Л.Р. №5-6: ДИНАМІЧНА ІДЕНТИФІКАЦІЯ ТИПІВ

План

5.1. Поняття рефлексії (відображення).

Рефлексія - це засіб, що дозволяє отримувати відомості про тип даних. Термін рефлексія, або відображення, походить від принципу дії цього засобу: об'єкт класу `Type` відображає базовий тип, який він представляє. Для отримання інформації про тип даних об'єкту класу `Type` робляться запити, а він повертає назад (відображає) інформацію, пов'язану з визначеним типом. Рефлексія є ефективним механізмом, оскільки вона дозволяє виявляти і використовувати можливості типів даних, відомі тільки під час виконання. Простір імен – `System.Reflection`.

5.2. Клас Type – ядро підсистеми рефлексії.

Клас `System.Type` становить ядро підсистеми рефлексії, оскільки він інкапсулює тип даних. Він містить багато властивостей і методів, якими можна користуватися для отримання інформації про тип даних під час виконання. Клас `Type` є похідним від абстрактного класу `System.Reflection.MemberInfo`.

У класі `MemberInfo` визначені наведені нижче властивості, доступні тільки для читання.

Властивість `MemberType` повертає тип `MemberTypes` - перерахування, в якому визначаються значення, що позначають різні типи членів. До їх числа відносяться наступні.

```
MemberTypes.Constructor
MemberTypes.Method
MemberTypes.Field
MemberTypes.Event
MemberTypes.Property
```

Методи класу `Type`, що найчастіше використовуються:

Властивості класу `Type`, що найчастіше використовуються:

5.3. Застосування рефлексії

За допомогою методів і властивостей класу `Type` можна отримати докладні відомості про тип даних під час виконання програми. Це досить ефективний засіб. Адже отримавши відомості про тип даних, можна відразу ж викликати його конструктори і методи або скористатися його властивостями. Отже, рефлексія дозволяє використовувати код, який не був доступний під час компіляції.

Прикладний інтерфейс `Reflection API` має ясну логічну структуру, а отже, зрозумівши одну його частину, неважко зрозуміти і все інше.

5.4. Отримання відомостей про методи

Список методів, що підтримуються окремим типом даних, можна отримати, використовуючи метод `MethodInfo[] GetMethods()`. Цей метод повертає масив об'єктів класу `MethodInfo`, які описують методи, що підтримуються типом. Клас `MethodInfo` володіє багатьма властивостями та методами, які дозволяють отримувати додаткову інформацію та виконувати певні дії над типом.

Наприклад, для отримання імені методу служить властивість `Name`. Особливий інтерес викликають два члена класу `MethodInfo`:

- доступна тільки для читання властивість `ReturnType` (тип, що повертає метод)
- і метод `ParameterInfo[] GetParameters()` – повертає список параметрів методу.

Відомості про параметри містяться в об'єкті класу `ParameterInfo`. У класі `ParameterInfo` визначено чимало властивостей і методів, що описують параметри. Особливе значення мають дві властивості: `Name` - являє собою рядок, що містить ім'я параметра, а `ParameterType` - описує тип параметра, який інкапсульований в об'єкті класу `Type`.

```
Dictionary<string, MethodInfo> GetClassMethodsMI(object obj)
{
	Dictionary<string, MethodInfo> res = new Dictionary<string, MethodInfo>();
	Type t = obj.GetType();
	MethodInfo[] methInfo = t.GetMethods(BindingFlags.DeclaredOnly | BindingFlags.Public |
BindingFlags.Instance | BindingFlags.NonPublic);
	foreach (MethodInfo method in methInfo)
	{
		string s = "(";
		ParameterInfo[] par = method.GetParameters();
		foreach (ParameterInfo parInfo in par)
		{
			s += parInfo.ParameterType.Name + " " + parInfo.Name + ", ";
		}
		if (s.Length > 2 & s.LastIndexOf(", ") == s.Length - 2)
			s = s.Remove(s.Length - 2, 2);
		res.Add(method.ReturnType.Name + " " + method.Name + s + ")",method);

	}
	return res;
}
```

У цьому прикладі використана друга форма методу GetMethods (), що дозволяє вказувати різні прапорці для фільтрування видобутих відомостей про методи:
`MethodInfo[] GetMethods(BindingFlags flags)`

Два або кілька прапорців можна об'єднати за допомогою логічної операції АБО. Але як мінімум прапорець Instance або Static слід вказувати разом з прапорцем Public або Nonpublic. В іншому разі не будуть видобуті відомості про жоден з методів.

5.5. Виклик методів за допомогою рефлексії

Як тільки методи, які підтримуються певним типом даних, стають відомі, їх можна викликати. Для цієї мети служить метод Invoke(), що входить до складу класу Methodlnfo:

```
object Invoke(object obj, object[] parameters)
```

де obj позначає посилання на об'єкт, для якого викликається метод. Для виклику статичних методів (static) в якості параметра obj передається порожнє значення (null). Будь-які аргументи, які повинні бути передані методу, вказуються у масиві parameters. Якщо ж аргументи не потрібні, то замість масиву parameters вказується порожнє значення (null). Крім того, кількість елементів масиву parameters має точно відповідати кількості переданих аргументів.

```
static void Main()
{
     Type t = typeof(MyClass);
     MyClass reflectOb = new MyClassA0, 20);
     int val;
     MethodInfo[] mi = t.GetMethods();
     // Вызвать каждый метод.
     foreach(Methodlnfo m in mi)
     {
          // Получить параметры.
          Parameterlnfo[] pi = m.GetParameters() ;
          if(m.Name.CompareTo("Set")==0 && pi[0].ParameterType==typeof(int))
          {
               object[] args = new object[2];
               args[0] = 9;
               args[l] = 18;
               m. Invoke(reflectOb, args);
          }
     else
          if(m.Name.CompareTo("Set")==0&& pi[0].ParameterType==typeof(double))
          {
               object[] args = new object[2];
               args[0] =1.12;
               args[l] = 23.4;
               m. Invoke(reflectOb, args);
          }
          else
               if(m.Name.CompareTo("Sum")==0)
               {
                    val = (int) m.Invoke(reflectOb, null);
                    Console.WriteLine("Сумма равна " + val);
               }
               else
                    if(m.Name.CompareTo("Show")==0)
                         {
                              m.Invoke(reflectOb, null);
                         }
```

```
}
```

5.6. Завдання для виконання.

1. Створити новий проект за шаблоном Windows Forms Application.
2. На форму додати елемент керування TreeView.
3. Описати клас, забезпечивши наявність:
   - 4-х властивостей різних типів, одна з яких мала б тип колекції;
   - 3-х методів;
   - 2-х конструкторів.
4. Описати метод, який би виводив перелік всіх властивостей (разом з типом та значенням) об’єкта описаного класу у TreeView.

Варіанти завдань

1. class Студент.
2. class Співробітник.
3. class Споруда.
4. class Мобільний.
5. class Корабель.
6. class Лікар.
7. class Меблі.
8. class Кінофільм.
9. class Тканина.
10. class Цифровий фотоапарат.
11. class Громадський транспортний засіб.
12. class Принтер.
13. class Зошит
14. class Настільна лампа