Unit Testing in Ruby

4010-350
Personal Software Engineering

Unit Testing Review

- Test a cohesive functional entity:
  - Class
  - Stand alone function or functions
- Verification testing – does the entity do what it's supposed to do.
- Greatly facilitated by unit test frameworks.
  - JUnit for Java
  - NUnit for .NET
  - Test::Unit for Ruby

Unit Testing in Ruby

- Test::Unit::TestCase
  - All unit test classes inherit from this class
  - Example: class MyClass < Test::Unit::TestCase
  - setup / teardown
  - test* methods (run in dictionary order)
- Assertions
  - assert(*boolean*, [*message*])
  - assert_[not_]equal(*exp*, *act*, [*message*])
  - assert_raise(*Exception*) *block*
  - assert_nothing_raised([*Exception*]) *block*
  - assert_[not_]nil(*obj*, [*message*])
  - Full list in *http://www.ruby-doc.org/stdlib*, test/unit, Test::Unit::Assertions

Queue: (queue.rb)

```
class Queue

  # Exception class for taking values from an empty queue.
  class Empty < StandardError
    def initialize
      super("Empty queue")
    end
  end

  # Initialization
  def initialize
    @contents = Array.new
    self
  end

  # Queue is empty if its size is zero
  def empty?
    size = 0
  end

  # Queue size - number of elements
  def size
    @contents.size
  end
```

Queue: (queue.rb)

```
  # Add a value to the tail of the queue
  def tail= value
    @contents[@contents.size] =  value
    value
  end

  # Return the first element in the queue without removing it
  def peek
    raise Empty if empty?
    @contents[0]
  end

  # Return and remove the first queue element
  def head
    value = peek
    @contents.delete_at(0)
    value
  end
end
```

TestQueue: (test_queue.rb)

```
require 'test/unit'
require 'queue'

class TestQueue < Test::Unit::TestCase
  def setup
    @tq = Queue.new
  end

  # Check proper empty queue behavior
  def test_001_new_queue
    assert( @tq.size == 0, "New queue size not zero" )
    assert( @tq.empty?, "New queue not empty" )
    assert_raise(Queue::Empty) { @tq.peek }
    assert_raise(Queue::Empty) { value = @tq.head }
  end
```

TestQueue: (test_queue.rb)

```
  # Check proper FIFO behavior.
  def test_002_fifo_check
    test_values = %w{ A B C }
    test_values.each { |v| @tq.tail = v }

    size = @tq.size
    tvlen = test_values.length
    assert( size == tvlen,
              "#{tvlen} element queue gives size of #{size}" )
    assert( ! @tq.empty?, "Non-empty queue reports empty" )

    test_values.each do |v|
      assert_nothing_raised() { qv = @tq.peek }
      assert_equal(v, qv, '@tq.peek:')

      assert_nothing_raised() { qv = @tq.head }
      assert_equal(v, qv, '@tq.head:')
    end

    assert_raise(Queue::Empty) { @tq.peek }
    assert_raise(Queue::Empty) { qv = @tq.head }
  end
end
```