Test-Driven Development

José Almeida, Microsoft
(josealm@microsoft.com)

Visual Studio .net

Outline

- What is TDD?
- TDD in Practice
- Endo-Testing
- Test-Driven UI Development
- Conclusion

Visual Studio .net

Test-Driven Development (TDD)

- Is a programming practice
- Unit tests are written before the domain code
- Namely:
  - Write a test that fails
  - Write code to make the test pass
  - Refactor
- Unit Tests and Refactoring are the tools of TDD

Unit Tests

- Test specific aspects of a functionality
- Execute rapidly
- Independent of each other
- Independent of the surrounding environment
- Independent of execution order
- Automated

Refactoring

- Change the internal structure of the code without changing it's external behavior.
- Associated with arithmetic factorization:

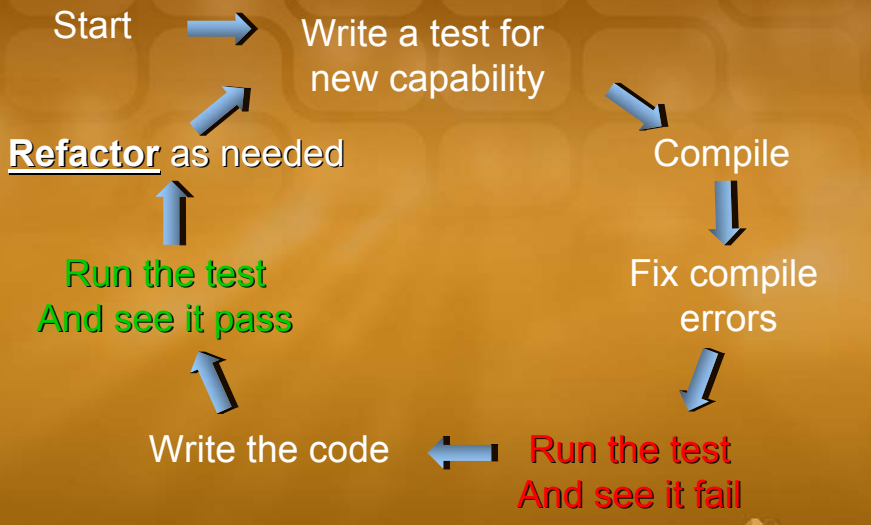
$$ab + ac = a(b+c)$$

- Same result but the expression is simplified.

Test List

- Task Based
  - 4-8 hour duration
  - 15-20 minute exercise at beginning
- Brainstorm a list of unit tests
- Do not get hung up on completeness, you can always add more later
- Describes completion requirements

Red/Green/Refactor



Visual Studio .net

demo

- Financial Service
  - A simple application that implements the following set of functionality:
    - Credit an Account
    - Debit an Account

Visual Studio .net

Characteristics of TDD

- TDD promotes code testability
- Testability is related to:
  - Strong cohesion
  - Loose coupling
  - No redundancy
  - Encapsulation
- These are good programming practices
- Striving for good tests results in better code

TDD Tenets

- Never write a single line of code unless you have a failing unit test
- Eliminate Duplication (Refactor mercilessly)

Observation

- It's harder to write unit tests for components located at the "edge" of the system:
  - Web Services
  - Data Access Layer
  - User Interface

Mock Objects

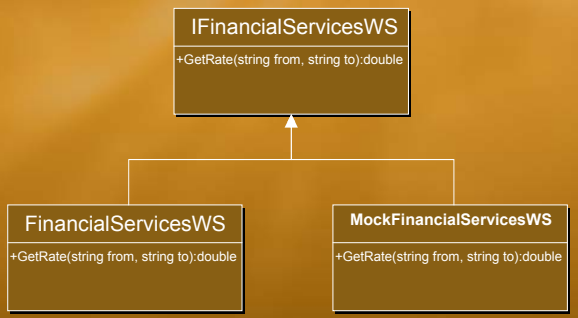
- Inherent challenges in testing dependant objects
  - Objects dependant on ephemeral objects produce unpredictable behavior
  - User Interfaces, Databases and the like are inherently ephemeral

Example

- How can we write a test for GetRate() when the result of FinancialServicesWS.GetRate() is unpredictable?

Mock Objects

- Solution:
  - Replace the unpredictable object with a testing version that produces predictable results



Steps to create a Dynamic Mock Object with DotNetMock

1. FinancialServicesWS is a proxy generated from the WSDL of the web service. Refactor the proxy to extract an interface: IFinancialServicesWS
2. Create a controller based on the extracted interface
3. Have the controller create a dynamic mock object
4. Specify the expected behavior of the mock object
5. Use the mock object in place of the real object in the test

Visual Studio .net

demo

Creating a Dynamic Mock Object

Visual Studio .net

Conclusion

- Advantages of Mock Objects:
  - Promote design to interfaces
  - Promote testability and isolation of tests
  - Promote decoupling
- Challenges of Mock Objects:
  - More classes to maintain
  - Requires breaking encapsulation to replace real object with mock object, sometimes resulting in less “elegant” code

User Interfaces

- Difficult to write automated tests to, because they are designed to be exercised by a user and often hide their programmatic interface.
- Inherent challenges in testing Windows Forms applications:
  - Are highly coupled
  - .Exe assemblies cannot be referenced from the IDE

NUnitForms

- NUnit Extension for testing Windows Forms applications

Steps to test a WinForms App

1. Place the forms in a separate class library
2. Create an Application Launcher that executes the forms
3. Reference the class library that contains the forms in the Test assembly
4. Design the Form
5. Use NUnitForms to write the tests
6. Write the code to pass the tests

demo

Unit Testing Windows Forms

Conclusion

- Advantages:
  - Easy to implement
  - Development of UI can be completely test-driven
  - Promotes decoupling
  - Enables creation of automated User Acceptance Tests
- Challenges:
  - Setup requires somewhat complex wiring

References

- "Test-Driven Development By Example", Kent Beck
- "Test-Driven Development in Microsoft .NET", James Newkirk, Alexei Vorontsov
- "Refactoring, Improving The Design Of Existing Code", Martin Fowler
- "Patterns Of Enterprise Application Architecture", Martin Fowler
- "The Humble Dialog Box", http://www.objectmentor.com/resources/articles/TheHumbleDialogBox.pdf

Visual Studio .net

Resources

- Visual Studio .NET 2005
  - http://msdn.microsoft.com/vstudio/teamsystem
- Visual Studio .NET 2003 Project Templates for NUnit:
  - http://www.pontonetpt.com/Downloads/317.aspx
- NUnit
  - http://www.nunit.org
- DotNetMock
  - http://sourceforge.net/projects/dotnetmock
- NUnitForms
  - http://nunitforms.sourceforge.net/
- C# Refactory
  - http://www.extreme-simplicity.net
- ReSharper
  - http://www.jetbrains.com/resharper

Visual Studio .net

Q & A

Visual Studio .net

Microsoft

Visual Studio .net