# **Reflection in C#**

جهت کار کردن و دسترسی به محتوای یک اسمبلی کاربرد دارد. می توانید به صورت dynamic یک instance از یک type بسازید یا متد های یک نوع را فراخونی کنید یا prop ها را مقدار دهی کنید و به آن ها دسترسی داشته باشید.

کاربرد اصلی دسترسی به محتوای assembly است. محتوای public private و ...

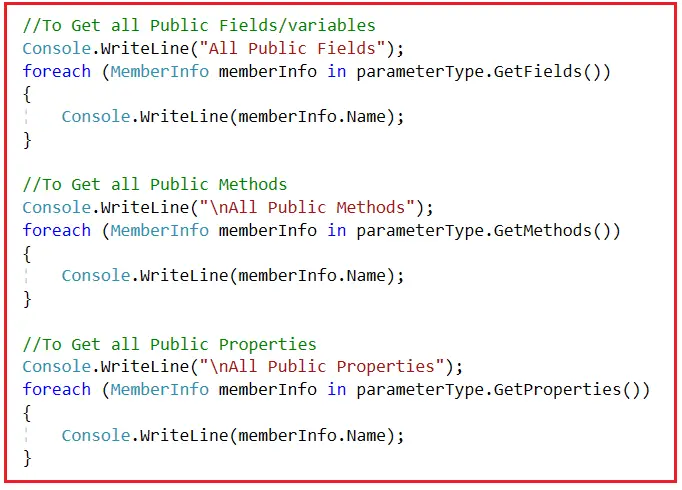
**مثال**

زمانی که در ویژوال استدیو از یک type یک object ایجاد می کنیم و دکمه دات را می زنیم تمامی گزینه های که intelligence می دهد تمامی prop method filed و ... امکان پذیر است به دلیل وجود reflection

**Sample**

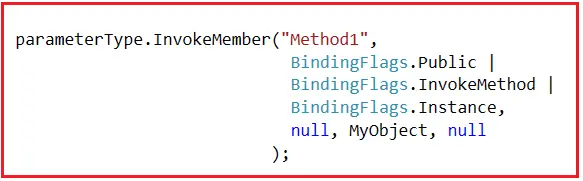
در ابتدا نیاز است که اسمبلی خود را Load کنیم با استفاده از Assembly.Loadfile(path) جهت دریافت reference مربوط به فایل assembly سپس نیاز است که class refence ها را دریافت کنیم از اسمبلی خود با استفاه از متد GetType سپس نیاز است که یک instance از کلاس خود ایجاد کنیم با استفاده از Activator.CreateInstance(myType) بعد از دریافت object می توانیم نوع Class را دریافت کنیم با استفاده از instance.GetType()

برای دسترسی به تمامی public member ها لازم است از متد GetMembers() استفاده کنیم و برای متد ها GetMethods() و فیلد ها GetFilelds() استفاده کنیم و پروپرتی ها GetPRoperties() .



**How to Invoke Methods Dynamically Using Reflection in C#?**

برای invoke کردن یک متد می توان از روش زیر استفاده کرد.



**Method Parameters**

1. **name: The string containing the name of the constructor, method, property, or field member to invoke. In our case it is Method1.**
2. **invokeAttr: A bitmask comprised of one or more System.Reflection.BindingFlags that specify how the search is conducted. The access can be one of the BindingFlags such as Public, NonPublic, Private, InvokeMethod, GetField, and so on. The type of lookup need not be specified. If the type of lookup is omitted, BindingFlags.Public | BindingFlags.Instance | BindingFlags.Static is used.**
3. **binder: An object that defines a set of properties and enables binding, which can involve the selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. -or- A null reference to use the System.Type.DefaultBinder. Note that explicitly defining a System.Reflection.Binder objects may be required for successfully invoking method overloads with variable arguments. Here, we are passing a null value.**
4. **target: The object on which to invoke the specified member. In our example, the object is MyObject.**
5. **args: An array containing the arguments to pass to the member to invoke. As our method does not take any arguments, we pass null here.**

**Sample**

**//Step2: Get the type**

**//Get the Assembly Reference**

**var MyAssembly = Assembly.LoadFile(@"D:\Projects\ReflectionDemo\SomeClassLibrary\bin\Debug\SomeClassLibrary.dll");**

**//Get the Class Reference**

**var MyType = MyAssembly.GetType("SomeClassLibrary.Class1");**

**//Create an instance of the type**

**dynamic MyObject = Activator.CreateInstance(MyType);**

**//Get the Type of the class**

**Type parameterType = MyObject.GetType();**

**//Step3: Browse the Metadata**

**//To Get all Public Fields/variables**

**Console.WriteLine("All Public Members");**

**foreach (MemberInfo memberInfo in parameterType.GetMembers())**

**{**

**Console.WriteLine(memberInfo.Name);**

**}**

**Console.WriteLine("\nInvoking Method1");**

**parameterType.InvokeMember("Method1",**

**BindingFlags.Public |**

**BindingFlags.InvokeMethod |**

**BindingFlags.Instance,**

**null, MyObject, null**

**);**

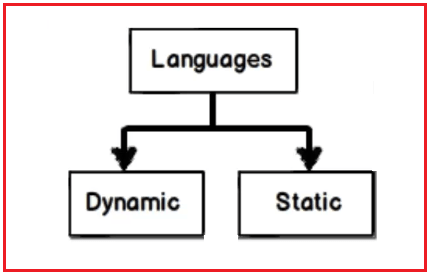
**Real Time Uses**

برای دسترسی به متا دیتا ها - در unit test – Late binding برای فایل های که در compile به آن ها دسترسی نداریم - زمانی که 2 پیاده سازی داریم از یک interface در فایل های مختلف پس لازم است که به صورت dynamic مشخص کنیم از کدام پیاده سازی استفاده شود.

# **Dynamic Type in C#**

برای جلوگیری از compile time checking کاربرد دارد و در زمان runtime عملیات چکینگ را انجام می دهد.

**Types of Programming Languages:**



Dynamic programing language (dynamic)

Strongly typed programing language (static)

**Strongly typed programing languages** زبان های هستند که **Data type checking** در زمان کامپایل رخ می دهد.

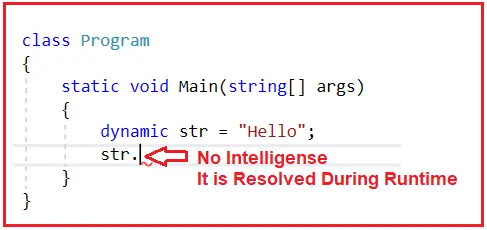
مثل دادن عدد به نوع String که خطای کامپایل می هد.

**Dynamic type** زبان های هستند که Data type checking در زمان run time انجام می شود.

مواقعی است که نوع داده را در زمان compile نمی دانیم پس نیاز است که data type checking را به زمان run time منتقل کنیم.

کلمه کلیدی dynamic کار ترکیبی انجام میدهد به این معنی که در زمان کامپایل type checking را انجام نمی دهد اما در زمان run time عملیات بررسی نوع داده رخ می دهد.

مثال زیر در زمان کامپایل داده را چک نمی کند اما در زمان اجرا بررسی می کند نوع آن را.



فهمیدن نوع داده در زمان اجرا با reflection امکان پذیر است.

**dynamic** str = "Hello";

Console.WriteLine**(**str.GetType**())**;

خروجی کنسول system string است.

**Static binding – early binding به معنای عملیات بایند در زمان کامپایل است.**

**نکته:** تبدیل نوع داده static به نوع داده dynamic در سی شارپ به صورت خود کار اتفاق می افتد emplicitly.

**int** int1 = 50;

**dynamic** dynamic1 = int1; //Explicit cast not required

Console.WriteLine**(**$"int1 = {int1} & dynamic1 = {dynamic1}"**)**;

//Convert from dynamic to int

**dynamic** dynamic2 = 100;

**int** int2 = dynamic2; //Explicit cast not required

**dynamic with complex data type**

**dynamic** dynamicStudent = new Student**()**

**{**

Name = "Anurag",

Barnch = "CSE",

Roll = 1001

نکته مهم در تبدیل نوع داده ها احتمال data loss است.

Dynamic parameter

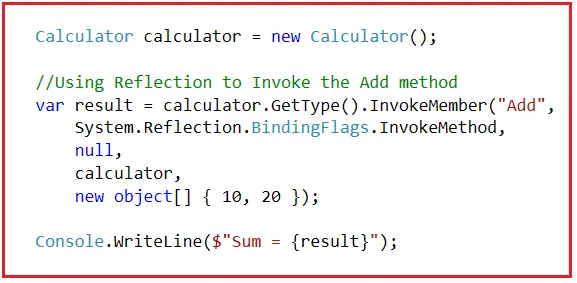
**public** **static** **void** DisplayValue**(dynamic** val**)**

**{**

Console.WriteLine**(**val**)**;

**}**

یکی از مزایای آن استفاده برای نگه داری json است



فراخوانی متد با reflection و ارسال 2 پارامتر به آن

**public** **class** Calculator

**{**

**public** **int** Add**(int** number1, **int** number2**)**

**{**

**return** number1 + number2;

**}{**

**با استفاده از Reflection نیاز به نوشتن کد زیادی برای دریافت متدی از یک کلاس هستیم اما با dynamic به سادگی می توان انجام داد به دلیل این که در زمان کامپایل type check انجام نمی شود**

**dynamic** calculator = new Calculator**()**;

var result = calculator.Add**(**10, 20**)**;عدم خطا چون کامپایل چک نمی کند و همچنین دارای متد ادد در نوع کلاس هستیم که زمان اجرا بررسی می وشد