多文件上传在Web应用中的实现方法研究

戚艳军[1,2],马光思[1]
(1.西安建筑科技大学 信控学院,陕西 西安 710055;
2.西北政法学院 网络中心,陕西 西安 710061)

摘 要:采用HTTP协议实现Web应用中的多文件上传有多种方式。文中针对其中的JavaBean和Struts框架两种方式,分析了从上传文件输入流中提取文件和筛选数据的方法,指出二者处理输入流时的差异。文中结合实例分别给出了它们的实现过程,并根据实际项目开发环境对二者的适用场合进行分析说明。

关键词:多文件上传;输入流; JavaBean; Struts
中图分类号:TP311.1 **文献标识码**:A **文章编号**:1005-3751(2006)04-0158-03

Implementation of Upload Multiple Files in Web Application

QI Yan-jun[1,2], MA Guang-si[1]
(1. School of Info. and Auto., Xi'an Univ. of Arch. & Tech., Xi'an 710055, China;
2. Network Center, Northwest University of Political Science and Law, Xi'an 710061, China)

Abstract: There are many ways to realize multiple files upload using HTTP protocol in Web application. Intruduced two methods, JavaBean and Struts framework, about upoading multiple files. Analyzed how to require file and data from data input stream, indicaded the difference of processing data input stream between them. With an example, the paper introduced process of upload files about two methods, and analyzed application occasion of two methods according to project and development environment.

Key words: upload multiple file; input stream; JavaBean; Struts

0 引 言

文件上传与下载是Web应用中的基本过程,研究相关的实现方法对应用开发有重要的实用价值。文件下载可以利用HTML提供的链接支持直接下载服务器上指定位置的文件,而文件上传则需要结合相应的动态网页技术,将文件上传到服务器指定位置。文献[1]讨论了利用JavaBean实现HTTP协议和FTP协议文件上传的两种方法;文献[2]已对Struts框架下单个文件上传进行了详细分析和讨论。文中在文献[1]和文献[2]的基础上,结合工程实践,给出了采用JSP技术和HTTP协议实现多文件上传的进一步研究和具体实现。

HTTP协议上传多个文件是将所有文件全部放进数据输入流中,各文件以分隔符区分,用相关的处理技术分析输入流,将文件逐一上传到服务器。数据输入流的具体格式参见文献[1]。采用JavaBean和Struts框架实现上传都需对数据输入流进行分解,二者对上传输入流的处理稍有差异[2]。

收稿日期:2005-07-27
基金项目:陕西省教育厅产业化培育项目(AJ05012)
作者简介:戚艳军(1974-),女,陕西西安人,助教,主要从事软件复用技术研究。

1 JavaBean实现多文件上传

在JSP中使用JavaBean实现Web应用的文件上传一般需要联合使用3个文件:HTML文件、JSP文件和Java类文件。其中,HTML文件提供界面,供用户选择上传文件;JSP负责调用JavaBean;JavaBean的Java类文件最后完成HTTP文件上传请求。

文献[1]已指出对多文件上传使用getDataSection(),getDataHeader()和save()方法,JavaBean从文件输入流中得到每一个上传文件的内容数据和每个文件的信息头。文中给出的方法不关心数据输入流中每一个文件的信息头,只需要判断文件分隔符和文件数据的起始位。在实现过程中,一旦分析到数据流中的文件信息头即略过不做,循环读取下一行信息。如果是文件分隔符,则跳出transferFileToServer()方法,表明已将一个文件上传到服务器,返回到uploadFile方法,读取输入流中的下一个文件。循环此过程,直到所有文件全部上传到服务器。具体实现过程如下:

Step1:首先初始化上传文件的目标路径、上传文件的大小(即文件的最大长度)及上传文件的后缀。

Step2:获取页面传递来的request请求,使用getInputStream()方法获得上传文件数据。循环读取每一个上传文件的绝对路径和文件后缀。

Step3:判断上传文件的合法性,即检查文件的后缀,看其是否在初始设置的后缀序列中。如果是,则读取文件数据,将该数据形成一个新的文件保存到服务器指定的位置;否则给出提示信息。

Step4:处理数据流,设置文件输出流,逐行读取输入流信息,将数据写入目标文件。

在 JavaBean 中用到的方法有文件上传处理命令和上传文件到服务器命令。

上传处理命令:public void uploadFile (HttpServletRequest request) 主要负责处理由表示层发送的 HTTP 文件上传请求,通过 getInputStream()方法获取文件上传输入流,将所有上传文件的数据全部放到该数据流中,判断上传文件后缀并调用 transferFileToServer 方法将文件上传到服务器指定的位置。实现代码如下:

```
public void uploadFile (HttpServletRequest request) throws IOException{
  //获取文件输入流
  sis = request.getInputStream();
  /*循环读取输入流,即分隔符后的信息,实现多个上传文件的信息读取*/
  while (rowInfo = sis.readLine(b, 0, b.length)) != -1){
    filePath = new String(b, 0, rowInfo);
  /*将读取的数据重新编码成一个新的字符串
  从数据输入流中读取文件信息头,从文件信息头中提取上传文件的绝对路径*/
  if(fileStart= filePath.indexOf("filename= \"")) != -1)
  {
    filePath = filePath.substring(fileStart + 10);
  //查找数据流中文件绝对路径起始位
  fileStart = filePath.indexOf("\"");
  //上传文件的绝对路径
  sourceFile = filePath.substring(0, fileStart);
  //取得文件后缀
  suffix=s.substring(fileStart + 1).toLowCase();
  /*判断文件后缀,看是否属于允许上传的文件类型,如果是,调用 transferFileToServer 方法,将文件上传到服务器;如不限制文件类型,可省略如下的 if 判断*/
  if(!(sourceFile.equals("")||(!(getsuffix.indexOf("."+suffix)>=0))))
  {//将文件上传到服务器
  transferFileToServer (sourceFile); }
  ++count; // count 为上传文件个数的计数器
  }}}
```

上传文件到服务器命令:private void transferFileToServer (String name)主要是将读入的文件上传到服务器。实现代码如下:

```
private void transferFileToServer(String name)
{//设置上传到服务器的文件名
  int start = name.lastIndexOf("\\");
  String fileName = name.substring(start+1);
  try{
  //上传第 i 个文件的文件名
  objectFileName = fileName;
  /*设置文件输出流,objectPath 为服务器上文件的放置路径*/
  FileOutputStream out = new FileOutput
  Stream(objectPath+ojectFileName);
  //初始化 a, index, newStr,这里略
  //逐行读取数据
 while ((a = sis.readLine(b, 0, b.length)) != -1){
  newStr = new String(b, 0, a);
 /*如果读取到数据流中的文件头 Content-Type:text/plain 信息,略过不做,继续读取下一行信息,即数据信息*/
 if((index=newStr.indexOf ("Content-Type:"))!= -1)
{break;}}
 //开始读取数据信息
 sis.readLine(b, 0, b.length);
 //循环读取输入流中文件的数据信息
 while ((a = sis.readLine(b, 0, b.length)) != -1)
 {
  newStr = new String(b, 0, a);
  /*如果是文件分隔符,则停止读数据流,跳转到下一个文件*/
  if (如果是文件分隔符){break;}
  out.write(b, 0, newStr);
 }
 out.close();
 }catch(IOExceptionioe){ioe.printStackTrace();}}
```

JSP 页面的实现:

在 HTML 页面加入表单元素,一次上传几个文件就添加几个文件表单元素,分别给表单元素命名,如:filename1,filename2 等。

```
<form action="uploadSubmit.jsp" enctype="MULTIPART/FORM-DATA" method="post">
 <table align="center">
 <tr><td>上传文件 1:</td><td>
 <input type="file" name="filename1"></td>
 </tr>
 <tr><td>上传文件 2:</td><td>
 <input type="file" name="filename2"></td>
 </tr>
 <tr><td>上传文件 3:</td><td>
 <input type="file" name="filename3"></td>
 </tr>
 ……//这里可据需要设置多个输入框
 </table>
 </form>
```

2　Struts 框架实现多文件上传

Struts 框架下实现多文件上传需借助于包 commons-fileupload.jar。由包中提供的接口 FormFile 负责获得客

户端向服务器发出的数据流,处理该数据流,从中分析出文件上传到服务器的相关参数和数据,将指定文件上传到服务器。包中相关的方法有 getFileName()(得到上传文件名),getContentType()(得到文件的内容类型),getInputStream(得到上传文件请求输入流),getFileData(得到上传文件的数据)等。它们联合起来对输入流进行预处理,不再需要另行编写具体代码分析数据流中的各种标识,判断每个上传文件数据的位置。即可以直接使用包中封装好的方法读取文件数据部分的数据流[3]。与 JavaBean 一样,采用 Struts 框架的表示层也可以为用户提供多个文件表单元素以供多个文件同时上传。

(1)FormBean 的实现。

FormBean 接收页面请求参数,使用 GetXxx 和 SetXxx 方法保存和获取表单中每个上传文件信息。其具体实现如下:

```
public class UploadForm extends ActionForm
{ /* 获取页面提交的文件上传信息,有多少个文件输入框,定义多少个变量 */
private FormFile filename1, filename2, filename3, filename4;
public FormFile getFilename1( )
{ return filename1; }
public void setFilename1(FormFile_filename1)
{ filename1 = _filename1; }
/* 以下各输入框的名称 filename2,filename3,filename4 的 getXxx 和 setXxx 方法和上面的一样,这里略 */
}
```

(2)ActionBean 的实现。

ActionBean 从 FormBean 中接收上传信息,得到多个文件的上传请求实例放到哈希表中,使用 Enumeration 容器依次获得各个上传文件,处理文件并上传到服务器[4,5]。其具体实现如下:

```
public class UploadAction extends Action
{
//类中的常规写法略,只介绍主要实现部分
//定义输入流
InputStream is = null;
//定义输出流
OutputStream os = null;
//定义字节缓冲
byte[] buffer = new byte[8192];
//定义目标路径
String objectPath="D:\ \";
//实例化 UploadForm
UploadForm theForm = (UploadForm) form;
java.util.Hashtable
files = theForm.getMultipartRequestHandler( ).getFileElements();
/* 将多个上传请求参数放到 Enumeration 容器中 */
java.util.Enumeration enumer=files.keys();
//如果容器有元素,循环取得上传文件
while(enumer.hasMoreElements())
{
String fileName= (String)enumer.nextElement();
FormFile file=(FormFile)files.get(fileName);
//取得上传文件的名称
String fileName1 = file.getFileName();
/* 为处理从容器中获得的上传文件信息,采用 getInputStream()方法得到输入流,将文件上传到服务器。这里注意,页面中虽然设置有多个文件的输入框,但是用户一次上传文件的个数可以不定,即允许有的输入框中信息为空,因此,在处理时应将为空的输入框略去不做,以免抛出异常 */
if(! fileName1.equals(""))
{
//文件在服务器上的绝对路径
objectName =objectPath + fileName1;
//取得文件输入流
is= file.getInputStream();
//定义文件输入流
os = new FileOutputStream(objectName);
try
{ //从输入流中读取数据,写入到输出流中
while ( (bytesRead = is.read(buffer, 0, 8192)) ! = -1)
{ os.write(buffer, 0, bytesRead); }
os.close();
is.close();
}catch(Exception e) { e.printStackTrace(); }
}}}
```

以上是 Struts 框架下 FormBean 和 ActionBean 的实现,表示层的实现与采用 JavaBean 的实现方式相同。

3 结束语

文中讨论了 JavaBean 和 Struts 框架下实现多文件上传的方法。从两种实现方式看出,采用 JavaBean 实现文件上传需要自行编写代码分析上传输入流,判断输入流中数据信息的位置,然后进行文件写入操作。Struts 框架封装了一些对上传文件处理的方法,不需要判断输入流中每一个文件内容的起始位,利用框架中提供的方法即可直接得到上传文件的数据。从两种实现方式可以看出,采用 JavaBean 读取输入流中的数据需要设置很多循环和条件等逻辑判断,实现时容易出错。Struts 框架已经实现了输入流中各文件数据的获取,使用方便,不易出错。但如果单为实现文件上传功能而动用 Struts 框架,则因需要较多的文件协调工作,增加系统额外开销就有些不值,读者可根据实际项目和应用工具,选用不同的实现方法。

参考文献:

[1] 王新芳,刘 杰. JavaBean 实现多个文件上传的两种方法[EB/OL]. http://www.phpx.com,2005-03-11.

(下转第 163 页)

务器(比如 WebSphere 或 Apache Tomcat)上。所有的调用消息都将被 SOAPRPC Servlet 捕获,它把这些消息路由到相应的网格服务。同时,可以把网格服务发布到统一描述、发现和集成(Universal Description, Discovery, and Integration, UDDI)注册中心或 Web 服务检查语言(Web Services Inspection Language, WSIL)文档。UDDI 的设计可以在其中发布和搜索商业伙伴的业务及他们的网格服务。UDDI 注册中心是一个存储这种信息以及网格服务位置的中心。有两种类型的 UDDI 注册中心:公共的和私有的。以应用程序开发者或服务提供者的身份把所有的网格服务发布到 IBM, Microsoft, HP 或 SAP 掌管的公共 UDDI 注册中心。如果想发布私有网格服务或机密网格服务,就应该使用私有 UDDI 注册中心。但用于测试目的或小规模集成时,将私人网格服务发布到 WSIL 文档却是最容易的,因为 WSIL 不需要 UDDI 注册中心就能够进行网格服务发现、部署和调用。这是因为 WSIL 提供了对已存在的服务描述文档的引用进行聚集的方法(这些文档已经被用许多种格式编辑过了)。然后这些检查文档在服务提供点处被提供,或者通过可以放置在内容媒体(比如 HTML)中的引用使其可用。如定位 WSIL 的 URL 的格式可能如下所示:http://www. myorg - wsd. com/inspection. wsil,而且,UDDI 注册中心和 WSIL 将被 WSIL wsiluddi 数据标记紧密地关联在一起。在 WSIL 中,引用指针被用来连接到发布在 UDDI 注册中心的业务或服务。

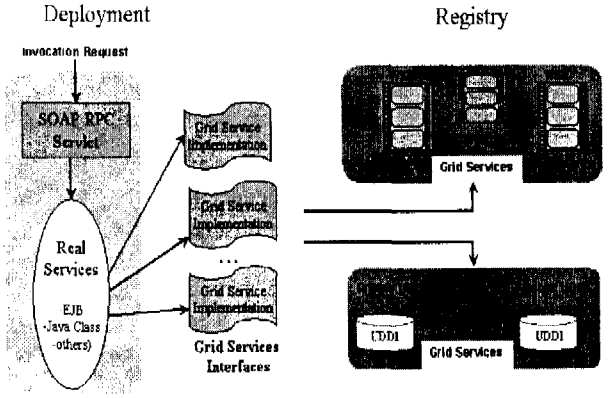

图 2　网格服务部署与发布示例的示意图

6　网格服务实例的创建与调用

用户应用程序在工厂上调用"创建网格服务(create Grid service)"请求,请求创建一个新的服务实例,并分配临时存储器以供这次计算之用。每个请求都涉及到用户和相关工厂之间的相互认证,接着是请求授权。每个请求都成功,结果是创建出了一个带某个初始生命周期的网格服务实例。同时还给这个新创建的网格服务实例提供了一个委托的代理凭证,该凭证允许这个实例代表用户执行更多的远程操作。由于网格服务是动态的并且是有状态的,所以每个网格服务实例都被分配了一个全局唯一的名称,即网格服务句柄(Grid Service Handle, GSH),它把一个特定的网格服务实例与其他所有的网格服务实例区分开来。例如"高级 UDDI 搜索服务(Advanced UDDI Search service)"使用它的代理凭证开始请求来自 UDDI 注册中心的数据,把中间结果放在本地存储器上。高级 UDDI 搜索服务还使用通知机制向用户应用程序提供对其状态的定期更新。同时,用户应用程序生成对它创建的网格服务实例的定期 keepalive 请求。

7　结束语

OGSA 属于正在不断发展的新技术、新领域,到目前为止 OGSA 的应用基本上还处于实验阶段,构造基于 OGSA 的网格系统的工具和规范仍有待进一步完善。随着与 Internet 联网的计算机设施社会化需求的增长,对 Web service、网格计算、OGSA 等现代分布式计算技术进行研究和创新具有十分重要的战略意义。

参考文献:

[1] Foster I, Kesselman C, Tuecke S, The Anatomy of the Grid: Enabling Scalable Virtual Organizations[J/OL]. International J. Supercomputer Applications, 2001, 15(3), www. globus. org/alliance/publications/papers/anatomy. pdf.

[2] Sotomayor B. The Globus Toolkit 3 Programmer's Tutorial [EB/OL]. http://www. casa - sotomayor. net/gt3 - tutorial - working/index. html, 2004 - 12 - 08.

[3] OGSA 规范[EB/OL]. http://www. gridforum. org/ ogsi - wg/ drafts/GS-Spec-draft03-, 2002 - 07 - 17.

[4] 王　毅,李智深,刘　鹏. Globus 开源网格基础平台简介[EB/OL]. http://www. net130. com, 2005 - 03 - 08.

[5] 邹德清,金　海. 纵览网格服务体系结构的演变(1),(2),(3)[EB/OL]. http://industry. ccidnet. com/pub/article/c28-a205603-p2. html, 2005 - 01 - 18.

[6] Foster I, Kesselman C, Jeffrey M, et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration[EB/OL]. www. globus. org/alliance/publications/papers/ogsa. pdf. , 2002 - 04 - 22.

[7] Foster I, Gannon D. The Open Grid Services Architecture Platform[EB/OL]. www. globus. org/alliance/publications/papers/ogsa. pdf. 2003 - 02 - 16.

(上接第 160 页)

[2] 戚艳军,马光思. 基于 Struts 框架实现 Web 应用中的文件上传[J]. 西安建筑科技大学学报(自然科学版), 2005(6): 270 - 273.

[3] Struts javadoc[EB/OL]. http://struts. apache. org/ api/index. html, 2005 - 04 - 15.

[4] Cavaness C, Friesen G, Keeton B. Java 完全探索[M]. 师夷工作室译. 北京:中国青年出版社, 2001. 689 - 691.

[5] 周　颖. 网络编程语言 JSP 实例教程[M]. 北京:电子工业出版社, 2002. 207 - 214.