NET integration

Starting with version 0.9 HBasic has some implemented functions to work with the DOTGNU C# compiler and the NET libraries. The access to NET libraries and the C# compiler from the DOTGNU project are in an alpha state but they might already be used in development and will be extended in future versions.

Contents

If you want to compile C# programs see information about compiling C# programs. To understand the difference between the different versions of NET integration in HBasic see *Example hello world programs*.

Introduction

This document will show how you can access different parts of a NET library. Before you can start the examples you have to install DOTGNU-NET support and import a NET library with the package-editor.

A NET library is a file that follows the standards defined by Microsoft for the .NET development framework. It may be a default library like mscorlib.dll or a library that has been created with the C# compiler from the DOTGNU project. We also started some effort to create a HBasic NET compiler which will directly create this NET compatible code from a HBasic program but it's in a very early alpha state.

In this document we will reference the components of a small NET library that we have created from a C# program. You can find the sourcecode and information how to compile the C# program here. This program declares one NET class *MyClass* in a namespace *FooBar* with some public defined elements. It is similar to an examples program in the Microsoft C# description. The class exports the following elements:

- Method *getval* which returns an integer value
- Field *MyField* of type integer
- Constant value *MyConstant* with value 34
- Property *MyProperty* with read and write access allowed
- Event *MyEvent*

To access this components you will need an instance of the class MyClass. A NET class may also export static components which may be called without creating a class instance. You can access this components with namespace.class.component. Otherwise you create a variable *Bar* with type MyClass and call a component with Bar.compname.

Since MyClass is derived from System.Object from mscorlib.dll it includes a reference to this basic library. When you load the compiled class-file into the package editor you can also find the class-list from this package.

In the following paragraphs we will show how you can access this components from HBasic code. Currently there is no support for handling events from NET classes. This will follow later. Another open problem is how to transfer complex types between HBasic and NET code. This has to be implemented step by step in the following versions. You can find some of the example programs in the subfolder code_examples/net of your HBasic installation directory.

Calling static NET methods

Before we start to create instances of MyClass from the library we will first show how to call a static method from the library. We choose a method WriteLine from the mscorlib.dll that we display a text on your terminal. Once again remember to install NET support and load the library with the package editor before you try to start this examples.

Create a new project and insert a button named *button1* into the form. Insert the following sourcecode and start the HBasic interpreter.

```
Sub button1_clicked()
   System.Console.WriteLine( "Hello NET world" )
End Sub
```

Example net_hello.bas: Example of a hello_world program with NET static method call

As a result the text "Hello NET world" should be printed whenever your click on the button in the form. The text will be printed to your console and not to the current form as in the HBasic *Print* command. In this example HBasic takes the string and passes it to the method *WriteLine* in the class *Console* of the *System* namespace. Since this method is declared static you may call it without an instance of the class *MyClass*.

Creating NET class instances

For the following examples we need an instance of the class MyClass. To get this class instance you declare a variable with type namespace.classname and assign a new class instance to this variable. Our own class has been declared with namespace FooBar. The two statements we need to declare a class instance are:

```
Dim v As FooBar.MyClass
v = New FooBar.MyClass()
```

The following examples show how to use this class instance.

Call methods for a net class

In this example we want to call the Method getval of MyClass. The result value will be stored in the variable i and display with the WriteLine method of the mscorlib.dll package.

```
Dim n As FooBar.MyClass

Sub button1_clicked()
   Dim i As Integer

   n = New FooBar.MyClass()
   i = n . getval()
   System.Console.WriteLine( i )
End Sub
```

Example net_callmethod.bas: Example of a call to a net method.

Reading and writing fields of a net class

A field is a variable that will be stored in the memory of a class instance. This means if you have two instances c1 and c2 of the class they may store different values because each class instance allocates it's own memory segment.

In this example we want to read and write the field MyField for an instance of the class MyClass. Like a normal variable each field of a NET class has a special type. The field MyField has the NET type int32. We will read and print the initial value of this field, assign a new value 2244 to it and read the value again to check that has been modified successfully.

```
Dim n As FooBar.MyClass

Sub button1_clicked()
   n = New FooBar.MyClass()
   Print n . MyField
   n . MyField = 2244
   Print n . MyField
End Sub
```

Example net_fieldaccess.bas: Example of access to a net field.

Reading constants of a net class

A constant is a special form of field in a NET library. A constant field has a value before any instance of the class has been allocated. You can read the value of a constant field when you have loaded the library file into HBasic. Values of a constant field cannot be changed and will not be stored in the memory of a class instance.

In this example we want to show how to read the value of the constant MyConst from the class MyClass. The result value has been defined in the sourcecode of the C# class definition. You can read a const value in every expression in the same way as reading a field value.

```
Dim n As FooBar.MyClass

Sub button1_clicked()
   n = New FooBar.MyClass()
   Print n . MyConst
End Sub
```

Example net_constaccess.bas: Example of access to a net constant.

Read or write property value of a net class

In this example we want to read and write the value of the property MyProperty of MyClass. Using a property is treated like using a field in the HBasic code. The difference in the implementation of the NET class is that readinf or writing a property value will call a method in the library implementation. The class method that is called when reading the property value may compute the return value with whatever algorithm that's required.

Compared to HBasic classes you may compare a NET field with a classlocal variable in HBasic and a NET property with a HBasic property.

```
Dim n As FooBar.MyClass

Sub button1_clicked()
   n = New FooBar.MyClass()
   Print n . MyProperty
   n . MyProperty = 1133
   Print n . MyProperty
End Sub
```

Example net_fieldaccess.bas: Example of access to a net property.