

مطابق استاندارد [ECMA-335](http://www.ecma-international.org/335/) قسمت دوم آن، یک اسمبلی از یک یا چند ماژول تشکیل می‌شود. هر ماژول از تعدادی نوع، enum و delegate تشکیل خواهد شد و هر نوع دارای تعدادی متد، فیلد، خاصیت و غیره می‌باشد. به همین جهت در حین کار با Reflection.Emit نیز این مراحل رعایت می‌شوند. ابتدا یک اسمبلی (AppDomain.DefineDynamicAssembly) ایجاد خواهد شد (یا از اسمبلی موجود استفاده می‌شود). سپس یک ماژول (AssemblyBuilder.DefineDynamicModule) را باید به آن اضافه کنیم (یا از ماژول اسمبلی جاری استفاده نمائیم). در ادامه یک Type باید به این ماژول اضافه شود (ModuleBuilder.DefineType) و اکنون می‌توان به این نوع جدید، سازنده (TypeBuilder.DefineConstructor)، متد (TypeBuilder.DefineMethod)، فیلد (TypeBuilder.DefineField)، خاصیت (TypeBuilder.DefineProperty) و رخداد (TypeBuilder.DefineEvent) اضافه کرد.

Assembly

Module

Type

Methods

Properties

Fields

Events

Enum

Delegate

```
using System;
using System.Reflection;
using System.Reflection.Emit;

namespace FastReflectionTests
{
    class Program
    {
        static void Main(string[] args)
```

```

{
    var name = "HelloWorld.exe";
    var assemblyName = new AssemblyName(name);
    // ایجاد یک اسمبلی جدید با قابلیت ذخیره سازی آن
    var assemblyBuilder = AppDomain.CurrentDomain.DefineDynamicAssembly(
        name: assemblyName,
        access: AssemblyBuilderAccess.RunAndSave);

    // افزودن یک ماژول به اسمبلی
    var moduleBuilder = assemblyBuilder.DefineDynamicModule(name);
    // تعریف یک کلاس در این ماژول
    var programmClass = moduleBuilder.DefineType("Program", TypeAttributes.Public);
    // افزودن یک متد به این کلاس
    // این متد خروجی ندارد اما ورودی آن شبیه به متد اصلی یک برنامه کنسول است
    var mainMethod = programmClass.DefineMethod(name: "Main",
        attributes: MethodAttributes.Public |
        MethodAttributes.Static,
        returnType: null,
        parameterTypes: new Type[] { typeof(string[]) });

    // تعیین بدنه متد اصلی برنامه
    var il = mainMethod.GetILGenerator();
    il.Emit(OpCodes.Ldstr, "Hello World!");
    il.Emit(OpCodes.Call, (typeof(Console)).GetMethod("WriteLine", new Type[] { typeof(string)
    }));
    il.Emit(OpCodes.Call, (typeof(Console)).GetMethod("ReadKey", new Type[0]));
    il.Emit(OpCodes.Pop);
    il.Emit(OpCodes.Ret);

    // تکمیل کار ایجاد نوع جدید
    programmClass.CreateType();

    // تعیین نقطه شروع فایل اجرایی برنامه کنسول تهیه شده
    assemblyBuilder.SetEntryPoint(((Type)programmClass).GetMethod("Main"));

    // ذخیره سازی این اسمبلی بر روی دیسک سخت
    assemblyBuilder.Save(name);
}
}
}

```

مراحلی را که توضیح داده شد، در کدهای فوق ملاحظه می‌کنید. انتخاب حالت دسترسی `AssemblyBuilderAccess.RunAndSave` سبب می‌شود تا بتوان نتیجه حاصل را ذخیره کرد. فایل `Exe` نهایی را اگر در برنامه `ILSpy` باز کنیم چنین شکلی دارد:

```

using System;
public class Program
{
    public static void Main(string[] array)
    {
        Console.WriteLine("Hello World!");
        Console.ReadKey();
    }
}

```