The Monad.Reader/Issue5/Software Testing With Haskell

From HaskellWiki

< The Monad.Reader | Issue5

Contents

1 Software Testing With Haskell

by ShaeErisson for The Monad.Reader Issue 5, Date: 2005-10-02T21:05:41Z

The two most commonly used libraries for testing software with Haskell are HUnit and QuickCheck. This article will shortly describe the two different approaches and show some demonstration code.

Both HUnit and QuickCheck are included with GHC 6.4.1 under the module names of Test.HUnit and Test.QuickCheck

1.1 HUnit

HUnit was written by Dean Herington and is available on SourceForge at http://hunit.sourceforge.net/ .

If you've ever used the xUnit (http://en.wikipedia.org/wiki/XUnit) framework in other programming languages, HUnit will feel familiar. The User's Guide (http://hunit.sourceforge.net/HUnit-1.0/Guide.html) includes a 'getting started' started section, and there are a thousand introductions to various flavors of the xUnit framework, so we'll mention HUnit only briefly.

From the user's guide:

> "Tests are specified compositionally. Assertions (http://hunit.sourceforge.net/HUnit-1.0/Guide.html#Assertions) are combined to make a test case (http://hunit.sourceforge.net/HUnit-1.0/Guide.html#TestCase) , and test cases are combined into tests (http://hunit.sourceforge.net/HUnit-

1.0/Guide.html#Tests) ."

Here's a short demo:

```
module ProtoHunit where

import Test.HUnit
import Test.HUnit

testList = TestList -- construct a TestList from a list of type TestCase
 [TestCase $ -- construct a TestCase from an assertion
 assertEqual "description" 2 (1 + 1) -- construct an assertion from a descriptive string, an expected result, an
 ]

t = runTestTT testList
```

1.2 QuickCheck

QuickCheck was written by Koen Claessan and John Hughes, and is available from Chalmers at http://www.cs.chalmers.se/~rjmh/QuickCheck/ .

QuickCheck takes a dramatically different approach to software testing. The programmer specifies a property that the code should follow, and the QuickCheck library generates random values and checks to see if the property always holds.

Some demonstration properties are given below.

```
module ProtoQuickCheck where
import Test.QuickCheck

-- this succeeds in one case of the input.
prop_Fail :: Int -> Bool
prop_Fail x =
 x == 1

-- this succeeds in three cases of the input.
prop_RevUnit :: [Int] -> Bool
prop_RevUnit x =
 reverse x == x

-- what's wrong with this picture?
prop_RevUnitConfusion :: [Int] -> Bool
prop_RevUnitConfusion x =
 reverse [x] == [x]

-- do you see a bug?
prop_RevApp :: [Int] -> [Int] -> Bool
prop_RevApp xs ys =
 reverse (xs ++ ys) == reverse xs ++ reverse ys

prop_RevRev :: [Int] -> Bool
prop_RevRev xs =
 reverse (reverse xs) == xs

(f === g) x = f x == g x

prop_CompAssoc :: (Int -> Int) -> (Int -> Int) -> (Int -> Int) -> Int -> Bool
prop_CompAssoc f g h = (f . (g . h)) === ((f . g) . h)

prop_CompCommut :: (Int -> Int) -> (Int -> Int) -> Int -> Bool
prop_CompCommut f g = (f . g) === (g . f)
```

```
-- this operator ==>
-- means filter inputs by that condition
-- below an x and y are only accepted if x is less than or equal to y
prop_MaxLe :: Int -> Int -> Property
prop_MaxLe x y = x <= y ==> max x y == y

instance Show (a -> b) where show _ = "<<function>>"
```

To test one of these properties, load the source into ghci and run `quickCheck prop_Fail`. One possible response is:

```
Falsifiable, after 0 tests:
-1
```

Since the type signature of prop_Fail is

```
Int -> Bool
```

QuickCheck generated an Int value and checked to see if the property held true. Since the value -1 is not equal to 1, the property is false.

1.3 Everything Else

The Haskell wiki has information on one button unit testing with the emacs haskell-mode.

Other libraries and applications that deal with software testing in Haskell are mentioned below, but are beyond the scope of this short introduction.

- Hat The Haskell Tracer.
- Plargeflarp (http://www.cs.mu.oz.au/~bjpop/plargleflarp/) (declarative debugger sometimes known as Buddha (http://www.cs.mu.oz.au/~bjpop/buddha/))
- Programatica (http://www.cse.ogi.edu/~hallgren/Programatica/) is a collection of tools, that include the ability to specify inline 'certificates'. 'Certificates' are tests, they can be static unit tests, QuickCheck properties, or automated proofs.

Retrieved from "http://www.haskell.org/haskellwiki/The_Monad.Reader/Issue5/Software_Testing_With_Haskell"

Category: Article
