

Unit-Tests mit OpenCTF

Michael Justin

Dieser Artikel gibt einen ersten Überblick zum Open Source Komponenten-Testframework OpenCTF.

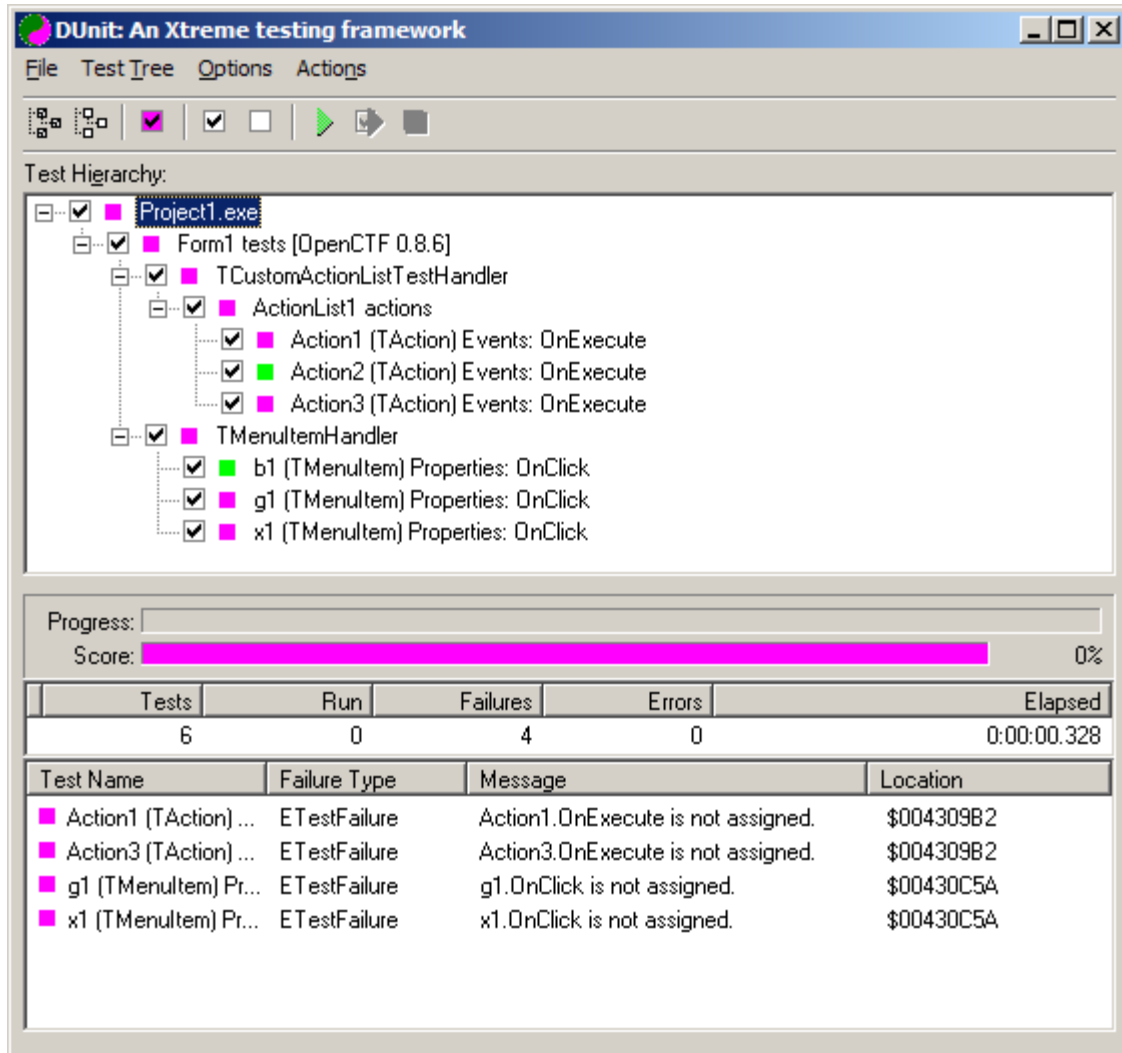


Abbildung 1: OpenCTF Testsuite im DUnit GUI TestRunner

Installation

Nach der Installation von OpenCTF befinden sich die Units für das Framework und die standardmäßig mitgelieferten Tests im Verzeichnis <Installationspfad>\source bzw. source\tests. Diese Verzeichnisse müssen im Suchpfad für das Testprojekt liegen, sowie das Verzeichnis für das DUnit-Framework.

Beispiel der Verzeichnisstruktur:

```
\Delphi
  \OpenCTF-0.8
    \doc
    \examples
    \source
      \extensions
      \tests
```

Mitgelieferte Tests

OpenCTF enthält (im Verzeichnis source\test) bereits eine kleine Auswahl Tests für verschiedene VCL-Standardkomponenten. Die Namen der Units für diese Tests beginnen mit 'ctfTest'. Zum Beispiel enthält die Unit ctfTestActnList einige Tests für Komponenten, die in der Unit ActnList deklariert sind.

Wie arbeitet OpenCTF?

OpenCTF kennt nach der Initialisierung im Prinzip nur zwei einfache Dinge (zwei Objektlisten):

- die Liste der zu untersuchenden Formulare
- die Liste der durchzuführenden Tests

Beide Listen werden in der Projektdatei festgelegt und sind daher dort einfach nachvollziehbar und editierbar. Es gibt keine separaten Konfigurationsdateien oder 'unsichtbaren' Abläufe im Hintergrund.

OpenCTF durchläuft nacheinander alle Komponenten der angegebenen Formulare, und prüft dabei, ob für die Klasse der Komponente (oder eine Vorfahrklasse) ein Test hinterlegt ist. Falls die Komponente diesen 'Aufnahmetest' besteht, wird dynamisch ein Testfall für das DUnit-Framework erzeugt und der Testsuite hinzugefügt. (Jeder Testfall enthält auch eine Objektreferenz auf die von ihm zu testende Komponente.)

Nachdem alle Testfälle feststehen, kann mit der DUnit-Standardmethode 'RunRegisteredTests' der eigentliche Testlauf gestartet werden. Da jeder Testfall die von ihm zu testende Komponente bereits 'kennt', ist in dieser Phase keine besondere Leistung des OpenCTF mehr erforderlich.

Angabe der zu testenden Formulare

Die zu testenden Formulare können direkt im Projekt Quelltext angegeben werden. Die Instanzen der zu testenden Formulare werden hierzu als ein Array-Parameter an die Prozedur 'RegisterForms' übergeben.

Beispiel:

```
RegisterForms([DataModule1, Form1, Form2, Form3]);
```

Die Reihenfolge der Objekte spielt dabei keine Rolle, sie legt nur die Testreihenfolge fest.

Wichtig ist nur, dass die Registrierung vor dem Aufruf der Methode RunRegisteredTests erfolgt.

Festlegung der durchzuführenden Tests

Für die Festlegung der durchzuführenden Tests gibt es zwei Möglichkeiten:

- automatische Registrierung
- manuelle Registrierung

Die mitgelieferten Beispieltests verwenden die automatische Registrierung, sie besteht einfach darin, die Units in der uses-Liste aufzunehmen. Alle in der angegebenen Units definierten Tests werden dann für das Testprojekt verwendet. (Technischer Hintergrund: die Tests werden durch Code im initialization-Abschnitt in OpenCTF registriert).

Beispiel für automatische Registrierung der Testklassen aus Unit ctfTestDB:

```
uses
  Forms,
  SysUtils,
  OpenCTF,
  ctfTestDB,
  ...
```

Seltener verwendet wird die manuelle Registrierung, daher wird sie hier nur kurz beschrieben. Eine manuelle Registrierung der Tests im OpenCTF Framework kann erfolgen, wenn gezielt nur einzelne Tests aktiviert werden sollen. Die Registrierung geschieht dann z.B. in der Projektdatei. (Es wird dann kein initialization-Abschnitt verwendet).

Die Standardinstallation von OpenCTF enthält zur Zeit nur Tests mit automatischer Registrierung.

Beispiel einer Delphi-Projektdatei

Die Delphi-Projektdatei für ein OpenCTF-Testprojekt besteht aus den Teilen

- uses-Liste mit den Units für die gewünschten Tests und die zu testenden Formulare/Datenmodule
- Code zum Erzeugen und Freigeben aller zu testenden Formulare/Datenmodule
- Code zur Übergabe der Formulare an OpenCTF
- Code zum Start der Tests

```
program Project1;
{$APPTYPE CONSOLE}
uses
  SysUtils,
  OpenCTF,
  ctfTestActnList,
  ctfTestMenus,
  TestForm in 'TestForm.pas' {Form1},
  TextTestRunner,
  GUITestRunner,
  TestFramework;
begin
  // First, create a form instance:
  Form1 := TForm1.Create(nil);
```

```

try
  // now create and add all tests for this form instance
  OpenCTF.RegisterForm([Form1]);
  // run the tests
  if FindCmdLineSwitch('console') then
    TextTestRunner.RunRegisteredTests(rxbHaltOnFailures)
  else
    GUITestRunner.RunRegisteredTests;
  finally
    // cleanup
    Form1.Free;
  end
end.
end.

```

Ausgabebeispiel für TextTestRunner

Um die Tests im Kommandozeilenmodus mit dem TextTestRunner auszuführen, kann zum Beispiel eine Batchdatei verwendet werden. Sie startet die Testanwendung mit dem Parameter /console aufgerufen und die Ausgabe in die Datei test.log umgeleitet wird:

```

call Project1 /console > test.log
pause

```

Ausgabe (Inhalt der Datei test.log):

```

DUnit / Testing
.....F....F....F.F
Time: 0:00:00.0

FAILURES!!!
Test Results:
Run:          0
Failures:     4
Errors:       0
There were 4 failures:
  1) Action1 (TAction) Events: OnExecute: ETestFailure
    at
      "Action1.OnExecute is not assigned."
  2) Action3 (TAction) Events: OnExecute: ETestFailure
    at
      "Action3.OnExecute is not assigned."
  3) g1 (TMenuItem) Properties: OnClick: ETestFailure
    at
      "g1.OnClick is not assigned."
  4) x1 (TMenuItem) Properties: OnClick: ETestFailure
    at
      "x1.OnClick is not assigned."

```

Anhang

Häufig gestellte Fragen zu OpenCTF

Bei der Ausführung von Tests kommt es zu Fehlermeldungen (z.B. EAccessViolation), weil die FormCreate- oder DataModuleCreate Methode ausgeführt wird und darin Code enthalten ist, der Objekte benutzt die im Unittest nicht instanziiert sind. Was kann ich tun?

Einfache Lösung, wenn es sich um eine GUI-Anwendung handelt, und das Unittest-Projekt als Konsolenanwendung erstellt wurde:

Mit der Funktion System.IsConsole kann man testen, ob es sich bei einer Anwendung um eine Konsolenanwendung handelt. Diese Methode kann man benutzen, falls der Unittest als Konsolenanwendung läuft (siehe Beispiel 1).

Kritischen Code kann man dann mit einem Test absichern:

```
// Im Unittest Funktion hier abbrechen:
if System.IsConsole then
    Exit;
// Sonst: gefährlichen Code ausführen
...
```

Tipps & Tricks

Dieser Abschnitt enthält einige Hilfestellungen rund um OpenCTF und DUnit.

Wie kann ein DUnit-Projekt wahlweise den GUI Testrunner oder den Text Testrunner nutzen?

Oft ist es sinnvoll, einen Unittest in der Delphi IDE über den GUI Testrunner auszuführen, und zugleich eine einfache Möglichkeit zu haben, diese Tests auch z.B. in einer Batchdatei laufen zu lassen.

Durch einen Kommandozeilenparameter wie z.B. **/console** kann einer Anwendung mitgeteilt werden, welcher Modus verwendet werden soll.

Siehe Beispiel 1.

Wie kann ein DUnit-Projekt als Kommandozeilenanwendung einen DOS-Rückgabewert zurückgeben, falls ein Test fehlgeschlagen ist?

Dazu muss der TestTextRunner lediglich mit dem Parameter **rxbHaltOnFailures** aufgerufen werden. Damit wird erreicht, dass nach dem Durchlaufen aller Tests ein Halt-Befehl mit einem Errorlevel größer Null ausgelöst wird.

Siehe Beispiel 1.

Wie kann ich ein Testprojekt mit Apache Ant ausführen?

Siehe Beispiel 2.

Beispiele

Beispiel 1

Dieses Beispiel zeigt ein Delphi Projekt, in dem ein Datenmodul und ein Formular mit OpenCTF Tests analysiert wird.

Das Projekt kann über einen Kommandozeilenparameter wahlweise im GUI- oder Textmodus gestartet werden.

```
program UnitTests;
{$APPTYPE CONSOLE}
uses
  Forms,
  SysUtils,
  OpenCTF,
  ctfTestDB,
  GUITestRunner,
  TextTestRunner,
  d_main in 'd_main.pas' {dm1: TDataModule},
  u_main in 'u_main.pas' {frmMain};
begin
  dm1 := Tdm1.Create(nil);
  frmMain := TfrmMain.Create(nil);
  try
    RegisterForms([dm1, frmMain]);
    if FindCmdLineSwitch('console') then
      TextTestRunner.RunRegisteredTests(rxbHaltOnFailures)
    else
      GUITestRunner.RunRegisteredTests;
  finally
    frmMain.Free;
    dm1.Free;
  end;
end.
```

Beispiel 2

Ausführung eines Dunit-Testprojekts in Apache Ant. Über die Property failonerror wird das Fehlschlagen des Builds erreicht, sobald bei der Ausführung der Tests ein Fehler auftrat. Die Ausgabe der Testprotokolle erfolgt in die Datei unittests.log.

```
<exec executable="${target}/bin/UnitTests.exe"
  dir="${target}/bin"
  failonerror="true"
  output="unittests.log" >
  <arg value="/console"/>
</exec>
```

Es werden aber in jedem Fall alle Tests durchgeführt - auch wenn zwischendurch ein Tests fehlschlägt, wird die Testsuite bis zum Ende abgearbeitet.