Donnerstag, 11. November 2004
11h00, Musensaal

Funktionales Testen von Oracle Web-Anwendungen mit Open-Source-Tools

Frank Berger
Shadow Connect GmbH, Krumbach

Schlüsselworte:

Funktionales Testen, Cactus, HTTPUnit, In-Container Testing, Canoo WebTest, Oracle ADF

Zusammenfassung

Der Artikel gibt zunächst einen Überblick über Open-Source-Tools für das Funktionale Testen von Web-Anwendungen. Danach wird eine Auswahl von Werkzeugen den einzelnen Anwendungsschichten einer mehrstufigen Web-Anwendung zugeordnet. In einem weiteren Schritt wird das Konzept des In-Container Testing mit Cactus kurz vorgestellt. Gefolgt von einem einfachen Beispiel für die Anwendung von HTTPUnit. Den Abschluß bildet ein Abschnitt über deklarative Beschreibung von Testfällen am Beispiel von Canoo WebTest.

Einleitung

Die Entwicklung von Software war noch nie so einfach wie heute. Für die Modellierung gibt es UML und die Implementierung erfolgt mit leistungsfähigen und umfangreichen Entwicklungsumgebungen. Auch für den Test der Software steht ein reichhaltiges Angebot an Werkzeugen zur Verfügung.

Trotzdem gibt es unzählige Software-Projekte, die scheitern und die Erwartungen der Kunden hinsichtlich der Qualität nicht erfüllen. Die Ursache für schlechte Software liegt also wahrscheinlich nicht an den verwendeten Tools oder Technologien, sondern an den Bedingungen, unter denen Software entwickelt wird. Hier wären

- verstärkter Kostendruck
- kürzere Produktlebenszyklen
- immer komplexere Software durch Forderung nach immer mehr Features und Implementierung von mehr Ausnahmefällen
- sich schnell verändernde Spezifikationen

als klassische Beispiele zu nennen.

In Abbildung 1 sind die 6 Bereiche der Softwarequalität nach [1] dargestellt. Die Punkte Funktionalität und Effizienz haben für viele Projekte eine herausragende Bedeutung. Um insbesondere die Forderung nach möglichst fehlerfreier Software unter den genannten Rahmenbedingungen umsetzen zu können, ist es erforderlich in allen Phasen der Software-Entwicklung eine durchgehende Teststrategie zu etablieren. Je nach Anforderung können hier Open-Source-Tools einen guten Beitrag leisten.

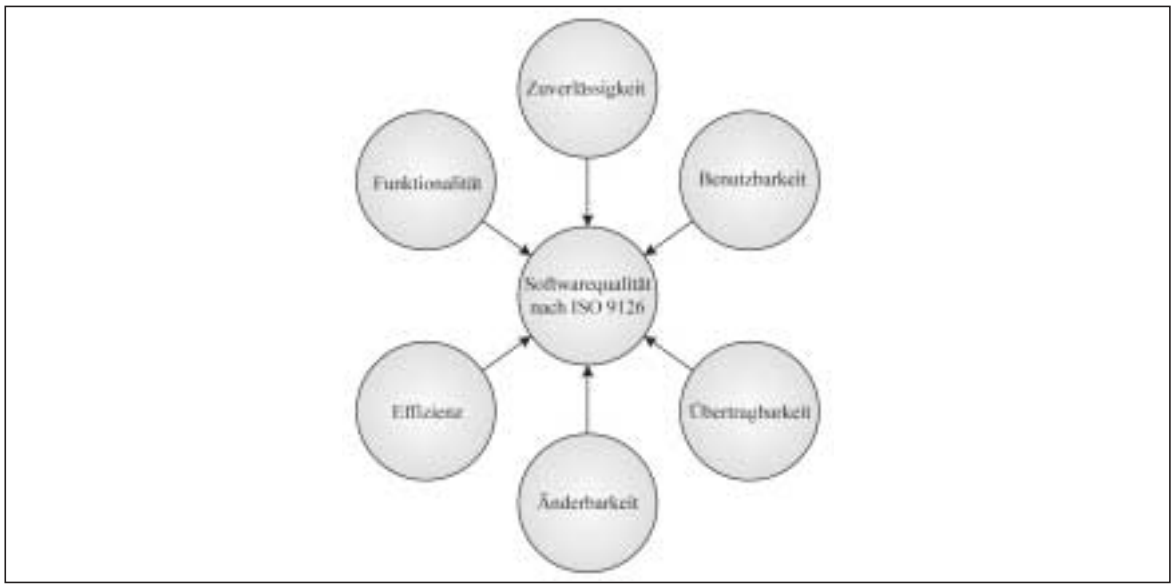

Abb. 1: Die 6 Bereiche der Softwarequalität

Die Firma Oracle verfügt über ein sehr reichhaltiges Produkt-Portfolio im Bereich der Web-Technologien. Beginnend bei Oracle Portal über das Oracle ADF hin zur nahtlosen Integration von Jakarta-Struts und nicht zuletzt in PL/SQL implementierte Web-Anwendungen. Daher soll im nachfolgenden Artikel der Schwerpunkt auf diese Web-Anwendungen gesetzt werden.

Überblick Open-Source Tools

Für den Bereich des Funktionalen Testens von Web-Anwendungen gibt es inzwischen eine unübersehbare Vielzahl von Open-Source-Tools. Die Werkzeuge bieten teilweise ganz unterschiedliche Funktionalität und lassen sich somit sehr gut kombinieren oder für unterschiedliche Zwecke und Situationen einsetzen.

Um hier einen ersten Überblick über die Bandbreite der Werkzeuge zu geben, erfolgt eine Kurzdarstellung einiger Werkzeuge und wie sie sich miteinander kombinieren lassen.

- **mechanize** [2] ist ein in Python geschriebener Web-Browser. Die Testfälle mit den einzelnen Browser-Aktionen werden hierbei ebenfalls in Python definiert. Ein ähnliches Konzept existiert in Perl mit den Paketen **HTTP::Recor-**

der [3] und **WWW::Mechanize** [4], sowie in Java mit **HTMLUnit** [5].

- **HTTPUnit** [6] arbeitet ähnlich wie HTMLUnit und ist auch in Java implementiert. Für HTTPUnit gibt es eine ganze Reihe von Erweiterungen. So steht mit **WebTest** [7] von der Firma Canoo eine sehr leistungsfähige Ergänzung bereit, mit der Testfälle in XML definiert werden können. Wegen seiner weiten Verbreitung wird HTTPUnit in einem eigenen Abschnitt noch näher vorgestellt.
- **Jameleon** [8] dient ebenfalls zum automatisierten Testen. Testfälle werden in XML und der Macro Sprache **Jelly** [9] (analog zu JSPs) definiert. Für Web-Anwendungen wird HTTPUnit mit eingebunden.
- **MaxQ** [10] realisiert den Ansatz eines Capture-Replay-Tools. HTTP-Requests werden mit dem eingebauten HTTP-Proxy aufgezeichnet und von MaxQ in Jython-Scripts umgewandelt. Es gibt auch Patches, um mit MaxQ XML-Testfälle für Canoo WebTest zu erstellen.
- **Anteater/AFT** [11] baut auf dem bekannten Build-Tool Ant auf. Testfälle werden auch hier in XML spezifiziert. Anteater eignet sich besonders für das Testen von SOAP-WebServices.
- **MockObjects** [12] bilden eine J2EE-Container Umgebung nach. Dadurch wird es möglich z.B. einzelne Servlets außerhalb von OC4J auszutesten. Dies erlaubt bei komplexen Aufgaben die Verkürzung des Code-Compile-Test-Zyklus.
- **Cactus** [13] und **JUnitEE** [14] führen das Konzept der Unit-Tests weiter zum sogenannten In-Container-Testing. Dabei wird der eigentliche Test-Treiber ebenfalls im Kontext des J2EE-Containers ausgeführt. Dieses Konzept eignet sich besonders für das Testen von Servlet-Filtern, Tag-Libraries und Session-Beans. Cactus wird im nachfolgenden Artikel ebenfalls noch näher vorgestellt.
- **dbUnit** [15] hat direkt zwar nichts mit Web-Anwendungen zu tun, leistet hierfür aber dennoch hilfreiche Dienste. Lassen sich damit doch vor dem eigentlichen Testlauf Datenbank-Tabellen mit Werten vorbelegen und nach dem Testlauf auf bestimmte Werte hin überprüfen. Durch die Integration mit Ant ist auch hier eine einfache Automatisierung realisierbar.
- **FIT** [16] dient als einfaches Werkzeug zur Definition von Akzeptanz-Tests in der Form von HTML-Tabellen. Mit **Fitnesse** [17] werden diese Definitionen in Form eines Wikis verwaltet. Mit **JwebUnit** [18] existiert auch eine Anbindung an HTTPUnit.

Welches Werkzeug ist das beste?

Diese Frage läßt sich in dieser Form leider nicht beantworten. So arbeiten die Tools teilweise mit sehr unterschiedlicher Test-Granularität. Wegen ihrer feinen Granularität werden MockObjects ähnlich wie JUnit vorrangig direkt von Entwicklern bei der Implementierung eingesetzt. Cactus und JUnitEE erweitern dieses Anwendungsfeld bezogen auf die Granularität nach oben hin zum Integrations-Test. HTTPUnit wiederum arbeitet mit sehr grober Test-Granularität (Black-Box-Testing) wie es zum Beispiel bei Akzeptanztests der Fall ist.

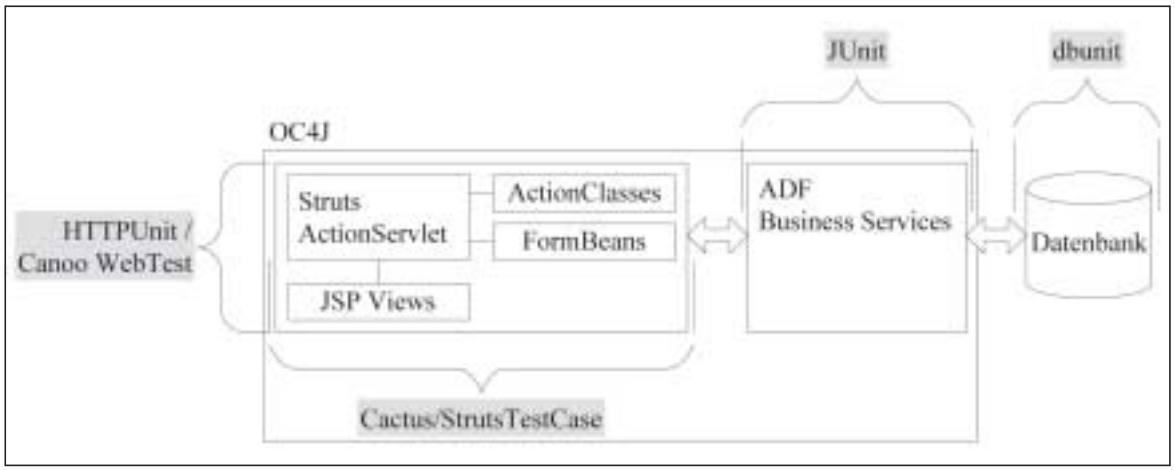

Abb. 2: Anwendung unterschiedlicher Test-Tools in einer mehrschichtigen Web-Anwendung

In Abbildung 2 ist die Struktur einer typischen Oracle ADF Anwendung dargestellt. Bezogen auf die einzelnen Stufen der Anwendung sind Werkzeuge für das Testen der jeweiligen Schicht angegeben. Im Idealfall sollten alle dargestellten Werkzeuge zum Einsatz kommen. So läßt sich eine sehr gute Abdeckung aller funktional relevanten Bereiche realisieren. Anzumerken gilt, dass der Aufwand für Erstellung und Pflege der Testfälle zur Präsentationsschicht hin steigt.

Neben der Unterscheidung nach Test-Granularität gibt es auch Einschränkungen hinsichtlich der verwendbaren Programmiersprache. So eignet sich Cactus nicht für den Test von PL/SQL-Anwendungen, eine PL/SQL-Web-Anwendung kann mit HTTPUnit aber sehr wohl getestet werden.

In-Container Tests mit Cactus

Beim In-Container Testing werden Unit-Tests direkt in einem J2EE-Container ausgeführt. Dies ist besonderrs bei serverseitigen Komponenten wie Servlets, Tag-Handlern oder EJBs vorteilhaft. Da die Unit-Tests im Kontext des J2EE-Containers laufen, läßt sich beispielsweise der Inhalt eines HttpServletRequest-Objektes sehr genau festlegen. Oder bei EJBs ist ein Zugriff über das Local-Interface möglich.

Web-Anwendungen auf der Basis des Oracle ADF verwenden meist Jakarta-Struts als View-Controller. In diesem Fall empfiehlt sich der Einsatz von StrutsTestCase [19] als Erweiterung von Cactus. Damit lassen sich sehr einfach Tests für StrutsAction-Objekte oder Struts FormBeans realisieren. Die Implementierung von Cactus-Testfällen erfolgt dabei analog zu JUnit.

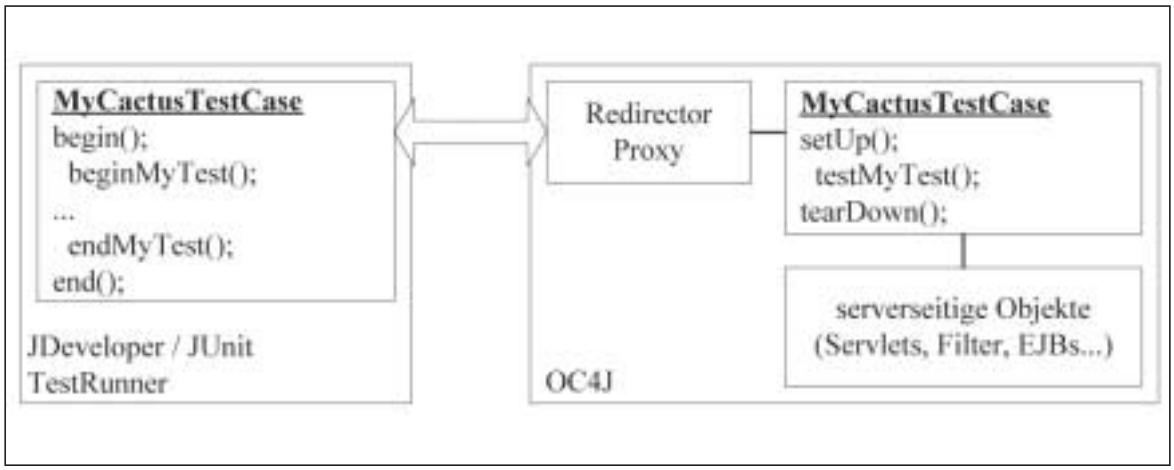

Abb. 3: In-Container Testing mit Cactus

Die von Cactus verwendete Architektur ist in Abbildung 3 dargestellt. Die Testklasse MyCactusTestCase wird dabei zweimal instanziert - sowohl auf der Clientseite in einem üblichen JUnit TestRunner wie auch auf der Serverseite. Die Kommunikation der beiden Instanzen erfolgt über einen Redirector-Proxy im J2EE-Container. Für Initialisierungs-aufgaben auf der Clientseite wurden analog zu den klassischen JUnit-Methoden `setUp()` und `tearDown()` die Methoden `begin(), beginMyTest()` und `endMyTest(), end()` eingeführt.

HTTPUnit im Detail

HTTPUnit überträgt die Funktionsweise von JUnit auf das Testen von Web-Applikationen. Die Testfälle werden hierbei analog zu JUnit in Form von Java Klassen definiert.

```
public class MyHttpUnitTestCase extends TestCase {
...
  public void testADFToyStoreIndex() throws Exception {
    WebConversation webConversation = new WebConversation();
    WebRequest index_request = new GetMethodWebRequest(
                                  "http://localhost:8988/ADFToyStore/");
    WebResponse index_response = webConversation.getResponse(index_request);
    assertEquals("ADF Toy Store Demo", index_response.getTitle());
  }
...
}
```

Abb. 4: HTTPUnit Beispiel Testmethode

Abbildung 4 zeigt ein einfaches Beispiel für einen Testfall. Zunächst wird durch Instanzieren der Klasse `WebConversation` ein Ausführungskontext erzeugt, der unter anderem zur Verwaltung von Cookies verwendet wird. Danach kann über `WebRequest` und `WebResponse` Objekte eine HTML-Antwortseite der Web-Anwendung abgefragt werden. Zum Schluß können wie von JUnit bekannt entsprechende Prüfungen der Antwortseite vorgenommen werden. Im dargestellten Beispiel wird der Titel der HTML-Seite geprüft.

HTTPUnit unterstüzt HTTP-Redirects und JavaScript. HTML-Antwortseiten können auch in der Form von XML-DOM Dokumenten weiterverarbeitet werden (z.B. mit XPath-Abfragen).

In der Praxis ist die manuelle Erzeugung und Verwaltung entsprechender Testfälle teilweise sehr aufwendig. Beim Erzeugen der Testfälle muss zunächst der Quellcode der HTML-Seiten ausgewertet werden, um anschließend die Codierung in Java vornehmen zu können. Die Verwendung von HTTPUnit Testfällen sollte daher auf ein notwendiges Maß begrenzt bleiben.

Im nachfolgenden Abschnitt wird ein XML-Wrapper für HTTPUnit vorgestellt, der hier ebenfalls für einige Arbeitserleichterung sorgen kann.

Deklarative Beschreibung von Testfällen

Die bisherigen Ausführungen orientierten sich stark an den Bedürfnissen von Softwareentwicklern. Für die Praxis müssen jedoch einige Restriktionen bedacht werden.

- Die Rahmenbedingungen von IT-Projekten gestatten es teilweise nicht, dass Testfälle z.B. in Java codiert werden (Zeit, Kosten).
- In vielen Projekten gibt es eine starke Trennung zwischen fachlichem Wissen der Anwendung und technischem Wissen bezüglich der Programmierung. Für das Erstellen von Akzeptanztestfällen ist meist fachliches Wissen erforderlich. Gibt es die beschriebene Spaltung in fachliches und technisches Wissen, ist meist wenig Know-How / Bereitschaft vorhanden die Testfälle z.B. in Java zu erstellen.
- Bei vielen Projekten müssen sehr viele Testfälle erstellt und gepflegt werden.

In der Praxis werden diese Probleme meist dadurch umgangen, dass die Testfälle entweder in einfacher Form deklarativ (z.B. in Excel oder in HTML-Tabellen) beschrieben werden, oder auf Capture-and-Replay-Tools zurückgegriffen wird.

Mit dem Werkzeug WebTest der Firma Canoo lassen sich beide Lösungsmöglichkeiten zusammen mit HTTPUnit verwirklichen. WebTest ist als Erweiterung zum Bulid-Tool Ant konzipiert und erlaubt damit die Beschreibung von Testfällen als XML.

```
<testSpec name="MyWebtestTestCase">
  <config host="localhost" port="8988"
          protocol="http" basepath="/ADFToyStore" />
  <steps>
    <invoke url="/" />
    <verifytitle text="ADF Toy Store Demo" />

    <clicklink label="Sign In" />
    <verifytitle text="Sign In" />

    <new_selectform name="loginform" />
    <new_setinputfield name="username" value="j2ee" />
    <new_setinputfield name="password" value="j2ee" />
    <clickbutton name="event_verifySignin" />
    <verifytitle text="Welcome to the ADF Toy Store Demo" />
  </steps>
</testSpec>
```

Abb. 5: Beispiel für einen WebTest XML Testfall

Das Beispiel in Abbildung 5 zeigt einen Ausschnitt aus einem derartigen XML-Testfall. Der Testfall besteht im wesentlichen aus der Konfiguration (Basis-URL) und nachfolgenden Schritten (Steps). Schritte können dabei Aktionen (`invoke, clicklink`), Verifikationen (`verifytitle, verifyxpath`) oder Kommandos für HTML-Form Daten (`new_selectform, new_setinputfield`) sein. Auf eine Trennung der Testdaten mit Properties wurde der Einfachheit halber verzichtet. Diese XML-Beschreibung könnte mit entsprechenden Makros z.B. auch aus Microsoft Excel-Tabellen generiert werden. Mit Click-O-Mat [20] steht aber auch ein Capture-and-Replay Werkzeug zur Verfügung.

Fazit

Durch den breiten Einsatz von Test-Tools lassen sich Funktionale Fehler in einer mehrstufigen Web-Anwendung schnell und zielsicher finden. Der Artikel zeigt aber auch deutlich, dass Open-Source-Tools in diesem Bereich mehr als Bausteine für eine Gesamtlösung zu sehen sind. Wer jedoch mit realistischem Blick nach Hilfsmitteln für eigene maßgeschneiderte Lösungen sucht, findet eine reichhaltige Auswahl an Werkzeugen und Frameworks.

Weiterführende Informationen/Links

[1] ISO 9126, Software Engineering - Product Quality
[2] mechanize, http://wwwsearch.sourceforge.net/mechanize/
[3] HTTP::Recorder, http://search.cpan.org/~leira/HTTP-Recorder/
[4] WWW::Mechanize, http://search.cpan.org/dist/WWW-Mechanize/
[5] HTMLUnit, http://htmlunit.sourceforge.net/
[6] HTTPUnit, http://www.httpunit.org/
[7] WebTest, http://webtest.canoo.com/manual/WebTestHome.html
[8] Jameleon, http://jameleon.sourceforge.net/
[10] MaxQ, http://maxq.tigris.org/

[11] Anteater/AFT, http://aft.sourceforge.net/
[12] MockObjects, http://www.mockobjects.com/FrontPage.html
[13] Cactus, http://jakarta.apache.org/cactus/index.html
[14] JUnitEE, http://www.junitee.org/
[15] dbunit, http://dbunit.sourceforge.net/
[16] FIT, http://fit.c2.com/
[17] Fitnesse, http://fitnesse.org/
[18] JwebUnit, http://jwebunit.sourceforge.net/
[19] StrutsTestCase, http://strutstestcase.sourceforge.net/
[20] WebTest Click-O-Mat,
http://webtest-community.canoo.com/wiki/space/Click-O-Mat

Kontaktadresse

Frank Berger
Bergstraße 22
D-89349 Burtenbach

Telefon: +49(0)8285-99510
E-Mail Frank.Berger@fm-berger.de
Internet: http://www.fm-berger.de

Shadow Connect GmbH
Carl-Reisch-Weg 12
D-86381 Krumbach

Telefon: +49(0)8282-99510
Fax: +49(0)8282-995111
E-Mail: Frank.Berger@shadowconnect.com
Internet: http://www.shadowconnect.com