使用 C#开发 ActiveX 控件（新）

摘要：ActiveX 是广泛应用于 IE 浏览器的一种 COM 技术，而使用 C#语言开发 ActiveX 控件技术由于属于非主流，所以存在不少技术难点。本文将从 ActiveX 控件的开发、发布、应用、部署和升级等方面，系统地介绍使用 C#开发 ActiveX 控件技术的方方面面，帮助你一步一步顺利掌握该技术。

前言

ActiveX 控件以前也叫做 OLE 控件，它是微软 IE 支持的一种软件组件或对象，可以将其插入到 Web 页面中，实现在浏览器端执行动态程序功能，以增强浏览器端的动态处理能力。通常 ActiveX 控件都是用 C++或 VB 语言开发，本文介绍另一种方式，在.NET Framework 平台上，使用 C#语言开发 ActiveX 控件。

虽然本文通篇都在讲如何使用 C#语言开发 ActiveX 控件，但我并不极力推荐使用这种技术，因为该技术存在明显的局限，即需要浏览器端安装.NET Framework（版本取决于开发 ActiveX 控件使用的.NET Framework 版本），该局限对于挑剔的互联网用户，几乎是不可接受的。所以，我建议以下几条均满足时，方可考虑使用该技术：

- 开发团队中没有人掌握使用 C++/VB 开发 ActiveX 控件技术；
- 该 ActiveX 控件不用于互联网；
- 用户对仅能使用 IE 浏览器访问表示可以接受；
- 用户对在浏览器端安装.NET Framework 组件表示可以接受。

另外，我建议如果不是因为控件的依赖库基于更高版本的.NET Framework，或需要更高版本的.NET Framework 提供的扩展功能（如需要 WCF 等），尽量在.NET Framework 2.0 上开发 ActiveX 控件，因为.NET Framework 2.0 只有 20M，相比 300M 的.NET Framework 3.5 和 40M 的.NET Framework 4.0 都要小很多，对客户端操作系统的要求也要低很多，并且随着 Windows 版本的不断升级换代，Windows Vista 以后的版本已经内置了.NET Framework 2.0。等到 Windows XP 系统寿终正寝之时，也将迎来该技术的春天。所以，别被我上面的建议夯退了，掌握该技术其实还是蛮有实用价值的，毕竟，C#高效的开发效率很有吸引力。

本文接下来将使用 C#语言开发一个 ActiveX 控件，实现对浏览器端的 MAC 地址遍历功能；另外，提供一个在 Web 静态页面中调用该控件的测试实例。本实例的开发环境为 Visual Studio 2010 旗舰版（SP1），目标框架为.NET Framework 2.0；浏览器端测试

环境为 Windows 7 旗舰版，IE8。

控件开发

使用 C#进行 ActiveX 控件开发过程其实很简单。首先，在解决方案中添加一个类库项目，目标框架使用.NET Framework 2.0，如图 1 所示：

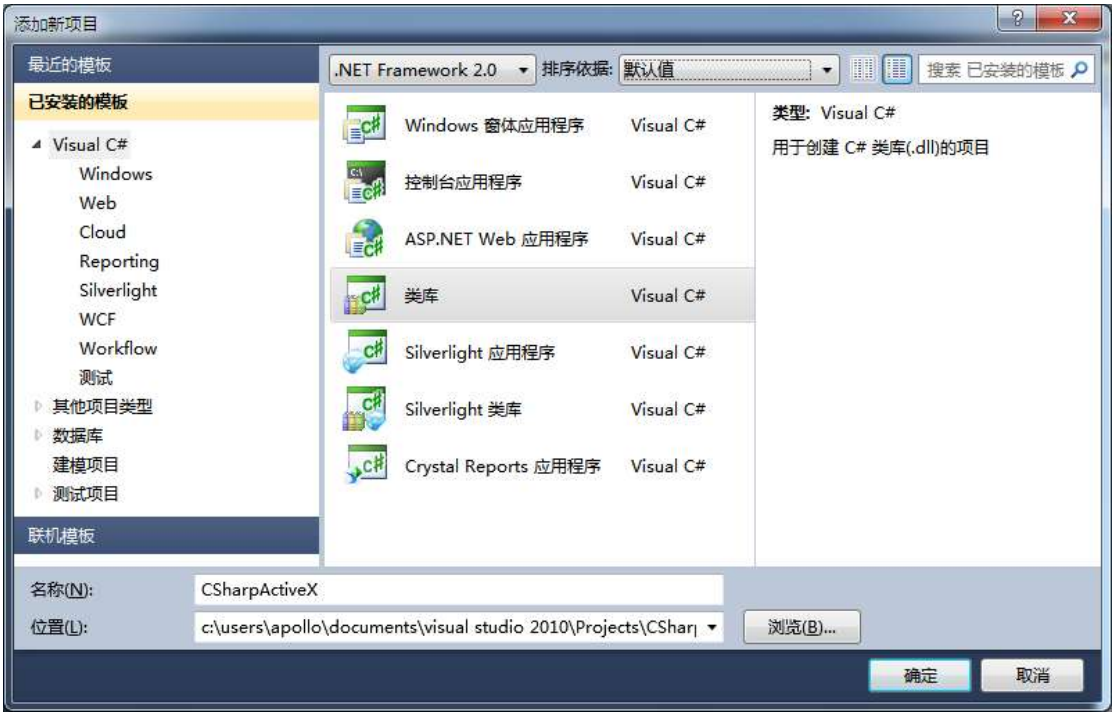

图 1 创建 ActiveX 控件类库

此处有一个关键操作，需要设置类库项目属性->程序集信息->使程序集 COM 可见，如图 2 所示：

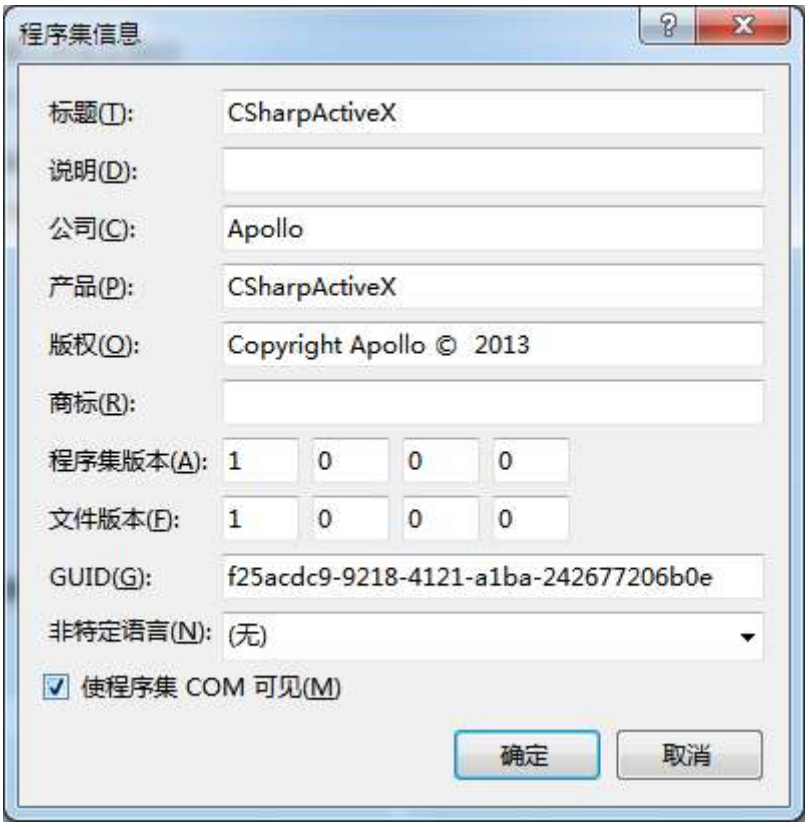

图 2 设置 ActiveX 控件类库程序集 COM 可见

ActiveX 类库的内容大致包括两部分，IObjectSafety 接口和实现该接口的控件类。考虑所有控件类都要实现 IObjectSafety 接口，可以将该接口的实现抽象为一个控件基类。

一、IObjectSafety 接口

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），接口内容如下：

```
[ComImport, Guid("CB5BDC81-93C1-11CF-8F20-00805F2CD064")]
[InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
public interface IObjectSafety
{
    [PreserveSig]
    int GetInterfaceSafetyOptions(ref Guid riid, [MarshalAs(UnmanagedType.U4)] ref int pdwSupportedOptions, [MarshalAs(UnmanagedType.U4)] ref int pdwEnabledOptions);

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

二、ActiveXControl 控件基类

```
public abstract class ActiveXControl : IObjectSafety
{
    #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-C000-000000000046}";
    private const string _IID_IPersistStream = "{00000109-0000-0000-C000-000000000046}";
    private const string _IID_IPersistPropertyBag = "{37D84F60-42CB-11CE-8135-00AA004BB
851}";

    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 pd
wEnabledOptions)
    {
        int Rslt = E_FAIL;

        string strGUID = riid.ToString("B");
        pdwSupportedOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER | INTERFA
CESAFE_FOR_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 dwEnabledOpt
ions)
    {
        int Rslt = E_FAIL;

        string strGUID = riid.ToString("B");
        switch (strGUID)
        {
            case _IID_IDispatch:
            case _IID_IDispatchEx:
                if (((dwEnabledOptions & dwOptionSetMask) == INTERFACESAFE_FOR_UN
TRUSTED_CALLER) &&
                        (_fSafeForScripting == true))
                    Rslt = S_OK;
                break;
            case _IID_IPersistStorage:
            case _IID_IPersistStream:
            case _IID_IPersistPropertyBag:
                if (((dwEnabledOptions & dwOptionSetMask) == INTERFACESAFE_FOR_UN
TRUSTED_DATA) &&
                        (_fSafeForInitializing == true))
                    Rslt = S_OK;
                break;
            default:
                Rslt = E_NOINTERFACE;
                break;
        }

        return Rslt;
    }

    #endregion
}
```

三、MacActiveX 控件类

```
[Guid("65D8E97F-D3E2-462A-B389-241D7C38C518")]
public class MacActiveX : ActiveXControl
{
    public string GetMacAddress()
    {
        var mc = new ManagementClass("Win32_NetworkAdapterConfiguration");
        var mos = mc.GetInstances();
        var sb = new StringBuilder();

        foreach (ManagementObject mo in mos)
        {
            var macAddress = mo["MacAddress"];

            if (macAddress != null)
                sb.AppendLine(macAddress.ToString());
        }

        return sb.ToString();
    }
}
```

注意，第一行指定的 Guid 值即为该 ActiveX 控件的唯一标识，请保证其唯一性。Guid 的生成有多种方法，你可以在系统目录的 Program Files 目录搜索一个名为 guidgen.exe 的工具，用该工具产生；也可以写一段测试代码，调用 Guid.NewGuid()方法产生；有的 Visual Studio 版本也提供了快捷方式，在“工具->生成 GUID”菜单下。另外，访问 MAC 需要添加对 System.Management 系统组件的引用。

到此，控件类库的开发工作就做完了，整个实现过程确实很简单。

发布

C#开发的 ActiveX 控件类库不像 OCX 那样可以直接通过 regsvr32.exe 注册（实际上，微软提供了替工具 regasm.exe，但由于这种方式要不能实现自动升级，所以本文就不介绍了），要使控件类库运行于浏览器端，可以采取两种方式，一种是将控件类库打包为 MSI 安装包，然后直接在浏览器端安装；另一种是将 MSI 再封装为一个 CAB 包，这个 CAB 包就是一个 ActiveX 控件了，可以将它随应用程序一并发布，浏览器端访问包含有该控件的页面时，就会自动提示安装了。接下来就后一种发布方式进行详细讲解。

一、安装项目

在解决方案中添加一个安装项目，如图 3 所示：

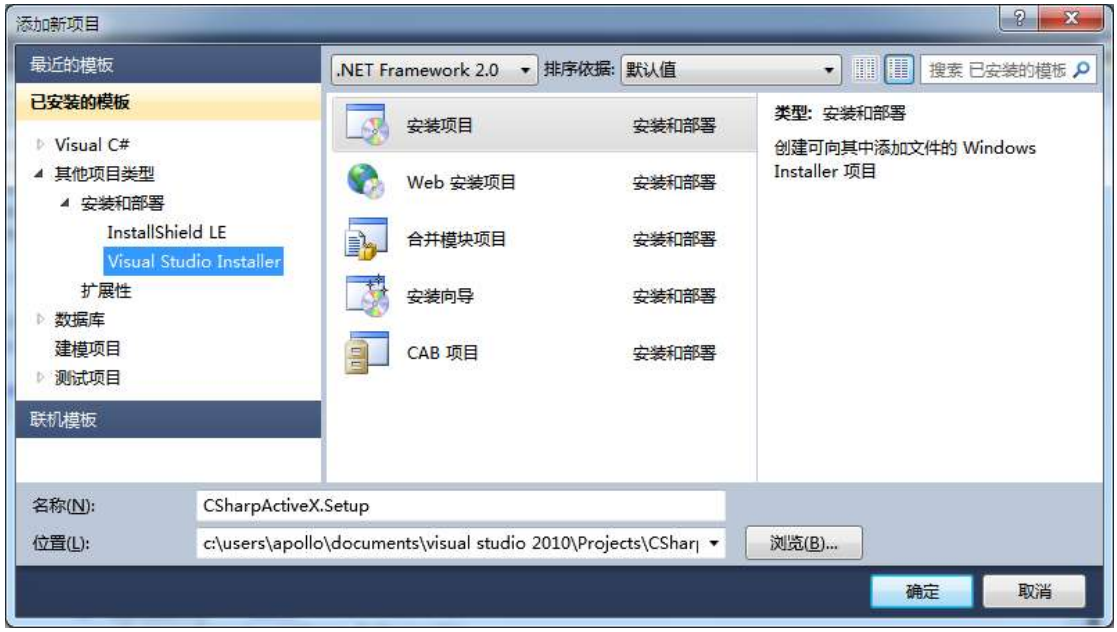

图 3 添加安装项目

右键点击新添加的安装项目，依次选择“添加->项目输出”菜单，打开添加项目输出组对话框，并选择 ActiveX 控件类库“CSharpActiveX”作为主输出，如图 4 所示：

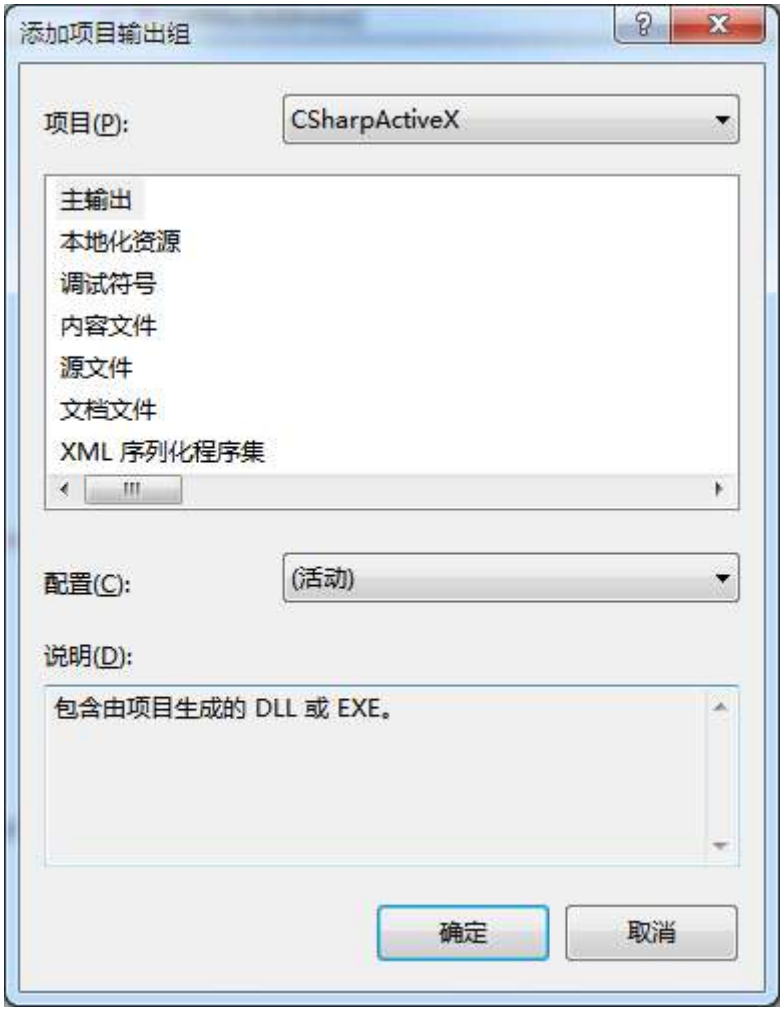

图 4 添加项目输出

双击安装项目检测到的依赖项“Microsoft .NET Framework”，打开安装项目的启动条件界面，选中“.NET Framework”项，如图 5 所示：

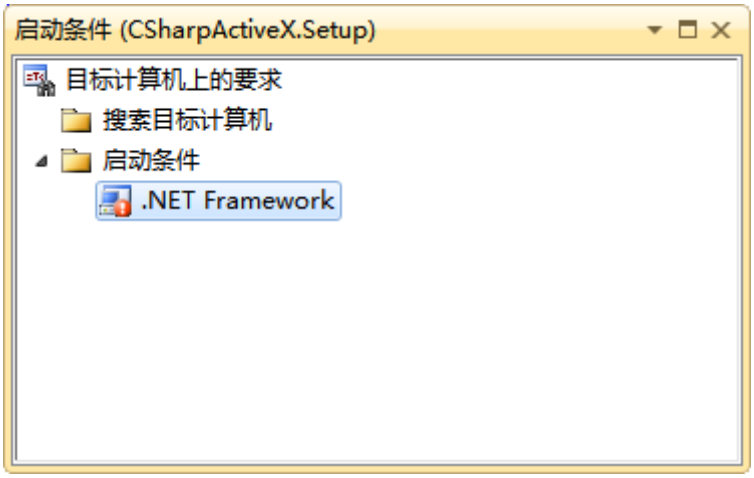

图 5 安装项目启动条件

按 F4 快捷键，打开属性窗口，设置.NET Framework 项的 Version 为“.NET Framework 2.0”，如图 6 所示：

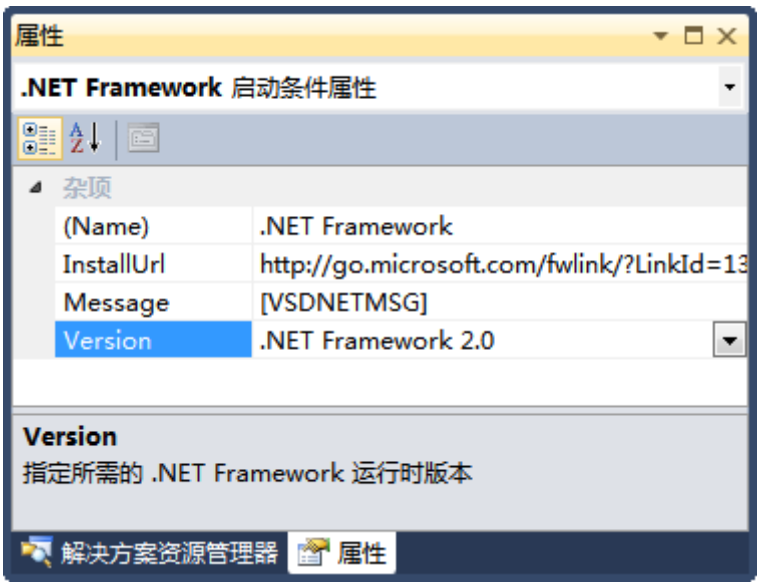

图 6 设置安装项目的依赖框架

下面这步很关键，选中“主输出来自 CSharpActiveX(活动)”项，如图 7 所示：

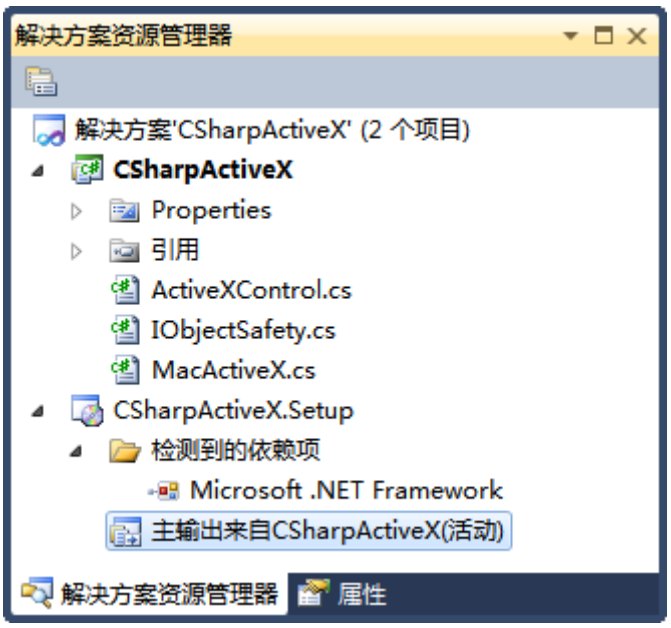

图 7 主输出内容项

设置主输出项内容的 Register 属性值为 vsdrpCOM，如图 8 所示：

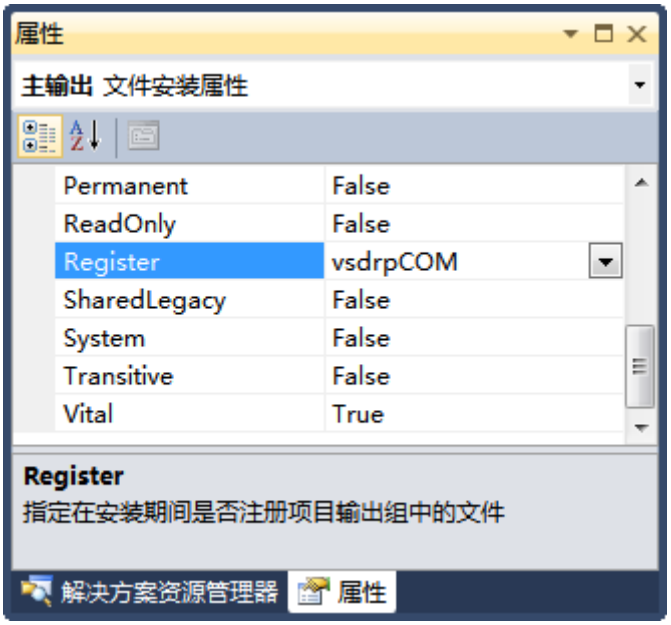

图 8 设置主输出项属性

二、制作 CAB 包

Visual Studio 2010 提供了 CAB 项目模板，但非常遗憾，无论我怎么设置，其生成的 CAB 安装包都不能在终端成功安装，最终只能放弃，转而选择了 makecab.exe 工具。源码提供了该打包工具，位于 CAB 目录下，共包含 makecab.exe、cab.ddf、installer.inf 和 makecab.bat 四个文件，其中 cab.ddf 和 installer.inf 文件需要简单说明下。

cab.ddf 文件定义了 CAB 文件的打包行为，内容包括打包参数，打包内容项以及输出

文件等。需要指出的是，使用 C#开发的 ActiveX 控件 CAB 包中需要包含 MSI 文件和 installer.inf 安装文件两部分。cab.ddf 文件内容如下：

```
.OPTION    EXPLICIT
.Set  Cabinet=on
.Set  Compress=on
.Set  MaxDiskSize=CDROM
.Set  ReservePerCabinetSize=6144
.Set  DiskDirectoryTemplate="."
.Set  CompressionType=MSZIP
.Set  CompressionLevel=7
.Set  CompressionMemory=21
.Set  CabinetNameTemplate="CSharpActiveX.CAB"
"installer.inf"
"CSharpActiveX.msi"
```

installer.inf 文件定义了 CAB 文件的安装行为，作为控件的一部分打入 CAB 包中，其内容如下：

```
[Setup  Hooks]
hook1=hook1

[hook1]
run=msiexec  /i  %EXTRACT_DIR%\CSharpActiveX.msi  /qn

[Version]
Signature=  "$CHICAGO$"
AdvancedInf=2.0
```

makecab.bat 文件是调用 makecab.exe 进行打包的批处理文件，内容如下：

```
makecab.exe    /f    "cab.ddf"
```

当生成安装项目后，将 CSharpActiveX.msi 文件拷贝到 CAB 目录下，就可以双击 makecab.exe 文件进行打包了，执行完成后会输出 CSharpActiveX.CAB 文件，这就是所谓的 ActiveX 控件了。

三、签名

IE 采用了 AuthentiCode 代码签名技术，对浏览器端安装 ActiveX 控件行为进行了控制。上面生成的 ActiveX 控件如果想在浏览器端成功安装，需要对浏览器进行设置，具体操作参见部署章节。

让所有用户都对 IE 进行设置，显得不太友好，为此，我们可以考虑使用 AuthentiCode 技术对 ActiveX 控件进行签名。Visual Studio 2010 附带的 signtool.exe（以前版本的

VS 提供的是另一个工具 signcode.exe）代码签名工具可以完成该工作（注意，并非一定要用微软提供的工具进行签名，只要按照 AuthentiCode 技术标准，使用 PKCS#7 标准定义的数据结构生成待签名文件的数字签名，并加入到待签名文件的 PE 结构中即可）。但需要先准备一个 PKCS#12（证书及私钥）文件(.pfx)，注意，该证书的增强型密钥用法须包含代码签名这项，如图 9 所示：

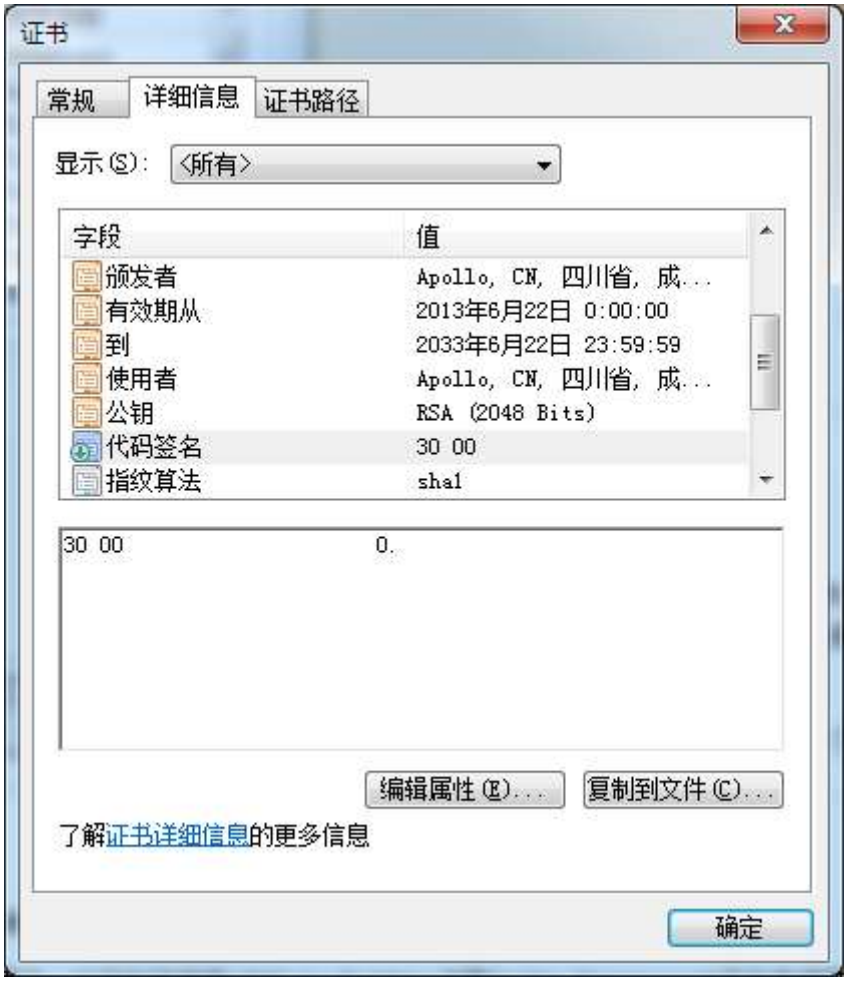

图 9 代码签名证书

本文源码提供了一份测试PKCS#12文件Apollo.pfx，PIN码为11111111。在Visual Studio 命令提示(2010)中，进入源码的 CAB 目录，输入如下命令即可对 ActiveX 控件进行签名操作了：

```
signtool sign –f Apollo.pfx –p 11111111 CSharpActiveX.CAB
```

图 10 对比了签名前后的 ActiveX 控件文件属性，可以看出，签名后的 ActiveX 控件属性中已经多了一项数字签名，表示该文件已经过签名。

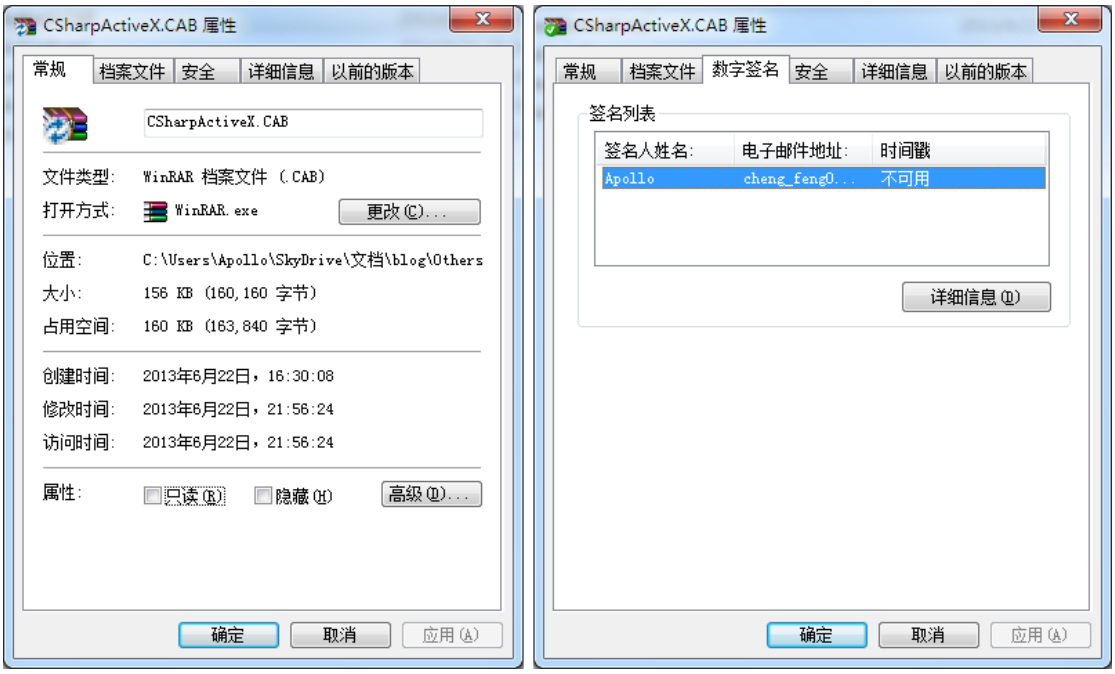

图 10 签名前后的 ActiveX 控件属性对比

出于方便考虑，本文源码的 CAB 目录下提供了一份 signtool.exe 工具的拷贝，这样就可以将签名命令加入 makecab.bat 文件中，修改后的 makecab.bat 我将其命名为 makecabsigned.bat，内容如下：

```
makecab.exe    /f    "cab.ddf"
signtool sign -f Apollo.pfx -p 11111111 CSharpActiveX.CAB
```

应用

ActiveX 控件用于 HTML 静态页面，执行于 IE 浏览器端。需要以<object>标签的形式引入页面文件，然后使用 Javascript 语言调用它。测试代码如下：

```
<html>
<head>
    <title>CSharpActiveX测试</title>
</head>
<body>
    <object id="cSharpActiveX" classid="clsid:65D8E97F-D3E2-462A-B389-241D7C38C518" co
debase="CSharpActiveX.CAB#version=1,0,0" style="display: none;"></object>
    <script type="text/javascript" language="javascript" defer="defer">
        var activeX = document.getElementById("cSharpActiveX");
        alert(activeX.GetMacAddress());
    </script>
</body>
</html>
```

注意，<object>标签的 classid 属性值即为 MacActiveX 类的 Guid 特性值。

部署

ActiveX 控件在 IE 浏览器端的部署会因 ActiveX 控件是否签名而有所区别。下面就以此分类进行说明。当然，首先需要将 test.htm 和 CSharpActiveX.CAB 文件部署到服务器上，假设部署后的访问地址为 http://192.168.1.1/test.htm。

一、部署未签名的 ActiveX 控件

未签名的 ActiveX 控件不受浏览器端信任，默认是不被允许安装的。需要先将站点添加为可信站点，具体步骤为：依次打开 IE“工具->Internet 选项”，在“安全”选项卡中，选中“可信站点”，如图 11 所示：

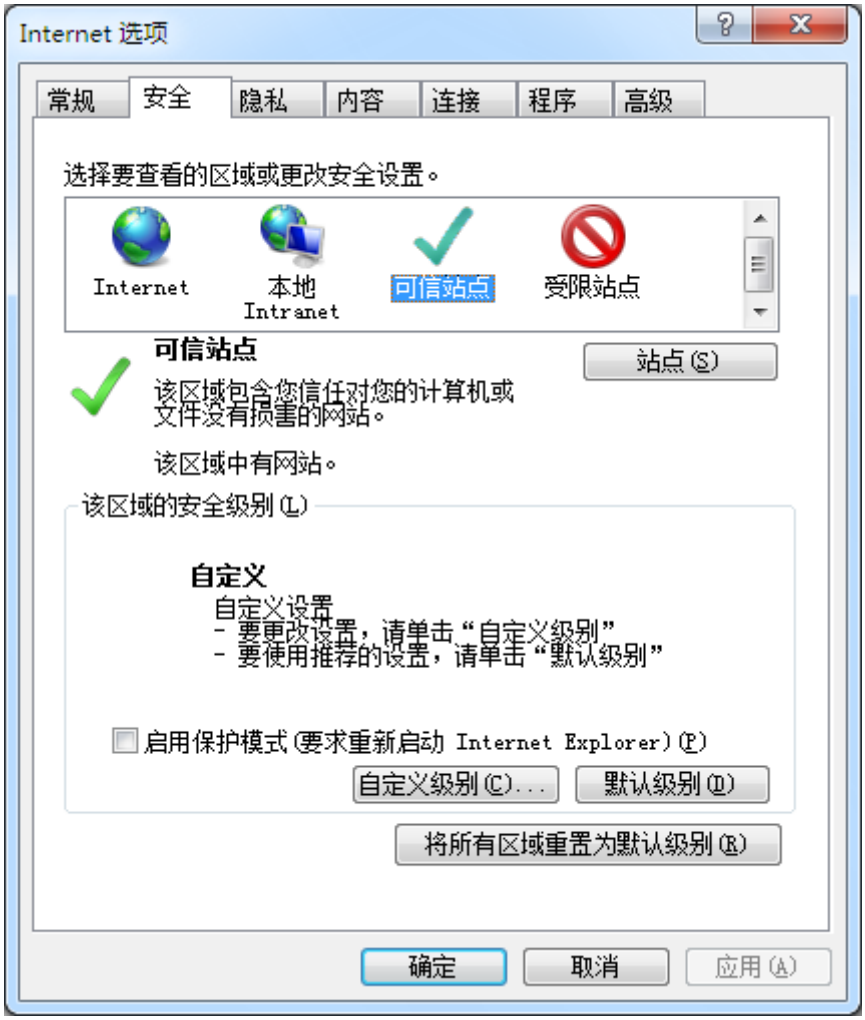

图 11 Internet 安全选项

点击“站点”按钮，打开可信站点管理对话框，将服务器站点添加到可信站点列表中，如图 12 所示：

图 12 可信站点对话框

回到“Internet 选项”对话框，点击“自定义级别”选项卡，打开可信站点的安全设置对话框，如图 13 所示：

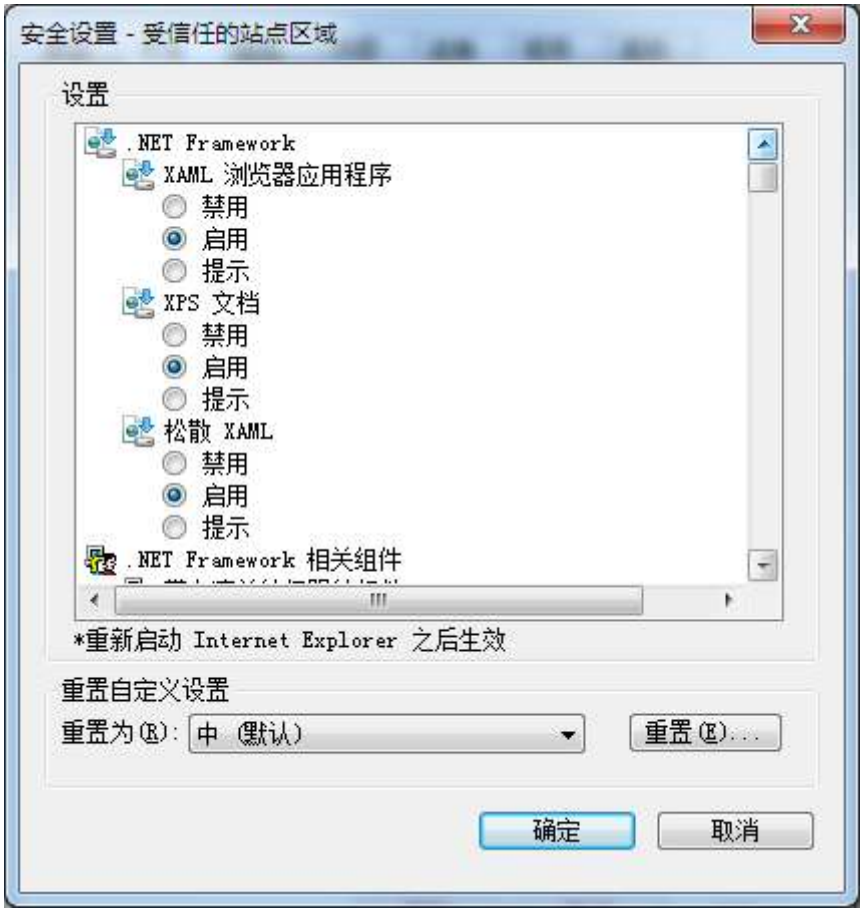

图 13 可信站点安全设置对话框

确认“对未标记为可安全执行脚本的 ActiveX 控件初始化并执行脚本”项设置为“启用”，“下载未签名的 ActiveX 控件”项设置为“提示”。

IE 设置完成后，访问 http://192.168.1.1/test.htm 测试页面（注意，Windows 7 需要“以管理员身份运行”IE 方可成功安装 ActiveX 控件），IE 便会提示加载 ActiveX 控件，如图 14 所示：

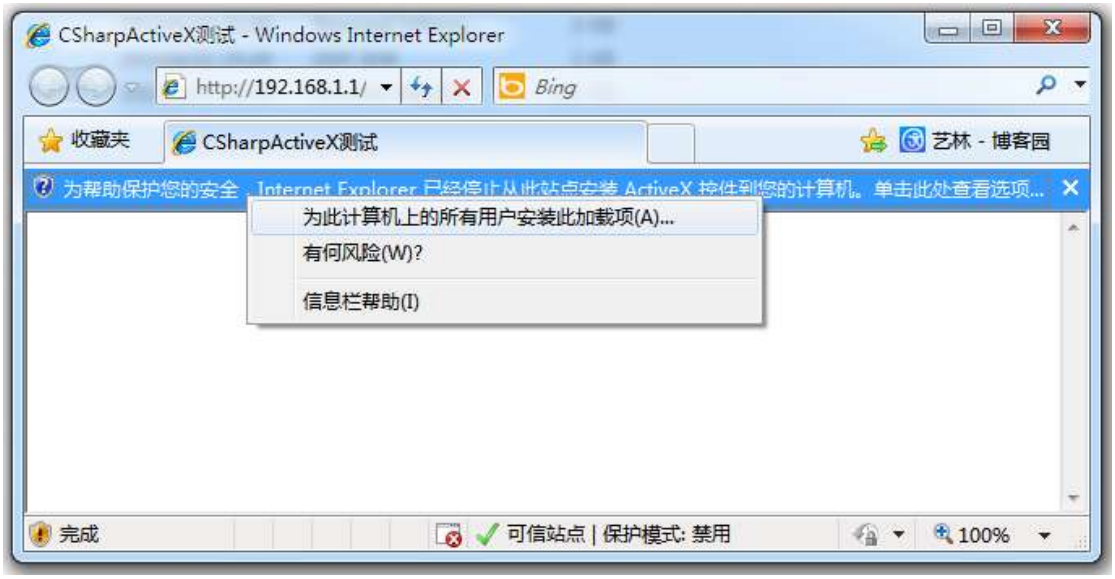

图 14 首次访问提示加载 ActiveX 控件

点击“为此计算机上的所有用户安装此加载项”，IE 将弹出安全警告，确认是否要安装该 ActiveX 控件，如图 15 所示：

图 15 ActiveX 控件安装安全警告

点击“安装”按钮，确认安装该 ActiveX 控件，待 IE 状态栏进度条完成，说明控件已安装完成，可以通过查看“卸载或更改程序”项来确认是否安装成功，如图 16 所示：

图 16 确认 ActiveX 控件成功安装

我们可以从 ActiveX 控件安装过程看出，浏览器端其实是以静默安装的方式完成对 CAB 包中的 MSI 安装文件的安装（有点拗口☺）。安装完成后，页面成功调用 ActiveX 控件，弹出接口调用结果（注意 Windows 7 需要重启 IE，且不能用“以管理员身份运行”方式启动，否则会再次提示安装 ActiveX 控件，但其实控件已经成功安装了，这个问题很奇怪），效果如图 17 所示：

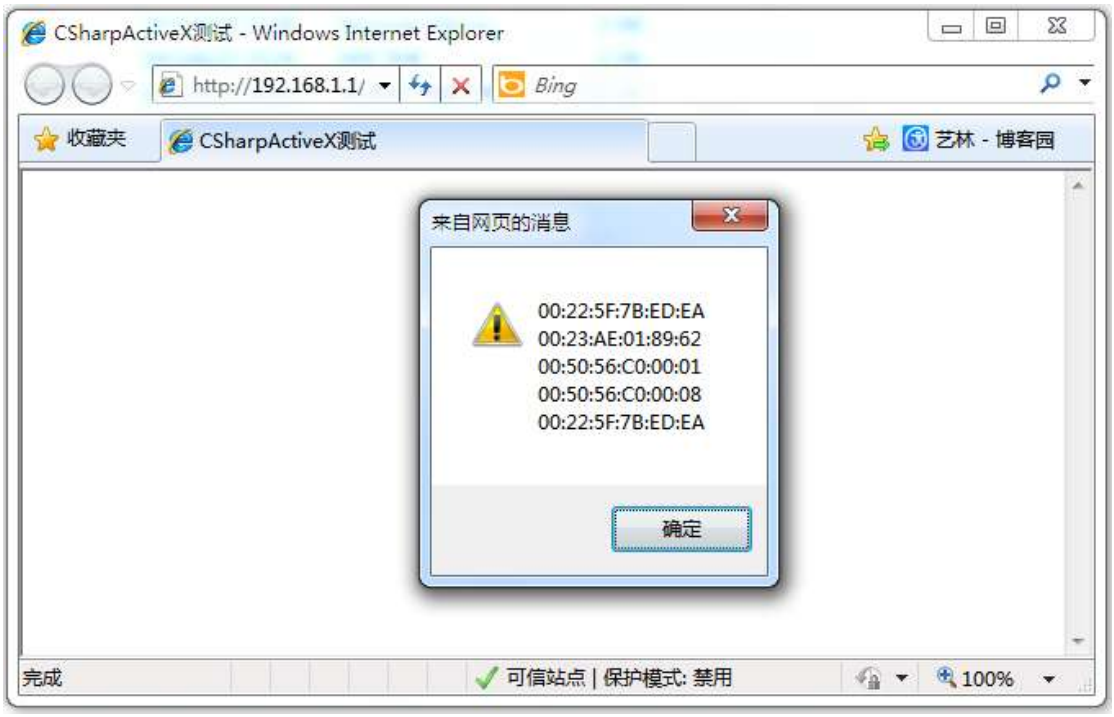

图 17 成功调用 ActiveX 控件接口

二、部署已签名的 ActiveX 控件

因为 IE 默认允许安装并运行收信任的已签名 ActiveX 控件，所以通过对 ActiveX 控件签名，可以有效简化浏览器端的配置工作。你仅需要安装签名所用的证书及其证书链文件（本文源码提供的签名文件所含证书是自签名证书，所以它的证书链就只是它自己）。打开源码 CAB 目录下的 Apollo.cer（与 Apollo.pfx 文件对应的数字证书文件）代码签名证书文件，如图 18 所示：

图 18 签名证书文件

点击“安装证书”按钮，将该证书安装到“受信任的根证书颁发机构”，如图 19 所示：

图 19 安装代码签名证书

打开 IE 的“工具->Internet 选项”对话框，选择“内容”选项卡，点击“证书”按钮，打开 IE 证书对话框，确认在“受信任的根证书颁发机构”选项卡中包含刚才导入的代码签名证书，如图 20 所示：

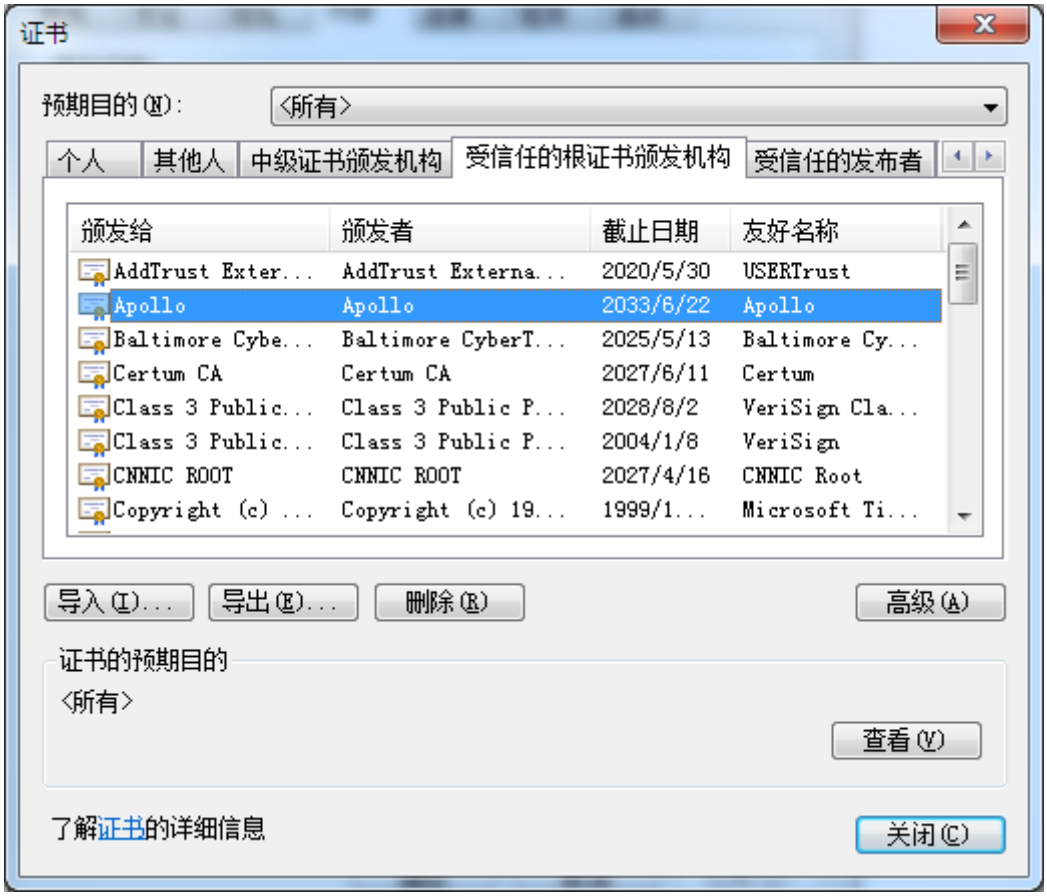

图 20 成功导入代码签名证书

此时，再访问测试页面 http://192.168.1.1/test.htm，IE 就会提示安装 ActiveX 控件了，而不再需要将站点添加到可信站点并设置 IE 选项了。

但是，如果用户不能接受初次安装需要导入代码签名证书及其证书链的方式，怎么办呢？从图 20 可以看到，Windows 其实默认内置了一些权威的 CA 机构证书，可以向这些机构申请一份代码签名证书及私钥文件来对 ActiveX 控件签名，这样就可以避免该问题了。但是，向权威的 CA 机构申请证书是需要付费的，所以需要权衡成本和易用性后，再做出选择。

升级

要使 C#编写的 ActiveX 控件支持自动升级，需要做四件事情，即升级 ActiveX 控件库版本、升级安装项目版本、设置安装项目注册表项版本和升级网页<object>版本。

一、升级 ActiveX 控件版本

打开 ActiveX 控件项目的“程序集信息”对话框，升级程序集版本和文件版本，如图 21 所示：

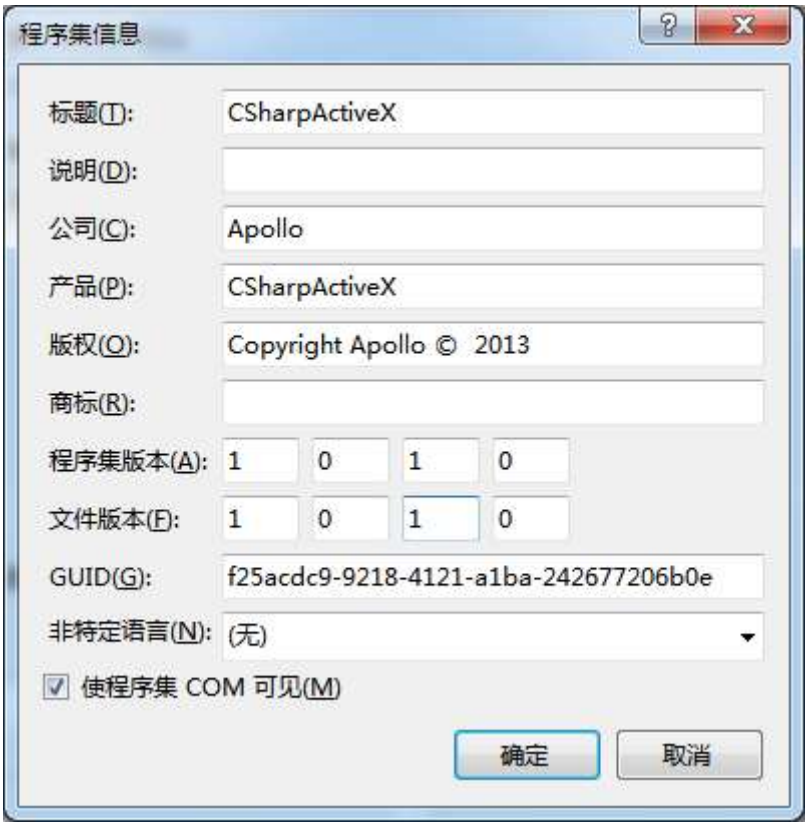

图 21 升级 ActiveX 控件版本

二、升级安装项目版本

选中安装项目，按 F4 快捷键打开安装项目的属性窗口，升级安装项目的版本，如图 22 所示：

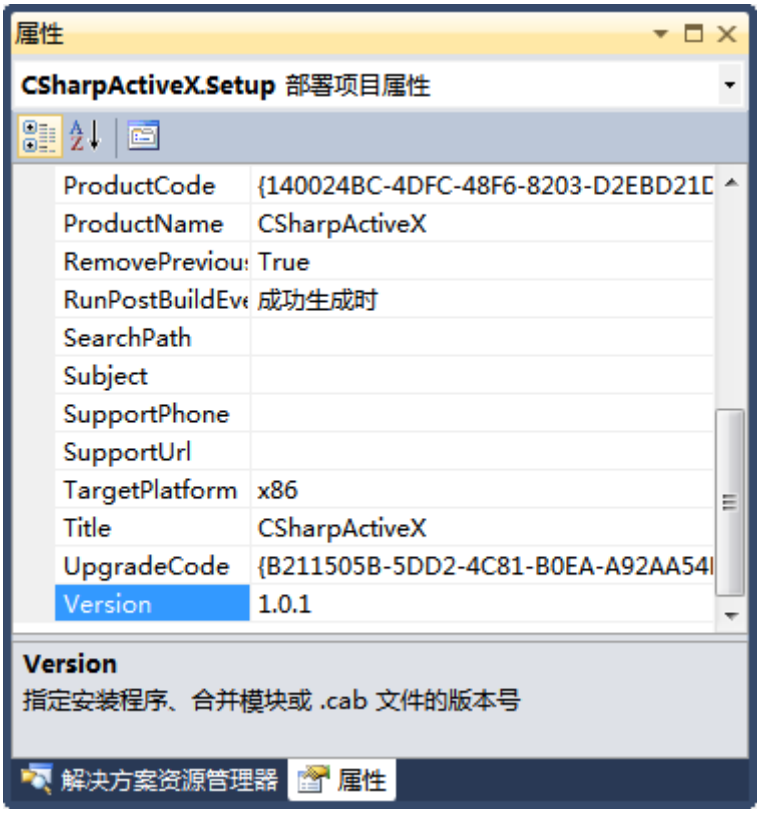

图 22 升级安装项目版本

注意，此处还有一项关键工作要做，就是设置 RemovePreviousVersions 属性值为 True，这样就会在升级时先自动卸载之前版本的控件。

三、设置安装项目注册表项版本

浏览器端检测 ActiveX 控件是否需要升级，是通过比对<object>标签的 codebase 属性值和本地 HKEY_CLASSES_ROOT/CLSID/{GUID}/InstalledVersion 键值是否相等来判断的。所以，如果要实现自动更新，需要手动添加该注册表项，并在每次升级控件时，相应更改该项键值。

右键点击安装项目，依次选择“视图->注册表”菜单，打开安装项目的注册表编辑界面，并在 HKEY_CLASSES_ROOT 节点下，建立 CLSID/{GUID}/InstalledVersion 注册表键路径，如图 23 所示：

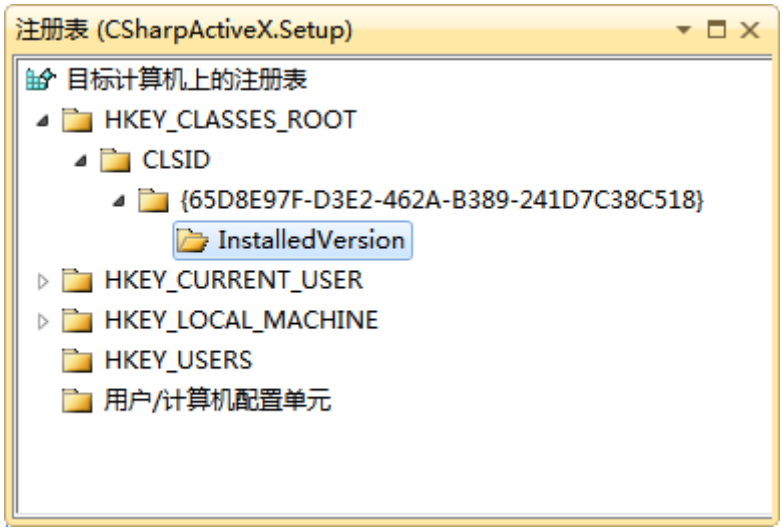

图 23 创建注册表键路径

右键点击 InstalledVersion 键节点，选择“新建->字符串值”菜单，新建一个名称为空（空名称会显示为“(默认值)”），值为当前控件版本号的键值，如图 24 所示：

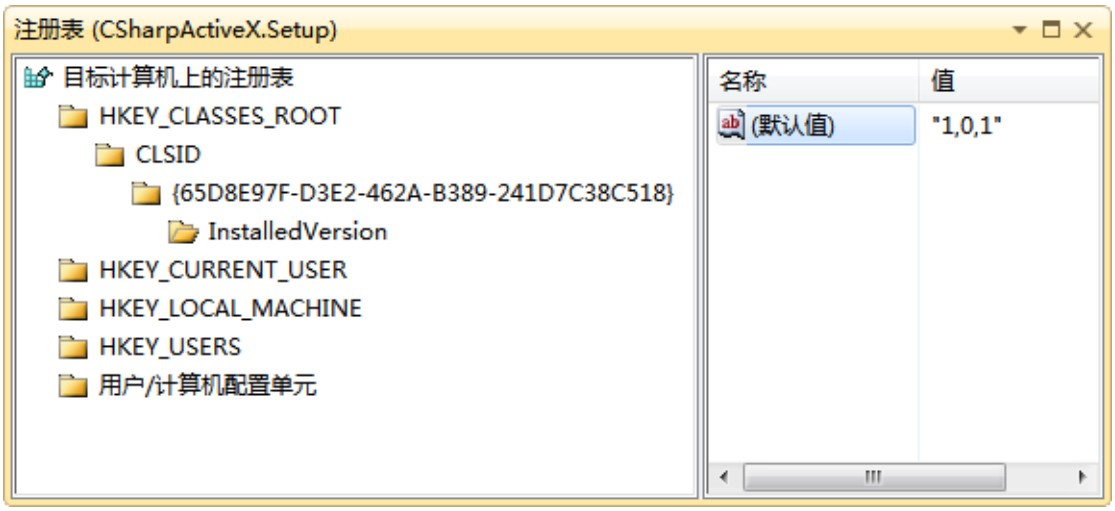

图 24 添加 InstalledVersion 默认键值

该步骤有几个地方需要特别说明。首先，{GUID}指的是 ActiveX 控件类的 GUID，对应本文 MacActiveX 类指定的 GUID，且该项需要包括左右花括号；其次，如果该安装项目用于发布多个 ActiveX 控件（类），需要创建多个{GUID}/InstalledVersion 路径；最后，InstalledVersion 的默认键值的主次版本号间是用“,”分隔，而不是“.”，后续升级时，需要同步升级该键值版本号。

四、升级网页<object>版本

最后，需要升级网页中的 ActiveX 对象引用版本号，如下用下划线标识部分：

```
<object id="csharpActiveX" classid="clsid:65D8E97F-D3E2-462A-B389-241D7C38C518" codeba
```

```
se="CSharpActiveX.CAB#version=1,0,1"  style="display:  none;"></object>
```

重新生成安装程序，打 CAB 包，将升级的页面及 ActiveX 控件（CAB 包）更新到服务器。此时，浏览器端重新访问时，就会提示/自动升级 ActiveX 控件了。

总结

本文是《使用 C#开发 ActiveX 控件》一文的升级版本，从 ActiveX 控件的开发、发布、应用、部署和升级整个生命周期，系统地介绍了使用 C#开发 ActiveX 控件技术的方方面面，对整个过程中可能遇到的一些技术难点进行了逐一讲解，并对其中涉及的一些知识进行了简单介绍。希望本文能够解答自上一篇文章发布以来众多网友提出的种种问题，帮助大家成功掌握这门技术。