0. 前言

ActiveX 控件以前也叫做 OLE 控件或 OCX 控件，它是一些软件组件或对象，可以将其插入到 WEB 网页或其它应用程序中。使用 ActiveX 插件，可以轻松方便的在 Web 页中插入多媒体效果、交互式对象以及复杂程序等等。

通常使用 C++或 VB 开发 ActiveX 控件，本文探讨一下在 Visual Studio 2005 环境中使用 C#开发 ActiveX 控件的技术实现。

1. 问题场景

在 C/S 架构的系统中，客户端要实现某些业务功能，可以通过安装相关的应用程序集来方便的实现。同样的需求，在 B/S 架构的系统里实现起来却比较困难。因为所有的程序都放在服务器端，客户端只是采用浏览器，通过 HTTP 协议来访问服务器端。比较成熟的解决办法是开发 ActiveX 控件安装到客户端，这样客户端的浏览器就可以访问本地的 ActiveX 控件来执行相关的本地操作。本文将要谈论的，就是使用 C#开发一个 ActiveX 控件实现读取并显示客户端的系统时间。

2. 开发环境

- Windows XP
- Visual Studio 2005
- .NET Framework 2.0（C#）

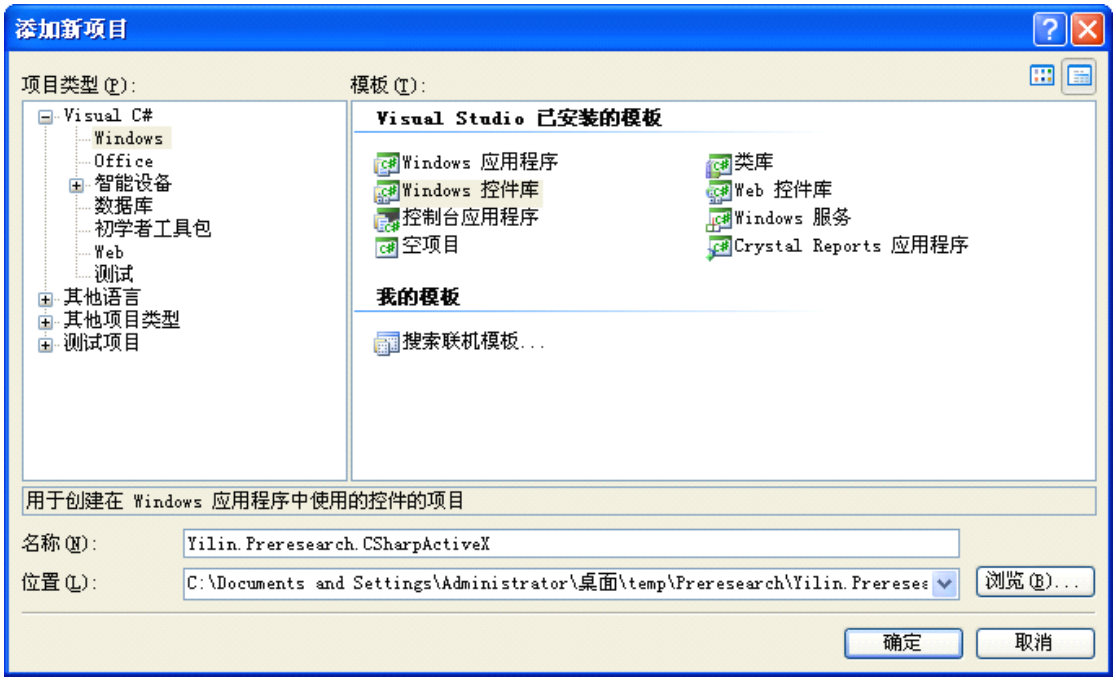
3. 实现过程

3.1.ActiveX 控件开发

在 Visual Studio 2005 开发环境中，可以使用 Windows 控件库项目实现 ActiveX 控件的开发，但是需要对项目做一些必要的设置。下面就来看看如何使用

Windows 控件库项目开发一个 ActiveX 控件。

首先创建一个应用程序解决方案，并添加一个 Windows 控件库项目：



更改“项目属性-应用程序-程序集信息”设置，勾选“使程序集 COM 可见”：

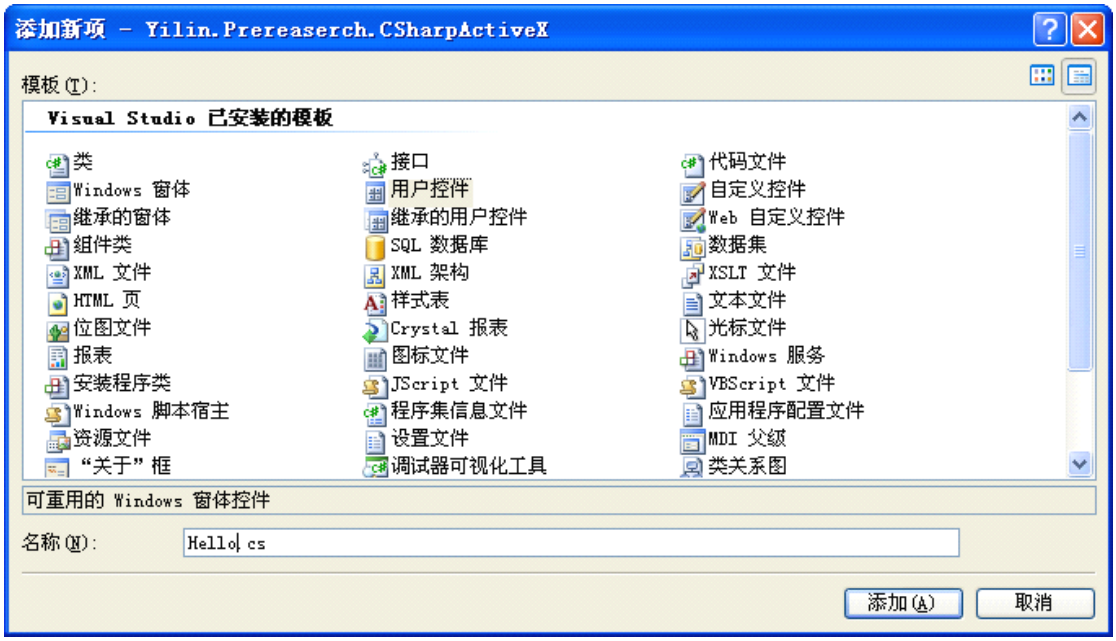
更改“项目属性-生成”设置，勾选“为 COM Interop 注册”（注意，此处如果实在 debug 状态下修改的，那在调到 release 状态下还需要再设置一次）：

修改 AssemblyInfo.cs 文件，添加[assembly: AllowPartiallyTrustedCallers()]项（需要引用 System.Security 名称空间）：

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Security;

[assembly: AssemblyTitle("Yilin.Preresearch.CSharpActiveX")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("10BAR")]
[assembly: AssemblyProduct("Yilin.Preresearch.CSharpActiveX")]
[assembly: AssemblyCopyright("Copyright © 10BAR 2009")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: AllowPartiallyTrustedCallers()]
[assembly: ComVisible(true)]
[assembly: Guid("114d1f0c-43b8-40ac-ae7c-5adccc19aef3")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

添加一个 Windows 用户控件：



按照开发 Windows 用户控件一样的思路完成该控件的开发，本例中主要实现了两个业务功能，一个是提供一个公共方法，用于读取 USBKey 中保存的签名证书，保存到本地 C 盘根目录下，并返回操作信息；另一个业务功能提供 UI 界面，包括一个 Button 控件和一个 Label 控件，Button 控件的 Click 事件调用前面提供的那个方法，并将返回信息显示到 Label 控件上。这样做可以达到两个目的，其一，ActiveX 控件提供公共方法供 B/S 程序直接调用，从后实现业务功能；其二，ActiveX 控件可以提供 B/S 程序 UI 界面，通过响应 B/S 程序中对 UI 的操作事件实现业务功能。

完成控件开发后，为了使该用户控件作为一个 ActiveX 控件进行使用，还需要做以下修改：

首先，为控件类添加 GUID，这个编号将用于 B/S 系统的客户端调用时使用（可以使用 工具-创建 GUID 菜单创建一个 GUID）：

```
Guid("4A44CF4E-F859-4328-AA22-3E9D7AFFF1AB")]
    public partial class Hello : UserControl
    {...
```

其次，为了让 ActiveX 控件获得客户端的信任，控件类还需要实现一个名为“IObjectSafety”的接口。先创建该接口（注意，不能修改该接口的 GUID 值）：

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
using System.Runtime.InteropServices;

namespace Preresearch.CSharpActiveX
{
    [ComImport, GuidAttribute("CB5BDC81-93C1-11CF-8F20-00805F2CD064")]
    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
    public interface IObjectSafety
    {
        [PreserveSig]
        int GetInterfaceSafetyOptions(ref Guid riid, [MarshalAs(UnmanagedType.U
4)] ref int pdwSupportedOptions, [MarshalAs(UnmanagedType.U4)] ref int pdwEnabl
edOptions);

        [PreserveSig()]
        int SetInterfaceSafetyOptions(ref Guid riid, [MarshalAs(UnmanagedType.U
4)] int dwOptionSetMask, [MarshalAs(UnmanagedType.U4)] int dwEnabledOptions);
    }
}
```

然后在控件类中继承并实现该接口：

```
#region IObjectSafety 成员

private const string _IID_IDispatch = "{00020400-0000-0000-C000-000000000046}";
private const string _IID_IDispatchEx = "{a6ef9860-c720-11d0-9337-00a0c90dcaa9}
";
private const string _IID_IPersistStorage = "{0000010A-0000-0000-C000-000000000
046}";
private const string _IID_IPersistStream = "{00000109-0000-0000-C000-0000000000
46}";
private const string _IID_IPersistPropertyBag = "{37D84F60-42CB-11CE-8135-00AA0
04BB851}";

private const int INTERFACESAFE_FOR_UNTRUSTED_CALLER = 0x00000001;
private const int INTERFACESAFE_FOR_UNTRUSTED_DATA = 0x00000002;
private const int S_OK = 0;
private const int E_FAIL = unchecked((int)0x80004005);
private const int E_NOINTERFACE = unchecked((int)0x80004002);

private bool _fSafeForScripting = true;
private bool _fSafeForInitializing = true;

public int GetInterfaceSafetyOptions(ref Guid riid, ref int pdwSupportedOptions,
 ref int pdwEnabledOptions)
{
```

```
    int Rslt = E_FAIL;

    string strGUID = riid.ToString("B");
    pdwSupportedOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER | INTERFACESAFE_FO
R_UNTRUSTED_DATA;
    switch (strGUID)
    {
        case _IID_IDispatch:
        case _IID_IDispatchEx:
            Rslt = S_OK;
            pdwEnabledOptions = 0;
            if (_fSafeForScripting == true)
                pdwEnabledOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER;
            break;
        case _IID_IPersistStorage:
        case _IID_IPersistStream:
        case _IID_IPersistPropertyBag:
            Rslt = S_OK;
            pdwEnabledOptions = 0;
            if (_fSafeForInitializing == true)
                pdwEnabledOptions = INTERFACESAFE_FOR_UNTRUSTED_DATA;
            break;
        default:
            Rslt = E_NOINTERFACE;
            break;
    }

    return Rslt;
}

public int SetInterfaceSafetyOptions(ref Guid riid, int dwOptionSetMask, int dw
EnabledOptions)
{
    int Rslt = E_FAIL;
    string strGUID = riid.ToString("B");
    switch (strGUID)
    {
        case _IID_IDispatch:
        case _IID_IDispatchEx:
            if (((dwEnabledOptions & dwOptionSetMask) == INTERFACESAFE_FOR_UNTR
USTED_CALLER) && (_fSafeForScripting == true))
                Rslt = S_OK;
            break;
        case _IID_IPersistStorage:
```

```
        case _IID_IPersistStream:
        case _IID_IPersistPropertyBag:
            if (((dwEnabledOptions & dwOptionSetMask) == INTERFACESAFE_FOR_UNTR
USTED_DATA) && (_fSafeForInitializing == true))
                Rslt = S_OK;
            break;
        default:
            Rslt = E_NOINTERFACE;
            break;
    }

    return Rslt;
}

#endregion
```

这样，一个 ActiveX 控件就开发完成了。

3.2.ActiveX 控件部署

ActiveX 控件可以使用 Visual Studio 2005 的安装项目进行部署。这与普通的 Windows Form 应用程序的部署几乎一样，只有一个地方需要注意，将前面创建的用户控件项目作为主输出项目，并设置其 Register 属性为 vsdrpCOM，如下图所示：

3.3.测试

建立一个 Web 应用程序项目，在测试页面的 HTML 代码中添加对 ActiveX 控件的引用，并且可以通过 Javascript 调用控件的公共成员（注意这里 clsid 后面的值即为前面为用户控件类设置的 GUID）：

```
<object id="csharpActiveX" classid="clsid:E5E0446C-8680-4444-9FC2-F837BC617ED9"
></object>
<input type="button" onclick="alert(csharpActiveX.SayHello());" value="显示当前
时间" />
```

将该 Web 应用程序项目发布到 IIS。另外找一台电脑作为客户端测试环境，确保它与服务器端网络连通，安装.NET Framework 2.0 和该 ActiveX 控件。安装完成后，就可以用浏览器访问服务器，进行测试了（你也可以在开发环境的系统中安装该 ActiveX 控件，并直接在 VS 2005 中运行 WebApp 项目查看结果）：

4. 总结

综上所述，在 Visual Studio 2005 环境中使用 C#开发 ActiveX 控件，技术实现上没有什么难度，唯一的问题就是客户端需要安装.NET Framework。鉴于 ActiveX 控件一般都是实现一些简单单一的功能，.NET Framework 2.0 已经完全可以应付，所以建议在.NET Framework 2.0 下开发。因为相对于.NET Framework 3.5 两百多兆的安装包，.NET Framework 2.0 安装包只有 20 多兆，用户相对容易接受一些。

5. FAQ

5.1.出现如下错误怎么解决？

经在网上查阅，该问题是 Visual Studio 2005 的一个 Bug，并不是每次都发生。我的解决办法是从 Visual Studio 2008 的安装目录里拷贝 regcap.exe 覆盖 Visual Studio 2005 的对应文件，文件目录一般为“~\Microsoft Visual Studio 8\Common7\Tools\Deployment\regcap.exe”。压缩包中提供了该文件的 Visual Studio 2008 版本。