在线实验，请到PC端体验

一、实验介绍

1.1 实验内容

课程主要介绍如何配合 Node.js 服务程序实现文件上传。课程将会从基础的 Node.js 实现开始简述其中的原理及重点，然后通过使用应用框架，第三方模块简化代码，简单实现文件上传功能。最后对文件上传功能进行一些扩展。

1.2 实验知识点

- node-formidable 的使用

1.3 实验环境

- Node.js 6.x

1.4 适合人群

本课程难度一般，属于初级级别课程，适合具有 Node.js 基础的用户，学习 Node.js 上传文件实战。

二、实验原理

2.1 数据上传

在 web 应用中，很多业务场景都需要实现一些数据上传，提交到服务端做判断、处理。比如常用来做信息判断和权限控制的 cookie 、session，用于查询的字符串等。这些简单、体积不大的数据都可以集中到 HTTP 的请求报文头中，但是仅仅依靠头部报文是无法存储大量的数据。

2.2 表单数据

一个简单的网页表单：

```
<form action="/upload" method="post">
    <label>username: </label>
    <input type="text" name="username" />
    <br />
    <label>password: </label>
    <input type="password" name="password" />
    <br />
    <input type="submit" name="submit" value="submit" />
</form>
```

Node.js 的 http 模块可以对报文的头部进行解析，然后出发 request 事件，并对请求中的内容进行处理、解析。

通过报头的 `Transfer-Encoding` 或 `Content-Length` 即可判断请求中是否带有内容：

```
var hasBody = function(req) {
    return 'transfer-encoding' in req.headers || 'content-length' in req.headers;
}
```

表单提交，请求报文头中的 `Content-Type` 字段值为 `application/x-www-form-urlenconded` ，Node.js 解析过后，我们便可以通过访问 `req.body` 获取表单提交的数据。

2.3 附件上传

通过以上可以了解到，可以通过判断请求报文头中的 Content-Type 字段值判断上传数据的格式，例如 JSON 和 XML 文件对应的值为 application/json 和 application/xml 。判断和解析原理也都基本相似。

此外大家还经常会遇到 file 类型的控件，此类型的表单需要指定表单属性 enctype 为 multipart/form-data ，如下：

```
<form action="/upload" method="post" enctype="multipart/form-data">
    <input type="file" name="file" />
    <br />
    <input type="submit" name="submit" value="submit" />
</form>
```

2.4 数据解析思路

- 必须使用 buffer 来进行 post 数据的解析
- 利用边界字符串（boundary）来分割各个字段数据
- 每个字段数据使用 \r\n 分割
- 利用上面的方法，我们确定了数据在 buffer 中的 start 和 end，利用 buffer.splice(start, end) 便可以进行文件写入了。

接着会使用比较流行的表单处理模块 node-formidable 结合应用框架简单实现文件上传实例。

三、实验步骤

3.1 框架搭建

1. 安装 Express 命令行工具： sudo npm install -g express-generator
2. 创建并初始化项目： express -ejs upload
3. 安装依赖模块： cd upload && npm install
4. 安装 bower 用于管理前端资源模块： sudo npm install -g bower
5. /upload/public 下执行命令： bower install bootstrap
6. 启动项目： npm start

打开文件 /upload/views/index.ejs 并输入如下代码：

```
<!DOCTYPE html>
<html>
  <head>
     <title><%= title %></title>
     <meta charset="utf-8">
     <!-- 引入 bootstrap 及相关文件 -->
     <link rel="stylesheet" type="text/css" href="/bower_components/bootstrap/dist/css/bootstrap.min.css" />
     <link rel="stylesheet" type="text/css" href="/stylesheets/style.css" />
     <script src="/bower_components/jquery/dist/jquery.min.js" type="text/javascript"></script>
     <script src="/js/script.js"></script>
  </head>

  <body>
       <div id="container" class="container">
          <!-- 添加信息提示 -->
          <% if (locals.success) { %>
              <div id="alt_sucess" class="alert alert-success">
                  <%- success %>
              </div>
          <% } %>

          <% if (locals.error) { %>
              <div id="alt_warning" class="alert alert-warning">
                  <%= error %>
              </div>
          <% } %>

          <!-- 表单 -->
          <form class="form-signin" role="form" method="post" enctype='multipart/form-data'>
              <div class="form-group">
                  <lebal for="fulAvatar">上传文件</lebal>
                  <input type="file" id="fulAvatar" name="fulAvatar" accept="image/*"/>
              </div>
              <button id="btnSub" class="btn btn-lg btn-primary" type="submit">上 传</button>
          </form>
       </div>
    </body>
</html>
```

这里就以上传图片为示例，所以在 **file** 类型的表单控件中，加入了 `accept="image/*"` 限制。

在提交表单时，一般需要前后端都各进行一次验证，以确保上传的安全性，所以新建 `/upload/public/js/script.js` 文件，敲入以下代码：

```
// 在 String 对象的 prototype 属性中添加我们自定义的 format() 方法
String.prototype.format = function (args) {
     var result = this;
     if (arguments.length > 0) {
       if (arguments.length == 1 && typeof (args) == "object") {
         for (var key in args) {
           if (args[key] != undefined) {
             var reg = new RegExp("({" + key + "})", "g");
             result = result.replace(reg, args[key]);
           }
         }
       }
       else {
         for (var i = 0; i < arguments.length; i++) {
           if (arguments[i] != undefined) {
             var reg = new RegExp("({)" + i + "(})", "g");
             result = result.replace(reg, arguments[i]);
           }
         }
       }
     }
     return result;
  }

$(function(){
    $('#btnSub').on('click',function(){
        var fulAvatarVal = $('#fulAvatar').val(),
        errorTip = '<div id="errorTip" class="alert alert-warning">{0}</div> ';

        $("#errorTip,#alt_warning").remove();

        if(fulAvatarVal.length == 0) {
            $("#container").prepend(errorTip.format('请选择要上传的文件'));
            return false;
        }

        // 验证上传文件后缀
        var extName = fulAvatarVal.substring(fulAvatarVal.lastIndexOf('.'),fulAvatarVal.length).toLowerCase();

        if(extName != '.png' && extName != '.jpg'){
            $("#container").prepend(errorTip.format('只支持png和jpg格式图片'));
            return false;
        }

        return true;
    })
});
```

前端上传验证实现后，我们需要编写后台程序，实现验证并上传图片。

3.2 使用 node-formidable 模块

安装 node-formidable，执行命令： npm install formidable --save

在 /upload/public 下新建文件夹 avatar 用于保存上传的图片

修改 /upload/routes/index.js 为：

```
// 导入必备的模块，全局设置一些公共变量
var express = require('express'),
    router = express.Router(),
    formidable = require('formidable'),
    fs = require('fs'),
    TITLE = '文件上传示例',
    AVATAR_UPLOAD_FOLDER = '/avatar/';

/* GET home page. */
router.get('/', function(req, res) {
    res.render('index', { title: TITLE });
});

router.post('/', function(req, res) {

    //创建上传表单
    var form = new formidable.IncomingForm();

    //设置编辑
    form.encoding = 'utf-8';

    //设置上传目录
    form.uploadDir = 'public' + AVATAR_UPLOAD_FOLDER;

    //保留后缀
    form.keepExtensions = true;

    //文件大小 2M
    form.maxFieldsSize = 2 * 1024 * 1024;

    // 上传文件的入口文件
    form.parse(req, function(err, fields, files) {

        if (err) {
            res.locals.error = err;
            res.render('index', { title: TITLE });
            return;
        }

        var extName = '';  //后缀名
        switch (files.fulAvatar.type) {
            case 'image/pjpeg':
                extName = 'jpg';
                break;
            case 'image/jpeg':
                extName = 'jpg';
                break;
            case 'image/png':
                extName = 'png';
                break;
            case 'image/x-png':
                extName = 'png';
                break;
        }

        if(extName.length == 0) {
            res.locals.error = '只支持png和jpg格式图片';
            res.render('index', { title: TITLE });
            return;
        }

        var avatarName = Math.random() + '.' + extName;
        var newPath = form.uploadDir + avatarName;
        fs.renameSync(files.fulAvatar.path, newPath);  //重命名
    });

    res.locals.success = '上传成功';
    res.render('index', { title: TITLE });
});

module.exports = router;
```

这样，简单的上传文件例子就完成了，执行 `npm start` 启动项目

上传的本地图片可以通过浏览器，浏览实验楼首页，下载一张课程图片即可。

选中并点击上传后，根据页面显示的提示信息即可判断是否上传成功。如果在 `/upload/public/avatar` 下有存在图片说明上床成功。

3.3 扩展

上传文件的逻辑功能已经实现，但还存在许多不足，例如：

- 刷新本页，会在此提交 post 请求；
- 没有图片预览；

通过使用 PRG（Post-Redirect-Get） 模式，将 post 提交后的返回结果重定向到新的一页，浏览器通过 get 那个页面来解决第一个问题。

修改 `index.js` 文件，保存上传图片的路径：

```
// ...
// 设置保存图片路径的数组
var IMGS_ARRAY = [];

router.post('/', function(req, res) {
    // here coding ...

    form.parse(req, function(err, fields, files) {
        // here coding ...


        var imgs = AVATAR_UPLOAD_FOLDER + avatarName;
        IMGS_ARRAY.push(imgs);
    })

    // ...
    // 注释掉这行
    // res.render('index', { title: TITLE });
    res.redirect('/details');
})

// 建立 detail 页面的 get 请求
router.get('/details', function(req, res) {
    res.render('details', {
        title: '图片列表',
        imgs: IMGS_ARRAY
    });
});
```

`/upload/views` 下新建页面 `detail.ejs`,代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <title><%= title %></title>
        <meta charset="utf-8">
        <link rel="stylesheet" type="text/css" href="/bower_components/bootstrap/dist/css/bootstrap.min.css" /
>
        <link rel="stylesheet" type="text/css" href="/stylesheets/style.css" />
        <script src="/bower_components/jquery/dist/jquery.min.js" type="text/javascript"></script>
    </head>
    <body>
        <div id="container" class="container">

            <a href="/">添加图片</a>
            <!-- 打印出所有上传的图片 -->
            <div>
            <% imgs.forEach(function(img) { %>
                <img src="<%= img %>" />
            <% }); %>
            </div>
        </div>
    </body>
</html>
```

四、总结

总的来说，项目很简单，只是讲解了 web 应用中经常遇到的一个业务场景，还有许多补充和完善的地方。大家可以学习其中的重要知识点，扩展出更强大的业务功能。

课程教师

Tryltry

共发布过**7**门课程

查看老师的所有课程 > (/teacher/9061)

前置课程

Node.js 教程 (/courses/44)

Node.js包教不包会 (/courses/493)

进阶课程

Node.js实现简单爬虫 (/courses/448)

Node.js实现私人笔记 (/courses/446)

Node.js 经典项目实战 (/courses/455)