R流
Visual Studio 2008 C# の驚異的な生産性を知る

2008年03月29日
R・田中一郎
http://blogs.wankuma.com/rti/
Microsoft MVP for Development Tools - Visual C#

アジェンダ

- はじめに
- コード比較
- 新機能の紹介
- 新機能の応用
- まとめ

はじめに

- つい先日発売した Visual Studio 2008
- 多くの機能が追加された
- 言語仕様の大幅な変更
- 生産性が向上
- C#を使って説明

コード比較

最初に C#3,0 の新機能を盛り込んだコードをご紹介します。

その後に、全く同じことを C#2.0 で実現するためのコードをご紹介します。

両者の違いをご確認下さい。

＜DEMO＞

コード比較 – C#3.0

```
public partial class Form3 : Form {
    public Form3() {
        InitializeComponent();
        var c = new[] {
            new { Code = 51 , Name = “ぽぴ王子”, Age = 18 },
            new { Code = 34 , Name = “R・田中一郎”, Age = 18 },
            new { Code = 111, Name = "IIJIMAS”. Age = 20 }};
            var q =
                from x in c
                where x.Age == 18
                orderby x.Name
                select new { Code = x.Code, Name = x.Name };
        listBox.DataSource = q.ToArray();
        listBox.ValueMember = "Code";
        listBox.DisplayMember = "Name";
        listBox.SelectedValueChanged +=  (sender, e) => Value = listBox.SelectedValue;
    }
    public object Value { get; private set; }
}
```

コード比較 – C#2.0

```
public class Member {
	private int _Code;
	public int Code {
		get { return this._Code; }
		set { this._Code = value; }
	}

	private string _Name;
	public string Name {
		get { return this._Name; }
		set { this._Name = value; }
	}

	private int? _Age;
	public int? Age {
		get { return this._Age; }
		set { this._Age = value; }
	}
}
```

コード比較 – C#2.0

```
public class Item {
	private int _Code;
	public int Code {
		get { return this._Code; }
		set { this._Code = value; }
	}

	private string _Name;
	public string Name {
		get { return this._Name; }
		set { this._Name = value; }
	}
}
```

コード比較 – C#2.0

```
public partial class Form2 : Form {
    public Form2() {
        InitializeComponent();

        Member popi = new Member();
        popi.Code = 51;
        popi.Name = “ぽぴ王子”;
        popi.Age = 18;

        Member rti = new Member();
        rti.Code = 34;
        rti.Name = “ R・田中一郎”;
        rti.Age = 18;

        Member iijimas = new Member();
        iijimas.Code = 111;
        iijimas.Name = "IIJIMAS";
        iijimas.Age = 20;

        Member[] c = new Member[] {popi, rti, iijimas };
```

コード比較 – C#2.0

```
        List<Item> list = new List<Item>();
        foreach (Member x in c) {
                if (x.Age == 18) {
                        Item item = new Item();
                        item.Code = x.Code;
                        item.Name = x.Name;
                        list.Add(item);
                }
        }
        list.Sort(
                delegate(Item a, Item b) {
                        return string.Compare(a.Name, b.Name); });
        listBox.DataSource = list;
        listBox.ValueMember = "Code";
        listBox.DisplayMember = "Name";
        listBox.SelectedValueChanged += delegate(object sender, EventArgs e) {
                Value = this.listBox.SelectedValue;
        };
}
```

コード比較 – C#2.0

```
        private object _Value;
        public object Value {
                get { return this._Value; }
                private set { this._Value = Value; }
        }
}
```

コード比較のまとめ

- C#3.0の新機能を上手に使う
  - –コードの記述量が減る
    - タイピング時間の減少
    - 可読性の向上
    - バグを含みにくいコード

新機能の紹介

C#3,0 の新機能を上手に使うことで全体的なコードの記述量が減られることがわかりました。

では、どのような新機能があるのでしょうか？

新機能の紹介 -暗黙的型付(Implicitly typed local variables)

C#3.0

```
var i = 5;
var s = “A”;
var v = GetValue();
var popi = new Member();
```

C#2.0

```
int i = 5;
string s = “A”;
double v = GetValue(); // GetValue の戻り値の型による
Member popi = new Member();
```

新機能の紹介 - 自動プロパティ(Automatic Properties)

C#3.0

```
public object Value { get; private set; }}
```

C#2.0

```
private object _Value;
public object Value {
        get { return this._Value; }
        private set { this._Value = Value; }
}
```

新機能の紹介 - オブジェクト初期化子(Object Initializers)

C#3.0

```
var popi = new Member {
    Code = 51 , Name = “ぽぴ王子”, Age = 18 },
```

C#2.0

```
Member popi = new Member();
popi.Code = 51;
popi.Name = “ぽぴ王子”;
popi.Age = 18;
```

新機能の紹介 -匿名型(Anonymous types)

C#3.0

```
var popi = new {
    Code = 51 , Name = “ぽぴ王子”, Age = 18 },
```

C#2.0

```
Member popi = new Member();
popi.Code = 51;
popi.Name = “ぽぴ王子”;
popi.Age = 18;
```

新機能の紹介 - コレクション初期化子(Collection initializers)

C#3.0

```
var c = new List<Member> { popi, rti, iijimas };
```

C#2.0

```
List<Member> c = new List<Member>();
c.Add(popi);
c.Add(rti);
c.Add(iijimas);
```

新機能の紹介 -ラムダ式(Rambda expressions)

C#3.0

```
listBox.SelectedValueChanged +=
    (sender, e) => Value = listBox.SelectedValue;
```

C#2.0

```
listBox.SelectedValueChanged +=
    delegate(object sender, EventArgs e) {
        Value = this.listBox.SelectedValue;
    };
```

新機能の紹介 - 拡張メソッド(Extension methods)

C#3.0

```
public static int GetByteLength(this string value) { /* 略 */ }
public void Hoge() {
    var byteLength = “あいうeo”.GetByteLength();
}
```

C#2.0

```
public static int GetByteLength(string value) { /* 略 */ }
public void Hoge() {
    var byteLength = GetByteLength(“あいうeo”);
}
```

新機能の紹介 - LINQ(Language Integrated Query)

C#3.0

```
var q =
    from x in c
    where x.Age == 18
    orderby x.Name
    select new { Code = x.Code, Name = x.Name };
```

新機能の紹介 - LINQ(Language Integrated Query)

C#2.0

```
List<Item> list = new List<Item>();
foreach (Member x in c) {
    if (x.Age == 18) {
        Item item = new Item();
        item.Code = x.Code;
        item.Name = x.Name;
        list.Add(item);
    }
}
list.Sort(delegate(Item a, Item b) {
    return string.Compare(a.Name, b.Name); });
```

新機能の応用

C#3,0 の新機能を上手に使うことで全体的なコードの記述量が減られることがわかりました。

では、どのような新機能があるのでしょうか？

＜DEMO＞

新機能の応用

```
public FormS() {
    InitializeComponent();
    var c = new[] { /* 省略 */ }; // 最初のサンプルと同じなので省略

    var q1 =
        from x in c
        where x.Age == 18
        orderby x.Name
        select new { Code = x.Code, Name = x.Name };

    var q2 = c
        .Where(x => x.Age == 18)
        .OrderBy(x => x.Name)
        .Select(x => new { Code = x.Code, Name = x.Name });

    var q3 = c
        .条件(x => x.Age == 18)
        .整列(x => x.Name)
        .選択(x => new { Code = x.Code, Name = x.Name });
```

```
    listBox1.Binding(q1.ToArray(), "Code", "Name");
    listBox2.Binding(q2.ToArray(), "Code", "Name");
    listBox3.Binding(q3.ToArray(), "Code", "Name");
}

public static void Binding<T>(
    this T x,
    object dataSource,
    string valueMember,
    string displayMember)
        where T : ListControl
{
    x.DataSource = dataSource;
    x.ValueMember = valueMember;
    x.DisplayMember = displayMember;
}
```

新機能の応用

```
public static IEnumerable<T> 選択<T>(
    this IEnumerable<T> value,
    Func<T, bool> func)
{
    var r = new List<T>();
    foreach(var x in value) if (func(x)) r.Add(x);
    return r;
}

public static IEnumerable<TR>選択<TS, TR>(
    this IEnumerable<TS> value,
    Func<TS, TR> func)
{
    return value.Select(x => func(x));
}
```

新機能の応用

```
public static IEnumerable<TS>整列<TS, TK>(
    this IEnumerable<TS> value,
    Func<TS, TK> func)
        where TK : IComparable
{
    var a = value.ToArray();
    for(var i = 0; i < a.Length; ++i) {
        var min = i;
        for(var j = i + 1; j < a.Length; ++j) {
            if (func(a[j]).CompareTo(func(a[min])) < 0) min = j;
        }
        if (i != min)  {
            var t = a[min];
            a[min] = a[i];
            a[i] = t;
        }
    }
    return a;
}
```

まとめ

- LINQ のための新機能
- LINQ だけではない！
- 上手に使って生産性アップ！

ご清聴
ありがとうございました

2008年03月29日

R・田中一郎

http://blogs.wankuma.com/rti/

Microsoft MVP for Development Tools - Visual C#