APEX Application Lifecycles
|> Managing the Change

Dietmar Aust
Opal-Consulting, Germany / Cologne
www.opal-consulting.de

OPAL Consulting

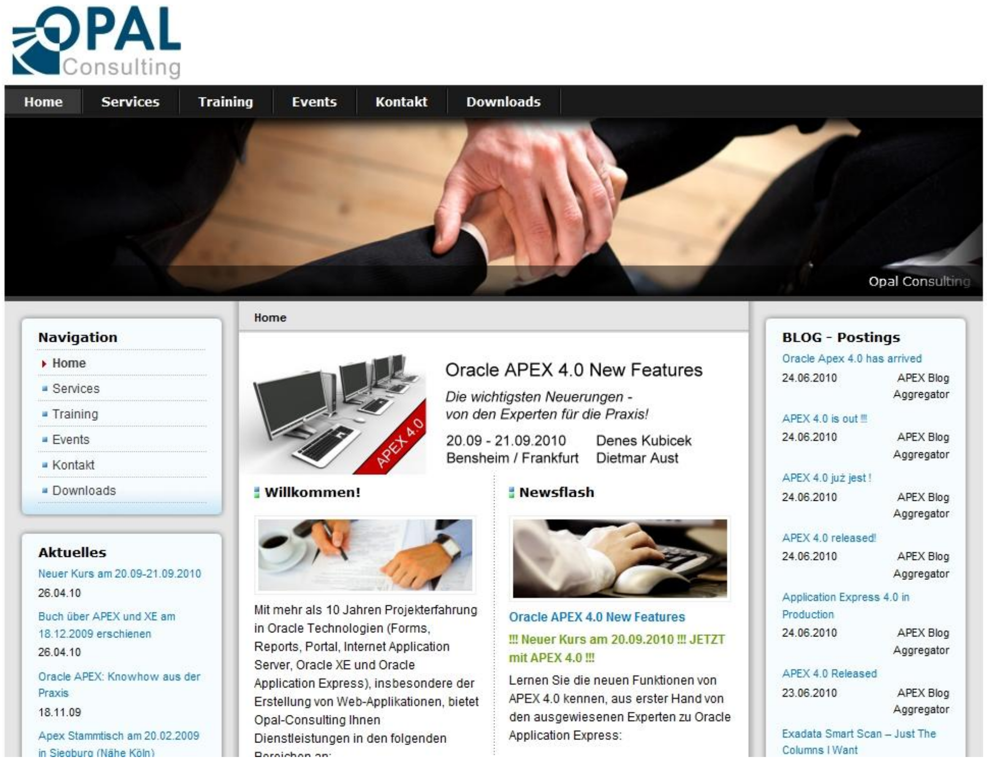
Version 1.1 / 28.10.2010 / D. Aust

Introducing Opal Consulting

- Building Oracle based Web Applications since 1997
  - Portal, Forms, Reports, OWA Toolkit, now APEX!
- Dipl.-Inform. Dietmar Aust, Freelance Consultant
  - Master's Degree in Computer Science (MSCS)
- 1997-2000: Consultant at Oracle Germany
- Since 09/2000: Freelance Consultant
- Blog: http://daust.blogspot.com/
- Regular presenter at ODTUG, DOAG

Introducing Opal Consulting

- Website (www.opal-consulting.de) built on APEX and Oracle XE as a proof of concept



Navigation

- Home
- Services
- Training
- Events
- Kontakt
- Downloads

Aktuelles

Neuer Kurs am 20.09-21.09.2010
26.04.10

Buch über APEX und XE am 18.12.2009 erschienen
26.04.10

Oracle APEX: Knowhow aus der Praxis
18.11.09

Apex Stammtisch am 20.02.2009 in Siegburg (Nähe Köln)

Home

Oracle APEX 4.0 New Features

Die wichtigsten Neuerungen -
von den Experten für die Praxis!

20.09 - 21.09.2010 Denes Kubicek
Bensheim / Frankfurt Dietmar Aust

Willkommen!

Mit mehr als 10 Jahren Projekterfahrung in Oracle Technologien (Forms, Reports, Portal, Internet Application Server, Oracle XE und Oracle Application Express), insbesondere der Erstellung von Web-Applikationen, bietet Opal-Consulting Ihnen Dienstleistungen in den folgenden Bereichen an:

Newsflash

Oracle APEX 4.0 New Features

!!! Neuer Kurs am 20.09.2010 !!! JETZT mit APEX 4.0 !!!

Lernen Sie die neuen Funktionen von APEX 4.0 kennen, aus erster Hand von den ausgewiesenen Experten zu Oracle Application Express:

BLOG - Postings

Oracle Apex 4.0 has arrived
24.06.2010 APEX Blog Aggregator

APEX 4.0 is out !!!
24.06.2010 APEX Blog Aggregator

APEX 4.0 już jest !
24.06.2010 APEX Blog Aggregator

APEX 4.0 released!
24.06.2010 APEX Blog Aggregator

Application Express 4.0 in Production
24.06.2010 APEX Blog Aggregator

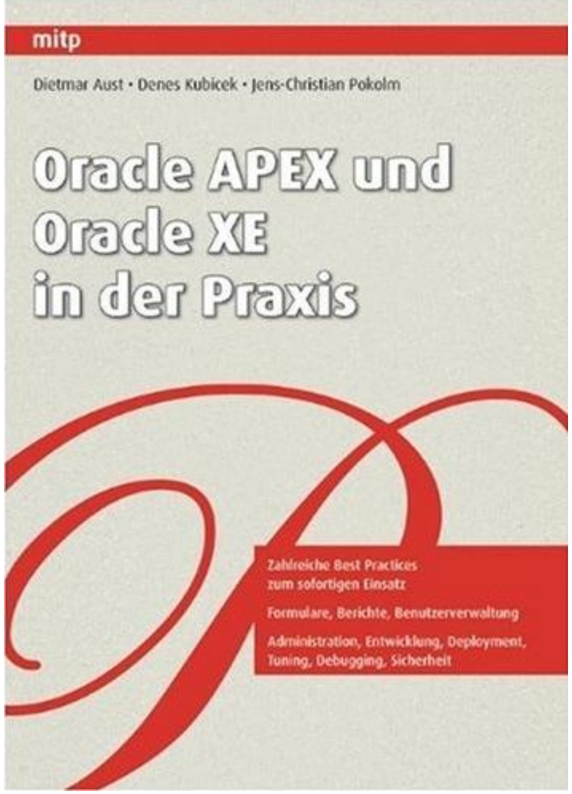
APEX 4.0 Released
23.06.2010 APEX Blog Aggregator

Exadata Smart Scan – Just The Columns I Want

Introducing Opal Consulting

- Giving APEX trainings regularly in Germany together with Denes Kubicek

- Co-authored a book in German on APEX and Oracle XE best practices
  - http://apex-xe-praxis.de/



mitp

Dietmar Aust • Denes Kubicek • Jens-Christian Pokolm

Oracle APEX und
Oracle XE
in der Praxis

Zahlreiche Best Practices
zum sofortigen Einsatz

Formulare, Berichte, Benutzerverwaltung

Administration, Entwicklung, Deployment,
Tuning, Debugging, Sicherheit

Why are you here?

- This topic is NOT SEXY, NOT COOL
- You don’t get any FREE BEER ;)
- Why are you here?
- It saves TIME, MONEY … and your NERVES
- It increases the quality of your software
- It increases the maintainability and transparency of your software
- Please spend more time with your family ;)

Why are you here?

- Everybody needs an Application Lifecycle Management (ALM) but nobody takes the time ... the costs are high ... and well hidden ;)
  - A good ALM / configuration management takes a lot of effort != trivial!

Why are you here?

- No standards in the Oracle world, the Java world seems to agree on Maven
  - Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
  - http://maven.apache.org
  - ANT (popular build tool)
  - Maven could be seen as (“ant with convention over configuration”)

Agenda

- The background project
- Challenges faced
- The components of a successful application lifecycle management (ALM)
- Implementation from A-Z with demo

OPAL Consulting

The Background Project

- Mainly developed at the German Telecom Shops for the Shop Management Application called „Spots".
- Concept evolved over the last three years
- Works really well (for us)
- Good understanding of the relevant issues

The Background Project

- Complexity
  - 2-3 Developers
  - Development since 05/2007
  - Single schema SHDB_200
  - **Tablespace names identical on dev, test, prod**
  - 200 Tables, 100 Packages, 3000 database objects
  - APEX application with 140 pages
- Releases
  - Usually four releases per year
  - In average 100 (50-250) changed / newly created objects
  - Usually four internal revisions with the test team

Technical Challenges

- Concurrently changing database packages
  - Overwriting changes
- Current state / version unclear
  - installation files
  - application files: APEX, WAR, JS, CSS
  - application version in database instance (dev, test, production)?
  - was script xyz (DDL or DML) already run on instance test?

Non Technical Challenges

- Which requirements were implemented in this patch?
  - => Release Notes!
- When did we install which version?
  - The problem we have now … could we have already fixed it?
- Keeping up with the documentation
  - Delta Release was fine ... complete system spec was outdated

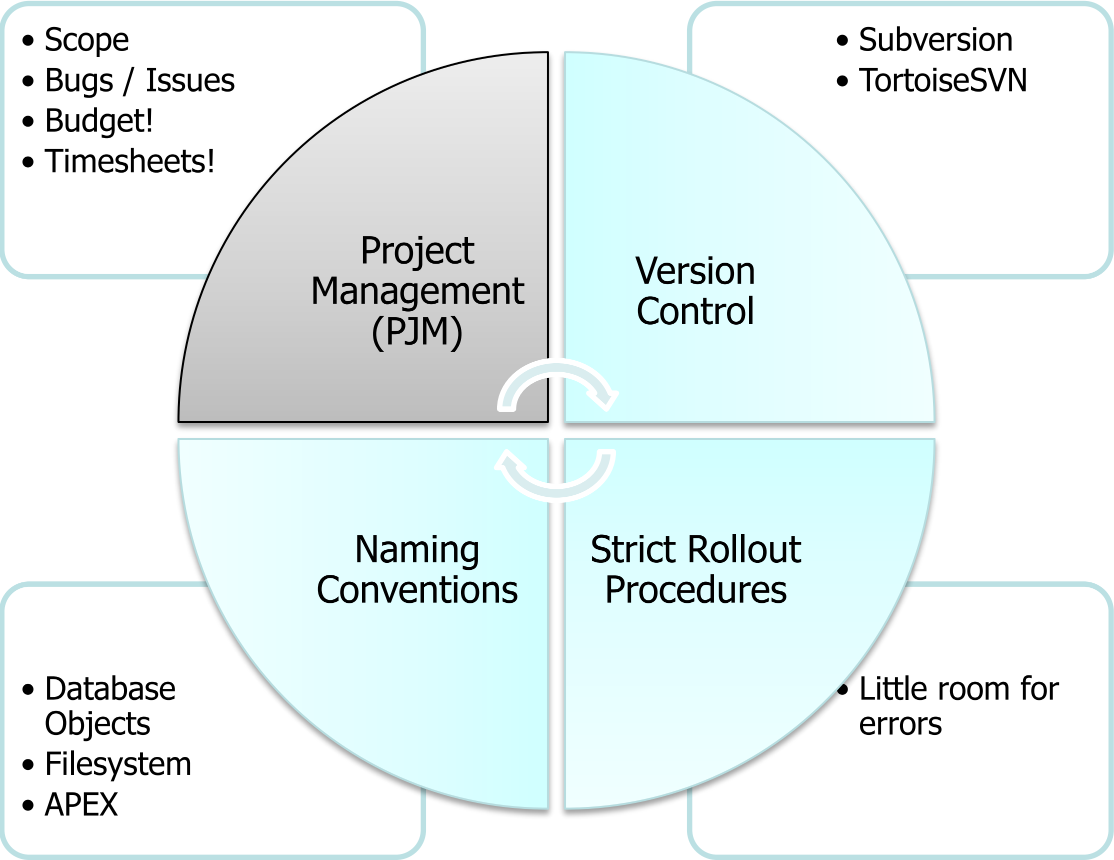
Non Technical Challenges

- Needed answers to the typical PJM related questions to organize our daily work
- At the end of the day the developer is asked
  - What is the scope of the release?
    - Must / Should / Could / Won’t
  - Who is doing what?
  - What are the open issues / questions?
  - What did we estimate for each task?
  - Are we in time/budget?
  - We are running late … why?

The Approach

- => We needed an approach that was
  - Simple (only few rules, KISS)
  - Transparent (don’t make me think, the good RTFM error ;)
  - Consistent
  - Safe (don’t make more mistakes than necessary ;)
- Which “objects” should be considered?
  - Files
  - Database objects
  - APEX objects
  - Documentation
  - Requirements

The Components Of A Successful ALM



- Scope
- Bugs / Issues
- Budget!
- Timesheets!

- Subversion
- TortoiseSVN

Project Management (PJM)

Version Control

Naming Conventions

Strict Rollout Procedures

- Database Objects
- Filesystem
- APEX

- Little room for errors

OPAL Consulting

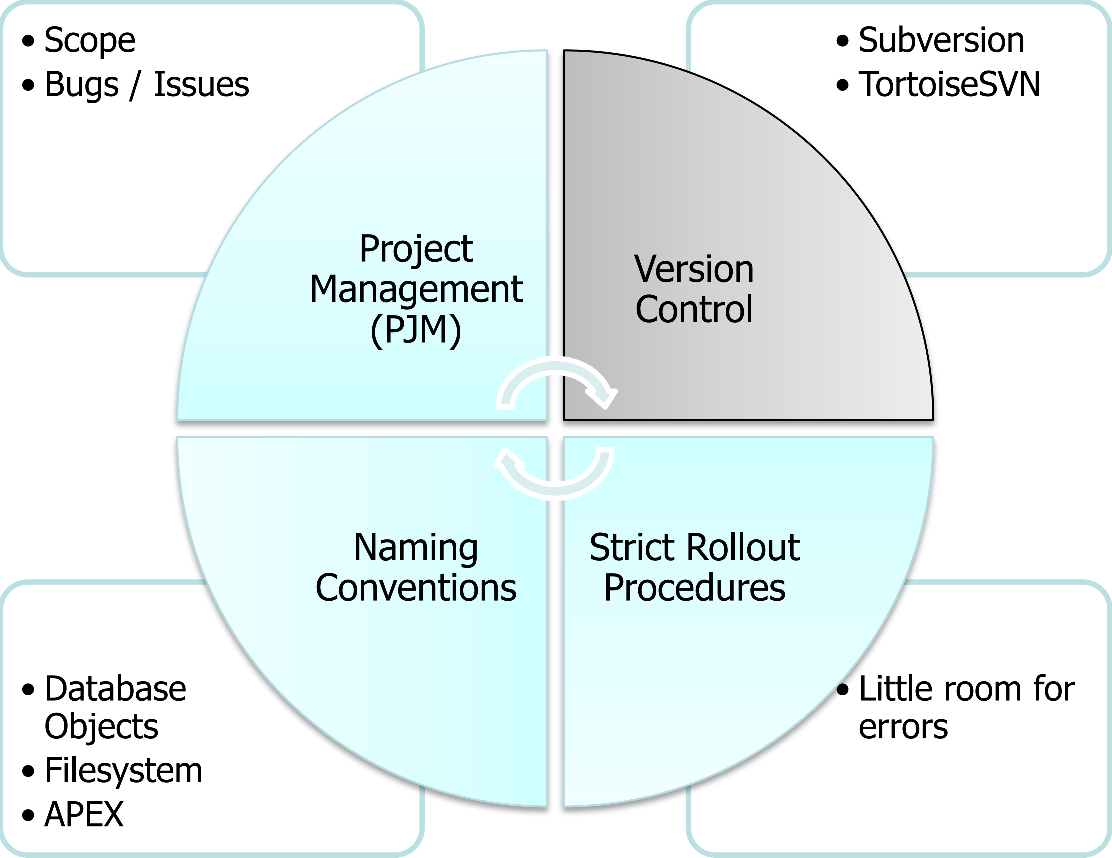
The Components: PJM

- PJM application for managing the scope
  - Lightweight, as a tool for developers
- Relevant features
  - Which requirements are included in version xyz?
    - Generate a list easily
    - We work on the requirements and spec online in the tool when meeting with the customer
  - Budgets, estimations and timesheets integrated
  - Milestones
  - Open issues, easy to generate a list for the next meeting with the client

The Components: PJM

- Demo:
  - Show PJM application
    - A tool for developers
    - Upcoming milestones
  - Requirements
    - Subrequirements, estimations, etc.
    - Open questions for the next customer meeting
    - Implemented features in Revision 1?
  - Timesheets
    - Unplanned activities
    - Timesheet export for the customer

The Components: Version Control



- Scope
- Bugs / Issues

Project Management (PJM)

- Subversion
- TortoiseSVN

Version Control

Naming Conventions

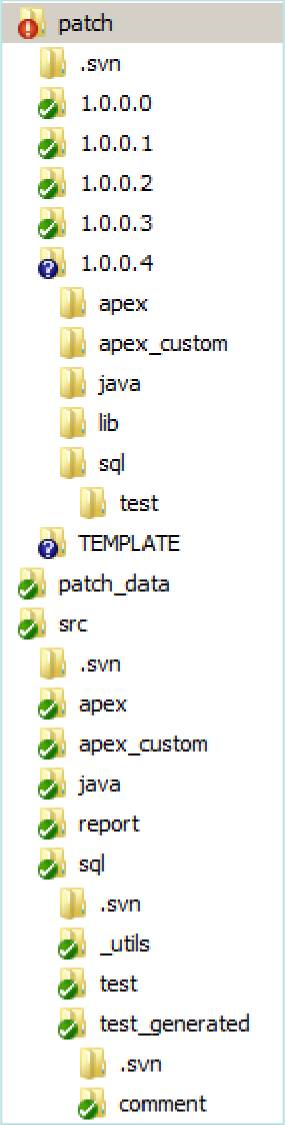
- Database Objects
- Filesystem
- APEX

Strict Rollout Procedures

- Little room for errors

The Components: Version Control

- Subversion + TortoiseSVN
  - http://tortoisesvn.net
  - Integration with Windows Explorer
  - Icon overlays, context menus
- Version control of database objects
  - Using the simple checkin/checkout mechanism in Toad
  - Exclusively locking a database object for modification

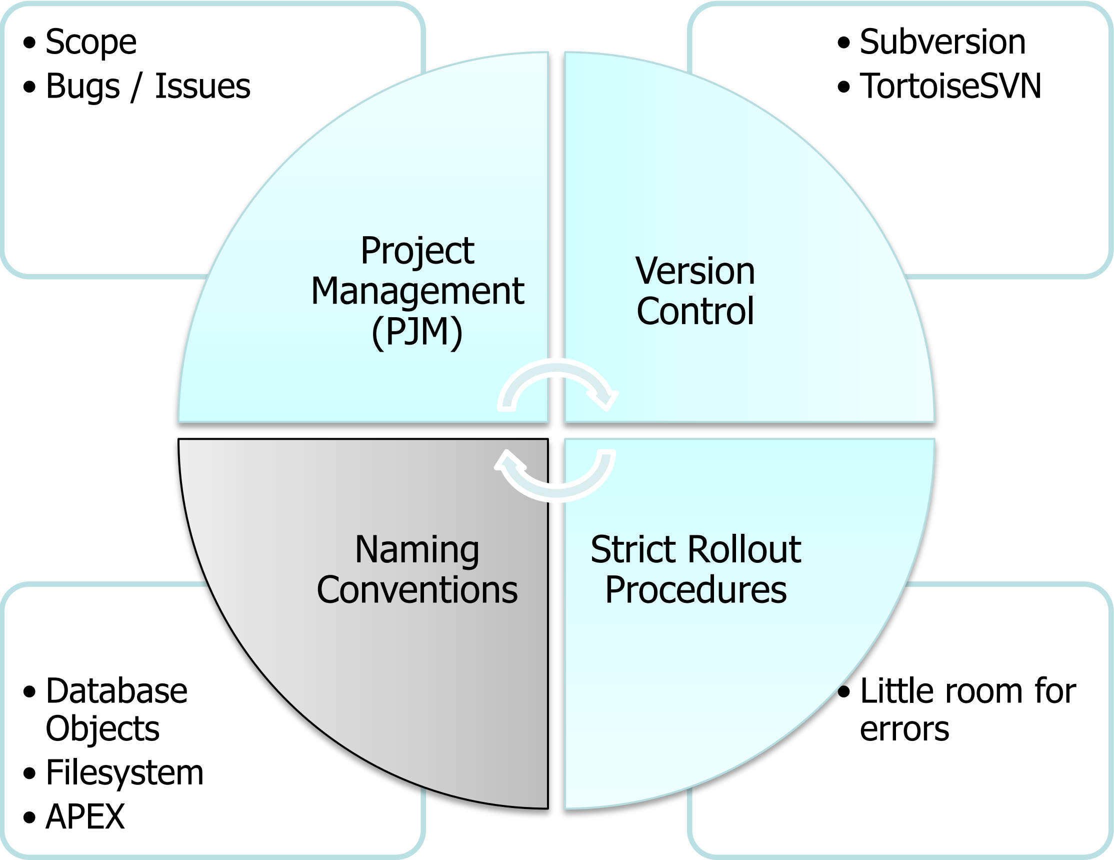


```
patch
  .svn
  1.0.0.0
  1.0.0.1
  1.0.0.2
  1.0.0.3
  1.0.0.4
    apex
    apex_custom
    java
    lib
    sql
      test
  TEMPLATE
patch_data
src
  .svn
  apex
  apex_custom
  java
  report
  sql
    .svn
    _utils
    test
    test_generated
      .svn
      comment
```

The Components: Version Control

- All files are under version control
- Changing database objects
  - Set an exclusive lock
  - Modify it directly in the database
  - Register the modified objects in the patch script immediately
    - Only for “true” ddl , e.g.: alter table add column
    - Empty files for the rest
  - Release the lock
  - Use subversion to control the filesystem

The Components: Naming Conventions



- Scope
- Bugs / Issues

- Subversion
- TortoiseSVN

Project Management (PJM)

Version Control

Naming Conventions

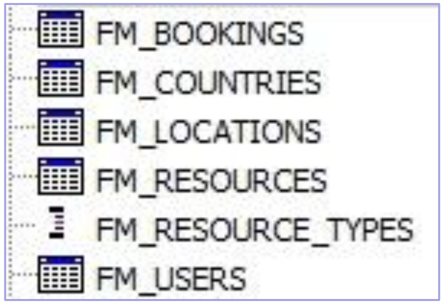
Strict Rollout Procedures

- Database Objects
- Filesystem
- APEX

- Little room for errors

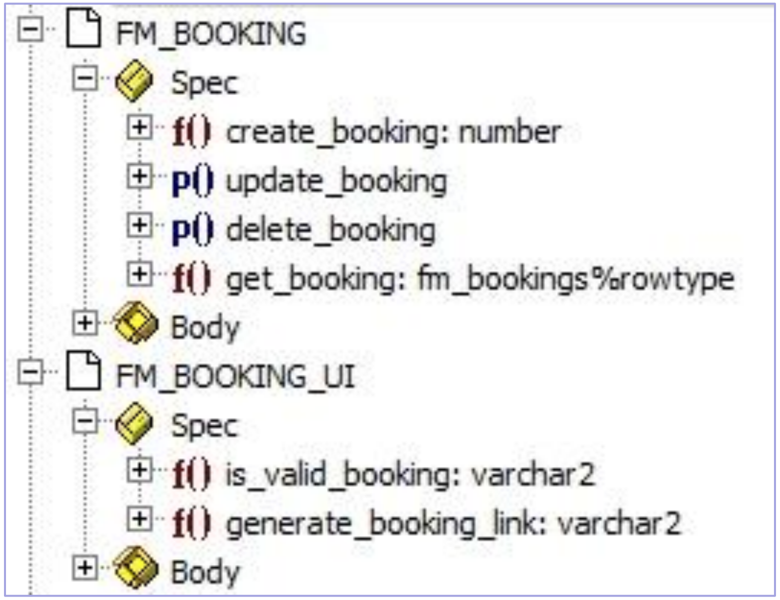
The Components: Naming Conventions

- Table names in plural



FM_BOOKINGS
FM_COUNTRIES
FM_LOCATIONS
FM_RESOURCES
FM_RESOURCE_TYPES
FM_USERS

- Packages in singular
  - CGUD conventions for functions / procedures
    - create, get, update, delete



- FM_BOOKING
  - Spec
    - f() create_booking: number
    - p() update_booking
    - p() delete_booking
    - f() get_booking: fm_bookings%rowtype
  - Body
- FM_BOOKING_UI
  - Spec
    - f() is_valid_booking: varchar2
    - f() generate_booking_link: varchar2
  - Body

The Components: Naming Conventions

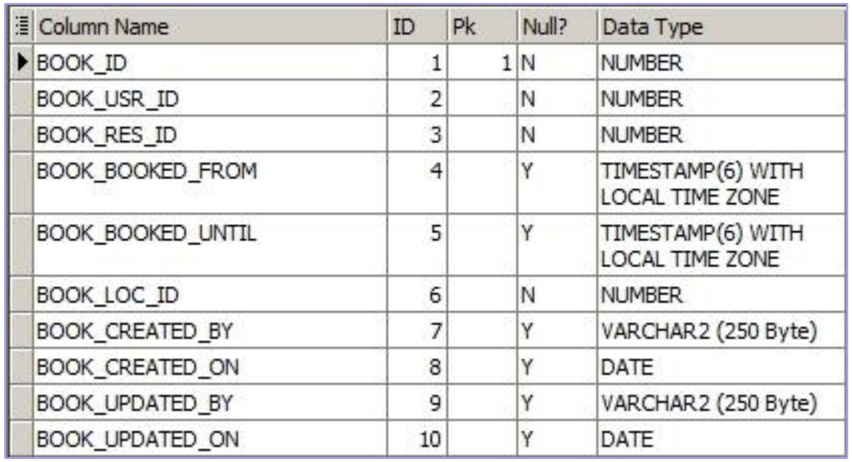
- Other object types have their type appended
  - Views: _v
  - Materialized views: _mv
  - Triggers: _trg
  - Primary keys: _pk
  - Indexes: _idx
  - …

The Components: Naming Conventions

- Data type of column can be guessed by its name
  - Boolean: is_valid_number
  - Date: created_on, valid_until
  - Varchar, i.e. user name: created_by, updated_by
- We don’t use **name** columns any more
  - Either we mean a (internal and unchangeable) **code** or (a possibly to be changed) **title**

The Components: Naming Conventions

- Sample table: FM_BOOKINGS



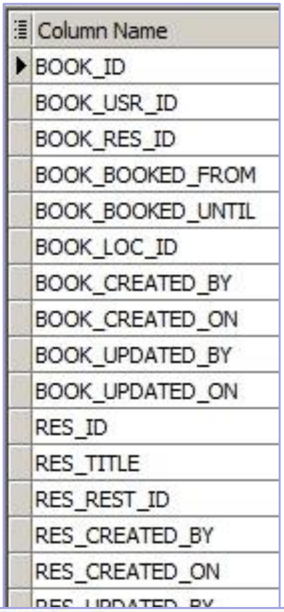
Column Name	ID	Pk	Null?	Data Type
BOOK_ID	1	1	N	NUMBER
BOOK_USR_ID	2		N	NUMBER
BOOK_RES_ID	3		N	NUMBER
BOOK_BOOKED_FROM	4		Y	TIMESTAMP(6) WITH LOCAL TIME ZONE
BOOK_BOOKED_UNTIL	5		Y	TIMESTAMP(6) WITH LOCAL TIME ZONE
BOOK_LOC_ID	6		N	NUMBER
BOOK_CREATED_BY	7		Y	VARCHAR2 (250 Byte)
BOOK_CREATED_ON	8		Y	DATE
BOOK_UPDATED_BY	9		Y	VARCHAR2 (250 Byte)
BOOK_UPDATED_ON	10		Y	DATE

- Prefix notation for columns
  - All columns are prefixed with the table short name / alias

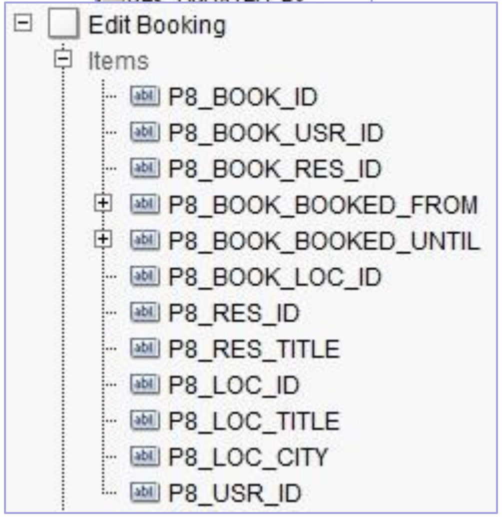
OPAL Consulting

The Components: Naming Conventions

- Prefix notation for columns
  - Impact on views, APEX page items
  - => TRANSPARENCY !!!
  - Sample: Edit Page on View FM_BOOKINGS_RL_V
    - A reference to page item P5_ID has to be explained / documented
    - References to P5_BOOK_ID or P6_USR_ID are transparent and self-documenting
    - No difference between data model and page items



Column Name
BOOK_ID
BOOK_USR_ID
BOOK_RES_ID
BOOK_BOOKED_FROM
BOOK_BOOKED_UNTIL
BOOK_LOC_ID
BOOK_CREATED_BY
BOOK_CREATED_ON
BOOK_UPDATED_BY
BOOK_UPDATED_ON
RES_ID
RES_TITLE
RES_REST_ID
RES_CREATED_BY
RES_CREATED_ON
RES_UPDATED_BY



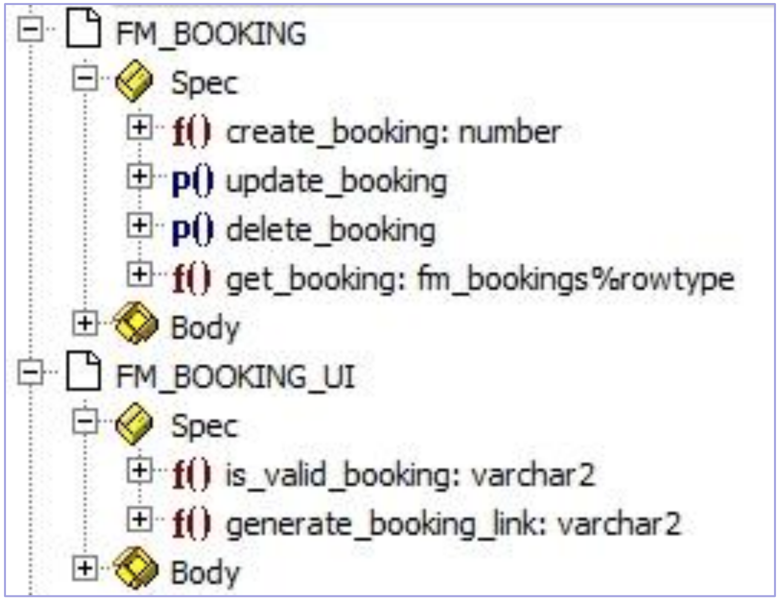
Edit Booking
Items
P8_BOOK_ID
P8_BOOK_USR_ID
P8_BOOK_RES_ID
P8_BOOK_BOOKED_FROM
P8_BOOK_BOOKED_UNTIL
P8_BOOK_LOC_ID
P8_RES_ID
P8_RES_TITLE
P8_LOC_ID
P8_LOC_TITLE
P8_LOC_CITY
P8_USR_ID

The Components: Naming Conventions

- Use domains for common columns
  - i.e. columns containing numbers should always be called **NO**
  - i.e. columns containing descriptions should always be called **DESC**
  - Use the same data type and length consistently

The Components: Naming Conventions

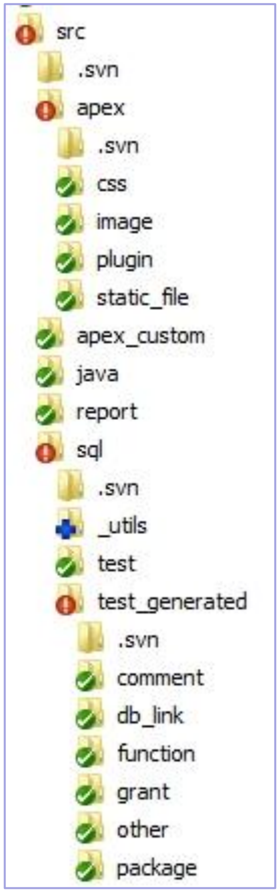
- Separate packages for UI and business logic
- Business packages
  - Use automated testing
- UI-packages
  - Specific logic just for our APEX applications
    - Generate a html link to another page
    - Page validations, referencing application state (v('P5_ID')) or using collections



```
FM_BOOKING
  Spec
    f() create_booking: number
    p() update_booking
    p() delete_booking
    f() get_booking: fm_bookings%rowtype
  Body
FM_BOOKING_UI
  Spec
    f() is_valid_booking: varchar2
    f() generate_booking_link: varchar2
  Body
```

The Components: Naming Conventions

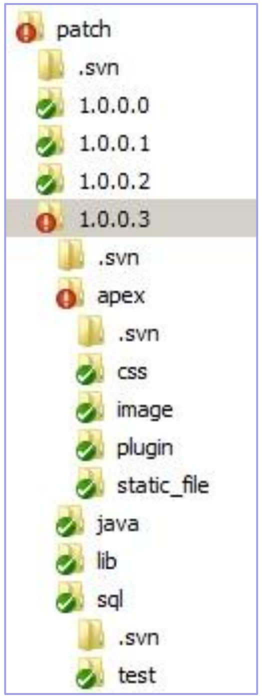
- File system layout - SRC
  - Source files
    - Organization by source type, then module or schema
  - src
    - apex (import into workspace)
      - static_file, image, css, plugin
    - apex_custom (virtual directory on the web server)
    - sql
      - schema1
      - schema2



```
src
  .svn
  apex
    .svn
    css
    image
    plugin
    static_file
  apex_custom
  java
  report
  sql
    .svn
    _utils
    test
    test_generated
      .svn
      comment
      db_link
      function
      grant
      other
      package
```

The Components: Naming Conventions

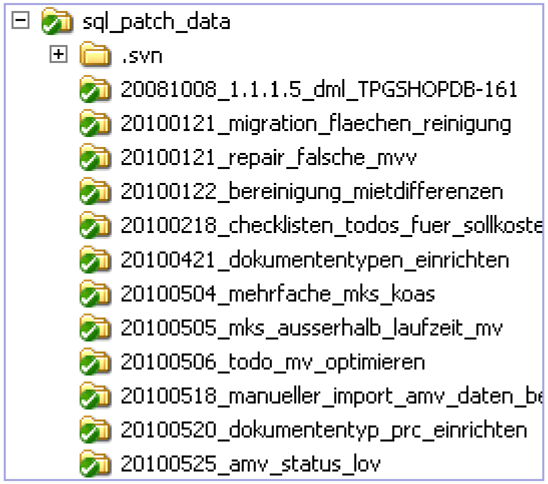
- File system layout - PATCH
  - Versions:
    - Major.Minor.Patch.Revision
      - Only communicate the first three
  - patch (new software release)
    - 1.0.0.0
      - apex
      - sql/schema1
      - sql/schema2
    - 1.0.0.1
    - 1.0.0.2



```
patch
  .svn
  1.0.0.0
  1.0.0.1
  1.0.0.2
  1.0.0.3
    .svn
    apex
      .svn
      css
      image
      plugin
      static_file
    java
    lib
    sql
      .svn
      test
```

The Components: Naming Conventions

- File system layout – PATCH_DATA
  - patch_data
    - (just a DML modification)
  - Not formalized yet, but
    - All patches are listed and under version control
    - Each execution is recorded in the database



```
⊟ sql_patch_data
    ⊞ .svn
      20081008_1.1.1.5_dml_TPGSHOPDB-161
      20100121_migration_flaechen_reinigung
      20100121_repair_falsche_mvv
      20100122_bereinigung_mietdifferenzen
      20100218_checklisten_todos_fuer_sollkoste
      20100421_dokumententypen_einrichten
      20100504_mehrfache_mks_koas
      20100505_mks_ausserhalb_laufzeit_mv
      20100506_todo_mv_optimieren
      20100518_manueller_import_amv_daten_be
      20100520_dokumententyp_prc_einrichten
      20100525_amv_status_lov
```

The Components: Naming Conventions

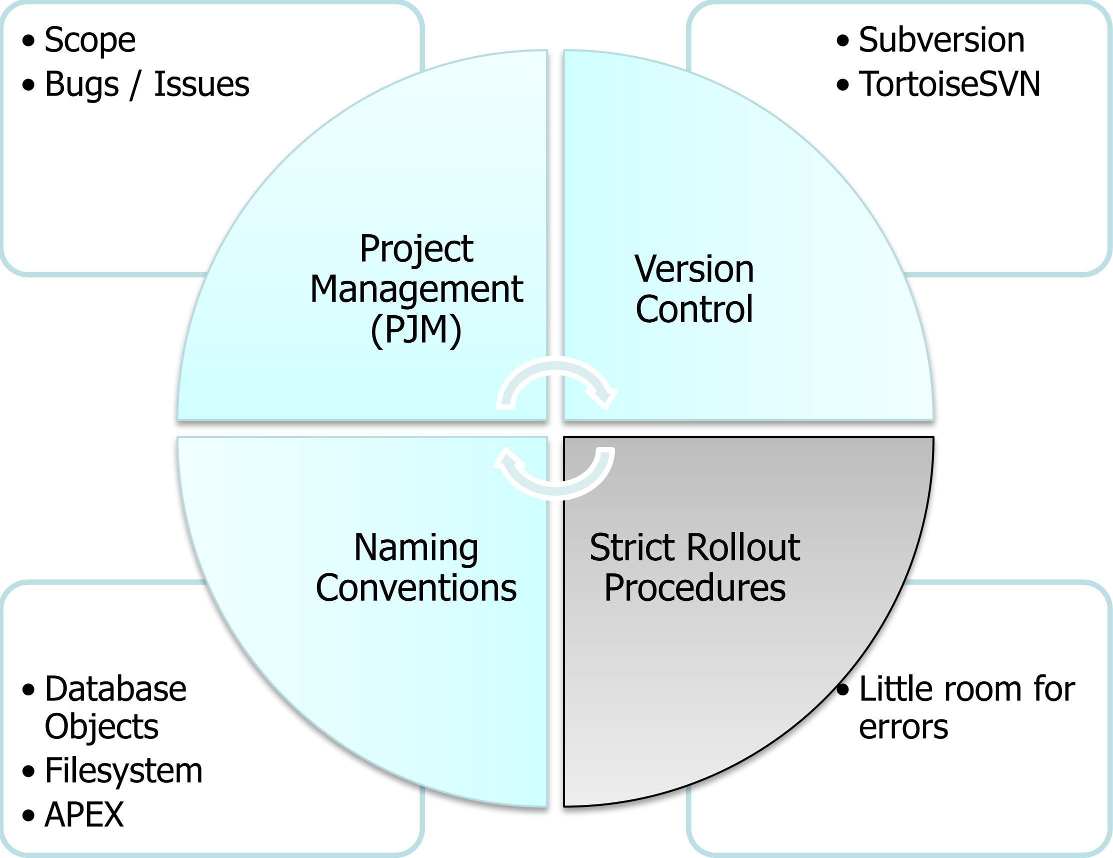
- Naming conventions for all files
  - In most cases: <object_name>.sql (e.g. for table, view, trigger, foreign key constraint, type, procedure, function)
  - Exception: (packages or types, they have a spec and body)
    - package_name.pks
    - package_name.pkb
  - All scripts are in lowercase => can be run on Windows and *nix
  - Data manipulations scripts: <table_name>_data.sql
    - Insert, Update, Delete
    - Insert into FM_BOOKINGS => fm_bookings_data.sql

The Components: Naming Conventions

- Naming conventions for all files
  - Easy to find changes:
    - When did we manipulate the configuration table?
    - dir /s *data.sql
    - When did we add the column xyz to table shdb_standorte?

OPAL Consulting

The Components: Naming Conventions



- Scope
- Bugs / Issues

Project Management (PJM)

- Subversion
- TortoiseSVN

Version Control

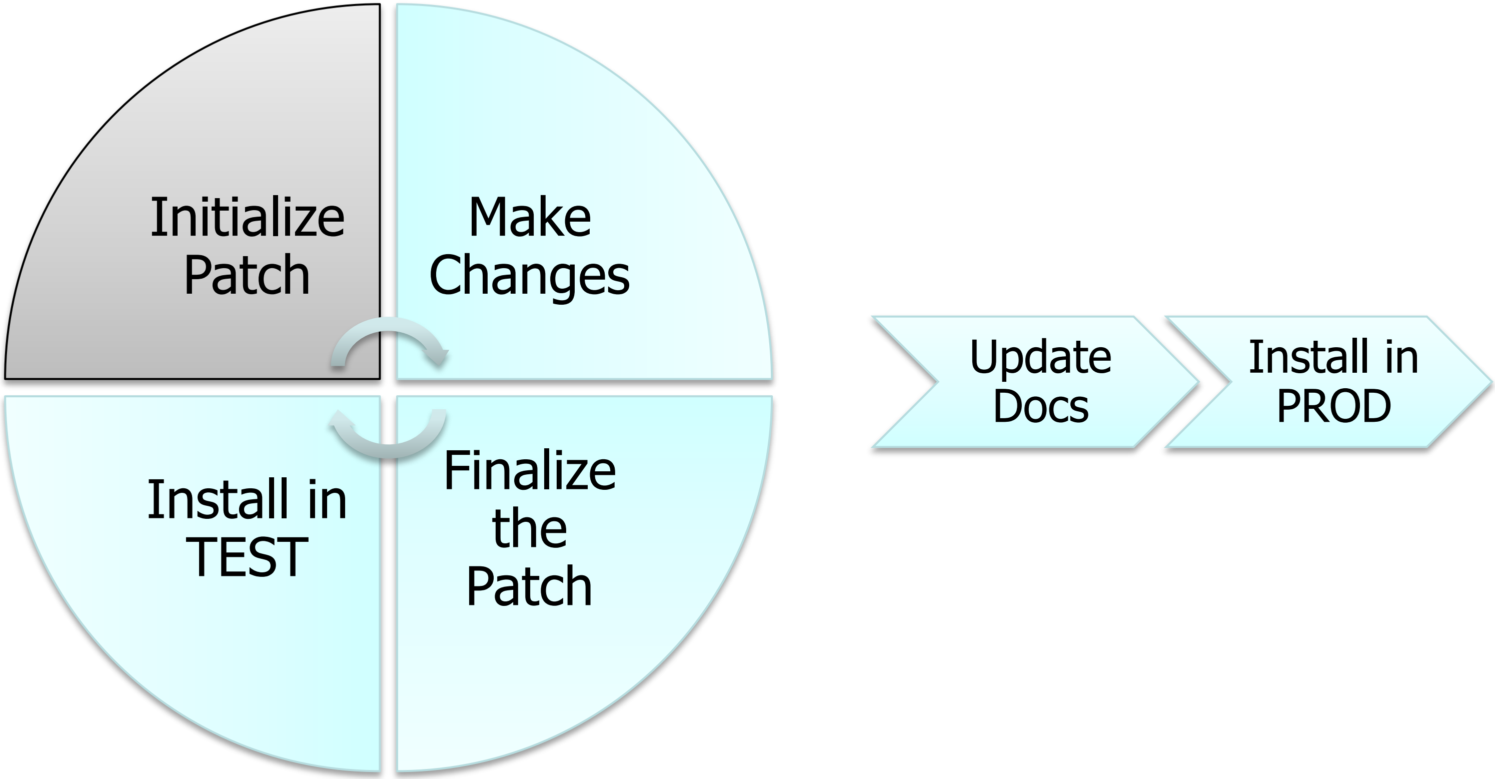
Naming Conventions

- Database Objects
- Filesystem
- APEX

Strict Rollout Procedures

- Little room for errors

Rollout from A-Z



Initialize Patch

Make Changes

Finalize the Patch

Install in TEST

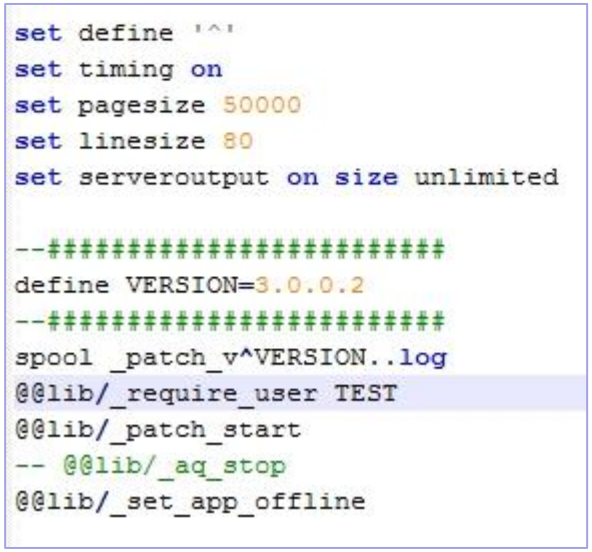
Update Docs

Install in PROD

Multiple iterations for internal testing

Rollout from A-Z: Initialize Patch

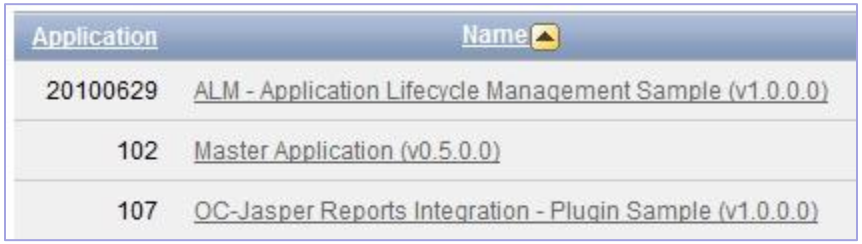
- Use a template to create the patch directory (zip file or ANT script)
  - Patch 1.0.0.x as a copy of the TEMPLATE directory
  - Change version number in the patch script `_patch.sql`



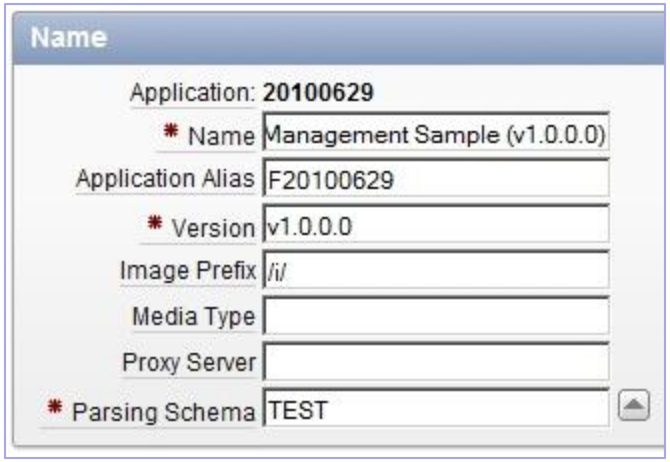
```
set define '^'
set timing on
set pagesize 50000
set linesize 80
set serveroutput on size unlimited

--#########################
define VERSION=3.0.0.2
--#########################
spool _patch_v^VERSION..log
@@lib/_require_user TEST
@@lib/_patch_start
-- @@lib/_aq_stop
@@lib/_set_app_offline
```

Rollout from A-Z: Initialize Patch

- Increase version number in your APEX application



Application	Name ▲
20100629	ALM - Application Lifecycle Management Sample (v1.0.0.0)
102	Master Application (v0.5.0.0)
107	OC-Jasper Reports Integration - Plugin Sample (v1.0.0.0)

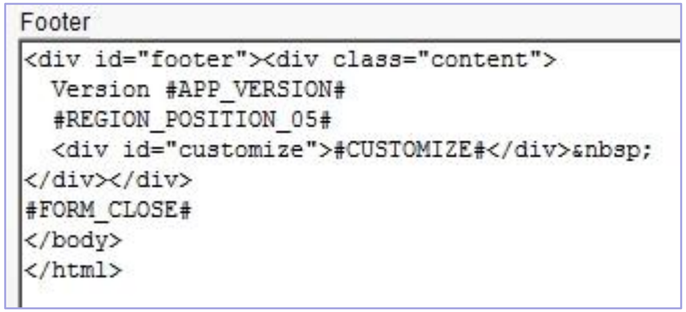


Name

Application:	**20100629**
* Name	Management Sample (v1.0.0.0)
Application Alias	F20100629
* Version	v1.0.0.0
Image Prefix	/i/
Media Type	
Proxy Server	
* Parsing Schema	TEST

Rollout from A-Z: Initialize Patch

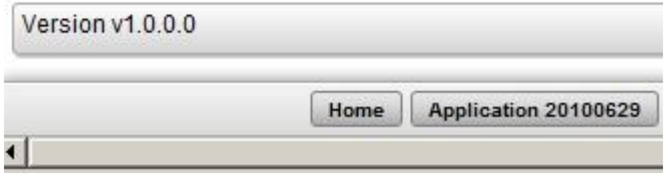
- Reference `#APP_VERSION#` in the page template footer



Footer

```
<div id="footer"><div class="content">
  Version #APP_VERSION#
  #REGION_POSITION_05#
  <div id="customize">#CUSTOMIZE#</div>&nbsp;
</div></div>
#FORM_CLOSE#
</body>
</html>
```

- Then it will appear in the application



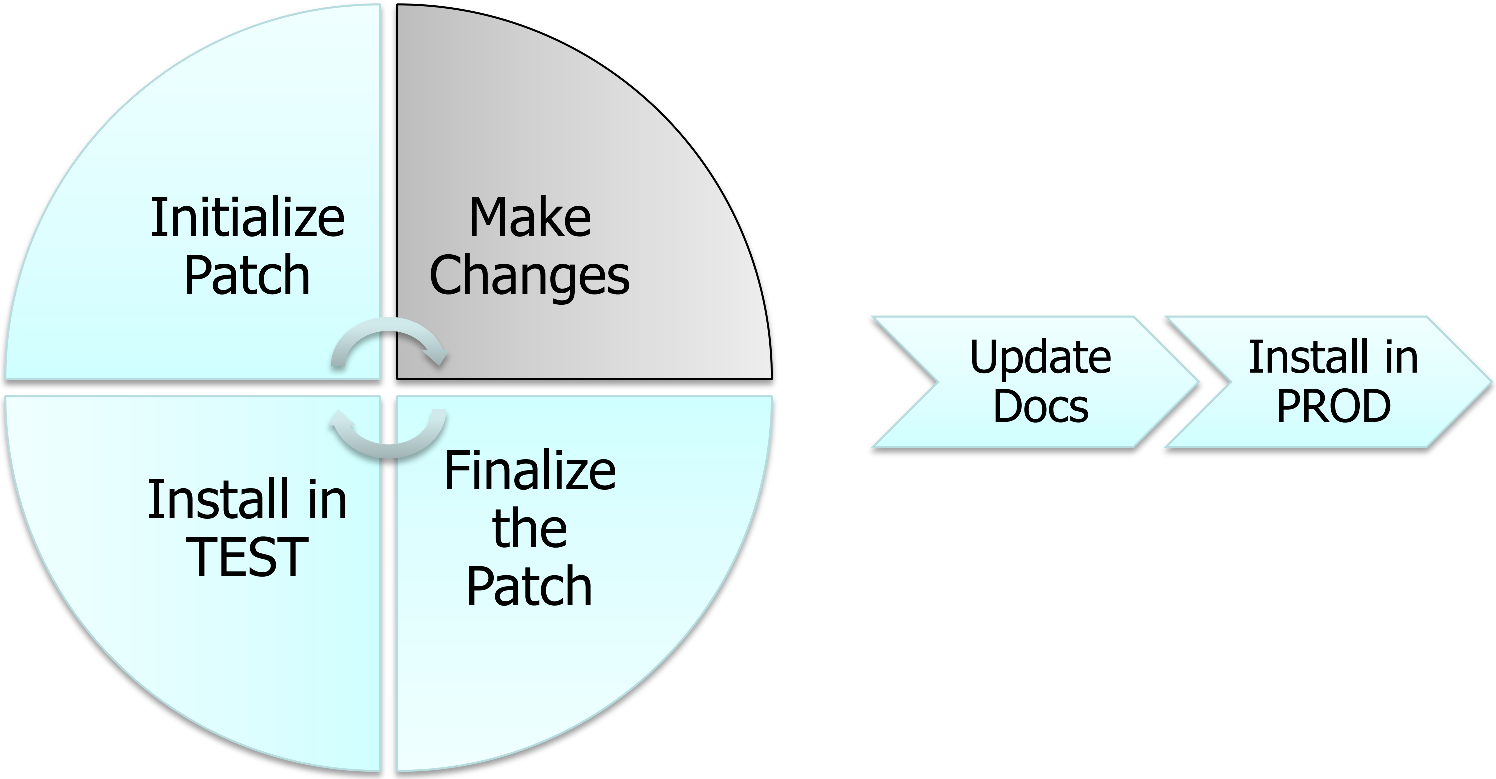
Version v1.0.0.0

Rollout from A-Z: Initialize Patch

- Demo:
  - Create new patch 1.0.0.4
    - Change version number
  - Modify APEX Application Version
    - 1.0.0.4

OPAL Consulting

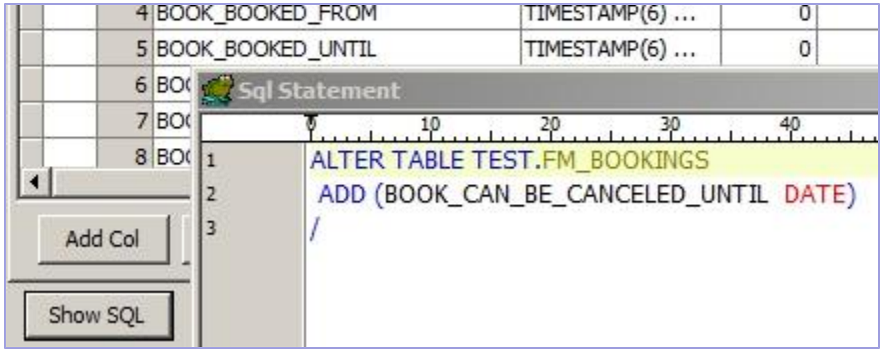
Rollout from A-Z: Make Changes



Multiple iterations for internal testing

Rollout from A-Z: Make Changes

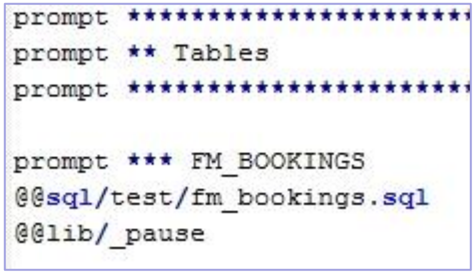
- ➢ Modify a table using toad and display the generated sql



4 BOOK_BOOKED_FROM	TIMESTAMP(6) ...	0
5 BOOK_BOOKED_UNTIL	TIMESTAMP(6) ...	0
6 BO(		
7 BO(		
8 BO(		

```
ALTER TABLE TEST.FM_BOOKINGS
 ADD (BOOK_CAN_BE_CANCELED_UNTIL  DATE)
/
```

- ➢ Add the file `fm_bookings.sql` to the `sql/test` directory and reference it in the `_patch.sql` script



```
prompt *********************
prompt ** Tables
prompt *********************

prompt *** FM_BOOKINGS
@@sql/test/fm_bookings.sql
@@lib/_pause
```

Rollout from A-Z: Make Changes

- If you modify any other object it only needs to be registered in the `_patch.sql` script.
  - Package, View, Procedure, Trigger and others can be completely generated from the database and copied over the empty files later.
  - And create the empty files for that in the `sql/test` directory

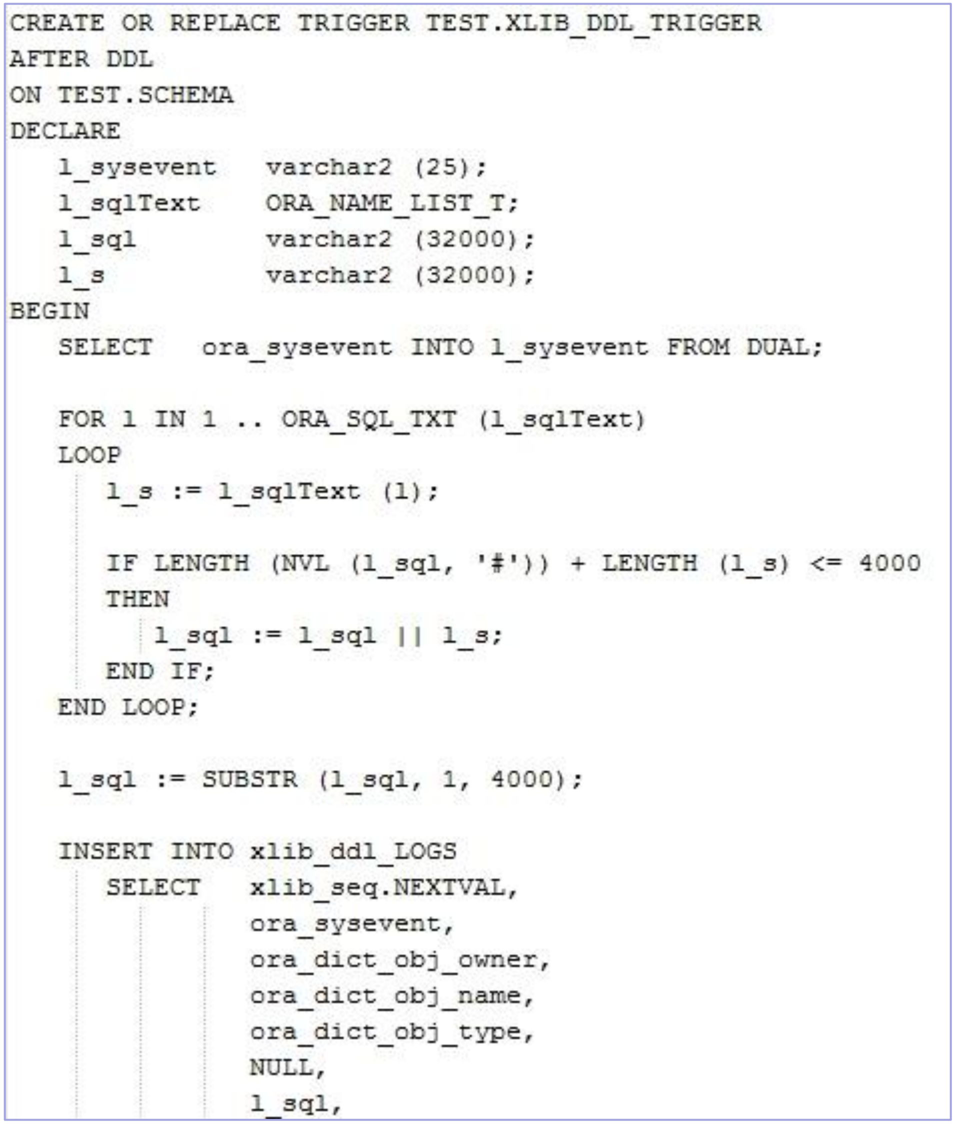
Rollout from A-Z: Make Changes

- The `_patch.sql` script defines an order how the objects have to be installed:
  - Types, Tables, Foreign keys, Views, Procedures, Functions, Packages Headers, Packages Bodies, Trigger, Data (DML) scripts, other scripts

- All scripts within a certain section have to be in alphabetical order (!!!) – helps with Subversion and things are easier to find
  - Show `_patch.sql` script from Spots

Rollout from A-Z: Make Changes

- Experimental DDL trigger to record changes

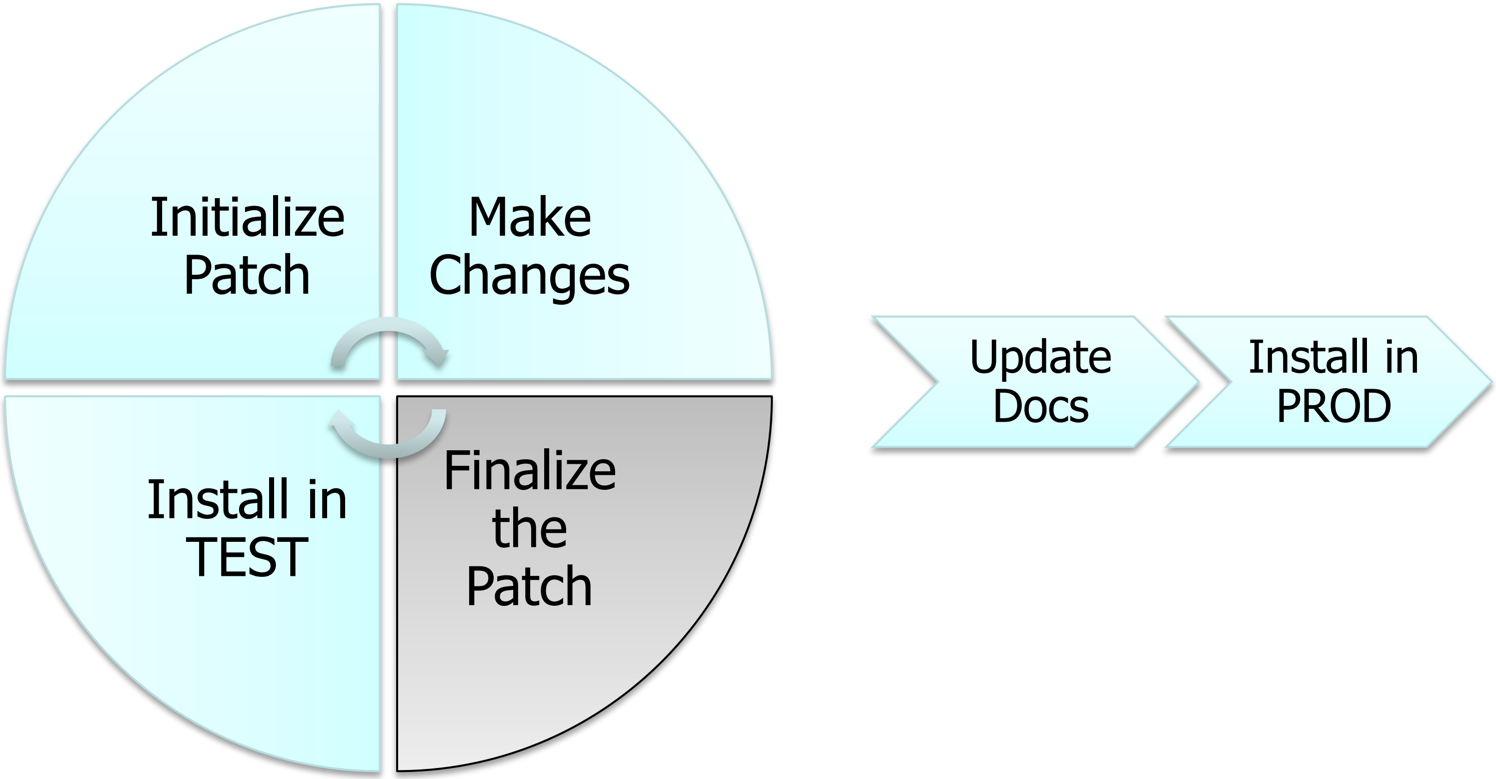


```
CREATE OR REPLACE TRIGGER TEST.XLIB_DDL_TRIGGER
AFTER DDL
ON TEST.SCHEMA
DECLARE
   l_sysevent   varchar2 (25);
   l_sqlText    ORA_NAME_LIST_T;
   l_sql        varchar2 (32000);
   l_s          varchar2 (32000);
BEGIN
   SELECT   ora_sysevent INTO l_sysevent FROM DUAL;

   FOR l IN 1 .. ORA_SQL_TXT (l_sqlText)
   LOOP
      l_s := l_sqlText (l);

      IF LENGTH (NVL (l_sql, '#')) + LENGTH (l_s) <= 4000
      THEN
         l_sql := l_sql || l_s;
      END IF;
   END LOOP;

   l_sql := SUBSTR (l_sql, 1, 4000);

   INSERT INTO xlib_ddl_LOGS
      SELECT   xlib_seq.NEXTVAL,
               ora_sysevent,
               ora_dict_obj_owner,
               ora_dict_obj_name,
               ora_dict_obj_type,
               NULL,
               l_sql,
```

Rollout from A-Z: Initialize Patch

- Demo:
  - Add column to FM_BOOKINGS
    - BOOK_CAN_BE_CANCELED_UNTIL
  - Add column to view FM_BOOKINGS_RL_V
  - Change package FM_BOOKING

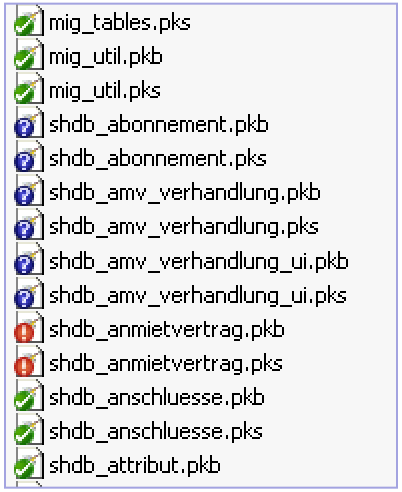
Rollout from A-Z: Finalize the Patch



Multiple iterations for internal testing

Rollout from A-Z: Finalize the Patch

- Extract the sources into the filesystem (again)
  - Subversion will highlight (!) the changed files
  - Copy them manually into the `sql/test` directory



✓ mig_tables.pks
✓ mig_util.pkb
✓ mig_util.pks
? shdb_abonnement.pkb
? shdb_abonnement.pks
? shdb_amv_verhandlung.pkb
? shdb_amv_verhandlung.pks
? shdb_amv_verhandlung_ui.pkb
? shdb_amv_verhandlung_ui.pks
! shdb_anmietvertrag.pkb
! shdb_anmietvertrag.pks
✓ shdb_anschluesse.pkb
✓ shdb_anschluesse.pks
✓ shdb_attribut.pkb

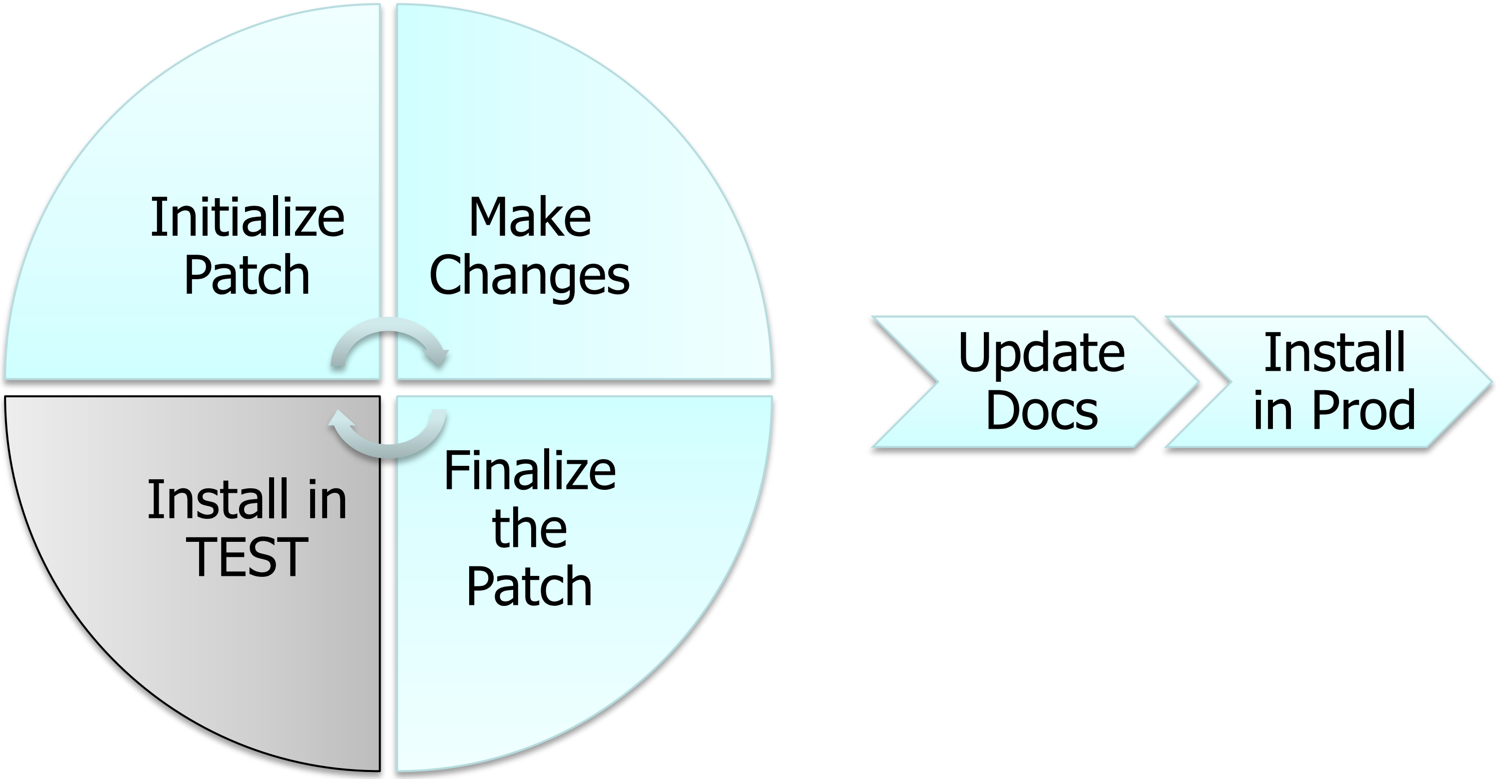
Rollout from A-Z: Finalize the Patch

- Export application file with version
  - `f20100629_alm_demo_v1.0.0.4`
- Copy all other relevant files (CSS, images, etc.) to the patch directory

Rollout from A-Z: Initialize Patch

- Demo:
  - Extract Sources again
  - Copy the modified files

Rollout from A-Z: Install in Test



Multiple iterations for internal testing

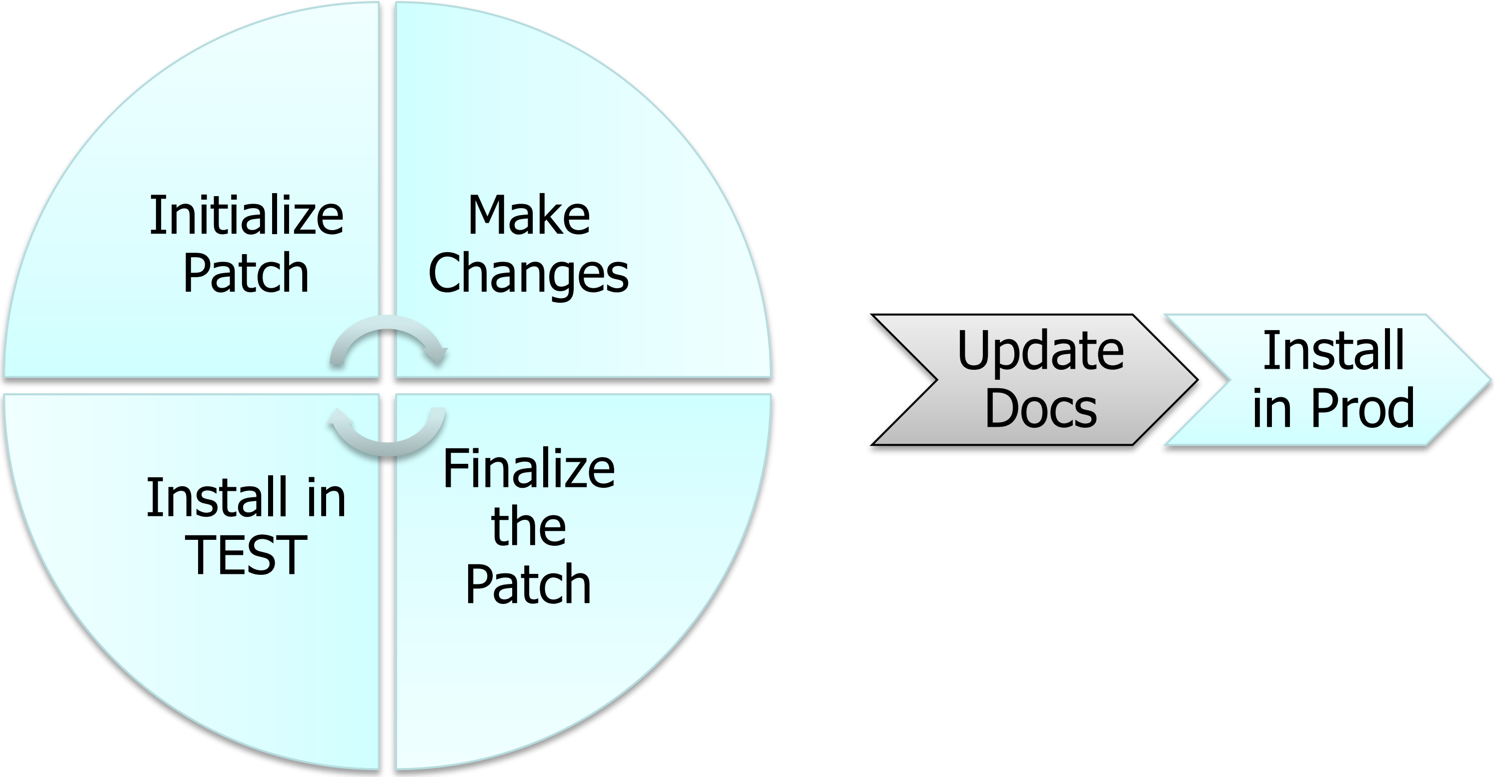
Rollout from A-Z: Install in Test

- Install the patch on the test system
  - Set restore point for flashback
    - SELECT name FROM v$restore_point;
    - create restore point BEFORE_REL_1_0_0_3;
  - Install patch
  - Flashback database if required
    - shutdown immediate;
    - startup mount;
    - flashback database to restore point BEFORE_REL_1_0_0_3;
    - alter database open resetlogs;
  - drop restore point BEFORE_REL_1_0_0_3;

Rollout from A-Z: Install in Test

- After testing, reverting and patching again (with modifications), the script is now complete
- Commit the current state of the directory `src/sql/test_generated` to subversion, so you can see the differences in the next patch

Rollout from A-Z: Update Docs



Initialize Patch

Make Changes

Install in TEST

Finalize the Patch

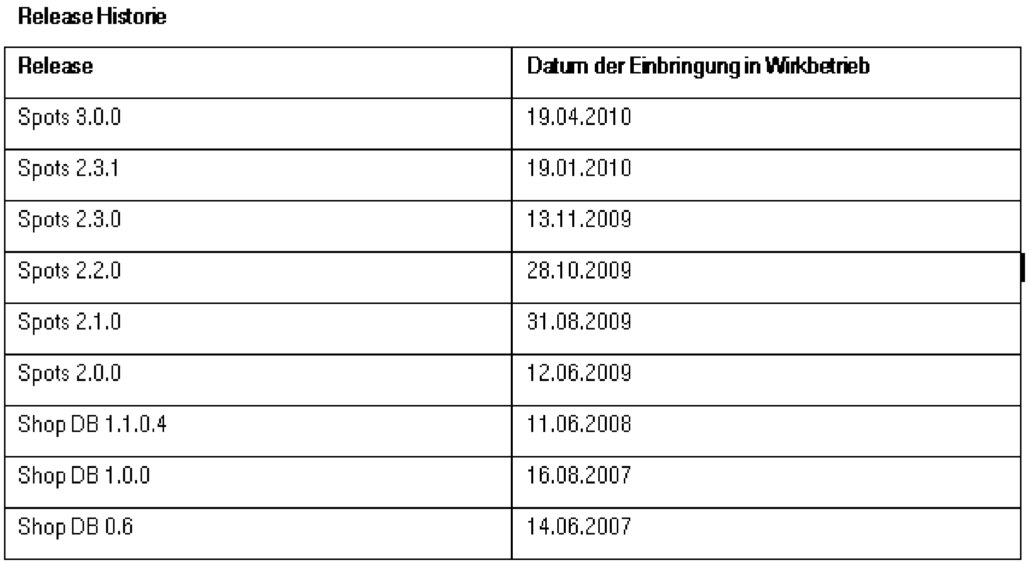
Update Docs

Install in Prod

Multiple iterations for internal testing

Rollout from A-Z: Update Docs

- Patch Version will be registered in table xlib_conf_values as VERSION
- Release history in operation guide or overall release notes



Release Historie

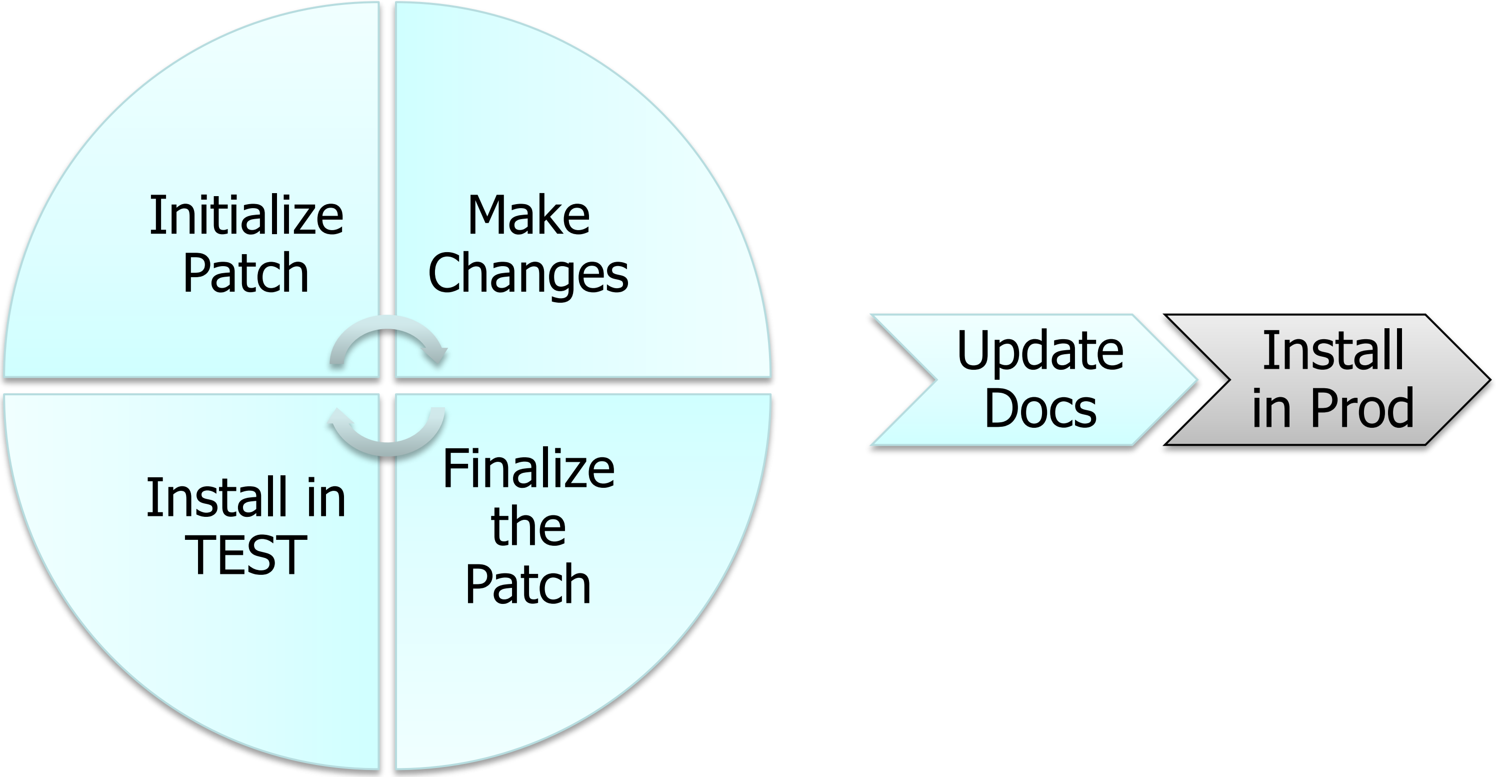
Release	Datum der Einbringung in Wirkbetrieb
Spots 3.0.0	19.04.2010
Spots 2.3.1	19.01.2010
Spots 2.3.0	13.11.2009
Spots 2.2.0	28.10.2009
Spots 2.1.0	31.08.2009
Spots 2.0.0	12.06.2009
Shop DB 1.1.0.4	11.06.2008
Shop DB 1.0.0	16.08.2007
Shop DB 0.6	14.06.2007

OPAL Consulting

Rollout from A-Z: Update Docs

- Update central use case document with delta documentation
  - Just copy/paste => no manual integration
- The technical documentation should be generated completety
  - Data model from data dictionary (use table and column comments!)
  - APEX from APEX dictionary
  - Database packages with pl/doc

Rollout from A-Z: Install In Prod



Multiple iterations for internal testing

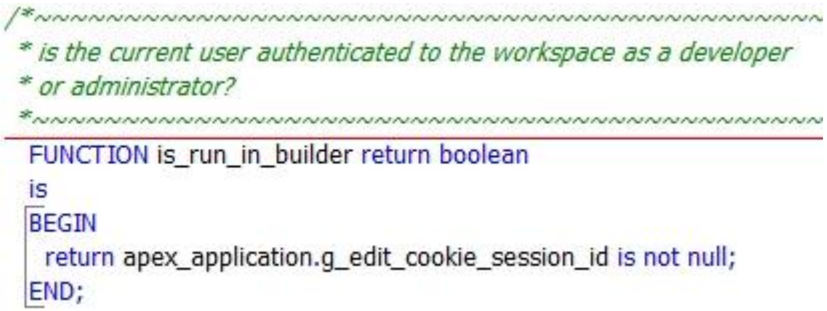
OPAL Consulting

Rollout from A-Z: Install In Prod

- Export current APEX application
- Backup database or set restore point for flashback
- When `_patch.sql` is run, the application was set offline
  - => only machines with registered IP-adresses could connect (or when run in the builder environment)

Rollout from A-Z: Install In Prod

- Set the application offline
  - Entry in configuration table
  - Application process to block users
- Internal testing (within the app builder)



```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * is the current user authenticated to the workspace as a developer
 * or administrator?
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 FUNCTION is_run_in_builder return boolean
 is
 BEGIN
   return apex_application.g_edit_cookie_session_id is not null;
 END;
```

- After final testing set application online again

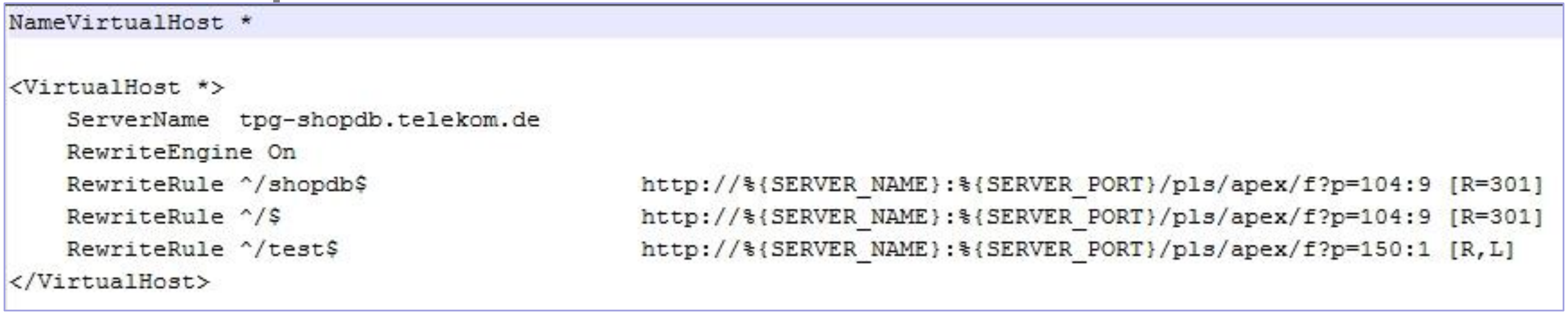
Easy to maintain? => Tips!

- Each execution of a script must be logged in the target system
- Install scripts => can be run multiple times
- Adaptive code
  - Dependant on instance type => not different versions of code on different systems!!!
  - Table XLIB_CONF_VALUES has configuration parameter INSTANCE_TYPE := DEV | TEST | PROD

Easy to maintain? => Tips!

- All things need an order !!!
  - Numbering of requirements, use cases, menu items, sorting in alphabetical order
  - => helps with Subversion, automatic merging of files, easier to look for specific entries

- Use Apache rewrites for the end-user URL



```
NameVirtualHost *

<VirtualHost *>
    ServerName  tpg-shopdb.telekom.de
    RewriteEngine On
    RewriteRule ^/shopdb$                   http://%{SERVER_NAME}:%{SERVER_PORT}/pls/apex/f?p=104:9 [R=301]
    RewriteRule ^/$                         http://%{SERVER_NAME}:%{SERVER_PORT}/pls/apex/f?p=104:9 [R=301]
    RewriteRule ^/test$                     http://%{SERVER_NAME}:%{SERVER_PORT}/pls/apex/f?p=150:1 [R,L]
</VirtualHost>
```

Easy to maintain? => Tips!

- apex_images
  - Don’t store files there !!!
- Use a custom virtual directory instead, e.g.: apex_custom
- Keep the workspace id identical on all systems

Easy to maintain? => Tips!

- Keep the application id identical on all systems, don’t use application aliases!!
  - Possible problems
    - User bookmarks reference to old application id
    - Interactive Reports loose the private reports
    - Script based deployment not possible in pre 4.0 environment
    - In APEX 4.0 you can use APEX_INSTALL to install into a different workspace, application and parsing schema
    - In APEX 4.0 you can fix the interactive report problem by modifying the offset with APEX_INSTALL

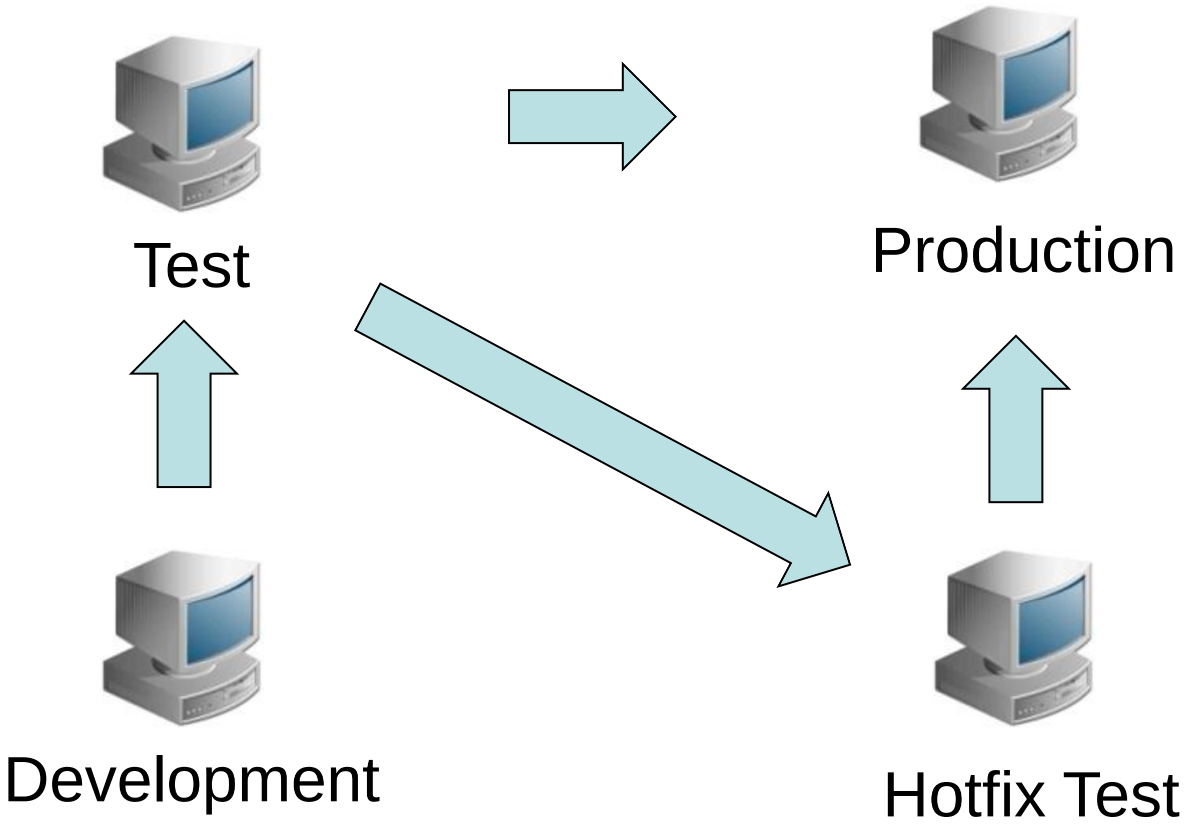
Parallel Development

- Working with multiple developers on different versions of the application
- Use cases:
  - Hotfix of a production issue
  - Parallelizing development efforts
- Different than ususal
  - Use central repository
  - Branching / Merging not advisable => use a separate application and copy over later

OPAL Consulting

Parallel Development

- Easy solution: Use multiple Oracle instances



Test

Production

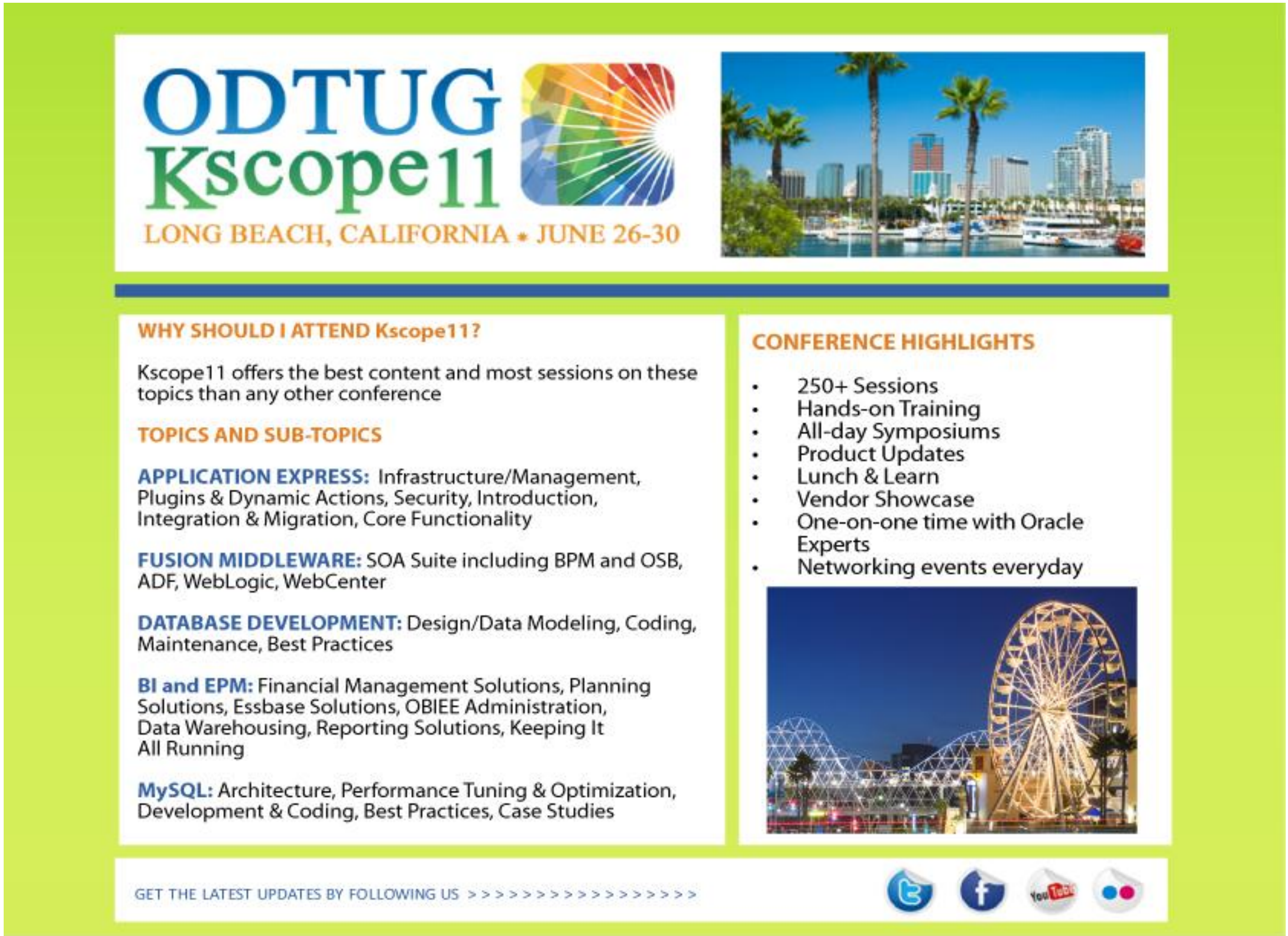
Development

Hotfix Test

Further developments

- Implementation with Maven 2 / 3
  - Add a target for SQL or PL/SQL projects
- Automatic generation of documentation
  - Data model
  - Application metadata from APEX
- Automated testing of business logic
- Automatic patch installer

Kaleidoscope 2011



ODTUG Kscope11

LONG BEACH, CALIFORNIA • JUNE 26-30

WHY SHOULD I ATTEND Kscope11?

Kscope11 offers the best content and most sessions on these topics than any other conference

TOPICS AND SUB-TOPICS

APPLICATION EXPRESS: Infrastructure/Management, Plugins & Dynamic Actions, Security, Introduction, Integration & Migration, Core Functionality

FUSION MIDDLEWARE: SOA Suite including BPM and OSB, ADF, WebLogic, WebCenter

DATABASE DEVELOPMENT: Design/Data Modeling, Coding, Maintenance, Best Practices

BI and EPM: Financial Management Solutions, Planning Solutions, Essbase Solutions, OBIEE Administration, Data Warehousing, Reporting Solutions, Keeping It All Running

MySQL: Architecture, Performance Tuning & Optimization, Development & Coding, Best Practices, Case Studies

CONFERENCE HIGHLIGHTS

- 250+ Sessions
- Hands-on Training
- All-day Symposiums
- Product Updates
- Lunch & Learn
- Vendor Showcase
- One-on-one time with Oracle Experts
- Networking events everyday

GET THE LATEST UPDATES BY FOLLOWING US > > > > > > > > > > > > > > > > > >

OPAL Consulting

Q&A

Q & A

- Contact:
  - Opal-Consulting Dietmar Aust
  - Web: http://www.opal-consulting.de
  - Blog: http://daust.blogspot.com/
  - E-Mail: dietmar.aust@opal-consulting.de

OPAL Consulting