Examples

This distribution of the Mono PHP Compiler includes a couple of examples contained in the directory `samples`. Please change to that directory if you would like to play around with them. The simplest one demonstrates the compilation and execution of a PHP script on its own. Another one additionally demonstrates how a compiled PHP script can be used from a class written in another .NET language. The other examples show how other .NET libraries can be accessed directly from within a PHP script.

Example (1)

In the file `Geometry.php` you'll find a PHP script defining some geometric classes to show the usage of some basic features that PHP offers:

```php
<?php

class Shape {
	const pi = 3.14159265;
}

class Circle extends Shape {
	public $radius;
	public function __construct($radius) {
		$this->radius = $radius;
	}
	public function GetArea() {
		return 0.5 * Shape::pi * $this->radius * $this->radius;
	}
	public function GetRound() {
		return 2 * Shape::pi * $this->radius;
	}
}

class Rectangle extends Shape {
	public $x;
	public $y;
	public function __construct($x, $y) {
		$this->x = $x;
		$this->y = $y;
	}
	public function GetArea() {
		return $this->x * $this->y;
	}
	public function GetRound() {
		return 2 * $this->x + 2 * $this->y;
	}
}
```

You can for example see some of PHP's object orientated features, so how classes, functions and constructors are defined, how classes inherit other classes and how class constants are defined.

In the next few statements we're playing a little with some objects. Like that you can see how objects are created, how their methods are invoked and how their members are accessed.

```
$c = new Circle(2);
echo '$c is a circle with radius 2\n';
echo 'area of $c = ' . $c->GetArea() . '\n';
echo 'round of $c = ' . $c->GetRound() . '\n\n';

$r = new Rectangle(2, 4);
echo '$r is a rectangle with lateral leghts ' . $r->x . ' and ' . $r->y . '\n';
echo 'area of $r = ' . $r->GetArea() . '\n';
echo 'round of $r = ' . $r->GetRound() . '\n\n';

echo '$c is ' . (!($c instanceof Shape) ? 'not ' : '') . 'a Shape' . '\n';
echo '$c is ' . (!($c instanceof Circle) ? 'not ' : '') . 'a Circle' . '\n';
echo '$c is ' . (!($c instanceof Rectangle) ? 'not ' : '') . 'a Rectangle' . '\n';
echo '$r is ' . (!($r instanceof Shape) ? 'not ' : '') . 'a Shape' . '\n';
echo '$r is ' . (!($r instanceof Circle) ? 'not ' : '') . 'a Circle' . '\n';
echo '$r is ' . (!($r instanceof Rectangle) ? 'not ' : '') . 'a Rectangle' . '\n';

?>
```

To compile the script, call

```
mono ../mPHP.exe Geometry.php
```

which will produce an executable file called `Geometry.exe`.
To run it, call

```
mono Geometry.exe
```

which will output the result as expected:

```
$c is a circle with radius 2
area of $c = 6.2831853
round of $c = 12.5663706

$r is a rectangle with lateral leghts 2 and 4
area of $r = 8
round of $r = 12

$c is a Shape
$c is a Circle
$c is not a Rectangle
$r is a Shape
$r is not a Circle
$r is a Rectangle
```

Example (2)

In the file `MathPHP.php` you'll find a PHP script defining two functions, one for calculating faculty numbers, the other one for fibonacci numbers:

```
<?php
class MathPHP {
	public static function Fib($a) {
		if ($a == 0)
			return 0;
		else if ($a == 1)
			return 1;
		else
			return self::Fib($a - 1) + self::Fib($a - 2);
	}
	public static function Fac($a) {
		if ($a == 0 || $a == 1)
			return 1;
		else
			return $a * self::Fac($a - 1);
	}
}
?>
```

Now we don't want to invoke them from inside the PHP script as in (1), but from a class written in another .NET language. For this purpose we'll compile the class with the `/target:library` option:

```
mono ../mPHP.exe /t:library MathPHP.php
```

which will produce an assembly file called `MathPHP.dll`.
Now have a look into the file `MathCS.cs` containing a C# class using the functions of the compiled PHP script:

```
public class MathCS {
	public static void Main(string[] args) {
		// calculating faculty of 5
		Object fac5 = MathPHP.Fac(5);
		System.Console.WriteLine("Faculty of 5 = " + fac5);
		// calculating fibonacci of 5
		object fib5 = MathPHP.Fib(5);
		System.Console.WriteLine("Fibonacci of 5 = " + fib5);
	}
}
```

We'll now compile this class. As it uses features of the compiled PHP script, it's important to refer to the assembly we just created and to the mPHPRuntime:

```
mcs /r:MathPHP.dll,mPHPRuntime.dll MathCS.cs
```

This will produce the executable file `MathCS.exe`. Calling `mono MathCS.exe` will output the result as expected:

```
Faculty of 5 = 120
Fibonaccy of 5 = 5
```

Example (3)

The other .php files all show how other .NET libraries can be accessed directly from within a PHP script. They all build simple Graphical User Interfaces using the Gnome libraries. I'll discuss one of these examples in this tutorial.

In the file `Button.php` you'll find a PHP script building a simple window containing a button. If this button is clicked, a short text will be written on the console.

```
<?php

using System;
using Gtk;

Application::Init();
$w = new Window("Gtk# Basics");
$b = new Button("Hit me");
add_event($w, "DeleteEvent", new DeleteEventHandler(null, Window_Delete));
add_event($b, "Clicked", new EventHandler(null, Button_Clicked));
$w->Add($b);
$w->SetDefaultSize(200, 100);
$w->ShowAll();
Application::Run();

function Window_Delete($o, DeleteEventArgs $args) {
      Application::Quit();
      $args->RetVal = true;
}

function Button_Clicked($o, EventArgs $args) {
      echo("Hello, World!\n");
}

?>
```

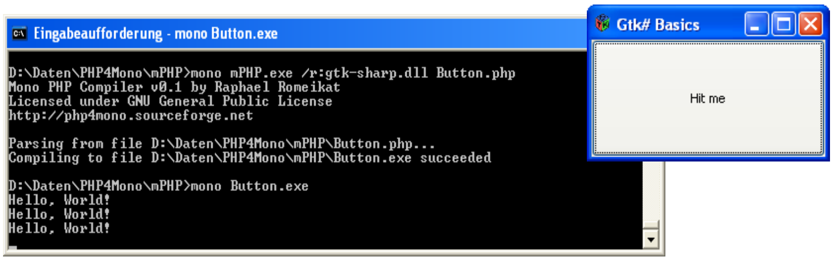
First of all, the two namespaces whose types are used in the script need to be declared. This is done by the using declaration which behaves in the same was as the one you know from C#. To compile this script, we now need to refer additionally to the Gtk# library which is used in this example. To do so, call

```
mono ../mPHP.exe /r:mPHPRuntime.dll,gtk-sharp.dll Button.php
```

which will produce an executable file called `Button.exe`.
To run it, call

```
mono Button.exe
```

which will show the window as expected as well as the short text when clicking the button:



```
D:\Daten\PHP4Mono\mPHP>mono mPHP.exe /r:gtk-sharp.dll Button.php
Mono PHP Compiler v0.1 by Raphael Romeikat
Licensed under GNU General Public License
http://php4mono.sourceforge.net

Parsing from file D:\Daten\PHP4Mono\mPHP\Button.php...
Compiling to file D:\Daten\PHP4Mono\mPHP\Button.exe succeeded

D:\Daten\PHP4Mono\mPHP>mono Button.exe
Hello, World!
Hello, World!
Hello, World!
```