PL/SQL Unit Tests mit SQL Developer

Perry Pakull

Principal Consultant

Trivadis AG

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

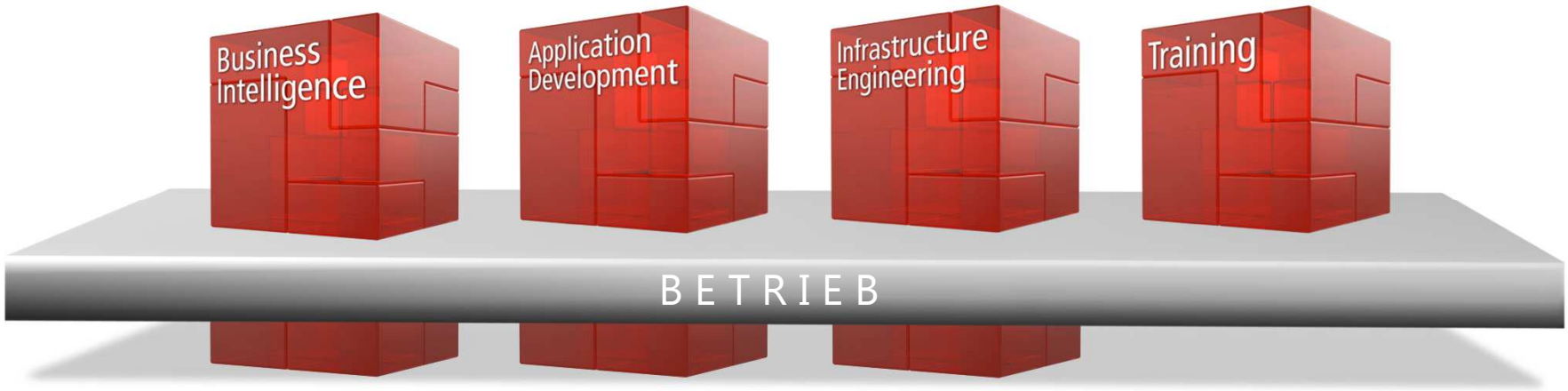
@PerryPakull

- Principal Consultant
  - Trivadis AG in Zürich (CH)
  - perry.pakull@trivadis.com
- Oracle Application Development
  - SQL und PL/SQL
  - Forms und Reports
  - APEX
  - BI Publisher
- Architektur, System Design, Datenmodellierung
- Modernisierung
  - Forms und Reports

Unser Unternehmen

Trivadis ist führend bei der IT-Beratung, der Systemintegration, dem Solution-Engineering und der Erbringung von IT-Services mit Fokussierung auf ORACLE und Microsoft Technologien im D-A-CH-Raum.

Unsere Leistungen erbringen wir aus den strategischen Geschäftsfeldern:



Trivadis Services übernimmt den korrespondierenden Betrieb Ihrer IT Systeme.

Mit über 600 IT- und Fachexperten bei Ihnen vor Ort

12 Trivadis Niederlassungen mit über 600 Mitarbeitenden

200 Service Level Agreements

Mehr als 4'000 Trainingsteilnehmer

Forschungs- und Entwicklungs-budget: CHF 5.0 Mio. / EUR 4.0 Mio.

Finanziell unabhängig und nachhaltig profitabel

Erfahrung aus mehr als 1'900 Projekten pro Jahr bei über 800 Kunden

Stand 12/2013

AGENDA

1. Einleitung
2. Unit Test Repository erstellen
3. Unit Tests erstellen und ausführen
4. Best Practice
5. Automatisierung von Unit Tests
6. Exportieren und Importieren von Unit Tests
7. Fazit

Einleitung

Unit Tests

- Unit Tests sind ein aufwendiger aber auch notwendiger Bestandteil eines Software Entwicklungsprozesses
- Vorhandene Frameworks im Oracle Datenbankumfeld sind gering
- SQL Developer bietet eine Vielzahl von Assistenten und Funktionen für den Entwickler, die den Einstieg in Unit Tests für PL/SQL Programme erleichtern
- Vortrag beschreibt die technischen Möglichkeiten des SQL Developers für die Erstellung, Verwaltung und Automatisierung von Unit Tests

Unit Test Repository erstellen

Unit Test Repository

SQL Developer benötigt ein Repository für Unit Test Funktionen

- Repository in einer Oracle Datenbank
  - Repository besteht aus Tabellen, Views und anderen Datenbankobjekten
- Eigenes Schema für das Repository
- Installation über SQL Developer Benutzeroberfläche

Installation

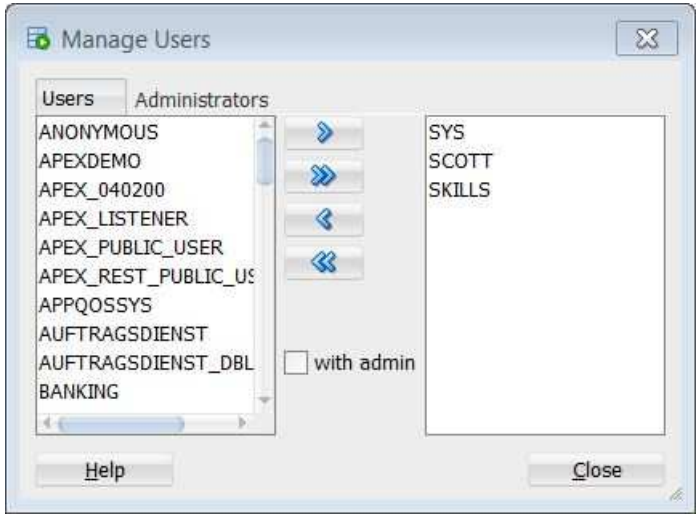
3 Schritte für die Installation des Repositories

- Datenbankbenutzer UNIT_TEST_REPOS für Repository anlegen
  - Verbindung mit DBA Berechtigungen erforderlich
  - Rechts-Klick "Other Users", Kontextmenü "Create User"
  - System Privileg CREATE SESSION
- Verbindung für Datenbankbenutzer UNIT_TEST_REPOS erstellen
  - Auswahl "Tools, Unit Test, Manage Users"
  - Neue Verbindung unit_test_repos
  - Berechtigungen werden vergeben
- Repository erstellen
  - Menüpunkt "Tools, Unit Test, Select Current Repository"
  - Verbindung unit_test_repos verwenden
  - Repository Objekte werden angelegt

Repository Berechtigungen

Berechtigungen über Datenbankrollen

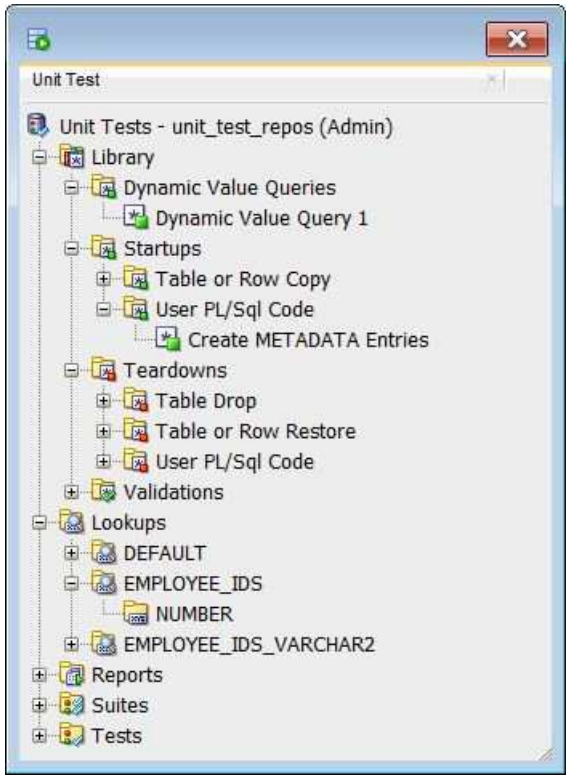
- Rolle UT_REPO_USER
  - User
  - Unit Tests erstellen
  - Unit Tests ausführen
- Rolle UT_REPO_ADMINISTRATOR
  - Administratoren
  - Betrieb Repository
  - Verwaltung Repository
  - Repository Berechtigungen vergeben



Repository Objekte

Objektkategorien im Repository

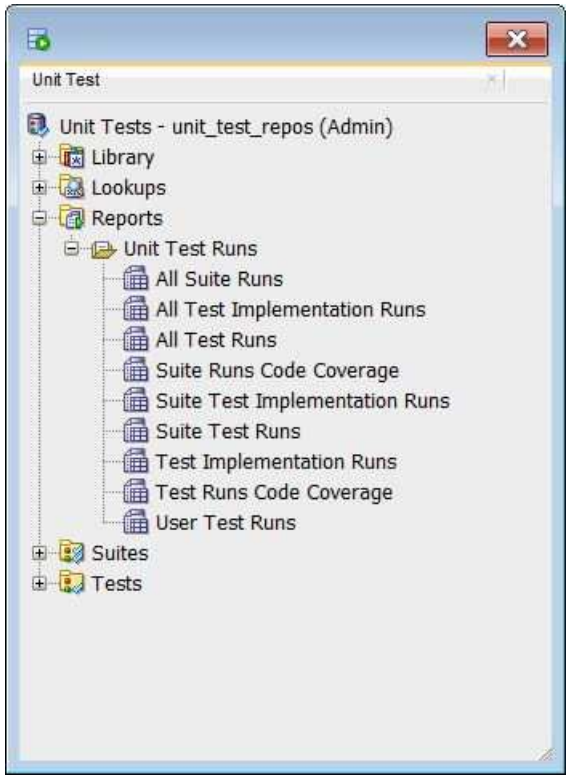
- Library
  - Wiederverwendbare Unit Test Komponenten
  - Dynamic Value Queries
  - Startups
  - Teardowns
  - Validations
- Lookups
  - Daten für Eingabeparameter
  - Kategorisiert nach Datentypen
  - Eine Test Implementierung pro Wert bei Neuanlage Unit Test



Repository Objekte

Objektkategorien im Repository

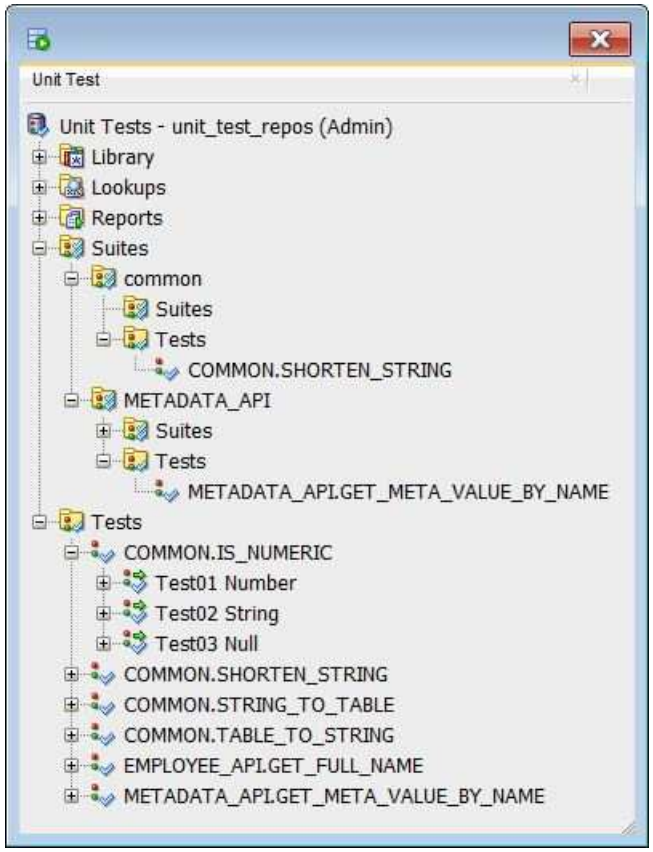
- Reports für Auswertungen der Testergebnisse
  - All Suite Runs
  - All Test Implementation Runs
  - All Test Runs
  - Suite Runs Code Coverage
  - Suite Test Implementation Runs
  - Suite Test Runs
  - Test Implementation Runs
  - Test Runs Code Coverage
  - User Test Runs (test runs grouped by user)



Repository Objekte

Objektkategorien im Repository

- Suites
  - Gruppe von Unit Tests
  - Kann Suites enthalten
- Tests
  - Unit Tests
    - Test Implementierungen

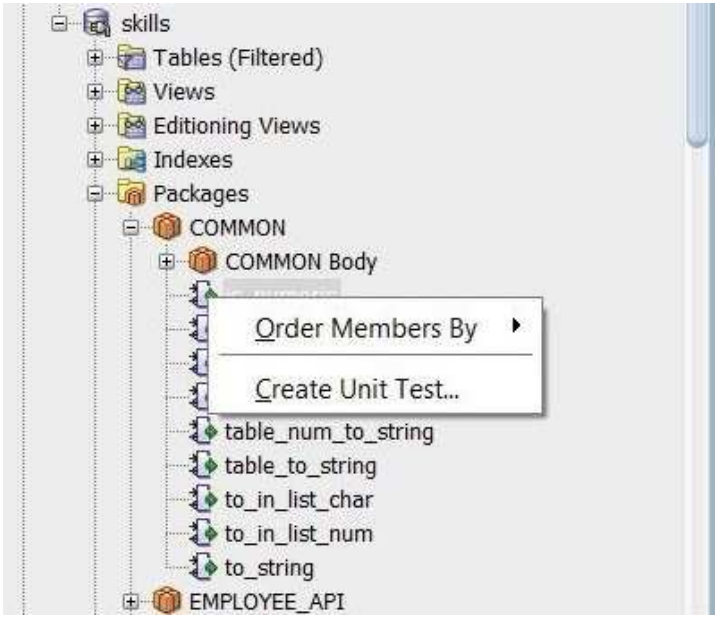


Unit Tests erstellen und ausführen

Unit Test erstellen

Einstieg in die Erstellung

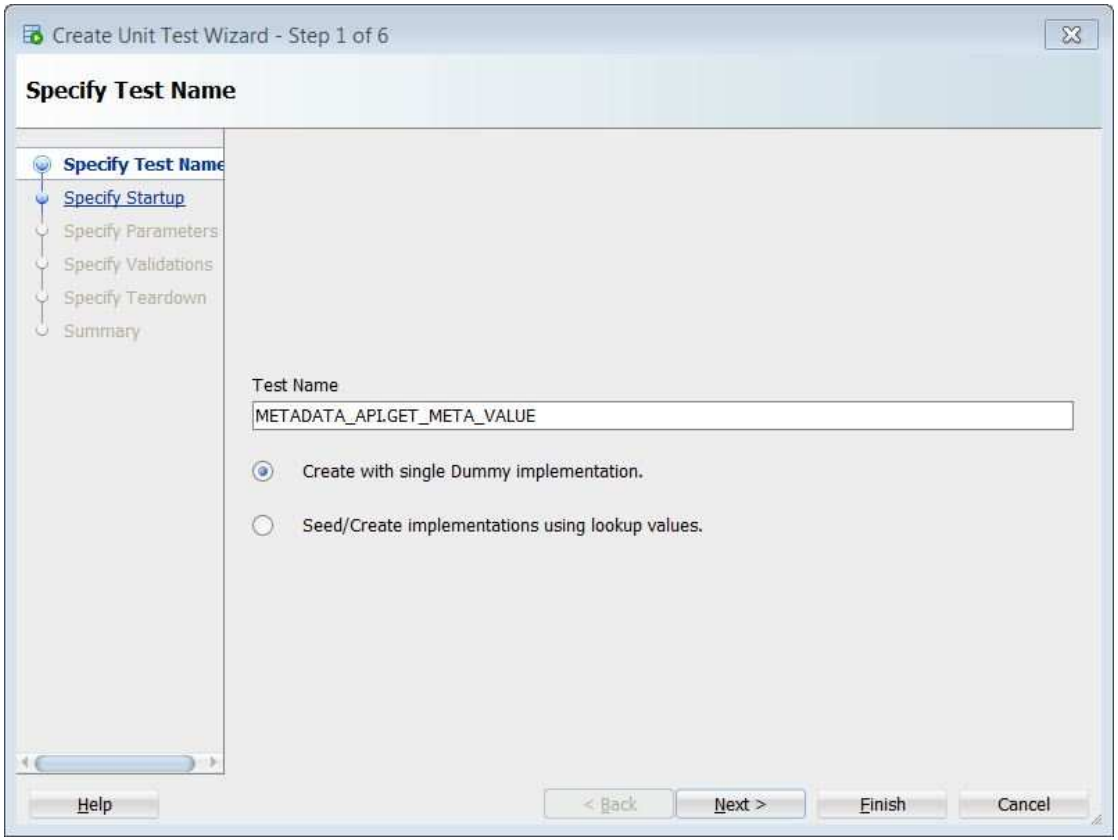
- Rechts-Klick auf eine Prozedur, Funktion oder Methode eines Packages im Navigator
- Rechts-Klick auf den Knoten Tests im Navigationsfenster für Unit Tests



Unit Test erstellen Schritt 1 Test Name

Test Name definieren

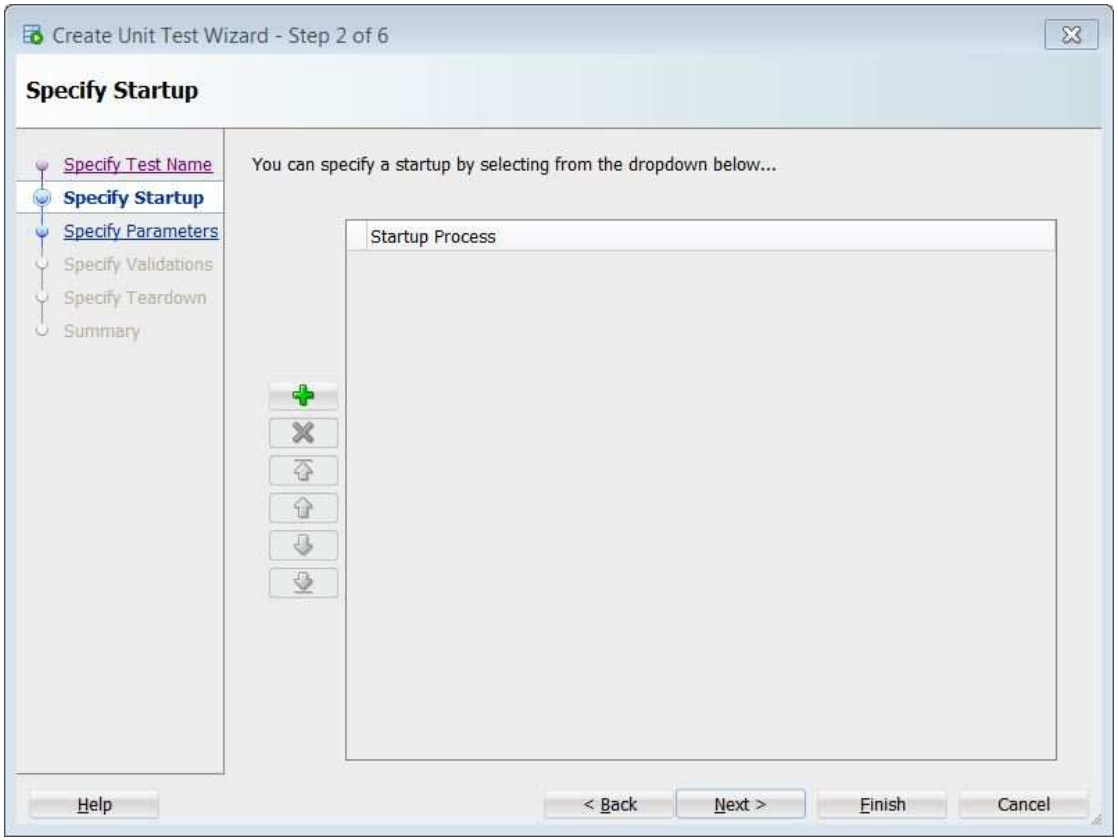
- Name wird vom Datenbankobjekt übernommen
- Eine Dummy Implementierung
- Option Lookup Values für mehr Implementierungen



Unit Test erstellen Schritt 2 Startup

Startup Prozesse

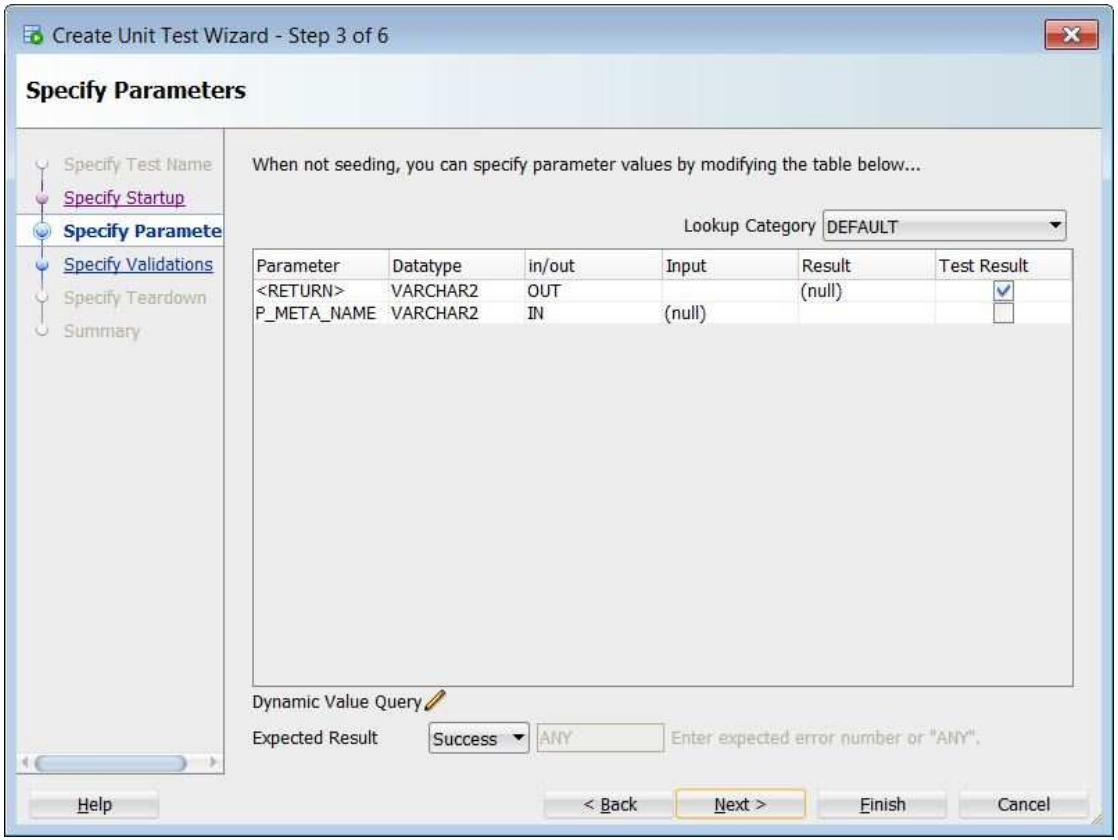
- Testkonstellation für Objekte und Daten aufbauen
- Prozesse
  - Table or Row Copy
  - User PL/SQL Code



Unit Test erstellen Schritt 3 Parameter

Parameterwerte eingeben

- IN Parameter
- OUT Parameter als Testergebnis
- Return-Werte für Funktionen als Testergebnis
- Dynamic Value Query
- Erwartetes Ergebnis



Unit Test erstellen Schritt 3 Parameter

Dynamic Value Query

- Syntax

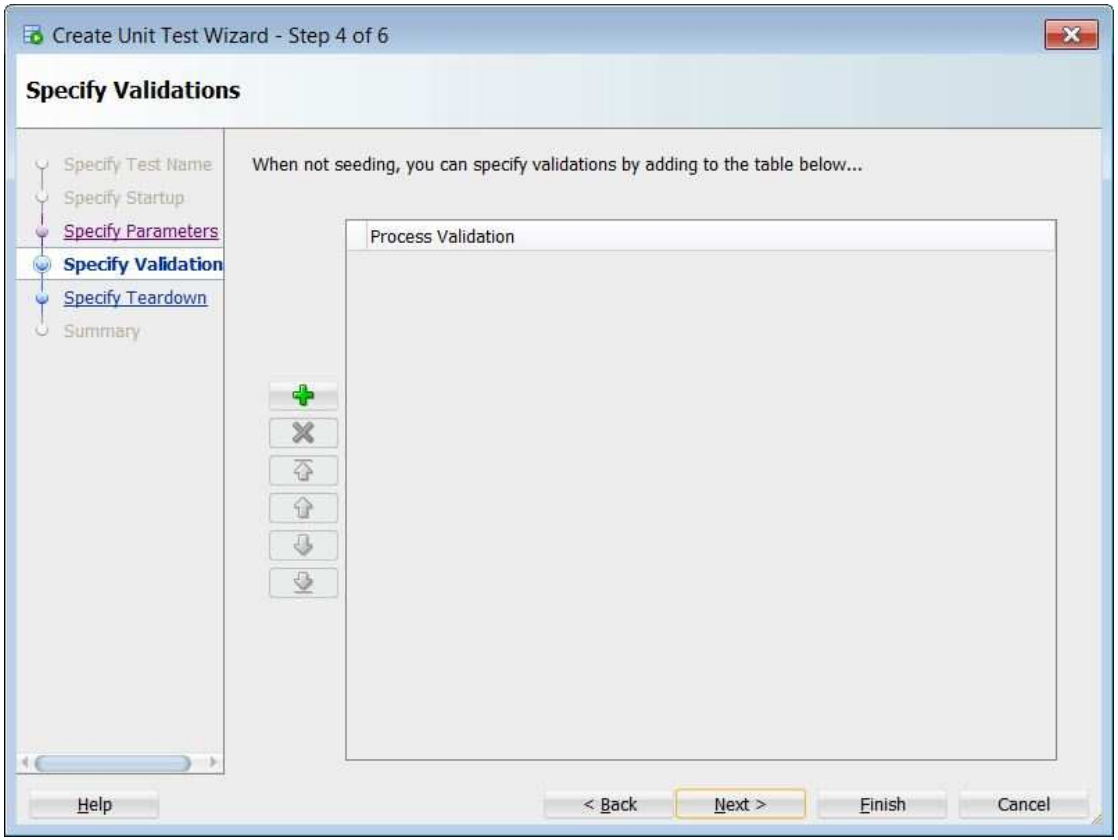
```
select ? as RETURNS$, ? as P_EMP_ID from ? where ?
```

- Alias für die Zuordnung der Werte zu den Parametern
- Implementierung wird entsprechend der Anzahl der Datensätze ausgeführt
- Komponente in der Unit Test Library

Unit Test erstellen Schritt 4 Validierungen

Validierungen

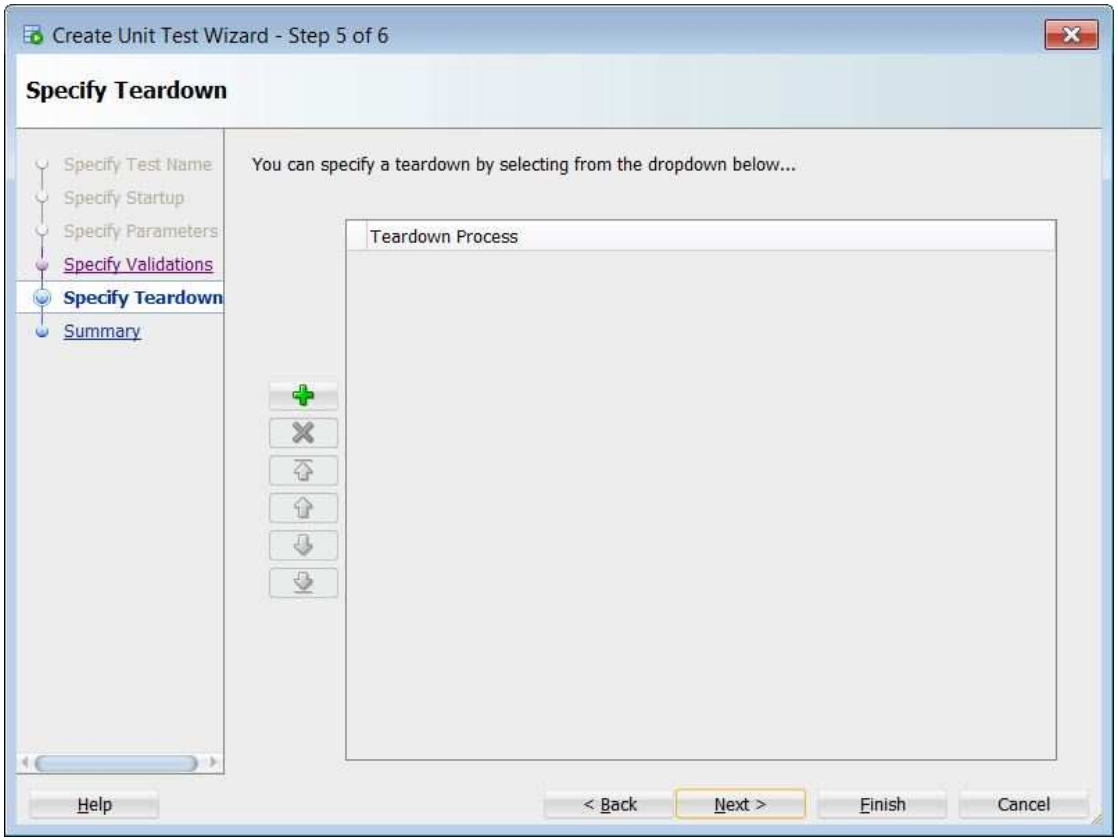
- Testergebnisse prüfen
- Prozesse
  - Boolean Function
  - Compare Query Results
  - Compare Tables
  - Query returning no rows
  - Query returning rows
  - User PL/SQL Code



Unit Test erstellen Schritt 5 Teardown

Teardown Prozesse

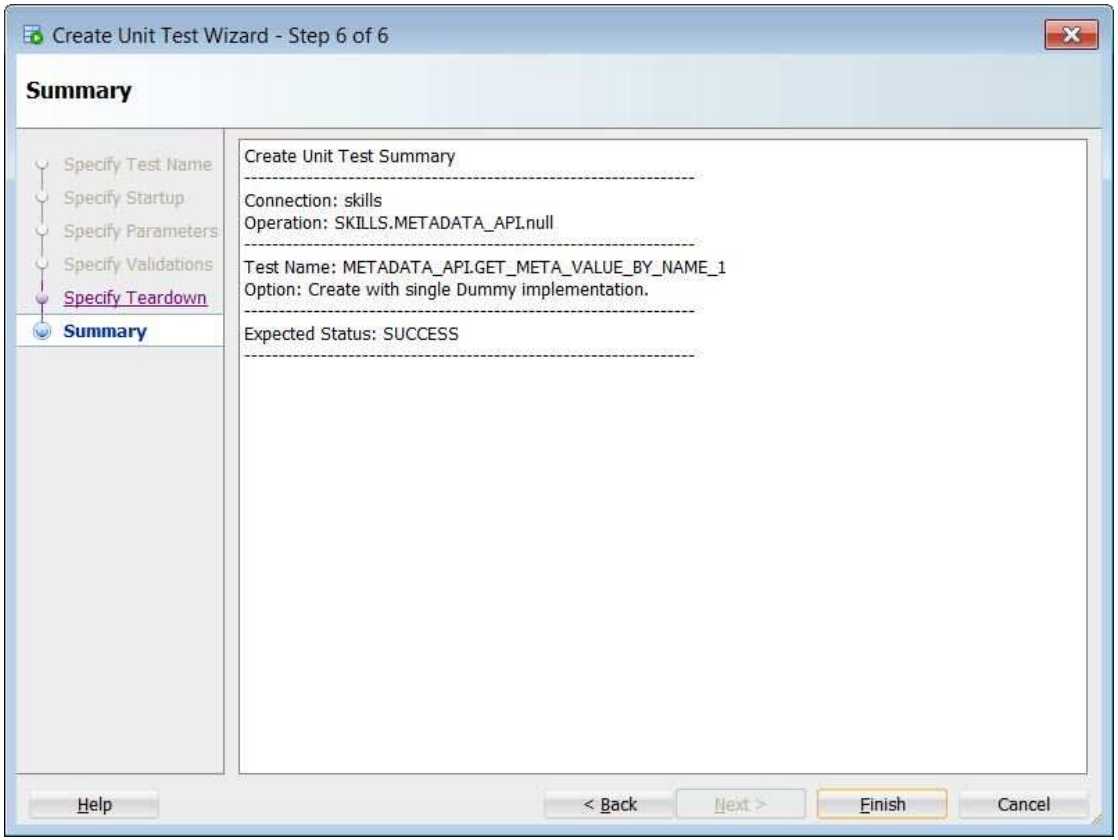
- Testkonstellation für Objekte und Daten aufräumen
- Prozesse
  - Table Drop
  - Table or Row Restore
  - User PL/SQL Code



Unit Test erstellen Schritt 6 Summary

Summary

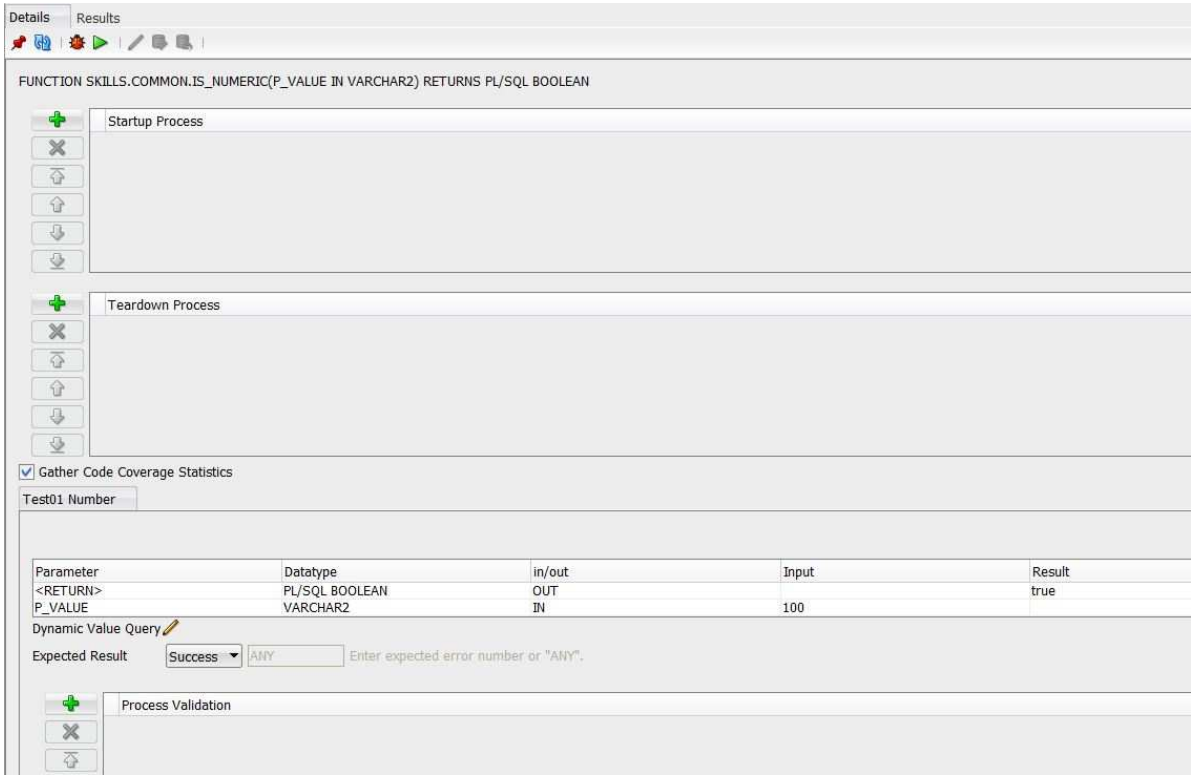
- Zusammenfassung der Definitionen und Eingaben



Unit Test Editor

Unit Test bearbeiten

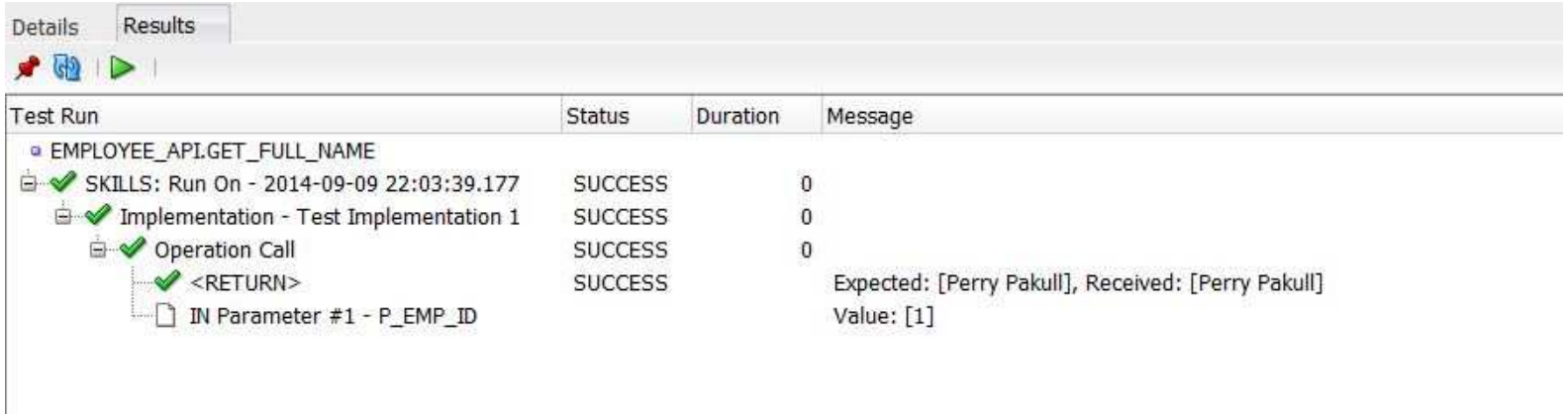
- Klick auf den Namen im Navigator
- Vollständige Übersicht
- Alle Komponenten änderbar
- Weitere Funktionen im Kontextmenü über Rechts-Klick
  - Löschen, Umbenennen, Kopieren, Implementierung ergänzen, Ausführen, Testergebnisse löschen, Synchronisation, Export



Unit Test ausführen

Unit Test ausführen mit F9

- Synchrone Ausführung
- Anzeige der Ergebnisse



Best Practice

Empfehlungen Strategie

- Lektüre der SQL Developer Hilfe für den Einstieg und das Verständnis
  - Vollständiges Tutorial
  - Oracle Empfehlungen für die Verwendung der Objekte und Funktionen
- Strategie für die Vorgehensweise, Granularität der Tests definieren
  - Ein Unit Test adressiert die kleinste testbare Einheit einer Software
    - Einzelne Prozeduren und Funktionen
    - Prozeduren und Funktionen in einem Package oder Object Type Body
  - Strategie für neue Applikationen
    - Parallel zur Entwicklung des Codes erfolgt Entwicklung der Unit Tests
  - Strategie für bestehende Applikationen
    - Zunächst Einteilung der PL/SQL Objekte in funktionale Gruppen
    - Gruppen als Suites abbilden
    - Schrittweise einzelne Unit Tests den Suites zuordnen

Empfehlungen Namenskonventionen

- Name für einen Unit Test ist auf 120 Zeichen begrenzt
- Name muss eindeutig sein
- Eine sinnvolle Benennung ist daher wichtig
- Name des Datenbankobjektes übernehmen
- Einfache Zuordnung der Tests zu den Datenbankobjekten
- Nummerierung durch eine sinnvolle Erweiterung im Namen anpassen
  - Mehr als ein Test pro Datenbankobjekt oder unterschiedliche Versionen
- Name des Schemas oder der Datenbank oder der Umgebung ergänzen
  - Datenbankobjekte mit gleichem Namen aus unterschiedlichen Schemas oder Datenbanken

Empfehlungen Test Implementierungen

- Benennung sinnvoll an den Inhalt der Test Implementierung anpassen
- Vorgeschlagene Nummerierung der Implementierungen ist nicht aussagekräftig
- Jeder Test sollte eine Implementierung mit den Standard-Eingabewerten enthalten
- Weitere Implementierungen mit NULL-Werten, Minimum- und Maximum-Werten

Empfehlungen Gruppierung

- Verwendung von Test Suites für eine Gruppierung der Tests
- Unit Tests zu einem Package in einer Suite mit dem Package Namen gruppieren
- Suites hierarchisch ordnen
- Master-Suite für die Ausführung aller Tests

Empfehlungen Unit Test Library

- Modularisierung der Komponenten
- Wiederverwendung möglich
- Objekte aus der Library kopieren oder vererben

Empfehlungen Command-Line Interface

- Automatisierte Ausführung der Tests mit Command-Line Interface
- Auswertung der Testergebnisse durch Abfragen der Ergebnis-Tabellen im Repository

Empfehlungen Repository Installation

- Unit Test Repository in eigenes Datenbankschema installieren
- Ein Repository pro Datenbank ist ausreichend
- Bei Bedarf können auch mehrere installiert werden

Automatisierung von Unit Tests

Software Entwicklungsprozesse

Moderne Software Entwicklung

- Continuous Integration
- Continuous Delivery
- Automatisierte Tests sind zentraler Bestandteil

SQL Developer Command-Line Interface

Ausführung von Unit Tests mit dem Command-Line Interface

```
sdcli64 unittest -run
                  -test | -suite
                  -id <id>|-name <name>
                  -repo <connection name>
                  -db <connection name>
                 {-return <return id>}
                 {-log <0,1,2,3>}
```

Parameter

test | suite	Unit Test oder Test Suite
id | name	Interne ID oder Name
repo	SQL Developer Verbindung für Unit Test Repository
db	SQL Developer Verbindung für Entwicklungsschema
return	ID für Testergebnisse

Export und Import

Export

Export der Unit Test Objekte über SQL Developer Benutzeroberfläche

- Rechts-Klick auf Unit Test, Suite, Dynamic Value Query, Startup, Teardown, Validation
  - Kontextmenü "Export to File"
- Mehrere Objekte markieren
  - Export in eine Datei
- Dateiformat XML
- Export enthält die markierten Objekte und alle abhängigen Objekte
  - Export Suite enthält alle zugeordneten Unit Tests
  - Abhängige Objekte aus Repository Library

Export Command-Line Interface

Export von Repository Objekten mit dem Command-Line Interface

```
sdcli64 unittest -export
                 -test | suite | validation | startup |
                         teardown | query
                 -id <id> | -name <name>
                 -repo <connection name>
                 -file <file name>
```

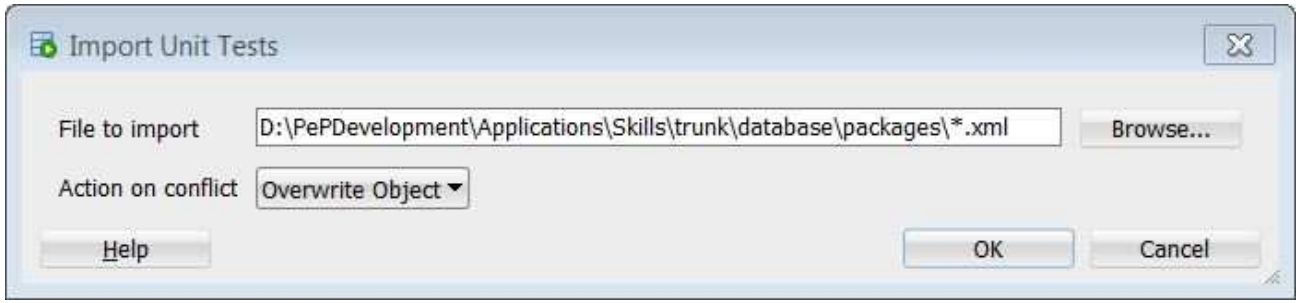
Parameter

object	Repository Objekt: Test, Suite, Validation Prozess, Startup Prozess, Teardown Prozess, Dynamic Value Query
id | name	Interne ID oder Name
repo	SQL Developer Verbindung für Unit Test Repository
file	Pfad/Name der Export Datei

Import

Import der Unit Test Objekte über SQL Developer Benutzeroberfläche

- Auswahl "Tools, Unit Test, Import from File"
- Alle Objekte werden neu angelegt
- Konfliktlösungsoptionen, wenn Objekte mit dem gleichen Namen bereits vorhanden sind
  - Überschreiben
  - Auslassen



Import Command-Line Interface

Import von Repository Objekten mit dem Command-Line Interface

```
sdcli64 unittest -imp
                  -repo <connection name>
                  -file <file name>
                 [-overwrite | -skip]
```

Parameter

repo	SQL Developer Verbindung für Unit Test Repository
file	Pfad/Name der Import Datei
overwrite | skip	Konfliktlösung

Verwendungsmöglichkeiten

Export und Import Verwendungsmöglichkeiten

- Transfer in ein anderes Repository
- Backup
- Export Dateien in Versionskontrolle übernehmen
- Automatisierung

Fazit

Gesamteindruck

- Der Gesamteindruck ist durchweg positiv
- Gute, integrierte Umgebung für die Entwicklung von Code und Tests

Vorteile

- Schlüssige und vernünftige Implementierung
- Einfache und intuitive Funktionen
  - Erstellung eines Tests
- Ergänzung der Funktionen durch sinnvolle Vorlagen, keine Einschränkungen
  - Startup Prozesse, Teardown Prozesse
- Command-Line Interface hilfreich für Automatisierung
  - Ausführung, Export/Import

Nachteile

- Lookup-Funktionalität bei der Erstellung der Test Implementierungen ist umständlich
  - Gewünschte Kategorie kann nicht aus einer Liste gewählt, sondern nur umständlich in den Einstellungen vorgegeben werden
- Validierungsprozesse können nicht auf benutzerdefinierte Datentypen zugreifen
  - Der Zugriff auf die Werte ist zwar möglich, aber nur über den Datentyp VARCHAR2
- Das Repository Navigationsfenster bietet keine Möglichkeit nach Tests oder anderen Objekten zu suchen
  - Bei vielen Objekten geht die Übersicht schnell verloren

Weitere Informationen...

Oracle Technology Network

http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html

Fragen und Antworten...

Perry Pakull

Principal Consultant

Telefon +41 79 264 88 37

perry.pakull@trivadis.com

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

20 JAHRE TRIVADIS We love IT.

trivadis makes IT easier.

Trivadis an der DOAG

Ebene 3 - gleich neben der Rolltreppe

Wir freuen uns auf Ihren Besuch.

Denn mit Trivadis gewinnen Sie immer.