Unit Testing with SQL Server Data Tools

Global Hebrew Virtual Chapter

August 17, 2015

Andrey Zavadskiy, Krasnodar, Russia
MCSE/MCSD/MCT

About me

- Solutions architect, SQL & .NET developer
- 20 years in IT industry
- Worked with SQL Server since 7.0 back in 2001
- Developed in Visual Basic, C#, ASP.NET, MVC, JavaScript, SharePoint
- MCSE, MCSD, MCT
- PASS speaker

http://andreyzavadskiy.com

https://www.facebook.com/andrey.k.zavadskiy

@AndreyZavadskiy

https://www.linkedin.com/in/zavadskiy

Session Goal

Provide a practical overview of how to use SQL Server Data Tools to create and run database unit tests for Microsoft SQL Server

Contents

- What is a unit test
- Database objects to be tested
- Unit test flow
- Creating and running unit test
- Debugging database objects in unit test
- Custom unit test conditions

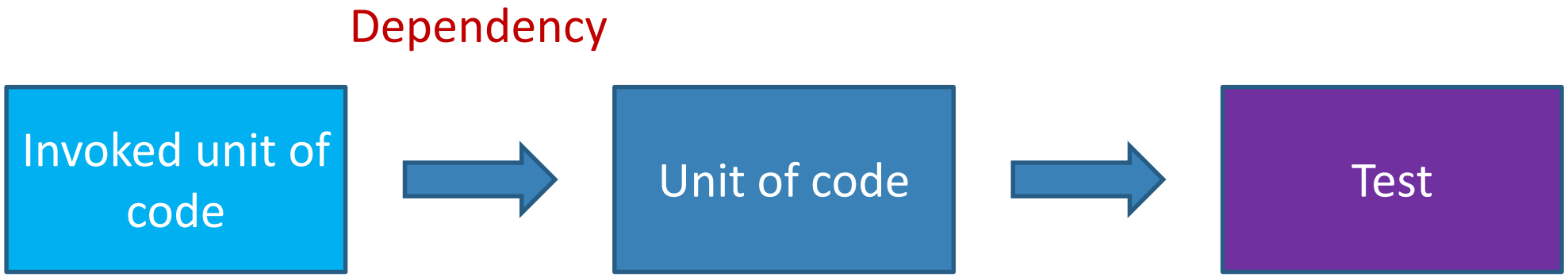
What is a unit test?

- Runs on a smallest piece of testable code
- Isolates from the other pieces of code
- Should be repeatable
- Gives the answer to only one question
- Usually created by developers

What for?

- Gives confidence in your code
- Confirms that product requirements are working
- Early error checking of code
- Instant visual feedback on errors
- Helps to check subsequent changes in code
- Provides documentation for other developers

Where are the bugs?



What can be tested?

Meta-data

- Table structure, field type and length
- Existence of objects

Constraints

- CHECK, DEFAULT, PRIMARY KEY, FOREIGN KEY, UNIQUE

T-SQL code

- Stored procedures, Functions, Triggers

Security permissions

Execution time

Data that can be tested

Scalar values

- Normal values
- Errors (incorrect values)
- Very big values
- NULL

Table values

- Rowset
- Empty rowset
- Very big rowset
- Metadata

DEMO

- Creating a database unit test project
- Unit Test Project Internals

Unit Test Flow

- Test initialize
- Unit test(s)
  - Pre-test
  - Test
  - Post-test
- Test cleanup

Unit Test Features

- Can have more than one test condition
- Can handle exceptions raised in database
- Can be run within a transaction
  - Particular test
  - All tests within a test class
- Can use a second connection for pre/post test phases

DEMO

- Positive test
- Negative test
- Running test in transaction
- Checking metadata
- Checking table equality

Debugging in unit tests

- Can debug only the T-SQL code to be tested
  - Breakpoint can be set inside the stored procedure, function or trigger
- Can’t debug the T-SQL code of the unit test itself

Steps to run debugging:

1. Enable SQL Server debugging on test project
2. Increase execution context timeout in app.config
3. Rebuild unit test project
4. Set breakpoints in the Transact-SQL code
5. Start debugging unit test

DEMO

Debugging a code from within a database unit test

Custom unit test conditions

- Are a Visual Studio IDE extensions
- Created in a separate project as a class library
- Compiled to a DLL
- How-to in MSDN article “Custom Test Conditions for SQL Server Unit Tests”
  - https://msdn.microsoft.com/en-us/library/jj860449(v=vs.103).aspx
- Example in my CodePlex project
  - https://ssdtconditions.codeplex.com/

DEMO

Using custom test conditions

Summary

- Good test scenario is a key to success
- SQL Server Data Tools is a framework for database unit testing
- Unit test has test class and T-SQL unit test(s)
- Unit test execution consists of 5 phases
- Debugging of T-SQL code during testing
- Assertion is made by various test conditions
- Allow customized test conditions

References

- MSDN: SQL Server Data Tools

  https://msdn.microsoft.com/en-us/library/hh272686(v=vs.103).aspx
- SSDT Team Blog

  http://blogs.msdn.com/b/ssdt/
- MSDN Forum

  https://social.msdn.microsoft.com/Forums/sqlserver/en-US/home?forum=ssdt

Questions?

Thank you for attending!