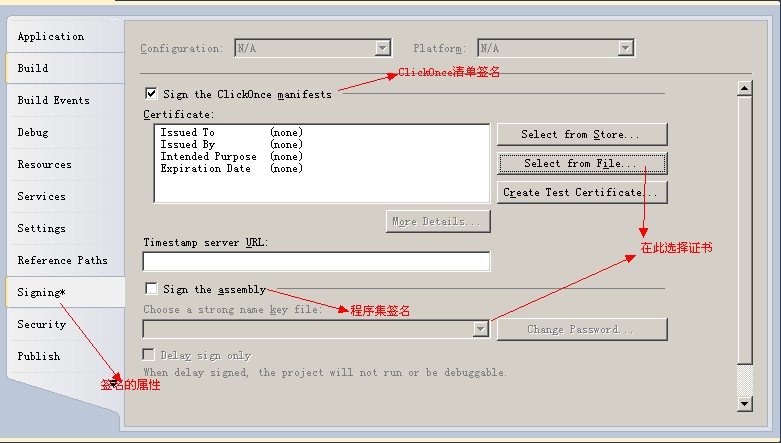
**WinForm程序安装、发布流程**

# 一、签名

所谓签名就是给应用程序一个身份，申请一个专利。签名的时候需要需要选择证书。就向我们上学一样，得奖了老师给你发个证书。如果不进行签名，杀毒软件会把你打包后的exe文件作为病毒处理。

签名的步骤：(1)右键点击所要发布的项目的属性。弹出如下图所示的窗口。

****

# 二、证书

在签名的部分提到了证书，在创建证书的时候我们一定要自己创建。因为在vs上面创建的证书默认到期时间都只有一年，这意味着每过一年你都要为你安装的应用程序再次创建一个证书。我们使用微软自带的vs控制台命令工具进行我们自己的自定义证书创建，可以设置有效期限。步骤如下：

打开Microsoft .NET Framework 的SDK命令提示，按以下步骤操作：

1. 创建一个自我签署的X.509证书（.cer）和一个.pvk私钥文件，用到makecert工具，命令如下：
2. makecert -r -n "CN=sqs3ng " -b 01/01/2011 -e 01/01/2020 -sv myselfName.pvk myselfName.cer  
   按提示设置私钥密码（也可以不使用密码）即可在当前目录生成相关文件

2、利用X.509证书（.cer）创建发行者证书 (.spc)，用到cert2spc工具，命令如下：  
cert2spc myselfName.cer myselfName.spc

3、从.pvk和.spc格式转换成.pfx格式，用到pvkimprt工具，命令如下：  
pvkimprt -pfx myselfName.spc myselfName.pvk

**注意此处需要下载并安装pvkimprt.exe，否则会提示"pvkimprt"不是内部或外部命令，也不是可运行的程序或批处理文件。**

pvkimprt工具下载地址如下，下载后要先解压。

http://download.microsoft.com/download/vba50/Utility/1.0/NT5/EN-US/pvkimprt.exe

按提示操作可导出.pfx证书，若第1步设置了私钥密码，此处需要输入验证

证书导出工具用于将.spc和.pvk文件导出为pfx证书文件。

导出时，会提示创建证书的密码

输入正确的密码后，出现导出向导界面，点击下一步，选择“是，导出密钥”，然后点击下一步，

出现证书类型选择界面，选择“个人信息交换 - PKCS #12(.PFX)(P)”（只有这一种可选，并且其下面的选项均不选择），再点击下一步，

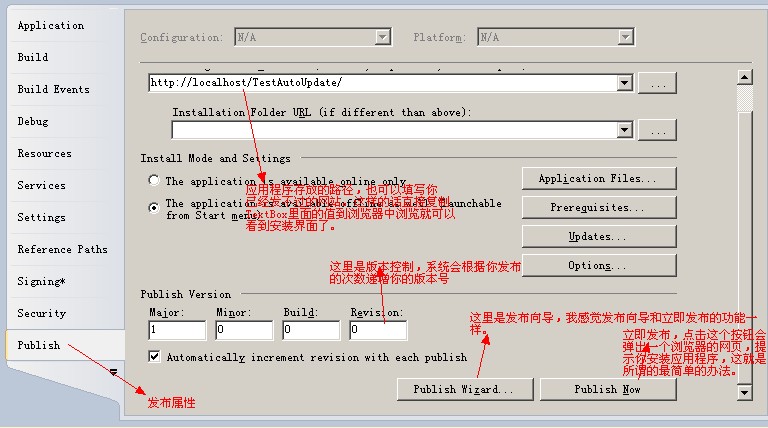
出现私钥保护密码选项，输入密码，然后点击下一步，在文本框中输入要生成的文件名（一般与证书名称一致），点击下一步

显示完成窗口，并显示导出证书的信息。

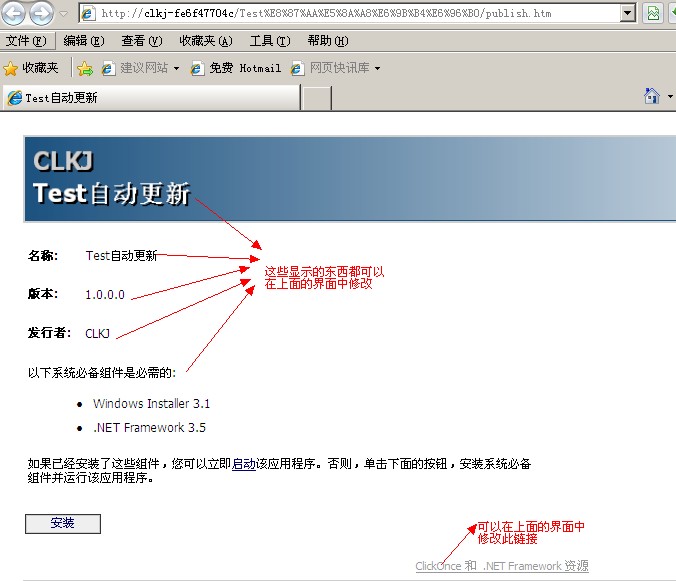
# 三、发布

## （1）web浏览器发布应用程序

首先我给大家介绍一种很快捷的方法，不用在vs上面添加部署的那个项目。也就是所谓的web浏览器发布应用程序。还是和上面一样。在所要发布的项目上面右击属性，选择publish属性

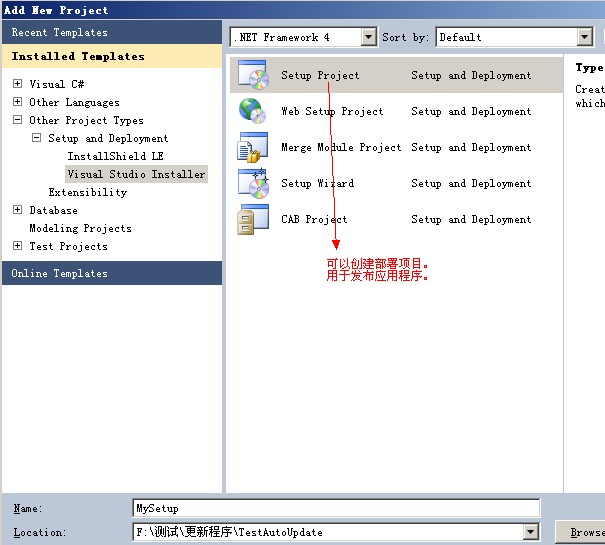
****

下图是应用程序再浏览器中的效果：

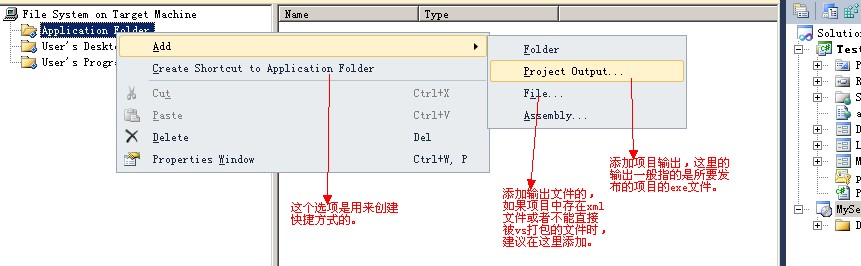
****

## （2）添加安装、部署项目

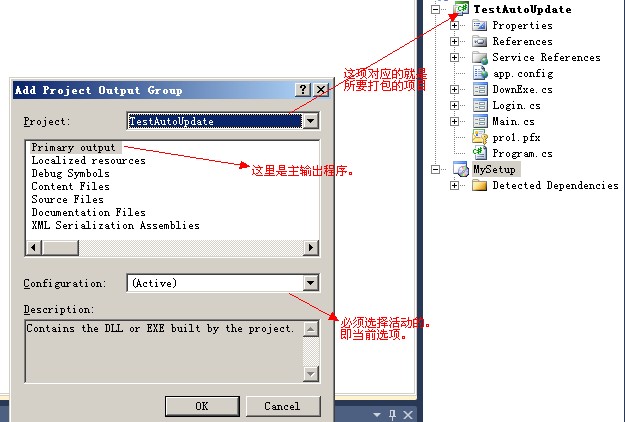
右键点击整个解决方案添加安装项目。如下图



接着就开始添加主要输出的文件，就是所谓打包的项目。如下图



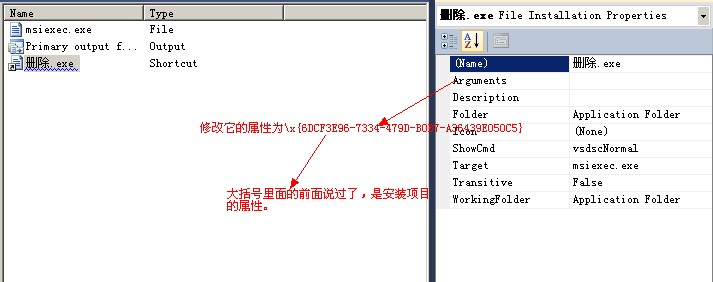
按照上面的步骤进入添加项目输出组窗口，如下图

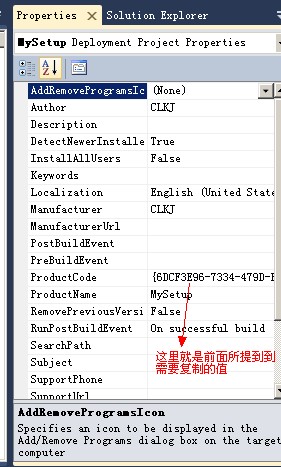


**接下来我来教你如何添加一个卸载程序**

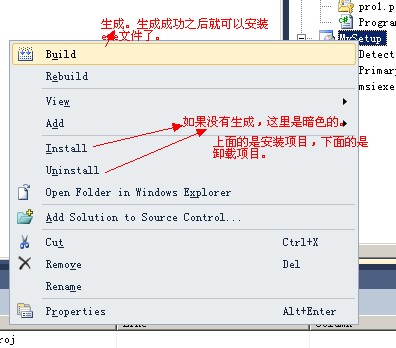
按照最上面的图片上显示的，添加一个输出的文件。找到C:\WINDOWS\system32目录下选择msiexec.exe文件。将msiexec.ex添加到文件夹中后，给它增加一个快捷方式，改名为delete.exe。之后修改delete.exe的属性Arguments为\x{6DCF3E96-7334-479D-B0B7-A36439E050C5}，大括号中的数字是安装项目的ProductCode属性(查找此属性必须是左键选中安装项目，不是右键点击查找属性)。

之后生成安装项目。待生成成功后，就可进行安装使用。具体视图如下：





下面的是安装和卸载的视图。



# 四、手动更新

手动更新就是客户在使用的时候，可以随时到网上去查看是否有最新版本的软件。下面是我的具体思路，已经得到实现：

1. 写一个webservice，提供一个获取服务器xml中版本的数据的方法。
2. 在WinForm应用程序启动的时候，首先访问webservice获取服务器的xml中的版本号，然后获取客户端的xml中的版本号。将两个版本号比较，若服务器中的版本号大，则提示到指定网站进行手动下载更新。

手动更新在这里就不为大家介绍了。下面详细说一下我最有收获的项目：自动更新。在自动更新里面含盖了上面的手动更新的所有步骤。

# 五、自动更新

自动更新的软件的目的在于让客户不在为了寻找最新软件花费时间。也不用去到开发商的网站上查找。客户端的软件自动会在程序启动前查找服务器上最新的版本。和自己当前软件的版本比较，如果服务器的是最新版本。客户端则进行自动下载、解压、安装。当然了下载是要有网络的，并且用户可以根据提示去完成操作。再也不用为找不到最新版本的软件而头疼。下面是我的大体思路，已经得到了实现：

1. 写一个webservice，提供一个获取服务器xml中版本的数据的方法。
2. 在WinForm应用程序启动的时候，首先访问webservice获取服务器的xml中的版本号，然后获取客户端的xml中的版本号。将两个版本号比较，若服务器中的版本号大，则提示提示可以在线更新应用程序。
3. 然后客户端自动下载网络上的RAR压缩包到本机的指定位置，进行自动解压缩。解压缩完毕之后，用进程打开所解压过的exe文件进行软件安装。同时关闭客户端软件所在的进程。

## （1）web项目中的代码

首先我给大家先展示下我的web项目中的webservice的代码，这里面跟简单，只有一个方法。项目需要发布到IIS上面。

### 1.1 webservice中的代码

/// <summary>

/// 获取xml中的软件最新版本

/// </summary>

/// <returns></returns>

[WebMethod]

public string GetEdition(string password)

{

string s = string.Empty;

//在使用此方法的时候必须先判断传进来的参数，看其是否有权限使用此方法。

if (password == "clkj ")

{

XDocument document = XDocument.Load(Server.MapPath("XMLEdition.xml"));

XElement element = document.XPathSelectElement("Content/Project/Edition");

if (element != null)

{

s = element.Value.ToString();

}

}

return s;

}

### 1.2 xml中的代码

Xml代码如下面所示，很简单

<?xml version="1.0" encoding="utf-8" ?>

<Content>

<Project id="p1">

<Name>test</Name>

<Edition>2.0</Edition>

</Project>

</Content>

## （2）WinForm项目中的代码

Web项目的代码就只有上面的一点，重点的还是在WinForm中。在WinForm中首先要添加web引用，将上述的webservice访问地址复制过来使用。下面我一步一步来为大家解析吧。

第一步：

### 2.1 xml中的代码

客户端的代码和服务器断的xml代码大致相同，不同的只用Edition元素里面的值。

<?xml version="1.0" encoding="utf-8" ?>

[**-**](file:///F:\测试\更新程序\TestAutoUpdate\TestAutoUpdate\bin\Debug\XMLEdition.xml) <Content>

[**-**](file:///F:\测试\更新程序\TestAutoUpdate\TestAutoUpdate\bin\Debug\XMLEdition.xml) <Project id="**p1**">

<Name>**test**</Name>

<Edition>**1.0**</Edition>

</Project>

</Content>

### 2.2 Program.cs代码(设置起始页的代码)

在Program.cs(WinForm中设置起始页的地方)这个类中添加代码

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

LoadMath();

}

private static void LoadMath()

{

//服务器上的版本号

string NewEdition = string.Empty;

//应用程序中的版本号

string OldEdition = string.Empty;

try

{

//获取webservice上的版本号

myService.WebServiceUpdateSoapClient c = new myService.WebServiceUpdateSoapClient();

NewEdition = c.GetEdition("clkj\_ws");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

try

{

//获取系统中xml里面存储的版本号

XDocument document = XDocument.Load("XMLEdition.xml");

XElement element = document.XPathSelectElement("Content/Project/Edition");

if (element != null)

{

OldEdition = element.Value.ToString();

}

}

catch (Exception exx)

{

MessageBox.Show(exx.Message);

}

double newE = double.Parse(NewEdition);

double oldE = double.Parse(OldEdition);

//比较两个版本号，判断应用程序是否要更新

if (newE > oldE)

{

//更新程序¨°

DialogResult dr = MessageBox.Show("发现新的版本是否要更新该软件", "系统提示?", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);

if (dr == DialogResult.OK)

{

//打开下载窗口

Application.Run(new DownUpdate ());

}

else

{

//若用户取消，打开初始界面

Application.Run(new Login());

}

}

}

### 2.3 Main.cs(登录后的主界面)的代码

这个可以省略,没有实际意义

### 2.4 DownUpdate.cs(更新页面)的代码

界面显示如下图



自动更新代码如下（其中更新按钮的name为btnDown，安装按钮的name为btnInstall）:

//debug目录，用于存放压缩文件t

string path = Application.StartupPath;

public DownExe()

{

InitializeComponent();

}

private void DownExe\_Load(object sender, EventArgs e)

{

btnInstall.Enabled = false;

}

//下载文件、自动解压缩文件

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

{

//自动下载压缩包,并且解压，最后关闭当前进程，进行安装

try

{

//设置进度条

List<int> resultList = new List<int>();

for (int i = 0; i < 100; i++)

{

resultList.Add(i);

}

//设置进度条的最大值和最小值

this.progressBar1.Maximum = resultList.Count;

this.progressBar1.Minimum = 0;

foreach (int item in resultList)

{

//下载压缩包

System.Net.WebClient client = new System.Net.WebClient();

client.DownloadFile(@"http://192.168.1.120/File/setup.rar", path + @"setup.rar");

this.progressBar1.Value++;

}

}

catch

{

MessageBox.Show("当前没有网络或者文件地址不正确");

}

if (!Exists())

{

MessageBox.Show("不支持RAR解压缩");

return;

}

//解a压1

try

{

unCompressRAR(path + "\\setup", path, "setup.rar", false);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

btnInstall.Enabled = true;

btnDown.Enabled = false;

}

//打开下载好的exe文件，并且关闭当前客户端所在的进程

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

{

if (File.Exists(path + @"\setup\cboxbeta2.0.0.7.exe"))

{

//打开下载好的exe文件

Process.Start(path + @"\setup\cboxbeta2.0.0.7.exe");

//关闭当前进程

Process[] proce = Process.GetProcesses();

foreach (Process p in proce)

{

//当前运行软件的进程名字

if (p.ProcessName == "TestAutoUpdate.vshost")

{

p.Kill();

break;

}

}

}

}

/// <summary>

/// 解压缩文件t

/// </summary>

/// <param name="unRarPatch">解压缩后的文件所要存放的路径?</param>

/// <param name="rarPatch">rar文件所在的路径</param>

/// <param name="rarName">rar文件名</param>

/// <param name="deleteFlag"></param>

/// <returns></returns>

public static string unCompressRAR(string unRarPatch, string rarPatch, string rarName, bool deleteFlag)

{

try

{

RegistryKey key = Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\WinRAR.exe");

string str = key.GetValue("").ToString();

key.Close();

if (!System.IO.Directory.Exists(unRarPatch))

{

System.IO.Directory.CreateDirectory(unRarPatch);

}

string str2 = "x \"" + rarName + "\" \"" + unRarPatch+"\""+" -y";

ProcessStartInfo info = new ProcessStartInfo();

info.FileName = str;

info.Arguments = str2;

info.WindowStyle = ProcessWindowStyle.Hidden;

info.WorkingDirectory = rarPatch;

Process process = new Process();

process.StartInfo = info;

process.Start();

process.WaitForExit();

process.Close();

if (deleteFlag && System.IO.File.Exists(rarPatch + @"\" + rarName))

{

System.IO.File.Delete(rarPatch + @"\" + rarName);

}

}

catch (Exception exception)

{

throw exception;

}

return unRarPatch;

}

/// <summary>

/// 判读是否支持RAR自动解压缩

/// </summary>

/// <returns></returns>

public static bool Exists()

{

return !string.IsNullOrEmpty(Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\WinRAR.exe").GetValue("").ToString());

}