# 第30章 インデクサ

前章でフィールドが配列になっていたとき、プロパティは使えません。

C#では、インデクサ(indexer)というものが利用できます。これを使うとオブジェクトを配列のように扱うことができます。インデクサは次のように宣言します。

```
要素のデータ型 this[int Index]
{
 get
 {
  return フィールド名[Index]
 }
 set
 {
  フィールド名[Index] = value;
 }
}
```

ちょっと注意すべき点は、インデクサには名前がありません。かわりにthisキーワードを 使っています。Indexはユーザーからもらってきた番号で名前は自由です。またIndexのデータ型はほとんどの場合int型ですが、絶対int型でないとダメだというわけでもありません。

通常インデクサは、他のクラスから利用するのでpublicとなります。

あれこれ説明するより、例を見た方がわかりやすいです。

```
// indexer01.cs

using System;

class MyIndexer
{
    string[] name;

    public string this[int n]
    {
        get
        {
            return name[n];
        }
        set
        {
            name[n] = value;
        }
    }

    public MyIndexer(int a)
    {
        name = new string[a];
    }
}


class indexer01
{
    public static void Main()
    {
        MyIndexer mi = new MyIndexer(5);

        string[] myname = new string[5]
            { "一郎", "次郎", "三郎", "四郎", "五郎"};

        for (int i = 0; i < 5; i++)
            mi[i] = myname[i];

        for (int i = 0; i < 5; i++)
            Console.WriteLine("mi[{0}] = {1}", i, mi[i]);
    }
}
```

MyIndexerクラスには、string型配列のnameフィールドがあります。

インデクサは、次のように宣言していますね。

```
public string this[int n]
{
 ...
}
```

そして、アクセッサは次のように宣言しています。

```
get
{
 return name[n];
}
set
{
 name[n] = value;
}
```

プロパティのアクセッサと同様です。ただ違うのは、nという配列の添え字が有る点です。

さて、呼び出し側では、まずMyIndexerクラスをインスタンス化しています。この時

```
MyIndexer mi = new MyIndexer(5);
```

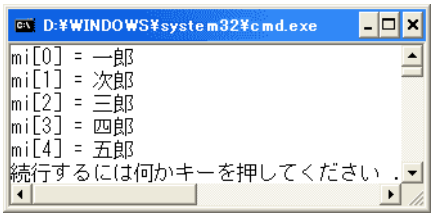
と、いうようにコンストラクタを利用して、５個分の要素を確保しています。

次に、このフィールドの配列に値を代入するわけですが、いきなり

### オブジェクト名[添え字]

でアクセスしています。決して「オブジェクト名.name[添え字]」ではありません。インデクサに名前が無いことからも推測できますね。これが、オブジェクトをあたかも配列のように扱えると書いた意味です。

では、実行結果を見てみましょう。




さて、最初にも書きましたが、インデクサの添え字は必ずしもint型である必要はありません。また、絶対に対応した配列のフィールドが必要というわけでもありません。

次の例を見てください。

```
// indexer02.cs

using System;

class MyIndexer
{
   public string this[string str]
    {
        get
        {
            switch (str)
            {
                case "一郎":
                    return "いちろう";
                case "次郎":
                    return "じろう";
                case "三郎":
                    return "さぶろう";
                case "四郎":
                    return "しろう";
                case "五郎":
                    return "ごろう";
                default:
                    return "読めません";
            }
        }
    }
}

class indexer02
{
    public static void Main()
```

```
    {
        string strFormat = "「{0}」の読み方は「{1}」です";
        MyIndexer mi = new MyIndexer();

        Console.WriteLine(strFormat, "次郎", mi["次郎"]);
        Console.WriteLine(strFormat, "五郎", mi["五郎"]);
        Console.WriteLine(strFormat, "猫", mi["猫"]);
    }
}
```

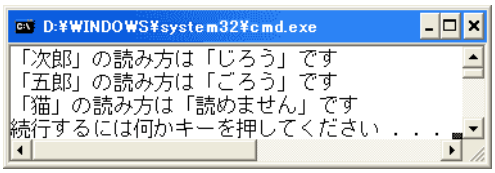
MyIndexerクラスには、フィールドはありません。インデクサのみです。しかも、インデクサの添え字はstring型です。

このインデクサのgetアクセッサでは、ユーザーが指定した添え字に対して、switch文で場合分けをして、文字列をreturnしています。setアクセッサはありません。

どうも、このインデクサは漢字の添え字に対して、その読みをreturnしているようです。こんなことをしてもよいのでしょうか。

結論はよいのです。

では、実行結果を見てみましょう。




なるほど、うまくいっていますね。

さて、インデクサやプロパティはそれ自身記憶領域を持ちません。従ってrefやoutとして メソッドの引数として渡すことはできません。

では、インデクサは配列のように高次元にすることができるのでしょうか。また添え字のデータ型が異なれば、メソッドのようにオーバーロードできるのでしょうか。さらに、インスタンスではなく、staticにして呼び出せるのでしょうか。いろいろ疑問がわいてきますね。

---

[C# Index] [総合Index] [Previous Chapter] [Next Chapter]

*Update 05/Sep/2006 By Y.Kumei*