CodeGen framework

Minor project

DINF3  
Kyle Schouten 0882858  
Sytse de Boer 0807098  
  
12-02-2016

Inhoud

[Redenen voor het gebruiken van het CodeGen framework 3](#_Toc443075675)

[Voorbeelden gebruik CodeGen in F# 3](#_Toc443075676)

[CodeGen Studio 5](#_Toc443075677)

# Redenen voor het gebruiken van het CodeGen framework

CodeGen maakt gebruik van het CodeDom framework van Microsoft. Hierin kan je in het geheugen classes, functies, variabelen etc. Dit kan uiteindelijk worden weggeschreven in een source file. Dit kan ik elke .net taal. Het voordeel hiervan ten opzichte van de huidige methode, namelijk het direct weggeschrijven in een bestand, is dat je hiermee geen platte tekst schrijft, maar code genereert. Doordat code genereerd wordt, is de kans op typfouten een stuk kleiner. Ook ben je nu niet gebonden aan een taal, maar kan je net zo gemakkelijk een andere .net taal gebruiken. Voor meer informatie over de functies van CodeGen, kunt u kijken in de help.

# Voorbeelden gebruik CodeGen in F#

public class EntryPoint  
{  
 public static bool Print(string s)  
 {  
 System.Console.WriteLine(s);  
 return true;  
 }  
}

Het codeblock hierboven wordt nu door F# genereerd door de volgende regel:  
sprintf "public class EntryPoint {\n public static bool Print(string s)\n{System.Console.WriteLine(s);\n return true;\n}\n}”

De nieuwe methode bouwt alles op in het geheugen en schrijft dit in een keer naar een bestand weg. Om dezelfde code te genereren kan je nu de volgende code gebruiken:

// Create the class

let cb = new ClassBuilder("EntryPoint", TypeAttributes.Public, false)

// Create the input parameter of the function

let parameterItems = [|new ParameterItem( Name = "s", Type = typeof<string>)|]

// Create the return statement

let returnStatement = new Statements.ReturnStatementLine()

returnStatement.Expressions.Add( 0, new Expressions.PrimitiveExpressionLine( Value = true ) )  
  
// Add the complete method to the class

cb.AddMethod<bool>("Print", parameterItems, CodeDom.MemberAttributes.Public ||| CodeDom.MemberAttributes.Static, [|statementLine;returnStatement|] ) |> ignore

// Create the Code compiler and publish it   
let cub = new CompileUnitBuilder("Bla")

let typeDecl = cb.GetDeclaration()

cub.AddClass(typeDecl) |> ignore

cub.PublishCode("c:\\ test.cs")

Dit is wel meer code dan de oude methode, maar nu wordt ook nog een Visual Basic, JScript en C++ bestand aangemaakt.

Het volgende codeblock is een voorbeeld van een conditie met alleen een true-block. De C# code die wordt gegenereerd staat in commentaar:

// if(printInput) System.Console.WriteLine(p.ToString());

let statementLine3 = new Statements.ConditionStatementLine()

statementLine3.Expressions.Add(0, new Expressions.VariableReferenceExpressionLine( VariableName = "printInput" ) )

let trueStatement = new Statements.ExpressionStatementLine()

trueStatement.Expressions.Add(0, new Expressions.MethodInvokeExpressionLine

( MethodName = "WriteLine",

TargetObject = new Expressions.TypeReferenceExpressionLine( Type = "System.Console" ),

Parameters = [|new Expressions.VariableReferenceExpressionLine( VariableName = "p" )|] ) )

statementLine3.TrueStatementLines <- [|trueStatement|]

De code hier onder laat zien hoe een interface gebouwd wordt.

let interface = new InterfaceBuilder(“TestInterface”)

interface.AddVoidMethod(“StartService”,[||]).AddProperty<bool>(“IsServiceStarted”)

.AddMethod<string>(“GetResult”,[| new ParameterItem( Name = "studentName", Type = typeof<string>)|])

Dit genereerd in C#

public interface TestInterface

{

bool IsServiceStarted { get; set; }

void StartService();

string GetResult( string studentName );

}

In VB genereerd dit als:  
Public Interface TestInterface

Property IsServiceStarted() As Boolean  
 sub StartService()  
 Function GetResult(ByVal studentName As String) As String  
End Interface

# CodeGen Studio

Met dit programma kan je door middel van een simpele interface code genereren. Dit was voornamelijk bedoeld voor het testen van onze functies, omdat dit moeilijk te debuggen is. Met deze tool is het alleen mogelijk om simpele kale classes, functies en fields te maken.