  最近在把一个网站打包成安装程序，这方面的文章网上有很多，也看了不少，但因为开发环境的不同，遇到了一些问题，便写下这篇文章记下整个流程（有很多资源都来自互联网，由于条目颇多，所以无法说明其来处，敬请谅解）。

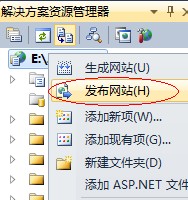
**一、开发环境**

        言归正传，先说明一下开发环境，win7 64位+visual studio 2010+IIS7。

**二、制作流程**

**1、发布网站**

        要制作网站自定义安装程序，首先，我们得有一个准备打包的网站。我这里是一个ASP.NET网站。接下来就要对这个网站进行预编译，右击“解决方案资源管理器”的网站，选择“发布网站”，即可将网站发布到本地机器上。预编译主要是将网站的源代码编译成相应的动态链接库。



**2、新建“安装项目”**

        接下来，在vs中打开“新建项目”->“其他项目类型”->“安装和部署”->“Visual Studio Installer”，选择“安装项目“，命名MyWebSite。“Web安装项目”也可进行网站安装程序的制作，但相对“安装项目”，其安装过程不好控制（虽然“安装项目”也有一些控制不到的地方，但目前已足够）。

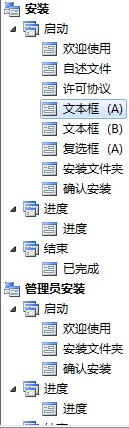
        在“安装项目”有如下按钮：文件系统编辑器、注册表编辑器、文件类型编辑器、用户界面编辑器、自定义操作编辑器、启动条件编辑器。

http://img.my.csdn.net/uploads/201304/02/1364908900_8919.jpg

**3、添加安装时显示的对话框**

        打开“用户界面编辑器”，分为安装与管理员安装两个部分。“安装”部分是我们要用到的，而“管理员安装”网上有人说是系统管理员将安装程序上传到某个网络位置时显示的对话框（不甚理解，以后用到再补充）。

    右键点击“启动”节点，选择“添加对话框”，依次添加“自述文件”、“许可协议”、“文本框（A）”、“复选框（A）”，如下图所示。



    这里先介绍文本框和复选框的设置，点击“文本框（A）”，在“属性”窗口中则可看到其相应设置（其他对话框类似），现列举出各选项含义，

    BannerBitmap:此文本框的横幅图片，即下图banner所指的那张图片，可以为bmp或jpg格式。后面会讲如何设置。

    BannerText:相当于此文本框的标题

    BodyText:Banner下对此文本框作用的说明

    Edit1Label:输入框上的标题

    Edit1Property:输入框的属性值，要获得输入框的内容，就得通过它

    Edit1Value:输入框默认显示的内容

    Edit1Visible:输入框可见性，false即隐藏



    复选框（A）设置与文本框（A）类似，只是其中CheckBox1Value值有点区别，如果勾选了某个复选框，则该复选框CheckBox1Value值为Checked（程序获得值为1），不选则为Unchecked（程序获得值为空）。

**4、新建安装类库**

    自定义安装的代码即在此类库中，它将完成根据用户输入的信息配置IIS站点等功能

    “文件”->“添加”->“新建项目”->“Visual C#”->“Windows”，选择“类库”，命名为“WebSetupClassLib”，删除自动生成的Class1.cs。

    右键点击类库名，“添加”->“新建项”->“Visual C#”项（常规），选择“安装程序类”，命名为“MyWebInstaller”。

    查看MyWebInstaller的代码，安装程序类其实是继承自Installer类（实现自定义安装的关键）的局部类：

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

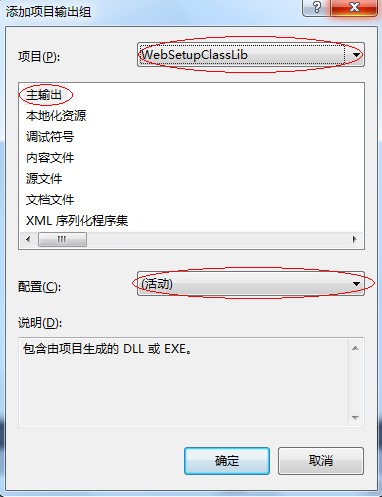
1. **using** System;
2. **using** System.Collections;
3. **using** System.Collections.Generic;
4. **using** System.ComponentModel;
5. **using** System.Configuration.Install;
6. **using** System.Linq;

9. **namespace** WebSetupClassLib
10. {
11. [RunInstaller(**true**)]    //安装程序集时调用自定义操作安装程序
12. **public** partial **class** MyWebInstaller : System.Configuration.Install.Installer
13. {
14. **public** MyWebInstaller()
15. {
16. InitializeComponent();
17. }
18. }
19. }

**5、添加自定义操作（相当于安装类库与程序安装等行为的连接）**

  先暂且不管安装程序类，右键单击安装类库，选择“生成”。

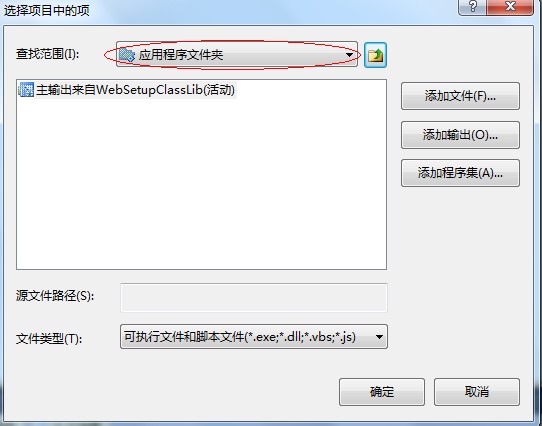
   接着右键单击安装项目“MyWebSite”，选择“添加”->“项目输出”，选择安装类库WebSetupClassLib（出现前提是执行上一步的“生成”），如下图。



   此时，安装项目“MyWebSite”下将多了两项，一个是“主输出来自WebSetupClassLib(活动)”，另一个是“检测到的依赖项”下多了一个“Microsoft .NET Framework”(因为安装类库WebSetupClassLib的目标框架是.NET Framework 4,安装项目需要提供对它的支持)。

   然后点击安装项目的“自定义操作编辑器”，里面有四个类似于文件夹的项，分别为安装、提交、回滚、卸载。提交指安装完后的动作，回滚指安装不成功程序回滚执行的动作。

   右击“安装”，选择“添加自定义操作”，在弹出的“选择项目中的项”对话框中选择应用程序文件夹，选中“主输出来自WebSetupClassLib(活动)”，确定。



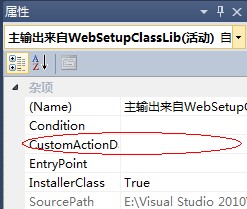
   这样在安装时，程序就会自动执行安装类库中的自定义安装操作（如：Install函数）。

   同样的方式，在“提交”和“卸载”也加入“主输出来自WebSetupClassLib(活动)”，回滚暂时没用到，不用加。

**6、获取用户输入信息**

 在自定义安装操作前，先得知道在安装程序类“MyWebInstaller”中如何获取网站安装过程中用户输入的信息（如web服务器地址，端口）。

   切换到“自定义操作”->“安装”项目下，点击“主输出来自WebSetupClassLib(活动)”，在“属性”窗口有个“CustomActionData”选项。



   在其中填入:/port=[PORT] /iis=[IISSERVER] /targetdir="[TARGETDIR]\"

   []内是属性名，如在文本框（A）中，有

    Edit1Label:web服务器地址

    Edit1Property:IISSERVER

    Edit1Value:192.168.1.100

   “web服务器地址”的属性名就是Edit1Property的值IISSERVER

   “web服务器地址”的值就自动赋给了iis，[TARGETDIR]是安装路径，程序定义，不是自定义，需如此书写"[TARGETDIR]\"，每项间必须有空格，根据需要自行加入。

   在MyWebInstaller中获得值：string webIP = this.Context.Parameters["iis"].ToString();    //web服务器IP

   即获得了用户输入的“web服务器地址”，如192.168.1.12。

**注意：在获取用户所选安装路径时，string physicaldir = this.Context.Parameters["targetdir"].ToString();  //网站物理路径**

**假设路径是“D:\test\”，则physicaldir的值是“D:\\test\\\\”（\\中一个\是转义字符,另一个才是表示路径的\）,所以，实际表         示路径是“D:\test\\”，显然不对（test后多了一个\）。  
        还要进行如下操作：physicaldir = physicaldir.Substring(0, physicaldir.Length - 1);**

**这里中了招，幸好有调试，才得以解决（但为何获得的路径会多个"\"?）。**

7、重写自定义安装相关方法

    要安装时执行自定义动作，就得重写Installer类中的相关函数（发现函数调用顺序OnBeforeInstall->OnAfterInstall->Install，猜测可能原因是前两个是事件，后一个Install是函数，因此前两个和后一个没有必然联系）。

    有两种方式：

    1、public override void Install(IDictionary stateSaver)

    2、在构造函数中加入this.BeforeInstall += new InstallEventHandler(MyWebInstaller\_BeforeInstall);

       然后定义函数private void MyWebInstaller\_Install(object sender, InstallEventArgs e){//自定义安装代码}

       但是此法不针对Install，如this.Install += new InstallEventHandler(MyWebInstaller\_Install);是不对的，因为Install只是一个方法而已，而非事件。

**A、重写安装（Installer）方法**

安装方法中包括以下内容：获取用户输入信息、新建IIS应用程序池、新建IIS站点、配置IIS站点、设置网站目录权限、修改web配置文件、网站ID写入注册表、创建桌面快捷方式、创建应用程序菜单项。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 程序安装
2. **public** **override** **void** Install(IDictionary stateSaver)
3. {
4. **base**.Install(stateSaver);
5. //System.Diagnostics.Debugger.Launch();
6. physicaldir = **this**.Context.Parameters["targetdir"].ToString();  //网站物理路径
7. physicaldir = physicaldir.Substring(0, physicaldir.Length - 1);
8. virtualdir = **this**.Context.Parameters["virtualdir"].ToString();  //网站虚拟路径
9. dbname = **this**.Context.Parameters["dbname"].ToString();  //数据库名
10. dbserver = **this**.Context.Parameters["dbserver"].ToString();  //数据库服务器名称
11. user = **this**.Context.Parameters["user"].ToString();  //数据库连接用户
12. pwd = **this**.Context.Parameters["pwd"].ToString();    //数据库连接密码
13. iis = **this**.Context.Parameters["iis"].ToString();    //web服务器IP
14. port = **this**.Context.Parameters["port"].ToString();  //站点端口
15. websitename = **this**.Context.Parameters["websitename"].ToString();  //即站点名
16. isrun = **this**.Context.Parameters["run"];   //安装完成后是否运行
17. NewWebSiteInfo siteinfo = **new** NewWebSiteInfo(iis, port, "", websitename, @physicaldir);//@意为忽略转义字符含义
18. CreateNewWebSite(siteinfo);
19. SetFileRole();
20. WriteWebConfig();
21. WriteToReg("WebSiteID");
23. **if** (**this**.Context.Parameters["deskcut"] == "1")    //创建桌面快捷方式
24. {
25. CreateDeskTopCut();
26. }
27. **if** (**this**.Context.Parameters["pmenu"] == "1")    //创建应用程序菜单项
28. {
29. CreateProCut();
30. }
32. }
33. #endregion

     B、IIS站点信息管理类

        在安装类库WebSetupClassLib中，添加一个类，类名为NewWebSiteInfo，该类主要是管理站点相关信息，包括站点IP、端口号、站点名等等。代码如下：

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. **namespace** SetupClassLibrary
2. {
3. **public** **class** NewWebSiteInfo
4. {
5. **private** **string** hostIP; // 主机IP
6. **private** **string** portNum; // 网站端口号
7. **private** **string** descOfWebSite; // 网站表示。一般为网站的网站名。如"www.myweb.com.cn"
8. **private** **string** nameOfWebSite;// 网站名称。如"我的网站"，此处即为在IIS管理器中的网站名称
9. **private** **string** webPath; // 网站的主目录。例如@"e:\\ mp"
11. **public** NewWebSiteInfo(**string** hostIP, **string** portNum, **string** descOfWebSite, **string** nameOfWebSite, **string** webPath)
12. {
13. **this**.hostIP = hostIP;
14. **this**.portNum = portNum;
15. **this**.descOfWebSite = descOfWebSite;
16. **this**.nameOfWebSite = nameOfWebSite;
17. **this**.webPath = webPath;
18. }
20. **public** **string** BindString
21. {
22. **get**
23. {
24. **return** String.Format("{0}:{1}:{2}", hostIP, portNum, descOfWebSite); //网站标识（IP,端口，主机头值）
25. }
26. }
28. **public** **string** PortNum
29. {
30. **get**
31. {
32. **return** portNum;
33. }
34. }
36. **public** **string** NameOfWebSite
37. {
38. **get**
39. {
40. **return** nameOfWebSite;
41. }
42. }
44. **public** **string** WebPath
45. {
46. **get**
47. {
48. **return** webPath;
49. }
50. }
51. }
52. }

   C、IIS站点管理方法

**获得DirectoryEntry对象实例**。

    DirectoryEntry使用ADSI（Active Directory Service Interfaces，活动目录服务接口）技术定位和管理网络上的资源，这里将使用它和微软为管理IIS7提供的API——Microsoft.Web.Administration共同管理IIS站点。

    也可使用WMI管理IIS站点，使用前需打开“IIS管理脚本和工具”。“IIS管理服务”不确定是否需要打开。

    如果系统使用的是IIS7，请确认是否打开“IIS 元数据库和IIS 6配置兼容性”功能。具体参见<http://blog.csdn.net/ts1030746080/article/details/8741399>

    DirectoryEntry需引用System.DirectoryServices组件，并在代码前加入using System.DirectoryServices;

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. **private** **string** entPath = String.Format("IIS://{0}/w3svc", "localhost");
2. **public** DirectoryEntry GetDirectoryEntry(**string** entPath)
3. {
4. DirectoryEntry ent = **new** DirectoryEntry(entPath);
5. **return** ent;
6. }

     判断站点是否存在。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 判断站点是否存在
2. /// <summary>
3. /// 返回true即站点不存在
4. /// </summary>
5. /// <param name="BindString"></param>
6. **private** **bool** EnsureNewSiteEnavaible(**string** BindString)
7. {
8. **bool** Isavaible = **false**;
9. DirectoryEntry rootEntry = GetDirectoryEntry(entPath);
11. //遍历所有站点
12. **foreach** (DirectoryEntry item **in** rootEntry.Children)
13. {
14. PropertyValueCollection serverBindings = item.Properties["ServerBindings"];
15. **if** (!serverBindings.Contains(BindString))
16. {
17. Isavaible = **true**;
18. }
19. **else**
20. {
21. **return** **false**;   //存在
22. }
23. }
24. **return** Isavaible;
25. }
26. #endregion

    获得新站点ID。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 获得站点新ID
2. /// <summary>
3. /// 取得现有最大站点ID，在其上加1即为新站点ID
4. /// </summary>
5. **private** **string** GetNewWebSiteID()
6. {
7. **int** siteID = 1;
8. DirectoryEntry rootEntry = GetDirectoryEntry(entPath);
9. **foreach** (DirectoryEntry de **in** rootEntry.Children)
10. {
11. **if** (de.SchemaClassName == "IIsWebServer")
12. {
13. **int** ID = Convert.ToInt32(de.Name);
14. **if** (ID >= siteID)
15. {
16. siteID = ID + 1;
17. }
18. }
19. }
20. websiteid = siteID.ToString().Trim();
21. **return** websiteid;
22. }
23. #endregion

    判断应用程序池是否存在。

***应用程序池和网站站点的关系？***

    简单来说，它们是两个不同的事物，应用程序池就像一个容器，它里面可以“装”多个网站，但它也是有容量的限制的。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 判断应用程序池是否存在
2. /// <summary>
3. /// 返回true即应用程序池存在
4. /// </summary>
5. /// <param name="AppPoolName">应用程序池名</param>
6. **private** **bool** IsAppPoolName(**string** AppPoolName)
7. {
8. **bool** result = **false**;
9. DirectoryEntry appPools = **new** DirectoryEntry("IIS://localhost/W3SVC/AppPools");
10. **foreach** (DirectoryEntry getdir **in** appPools.Children)
11. {
12. **if** (getdir.Name.Equals(AppPoolName))
13. {
14. result = **true**;
15. }
16. }
17. **return** result;
18. }
19. #endregion

    创建站点。

此段代码包括：创建应用程序池、创建站点、设置网站托管模式和.NET运行版本、应用程序配置。

    兼容IIS6和IIS7，ServerManager是Microsoft.Web.Administration中的类，需引用Microsoft.Web.Administration.dll,目录为%WinDir%\System32\InerSrv,win7中即为C:\Windows\System32\InerSrv文件夹下。最好将Microsoft.Web.Administration.dll复制到自己的安装程序的目录下，再引用，因为有可能VS会发出无法使用类似的警告（具体警告内容不记得了，不过大概就如此了），即使有警告应该也不影响程序运行。

    代码顶端加上

      using Microsoft.Web.Administration;

      using System.Diagnostics;

//进程类，Process

***注意：***vdEntry.Properties["DefaultDoc"][0] = "login.aspx";是设置网站默认文档，即访问网站是看到的第一个页面。若在这里设置了默认文档，在配置文件（web.config）中不需再次设置，否则会出错。即要么在这里设置，要么在web.config中设置，只能取其一。

***web.config中设置网站默认文档：***

**[html]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. **<system.webServer>**
2. **<defaultDocument>**
3. **<files>**
4. **<add** value="login.aspx"**/>**
5. **</files>**
6. **</defaultDocument>**
7. **</system.webServer>**

    创建站点代码如下：

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 创建站点
2. **public** **void** CreateNewWebSite(NewWebSiteInfo siteInfo)
3. {
4. **if** (!EnsureNewSiteEnavaible(siteInfo.BindString))
5. {
6. **throw** **new** Exception("该网站已存在" + Environment.NewLine + siteInfo.BindString);
7. }
8. DirectoryEntry rootEntry = GetDirectoryEntry(entPath);
10. **string** newSiteNum = GetNewWebSiteID();
11. DirectoryEntry newSiteEntry = rootEntry.Children.Add(newSiteNum, "IIsWebServer");
12. newSiteEntry.CommitChanges();
14. newSiteEntry.Properties["ServerBindings"].Value = siteInfo.BindString;
15. newSiteEntry.Properties["ServerComment"].Value = siteInfo.NameOfWebSite;
16. newSiteEntry.CommitChanges();
17. DirectoryEntry vdEntry = newSiteEntry.Children.Add("root", "IIsWebVirtualDir");
18. vdEntry.CommitChanges();
19. **string** ChangWebPath = siteInfo.WebPath.Trim().Remove(siteInfo.WebPath.Trim().LastIndexOf('\\'), 1);
20. vdEntry.Properties["Path"].Value = ChangWebPath;

23. vdEntry.Invoke("AppCreate", **true**);//创建应用程序
25. vdEntry.Properties["AccessRead"][0] = **true**; //设置读取权限
26. vdEntry.Properties["AccessWrite"][0] = **true**;
27. vdEntry.Properties["AccessScript"][0] = **true**;//执行权限
28. vdEntry.Properties["AccessExecute"][0] = **false**;
29. //string defaultdoc = vdEntry.Properties["DefaultDoc"][0].ToString();//获取默认文档集
30. //vdEntry.Properties["DefaultDoc"][0] = "login.aspx";//设置默认文档
31. vdEntry.Properties["AppFriendlyName"][0] = "WebManager"; //应用程序名称
32. vdEntry.Properties["AuthFlags"][0] = 1;//0表示不允许匿名访问，1表示可以匿名访问，3为基本身份验证，7为windows继承身份验证
33. vdEntry.CommitChanges();

36. #region 针对IIS7
37. DirectoryEntry getEntity = **new** DirectoryEntry("IIS://localhost/W3SVC/INFO");
38. **int** Version = **int**.Parse(getEntity.Properties["MajorIISVersionNumber"].Value.ToString());
39. **if** (Version > 6)
40. {
41. #region 创建应用程序池
42. **string** AppPoolName = "WebManager";
43. **if** (!IsAppPoolName(AppPoolName))
44. {
45. DirectoryEntry newpool;
46. DirectoryEntry appPools = **new** DirectoryEntry("IIS://localhost/W3SVC/AppPools");
47. newpool = appPools.Children.Add(AppPoolName, "IIsApplicationPool");
48. newpool.CommitChanges();
49. }
50. #endregion
52. #region 修改应用程序的配置(包含托管模式及其NET运行版本)
53. ServerManager sm = **new** ServerManager();
54. sm.ApplicationPools[AppPoolName].ManagedRuntimeVersion = "v4.0";
55. sm.ApplicationPools[AppPoolName].ManagedPipelineMode = ManagedPipelineMode.Integrated; //托管模式:Integrated为集成 Classic为经典
56. sm.CommitChanges();
57. #endregion
59. vdEntry.Properties["AppPoolId"].Value = AppPoolName;
60. vdEntry.CommitChanges();
61. }
62. #endregion

65. //启动aspnet\_regiis.exe程序
66. **string** fileName = Environment.GetEnvironmentVariable("windir") + @"\Microsoft.NET\Framework\v4.0.30319\aspnet\_regiis.exe";
67. ProcessStartInfo startInfo = **new** ProcessStartInfo(fileName);
68. //处理目录路径
69. **string** path = vdEntry.Path.ToUpper();
70. **int** index = path.IndexOf("W3SVC");
71. path = path.Remove(0, index);
72. //启动ASPnet\_iis.exe程序,刷新脚本映射
73. startInfo.Arguments = "-s " + path;
74. startInfo.WindowStyle = ProcessWindowStyle.Hidden;
75. startInfo.UseShellExecute = **false**;
76. startInfo.CreateNoWindow = **true**;
77. startInfo.RedirectStandardOutput = **true**;
78. startInfo.RedirectStandardError = **true**;
79. Process process = **new** Process();
80. process.StartInfo = startInfo;
81. process.Start();
82. process.WaitForExit();
83. **string** errors = process.StandardError.ReadToEnd();
84. **if** (errors != **string**.Empty)
85. {
86. **throw** **new** Exception(errors);
87. }
88. }
89. #endregion

    设置网站目录权限。

    某些情况下，也可不进行此步骤。

    命名空间：using System.Security.AccessControl;

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 设置文件夹权限
2. /// <summary>
3. /// 设置文件夹权限 处理给EVERONE赋予所有权限
4. /// </summary>
5. /// <param name="FileAdd">文件夹路径</param>
6. **private** **void** SetFileRole()
7. {
8. //physicaldir = physicaldir.Remove(physicaldir.LastIndexOf('\\'), 1);
9. DirectorySecurity fSec = **new** DirectorySecurity();
10. fSec.AddAccessRule(**new** FileSystemAccessRule("Everyone", FileSystemRights.FullControl, InheritanceFlags.ContainerInherit | InheritanceFlags.ObjectInherit, PropagationFlags.None, AccessControlType.Allow));
11. System.IO.Directory.SetAccessControl(physicaldir, fSec);
12. }
13. #endregion

    D、写入数据库连接字符串至配置文件

    也可考虑将数据库连接字符串写入注册表。

    web.config中连接字符串格式如下：

**[html]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. **<configuration>**
2. **<appSettings/>**
3. **<connectionStrings>**
4. **<add** name="conn" connectionString="Database=MyWebDB;Server=.;Uid=sa;Pwd=123456;"**/>**
5. **</connectionStrings>**

**注意：**若数据库服务器名称（即上文的Server）不设置为“.”，而是设置成具体IP，这就需要数据库打开远程访问功能，否则连接出错。

    CODE如下：

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 写入连接字符串至配置文件
2. **private** **void** WriteWebConfig()
3. {
4. //加载配置文件
5. **string** path = physicaldir;
6. **string** pathpwd = pwd;
7. System.IO.FileInfo FileInfo = **new** System.IO.FileInfo(physicaldir + "web.config");
8. **if** (!FileInfo.Exists)
9. {
10. **throw** **new** InstallException("缺少配置文件 :" + physicaldir + "web.config");
11. }
12. System.Xml.XmlDocument xmlDocument = **new** System.Xml.XmlDocument();
13. xmlDocument.Load(FileInfo.FullName);
15. //写入连接字符串
16. **foreach** (System.Xml.XmlNode Node **in** xmlDocument["configuration"]["connectionStrings"])
17. {
18. **if** (Node.Name == "add")
19. {
20. **if** (Node.Attributes.GetNamedItem("name").Value == "conn")
21. {
22. Node.Attributes.GetNamedItem("connectionString").Value = String.Format("Database={0};Server={1};Uid={2};Pwd={3};", dbname, dbserver, user, pwd);
24. }
26. }
27. }
28. xmlDocument.Save(FileInfo.FullName);
29. }
30. #endregion

    E、注册表操作

注册表编辑器也可操作注册表，不爽的，所以代码操作。在操作注册表时，有出现过对注册表修改，没有任何结果，只是调试看到抛出异常，不明所以。

    regedit打开注册表，看看原因。右键点击注册表中的HKEY\_CURRENT\_USER，选择“权限”。可以看到“RESTRICTED”（限制）用户并没有“完全控制”权限。想想可能原因就在这儿，是不是没有修改注册表该项的权限，勾上试试，运行，果然可以修改注册表了。但这样似乎有些麻烦，有没有更直接的办法？

    当然有。既然问题出在权限上，那就在执行时赋给我们相应权限，如下所示，即可访问并修改注册表了。

    OpenSubKey("Software",RegistryKeyPermissionCheck.ReadWriteSubTree,RegisTryRights.ChangePermissions)

    若操作注册表项“HKEY\_LOCAL\_MACHINE”,RegisTryRights.ChangePermissions需改为RegisTryRights.FullControl,否则无法操作。假设向“HKEY\_LOCAL\_MACHINE”的“SOFTWARE”项中加入子项“Test”；打开注册表，在SOFTWARE项下没有刚加的Test子项，其实打开SOFTWARE下的Wow6432Node子项，会发现Test，访问此项的方式仍用SOFTWARE->Test。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. 命名空间：**using** Microsoft.Win32;

**网站ID写入注册表**（主要是供卸载软件时删除站点是使用）

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 网站ID写入注册表
2. **private** **void** WriteToReg(**string** name)
3. {
4. RegistryKey rk = Registry.CurrentUser.OpenSubKey("Software",RegistryKeyPermissionCheck.ReadWriteSubTree,RegistryRights.ChangePermissions);
5. RegistryKey MywebSite = rk.CreateSubKey("MyWebSite");
6. MywebSite.SetValue(name, websiteid,RegistryValueKind.DWord);
7. rk.Close();//修改内容刷新到磁盘
8. }
9. #endregion

**读取注册表中的值**

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 根据注册表项名读取值
2. **private** **string** ReadFromReg(**string** name)
3. {
4. **string** registData;
5. RegistryKey rk = Registry.CurrentUser.OpenSubKey("Software", RegistryKeyPermissionCheck.ReadSubTree, RegistryRights.ChangePermissions);
6. RegistryKey MyWebSite = rk.OpenSubKey("MyWebSite",**true**);
7. registData = MyWebSite.GetValue(name).ToString();
8. **return** registData;
9. }
10. #endregion

**删除注册表项**

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 删除注册表项
2. **private** **void** DeleteReg(**string** name)
3. {
4. RegistryKey rk = Registry.CurrentUser.OpenSubKey("Software", RegistryKeyPermissionCheck.ReadWriteSubTree, RegistryRights.ChangePermissions);
5. rk.DeleteSubKeyTree(name);
6. rk.Close();
7. }
8. #endregion

**F、创建桌面快捷方式**

           如果是要创建一个已经存在的程序的快捷方式，则可以在“文件系统”中的右键点击“应用程序文件夹”，选择“添加”进此程序，再右键点击添加的程序，选择“创建\*\*\*\*的快捷方式”，在将改快捷方式剪切到“用户桌面”文件夹中，搞定。

          但如果要创建快捷方式的程序事先并不存在，比如一个网站的地址，下面的方式即可解决。

          需引用名为Windows Script Host Object Model的COM组件。而命名空间却为：using IWshRuntimeLibrary;很特别~~~

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 创建桌面快捷方式
2. **private** **void** CreateDeskTopCut()
3. {
4. **string** dk = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Desktop);//得到桌面文件夹路径
6. IWshRuntimeLibrary.WshShell shell = **new** IWshRuntimeLibrary.WshShell();
7. IWshRuntimeLibrary.IWshShortcut shortcut = (IWshRuntimeLibrary.IWshShortcut)shell.CreateShortcut(@dk + "\\我的网站.lnk");
8. shortcut.TargetPath = @"%HOMEDRIVE%/Program Files\Internet Explorer\IEXPLORE.EXE";
9. shortcut.Arguments = @"http://" + iis + ":" + port;    //参数，这里就是网站地址
10. shortcut.Description = "我的网站快捷方式";
11. shortcut.IconLocation = @"%HOMEDRIVE%/Program Files\Internet Explorer\IEXPLORE.EXE, 0";//图标，以IE图标作为快捷方式图标
12. shortcut.WindowStyle = 1;
13. shortcut.Save();
14. }
15. #endregion

        G、创建应用程序菜单项

         卸载程序是win7“C:\Windows\System32”目录下的msiexec.exe程序，将其添加到“文件系统”中的“应用程序文件夹”里。按照上面所述方式为它创建一个快捷方式，右键点击快捷方式可选择“重命名”为其重命名，左键点击快捷方式，在“属性”窗口中可看到“Arguments”选项，在其中加入/x {ProductCode}，ProductCode为产品代码，左键点击创建的安装项目，可在其“属性”窗口中看到ProductCode。例：ProductCode为：44533S25-3342-89F4-H852-7KF682G90991，则Arguments中输入：/x {44533S25-3342-89F4-H852-7KF682G90991}，注意***/x后有一个空格***。卸载程序创建成功，此时可将msiexec.exe隐藏，“属性”窗口，“Hidden”选为True即可。

        直接使用msiexec.exe是不能卸载程序的，因为没有“Arguments”选项，所以它并不知道你要卸载的是哪个程序。必须得创建快捷方式，并加入ProductCode，它才能知道你到底要卸载哪个程序。msiexec.exe直接运行将没有任何作用（只出现对msiexec作简要说明的对话框而已），所以将它隐藏，一定程度上避免用户运行它。

        卸载程序的快捷方式添加到——用户的“程序”菜单，是不是就能在应用程序菜单中创建卸载程序呢？答案是肯定，但是如果命名不好的话容易造成误解，只是看到程序菜单项中有个卸载程序，而不知道它到底是卸载谁的。那么就在用户的“程序”菜单文件夹中在创建一个“我的网站”的文件，再将卸载的快捷方式放进去，这样用户就知道这是“我的网站”的卸载程序了。

        但这样又有另外一个问题，即我们已在应用程序菜单中创建了一个"我的网站"的文件夹，里面放了一个到我的网站的快捷方式，程序会如何处理它们？其实，这样做，安装程序会在应用程序菜单中创建两个文件夹，名字都为“我的网站”，一个放了到我的网站的快捷方式，一个放了到卸载程序的快捷方式。显然，这不是我们想要的。

        肿么办？很简单，不用上述方法。在创建我的网站的快捷方式时，同时创建卸载程序的快捷方式。

**注意：必须为卸载方式提供参数ProductCode，如：shortcut3.Arguments = @"/x {44533S25-3342-89F4-H852-7KF682G90991}";**，不这样做，而想把它的目标指向卸载程序的快捷方式或卸载程序都是不行的。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 创建应用程序菜单项
2. **private** **void** CreateProCut()
3. {
4. **string** prodir = System.Environment.GetFolderPath(System.Environment.SpecialFolder.StartMenu);   //得到应用程序菜单文件夹路径
5. prodir += "\\" + "我的网站程序菜单";
6. **if** (!System.IO.Directory.Exists(prodir))
7. {
8. System.IO.Directory.CreateDirectory(prodir);
9. }
10. IWshRuntimeLibrary.WshShell shell2 = **new** IWshRuntimeLibrary.WshShell();
11. IWshRuntimeLibrary.IWshShortcut shortcut2 = (IWshRuntimeLibrary.IWshShortcut)shell2.CreateShortcut(@prodir + "\\我的网站.lnk");
12. shortcut2.TargetPath = @"%HOMEDRIVE%/Program Files\Internet Explorer\IEXPLORE.EXE";
13. shortcut2.Arguments = @"http://" + iis + ":" + port;    //参数
14. shortcut2.Save();
15. IWshRuntimeLibrary.WshShell shell3 = **new** IWshRuntimeLibrary.WshShell();
16. IWshRuntimeLibrary.IWshShortcut shortcut3 = (IWshRuntimeLibrary.IWshShortcut)shell3.CreateShortcut(@prodir + "\\卸载.lnk");
17. shortcut3.Description = "我的网站卸载程序";
18. shortcut3.Arguments = @"/x {44533S25-3342-89F4-H852-7KF682G90991}";    //参数
19. shortcut3.TargetPath = @physicaldir + "Uninstall.exe";<span style="WHITE-SPACE: pre">   </span>//卸载程序被重命名为Uninstall.exe
20. shortcut3.Save();
21. }
22. #endregion

**H、程序安装完后自启动**

OnCommitted()为程序安装完后执行的动作，p.WaitForExit();在程序还没装完的时候就开始执行了，肯定得不到正确结果。p.WaitForInputIdle();会在程序安装完后开始执行，但还没关掉最后的安装成功窗口，就开始执行了（可使进程判断用户关闭安装窗口后再开始执行）。

      同时，打开的IE窗口和平常的不一样，很简洁。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 程序安装完自启动
2. **protected** **override** **void** OnCommitted(IDictionary savedState)
3. {
4. **base**.OnCommitted(savedState);
5. **if** (isrun == "1")
6. {
7. Process p = **new** Process();
8. p.StartInfo.FileName = "IExplore.exe";
9. p.StartInfo.Arguments = @"http://" + iis + ":" + port; ;
10. p.StartInfo.WindowStyle = ProcessWindowStyle.Maximized;
11. p.Start();
12. p.WaitForInputIdle();
13. //p.WaitForExit();
15. }
16. }
17. #endregion

        I、删除站点

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 删除站点
2. /// <summary>
3. ///  删除一个网站。根据网站ID（注册表中获得）删除。
4. /// </summary>
5. /// <param name="siteID">网站ID</param>
6. **public** **void** DeleteWebSiteByID(**string** siteID)
7. {
8. **string** siteEntPath = String.Format("IIS://{0}/w3svc/{1}", "localhost", siteID);
9. DirectoryEntry siteEntry = GetDirectoryEntry(siteEntPath);
10. DirectoryEntry rootEntry = GetDirectoryEntry(entPath);
11. rootEntry.Children.Remove(siteEntry);
12. rootEntry.CommitChanges();
13. }
14. #endregion

**J、删除空的应用程序池**

          卸载时，若网站所在应用程序池内已没有任何应用程序，即其中没有网站了，就删除安装网站时创建的应用程序池（当然前提是该应用程序池存在）。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 删除空应用程序池
2. /// <summary>
3. /// 删除没有站点的指定应用程序池
4. /// </summary>
5. /// <param name="AppPoolName">程序池名称</param>
6. /// <returns>true删除成功 false删除失败</returns>
7. **private** **bool** DeleteAppPool(**string** AppPoolName)
8. {
9. **bool** result = **false**;
10. DirectoryEntry appPools = **new** DirectoryEntry("IIS://localhost/W3SVC/AppPools");
11. **foreach** (DirectoryEntry getdir **in** appPools.Children)
12. {
13. **if** (getdir.Name.Equals(AppPoolName))
14. {
15. **try**
16. {
17. DirectoryEntry findPool = appPools.Children.Find(AppPoolName, "IIsApplicationPool");
18. **object**[] s = findPool.Invoke("EnumAppsInPool", **null**) **as** **object**[];
19. **if** (s.Length == 0)  //应用程序池下没有站点
20. {
21. getdir.DeleteTree();
22. result = **true**;
23. }
24. }
25. **catch**
26. {
27. result = **false**;
28. }
29. }
30. }
31. **return** result;
32. }
33. #endregion

**K、卸载程序**

         卸载程序时执行的动作，包括删除站点、删除快捷方式（如果存在）、应用程序菜单相应文件夹（如果存在）。

**[csharp]** [view plaincopy](http://blog.csdn.net/ts1030746080/article/details/8753203)

1. #region 程序卸载
2. **public** **override** **void** Uninstall(IDictionary savedState)
3. {
4. **base**.Uninstall(savedState);
5. DeleteWebSiteByID(ReadFromReg("WebSiteID"));
6. DeleteAppPool("WebManager");
7. DeleteReg("MyWebSite");
8. **string** dk = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Desktop);
9. dk = @dk + "\\我的网站.lnk";
10. **if** (System.IO.File.Exists(dk))
11. {
12. //如果存在则删除
13. System.IO.File.Delete(dk);
14. }
15. **string** prodir = System.Environment.GetFolderPath(System.Environment.SpecialFolder.StartMenu);   //得到应用程序菜单文件夹
16. prodir += "\\" + "我的网站程序菜单";
17. **if** (System.IO.Directory.Exists(prodir))
18. {
19. System.IO.Directory.Delete(prodir, **true**);
20. }
21. }
22. #endregion

**L、自述文件、许可协议**  
         “用户界面”编辑器中，可加入“自述文件”、“许可协议”对话框。先将写好的自述文件和许可协议“添加”到“应用程序文件夹”，再在“自述文件”、“许可协议”对话框的“属性”窗口中的ReadmeFile、LicenseFile分别选择对应的文件即可。

         许可协议采用rtf格式，自述文件也可采用此种格式。可用word编辑，**但需要在word中新建一个空白文档，在其中写入内容，最后保存为rtf格式。若使用doc格式编辑，然后改后缀为rtf，这样会造成乱码**。  
       **M、加入网站**  
        接下来就要将我们预编译（发布）后的网站加入到安装程序类中了。

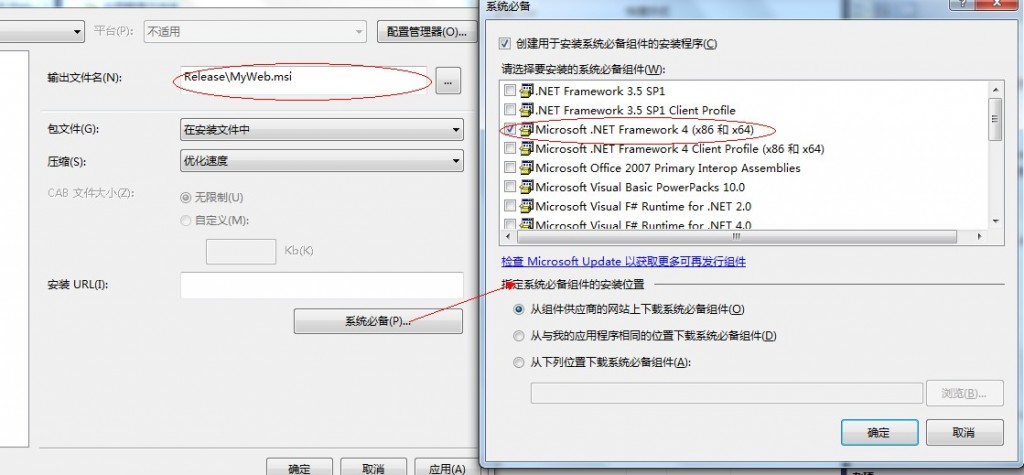
        一种方式是在“文件系统”中按照网站目录，逐个添加完网站内所有文件和文件夹到“应用程序文件夹”中，倘若网站内文件夹或文件很多，太繁琐（在同一个目录下，添加多个文件时，可用ctrl选中多个文件一次添加，但目录结构太复杂也很麻烦）。

        另一种方式，“文件”->"添加"->“现有网站”。将网站加进去，会提示网站预编译什么的一段废话，不用管。再右键点击安装项目名，选择“添加”->“文件”，选择网站，添加即可。安装项目下就会多一个子项“内容文件来自.....” ，一次搞定。

**N、 自带程序运行依赖项的安装**

       安装好的程序运行时，有一些运行必备条件，比如网站是基于.Net FrameWork 4.0做的，那要运行它就需要电脑上装有.Net FrameWork 4.0框架。如何让安装程序自动检测电脑上是否安装.Net FrameWork 4.0，并给没有安装.Net FrameWork 4.0的电脑自动安装？

       在安装项目的"检测到的依赖项"中，生成类库时加入了“Microsoft .NET Framework”。右键单击安装项目选择属性（可在此选择输出的安装程序的名字），在属性对话框中选择“系统必备”，便可选择相应安装必备组件，“指定系统供应商的网站上下载系统必备组件”中第一个从网上下，第二个从安装程序中找。



        若选择第二个，这可看到项目的“【项目名】”->“【项目名】”->"Debug"（或Release）文件下多了一个DotNetFX40文件夹，里面装的就是.Net FrameWork 4.0。

        在“文件系统”的“应用程序文件夹”中加入.Net FrameWork 4.0。

        再在“启动条件编辑器”中选择“启动条件”下的“.NET Framework”，在它的属性窗口中，修改InstallUrl为“DotNetFX40\”

**O、生成安装项目**

       生成安装项目，解决方案配置debug、release看情况选择（一个调试版，一个发布版）。有网站预编译什么的错误，不用管，不影响。

**P、注意事项**

        确保“启动条件”下、系统必备、类库、网站的.Net FrameWork版本都一致。

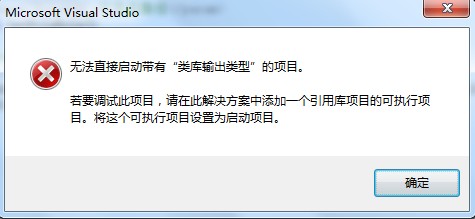
        确保平台相同，安装项目的属性窗口（右键单击所出现的属性窗口）中的“TargetPlatfrom”平台默认为x86，但win7下卸载程序运行平台是x64，会出现警告，说两者目标平台不一致（实际暂不碍事），其属性窗口还可设置安装项目的一些属性，AddRemovePro是设置卸载程序的图标。

        如果卸载程序所控制的安装类库和待卸载程序的安装类库（已生成为dll文件）不一样，则可能造成程序无法卸载，系统提示程序无法完全卸载，需在控制面板中的程序和功能中彻底卸载，然后回滚，程序卸载失败，重启也不行。这时需要将最新的安装类库dll替换掉待卸载程序中的相应dll，再卸载即可。

        在安装程序时，选择为“任何人”或“只为我”安装时，一般选任何人，若选只为我，这有可能出现安装的文件夹无法删除，导致无法卸载，也无法再次安装到此目录下。若出现了这种情况可，使用管理员获得此文件夹权限，关闭vs，为安装程序所在文件夹添加everyone用户，并赋给它所有权限，在删除安装文件夹中的网站根目录，再删除其上级安装目录（若存在的话）。

**Q、安装项目的调试**

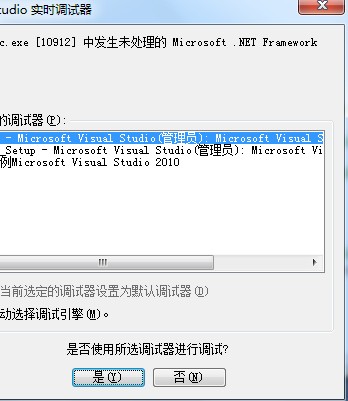
         安装项目直接调试（F5）有如下错误：



         所以不能直接调试，有三种其他的调试方法：

**1、System.Diagnostics.Debugger.Launch();**

               在要调试的代码前加上System.Diagnostics.Debugger.Launch();，右键单击安装项目，选择安装，当安装程序执行到加入此句的地方时，会弹出如下提示框：

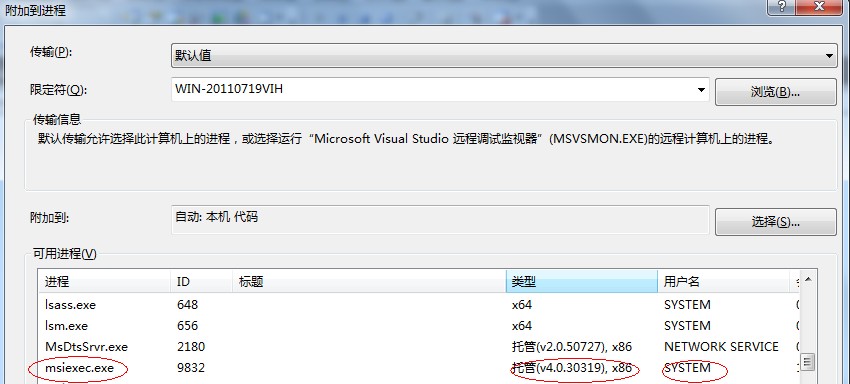


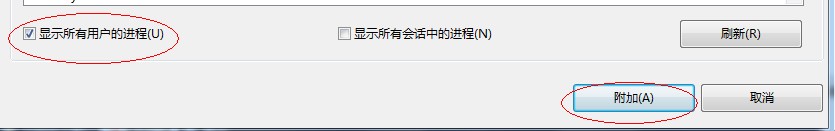
           选择一个实例进入其中就可调试代码了。

**2、使用MessageBox中断程序进行调试**

          在要调试的代码前加入MessageBox.Show("Debug Me");  [添加引用System.Windows.Forms]，MessageBox中的内容任意，它只是起中断安装程序的作用。

          右键单击安装项目，选择安装，当安装程序执行到加入此句的地方时，会弹出消息框”Debug Me“，不管他，切到VS窗口，此时使用ctrl+alt+p，附加进程，选择“显示所有进程”，选择如下进程（***注意所选msiexec进程的类型和用户名***），附加。





          此时，在调试代码处加入断点，成功加入后，点击全部中断（ctrl+alt+break），程序开始从MessageBox处开始执行，再点击继续（F5），程序会弹出MessageBox对话框，这是就可点击对话框的确定按钮了，然后程序直接跳到断点处，逐过程（F10）或断点调试都可以。

**3、使用 InstallUtil.exe**

          将调试首选项设置为启动 InstallUtil.exe（位于 \winnt\Microsoft.net\Framework\*版本*），并将其作为参数传递给程序集。 按 F5 时，命中断点。 InstallUtil.exe 将采用与 MSI 相同的方式运行自定义操作。