學習

unit-testing

#unit-testing

1:

。。

。。

。。

。

;。

TDD。 TDD;。

。

Examples

-
-
-

'Arrange-Act-Assert''Given-When-Then'。

NUnitC。

```
[TestFixture]
public CalculatorTest
{
   [Test]
   public void Add_PassSevenAndThree_ExpectTen()
   {
       // Arrange - setup environment
       var systemUnderTest = new Calculator();

       // Act - Call system under test
       var calculatedSum = systemUnderTest.Add(7, 3);

       // Assert - Validate expected result
       Assert.AreEqual(10, calculatedSum);
  }
}
```

。

。。。。

```
// Test that oneDayFromNow returns a value 24*60*60 seconds later than current time

let systemUnderTest = new FortuneTeller()       // Arrange - setup environment
systemUnderTest.setNow(() => {return 10000})    //   inject a stub which will
```

```
                                                //   return 10000 as the result

let actual = systemUnderTest.oneDayFromNow()    // Act - Call system under test

assert.equals(actual, 10000 + 24 * 60 * 60)     // Assert - Validate expected result
```

oneDayFromNowDate.now。 。

。 。 。 。

```
// Test that squareOfDouble invokes square() with the doubled value

let systemUnderTest = new Calculator()          // Arrange - setup environment
let square = spy()
systemUnderTest.setSquare(square)               //   inject a spy

let actual = systemUnderTest.squareOfDouble(3)  // Act - Call system under test

assert(square.calledWith(6))                    // Assert - Validate expected interaction
```

Java + JUnit

JUnitJava。

。

```
public class BankAccount {
    private int balance;

    public BankAccount(int i){
        balance = i;
    }

    public BankAccount(){
        balance = 0;
    }

    public int getBalance(){
        return balance;
    }

    public void deposit(int i){
        balance += i;
    }

    public void withdraw(int i){
        balance -= i;
        if (balance < 0){
            balance -= 10; // penalty if overdrawn
        }
    }
}
```

BankAccount。

```
import org.junit.Test;
import static org.junit.Assert.*;

// Class that tests
public class BankAccountTest{

    BankAccount acc;

    @Before                         // This will run **before** EACH @Test
    public void setUptestDepositUpdatesBalance(){
        acc = new BankAccount(100);
    }

    @After                          // This Will run **after** EACH @Test
    public void tearDown(){
    // clean up code
    }

    @Test
    public void testDeposit(){
       // no need to instantiate a new BankAccount(), @Before does it for us

        acc.deposit(100);

        assertEquals(acc.getBalance(),200);
    }

    @Test
    public void testWithdrawUpdatesBalance(){
        acc.withdraw(30);

        assertEquals(acc.getBalance(),70); // pass
    }

    @Test
    public void testWithdrawAppliesPenaltyWhenOverdrawn(){

        acc.withdraw(120);

        assertEquals(acc.getBalance(),-30);
    }
}
```

NUnitC

```
using NUnit.Framework;

namespace MyModuleTests
{
    [TestFixture]
    public class MyClassTests
    {
        [TestCase(1, "Hello", true)]
        [TestCase(2, "bye", false)]
        public void MyMethod_WhenCalledWithParameters_ReturnsExpected(int param1, string
param2, bool expected)
        {
        //Arrange
        var foo = new MyClass(param1);
        //Act
        var result = foo.MyMethod(param2);

        //Assert
        Assert.AreEqual(expected, result);
        }
    }
}
```

python

```
import unittest

def addition(*args):
    """ add two or more summands and return the sum """

    if len(args) < 2:
        raise ValueError, 'at least two summands are needed'

    for ii in args:
        if not isinstance(ii, (int, long, float, complex )):
            raise TypeError

    # use build in function to do the job
    return sum(args)
```

```
class Test_SystemUnderTest(unittest.TestCase):

    def test_addition(self):
        """test addition function"""

        # use only one summand - raise an error
        with self.assertRaisesRegexp(ValueError, 'at least two summands'):
            addition(1)

        # use None - raise an error
        with self.assertRaises(TypeError):
            addition(1, None)

        # use ints and floats
        self.assertEqual(addition(1, 1.), 2)

        # use complex numbers
        self.assertEqual(addition(1, 1., 1+2j), 3+2j)

if __name__ == '__main__':
    unittest.main()
```

XUnit

```
using Xunit;

public class SimpleCalculatorTests
{
    [Theory]
    [InlineData(0, 0, 0, true)]
    [InlineData(1, 1, 2, true)]
    [InlineData(1, 1, 3, false)]
    public void Add_PassMultipleParameters_VerifyExpected(
        int inputX, int inputY, int expected, bool isExpectedCorrect)
    {
        // Arrange
        var sut = new SimpleCalculator();

        // Act
        var actual = sut.Add(inputX, inputY);

        // Assert
        if (isExpectedCorrect)
        {
            Assert.Equal(expected, actual);
        }
        else
        {
            Assert.NotEqual(expected, actual);
        }
    }
}

public class SimpleCalculator
{
    public int Add(int x, int y)
    {
        return x + y;
    }
}
```

https://riptutorial.com/zh-TW/unit-testing/topic/570/

2: Visual Studio for C

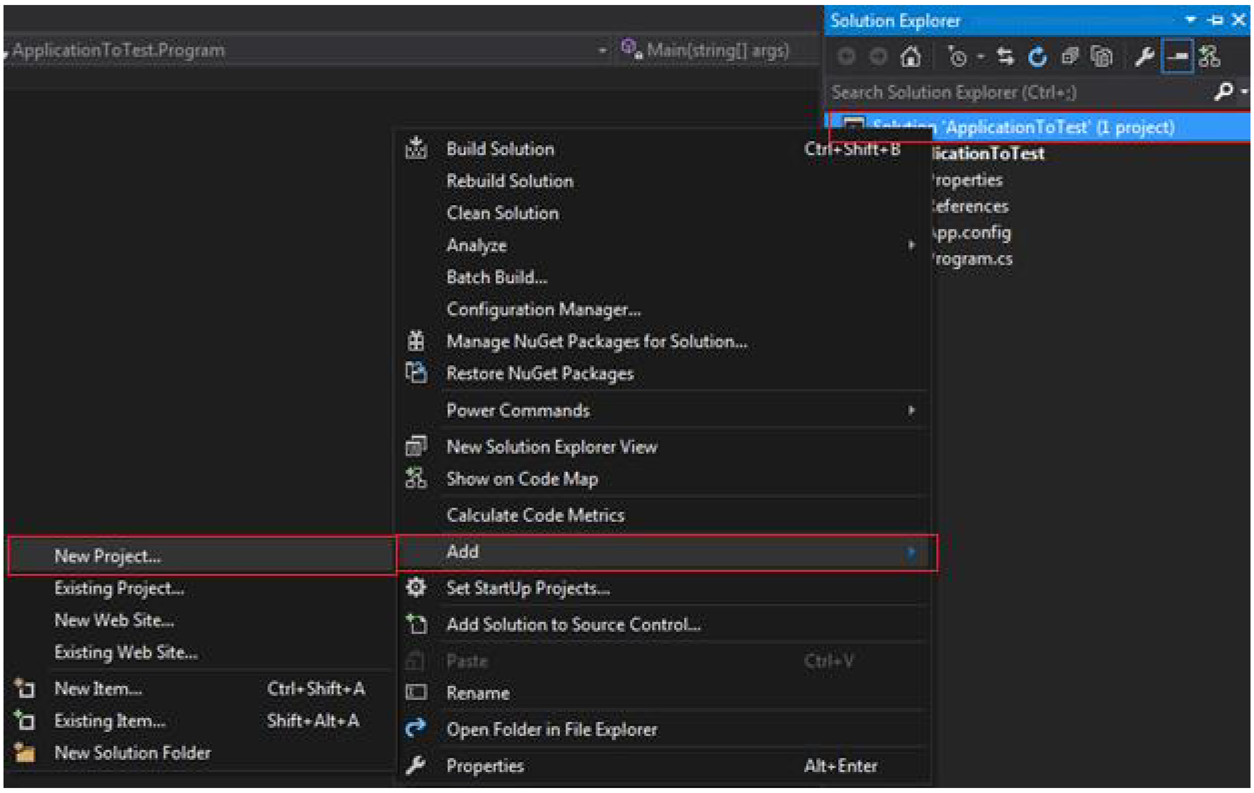
。

MSTestVisual Studio。

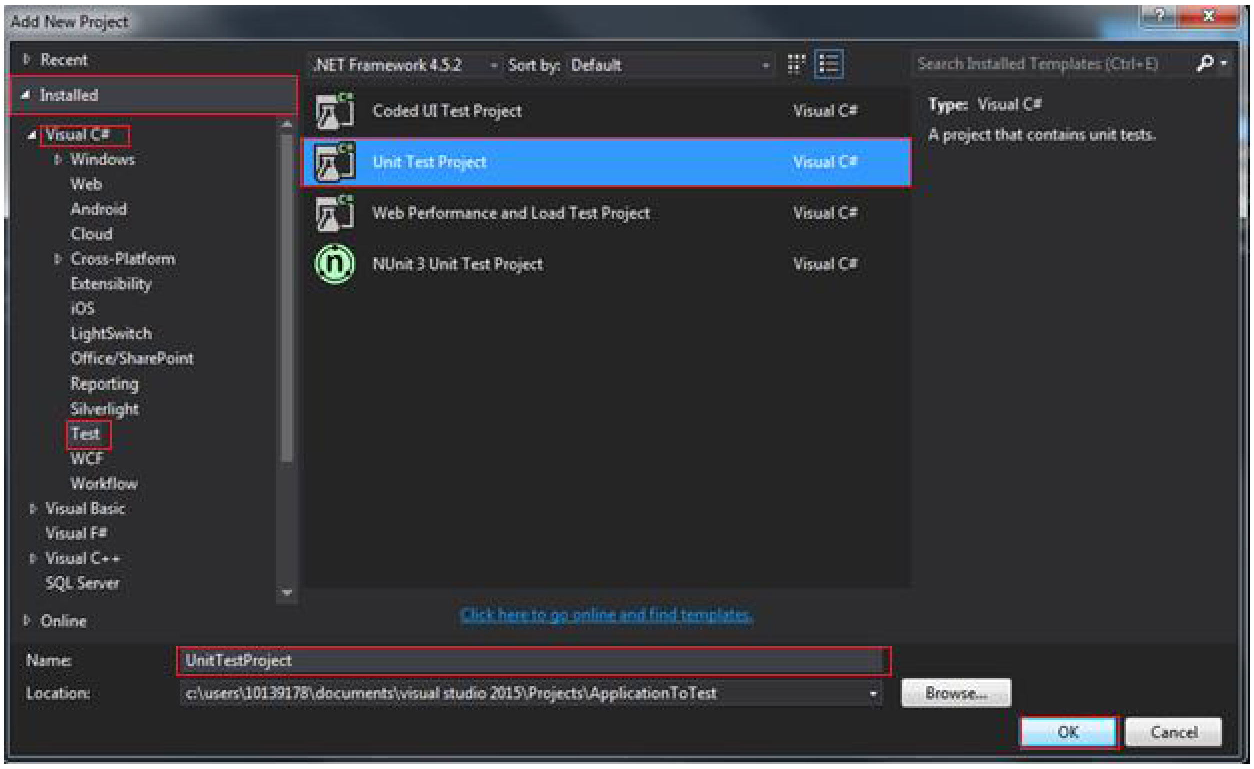
Visual Studio 2015。

Examples

- C
- - > - >...
- 1



- - > Visual C - >
- Unit Test Project
- “”
- 2


Add New Project

.NET Framework 4.5.2 · Sort by: Default · Search Installed Templates (Ctrl+E)

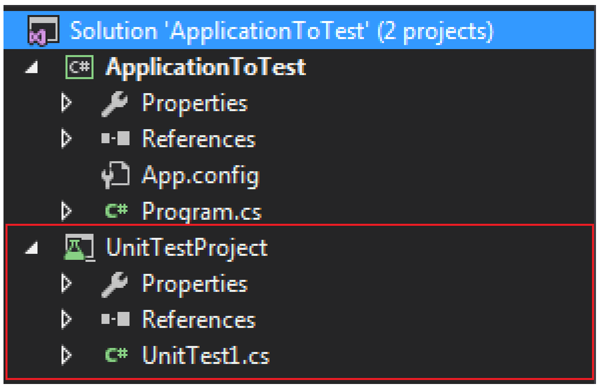
Installed > Visual C# > Test

Coded UI Test Project	Visual C#
Unit Test Project	Visual C#
Web Performance and Load Test Project	Visual C#
NUnit 3 Unit Test Project	Visual C#

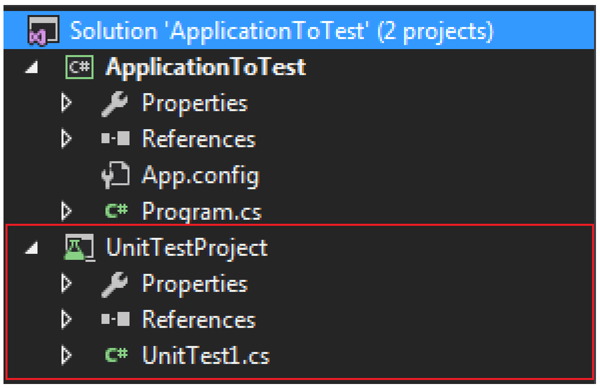
Type: Visual C#
A project that contains unit tests.

Name: UnitTestProject
Location: c:\users\10139178\documents\visual studio 2015\Projects\ApplicationToTest

-
- 3


```
Solution 'ApplicationToTest' (2 projects)
  ApplicationToTest
    Properties
    References
    App.config
    Program.cs
  UnitTestProject
    Properties
    References
    UnitTest1.cs
```

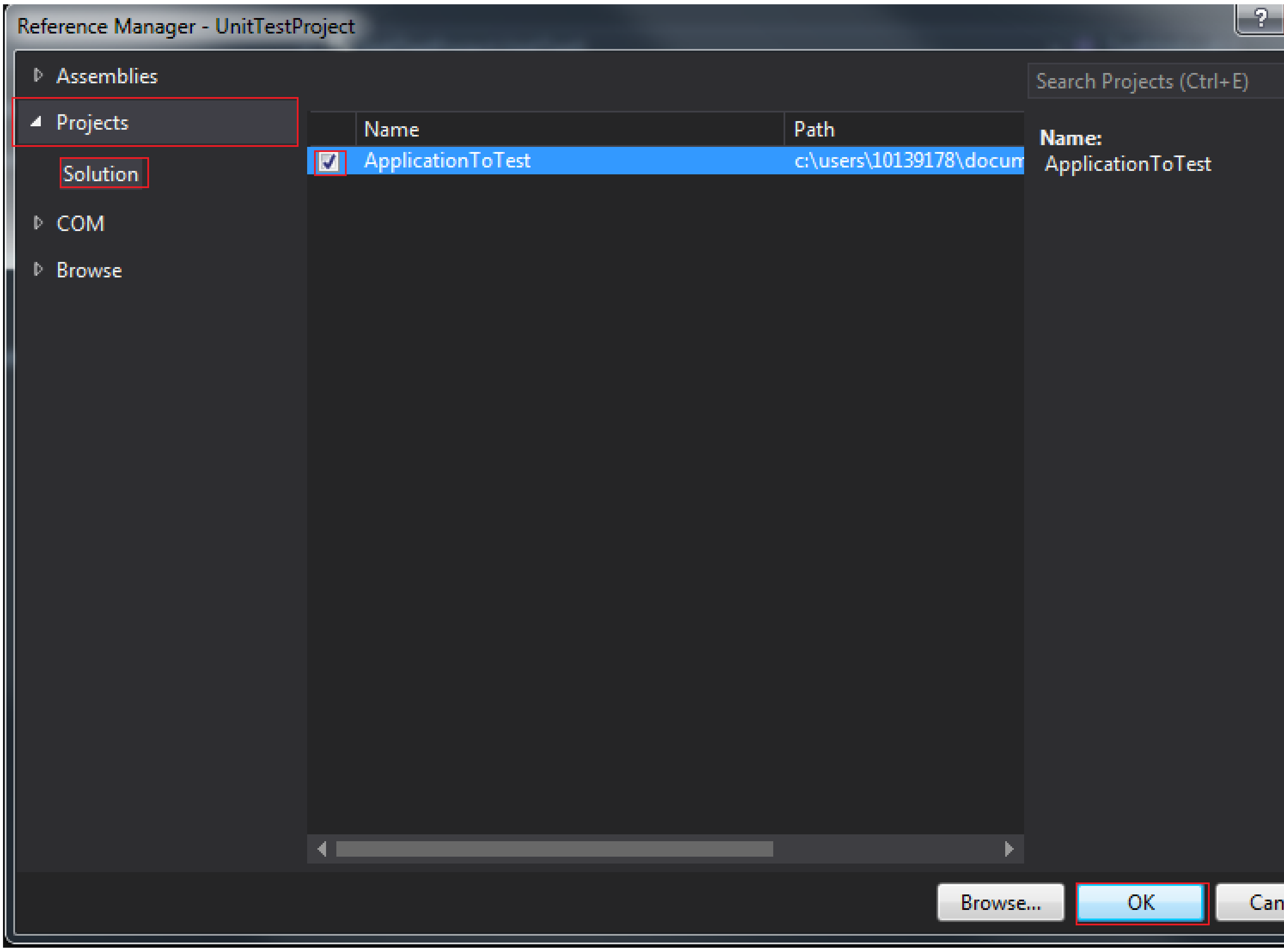
-
- - >...
- 3


```
Solution 'ApplicationToTest' (2 projects)
  ApplicationToTest
    Properties
    References
    App.config
    Program.cs
  UnitTestProject
    Properties
    References
    UnitTest1.cs
```

-
- - >
-

- >“”

- 4



	Name	Path
☑	ApplicationToTest	c:\users\10139178\docum

Name: ApplicationToTest

1

-
-

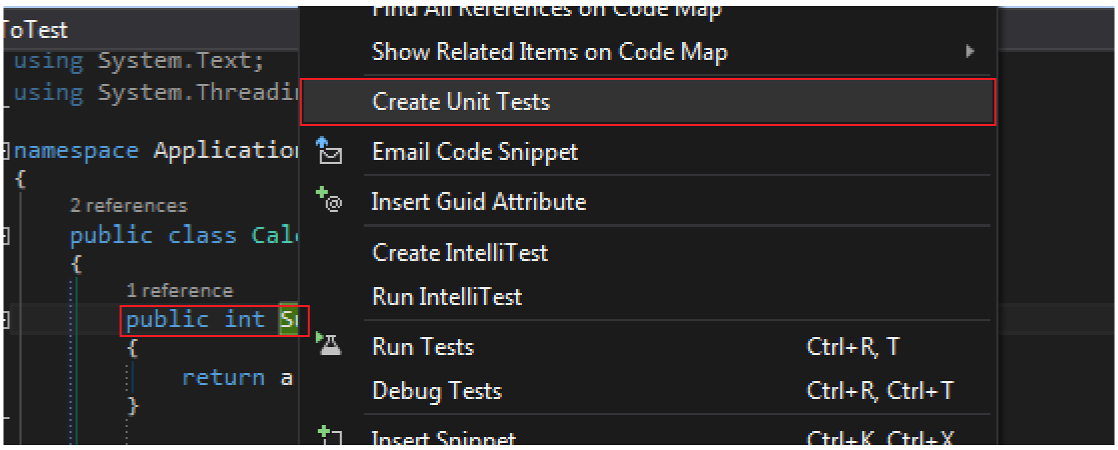
```
[Testclass]
public class UnitTest1
{
    [TestMethod]
    public void TestMethod1()
    {
        //Arrange
        ApplicationToTest.Calc ClassCalc = new ApplicationToTest.Calc();
        int expectedResult = 5;

        //Act
        int result = ClassCalc.Sum(2,3);

        //Assert
        Assert.AreEqual(expectedResult, result);
        }
    }
```

2

-
- - >
- 4



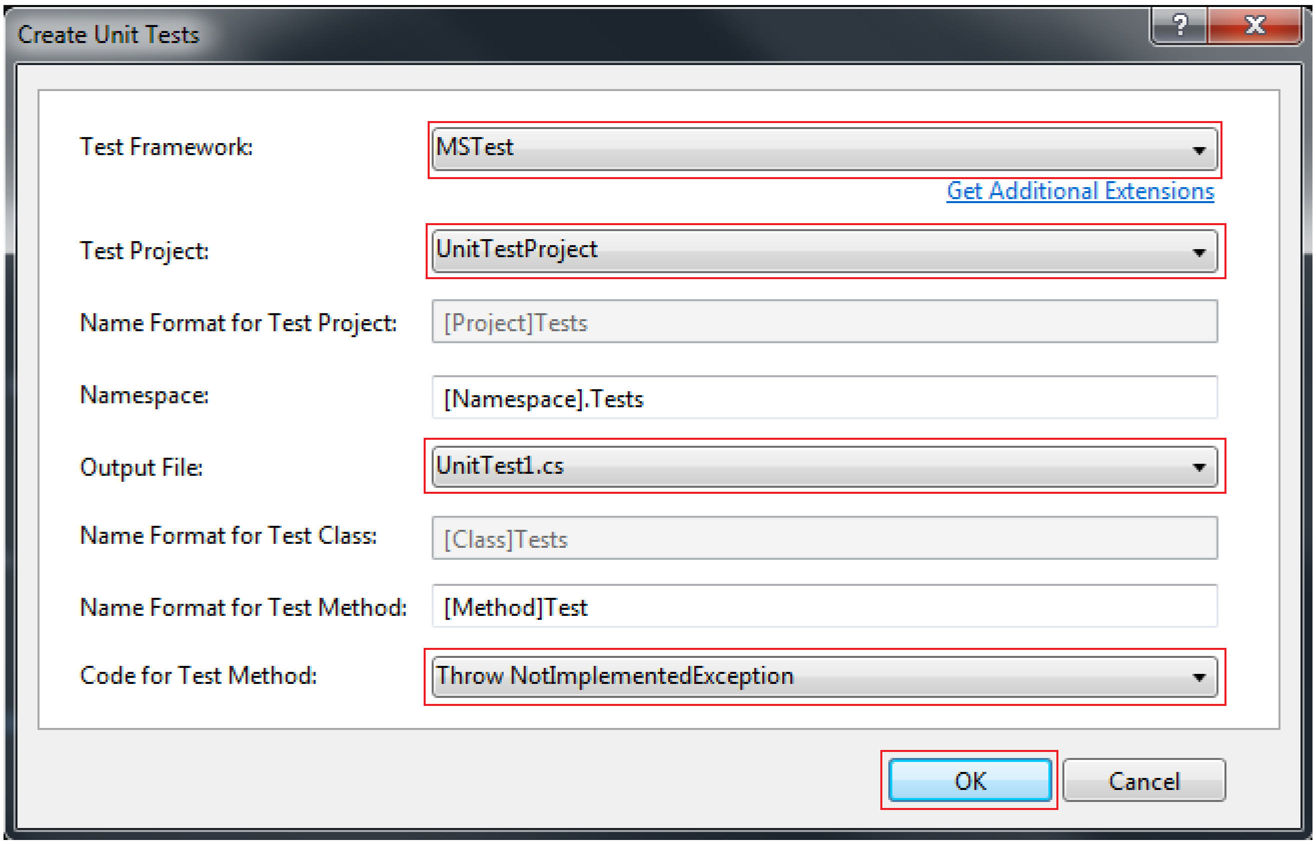
```
oTest
 using System.Text;
 using System.Threadi
namespace Applicatio
 {
     2 references
     public class Cal
     {
         1 reference
         public int S
         {
             return a
         }
```

Find All References on Code Map
Show Related Items on Code Map
Create Unit Tests
Email Code Snippet
Insert Guid Attribute
Create IntelliTest
Run IntelliTest
Run Tests Ctrl+R, T
Debug Tests Ctrl+R, Ctrl+T
Insert Snippet Ctrl+K, Ctrl+X

- MSTest
- Test Project
- Output File
- “”
-

◦ Test ProjectOutput File to◦

- 5


Create Unit Tests

Test Framework: MSTest

Test Project: UnitTestProject

Name Format for Test Project: [Project]Tests

Namespace: [Namespace].Tests

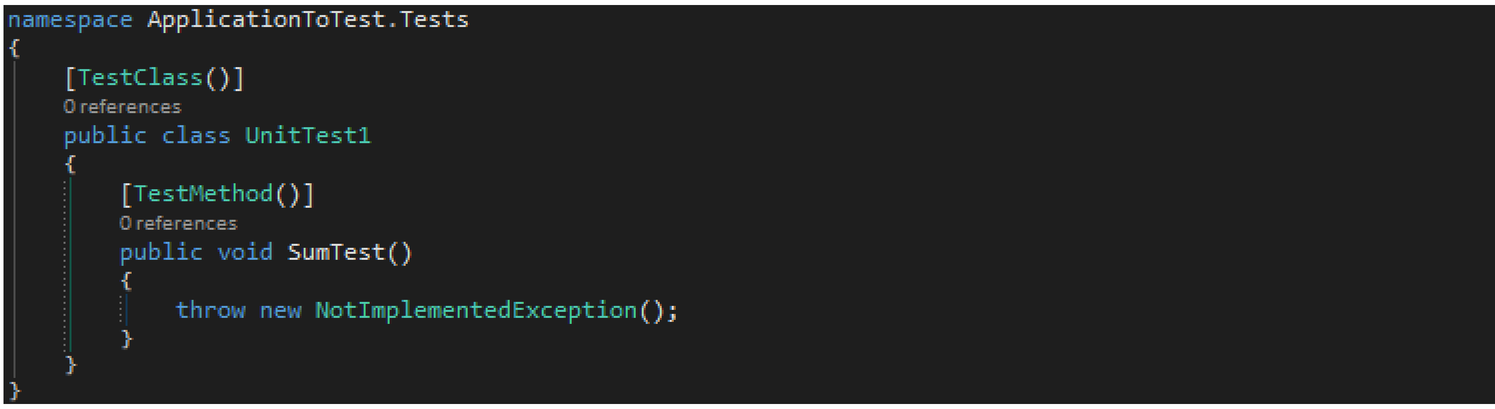
Output File: UnitTest1.cs

Name Format for Test Class: [Class]Tests

Name Format for Test Method: [Method]Test

Code for Test Method: Throw NotImplementedException

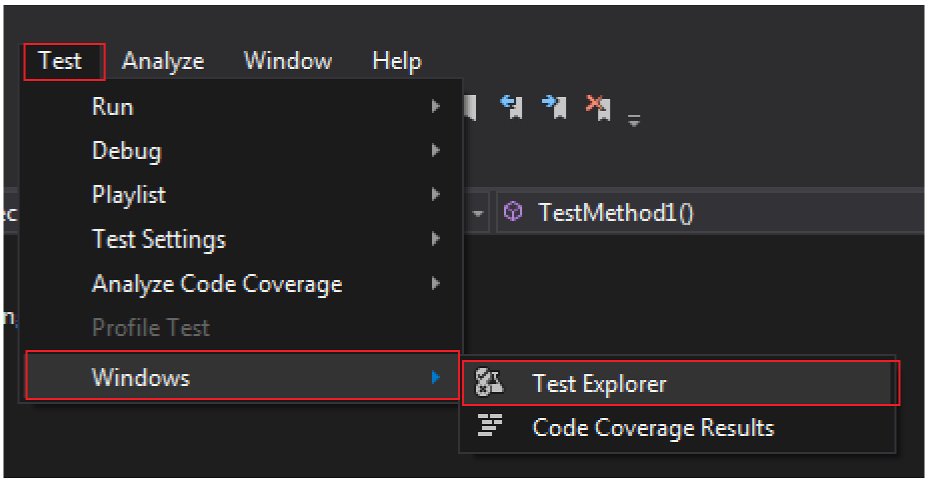
-
- 6



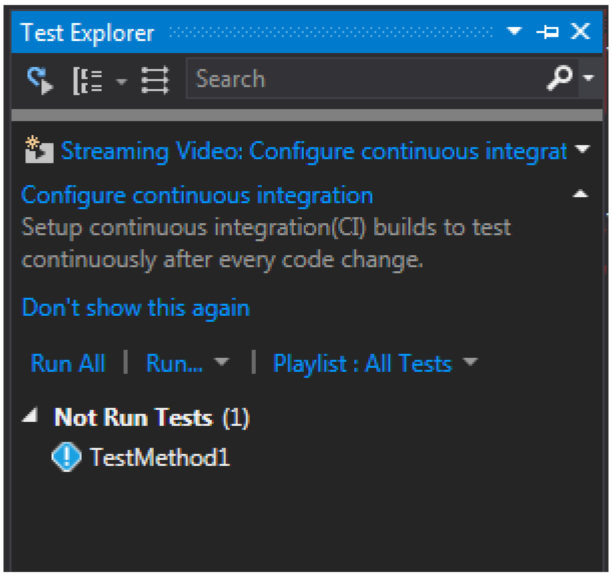
```
namespace ApplicationToTest.Tests
{
    [TestClass()]
    0 references
    public class UnitTest1
    {
        [TestMethod()]
        0 references
        public void SumTest()
        {
            throw new NotImplementedException();
        }
    }
}
```

Visual Studio

- - > Windows - >
- 1



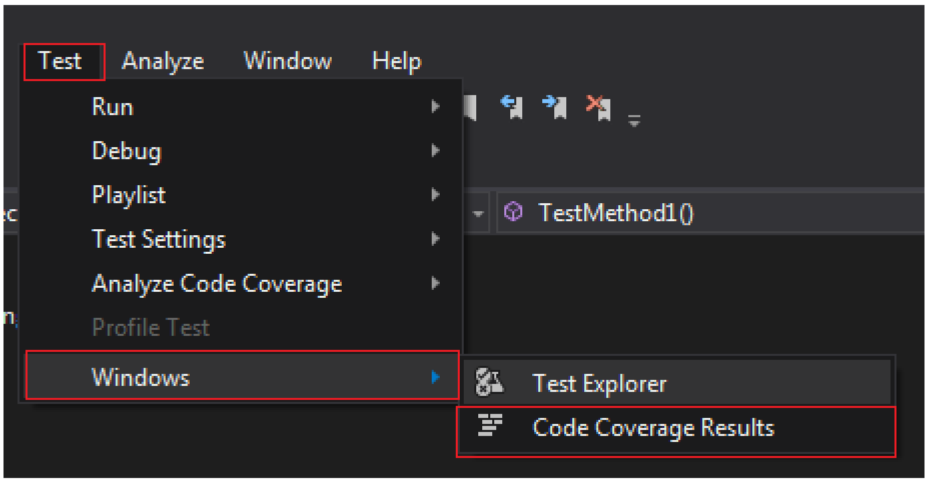
-
- 2


Test Explorer
Search
Streaming Video: Configure continuous integrat
Configure continuous integration
Setup continuous integration(CI) builds to test continuously after every code change.
Don't show this again
Run All | Run... | Playlist : All Tests
Not Run Tests (1)
TestMethod1

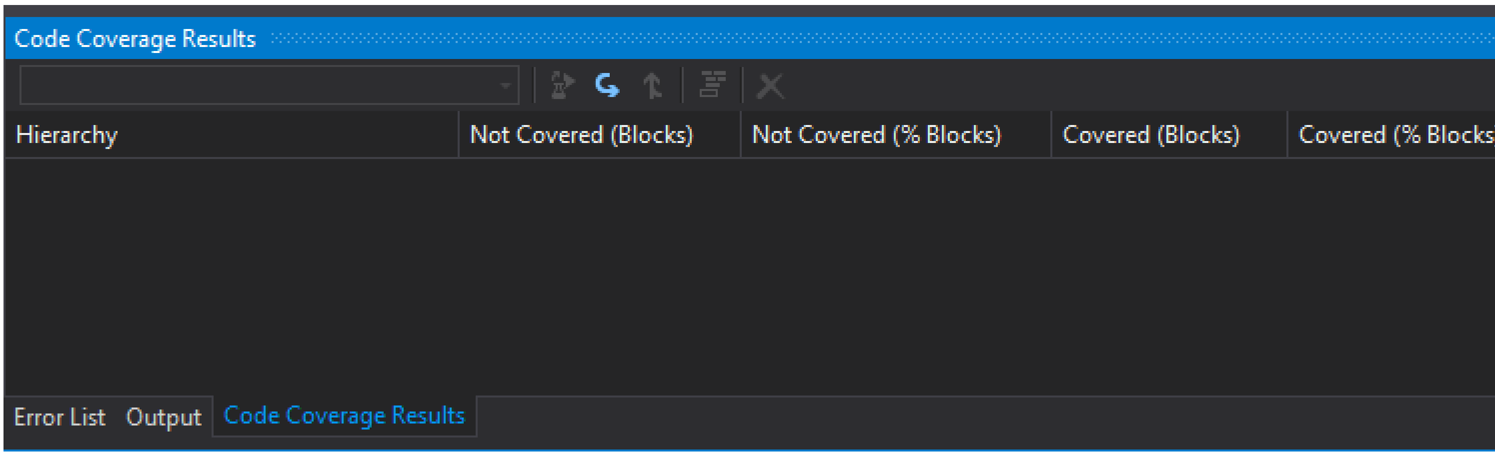
-
-
- ...
- “”1

Visual Studio

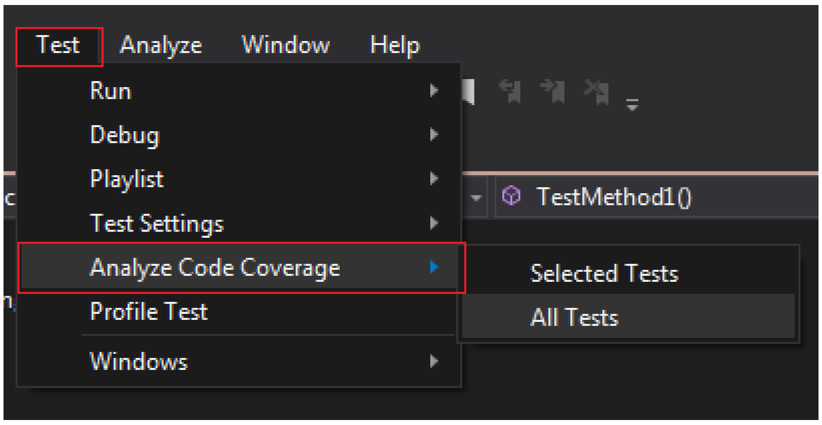
- - > Windows - >
- 1


Test
Analyze
Window
Help
Run
Debug
Playlist
Test Settings
Analyze Code Coverage
Profile Test
Windows
TestMethod1()
Test Explorer
Code Coverage Results

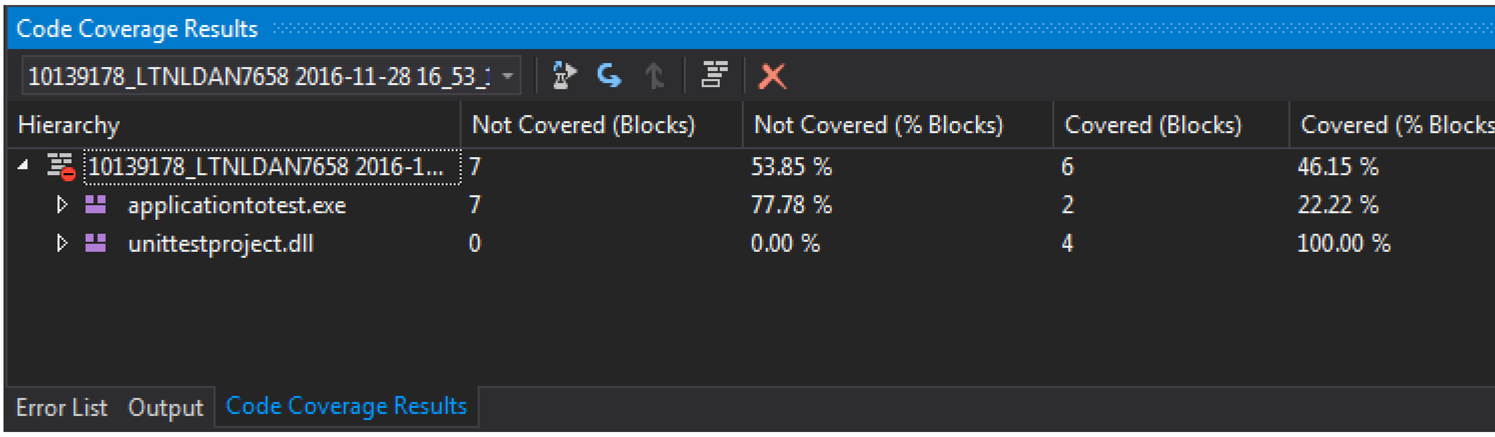
-
- 2


Code Coverage Results
Hierarchy
Not Covered (Blocks)
Not Covered (% Blocks)
Covered (Blocks)
Covered (% Blocks
Error List
Output
Code Coverage Results

-
- “” - >“”
- 3


Test
Analyze
Window
Help
Run
Debug
Playlist
Test Settings
Analyze Code Coverage
Profile Test
Windows
TestMethod1()
Selected Tests
All Tests

-
-
- 4


Code Coverage Results

10139178_LTNLDAN7658 2016-11-28 16_53_

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks
10139178_LTNLDAN7658 2016-1...	7	53.85 %	6	46.15 %
applicationtotest.exe	7	77.78 %	2	22.22 %
unittestproject.dll	0	0.00 %	4	100.00 %

Visual Studio for C https://riptutorial.com/zh-TW/unit-testing/topic/9953/visual-studio-for-c-

3:

。。。。

```
if (DateTime.Now.Date > processDate)
{
    // Do some processing
}
```

。。。。

IOC

;。 “”IoC。

;HTTP。 。 IoC*Singleton* 。

WebContext;。

。

Examples

。 。。

。 ILogger 。

```
public class RecordProcessor
{
    readonly private ILogger _logger;

    public RecordProcessor(ILogger logger)
    {
        _logger = logger;
    }

    public void DoSomeProcessing() {
        // ...
        _logger.Log("Complete");
    }
}
```

。 SRP。 。

```
[Test]
public void RecordProcessor_DependencyInjectionExample()
{
    ILogger logger = new FakeLoggerImpl(); //or create a mock by a mocking Framework

    var sut = new RecordProcessor(logger); //initialize with fake impl in testcode
    Assert.IsTrue(logger.HasCalledExpectedMethod());
}
```

。。

。Logger 。ConcreteLoggerILogger。 ILogger。

```
public class RecordProcessor
{
    public RecordProcessor()
    {
        Logger = new ConcreteLogger();
    }

    public ILogger Logger { get; set; }

    public void DoSomeProcessing()
    {
        // ...
        _logger.Log("Complete");
    }
}
```

Property Injection。

。。。

```
public void ProcessRecords(DateTime currentDate)
{
    foreach(var record in _records)
    {
        if (currentDate.Date > record.ProcessDate)
        {
            // Do some processing
        }
    }
}
```

/ DI

。/ Inversion of Control Containers。。。

```
public interface ILogger {
    void Log(string message);
}

public class ConcreteLogger : ILogger
{
    public ConcreteLogger()
    {
        // ...
    }
    public void Log(string message)
    {
        // ...
    }
}
public class SimpleClass
{
    public SimpleClass()
    {
        // ...
    }
}

public class SomeProcessor
{
    public SomeProcessor(ILogger logger, SimpleClass simpleClass)
    {
        // ...
    }
}
```

SomeProcessor ILoggerSimpleClass。 Unity。

。 。

```
// Register the container
var container = new UnityContainer();

// Register a type mapping.  This allows a `SimpleClass` instance
// to be constructed whenever it is required.
container.RegisterType<SimpleClass, SimpleClass>();

// Register an instance.  This will use this instance of `ConcreteLogger`
// Whenever an `ILogger` is required.
container.RegisterInstance<ILogger>(new ConcreteLogger());
```

```
var processor = container.Resolve<SomeProcessor>();
```

https://riptutorial.com/zh-TW/unit-testing/topic/597/

4:

◦ ◦ ◦

Examples

```
[Test]
Test1() {...} //Cryptic name - absolutely no information

[Test]
TestFoo() {...} //Name of the function - and where can I find the expected behaviour?

[Test]
TestTFSid567843() {...} //Huh? You want me to lookup the context in the database?
```

◦ ◦

◦ ◦

```
[Test]
public void GetOption_WithUnkownOption_ReturnsEmptyString() {...}
[Test]
public void GetOption_WithUnknownEmptyOption_ReturnsEmptyString() {...}
```

EnsureThat_ ◦ “EnsureThat_”

```
[Test]
public void EnsureThat_GetOption_WithUnkownOption_ReturnsEmptyString() {...}
[Test]
public void EnsureThat_GetOption_WithUnknownEmptyOption_ReturnsEmptyString() {...}
```

◦

```
[TestFixture]
public class OptionsTests //tests for class Options
{
    ...
}
```

◦

-

```
[Test]
public void EnsureThat_IsLeapYearIfDecimalMultipleOf4() {...}
[Test]
public void EnsureThat_IsNOTLeapYearIfDecimalMultipleOf100 {...}
[Test]
public void EnsureThat_IsLeapYearIfDecimalMultipleOf400 {...}
```

-

。

MakeSut

Testcode。 MakeSut

-
-
- 。

```
[Test]
public void TestSomething()
{
    var sut = MakeSut();

    string result = sut.Do();
    Assert.AreEqual("expected result", result);
}
```

MakeSut

```
private ClassUnderTest MakeSUT()
{
    return new ClassUnderTest();
}
```

```
private ScriptHandler MakeSut(ICompiler compiler = null, ILogger logger = null, string
scriptName="", string[] args = null)
{
    //default dependencies can be created here
    logger = logger ?? MockRepository.GenerateStub<ILogger>();
    ...
}
```

MakeSutTestrunner。

。 MakeSut。 testrunner。

https://riptutorial.com/zh-TW/unit-testing/topic/6074/-

5: Java

◦ ◦

◦

Examples

◦

◦

1

2

3

4

5mm <

6

7

8

◦

public class SimpleLoopTest {

private int [] numbers = {5-77,8-11,4,1-20,6,2,10};

```
/** Compute total of  positive numbers in the array
 *  @param numItems number of items to total.
 */
public int findSum(int numItems)
{
    int total = 0;
    if (numItems <= 10)
    {
        for (int count=0; count < numItems; count = count + 1)
        {
          if (numbers[count] > 0)
             {
                total = total + numbers[count];
             }
        }
    }
    return total;
}
```

}

TestPassTestCase {

```
public void testname() throws Exception {

    SimpleLoopTest s = new SimpleLoopTest();
    assertEquals(0, s.findSum(0));     //Test 1
    assertEquals(0, s.findSum(-1));    //Test 2
    assertEquals(5, s.findSum(1));     //Test 3
    assertEquals(5, s.findSum(2));     //Test 4
    assertEquals(17, s.findSum(5));    //Test 5
    assertEquals(26, s.findSum(9));    //Test 6
    assertEquals(36, s.findSum(10));   //Test 7
    assertEquals(0, s.findSum(11));    //Test 8
}
```

}

。

/。

。。Test3 / Test4 / Test5 / Test6 / Test7。

。。。

。

Java https://riptutorial.com/zh-TW/unit-testing/topic/10116/-java-

6:

SetUpTearDown

。

。 - 。

。 。 “”CUT“”SUT

。 。

SetUpTearDown

SetUpTearDown。

SetUpTearDown。 。 “”。

。 。 。 。

CUT。 StubsMocks。

- 。
- CUT。

1.

。

。

3.

。 。 。 。

4.

。 。

5.

。

1.AAA

。 。

“”。

2.

。。。

3.“”

。。。

4.

。。

-
-
- MaxIntMinInt
- 361
-
- 2GB

5.

。。

6.

。。。

7.

。。

8.

。。bug。。

。。。

9.

。。

10.

。SetUpTearDown。

11.。

。。

。

。 “DividedByZeroShouldThrowException”。

12.

。

13.CheckTrueAssert.IsTrue

。 CheckTrueAssert.IsTrueCheckEqualsAssert.IsEqual。

CheckTrue“TrueFalse。 ”

。

CheckEquals

“73”。

CheckTrueAssert.IsTrue。

14.

。 。 。 。 。

15.

。 。 。 。

Examples

C

。

ApplicationToTest。 Calc。 Sum。

Sum

```
public void Sum(int a, int b)
{
    return a + b;
}
```

```
[Testclass]
public class UnitTest1
{
    [TestMethod]
    public void TestMethod1()
    {
        //Arrange
        ApplicationToTest.Calc ClassCalc = new ApplicationToTest.Calc();
        int expectedResult = 5;

        //Act
        int result = ClassCalc.Sum(2,3);

        //Assert
        Assert.AreEqual(expectedResult, result);
    }
}
```

https://riptutorial.com/zh-TW/unit-testing/topic/9947/

7:

Examples

```
[Test]
public void Calculator_Add_ReturnsSumOfTwoNumbers()
{
    Calculator calculatorUnderTest = new Calculator();

    double result = calculatorUnderTest.Add(2, 3);

    Assert.AreEqual(5, result);
}
```

HairLength ShaveHeadHairLengthShaveHead。

```
public class Person
{
    public string Name;
    public int HairLength;

    public Person(string name, int hairLength)
    {
        this.Name = name;
        this.HairLength = hairLength;
    }

    public void ShaveHead()
    {
        this.HairLength = 0;
    }
}

[Test]
public void Person_ShaveHead_SetsHairLengthToZero()
{
    Person personUnderTest = new Person("Danny", 10);

    personUnderTest.ShaveHead();

    int hairLength = personUnderTest.HairLength;

    Assert.AreEqual(0, hairLength);
}
```

。 NUnitAssert.Throws。 。

```
[Test]
public void GetItem_NegativeNumber_ThrowsArgumentInvalidException
{
    ShoppingCart shoppingCartUnderTest = new ShoppingCart();
    shoppingCartUnderTest.Add("apple");
    shoppingCartUnderTest.Add("banana");
    double invalidItemNumber = -7;

    bool exceptionThrown = false;

    try
    {
        shoppingCartUnderTest.GetItem(invalidItemNumber);
    }
    catch(ArgumentInvalidException e)
    {
        exceptionThrown = true;
    }

    Assert.True(exceptionThrown);
}
```

https://riptutorial.com/zh-TW/unit-testing/topic/6330/

8:

。 。

Examples

。 “”。

```
public interface IRecordProvider {
    IEnumerable<Record> GetRecords();
}
```

```
public bool ProcessRecord(IRecordProvider provider)
```

。

```
public class RecordProviderStub : IRecordProvider
{
    public IEnumerable<Record> GetRecords()
    {
        return new List<Record> {
            new Record { Id = 1, Flag=false, Value="First" },
            new Record { Id = 2, Flag=true, Value="Second" },
            new Record { Id = 3, Flag=false, Value="Third" }
        };
    }
}
```

。

```
var stub = new RecordProviderStub();
var processed = sut.ProcessRecord(stub);
```

MockStub。 StubMocking。

“”。

Moq

```
var stub = new Mock<IRecordProvider>();
stub.Setup(provider => provider.GetRecords()).Returns(new List<Record> {
    new Record { Id = 1, Flag=false, Value="First" },
    new Record { Id = 2, Flag=true, Value="Second" },
    new Record { Id = 3, Flag=false, Value="Third" }
});
```

```
var processed = sut.ProcessRecord(stub.Object);
```

。。

ProcessRecord Record UseValue Flag==true 。

```
var stub = new Mock<IRecordProvider>();
stub.Setup(provider => provider.GetRecords()).Returns(new List<Record> {
    new Record { Id = 1, Flag=false, Value="First" },
    new Record { Id = 2, Flag=true, Value="Second" },
    new Record { Id = 3, Flag=false, Value="Third" }
});
```

IServicemock

```
var mockService = new Mock<IService>();
mockService.Setup(service => service.UseValue(It.IsAny<string>())).Returns(true);
```

。

```
var sut = new SystemUnderTest(mockService.Object);

var processed = sut.ProcessRecord(stub.Object);
```

。 UseValue “Second”Flag==true 。

```
mockService.Verify(service => service.UseValue("Second"));
```

https://riptutorial.com/zh-TW/unit-testing/topic/615/

S. No		Contributors
1		Andrey, Carl Manaster, Community, Farukh, forsvarir, Fred Kleuver, mahei, mark_h, Quill, silver, Stephen Byrne, Thomas Weller, zhon
2	Visual Studio for C	DarkAngel
3		forsvarir, kayess, mrAtari, Pavel Voronin, Stephen Byrne
4		mrAtari, RamenChef, Shrinivas Patgar, user2314737
5	Java	Remya
6		DarkAngel
7		Danny
8		forsvarir