HERZLICH WILLKOMMEN

Continuous Database Integration

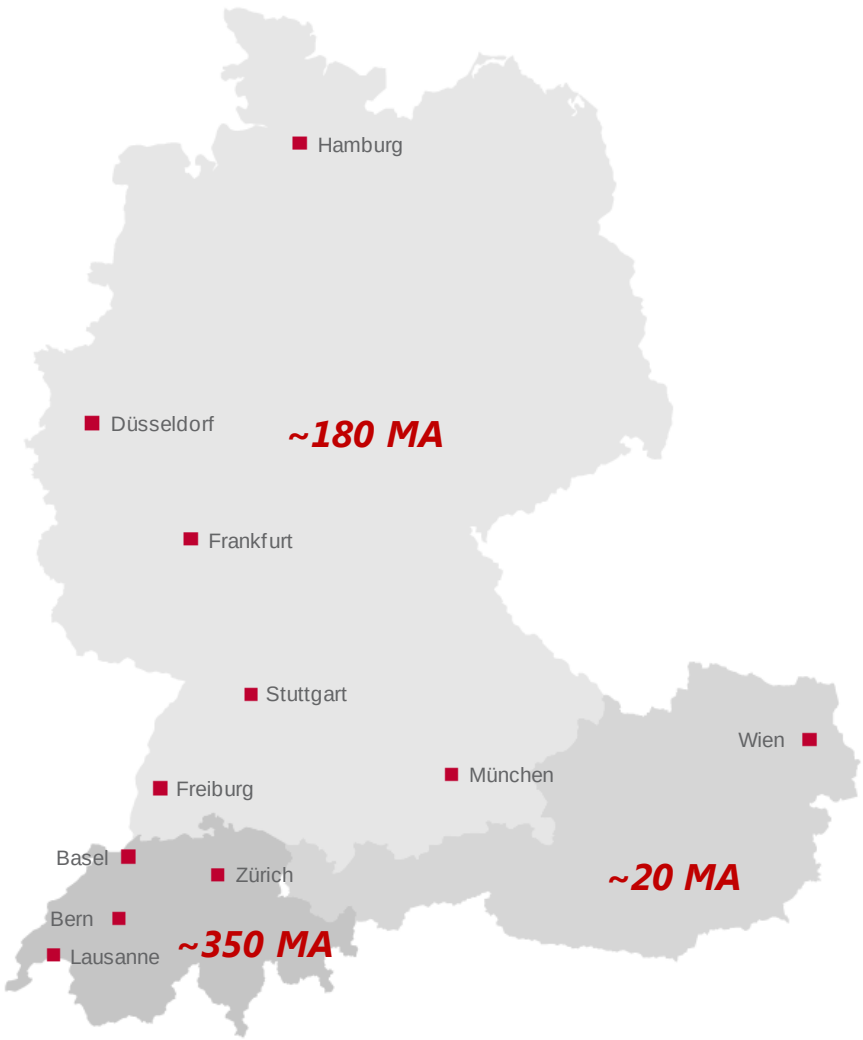
Andrej Pashchenko

Köln, 22.09.2011

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

trivadis
makes IT easier.

Trivadis Facts & Figures



11 Trivadis Niederlassungen mit über 550 Mitarbeitern

Finanziell unabhängig und nachhaltig profitabel

Kennzahlen 2010

- Umsatz CHF 101 / EUR 73 Mio.
- Dienstleistungen für über 700 Kunden in mehr als 1'800 Projekten
- Über 170 Service Level Agreements
- Mehr als 5'000 Trainingsteilnehmer
- Forschungs- und Entwicklungs-budget: CHF 5.0 / EUR 3.6 Mio.

ORACLE Platinum Partner

Microsoft GOLD CERTIFIED Partner

trivadis makes IT easier.

AGENDA

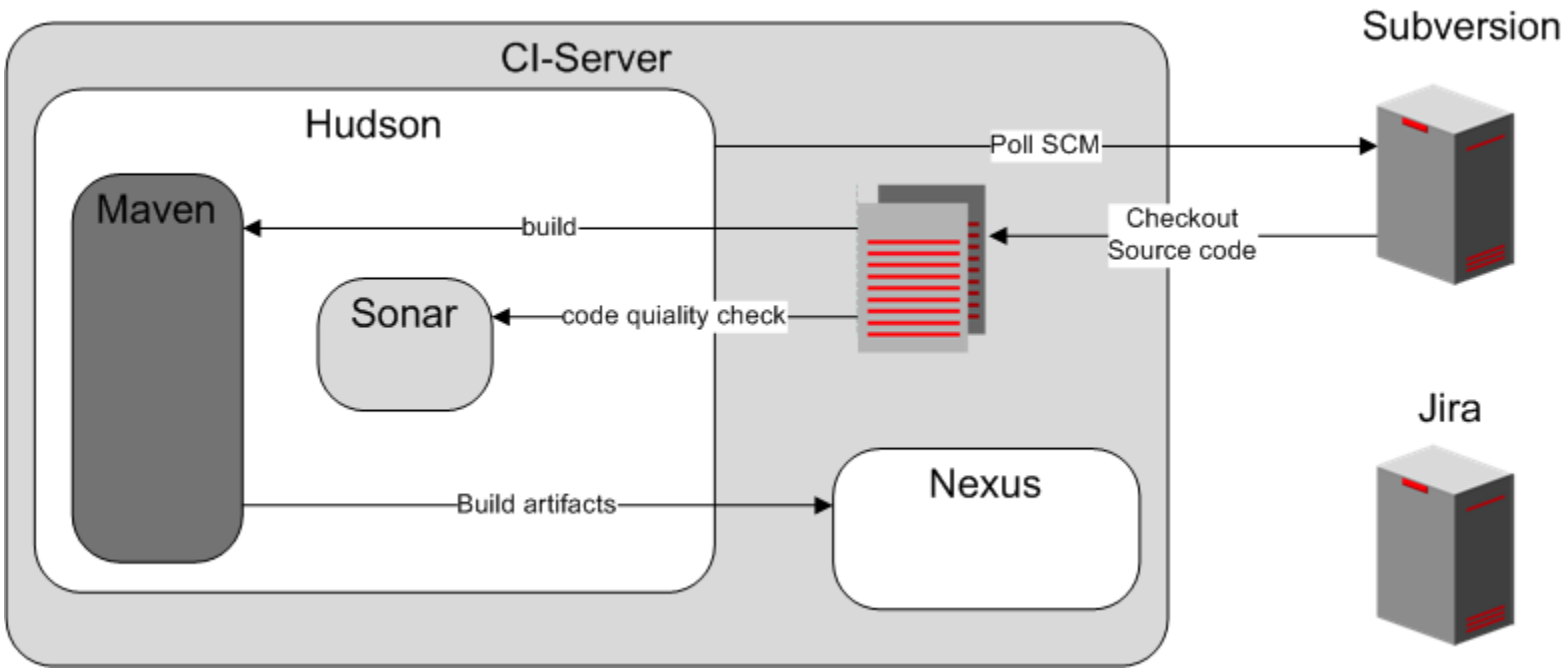
- Einführung: CI und CDBI
- Build-Prozess
  - DB-Change-Tool
  - Verbindungsaufbau
- Unit-Tests
- What's next?
- Referenzen

Einführung

- Continuous Integration ist eine Best Practice:
  - Frühzeitig und häufig integrieren
- Vorteile:
  - Risikominimierung: nachgelagerte Integration erschwert die Sichtbarkeit des Projektstandes
  - Integrationsprobleme frühzeitig erkennen
  - Fehler sind einfacher zu beheben
  - Aktueller Stand ist immer verfügbar als Demo oder für QA
- Nachteile:
  - Initialer Aufwand
  - Test-Suite für automatisierte Tests benötigt
- Voraussetzungen für Continuous Integration:
  - Schneller, vollautomatisierter Build, der Tests beinhaltet
  - Ein zentrales Quellcode-Repository
  - Der Build wird in einem Produktionsklon durchgeführt

Einführung

Was fehlt hier?

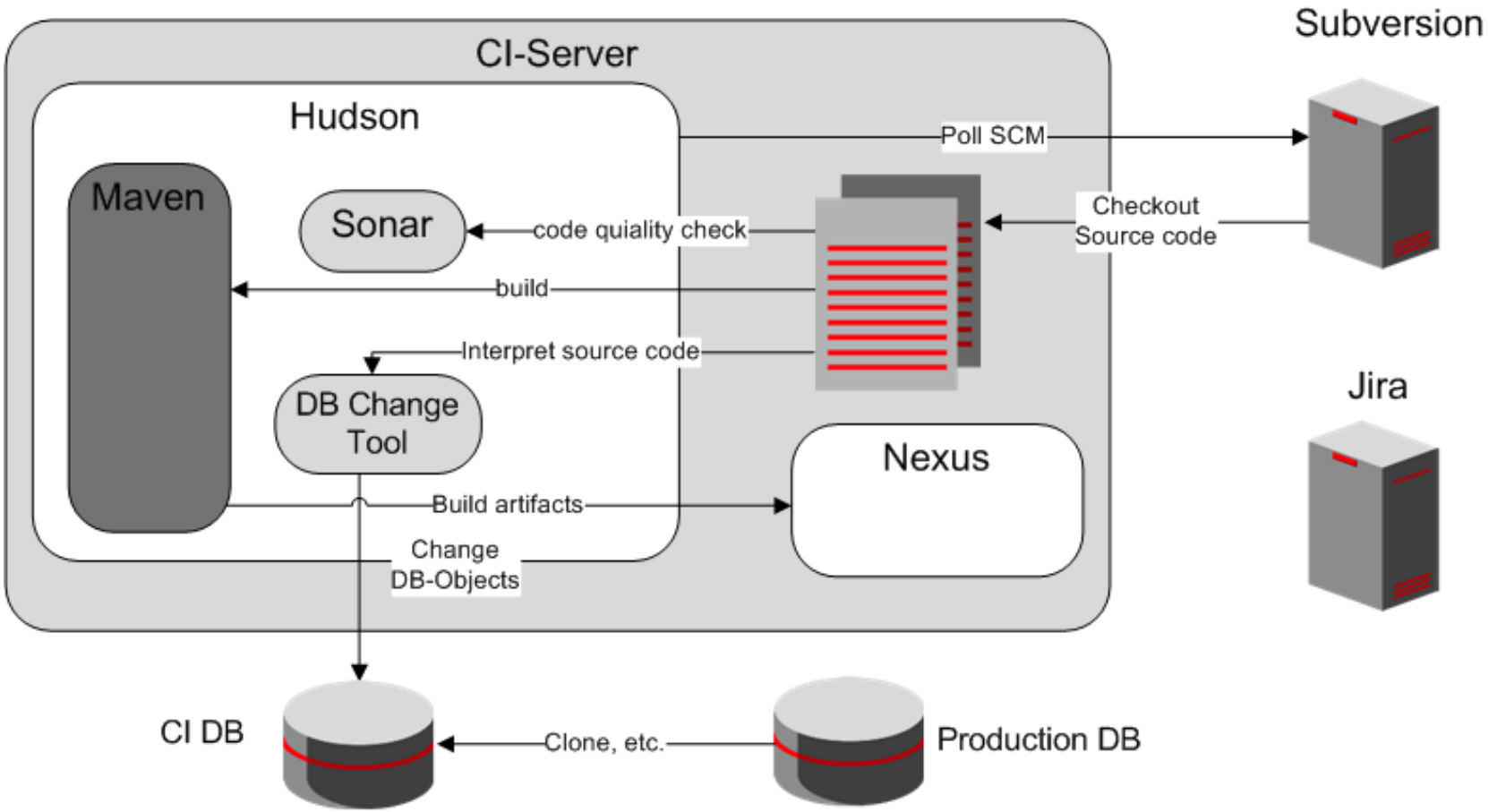


Einführung

Warum ist die Datenbankentwicklung anders?

- Unmöglich komplett neu aufzubauen
- Build ist eine Änderung an einem existierenden DB-Schema
- Benötigt einen DB-Stand, auf den die Änderungen angewendet werden
- Normalerweise besteht die Änderung aus:
  - Änderungen an Datenstrukturen
  - Datenmigration
  - Änderungen am Datenzugriffscode
- Objekte wie Views oder PL/SQL können immer überschrieben werden
- Andere Objekte wie Tabellen erfordern Änderung durch ALTER-DDL-Statements und womöglich eine Datenmigration

Einführung



AGENDA

- Einführung: CI und CDBI
- Build-Prozess
  - DB-Change-Tool
  - Verbindungsaufbau
- Unit-Tests
- What's next?
- Referenzen

Build-Prozess: DB-Change Tool

Was leistet ein DB-Change Tool?

- Hilft die Build-Logik aus dem Quelltext auszulagern:
  - Was ist bereits ausgeliefert worden?
  - Was ist in einer Fehlersituation zu tun?
  - In welcher Reihenfolge sollen die Änderungen angewendet werden?
  - Protokollieren

Anforderungen:

- Alle benötigten Änderungsskripte in definierter Reihenfolge und nur einmalig ausführen
- Im Fehlerfall die Weiterverarbeitung stoppen
- Protokoll-Tabelle in der DB, um Ausführung und den Erfolgsstatus festzuhalten
- Abgestimmt mit der Art und Weise wie der Quellcode organisiert ist (Verzeichnisstruktur, Inhalt der Skripte, Verbindungsaufbau)

DB-Change Tool

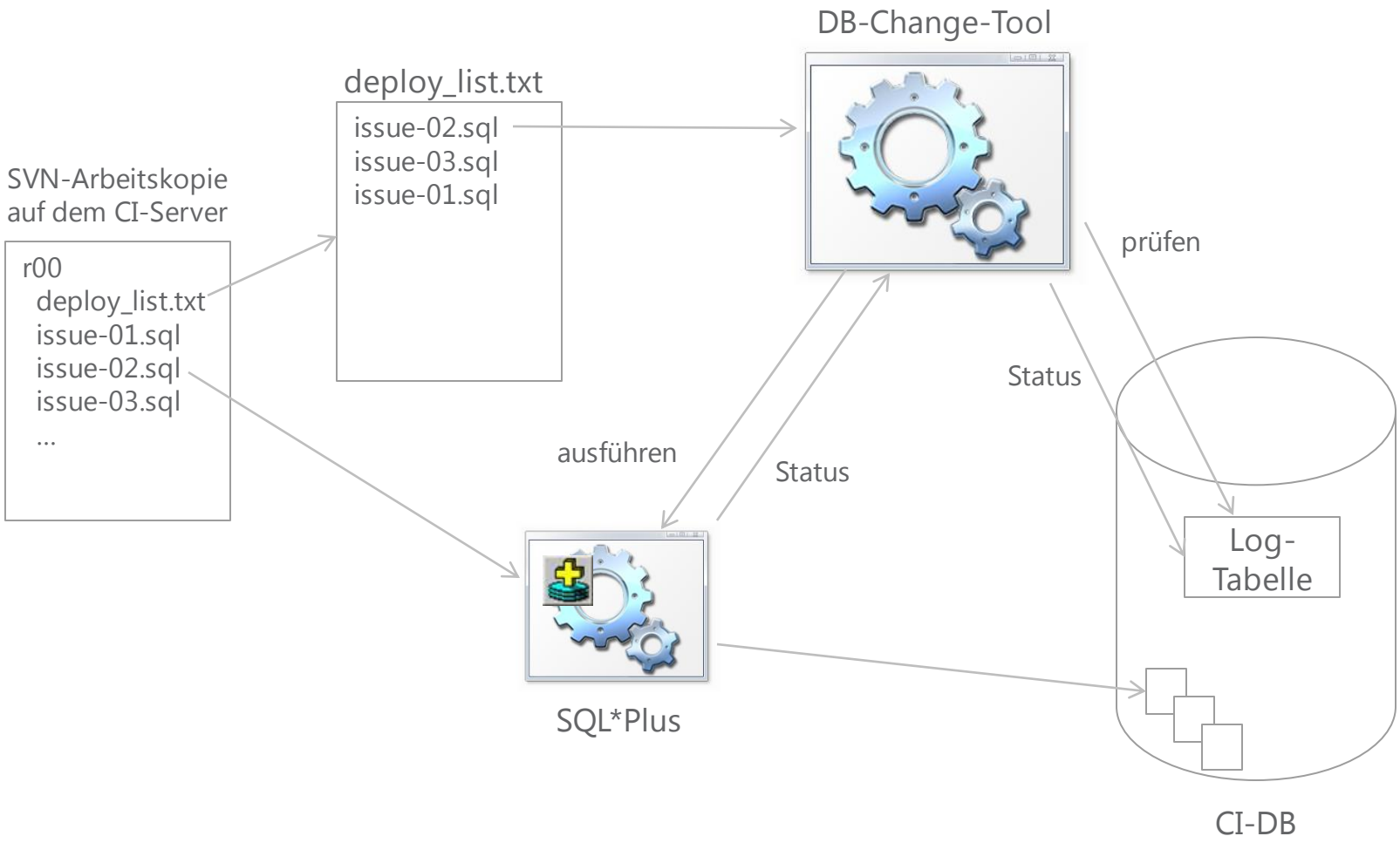
- Vorhandene Open-Source DB-Change Tools:
  - Liquibase, http://www.liquibase.org/
  - Flyway, http://code.google.com/p/flyway/
  - Dbmaintain, http://www.dbmaintain.org/
  - c5-db-migration, http://code.google.com/p/c5-db-migration/
  - Dbdeploy, http://dbdeploy.com
  - Autopatch, http://autopatch.sourceforge.net/
  - migratedB, http://migratedb.sourceforge.net/
- Nachteile:
  - Sehr dünne Benutzer- und Entwickler-Basis
  - Zum Teil sehr sparsame Dokumentation
  - Weiterentwicklung kann jederzeit eingestellt werden
  - Explizite Reihenfolge der Änderungen kann nur bei Liquibase (als XML) definiert werden

DB-Change Tool

- Alternative: Trivadis-Lösung auf der Basis der Shell- und SQL-Skripte
- Vorteile:
  - Gut abgestimmt mit evtl. vorhandener Vorgehensweise bei Auslieferung
  - Ausführungsumgebung SQL*Plus
  - Oracle Client Features nutzen
- Auszug aus POM.XML

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <executions>
    <execution>
      <id>sqldeploy</id>
      <phase>process-resources</phase>
      <goals> <goal>exec</goal> </goals>
      <configuration>
        <executable>dbd_start.sh</executable>
        <workingDirectory>${basedir}/src/main/db/${change_dir}/</workingDirectory>
      </configuration>
    </execution>
  </executions>
</plugin>
```

DB-Change Tool



AGENDA

- Einführung: CI und CDBI
- Build-Prozess
  - DB-Change-Tool
  - Verbindungsaufbau
- Unit-Tests
- What's next?
- Referenzen

DB-Change Tool

- Verbinden mit Oracle Secure External Password Store
  - Verbindungsdaten (Benutzer, Passwort, TNS-Alias) werden verschlüsselt in einem Wallet abgespeichert
- SQLNET.ORA

```
WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = C:\tns) ) )

SQLNET.WALLET_OVERRIDE = TRUE
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_VERSION = 0
```

- TNSNAMES.ORA

```
crm_at_cidb = (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = yourhost.com)(PORT = 1521))
    (CONNECT_DATA = (SERVER = DEDICATED)(SID = cidb)  )  )
```

- Wallet anlegen und Verbindungsdaten speichern

```
>mkstore -wrl c:\tns -create
…
>mkstore -wrl c:\tns -createCredential crm_at_cidb crm crm
…
>sqlplus /@crm_at_cidb
```

DB-Change Tool

- Beispiel: eine neue Spalte durch ETL-Schiene ziehen
  - Übersichtlich
  - Skripte kann man aufgabenbezogen gestalten
  - Der Entwickler muss die echten Verbindungsdaten nicht wissen

```
-- View-Definition in der Quell-DB
connect /@crm_at_db_src
@../crm_at_db_src/views/kunde.vw

-- Tabelle im Stage erweitern
connect /@stage_at_db_dwh

ALTER TABLE stg_kunde ADD (cust_category VARCHAR2(100));

-- Tabelle im Core erweitern
connect /@core_at_db_dwh

ALTER TABLE co_kunde ADD (cust_category VARCHAR2(100));

-- Dimension im Mart erweitern
connect /@mart_at_db_dwh

ALTER TABLE dim_kunde ADD (cust_category VARCHAR2(100));
```

AGENDA

- Einführung: CI und CDBI
- Build-Prozess
  - DB-Change-Tool
  - Verbindungsaufbau
- Unit-Tests
- What's next?
- Referenzen

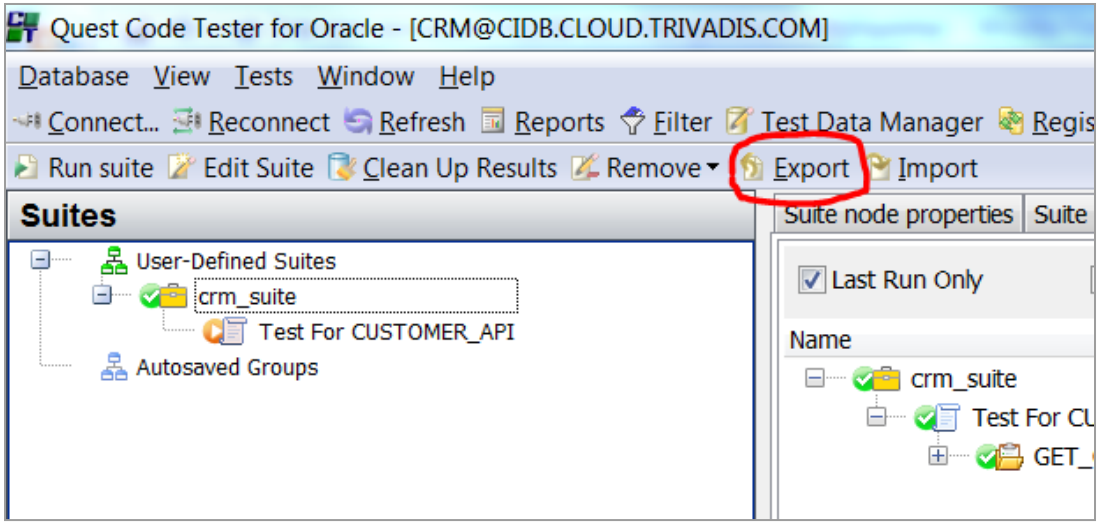
Unit-Tests

- Quest Code Tester for Oracle
  - Kostenpflichtiges aber komfortables Tool für das Testen von PL/SQL
  - Nimmt viel Arbeit durch das Generieren von Test-Code ab
  - Stellt PL/SQL API für Automatisierung zur Verfügung
  - Nachfolger von utplsql

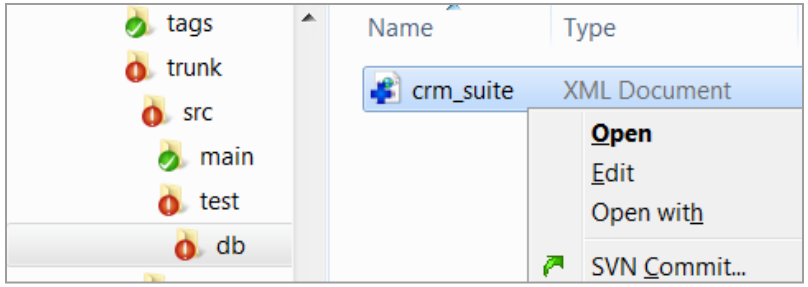
- Anbindung an CI:
  - Testdefinitionen über VCS verteilen
  - Tests ausführen
  - Testergebnisse bereitstellen

Unit-Tests

- Testdefinitionen über VCS verteilen
  - Export von Testdefinitionen als XML



  - XML-Datei einchecken



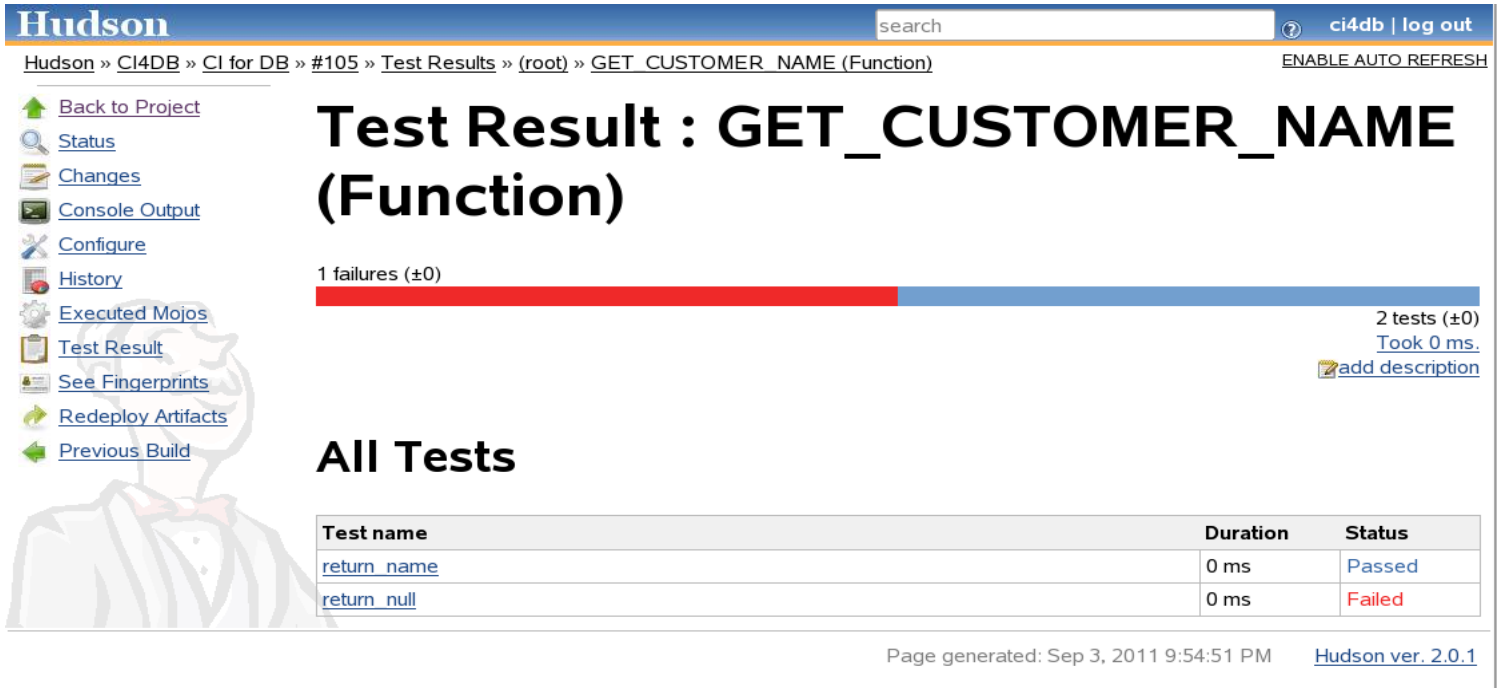
Unit-Tests

- Testdefinitionen über VCS verteilen
  - Import der XML-Definitionen mittels PL/SQL API über CI-Server
    - Kopieren der XML-Datei auf den DB-Server und Laden als BFILE:
      - Vorteil: Einfachere Implementierung
      - Nachteil: Zusätzliche Konfiguration (Zugang zum DB-Server), Directory und Rechte in der DB
    - Importieren der XML-Datei als CLOB direkt vom Client (CI-Server) aus:
      - Vorteil: Keine zusätzliche Konfiguration
      - Nachteil: Implementierung ist nicht mehr trivial
    - Aufruf von PL/SQL API (Package QU_XMLDOM_IMPORT)

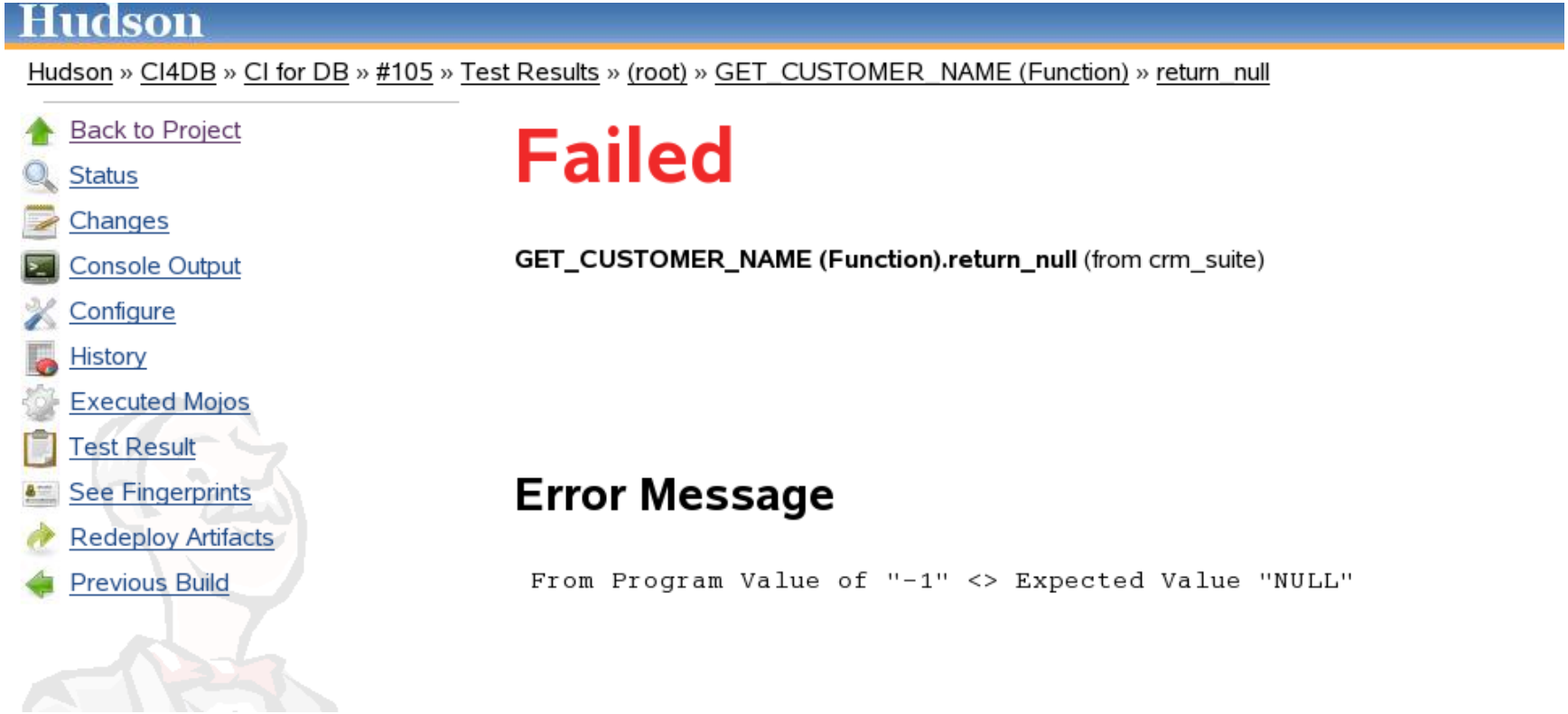
```
begin
 -- Load the export, preparing for import.
  qu_xmldom_import.init_xml(:xml_import);
  qu_xmldom_import.set_options( … );
  qu_xmldom_import.set_for_mapping(prog_owner_in => 'CRM'
                                   ,harn_owner_in => 'CRM' );
  -- Perform the import.
  qu_xmldom_import.import_as_xml;
end;
/
```

Unit-Tests

- Ausführen von Testdefinitionen
  - PL/SQL API von Code Tester, Package QU_TEST
    - Bietet verschiedene Möglichkeiten Test-Suites und einzelne Test-Cases auszuführen
  - Über PL/SQL lassen wir die Ergebnisse im JUnit-XML-Format ausgeben (Surefire-Report), sodass sie von Hudson interpretiert werden:



Unit-Tests



AGENDA

- Einführung: CI und CDBI
- Build-Prozess
  - DB-Change-Tool
  - Verbindungsaufbau
- Unit-Tests
- What's next?
- Referenzen

What's next?

- Trivadis SQL & PL/SQL Coding Guidelines, http://www.trivadis.com/de/technologie/oracle/oracle-application-development/oracle-sql-und-plsql.html
  Integration von Trivadis Code-Checker
- Code Qualitätsmetriken: Sonar Plugin für PL/SQL
- PL/SQL Inline-Dokumentation, z.B. PLDoc
- APEX
- Data Warehouse (OWB, ODI)

AGENDA

- Einführung: CI und CDBI
- Build-Prozess
  - DB-Change-Tool
  - Verbindungsaufbau
- Unit-Tests
- What's next?
- Referenzen

Referenzen

- Scott Ambler, Pramod Sadalage: Refactoring Databases – evolutionary database design, Addison Wesley, 2006
- Paul M. Duvall: Continuous Integration , Addison Wesley, 2007
- Gunter Popp: Konfigurationsmanagement mit Subversion, Maven und redmine, 3. Aufrlage, dpunkt.verlag, 2009
- QCTO Community Blog
  http://toadworld.com/BLOGS/tabid/67/EntryId/493/Continuous-Integration.aspx

VIELEN DANK.

Trivadis GmbH

Andrej Pashchenko

Werdener Str. 4
40227 Düsseldorf

Tel. +49 211 58 66 64 70
Fax + 49 211 58 66 64 71

info@trivadis.com
www.trivadis.com

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