apps associates
extreme expertise

Entwicklung von qualitativ hochwertigen APEX Anwendungen

Sven Böttcher

Consultant, Apps Associates GmbH

Apps Associates

ORACLE Platinum Partner

Apps Associates

- gegründet 2002 in Boston (HQ)
- seit 2003 GDC in Hyderabad
- seit 2006 in Dortmund
- 650+ Mitarbeiter auf drei Kontinenten
- Acht Standorte
- 41 Mio. USD Umsatz in 2014

Performance. Wachstum. Qualität.

- Oracle Platinum Partner
- Sechs zertifizierte Oracle Spezialisierungen
- Amazon Web Services Advanced Consulting Certified
- CMMI Level 3 Certification
- SSAE16 Compliance
- Flexibles & Kosteneffizientes “Global Delivery Model”
- Mitglied der ‘Scope Alliance’

scope alliance

CMMI LEVEL 3

ORACLE Platinum Partner
Specialized
Oracle Business Intelligence Applications 7

ORACLE Platinum Partner
Specialized
Oracle Database 11*g*

ORACLE Platinum Partner
Specialized
Oracle E-Business Suite R12.1 Supply Chain Management

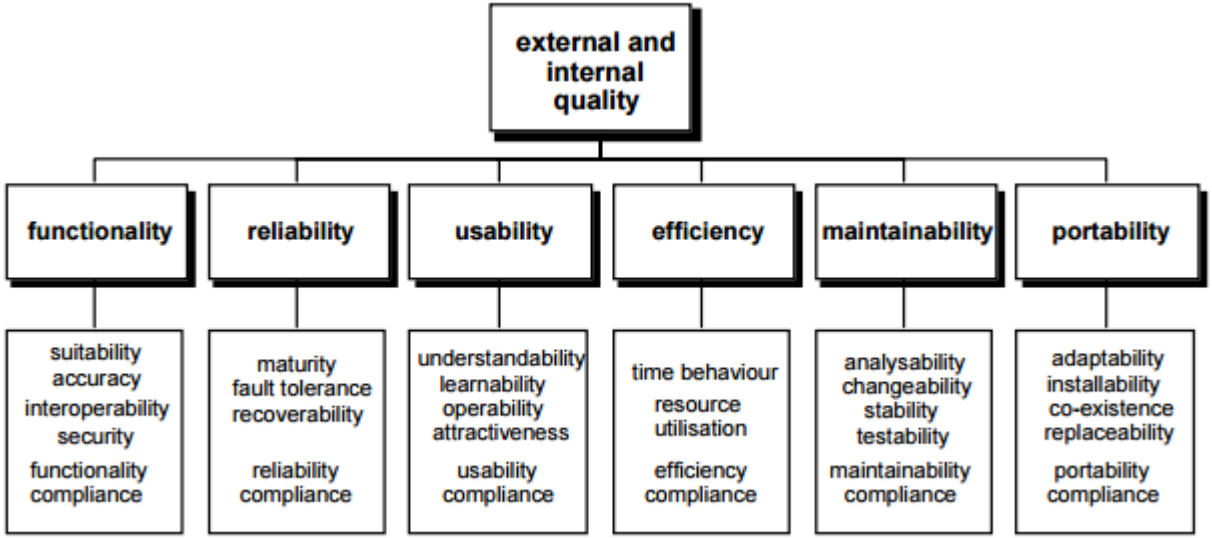
ORACLE Platinum Partner
Specialized
Service-Oriented Architecture

Agenda

- Continous Integration
- Jenkins als CI Server
- Jenkins mit Maven als Build Tool
- Jenkins mit RSpec als Test Tool
- Selenium (Webdriver) als Test Tool

Qualität von Software

- Qualitätsmodell nach ISO/IEC 9126



Quelle: SO/IEC FDIS 9126-1

Maßnahmen zur Qualitätssicherung

- Vorgehensmodelle
- Dokumentation
- Refactoring
- Code Reviews
- Entwicklungsstandards
- Best Practices
- Softwaretests
- Testgetriebene Entwicklung
- Continous Integration
- …

APEX Best Practices

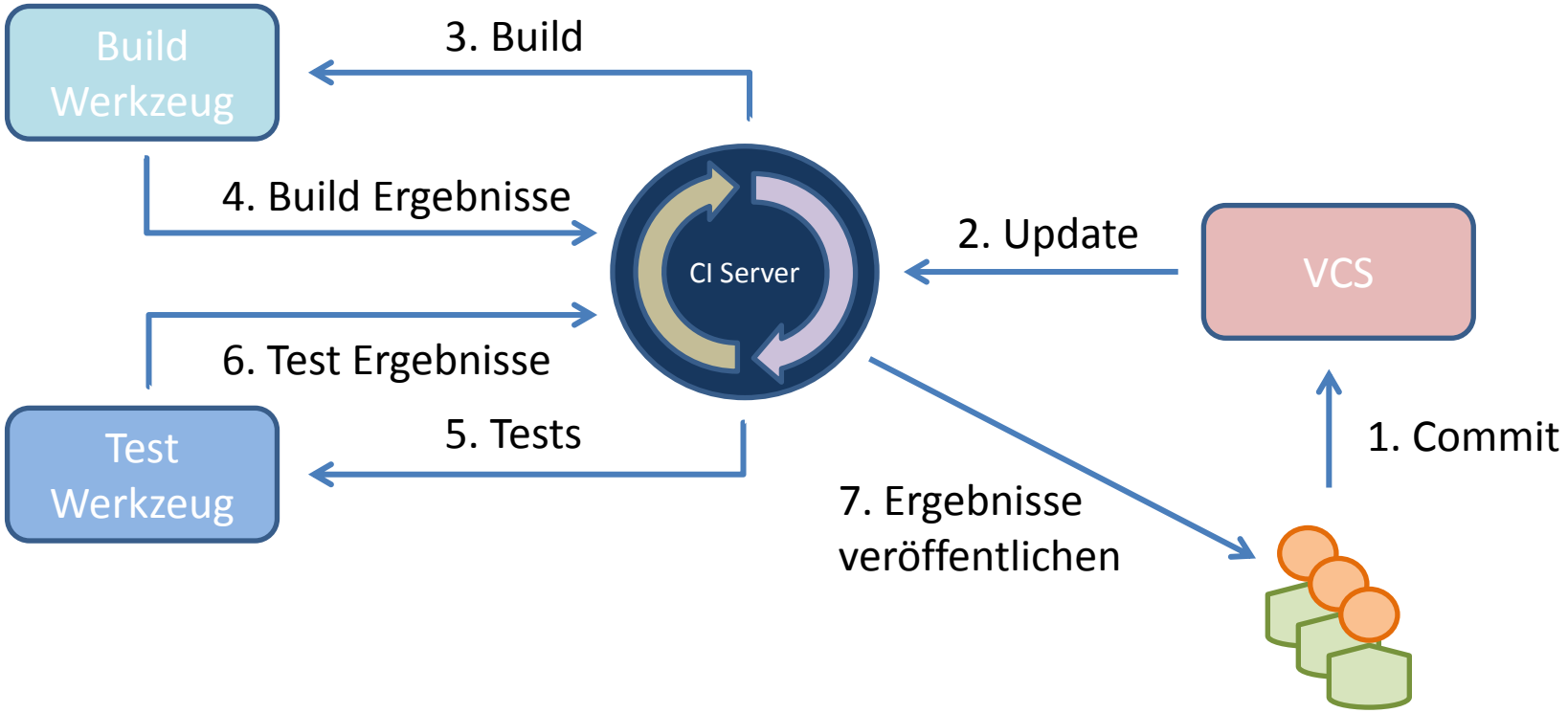
- Berichte: Auf APEX Seiten vs. in Views implementieren
- JavaScript: Auf APEX Seiten vs. als statische Komponente
- Tabular Forms: Assistenten vs. manuell implementieren mit APEX Items
- Zugriffssteuerung: Bedingung von Elementen vs. Autorisierungsschema
- AJAX: Dynamic Actions vs. manuell implementieren (apex.server.process)
- PLSQL: Auf APEX Seiten vs. in Packages/Funktionen/Prozeduren
- …

Continous Integration

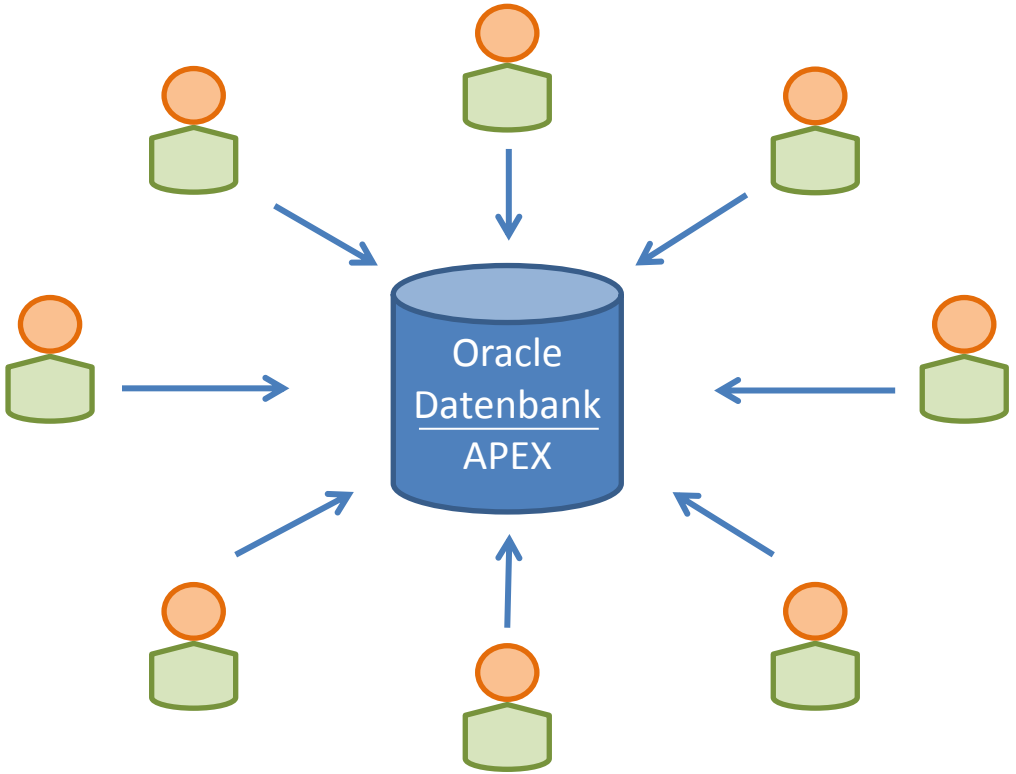
- Verfahren zur Steigerung der Softwarequalität
  - Kontinuierliches zusammenfügen und „bauen“ der Software
  - Kontinuierliche Ausführung von Softwaretests

- Unterstützung durch CI Server (Hudson, Jenkins, Bamboo,…)
  - Automatisiertes bauen und testen
  - Ausgelöst durch Änderungen im Versionskontrollsystem
  - Überblick über den Status des Softwareprojektes
  - I.A. lassen sich weitere Operationen zur Qualitätssteigerung integrieren

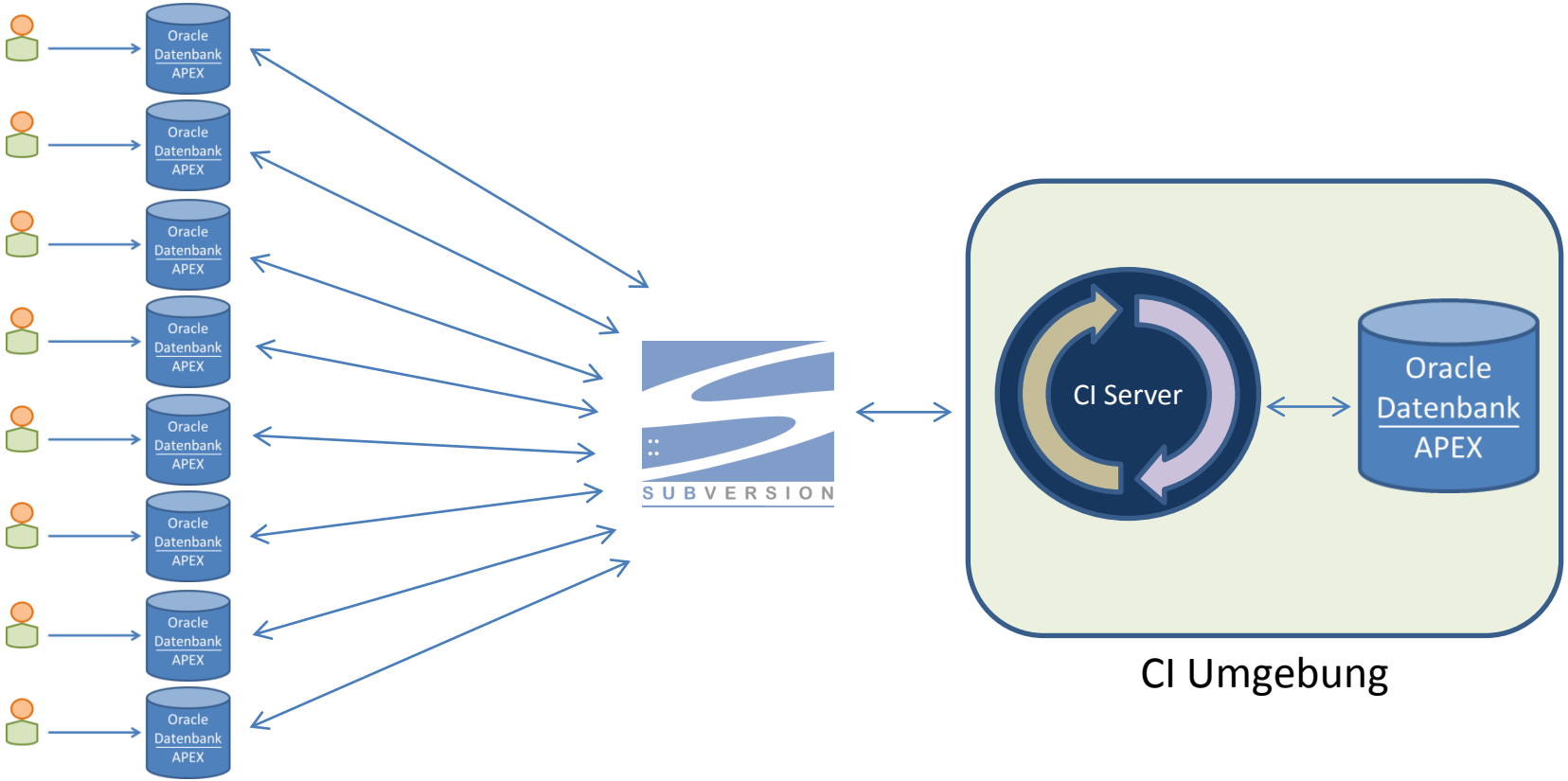
Continous Integration



APEX/PLSQL Entwicklung im Team

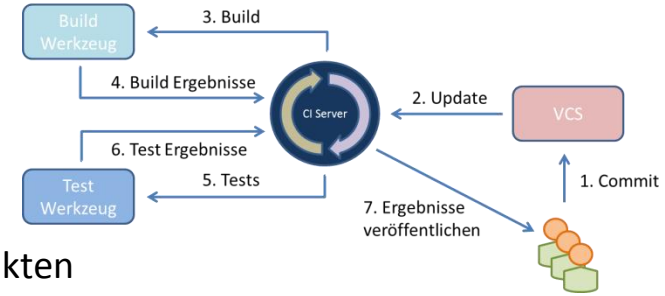


APEX und Continous Integration



APEX und Continous Integration

- Beispiel Architektur CI Server
  - CI Server: Jenkins
  - Versionskontrollsystem: SVN
  - Build-Werkzeug: Maven
    - ojdbc Treiber
    - plsql-maven-plugin → „bauen“ von Datenbankobjekten
    - exec-maven-plugin → „bauen“ der APEX Anwendung mit sqlplus
  - Test-Werkzeug für Unit Tests: ruby-plsql-spec
  - Test-Werkzeug für APEX Tests: Selenium Webdriver



- SVN mit dem SQL Developer
  - DDL und PLSQL muss in Dateien gespeichert werden
  - Export von APEX Anwendungen/Seiten

APEX/PLSQL Build mit Maven

Jenkins

APEX/PLSQL Build mit Maven

```
<build>
    <plugins>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>sql-maven-plugin</artifactId>
            <version>1.4</version>
            <dependencies>
                <dependency>
                    <groupId>com.oracle.jdbc</groupId>
                    <artifactId>ojdbc7</artifactId>
                    <version>12.1.0.1</version>
                </dependency>
            </dependencies>
            <configuration>
                <driver>oracle.jdbc.driver.OracleDriver</driver>
                <url>jdbc:oracle:thin:@192.168.56.102:1521:workshop</url>
                <username>ci_test</username>
                <password>ci_test</password>
                <skip>false</skip>
            </configuration>
```

APEX/PLSQL Build mit Maven

```
            <executions>
                <execution>
                    <id>test</id>
                    <phase>process-resources</phase>
                    <goals>
                        <goal>execute</goal>
                    </goals>
                    <configuration>
                        <autocommit>true</autocommit>
                        <delimiter>/</delimiter>
                        <delimiterType>row</delimiterType>
                        <onError>abort</onError>
                        <srcFiles>
                            <srcFile>ci_test/trunk/DDL/t_benutzer.sql</srcFile>
                            <srcFile>ci_test/trunk/DDL/t_benutzer_pk.sql</srcFile>
                            <srcFile>ci_test/trunk/DDL/t_benutzer_uk1.sql</srcFile>
                            <srcFile>ci_test/trunk/DDL/seq_t_benutzer.sql</srcFile>
                            <srcFile>ci_test/trunk/PLSQL/benutzerverwaltung.pks.sql</srcFile>
                            <srcFile>ci_test/trunk/PLSQL/benutzerverwaltung.pkb.sql</srcFile>
                        </srcFiles>
                    </configuration>
                </execution>
            </executions>
        </plugin>
```

APEX/PLSQL Build mit Maven

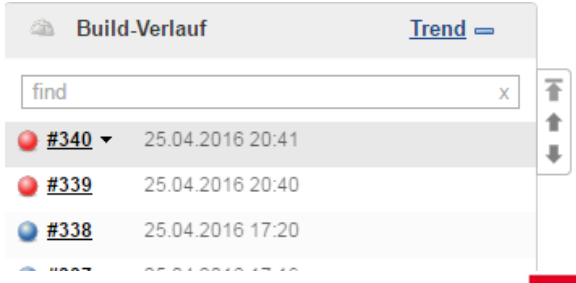
- Problem: Werden bereits existierende Datenbankobjekte jedes mal beim Build erzeugt, kommt es zu einem Fehler
- 2 Lösungen:
  - Alle Datenbankobjekte vor dem Erstellen löschen → problematisch beim Deployment z.B. auf Produktivumgebungen
  - Besser: execute emmediate mit Exception Block verwenden

```
BEGIN
  EXECUTE IMMEDIATE
    'CREATE SEQUENCE SEQ_T_BENUTZER
     INCREMENT BY 1 START WITH 1
     MAXVALUE 9999999999999999999999999999999999999999999999999
     MINVALUE 1';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE = -955 THEN
      NULL;
    ELSE
      RAISE;
    END IF;
END;
/
```

APEX/PLSQL Build mit Maven

- Kommt es beim kompilieren von DDLs zu einem Fehler, ist dies in Jenkins ersichtlich

```
[ERROR] Failed to execute:  CREATE TABLE "T_BENUTZER"
 (      "BENUTZER_ID" NUMBER,
 "VORNAME" VARCHAR2(200 BYTE),
 "NACHNAME" VARCHAR2(600 BYTE))

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 10.498 s
[INFO] Finished at: 2016-04-25T20:41:28+02:00
[INFO] Final Memory: 15M/71M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.codehaus.mojo:sql-maven-plugin:1.4:execute (test) on project test-app: ORA-00955: Es gibt bereits ein Objekt mit diesem Namen -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
[JENKINS] Archiving /var/lib/jenkins/workspace/APEX_CONNECT_2016_BUILD/pom.xml to com.appsassociates.apex-connect/test-app/1/test-app-1.pom
channel stopped
Finished: FAILURE
```



APEX/PLSQL Build mit Maven

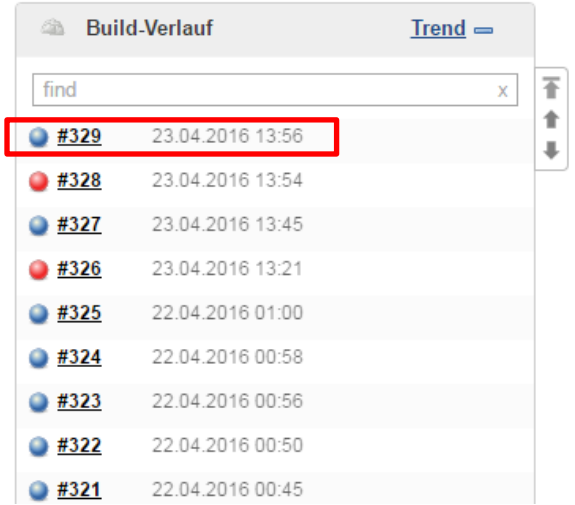
- Problem: Für Funktionen/Prozeduren/Packages gilt das nicht → Package wird mit Warnung kompiliert und Jenkins erkennt das Build als erfolgreich
- Beispiel: Package „benutzerverwaltung“ ist fehlerhaft

```
CREATE OR REPLACE PACKAGE BODY benutzerverwaltung AS

  FUNCTION erstelle_benutzer(
                i_vorname    IN t_benutzer.vorname%TYPE,
                i_nachname   IN t_benutzer.nachname%TYPE) RETURN NUMBER AS
  BEGIN

    INSERT INTO t_benutzer1(
      benutzer_id,
      vorname,
      nachname)
    VALUES(
      seq_t_benutzer.NEXTVAL,
      i_vorname,
      i_nachname);

    COMMIT;

    RETURN 1;

    EXCEPTION
      WHEN OTHERS THEN
        ROLLBACK;
        RETURN 0;

  END erstelle_benutzer;

END benutzerverwaltung;
```

APEX/PLSQL Build mit Maven

- Problem: Für Funktionen/Prozeduren/Packages gilt das nicht → Package wird mit Warnung kompiliert und Jenkins erkennt das Build als erfolgreich

```
[INFO]
[INFO] --- sql-maven-plugin:1.4:execute (test) @ test-app ---
[INFO] Executing file: /tmp/t_benutzer.323453694sql
[INFO] Executing file: /tmp/t_benutzer_pk.1559688219sql
[INFO] Executing file: /tmp/t_benutzer_uk1.1129476005sql
[INFO] Executing file: /tmp/seq_t_benutzer.844398631sql
[INFO] Executing file: /tmp/benutzerverwaltung.pks.276117888sql
[INFO] Executing file: /tmp/benutzerverwaltung.pkb.1957875124sql
[INFO] 6 of 6 SQL statements executed successfully
[INFO]
```



APEX/PLSQL Build mit Maven

- exec-maven-plugin → Kompilieren von Funktionen/Prozeduren/Prozeduren und „bauen“ der APEX Anwendung mit sqlplus

```
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.4.0</version>
                <executions>
                    <execution>
                        <phase>process-resources</phase>
                        <goals>
                            <goal>exec</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <executable>sqlplus64</executable>
                    <environmentVariables>
                        <LD_LIBRARY_PATH>/usr/lib/oracle/12.1/client64/lib/</LD_LIBRARY_PATH>
                    </environmentVariables>
                    <arguments>
                        <argument>ci_test/ci_test@192.168.56.102:1521/workshop</argument>
                        <argument>@install_database_objects/install_database_objects.sql</argument>
                    </arguments>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

APEX/PLSQL Build mit Maven

- exec-maven-plugin → Fehlermeldung durch raise_application_error im sql Skript

```
WHENEVER SQLERROR EXIT SQL.SQLCODE

set define off
/
@install_database_objects/install_packages.sql

@install_database_objects/install_apex.sql

show errors

declare
 l_count_errors number;
begin
 select count(*)
  into l_count_errors
  from all_errors;

 if l_count_errors > 0 then
  raise_application_error(-20001,'Fehler beim komilieren einer PL/SQL Prozedur');
 end if;

end;
/

exit 0;
```

APEX/PLSQL Build mit Maven

- Fehlerhaftes Build in Jenkins


```
Errors for PACKAGE BODY BENUTZERVERWALTUNG:

LINE/COL ERROR
-------- -----------------------------------------------------------------
8/5      PL/SQL: SQL Statement ignored
8/17     PL/SQL: ORA-00942: table or view does not exist
declare
*
ERROR at line 1:
ORA-20001: Fehler beim komilieren einer PL/SQL Prozedur
ORA-06512: at line 9


Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 45.592 s
[INFO] Finished at: 2016-04-24T08:43:51+02:00
[INFO] Final Memory: 14M/71M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.codehaus.mojo:exec-maven-plugin:1.4.0:exec (default) on project test-app: Command execution failed. Process exited with an error: 33 (Exit value: 33) -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
[JENKINS] Archiving /var/lib/jenkins/workspace/APEX_CONNECT_2016_BUILD/pom.xml to com.appsassociates.apex-connect/test-app/1/test-app-1.pom
channel stopped
Finished: FAILURE
```

APEX/PLSQL Build mit Maven

- „Bauen“ der APEX Anwendung

```
begin
  apex_application_install.set_workspace_id(5512166499766120);
  apex_application_install.set_application_id(102);
  apex_application_install.set_schema('CI_TEST');
  apex_application_install.generate_offset;
end;
/

@ci_test/trunk/APEX/CI_TEST.sql
@ci_test/trunk/APEX/PAGE_2.sql
```

Workspace ID der CI Datenbank!

Softwaretests

- Black-Box-Tests
  - Spezifikationsgetrieben
  - Das „innere“ der Software ist nicht bekannt
  - Werden i.A. durch den Kunden spezifiziert (äquivalenzklassenbasiert, grenzwertbasiert,…)
  - ➢ Entspricht die Software der Spezifikation?

- White-Box-Tests
  - Strukturgetrieben
  - Das „innere“ der Software ist der Testgegenstand
  - Werden i.A. durch den Entwickler spezifiziert (Anweisungsüberdeckung, kontrollflussorientierte Verfahren,…)
  - ➢ Enthält die Software Fehler?

PL/SQL Unit Tests mit ruby-plsql-spec (ruby-plsql+RSpec)

Jenkins

PL/SQL Unit Tests mit ruby-plsql-spec (ruby-plsql+RSpec)

- RSpec → Behavior Driven Development (BDD) / Test Framework für Ruby
  - Strukturierung der Tests:
    - Describe: Beschreibt eine Funktionalität bzw ein Verhalten
    - Context: Fasst Verhaltensbeschreibungen, die unter einem bestimmten Kontext bzw. Zustand gelten sollen, zusammen → Strukturierung von „Describe“ Blöcken
    - It: Beschreibt ein bestimmtes Verhalten bzw. einen konkreten Testfall
  - Expectations:
    - expect(actual).to eq(expected)
    - expect(actual).to be [>|<|>=|<=] expected
    - expect(actual).to be true
    - expect(actual).to be nil
    - expect(actual).not_to nil
    - …
- ruby-plsql → Ruby API für SQL/PLSQL

PL/SQL Unit Tests mit ruby-plsql-spec (ruby-plsql+RSpec)

- Unit Test mit ruby-plsql-spec

```
require_relative "../spec_helper.rb"

describe "erstelle_benutzer" do

  it "soll Benutzer einfuegen" do
    @user_name = "Sven"
    expect(plsql.benutzerverwaltung.erstelle_benutzer("Sven","Boettcher")).to equal(1)
    plsql.t_benutzer.delete(:vorname => 'Sven' , :nachname => 'Boettcher')
    plsql.commit
  end

  it "soll Benutzer nicht einfuegen" do
    benutzer = {:benutzer_id => plsql.seq_t_benutzer.nextval, :vorname => 'Sven', :nachname => 'Boettcher'}
    plsql.t_benutzer.insert benutzer
    expect(plsql.benutzerverwaltung.erstelle_benutzer("Sven","Boettcher")).to equal(0)
    plsql.t_benutzer.delete(:vorname => 'Sven' , :nachname => 'Boettcher')
    plsql.commit
  end

end
```

PL/SQL Unit Tests mit ruby-plsql-spec (ruby-plsql+RSpec)

- Code Coverage

CI_TEST.BENUTZERVERWALTUNG

NAME	TOTAL LINES	ANALYZED LINES	TOTAL COVERAGE	CODE COVERAGE
CI_TEST.BENUTZERVERWALTUNG	28	7	100.00%	100.00%

```
1   PACKAGE BODY benutzerverwaltung AS
2
3     FUNCTION erstelle_benutzer(
4               i_vorname   IN t_benutzer.vorname%TYPE,
5               i_nachname  IN t_benutzer.nachname%TYPE) RETURN NUMBER AS
6     BEGIN
7
8       INSERT INTO t_benutzer(
9          benutzer_id,
10         vorname,
11         nachname)
12       VALUES(
13         seq_t_benutzer.NEXTVAL,
14         i_vorname,
15         i_nachname);
16
17      COMMIT;
18
19      RETURN 1;
20
21      EXCEPTION
22        WHEN OTHERS THEN
23          ROLLBACK;
24          RETURN 0;
25
26    END erstelle_benutzer;
27
28  END benutzerverwaltung;
```

APEX Tests mit dem Selenium Webdriver

Jenkins

APEX Tests mit dem Selenium Webdriver

- Selenium:
  - Testumgebung für Webanwendungen (Firefox Plugin)

- Selenium Webdriver
  - API für Selenium
  - Ruby binding for Selenium → Verwendung von Selenium in Ruby/RSpec
  - ➢ Tests können in RSpec formuliert werden!

- Mit ruby-plsql kann gegen die Datenbank getestet werden

APEX Tests mit dem Selenium Webdriver

- Selenium Webdriver mit Ruby
  - Firefox Driver erstellen: @driver = Selenium::WebDriver.for(:firefox)
  - URL öffnen: @driver.navigate.to(@url)
  - Wert in Formularfeld schreiben:
    @driver.find_element(:id, “ID”).send_keys(“Wert”)
  - Auf einen Button klicken: @driver.find_element(:id,“ID").click

APEX Tests mit dem Selenium Webdriver

```
require "selenium-webdriver"
require_relative "spec_helper.rb"

describe "Benutzerverwaltung" do

    before(:all) do
        @driver = Selenium::WebDriver.for(:firefox)
    end

    before(:each) do
        @url = "http://192.168.56.102:8080/ords/f?p=102"
        @driver.navigate.to(@url)
        @driver.find_element(:id, "P101_USERNAME").clear
        @driver.find_element(:id, "P101_USERNAME").send_keys("admin")
        @driver.find_element(:id, "P101_PASSWORD").send_keys("Oracle1234!")
        sleep 5
        @driver.find_element(:xpath,"//button[@type='button']").click
        @url = @driver.current_url
    end

    after(:all) do
        @driver.quit
    end
```

APEX Tests mit dem Selenium Webdriver

```
    it "soll Benutzer einfuegen" do
        @url = @url.gsub("f?p=102:1","f?p=102:2")
        @driver.navigate.to(@url)
        @driver.find_element(:id,"P2_VORNAME").send_keys("Sven")
        @driver.find_element(:id,"P2_NACHNAME").send_keys("Boettcher")
        @driver.find_element(:id,"speichern").click
        expect(plsql.t_benutzer.count("WHERE vorname = :vorname and nachname = :nachname", "Sven", "Boettcher")).to eq(1)
        plsql.t_benutzer.delete(:vorname => 'Sven' , :nachname => 'Boettcher')
        plsql.commit
    end

    it "soll Benutzer nicht einfuegen" do
        benutzer = {:benutzer_id => plsql.seq_t_benutzer.nextval, :vorname => 'Sven', :nachname => 'Boettcher'}
        plsql.t_benutzer.insert benutzer
        plsql.commit
        @url = @url.gsub("f?p=102:1","f?p=102:2")
        @driver.navigate.to(@url)
        @driver.find_element(:id,"P2_VORNAME").send_keys("Sven")
        @driver.find_element(:id,"P2_NACHNAME").send_keys("Boettcher")
        @driver.find_element(:id,"speichern").click
        sleep 5
        expect(plsql.t_benutzer.count("WHERE vorname = :vorname and nachname = :nachname", "Sven", "Boettcher")).to eq(1)
        plsql.t_benutzer.delete(:vorname => 'Sven' , :nachname => 'Boettcher')
        plsql.commit
    end

end
```

APEX Best Practices

- Probleme Entwicklung im Team
- Architektur für Teamentwicklung
- Installation JENKINS, plsql-jenkins-plugin,sql-ruby/sql-ruby-spec
- Konfiguration Maven build Prozess (pom) für plsql
- Konfiguration JENKINS für plsql build Prozess
- Konfiguration sql-ruby-spec für Unit test
- Kann mit sql-ruby-spec Session info gesetzt werden?
- Wie sollten DDL aussehen / Exception Blöcke um doppelte Objekte zu vermeiden:

```
WHEN OTHERS THEN
  IF SQLCODE = -955 THEN
    NULL;
  ELSE
    RAISE;
  END IF;)
```

- SQL Developer und svn
- Build von PLSQL mit maven exec plugin → bei Fehlerhaften Compileprozessen kann der build Prozess als fehlerhaft markiert werden
- CI von APEX Applikationen → Fehler, wenn z.B. Funktionen umbenannt werden?

Fazit

- Continous Integration ist mit APEX möglich
- Die Art und Weise der Entwicklung ändert sich
- Das Schreiben von Tests kann sehr zeitaufwändig sein
- Es wird mehr Infrastruktur benötigt

- Aber:
  - Die Qualität der Software hinsichtlich der Korrektheit kann stark verbessert werden
  - Das Deployment auf UAT-/Produktiv- und anderen System kann anschließend sehr einfach von statten gehen oder sogar automatisch erfolgen (Continous Deployment)

Vielen Dank

ORACLE Platinum Partner

amazon web services | Partner Network
ADVANCED CONSULTING PARTNER

Microsoft CERTIFIED Partner

North America
- Boston (Headquarters)
- New York
- Atlanta
- Chicago

Asia
- India
  Global Development Center

Europe
- Germany
- Netherlands

Middle East
- Oman
- Dubai

www.appsassociates.com