Unit Testing T-SQL code with tSQLt

The Lay of the Land

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

pluralsight
hardcore developer training

What is a unit test?

- **Test on a discrete unit of code**
  - Does not test how it interacts with other code units
- **Should not be affected by other units**
- **Isolates the unit under test from the rest of the code.**
- **Repeatable**
  - Ability to automate

What is a unit test?

- **Tests one question**
  - A series of discrete tests will be done on each unit
- **Should reflect a requirement for the unit**
- **Forms only part of the gamut of testing**
  - You still need to test overall function, speed, etc.
- **Usually written by the development team**

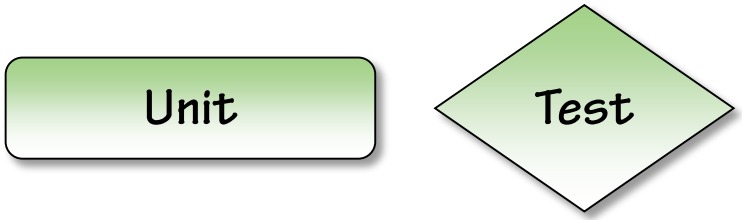
What does a unit test give me?

- **Certainty**
- **Requirements documentation**
  - Confirmation that all requirements are tested and working
- **Ability to code modules out of order**
  - Because interfaces / interactions are defined
- **Simpler error checking**

Finding Bugs

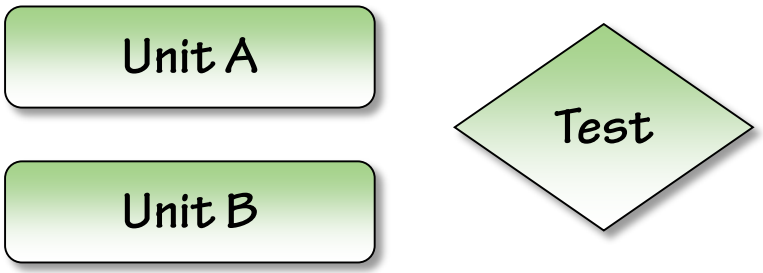
Single Unit

- Error in the unit
- Error in the test



Multiple Units (A and B)

- Error in unit A
- Error in unit B
- Error in Both Units
- Error in the way units A and B interact together
- Many more combinations if you have more than 2 units!
- Error in the test



When do I write unit tests?

- **After development?**
  - Retrofitting tests to existing code
  - Not ideal for new developments
- **Before development?**
  - Allows you to think out code structure first
  - Focuses development on requirements
  - Part of Extreme Programming and other Agile methodologies
- **During development?**
  - Tests need to be revised in an agile environment
  - Requirements can change requiring new tests
- **Answer - All three!**

Introduction to tSQLt

- **SQL Based**
  - Tests are stored procedures
- **Runs entirely in the database layer**
- **Tests run within a framework**
- **Accessible in existing Development tools**

Introduction to tSQLt

- Group Tests into a Class
- Set Up
- Mock objects to remove dependencies
- http://tsqlt.org

Red Gate SQL Test

- **Nice interface to running tSQLt**
  - Integrated into SQL Server Management Studio (SSMS)
  - Easier integration means better adoption
  - Fewer guesses about tests that are available
- **SQL Test is NOT required to use tSQLt**
  - But it does make life easier!
- **http://www.red-gate.com/products/sql-development/sql-test/**

Anatomy of a test

- **Setup**
  - Isolate dependencies
  - Insert test data
  - Specify expected result
- **Run object under test**
  - Call Stored procedure
- **Assess result**
  - Does the actual result match the expected result?

Running a test

- **Start Transaction**
- **Run Setup Procedure (if any)**
- **Run Test**
  - Setup
  - Run object under test
  - Assess result
- **Rollback transaction (keeping results of test)**

Running all tests in a test class

- **Look up the tests in the class, and for each test in the class:**
  - Start Transaction
  - Run Setup Procedure (if any)
  - Run Test
    - Setup
    - Run object under test
    - Assess result
  - Rollback transaction (keeping results of test)

Running all tests in a database

- **Look up all test classes (schemas marked as test classes) in the database**
- **For each test class:**
  - Look up the tests in the class, and for each test in the class:
    - Start Transaction
    - Run Setup Procedure (if any)
    - Run Test
      - Setup
      - Run object under test
      - Assess result
    - Rollback transaction (keeping results of test)

Summary

- **Unit tests form part of our testing armoury**
- **Unit tests require us to**
  - Think about code as discrete units
  - Think about what we are trying to achieve
- **And**
  - Free us from guesswork about what we are coding
  - Allow us to be lazy - only code what is required
  - Minimise overhead of maintenance

- **tSQLt - a framework for database unit tests**

- **SQL Test – a way of integrating tSQLt into SSMS**