Behaviour Driven Development

Following excerpt is from http://behaviour-driven.org/Introduction

“Experienced practitioners, particularly those that have been involved in helping other developers learn the practice, note a life-cycle to TDD's learning and adoption:

1. The developer starts writing unit tests around their code using a test framework like JUnit or NUnit, etc.
2. As the body of tests increase the developer begins to enjoy a strongly increased sense of confidence in their work.
3. At some point the developer has the insight (or are shown) that writing the tests before writing the code, helps them focus on only writing the code that they need.
4. The developer also notices that when they return to some code that they haven’t seen for a while, the tests serve to document how the code works.
5. A point of revelation occurs when the developer realises that writing tests in this way helps them to “discover” the API to their code. TDD has now become a design process.
6. Expertise in TDD begins to dawn at the point where the developer realizes that TDD is not about testing, it is about defining behaviour.
7. Behaviour is about the interactions between components of the system and so the use of mocking is a fundamental to advanced TDD.

We have observed this progression in many developers, but unfortunately while most, with a little help, find their way to step 4, many miss the big wins found at steps 5, 6 and 7.

We started wondering why this was, and wondering what we could do to help people get to the good stuff more quickly. Looking closely at this evolution in understanding, while step 1 may be considered to be related to testing the rest really are not. For the remainder of the steps in this process, the test aspect is very much a secondary concern. The issue at the heart of this learning process is the behaviour of the system. Developers gain confidence in the systems they build when the behaviour of that system is confirmed.

By writing tests in advance of code, the developer defines the behavioural intent of the system that they are developing. Tests that document the behavioural intent of the system are most valuable as documentation of that system. By specifying the behaviour of the code in tests, developers are able to best consider the interface from the perspective of the consumer of the service rather than as the producer – thinking more abstractly about the nature of the solution and so hiding technical detail.

This focus on behaviour ahead of testing is a focus on the real value in TDD, and in our experience is the mark of the genuinely experienced TDD practitioner.”

Marcus Crafter <crafterm@redartisan.com>

Installing and using RSpec with Rails

These instructions relate to the installation and usage of the current release of RSpec and RSpec on Rails 1.0.8:

... create or enter Rails application work area (following commands are all on one line)...

... first install rspec itself:

```
$> script/plugin install
svn://rubyforge.org/var/svn/rspec/tags/CURRENT/rspec
```

... then the rspec_on_rails plugin:

```
$> script/plugin install
svn://rubyforge.org/var/svn/rspec/tags/CURRENT/rspec_on_rails
```

... (or use piston, svn externals, or similar)...

... seed your application with necessary rspec helper files (one time operation) ...

```
$> script/generate rspec
```

... and then you can generate models, controllers, or resources using the respective rspec_* generators ...

```
$> script/generate rspec_scaffold user
```

To run a particular spec you can use the Rake task or the command line:

```
$> rake spec
```

Will run all specs in your application. You can also specifically target models, controllers and views using the rake spec:models, spec:controllers, spec:views targets respectively.

```
$> spec spec/model/user_spec.rb
```

Will run an individual test. You can choose different formats using the -f parameter.

```
$> spec -f s spec/model/user_spec.rb
```

Will run an individual test and report the results in specdoc format. rdoc and html are also supported.

In TextMate ⌘D will run all specs in the currently opened file.

See http://rspec.rubyforge.org/ for more information, and the rspec users & developers mailing list.

Marcus Crafter <crafterm@redartisan.com>