.sql
- Plugins einzeln verfügbar



# Anwendungsimport

- Kommandozeilenimport

```
begin
  apex_application_install.set_workspace_id(3138423103500405);
  apex_application_install.set_application_id(199);
  apex_application_install.set_schema('{<i>DB_SCHEMA</i>}');
  apex_application_install.set_offset(2763898718743);
end;
/
```

- set\_workspace => nötig bei Import in anderen Workspace als der Export
- set\_application => nötig falls andere ID vergeben werden soll
- set\_schema => Ziel DB Schema
- set\_offset => Bei Installation in gleichen Workspace aber anderer ID nötig

# Anwendungsimport

- Anwendungen im Workspace
  - mit gleichem ALIAS => ALIAS umbenennen
    - Absicherung dass der ALIAS nur einmal vorkommt im Workspace
  - mit gleichem ALIAS => Deaktivieren (alte Version)
  - Importieren unter „Versions-ID“ als Nummer
    - Dadurch kann man die Anwendungen in alten Versionen zum Vergleich heranziehen

# Anwendungsimport

- Prüfung der installierten APEX Version

```
select version_no from apex_release
```

- Prüfung der installierten Sprachpakete

```
select t.translated_application_id
       ,t.translated_app_language
       ,nvl2(app.name,'Yes', 'No') installed
from (select m.translated_application_id,m.translated_app_language from
apex_application_trans_map m
      where m.translated_application_id between 4001 and 4009
      union
      select 4000 translated_application_id,'en' translated_app_language from dual) t
left join apex_040200.www_flows app on app.id = t.translated_application_id
order by translated_app_language
```

# Anwendungsimport

- Build Options

- Supporting Objects

- Validation

- Prüfung auf vorhandenen Anwendungsversion

```
select a.version from apex_applications a where a.ALIAS = <ALIAS>
```

- Upgrade

- Aktualisierung der Version in eigenen Tabellen oder Eintrag in LOG bei erfolgreichem Import

A photograph of a woman with long, wavy blonde hair and blue eyes, wearing a light-colored blazer over a dark top with a large blue bow. She is looking towards the right of the frame. In the background, another person with blonde hair is seen from behind, and a computer monitor is visible. The scene is set in a professional office environment with warm lighting.

# Installation der Datenbankobjekte

# Versionsprüfung

- Versionsnummer
  - 1.2.3.4
    - 1 – Die erste Ziffer ändert sich nur bei einer Neuentwicklung oder grundlegenden Überarbeitung
    - 2 – Die zweite Ziffer ändert sich nur bei Änderungen innerhalb der Datenbank
    - 3 – Die dritte Ziffer ändert sich nur bei Änderungen in der APEX Anwendung
    - 4 – Je nach Art der Auslieferung kann die 4.te Ziffer als Hotfixnummer genutzt werden.
  
- Prüfung der Version anhand zentraler Tabelle
  - Fehlermeldung bei nicht korrekter Vorversion
  
- Prüfung anhand der installierten APEX Anwendung

# Initialinstallation - Tablespace

- Parametrisierte Erstellung von Tablespaces

- User/Schema &1
- Verzeichnis der Datafiles &2

- Verzeichnis der Datafiles liegt standardmäßig im Oracle Installationsverzeichnis unter oradata\SID - Beispiel: c:\oracle\oradata\orcl

```
PROMPT CREATE TABLESPACE &1  
CREATE TABLESPACE &1  
DATAFILE '&2&1..dbf' SIZE 100M  
AUTOEXTEND ON NEXT 100M MAXSIZE 1000M;
```

- eigenen Tablespace für

- Indices
- Blob Spalten bzw. Tabellen mit Blob Spalten



# Initialinstallation – User/Schema

- User/Schema & Tablespace zwecks einfacher Handhabung mit gleichem Namen versehen
  - Passwort &3

```
PROMPT User &1 erstellen  
CREATE USER &1 IDENTIFIED BY &3  
DEFAULT TABLESPACE &1  
TEMPORARY TABLESPACE temp  
QUOTA UNLIMITED ON &1;
```

- Temporary Tablespace optional auch anders wählbar
- An Quota denken!
- Falls benötigt Zugriff auf weitere Tablespaces (von Schnittstellen) vergeben

```
ALTER USER &1 QUOTA UNLIMITED ON &4;
```

# Initialinstallation - Grants

- Allgemeine Grants

- Am besten ohne „any“ arbeiten

```
grant create procedure to &1;  
grant create public synonym to &1;  
grant create sequence to &1;  
grant create session to &1;  
grant create table to &1;  
grant create trigger to &1;  
grant create type to &1;  
grant create view to &1;
```

- Ggf. auch Public Synonyme wieder entfernen

```
grant drop public synonym to &1;
```

# Initialinstallation - Grants

- spezifische Rechte

- Verschlüsselung `grant execute on dbms_crypto to &1;`

- Scheduler oder Jobs anlagen `grant execute on dbms_scheduler to &1;`  
`grant create job to &1;`

- auf Entwicklung Debug Rechte `grant debug connect session to &1;`

- Zugriff auf Directory `GRANT READ,write ON DIRECTORY dbexport TO &1;`

- Volltextsuche `grant execute on CTXSYS.CTX_DDL to &1;`

# Skripte

- Keine Schemabezeichner

- Entweder mittels Parameter oder Zuordnung durch Ordnerstruktur und später Variable je nach Umgebung definierbar

```
set sqlblanklines on;
```

- SQL Plus Probleme

- Leerzeichen in Views

```
set define off;
```

- &-Zeichen (ohne als Parameter)

- Maximale Zeilenlänge von 2500 Zeichen

# Skripte

- PLSQL Objekte mit force
  - Objekt existiert immer und ist als invalid zu sehen, falls es fehlschlägt
- DML
  - Abfrage mit gleicher „where-Bedingung“ zur Überprüfung
- DDL
  - Abfrage auf user\_objects nach Anpassung
- Exception Handling

# Skripte

- **Keywörter verwenden.**
  - `Dbms_output.put_line` / `PROMPT`
    - `INFO` -> erfolgreiche Durchführung eines Skriptes
    - `WARNING` -> Skript wurde bereits vorher ausgeführt
    - `ERROR` -> Skript ist fehlgeschlagen oder wurde nicht korrekt ausgeführt
- **Body & Spec zusammen**
- **PLSQL Abschluss mit /**

# Skripte – re-run-Fähigkeit - DML

- Problem DML Skripten
  - bei zweitem oder mehreren Durchläufen beim Kunden
  - bei Testumgebung damit diese nicht jedes Mal neu aufgesetzt wird
- Select count verwenden anstatt merge
- Update mit Where Bedingung
  - Beides führt DML aus.
- DML in Verbindung mit Objekten die verändert werden
  - Conditional Compilation

```
$IF THEN  
...  
$END IF;
```

# Skripte – re-run-Fähigkeit – DDL

- Problem bei DDL Skripten
  - Objekt existiert bereits bzw. ist bereits geändert
- Lösungen
  - Abfrage auf user\_objects
  - Nutzung von execute Immediate



Prompt insert into USER\_MESSAGES 1438

declare

l\_count number;

c\_msg\_nr constant number := 1438;

l\_action varchar2(32767) := 'INSERT INTO USER\_MESSAGES ' || c\_msg\_nr;

begin

select count(1) into l\_count from user\_messages u where u.umsg\_message\_nr = c\_msg\_nr;

if l\_count = 0

then

insert into user\_messages(umsg\_message\_nr, umsg\_language, umsg\_message) values (c\_msg\_nr, 'DE', ,Dies ist ein beliebiger Text.);

dbms\_output.put\_line('INFO: ' || l\_action || ' ist ausgefuehrt worden!');

else

dbms\_output.put\_line('WARNING: ' || l\_action || ' wurde schon ausgeführt');

end if;

select count(1) into l\_count from user\_messages u where u.umsg\_message\_nr = c\_msg\_nr;

if l\_count = 0

then

dbms\_output.put\_line('ERROR: ' || l\_action || ' ist fehlgeschlagen!');

end if;

exception

when others then

dbms\_output.put\_line('ERROR: ' || l\_action || ' konnte NICHT ausgefuehrt werden!' || substr(sqlerrm,0,200));

end; /

# Grants & Synonyme

- pro Schema in einer Datei definiert
- Berechtigung und übergreifende Vorgaben so ausgelagert
- Schema mittels Parameter
- Prozedur zum Anlegen & Löschen nutzbar



synonyms.sql



grants.sql

# Abhängigkeiten

- Erstellung neuer Objekte
  - pro Objekt eine Datei
- DML & DDL trennen
- Beispiele:
  - Neue not NULL Spalte
  - Datenmigration
  - Verwendung von Objekten durch weitere Objekte (Tabelle in Package etc.)

# Skripte - Recompile

- Recompile (mit Berechtigung auf DBMS\_UTILITY)

```
EXEC DBMS_UTILITY.compile_schema(schema => 'SCOTT');
```

- Durchlaufen aller Objekte im Schema die invalid sind

```
SELECT OBJECT_TYPE, OBJECT_NAME  
FROM   user_objects  
WHERE  status != 'VALID'  
AND    object_type in ('PACKAGE','PACKAGE BODY','FUNCTION','PROCEDURE',  
, 'TRIGGER','VIEW','TYPE BODY','TYPE')  
order by OBJECT_TYPE, OBJECT_NAME;
```

- Mittels dbms\_sql.execute neu kompilieren
  - Ggf. mehrfach recompilieren notwendig



recompileall.sql

# Abhängigkeiten – allgemeine Reihenfolge

1. Schema

2. DDL

3. PLSQL

1. Type

2. Restliche PLSQL Objekte

4. DML

- Innerhalb jedes Bereichs Sortierung nach Name
- Vor jedem DML ein Recompile durchführen
- Zusätzlich händische Definition von Abhängigkeiten
  - 1. Spalte Anfügen (DDL)                      2. Spalte befüllen (DML)
  - 3. Spalte NOT NULL (DDL)

# Abhängigkeiten - Anwendung

Willkommen: OLEMM Drückans...

Tickets   Objekte   Objekt - Ticket - Zuordnungen   Versionen   **Vorgänger**   DDL Änderungen   Implikationsanalyse   QS - Objekte   QS - Personen   QS - Person - Objekt - Zuordnungen

Vorgänger Abbrechen Löschen Weiterleiten nicht in Vorgängerlogik zugeordnete Objekte des

Version  Schema  Ticket

<input type="checkbox"/> Objekt	Vorgänger	angelegt am	angelegt von	geändert am	geändert von
<input type="checkbox"/> rr_bereiche_rollen_zuordnungen_update_1172.sql	rr_rollen_update_1172.sql	21.08.2015 17:51:38	OLEMM	(null)	(null)
<input type="checkbox"/> rr_gruppen_rollen_zuordnungen_update_1172.sql	rr_rollen_update_1172.sql	21.08.2015 17:51:54	OLEMM	(null)	(null)

1 - 2 Zeile hinzufügen

Reihenfolge der Scripte - Phase 1

Schema	Installationsdatei	Verzeichnis	Datei	Bugzilla	Jira	Ebene
FPP	FPP_1_DML	DML	rr_rollen_update_1172.sql			0
FPP	FPP_1_DML	DML	rr_bereiche_rollen_zuordnungen_update_1172.sql			1
FPP	FPP_1_DML	DML	rr_gruppen_rollen_zuordnungen_update_1172.sql			1

1 - 3

Reihenfolge der Scripte - Phase 2 (keine Vorgänger)

Schema	Install File Name	Verzeichnis	Objekt	Bugzilla	Jira
EXP	EXP_2	DDL	exp_anteilsscheine_add_cols_1196.sql		
	EXP_2	DDL	exp_anteilsscheine_drop_col_1044.sql		
	EXP_2	DDL	exp_anteilsscheine_hist_add_cols_1196.sql		



# Datapump

# Datapump

- Impdp und expdp Bestandteil der Datenbank
  - Import & Export mit beliebigen Kriterien
    - Mit oder ohne Daten
    - Welches Schema
    - Remapping von Schema, Tablespace etc.
    - Kapselung von „Charset“ Problemen
    - Mit oder ohne Grants
    - Export & Import unabhängig
    - Ausschluss von Objekten
- Imp und exp „alt“



# Datapump – Voraussetzungen & Hinweise

- Zugriff auf Verzeichnis des Servers
- Anlegen eines Directory mit Lesen (import) und Schreiben (export)
  - Für das Erzeugen einer LOGFILE werden auch beim Import Schreibrechte benötigt
- Pro Schema eine Datei erzeugen
  - Parallele Verarbeitung möglich
- Bei Import & Export immer mit Logfile arbeiten
- Zwecks Übersicht parfile benutzen
  - Parfile auch sinnvoll bei unterschiedlichen Umgebungen
  - Oder bei Import von Kundendaten zum Ausschluss von Datenbanklinks etc

# Datapump - Import

- Beispiel:

- impdp user/password@orcl  
dumpfile=<ENVIRONMENT>\_<SCHEMA>\_<VERSION>.dmp  
directory=DBEXPORT  
LOGFILE=<ENVIRONMENT>\_<SCHEMA>\_<VERSION>.log  
REMAP\_TABLESPACE=OLD\_TABLESPACE:NEW\_TABLESPACE

- Probleme

- Datei pro Server auf dem Server nötig
- XMLTypes können Probleme verursachen
- DBMS\_Scheduler kann Probleme erzeugen
- Fehlerliste bei Import teils sehr lang (viele Warnungen)



# Statische Dateien

# Statische Dateien

- Dateitypen
  - Grafiken
  - CSS
  - JavaScript
- Minify bei JavaScript nutzen
- Entwicklung nicht lokal möglich
- Auch auf Charset achten
- Kapselung von Dateien in Templates mittels substitutions

# Statische Dateien

- In Anwendung
  - Einfacher zu importieren & exportieren
  - Mit APEX 5 direkt integriert beim Export & Import
  - Caching Problematik mit APEX 5 geringer (Versionierung)
  - Performance oft vernachlässigbar
- Auf Betriebssystem
  - Dateien müssen außerhalb von APEX bereitgestellt werden
  - Automatisierte Bereitstellung schwierig
  - Performance besser – Caching & Kompression über Apache besser
  - Pflege einfacher
  - APEX Anwendung selber ist kleiner und besser versionierbar



# Vielen Dank.

**MT AG**

Balcke-Dürr-Allee 9  
40882 Ratingen

Telefon: +49 (0) 21 02 309 61-0  
Telefax: +49 (0) 21 02 309 61-101

E-Mail: [info@mt-ag.com](mailto:info@mt-ag.com)  
[www.mt-ag.com](http://www.mt-ag.com)