Упражнения: Използване на външна библиотека за работа с JSON

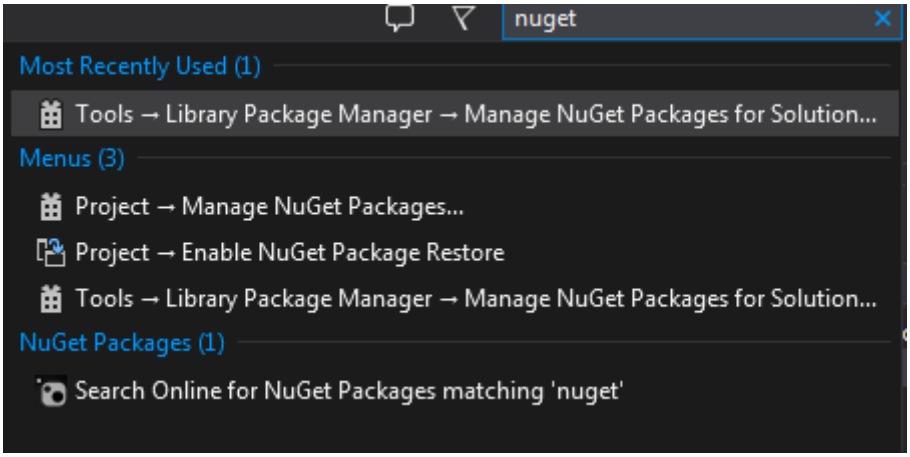
Ще направим малък примерен проект за работа с JSON формата за пренос на данни. За целта създайте проект за конзолно приложение **JsonApp**

1. Инсталиране на Json.NET чрез NuGet

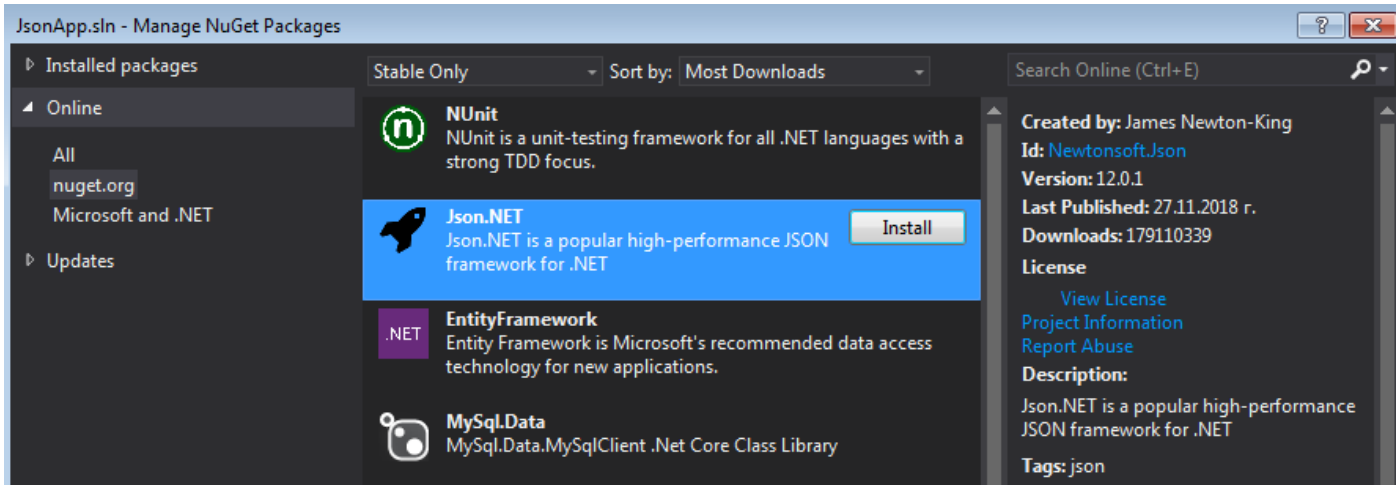
Сега трябва да инсталираме **Json.NET** библиотеката чрез **NuGet**. За целта:

- Натиснете **Ctrl + Q**, за да използвате **Quick Launch**
- Въведете **nuget**

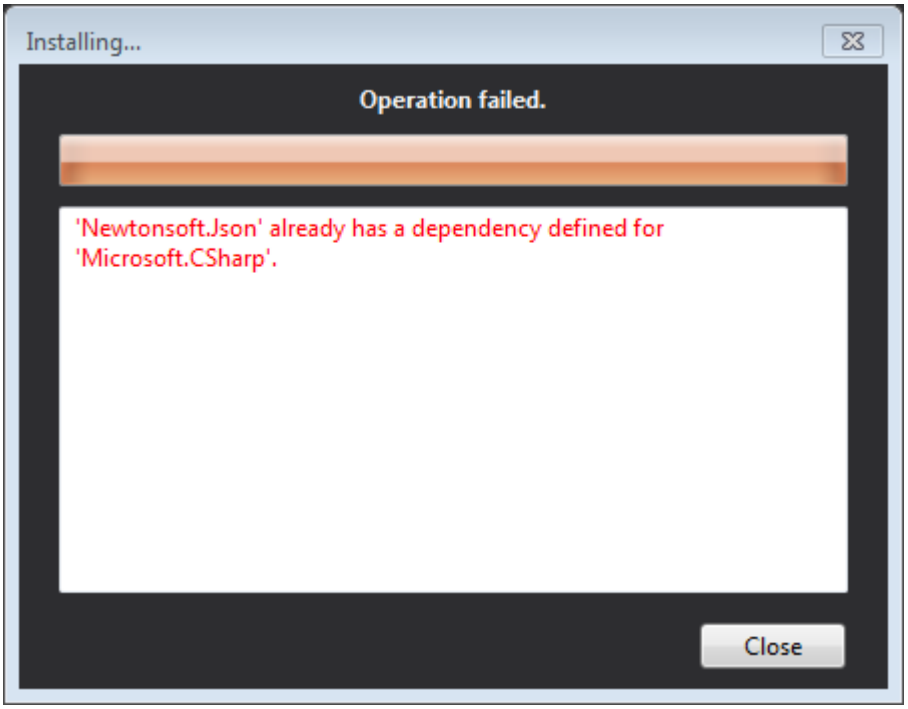
Изберете **Tools -> Library Package Manager -> Manage NuGet Packages for Solution...**



Намерете **Json.NET** и инсталирайте:

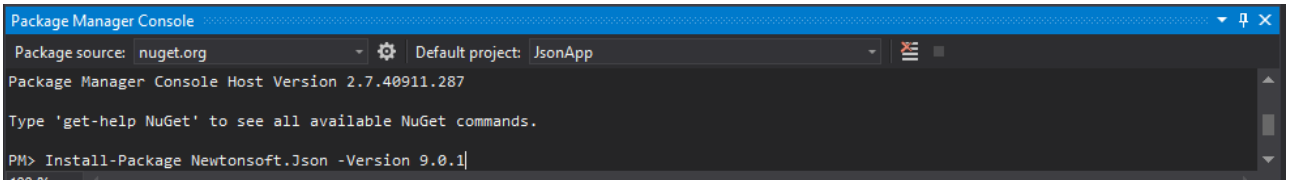


Забележка: Ако използвате по-стара версия на Visual Studio, вероятно ще получите сходна грешка:



Това се случва понеже актуалната версия (към Декември 2018-Януари 2019) изисква по-висока версия на Visual Studio. За нашите цели ще инсталираме по-стара версия, като за избор на версия тук ще демонстрираме инсталация през **конзолата на пакетния мениджър /Package Manager Console/**:

- Достъпете чрез **Ctrl + Q** Quick launch
- Въведете package manager console и я изберете
- В конзолата въведете:
  Install-Package Newtonsoft.Json -Version 9.0.1



2. Създаване на клас Product

Създайте съвсем обикновен клас Product със свойства за:

- Id (**int**) – номер на продукта
- Name (**string**) – име на продукта
- Price (**decimal**) – цена на продукта
- Stock(**int**) – наличност на продукта
- Expiry (**DateTime**) – срок на годност на продукта

3. Преобразуване на Product към JSON и JSON към Product

Ще разгледаме няколко примера за работа с библиотеката.

4. Сериализиране на обект

Първо ще създадем един обект от клас **Product**. След това използвайки библиотеката ще извършим **сериализация** на обекта, т.е. да го превърнем в **JSON** низ. Кодът е както следва:

```
Product product = new Product(1, "Test product", 100.01m, 100, new DateTime(2019, 06, 30));
string json = JsonConvert.SerializeObject(product);
Console.WriteLine("Single product object:");
Console.WriteLine(json);
Console.WriteLine(new string('-', 50));
```

За да може да ползваме класа **JsonConvert**, ще трябва да добавим и `using Newtonsoft.Json;` в началото на **Program.cs**

5. Сериализиране на списък от продукти

По сходен начин можем да реализираме и сериализирането на списък от продукти:

```
List<Product> products = new List<Product>()
{
    new Product(1, "Milk", 2.59m, 100, new DateTime(2019, 06, 30)),
    new Product(2, "Lyutenitsa", 2.39m, 100, new DateTime(2019, 08, 30)),
    new Product(3, "Rice", 1.50m, 100, new DateTime(2020, 03, 30)),
    new Product(4, "Salt", 100.01m, 100, new DateTime(2019, 10, 30)),
};

string jsonList = JsonConvert.SerializeObject(products);
Console.WriteLine("List of products:");
Console.WriteLine(jsonList);
Console.WriteLine(new string('-', 50));
```

6. Десериализиране на JSON

Сега нека да видим как работи и обратния процес, ще десериализираме **JSON** низ, в който е описан масив от размери на дрехи. За целта ще работим с класа **JArray**, за който ще имате нужда също от `using Newtonsoft.Json.Linq;` в горната част на **Program.cs**. Кодът десериализира низа, превръщайки го в масив, на който можем да приложим **foreach** или произволна **LINQ** операция.

```
//Deserialize a list of json data:
string jsonSizes = @"['Small','Medium','Large']";
JArray a = JArray.Parse(jsonSizes);
foreach (var item in a)
{
    Console.WriteLine(item);
}
```

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).