Microsoft®

.net™

Програмиране за .NET Framework

http://www.nakov.com/dotnet/

Отражение на типовете (Reflection)

Ивайло Христов
софтуерен разработчик

Необходими знания

- Базови познания за .NET Framework и Common Language Runtime (CLR)
- Базови познания за езика C#
- Базови познания за MSIL

Microsoft .net

Съдържание

- Какво е Global Assembly Cache?
- Какво е Reflection?
- Зареждане на асемблита
- Извличане информация за асембли
- Премахване на асемблита от паметта
- Изучаване членовете на тип
- Извличане на методи и параметрите им
- Извличане на параметрите на метод
- Динамично извикване на методи
- Reflection Emit

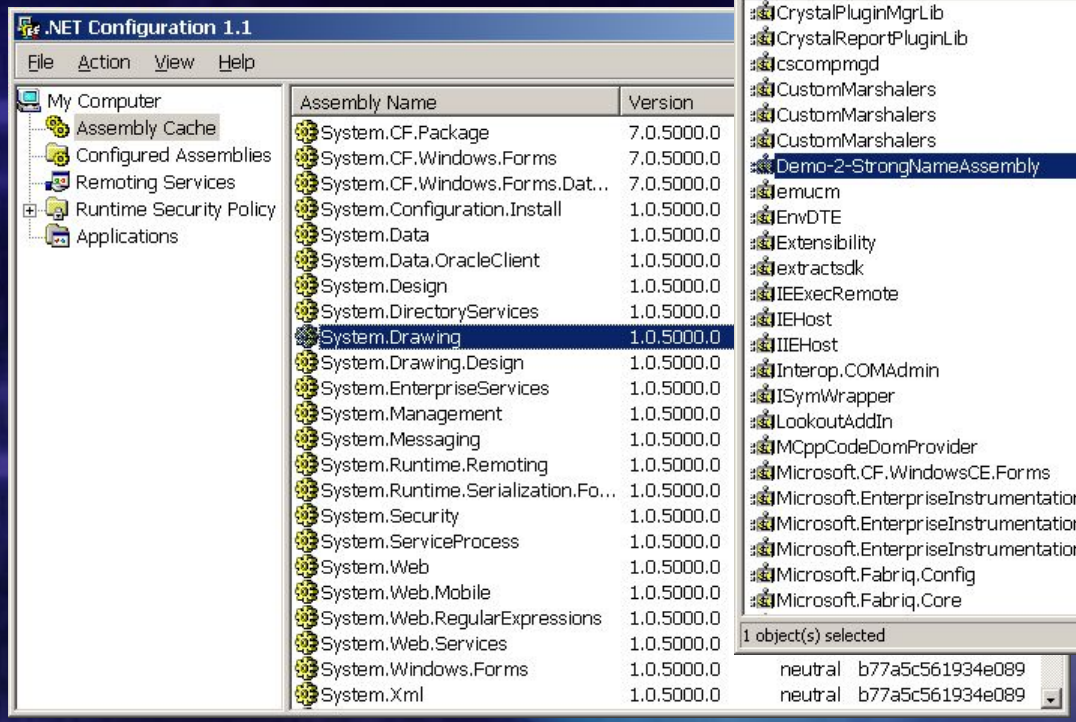
Какво е Global Assembly Cache?

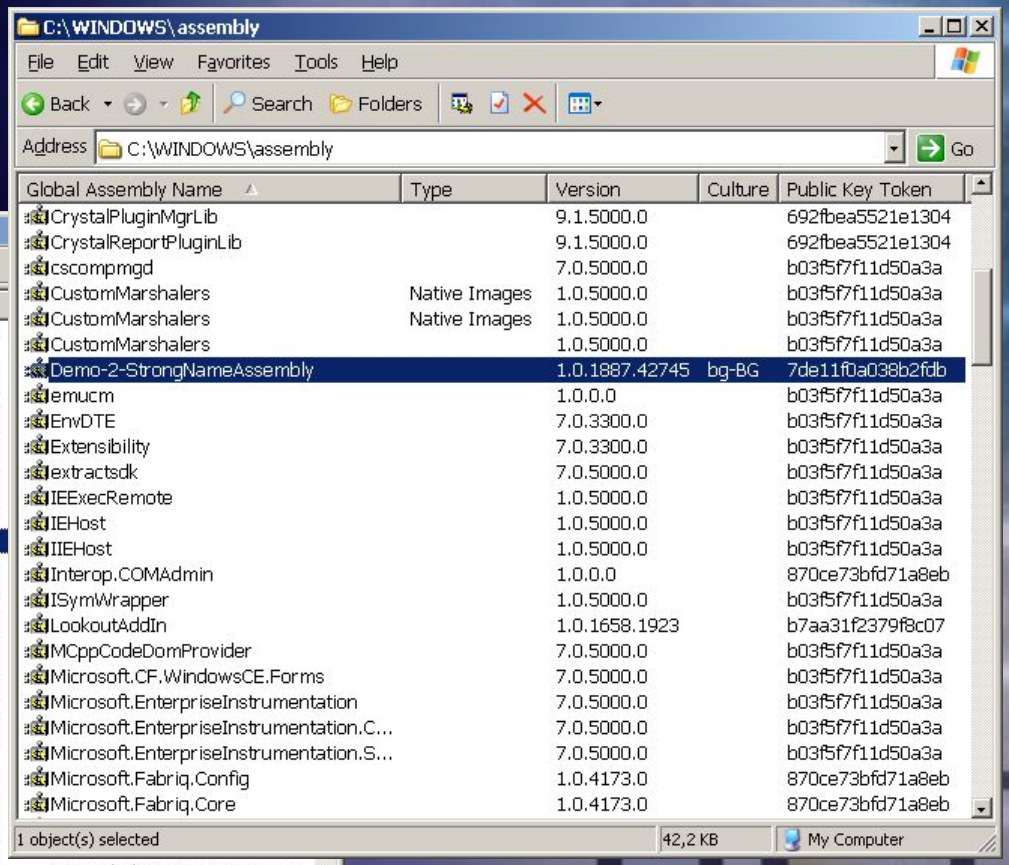
- Global Assembly Cache (GAC) е централно хранилище за споделени асемблита
- Асемблитата в GAC са достъпни за ползване от всички .NET приложения на машината
- Асемблитата в GAC имат силно име, което ги идентифицира уникално
- Не добавяйте асемблита в GAC освен, ако не е абсолютно необходимо

Microsoft .net

Демонстрация #1

- Преглед на GAG през Windows Explorer и през Administrative Tools





Какво е Reflection?

- Отражението на типовете (reflection) е възможността да получаваме информация за типовете по време на изпълнение на програмата
- С Reflection .NET приложенията могат:
  - да изучават метаданните на асемблита
  - да изучават типовете в дадено асембли
  - динамично да извикват методи
  - динамично да създават нови асемблита, да ги изпълняват и да ги запазват като файл

Microsoft .net

Зареждане на асемблита

- Зареждане чрез `System.Reflection.Asembly.Load(…)`
  - Приема като параметър:
    - името на асемблито
    - обект от тип `AssemblyName` – описва асемблито
  - Търси асембли със зададеното описание (probing) и ако го намери го зарежда
  - Ако асемблито не бъде намерено предизвиква `FileNotFoundException`

```
Assembly.Load("SomeAssembly.dll");
```

Зареждане на асемблита

- Зареждане чрез `System.Reflection.Asembly.LoadFrom(…)`
  - Приема като параметър пътя до асемблито
  - Прочита подадения файл
  - Извиква вътрешно `Load(…)`
  - По-бавно от `Load(…)`
  - Ако асемблито не бъде намерено се хвърля `FileNotFoundException`

```
Assembly.LoadFrom(@"C:\Tools\MyAss.dll");
```

Извличане информация за асембли

- Свойства на `System.Reflection.Assembly` за извличане информация за асембли
  - `FullName`
    - пълното име на асемблито, включващо версия, култура и ключ (Public Key Token)
  - `Location`
  - `EntryPoint`
  - `GlobalAssemblyCache`

Microsoft .net

Извличане информация за асембли

- Свойства на `System.Reflection.Assembly` за извличане информация за асембли
  - `FullName`
  - `Location`
    - пътят, от където е заредено асемблито
  - `EntryPoint`
  - `GlobalAssemblyCache`

Microsoft .net

Извличане информация за асембли

- Свойства на `System.Reflection.Assembly` за извличане информация за асембли
  - `FullName`
  - `Location`
  - `EntryPoint`
    - методът, от който ще започне изпълнението на асемблито
  - `GlobalAssemblyCache`

Microsoft .net

Извличане информация за асембли

- Свойства на `System.Reflection.Assembly` за извличане информация за асембли
  - `FullName`
  - `Location`
  - `EntryPoint`
  - `GlobalAssemblyCache`
    - булева стойност, която показва дали асемблито е било заредено от GAC

Демонстрация #2

- Зареждане на асемблита

Demo-2-DoubleLoad - Microsoft Visual C# .NET [design] - AssemblyDoubleLoad.cs

File Edit View Project Build Debug Tools Window Help

AssemblyDoubleLoad.cs

AssemblyDoubleLoad

```
static private void ShowAllAsembli
{
    foreach(Assembly assembly in A
    {
        Console.WriteLine(assembly
        Console.WriteLine(assembly
        Console.WriteLine();
    }
}

static void Main(string[] args)
{
    // List all assemblies at the l
    Console.WriteLine(" ----------
    ShowAllAsemblies();

    // Load mscorlib.dll with Asse
    Assembly.Load("mscorlib.dll");
    Console.WriteLine("------ Afte
    ShowAllAsemblies();

    // Load mscorlib.dll from a different path
    Assembly.LoadFrom(@"..\..\mscorlib.dll");
```

"C:\MS Content and Curriculum\PPT\-Lecture-18-Reflection\Demo-2-DoubleLoad\bin\Debug\Demo2.exe"

```
------------------- In the beginning -------------------
mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
c:\windows\microsoft.net\framework\v1.1.4322\mscorlib.dll

Demo2, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
C:\MS Content and Curriculum\PPT\-Lecture-18-Reflection\Demo-2-DoubleLoad\bin\De
bug\Demo2.exe

------ After loading Mscorlib with Assembly.Load() ------
mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
c:\windows\microsoft.net\framework\v1.1.4322\mscorlib.dll

Demo2, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
C:\MS Content and Curriculum\PPT\-Lecture-18-Reflection\Demo-2-DoubleLoad\bin\De
bug\Demo2.exe

 --- After loading mscorlib.dll from a different path ---
mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
c:\windows\microsoft.net\framework\v1.1.4322\mscorlib.dll

Demo2, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
C:\MS Content and Curriculum\PPT\-Lecture-18-Reflection\Demo-2-DoubleLoad\bin\De
bug\Demo2.exe

mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
c:\ms content and curriculum\ppt\-lecture-18-reflection\demo-2-doubleload\mscorl
ib.dll

Press any key to continue
```

Properties | Dynamic Help

Ready | Ln 1 | Col 1 | Ch 1 | INS

Премахване на асемблита от паметта

- Не се подържа възможността да се премахне едно асембли
- Възможно е премахването на всички асемблита в даден домейн
- Не се препоръчва да се използва поради голямата опасност от грешки

Microsoft .net

Изучаване на типовете в асембли

- `System.Type` – отправна точка за извършване на манипулации с типове и обекти
- Чрез `System.Type` можем да получим всички членове на даден тип:
  - полета
  - методи
  - свойства
  - събития
  - вложени типове
- Чрез `Assembly.GetTypes()` извличаме типовете от дадено асембли

Microsoft .net

Изучаване на типовете в асембли

- `Sytem.Type` дефинира множество от свойства и методи за изучаване информацията за даден тип:
  - Свойства:

```
BaseType, Attributes, FullName, IsAbstract,
IsArray, IsByRef, IsClass, IsCOMObject,
IsEnum, IsInterface, IsPublic, IsSealed,
IsValueType, Name, …
```

  - Методи:

```
GetConstructors(), GetEvents(), GetFields(),
GetInterfaces(), GetMembers(), GetMethods(),
GetNestedTypes(), GetProperties(),
InvokeMember(), IsInstanceOfType()
```

Изучаване членовете на тип

Взимаме текущото асембли

Получаваме всички типове в асемблито

```
Assembly currAssembly =
Assembly.GetExecutingAssembly();

foreach(Type type in currAssembly.GetTypes())
{
    foreach(MemberInfo member in type.GetMembers())
    {
        Console.WriteLine(member.MemberType);
        Console.WriteLine(member.Name);
    }
}
```

GetMembers() връща масив от членовете на типа

Демонстрация #3

- Изследване на типовете в асембли

Demo-3-AssemblyTypesInfo - Microsoft Visual C# .NET [design] - AssemblyTypesDemo.cs

File Edit View Project Build Debug Tools Window Help

Debug table1

AssemblyTypesDemo.cs

SomeStructure

```
    static void Main()
    {
        Assembly currentAssembly;
        currentAssembly = Assembly.GetExecutingAssembly();

        foreach (Type type in currentAssembly.
        {
            MemberInfo[] members = type.FindMe
                MemberTypes.Method,
                BindingFlags.NonPublic | Bindi
                Type.FilterName,
                "*");
            foreach (MemberInfo member in memb
            {
                Console.WriteLine("{0}.{1}()",
            }

            Console.WriteLine();
        }
    }
}
```

Solution Explorer - Demo-3-Assemb...

ution 'Demo-3-AssemblyTypesInfo' (1 pr

Demo-3-AssemblyTypesInfo

References

AssemblyTypesDemo.cs

"C:\MS Content and Curriculum\PPT\-Lecture-18-Reflection\...

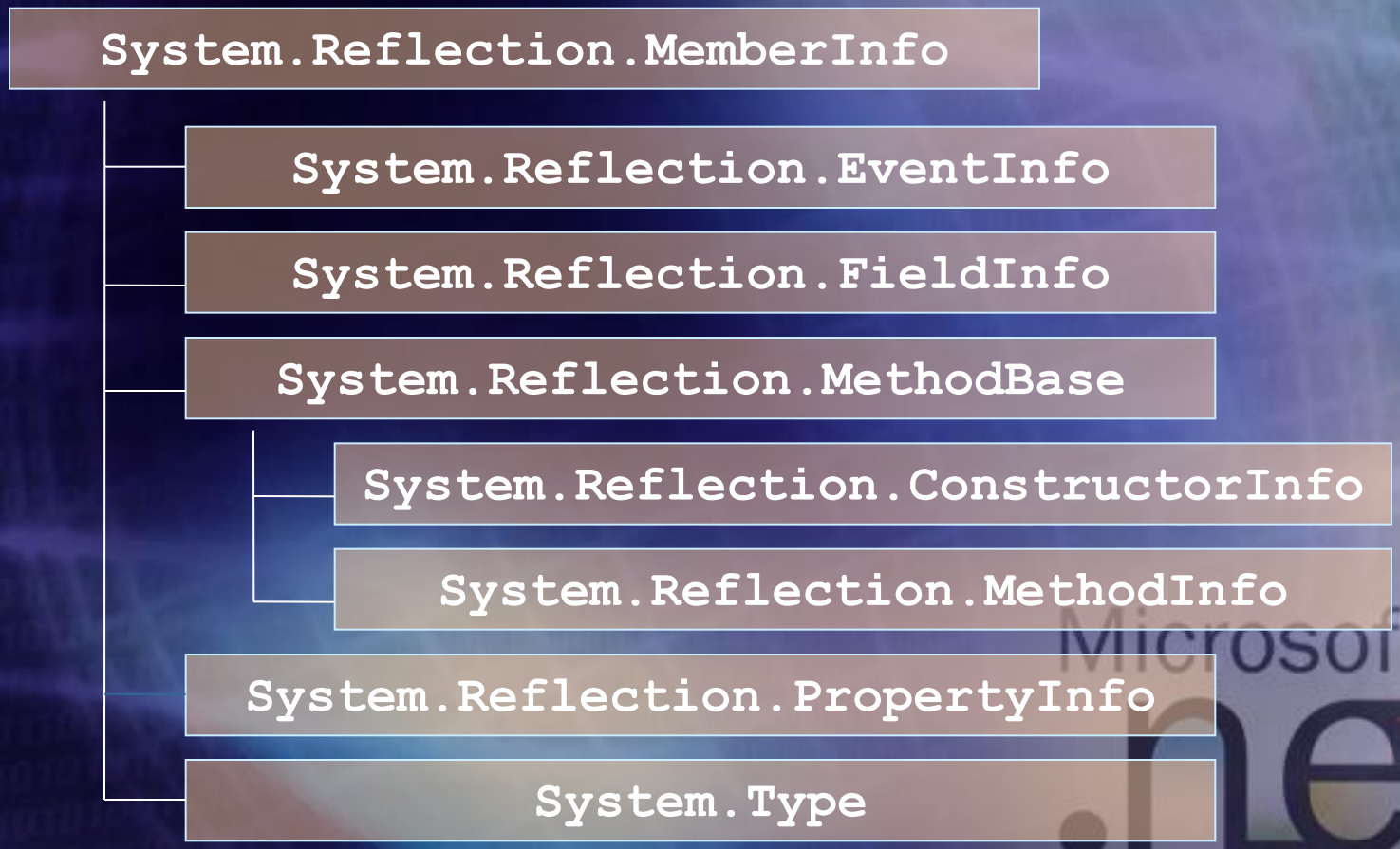
```
SomeStructure.Finalize()
SomeStructure.MemberwiseClone()

AssemblyTypesDemo.Finalize()
AssemblyTypesDemo.SomePrivateMethod()
AssemblyTypesDemo.MemberwiseClone()

Press any key to continue
```

Properties Dynamic Help

Ready Ln 1 Col 1 Ch 1 INS

Класове за видовете членове

- За всеки вид членове има съответен клас, който ги описва:



Извличане на методи и параметрите им

- `Type.GetMethod()` – връща отражението на даден метод (`MethodInfo`)
- `MethodInfo.GetParameters()` – извлича параметрите на даден метод

```
MethodInfo someMethod =
    myType.GetMethod("SomeMethod");
foreach(ParameterInfo param in
    someMethod.GetParameters())
{
    Console.WriteLine(param.ParameterType);
}
```

Динамично извикване на метод от асембли (Late Binding)

1. Създаваме инстанция на типа, чрез някой от статичните методи на класа `Activator`:
   - `CreateInstance(…)` – създава инстанция на посочения тип
   - `CreateInstanceFrom(…)` – инстанцира определен тип от дадено асембли
   - `CreateComInstanceFrom(…)` – създава инстанция на COM обект
2. Динамично извикаме методите на типа чрез `System.MethodInfo.Invoke(…)`

Динамично извикване на метод

```
// Load the assembly mscorlib.dll
Assembly mscorlibAssembly =
    Assembly.Load("mscorlib.dll");

// Create an instance of DateTime by calling
// new DateTime(2004, 1, 5)
Type systemDateTimeType = mscorlibAssembly.
    GetType("System.DateTime");
object[] constructorParams =
    new object[] {2004, 1, 5};
object dateTimeInstance =
    Activator.CreateInstance(
    systemDateTimeType, constructorParams);
```

Параметри за конструктора на DateTime

(примерът продължава)

Динамично

Параметри за метода, който извикваме. Може да има няколко метода с еднакво име, но с различни параметри.

```
// Invoke DateTime.AddDays(10)
Type[] addDaysParamsTypes =
    new Type[] {typeof(System.Double)};
MethodInfo addDaysMethod = systemDateTimeType.
    GetMethod("AddDays", addDaysParamsTypes);
object[] addDaysParams = new object[]{10};
object newDateTimeInst = addDaysMethod.Invoke(
    dateTimeInstance, addDaysParams);

// Get the value of the property DateTime.Date and print it
PropertyInfo datePropertyInfo =
    systemDateTimeType.GetProperty("Date");
object datePropValue = datePropertyInfo.GetValue(
    newDateTimeInst, null);
Console.WriteLine("{0:dd.MM.yyyy}", datePropValue);
```

`GetValue()` използва втория си аргумент само ако свойството е индексатор

Демонстрация #4

- Зареждане на тип от асембли и извикване на методи

Demo-4-LateBinding - Microsoft Visual C# .NET [design] - LateBindingDemo.cs

File Edit View Project Build Debug Tools Window Help

Debug | table1

LateBindingDemo.cs

LateBindingDemo | Main(string[] args)

```
static void Main(string[] args)
{
    // Load the assembly mscorlib.dll
    Assembly mscorlibAssembly = Assembly.Load("mscorlib.dll");

    // Create an instance of DateTime by calling
    // new DateTime(2004, 1, 5)
    Type systemDateTimeType =
        mscorlibAssembly.GetType("System.DateTime");
    object[] constructorParams = new object[] {2004, 1, 5};
    object dateTimeInstance = Activator.CreateInstance(
        systemDateTimeType, constructorParams);

    // Invoke DateTime.AddDays(10)
    Type[] addDaysParamsTypes =
        new Type[] {typeof(System.Double)};
    MethodInfo addDaysMethod = systemDateTimeType.GetMethod(
        "AddDays", addDaysParamsTypes);
    object[] addDaysParams = new object[]{10};
    object newDateTimeInstance =
        addDaysMethod.Invoke(dateTimeInstance, addDaysParams);

    // Get the value of the property DateTime.Date and print it
```

Solution Explorer - Demo-4-LateBin...

Solution 'Demo-4-LateBinding' (1 pr

Demo-4-LateBinding

References

LateBindingDemo.cs

Solution Explorer | Index

Dynamic Help

Help

Main (C#)

Code and Text Editor

Coding Techniques and Programm

Samples

Visual C# Samples

Office Development Samples

Office Development Code Example

Smart Device Samples

Visual Studio Samples

Properties | Dynamic Help

Build succeeded | Ln 25 | Col 37 | Ch 31 | INS

Какво е Reflection Emit?

- `Reflection.Emit`
  - Създаване на цели асемблита
  - Запазване на асемблита на диска
  - Изпълнение на асемблита
  - Изпълнение и запазване на асемблита
- `Reflection.Emit` ни позволява да създадем асемблита от нулата
  - Модули
  - Типове
    - Конструктори
    - Методи
    - Събития
    - Свойства

Microsoft .net

Използване на Reflection Emit

- Пространството `System.Reflection.Emit` предоставя набор от класове за създаване на части от асемблита:
  - Асемблита – `AssemblyBuilder`
  - Модули – `ModuleBuilder`
  - Типове – `TypeBuilder`
    - Конструктори – `ConstructorBuilder`
    - Методи – `MethodBuilder`
    - Свойства – `PropertyBuilder`
    - Събития – `EventBuilder`

Microsoft .net

Използване на Reflection Emit

- Чрез класа `System.Reflection.Emit.ILGenerator` се генерират MSIL инструкции
  - Представляват MSIL изпълним код
  - Могат да се добавят в даден метод
  - `Emit(…)` – добавяме в поток последователност от MSIL инструкции
  - `EmitWriteLine(…)` – добавя инструкциите за отпечатване на низ
- Създаване на изпълними асемблита :
  - `AssemblyBuilder.SetEntryPoint(…)`

Динамично генериране на асембли

```
AssemblyName assemblyName = new AssemblyName();
assemblyName.Name = "DynamicAssembly";

AssemblyBuilder newAssembly = AppDomain.
  CurrentDomain.DefineDynamicAssembly(
    assemblyName, AssemblyBuilderAccess.RunAndSave);

ModuleBuilder newModule =
  newAssembly.DefineDynamicModule(
    "NewModule", " EmitedAssembly.exe ");

TypeBuilder newType = newModule.DefineType(
  "HelloWorldType", TypeAttributes.Public);

MethodBuilder newMethod = newType.DefineMethod(
  "WriteHello", MethodAttributes.Static |
  MethodAttributes.Public, null, null);
```

(примерът продължава)

Динамично генериране на асембли

```
ILGenerator msilGen = newMethod.GetILGenerator();
msilGen.EmitWriteLine(
  "Hello World! Today is " + DateTime.Now);
msilGen.Emit(OpCodes.Ret);

Type helloWorldType = newType.CreateType();
Object instance =
  Activator.CreateInstance(helloWorldType);

MethodInfo helloWorldMethod =
  helloWorldType.GetMethod("WriteHello");
helloWorldMethod.Invoke(instance, null);

newAssembly.SetEntryPoint(helloWorldMethod);
newAssembly.Save("EmitedAssembly.exe");
```

Демонстрация #5

- Динамично създаване на асембли

Demo-5-ReflectionEmit - Microsoft Visual C# .NET [design] - ReflectionEmitDemo.cs

File Edit View Project Build Debug Tools Window Help

ReflectionEmitDemo.cs

ReflectionEmitDemo | Main()

```
static void Main()
{
    AssemblyName assemblyName = new AssemblyName();
    assemblyName.Name = "DynamicAssembly";

    // Create new assembly
    AssemblyBuilder newAssembly =
        AppDomain.CurrentDomain.DefineDynamicAssembly(
            assemblyName, AssemblyBuilderAccess.RunAndSave);

    // Create new module in the new assembly
    ModuleBuilder newModule = newAssembly.DefineDynamicModule(
        "NewModule", "EmitedAssembly.exe");

    // Create new type in the new module
    TypeBuilder newType = newModule.DefineType(
        "HelloWorldType", TypeAttributes.Public);

    // Create new method in the new type
    MethodBuilder newMethod = newType.DefineMethod(
        "WriteHello", MethodAttributes.Static | MethodAttributes.Public,
        null, null);
```

Solution Explorer - Demo-5-Reflecti...

Solution 'Demo-5-ReflectionEmit' (1

Demo-5-ReflectionEmit

References

ReflectionEmitDemo.cs

Solution Explorer | Index

Properties

Properties | Dynamic Help

Ready | Ln 4 | Col 1 | Ch 1 | INS

Отражение на типовете (Reflection)

Въпроси?

Microsoft .net

Упражнения

1. Какво е Global Assembly Cache? За какво служи?
2. Опишете поне един начин за преглеждане на асемблитата от Global Assembly Cache.
3. Да се реализира Windows Forms приложение, което позволява да се зарежда избрано от потребителя асембли и показва информация за него (път от където е заредено, дали е заредено от GAC, входната му точка и т.н.) .
4. Да се реализира конзолно приложение, което зарежда асемблито `mscorlib.dll` и отпечатва имената на всички типове в него.
5. Да се реализира конзолно приложение, което зарежда асемблито `mscorlib.dll` и намира всички методи на типа `System.DateTime`, който е дефиниран в него.

Упражнения

6. Съставете Windows Forms приложение, което зарежда асембли, името на което се избира от потребителя и извлича от него имената и параметрите на конструкторите на всички типове, дефинирани в него.
7. Дефинирайте интерфейс `ICalculatable`, който дефинира метод `double Calculate(int[])`. Напишете конзолно приложение, което чете от текстов файл редица от числа, намира всички асемблита от зададена директория, в които има имплементация на `ICalculatable` и чрез всяко от тях извършва пресмятането `Calculate(…)` и отпечатва резултата. Тествайте като създадете две асемблита, в които има тип, имплементиращ `ICalculatable`. Едното асембли трябва да изчислява средно аритметично, а другото сума на елементите от подадения масив.

Упражнения

8. Съставете програма, която прочита въведена текстова последователност и създава асембли съдържащо тип, който съдържа метод отпечатващ тази текстова последователност. Генерираното асембли трябва да бъде съхранено, като изпълним файл.

Microsoft .net

Използвана литература

- Jeffrey Richter, Applied Microsoft .NET Framework Programming, Microsoft Press, 2002, ISBN 0735614229
- Jesse Liberty, Programming C#, 3rd Edition, O'Reilly, 2003, ISBN 0596004893
- Professional C#, 3rd, Wrox Press, 2004, ISBN 0764557599
- Георги Иванов, Отражение на типовете (Reflection) – http://www.nakov.com/dotnet/2003/lectures/Reflection.doc
- MSDN Library – http://msdn.microsoft.com

Microsoft

БАРС
Българска асоциация на разработчиците на софтуер

НАРС
НАЦИОНАЛНА АКАДЕМИЯ ПО РАЗРАБОТКА НА СОФТУЕР