Unit Testing T-SQL code with tSQLt

How do I unit test in my day to day work?

Dave Green
Twitter: @d_a_green
Email: dave@dgta.co.uk

pluralsight
hardcore developer training

How to approach writing tests

“________________________?”

How to approach writing tests

Think about the question that needs an answer - before you code

Can come from :

- Formal requirement
- Interface with code
- Customer
- Other design decision

How to approach writing tests

Ask simple yes/no questions.

**If you can’t ask it as a simple question,
it may not be simple enough to test.**

Setting up a scenario

Types of data:

- Typical (Normal)
- Extreme (Edge Case)
- Erroneous (Incorrect)

We want relevant, realistic test scenarios

The minimum possible quantity of data to allow our test to work properly

Testing tests

- Tests are part of your database code
- A test failure indicates that EITHER the TEST or THE CODE under test are not performing as expected.
- Review tests as you would other code
- We don’t unit test our unit tests

  - so don’t let them into the UAT / PRODUCTION environments

Do I need to test everything?

- **Testing code will almost always take more time than writing it**

- **Some trivial methods don’t pay back the investment in testing – consider peer review instead for these.**

- **Generally you will want to at least put some basic testing in place**
  - Places where code interfaces
  - To cover specified requirements of the system.

What if my test fails?

Write / fix the code – or test – promptly

Not doing so means the framework is not allowed to do its job
- to give confidence in the code.

What if my test fails?

- Writing tests takes your time away from coding
- If you lose faith in the unit tests, they will be less effective
- They can become neglected and drift away from current functionality
- Bad unit tests are worse than no unit tests

Do you work alone?

Managers

Successors

Developers

Testers

Clients

Working with others means:

Tests should be findable

Naming Conventions

Simply a way of standardising what we call things

Easy to see what others are doing

How do we decide on a convention?

- **Best done at the start of development**
- **If possible, share a standard with other teams**
  - Company-wide standard
- **Limit of 128 character length**
- **Human readable test names**

How do we decide on a convention?

Unanimously

Where do we need one?

- **Test Class Naming**
  - We have used name of object as our test class in this course
  - Easy to find tests on an object
  - Could be types of test

DEMO

- **Current test classes in the database**

Test Naming

- Helps to communicate the purpose of the test
- Easier to find tests on particular subject
- Helps bug fixing
- Assess test coverage

Example

- Demo of list of current tests.
- Rename test in SQL Test (Note, can rename in object explorer for tSQLt)

Working with others means:

Tests should be understandable

Understandable tests

- **Documentation**
  - Why, not just what
- **Where does this test tie back to requirements?**
  - Naming Convention?
- **Standards**
  - Create example and actual tables in the schema

Understandable tests

- Meaningful test failure messages
- Use Setup procedures
- Consistency makes for simple, repeatable, understandable tests

Working with others means:

Tests should be shareable

Part-baked Tests

- A test should not pass when the criteria hasn’t been met.
- First instruction should be a call to fail.
- Then remove it when you have coded your test.
- That way a shared development system doesn’t create a misleading impression.

Working with others means:

Tests should be part of the development system

Source Control

- **tSQLt objects are normal database objects**
  - Some have Extended properties
  - CLR
- **Can be source controlled like other DB objects**
- **Source controlling objects can tell you what has changed**
  - Helps bug fix tests that now fail.

Where do unit tests belong?

- **DEVELOPMENT system**
  - May be shared?
- **Source Control**
- **Possibly Test system (Not UAT)**
- **Continuous Integration**
  - Run all tests each code change

Removing tests from the database

- **Two distinct components – tSQLt and the unit Tests**

- **Remove tests with tSQLt.DropClass**
    - Loop through each class and drop them one by one

- **Remove Unit test framework with tSQLt.Uninstall**

Demo

- **Removing test classes from the database**
  - Not reversible! Have a backup handy..

- **Removing tSQLt using Uninstall**

When should I run unit tests?

- **During development of code**
  - Agile or Waterfall models
- **When committing code**
  - New developments and bug fixes
  - To check no existing functionality has been broken
- **Continuous Integration server**
  - Can be used to reject code, or prioritise a fix
  - Can give confidence to development team and management

Summary

- **Think about the Question**
- **Adopt development standards**
  - Such as Naming Conventions
- **Tests need to be in a good state to share with others**
- **We can remove the unit tests and framework from the database**