Institut für Systemsoftware
O.Univ.-Prof. Dr. Hanspeter Mössenböck

JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Entwicklung eines Eclipse-Plugins für Regressionstests von serverbasierten Webanwendungen

Masterarbeitsthema für Günter Öller
Matrikelnummer: 0355112
Email: g_oeller@gmx.at

Webapplikationen haben mit einem starken Technologiewandel zu kämpfen. Heute existierende Webanwendungen können deshalb nicht stehen bleiben, sondern unterliegen einer ständigen, im Vergleich zu Desktopapplikationen noch rasanteren Entwicklung.

Im Rahmen dieser Diplomarbeit soll ein Eclipse-Plugin erstellt werden, das bei Regressionstests für kleine bis mittlere Webapplikationen eine Alternative zu den meist sehr teuren und komplexen existierenden Web-Test-Werkzeugen bietet.

Das zu entwickelnde Eclipse-Plugin soll eine benutzerfreundliche Lösung für das funktionale Testen von Webapplikationen bieten. Dabei soll der Fokus auf das Regressionstesten einer bereits existierenden Webanwendung gelegt werden. Wichtig ist, dass Änderungen in der zu testenden Webapplikation keinen großen Umstellungsaufwand für die Regressionstests nach sich ziehen.

Das Haupanwendungsszenario des Plugins stellt sich wie folgt dar:

1. Der Benutzer nimmt eine zu gewährleistende Funktionalität über einen Recorder auf. Dazu spielt er diese Funktionen in der zu testenden Applikation durch und lässt einen Proxy mitlaufen, der alles protokolliert.
2. Der Benutzer bearbeitet den so erstellten Test. Er führt Variable ein und legt Assertions an.
3. Zum Verifizieren der Funktionalität kann der Benutzer den Test (mit bzw. ohne Anzeige im Browser) abspielen und bekommt abschließend einen Bericht.

Das Plugin soll auf dem TPTP-Framework [1] aufbauen, welches bereits die Grundkomponenten in einer Beispielimplementierung enthält: Recorder, Testeditor und Report-Generator

Diese Funktionalität soll mit folgenden Zusatzdiensten, die gegliedert sind nach Phasen und Wichtigkeit, erweitert bzw. überlagert werden:

Test Aufzeichnung / Test Erstellung

Basisfunktionalität

- Filtern von Seiten-Typen während dem Aufzeichnen, zB Bilder, Java-Script-Dateien, Stylesheet-Dateien
- Der Recorder muss *Basic Authentication* und *Forms Authentication* aufzeichnen können.
- Es soll das Eclipse-Browser-View für das Aufzeichnen verwendet werden. Hier kann alternativ ein interner oder ein externer Browser ausgewählt werden.
- Der Recorder muss als Proxy im verwendeten Browser gesetzt werden. Dadurch kann der Recorder alles mitprotokollieren. Das Setzen des Proxies im Browser zum Aufzeichnen und das Zurücksetzen soll automatisch vom Recorder gemacht werden.

Wichtige Erweiterung

- *Web-Crawler*-Funktion: es wird während des Aufzeichnens von der aktuellen HTML-Seite aus jeder Link für eine konfigurierbare Tiefe weiterverfolgt und aufgezeichnet.
- *Form-Test*-Funktion: es wird von der aktuellen HTML-Seite aus jede Button-Pressed-Kombination in einem HTML-Formular für eine konfigurierbare Anzahl dieser Aktionen durchgespielt und aufgezeichnet.

Zusätzliche Funktion

- *Recorder-Loop*-Funktion: mehrmaliges Ausführen eines Requests mit variablen Parametern, die über *Data Pools* bzw. *Map*-Strukturen (siehe dazu Phase: Nachbearbeitung) befüllt werden oder die durch Parametertransformationen verändert werden (zB erhöhen).

Nachbearbeitung

Basisfunktionalität

- Die Struktur des erstellten Tests soll in einer Outline-ähnlichen Weise dargestellt werden.
- Requests sollen über eine *Property View* bearbeitet werden können. Dabei sollen sowohl Post- wie auch Get-Parameter automatisch erkannt und vom Benutzer geändert werden können.
- Die Anzeige des Response-Bodies soll über *pluggable Views* gelöst werden. Implementiert bzw. eingebunden werden zumindest Ansichten für HTML, DOM und HTML-Quelltext.
- Assertions müssen zu aufgenommenen Requests zugeordnet werden können.

- Verschieden Arten von Assertions:
  - *Content Check*: Ist ein bestimmter Text vorhanden bzw. nicht vorhanden? Sind die Werte der Map-Strukturen vorhanden?
  - *Response Check*: Header-Felder, Antwortzeit, Response-Größe, Response-Code usw.
  - Evtl. *Summary Check*: zB die Summe von Werten oder die Anzahl, das Minimum oder das Maximum von Zeilen in einer Tabelle usw.
  - und *Programmatic Check*: konfigurierbare Java-Testklasse unter Verwendung des HTMLUnit-Test-Frameworks [2])
- Es soll ein XML-Format für Standard-Assertions wie für Content, Response und Summary Check definiert werden.
- Die XML-Dateien der Standard-Assertions werden von HTMLUnit-Testfällen abgearbeitet.
- Erstellung von Content Check-Assertions mit RegEx.
- Erstellung von Content Check-Assertions aus der DOM-View: direkte Auswahl oder XPath-Ausdruck verwendbar.
- Erstellung von Content Check-Assertions durch eine Auswahl in einer HTML-Seite.
- Content Check-Assertions und Responses sollen vor dem Vergleichen normalisierbar sein: führende und abschließende Whitespaces entfernen, Tags nach einer Konvention formatieren und XHTML erzeugen.
- Erstellung von Map-Strukturen mittels Suche nach einem Template im HTML-Quelltext. Das Template gibt an wie ein Name-Wert Paar aussieht. Hier könnte auch wieder RegEx verwendet werden.
- Filter-Funktion für die Ergebnisbaum-Anzeige. ZB Requests und Responses mit fehlgeschlagenen Assertions oder nur Bilder anzeigen etc.
- Assertions sollen als Tools abgespeichert werden können und wieder verwendbar sein.
- Kopieren und Einfügen soll im Ergebnisbaum unterstützt werden.
- Eine Content Check-Assertion auf einen ausgewählten Response aus dem Ergebnisbaum testen können.

Wichtige Erweiterung

- Assertions sollen auch über einen gesamten Block von Request gelegt werden können.
- Assertions sollen ab einem bestimmten Punkt im Ergebnisbaum eingeschaltet und bei einem anderen Punkt wieder ausgeschaltet werden können.

- Zum Testen von Content Check-Assertions kann der aktuell angezeigte Browserinhalt verwendet werden.
- RegEx-Implementierung pluggable umsetzen (als Standard Apache Jarkata ORO [3] oder Eclipse-Regular Expressions verwenden).
- Map-Strukturen mit benutzerdefinierten Datentypen.
- Der Name in Name-Wert-Paaren von Map-Strukturen soll nicht ein einziger String sein, sondern eine Liste von möglichen Schreibweisen (Internationalisierung).
- Suchen und Ersetzen Funktion im gesamten bzw. durch Filtern eingeschränkten Ergebnisbaum. Dadurch können zB generell geänderte Bezeichner in Responses oder in Assertions leicht geändert werden.
- Anzeige der Content Check-Assertions im Referenzquelltext, in der HTML-Darstellung und in DOM-Darstellung. Es soll auch möglich sein die Assertion in den aufgezeichneten Responses zu betrachten. Dabei soll der ausgewählte Abschnitt im Response angezeigt bzw. markiert werden.
- Ersetzten des Referenz-Responses durch einen gerade aufgezeichneten Response.
- Content Check-Assertions im HTML-Quelltext mit RegEx bearbeiten. Dabei soll der gesuchte HTML-Quelltext automatisch in einen regulären Ausdruck verwandelt werden. ZB Escape für Sonderzeichen.

Zusätzliche Funktion

- Der HTML-Quelltext des Referenz-Responses soll ebenfalls editierbar sein.
- Test-Templates erstellen zB 1) Login 2) Logged-In-Assertion 3) <spezieller Test> 4) Logout 5) Logged-Out-Assertion

Verifikation / Test Ausführung

Basisfunktionalität

- Es soll möglich sein Konstanten zu definier und für einen Testlauf umzustellen. ZB für Server, Port und Anwendungskontext.
- Beim Abspielen der Tests soll es möglich sein dynamische Parameter zu verwenden, die aus den vorhergehenden Requests oder Responses gelesen werden.
- Vorher definierte Parameter und Konstante sollen in Requests und Assertions verwendbar sein.
- Ant-Task zum Ausführen von Tests.

- Einstellbares Fehlerverhalten bei fehlgeschlagenen Assertions: Proceed, Retry, Retry Whole Test (Suit), Stop. Weiters ein Schalter der angibt, ob ein Popup erscheinen soll, das den Testlauf pausiert.
- Tests ablaufen lassen mit und ohne Anzeige im Browser. Geschwindigkeitsvorteil, wenn HTML nicht dargestellt werden muss.
- Abspielen von Blöcken: gesamte Suite, einzelne Tests oder einzelne Requests.

Wichtige Erweiterung

- User Think Time aufzeichnen und abspielen; Standard-Think Time einstellen; Schalter zum Ignorieren der Think Time während des Abspielens.
- Breakpoints sollen für das Abspielen des Tests im Ergebnisbaum gesetzt werden können.
- Der Name von Name-Wert-Paaren wird nicht nur Eins-Zu-Eins im Response gesucht, sondern es wird auch das verwendete Template herangezogen. Es wird dann die Stelle im Response für einen Wertvergleich gewählt, die diesem Namen am ähnlichsten ist. (Ähnlichkeit von Text? Algorithmus noch nicht bekannt?)

Zusätzliche Funktion

- Es soll möglich sein die nächsten Schritte eines Testes als Aktionen auf Elemente eines vorhergehenden Responses zu definieren. ZB erster Link in der ersten Tabelle der HTML-Seite wird angeklickt.

Testberichte

Basisfunktionalität

- Abspeichern des Testberichts als XML-Datei: Testbericht beinhaltet zumindest: Dauer des Tests, Anzahl der Requests, Anzahl der fehlerhaften und erfolgreichen Assertions, Vergleich der erwarteten und der tatsächlichen Werte bei fehlgeschlagenen Assertions.
- Anzeige des Testberichts als HTML; über XSL-Transformierung.
- Vergleichen der HTML-, Quelltext- und DOM-Darstellung von gleichen Responses unterschiedlicher Testläufe bzw. des Referenz-Response.

Wichtige Erweiterung

- Gefundene Fehler, d.h. fehlgeschlagene Assertions, mit Bug-Tracking-Systemen verbinden. Das Bug-Tracking-System pluggable umsetzen. Beispielschnittstelle mit TRAC [4].
- Zurücksetzen des Ergebnisbaums, um alle Testausführungsergebnisse zu löschen.

Zusätzliche Funktion

- (zurzeit noch keine bekannt)

Generelle Einstellungen / Vorbereitungen

Basisfunktionalität

- Parameter für Requests, die aus einem Data Pool befüllt werden. Data Pool ist als fixe Liste von Werten vom Benutzer zu definieren.
- Das Plugin muss Redirects und Cookies richtig aufzeichnen und abspielen können.
- Es muss das durch TPTP definierte Testmodell verwendet werden. Wenn nötig sind Erweiterungen mit Hilfe von EMF erlaubt.
- Es muss möglich sein den Ergebnisbaums im XML-Format abzuspeichern. Dabei sollen auch, alle Responses und Testergebnisse abgespeichert werden können.
- Es soll eine schrittweise Benutzerführung gewählt werden (ähnlich einem Wizzard), wobei es immer möglich sein muss, in jede der vorherigen Phasen zurückzukehren.

Wichtige Erweiterung

- Parameter und Data Pools aus CSV-Dateien lesen.
- Parameter und Data Pools aus XML-Dateien lesen.
- Funktion zum Löschen des Browser-Caches, der -History und der -Cookies vor einer Testausführung.
- Testen von Datei-Uploads.
- Installation des Eclipse-Plugins via Eclipse-Update-Sites.

Zusätzliche Funktion

- Parameter soll durch eine Zufallszahl oder durch einen Zufallswert aus einer Liste definiert werden können.
- HTTPS Unterstützung.
- IP-Spoofing soll möglich sein.
- Scheduling von Tests soll möglich sein.
- Es soll möglich sein nach Beendigung eines Tests eine E-Mail mit dem HTML-Testbericht an eine oder mehrere vorher definierte E-Mail-Adresse(n) zu senden.

Zurzeit nicht benötigt:

- Java-Script Unterstützung.
- Browser-Popups testen
- Behandlung von Multiframe-HTML-Seiten
- Applet
- Flash/Flex
- ActiveX
- Multiple Browser-Fenster
- SSL-Zertifikate

Wie aus dieser Übersicht hervorgeht, soll das Eclipse-Plugin dem Benutzer besonderen Komfort im Bereich der Assertion-Erstellung aus einem Referenz-Response bieten. Dadurch wird es den Benutzern ermöglicht, schnell und einfach Tests für Hunderte oder Tausende von HTML-Seiten zu erstellen.

Das zu entwickelnde Eclipse-Plugin soll die Motivation von Webentwickler bei der Erstellung von Regressionstest steigern und damit auch neue Technologien fördern. Durch Einsetzen von Regressionstests werden sich die Entwickler viel sicherer fühlen, wenn sie in einem bereits laufenden Projekt auf eine neue Technologien umsteigen. Sie brauchen nach der Umstellung nur den Regressionstest laufen lassen und auf „Grün“ zu warten.

Zurzeit hindern meist der hohe Preis der guten Werkzeuge und die aufwendige Erstellung von umfangreichen Testfällen die Entwickler an der Umsetzung eines durchgehenden Regressionstests für ihre Anwendungen. Das in dieser Masterarbeit zu erstellende Eclipse-Plugin soll hier die Lücke füllen und den Webentwicklern diese Arbeit erleichtern.

Betreuer: Prof. Dr. Hanspeter Mössenböck

Beginn: Oktober 2007

Referenzen

1 [http://www.eclipse.org/tptp/]

2 [http://htmlunit.sourceforge.net]

2 [http://jakarta.apache.org/oro/]

4 [http://trac.edgewall.org]