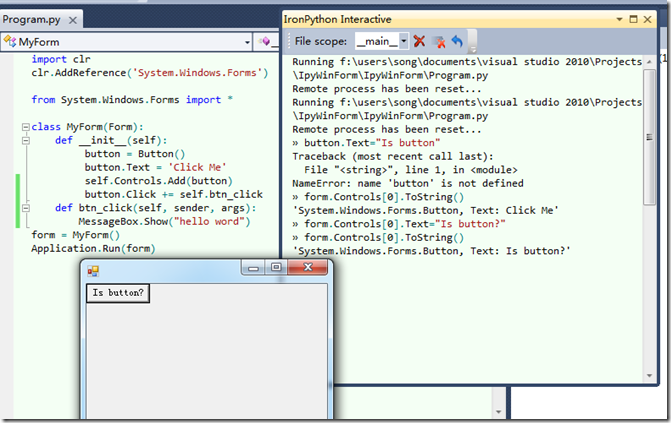
IronPython 简 介

IronPython 是一种在 .NET 及 Mono上的 Python 实现，由微软的 Jim Hugunin 所发起，是一个开源的项目，基于微软的 DLR 引擎；托管于微软的开源网站 CodePlex([www.codeplex.com](http://www.codeplex.com/))。

开源项目链接：<http://ironpython.codeplex.com/>

以下是本人一个简单的Winform程序，使用Interative在运行时的修改，动态语言的魅力就在于此吧！？

[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/201105211217432078.png)

简单说明：新建Ipy Winform程序，为button添加btn\_click事件，运行

输入button.Text="Is button"回车后出错（如图）

思考 button为运行时动态添加，非全局类型，故改用form获取其第一个子控件既为button

其它略，见图

链接：

【51CTO精选译文】 [IronPython入门：什么是IronPython？](http://developer.51cto.com/art/200910/156377.htm)

博客园 [小蟒蛇IronPython 团队](http://www.cnblogs.com/team/ipy/)

IronPython  IDE

版本：

当前语言正式版本：[IronPython 2.7](http://ironpython.codeplex.com/releases/view/54498)

主流 IDE:

推荐：Visual Studio 2010 : [IronPython 2.7 for .NET 4.0](http://ironpython.codeplex.com/releases/view/54498) (登录网站2.7下载默认为vs2010，安装后vs新建项目出现IronPython)

or Visual Studio 2008  ： IronPython 2.6.1 RC1 for .NET 2.0 （注意版本的选择，我在XP下配置了改方案）

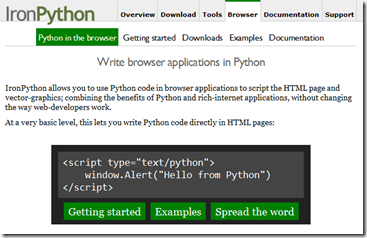
or [pydev + Eclipse](http://pydev.org/download.html)||Aptana Studio 3 (既然你是慕名C#而来，为啥选择Eclipse，呵呵)

or [Visual Studio 2008 shell + ironpythonstudio](http://ironpythonstudio.codeplex.com/) （2008年最后一次更新，不过可视化比较优秀，建议使用以上方案）

IronPython 其  它

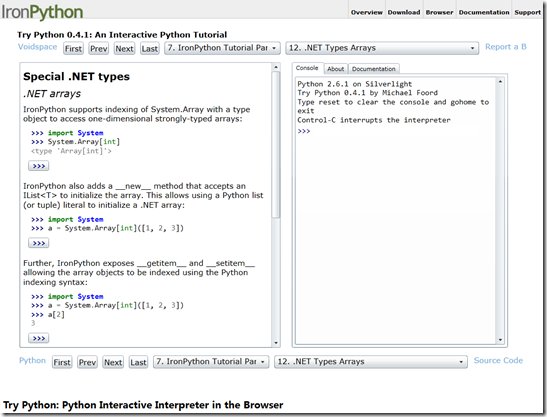
1.  <http://ironpython.net/browser/> （英文，墙外）

介绍了 IronPython 在浏览器上作为脚本语言的使用

[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/201105211217479141.png)

2.  <http://ironpython.net/try/>   （英文，墙外）

使用silverlight（IronPython开发），Python/IronPython 在线代码编辑器，同时提供了简单的语法教程

[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/201105211217484058.png)

目标：

在WinForm程序中，让IronPython 与C#进行交互。

说明：

本例中在WinForm中嵌入了Ipy脚本，在WinForm执行与输出Ipy脚本运行结果，在WinForm运行时修改窗体属性与控件属性

测试环境：

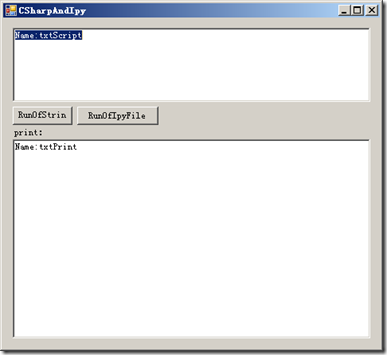
Window：VS2010+IronPython 2.7

[参考： IronPython <零> 前言](http://www.cnblogs.com/SongSharp/archive/2011/05/21/2052688.html)

过程：

1. C#部分

使用C#语言新建一个WinForm，界面如下

[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/201105222247214729.png)

代码：

using System;

using System.Windows.Forms;

using IronPython.Hosting;

using System.Drawing;

namespace CSharpAndIpy

{

public partial class FrmMain : Form {

public FrmMain()

{

InitializeComponent();

}

private void btnRunOfString\_Click(object sender, EventArgs e)

{

try {

//创建一个IpyRunTime，需要2-3秒时间。建议进入全局时加载，此为演示 var engine = IronPython.Hosting.Python.CreateEngine();

var code = engine.CreateScriptSourceFromString(txtScript.Text);

var actual = code.Execute<object>();//执行code脚本，获取输出值actual txtPrint.Text = actual.ToString();

}

catch (Exception ex)

{

txtPrint.Text = ex.ToString();

}

}

//执行事先准备的my.py脚本 private void btnRunOfIpyFile\_Click(object sender, EventArgs e)

{

var engine = Python.CreateEngine();

var scope = engine.CreateScope();

scope.SetVariable("Ipy\_this", this);//将this Set 到Ipy脚本的Ipy\_this值中 var code = engine.CreateScriptSourceFromFile("my.py");//执行my.py code.Execute(scope);

}

/// <summary> /// 测试方法，要为public才能在Ipy脚本中访问 /// </summary> public void testMethod(int R,int G,int B)

{

try {

txtScript.BackColor = Color.FromArgb(R, G, B);

}

catch { }

}

}

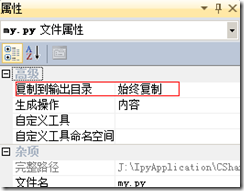
}

2：Ipy部分

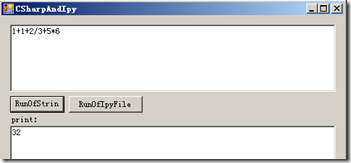
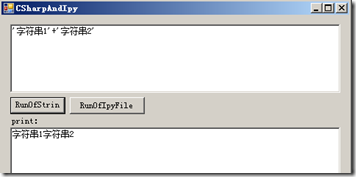
my.py 脚本(小例就简单3行吧)注：txtPrint控件被我设置为public

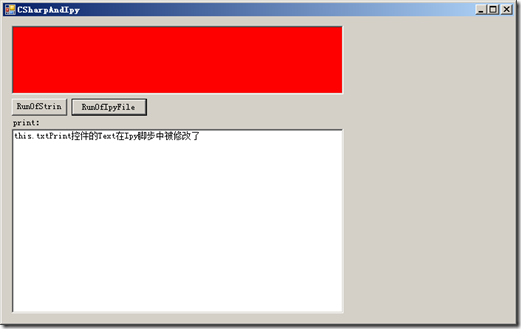
Ipy\_this.Width = 700

Ipy\_this.testMethod(255,0,0) Ipy\_this.txtPrint.Text = 'this.txtPrint控件的Text在Ipy脚本中被修改了'

[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/201105222247348849.png)当然，手动复制到debug目录也可以

运行结果：

[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/20110522224742531.png)[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/201105222247571945.png)

[](http://images.cnblogs.com/cnblogs_com/SongSharp/201105/201105222248237568.png)

编后感：读过很多来自前辈关于Ipy应用的文章，个人之见，用于不用应该与项目需求息息相关。而用于不用，Ipy就在那里，版本只增不减。以此简单小例，做一个Ipy的应用启发：是否我们可以在config加入Ipy脚本代码，来达到动态设置效果OR写上一个简单的计算器控件OR是否我们可以为WinForm小游戏写一个小游戏脚本...等等

源代码：[CSharpAndIpy.rar](http://files.cnblogs.com/SongSharp/CSharpAndIpy.rar)

**C# 4.0 动态调用 IronPython**

想一下这样的应用场景 现在我们使用NHibernate是使用xml来配置数据库链接 数据库方言 等等等等

但是通过调用IronPython来执行 \*.py 我们只需要在 py文件中写一个方法 然后在这个方法中直接写代码配置就可以了

而这个方法接受的参数是一个new Configuration()的实例 这样我们就可以做到在这个py中用python的代码来进行配置

是不是很棒呢？当然理论上是可行的 具体行不行呢？我们来做一个小例子

新建一个解决方案 InvokeIronPython

在这个解决方案下建两个项目

类库项目：Services

Console项目： ConsoleApp

在Services项目下建一个EchoServices.cs 代码如下

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Services {

     public class EchoServices {

         public string Echo(string message) {

             return message;

}

}

}

在ConsoleApp项目添加对IronPython的一些dll的引用（当然你要装了IronPython才会有这些dll。。。）

这些文件你可以在IronPython的安装目录下找到

IronPython.dll, IronPython.Modules.dll, Microsoft.Scripting.dll, Microsoft.Dynamic.dll

好了 现在我们在ConsoleApp目录下写一个文件

test.py 内容如下：

当然这个文件的属性是要设置copy if newer（你懂的-\_-）

代码如下

import clr #我们要调用C#的类库

clr.AddReference("Services") #添加对类库的引用

from Services import EchoServices #你懂的。。。

def zerg():

 es = EchoServices()

 return es.Echo("这是一个来自Services程序集的EchoServices类实例的Echo方法的执行返回结果！")

接下来是program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using IronPython.Hosting;

using Microsoft.Scripting.Hosting;

using Services;

namespace ConsoleApp

{

class Program

{

         static void Main(string[] args)

         {

             EchoServices es = new EchoServices();

             var ipy = Python.CreateRuntime();

             dynamic test = ipy.UseFile("test.py");

             Console.WriteLine(test.zerg());

         }

     }

 }

大功告成 你可以运行一下看看

这是在C# 4.0中动态调用IronPython脚本的一种方式，至于有没有用就要看你的应用场景了，呵呵。

原文：http://www.congci.com/item/csharpandironpython