APEX-Anwendungen testen wie von Geisterhand mit PhantomJS und Casper

Andreas Wismann
WHEN OTHERS
D-45264 Kaarst

Schlüsselworte

APEX, Application Testing, JavaScript, PhantomJS, CasperJS

Einleitung

Web-Anwendungen, die mit APEX erstellt werden, bestehen nicht selten aus einer großen Anzahl von Einzelseiten im Rahmen einer umfangreichen Gesamtapplikation. Formulare und Reports summieren sich auf mehrere dutzend bis hunderte Application Pages. Hierdurch entsteht die Herausforderung, die Funktionalität, Korrektheit und Performance permanent unter Kontrolle zu behalten, um nicht zu riskieren, dass der Endanwender zum Betatester oder gar zum Feuermelder wird.

Wo Probleme lauern

APEX besteht als Webapplikation aus einer Vielzahl von Komponenten, die zur Laufzeit aus der Datenbank gelesen und ausgeführt werden. Zwar wurden diese Komponenten (Reports, Formulare, Wertelisten etc.) zur Programmierzeit validiert, anschließend werden sie jedoch nur unmittelbar zur jeweiligen Laufzeit geprüft. Dies unterscheidet APEX beispielsweise von Oracle Forms, wo die gesamte Anwendung zu jedem Zeitpunkt einen validen Status haben muss, um überhaupt gestartet werden zu können – ähnlich einer Datenbank-View. Dies hat Vor- und Nachteile. Der Vorteil bei APEX besteht insofern, dass die Anwendung auch dann noch verwendet werden kann, wenn nur Teile (etwa weniger wichtige Einzelseiten) keinen gültigen Zustand haben. Darum ist es aber umso wichtiger, eventuell nicht funktionierende Teile zu kennen, damit eine Bewertung des Benutzungsrisikos möglich ist.

Scheinbar zufällig …

Selbst wenn die Anwendung selbst nicht aktiv verändert wurde, können sich doch Abweichungen vom gewünschten Verhalten ergeben. Interessanterweise entstehen solche Veränderungen nicht immer im Zusammenhang mit einem bestimmten, planbaren Ereignis (etwa dem Deployment einer neuen Anwendungsversion), sondern „passieren zwischendurch“. Typische Fälle, in denen sich die Anwendung anders als erwartet verhält, sind:

- Erhöhung der Ausführungszeit eines Reports
- Veränderte Ergebnismengen in Reports (keine, zu wenige, falsche oder duplizierte Zeilen)
- Nicht-Funktionieren einzelner Seiten aufgrund zeitweise invalider Ressourcen
- Nichtverfügbarkeit von Seiten oder Seitenbestandteilen wegen fehlender Benutzerrechte
- Deadlock beim schreibenden Zugriff auf eine Tabelle
- Nicht mehr funktionierende Dynamic Actions

Finde den Fehler!

Einerseits bringt APEX ein (immer noch zu selten genutztes) Bordmittel mit, um jederzeit die Integrität der wichtigsten Anwendungskomponenten sicherzustellen: den APEX Advisor, zu finden unter Application Builder > Utilities > Advisor. Dieses Werkzeug prüft quasi auf Knopfdruck, ob die SQL-Statements ausführbar sind, ob die PL/SQL-Blöcke problemlos kompiliert werden können, ob Hyperlinks (Branches) ins Leere führen würden, ob alle Bedingungen gültig formuliert sind, referenzierte Seiten existieren, Variablen korrekt deklariert sind und so weiter.

Dabei handelt es sich natürlich um formelle Tests des Quellcodes. Ob dieser Code tatsächlich bewirkt was der Benutzer erwartet, bleibt unbeantwortet. Zudem kann der Advisor nicht hellsehen: Denn ob es tatsächlich ein Bug ist, dass ein Report neuerdings mit „no data found“ antwortet, muss letztlich der IT-Mitarbeiter oder der fachliche Benutzer beantworten. Daher prüft der Advisor diese Fragestellung erst gar nicht.

Hier kommen nun die Test-Tools ins Spiel.

Das Konzept

Testsoftware dient dazu, den tatsächlichen Output eines Programms mit dem erwarteten Output zu vergleichen. Der Testentwickler (zum Teil auch der Benutzer) definiert Szenarien, die ausgeführt und deren Ausgaben zur Laufzeit mit der Vorgaben verglichen werden.

Entscheidend ist, dass die Anwendung also tatsächlich „im Live-Betrieb“ getestet wird, und nicht nur anhand der Einsatzfähigkeit ihrer einzelnen Komponenten wie beim APEX Advisor. So ähnlich, als würde man einen Süßigkeitenautomaten testen: Geld einwerfen, Auswahl treffen und prüfen, ob unten tatsächlich der Schokoriegel entnommen werden kann und das Wechselgeld korrekt zurückgegeben wird.

Genau das, im übertragenen Sinne für Webanwendungen, leisten PhantomJS und CasperJS.

Phantom JS und CasperJS, Kurprofil

PhantomJS ist ein „Headless“ Browser, was soviel bedeutet wie Browser „ohne Anwendungsoberfläche“. Es basiert – derzeit noch – auf der Webkit-Engine (die auch vom Safari-Browser implementiert wird) und bietet eine JavaScript-Programmierschnittstelle für Eingaben und Ausgaben. PhantomJS wurde nicht speziell für APEX entworfen, sondern kann für die Überprüfung beliebiger Webseiten-Inhalte herangezogen werden: Vom Testen einer Oracle ADF-Anwendung bis hin zur Überwachung des Preises einer Spiegelreflexkamera im Online-Shop.

Mit PhantomJS können sogar Screenshots automatisch erstellt werden. Die Software ist kostenfrei verfügbar; zur Installation unter Windows, MacOS oder Linux genügt es, sie von der Seite http://phantomjs.org herunterzuladen und auf den Rechner zu kopieren – es ist keine besondere Administratorberechtigung erforderlich.

CasperJS ergänzt PhantomJS als Testwerkzeug, wenngleich es auch separat verwendet werden kann. Hauptsächlich dient es der programmierbaren Navigation durch umfangreiche Webseitenbestände und bringt ebenfalls ein JavaScript-API mit. Eine typische Anwendung für CasperJS ist das Screen Scraping, also das gezielte Auslesen von Text aus Webseiten.

Auch CasperJS ist unter den liberalen Bedingungen der MIT License frei verfügbar.

Eine Art „Hallo Welt“ am Beispiel einer APEX-Seite

Die Anwendung phantomjs.exe (unter Windows) präsentiert sich als Fenster, welches einem DOS-Kommandofenster ähnelt. Es nennt sich REPL (Read-Eval-Print-Loop) und kann gespeicherte Skripte ausführen.

Als erstes Beispiel hier ein Skript, das eine Webseite aufruft und, sofern die Webseite existiert, davon einen Screenshot im PNG-Format erstellt:

```
var page = require('webpage').create();
page.open('http://apex.oracle.com', function(status) {
  console.log("Status: " + status);
  if(status === "success") {
    page.render('apex-screenshot.png');
  }
  phantom.exit();
});
```

Dieses Skript kann unter dem Namen „apex-screenshot.js“ gespeichert und in REPL-Fenster mit

```
phantomjs apex-screenshot.js
```

aufgerufen werden. Wobei auch die Möglichkeit besteht, weitere Parameter an der Kommandozeile zu übergeben, etwa die Adresse der aufzurufenden Webseite. Diese Parameter werden von PhantomJS aus einem Array namens system.args ausgelesen.

Was kann getestet werden?

Die Ausgabe einer Webseite kann daraufhin untersucht werden, ob sie bestimmte Bestandteile enthält. Dazu kommt im Wesentlichen dieselbe Architektur des DOM (Document Object Model) zum Tragen, die man für CSS-Regeln oder jQuery-Skripte verwendet.

Ein PhantomJS-Skript könnte beispielsweise nach folgendem Pseudo-Code vorgehen:

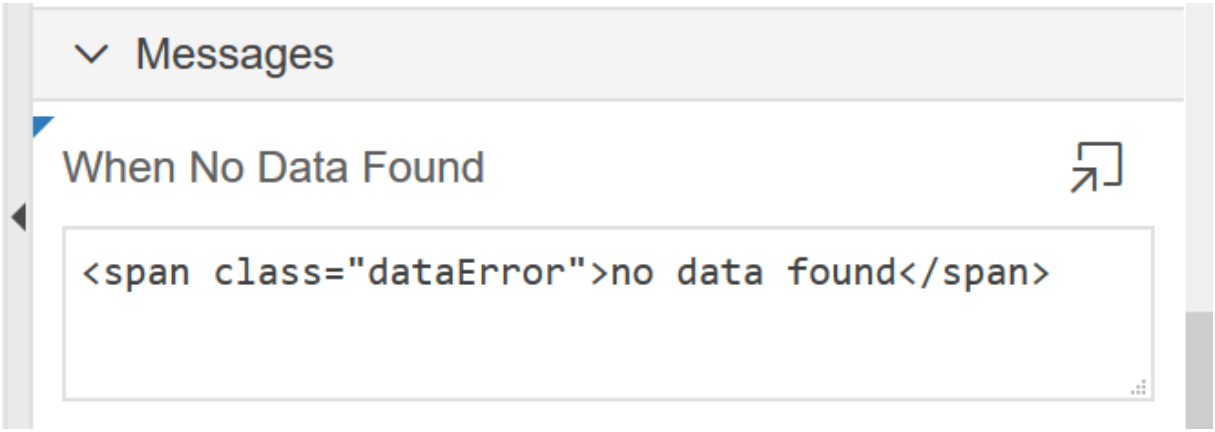
- Rufe die Website auf: server/apex/f?p=1234:567
- Protokolliere die Zeit bis zum vollständigen Aufbau der Seite
- Prüfe und protokolliere, ob der Report XY angezeigt wird
- Prüfe und protokolliere, ob der Report Datenzeilen enthält
- Prüfe, ob in der letzten Spalte „rechts unten“ ein Betrag größer 1000 erscheint; wenn nicht, erzeuge einen Eintrag in der PROBLEME-Datei

Besonderheiten beim Einsatz mit APEX

APEX verwendet unter anderem Cookies zur Identifizierung des Benutzers nach dem Login. Daher ist es entscheidend, dass auch CasperJS Cookies beherrscht und auch ermöglicht, diese skriptgesteuert anzulegen bzw. auszulesen.

Um die APEX-Anwendung möglichst optimal auf das Zusammenspiel mit CasperJS vorzubereiten, hilft es, schon bei der Programmierung bestimmte Marker zu verwenden:

- Jede Region einer APEX-Seite sollte ihre eindeutige HTML-ID erhalten
- In Templates lassen sich Markierung-Klassen als HTML-Attribute hinzufügen
- Die Report-Anzeige „no data found“ wird mit umgeben



Benutzerinteraktion
Selbstverständlich kann auch die Interaktion des Benutzers mit der APEX-Seite simuliert werden, zum Beispiel Button-Klicks oder die Bedienung des Anwendungs-Navigationsmenüs. So kann ein kompletter Bearbeitungsvorgang nachgestellt und dessen Ergebnisse getestet werden, etwa das Anlegen eines neuen Datensatzes in einem interaktiven Formular. Sollte der Server eine Fehlermeldung zurückgeben, so wäre PhantomJS in der Lage, diese anhand eindeutiger Merkmale (etwa dem gefundenen Textbestandteil ORA-) zu erkennen und zu protokollieren. Aber auch hier bietet APEX durch seine offenen Programmierschnittstellen vielfältige Möglichkeiten, die Fehlererkennung zu vereinfachen, etwa durch die Benutzung des package APEX_ERROR. Mehr dazu im Vortrag.

Ausblick auf den Vortrag
Die Präsentation zeigt anhand praxisnaher Live-Demos, wie man mit PhantomJS und CasperJS typische Testszenarien bewältigt und protokolliert. Das Hauptaugenmerk liegt darauf, bereits durch geschicktes Programmieren den Testvorgang erheblich zu vereinfachen.

Kontaktadresse:
Andreas Wismann
WHEN OTHERS Inh. Andreas Wismann
Hirschstr. 10
D-41564 Kaarst

Telefon: +49 (0) 2131 - 314 9966
mobil: +49 (0) 176 - 7800 3109
E-Mail: wismann@when-others.com

Internet: http://when-others.com