Ausarbeitung zum Seminar

Web-Testen

mit

# JUnit und HttpUnit

Referent: Kai Schmitz-Hofbauer Betreuer: Dipl. Inform. Peter Ziesche

3. Februar 2003

# Inhaltsverzeichnis

# 1 Einleitung

*„Gestern hat es noch funktioniert !”* oder *„Ich dachte, das hätte ich bereits korrigiert!”* sind Aussagen, die vielen bekannt sein dürften, die bereits mit Software-Entwicklung in Berührung gekommen sind. Diese Aussprüche fallen meistens dann, wenn das Testen der entsprechenden Software vernachlässigt wurde. Software wird heutzutage in den verschiedensten Bereichen eingesetzt. Der Einsatz von Softwaresystemen hat mittlerweile Einzug in sämtliche Geschäftsbereiche erhalten. Es bestehen hohe Anforderung an die Qualität neu entwickelter Software. Die Qualtitätssicherung rückt immer mehr ins Rampenlicht. In den letzten Jahren durchlief die Softwareentwicklung stetige Änderungen. Es wurden und werden immer neuere und „bessere” Techniken entwickelt und angewendet. Auch die Komplexität von Software-Systemen steigt von Jahr zu Jahr. Daraus resultiert ein Bedarf an neuen Test-Methoden und Werkzeugen. Im Rahmen dieser Seminararbeit wird das Testen einer Web-Anwendung behandelt. Eine Web-Anwendung zeichnet sich dadurch aus, dass ein Dienst mehreren Benutzern über das Internet zur Verfügung gestellt wird. Der Benutzer kann in der Regel mit einem Web-Browser den entsprechenden Dienst nutzen. Typische Beispiele für Web-Anwendungen sind *online-banking, online-Versteigerungen, e-commerce* oder *e-learning-Plattformen.* Web-Anwendungen arbeiten oft mit dynamischen HTML-Seiten, d. h. auf einem Server wird eine HTML-Datei erzeugt, deren Inhalt meist abhängig von einer Benutzerinteraktion ist, und anschließend zum Web-Browser geschickt. Dort kann die Seite angezeigt werden. Da sich die Architektur einer Web-Anwendung gravierend von der Architektur einer „herkömmlichen” Anwendung unterscheidet und daher auch der Test-Prozess anders verläuft, muss auf spezielle Testtechniken- und Werkzeuge zurückgegriffen werden. In dieser Ausarbeitung wird das Web-Testen mit Hilfe von HttpUnit beschrieben. HttpUnit ist ein Java-Paket, mit dem es möglich ist, HTML-Webseiten (automatisiert) zu testen. Sämtliche Tests wurden an der e-learning-Plattform W3L durchgeführt, die derzeit unter der Leitung von Herrn und Frau Prof. Dr. H. Balzert als Gemeinschaftsprojekt der Ruhr-Universität Bochum und der Fachhochschule Dortmund entwickelt wird.

# 2 Automatisiertes Testen

Soll ein Software-Produkt getestet werden, so werden Tests of manuell durchgeführt. Damit ist Folgendes gemeint: Ein Software-Tester bedient persönlich den Testling und überprüft „per Hand”, ob sich das Produkt entsprechend den Produktspezifikationen verhält. Moderne Test-

werkzeuge haben den Vorteil, dass Tests automatisiert durchgeführt werden können. Automatisierte Testverfahren haben gegenüber manuellem Testen folgende Vorteile:

- **Geschwindigkeit:** Eine automatisierte Testausführung benötigt lediglich einen Bruchteil der Zeit einer manuellen Durchführung und ist außerdem nicht so fehleranfällig.

- **Präzision:** Nach mehreren manuell durchgeführten Tests leidet die Konzentration des Testers. Infolgedessen schleichen sich oft Fehler ein. Durch automatisiertes Testen kann derselbe Test beliebig oft hintereinander mit der gleichen Präzision ausgeführt werden. Flüchtigkeitsfehler wie z. B. Eingabefehler oder Ablesefehler können ausgeschlossen werden.

- **Wiederverwendbarkeit:** Erstellte Test lassen sich jederzeit, z. B. zum Zweck eines Regressionstests, wieder verwenden. Existierende Programme können bei einer erneuten Freigabe in einer leicht modifizierten Form erneut getestet werden. Die Testprogramme müssen dafür nur leicht verändert werden.

- **Effizienz:** Manuelles Testen ist sehr zeitaufwändig. Bei der Automatisierung ist zwar ein einmaliger Aufwand für die Testimplementierung notwendig, für weitere Testdurchführungen sind aber nur noch Wartungsarbeiten nötig. Die dadurch eingesparte Arbeitszeit kann anderweitig genutzt werden.

Es sei an dieser Stelle ausdrücklich darauf hingewiesen, dass Testwerkzeuge, die eine automatisierte Testdurchführung unterstützen, niemals die Testperson ersetzen können. Sie unterstützen und vereinfachen lediglich deren Arbeit. So müssen die Testfälle nach wie vor selbst entworfen werden. Der Entwurf von Testfällen ist ein kreativer, intellektuell herausfordernder Prozess und ist daher für eine Automatisierung ungeeignet.

# 3 Unit-Tests

Bei umfangreichen Softwareprojekten wäre es nicht sinnvoll, erst am Ende das fertige Projekt zu testen. Es ist sinnvoll einzelne Komponenten *(Units)*, bei denen es sich um kleine, abgeschlossene Einheiten des Gesamtprojektes handelt, zu testen. Bei diesen Komponenten kann es sich z. B. um einzelne Funktionen, Klassen oder um Webseiten handeln. Bei diesem Vorgehen erweist es sich als vorteilhaft, dass Fehler leichter lokalisiert und anschließend korrigiert

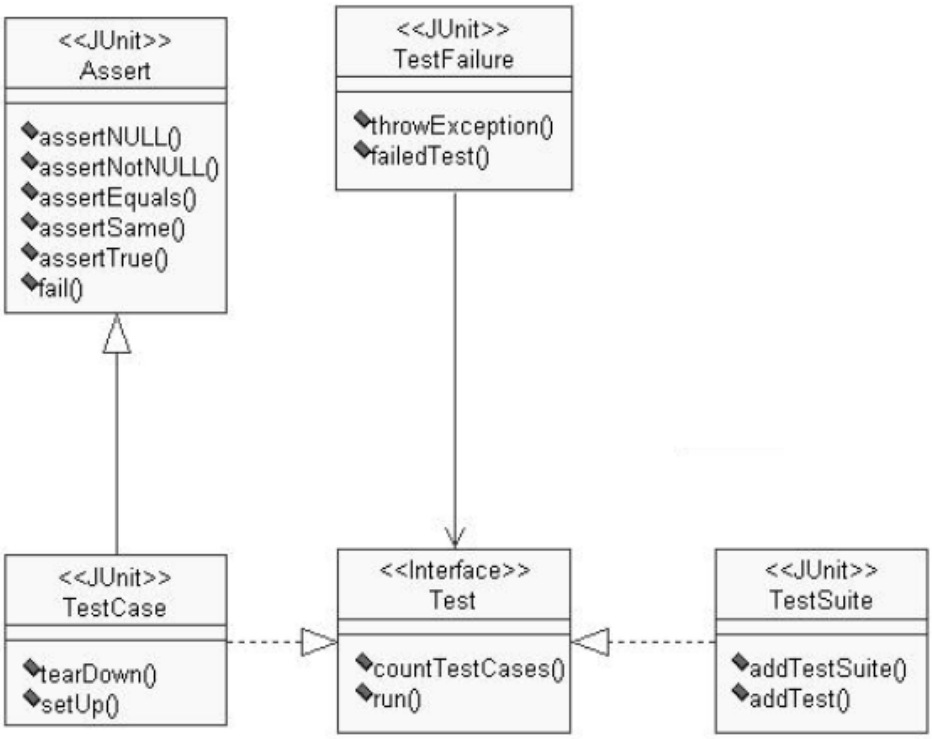
werden können. Die Durchführung von Unit-Tests soll im Folgenden anhand von JUnit, einem Java-Paket zur Ausführung automatisierter Unit-Tests, und HttpUnit, einer Erweiterung von JUnit für verschiedene Web-Probleme, erläutert werden.

## 3.1 JUnit

JUnit richtet sich in erster Linie an den Softwareentwickler und nicht an das Qualitätsmanagment. JUnit ist als Testumgebung von *eXtreme Programming (XP)* berühmt geworden. Bei *XP* handelt es sich um eine Entwicklungsmethode, bei der zum Thema Testen empfohlen wird, dass der Testfall geschrieben wird, bevor überhaupt das zu testende Modul dazu implementiert wird. Der Softwareentwickler muss sich vor Beginn der Implementierung über die genauen Spezifikationen im Klaren sein. Mit dem Test legt er einen Rahmen dazu fest. Bei der späteren Implementierung hat man Kontrolle darüber, ob die Spezifikationen, die man vorher in Form eines Tests festgelegt hat, erfüllt sind oder nicht. Werden Tests und Software in dieser Reihenfolge entwickelt, wird sichergestellt, dass jedes Modul durch Tests geschützt ist. Dieses Vorgehen wird auch als *testgetriebene Entwicklung* bezeichnet.

### 3.1.1 Die wichtigsten Klassen von JUnit

JUnit ist ein kleines, mächtiges Java-Framework zum Schreiben und Ausführen automatischer Unit Tests. Die Tests werden direkt in Java geschrieben. Die folgenden Klassen sind die wichtigsten Klassen des JUnit-Frameworkes :




**junit.framework.TestCase** Die Klasse TestCase ist eine abstrakte Basisklasse, die als Grundlage für sämtliche Testfälle dient. Sämtliche Klassen, die Testfälle implementieren, müssen von TestCase erben. Für jeden Testfall muss in so einer Klasse eine öffentliche Methode implementiert werden. Per Konvention sollte der Methodenname mit *test* beginnen, gefolgt von einem aussagekräftigen Bezeichner, der angibt, was getestet werden soll (z. B. `testLogin()`). Die Ausführung der Tests wird durch die Methode `runTest` angestoßen. Dies geschieht unter Verwendung des `Reflection-APIs` von Java. Die Klasse TestCase stellt außerdem die Methoden `setUp()` und `tearDown()` zur Verfügung. Mit `setUp()` können Initialisierungen der Testklasse, wie z. B. Instanziierung benötigter Objekte, durchgeführt werden, mit `tearDown()` Aufräumarbeiten durchgeführt werden. Die beidem Methoden werden vor und nach der Ausführung jedes Testfalles aufgerufen, so dass die Unabhängigkeit der Testfälle gewährleistet ist.

**junit.framework.Assert** Die Klasse Assert stellt einige Methoden zur Verfügung, mit der möglich ist, Werte und Bedingungen zu testen. Diese müssen erfüllt sein, damit der Test erfolgreich durchläuft. Schlägt bei der Durchführung aller Tests ein Test fehl, so wird bei der Testausführung eine Fehlermeldung mit Angabe des fehlgeschlagenen Tests ausgegeben. Diese Aufgaben werden durch spezialisierte `assert`-Methoden erledigt. Die wichtigsten sollen im

Folgenden kurz erläutert werden :

- `assertTrue(boolean condition)` prüft, ob die logische Bedingung erfüllt, also wahr ist.
- `assertFalse(boolean condition)` prüft, ob die logische Bedingung nicht erfüllt, also falsch ist.
- `assertNull(Object object)` prüft, ob eine Referenz null ist.
- `assertNotNull(Object object)` prüft, ob eine Referenz von null verschieden ist.
- `assertEquals(Object expected, Object actual)` überprüft die angegebenen Objekte auf Gleichheit. Diese Methode existiert nicht nur für Parameter vom Typ `Object`, sondern auch für sämtliche primitive Datentypen *(z. B. int, double, float etc)*.
- `assertSame(Object expected, Object actual)` überprüft, ob zwei Referenzen auf das gleiche Objekt verweisen.
- `fail(String message)` bricht den laufenden Test mit der angegeben Fehlermeldung ab.

Die im Testcode durch `assert`-Anweisungen kodierten Behauptungen werden von der Klasse Assert automatisch verifiziert. Im Fehlerfall wird der laufende Testfall sofort abgebrochen und eine `AssertionFailedException` mit einem entsprechenden Fehlerprotokoll ausgelöst. Diese Klasse ist eine direkte Oberklasse von TestCase. Sämtliche assert-Methoden werden an TestCase vererbt.

**junit.framework.TestSuite** Die Klasse TestSuite bietet ein Grundgerüst, um beliebig viele Testfälle zusammenzufassen. Einer TestSuite können beliebig viele Tests und selbst andere Testsuiten hinzugefügt werden. Um entsprechende Objekte einer TestSuite hinzuzufügen, stehen die Methoden `addTest()` und `addTestSuite()` zur Verfügung. Auf diese Art und Weise lassen sich die Testfälle eines Projektes entsprechend ihren Aufgaben sehr gut strukturieren und organisieren.

**junit.xxx.TestRunner** Die Klasse Testrunner dient zum Start und zur Ergebniskontrolle der Tests. Es gibt drei Varianten :

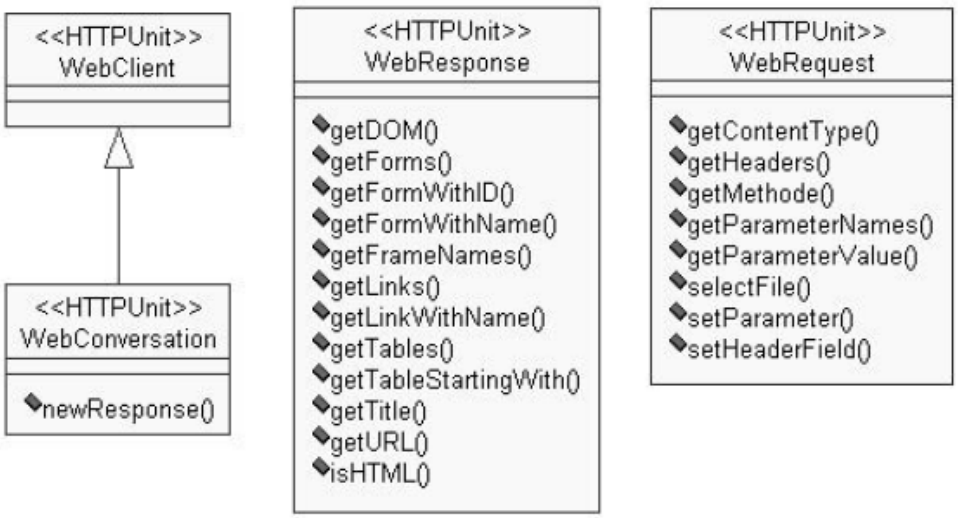
1. `junit.textui.TestRunner`(textbasierte Variante des TestRunners)
2. `junit.awtui.TestRunner` (graphische Variante des TestRunners auf awt-Basis)
3. `junit.swing.TestRunner` (graphische Variante des TestRunners auf swing-Basis)

Es sei noch erwähnt, dass es sich bei JUnit-Tests um *Black-Box-Tests* haldelt.

## 3.2 HttpUnit

Normalerweise greift man auf eine Web-Seite mit einem Browser zu. Dieses Vorgehen ist aber für eine automatisierte Testdurchführung ungeeignet. Es ist wünschenswert, auf Webseiten direkt aus einem Programm wie z. B. einem JUnit-`TastCase`, zuzugreifen. HttpUnit stellt diese Funktionalität zur Verfügung. Es bietet sowohl die Möglichkeit, auf die Inhalte einer Seite zuzugreifen als auch Informationen über die Verbindung zu erfragen. HttpUnit ist genau wie JUnit in Java geschrieben.

### 3.2.1 Die wichtigsten Klassen von HttpUnit




Die wichtigsten Klassen sollen im Folgenden kurz vorgestellt werden.

**com.meterware.httpunit.WebConversation** Die Klasse WebConversation repräsentiert den Kommunikationsvorgang zwischen der Test-Klasse und einem Web-Server. Die gesamte Kom-

munikation wird über diese Klasse abgewickelt. Sie emuliert das Verhalten und die Funktionalität eines Web-Browsers.

**com.meterware.httpunit.WebRequest** Die Klasse WebRequest repräsentiert eine Anfrage, die an einen WebServer gesendet wird. Im Konstruktor kann dieser Klasse eine entsprechende URL übergeben werden.

**com.meterware.httpunit.WebResponse** Die Klasse WebResponse repräsentiert die Antwort eines Web-Servers auf eine Anfrage (WebRequest). An ein Objekt dieser Klasse gelangt man auf folgende Weise:

1. Erzeugen einer neuen WebConversation mit
   ```
   WebConversation conversation = new WebConversation();
   ```
2. Erstellen einer neuen Anfrage mit
   ```
   WebRequest request  = new GetMethodWebRequest(url);
   ```
3. Anfrage abschicken und Antwort einholen mit
   ```
   WebResponse response = conversation.newResponse(request);
   ```

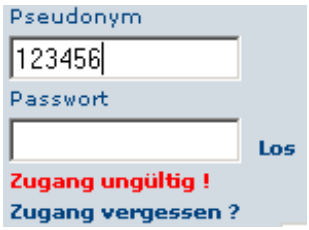
Objekte der Klasse WebResponse ermöglichen den Zugriff auf alle Elemente einer Web-Seite. Für den Zugriff auf die in die Seite eingebetteten Elemente, wie z. B. Tabellen oder Formulare, gibt es die Klassen WebTable, WebForm, für Verweise die Klasse WebLink, für Bilder die Klasse WebImage etc., die durch die entsprechende get-Methode des WebResponse-Objektes zurückgeliefert werden.

# 4 Download und Installation

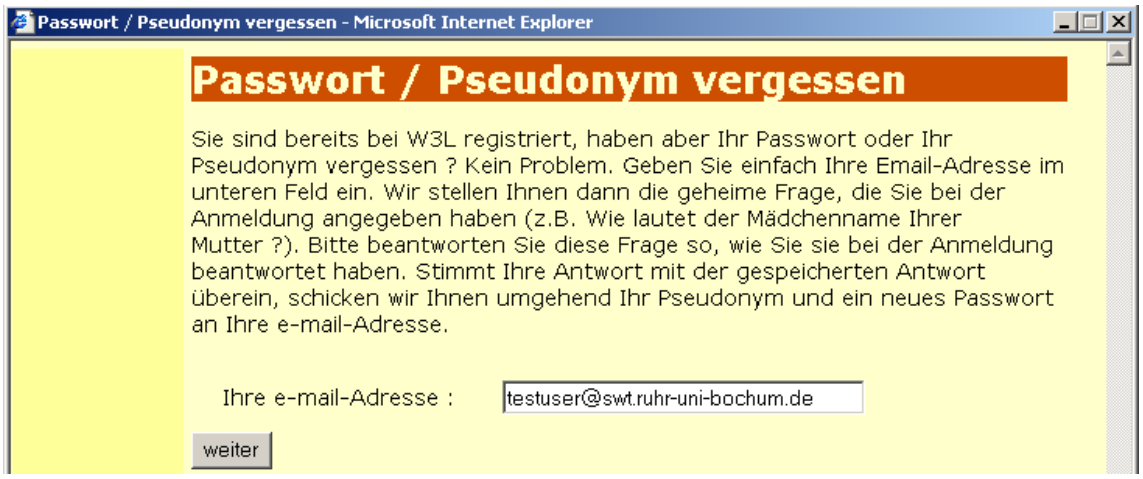
Das HttpUnit-Paket kann bei `http://www.httpunit.org` heruntergeladen werden. Das heruntergeladene Paket muss entpackt und in ein Verzeichnis der Wahl kopiert werden. Das Archiv `httpunit.jar` und sämtliche Archive, die sich im Unterverzeichnis `jars` befinden, müssen in den `CLASSPATH` aufgenommen werden. In dem Unterverzeichnis `jars` befindet sich u. a. auch das JUnit-Paket.

# 5 Durchführen von Tests

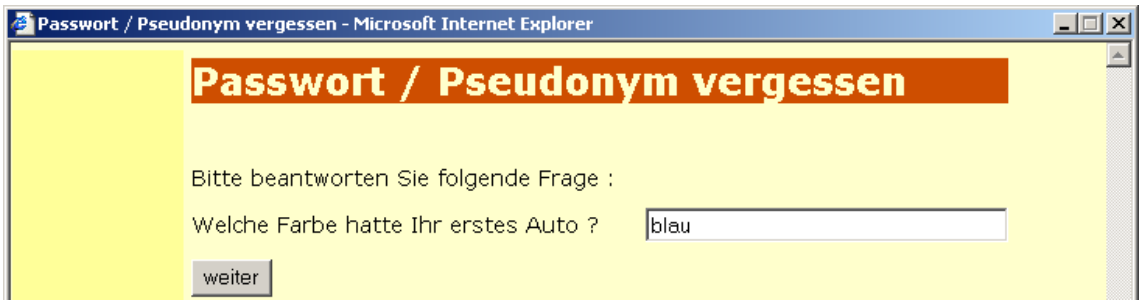
Im Rahmen dieses Seminares wurde eine Web-Seite der e-learning-Plattform W3L mit JUnit und HttpUnit getestest. Bei der zu testenden Seite handelt sich um die Seite, die aufgerufen werden kann, falls ein W3L-Benutzer sein Pseudonym bzw. sein Passwort vergessen hat. Schlägt der Versuch, sich bei W3L anzumelden fehl, so wird unter den Eingabefeldern ein Verweis (Link) mit dem Text *Zugang vergessen?* angezeigt.




Folgt man diesem Verweis, so gelangt man zu der zu testenden Web-Seite. Hier wird der Benutzer aufgefordert seine e-mail-Adresse anzugeben.

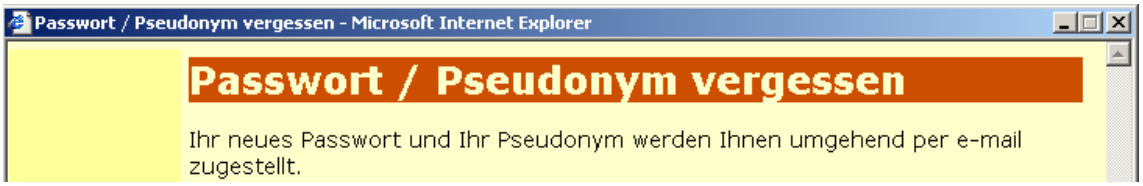



Konnte zu der angegebenen e-mail-Adresse ein bereits bei W3L registrierter Benutzer gefunden werden, so wird ihm eine geheime Frage gestellt, die er beantworten muss.




Die geheime Frage muss bei der Anmeldung bei W3L aus einer Liste von möglichen Fragen ausgewählt werden. Gleichzeitig muss eine dazugehörige Antwort angegeben werden. Bei er-

folgreicher Beantwortung dieser Frage werden dem Benutzer dann das Pseudonym und ein neues Passwort per e-mail zugestellt.




Es sollen drei Testfälle ausgeführt werden :

1. Es wird eine e-mail-Adresse angegeben, unter der kein Benutzer im System registriert ist.
2. Es wird eine gültige e-mail-Adresse angegeben, aber die geheime Frage falsch beantwortet.
3. Es wird eine gültige e-mail-Adresse angegeben und die geheime Frage korrekt beantwortet.

Bei der zu testenden Seite handelt es sich um eine *Java Server Page* (JSP), also um eine dynamische Webseite. Die Inhalte der Seite sind, wie bereits in der Einleitung angesprochen, abhängig von Benutzereingaben. Als Entscheidungsgrundlage, ob die Tests erfolgreich sind, dienen die Inhalte der Web-Seite. Konnte z. B. zu einer angegebenen e-mail-Adresse ein W3L-Benutzer gefunden werden, so muss auf der folgenden Web-Seite eine gültige geheime Frage zu finden sein. Konnte zu der angegebenen e-mail-Adresse kein gültiger Benutzer gefunden werden, so darf auf der Folgeseite keine gültige geheime Frage zu finden sein. Wurde die geheime Frage richtig beantwort, so muss auf der Folgeseite der Satz *Ihr neues Passwort und Ihr Pseudonym werden Ihnen umgehend per e-mail zugestellt* angezeigt werden. Wurde die Frage falsch beantwortet, muss der Satz *Sie haben die Frage nicht richtig beantwortet* auf der Web-Seite zu finden sein.

## 5.1 Entwicklung der Testklasse

Die Klasse `SimplePwtVergessenTest.java` soll sämtliche Testfälle enthalten. Aus diesem Grund muss sie von `TestCase` erben, wodurch auch sämtliche `assert`-Methoden vererbt werden. Der folgende Codeausschnitt zeigt den Beginn der Klasse :

```
import junit.framework.*;
import com.meterware.httpunit.*;

public class SimplePwtVergessenTest extends TestCase
{

  private String urlbase  = null;
  private WebConversation conversation = null;
  private WebRequest  request = null;
  private WebResponse response = null;
```

```
public SimplePwtVergessenTest()
{
  this.urlbase = "http://www.swt.ruhr-uni-bochum.de/w3l/jsp/";
  conversation = new WebConversation();
}

public static void main(String args[])
{
  junit.textui.TestRunner.run (suite());
}

public static Test suite()
{
  return new TestSuite(PwtVergessen.class);
}
```

Um die Klasse als Test starten zu können, muss sie zunächst bei einer TestSuite angemeldet werden. Zu diesem Zweck wird die Klasse PwtVergessenTest bei der TestSuite registriert. Dies geschieht in der `main`- Methode. Dort wird der Methode `run(...)` der Klasse TestRunner ein mit unserer Klasse initialisiertes Objekt der Klasse TestSuite übergeben. Die Erzeugung des TestSuite-Objektes erfolgt in der Methode suite(). Auf die Methoden `setUp()` und `tearDown()` wird in diesem Beispiel verzichtet, da keine Notwendigkeit für besondere Initialisierungen besteht.

## 5.2 Hilfsmethoden

Um die eigentlichen Testmethoden übersichtlich zu halten, wird bestimmte Funktionalität in andere Methoden ausgelagert. Die folgenden Methoden sind davon betroffen.

```
/*
 * Diese Methode liefert eine Response-Objekt fuer die Passwort-Vergessen-Web-Seite.
 */
private WebResponse getPwtVergessenResponse() throws Exception
{
    //
    // die w3l-Startseite aufrufen...
    //
    request = new GetMethodWebRequest( urlbase+"startportal.jsp" );
    request.setParameter("principal","W3L");
    response = conversation.getResponse(request);
    //
    // Das Login-Frame aufrufen...
    //
    WebResponse flogin = conversation.getFrameContents("login_startportal");
    WebForm loginForm = flogin.getForms()[0];
    request = loginForm.getRequest();
    //
    // einen ungueltigen Zugang eingeben...
    //
    request.setParameter("user","123456789");
    request.setParameter("password","");
    response = conversation.getResponse( request );
    //
    // Die Seite Passwort / Pseudonym vergessen aufrufen...
    //
    WebLink link = response.getLinkWith("Zugang vergessen ?");
    assertNotNull(link); // der Verweis muss vorhanden sein...
    request = new GetMethodWebRequest(urlbase+link.getURLString());
    response = conversation.getResponse(request);
    String title = response.getTitle();
    //
    // Anhand des Titels sicherstellen, dass die richtige Seite aufgerufen wurde...
    //
    assertTrue( "richtiger Titel", title.indexOf( "Passwort" ) != -1 );

    return response;
}

/*
 * Diese Methode gibt die angegebene e-mail-Adresse fuer das entprechende Eingabefeld an
 * und schickt das Formular ab. Es wird ein Response-Objekt fuer die Folge-Seite zurueckgegeben.
 */
private WebResponse submitMailForm(String mail) throws Exception
{
   WebForm mailForm = response.getForms()[0];
   request = mailForm.getRequest();
   request.setParameter("mail",mail);
   response = conversation.getResponse(request);
```

```
    return response;
}

/*
 * Diese Methode gibt die angegebene Antwort fuer das entprechende Eingabefeld an
 * und schickt das Formular ab. Es wird ein Response-Objekt fuer die Folge-Seite zurueckgegeben.
 */
private WebResponse submitAnswerForm(String answer) throws Exception
{
    WebForm answerForm = response.getForms()[0];
    request = answerForm.getRequest();
    request.setParameter("antwortGeheimeFrage",answer);
    response = conversation.getResponse(request);

    return response;
}
```

Die Methode getPwtVergessenResponse() ruft zunächst die W3L-Startseite auf. Im Anschluss wird das Login-Frame (`conversation. getFrameContents(...)`) und das darin enthaltene Formular (`flogin. getForms()[0]`) ermittelt. Danach wird mit `setParameter(...)` ein Pseudonym (`user`) und ein Passwort (`password`) angegeben. Nachdem das Formular mit den „falschen” Eingaben abgeschickt wurde, wird aus der dazu gehörigen Antwort der *Zugang vergessen ?* - Link extrahiert (`response. getLinkWith(...)`). Die verwiesene Seite wird ebenfalls aufgerufen. Ausserdem wird ein Response-Objekt, das diese Seite repräsentiert, zurückgegeben.

Die Methode submitMailForm() gibt in das Formular der *Passwort / Pseudonym - Vergessen*-Seite die ihr übergebene e-mail-Adresse ein und schickt das Formular ab. Das die Antwort repräsentierende Response-Objekt wird zurückgegeben. Die Methode submitAnswerForm() erledigt das Gleiche mit der Antwort auf die geheime Frage.

## 5.3 Testmethoden

Die folgenden Methoden führen die eigentlichen Tests aus:

```
/*
 * Testet die Web-Seite unter Angabe einer gueltigen e-mail-Adresse und
 * einer gueltigen Antwort auf die geheime Frage
 */
public void testValidMailValidAnswer () throws Exception
{
  response = getPwtVergessenResponse();
  //
  // Die email-Adresse angeben
  //
  response = submitMailForm("testuser@swt.ruhr-uni-bochum.de");
  //
  // Ueberpruefen, ob eine gueltige geheime Frage gestellt wird.
  //
  assertTrue("geheime Frage muss gestellt werden !", hasValidQuestion(response.getText()));
  //
  // Die Antwort auf die geheime Frage eingeben...
  //
  response = submitAnswerForm("blau");
  //
  // Die Reaktion abwarten...
  //
  assertTrue(">>e-mail wird zugestellt<< muss angezeigt werden !", response.getText()
    .indexOf("Ihr neues Passwort und Ihr Pseudonym werden Ihnen umgehend per e-mail zugestellt.") != -1);
}

/*
 * Testet die Web-Seite unter Angabe einer gueltigen e-mail-Adresse und
 * einer ungueltigen Antwort auf die geheime Frage.
 */
public void testValidMailInvalidAnswer() throws Exception
{
  response = getPwtVergessenResponse();
  //
```

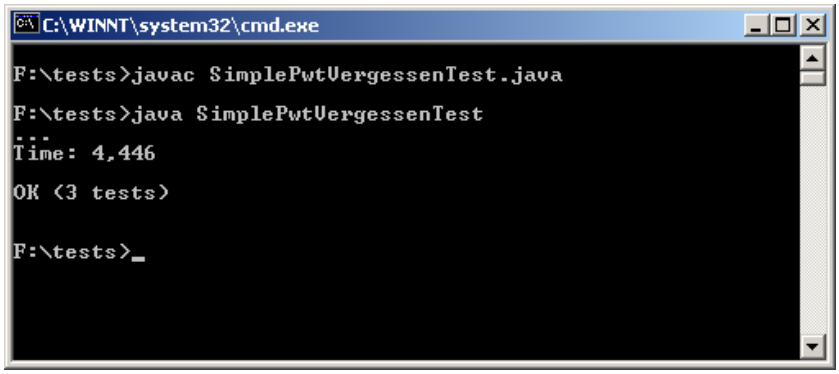
```
      // Die email-Adresse angeben
      //
      response = submitMailForm("testuser@swt.ruhr-uni-bochum.de");
      assertTrue("geheime Frage muss gestellt werden !", hasValidQuestion(response.getText()));
      //
      // Die Antwort eingeben...
      //
      response = submitAnswerForm("falsche Antwort");
      //
      // Die Reaktion abwarten...
      //
      assertTrue("",
         response.getText().indexOf("Sie haben die Frage nicht richtig beantwortet") != -1);
    }

    /*
     * Testet die Web-Seite unter Angabe einer ungueltigen e-mail-Adresse
     */
    public void testInvailidMail() throws Exception
    {
      response = getPwtVergessenResponse();
      //
      // Die e-mail-Adresse angeben...
      //
      response = submitMailForm("schwach@sinn.de");
      //
      // Jetzt darf keine gueltige geheime Frage angezeigt werden...
      //
      assertFalse(hasValidQuestion(response.getText()));
    }
}
```

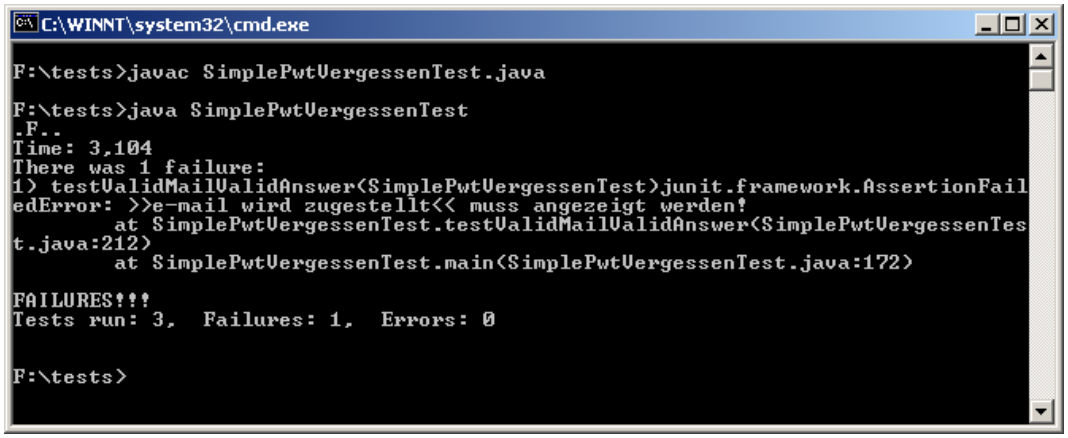
Es sei an dieser Stelle darauf hingewiesen, dass die Methode `hasValidQuestion()` lediglich die Aufgabe hat zu überprüfen, ob die ihr übergebene Zeichenkette eine gültige geheime Frage enthält. Da die Implementierung trivial ist, wurde die Methode aus Gründen der Übersichtlichkeit an dieser Stelle nicht mit aufgeführt. Die Testmethoden üebrprüfen unter Zuhilfenahme der `assert`-Methoden, ob die zu erfüllenden Bedingungen eingehalten werden.

## 5.4 Testausgabe

Zur Ausführung der Tests wird auf die Klasse `junit.textui.TestRunner` zurückgegriffen. Vor der Ausführung muss zunächst eine Kommandozeile aufgerufen werden. Die Tests können nun kompilert und ausgeführt werden.




Für jede erfolgreich ausgeführte Testmethode gibt JUnit einen Punkt aus. Falls eine `assert`-Methode fehlschlägt, wird ein **F** für *failed* ausgegeben, gefolgt von der Ausgabe der dazu gehörigen `AssertionFaildException` mit einem entsprechenden Fehlerprotokoll.




Falls es während der Testausführung zu einem Fehler kommt, also falls innerhalb der Testmethode eine Exception ausgelösst wird und diese nicht abgefangen wird, wird **E** für *error* ausgegeben.

## 5.5 Ergänzungen

Zu dem oben beschriebenen Tests sollen noch zwei Sachverhalte erwähnt werden :

1. Bei den durchgeführten Tests wurde vorrausgesetzt, dass der den Tests zu Grunde liegende Testbenutzer bereits im System vorhanden ist. Dies muss aber nicht zwingend der Fall sein. Darum ist es ratsam vor Ausführung eines Tests zu überprüfen, ob der Testbenutzer bereits im System existiert. Ist dies nicht der Fall, so muss er zuerst angelegt werden. Anschliessend können dann die Tests ausgeführt werden.

2. Handelt es sich bei der zu testenden Web-Seite um eine JSP und benutzt diese JSP ein *JavaBean*, so kann die JavaBean auch unabhänngig von der JSP mit JUnit (ohne HttpUnit) getestet werden.

# 6 Fazit

Das Testen von dynamischen HTML-Seiten wird durch HttpUnit gut unterstützt. Leider wird JavaScript nur in geringem Maße unterstützt. Browserspezifische JavaScript-Anweisungen werden gar nicht unterstützt. Da JavaScript bei vielen Web-Anwendungen eingesetzt wird, ist die Kombination von JUnit und HttpUnit zum Testen von Web-Anwendungen nur eingeschränkt

tauglich.

JUnit ist ein kleines, mächtiges Java-Framework zum Schreiben und Ausführen automatischer Unit Tests, das mittlerweile weit verbreitet ist und in zahlreichen Projekten eingesetzt wird. Im Bereich Web-Testen kann es, wie im vorangegangenen Abschnitt beschrieben, eingesetzt werden, um JavaBeans zu testen. In der Praxis werden oft Testwerkzeuge wie z. B. JUnit und HttpUnit erst dann angeschafft, wenn das Projekt bereits in Schieflage geraten ist. Dann ist es aber oft bereits zu spät. Es ist empfehlenswert, sich rechtszeitig mit solchen Testumgebungen vertraut zu machen.

# Literatur

[1] Projekthomepage HttpUnit
http://www.httpunit.org

[2] Projekthomepage JUnit
http://www.junit.org

[3] JUnit Cookbook
http://junit.sourceforge.net/doc/cookbook/cookbook.htm

[4] Artikel JUnitTest Infected - Programmers Love Writing Tests
http://junit.sourceforge.net/doc/testinfected/testing.htm

[5] Artikel Unit Testing mit JUnit
http://www.frankwestphal.de/UnitTestingmitJUnit.html

[6] JUnitverschickt.doc
http://home.carolina.rr.com/mitrovic/junit/JUnitverschickt.doc

[7] Projektarbeitarbeit Automatisiertes Erstellen und Testen von Web-Anwendungen von Peter Kesch / FH Dortmund 2002