Database development unit test with tSQLt

Sergio Govoni

Slide and demos: https://bit.ly/3xwlyse

Sergio Govoni

centro software

DelphiDay italian conference

ONLINE EDITION

wintech italia

linkedin.com/in/sgovoni

github.com/segovoni

twitter.com/segovoni

DelphiDay
italian conference
ONLINE EDITION

wintech
italia

AGENDA

- ➔ Introduction to Unit Testing
- ➔ Introduction to tSQLt framework
- ➔ The anatomy of a unit test
- ➔ Our first unit test will be for a Trigger

Types of database unit tests

- ➔ Structural Testing
- ➔ Functional Testing
- ➔ Non-functional Testing

What is a unit test?

- Unit testing is a software testing level that aims to test a discrete piece of code. The “unit” refers to the **smallest** piece of code that can be tested **separately**
- Unit test
  - Must be repeatable
  - Isolates the code under test from the rest of the code
  - Doesn’t test how the unit interacts with other units

What is a unit test?

- ➔ Unit test have to test one question at a time and that question should reflect a requirement for our code
- ➔ Unit testing is about confirming that all the individual parts work, not that they work together
- ➔ Unit tests is usually written by the Development Team

System Under Test (SUT)

- In a database solution, the “unit” you want to test is typically a stored procedure, a trigger or a user-defined function
- It is very important to define the **System Under Test** (SUT) first and isolate it!
- System Under Test must not be influenced by other procedures or functions called within the one you want to test

What does a unit test give me?

- ➔ Unit tests convey safety
- ➔ Unit tests provide documentation of the software requirements
- ➔ Unit tests are preparatory to the design phase (TDD methodology), they force you to think how to organize properly
- ➔ Unit tests simplify the error checking process

When should I write the unit tests?

- ➔ Before starting the development?
  - ➔ Focusing on the actual requirements, it minimizes the work for the Developer
- ➔ During development?
  - ➔ Some tests may need to be reviewed due to development force a change in requirements
- ➔ After completing the development?
  - ➔ It's the only option, not the optimal one, if you have already written the code

Introduction to tSQLt

- The framework tSQLt was developed by Sebastian Meine and Dennis Lloyd, it's an open source framework for implementing unit tests in T-SQL for SQL Server and Azure SQL Database
- It works with
  - All editions of SQL Server starting from SQL Server 2005 SP2
  - Azure SQL Database
- It requires SQL CLR enabled

Benefits of using tSQLt

- ➔ Unit tests will be written in T-SQL, you don't need to learn new programming language
- ➔ Data manipulation will be rolled back at the end of the test so you don't need any data cleanup
- ➔ Mock objects are supported
- ➔ tSQLt can be integrated into SSDT projects or 3rd party tools

Benefits of using tSQLt

- ➔ Tests can be grouped within a single schema
- ➔ You can use a setup routine for a group of tests or class
- ➔ The output can be in plain text or XML

tSQLt Setup

- Download tSQLt scripts from tsqlt.org/downloads
- Enable CLR at the SQL Server instance level
- In each development database you want to install tSQLt
  - Enable TRUSTWORTHY property
  - Execute tSQLt.class.sql

Demo: tSQLt setup

The anatomy of a unit test

- ➔ Arrange (or Assemble)
  - ➔ Preparation of data on which the test will run
  - ➔ Isolation of the code from any external dependencies
- ➔ Act
  - ➔ The SUT will be executed and the output has been acquired as a result
- ➔ Assert
  - ➔ The expected result will be compared with the obtained one, the test will fail or will have a positive outcome according to this comparison

How does a test run with tSQLt

- When we run a test through tSQLt, the framework traces the running tests and starts a dedicated transaction
- The configuration procedure for the test class will be executed (if existing) and afterward the test will be executed

How does a test run with tSQLt

- At the end of the test, the tSQLt framework will rollback the dedicated transaction
- The results will be stored in the `tSQLt.TestResult` table

Our first unit test will be for a Trigger

Our first unit test will be for a Trigger

- You developed a Trigger for AdventureWorks2017 to prevent the insertion of new products with values less than 10 as a “safety stock”
- The Company Adventure Works LTD wishes to always have a warehouse stock of no less than 10 units for each product
- To make our trigger simple, it will only respond to the OnInsert event, for INSERT commands

Possible test

- ➔ Try to insert one correct row
- ➔ Try to insert one wrong row and observe the error caught by the Trigger
- ➔ Try to insert multiple rows in a single statements
  - ➔ Wrong and correct rows
  - ➔ Does the order matter?
- ➔ Let's start!! ☺

Demo: Let’s write our first unit test!

Isolate dependencies

- ➔ Tests often depend on data or the result of a stored procedure or function
  - ➔ This affects our test's repeatability
- ➔ We can't rely on data being unchanged
- ➔ We need to isolate
  - ➔ Data
  - ➔ Table constraints
  - ➔ Stored procedures and Triggers
  - ➔ Functions

Isolate tables

`tSQLt.FakeTable`

It creates a copy of the table without constraints and data The @TableName parameter indicates the table to fake

https://bit.ly/3fWrNML

Apply trigger

`tSQLt.ApplyTrigger`

Combined with `tSQLt.FakeTable` allows you to test triggers individually, in isolation of table constraints and other triggers on the table

https://bit.ly/3g2ZYm4

Isolate stored procedures

`tSQLt.SpyProcedure`

It replaces the execution of the stored procedure with the specified command

A log table named @ProcedureName_SpyProcedureLog is created and a new log entry is made for each fake execution of the “spied” procedure

https://bit.ly/2XWVZBI

DelphiDay italian conference

Isolate functions: Replace with a stub

```
tSQLt.RemoveObject
```

It removes the original object by renaming it in favor of the new one. It outputs a constant value or a value from a stub (mock) instead of the result of the real function

https://bit.ly/30V4jDr

Isolate functions: Replace with a fake

`tSQLt.FakeFunction`

It avoids executing the logic of a complex function

Both the original and the fake function must exist

The fake function may check the parameters value and simply return two constant values, one for successful check and another one for parameters check error

https://bit.ly/2PSmmE9

Summary

- ➔ We discussed about the importance of unit testing applied to database development
- ➔ We learn how to use tSQLt framework
- ➔ We learned how to write our first unit test using tSQLt

Summary

Think about triggers, SP and complex functions you wrote (those with more than 200 lines of code just to be clear ☺) would you feel safe in modifying them?

If the answer is “No” the first thing to do is to write the unit tests, now you know how!

Resources

- Articles
  - What it is and why it is important for T-SQL code
    - https://bit.ly/3tfUTP8
  - The tSQLt framework and the execution of a test
    - https://bit.ly/3oFy1Fb
  - How to write your first unit test for T-SQL code
    - https://bit.ly/3qkBRFq

- tSQLt - https://tsqlt.org

Resources

- Videos
  - Implementare unit testing su Sql Server - Alessandro Alpi
    - https://www.youtube.com/watch?v=7tIHg3POea0
  - Continuous Deployment, portare SQL Server nel mondo DevOps - Alessandro Alpi
    - https://www.youtube.com/watch?v=1jBm4MFFIPg
- Tools
  - Red-Gate SQL Test
    - https://www.red-gate.com/products/sql-development/sql-test/
  - TSQLUnit
    - https://github.com/aevdokimenko/tsqlunit

DelphiDay italian conference

THANK YOU

DelphiDay
italian conference