

Date

# Visual Studio 2010 DSL 开发与设计实践系列

## 第 8 讲 DSL 的代码生成（上）

# 自我介绍

- 陈俊先
- .NET 高级讲师

# 内容提要

- 代码文档对象模型(**CodeDOM**)
- **CodeCompileUnit**
- **CSharpCodeProvider**
- **CompilerResults**
- 生成类
- 生成类成员
  - 字段
  - 属性
  - 方法
- 生成方法调用
- 赋值语句

# 代码文档对象模型(CodeDOM)

- CodeDOM 提供了表示许多常见的源代码元素类型的类型。您可以设计一个生成源代码模型的程序，使用 CodeDOM 元素构成一个对象图。可以使用受支持的编程语言的 CodeDOM 代码生成器，将该对象图呈现为源代码。CodeDOM 也可以用于将源代码编译成二进制程序集。
- **CodeDOM 图的结构**
- CodeDOM 图的结构类似于一个容器树。每个可编译的 CodeDOM 图的最顶部（即根位置）的容器都是 CodeCompileUnit。源代码模型的每个元素都必须通过图中 CodeObject 的属性链接到图中。

# CodeCompileUnit

- CodeCompileUnit 为 CodeDOM 程序图形提供容器。
- CodeCompileUnit 包含以下几个集合：可以存储包含 CodeDOM 源代码图形的 [CodeNamespace](#) 对象的集合、项目引用的程序集的集合，以及项目程序集的属性集合。

# CSharpCodeProvider

- 提供对 C# 代码生成器和代码编译器的实例的访问。
- 通过GenerateCodeFromCompileUnit()方法产生C#代码。
- 通过CompileAssemblyFromFile()方法编译源代码。

# CompilerResults

- 从编译器返回的编译结果。
- Errors属性表示错误结果。

# Demo (CodeDOM\_HelloWorld)

# 生成类

- CodeTypeDeclaration newType = new CodeTypeDeclaration(
  - "TestType");
  - newType.Attributes = MemberAttributes.Private;
  - newType.BaseTypes.Add( "BaseType" );
- 源码:
- // class TestType : BaseType
- // {
- // }

# 生成字段

- // 生成类
- CodeTypeDeclaration type1 = new CodeTypeDeclaration("FieldTest");
- // 生成字段
- CodeMemberField field1 = new CodeMemberField("System.String", "TestStringField");
- type1.Members.Add( field1 );
- // 构造方法
- CodeConstructor constructor1 = new CodeConstructor();
- constructor1.Attributes = MemberAttributes.Public;
- type1.Members.Add( constructor1 );
  
- 源码:
- // public class FieldTest
- // {
- // private string testStringField;
- //
- // public FieldTest()
- // {
- // }
- // }

# 生成属性

- `// Declares a property of type String named StringProperty.`
- `CodeMemberProperty property1 = new CodeMemberProperty();`
- `property1.Name = "StringProperty";`
- `property1.Type = new CodeTypeReference("System.String");`
- `property1.Attributes = MemberAttributes.Public;`
- `property1.GetStatements.Add( new CodeMethodReturnStatement( new CodeFieldReferenceExpression(new CodeThisReferenceExpression(), "testStringField") ) );`
- `property1.SetStatements.Add( new CodeAssignStatement( new CodeFieldReferenceExpression(new CodeThisReferenceExpression(), "testStringField"), new CodePropertySetValueReferenceExpression());`

- 源码:
- `// public virtual string StringProperty`
- `// {`
- `// get`
- `// {`
- `// return this.testStringField;`
- `// }`
- `// set`
- `// {`
- `// this.testStringField = value;`
- `// }`
- `// }`

# 生成方法

- CodeMemberMethod method1 = new CodeMemberMethod();
  - method1.Name = "MyMethod1";
  - method1.ReturnType = new CodeTypeReference("System.String");
  - method1.Parameters.Add( new CodeParameterDeclarationExpression("System.String", "text") );
  - method1.Statements.Add( new CodeMethodReturnStatement( new CodeArgumentReferenceExpression("text") ) );
- 
- 源码:
  - private string MyMethod1(string text)
  - {
    - return text;
  - }

# 生成方法调用

- CodeMethodInvokeExpression methodInvoke =  
new CodeMethodInvokeExpression(
  - new CodeThisReferenceExpression(),
  - "MyMethod1",
  - new CodeExpression[] { new  
CodePrimitiveExpression(true) } );
- 源码:
- this. MyMethod1(true);

# 生成赋值语句

- `CodeAssignStatement as1 = new CodeAssignStatement(new CodeVariableReferenceExpression("i"), new CodePrimitiveExpression(10));`
- 源码:
- `// i=10;`

# Demo(SampleCodeDom)

# CodeDOM的缺点

- 1、硬编码繁琐。有时，编码长度比生成的程序还长。
- 2、缺少灵活性。



© 2009 Microsoft Corporation. All rights reserved. Microsoft, MSDN, the MSDN logo, and [list other trademarks referenced] are trademarks of the Microsoft group of companies. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.