Тема 12

Шаблонни типове в C#

1. Същност и декларираме на шаблонен тип в C#

Много често възниква проблемът програмистът да създава класове, които са сходни по функционалност, а се различават само по типа на обектите, с които работят. Например, необходимо е да се изгради списък като елементите на списъка са цели числа. В този случай полетата в класа ще са от целочислен тип. Методите, който биха били включени в подобен клас като добавяне, изтриване и търсене на елемент в списъка, биха работили с променливи от целочислен тип. Ако същата задача бъде поставена за изграждане на списък от дробно-десетични стойности или на низове, структурата и функционалността на следващите два класа ще бъдат идентични с тези на първия клас като единствената разлика ще бъде в типовете на данните.

Могат да бъдат дадени още много други такива примери за класове, които биха се различавали единствено по типовете на полетата. Например, ако трябва да се описват класове съответстващи на група недвижими имоти, които могат да са къща, етаж от къща или апартамент и да се определя дали обектът се продава, купува или отдава под наем, функционалността ще е идентична и ще се различава единствено по типа на обектите.

В подобни случаи възможностите на език C# е предлагат създаването на шаблонни типове (generics). Чрез шаблонните типове функционалността на един клас може да бъде приложена към множество обекти от различни типове. Шаблонните типове са класове, структури, интерфейси и методи, на които се предават като параметри типове. В последствие, при инстанцирането (създаването на програмни обекти) на класове и структури, ще бъдат използвани конкретните типове, които се предават като параметри.

Декларацията на типизиран (шаблонен) клас в C# има следния общ вид:

class ИмеТипизиранКлас <Type>
{

}

Type се явява идентификатор, който съответства на параметър на тип. Този тип е неизвестен по време на декларацията на класа.

На практика, типизирането на клас т.е. създаването на шаблонен клас е чрез добавянето на параметър към декларацията, съответстващ на неизвестен тип. Едва при инстанцирането на класа този параметър ще бъде заменен с конкретен тип.

Пример:

```
public class Generic<T>
{
   public T Field;
```

```
}
```

Параметърът тип задължително се поставя между символите < > както в декларацията, така и при инстанцирането на класа. За да бъде инстанциран типизиран клас се използва следния синтаксис:

ИмеТипизиранКлас <тип> имеОбект= new ИмеТипизиранКлас <тип>();

В този случай тип е конкретен тип, който замества парамертизирания.

```
 Generic<string> g = new Generic<string>();
g.Field = "A string";
//...
Console.WriteLine("Generic.Field          = \"{0}\"", g.Field);
        Console.WriteLine("Generic.Field.GetType() = {0}", g.Field.GetType().FullName);

Generic<int> i = new Generic<int>();
i.Field = 10;
Console.WriteLine("Generic.Field          = \"{0}\"", i.Field);
        Console.WriteLine("Generic.Field.GetType() = {0}", i.Field.GetType().FullName);
```

Пример:

```
public class Point<T>
    {
        private T x, y;
        public T X
        {
            get { return x; }
            set {x=value;}
        }
        public T Y
        {
            get { return y; }
            set { y = value; }
        }

        public void Info()
        {
            Console.WriteLine("Generic Type: {0}", this.x.GetType());
            Console.WriteLine("x={0}, y={1}", x, y);
        }
    }

Point<int> p1 = new Point<int>();
        p1.X = 100;
        p1.Y = 100;
        p1.Info();

        Point<double> p2 = new Point<double>();
        p2.X = 10.01;
        p2.Y = 100.12;
        p2.Info();
```

Пример:

```
public class MyGenericArray<T> {
    private T[] array;

    public MyGenericArray(int size) {
      array = new T[size + 1];
    }
    public T getItem(int index) {
      return array[index];
    }
    public void setItem(int index, T value) {
      array[index] = value;
    }
  }
  class Tester {
    static void Main(string[] args) {

      //declaring an int array
      MyGenericArray<int> intArray = new MyGenericArray<int>(5);

      //setting values
      for (int c = 0; c < 5; c++) {
        intArray.setItem(c, c*5);
      }

      //retrieving the values
      for (int c = 0; c < 5; c++) {
        Console.Write(intArray.getItem(c) + " ");
      }

      Console.WriteLine();

      //declaring a character array
      MyGenericArray<char> charArray = new MyGenericArray<char>(5);

      //setting values
      for (int c = 0; c < 5; c++) {
        charArray.setItem(c, (char)(c+97));
      }

      //retrieving the values
      for (int c = 0; c< 5; c++) {
        Console.Write(charArray.getItem(c) + " ");
      }
      Console.WriteLine();

      Console.ReadKey();
    }
  }
```

2. Използване на неизвестни типове в дефиниции на полета и методи

Неизвестните типове в тялото на класа се използват при дефинициите на полета и методи

Например, поле може да бъде дефинирано по следния начин:

Модификатор Т имеПоле;

Като Т се явява неизвестния тип

При дефиницията на метод неизвестният тип може да се използва като тип на върнат резултат или като параметър на метода.

Пример:

```
Модификатор Т ИмеМетод(параметри)
{
}

Модификатор Т ИмеМетод(Т име параметър)
{
}
```

3. Разгъване на шаблонен клас

Когато се компилира шаблонен клас (да се има предвид, че преводът е до код на междинен език IL-код) транслираният код съдържа информация, че се класът е типизиран т.е. че се работи с неопределени до момента типове. По време на изпълнение, когато се използва типизирания клас, се създава ново описание на класа като неопределения до момента тип вече се заменя с конкретен тип и се създава ново описание на класа. Новото описание е идентично с това на типизирания клас с тази разлика, че на всякъде типът Т е заменен с конкретния. Това ново описание на класа се използва за да се създаде обект от класа при инстанцирането на шаблонния клас. Затова инстанцирането на шаблонен клас се нарича още „разгъване на шаблонен клас”

4. Типизиране на методи

Подобно на класовете методите също могат да бъдат типизирани. Типизирането на метод се налага когато не е известно от какъв тип ще са параметрите на метода и означава, че в него ще бъдат използвани неизвестни типове. Едва при извикването на метода се определя кои ще бъдат неизвестните типове като типовете параметри се заменят с конкретни.

Синтаксисът на типизирането на метод е следният:

Модификатор тип ИмеМетод <К> (списък параметри)
{
}

В случая К е параметър на тип, който при дефиницията на метода е неизвестен.

Пример:

```
public static void Swap<K>(ref K a, ref K b)
      {
         K c;
         c = a;
         a = b;
         b = c;
      }
```

Извикването на типизиран метод е по следния начин:

```
int x = 10, y = 20;
            Console.WriteLine("x={0} y={1}",x,y);
            Swap <int>(ref x, ref y);
     Console.WriteLine("x={0} y={1}", x, y);
     string s1 = "Hello";
     string s2 = "World";
     Console.WriteLine(s1 + " " + s2);
     Swap<string>(ref s1, ref s2);
     Console.WriteLine(s1 + " " + s2);

         Point<int> p3 = new Point<int>();
         p3.X = 50;
         p3.Y = 200;
         Swap(ref p1, ref p3);
         p1.Info();
         p3.Info();
```

```
public void Swap <K> (ref K a, ref K b)
{
  K c;
  c=a;
  a=b;
  b=c;
}
```

Пример за извикване на типизиран метод:

```
int x=5;
int y=4;
Swap<int> (ref x, ref y);


String s1 =”Hello”;
String s2=”World”;
Swap<string>(ref s1, ref s2);
```

Относно въпроси по темата на адрес: ln_zh_st@yahoo.com