aardio 范例: C# 使用 InvokeMember 调用 aardio

```
//C# 使用 InvokeMember 调用 aardio
import dotNet;
var compiler = dotNet.createCompiler("C#");
compiler.Parameters.CompilerOptions = "/optimize";

compiler.Source = /******
using System;
using System.Reflection;
using System.Collections;

namespace CSharpLibrary
{
    public class DynObject : IEnumerable
    {
        private object target;
        private Type type;
        public DynObject(object aardioObject) { target = aardioObject; type = target.GetType(); }
        IEnumerator IEnumerable.GetEnumerator() {return (target as IEnumerable).GetEnumerator(); }

        public object InvokeMember(string method, params object[] args) { return type.InvokeMember(method, BindingFlags.InvokeMethod, null, target, args); }
        public object InvokeMember(int dispId, params object[] args){ return type.InvokeMember("[DispId=" + dispId + "]", BindingFlags.InvokeMethod, null, target, args); }
        public object Invoke(params object[] args) {return type.InvokeMember("", BindingFlags.InvokeMethod, null, target, args); }

        public object this[string index]
        {
            get { return type.InvokeMember(index, BindingFlags.GetProperty, null, target, null); }
            set { type.InvokeMember(index, BindingFlags.SetProperty, null, target, new object[] { value }); }
        }
    }

    public class Object
    {
        public object Hello(object aardioObject)
        {

           //转换 aardio 对象
            DynObject tab = new DynObject(aardioObject);

            //调用成员函数
            tab.InvokeMember("执行aardio", " console.log('在C#中执行aardio代码') ");

             //调用 对象自身（如果对象不是函数而是表，调用表对象的 _call 元方法）
            tab.Invoke(12, 3, this);

            //遍历 aardio 对象
            foreach (object item in tab)
            {
                Console.WriteLine("C#中遍历aardio对象键名: " + item);
            }

            //修改属性
            tab["属性名"] = 456;

            //读取属性
            return tab["属性名"];
        }
    }
}
******/

compiler.import("CSharpLibrary");

//使用 C# 编写的类构造对象实例
var cSharpObject = CSharpLibrary.Object();

import console;
var aardioObject =  {
    属性名 = 123;
    执行aardio = function( code ){
        loadcode(code)();
        return 123;
    };
    @{
        _call = function(a,b,netObject){

            /*
            aardio 函数传给 .NET 时（委托、事件），
            aardio 会在原函数外包装一层代理函数，并自动处理回调参数。
            回调 aardio 函数的参数中的 .NET 对象会自动封装为 dotNet.object 。

            但 .NET 反过来直接回调 aardio 对象的成员函数时，
            回调 aardio 函数的参数中的 .NET 对象没有经过处理，不会自动封装为 dotNet.object 。

            原生 .NET 对象在 aardio 中存为 COM 对象，
            而原生 COM 接口不支持 .NET 的很多特性，例如重载（函数名相同，参数不同）。

            .NET 原生对象传入 com.IsNetObject() 会返回非 0 值。
            而 dotNet.object 传入 dotNet.getObject() 会返回非 null 值。

            解决方案：
            如下将原生 .NET 对象转换为 aardio 中的 dotNet.object 。
            */
            netObject = dotNet.object(netObject);

            //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("aardio 对象被 C# 调用了，参数:",a,b)
            return a + b;
        }
    }
}

//调用实时编译的 C# 函数
var ret = cSharpObject.Hello( aardioObject );

console.dump(ret);
console.dumpJson(aardioObject);
console.pause();
```

Markdown 格式