Softwaretests in Visual Studio 2010 Ultimate

Vergleich mit Java-Testwerkzeugen

Alexander Schunk
Marcel Teuber
Henry Trobisch

Inhalt

1. Vergleich der Unit-Tests

Bei der Erstellung von Unit-Tests sind die Unterschiede zwischen JUnit 4 und den in Visual Studio Ultimate bereitgestellten Unit-Tests eher gering. So sieht beispielsweise ein Test in JUnit4 wie folgt aus:

```
@Test
public void testKonstruktor()
{
	//auszufuehrender Code
}
```

In C# wird ein solcher Test mit folgender Syntax beschrieben:

```
[TestMethod]
public void testKonstruktor()
{
	//auszufuehrender Code
}
```

Abgesehen von der anderen Syntax ist es bei Visual Studio möglich, den vor und nach jedem Test auszuführenden Code direkt dem gewünschten Test zuzuordnen. Zwar ist es auch in JUnit 4 möglich, mittels der Annotation „@before“ verschiedene vorher auszuführende Codes zu erstellen, eine bestimmte Ausführungsreihenfolge wird aber nicht zugesagt.

Bei Visual Studio entsteht mit dem zwingend erforderlichem Testprojekt und den dazugehörigen Testlisten ein Mehraufwand, der allerdings bei wachsendem Programmumfang eine sehr gute Strukturierung der eigenen Tests ermöglicht.

2. Vergleich der Codeabdeckungs-Tests

In Visual Studio 2010 Ultimate ist die Möglichkeit, Abdeckungstests durchzuführen, zwar verfügbar, allerdings beim Start eines neuen Projektes erst einmal deaktiviert. Diese Option muss auf recht umständlichen Weg erst aktiviert werden. Sobald die Codeabdeckung aktiviert ist, wird diese dann auch bei dem Ausführen der erstellten Tests gleich mit ausgeführt.

Um Codeabdeckungstests für Java auszuführen, ist es, je nach Entwicklungsumgebung, nötig, die dafür benötigte Software zu installieren. Zusätzlich dazu muss die Option, Abdeckungstests durchzuführen, auch aktiviert werden, diese ist aber einfach zu finden. Die Abdeckungstests sind hier unabhängig von den restlichen Tests und müssen separat gestartet werden.

3. Vergleich der GUI-Tests

In Visual Studio 2010 sind die GUI Testmöglichkeiten bereits vorinstalliert und können so genutzt werden. Für das IDE „Eclipse“ muss man erst FEST oder Marathon nachinstallieren bzw. die jar-Datei einbinden. Das Aufzeichnen in Visual Studio 2010 erfolgt ähnlich wie in Marathon von Eclipse. Hierzu wird eine Aufzeichnung gestartet, anschließend startet man die GUI und gibt seine Werte in das Formular ein und führt Operationen mit den Befehlsschaltflächen aus. Zum Abschluss wird die Aufzeichnung beenden.

Die Möglichkeit GUI-Tests selber zu schreiben ist in Visual Studio ein wenig erschwerend. Man kann die Aufzeichnung nehmen und den Code darin verändern jedoch nicht so sauber getrennt wie in Eclipse mit FEST in Java. Wobei es sicherlich auch Software zum nachinstallieren für Visual Studio gibt die eine ähnliche Funktion von FEST in Visual Studio ermöglicht.

4. Vergleich der Performance-Tests

Visual Studio 2010 Ultimate bietet für Codeanalyse bereits integrierte Mittel an. Im Vergleich zu dem im Praktikum getesteten Netbeans-Profiler fällt auf, dass in Visual Studio mehr Testmöglichkeiten auf Systembelastung möglich sind.

Beide Analysetools sind sehr einfach zu bedienen und bereiten das Ergebnis visuell und textuell noch einmal auf. Visual Studio bietet hier die etwas bessere Übersicht, gerade im Hinblick auf den langsamen Teil des getesteten Programms, welcher direkt in der Ergebnisübersicht aufgezeigt wird.

5. Vergleich der Möglichkeiten des Mocking

In Visual Studio 2010 gibt es keine automatische Möglichkeit des Mockens. In Eclipse muss man die jmock.jar den genutzten Bibliotheken hinzufügen, um richtige Mocks zu erstellen. Für Visual Studio kann man hier „NUnit“ installieren und die nunit_framework.dll sowie die nunit_mocks.dll einbinden. Die Syntax ist weites gehend identisch in beiden Sprachen, leichte Unterschiede in den Annotationen sowie in den Aufrufenden Klassen sind möglich.

C#	Java
`private DynamicMock ListensucherMock;` `= new DynamicMock(typeof(IListensucher));`	`private Mockery ListensucherMock =` `new Mockery();`
`test.sucherobj =` `(IListensucher)ListensucherMock.MockInstance;`	`test.sucherobj =` `context.mock(IListensucher.class);`

6. Weitere Testarten in Visual Studio

Neben den bereits genannten Möglichkeiten für beide Programmiersprachen bietet Visual Studio 2010 Ultimate Auslastungstests an. Diese sind sehr umfangreich und verlangen eine gewisse Einarbeitung da nicht immer alle Funktionen und Begriffe selbsterklärend sind. Da der Fokus der Auslastungstests auf Multiuseranwendungen und verteilte Anwendungen liegt, ist der Erwerb von Zusatzsoftware und Lizenzen unumgänglich, da mit der Standardinstallation von Visual Studio nur eine begrenzte Anzahl an Zugriffen durch mehrere Benutzer simuliert werden kann. Daher lohnt sich diese Testmöglichkeit für kleinere und lokale Anwendungen nur bedingt.

7. Fazit

Die Möglichkeiten, Software zu testen, sind sowohl für Java als auch C# in Visual Studio sehr umfangreich, auch wenn teilweise die Installation von Zusatzsoftware nötig ist. Visual Studio ist merklich auf den professionellen Bereich ausgelegt, was an dem Umfang der Testmöglichkeiten und deren Verwaltungsaufwand zu sehen ist. Große Unterschiede zwischen den Programmiersprachen allgemein sind kaum festzustellen. Wenn das Testen mit Mocks nicht unbedingt notwendig ist, ist eine Installation von Zusatzsoftware nicht notwendig. Anzumerken ist, dass die besprochenen integrierten Testmöglichkeiten in Visual Studio lediglich in der Premium- als auch in der Ultimate-Version zur Verfügung stehen, die Auslastungstests beschränken sich sogar nur auf die Ultimate-Version. Da dies unser erster Kontakt mit der Sprache C# und der Entwicklungsumgebung Visual Studio, war dies ein sehr interessanter Einblick.

8. Arbeitsaufteilung

Unit Tests in Visual Studio:	Alexander Schunk , Henry Trobisch
Unit Tests mit NUnit:	Marcel Teuber, Alexander Schunk
Mocking:	Marcel Teuber
Codeabdeckung :	Alexander Schunk
Oberflächentests:	Marcel Teuber
Leistungsanalyse:	Henry Trobisch
Auslastungstests:	Henry Trobisch, Alexander Schunk
Dokumentation und HowTo:	Alexander Schunk, Marcel Teuber, Henry Trobisch