基于 asp.net 的文件上传和下载

李颖云,张克

(陕西职业技术学院 计算机科学系,陕西 西安 710100)

摘要:文件的上传和下载是网络中非常重要的两种应用,本文介绍了使用 asp.net 控件实现文件上传的基本方法。解决了当上传大文件时出现各种问题,并给出了几种解决方案和技巧。另外,下载文件用二进制读写的方式实现。

关键词:asp.net;上传;大文件;下载

中图分类号:TP311　　**文献标识码**:A　　**文章编号**:1009-3044(2007)06-11549-02

UpLoad and Download File Based on Asp.net

LI Ying-yun,ZHANG Ke

(Shanxi Institute of Vocational and Technical,xi' an 710100,China)

Abstract: This paper introduces several Methods of upload and download file on Asp.net, Analyses some questions existing in upload big file, provides a solution for these questions. And also explains a method of download file in binary access.

Key words: Asp.net;Upload;Big File;Download

1 引言

文件的上传和下载是网络中非常重要的两种应用。其实现方法主要有 FTP 方式和 HTTP 方式两种,FTP(File Transfer Protocol)是指文件传输协议,主要用来在网络上传输文件。这种方式虽说文件传输稳定、系统资源占用率低、对文件大小没有限制,但服务器的部署比较复杂,需要使用特定的软件来完成上传和下载,而且功能单一、权限设置复杂,一般用于专业的文件传输,对于普通用户来说实用性不高。

HTTP(Hyper Text Transfer Protocol)是指超文本传输协议,是互联网上应用最为广泛的一种网络传输协议。所有的 WWW 页面都必须遵守这个标准。设计 HTTP 最初的目的是为了提供一种发布和接收 HTML 页面的方法。相比于 FTP,HTTP 使用浏览器作为客户端软件,界面友好,操作简单,可以通过 WEB 表单机制实现文件的上传,利用超级链接和二进制读写的方法实现文件的下载。因为 HTTP 的文件上传和下载可以集成在 web 页面中,易用性、实用性较强,已经被广泛的应用于论坛、电子邮件、网络硬盘等实际应用中。本文主要介绍使用 asp.net 技术完成文件的上传和下载。

2 上传的实现

在 Asp.net 出现之前,完成 HTTP 方式的文件上传是一件很复杂的事情。Asp.net 作为微软的新一代网络程序开发平台,提供了专门用来进行文件上传的控件 HtmlInputFile,使用这个控件可以很方便的进行文件的上传。

该控件的语法为:<input id="控件名称" type="file" runat="server">

将上传文件保存在服务器指定文件夹下的语法为:

控件名称.PostedFile.SaveAs(服务器端物理路径)

此外还可以使用 ContentLength、ContentType 和 FileName 属性,分别获取文件大小、文件类型和在客户端的路径,从而对文件的类型、大小、文件名等进行特别的限定,保证系统的安全和稳定性。

具体上传代码如下(本文所列举代码都使用 C# 语言):

```
//因用到了 System.IO.Path 类的 GetFileName 方法,需导入 System.IO 名称空间
<%@ Import Namespace="System.IO" %>
<script language="C#" runat="server">
 private void Enter_Click(object sender,EventArgs e)
 {String filename,filepath ;
 //Path.GetFileName 方法可以从任何一个路径字符串中获取文件名称
 filename= Path.GetFileName(uploadfile.PostedFile.FileName);
 //filepath 是服务器端保存上传文件的物理路径
 filepath= Server.MapPath("upload") + filename;
 //保存上传文件
 uploadfile.PostedFile.SaveAs(filepath);
 //下面显示上传文件详细信息
 message.InnerHtml = "文件已经成功上载,详细信息如下";
 message.InnerHtml += "<br>保存路径:" + filepath ;
 message.InnerHtml += "<br>文件名称:" + filename;
message.InnerHtml += "<br>文件大小:" + uploadfile.PostedFile.ContentLength + "字节";
 message.InnerHtml += "<br>文件类型:" + uploadfile.PostedFile.ContentType;
 message.InnerHtml += "<br>客户端路径:" + uploadfile.PostedFile.FileName;
 message.InnerHtml += "<br><a href=//upload/" + filename + "//>察看文件</a>" ;}
</script>
<html>
<body>
 <h4 align="center">文件上传</h4>
 <form enctype="multipart/form-data" runat="server">
  选择文件:<input id="uploadfile" type="file" runat="server">
 <br><input type="button" id="enter" value="提交" OnServerClick="Enter_Click" runat="server">
 <p><span id="message" style="font: 9pt 宋体;" runat="server" />
 </form>
</body>
```

收稿日期:2007-01-29

作者简介:李颖云,女,陕西职业技术学院计算机科学系讲师、研究生;张克,男,陕西职业技术学院计算机科学系,讲师。

```
</html>
```

使用这种方式实现基本的文件上传的确是非常方便,但在实际项目应用中发现,在 asp.net 中文件上传的默认大小为 4MB。如果通过客户端直接上传超过 4MB 的文件,会出现 DNS 错误,而且因为是客户端错误, 使用 Try…Catch 机制在服务器端无法捕捉,严重的影响了程序的健壮性,增加了最终客户使用的难度。

对于这个问题,一般情况下,我们可以采用更改 WEB.Config 文件中特定选项的方法来解决,如下例:

Web.config 文件

```
<system.web>
<httpRuntime
MaxRequestLength="80690"
ExecutionTimeout="6000"
/></system.web>
```

其中 maxRequestLength 就是用来限制上传文件大小的,单位为 KB, 可以通过修改这一项的值来增加 asp.net 允许上传文件的最大值。另一项 ExecutionTimeout 用来设定执行请求的最大时间值,单位为秒。在这个例子中上传文件的最大值被重新设置成了 80MB,另外超时时间也设置成了 6000 秒,解决了 asp.net 中 4MB 文件上传的限制。

但是,这种方法也不是很完善,首先,MaxRequestLength 的值并不能无限的扩大, 因为在上传文件时,ASP.NET 会将客户端文件载入服务器内存后,再加以处理,所以如果上传文件超出了服务器可用内存,仍然会出错。经测试,在配置 512MB 内存的服务器上,最大可以上传 160MB 左右的文件,如果超过这个大小,就会发生错误;其次,在文件上传过程中无法给出传输进度的提示,当文件很大时,会使用户无法了解上传进度,甚至认为传输失败;另外,上传文件的大小超过 MaxRequestLength 的值后,还是会出现客户端错误。所以这种方法比较适合于进行几百 M(具体要根据服务器的内存大小来确定)左右的文件上传。

另一种解决的方案就利用隐含的 HttpWorkerRequest,用它的 GetPreloadedEntityBody 和 ReadEntityBody 方法从 IIS 为 ASP.NET 建立的 pipe 里分块读取数据,对于每块分块进行分析并存储为临时文件,最后再转存为指定的文件。

其中分块读取数据的基本代码如下:

```
IserviceProvider provider=(IServiceProvider)HttpContext.Current;
HttpWorkerRequest request1 =(HttpWorkerRequest)provider.GetService(typeof(HttpWorkerRequest));
// 为上传数据建立缓冲区
byte[] buffer;
// 返回 HTTP 请求正文已被读取的部分
buffer= request1.GetPreloadedEntityBody();
// 如果是附件上传
if(! request1.IsEntireEntityBodyIsPreloaded())
{// 分块大小
Int n=1024;
//定义临时缓冲区
byte[] tempBuff=new byte[n];
// 循环分块读取,直到所有数据读取结束
while(request1.ReadEntityBody(tempBuff,n)>0)
{//对读取到临时缓冲区中的数据进行分析并存储
……}}
```

利用这种方式,通过对分块数据的分析还可以获得上传的进度,从而实现上传进度条。这种方法实现效率较高,不受服务器内存大小的限制,经测试,在配制 512MB 内存的服务器上可稳定上传 1GB 的数据文件,但实现方法相对要复杂的多。

最后, 还有一种方案就是使用基于 asp.Net 的第三方组件如 AspnetUpLoad 等来实现。这种方法实现简单,无上传文件大小限制,稳定性、上传效率等方面都非常不错,而且界面友好,普遍都提供了上传进度指示等附加功能。但存在的问题是,优秀的组件一般都为商业软件,需要付费才能使用。

3 下载的实现

在 web 页面中下载的实现相对于上传要简单的多,一般页面使用的办法都是给要下载的文件建立一个超级链接,在客户端点击超级链接后,弹出下载对话框进行下载。这种方式对于使用 rar 或 zip 压缩的文件来说比较适合,但如果源文件为 doc、txt、bmp 等常用格式,而客户端又对这些格式的文件进行了关联,那么将不会弹出下载窗口,而是直接在当前位置打开了文件。这样对一些初级用户来说,会认为系统出现了问题,没有完成下载,影响系统的正常使用。

为了解决这个问题,可以用二进制读写的方法来实现,使用 FileStream 对象的 read() 方法读取文件到字节数组中,再用 Response. BinaryWrite()方法向客户端输出。

具体代码:

```
//FilePath 为要下载文件的具体路径
FileStream fs = new FileStream(FilePath, FileMode.Open);
//建立缓冲区
byte[] bytes = new byte[(int)fs.Length];
//读取文件到缓冲区
fs.Read(bytes, 0, bytes.Length);
fs.Close();
Response.Clear();
//向浏览器输出,首先指明 ContentType
Response.ContentType = "application/octet- stream";
//向输出流中添加文件名,并进行必要的编码
Response.AddHeader("Content- Disposition", "attachment; filename=" +HttpUtility.UrlEncode (fi.Name,System.Text.Encoding.GB2312).Replace("+"," "));
//输出二进制文件
Response.BinaryWrite(bytes);
Response.End();
```

使用这种方法,无论是何种类型的文件,在希望下载时都会弹出选择窗口让用户确认“下载”或者“打开”。

4 结束语

使用 asp.net 的控件可以比较容易的实现文件的上传和下载,但毕竟 HTTP 不是专门用来进行文件传输的, 所以还存在很多的不足。例如:服务器内存占用率相当高,特别是在上传、下载文件较大的时候。所以,如果经常性的要进行大型文件(例如超过 2G)的上传下载,用户群体又相对专业时,还是使用 FTP 更合适一些。但对于要求操作简单、界面友好且文件传输量不大的场合,上面介绍的方法还是具有很大的优势和实用性。

参考文献:

[1]尚俊杰.asp.net 程序设计[M].北方交通大学出版社, 2004.
[2]苏贵洋.asp.net 网络编程[M].电子工业出版社, 2005.
[3]运用 AspnetUpLoad 组件实现基于 Web 的文件上传[J].计算机时代, 2006(6).