Модульное тестирование кода C# с использованием MSTest и .NET

Создание исходного проекта

Откройте окно оболочки. Создайте каталог с именем *unit-testing-using-mstest* для хранения решения. В этом каталоге выполните команду dotnet new sln, чтобы создать файл решения для библиотеки классов и тестового проекта. Создайте каталог PrimeService. Ниже приведена актуальная структура каталогов и файлов:

```
/unit-testing-using-mstest
    unit-testing-using-mstest.sln
    /PrimeService
```

Перейдите в каталог *PrimeService* и выполните команду `dotnet new classlib`, чтобы создать исходный проект. Переименуйте *Class1.cs* в *PrimeService.cs*. Замените код в файле следующим кодом, чтобы создать неудачную реализацию класса *PrimeService*.

```
using System;

namespace Prime.Services
{
    public class PrimeService
    {
        public bool IsPrime(int candidate)
        {
            throw new NotImplementedException("Please create a test first.");
        }
    }
}
```

Вернитесь в каталог *unit-testing-using-mstest*. Выполните команду `dotnet sln add`, чтобы добавить проект библиотеки классов в решение.

```
dotnet sln add PrimeService/PrimeService.csproj
```

Создание тестового проекта

Создайте каталог *PrimeService.Tests*. Ниже представлена структура каталогов:

```
/unit-testing-using-mstest
    unit-testing-using-mstest.sln
```

```
    /PrimeService
        Source Files
        PrimeService.csproj
    /PrimeService.Tests
```

Перейдите в каталог *PrimeService.Tests* и создайте проект с помощью `dotnet new mstest`. Команда dotnet new создает тестовый проект, который использует `MSTest` в качестве библиотеки тестов. Шаблон настраивает средство запуска тестов в файле *PrimeServiceTests.csproj*.

```
<ItemGroup>
  <PackageReference Include="Microsoft.NET.Test.Sdk" Version="16.7.1" />
  <PackageReference Include="MSTest.TestAdapter" Version="2.1.1" />
  <PackageReference Include="MSTest.TestFramework" Version="2.1.1" />
  <PackageReference Include="coverlet.collector" Version="1.3.0" />
</ItemGroup>
```

Тестовый проект требует других пакетов для создания и выполнения модульных тестов. Команда `dotnet new` на предыдущем шаге добавила пакет SDK, платформу тестирования и средство запуска тестов `MSTest`, а также модуль `Coverlet` для отчетов об объеме протестированного кода.

Добавьте библиотеку классов `PrimeService` в качестве еще одной зависимости проекта. Использование команды `dotnet add reference`:

```
dotnet add reference ../PrimeService/PrimeService.csproj
```

Ниже показан окончательный макет решения:

```
/unit-testing-using-mstest
    unit-testing-using-mstest.sln
    /PrimeService
        Source Files
        PrimeService.csproj
    /PrimeService.Tests
        Test Source Files
        PrimeServiceTests.csproj
```

Перейдите в каталог *unit-testing-using-mstest* и выполните команду `dotnet sln add`.

```
dotnet sln add ./PrimeService.Tests/PrimeService.Tests.csproj
```

Создание первого теста

Напишите один неудачный тест, обеспечьте его прохождение, а затем повторите процесс. Удалите *UnitTest1.cs* из каталога *PrimeService.Tests* и создайте файл C# *PrimeService_IsPrimeShould.cs* со следующим содержимым:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Prime.Services;

namespace Prime.UnitTests.Services
{
    [TestClass]
    public class PrimeService_IsPrimeShould
    {
        private readonly PrimeService _primeService;

        public PrimeService_IsPrimeShould()
        {
            _primeService = new PrimeService();
        }

        [TestMethod]
        public void IsPrime_InputIs1_ReturnFalse()
        {
            bool result = _primeService.IsPrime(1);

            Assert.IsFalse(result, "1 should not be prime");
        }
    }
}
```

Атрибут `TestClass` обозначает класс, содержащий модульные тесты. Атрибут `TestMethod` указывает, что метод — это метода теста.

Сохраните этот файл и выполните `dotnet test`, чтобы создать тесты и библиотеку классов, а затем запустите тесты. Средство запуска тестов `MSTest` содержит точку входа в программу для выполнения тестов. dotnet test запускает средство выполнения тестов с помощью проекта модульного теста, который вы создали.

Тест не будет пройден. Вы еще не создали реализацию. Чтобы тест был пройден, напишите простейший код в классе *PrimeService*, который работает:

```
public bool IsPrime(int candidate)
{
    if (candidate == 1)
    {
        return false;
    }
    throw new NotImplementedException("Please create a test first.");
}
```

Добавление дополнительных возможностей

Теперь, когда тест проходит успешно, пора создать дополнительные тесты. Есть еще ряд элементарных случаев с простыми числами: 0, -1. Можно добавлять новые тесты с помощью атрибута `TestMethod`, но это скоро станет утомительным. Есть другие атрибуты `MSTest`, которые позволяют создавать наборы похожих тестов. Метод теста может выполнять тот же код, но иметь разные входные аргументы. С помощью атрибута `DataRow` можно указать значения для этих входных аргументов.

Вместо того чтобы создавать новые тесты, используйте эти два атрибута, чтобы создать единый управляемый данными тест, Тест на основе данных — это метод, который проверяет несколько значений меньше 2 (наименьшего простого числа). Добавьте новый метод теста в файл *PrimeService_IsPrimeShould.cs*.

```
[TestMethod]
[DataRow(-1)]
[DataRow(0)]
[DataRow(1)]
public void IsPrime_ValuesLessThan2_ReturnFalse(int value)
{
    var result = _primeService.IsPrime(value);

    Assert.IsFalse(result, $"{value} should not be prime");
}
```

Выполните команду `dotnet test`, и два из этих тестов завершаются ошибкой. Для успешного выполнения всех тестов нужно изменить предложение if в начале метода `IsPrime` в файле *PrimeService.cs*:

```
if (candidate < 2)
```