# 19.文件上传

## 19.1.文件上传概述

**文件上传是什么**

在web开发中经常需要从客户端向服务器上传文件，如：上传照片、上传新闻图片、上传附件等等。这些都需要通过WEB开发中的文件上传技术实现。

**文件上传步骤**

实现web开发中的文件上传功能只需要两个步骤：

（1）提供一个带有文件上传项的表单。

（2）在servlet中读取处理上传的文件，保存到服务器中。

## 19.2.文件上传实现

### 19.2.1.在用户页面中添加上传输入项

提供一个带有文件上传项的表单

<input type="file" name="filex"/>

注意：

（1）文件上传的输入框必须有name属性才能被上传

（2）文件上传的表单必须是post提交

（3）文件上传的表单必须设置enctype=multipart/form-data

### 19.2.2.在服务器端编写文件上传程序

方式一：

可以通过javaee原生的api中request.getInputStream来获取请求中实体 内容的流，获取到流中的数据后，解析、处理非常麻烦

方式二：

为了简化这个处理过程，Apache 开源组织提供了一个用来处理表单文件上传的一个开源组件（Commons-fileupload ），让开发人员轻松实现web文件上传功能。

### 19.2.3.上传组件(Apache commons-fileupload)使用

首先需要下载并导入该组件相应的支撑jar包： Commons-fileupload和commons-io（课前资料中已经提供）

然后编程实现，步骤如下：

**（1）创建文件上传工厂类：DiskFileItemFactory对象，设置缓冲区大小和临时文件目录。**

DiskFileItemFactory factory = new DiskFileItemFactory(sizeThreshold, repository);

创建时，需要指定内存缓冲区大小和临时文件存放位置。

sizeThreshold - 指定内存缓冲区的大小

repository - 指定临时文件存放的位置

文件上传时，需要将请求的实体内容全部读取后，才能做处理，此时需要将实体内容缓冲起来。内存缓冲快，但是耗费内存；文件缓冲慢，但是可以存放大量数据。

所以此处提供了两个选项：如果数据大小小于sizeThreshold，则使用内存做缓冲，速度快。如果文件大小超过了该限制，则在repository指定的位置下创建临时文件来缓冲数据。

**（2）创建文件上传核心类：使用DiskFileItemFactory 对象创建ServletFileUpload对象，并设置上传文件的大小限制。**

ServletFileUpload fileUpload = new ServletFileUpload(factory);

常用方法有：

//判断当前表单是否是一个enctype为multipart/form-data类型的表单

boolean isMultipartContent(HttpServletRequest request)

//设置单个上传文件的最大值。

setFileSizeMax(long fileSizeMax)

//设置上传文件总量的最大值。

setSizeMax(long sizeMax)

//解决上传文件名乱码问题。

setHeaderEncoding(String encoding)

//实时监听文件上传状态。

setProgressListener(ProgressListener pListener)

//解析request对象，并把表单中的每一个输入项包装成一个fileItem 对象，并返回一个保存了所有FileItem的list集合。

List parseRequest(HttpServletRequest request)

1. **对list进行迭代，得到文件上传项 FileItem，并判断是否是上传文件。**

//判断当前FileItem是不是一个普通字段项，如果返回true表示这是一个普通字段项，返回false表示是一个文件上传项

boolean isFormField()

//如果是普通字段项

String getFieldName() //获取字段项的名称

String getString(String encode) //获取字段项的值，可以用encode属性进行设置。

//如果是文件上传项

String getName() //获取文件名（有些浏览器会携带客户端路径）

InputStream getInputStream() //获取文件内容的流

delete() //在关闭FileItem输入流后，删除临时文件

### 19.2.4.上传文件注意问题

**（1）ie上传文件时，文件名Bug处理**

**当ie上传文件时，文件名称会携带路径。**

if(fname.contains("\\")){

fname = fname.substring(fname.lastIndexOf("\\"));

}

**（2）文件上传保存位置问题**

文件上传保存的位置一定不能被外界直接访问，防止用户用浏览器访 问、下载资源或执行jsp恶意代码。

所以，文件要么保存在WEB-INF下保护起来，要么放在本地磁盘其他位 置，保证通过浏览器无法直接访问。

**（3）上传文件重名问题**

多个上传名称相同时，文件会发生覆盖。应该想办法让文件名尽量不要重复。

在文件名的前面拼接UUID来保证文件名绝对不会重复：

fname = UUID.randomUUID().toString()+"\_"+fname;

**（4）上传文件目录下文件过多**

一个文件夹下文件过多，会造成访问缓慢，甚至有可能无法访问。所以，应该想办法将这些文件分目录存储。

分目录存储的算法可以有很多：

1.按照上传时间进行目录分离 （周、月 ）

2.按照上传用户进行目录分离 ----- 为每个用户建立单独目录

3.按照固定数量进行目录分离 ------ 假设每个目录只能存放3000个文件 ，每当一个目录存满3000个文件后，创建一个新的目录

4.按照唯一文件名的hashcode 进行目录分离

计算文件名的hash - 散列

将hash值转换为16进制的字符串 - 共8位 值0-9 a-f

将这个hash字符串截取出每一位作为一级目录

最终将上文件放置到8级hash目录下，共有16^8个可能的文件夹分目录存放文件，从而保证一个文件夹文件不会过多

代码示例：

|  |
| --- |
| 在day17中的index.jsp中<body>添加如下代码：  <form action=*"*${ pageContext.request.contextPath }*/servlet/UploadServlet"* method=*"post"* enctype=*"multipart/form-data"*>  用户名 <input type=*"text"* name=*"username"*/>  密码<input type=*"text"* name=*"password"*/>  选择文件<input type=*"file"* name=*"fx"*/>  <input type=*"submit"*/>  </form> |
| 导入开发包：commons-fileupload-1.2.1.jar和commons-io-1.4.jar，放在WEB-INF下的lib文件夹中，并且在WEB-INF下创建临时文件夹temp和上传文件夹upload，然后创建servlet：UploadServlet，并添加如下代码：  **try** {  //创建文件上传工厂  DiskFileItemFactory factory = **new** DiskFileItemFactory(100, **new** File(**this**.getServletContext().getRealPath("WEB-INF/temp")));  //创建文件上传的核心类  ServletFileUpload fileUpload = **new** ServletFileUpload(factory);  //判断当前文件上传的表单是否满足enctype=multipart/form-data  **if**(!fileUpload.*isMultipartContent*(request)){  **throw** **new** RuntimeException("请使用正确的文件上传表单上传数据！");  }  //单个文件不能超过1M  fileUpload.setFileSizeMax(1024\*1024);  //总大小不超过10M  fileUpload.setSizeMax(1024\*1024\*10);  //解决上传文件名乱码问题  fileUpload.setHeaderEncoding("utf-8");  //解析request对象，并把表单中的每一个输入项包装成一个fileItem 对象，并返回一个保存了所有FileItem的list集合。  List<FileItem> items = fileUpload.parseRequest(request);  //遍历每个item分别做处理  **for** (FileItem item : items) {  **if**(item.isFormField()){//普通字段项  String name = item.getFieldName();  String value = item.getString("utf-8");  System.*out*.println(name+":"+value);  }**else**{//文件上传项  String fname = item.getName();  //处理ie文件名bug  **if**(fname.contains("\\")){  fname = fname.substring(fname.lastIndexOf("\\"+1));  }  //处理文件名 使其不会重复  fname = UUID.*randomUUID*().toString()+"\_"+fname;    //文件分目录处理  //----获取文件名的hash 转换为16进制字符串表现形式 由于文件名随机 所以 hash值也是散列的  String hash = Integer.*toHexString*(fname.hashCode());  //----如果hash不足8位则拼接足够8位的0  **while**(hash.length()<8){  hash += "0";  }  //----遍历hash值字符串的每一个字符作为一级目录拼接  String path = "WEB-INF/upload/";  **for** (**int** i = 0; i < hash.length(); i++) {  path += hash.charAt(i)+"/";  }  //----创建出该目录  **new** File(**this**.getServletContext().getRealPath(path)).mkdirs();    //得到输入流  InputStream in = item.getInputStream();  //得到输出流,路径就是上面 hash拼接出的路径+文件名  OutputStream out = **new** FileOutputStream(**this**.getServletContext().getRealPath(path+fname));    //输出数据到文件  **byte**[] data = **new** **byte**[1024];  **int** n = -1;  **while**((n=in.read(data)) != -1){  out.write(data, 0, n);  }  //关闭流  in.close();  out.close();  //删除缓存文件  item.delete();  }  }  } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** RuntimeException();  } |

### 19.2.5.文件上传进度监听

ServletFileUpload类提供了如下方法，监听文件上传时的进度信息：

public void setProgressListener(ProgressListener pListener)

设置监听器，文件上传程序会自动执行,监听器中 update方法

public void update(long pBytesRead, long pContentLength, int pItems)

在方法中可以获得 文件总大小、已经上传大小和 上传第几个元素

能否根据上面三个参数计算：剩余大小、传输速度、已用时间、剩余时间

(1) 已用时间 = 当前时间 - 开始时间

(2) 速度 = 已经上传大小/已用时间

(3) 剩余大小 = 总大小- 已经上传大小

(4) 剩余时间 = 剩余大小/速度

代码示例：

|  |
| --- |
| 在WebRoot下创建js文件夹，并将jquery包导入，然后在jsp页面中添加如下内容：；  在form表单中，添加div，用来显示进度条：  <div id=*"out\_div"*>  <div id=*"in\_div"*></div>  </div>  在<head>中，添加如下css属性，设置进度条的样式：  <style type=*"text/css"*>  *#out\_div*{  border: *1px solid gray*;  width:*150px*;  height:*20px*;  }  *#in\_div*{  width:*0%*;  height:*20px*;  background: *blue*;  }  </style>  在<head>中，引入jQuery类库，并为表单提交事件增加函数触发,增加定时器,每隔一段时间,通过ajax发送请求,访问服务器,获取最新的文件上传进度,展示到页面：  <!-- 引入jQuery类库 -->  <script type=*"text/javascript"* src=*"*${pageContext.request.contextPath }*/js/jquery-1.4.2.js"*></script>  <script type=*"text/javascript"*>  /\* 文档就绪事件 \*/  $(**function**(){  /\* 为form表单添加提交触发事件 \*/  $("form").submit(**function**(){  //定时器，每隔指定时间执行指定函数  window.setInterval(**function**(){  /\* 通过ajax发送请求，并将响应百分比填充到div中 \*/  $.get("${pageContext.request.contextPath }/servlet/UploadProgressServlet", **function**(data){  $("#in\_div").width(data);  });  }, 10);  });  });  </script> |
| 在UploadServlet中，获取List<FileItem> items之前添加如下代码：  //设置上传文件的监听  fileUpload.setProgressListener(**new** ProgressListener() {  //开始时间  Long begin = System.*currentTimeMillis*();  **public** **void** update(**long** pBytesRead, **long** pContentLength, **int** pItems) {  System.*out*.print("已经读了"+pBytesRead+"字节了");  System.*out*.print("总共"+pContentLength+"字节");  System.*out*.print("正在读第"+pItems+"个文件");  //剩余字节  **long** leftBytes = pContentLength - pBytesRead;  System.*out*.print("剩余"+leftBytes+"字节。。");  //结束时间  **long** now = System.*currentTimeMillis*();  //已经用时  **long** usetime = (now - begin)/1000 ;  System.*out*.print("已经用时" + usetime+"秒。。");  //上传速度  **long** speed = usetime == 0 ? 0 : pBytesRead / usetime / 1024;  System.*out*.print("上传速度" + speed+"KB/s。。");  //上传百分比  **double** per = pBytesRead \* 10000 / pContentLength / 100.0;  System.*out*.print("上传百分比" + per + "%。。");  //剩余时间  **long** lefttime = speed == 0 ? 0 : leftBytes /1024 / speed;  System.*out*.println("大致剩余时间"+lefttime+"秒");  request.getSession().setAttribute("progress", per);  }  }); |
| 创建servlet：UploadProgressServlet，并添加如下代码：  **double** per = (Double) request.getSession().getAttribute("progress");  response.getWriter().write(per+"%"); |

## 19.3.商品添加功能

### 19.3.1.搭建easymall后台页面

将课前资料中的后台页面backend拷贝到WebRoot下，并修改\_left.jsp中商品添加<div>：

|  |
| --- |
| <div><a href=*"*${ app }*/backend/manageAddProd.jsp"* target=*"rightFrame"*>> 商品添加</a></div> |

### 19.3.2.设计商品相关表

在easymall数据库中创建商品种类表和商品表

商品种类表

create table prod\_category(

id int primary key auto\_increment, -- 商品种类编号

cname varchar(255) -- 商品种类名称

);

商品表

create table prod(

id int primary key auto\_increment, -- 商品编号

name varchar(255), -- 商品名称

price double, -- 商品单价

cid int, -- 商品种类编号(外键)

pnum int, -- 商品库存

imgurl varchar(255), -- 图片地址

description varchar(255) -- 商品描述

);

### 19.3.3.后台开发

在bean包中创建类：ProdCategory，添加如下属性并添加get和set方法：

|  |
| --- |
| **private** **int** id;  **private** String cname; |

在bean包中创建类：ProdCategory，添加如下属性并添加get和set方法：

|  |
| --- |
| **private** **int** id;  **private** String name;  **private** **double** price;  **private** **int** cid;  **private** **int** pnum;  **private** String imgurl;  **private** String description;  **private** String cname; |

在后台页面manageAddProd.jsp中，<form>表单提交地址改为：

|  |
| --- |
| <form action=*"*${ app }*/servlet/ManageAddProdServlet"* method=*"POST"* enctype=*"multipart/form-data"*> |

在web包中创建包：cn.tedu.web.backend，在此包下创建servlet：ManageAddProdServlet，并在WEB-INF下创建文件夹：temp和upload，分别作为临时文件和商品图片的存放位置，然后导入文件上传相关的jar包：commons-fileupload-1.2.1.jar和commons-io-1.4.jar，在ManageAddProdServlet中添加如下代码：

|  |
| --- |
| **try** {  ServletContext sc = **this**.getServletContext();  Map<String,String> paramMap = **new** HashMap<String,String>();  //1.上传商品图片  //生成文件上传工厂  String temp = "WEB-INF/temp";  String upload = "WEB-INF/upload";  DiskFileItemFactory factory = **new** DiskFileItemFactory(100,**new** File(sc.getRealPath(temp)));    //创建文件上传核心类  ServletFileUpload fileUpload = **new** ServletFileUpload(factory);  //判断是否是正确的文件上传表单  **if**(!fileUpload.*isMultipartContent*(request)){  **throw** **new** MsgException("请使用正确的表单上传数据！");  }  //设定上传文件的大小限制  fileUpload.setFileSizeMax(1024\*1024);  fileUpload.setSizeMax(1024\*1024);  //设定编码解决中文文件名乱码  fileUpload.setHeaderEncoding(sc.getInitParameter("encode"));    //解析请求 获取FileItem集合  List<FileItem> items = fileUpload.parseRequest(request);  //处理item  **for** (FileItem item : items) {//普通字段项  **if**(item.isFormField()){  String name = item.getFieldName();  String value = item.getString(sc.getInitParameter("encode"));  System.*out*.println(name+":"+value);  paramMap.put(name, value);  }**else**{//文件上传项  String fname = item.getName();  //ie文件名bug  **if**(fname.contains("\\")){  fname = fname.substring(fname.lastIndexOf("\\")+1);  }  //文件名重复  fname = UUID.*randomUUID*().toString()+"\_"+fname;  //分目录存储,计算存储路径  String hash = Integer.*toHexString*(fname.hashCode());  **while**(hash.length()<8){  hash += "0";  }  **for** (**int** i = 0; i < hash.length(); i++) {  upload += "/"+hash.charAt(i);  }  **new** File(sc.getRealPath(upload)).mkdirs();  String savePath = sc.getRealPath(upload+"/"+fname);  paramMap.put("imgurl", upload+"/"+fname);    //上传文件  InputStream in = item.getInputStream();  OutputStream out = **new** FileOutputStream(savePath);  **byte**[] data = **new** **byte**[1024];  **int** len = -1;  **while**((len=in.read(data)) != -1){  out.write(data,0,len);  }  in.close();  out.close();  //删除临时文件  item.delete();  }  }    //2.将商品信息保存到数据库  ProdService service = BasicFactory.*getFactory*().getInstance(ProdService.**class**);  //创建prod对象封装数据  Prod prod = **new** Prod();  prod.setName(paramMap.get("name"));  prod.setPrice(Double.*parseDouble*(paramMap.get("price")));  prod.setCname(paramMap.get("cname"));  prod.setPnum(Integer.*parseInt*(paramMap.get("pnum")));  prod.setImgurl(paramMap.get("imgurl"));  prod.setDescription(paramMap.get("description"));    //调用service增加商品  service.addProd(prod);    //3.提示成功回到管理主页  response.getWriter().write("恭喜您，商品添加成功！");  response.setHeader("refresh", "1,url="+request.getContextPath()+"/backend/\_right.jsp");    } **catch** (MsgException e) {  response.getWriter().write(e.getMessage());  **return**;  }**catch** (Exception e) {  e.printStackTrace();  **throw** **new** RuntimeException();  } |

在service包中创建接口：ProdService，并添加如下代码：

|  |
| --- |
| /\*\*  \* 添加商品  \* **@param** prod 封装了商品信息的bean  \* **@throws** MsgException 消息异常  \*/  **void** addProd(Prod prod)**throws** MsgException; |

在service包中创建实现类：ProdServiceImpl，实现ProdService接口，并在配置文件：config.properties中添加如下配置信息：

|  |
| --- |
| ProdService=cn.tedu.service.ProdServiceImpl |

在ProdServiceImpl中添加如下代码：

|  |
| --- |
| **private** ProdDao dao = BasicFactory.*getFactory*().getInstance(ProdDao.**class**);    **public** **void** addProd(Prod prod) **throws** MsgException {  //1.根据商品种类名称查询商品种类表  String cname = prod.getCname();  ProdCategory findPc = dao.findProdCategoryByCname(cname);    //2.处理商品种类  **int** cid = 0;  **if**(findPc == **null**){  //如果找不到，则向商品种类表中加入新的商品种类  ProdCategory pc = **new** ProdCategory();  pc.setCname(cname);  dao.addProdCategory(pc);  //并获取新增加的商品的id 作为外键保存在商品信息里  ProdCategory findPc2 = dao.findProdCategoryByCname(cname);  cid = findPc2.getId();  }**else**{  //如果查找到，则使用该商品种类的id作为外键保存在商品信息里  cid = findPc.getId();  }    //3.将商品信息加入商品表  prod.setCid(cid);  dao.addProd(prod);  } |

在dao包中创建接口：ProdDao，并添加如下代码：

|  |
| --- |
| /\*\*  \* 根据商品种类名称查询商品种类信息  \* **@param** cname 商品种类名称  \* **@return** 查找到的商品种类信息，如果找不到就返回null  \*/  ProdCategory findProdCategoryByCname(String cname);  /\*\*  \* 向商品种类表中增加记录  \* **@param** pc 封装了商品种类信息的bean  \*/  **void** addProdCategory(ProdCategory pc);  /\*\*  \* 将商品信息加入数据库  \* **@param** prod 封装了商品信息的bean  \*/  **void** addProd(Prod prod); |

在dao包下创建实现类：ProdDaoImpl，实现ProdDao接口，并在配置文件：config.properties中添加如下配置信息：

|  |
| --- |
| ProdDao=cn.tedu.dao.ProdDaoImpl |

在ProdDaoImpl中添加如下代码：

|  |
| --- |
| **public** ProdCategory findProdCategoryByCname(String cname) {  Connection conn = **null**;  PreparedStatement ps = **null**;  ResultSet rs = **null**;  **try**{  conn = JDBCUtils.*getConnection*();  String sql = "select \* from prod\_category where cname = ?";  ps = conn.prepareStatement(sql);  ps.setString(1, cname);  rs = ps.executeQuery();  **if**(rs.next()){  ProdCategory pc = **new** ProdCategory();  pc.setId(rs.getInt("id"));  pc.setCname(rs.getString("cname"));  **return** pc;  }**else**{  **return** **null**;  }  }**catch** (Exception e) {  e.printStackTrace();  **throw** **new** RuntimeException(e);  } **finally** {  JDBCUtils.*close*(conn, ps, rs);  }  }  **public** **void** addProdCategory(ProdCategory pc) {  Connection conn = **null**;  PreparedStatement ps = **null**;  ResultSet rs = **null**;  **try**{  conn = JDBCUtils.*getConnection*();  String sql = "insert into prod\_category values (null,?)";  ps = conn.prepareStatement(sql);  ps.setString(1, pc.getCname());  ps.executeUpdate();  }**catch** (Exception e) {  **throw** **new** RuntimeException(e);  } **finally** {  JDBCUtils.*close*(conn, ps, rs);  }  }  **public** **void** addProd(Prod prod) {  Connection conn = **null**;  PreparedStatement ps = **null**;  ResultSet rs = **null**;  **try**{  conn = JDBCUtils.*getConnection*();  String sql = "insert into prod values (null,?,?,?,?,?,?)";  ps = conn.prepareStatement(sql);  ps.setString(1, prod.getName());  ps.setDouble(2, prod.getPrice());  ps.setInt(3, prod.getCid());  ps.setInt(4, prod.getPnum());  ps.setString(5, prod.getImgurl());  ps.setString(6, prod.getDescription());  ps.executeUpdate();  }**catch** (Exception e) {  **throw** **new** RuntimeException(e);  } **finally** {  JDBCUtils.*close*(conn, ps, rs);  }  } |