Testkonzept Übungsverwaltung

Einführung

Automatische Tests von Klassen sind ein wesentlicher Bestandteil des „Xtreme Programming“-Paradigmas. Im wesentlichen unterteilt es sich in zwei Teile: Komponenten- und Systemtests. Dabei werden in jeweils speziellen Formaten Erwartungen an die Funktionalität des Codes in songenannten „testcases“ festgehalten und gegen den Code laufengelassen, um festzustellen, ob implementierte Methoden wirklich die erwarteten Ausgaben erzeugen. In unserem Projekt verwenden wir für Komponententests das Framework „JUnit“ und für die Systemtests „Cactus“, eine erweiterte Form von JUnit für den Test von J2EE-Code.

Komponententests

Die automatischen Tests zu den einzelnen Komponenten erfolgen einzeln für jedes Paket sowie für jede implementierte Klasse. Dazu wird zu jeder Klasse eine Testklasse erzeugt, die von *junit.framework.TestCase* abgeleitet wird. Zu jeder zu testenden Methode *XXXX* wird eine *testXXX*-Methode in der Testklasse erstellt. In dieser erfolgt der Aufruf der zutestenden Methode zusammen mit jeweils passenden Parametern. Danach werden in diversen *AssertXXX* –Anweisungen die zu erfüllenden Erwartungen formuliert. Sollten diese Erwartungen nicht erfüllt werden, bricht der Test an dieser Stelle mit der Ausgabe des stacktraces ab. Sobald also alle definierten Tests erfüllt werden, bedeutet dies, dass alle im Vorfeld erkannten möglichen Fehler ausgeschlossen sind.

Eine Beispieltestklasse ist folgende:

```
import java.util.*;
import junit.framework.*;
public class SimpleTest extends TestCase {
   public SimpleTest(String name) {
      super(name);
   }
  public void testEmptyCollection() {
      Collection collection = new ArrayList();
      assertTrue(collection.isEmpty());
   }
   public static Test suite() {
      return new TestSuite(SimpleTest.class);
   }
  public static void main(String args[]) {
      junit.textui.TestRunner.run(suite());
   }
}
```

In diesem Beispiel wir getestet, ob eine neu angelegte Collection leer ist (*assertTrue(collection.isEmpty())).* Gibt die Methode isEmpty() false zurück, würde der Test abbrechen.

Es gibt verschiedene Möglichkeiten, die so erstellten Tests auszuführen. Über die main()-Methode kann gezielt eine einzelne Testklasse durchlaufen werden. Eine weitere Möglichkeit ist es, eine TestSuite zu erstellen, welche eine Aggregation von TestCases darstellt, die alle durchlaufen werden. In unserem Projekt sollen die Testklassen jeweils eines Paketes zu einer TestSuite, sowie die Suites des Projektes zu einer Gesamt-TestSuite zusammengefasst werden, so dass sichergestellt ist, das mit einem Test alle Komponenten einer funktionellen Einheit

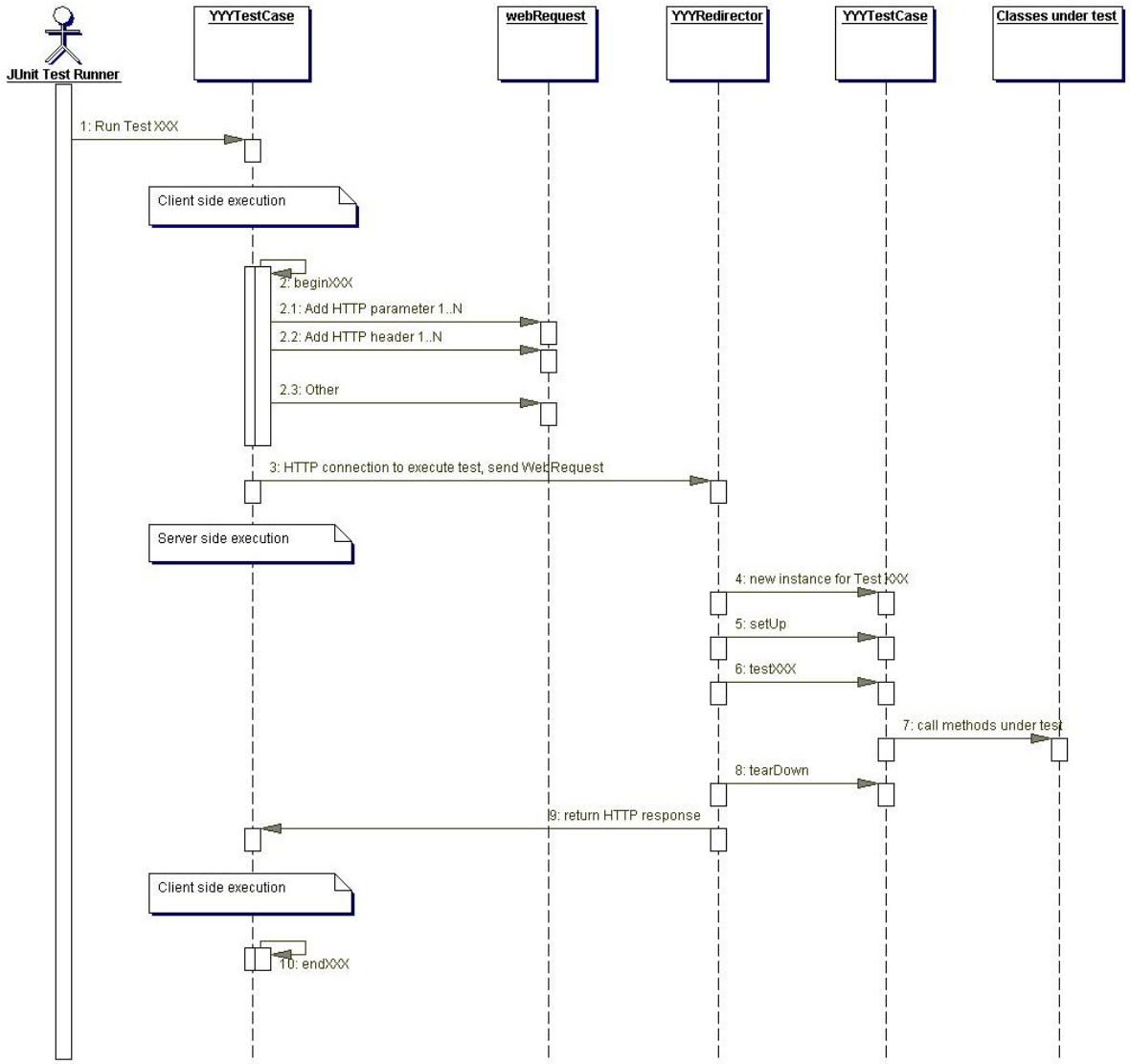
durchlaufen werden. Die genauentechnischen Details der Implementierung der Tests werden noch im Einzelnen in der Gruppe besprochen.

Quelle: http://sourceforge.net/projects/junit/

Systemtests

Ein Systemtest der gesamten Anwendung ist insofern schwierig, da es sich um eine Internetanwendung handelt. Wir verwenden hierfür das „Cactus“-Framework, welches eine Erweiterung von JUnit darstellt. Die Funktion ist folgende: Es wird auf ähnliche Art wie bei JUnit ein TestCase erzeugt. In diesem stehen drei Methoden für jeden auszuführenden Test: Eine *beginXXX()*-Methode, eine *testXXX()*-Methode und eine *endXXX()*-Methode. In der begin-Methode werden Parameter für den http-Request festgelegt wie beispielsweise die Ursprungs-URL, etwaige Cookies etc. Die *testXXX()*-Methode führt die eigentlichen Tests aus und die korrespondierende *endXXX()*-Methode prüft die Rückgabe des Servlets, wie z.B. den html-Output, Cookies...
Um diese Tests durchzuführen muss Cactus im Servletcontainer der zu testenden Anwendung sowie lokal installiert sein. Beim Ausführen des Tests initialisiert Cactus in den *beginXXX()*-Methoden die Anfragen, sendet diese an eine Proxy-Klasse im Servletcontainer, welche die dortige *testXXX()*-Methode ausführt, die Ergebnisse zurückschickt, wo wiederum das lokale Cactus mit Hilfe der *endXXX()*-Methode die Ergebnisse evaluiert.

Diagramm:



Auf diese Weise können nun Systemtests am laufenden System durchgeführt werden. Auch diese Tests sollen in unserem Projekt wieder in einer jeweils zu aktualisierenden TestSuite zusammengefasst werden, so dass möglichst einfach umfassende Tests automatisch ausgeführt werden können.