# 目录

1. Node.js 实现文件上传 **完成**

2.发送邮件 **进行中**

3.两个服务器实现共享数据

4.单独开设应用发布链接文件

5.云服务器绑定域名\windows server下nodejs监听80端口

6.Express的升级——locomotive

7.私有NPM仓库的搭建

8.手动负载均衡

9.发布网站的图标

10.nodejs流程控制问题

11.分页

12.小型web应用的新模式探究

# 一 Node.js 实现文件上传

概述：文件上传功能是社交网站、电子商务网站等必备的功能。文件上传功能可分为两部分来实现，一是客户端部分（浏览器），二是服务器部分。

客户端部分主要是实现选择文件、上传进度、上传状态的功能，服务器部分主要是实现接收文件、并将文件存储或者移动至相应的目录下的功能。

## 1 客户端部分

### 1.1简单的文件上传

如果仅仅可以上传文件而不需要了解上传进度、文件大小等信息，那么HTML的<input type=”file”/>标签就能够实现。

|  |
| --- |
| 代码清单 使用input标签实现简单的文件上传 |
| <form method=”post” action=”” enctype=” multipart/form-data” >  <input type=”file” />  <button type=”submit”>提交</button>  </form> |

其中，enctype是指定将数据会发到服务器时，浏览器使用的编码类型，其值有三种：

①application/x-www-form-urlencoded

这是标准的编码格式，如果在<form>标签里没有设置enctype属性，则默认的是此值。该属性会在发送到服务器之前，所有字符都会进行编码（空格转换为“+”，特殊符号转换为 ASCII HEX值）

②multipart/form-data

不对字符编码，窗体数据被编码为一条消息，页上的每个控件对应消息中的一部分。在使用文件上传功能时，必须设置该值。

③text/plain

窗体数据以纯文本的形式进行编码，其中不含任何控件或格式字符。（空格转换为“+”加号，但不对特殊符号编码）。

### 1.2带有上传进度功能的文件上传

文件上传进度功能的实现有两种方式，一是使用Ajax的XMLHTTPRequest，自己编写；二是使用一些JQuery上传插件，比如Plupload、jQuery File Upload、Uploadify等。

这里介绍一下JQuery插件Plupload的特点及使用，Plupload的下载地址为：http://www.plupload.com/download/

#### 1.2.1 Plupload的功能和特点

①拥有多种上传方式：HTML5、flash、silverlight以及传统的<input type=”file” />。Plupload会自动侦测当前的环境，选择最合适的上传方式，并且会优先使用HTML5的方式。所以你完全不用去操心当前的浏览器支持哪些上传方式，Plupload会自动为你选择最合适的方式；

②支持以拖拽的方式来选取要上传的文件；

③支持在前端压缩图片，即在图片文件还未上传之前就对它进行压缩；

④可以直接读取原生的文件数据，这样的好处就是例如可以在图片文件还未上传之前就能把它显示在页面上预览；

⑤支持把大文件切割成小片进行上传，因为有些浏览器对很大的文件比如几G的一些文件无法上传。

#### 1.2.2 Plupload的使用

Plupload的使用分五个步骤：

①引入Plupload的JS文件；

②实例化一个Plupload的对象，并设置参数；

③进行初始化；

④绑定事件，如文件添加事件、开始上传事件、上传进度事件、上传完成事件等；

⑤为绑定的事件添加动作。

下面是详细介绍：

一个简单的demo如下所示，该demo使用了Bootstrap。

|  |
| --- |
| 代码清单 plupload的使用 |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1">  <title>上传文件</title>  <meta charset="utf-8" />  <!--引入Bootstrap -->  <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">  <script src="http://libs.baidu.com/jquery/2.0.0/jquery.min.js"></script>  <script src="js/bootstrap.min.js"></script>  <!--引入plupload的源代码 -->  <script src="../js/plupload/plupload.full.min.js"></script>  <!—设置一下样式 -->  <style>  .wraper{ padding: 30px 0;}  .btn-wraper{ text-align: center;}  .btn-wraper input{ margin: 0 10px;}  #file-list{ width: 350px; margin: 20px auto;}  #file-list li{ margin-bottom: 10px;}  .file-name{ line-height: 30px;}  </style>  </head>  <body>  <div class="container">  <h2>这是一个简单的上传文件demo</h2>  </div>  <!—设置一个选择文件按钮、一个上传文件按钮和一个列表用来添加被选择文件的信息 -->  <div id="fileUpload">  <div class="wraper">  <div class="btn-wraper">  <input type="button" value="选择文件" id="browse" />  <input type="button" value="开始上传" id="upload-btn" />  </div>  <ul id="file-list">  </ul>  </div>  <script>  var uploader = new plupload.Uploader({ //实例化一个plupload上传对象  browse\_button : 'browse', //触发文件选择对话框的按钮，为那个元素的id  url : ' /saveFile/:param?/: param?', //要上传的地址  flash\_swf\_url : '../js/plupload/Moxie.swf', //swf文件，当需要使用swf方式进行上传时需要配置该参数  silverlight\_xap\_url : '../js/plupload/Moxie.xap' //silverlight文件，当需要使用silverlight方式进行上传时需要配置该参数  });  uploader.init(); //调用init()方法初始化  //绑定文件添加进队列事件  uploader.bind('FilesAdded',function(uploader,files){  for(var i = 0, len = files.length; i<len; i++){  var file\_name = files[i].name; //文件名  //构造html来更新UI  var html = '<li id="file-' + files[i].id +'">' +'<p class="file-name">' + file\_name + '</p><p class="progress"></p></li>';  $(html).appendTo('#file-list');  }  });  //绑定文件上传进度事件  uploader.bind('UploadProgress',function(uploader,file){  $('#file-'+file.id+' .progress').css('width',file.percent + '%'); //控制进度条  $('#file-'+file.id+' .progress-bar').text(file.percent + '%');  //控制进度条显示当前上传进度  });  //绑定文件上传完成事件  uploader.bind('FileUploaded',function(uploader,file,responseObject){  if(file.loaded==file.origSize){  $('#file-'+file.id+' .status').text('上传成功!');  }else{  $('#file-'+file.id+' .status').text('上传失败!');  }  });  //绑定文件开始上传事件  uploader.bind('UploadFile',function(uploader,file){  $('#file-'+file.id+' .status').text('正在上传!');  });  //上传按钮  $('#upload-btn').click(function(){  uploader.start(); //开始上传  });  </script>  </div>  </body>  </html> |

## 2 服务器部分

服务器部分主要是对客户端上传的文件进行接收，可以使用connect-busboy中间件来实现，其它也可以实现接收文件功能的中间件有：formidable、connect-form等。这里详细介绍connect-busboy的使用方法。

### 2.1 connect-busboy安装

npm install connect-busboy

Github 地址：https://github.com/mscdex/connect-busboy

### 2.2 Express中的路由设置

|  |
| --- |
| 代码清单 设置路由 |
| var express = require('express');  var router = express.Router();  Var upload=require(‘upload.js’);  router.post('/saveFile/:param?/: param?',upload.saveFile); |

### 2.3 模块代码

upload.js的代码如下：

|  |
| --- |
| 代码清单 模块代码 |
| var fs=require('fs');  var express=require('express');  var busboy = require('connect-busboy');  var bodyParser = require('body-parser');  exports.saveFile=function(req,res,next){  var fstream;  req.pipe(req.busboy);  req.busboy.on('file', function (fieldname, file, filename) {  console.log("Uploading: " + filename);  fstream = fs.createWriteStream('public/files/'+filename);  file.pipe(fstream);  fstream.on('close', function () {  res.send('<script>alert("video file successfully uploaded!")</script>');  res.end();  });  });  } |

如果想要实现根据文件名把文件分别放置到不同文件夹下的功能，例如，如果文件名是.flv，则把它放到”video”文件夹下，如果是其它的，则放到”courseware”文件夹下，可以修改上述代码如下：

|  |
| --- |
| 代码清单 模块代码 |
| exports.saveFile=function(req,res,next){  var path='./public/files/';  var fstream;  req.pipe(req.busboy);  req.busboy.on('file', function (fieldname, file, filename) {  var patten=/.+\.flv/;  if(patten.test(filename)){  var pathNew=path+'video/'+filename;  }else{  var pathNew=path+'courseware/'+ filename;  };  console.log("Uploading: " + filename);  fstream = fs.createWriteStream(pathNew);  file.pipe(fstream);  fstream.on('close', function () {  res.send('<script>alert("video file successfully uploaded!")</script>');  res.end();  });  });  } |

## 3.参考资源

（1）Plupload官网:http://www.plupload.com/

（2）Plupload 上传控件使用指南: http://www.faceye.net/search/114587.html

（3）前端上传组件Plupload使用指南: http://www.cnblogs.com/2050/p/3913184.html

# 二 Node.js发送邮件功能

发送邮件功能有很多用途：

1. 服务器出错时通知维护人员；

2. 用户注册时验证邮箱；

3. 给用户发送特定内容的邮件；

Node.js中发送邮件功能可以使用“nodemailer”模块实现。Github项目地址：https://github.com/andris9/Nodemailer。

## 1 模块介绍

Nodemailer的主要特点包括：

（1）支持Unicode编码

（2）支持Window系统环境

（3）支持HTML内容和普通文本内容

（4）支持附件(传送大附件)

（5）支持HTML内容中嵌入图片

（6）支持SSL/STARTTLS安全的邮件发送

（7）支持内置的transport方法和其他插件实现的transport方法

（8）支持自定义插件处理消息

（9）支持XOAUTH2登录验证

## 2 安装模块

npm install nodemailer

## 3 模块使用

### 3.1 基本使用

（1）新建一个js文件，命名为“sendMail.js”，其代码如下：

|  |
| --- |
| 代码清单 发送邮件模块 |
| var nodemailer = require('nodemailer');  // 开启SMTP连接池  var transporter = nodemailer.createTransport({  service: 'QQ', //邮箱类型  auth: {  user: '180xxxx59@qq.com', //邮箱地址  pass: 'xxxxxxxxxxxxxxxxxxx' //密码  }  });  var mailOptions = { //设置发送邮件选项  from: 'node server <1805377859@qq.com>', //设置发件人地址，'<>'中的地址应该与auth\_user中的地址一致  to: 'xxxxxx@xxx.com', //收件人地址  subject: ‘ xxxxxx’, // 邮件的主题  text: '', // 邮件的纯文本内容  html: '<b></b>' // 邮件的内容，带有HTML格式  };  exports.sendMail=function(data){ //参数data为一个对象，包含将要发送邮件的主题、内容及收件人  mailOptions.to=data.to;  mailOptions. subject =data. subject;  mailOptions.text=data.text;  mailOptions.html= data.html;  transporter.sendMail(mailOptions, function(error, info){  if(error){  return console.log(error);  }  console.log('邮件发送成功' + info.response +' 于'+new Date().toLocaleString() );  });  }; |

（2）为发送邮件的邮箱设置开启SMTP服务，腾讯QQ邮箱开启SMTP服务的过程如下：

|  |
| --- |
| ①进入QQ邮箱，点击“设置”按钮进入设置界面： |
|  |
| ②在邮箱设置中点击“账户”选项卡： |
|  |
| ③找到如下部分并把SMTP服务设置为开启： |
|  |
| ③新版QQ邮箱在第三方客户端登录时需要输入授权码以代替邮箱密码，验证密保手机后，可得到邮箱的授权码： |
|  |

（3）在其它模块中调用此方法：sendMail.sendMail(data);

### 3.2 高级使用

## 4 参考资源

（1）**Nodejs发邮件组件Nodemailer**

http://blog.fens.me/nodejs-email-nodemailer/

分页问题：

三种思路：

1.刷新整个页面。<li ><a href=’’></a></li>

2.使用ajax不刷新整个页面

3.从一开始从服务器上获取全部数据，然后使用JQuery的hidden等控制元素的可见与不可见。