# Начало работы с Live Unit Testing

При включении функции Live Unit Testing в решении Visual Studio она визуально описывает охват тестов и их состояние. Она также динамически выполняет тесты при каждом изменении кода и немедленно уведомляет, если изменение вызвало сбой.

Live Unit Testing можно использовать для тестирования решений, предназначенных как для .NET Framework, так и для .NET Core. С помощью этого руководства вы научитесь использовать Live Unit Testing, создав простую библиотеку классов, ориентированную на .NET Standard, а также создадите проект MSTest, ориентированный на .NET Core, для тестирования.

Полноценное решение на C# можно скачать из репозитория [MicrosoftDocs/visualstudio-docs](https://github.com/MicrosoftDocs/visualstudio-docs/tree/master/docs/test/samples/csharp/UtilityLibraries/) на сайте GitHub.

## Предварительные требования

Для работы с руководством нужно установить выпуск Visual Studio Enterprise с рабочей нагрузкой **Кроссплатформенная разработка .NET Core**.

## Создание решения и проекта библиотеки классов

Начните с создания решения Visual Studio с именем UtilityLibraries, состоящего из одного проекта библиотеки классов .NET Standard, StringLibrary.

Решение — это просто контейнер для одного или нескольких проектов. Чтобы создать пустое решение, откройте Visual Studio и сделайте следующее:

1. Выберите **Файл**>**Создать**>**Проект** в меню верхнего уровня Visual Studio.
2. Введите **решение** в поле поиска шаблона и выберите шаблон **Пустое решение**. Присвойте проекту имя **UtilityLibraries**.
3. Завершите создание решения.

После создания решения вы создадите библиотеку классов с именем StringLibrary, которая содержит несколько методов расширения для работы со строками.

1. В **обозревателе решений** щелкните решение UtilityLibraries правой кнопкой мыши и последовательно выберите пункты **Добавить**>**Новый проект**.
2. Введите **библиотека классов** в поле поиска шаблона и выберите шаблон **Библиотека классов (.NET Standard)** . Нажмите кнопку **Далее**.

**Примечание**

Так как наша библиотека ориентирована скорее на платформу .NET Standard, чем на ее конкретную реализацию .NET, ее можно вызвать из любой реализации .NET, которая поддерживает эту версию .NET Standard. Дополнительные сведения см. в статье **.NET Standard**.

1. Присвойте проекту имя **StringLibrary**.
2. Щелкните **Создать**, чтобы создать проект.
3. Замените весь существующий код, отображаемый в редакторе кода, следующим кодом:

C#Копировать

using System;

namespace UtilityLibraries

{

public static class StringLibrary

{

public static bool StartsWithUpper(this string s)

{

if (String.IsNullOrWhiteSpace(s))

return false;

return Char.IsUpper(s[0]);

}

public static bool StartsWithLower(this string s)

{

if (String.IsNullOrWhiteSpace(s))

return false;

return Char.IsLower(s[0]);

}

public static bool HasEmbeddedSpaces(this string s)

{

foreach (var ch in s.Trim())

{

if (ch == ' ')

return true;

}

return false;

}

}

}

StringLibrary имеет три статических метода:

* + StartsWithUpper возвращает true, если строка начинается с прописной буквы; в противном случае он возвращает false.
  + StartsWithLower возвращает true, если строка начинается со строчной буквы; в противном случае он возвращает false.
  + HasEmbeddedSpaces возвращает true, если строка содержит внедренный пробел; в противном случае он возвращает false.

1. Выберите **Сборка**>**Сборка решения** в меню верхнего уровня Visual Studio. Сборка должна выполниться успешно.

## Создание тестового проекта

Следующим шагом является создание проекта модульного теста для тестирования библиотеки StringLibrary. Создайте модульные тесты, выполнив следующие действия:

1. В **обозревателе решений** щелкните решение UtilityLibraries правой кнопкой мыши и последовательно выберите пункты **Добавить**>**Новый проект**.
2. Введите **модульный тест** в поле поиска шаблона, выберите **C#** в качестве языка, а затем выберите **Проект модульного теста** для шаблона .NET Core. Щелкните **Далее**.

**Примечание**

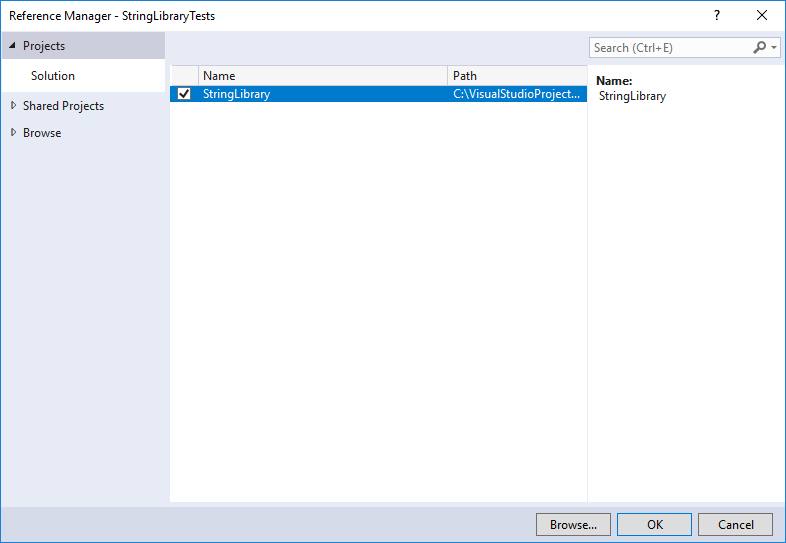
Начиная с Visual Studio 2019 версии 16.9, имя шаблона проекта MSTest изменено с **Проект модульного теста MSTest (.NET Core)** на **Проект модульного теста**.

1. Назовите проект **StringLibraryTests** и щелкните **Далее**.
2. Выберите рекомендуемую версию целевой платформы или .NET 6 и щелкните **Создать**.

**Примечание**

Этот учебник по началу работы использует Live Unit Testing с платформой тестирования MSTest. Вы также можете использовать платформы тестирования xUnit и NUnit.

1. Проект модульного теста не может автоматически получить доступ к тестируемой библиотеке классов. Вы предоставляете доступ к библиотеке тестов, добавив ссылку на проект библиотеки классов. Для этого щелкните проект StringLibraryTests правой кнопкой мыши и выберите **Добавить**>**Ссылка**. В диалоговом окне **Диспетчер ссылок** щелкните вкладку **Решение** и выберите проект StringLibrary, как показано на следующей иллюстрации.



1. Замените стандартный код модульного теста, включенный в шаблон, следующим кодом:

C#Копировать

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using UtilityLibraries;

namespace StringLibraryTest

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestStartsWithUpper()

{

// Tests that we expect to return true.

string[] words = { "Alphabet", "Zebra", "ABC", "Αθήνα", "Москва" };

foreach (var word in words)

{

bool result = word.StartsWithUpper();

Assert.IsTrue(result,

$"Expected for '{word}': true; Actual: {result}");

}

}

[TestMethod]

public void TestDoesNotStartWithUpper()

{

// Tests that we expect to return false.

string[] words = { "alphabet", "zebra", "abc", "αυτοκινητοβιομηχανία", "государство",

"1234", ".", ";", " " };

foreach (var word in words)

{

bool result = word.StartsWithUpper();

Assert.IsFalse(result,

$"Expected for '{word}': false; Actual: {result}");

}

}

[TestMethod]

public void DirectCallWithNullOrEmpty()

{

// Tests that we expect to return false.

string[] words = { String.Empty, null };

foreach (var word in words)

{

bool result = StringLibrary.StartsWithUpper(word);

Assert.IsFalse(result,

$"Expected for '{(word == null ? "<null>" : word)}': " +

$"false; Actual: {result}");

}

}

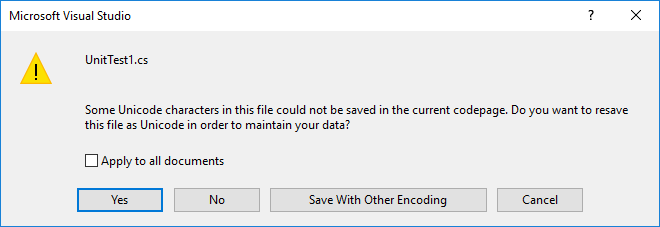
}

}

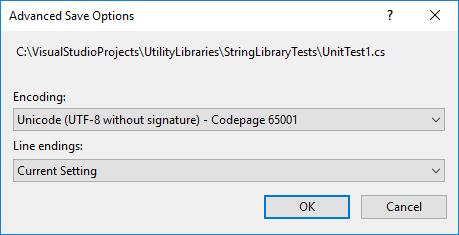
1. Сохраните проект, выбрав значок **Сохранить** на панели инструментов.

Так как код модульного теста включает некоторые отличные от ASCII символы, появится указанное ниже диалоговое окно с предупреждением о том, что при сохранении файла в формате ASCII по умолчанию некоторые символы будут потеряны.

1. Нажмите кнопку **Сохранить в другой кодировке**.



1. В раскрывающемся списке **Кодировка** диалогового окна **Дополнительные параметры сохранения** выберите параметр **Юникод (UTF-8, без сигнатуры), кодовая страница 65001**, как показано на следующей иллюстрации:



1. Скомпилируйте проект модульного теста, выбрав **Сборка**>**Перестроить решение** в меню верхнего уровня Visual Studio.

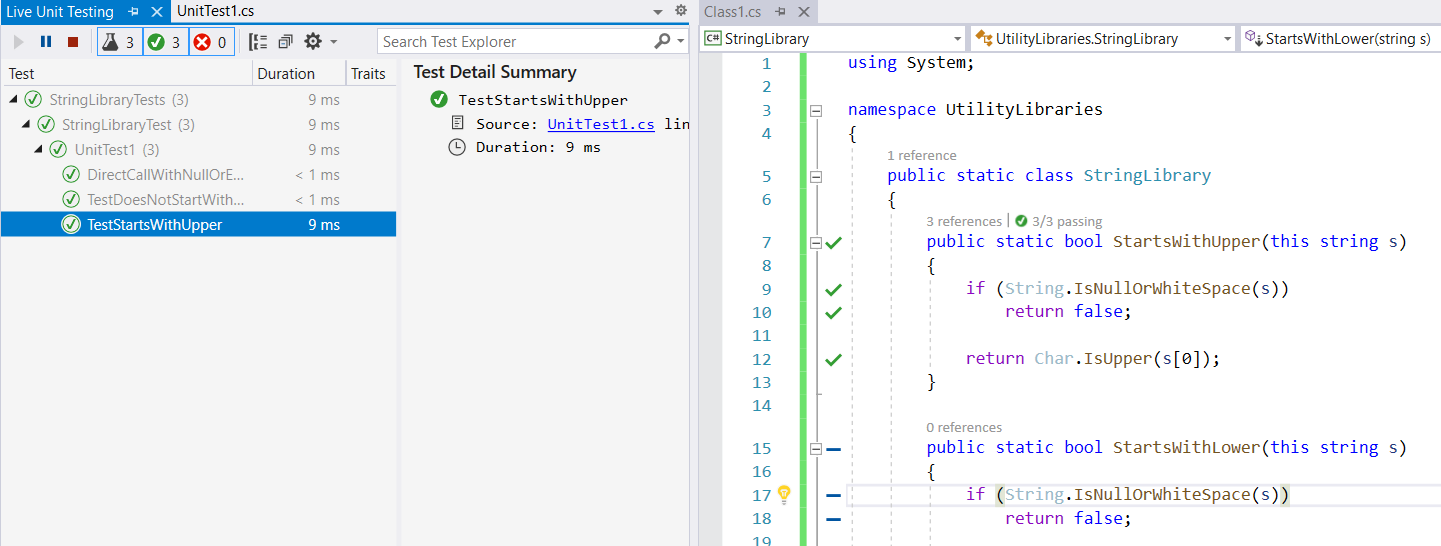
Вы создали библиотеку классов, а также несколько модульных тестов для нее. Вы завершили подготовку к использованию Live Unit Testing.

## Включение Live Unit Testing

Вы уже написали тесты для библиотеки классов StringLibrary, но еще не запускали их. После включения функция Live Unit Testing выполняет их автоматически. Для этого сделайте следующее:

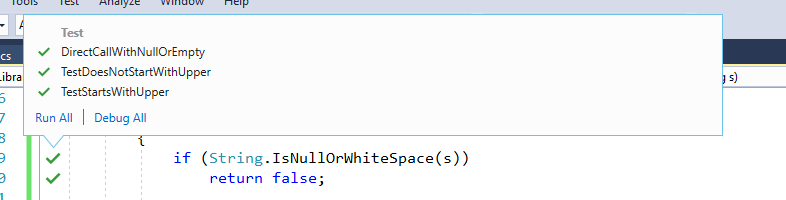
1. При необходимости выберите окно редактора кода, содержащее код для StringLibrary. Это либо Class1.cs для проекта C#, либо Class1.vb для проекта Visual Basic. (Этот шаг позволяет визуально оценить результат тестов и объем протестированного кода после включения Live Unit Testing.)
2. В меню верхнего уровня Visual Studio выберите **Тест**>**Live Unit Testing**>**Запустить**.
3. Проверьте конфигурацию Live Unit Testing, убедившись, что корневой каталог репозитория содержит путь к исходным файлам для проекта служебной программы и тестового проекта. Выберите пункт **Далее**, затем — **Готово**.
4. Visual Studio выполнить повторное создание проекта и запустит Live Unit Test, который автоматически выполняет все тесты.

По окончании выполнения **Live Unit Testing** отображает как общие результаты, так и результаты отдельных тестов. Кроме того, в окне редактора кода выводится графическое представление для результатов тестов и объема протестированного кода. Как показано на следующей иллюстрации, все три теста прошли успешно. Кроме того, видно, что наши тесты охватили все ветви кода в методе StartsWithUpper и все эти тесты были выполнены успешно (на что указывает зеленая галочка "✓"). Наконец, показано, что ни один из других методов в StringLibrary не имеет объем протестированного кода (на что указывает синяя линия "➖").

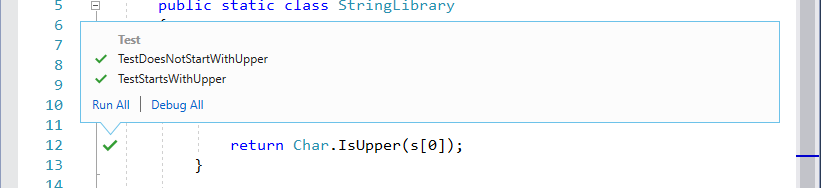


Вы также можете получить более подробные сведения о покрытии и результатах тестов, щелкнув отдельный значок объема протестированного кода в окне редактора кода. Чтобы изучить эту информацию, сделайте следующее:

1. Щелкните зеленую галочку в строке, где if (String.IsNullOrWhiteSpace(s)) считывается в методе StartsWithUpper. Как показано на следующей иллюстрации, Live Unit Testing указывает, что эту строку кода охватили три теста и все они были выполнены успешно.



1. Щелкните зеленую галочку в строке, где return Char.IsUpper(s[0]) считывается в методе StartsWithUpper. Как показано на следующей иллюстрации, Live Unit Testing указывает, что эту строку кода охватили только два теста и все они были выполнены успешно.



Основной поднятый Live Unit Testing вопрос — это неполный объем протестированного кода. Ему и посвящен следующий раздел.

## Увеличение объема протестированного кода

В этом разделе вы расширите область действия модульных тестов, чтобы они захватывали метод StartsWithLower. При этом Live Unit Testing продолжит динамически тестировать ваш код.

Чтобы увеличить объем протестированного кода, захватив метод StartsWithLower, сделайте следующее:

1. Добавьте следующие методы TestStartsWithLower и TestDoesNotStartWithLower в файл исходного кода теста проекта:

C#Копировать

// Code to add to UnitTest1.cs

[TestMethod]

public void TestStartsWithLower()

{

// Tests that we expect to return true.

string[] words = { "alphabet", "zebra", "abc", "αυτοκινητοβιομηχανία", "государство" };

foreach (var word in words)

{

bool result = word.StartsWithLower();

Assert.IsTrue(result,

$"Expected for '{word}': true; Actual: {result}");

}

}

[TestMethod]

public void TestDoesNotStartWithLower()

{

// Tests that we expect to return false.

string[] words = { "Alphabet", "Zebra", "ABC", "Αθήνα", "Москва",

"1234", ".", ";", " "};

foreach (var word in words)

{

bool result = word.StartsWithLower();

Assert.IsFalse(result,

$"Expected for '{word}': false; Actual: {result}");

}

}

1. Измените метод DirectCallWithNullOrEmpty, добавив следующий код сразу после вызова метода Microsoft.VisualStudio.TestTools.UnitTesting.Assert.IsFalse.

C#Копировать

// Code to add to UnitTest1.cs

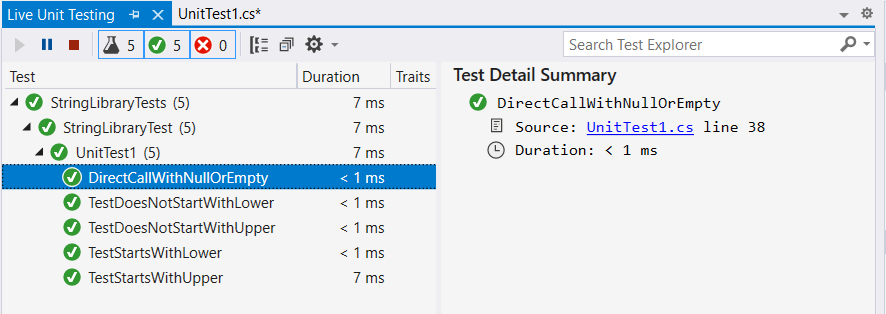
result = StringLibrary.StartsWithLower(word);

Assert.IsFalse(result,

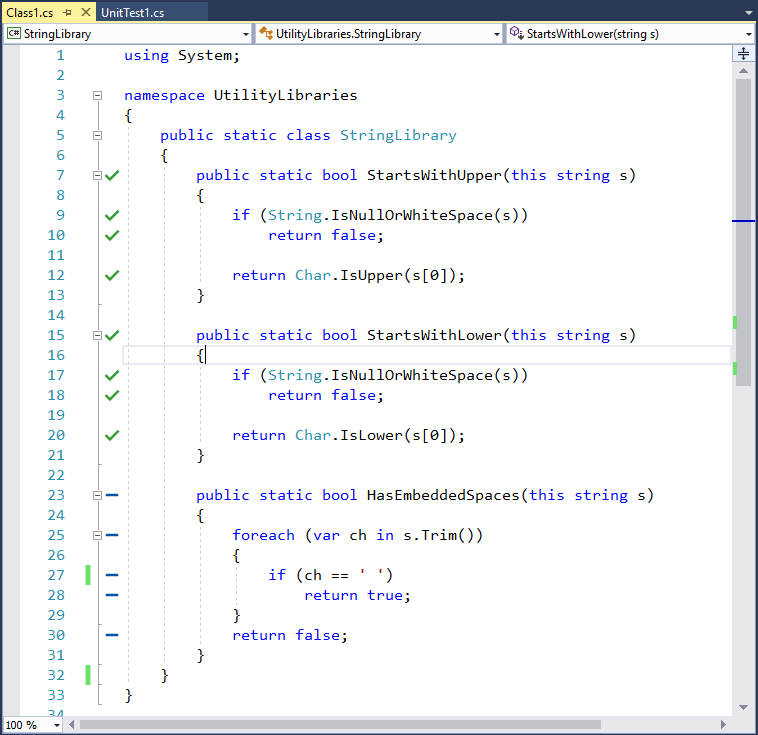
$"Expected for '{(word == null ? "<null>" : word)}': " +

$"false; Actual: {result}");

1. Live Unit Testing автоматически выполняет новые и измененные тесты, когда вы изменяете исходный код. Как показано на следующей иллюстрации, все тесты, включая два добавленных и один измененный, были выполнены успешно.



1. Перейдите в окно, содержащее исходный код для класса StringLibrary. Live Unit Testing показывает, что объем протестированного кода расширен и распространяется на метод StartsWithLower.



В некоторых случаях успешные тесты в **обозревателе тестов** могут быть выделены серым. Это означает, что тест выполняется прямо сейчас либо не запускался повторно из-за отсутствия изменений в коде, которые повлияли бы на тест, со времени последнего его выполнения.

Пока что все наши тесты прошли успешно. В следующем разделе мы рассмотрим, что делать в случае сбоя теста.

## Обработка сбоя при тесте

В этом разделе вы узнаете, как использовать Live Unit Testing для идентификации, диагностики и устранения сбоев тестов. Для этого вы расширите объем протестированного кода, захватив метод HasEmbeddedSpaces.

1. Добавьте следующий метод в файл теста:

C#Копировать

[TestMethod]

public void TestHasEmbeddedSpaces()

{

// Tests that we expect to return true.

string[] phrases = { "one car", "Name\u0009Description",

"Line1\nLine2", "Line3\u000ALine4",

"Line5\u000BLine6", "Line7\u000CLine8",

"Line0009\u000DLine10", "word1\u00A0word2" };

foreach (var phrase in phrases)

{

bool result = phrase.HasEmbeddedSpaces();

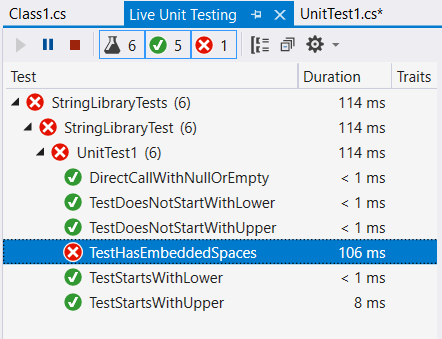
Assert.IsTrue(result,

$"Expected for '{phrase}': true; Actual: {result}");

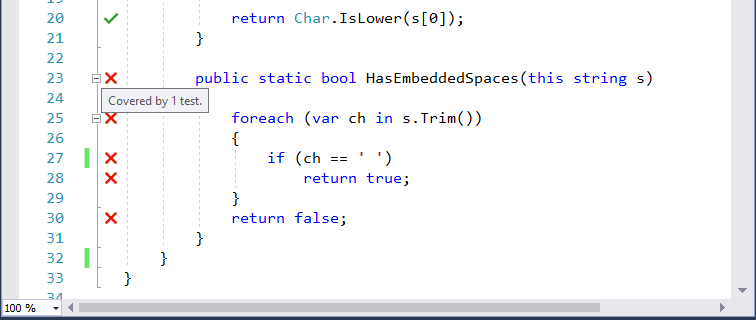
}

}

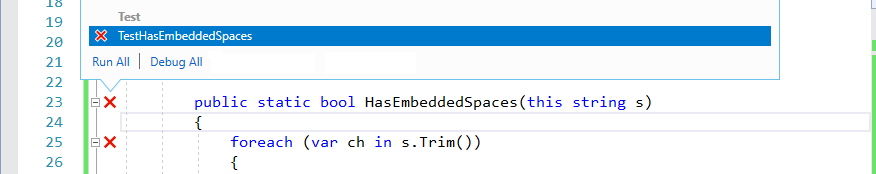
1. При выполнении теста Live Unit Testing указывает, что произошел сбой метода TestHasEmbeddedSpaces, как показано на следующей иллюстрации:



1. Выберите окно, где отображается код библиотеки. Для Live Unit Testing был расширен объем протестированного кода с захватом метода HasEmbeddedSpaces. Эта функция также сообщает о сбое теста, добавляя красные значки "🞩" для строк, охваченных тестами со сбоями.
2. Наведите указатель мыши на строку с сигнатурой метода HasEmbeddedSpaces. Live Unit Testing отображает подсказку, сообщающую, что метод охвачен одним тестом, как показано на следующей иллюстрации:

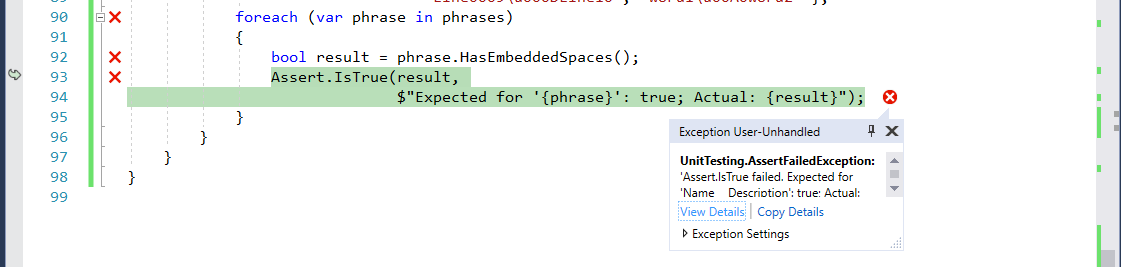


1. Выберите непройденный тест **TestHasEmbeddedSpaces**. Live Unit Testing предлагает вам ряд возможностей, в том числе выполнение всех тестов и отладку всех тестов, как показано на этой иллюстрации:

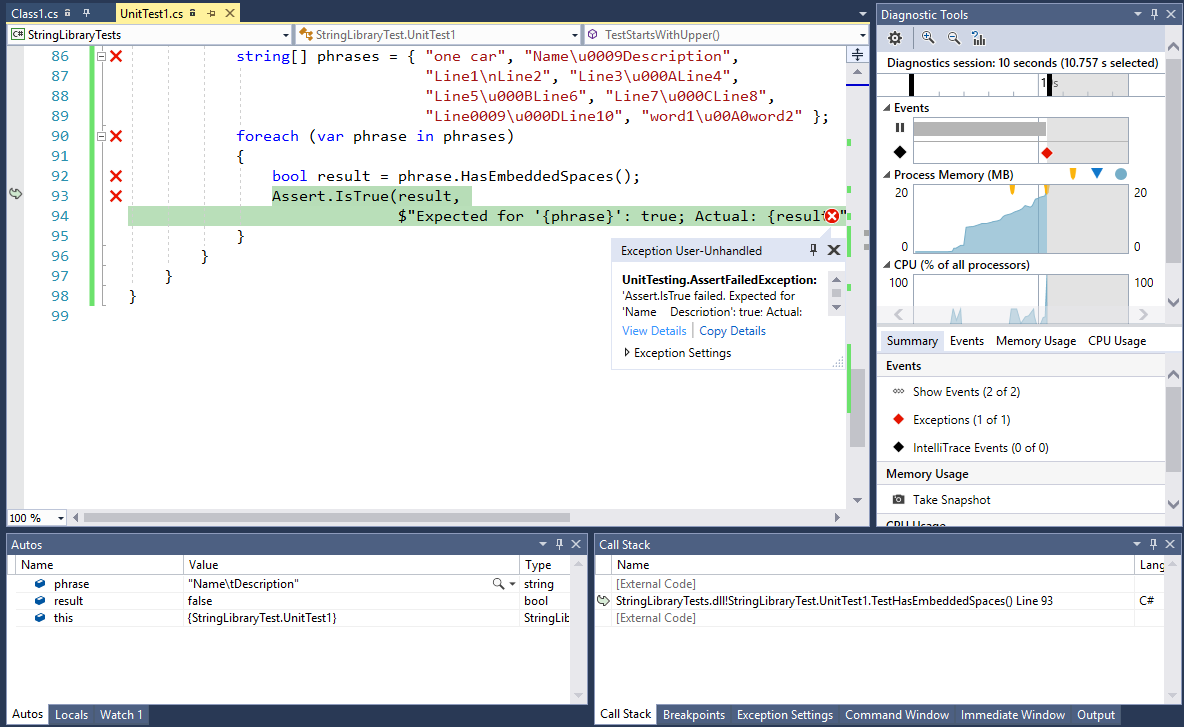


1. Выберите **Отладить все**, чтобы отладить непройденный тест.
2. Visual Studio выполняет тест в режиме отладки.

В тесте каждая строка массива назначается переменной с именем phrase, которая передается в метод HasEmbeddedSpaces. Когда выражение утверждения в первый раз принимает значение false, выполнение программы приостанавливается и вызывается отладчик. Диалоговое окно исключения, полученное в результате непредвиденного значения в вызове метода Microsoft.VisualStudio.TestTools.UnitTesting.Assert.IsTrue, показано на следующей иллюстрации.



Кроме того, для диагностики непройденного теста мы можем использовать все предоставляемые Visual Studio средства отладки, как показано на следующей иллюстрации:



Обратите внимание, что в окне **Видимые** переменная phrase имеет значение "Name\tDescription", что соответствует второму элементу массива. Метод теста ожидает, что HasEmbeddedSpaces возвращает true при передаче данной строки; вместо этого он возвращает false. Очевидно, он не распознает символ табуляции "\t" как внедренный пробел.

1. Выберите **Отладка**>**Продолжить** нажмите клавишу **F5** или кнопку **Продолжить** на панели инструментов, чтобы продолжить выполнение программы тестирования. Так как возникло необработанное исключение, тест был завершен. Это дает достаточно сведений для предварительного исследования ошибки. Либо подпрограмма тестирования TestHasEmbeddedSpaces сделала неверное допущение, либо HasEmbeddedSpaces неправильно распознает все внедренные пробелы.
2. Чтобы выявить и устранить проблему, начните с метода StringLibrary.HasEmbeddedSpaces. Посмотрите на сравнение в методе HasEmbeddedSpaces. В нем внедренный пробел считается равным U+0020. Однако в стандарте Юникод есть ряд других пробелов. Это дает основания полагать, что код библиотеки был неправильно протестирован на пробелы.
3. Замените сравнение на равенство вызовом метода System.Char.IsWhiteSpace:

C#Копировать

if (Char.IsWhiteSpace(ch))

1. Live Unit Testing автоматически перезапускает метод непройденного теста.

Live Unit Testing отображает обновленные результаты, в том числе в окне редактора кода.