Java&J2EE应用与开发

学号1： 201592038

姓名1： 阙建明

班级1： 软网1503

学号2： 201592442

姓名2： 邹洋奕

班级2： 软网1503

大作业得分点

编码规范及易读性（15分）：

功能实现（35分）：

异常处理（25分）：

文档质量（25分）：

大作业总成绩（占60%）：

平时作业总成绩（占40%）：

期末总成绩（100分）：

目录

[1 功能要求分析 3](#_Toc488081329)

[1.1 FileServer 3](#_Toc488081330)

[1.2 StorageNode 3](#_Toc488081331)

[1.3 FileClient 3](#_Toc488081332)

[1.4 监控程序 4](#_Toc488081333)

[2 具体实现思路 4](#_Toc488081334)

[3 各模块及主要功能展示 5](#_Toc488081335)

[3.1 FileServer模块 5](#_Toc488081336)

[3.1.1 ComunicateWithStorageStrategy 5](#_Toc488081337)

[3.1.2 FCStrategy 6](#_Toc488081338)

[3.1.3 FileStorageServer+FileStorageDataDealRunnable 7](#_Toc488081339)

[3.2 FileClient模块 7](#_Toc488081340)

[3.2.1 CommunicateWithFileServerOperator 8](#_Toc488081341)

[3.2.2 FileTransOperator 10](#_Toc488081342)

[3.2.3 FileTransToStorageSupport 10](#_Toc488081343)

[3.3 FileStorage 11](#_Toc488081344)

[3.3.1 CommunicateWithServerStrategy 11](#_Toc488081345)

[3.3.2 TransWithClientStrategy 12](#_Toc488081346)

[3.4 Monitor 12](#_Toc488081347)

[4 不足改进 13](#_Toc488081348)

[4.1 问题一及解决方案 13](#_Toc488081349)

[4.2 问题二及解决方案 13](#_Toc488081350)

[5 引用类库文献 14](#_Toc488081351)

# 功能要求分析

本应用基于Socket编程模拟实现文件的分布式存储，需要实现四个模块的功能。

## FileServer

FileServer服务器，主要用于负载均衡，管理文件信息和存储节点信息。在文件上传时，客户端需向FileServer请求存储节点的信息，此时服务器在分配存储节点时要考虑负载均衡。在FileServer上通过Map集合保存存储节点的信息以及文件的信息。服务器宕机之前将Map集合中存储的信息序列化到文件中，下次启动的时候再从文件中序列化回来。

## StorageNode

StorageNode服务器主要用于存储用户上传的文件，本地不能保存过多的信息。需要一个配置文件，配置StorageNode的IP、端口、容量、文件存储路径等信息，在程序启动的时候要加载配置文件的信息。文件在保存的时候文件名为FileServer给改文件分配的UUID的值，不能使用原文件的名字。在收到文件的时候要向FileServer请求同组服务器的信息，将刚收到的文件备份到同组的其他节点服务器。

## FileClient

FileClient应用程序主要实现文件的上传，下载，删除等功能。在上传时需要向FileServer请求可用的节点服务器组的列表，FileClient再根据收到的节点服务器的信息，将文件上传到节点服务器。如果主节点服务器不可用，或者在传输过程中与节点服务器的连接中断，则应该有容错的功能，将文件上传到同组的备份节点服务器上。在下载的时候FileClient需凭借UUID向FileServer询问文件具体在哪个节点服务器上保存了，再根据收到的信息，找到对应的节点服务器下载文件。在删除文件时，FileClient将删除文件的请求发给FileServer，委托FileServer删除掉存储在节点服务器上的文件。

## 监控程序

监控程序，需要对节点服务器的状态进行监控，并以表格的方式显示出来。

# 具体实现思路

首先四个功能模块分属于四个项目。统一协商好一套协议，在通信过程中，根据收到的不同的code执行不同的行为。

在Socket通信过程中，统一采用DataInputStream、DataOutputStream。通信过程中如果传输基本数据类型，则直接调用上述两个流操作对象对应的函数即可；如果传输对象，则引入GSON库，将对象序列化为JSON格式的字符串，传到目标主机后在用GSON库将JSON 串反序列化为原对象；如果传输文件，则建立缓冲数组，分配传输字节数组即可。

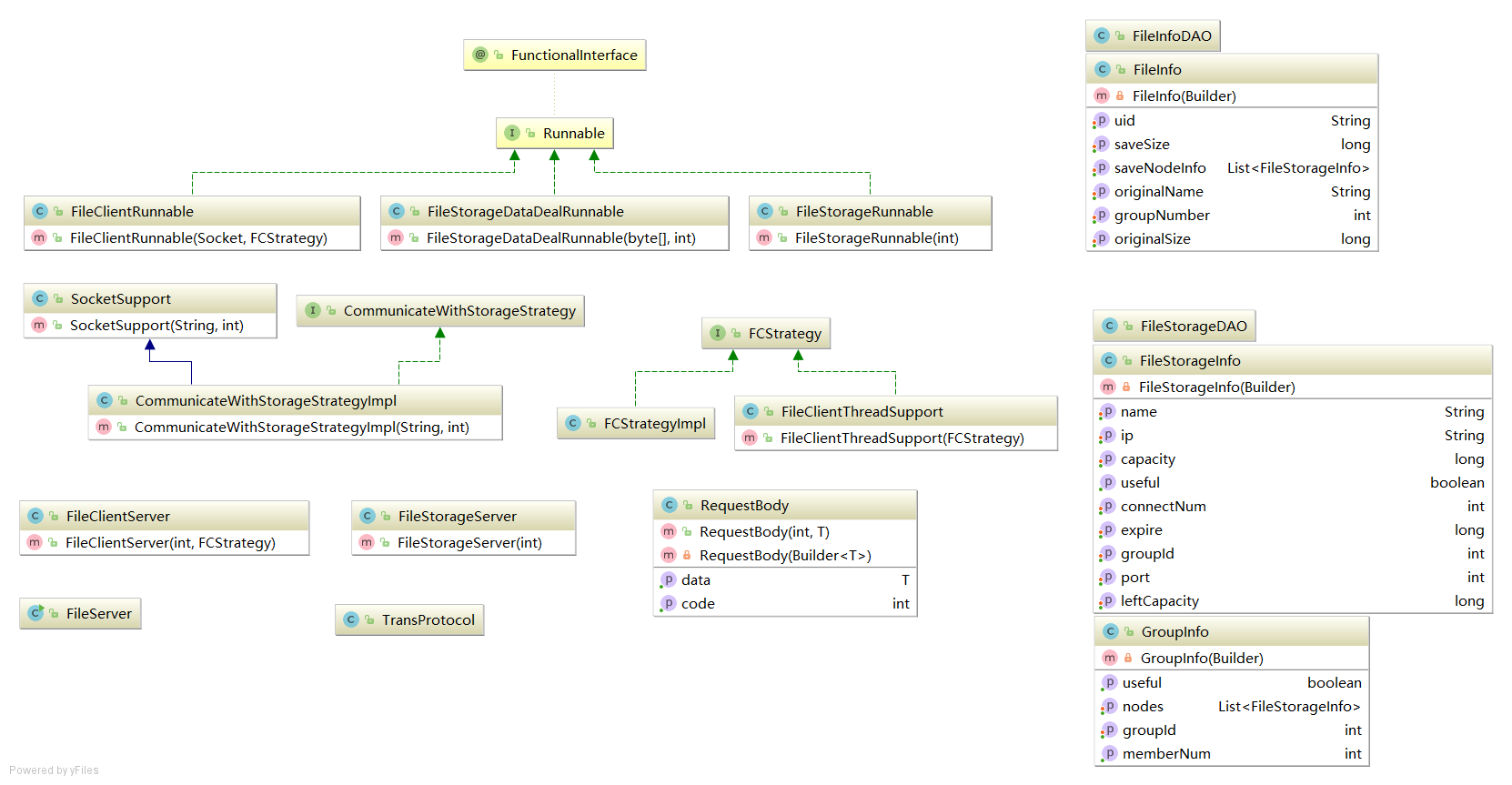
为了实现FileServer服务器对节点服务器的管理，节点服务器自启动开始便向FileServer发送UDP packet作为心跳包。实时向FileServer汇报自己的情况。对于UDP通信不可靠问题，在数据末尾添上CRC校验码，FileServer在根据收到的信息计算CRC校验码，并进行比对，判断本次传输是否有效。

为了保护用户的文件安全，不能将原文件存于节点服务器，在文件上传的时候将文件分块压缩、加密，先向输出流中输出本数据块在压缩、加密以后的大小，紧接着再将处理后的数据块输出。在节点服务器上接收的时候先读取一个整形数，表示下一个传输的数据块的长度，再调用readFully方法读取下一个块，输出到文件中的时候也是先输出块的长度，紧接着输出数据块。文件下载的时候采用同样的思路读取、解压、解密，最后还原出原文件。

在监控程序方面则采用JavaWeb方式实现，在JavaWeb服务器收到Get请求后随即向FileServer发起Socket通信，获取到各存储节点的信息，再将数据通过JSP页面显示出来，同时设定每隔一秒自动刷新，以达到实时监控的效果。

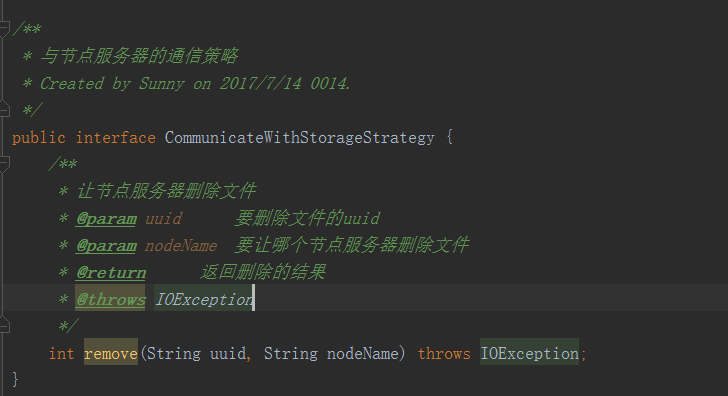
# 各模块及主要功能展示

## FileServer模块



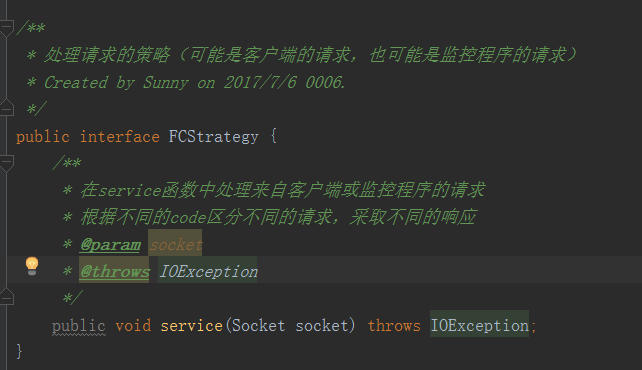
### ComunicateWithStorageStrategy

由服务器发起的与节点服务器间的通信只有一个，那就是删除文件的时候，服务器在收到客户端的委托请求后，向对应的节点服务器发送删除文件的请求。

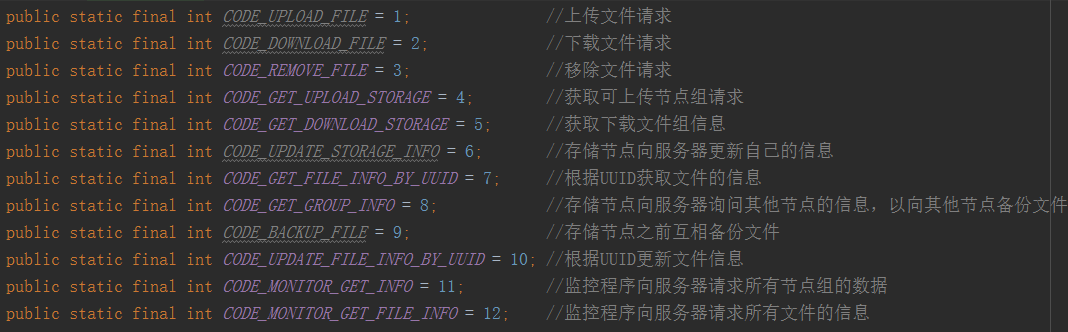


### FCStrategy

* 这个策略用于应对客户端和监控程序的请求，支持各种信息的增删改查以及代理客端删除，文件等。在service函数具体实现的时候根据不同的Code区分不同请求的目的，并作出相应的响应动作。



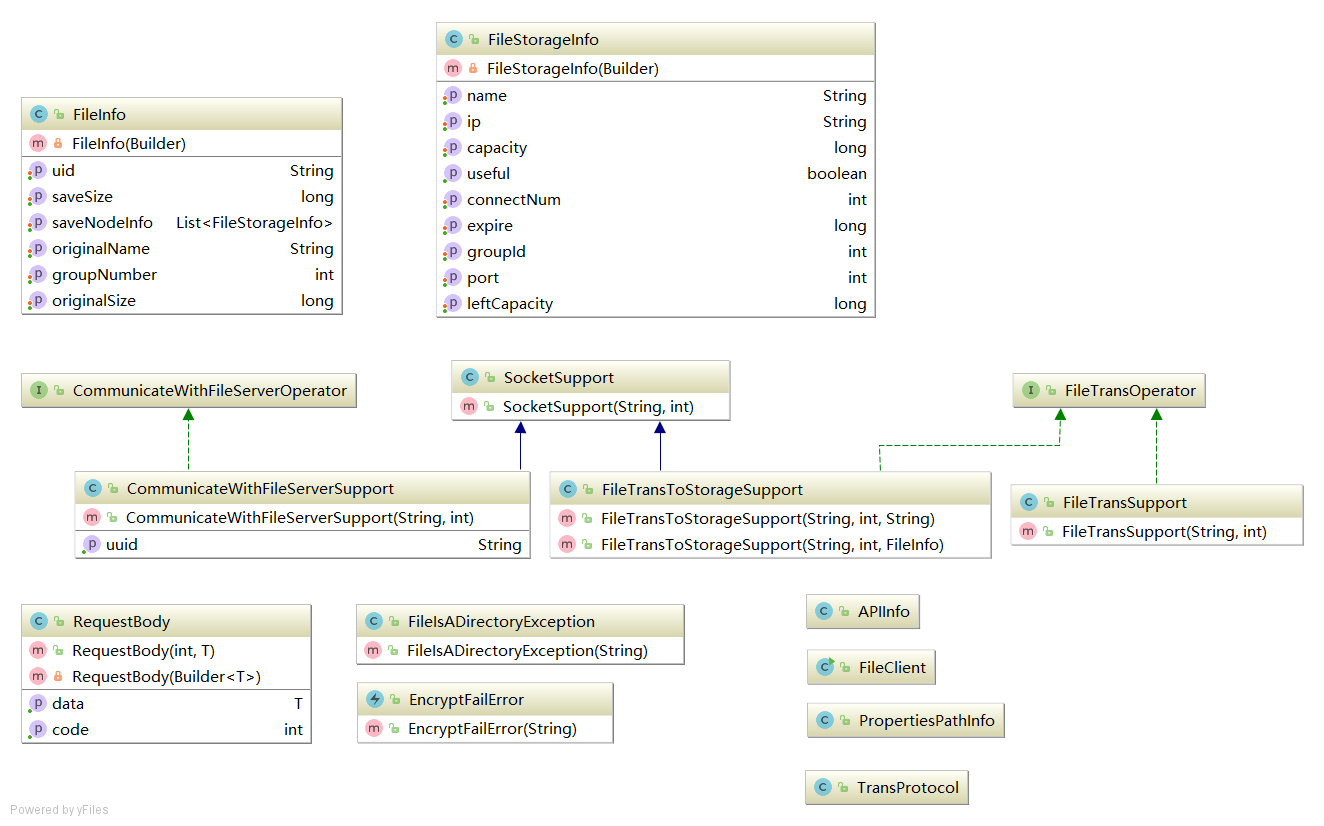
* 以下是不同的协议类中定义的不同的code代表不同的请求



### FileStorageServer+FileStorageDataDealRunnable

这两个类主要负责接收来自节点服务器发送的心跳包。其中FileStorageServer负责接收UDP报文，并将其放入阻塞队列当中，而FileStorageDataDealRunnable则跑在另一个线程，负责从阻塞队列中取报文并处理。（心跳包有三种，一种只用于更新节点服务器信息；一携带删除文件请求；一种携带添加文件请求。）

## FileClient模块



### CommunicateWithFileServerOperator

该策略接口主要定义了客户端与服务器之间的通信规则，要求实现该接口的类必须实现其定义的五个函数，分别为：

* **List<FileStorageInfo> getUploadAvailableStorage(String path)**

用于在上传文件时向服务器询问应该上传到哪一个节点服务器组。

* **List<FileStorageInfo> getDownloadAvailableStorage(String uuid);**

用于在下载文件的时候将UUID传给服务器，询问应该到哪一个节点组去下载该UUID对应的文件。

* **FileInfo getFileInfoByUUID(String uuid);**

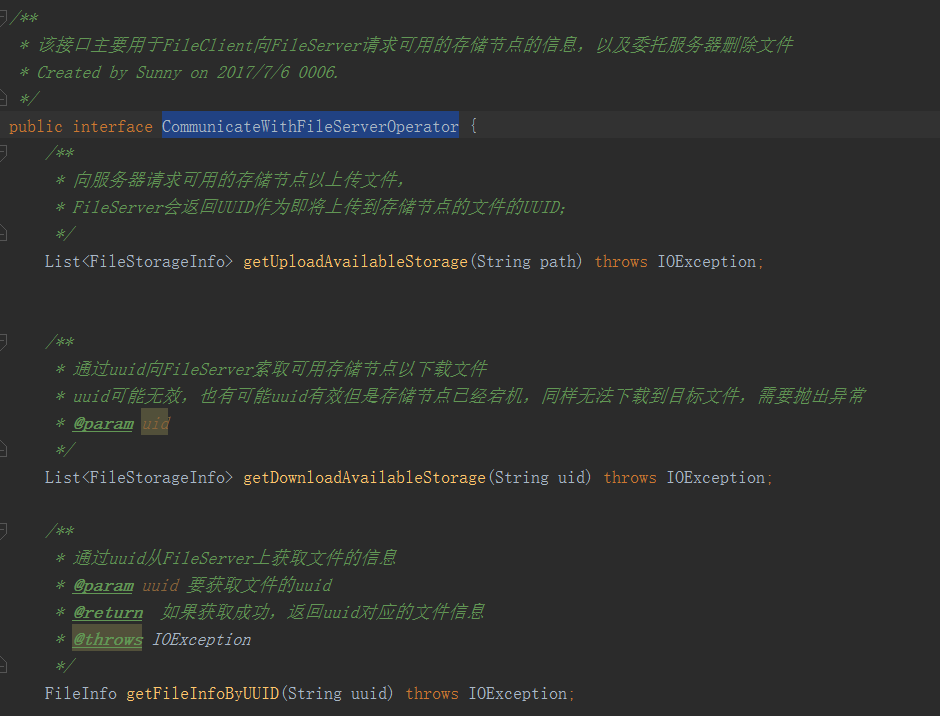
通过UUID向服务器查询该UUID对应的文件信息。

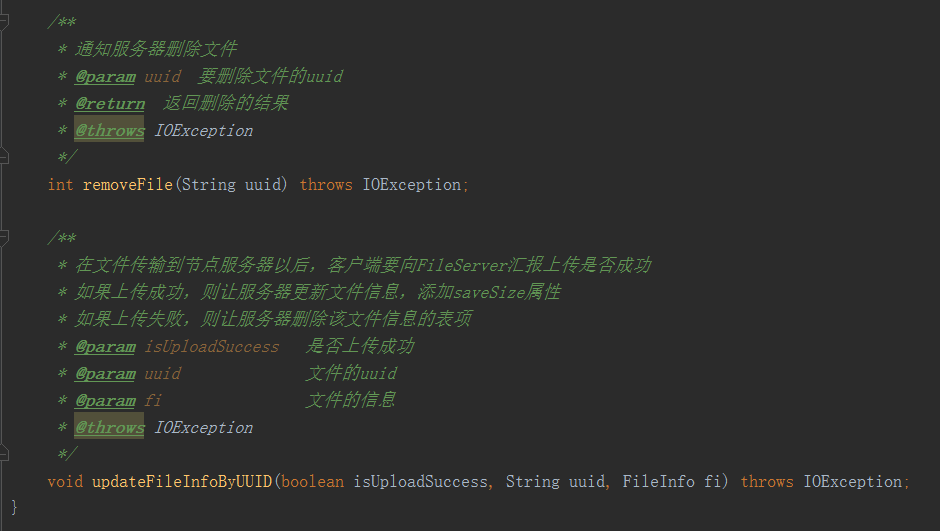
* **int removeFile(String uuid);**

委托服务器删除所有存储在节点服务器上该UUID对应的文件。

* **void updateFileInfoByUUID(boolean isUploadSuccess, String uuid, FileInfo fi);**

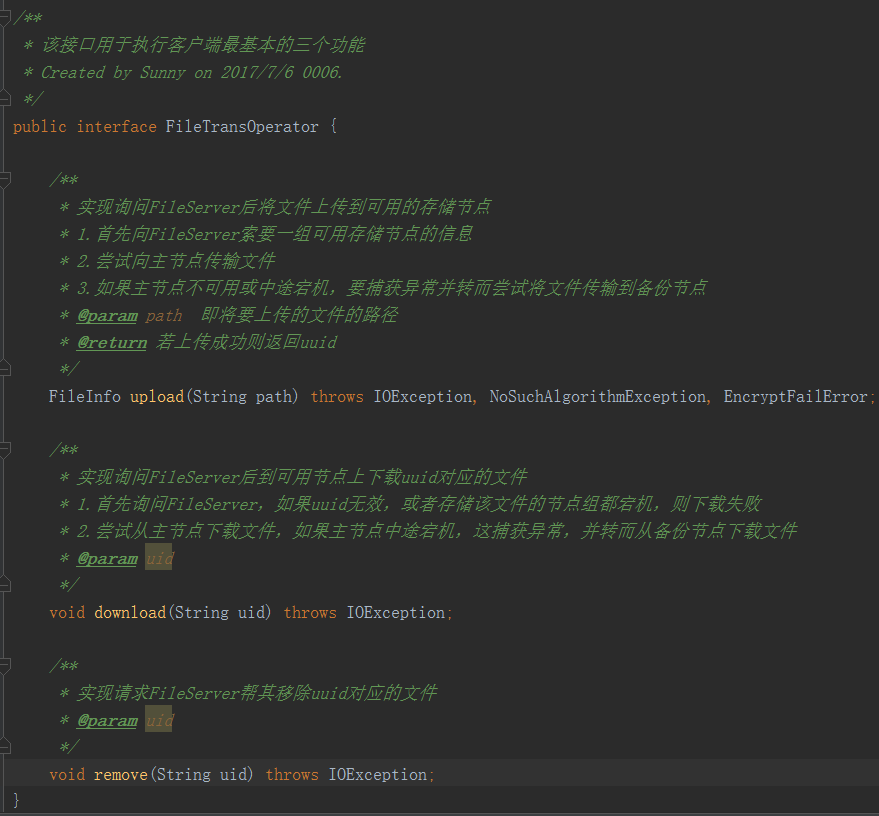
用于通知服务器本次文件传输的情况，如果文件传输成功，则让服务器更新存在服务器上该文件的信息（文件在节点服务上保存的真是大小）。如果文件上传失败，则让服务器删除该文件对应的表项。





### FileTransOperator

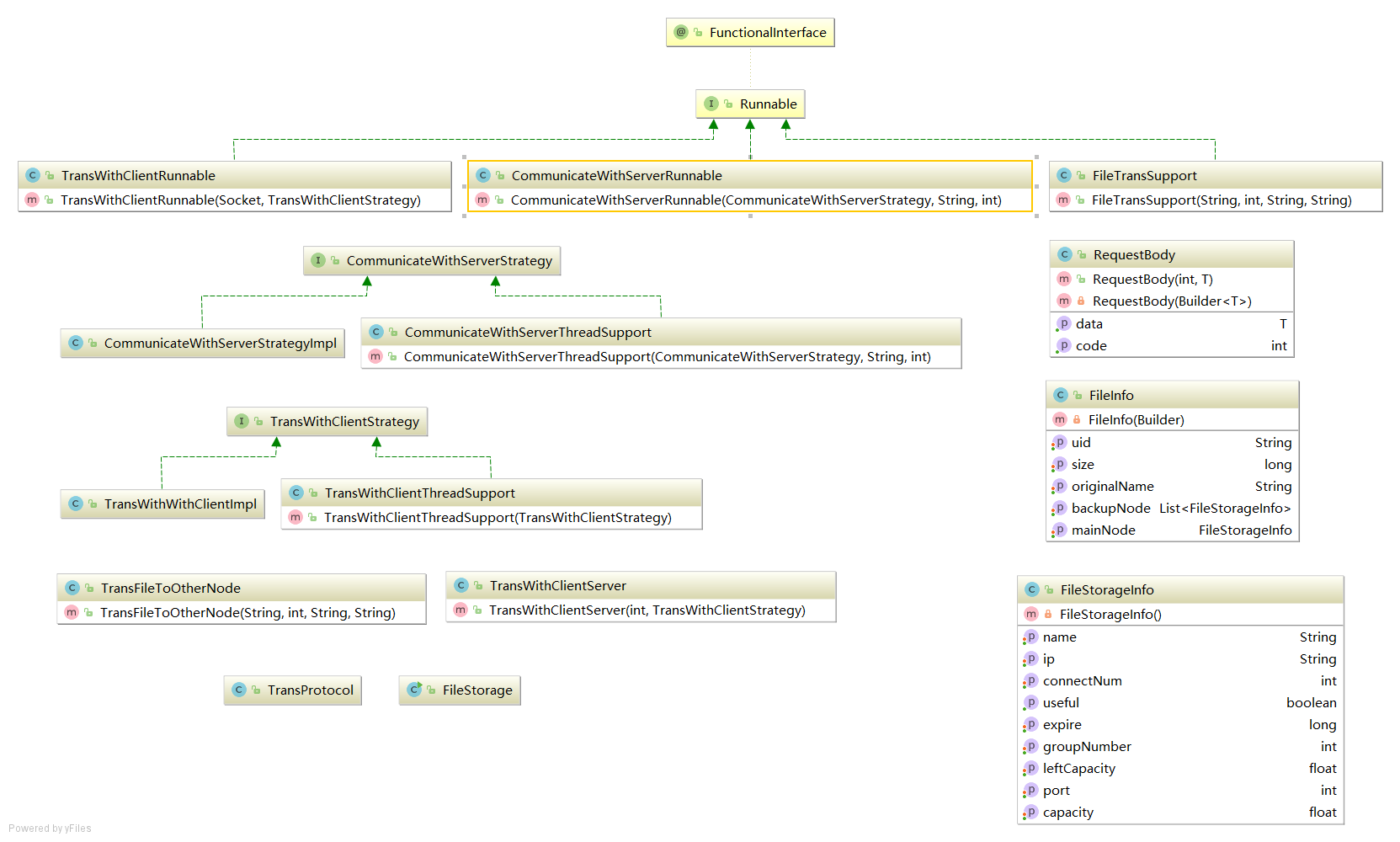
该策略主要定义了提供给用户的三个接口，分别为上传、下载和删除文件。具体实现的时候会分别依赖于与服务器和节点服务器的通信策略。



### FileTransToStorageSupport

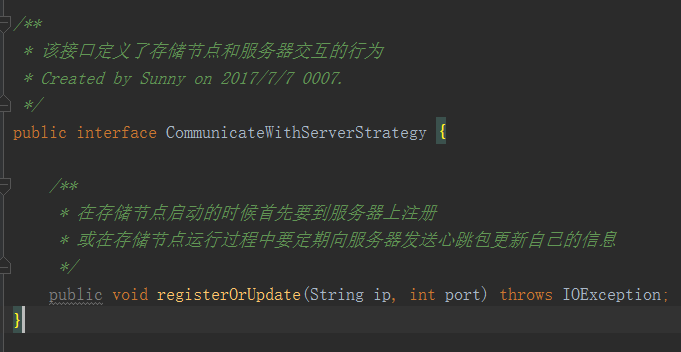
该类提供了客户端和节点服务器之间的通信。其实现了FileTransOperator，并实现了其中的download和upload方法，remove方法为空方法（因为删除文件操作不是由客户端直接向节点服务器请求的，而是向服务器发请求，委托其删除文件）。

## FileStorage



### CommunicateWithServerStrategy

该策略定义了节点服务器与服务器间通信的策略，就是向服务器注册或更新自己。具体实现时是每隔两秒向服务器发一个心跳包，这个心跳包中包含了本节点服务器的IP、端口、容量等信息。同时在心跳中也可携带其他信息，如通知服务器，本节点新添文件，或什么文件已删除。



### TransWithClientStrategy

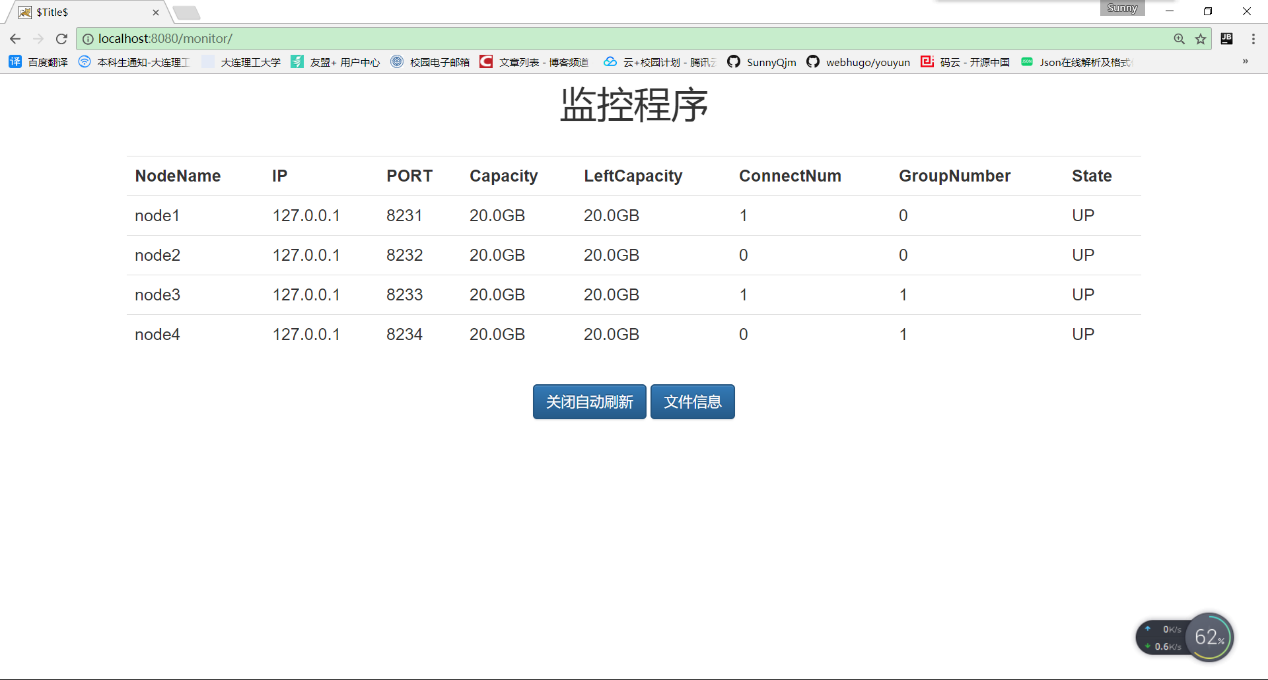
该策略接口定义了如何应对来自客户端和服务器的请求。都是把Socket请求传给service函数，函数里面再根据不同的Code区分不同的请求，做出不同响应。

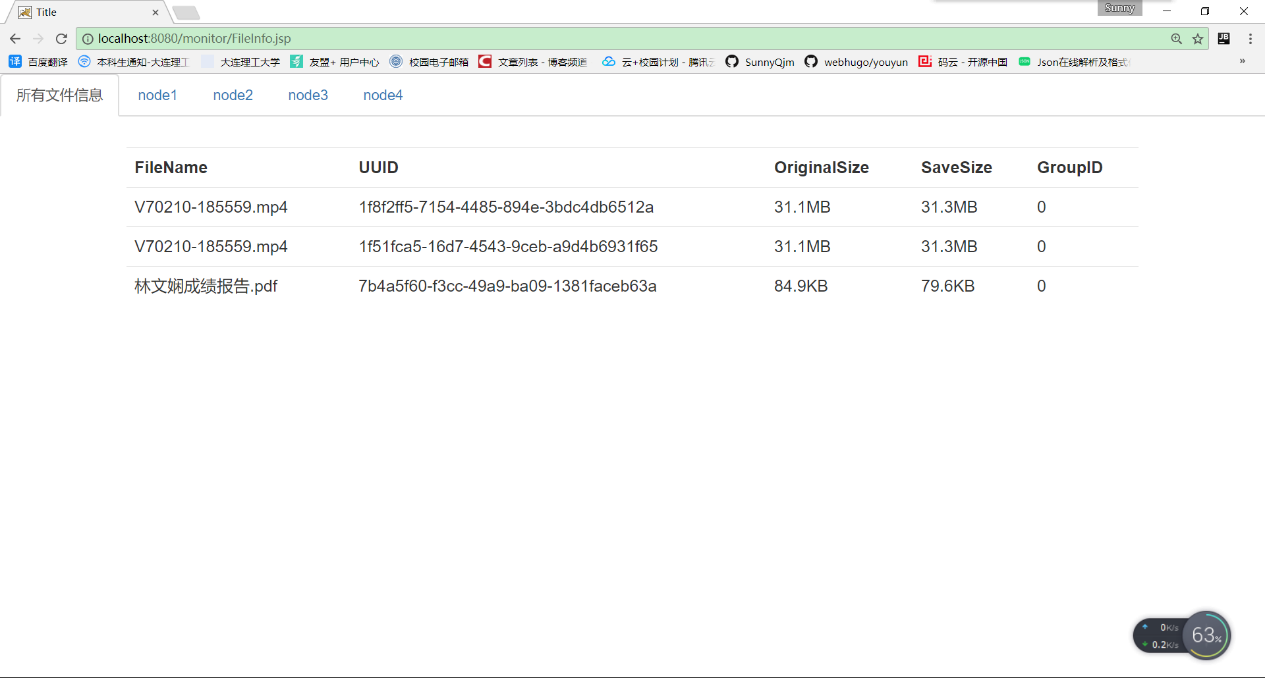


## Monitor

监控程序用JavaWeb实现。原理相对简单，就是在用户用浏览器请求监控程序页面的时候，在JavaWeb服务器端用一个拦截器拦截该请求，并在拦截器中通过向FileServer拉取数据。并将数据通过request对象传给JSP。JSP再根据收到的数据，动态的向表格中添加表项。

整体的界面效果引入了bootstrap框架，使得界面稍微美观一点。



****

# 不足改进

## 问题一及解决方案

问题一：由于在实现过程中每组的节点服务器有两个，在主存储节点宕机的情况下，文件只能传到备份节点服务器上，且此时无法将收到的文件备份到主存储节点。则在下次下载该文件的时候如果这个备份节点宕机，则会下载不到该文件。

解决方案：一种解决方案是每组的服务器多开几个，降低文件下载不到的概率。另一种解决方法是在FileServer记录每个文件保存的情况，每当一个节点服务器接入的时候，就向FileServer询问同组的其它节点的哪些文件自己没有备份，就向这些节点请求文件备份到本节点服务器。

## 问题二及解决方案

问题二：在删除文件时，可能有些存有该文件的节点服务器已经宕机，没有办法删除干净，如果直接删除FileServer中该文件信息的表项，则在节点服务器上就会有冗余文件，成为垃圾，无法访问到。

解决方案：在用户请求删除文件时，FileServer在该表项上标记一下该文件已删除，然后返回给用户删除成功，接着FileServer遍历每个存有该文件的节点，试图删除，定期的便利一遍被标记为删除的文件试图删除它，等到所有存有该文件的节点都删除了该文件，FileServer再将该文件的表项，从Map集合中删除。

# 引用类库文献

【1】Gson:2.8.1

【2】姜老师提供的Tool.java工具类