فریمورک توسعه پذیری مدیریت شده (MEF) در .Net Core

در این بخش می خواهیم به بررسی MEF بپردازیم. MEF در پلاگین های توسعه پذیری سوم شخص کاربرد دارد. همچنین به کمک آن می توان در برنامه ها معمولی از مزایای معماری پلاگین گونه ی با همراهی آزادانه بهره برد.

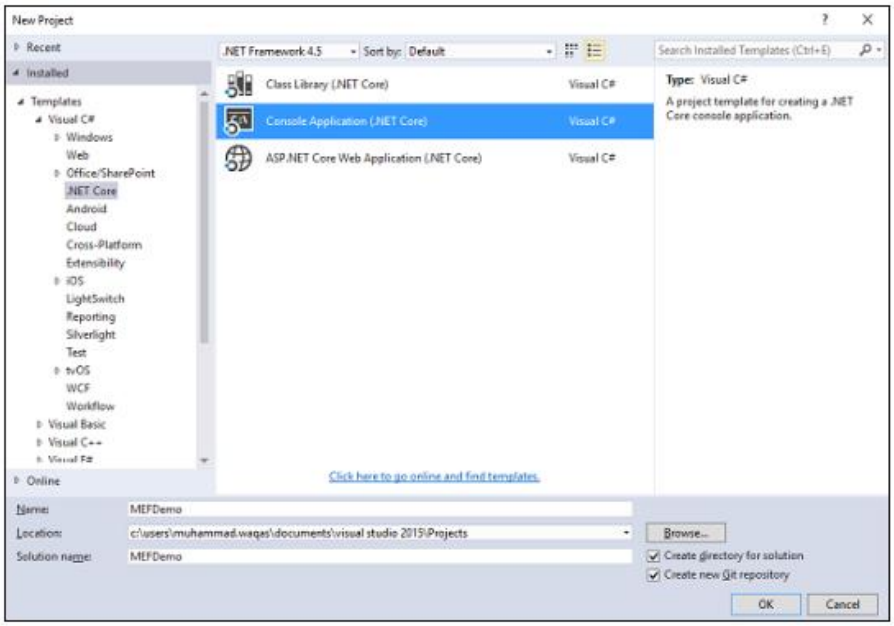
- MEF کتابخانه ای است که می توان به کمک آن برنامه های توسعه پذیر و سبک را ایجاد کرد.
- برنامه نویسان با کمک MEF می توانند بدون نیاز به پیکربندی، افزونه ها را کشف و از آن ها استفاده کنند.
- MEF بخش جدایی ناپذیر .NET Framework 4 است و در هر جایی که از .NET Framework استفاده شود در دسترس است. همچنین MEF انعطاف پذیری، قابلیت نگهداری و آزمون پذیری برنامه های بزرگ را بهبود می بخشد.
- می توانید در برنامه های کلاینت خود از MEF استفاده کنید. تفاوتی ندارد که این برنامه ها از Windows Forms، WPF و یا فناوری دیگری استفاده کنند. حتی می توانید در برنامه های سرور که از ASP.NET بهره می برند نیز از آن استفاده کنید.
- بخشی از MEF به صورت Microsoft.Composition به .NET Core وارد شده است.
- تنها System.Composition به این بخش وارد شده است و دیگر خبری از System.ComponentModel.Composition نیست. این یعنی ما دیگر کاتالوگ هایی در اختیار نداریم که بتوانند نوع ها را از اسمبلی های موجود در یک دایرکتوری بارگیری کنند.

در این بخش ما تنها به چگونگی استفاده از MEF در برنامه ها .NET Core می پردازیم.

بیایید به مثال ساده ای بپردازیم که در آن در برنامه ی کنسول .NET CORE از MEF استفاده شده باشد. بنابراین یک پروژه ی کنسول .NET Core جدید را ایجاد کنید.

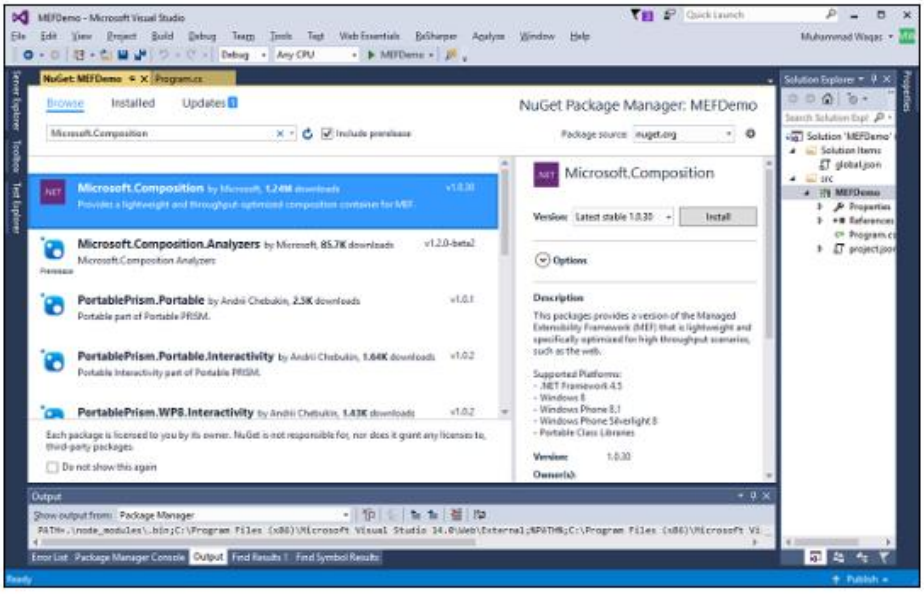
در سمت چپ پنجره .NET Core → Visual C# → Templates را انتخاب کنید و سپس در بخش میانی صفحه Console Application (.NET Core) را انتخاب کنید.

در بخش نام اسم پروژه را وارد کنید و بر روی OK کلیک کنید.

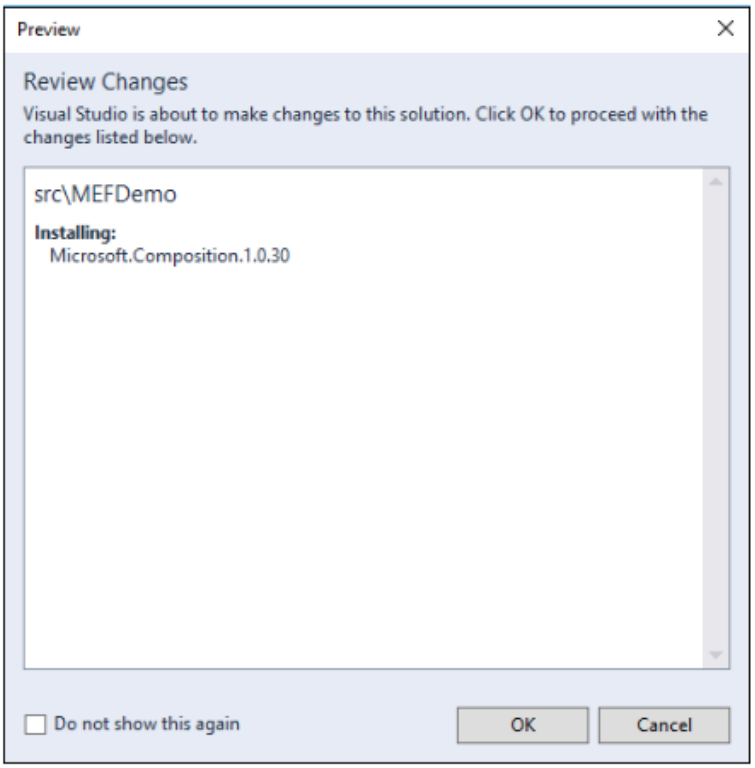


پس از ایجاد شدن این پروژه ما باید رفرنس Microsoft.Composition را اضافه کنیم، به گونه ای که بتوانیم از MEF استفاده کنیم. برای انجام این کار بر روی پروژه ی موجود در Solution Explorer و Manage NuGet Packages... کلیک راست کنید.

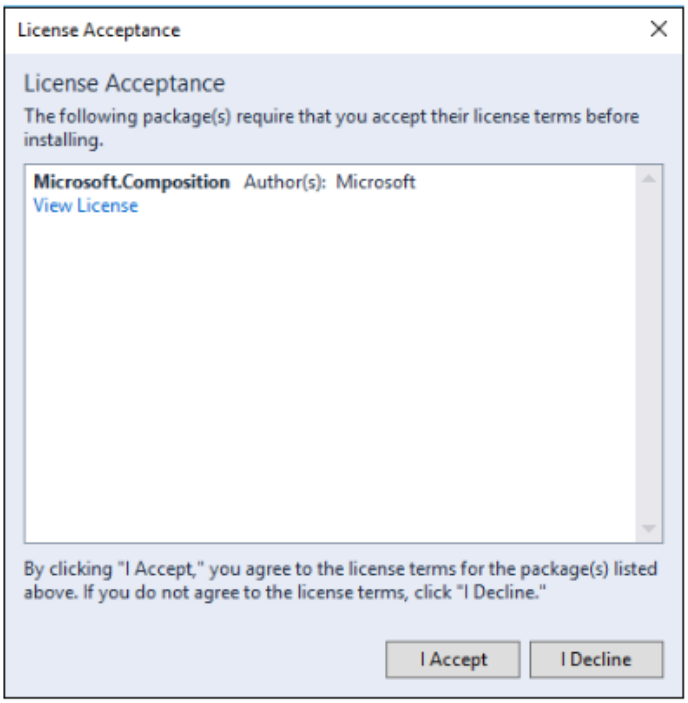
Microsoft.Composition را جستجو کنید و بر روی Install کلیک کنید.



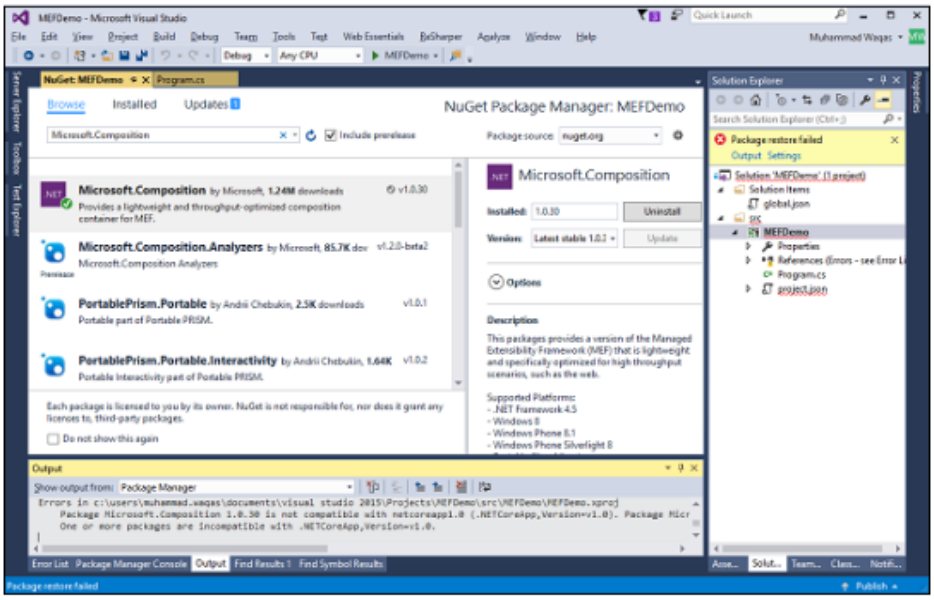
بر روی OK کلیک کنید.



Accept ارا بزنید.



بعد از تمام شدن فرآیند نصب، در رفرنس ها با خطا مواجه می شوید.



فایل project.json را باز کنید.

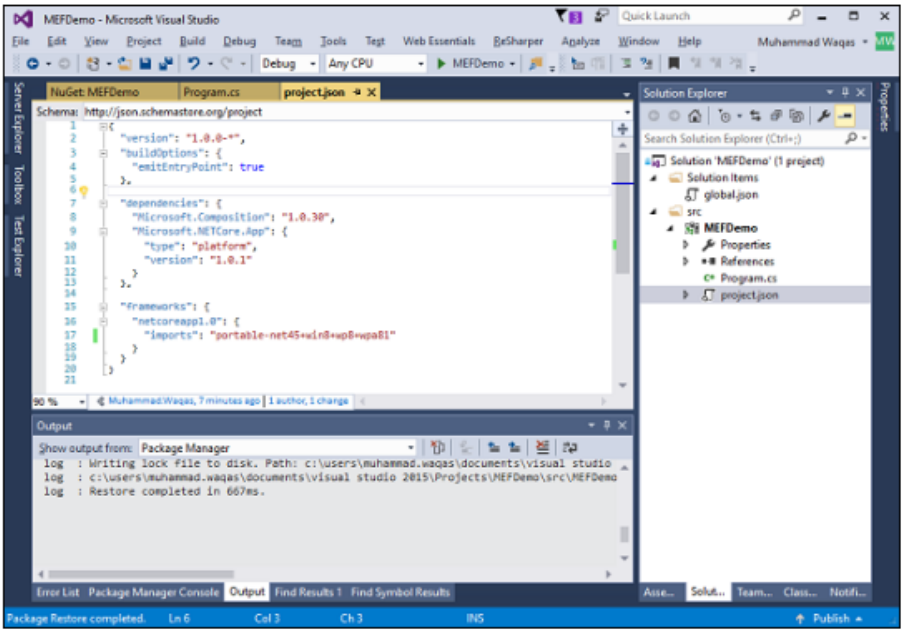
```
{
 "version": "1.0.0-*",
 "buildOptions": {
   "emitEntryPoint": true
 },

 "dependencies": {
   "Microsoft.Composition": "1.0.30",
   "Microsoft.NETCore.App": {
     "type": "platform",
     "version": "1.0.1"
   }
 },

 "frameworks": {
```

```
    "netcoreapp1.0": {
      "imports": "dnxcore50"
    }
  }
}
```

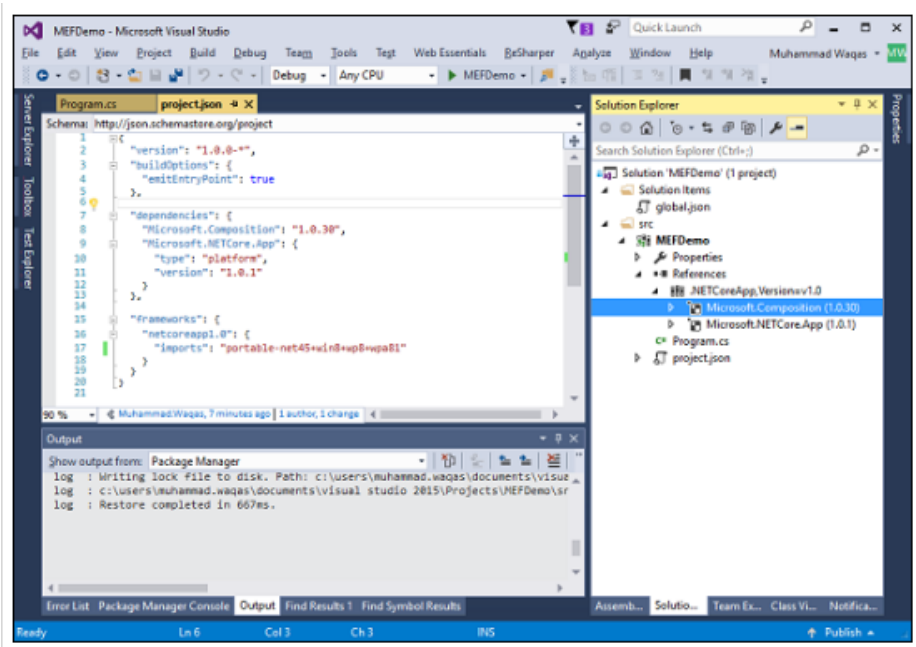
بعد از انجام این کار وابستگی Microsoft.Composition اضافه می شود، اما مشکل اینجاست که این بسته با dnxcore50 سازگاری ندارد. بنابراین ما باید portablenet45+win8+wp8+wpa81 را وارد کنیم. جای فایل project.json را با کد زیر عوض کنید.

```
{
  "version": "1.0.0-*",
  "buildOptions": {
    "emitEntryPoint": true
  },
  "dependencies": {
    "Microsoft.Composition": "1.0.30",
    "Microsoft.NETCore.App": {
      "type": "platform",
      "version": "1.0.1"
    }
  },
  "frameworks": {
    "netcoreapp1.0": {
      "imports": "portable-net45+win8+wp8+wpa81"
    }
  }
}
```

این فایل را ذخیره کنید تا این خطا برطرف شود.

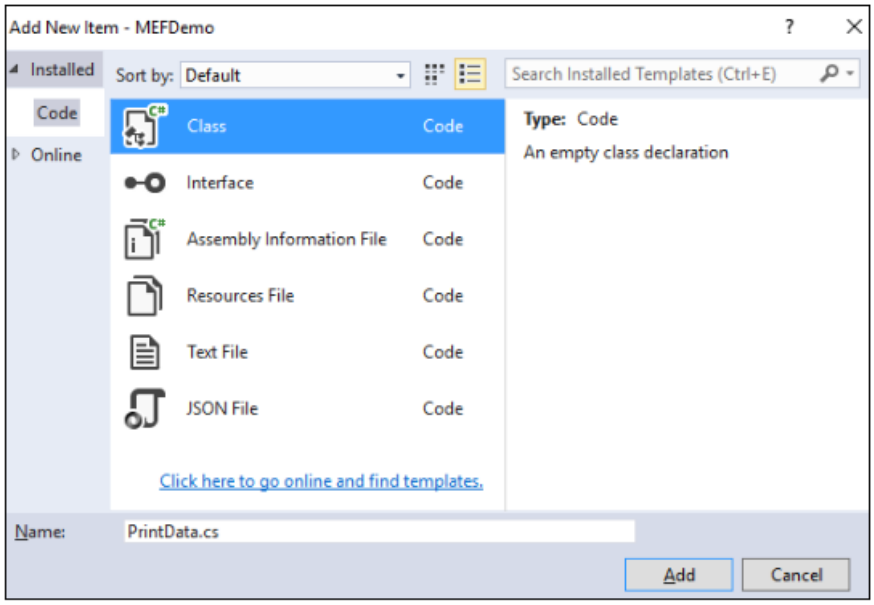


اگر رفرنس ها را گسترش دهید، می توانید رفرنسی از Microsoft.Composition را ببینید.



ابتدا ما باید رابطی ایجاد کنیم که بعدا صادر شود و باید این رابط را پیاده سازی کنیم و کلاس را به همراه صفت export آذین کنیم. کلاس جدیدی را اضافه کنید.

در بخش نام، اسم کلاس خود را وارد کنید و بر روی Add کلیک کنید.



کد زیر را به فایل PrintData.cs اضافه کنید.

```
using System;

using System.Collections.Generic;

using System.Composition;

using System.Linq;

using System.Threading.Tasks;


namespace MEFDemo {

  public interface IPrintData {

    void Send(string message);

  }

  [Export(typeof(IPrintData))]

  public class PrintData : IPrintData {

    public void Send(string message) {

      Console.WriteLine(message);
```

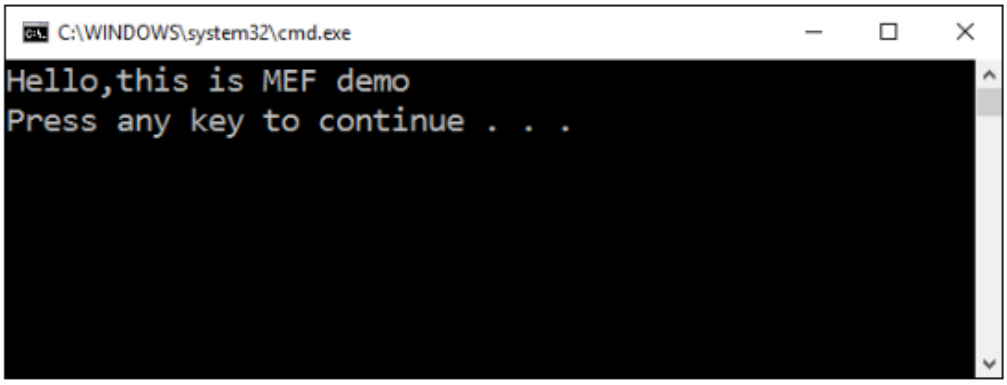
```
    }
  }
}
```

همان طور که در بالا نیز اشاره شد، کاتالوگ ها در فضای Microsoft.Composition موجود نیستند. بنابراین این فضای نام تمامی انواع اسمبلی را به همراه صفت export بارگیری می کند و همان طور که در متد Compose موجود در فایل Program.cs مشهود است، به صفت import پیوند می دهد.

```
using System;
using System.Collections.Generic;
using System.Composition;
using System.Composition.Hosting;
using System.Linq;
using System.Reflection;
using System.Threading.Tasks;


namespace MEFDemo {
  public class Program {
    public static void Main(string[] args) {
      Program p = new Program();
      p.Run();
    }
    public void Run() {
      Compose();
      PrintData.Send("Hello,this is MEF demo");
    }
    [Import]
    public IPrintData PrintData { get; set; }
```

```
    private void Compose() {
      var assemblies = new[] { typeof(Program).GetTypeInfo().Assembly };
      var configuration = new ContainerConfiguration()
        .WithAssembly(typeof(Program).GetTypeInfo().Assembly);

      using (var container = configuration.CreateContainer()) {
        PrintData = container.GetExport<IPrintData>();
      }
    }
  }
}
```

حالا برنامه ی خود را اجرا کنید تا این برنامه با نمونه سازی کلاس PrintData اجرا شود.



برای کسب اطلاعات بیشتر درباره ی MEF می توانید به این لینک مراجعه کنید.

https://msdn.microsoft.com/en-us/library/dd460648%28v=vs.110%29.aspx