摘要：
本文针对目前C/S模式下编写的应用程序可维护性差的特点，提出了一套自动在线升级的解决方案，分析了在线升级的困难及实现原理，并给出了实现升级的部分代码，具有实际参考价值和现实意义。本文程序代码均在.Net Framework 1.1 和Windows2000下测试通过。
关键词：C#；在线升级；自动升级；下载；XML 文档

1 前言
长期以来，广大程序员为到底是使用Client/Server，还是使用Browser/Server结构争论不休，在这些争论当中,C/S结构的程序可维护性差，布置困难，升级不方便，维护成本高就是一个相当重要的因素。有很多企业用户就是因为这个原因而放弃使用C/S。然而当一个应用必须要使用C/S结构才能很好的实现其功能的时候，我们该如何解决客户端的部署与自动升级问题？部署很简单，只要点击安装程序即可，难的在于每当有新版本发布时，能够实现自动升级。现在好了，我们的目标很简单，我们希望开发一个与具体应用无关的能够复用的自动升级系统。下面我为大家提供了一套可复用的用C#编写的自动升级系统。
2 实现软件的自动升级存在的困难
第一，为了查找远程服务器上的更新，应用程序必须有查询网络的途径，这需要网络编程、简单的应用程序与服务器通讯的协议。
第二是下载。下载看起来不需要考虑联网的问题，但要考虑下载用户请求的文件，以及在没有用户同意时下载大文件。友好的自动更新应用程序将使用剩余的带宽下载更新。这听起来简单，但却是一个技术难题，幸运的是已经有了解决方法。
第三个考虑因素是使用新版应用程序更换原应用程序的过程。这个问题比较有趣，因为它要求代码运行时将自己从系统删除，有多种办法可以实现该功能，本文程序主要通过比较新旧版本的日期号来实现替换新版本应用程序的功能。
3 实现软件自动在线升级的原理
写两个程序，一个是主程序；一个是升级程序；所有升级任务都由升级程序完成。
1.启动升级程序，升级程序连接到网站，下载新的主程序（当然还包括支持的库文件、XML配置文档等）到临时文件夹；
2.升级程序获取服务器端XML配置文件中新版本程序的更新日期或版本号或文件大小；
3.升级程序获取原有客户端应用程序的最近一次更新日期或版本号或文件大小，两者进行比较：如果发现升级程序的日期大于原有程序的最新日期，则提示用户是否升级；或者是采用将现有版本与最新版本作比较，发现最新的则提示用户是否升级；也有人用其

它属性如文件大小进行比较，发现升级程序的文件大小大于旧版本的程序的大小则提示用户升级。本文主要采用比较新旧版本更新日期号来提示用户升级。

4.如果用户选择升级，则获取下载文件列表，开始进行批量下载文档；

5.升级程序检测旧的主程序是否活动，若活动则关闭旧的主程序；

6.删除旧的主程序，拷贝临时文件夹中的文件到相应的位置；

7.检查主程序的状态，若状态为活动的，则启动新的主程序；

8.关闭升级程序，升级完成[4]。

4 用C#实现在线升级的关键步骤

这里我主要使用日期信息来检测是否需要下载升级版本。

服务器升级配置文件

4.1 准备一个XML配置文件：名称为AutoUpdater.xml，作用是作为一个升级用的模板，显示需要升级的信息。

```
<?xml version="1.0"?>
<!--xml 版本号-->
<AutoUpdater>
    <URLAddres URL="http://192.168.198.113/vbroker/log/"/>
    <!--升级文件所在服务器端的网址-->
    <UpdateInfo>
        <UpdateTime Date = "2005-02-02"/>
        <!--升级文件的更新日期-->
        <Version Num = "1.0.0.1"/>
        <!--升级文件的版本号-->
    </UpdateInfo>
    <UpdateFileList>
        <!--升级文件列表-->
```

```
        <UpdateFile FileName = "aa.txt"/>
        <!--共有三个文件需升级-->
        <UpdateFile FileName = "VB40.rar"/>
        <UpdateFile FileName = "VB4-1.CAB"/>
    </UpdateFileList>
    <RestartApp>
        <ReStart Allow = "Yes"/>
        <!--允许重新启动应用程序-->
        <AppName Name = "TIMS.exe"/>
        <!--启动的应用程序名-->
    </RestartApp>
</AutoUpdater>
```

从以上 XML 文档中可以得知升级文档所在服务器端的地址、升级文档的更新日期、需要升级的文件列表，其中共有三个文件需升级：aa.txt、VB40.rar、VB4-1.CAB。以及是否允许重新启动应用程序和重新启动的应用程序名。

4.2 获取客户端应用程序及服务器端升级程序的最近一次更新日期：通过 GetTheLastUpdateTime（）函数来实现。

```
private string GetTheLastUpdateTime(string Dir)
{
    string LastUpdateTime = "";
    string AutoUpdaterFileName = Dir + @"\AutoUpdater.xml";
    if (!File.Exists(AutoUpdaterFileName))
        return LastUpdateTime;
    //打开 xml 文件
    FileStream myFile = new FileStream(AutoUpdaterFileName, FileMode.Open);
    //xml 文件阅读器
    XmlTextReader xml = new XmlTextReader(myFile);
```

```
    while (xml.Read())
    {
        if (xml.Name == "UpdateTime")
        {
            //获取升级文档的最后一次更新日期
            LastUpdateTime = xml.GetAttribute("Date");
            break;
        }
    }
    xml.Close();
    myFile.Close();
    return LastUpdateTime;
}
```

通过 XmlTextReader 打开 XML 文档，读取更新时间从而获取 Date 对应的值，即服务器端升级文件的最近一次更新时间：函数调用实现：

```
//获取客户端指定路径下的应用程序最近一次更新时间
string thePreUpdateDate = GetTheLastUpdateTime(Application.StartupPath);
//Application.StartupPath 指客户端应用程序所在的路径。

//获得从服务器端已下载文档的最近一次更新日期
string theLastsUpdateDate = GetTheLastUpdateTime(theFolder.FullName);
//theFolder.FullName 指在升级文档下载到客户机上的临时文件夹所在的路径。
```

4.3 比较日期

客户端应用程序最近一次更新日期与服务器端升级程序的最近一次更新日期进行比较。

```
//获得已下载文档的最近一次更新日期
string theLastsUpdateDate = GetTheLastUpdateTime(theFolder.FullName);
if (thePreUpdateDate != "")
```

```
{
    //如果客户端将升级的应用程序的更新日期大于服务器端升级的应用程序的更新日期
    if (Convert.ToDateTime(thePreUpdateDate) >= Convert.ToDateTime(theLastsUpdateDate))
    {
        MessageBox.Show("当前软件已经是最新的，无需更新！",
            "系统提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
        this.Close();
    }
}
this.labDownFile.Text = "下载更新文件";
this.labFileName.Refresh();
this.btnCancel.Enabled = true;
this.progressBar.Position = 0;
this.progressBarTotal.Position = 0;
this.progressBarTotal.Refresh();
this.progressBar.Refresh();

//通过动态数组获取下载文件的列表
ArrayList List = GetDownFileList(GetTheUpdateURL(), theFolder.FullName);
string[] urls = new string[List.Count];
List.CopyTo(urls, 0);
```

将客户端升级的应用程序的日期与服务器端下载的应用程序日期进行比较，如果前者大于后者，则不更新；如果前者小于后者，则通过动态数组获取下载文件的列表，开始下载文件。通过 BatchDownload（）函数来实现。升级程序检测旧的主程序是否活动，若活动则关闭旧的主程序；删除旧的主程序，拷贝临时文件夹中的文件到相应的位置；检查主程序的状态，若状态为活动的，则启动新的主程序。

```
private void BatchDownload(object data)
```

```
{
    this.Invoke(this.activeStateChanger, new object[] { true, false });
    try
    {
        DownloadInstructions instructions = (DownloadInstructions) data;
        //批量下载
        using (BatchDownloader bDL = new BatchDownloader())
        {
            bDL.CurrentProgressChanged += new DownloadProgressHandler(this.SingleProgressChanged);
            bDL.StateChanged += new DownloadProgressHandler(this.StateChanged);
            bDL.FileChanged += new DownloadProgressHandler(bDL_FileChanged);
            bDL.TotalProgressChanged += new DownloadProgressHandler(bDL_TotalProgressChanged);
            bDL.Download(instructions.URLs, instructions.Destination, (ManualResetEvent) this.cancelEvent);
        }
    }
    catch (Exception ex)
    {
        ShowErrorMessage(ex);
    }
    this.Invoke(this.activeStateChanger, new object[] { false, false });
    this.labFileName.Text = "";
    //更新程序
    if (this._Update)
    {
        //关闭原有的应用程序
        this.labDownFile.Text = "正在关闭程序....";
```

```
        System.Diagnostics.Process[] proc = System.Diagnostics.Process.GetProcessesByName("TIMS");
        //关闭原有应用程序的所有进程
        foreach (System.Diagnostics.Process pro in proc)
        {
            pro.Kill();
        }
        DirectoryInfo theFolder = new DirectoryInfo(Path.GetTempPath() + "JurassicUpdate");
        if (theFolder.Exists)
        {
            foreach (FileInfo theFile in theFolder.GetFiles())
            {
                //如果临时文件夹下存在与应用程序所在目录下的文件同名的文件，则删除应用程序目录下的文件
                if (File.Exists(Application.StartupPath + "\\" + Path.GetFileName(theFile.FullName)))
                    File.Delete(Application.StartupPath + "\\" + Path.GetFileName(theFile.FullName));
                //将临时文件夹的文件移到应用程序所在的目录下
                File.Move(theFile.FullName, Application.StartupPath + "\\" + Path.GetFileName(theFile.FullName));
            }
        }
        //启动安装程序
        this.labDownFile.Text = "正在启动程序....";
        System.Diagnostics.Process.Start(Application.StartupPath + "\\" + "TIMS.exe");
        this.Close();
    }
}
```

这段程序是实现在线升级的关键代码，步骤有点复杂：首先用 Invoke 方法同步调用状态改变进程，然后调用动态链接库中批量下载(BatchDownloader.cs)类启动批量下载，接着判断原有的主应用程序有没有关闭，如果没关闭,则用 Process.Kill()来关闭主程序。

接下来，判断临时文件夹下(即下载升级程序所在的目录)是否存在与应用程序所在目录下的文件同名的文件，如果存在同名文件,则删除应用程序目录下的文件，然后将临时文件夹的文件移到应用程序所在的目录下。最后重新启动主应用程序。这样更新就完成了。