《C#入门经典》

图书基本信息

书名：《C#入门经典》

13位ISBN编号：9787302053330

10位ISBN编号：7302053332

出版时间：2002-04-01

出版社：清华大学出版社

作者：Karli Watson

页数：873

《C#入门经典》

内容概要

C#是Microsoft专用在 .NET Framework平台上进行开发的一门新型编程语言。.NET Framework由托管代码执行的运时环境和几乎可以完成所有编程任务的众多类库组成。虽然很多语言都能够编写NET代码，但C#是惟一针对.NET Framework 而设计的语言,因此在今后几年内，C#将会成为编写.NET 应用程序的首选。

本书是您在学习编写C#程序时的必备向导,它逐步阐明了C# 和.NET 的关键概念.本书首先介绍C#语言和基础知识,然后深入探讨如何利用C#进行面向对象的编程,如何利用C#编写Windows 应用程序,以及如何用C#创建动态Web页和 Web服务。最后的两个案例分析提供了多个完整用的C#应用程序范例，同时展示了利用C#编写应用程序的方法。 本书适用于初学者，以及相对缺乏编程经验、但又想从不支持面向对象编程技术的语言转移.NET Framework的程序员。

《C#入门经典》

书籍目录

《C#入门经典》

精彩短评

1、不做码农很久了，唉
2、天天都在实验室读
3、经典,值得推荐
4、深入简出的一本书,适合初学者,但对于framework过于依赖,真想学好c语言的话,还是找本MFC看吧
5、真得看完我该成仙了
6、当初学C#1.1时看的，用来入门确实不错。
7、入门不错
8、我看的第一本C#书籍，但是翻译的跟屎一样
9、废话太多，干货太少。
10、不是一般的经典

《C#入门经典》

精彩书评

1、回到办公室一眼就看见這個C#入门经典，就怒火丛生，比狗屎还要烂的一本书，语句都不通顺的一本书，还他妈的非看它不可，封面上这个长毛胖子在C#高级编程的封面上居然也赫然在列。一看见这张脸我两只眼睛就打颤，连连向我哀告：求求你，我们只有兄弟俩，一个没了就少一半，两个没了就全玩完了，别让我们看这张藩属脸了……我也是没办法，书总有一不小心合上的时候，一合上，要是正面朝上的话，那不就……只好委屈了这双眼睛了。其实我也巴不得就把这破书烧了，自吹自擂的wrox出版社就用这种次等图书欺诈消费者，真是神人共愤，天诛地灭。此书的封底大言不惭：本书的对象本书是一本浅显易懂的C#入门手册，适用于初学者、相对缺乏编程经验的程序员……那么本书正适合您。其实这本书一点都不浅显易懂，倒不是涉及的编程技术概念非常高深，而是此书的叙述方式简直狼心狗肺惨绝人寰。如果是英文原版的话可能情况还能好点，多少句子应该是读得通的。此书对概念和技术解释极端的术语化，以技术解释技术，只在操作层面解释技术，从来不深层解释，以至于一开始的时候让我对一些基本的概念和技术理解起来都非常困难。这本书与其说是入门经典，倒不如说是一本技术参考手册。连篇累牍的平铺直叙，毫无表情的代码分析，号称PROGRAMMER TO PROGRAMMER的红皮书，听起来像是一对一的悉心辅导，其实是一堆枯燥透顶的抽象教条的罗列。就拿今天早上刚刚开始看的Windows程序部署来说，本人从来没有做过程序部署这种工作，程序部署包括些什么我都不知道，我只知道我得把我调试过的程序做成一个安装程序包，才能给别人用——结果我是问了老Y才确定原来17章部署Windows应用程序就是说怎么把程序做成一个安装包——凭书里面说的部署的含义，我压根就看不出来这就是我要的“ 部署“ 。汗一个，这些虚妄至极误人子弟的编程作家。加上康博之流的蹩脚翻译，此书更加语塞不通，臭不堪读，连基本的汉语阅读习惯都在这里被彻底颠覆。前言不搭后语，原本前面一句就说的模棱两可，后一句就王顾左右而言他了——英文作者在这方面也难辞其咎，一并声讨！

《C#入门经典》

章节试读

1、《C#入门经典》的笔记-第一章

C#实现在线软件自动升级程序

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。

一、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。

二、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。

三、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。

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。

以下为引用的内容：

<?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。

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 此文来自轻狂天下

通过GetTheLastUpdateTime（）函数来实现。

以下为引用的内容：

```
private string GetTheLastUpdateTime(string Dir)
{
string LastUpdateTime = "";
string AutoUpdaterFileName = Dir + @"\AutoUpdater.xml";
if(!File.Exists(AutoUpdaterFileName))
return LastUpdateTime;
FileStream myFile = new FileStream(AutoUpdaterFileName,FileMode.Open);
XmlTextReader xml = new XmlTextReader(myFile);
  while(xml.Read())
{
if(xml.Name == "UpdateTime")
{
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。

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)&gt;=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); bitsCN_com
}
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。

《C#入门经典》

版权说明