Automatisiertes Testen für APEX

Dr. Gudrun Pabst

Trivadis GmbH
Lehrer-Wirth-Straße 4
81829 München

gudrun.pabst@trivadis.com

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

trivadis
makes IT easier.

AGENDA

- **Einführung**
- Werkzeuge
- Vorgehen
- Zusammenfassung

Einführung

- Arten von Tests
  - Unit Tests (Modultests / Komponententests)
  - Integrationstests
  - Abnahmetests
  - Lasttests
  - Performancetests
  - …
- Hier:
  - Architektur-Prüfungen
  - Unit Tests
  - Integrationstests

Einführung

- Ziel der Tests
  - Qualitätssicherung
  - Prüfen der Umsetzung von Funktionalitäten
  - Sicherstellen der Funktionalitäten auch nach Änderungen
- Manuelles Testen
  - langwierig, langweilig
  - fehleranfällig
  - benötigt Vorarbeiten (Checklisten …), um wiederholbar zu sein
- Automatisiertes Testen
  - Übernahme der Routineaufgaben durch den Computer
  - wiederholbar
  - schnell durchführbar

Einführung

- APEX-Anwendungen bestehen aus
  - Datenmodell
  - APEX-Oberfläche für Benutzereingaben
  - Geschäftslogik zur Verarbeitung der Eingaben in Datenbankpackages
- Prüfungen
  - vor der Programmierung: Validierung des Datenmodells
  - Test der programmierten Anwendung(steile):
    - Test der Geschäftslogik
    - Test der korrekten Funktion der Oberfläche

AGENDA

- Einführung
- **Werkzeuge**
- Vorgehen
- Zusammenfassung

Validierung des Datenmodells

- Prüfungen:
  - Existenz von Primary Keys, Unique Keys, Foreign Keys
  - Einhaltung von Namenskonventionen
  - Existenz von Kommentaren
  - …
- Im Modellierungswerkzeug:
  - z.B. SQL Developer Data Modeler: „Design Rules“
- In der Datenbank:
  - Skripte → Ausführung in einer SQL-Oberfläche

Werkzeuge für DB-seitige Tests

Wikipedia:
(http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks#PL.2FSQL)

- SQL Developer (Oracle)
- utPLSQL
- Code Tester for Oracle (Quest)
- pl/unit
- PL/SQL Unit Testing for Oracle (PLUTO)
- ruby-plsql-spec
- DBFit

Werkzeuge für Tests von Web-Oberflächen

Wikipedia: (http://en.wikipedia.org/wiki/List_of_web_testing_tools)

- iMacros
- QF-Test
- Ranorex Studio
- Sahi
- Selenium
- SOAtest
- TestComplete
- TOSCA Testsuite
- Test Studio
- WatiN
- Watir

Verwendete Werkzeuge für die Tests

Verwendete Werkzeuge:

- SQL Developer
  - kein zusätzliches Tool nötig
  - keine Kosten
- Selenium
  - verschiedene Varianten:
    - Plugin für Firefox (Selenium IDE) zum Aufzeichnen von Testfällen
    - API für verschiedene Programmiersprachen zur Programmierung von Tests

AGENDA

- Einführung
- Werkzeuge
- **Vorgehen**
- Zusammenfassung

Erstellen des Testkonzepts

- Welche Funktionalität soll getestet werden?
- Welchen Umfang soll der Test haben?
- Positiv-Fall:
  - Welche Eingaben führen zu einem korrekten Ergebnis?
  - Wie sieht das korrekte Ergebnis aus?
- Negativ-Fall:
  - Welche Eingaben führen zu einem Fehlerfall?
  - Wie sieht der Fehlerfall aus?
- Welchen Datenbestand muss das System aufweisen?

Erstellen des Testkonzepts – Probleme

- Konzeptionelle Probleme:
  - Sind die Funktionalitäten / Testfälle korrekt gewählt?
  - Ist der gewählte Datenbestand passend?
  - Sind die Testfälle vollständig?
  - Ist die gewünschte Funktionalität durch den Test korrekt abgebildet?
- Technische Probleme:
  - Wie wird die korrekte Verarbeitung identifiziert?
  - Wodurch ist der Fehlerfall definiert?

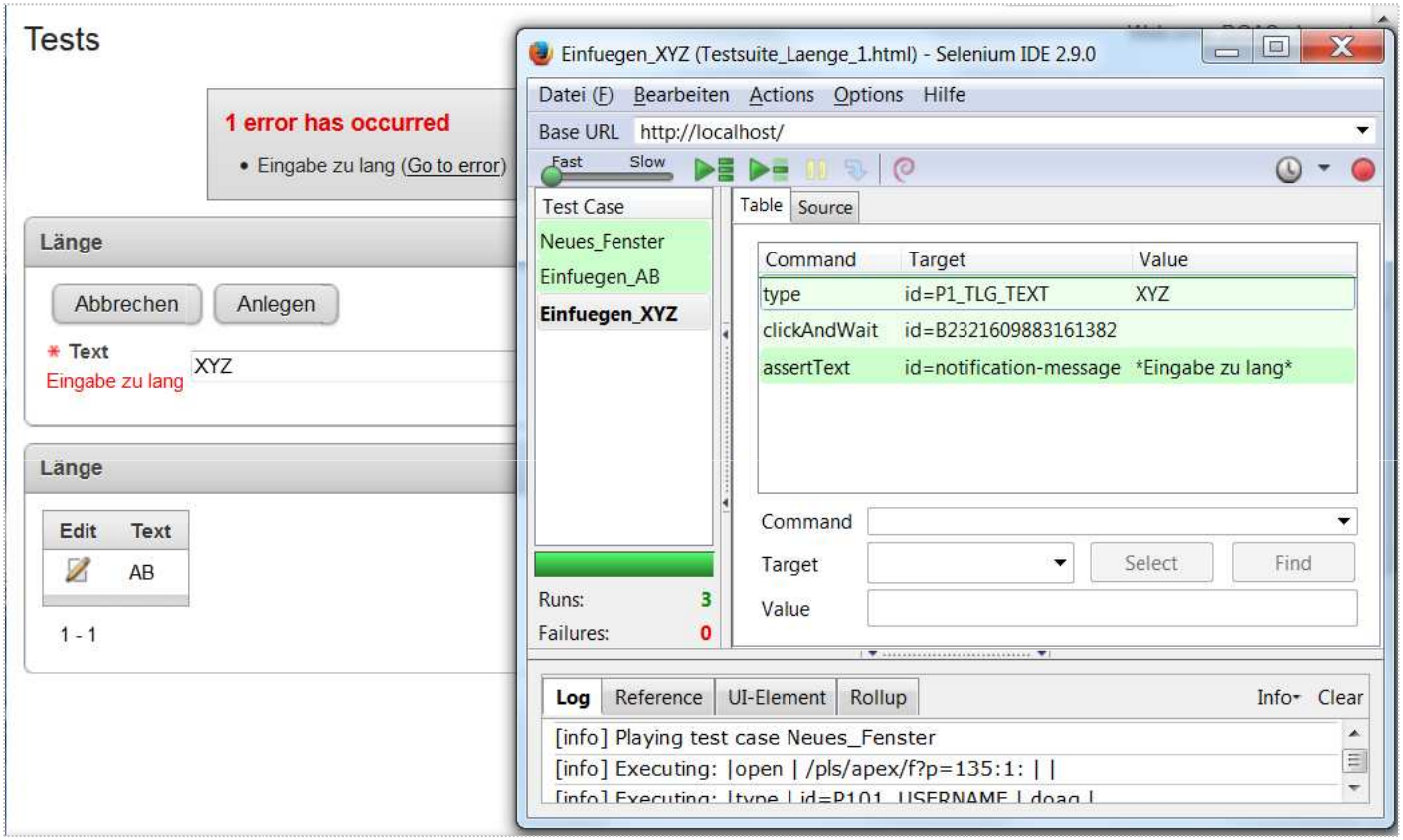
Erstellen des Testkonzepts

- Beispiel:
  - Zu testende Anforderung:
    Der Text in der Spalte TEXT darf maximal 2 Zeichen lang sein und beliebigen Text enthalten.
  - Problem: Die Programmierung ist fehlerhaft.

- Vorgeschlagener Test:
  - Datenbestand:
    - Die Tabelle ist leer.
  - Positiv-Fall:
    - Das Einfügen des Texts AB ist erfolgreich.
  - Negativ-Fall:
    - Das Einfügen des Texts XYZ führt zum Fehler.

Erstellen des Testkonzepts

Ergebnis des Tests:



- Test wurde erfolgreich beendet
- vorhandene Programmierfehler wurden nicht erkannt!

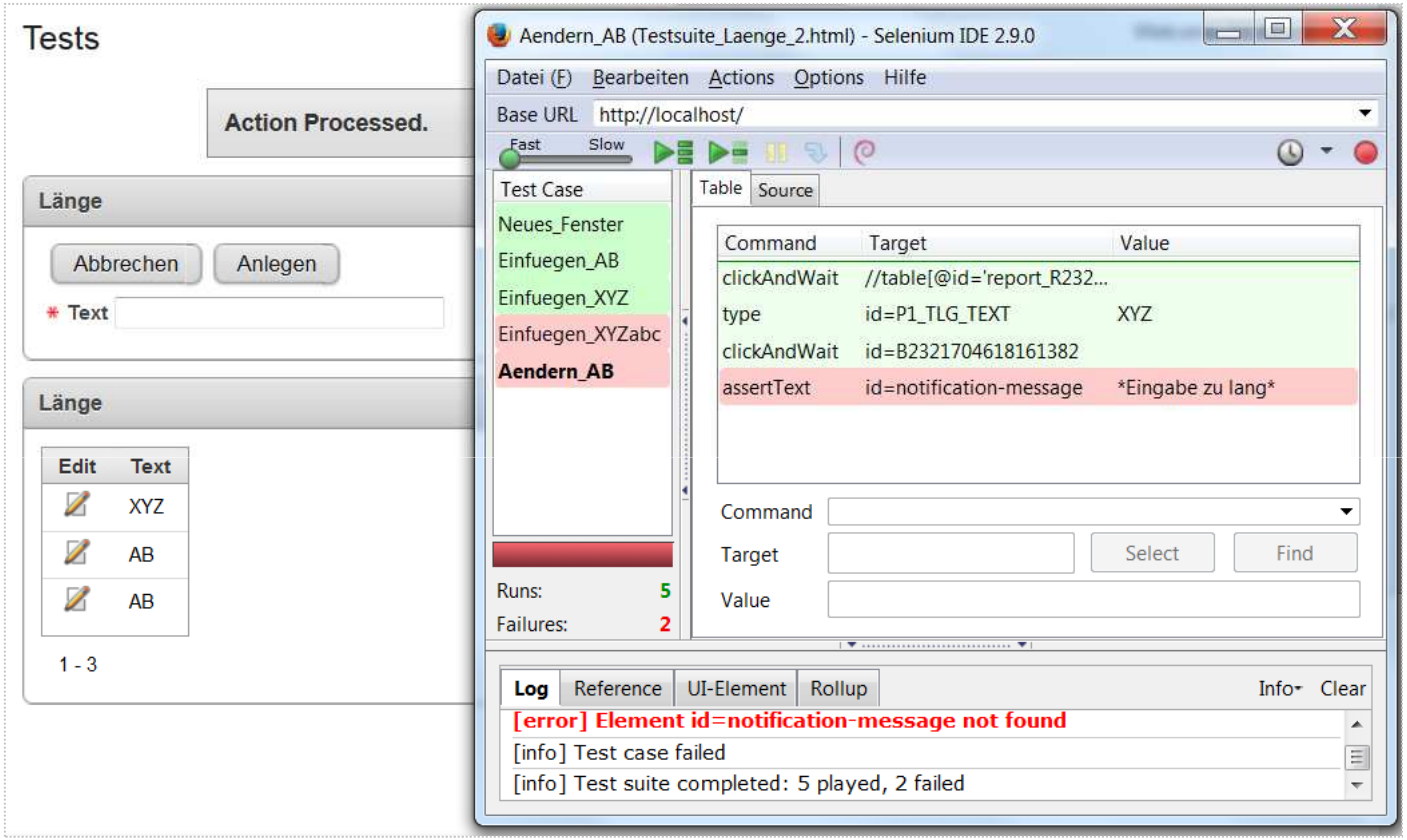
Erstellen des Testkonzepts

Probleme im Beispiel:

- Programmierfehler
  - Die Länge wird nur bei Inserts geprüft.
  - Längenprüfung: Statt „>=3“ wurde nur „=3“ programmiert.
- Umfang des Tests
  - nur Inserts, keine Updates
  - nur Texte aus zwei bzw. drei Zeichen
  - nur Test von ASCII-Zeichen → Annahme: unproblematisch
- Ergebnis des Tests
  - Der Test wird korrekt durchlaufen.
  - Die fehlende Prüfung bei Updates wird nicht bemerkt.
  - Die falsche Einschränkung wird nicht festgestellt.

Erstellen des Testkonzepts

Erweiterung des Tests:



- Test schlägt fehl
- vorhandene Programmierfehler wurden erkannt

Erstellen des Testkonzepts

- Welche Funktionalitäten sollen getestet werden?
  - Aus der Spezifikation
  - Abwägen von Aufwand und Nutzen
- Welchen Umfang soll der Test haben?
  - „alles“ → nicht möglich
  - Festlegen von **relevanten** Testdaten und –fällen, zum Beispiel:
    - Wenn es für „AB“ funktioniert, kann dann angenommen werden, dass es auch für „ßä“ und für „ל१ז“ funktioniert?
    - Sind Inserts, Updates und Deletes getrennt zu betrachten?
  - Festlegen von mehreren unterschiedlichen Testdatensätzen für den Positiv- und den Negativfall

Implementierung der Tests

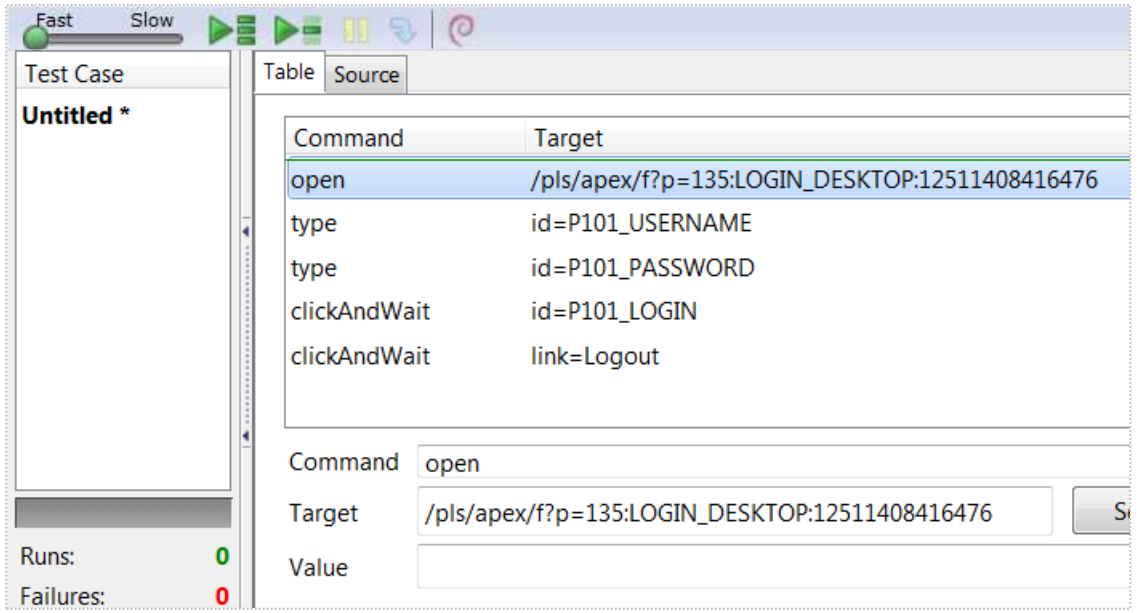
- Erstellen einer Initialisierungsroutine für die Datenbank:
  - Anlegen der Strukturen
  - Einfügen des benötigten Datenbestands
- Erzeugen der Tests
  - Programmieren der Testroutinen
  - Aufzeichnen von Eingaben mit Selenium
  - Anpassen der aufgezeichneten Eingaben:
    - Erweitern um Prüfungen
    - Anpassen von Session-spezifischen Stellen

Implementierung der Tests für ein Beispiel

- Beispiel:
  - Die Spalte LAENDER_KUERZEL in der Tabelle LAENDER darf nur eindeutige Werte aufweisen.
- Datenbestand:
  - Es gibt drei Datensätze in der Tabelle.
  - Die Datensätze haben die Länderkürzel DE , EN , FR .
- Positiv-Fall:
  - Das Einfügen eines Landes mit Länderkürzel AT ist erfolgreich.
  - Das Ändern des Länderkürzels für den Datensatz AT auf IT ist erfolgreich.
- Negativ-Fall:
  - Das Einfügen eines Landes mit Länderkürzel DE führt zum Fehler.
  - Das Ändern des Länderkürzels für den Datensatz EN auf DE führt zum Fehler.

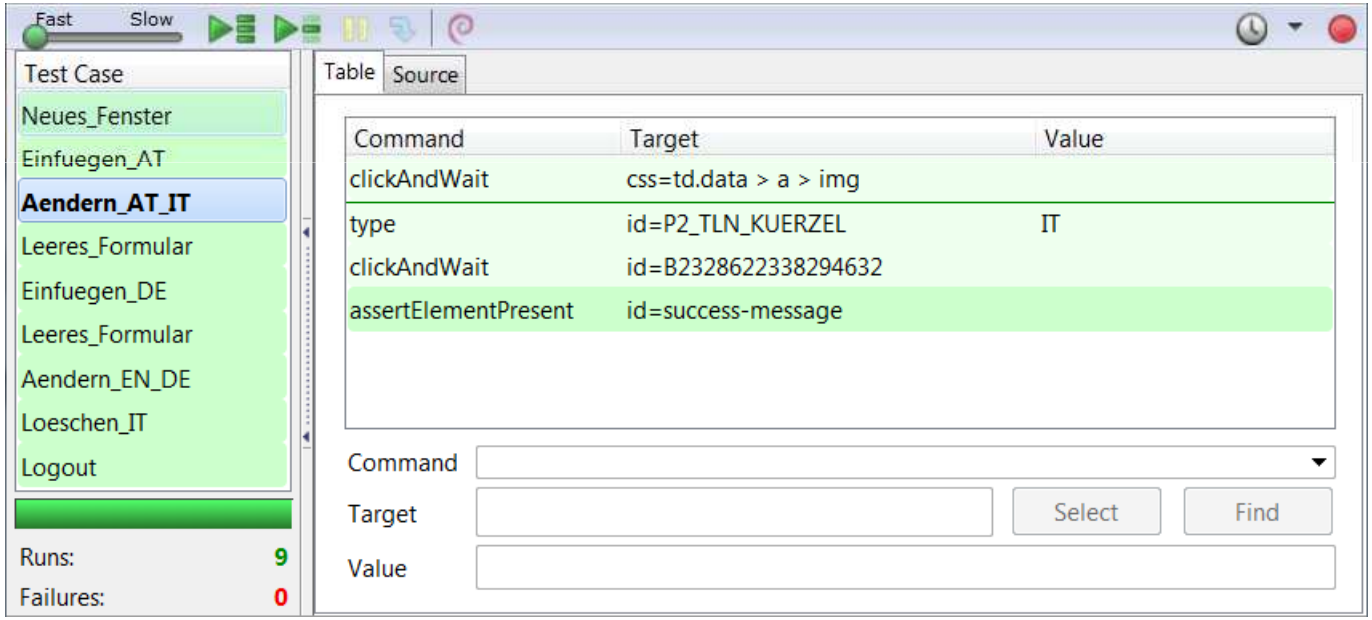
Implementierung der Tests für ein Beispiel

- Datenbestand
  - Der korrekte Datenbestand ist bereits hergestellt.
- Implementieren der Testfälle
  - Verwendetes Tool: Selenium
  - Aufzeichnen der Tests
  - Nachbearbeitung



Implementierung der Tests – Probleme

- Ist die Initialisierung der Datenbank korrekt?
- Entsprechen die programmierten Testfälle dem Testkonzept?
- „Wer testet die Tests?“



Umsetzen von Änderungen in der Anwendung

- Erstellen neuer Tests zum Prüfen neuer Anforderungen
- Bereits definierte Tests
  - sollen sicherstellen, dass eingebaute Funktionalität weiter funktioniert
  - aber geänderte Funktionalitäten erfordern geänderte Tests
- Ermitteln und Anpassen der Tests, die von den Änderungen betroffen sind:
  - Änderungen an den Strukturen und dem initialen Datenbestand
  - Änderung der Testdurchführung
  - ggf. Änderungen der Gültigkeitskriterien durch Textänderungen
- Wartung der Tests ist aufwendig!

AGENDA

- Einführung
- Werkzeuge
- Vorgehen
- **Zusammenfassung**

Zusammenfassung

- Erstellen sinnvoller automatisierter Tests aufwendig
- Anpassen der Tests bei Änderungen notwendig
- Nicht jede Funktionalität mit automatisierten Tests prüfen:
  - Wie wahrscheinlich ist das Auftreten eines Fehlers an dieser Stelle?
  - Wie hoch ist der Schaden / die Auswirkung, wenn der Fehler auftritt?
  - Wie hoch ist der Aufwand, einen automatisierten Test für die Funktionalität zu erstellen?

VIELEN DANK

Dr. Gudrun Pabst

Trivadis GmbH
Lehrer-Wirth-Straße 4
81829 München

gudrun.pabst@trivadis.com

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