文件管理系统使用说明

# 1. 简介

文件管理系统 为公司提供统一的文件存储方案，可以通过api 或 界面 的方式上传文件。

## 项目的特点和功能

1. 界面管理：统计、预览、查询。
2. 权限控制：可控制每个文件的访问权限 和 界面的菜单权限。
3. 重复文件校验：校验文件md5值，确保相同的文件在数据库只存一份。
4. 图片转换：可以转换多个缩略图，方便在移动设备上预览。
5. 视频转换（m3u8切片）：支持多码率转换，提高播放效率，可以根据网络情况自动选取合适的视频流播放，可以记录用户的播放进度。
6. 可在线预览各种office文档：包括 microsoft office、libreoffice、wps office，原理是在服务端转换成pdf。
7. Zip、rar解压：可在线预览压缩包里面的文件。

## 文件存到mongodb vs 文件存到系统磁盘

该系统是使用mongodb的 GridFS 来存储文件的，针对把文件存到 mongodb 和 磁盘文件系统 的优缺点做一下说明。

使用GridFS 存储的优点：

1. 如果你的磁盘文件系统 对文件夹中的文件个数有限制，使用 Mongodb的 GridFS 来存储文件可以避免这个问题。
2. 如果你想对公司中 每个软件中的文件 做统一的管理，并且有界面可操作文件，该系统可以胜任。
3. 如果你想对每一个文件添加权限控制，不同的人查看不同的文件，使用该系统可以满足要求。
4. 随着文件的增多，查找和存储文件都成了挑战，而mongodb天生是分布式的，使用它可以非常容易的实现横向扩展

使用文件磁盘系统的优点：

1. 当你的项目对性能有着极致追求的时候，直接把文件存储到磁盘上会更有优势。

把文件存储到mongodb 的确有一些性能上的损失，但是却可以获得 更多其他的好处，而且经过测试：单台服务器上存储100GB 的文件，依然可以实时的获取文件。

## 系统工作流程图

1. 上传文件：通过 web界面 或者API 上传的文件 会暂时保存在一个共享目录（TempFile）里面。上传完成之后，其他系统上的转换程序（Convert）会获取该文件 校验md5、进行相应的转换之后，存入MongoDB中。
2. 下载文件：通过 web界面 或者 API 获取的文件 会首先从MongoDB 中查询，如果有文件，则立即返回；如果没有，则从TempFile中获取文件返回。

一般情况下如果转换程序（Convert）暂时挂掉了 或者 转换任务过重，文件还没来得及存入数据库，就会临时从TempFile中获取文件返回。



其中从API传送过来的文件来自不同的项目 ，Convert 可以部署在多台不同的机器上，协同转换。

# 2. 安装步骤

有一些软件是可以参考网上的资料的。不一定非得按照本文档。

## 2.1 Mongodb安装

1) 安装：到官网下载最新的Mongodb安装包，双击并按照默认的方式安装好，安装好之后在 “C:\Program Files\MongoDB\Server\4.0\bin\ ”目录下应该有一系列的安装文件。

2) 修改配置：打开mongodb的配置文件 mongod.cfg ，在storage 节点下的 dbPath节点中配置数据库的路径(D:\mongodata\)，在systemLog节点下的path节点中 配置日志文件保存位置(C:\Program Files\MongoDB\Server\4.0\log\mongodb.log)，其他默认。

3) 把mongodb安装为windows服务：在mongodb目录（C:\Program Files\MongoDB\Server\4.0\bin\ ）下按住shift键然后鼠标右键打开cmd命令行 输入命令：

mongod.exe --install --config "C:\Program Files\MongoDB\Server\4.0\bin\mongod.cfg"

服务安装好之后，把服务设置为自动启动。

## 2.2 LibreOffice安装

LibreOffice是开源软件，主要用作各种类型的office文件转pdf，方便在线预览。到官网下载安装好之后默认应该在目录 C:\Program Files\LibreOffice\ 下。

## 2.3 FileServiceWeb 部署

1) 请确保计算机已安装最新版的 .net framework和6.0以上的IIS版本。

2) 解压FileServiceWeb.zip 压缩包，在IIS中新建一个站点，把FileServiceWeb目录放入新建的站点中，应用程序池选择集成身份认证。

3) 打开浏览器预览站点，如果能看到登录界面，则安装完成。

4) 设置项目的目录（App\_Data\FileServiceTempFiles\）为共享目录。

5) 站点默认的账户名 为 local 密码为123，但是为了安全考虑，该账户只能在本机登录，要远程登录请自行添加其他的用户名和密码。

## 2.4 FileService.Converter 部署

1) 请确保计算机已安装最新版的 .net framework。

2) 解压FileService.Converter.zip压缩包，打开FileService.Converter.exe.config文件，

修改handlerId 节点的value值（转换服务器编号，每个服务器上的配置文件中这个value值必须不一致）。

修改mongodb节点的value值（mongodb数据库连接字符串，默认是 mongodb://localhost:27017/admin）。

修改libreOffice节点的value值（libreoffice可执行程序的路径，默认是 C:\Program Files\LibreOffice\program\soffice.exe）。

修改 sharedFolder 节点的value 值（2.3的第4步配置的共享目录的路径，\\{computerName or ip}\FileServiceTempFiles\）

修改 sharedUserName 节点的value值（登录共享目录的用户名，如果没有则填入””）

修改sharedUserPwd 节点的value 值（登录共享目录的密码，如果没有则填入 “”）

3) 双击 install-service.bat 系统自动把它安装为服务，服务名默认是 FileService.Converter 。把服务设置为自动启动并启动服务。

# 3. Web管理界面

Web管理界面默认的登录账户为 local 密码为 123，该账户只能在本机登录，登录之后请添加其他的账户和密码确保安全。

## 3.1 概况统计

主要统计各个系统的文件总数、文件走势、文件下载量、服务器、集群服务器、分片集群服务器的基本状态和数据存储情况。

## 3.2 处理器

主要展示转换服务器的基本信息（当前转换服务器还有多少转换任务、服务器状态，当服务器任务总数出现异常时，可以通过清空功能矫正总数）

## 3.3 转换任务

主要显示转换信息，如果一张图片或视频有多个转换任务，则在这里显示多条信息。

处理器：显示当前任务由那台服务器转换。

状态：当前任务的状态（wait running updated completed）。

百分比：转换任务的进度。

重做：如果转换任务发生了异常，找出异常并且修复异常之后点击 重做 ，这时处理次数会+1。

更新任务：通过点击文件编号，在下方显示任务详细信息，修改任务之后点击 更新，这时候任务的状态是 updated，需要点击重做来重新执行任务，重新执行任务完成之后处理次数会+1。

当文件有缓存的时候处理器编号上会有一个©符号，通过点击清除缓存文件来清除缓存，清除缓存只会清除 转换完成之后的文件 的缓存。

## 3.4 资源列表

资源列表有两种显示模式：列表模式和图模式，列表模式显示的信息量较多，而图模式可以更加方便的预览。

### 3.4.1 上传图片

在 input 框中选择图片文件，此时如果没有 转换缩略图 和 权限控制 需求，可以直接点击上传；如果有 转换缩略图需求 或者 权限控制需求 则 添加转换之后 再上传。

 **添加转换任务**：点击右侧的+开启

输出格式：转换之后的文件的格式，默认和上传格式一致。

标识：缩略图的标识，一般填写有意义的字符串。

模式：

1. 缩放：指定一个宽高，系统会把原图缩放到指定的宽高，图片可能会发生变形。
2. 剪切：从原图的 左上点 出发，指定需要剪切的起始点（top，left），指定剪切的宽度（width）高度（height），图片会剪切指定矩形区域。
3. 按宽度： 指定一个宽度（高度按原图比率来计算）来缩放，图片不会发生变形。
4. 按高度：指定一个高度（宽度按原图比率来计算）来缩放，图片不会发生变形。

当参数正确时，确定按钮会变成可点击状态。

 **添加权限**：点击右侧的+开启

其中公司、部门、用户需要提前设置，请参考文档的 用户管理 部分，

权限类型：

仅当前：仅当前选中的部门有权限查看文档。

当前及下属：当前选中的部门以及其下属有权限查看文档。

当前及上级：当前选中的部门以及其上级有权限查看文档。

添加好部门和用户之后点击 确定 添加已选好的权限。

设置好缩略图和权限之后点击上传即完成上传操作。

权限案例1：如果只设置了公司，没有部门和用户，则只要属于该公司的人员都可访问该文件。

权限案例2：如果设置了公司和用户，没有设置部门，则属于该公司的该用户可访问，其他人不能访问该文件。

权限案例3： 如果设置了公司和部门，没有设置人员，则属于该公司的该部门的都能访问，其他人不可访问该文件。

权限案例4：如果同时设置了公司、部门、用户，则属于该公司的该部门和该公司的该用户都有权限访问该文件，其他人不能访问。



### 3.4.2 上传视频

在 input 框中选择视频文件，一般情况下需要2种流的转换：一种pc端（原画），一种移动端（中等）方便以后的播放。

**添加转换任务：**点击右侧的+开启



输出格式：目前只能是http live stream 视频流（m3u8）,

质量：视频质量，质量越好 切片文件的大小越大，越适合 网速好的环境；适量越差切片文件的大小越小，越适合 网速慢的环境。

标识：转换视频的标识，一般填入有意义的字符串。

**添加权限：**请参考 上传图片 添加权限 节点。

### 3.4.3 上传附件

在 input 框中选择附件，添加权限之后直接提交。对于office类型的附件（microsoft office，libre office，wps office），系统会自动创建转换pdf的任务，转换完成之后，便可以点击相应的文件实现在线预览；对于zip、rar类型的附件，系统会自动创建解压的任务，解压完成之后，通过选择相应的文件，在下方的子文件列表中选择预览。

### 3.4.4 添加子文件

对于图片和视频文件，如果在上传的时候不确定转换任务，可以在上传之后在 缩略图列表 或者m3u8列表 中继续添加子文件，相关的参数和上传时候的参数一致。

### 3.4.5 修改权限&分享文件

对于自己上传的文件，选中文件之后，会出现 修改权限 和 分享文件 2个功能。



通过点击 访问权限 中的相应的公司信息，将在文本框中显示权限的详情，修改之后点击保存就修改完成。

默认情况下 文件管理系统 需要登录才能预览文件，而分享的文件会生成一个连接，访问这个连接可以 不登录系统 直接预览文件，可以按照需求设置访问密码和过期时间。

## 3.5 操作日志

### 3.5.1 日志

这里显示通过界面或者api所有的操作日志。

### 3.5.2 文件回收站

从资源列表中 删除的文件 会暂时回到文件回收站，从文件回收站中删除的文件会真正的从磁盘上消失，包括随文件一起的转换任务，缩略图，文件分享信息，视频截图信息。

## 3.6 配置管理

### 3.6.1 扩展名

系统安装的时候默认会添加一些扩展名。

在文件上传的时候 会通过这个表 检测该文件的行为，如果待上传的文件在配置的列表中的行为是block，则阻止该类型的文件上传（返回的文件id为24个0），可以通过配置block阻止某种类型的文件上传。

在文件预览的时候 会通过这个表 检测该文件的类型。

如果文件类型是image，则按照图片的规则预览。

如果文件类型是video，则按照视频的规则预览。

如果文件类型是office或者pdf，则按照pdf的规则预览。

如果文件类型是text，则按照文本的规则预览。

如果某种类型的文件无法预览，而该文件又是文本类型的，则需要在这里配置一下文件的扩展名和类型（text），配置好之后就可以预览了。

### 3.6.2 应用程序

在这里配置与API调用相关的一系列问题，默认有一个 FileServiceWeb，通过Web管理界面上传的文件全部属于该应用程序；也可以配置行为为block来阻止 某个应用程序 往 文件管理系统 上传文件，其中配置的 认证编码 是之后API要使用的authCode，配置的图片转换或者视频转换是指：当用户使用API中的Upload??DefaultConvert接口上传文件的时候不需要指定转换规则，默认使用配置好的转换规则。

FileClient fileClient = new FileClient(authCode, remoteUrl)

一个应用程序需要对应的在这里配置一条记录，这可以方便以后的文件统计。

## 3.7 用户管理

### 3.7.1 公司

 系统默认会生成一个叫 Company的公司，该公司没有子部门，需要手动添加。点击公司编号，下方会显示更新、添加子部门、删除栏。

如果公司原有的系统中已经有 部门名称 和 部门编码，则添加子部门的时候要和原有系统的 部门名称 和 部门编号 对应好，这有助于以后通过API添加文件的时候设置权限。

通过点击部门详情里面的子部门，可以对点击的部门继续添加子部门，子部门最多为15层。

可以通过拖动部门详情里面子部门的位置，来调整阻止结构，拖动完成之后记得点击保存。

### 3.7.2 用户

用户是通过用户名作为唯一标识的，如果尝试添加相同的用户名会报错，维护好用户对以后权限操作也有好处。

# 4. API(C#)

调用之前请添加对 FileService.Client.dll 的引用（FileService.Client.dll

一般在附带的压缩包中，另外在nuget上面也有一个，名称为： WangXYang.FileService.Client）

在随后的调用中：

authCode 是指在 配置管理 -> 应用程序 中添加的记录的 认证编码

remoteUrl 是指FileServiceWeb 部署好之后的url地址。

所有的接口调用成功后的返回值中code为0，Message为 success。

## 4.1 上传、下载图片

FileClient fileClient = new FileClient(authCode, remoteUrl);

### 4.1.1 上传图片

方式一：

var result = fileClient.UploadImage(fileName, fileStream);

int code = result.Code; //返回码，0代表成功，其他代表失败。

IEnumerable<ImageFileResult> res = result.Result; //返回的文件内容,因为上传的是单张图片，res列表的长度为1，

方式二：使用系统设置好的转换规则来上传文件

var result = fileClient.UploadImageDefaultConvert(fileName, fileStream);

int code = result.Code; //返回码，0代表成功，其他代表失败。

IEnumerable<ImageFileResult> res = result.Result; //返回的文件内容,因为上传

//res.First().FileId:返回的文件id（以后获取文件的依据）。

//res.First().FileName:返回的文件名。

//res.First().FileSize:返回文件的大小，字节。

//res.First().Thumbnail:当前文件的缩略图列表，因为没有添加转换任务，该列表为空。

### 4.1.2 上传图片的其他重载方法

ImageConvert：转换缩略图任务描述类。

ImageConvert.Flag：字符串，缩略图的标识，请填写有意义的字符串（mobile，pad，small）。

ImageConvert.Format：缩略图的格式。

ImageConvert.Model：转换缩略图的模式，

ImageConvert.X：仅仅在Cut模式下有效，指从原图左上角算起 水平轴（X轴）的坐标。

ImageConvert.Y：仅仅在Cut模式下有效，指从原图左上角算起 垂直轴（Y轴）的坐标。

ImageConvert.Width：缩略图的宽度。

ImageConvert.Height：缩略图的高度。

UserData：用户信息，设置文件权限必传该选项。

UserData.UserName：用户名称，文件的上传者，文件的拥有者。

UserData.UserAgent：用户浏览器信息，Request.UserAgent。

UserData.UserIp：用户ip信息，Request.UserHostAddress。

UserAccess：文件的访问权限信息，设置权限的前提是设置了 UserData 信息。

UserAccess.Company：文件的可访问公司代码（companyCode）。

UserAccess.DepartmentCodes：文件的可访问部门代码列表。

UserAccess.AccessUsers：文件的可访问用户列表（用户名）。

### 4.1.3 下载源图片

var file = fileClient.DownloadFile(id) //id为上传完成之后返回的源文件id

// file.FileName:文件名。

// file.ContentType:文件的contentType。

// file.ContentLength:文件的contentLength。

// file.FileStream:图片的文件流对象

//如果是mvc项目，则直接在Action中返回文件 return File(file.FileStream, file.ContentType, file.FileName);

//如果是aspx页面，则用该方法下载文件：

//file.FileStream.CopyTo(Response.OutputStream);

//Response.ContentType = file.ContentType;

//Response.AppendHeader("Content-Disposition", "attachment;filename=" + file.FileName);

//Context.Response.AddHeader("Content-Length",file.ContentLength.ToString());

//Response.Flush();

### 4.1.4 下载缩略图

var result = fileClient.DownloadThumbnail(id); //id为上传完成之后返回的缩略图列表中缩略图的id

//返回值已在上面介绍过。

//有的时候源图片刚上传完成，立即获取缩略图，这个时候有可能缩略图在服务端还没有转换完成，为了查询转换进度，使用一个接口来获取进度，当进度到100或者state状态是completed 时再去获取缩略图。

var state = fileClient.GetFileState(id); //id是指源图片id

//state.Code 返回码，0代表成功，其他代表失败。

//state.Result.FileId 源图片id。

//state.Result.FileName 源图片名称。

//state.Result.StateList 转换状态列表。

//state.Result.StateList[i].FileId 缩略图文件id。

//state.Result.StateList[i].Server那台服务器正在处理该任务。

//state.Result.StateList[i].TotalCount 服务器还有多少任务在等待中。

//state.Result.StateList[i].State 任务的状态 wait processing completed

//state.Result.StateList[i].Percent 任务的转换进度 1-100。

### 4.1.5 通过源文件id下载缩略图

var result = fileClient.DownloadThumbnailFromSourceId(id); //id为上传完成之后返回的源文件id。

//返回值已在上面介绍过。

//值得注意的是，该方法默认返回第一个缩略图（如果有多张缩略图的话），如果没有缩略图，则返回原图片。

### 4.1.6 通过源文件id和flag下载缩略图

var result = fileClient.DownloadThumbnailFromSourceIdFlag(id,flag); //id为上传完成之后返回的源文件id,flag为用户设置的flag。

//返回值已在上面介绍过。

//如果没有缩略图，则返回原图片。

## 4.2 上传、下载视频

FileClient fileClient = new FileClient(authCode, remoteUrl);

### 4.2.1 上传视频

方式一：

var result = fileClient.UploadVideo(fileName, fileStream);

int code = result.Code; //返回码，0代表成功，其他代表失败。

IEnumerable<VideoFileResult> res = result.Result; //返回的文件内容,因为上传的是单个视频，res列表的长度为1。

方式二：使用系统设置好的转换规则来上传文件

var result = fileClient.UploadVideoDefaultConvert(fileName, fileStream);

int code = result.Code; //返回码，0代表成功，其他代表失败。

IEnumerable<VideoFileResult> res = result.Result; //返回的文件内容,因为上传的是单个视频，res列表的长度为1。

//res.First().FileId:返回的文件id（以后获取源文件的依据）。

//res.First().FileName:返回的文件名。

//res.First().FileSize:返回文件的大小，字节。

//res.First().Videos:当前视频的m3u8转换视频列表，因为没有添加转换任务，该列表为空。

//注意：上传的视频可以是任意格式，如果不添加转换任务，你只能下载源文件，无法播放。因为H5播放器只支持m3u8播放器

### 4.2.2 上传视频的其他重载方法

VideoConvert：视频转换描述类。

VideoConvert.Flag：转换视频的标识，请填写有意义的字符串（mobile，pad…）。

VideoConvert.Format： 转换视频的格式，目前固定填 VideoFormat.M3u8。

VideoConvert.Quality： 转换视频的质量，目前有4种（Original：原样转换，Lower：稍低转换，Medium：中等转换，Bad：差转换，越差的转换视频切片越小，越适合网络差的环境）。

UserData：用户信息，设置文件权限必传该选项。

UserData.UserName：用户名称，文件的上传者，文件的拥有者。

UserData.UserAgent：用户浏览器信息，Request.UserAgent。

UserData.UserIp：用户ip信息，Request.UserHostAddress。

UserAccess：文件的访问权限信息，设置权限的前提是设置了 UserData 信息。

UserAccess.Company：文件的可访问公司代码（companyCode）。

UserAccess.DepartmentCodes：文件的可访问部门代码列表。

UserAccess.AccessUsers：文件的可访问用户列表（用户名）。

### 4.2.3 获取视频转换进度

为了更好的用户体验，允许用户通过api获取当前文件的转换进度。

var state = fileClient.GetFileState(id); //id是指源文件id

//state.Code 返回码，0代表成功，其他代表失败。

//state.Result.FileId 源文件id。

//state.Result.FileName 源文件名称。

//state.Result.StateList 转换状态列表。

//state.Result.StateList[i].FileId 转换文件id。

//state.Result.StateList[i].Server那台服务器正在处理该任务。

//state.Result.StateList[i].TotalCount 服务器还有多少任务在等待中。

//state.Result.StateList[i].State 任务的状态 wait processing completed

//state.Result.StateList[i].Percent 任务的转换进度 1-100。

### 4.2.4 播放视频

播放视频其实是播放的ts切片流，但前提是上传视频的时候添加了转换任务。在上传完成之后的返回值里面（Videos）包含了转换的详细信息。

播放器（hlsplayer.zip）：h5播放器，以下为播放器的代码样例。

<!DOCTYPE html>

<html>

<head>

<script src="jquery.js"></script>

<script src="~/hlsplayer/hlsplayer.min.js"

script-flowplayer="~/hlsplayer/flowplayer.swf"

script-flowplayerhls="~/hlsplayer/flowplayerhls.swf"

script-auto-init="true"></script>

</head>

<body>

<video controls="controls" width="900" height="600" class="hlsplayer">

<source src="/home/m3u8/{id}" />

</video>

</body>

</html>

其中script标签中的script-flowplayer、script-flowplayerhls是为低版本浏览器设计的flash播放器，一般用不到，也可以不设置这两项。

script-auto-init 设置为true 意思为等待页面加载完毕之后，自动查找页面上具有 class=” hlsplayer” 的标签进行视频初始化工作，这是默认行为；也可以设置为 false,然后手动在javascript中调用hlsplayer() 方法进行初始化工作。

/home/m3u8/{id} 对应服务端方法:

[HttpGet]

public ActionResult M3u8(string id)

{

DownloadFileItem item = fileClient.DownloadM3u8(id);

return File(item.FileStream, item.ContentType, item.FileName);

}

其中的id 是指转换视频id（Videos内部的FileId、m3u8文件id）。

等等！如果在上传的视频的时候添加了2个转换任务，分别对应pc端播放和mobile端播放，还有视频下载的需求，那么在我自己的系统中是不是就需要保存3个文件id呢？现在只需要保存一个源文件id就够了！

通过源文件id获取自适应流：

DownloadFileItem item= fileClient.DownloadM3u8MultiStream(id);

该方法通过源视频文件id返回一个播放流列表，hlsplayer.min.js自动判断：在网络条件好的时候选取播放效果好的视频流播放，网络条件差的时候选取比较流畅的视频流播放！

### 4.2.5 记录用户播放时间

有时为了更好的用户体验，可以记录每个用户每个视频的播放时间，等到用户下次访问时直接跳到上次播放的位置，该功能已经集成在hlsplayer.min.js的方法里面，默认是开启的，不需要该功能可以在引用的hlsplayer.min.js 的标签上加上script-tstime-record=”false”。

服务端获取m3u8切片列表的方法的修改如下：

[HttpGet]

public ActionResult M3u8(string id)

{

string tsTime = Request.Headers["TsTime"];

DownloadFileItem item = fileClient.DownloadM3u8(id, new UserData() { UserName = userName, UserTsTime = tsTime });

Response.AddHeader("TsTime", item.UserTsTime);

return File(item.FileStream, item.ContentType, item.FileName);

}

//或者

[HttpGet]

public ActionResult M3u8MultiStream(string id)

{

string tsTime = Request.Headers["TsTime"];

DownloadFileItem item = fileClient.DownloadM3u8MultiStream(id, new UserData() { UserName = userName, UserTsTime = tsTime });

Response.AddHeader("TsTime", item.UserTsTime);

return File(item.FileStream, item.ContentType, item.FileName);

}

//userName 是当前登录的用户名，唯一标识。

当用户请求m3u8列表的时候，根据userName查询出视频当前的播放进度，放在headers中返回到前端，具体前端的操作在hlsplayer.min.js中已经封装好。当用户请求ts切片的时候会在headers中带上当前视频播放的时间，并存到服务端。

### 4.2.6 下载视频

DownloadFileItem file = fileClient.DownloadFile(id);

// file.FileName:文件名。

// file.ContentType:文件的contentType。

// file.ContentLength:文件的contentLength。

// file.FileStream:视频的文件流对象

//如果是mvc项目，则直接在Action中返回文件 return File(file.FileStream, file.ContentType, file.FileName);

//如果是aspx页面，则用该方法下载文件：

//file.FileStream.CopyTo(Response.OutputStream);

//Response.ContentType = file.ContentType;

//Response.AppendHeader("Content-Disposition", "attachment;filename=" + file.FileName);

//Context.Response.AddHeader("Content-Length",file.ContentLength.ToString());

//Response.Flush();

### 4.2.7 视频截图

1）通过api上传视频截图的两种方式

var response = fileClient.UploadVideoCapture(fileId,fileBase64);

//默认的文件名和视频的名称一致

//fileId:视频源文件id。

//fileBase64:图片的base64字符串。

var response = fileClient.UploadVideoCapture(fileId,fileName,stream,null);

//fileId:视频源文件id。

//fileName:截图的文件名。

//stream:图片的文件流。

2）通过hlsplayer.min.js 封装的方法在前端截图

//该方法不是自动触发的，需要点击某个按钮事件来触发

$(".hlsplayer").videoCP(1,function (img) { //1是指按原画面1:1比率截图

console.log(img.src); //图片文件的base64字符串

});

3）获取视频截图文件id列表

var result = fileClient.GetVideoCaptureIds(id); //id:源文件id

//result.Code; 返回码，0代表成功，其他代表失败。

//result.Result: 截图文件的id列表。

4）删除视频截图文件

var result = fileClient.DeleteVideoCapture(id); //id:视频截图文件id

## 4.3 上传、下载附件

FileClient fileClient = new FileClient(authCode, remoteUrl);

### 4.3.1 上传附件

var result = fileClient.UploadAttachment(fileName, fileStream);

int code = result.Code; //返回码，0代表成功，其他代表失败。

IEnumerable<AttachmentFileResult> res = result.Result; //返回的文件内容,因为上传的是单个附件，res列表的长度为1。

//res[i].FileId:返回的文件id（以后获取文件的依据）。

//res[i].FileName:返回的文件名。

//res[i].FileSize:返回文件的大小，字节。

### 4.3.2 上传附件的其他重载方法

UserData：用户信息，设置文件权限必传该选项。

UserData.UserName：用户名称，文件的上传者，文件的拥有者。

UserData.UserAgent：用户浏览器信息，Request.UserAgent。

UserData.UserIp：用户ip信息，Request.UserHostAddress。

UserAccess：文件的访问权限信息，设置权限的前提是设置了 UserData 信息。

UserAccess.Company：文件的可访问公司代码（companyCode）。

UserAccess.DepartmentCodes：文件的可访问部门代码列表。

UserAccess.AccessUsers：文件的可访问用户列表（用户名）。

### 4.3.3 下载附件

DownloadFileItem file = fileClient.DownloadFile(id); //附件id

// file.FileName:文件名。

// file.ContentType:文件的contentType。

// file.ContentLength:文件的contentLength。

// file.FileStream:附件文件流对象

//如果是mvc项目，则直接在Action中返回文件 return File(file.FileStream, file.ContentType, file.FileName);

//如果是aspx页面，则用该方法下载文件：

//file.FileStream.CopyTo(Response.OutputStream);

//Response.ContentType = file.ContentType;

//Response.AppendHeader("Content-Disposition", "attachment;filename=" + file.FileName);

//Context.Response.AddHeader("Content-Length",file.ContentLength.ToString());

//Response.Flush();

# 5. API(JavaScript)

对于部署到外网的FileServiceWeb来说，可以直接使用JavaScript接口来上传和下载文件，调用之前请添加对 file-service-api-{version}.js的引用。

在随后的调用中：

authCode 是指在 配置管理 -> 应用程序 中添加的记录的 认证编码

remoteUrl 是指FileServiceWeb 部署好之后的url地址。

所有的接口调用成功后的返回值中code为0，Message为 success，所有需要下载文件 的接口返回值为 下载文件的url，请使用window.open(url)来下载文件。

## 5.1 上传、下载图片

var fileClient = new FileClient(authCode, remoteUrl);

### 5.1.1 上传图片

fileClient.uploadImage(file, imageConvert, userAccess, success, progress, error, userName);

//file: File对象 或者 FileList对象。

//imageConvert: 图片转换任务描述对象或者对象数组，没有转换任务设置null，案例：{ flag: "mobile", format: 0, model: 2, x: 0, y: 0, width: 300, height: 0 }。

//flag: 字符串，缩略图的标识，请填写有意义的字符串（mobile，pad，small）。

//format: 缩略图的格式，0:和原图保持一致；1:jpeg, 2:png, 3:gif, 4:bmp。

//model: 转换模式，0:缩放, 1:剪切, 2:按照宽度, 3:按照高度 。

//x: 仅仅在Cut模式下有效，指从原图左上角算起 水平轴（X轴）的坐标。

//y: 仅仅在Cut模式下有效，指从原图左上角算起 垂直轴（Y轴）的坐标。

//width: 缩略图的宽度。

//height: 缩略图的高度。

//userAccess: 文件的访问权限信息对象或数组，没有权限信息设置 null，设置权限的前提是设置了 userName(文件所有者)，案例：{ company: "", departmentCodes: [], accessUsers: [] }

// company: 公司编号。

// departmentCodes: 部门编号列表。

// accessUsers: 可访问的用户名称列表。

//success: 上传成功之后调用。

//progress: 上传进度事件。

//error: 上传出现错误的时候调用。

//userName: 文件所有者，设置权限必备。

案例（上传没有访问权限的图片）：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title>

<script src="file-service-api-1.0.js"></script>

</head>

<body>

<input type="file" name="file" id="file" /><br />

</body>

</html>

<script>

var fileClient = new FileClient(authCode,remoteUrl);

document.getElementById("file").onchange = function () {

var files = this.files;

fileClient.uploadImage(files, null, null,

function (data) {

console.log(data);

},

function (percent) {

console.log(percent);

},

null,

null,

)

}

</script>

### 5.1.2 上传图片的其他参数介绍

imageConvert: 图片转换描述对象，一个对象或者一个对象数组。

{flag:”moblie”,format:0,model:0,x:0,y:0,width:300,height:300} //转换一个标识为mobile，缩略图格式和上传图片格式一致，按照 width=300，height=300进行缩放的缩略图。

userAccess: 权限描述对象，一个对象或者一个对象数组。

{company:”ad03c1ddb01b”,departmentCodes:[“cfdf6aed4ee7”],accessUsers:[“li”]} //该图片只有在ad03c1ddb01b公司，部门编码为cfdf6aed4ee7，或者用户名为li的人可以访问。

userName: 图片的所有者。

### 5.1.3 下载源图片

var url = fileClient.getFileUrl(fileId); //通过源图片id获取下载的url

window.open(url); //下载文件

### 5.1.4 下载缩略图

var url = fileClient.getThumbnailUrl(thumbId); //通过缩略图id获取下载的url

window.open(url); //下载文件

### 5.1.5 通过源文件id下载缩略图

var url = fileClient.getThumbnailFromSourceIdUrl(fileId); //通过源图片id获取下载的url

### 5.1.6 通过源文件id和falg下载缩略图

var url = fileClient.getThumbnailFromSourceIdFlagUrl(fileId,flag); //通过源图片id和flag获取下载的url

window.open(url); //下载文件，如果没有缩略图，则下载源文件

## 5.2 上传、下载视频

var fileClient = new FileClient(authCode, remoteUrl);

### 5.2.1 上传视频

fileClient.uploadImage(file, videoConvert, userAccess, success, progress, error, userName);

//file: File对象 或者 FileList对象。

//videoConvert: 视频转换任务描述对象或者对象数组，没有转换任务设置null，案例：{ flag: "mobile", format: 0, quality: 1}。

//flag: 字符串，转换视频的标识，请填写有意义的字符串（mobile，pad）。

//format: 缩略图的格式，0:m3u8。

//quality: 转换视频的质量，0:原画, 1:稍低, 2:中等, 3:差 。

//userAccess: 文件的访问权限信息对象或数组，没有权限信息设置 null，设置权限的前提是设置了 userName(文件所有者)，案例：{ company: "", departmentCodes: [], accessUsers: [] }

// company: 公司编号。

// departmentCodes: 部门编号列表。

// accessUsers: 可访问的用户名称列表。

//success: 上传成功之后调用。

//progress: 上传进度事件。

//error: 上传出现错误的时候调用。

//userName: 文件所有者，设置权限必备。

案例（上传没有访问权限的视频）：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title>

<script src="file-service-api-1.0.js"></script>

</head>

<body>

<input type="file" name="file" id="file" /><br />

</body>

</html>

<script>

var fileClient = new FileClient(authCode,remoteUrl);

document.getElementById("file").onchange = function () {

var files = this.files;

fileClient.uploadVideo(files, {flag:"mobile",format: 0,quality:2}, null,

function (data) {

console.log(data);

},

function (percent) {

console.log(percent);

},

null,

null,

)

}

</script>

### 5.2.2 上传视频的其他参数介绍

videoConvert: 视频转换描述对象，一个对象或者一个对象数组。

{flag:"mobile",format: 0,quality:2} : 转换一个标识为mobile，格式为m3u8，质量中等的视频。

userAccess: 权限描述对象，一个对象或者一个对象数组。

{company:”ad03c1ddb01b”,departmentCodes:[],accessUsers:[“li”]} //该视频只有在ad03c1ddb01b公司，用户名为li的人可以访问。

userName: 视频的所有者。

### 5.2.3 获取视频转换进度

fileClient.getFileState(fileId, function (data) {

console.log(data);

});

//fileId: 视频源文件id。

//data.Code 返回码，0代表成功，其他代表失败。

//data.Result.FileId 源文件id。

//data.Result.FileName 源文件名称。

//data.Result.StateList 转换状态列表。

//data.Result.StateList[i].FileId 转换文件id。

//data.Result.StateList[i].Server那台服务器正在处理该任务。

//data.Result.StateList[i].TotalCount 服务器还有多少任务在等待中。

//data.Result.StateList[i].State 任务的状态 wait processing completed

//data.Result.StateList[i].Percent 任务的转换进度 1-100。

### 5.2.4 播放视频

播放视频其实是播放的ts切片流，但前提是上传视频的时候添加了转换任务。在上传完成之后的返回值里面（Videos）包含了转换的详细信息。

播放器（hlsplayer.zip）：h5播放器，以下为播放器的代码样例。

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title>

<script src="file-service-api-1.0.js"></script>

<script src="scripts/jquery-3.2.1.min.js"></script>

<script src="hlsplayer.min.js"

script-flowplayer="flowplayer.swf"

script-flowplayerhls="flowplayerhls.swf"

script-auto-init="false"></script>

</head>

<body>

<video controls="controls" width="500" height="300" class="hlsplayer">

<source src="" id="video\_source"/>

</video>

</body>

</html>

<script>

var fileClient = new FileClient(authCode, remoteUrl);

var url = fileClient.getM3u8Url(m3u8Id);

$("#video\_source").attr("src", url);

hlsplayer();

</script>

其中script标签中的script-flowplayer、script-flowplayerhls是为低版本浏览器设计的flash播放器，一般用不到，也可以不设置这两项。

script-auto-init 设置为true 意思为等待页面加载完毕之后，自动查找页面上具有 class=” hlsplayer” 的标签进行视频初始化工作，这是默认行为；也可以设置为 false,然后手动在javascript中调用hlsplayer() 方法进行初始化工作。

m3u8Id: 转换视频的id。

如果在上传的视频的时候添加了2个转换任务，分别对应pc端播放和mobile端播放，还有视频下载的需求，那么在我自己的系统中是不是就需要保存3个文件id呢？现在只需要保存一个源文件id就够了！

把 var url = fileClient.getM3u8Url(m3u8Id); 换成

var url = fileClientgetM3u8MultiStreamUrl(fileId) //fileId是源文件id。

该方法通过源视频文件id返回一个获取多种码率视频流url，hlsplayer.min.js自动判断：在网络条件好的时候选取播放效果好的视频流播放，网络条件差的时候选取比较流畅的视频流播放！

### 5.2.5下载视频

var url = fileClient.getFileUrl(fileId); //通过源文件id获取下载的url

window.open(url); //下载文件

### 5.2.6 视频截图

1）通过视频截图的方式：

$("#cp").click(function () {

$(".hlsplayer").videoCP(1, function (img) { //1是指按原画面1:1比率截图

uploadCp(img.src);

});

});

function uploadCp(base64) {

// fileId:视频源文件id

fileClient.uploadVideoCapture(fileId, base64, function (data) {

console.log(data);

});

}

2）通过文件框上传另外的视频截图：

document.getElementById("file").onchange = function () {

var files = this.files;

fileClient.uploadVideoCapture(fileId, files,function (data) {

console.log(data);

})

}

//fileId: 视频源文件id

3）获取视频截图文件id列表

fileClient.getVideoCaptureIds(fileId, function (data) { //fileId: 视频源文件id

console.log(data);

});

//data.Code; 返回码，0代表成功，其他代表失败。

//data.Result: 截图文件的id列表。

4）删除视频截图文件

fileClient.deleteVideoCapture(fileId, function (data) {//fileId:视频截图文件id

console.log(data);

});

## 5.3 上传、下载附件

var fileClient = new FileClient(authCode, remoteUrl);

### 5.3.1 上传附件

fileClient.uploadAttachment(file, userAccess, success, progress, error, userName);

//file: File对象 或者 FileList对象。

//userAccess: 文件的访问权限信息对象或数组，没有权限信息设置 null，设置权限的前提是设置了 userName(文件所有者)，案例：{ company: "", departmentCodes: [], accessUsers: [] }

// company: 公司编号。

// departmentCodes: 部门编号列表。

// accessUsers: 可访问的用户名称列表。

//success: 上传成功之后调用。

//progress: 上传进度事件。

//error: 上传出现错误的时候调用。

//userName: 文件所有者，设置权限必备。

案例（上传没有访问权限的图片）：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title>

<script src="file-service-api-1.0.js"></script>

</head>

<body>

<input type="file" name="file" id="file" /><br />

</body>

</html>

<script>

var fileClient = new FileClient(authCode,remoteUrl);

document.getElementById("file").onchange = function () {

var files = this.files;

fileClient.uploadAttachment(files,null,function (data) {

console.log(data);

})

}

</script>

### 5.3.2 上传附件的其他参数介绍

userAccess: 权限描述对象，一个对象或者一个对象数组。

{company:”ad03c1ddb01b”,departmentCodes:[“cfdf6aed4ee7”],accessUsers:[]} //该附件只有在ad03c1ddb01b公司，部门编码为cfdf6aed4ee7的人可以访问。

userName: 视频的所有者。

### 5.3.3 下载附件

var url = fileClient.getFileUrl(fileId); //通过源文件id获取下载的url

window.open(url); //下载文件

# 6. 数据副本集（附录）

对一般的公司来说，一台数据服务器已经够用了，如果要确保数据的安全性和服务器的稳定性，就需要配置副本集了。

副本集的最终目的是配置3台服务器，一台主节点，一台从节点，一台仲裁节点，其中主节点和从节点的数据互为备份（数据一致），仲裁节点只有在 主节点或从节点挂掉了 起 选取另外的主节点 作用（这台服务器对资源要求不高，可以灵活部署），这3台服务器挂掉了任何一台，对整个服务没有任何影响。

 也可以不设仲裁节点，设置1台主节点，2台从节点，这样数据就在这3台服务器上互为备份，挂掉任何一台，对整个服务也没有任何影响。

首先在3台服务器上分别安装mongodb，安装步骤请参考2.1节点，安装完成之后，分别打开3个配置文件添加如下配置节点 并重启mongodb服务。

replication:

replSetName: MyReplSetName

随意找一个服务器用 cmd命令行定位到mongodb安装目录（C:\Program Files\MongoDB\Server\4.0\bin\）。

输入命令行：

mongo.exe ip:port #登录到mongodb服务器

rs.Initiate() #初始化一个集群

rs.add("ip:port") #往集群中添加从节点

rs.addArb("ip:port") #往集群中添加仲裁节点

配置完成之后可以输入命令 rs.status() 查看集群状态，修改web管理界面的数据库连接字符串（样例：mongodb://ip:port,ip:port,ip:port/admin），修改完成之后可以在web管理界面的 概况统计 的服务器节点 查看到集群的状态。

# 7. 数据分片（附录）

系统运行一段时间之后可能数据量非常多，当一台数据服务器的存储容量已经不能满足继续存储需求的时候，就需要对数据进行分片存储了。在系统设计的早期就已经考虑到了分片需求并且已经默认在集合中设置好了分片键（关于分片键的知识请上网查找相关资料了解）。

分片的服务器结构图



mongos是一台普通的mongodb服务器，但是不存数据，只是缓存配置集群（configset）的数据。

replsetA、replsetB都是普通的副本集，里面的主节点（Primary）和从节点（Secondary）的数据互为备份。

replsetA、replsetB分别保存了整个数据集的一部分数据。

配置集群（configset）中挂掉一台 或者 副本集A（replsetA）中挂掉一台 或者 副本集B（replsetB）中挂掉一台 对整个服务没有任何影响。

配置集群（configset）不能配置仲裁节点。

配置集群（configset）和仲裁节点（arbiter）都是对资源要求不高的服务器，可以灵活配置。mongos一般和web服务器配置在一起，算一算除了mongos一共有9台服务器，但是非数据服务器可以合理的和其他服务器公用一台来节约资源。

在配置replsetA副本集的时候在配置文件中添加以下配置

replication:

replSetName: replsetA

sharding:

clusterRole: shardsvr

在配置 replsetB副本集的时候在配置文件中添加以下配置

replication:

replSetName: replsetB

sharding:

clusterRole: shardsvr

在配置configset 集群的时候在配置文件中添加以下配置

replication:

replSetName: configSet

sharding:

clusterRole: configsvr

在配置mongos服务器的时候请删除配置文件中的storage以及它下属节点，因为mongos不存数据，并在配置文件中添加以下配置

sharding:

configDB: configSet /ip:port,ip:port,ip:port

注意：

configDB配置的是configset 集群的集群名称和集群服务器列表。

在把mongos配置成服务的形式运行的时候使用以下命令行（不能用mongod.exe）

mongos.exe --install --config "C:\Program Files\MongoDB\Server\4.0\bin\mongod.cfg"

通过以上的配置我们知道：mongos服务器已经知道配置集群的存在，但是还不知道replSetA和replSetB的存在，现在通过命令行添加2个副本集。

在mongos服务器上使用命令行：

mongo.exe ip:port #登录到mongos服务器

sh.addShard("replsetA/ip:port,ip:port,ip:port") #添加副本集A（replsetA），其中的参数分别是副本集A的名称和副本集A的机器列表。

sh.addShard("replsetB/ip:port,ip:port,ip:port") #添加副本集B（replsetB），其中的参数分别是副本集B的名称和副本集B的机器列表。

到此分片集群服务器已经配置完毕，修改web管理界面的数据库连接字符串为mongos的服务器（样例：mongodb://127.0.0.1:27017/admin），修改完成之后可以在web管理界面的 概况统计 的 服务器 节点 查看到集群的状态，以下操作是对集合的分片操作。

sh.enableSharding("FileService") #开启数据库FileService的分片功能。

sh.shardCollection("FileService.fs.chunks",{"files\_id" : 1,"n" : 1}) #对FileService数据库的fs.chunks集合开启分片，分片键为{"files\_id" : 1,"n" : 1}。

sh.shardCollection("FileService.convert.chunks",{"files\_id" : 1,"n" : 1}) #对FileService数据库的convert.chunks集合开启分片，分片键为{"files\_id" : 1,"n" : 1}。

对于开启分片的集合 里面的数据就会均匀分布到 replsetA和replsetB中去，系统中大部分表预先设置好的分片键为{ "From" : 1,"CreateTime" : 1}，查询这些表的数据的时候要尽量带上From参数，这样mongos会通过From参数把查询路由到相应的分片，可以提高查询效率，如果数据进一步增加的话，还可以继续添加replsetC …集群来分担数据量。

随着系统的运行，有一些存二级制文件的集合 大小会比其他的存记录的集合增长的要快，还有一些集合每天产生的数据量较大比如Log，Download，那么这些集合也有可能需要分片。

最有可能需要分片的集合：

fs.chunks ：这是存储源文件chunks的集合，系统已经默认设置好了分片键 {"files\_id" : 1,"n" : 1}。

convert.chunks：这是存储转换文件 chunks的集合，系统已经默认设置好了分片键 {"files\_id" : 1,"n" : 1}。

VideoCapture：这是存储视频截图文件的集合，预先设置好的分片键为 { "From" : 1,"CreateTime" : 1}

Ts：这是存储视频切片文件的集合，预先设置好的分片键为 { "From" : 1,"CreateTime" : 1}

Thumbnail：这是存储图片缩略图文件的集合，预先设置好的分片键为 { "From" : 1,"CreateTime" : 1}

Log：这是存储操作日志文件集合，预先设置好的分片键为 { "From" : 1,"CreateTime" : 1}

FilePreview 和 FilePreviewBig：这是存储图片小图标的集合，预先设置好的分片键为 { "From" : 1,"CreateTime" : 1}

Download：这是存储下载量的集合，预先设置好的分片键为 { "From" : 1,"CreateTime" : 1}