# aardio 范例: .NET 委托自动支持 aardio 函数对象

```
//.NET 委托自动支持 aardio 函数对象
//.NET 静态委托回调自动支持 aardio 的回调函数，兼容低版本 .NET 2.0
import dotNet;
var compiler = dotNet.createCompiler("C#");
compiler.Source = /******
namespace CSharpLibrary
{
    public class Object
    {
        //定义一个委托类型
        public delegate int TestDelegateType(string str,int a);

        //定义一个委托类型字段
        public TestDelegateType callback;

        //函数与委托的参数、返回值类型要一致
        public int TestCall(string str,int a){
            return a + 900;
        }

        //函数参数传入委托
        public int Hello(TestDelegateType func)
        {
            return func("你好",123);
        }

        //函数返回委托
        public TestDelegateType GetTestDelegate( )
        {
            var n = callback("你好",123);

            //函数转换为委托
            //return new TestDelegateType(TestCall);

            //.NET 2.0 开始支持隐式转换,可以不用再写 new 了。
            return TestCall;
        }
    }
}
******/

compiler.import("CSharpLibrary"); //编译 C# 代码并导入名字空间
var netObj = CSharpLibrary.Object();//创建 .NET 对象

import console;

//函数参数传入委托
netObj.Hello(function(){
    //无论.NET 回调是否在同一线程，被 .NET 回调的 aardio 函数总是在原调用线程执行（不必考虑多线程规则与同步）。
    console.log("3、aardio 函数被 C# 调用了,参数:",a,b)
    return 2; //如果不返回委托指定类型的返回值会导致报错，委托返回值类型为 void 时这里可以不返回值。
})

//委托回调可以直接赋值为 aardio 函数对象
netObj.callback = function(a,b){
    console.log("1、aardio 函数被 C# 调用了,参数:",a,b)
    return 1;
}

//也可以这样追加委托回调函数
dotNet.delegate.combine(
    netObj,"callback",function(a,b){
        console.log("2、aardio 函数被 C# 调用了,参数:",a,b)
        return 2;
    }
)

//获取委托的回调对象数组
var list = netObj.callback.GetInvocationList()
console.log(list[1]);

//调用 C# 函数，可以在返回值里返回 .NET 委托
var resultDelegate = netObj.GetTestDelegate();

//直接调用 .NET 委托。
```

```
var ret = resultDelegate("调用函数返回的委托",123);
console.log("调用 C# 委托的返回值",ret);

//通过对像成员获取 .NET 委托对象
var callback  = netObj.callback;

//可以在 aardio 中直接调用 .NET 委托对象
callback("测试",123)

//下面这样写是作为对象的成员函数调用，委托字段不是成员函数所以会报错
//netObj.callback("测试",123)

console.pause();
```

Markdown 格式