Testovanie kódu pomocou Unit Testing nástroja Caché

Koľkí z nás vývojárov môžu na otázku, či testujú svoj kód, odpovedať úprimne: áno? Asi väčšina, ale koľkí z nás môžu kladne odpovedať na otázku: „Ste si naozaj istí, že testujete svoj kód dostatočne?" Pre všetkých, ktorí chcú mať istotu, že svoj kód pred nasadením u zákazníkov dobre otestujú a doladia, je určený tento príspevok.

Jeden z prístupov k efektívnemu písaniu funkčného kódu je programovanie riadené testovaním, v anglickom origináli test driven programming. V skratke to znamená, že programátor predtým, než vôbec začne písať kód, si sadne a napíše všetky možné testy budúceho kódu. Potom napíše kód, aplikuje testy a cyklus opakuje. Asi sa zhodneme na tom, že v praxi je pomerne ťažké dodržať tento prístup, lebo sme všetci netrpezliví, nebudeme sa predsa zdržovať písaním niečoho iného ako cieľového kódu. No keď už testy nepíšeme pre všetky možné situácie a vopred, aspoň tie hlavné je veľmi vhodné vytvoriť, hoci aj súbežne alebo dodatočne ku kódu.

Caché ponúka už nejaký čas súbor tried obsiahnutých v balíčku % UnitTesting. Práca s balíčkom je jednoduchá a je detailne opísaná v on-line dokumentácii, takže tu len stručne s malou ukážkou.

1. Vytvoríme triedy s aplikačným kódom, ktorý neskôr budeme chcieť testovať.

2. Vytvoríme jednu alebo viac testovacích tried oddelením od % UnitTesting.TestCase.

3. Pre každú testovanú funkčnosť definujeme samostatnú metódu začínajúcu sa reťazcom "Test". To je dôležité, pretože iba takto pomenované metódy sa budú následne vykonávať v priebehu testovania.

4. Teraz exportujeme triedy ako XML do vopred určeného testovacieho adresára. Koreňový testovací adresár je daný hodnotou globale ^ UnitTestRoot, ktorú nezabudneme nastaviť. Export tried je dôležitý, pretože východiskovým správaním unit test managera je zmazanie definícií testovacích tried po vykonaní testov.

5. V termináli spustíme súbor testov príkazom d # # class (% UnitTest.Manager). RunTest ([názov adresára])>, kde [názov adresára] označuje buď koreňový adresár (ak je prázdny), alebo podadresár s exportovaným triedami v súboroch XML.

6. V priebehu vykonávania jednotlivých testov sa bude na termináli zobrazovať výpis jednotlivých metód a ich výsledkov, na konci výpisu potom bude uvedený odkaz na stránku CSP s výsledkami.

Pri tvorbe testovacích tried môžeme využiť preddefinované makrá a metódy pochádzajúce z triedy % UnitTest.TestCase. Jednak môžeme nastaviť pomocné premenné a upratať po sebe, jednak pomocou makier vykonávať vyhodnocovanie rôznych druhov výrazov. Všetky makrá s výnimkou logovania sa začínajú reťazcom $ $ $ Assert .... Napríklad makro $ $ $ SAssertStatusOK kontroluje, či návratová hodnota typu % Status vracia hodnotu OK (1) alebo chybový stav. Miesto makier môžeme volať aj metódy začínajúce sa na "Assert", pričom ako prvý argument zadáme hodnotu 1 (autoquoted). A teraz jednoduchý príklad:

```
Class tt.Tip18  [  Abstract ]
 {
  classmethod Secti  (a as %Float, b as %Float)
as %Float
   {
       quit a + b
   }
□ classmethod  Podel  (a as %Float, b as %Float)
as %Float
   {
                try  {
                       set  c=a/b
                }  catch  (ex)   {
                       //   jednoduse eskalujeme
vzjimku volajicimu kodu
                       throw ex
                }
                quit  c
   }
  }
```

Táto jednoduchá trieda implementuje dve metódy - Secti a Podel. My budeme testovať nasledujúce prípady:

- Secti(1,2) = 3
- Secti(1,2) != 4
- Podel(6,2) = 3
- Podel(1,0) vracia chybu DIVIDE

Podľa uvedeného návodu zostavíme testovaciu triedu:

```
   Class tt.unitest.Tip18 Extends
%UnitTest.TestCase
   {
□Method TestSectiOK()
    {
            do  ..AssertEqualsViaMacro
(1,##class(tt.Tip18).Secti(1,2),3,"Secti(1,2)
=3")
    }
□Method TestSectiNOK()
    {
            do  ..AssertEqualsViaMacro
(1,##class(tt.Tip18).Secti(1,2),4,"Secti(1,2)
!=4")
    }
□Method TestTestPodelOK()
    {
            do  ..AssertEqualsViaMacro
(1,##class(tt.Tip18).Podel(6,2),3,"Podel(6,2)
=3")
    }
□Method TestPodelNulou()
  {
            try {
                  set res=##class(tt.Tip18)
.Podel(1,0)
          }  catch  (ex)   {
                  do  ..LogMessage ("Zachycena
ocekavana vyjimka  "_ex.DisplayString  ())
        }
        Quit
  }
}
```

A teraz už môžeme otestovať:

```
SYM>d
##class(%UnitTest.Manager).RunTest("SYM")

=============================================
==================================
Directory: D:\Cache\DataSets\MASTER\Unit Test
Root\SYM\
=============================================
==================================
SYM begins ...
Load of directory started on 01/29/2010
15:38:45 '*.xml;*.XML'


Loading file D:\Cache\DataSets\MASTER\Unit
Test Root\SYM\Tip18.xml as xml
Imported class: tt.unitest.Tip18


Compilation started on 01/29/2010 15:38:45
with qualifiers 'ck'
Compiling class tt.unitest.Tip18
Compiling routine tt.unitest.Tip18.1
Compilation finished successfully in 0.058s.

Load finished successfully.

  tt.unitest.Tip18 begins ...
    TestPodelNulou() begins ...
      LogMessage:Zachycena ocekavana vyjimka
18 zPodel+2^tt.Tip18.3
      LogMessage:Duration of execution:
.000122 sec.
    TestPodelNulou passed
    TestPodelOK() begins ...
      AssertEquals:Podel(6,2)=3 (passed)
      LogMessage:Duration of execution:
.000042 sec.
    TestPodelOK passed
    TestSectiNOK() begins ...
      AssertNotEquals:Secti(1,2)!=4 (passed)
      LogMessage:Duration of execution:
.000046 sec.
    TestSectiNOK passed
    TestSectiOK() begins ...
      AssertEquals:Secti(1,2)=3 (passed)
      LogMessage:Duration of execution:
.000032 sec.
    TestSectiOK passed
  tt.unitest.Tip18 passed
SYM passed

Use the following URL to view the result:
http://KUTAC:57772/csp/samples/%25UnitTest.Re
port.cls?NS=SYM&INDEX=2
SYM>
```

V budúcej časti sa budeme venovať využitiu používateľskej projekcie pre Unit Testing. Ďalšie zaujímavé diely seriálu Tipy a triky s Caché, ktoré vychádzajú už od roku 2005, nájdete na stiahnutie tu: http://www.intersystems.cz/education/university/serialy/tipyTrikyCache.html

DAN KUTÁČ, InterSystems

Prostredníctvom Google Apps sa teraz dostanete k 60 službám

Prihlásením ku Google Apps sa dostanete k desiatkam nových služieb od Googlu. Google Apps je jeden z najlepších produktov, s ktorými Google doteraz prišiel. Mnohým používateľom však prekážalo, že hoci ste mali účet na Google Apps, stále ste sa s ním nemohli pripájať k populárnym službám, ako je Picasa, Blogger alebo Google Voice, nehovoriac o tých menej populárnych, ako napr. Google Base. Ak ste sa prihlásili ku Google Apps, mali ste totiž prístup len ku Gmailu, Google Docs, Google Sites a niekoľkým ďalším službám. To sa zmenilo. Prihlásením ku Google Apps sa teraz dostanete k viac než šiestim desiatkam služieb od Googlu. Prečo to trvalo tak dlho? Podľa oficiálneho vyjadrenia Googlu si spoločnosť chcela byť stopercentne istá, že jej zázemie taký nápor zvládne. Navyše si vraj prístup k Picase, Bloggeru a ďalším službám cez Google Apps ľudia veľmi žiadali. Google podotkol, že jednotlivé spoločnosti využívajúce Google Apps budú môcť prístup k jednotlivým službám pre svojich zamestnancov povoliť alebo zakázať.