

CAutoupdater 通用

自动升级组件

用户手册

1、CAutoupdater 通用自动升级组件介绍

1.1、组件介绍

C/S 构的特点是能充分发挥客户端的处理能力，很多工作可以由客户端处理后再提交给服务器，对应的优点就是客户端响应速度快模式客户端以其强大的功能，丰富的表现力受到相当大部分用户的青睐，但是客户端部署，维护升级的成本却是非常的高的。

C/S 客户端需要安装专用的客户端软件及运行环境。首先涉及到安装的工作量，其次任何一台电脑出问题，如病毒、硬件损坏，都需要进行安装或维护。特别是有很多分部或专卖店的情况，不是工作量的问题，而是路程的问题。还有，系统软件升级时，每一台客户机需要重新安装，其维护和升级成本非常高。

CAutoupdater 组件就是为了解决 C/S 在维护升级方面的问题而存在的，本组件可以快速建立升级部署应用，完全控制整个升级部署过程，全功能的、快速、易于使用,设立发布你的客户端应用。

本组件以最简单的模式提供给客户端在线部署和自动升级功能,彻底解决升级部署方面的后顾之忧。组件中提供的接口以最简单的方式来对升级过程提供完全控制。CAutoupdater 组件为你的软件集成稳定/高效的自动升级功能，是本升级组件的宗旨。

CAutoupdater 组件四步完成升级工作：检查新版本、下载到临时目录、检查本地文件、拷贝到工作目录。

无需特被的服务器，传统的 Web 服务即可，如 IIS 等。服务器部署最新版的软件文件（文件列表保存在一个 XML 文件中），CAutoupdater 组件对比分析服务器和客户端的 XML 对比配置文件，不同就更新。

CAutoupdater 组件采用 C#语言编写，可应用于目前大多流行语言开发的软件之中，不依赖任何类库，稳定，高效。

2.2、CAutoupdater 组件升级原理及过程。

2.2.1、CAutoupdater 组件工作原理。

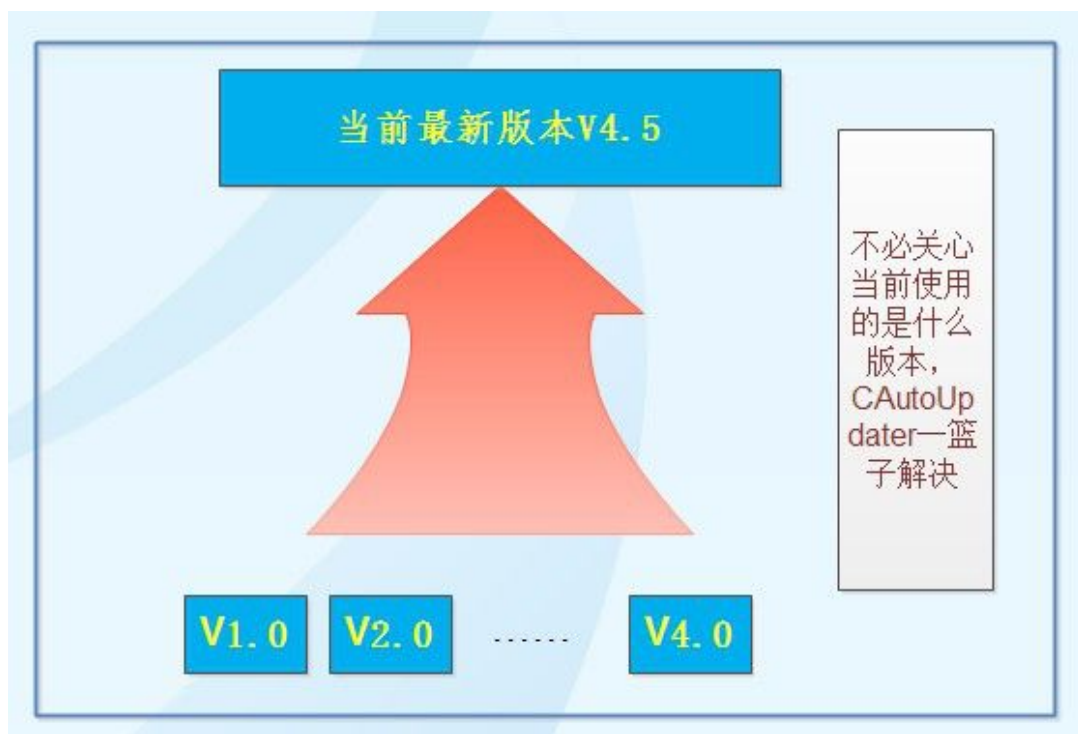


图 2.1.1.1 不必关心当前使用的版本

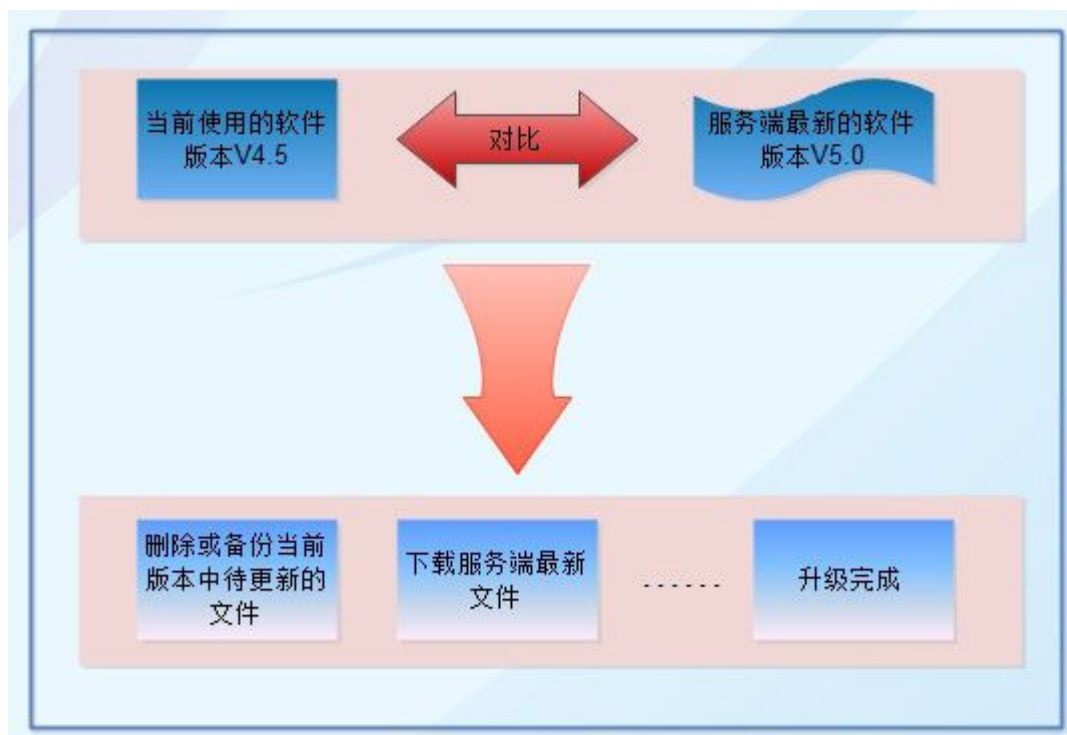


图 2.1.1.2 对比分析服务端与客户端使用的软件版本

2.2.2、服务端部署。

要使用 CAutoupdater 组件必须部署服务端，服务端的部署很简单，在这儿我们选择传统的 IIS 进行部署。假设我们升级的相关文件放在服务端的 D 盘 CAutoupdater 文件夹下，打开 IIS，新建一个网站，将其物理路径指向：D:\CAutoupdater，具体 IIS 的配置可参考相关文章，在这儿不再介绍。在这儿重点介绍服务端 [AutoupdateService.xml](#) 文件。

```
AutoupdateService.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <updateFiels>
3   <file path="RDIFramework.Utilities.dll"
4     url="http://localhost:8010/RDIFramework.Utilities.dll"
5     lastver="2.5.0.0"
6     size="642048"
7     md5="14B6A7A55BFC98348949E13F36297BF6"
8     needRestart="False" />
9 </updateFiels>
```

图 2.2.1 AutoupdateService.xml 文件

通过 [AutoupdateService.xml](#) 文件可以看到，目前我们新增了一个待更新的文件 `RDIFramework.Utilities.dll`，其版本号通过 `lastver` 配置项提供，文件大小通过 `size` 配置项提供，`md5` 通过 `md5` 提供，还有一个配置项 `needRestart`，是否需要重启应用程序，意思是说，更新本文件后，主程序是否需要重新启动。`url` 配置项就是待更新文件实际的服务端地址，客户端通过这个地址来下载这个文件。

客户端是如何判断当前是否需要更新呢？这儿主要是通过文件的 `md5`、`lastver`（最新的版本号）与 `size`（文件大小）来判断，三者只要取其一即可，在 `CAutoupdater` 组件中默认是取文件的 `md5` 进行判断，只要客户端对应文件与服务端对应文件的 `md5` 不一样，则更新。下面我们来看一下

如何自动生成服务端的配置文件（AutoupdateService.xml），如下图所示：

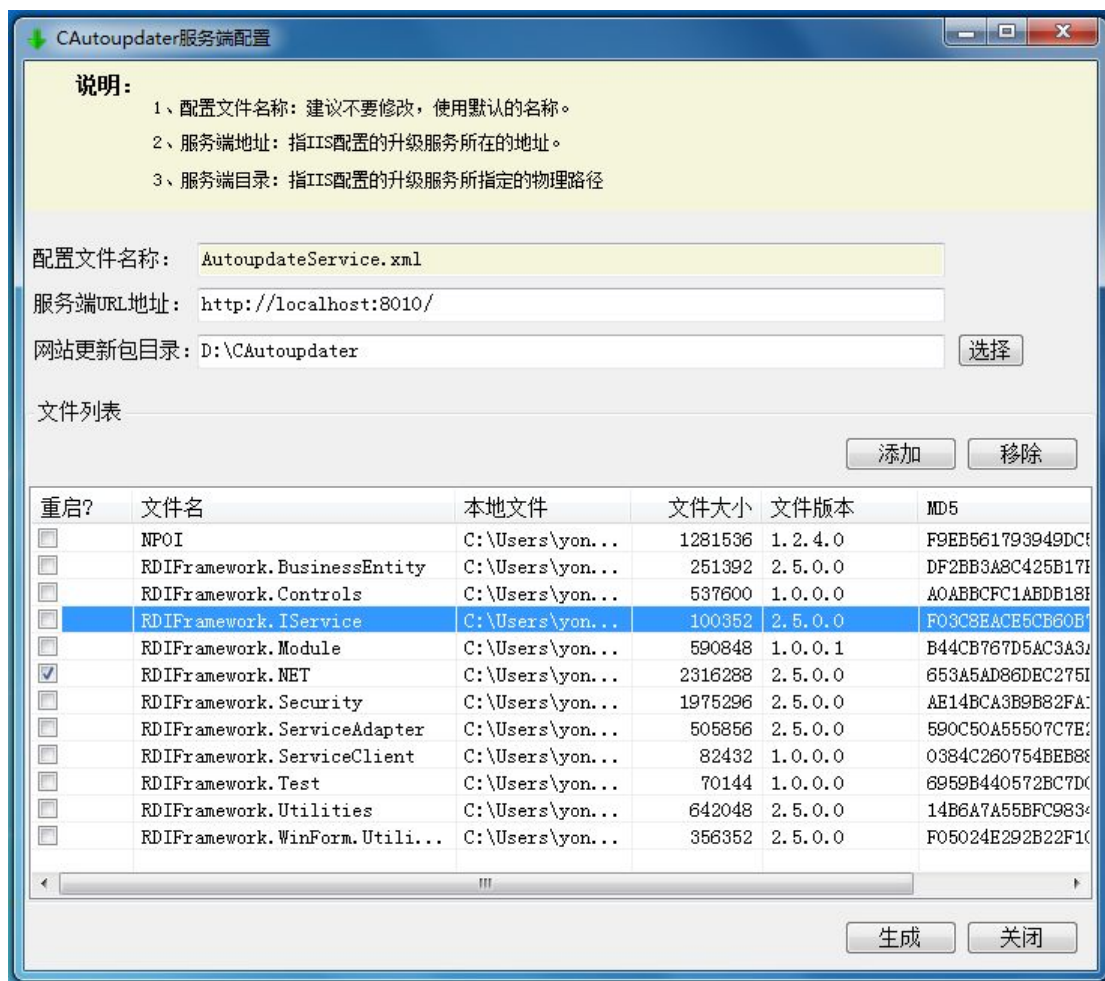


图 2.2.2.1 CAutoupdater 服务端配置

在图 2.2.2.1 中，配置文件的名称默认不允许修改，服务端地址就是我们配置 IIS 时的 URL 地址，在我的这个实例中是：<http://localhost:8010/>，服务端目录就是 IIS 对应 URL 地址所在的物理地址，在我的实例中是：D:\CAutoupdater，在界面下方的文件列表中，我们可以添加需要更新的文件，对于选择错误的文件，也可以对其选中后移除，可以看到文件列表的第一列为“重启？”，如果选中，则表示客户端升级后，需要重启客户端主程序，以完成更新，这需要根据实际的项目要求做相应的设置，没有定论。单击“生成”按钮，即可自动生成服务端的配置文件

AutoupdateService.xml，同时把文件列表中的文件拷贝到指定的“服务端目录”中去。这样，就完成了服务端的配置。

同时需要说明的是，同时要修改 CAutoUpdater 项目文件中的“Autoupdater.config”文件，此文件实例设置如下：

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

<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <Enabled>true</Enabled>

    <ServerUrl>http://localhost:8010/AutoupdateService.xml</ServerUrl>

</Config>
```

其中的：<http://localhost:8010/AutoupdateService.xml> 就是升级服务器上升级配置文件的 URL 地址。在客户端中也要包含这个文件，以让客户可以通过这个文件中配置的升级服务器的地址，找到待升级的文件。

2.3、客户端部署。

客户端的配置非常简单，只需引用“AutoUpdater.dll”文件，然后在需要的位置（一般是在程序启动时检查更新，也可让用户手动检查更新），我一般都是在程序启动时（在Program.cs文件的Main函数中）检查待更新的文件，参考代码为：

```
#region check and download new version program
bool bHasError = false;
IAutoUpdater autoUpdater = new AutoUpdater();
try
{
    autoUpdater.Update();
}
catch (WebException ex)
```

```
{
    LogHelper.WriteException(ex);
    MessageBoxHelper.ShowErrorMsg("连接自动升级服务器出错， 请检查网络连接或联系软件提供商");
}

bHasError = true;
}
catch (XmlException ex)
{
    LogHelper.WriteException(ex);
    bHasError = true;
    MessageBoxHelper.ShowErrorMsg("AutoUpdate Error:Download the upgrade file error");
}
catch (NotSupportedException ex)
{
    LogHelper.WriteException(ex);
    bHasError = true;
    MessageBoxHelper.ShowErrorMsg("AutoUpdate Error:Upgrade address configuration error");
}
catch (ArgumentException ex)
{
    LogHelper.WriteException(ex);
    bHasError = true;
    MessageBoxHelper.ShowErrorMsg("AutoUpdate Error:Download the upgrade file error");
}
catch (Exception ex)
{
    LogHelper.WriteException(ex);
    bHasError = true;
    MessageBoxHelper.ShowErrorMsg("AutoUpdate Error:An error occurred during the upgrade process");
}
finally
{
    if (bHasError == true)
    {
        try
        {
            autoUpdater.Rollback();
        }
        catch (Exception ex)
        {
            LogHelper.WriteException(ex);
        }
    }
}
```



```
}  
#endregion
```

客户端启动时，检查若有升级，就会弹出下面的窗口。



用户单击“跳过”按钮，可跳过升级，直接进入登录界面，单击“确定”或系统在用户没有任何操作的情况下默认等待10秒就自动进入升级程序，进行框架的升级，如下图所示：

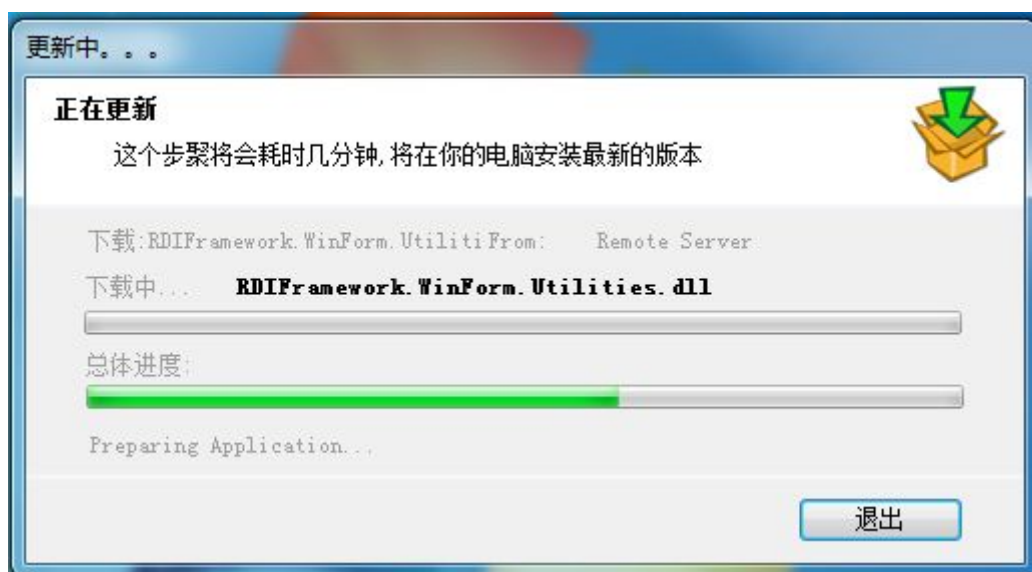


图 4.2.2 框架升级进度

框架升级完成，如果我们在升级的服务端设置了需要重新启动，则会弹出下面的窗口，单击“确定”立即进入系统，或等待 5 秒自动进入系统。
如下图所示：



RDIFramework.NET，基于.NET 的快速信息化系统开发、整合框架，给用户和开发者最佳的.NET 框架部署方案。

作者：[EricHu](#)

Email: 406590790@qq.com

QQ : 406590790

框架博客：

<http://www.cnblogs.com/huyong>

<http://blog.csdn.net/chinahuyong>

邮件交流：406590790@qq.com

购买地址：<http://yonghu86.taobao.com>

关于作者：

关于作者：高级工程师、信息系统项目管理师、DBA。专注于微软平台项目架构、管理和企业解决方案，多年项目开发与管理经验，曾多次组织并开发多个大型项目，在面向对象、面向服务以及数据库领域有一定的造诣。现主要从事基于 RDIFramework.NET 框架的技术开发、咨询工作，主要服务于金融、医疗卫生、铁路、电信、物流、物联网、制造、零售等行业。

如有问题或建议，请多多赐教！

