APEX goes UNIT Testing

Oliver Lemm
MT AG
Ratingen

Schlüsselworte
APEX, UNIT Testing

Einleitung

Der Vortrag zeigt wie man effektiv und qualitätsunterstützend mit Hilfe von UNIT Tests APEX Anwendungen entwickeln kann. Dabei wird mit Hilfe des APEX Repositories und der Oracle Metadaten automatisiert geprüft, welche Vorgaben in APEX und im Datenmodell eingehalten werden.
Die Prüfungen zeigen ähnlich wie beim internen APEX Advisor Informationen pro APEX Seite bzw. pro Datenbankobjekt an. Im Vergleich zum Advisor, werden die Prüfungen aber speziell auf die Vorgaben des Projekts erstellt und erstrecken sich auch über das Datenmodell und ggf. sogar über vorhandene Daten. Der Advisor und das entwickelte Testpackage der MT AG ergänzen sich dabei problemlos. Zusätzlich werden die Informationen zur Laufzeit direkt bei der Entwicklung angezeigt.

Die Ausgangssituation

In jeder Webanwendung ist ein Ziel dem Benutzer einheitliches Layout und eine übersichtliche Anwendung zur Verfügung zu stellen. Eine verständliche und gut zu bedienende Anwendung wird einerseits durch ein gutes Fach- und Umsetzungskonzept definiert, aber muss auch innerhalb der Entwicklung eingehalten werden.
Auch die Art der Verarbeitung von Daten, sowie Validierungen und die Benutzerführung müssen innerhalb der Anwendung gleich ablaufen. Andernfalls wird sich ein Benutzer wundern, wenn beispielsweise auf einer Seite Daten dynamisch bei Eingabe nachgeladen werden und auf der nächsten Seite erst immer ein Button betätigt werden muss, damit eine Aktion stattfindet.

Das Konzept

Innerhalb des Fachkonzepts werden die grundlegenden Entscheidungen getroffen, welche Funktionalitäten und welches Layout die Anwendung später besitzt.
Im Umsetzungskonzept werden dann die Techniken und Mechaniken beschrieben, wie die Vorgaben des Fachkonzepts umgesetzt werden. Daraus resultieren die Vorgaben, wie die einzelnen APEX Komponenten über die Templates angepasst werden müssen, sowie Eigenschaften über Ausrichtung, Labels, Conditions und viele weitere Eigenschaften innerhalb APEX.
Ein großer Teil dieser Eigenschaften ist innerhalb des APEX Repositories und somit über Views, welche mit „APEX_APPLICATION_“ beginnen ermittelbar.
Neben den Vorgaben innerhalb der Anwendung, kann man zusätzlich noch Prüfungen innerhalb der Datenbank durchführen.
Zuletzt können, je nach Projekt, zusätzlich Prüfungen auf Basis von Tabelleninhalten erfolgen die den Funktionalen Ablauf, sowie Abhängigkeiten zu APEX Objekten prüfen.

Bei all diesen Prüfungen sollte der Ansatz des UNIT Testing benutzt werden, der bei jeder Prüfung einzeln zurückgibt, ob diese positiv oder negativ getestet wurde. Neben der Information, dass eine Prüfung fehlgeschlagen ist, werden Informationen zum jeweiligen Fehler gesammelt und nachher ausgegeben.
Damit diese Prüfungen auch garantiert zur Laufzeit immer durchgeführt werden, ist der Aufruf dieser Prüfungen in APEX über eine Region auf Seite 0 eingebaut. Das Ergebnis der Prüfungen wird dabei in einer Region unterhalb der anderen Regionen angezeigt.

Zusätzlich wird die Region nur dann angezeigt, wenn man sich im Entwicklungsmodus befindet. Dies ist über eine Condition (PL/SQL Expression) in der Region realisiert:

```
APEX_APPLICATION.G_EDIT_COOKIE_SESSION_ID IS NOT NULLL
```

Bei diesem Ansatz wurden alle Funktionen so entworfen, dass die Prüfungen über das Repository mit dem Parameter der APEX Seite (PAGE_ID) durchgeführt werden können. Dies sorgt dafür, dass der jeweilige Entwickler auch nur die Fehler auf der Seite angezeigt bekommt, die direkt in Relation zur momentanen Seite stehen.
Zusätzlich besteht die Möglichkeit alle Seiten der Anwendung zu prüfen, indem man die einzelnen Seiten durchläuft und die Komponenten prüfen, welche nicht direkt einer Seite zugeordnet werden können.

Bei allen Prüfungen wurde eine Einstufung in eine Fehlerkategorie getroffen, wie kritisch eine Vorgabe ist. Dies kann beim Aufruf der Prüfungen verwendet werden, um zum Beispiel nur die Fehler anzuzeigen, die eine Ausführung der Anwendung verhindern.

Die Umsetzung
Bei der Umsetzung wurden die 3 unterschiedlichen Arten der Prüfungen

1. Prüfungen auf dem APEX Repository & APEX Views
2. Prüfungen auf den Datenbankobjekten
3. Prüfungen auf den Tabelleninhalten

jeweils in einzelne Packages ausgelagert.

Prüfungen auf dem APEX Repository & APEX Views

Hierbei wurden zuerst die Eigenschaften identifiziert die geprüft werden sollten, als auch die Einschränkungen definiert, welche möglich sein sollten um gezielt einzelne Komponenten testen zu können.

Die Funktion check_application prüft dabei die gesamte Applikation und muss mindestens mit der jeweiligen APP_ID aufgerufen werden.
Zusätzlich besteht die Möglichkeit einzelne Seiten auszuschließen oder eine Liste mit Seitennummer anzugeben, welche geprüft werden müssen.
Als weiterer optionaler Parameter kann die Einstufung eines Fehlers beachtet werden, dass zum Beispiel nur kritische Fehler angezeigt werden. Per Standard werden aber alle Prüfungen durchgeführt.
Daneben besteht die Möglichkeit zu definieren, welche APEX Komponenten ggf. nicht geprüft werden. Auch an dieser Stelle werden per default alle Komponenten auf die Vorgaben geprüft.

Die Prüfung der einzelnen Komponenten geschieht dabei über den Aufruf der Funktion check_page.

Die Funktion check_page benötigt neben der APP_ID zwingend eine APP_PAGE_ID. Durch die APEX Variablen :APP_ID und :APP_PAGE_ID ist der Aufruf innerhalb der Seite 0, jeweils mit der Seitennummer auf, welcher sich der Entwickler befindet möglich:

```
declare
 l_result boolean;
 l_errors varchar2(32767);
begin
 l_result := qs_apex_pkg.check_page(p_application_id => :APP_ID,
```

```
                                        p_page_id       => :APP_PAGE_ID,
                                        p_messages      => l_errors);
 if l_result
 then
  htp.prn('Seite enthält keine Fehler bei der automatischen
Prüfung');
 else
  htp.prn('<pre>' || l_errors || '</pre>');
 end if;
end;
```

Egal welche Seite der Entwickler nun bearbeitet oder neu erstellt, er bekommt direkt das Ergebnis der Prüfung angezeigt bzgl. der Vorgaben die ggf. nicht eingehalten worden sind. Fügt man zum Beispiel eine Region hinzu und benutzt ein Template was nicht referenziert werden darf, so wird dies sofort auf der Seite dargestellt.

Zusätzlich kann man die Funktionen zur Prüfung einer Seite, als auch zur Prüfung der gesamten Anwendung (inklusive der Komponenten die nicht seitenspezifisch sind) auch ohne eine APEX Sessions direkt in der Datenbank über sqlplus aufrufen.

Prüfungen der einzelnen APEX Seitenkomponenten
Innerhalb des MT Test-Packages existieren nun folgende weitere Routinen:

- check_branches
- check_processes
- check_items
- check_buttons
- check_regions

Diese Funktionen basieren im Grunde auf folgenden Views:

- apex_application_page_branches => Weiterleitungen/Umleitungen
- apex_application_page_proc => Prozesse
- apex_application_page_items => Seitenelemente
- apex_application_page_buttons => Buttons
- apex_application_page_regions => Regionen

und noch die View für die Seite selber

- apex_application_pages => Seite

Alle Beispiele innerhalb der Prüfungen können je nach Projekt variieren. Es empfiehlt sich eine größt mögliche Menge an Prüfungen, basierend auf den Vorgaben zu definieren.

Folgende Eigenschaften werden dabei pro Objekt geprüft. Die Prüfung ist dabei so angelegt, dass alle Elemente ausgegeben werden, die nicht die Vorgabe erfüllen:

1. Weiterleitungen (Branches)
   I. branch_action <> '&REQUEST.'
      Es empfiehlt sich in APEX den Request auch beim Aufbau der Seite auswerten, daher wird eine Prüfung vorgenommen

2. Prozesse (Processes)

I. PROCESS_TYPE = ‚Automated Row Fetch‘ AND PROCESS_POINT <> ‘On Load – Before Header’
   Ein Prozess zum Laden der Daten sollte immer beim gleichen Verarbeitungspunkt durchgeführt warden (beim Aufbau der Seite, vor dem HTML Gerüst)
II. PROCESS_NAME = ‚SAVE_PAGE‘ AND SUCCESS_MESSAGE <> ‚&AI_APP_USER_MESSAGE.‘
   Jeder Verarbeitungsprozess muss eine Positivmeldung besitzen.
III. RUN_PROCESS <> ‘Once Per Page Visit (default)’
   Standardmäßig sollte jeder Prozess pro Aufruf der Seite durchgeführt werden.
IV. ERROR_DISPLAY_LOCATION <> ‘INLINE_IN_NOTIFICATION‘
   Jeder Fehler sollte ab APEX 4.x den Fehler innerhalb der Seite darstellen und keine Umleitung mehr durchführen.

3. Eingabefelder (Items)
   I. Substr(ITEM_LABEL_TEMPLATE,0,) <> ‚XXX‘
      Hierbei gilt die Vorgabe, dass das Kürzel ‚XXX‘ (nur als Beispiel) im Namen des Templates vorhanden sein muss. Dadurch kann man sicherstellen, dass nur Templates verwendet werden, die für diese Anwendung angepasst wurden.
   II. LABEL_ALIGNMENT not in (‚Left‘) AND DISPLAY_AS not in ('Hidden', 'Checkbox', 'Radio Group', 'Textarea', 'Shuttle')
      Hierbei wird die Ausrichtung des Labels geprüft, die fülle alle Standardeingabefelder links definiert ist.
   III. DISPLAY_AS = ‚Checkbox‘ AND LOV_DISPLAY_EXTRA = ‚YES‘
      Eine Checkbox sollte keine anderen Werte zur Auswahl stellen, als die definierten Werte der Werteliste

4. Buttons
   I. BUTTON_TEMPLATE = ‚XXX Seite zurücksetzen‘ AND LABEL <> ‚Zurücksetzen‘
      Ein Button der das Template für Seite zurücksetzen enthält muss auch die Bezeichnung/Label tragen
   II. BUTTON_TEMPLATE = ‚XXX Seite zurücksetzen‘ AND BUTTON_SEQUENCE <> 10
      Der Button Zurücksetzen sollte immer als erster Button definiert sein.
   III. BUTTON_TEMPLATE = ‚XXX Seite zurücksetzen‘ AND EXECUTE_VALIDATIONS <> ‚No‘
      Der Button zum Zurücksetzen der Seite darf keine Validierungen auslösen.
   IV. BUTTON_TEMPLATE = ‚XXX Seite zurücksetzen‘ AND REGION_TEMPLATE <> ‚XXX Buttons‘
      Der Zurücksetzen Button muss immer in der Region mit dem entsprechenden Button-Template liegen.
   V. BUTTON_TEMPLATE = , XXX Seite zurücksetzen‘ AND DISPLAY_POSITION <> ‚TOP‘
      Der Zurücksetzen Button muss immer gleich ausgerichtet sein.
5. Regionen (Regions)
   I. TEMPLATE = ‚XXX Buttons‘ AND DISPLAY_POSITION <> 'Page Template Region Position 3'
      Die Button Region muss immer in der gleichen Region liegen.

6. Seite (Page)

I. PAGE_GROUP is null
Jede Seite der Anwendung muss einer Seitengruppe zugeordnet sein.

Diese Prüfungsbeispiele zeigen, welche Möglichkeiten existieren Vorgaben innerhalb APEX einfach und effektiv zu prüfen.

Prüfungen auf den Datenbankobjekten
Damit eine Prüfung im Rahmen einer Seite diese Eigenschaften überprüfen kann empfiehlt es sich hierfür 2 Ansätze zu verfolgen.
Einerseits wurden Objekte wie Views und Packages erstellt, welche pro APEX Seite die Datenbankobjekte kapseln, welche auf dieser Seite verwendet werden. Bei dieser Kapselung werden die Objekte mit der Seitennummer im Namen erstellt. Eine View für Seite 10 würde dann zum Beispiel XXX_P0010_BENUTZER_V heißen (bei Packages äquivalent). Diese Objekte kann man nun auf Invalidität prüfen. Zusätzlich kann geprüft werden ob eine Seite 10 in APEX existiert, oder ob eine solche View auf Seite 10 auch benutzt wird.

Der Zweite Ansatz ist, dass zusätzlich in einer Tabelle für Metainformationen zum jeweiligen Projekt, die Objekte (Tabelle & Spalte) festgehalten werden, welche auf der Seite verwendet werden. Auf diesen Inhalten kann dann geprüft werden, ob diese Objekte entsprechend vorhanden sind und ob Vorgaben bzgl. der Namenskonventionen eingehalten wurden.
Zusätzlich kann geprüft werden ob jeder Primär- und Fremdschlüssel einen Index besitzt und viele weitere Eigenschaften.

Prüfungen auf den Tabelleninhalte
Innerhalb der Daten von APEX Anwendung gelten immer wieder diverse Vorgaben und Metainformationen die vorhanden sein müssen. Es können beispielsweise Referenztabellen auf Inhalte oder themenspezifische Daten auf Abhängigkeit geprüft werden.
Wenn auch hier die Möglichkeit besteht einen Bezug zur jeweiligen APEX Seite herzustellen, kann es oft enorm hilfreich sein, diese Informationen auszugeben um den Entwickler bei der Entwicklung evtl. vorhandene fehlerhafte bzw. nicht vorhandene Daten aufzuzeigen.
Werden zum Beispiel in Wertelisten keine Daten angezeigt, oder Regeln zur Prüfung von Eingaben fehlen, so sind diese Fehler direkt bei der Entwicklung auf der Seite ersichtlich und die Ausgaben können enorm viel Zeit in der Problemanalyse und Entwicklung sparen.

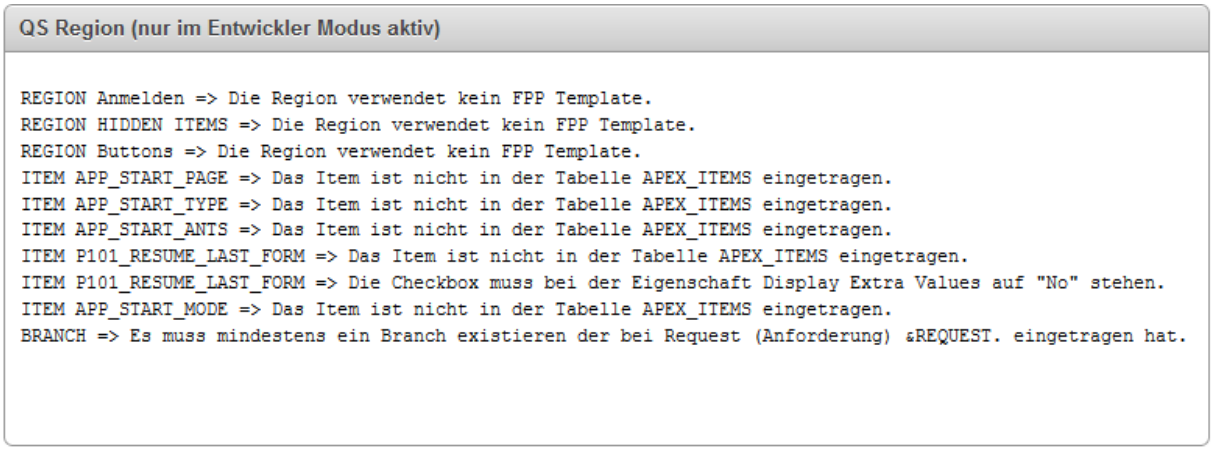

Abbildung 1: Beispielausgabe von Fehlermeldungen

Der Ausblick
In der ersten Version galt es primär zu identifizieren wie ein solcher Ansatz bei der Entwicklung und QS die Qualität der Anwendung steigert.
Um eine solche Funktionalität einfach in jedes neue Projekt zu integrieren, würde es sich anbieten die Prüfungen innerhalb von einzelnen SQL-Anfragen in einer Tabelle mit der jeweiligen Fehlermeldung auszulagern und umso die Prüfungen einfach und variabel anzupassen.
Außerdem wäre eine Kapselung in ein Region-Plugin denkbar um die Integration in andere Projekte zur erleichtern.
Zuletzt besteht die Möglichkeit mit Hilfe von Datenbankjobs und Mailversandt oder anderen Systemen einen Prozess zu schaffen, der einen jeden Entwickler, welcher für die jeweilige Seite verantwortlich ist eine Auflistung über die Seiten zu geben, welche er entwickelt. So könnte zu jedem Zeitpunkt dargestellt werden, welche Seiten, gemäß den Vorgaben korrekt funktionsfähig sind.

Fazit
APEX eignet sich perfekt zur automatisierten Analyse per UNIT Tests und zur Durchführung von Modultests. Außerdem werden mit Hilfe eines solchen Prozesses Vorgaben besser eingehalten und Fehler direkt zur Entwicklungszeit angezeigt und können dann sofort behoben werden.
Sowohl für den Entwickler, als auch die QS und nicht zuletzt für den Anwender wird die Qualität der Anwendung enorm gesteigert. Kommen neue Entwickler zum späteren Zeitpunkt zur Entwicklung der Anwendung hinzu, so werden evtl. nicht bekannte Vorgaben die ggf. nicht beachtet werden sofort angezeigt.
Zuletzt ermöglicht die gesamte Prüfung der Anwendung, selbst einer Person die technisch nicht mit dem Projekt vertraut ist, einen Überblick zu gewinnen an welchen Seiten noch Fehler vorhanden sind oder ob eine Anwendung gemäß den Vorgaben entwickelt wurde.

Kontaktadresse:
Oliver Lemm
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon:	+49 (0) 2102 309 61 - 0
Fax:	+49 (0) 2102 309 61 - 10
E-Mail	oliver.lemm@mt-ag.com
Internet:	www. mt-ag.com
Twitter:	https://twitter.com/OliverLemm