بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

# تست واحد (unit testing) ASP.NET Web API 2

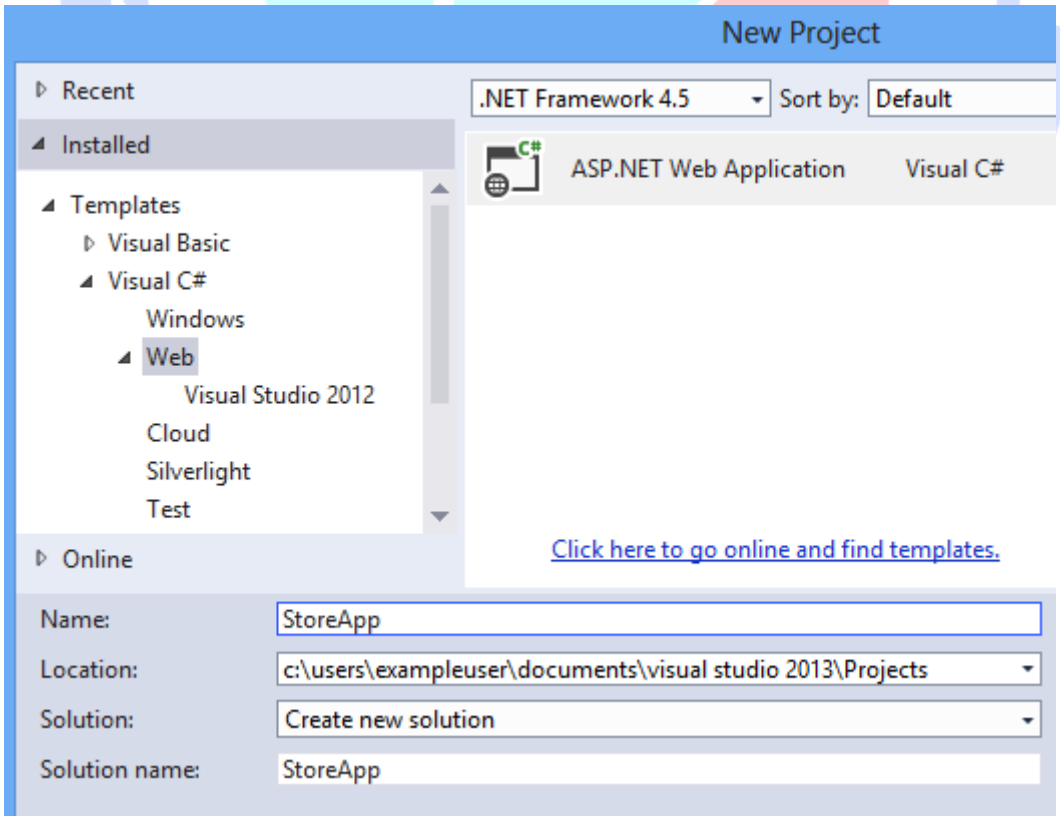
مدرس : مهندس افشین رفوآ

دوره آموزشی Web API

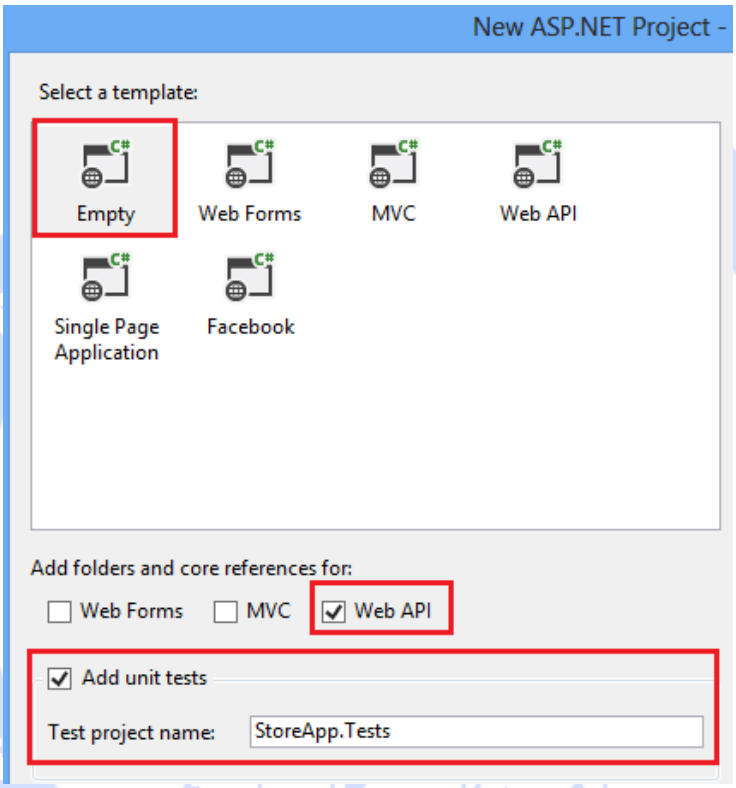
در این مقاله می خواهیم نشان دهیم چگونه به پروژه خودتان یک unit test اضافه کنید و متدهای تست که مقادیر بازگشتی از یک Controller را بر می گرداند بنویسید.

## ساختن یک برنامه با unit test

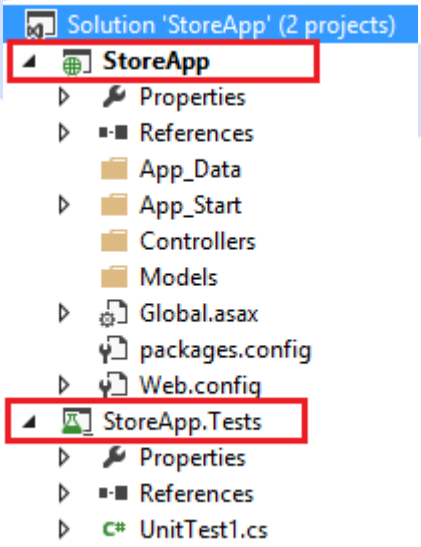
یک برنامه Asp.net به نام StoreApp بسازید.



در پنجره Asp.net روی Empty کلیک کنید و فولدر و reference های Web API را اضافه کنید. گزینه Add unit tests را انتخاب کنید. به صورت اتوماتیک نام StoredApp.Tests برای unit test انتخاب می شود که می توانید از همین نام استفاده کنید.

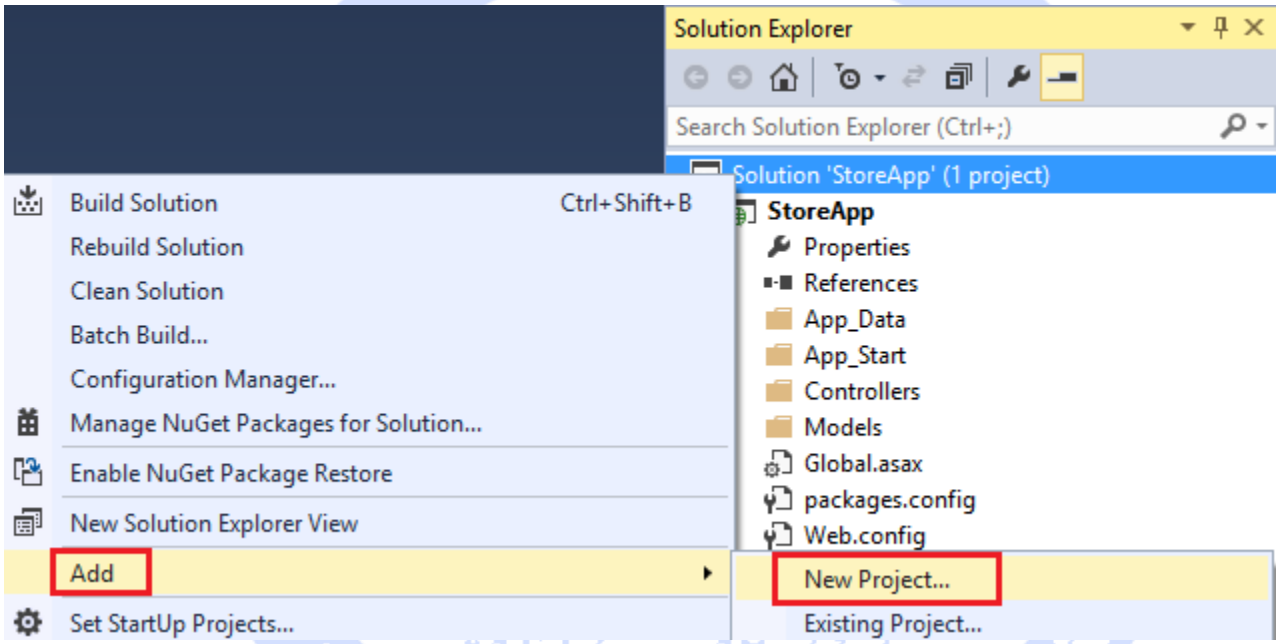



بعد از ساخته شدن برنامه ، دو پروژه برای شما نشان داده می شود:

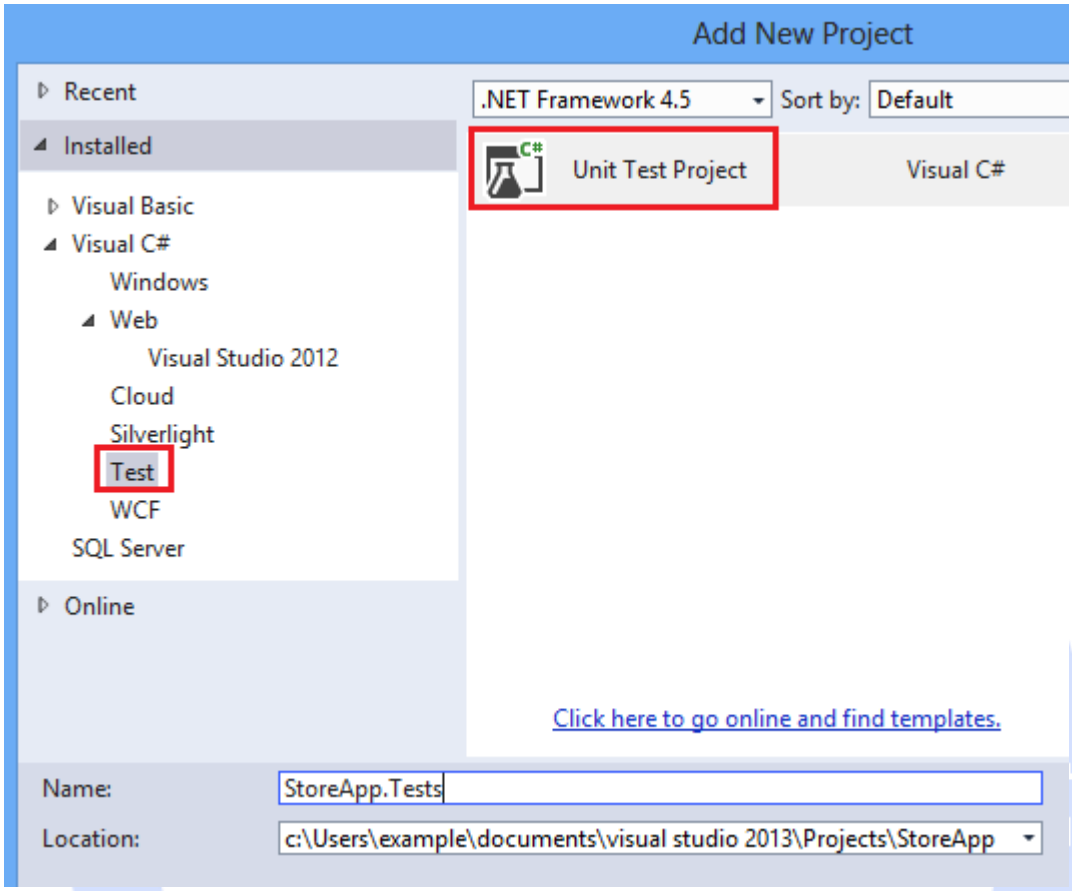


## اضافه کردن unit test به برنامه ی از قبل ساخته شده

اگر شما در زمان نوشتن یک برنامه unit test را اضافه نکرده باشید می توانید در هر زمان دیگری آن را اضافه کنید. برای مثال، فرض کنید در حال حاضر برنامه ای به نام StroeApp دارید و می خواهید unit test را به آن اضافه کنید. برای اضافه کردن unit test روی solution راست کلیک کنید و Add را انتخاب و New Project را انتخاب کنید.




در سمت چپ روی Test کلیک کنید و سپس Unit Test Project را انتخاب کنید و نام آن را StoreApp.Tests بگذارید.




حالا شما می توانید unit test را در پروژه خود ببینید. در unit test ، referenceهای آن را به پروژه اصلی اضافه کنید.

## تنظیم کردن برنامه ی Web API 2

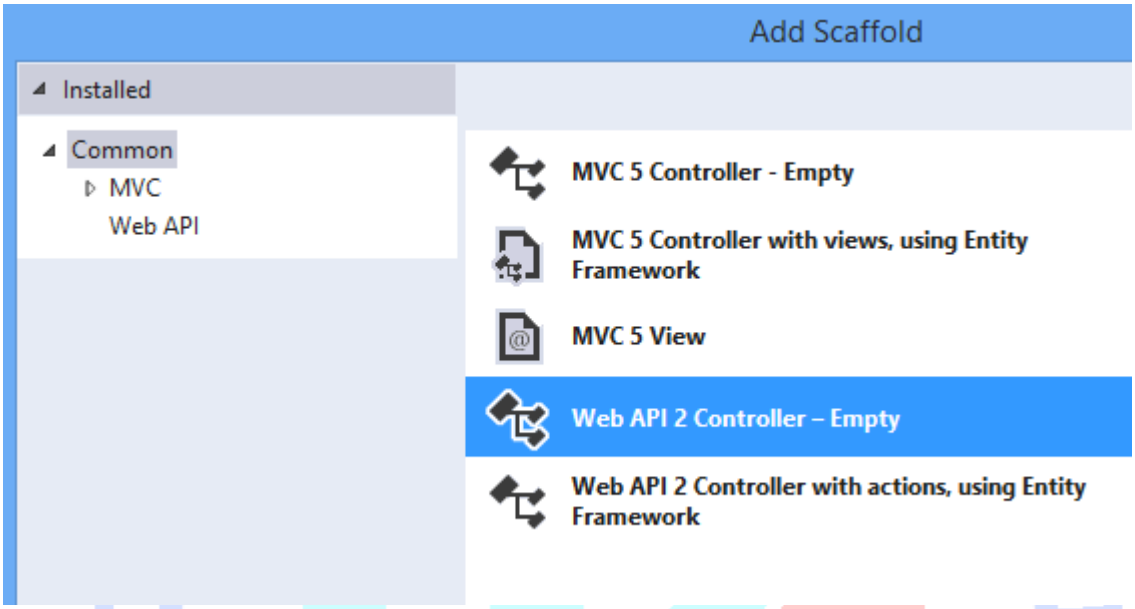
در پروژه StoreApp خودتان، یک کلاس به فولدر Models به اسم Product.cs اضافه کنید و محتوای آن را با کد های زیر عوض کنید.

```
namespace StoreApp.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }
    }
}
```

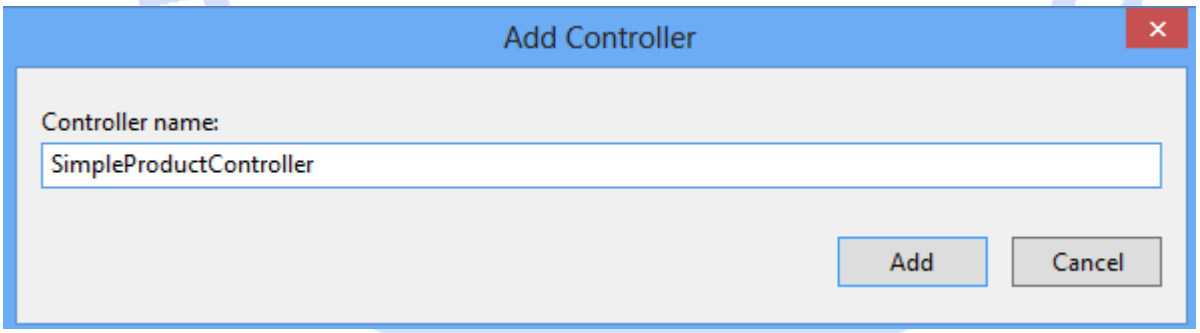
برنامه خود را Build کنید.

روی فولدر Controllers کلیک راست کنید و Add را انتخاب کرده و New Scaffolded Item را انتخاب کنید.

سپس Web API 2 Controllers-Empty را انتخاب کنید.




نام controller را SimpleProductController بگذارید و روی Add کلیک کنید.




کد های درون آن را با کد زیر عوض کنید. برای ساده کردن این مثال، اطلاعات به جای دیتابیس، در یک لیست ذخیره می شود. لیستی که در این کلاس تعریف شده، اطلاعات محصول(production) را نمایش می دهد. توجه داشته باشید که controller شامل یک constructor می باشد که از لیست Product ، یک پارامتر می گیرد. این constructor باعث test اطلاعات توسط unit test می شود. همچنین این controller شامل دو متد async برای نمایش unit test کردن متدهای ناهمسان(asynchronous) می شود. این متد های async برای کم کردن

کد های اضافی با فراخوانی Task.FromResult پیاده سازی می شوند اما معمولا این متدها شامل منابعی فشرده از عملگر ها می باشند.

```
using System;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web.Http;
using StoreApp.Models;
namespace StoreApp.Controllers
{
    public class SimpleProductController : ApiController
    {
        List<Product> products = new List<Product>();
        public SimpleProductController() { }
        public SimpleProductController(List<Product> products)
        {
            this.products = products;
        }
        public IEnumerable<Product> GetAllProducts()
        {
            return products;
        }
        public async Task<IEnumerable<Product>> GetAllProductsAsync()
        {
            return await Task.FromResult(GetAllProducts());
        }
        public IHttpActionResult GetProduct(int id)
        {
            var product = products.FirstOrDefault((p) => p.Id == id);
            if (product == null)
            {
                return NotFound();
            }
            return Ok(product);
        }
        public async Task<IHttpActionResult> GetProductAsync(int id)
        {
            return await Task.FromResult(GetProduct(id));
        }
    }
}
```

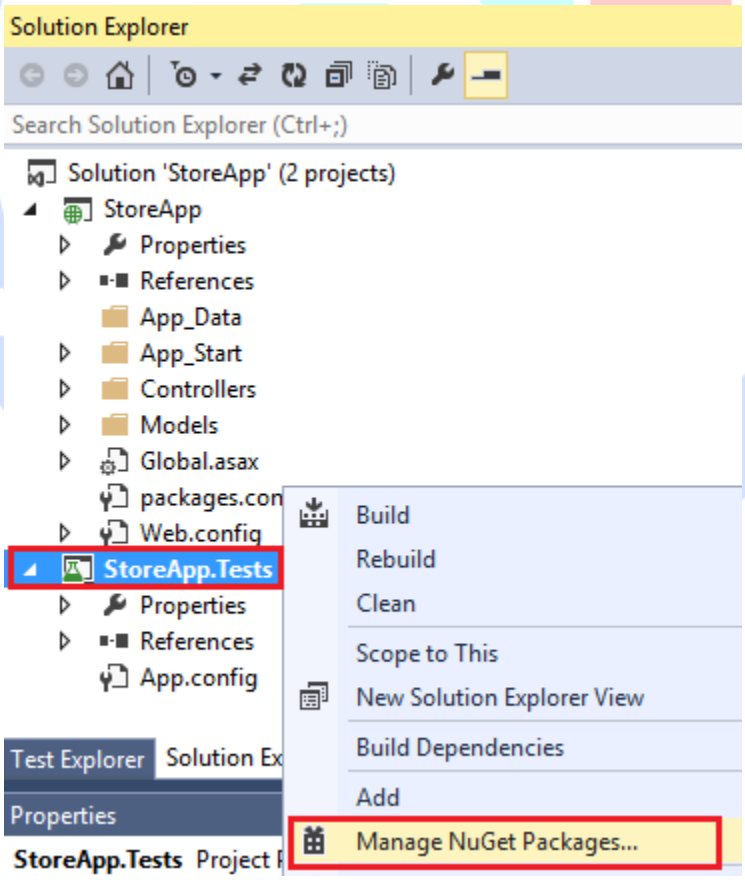
متد GetProduct رابط IHttpActionResult را بر می گرداند. IHttpActionResult یکی از ویژگی های جدید Web API 2 می باشد و unit test را ساده تر کرده است. Namespace مربوط به کلاس هایی که رابط IHttpActionResult را پیاده سازی می کنند، system.Web.Http.Results می باشد. این کلاس ها پاسخ های احتمالی از یک درخواست action را نمایش می دهند و آن را با کد های وضعیت (status code) HTTP تطبیق می دهند.

برنامه را build کنید.

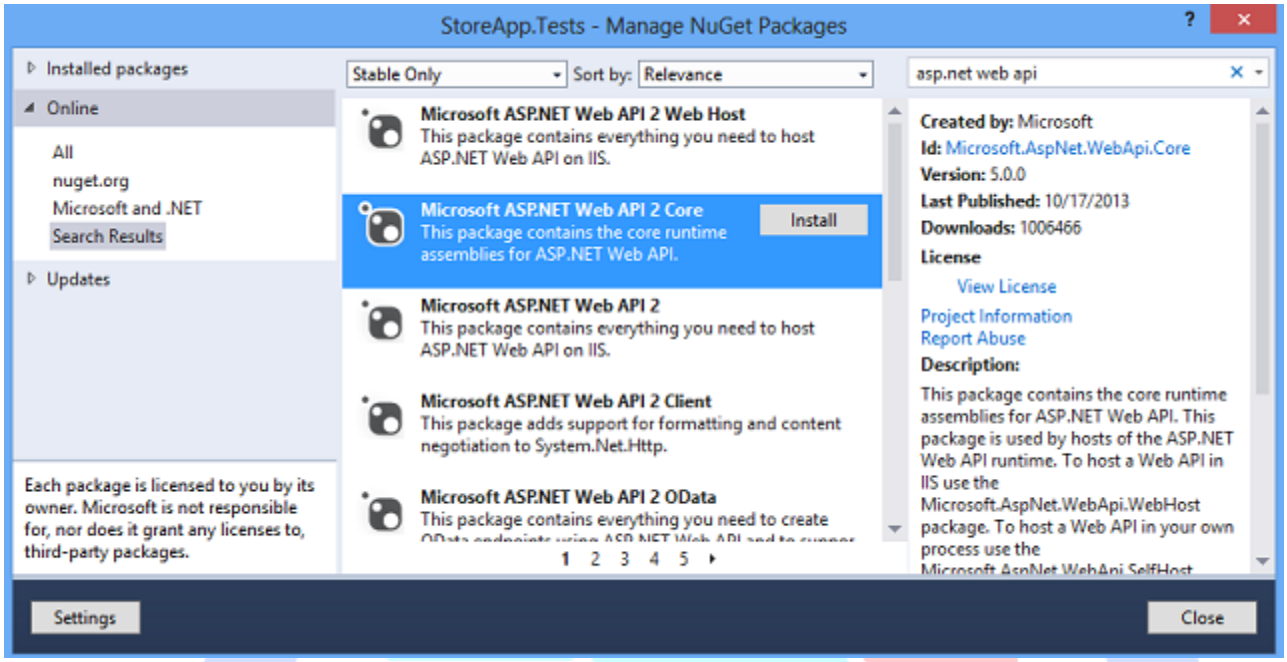
## نصب کردن بسته های NuGet در test

وقتی شما از قالب(template) Empty برای ساختن برنامه خود استفاده می کنید, هیچ کدام از بسته های NuGet برای unit test پروژه (StoreApp.Tests) نصب نشده است. templateهای دیگر، مثل Web API شامل بعضی از بسته های NuGet برای unit test می باشد. برای این مقاله، شما باید بسته Microsoft Web API 2 Core را به پروژه test اضافه کنید.

روی StoreApp.Tests کلیک راست کنید و Manage NuGet Packages را انتخاب کنید. توجه داشته باشید حتما باید StoreApp.Tests را برای اضافه کردن بسته ها به پروژه انتخاب کنید.

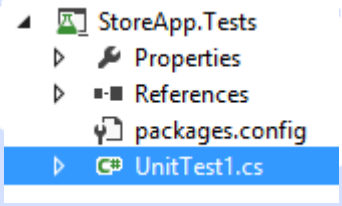



بسته Microsoft Asp.net Web API 2 Core را پیدا کرده و نصب کنید:




ساخت test

به صورت پیش فرض، test project شما شامل یک فایل خالی به نام UnitTest1.cs می باشد. این فایل attribute هایی که برای ساخت متد های test مورد استفاده قرار می گیرد نشان می دهد. برای unit test شما، می توانید فایل خودتان را بسازید یا از آن استفاده کنید.




در این مقاله، شما می توانید کلاس test خودتان را بسازید. می توانید فایل UnitTest1.cs را حذف کنید. یک کلاس به نام **TestSimpleProductController.cs** **اضافه کنید و کد های درون آن را با کد های زیر عوض کنید:**

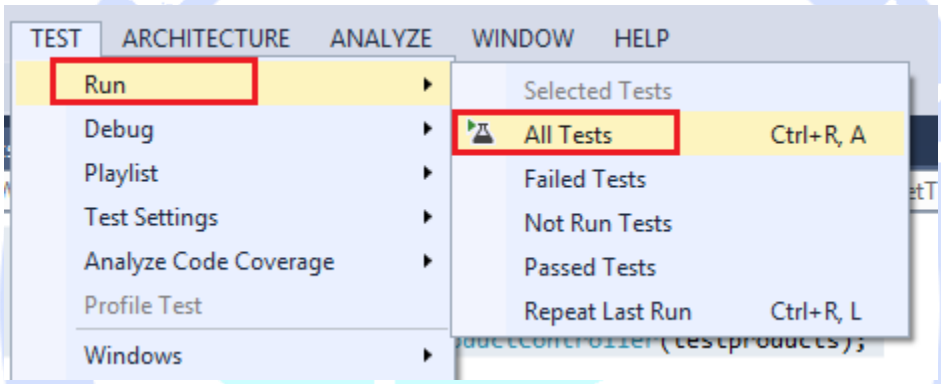
```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Web.Http.Results;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using StoreApp.Controllers;
using StoreApp.Models;
```

```
namespace StoreApp.Tests
{
    [TestClass]
    public class TestSimpleProductController
    {
        [TestMethod]
        public void GetAllProducts_ShouldReturnAllProducts()
        {
            var testProducts = GetTestProducts();
            var controller = new SimpleProductController(testProducts);

            var result = controller.GetAllProducts() as List<Product>;
            Assert.AreEqual(testProducts.Count, result.Count);
        }
        [TestMethod]
        public async Task GetAllProductsAsync_ShouldReturnAllProducts()
        {
            var testProducts = GetTestProducts();
            var controller = new SimpleProductController(testProducts);

            var result = await controller.GetAllProductsAsync() as List<Product>;
            Assert.AreEqual(testProducts.Count, result.Count);
        }
        [TestMethod]
        public void GetProduct_ShouldReturnCorrectProduct()
        {
            var testProducts = GetTestProducts();
            var controller = new SimpleProductController(testProducts);

            var result = controller.GetProduct(4) as OkNegotiatedContentResult<Product>;
            Assert.IsNotNull(result);
            Assert.AreEqual(testProducts[3].Name, result.Content.Name);
        }
        [TestMethod]
        public async Task GetProductAsync_ShouldReturnCorrectProduct()
        {
            var testProducts = GetTestProducts();
            var controller = new SimpleProductController(testProducts);

            var result = await controller.GetProductAsync(4) as
OkNegotiatedContentResult<Product>;
            Assert.IsNotNull(result);
            Assert.AreEqual(testProducts[3].Name, result.Content.Name);
        }
        [TestMethod]
        public void GetProduct_ShouldNotFindProduct()
        {
            var controller = new SimpleProductController(GetTestProducts());

            var result = controller.GetProduct(999);
            Assert.IsInstanceOfType(result, typeof(NotFoundResult));
        }
        private List<Product> GetTestProducts()
        {
            var testProducts = new List<Product>();
            testProducts.Add(new Product { Id = 1, Name = "Demo1", Price = 1 });
            testProducts.Add(new Product { Id = 2, Name = "Demo2", Price = 3.75M });
            testProducts.Add(new Product { Id = 3, Name = "Demo3", Price = 16.99M });
            testProducts.Add(new Product { Id = 4, Name = "Demo4", Price = 11.00M });
            return testProducts;
        }
```

```
    }
}
```

## اجرا کردن test

حالا شما می توانید test ها را اجرا کنید. همه ی متد هایی که با TestMethod attribute مشخص شده اند، آزمایش می شوند. از منو Test می توانید test ها را اجرا کنید:




پنجره Test Explorer را باز کنید و به نتایج بدست آمده از test ها توجه کنید: