快速开始

引入资源

本组件依赖于百度的上传组件`WebUploader`，并且后端代码不支持原生java文件上传，需要在`SpringMVC`或者`Springboot`的配合下完成带进度条的**断点续传**功能

```
  <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.c
ss"/>
    <link rel="stylesheet" type="text/css" href="/css/fileinput.cs
s"/>
    <link rel="stylesheet" type="text/css" href="/css/docs.css"/>

...

<script type="text/javascript" src="/js/jquery.js"></script>
<script type="text/javascript" src="/webuploader/webuploader.js">
</script>
<script type="text/javascript" src="/js/cyhup.js"></script>
<script>YOUR CODE</script>
</body>
```

前端部分

默认样式

需要对上传按钮添加id`filePicker`,以便本组件能够捕捉到上传按钮，并且添加点击事件，默认为选择文件后即上传

进度条

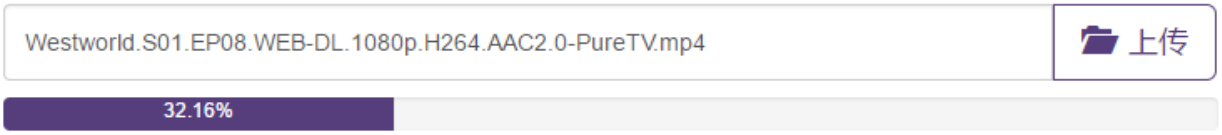
```
 <div class="col-md-12 margin">
                        <div class="input-group">
                            <div tabindex="-1" class="cyh-fileName
form-control">
                            </div>
                            <div class="input-group-btn">
                                <div id="filePicker" class="btn bt
n-lg btn-outline btn-file"><i
                                                class="glyphicon glyphicon-
folder-open"></i>  上传
                                            <input id="fileToUpload" clas
s="file" type="file" multiple=""
                                                   data-preview-file-typ
e="any"
                                                   data-upload-url="#" dat
a-preview-file-icon=""></div>
                            </div>
                        </div>
                    </div>
                    <div class="col-md-12 margin">
                        <div id="progress" class="display-none">
                            <div class="progress">
                                <div id="progress-bar" class="progr
ess-bar" role="progressbar" aria-valuenow="2"
                                     aria-valuemin="0" aria-valuema
x="100"
                                     style="background-color: #563d
7c; height:2em;min-width: 2em; width: 6%; display:inline;">
                                    4%
                                </div>
                            </div>
                        </div>
                    </div>
```

暂停\取消\下载按钮

需要要在前两个按钮的class中分别添加`cyh-pause` `cyh-cancel`，下载按钮可以自定义

```
<div class="col-md-12 margin" style="height:55px;">
                        <div class="col-md-2">
                            <button class="cyh-cancel btn btn-lg bt
n-outline " style="margin:0 auto;">取消</button>
                        </div>
                        <div class="col-md-3"></div>
                        <div class="col-md-2">
                            <form id="downloadForm" action="" metho
d="get">
                                <button type="submit" id="download"
class="btn btn-lg btn-outline"
                                        style="margin:0 auto; displ
ay:none;">
                                    下载
                                </button>
                            </form>
                        </div>
                        <div class="col-md-3"></div>
                        <div class="col-md-2">
                            <button class="cyh-pause btn btn-lg bt
n-outline" style="margin:0 auto; " data-bind="pause">暂停</button>
                        </div>
                    </div>
```

效果



js实例化

组件采用了js对象作为构造函数参数以实现实例化，为了使用本组件的基本功能，至少应该提供以下参数

```
 var BASE_URL = "http://localhost:8011";
    var UPLOAD_FUNC = '/upload';
    var DOWNLOAD_FUNC = '/download';
    var fileInfo = {};
    var md5 = "";
    var upload = new cyhup({
        baseUrl: BASE_URL,       // 项目根目录
        uploadFunc: UPLOAD_FUNC, //上传函数
        chunked: true,          //是否分块,决定是否有断点续传
        beforeSendFileSkip: function (res) {  //文件ajax提交验证存在
            $('#item1').find("p.state").text("文件重复，已跳过");
        },
        uploadProgress: function (file, percentage) {  //文件上传进度
            $("#progress-bar").text(Math.round(percentage * 10000)
/ 100 + '%')
                    .css("width", percentage * 100 + '%');
        },
        afterSendFile: function (file) {  //所有分块上传完毕后调用此函数
            $("#progress-bar").css("width", 100 + '%');
            $("#progress-bar").text("文件上传成功...");
            $('#item1').find("p.state").text("文件上传成功...");
        }
    });
```

后端部分

后端部分处理是否**断点续传**，`md5`验证，文件分块的**合并**，临时文件的**删除**等逻辑进行了封装，只需要一个函数搞定**上传前，上传中，上传后**的全部过程。函数会根据前端部分传入的参数进行文件上传各个阶段的判断，并进行对应的处理（参数全部使用pojo类`CyhParaters`进行了封装），并且返回值也封装到了`Feedback`中，如果想自定义返回值，可以继承`Feedback`类。确保使用了`SpringMVC` 或者`Springboot`.

```
@RequestMapping("/upload")
    @ResponseBody
    public Feedback up(
            @RequestParam(value = "fileToUpload", required = false)
                    MultipartFile file, CyhParaters cp
    ) throws IOException, NoSuchAlgorithmException,CyhUploadExcepti
on{
        String filePath = "H:\\upload\\";    // 文件上传后的路径
        CyhUpload upload = new CyhUpload();
        Feedback feedback =  upload.upload(filePath,file,cp);
        CyhFile fileInfo =  upload.getFileInfo();
        if(fileInfo != null){
            System.out.println(fileInfo);
        }
       return  feedback;
    }
```

> **注意：**在前端配置文件上传的大小限制之后，如果 `chunked` 设置为 `false` ，即不分块上传，那么在后端必须配置对文件大小的限制，否则限制不生效或出错， `Springboot` 中的配置如下

```
server.port=8011
spring.http.multipart.max-file-size=10000MB
spring.http.multipart.max-request-size=10000MB
```

前端参数配置

组件的前端部分提供了许多个性化的参数，以便实现不同的功能，需要将各项配置组合成 `javascript` 对象，传入组件的构造函数中，以实现实例化

基本类型参数

- **baseUrl** {String}[必选] 指定项目的根目录
- **uploadFunc**{String}[必选] 指定上传处理函数
- **chunked**{Boolean}[必选]是否分块，为true则分块，且分块大小参数才会被使用，断点续传必须指定本参数为true
- **fileSizeLimit**{int}[可选] [默认:200m]限制文件上传的总大小
- **fileSingleSizeLimit**{int}[可选][默认:50m]限制单个文件上传的大小
- **chunkSize**{int}[可选][默认:5m]指定上传的分块大小
- **accept**{Object}[可选]限制文件上传的类型，参数对象中，包含extensions参数，可赋

值为 ‘mp4,gif,jpg….’

复杂类型参数(均可选)

- **fileInfo**: {function}[参数：Info] 文件上传前可获取文件的相关信息
- **uploadError**: {function}上传出错时的处理函数
- **beforeSendFile**: {function}所有分块上传前调用此函数
- **md5Progress**: {function}[参数：percentage(转码进度)]文件MD5转码的过程
- **afterMd5**: {function}[参数：fileMd5(文件的Md5码)]Md5完成后，可获取文件的md5码
- **beforeSend**: {function}每个分块上传前调用此函数
- **onMerge**: {function}后台正在合并时调用函数
- **afterSendFile**: {function}所有分块上传完毕后调用此函数
- **beforeSendFileSkip**: {function}文件提交验证后，文件存在
- **fileQueued**: {function}文件被加入上传队列后调用
- **uploadProgress**: {function}[参数:file,percentage]文件上传过程中的进度信息
- **all**: {function}[参数:type]文件上传会触发的所有事件，包括`beforeFileQueued` `beforeFileQueued` `filesQueued` `fileDequeued` `reset` `startUpload` `stopUpload` `uploadStart` `uploadBeforeSend` `uploadAccept` `uploadProgress` `uploadError` `uploadSuccess` `uploadComplete` `error`,对于各个事件的解释详见 http://fex.baidu.com/webuploader/doc/index.html#WebUploader_Uploader_events

一个典型的样例

```
 var BASE_URL = "http://localhost:8011";
    var UPLOAD_FUNC = '/upload';
    var DOWNLOAD_FUNC = '/download';
    var fileInfo = {};
    var md5 = "";
    var upload = new cyhup({
        baseUrl: BASE_URL,                      // 项目根目录
        uploadFunc: UPLOAD_FUNC,
//上传函数
        fileSizeLimit: 2000 * 1024 * 1024,                      //文
件总大小
        fileSingleSizeLimit: 4 * 500 * 1024 * 1024,             //文
件单个大小
        chunked: false,                                         //是
否分块
        chunkSize: 5 * 1024 * 1024,                             //分
块大小
        //accept:{
        //    extensions: 'mp4,gif,jpg,jpeg,bmp,png'            //允
许上传的文件后缀名
        //},
        fileInfo: function (Info) {
            fileInfo = Info;
            $("#fileName").text(Info.fileName);
            $("#fileSize").text(Info.fileSize + " Bytes");
            $("#fileType").text(Info.contentType);
        },
        uploadError: function () {                              //上传
出错
            alert("上传失败!  请重试! ");
        },
        beforeSendFile: function (file) {                       //所
有分块上传前调用此函数
            $('#progress').fadeIn(1000);                        //进度条显
示
            $(".cyh-fileName").text(file.name);
        },
        md5Progress: function (percentage) {                    //文件MD
5转码的过程
            $('#item1').find("p.state").text("正在读取文件信息...");
        },
        afterMd5: function (fileMd5) {
//Md5完成后
            md5 = fileMd5;
            $('#item1').find("p.state").text("成功获取文件信息...");
        },
        beforeSend: function () {                               //每个分块
上传前调用此函数
```

```
        },
        onMerge: function () {                                  //正在
合并
            $("#progress-bar").text("合并中...");
            $('#item1').find("p.state").text("合并中...");
        },
        afterSendFile: function (file) {                        //所有
分块上传完毕后调用此函数
            $("#progress-bar").css("width", 100 + '%');
            $("#progress-bar").text("文件上传成功...");
            $('#item1').find("p.state").text("文件上传成功...");
            $("#progress").fadeOut(3000);          //动画 变透明
            $("#download").fadeIn(2000);
            $("#downloadForm").attr("action", BASE_URL + DOWNLOAD_F
UNC + "/" + fileInfo.extensionName + "/" + md5);
            // $("#progress-bar").css("width", 100 + '%');
        },
        beforeSendFileSkip: function (res) {            //文件ajax提
交验证存在
            $('#item1').find("p.state").text("文件重复，已跳过");
        },
        fileQueued: function (file) {                           //文件
加入队列

        },
        uploadProgress: function (file, percentage) {           //文
件上传进度
            $("#progress-bar").text(Math.round(percentage * 10000)
/ 100 + '%').css("width", percentage * 100 + '%');
        },

        all: function (type) {                                  //上传过程中
所有的触发类型
            console.log(type);
            if (type == "error") {
                alert("上传失败！");
            }
        }
    });
```