测试

一个简单测试，不要把他当成测试二看应用！

overview

本文将以以下三个功能来证明测试的优势

- 可维护
- 可扩展性
- 可读性

可读性

单元测试的结构

为了能清晰表达测试意图和结果，应该使用“三段式”结构来 when-then-then结构来写测试。

例如

```
[Test]
[Description("Verify that the Sum method adds two numbers")]
public void Sum_ShouldAddTwoNumbers()
{
    // Arrange
    var calculator = new Calculator();
    int a = 5;
    int b = 3;
    int expectedSum = 8;

    // Act
    int actualSum = calculator.Sum(a, b);

    // Assert
    Assert.AreEqual(expectedSum, actualSum);
}
```

没有测试

如果一个测试没有意义，即使他不报错或断言通过，也不了解被测试的单元的测试，

常见现象

```
[Test]
public void test1()
{
    var result = Calculator.Sum(5, 3);
    Console.WriteLine("result: " + result);
    // 没有断言，测试会通过，但没有实际验证
}
```

问题

缺乏Assertions，要知道发生了什么

```
[Test]
public void test2()
{
    var result = Calculator.Sum(5, 3);
    if (result == 8)
        Console.WriteLine("Test passed");
    else
        Console.WriteLine("Test failed");
}
```

问题

使用Console.WriteLine，需要查看输出，自动化困难

可维护性

逻辑修改不少

逻辑修改不少的情况下，但仅仅是测试代码发生修改——即单元测试本身的逻辑发生修改，则说明测试代码本身的逻辑发生了变化，测试代码的逻辑发生了变化，测试代码的逻辑发生了变化，测试代码