کتابخانه ی آزمایش برای .Net Core

در این بخش می خواهیم StringLibrary خود را تست کنیم. برای این کار بایدچیدمان پروژه های خود را به گونه ای مجددا انجام دهیم که بتوانیم از قراردادهای پیش فرض پیروی کنیم.

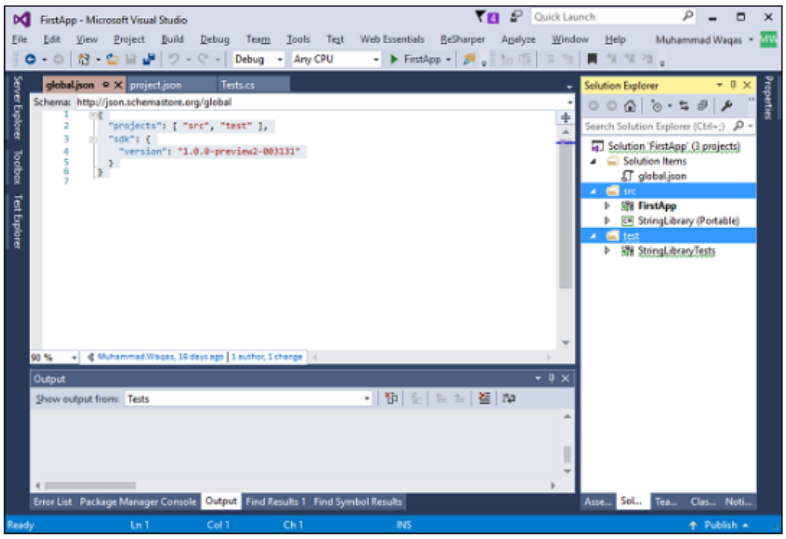
فایل global.json را باز کنید.

```
{
  "projects": [ "src", "test" ],
  "sdk": {
    "version": "1.0.0-preview2-003131"
  }
}
```

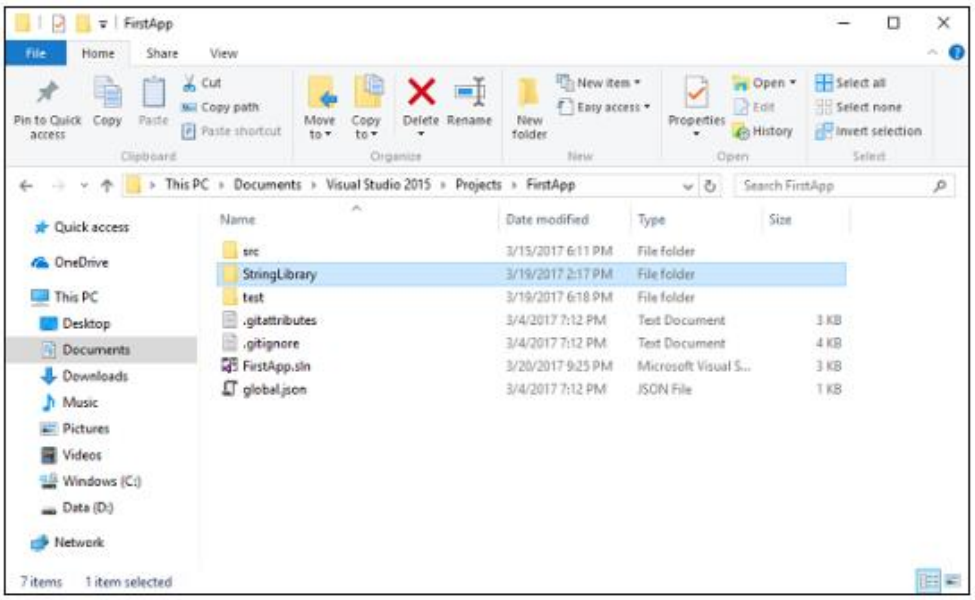
در بالای این فایل می توانید تنظیمات پروژه را ببینید. این تنظیمات برخی از پوشه ها همچون src و test را تنظیم می کنند.

با توجه به قرارداد ما باید در این پوشه ها، پروژه هایی را داشته باشیم. این قرارداد، قرارداد جدیدی است که ما نیز از آن به عنوان بخشی از .NET Core استفاده می کنیم.

در Solution Explorer می توانید ببینید که هم پروژه ی کنسول و هم پروژه ی کتابخانه ای داخل پوشه ی src قرار دارند. این درحالی است که پروژه ی آزمایش داخل پوشه ی test قرار دارد.

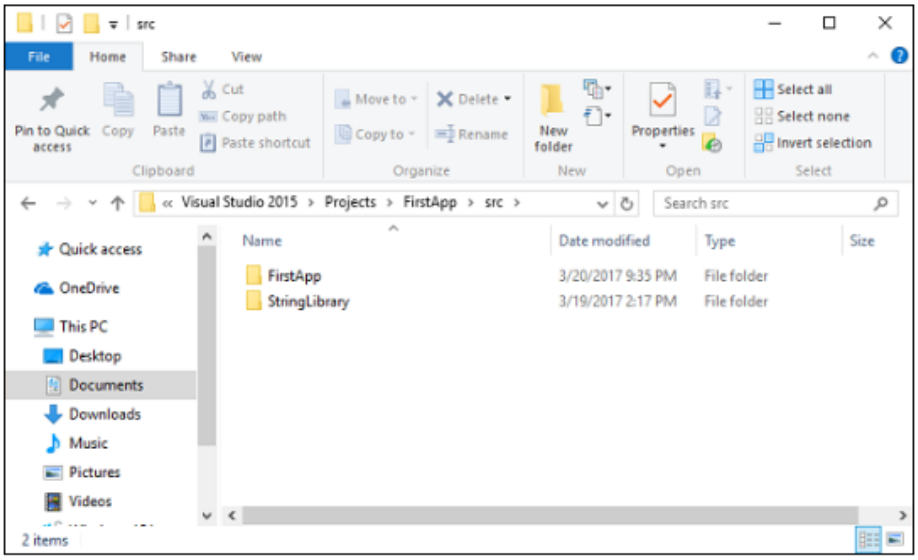


همچنین ساختار پروژه های موجود در Solution Explorer بیانگر مکان فیزیکی پروژه ها در دیسک نیست. پوشه ی Solution را باز کنید تا متوجه شوید که پروژه ی StringLibrary داخل پوشه ی src نیست.

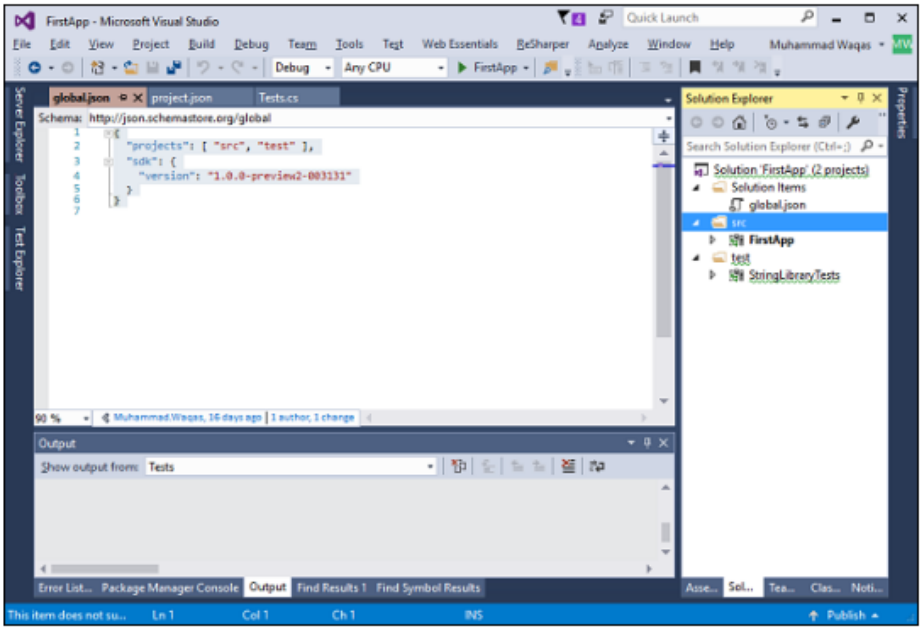


همان طور که می بینید، هر دو پوشه ی src و test داخل قرارداد مشخص شده در فایل global.json قرار می گیرند. با این حال پروژه ی StringLibrary خارج از قرارداد قرار دارد. پروژه ی StringLibrary را داخل پوشه ی src اضافه کنید.

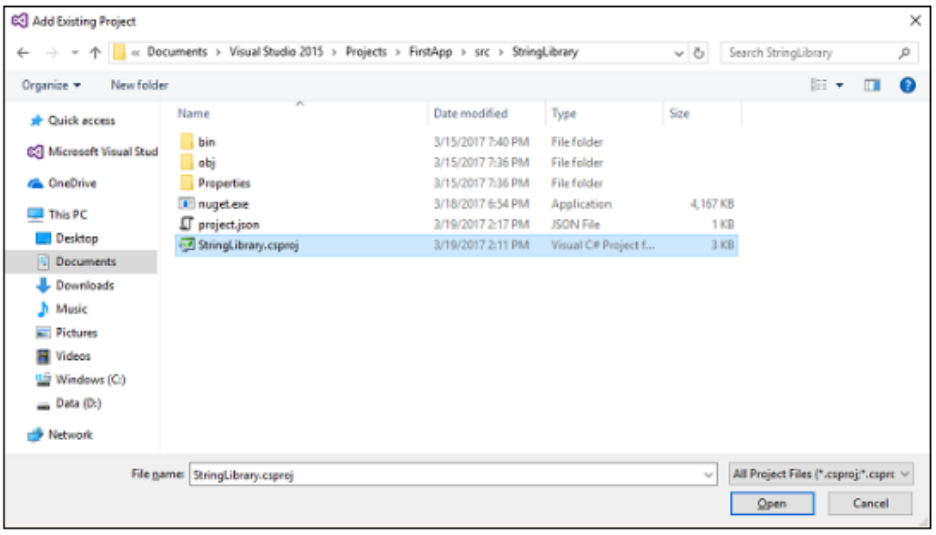
در پوشه ی src ما دو پروژه داریم و باید مشکل موجود را حل کنیم؛ به گونه ای که بتوانیم به درستی از تمامی پروژه ها استفاده کنیم. به ویژوال استودیو برگردید و بر روی پروژه ی StringLibrary کلیک راست کنید و در نهایت گزینه ی Remove را انتخاب کنید. این کار باعث نمی شود که این پروژه حذف شود، بلکه صرفا آن را جا به جا می کند.



حالا بر روی پوشه ی src کلیک راست کنید و Add → Existing Project... را انتخاب کنید.



داخل پروژه ی StringLibrary که در حال حاضر در پوشه ی src قرار دارد، به دنبال فایل StringLibrary.csproj بگردید. آن را انتخاب کنید و بر روی Open کلیک کنید.



حالا باید رفرنس StringLibrary را از فایل project.json موجود در برنامه ی کنسول جابه جا کنید.

```
{
 "version": "1.0.0-*",
 "buildOptions": {
   "emitEntryPoint": true
 },
 "dependencies": {
   "Microsoft.NETCore.App": {
     "type": "platform",
     "version": "1.0.1"
   },
   "NuGet.CommandLine": "3.5.0",
   "System.Runtime.Serialization.Json": "4.0.3"
 },
 "frameworks": {
   "netcoreapp1.0": {
     "dependencies": { },
     "imports": "dnxcore50"
```

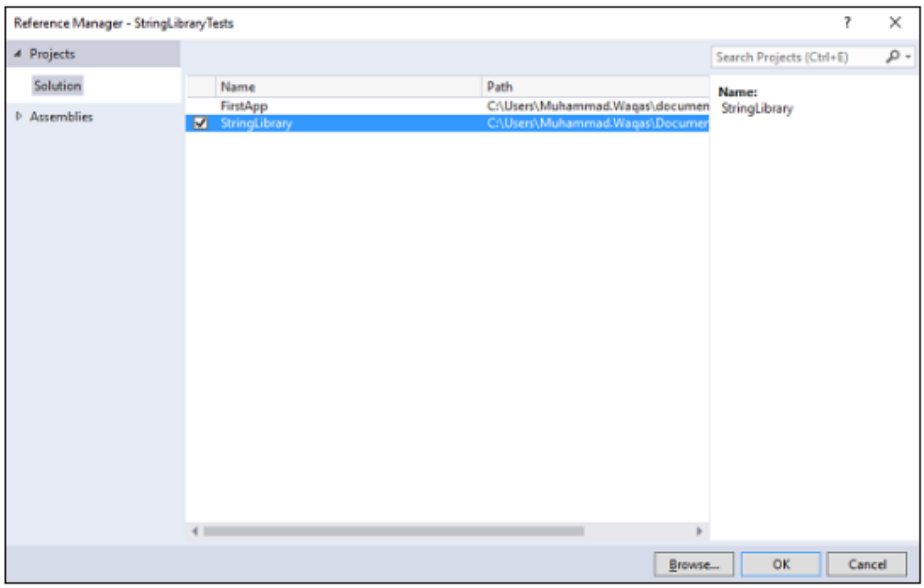
```
    }
  }
}
```

تغییرات را ذخیره کنید و بعد از آن مجددا در پروژه ی کنسول خود رفرنسی از StringLibrary را اضافه کنید.

```
{
  "version": "1.0.0-*",
  "buildOptions": {
    "emitEntryPoint": true
  },
  "dependencies": {
    "Microsoft.NETCore.App": {
      "type": "platform",
      "version": "1.0.1"
    },
  "NuGet.CommandLine": "3.5.0",
    "System.Runtime.Serialization.Json": "4.0.3"
  },
  "frameworks": {
    "netcoreapp1.0": {
      "dependencies": {
        "StringLibrary": {
          "target": "project"
        }
      },
      "imports": "dnxcore50"
    }
  }
```

```
}
```

بعد از این کار تمامی اجزا باید بتوانند مجددا کار کنند و شما نیز باید بتوانید ابتدا StringLibrary و سپس FirstApp (پروژه ی کنسول) را بدون هیچ خطایی ایجاد کنید. حالا بیایید با استفاده از xunit عملکرد StringLibrary را تست کنیم. برای این کار باید رفرنس StringLibrary را داخل پروژه ی آزمایش خود اضافه کنیم. بر روی رفرنس های پروژه ی StringLibraryTests کلیک راست کنید و ...Add Reference را انتخاب کنید.

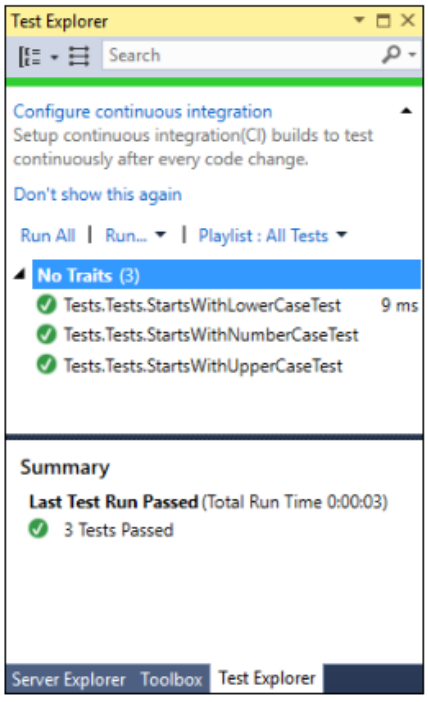


بر روی OK کلیک کنید تا رفرنسی از StringLibrary به پروژه ی آزمایش ما اضافه شود. حالا کد زیر را در فایل Tests.cs جایگذاری کنید.

```
using System;

using Xunit;

using StringLibrary;


namespace Tests {

  public class Tests {

    [Fact]

    public void StartsWithUpperCaseTest() {

      string input = "Mark";
```

```
    Assert.True(input.StartsWithUpper());
  }
  [Fact]
  public void StartsWithLowerCaseTest() {
    string input = "mark";
    Assert.True(input.StartsWithLower());
  }
  [Fact]
  public void StartsWithNumberCaseTest() {
    string input = "123";
    Assert.True(input.StartsWithNumber());
  }
 }
}
```

همان طور که می بینید ما در حال حاضر سه متد تست داریم که عملکرد StringLibrary را آزمایش می کنند. بر روی لینک Run All کلیک کنید تا خروجی زیر در Test Explorer نمایش داده شود.



علاوه بر این می توانید این تست ها را از طریق خط فرمان اجرا کنید. cmd را باز کنید و دستور dotnet test را اجرا کنید.