Unit testing

PBA Software Development

Test

Henrik Kühl

BUSINESS ACADEMY SOUTHWEST

Unit test definition

- A unit test is an automated piece of code that invokes the unit of work being tested, and then checks some assumptions about a single end result of that unit.
  - Is almost always written using a unit testing framework.
  - Its scope can span as little as a method or as much as multiple classes.
  - Can be written easily and runs quickly.
  - Is trustworthy, readable, repeatable, independent and maintainable.
  - It's consistent in its results as long as production code hasn't changed.

Unit of work

- A unit of work is the sum of actions that take place between the invocation of a public method in the system and a single noticeable end result by a test of that system.
- An end result is any of the following:
    1. The invoked method's return value or an exception.
    2. A noticeable change to the state of the system that can be determined without interrogating private state.
    3. There is a call to a third-party system over which the test has no control, and that third-party system doesn't return any value.

Integration test

- Integration testing is testing a unit of work without having full control over it and using one or more of its external dependencies.

- Examples of external dependencies:
  - File system
  - Database
  - Web service (or other forms of distributed programming)

- Integration tests should be separated from unit tests.

Basic naming conventions for unit tests

- Project
  - Create a test project named [ProjectUnderTest].UnitTests
- Class
  - Create a class named [ClassName]Tests
- Unit of work
  - Create a test method with the following name:
  - [UnitOfWorkName]_[ScenarioUnderTest]_[ExpectedBehavior]

Example: Unit under test

```
namespace Calculator
{
    public class Calc
    {
        public int Add(int a, int b)
        {
            return a + b;
        }
    }
}
```

Example: Unit test class (xUnit)

```
public class CalcTests
{
    [Fact]
    public void Add_TwoAndThree_ReturnsFive()
    {
        // Arrange
        Calc c = new Calc();

        // Act
        int result = c.Add(2, 3);

        // Assert
        Assert.Equal(5, result);
    }
}
}
```