Blog

Entries

Summary

Listed by:

Date

February 2010

January 2010

December 2009

November 2009

October 2009

September 2009

August 2009

July 2009

June 2009

May 2009

April 2009

March 2009

February 2009

January 2009

December 2008

November 2008

October 2008

September 2008

August 2008

< Previous Next >

August 25

自Delphi 7以来的新语法特性

New Delphi language features since Delphi 7

内联函数（Inlining）

D7中的inline关键字作为保留字并不会对编译器产生实际作用，在2009中此关键字起到内嵌到代码中起到实际作用。语法如下：

```
function foo: Integer; inline;
```

内部函数/过程也可以使用，但在D2009测试版中，类方法的内部函数使用inline后不认Self指针；类的子过程/子函数，也可以使用inline关键字，但没有实际效果，且虚方法/继承方法（virtual/override）不能使用。

重载运算符（Operator Overloading）

可以重载部分运算符，如+、-、类型转换等，在D2006只支持到record，但从2007开始支持到Class，以下示例修改自官网：

```
TMyClass = class
  // Addition of two operands of type TMyClass
  class operator Add(a, b: TMyClass): TMyClass;
  // Subtraction of type TMyClass
  class operator Subtract(a, b: TMyClass): TMyclass;
  // Implicit conversion of an Integer to type TMyClass
  class operator Implicit(a: Integer): TMyClass;
  // Implicit conversion of TMyClass to Integer
  class operator Implicit(a: TMyClass): Integer;
  // Explicit conversion of a Double to TMyClass
  class operator Explicit(a: Double): TMyClass;
end;

class operator TMyClass.Add(a, b: TMyClass): TMyClass;
begin
  //...
end;

var
  x, y: TMyClass
begin
  x := 12; // Implicit conversion from an Integer
```

July 2008
June 2008
May 2008
April 2008
March 2008
February 2008
January 2008
December 2007
November 2007
October 2007
September 2007
August 2007
July 2007
June 2007
May 2007
April 2007
March 2007
February 2007
January 2007
December 2006
November 2006
October 2006
September 2006
August 2006
July 2006
June 2006
May 2006

```
    y := x + x; // Calls TMyClass.Add(a, b: TMyClass): TMyClass
  end;
```

类助手（Class Helpers）

Helper是对原Class的扩展，是我们在不修改原类的基础上增加类方法，并加入原类的空间中。在Delphi中，对对象的调用实际上采用了两个步骤，首先是把对象地址放入eax寄存器中，然后call类方法，所以如果不使用继承类增加数据的话，用父类调用继承类的方法是没问题的，所以其实这样的方法在D7中也可以使用，但却很麻烦。所以Class Helper起到的就是这个作用，在Class Helper中可以增加的就是与实例无关的内容，所以任何需要增加实例Size的活VMT的功能不能声明，例如变量、虚方法等，但只占用类空间的没关系，如class var。在应用上我们可以通过这种方法方便的给VCL一类控件加上某个属性。

```
  TFoo = class helper for TControl
    private
      function GetA: Integer;
    public
      class var X: Integer;
      procedure MSG(var Message: TMessage); message WM_MYMESSAGE;
      procedure ProcFoo;
      property A: Integer read GetA;
    end;
  // ...
  procedure TForm1.Foofoo;
  begin
    ProcFoo; // TControl -> TWinControl -> TScrollingWinControl-> TCustomForm -> TForm -
> TFrom1: Call TFoo.ProcFoo
  end;
```

strict关键字（Keyword “strict”）

众所周知，在Delphi中，类的private和protected域中的变量可以被同一单元中可以自由的被访问（Delphi的类没有“友元”的概念，但同一个unit中可以说自动友元化了），而并非是真正的私有或只能被继承类访问。而strict关键字的作用就是使该内容变成严格OO意义上的private/protected作用域，这点没有什么多说的。语法：

```
  strict private
    // Blah...
  strict protected
    // Blah...
```

结构方法（Records with Methods）

也没什么特别的，就是和class差不多，就一个不用创建和销毁、不能继承、没有作用域之类的类，很容易掌握，所以这里就不多介绍了。但是很有意思的是带参数的constructor可以通过编译，可能是为了初始化的方便吧。

抽象类和固实类（Abstract and Sealed Classes）

这两个概念在OO中也并不陌生，抽象类是不应该创建实例的（但D2006起的编译器就不给检查，连个Warning都没有，这还有啥用啊 -.- ），而固实类是不能被继承的。语法：

```
  TAnAbstractClass = class abstract  // or (TParentClass)
```

April 2006
March 2006
February 2006
January 2006
December 2005
November 2005
October 2005
September 2005
August 2005
July 2005
June 2005
May 2005
April 2005

```
  // Blah...
end;
TASealedClass = class sealed(TAnAbstractClass) // or empty
  // Blah...
end;
```

类常量、类变量、类属性与静态类方法（Class const/var/property and Static Class Methods）

老的Delphi中只提供了类方法，而没有提供类变量、类常量和类属性，这真的是很不方便。这里先区分一下我所使用的类（Class）和对象（Object）即类的实例（Instance of Class）。当在Delphi中声明一个类的时候，这个类是有实际地址的，该地址记录了许多类的相关信息，比如实例的Size啊、虚方法信息啊一堆东西，而创建一个对象的时候则把类实例化，在堆（Heap）中分配一块地址，包括内部数据和VMT之类的东西。在调用实例的时候，首先要知道对象地址，然后才能访问内部变量和调用方法时使用Self指针即实例地址；而在调用类方法的时候，eax中的并不是实例的地址而是类的地址，然后再call方法，这时的Self指针并非实例地址而是类地址。所以对于每一个类和继承类来说，包括它和它的继承类的所有实例，类变量、常量都是同一个，这样就存在了一个唯一的可供使用的变量或常量，方便同步并且不需要使用较多的内存（可以参考C#中的类，不过C#中不允许从实例直接访问类变量、常量、方法）。而静态类方法则是在使用这个类方法的时候不传入class地址，也就是说没有Self指针，这样的类方法的访问开销要小于普通的类方法；这自然也就意味着，该类方法不能被继承（不能virtual/override）。另外，类属性的get/set方法必须使用静态类方法。

```
TFooClass = class
private
  class procedure SetFoo(const Value: Integer); static; // A Static Class Method
protected
  class var FX : Integer; // class var
public
  const FC: Integer = 10; // class const
  class procedure VirtualProc; virtual;
  class property X: Integer read FX write FX; // class property
  class property Foo: Integer read FC write SetFoo;
end;
```

类内部类型与嵌套类（Class Types and Nested Classes）

可以说现在的Class的域几乎相当于原来的整个unit，以前不能放里面的元素现在都可以放里面了，这个也没什么好多说的，试验一下就很容易明白了。

终方法（Final Methods）

这个也是建立在虚方法的基础上的，在override后使用final关键字，则表示该虚方法不能再被子类继承下去了。

```
TAClass  = class
public
  procedure Foo; virtual;
end;
TFinalMethodClass  = class(TAClass)
public
  procedure Test; override; final; // A Final Method
end;
```

For-in循环（For-in Loop）

这个应该是受.Net影响吧，支持遍历一个数组或提供了GetEnumerator函数的类。GetEnumerator要求返回一个类的实例，该类包含有Current属性和MoveNext方法。

```
procedure Foo(List: TStrings);
  i : Integer;
  lst : array[0..100]of Integer;
  s : string;
begin
  for i in lst do ;
  for s in List do ; // Support of TStrings.GetEnumerator
end;
```

Next: Delphi 2009测试版的新语法特性
Back: 引子

9:57 PM | Blog it | 计算机与 Internet

Comments

To add a comment, sign in with your Windows Live ID (if you use Hotmail, Messenger, or Xbox LIVE, you have a Windows Live ID). Sign in

Don't have a Windows Live ID? Sign up

Trackbacks

The trackback URL for this entry is:
http://egust.spaces.live.com/blog/cns!98563909D0913C04!548.trak
Weblogs that reference this entry

- None