

哈尔滨理工大学 计算机科学与技术学院

《云计算及虚拟化技术》

项目实践报告

题	目	: 基于 HDFS+HBase+SpringMVC 的校 园网盘系统设计与实现		
班	级	: 大数据 19-1 班		
专	业	: 数据科学与大数据技术		
姓	名	: 马超		
学	号	: 1914050120		
指	导	教	师	: 李金刚
日	期	: 2021-12-22		

目录

一、需求分析	1
二、系统设计	8
(一) 系统中的数据定义	8
(二) 系统的概要设计	10
(三) 系统的详细设计	16
(四) 系统的核心算法	23
三、系统编码及运行	24
(一) 系统开发涉及的软件	24
(二) 系统运行界面及结果	24
四、系统测试	28
五、总结	32
附录 (源代码)	34

一、需求分析

（一）系统开发背景

近几年，随着互联网的快速发展与应用，网络、数字设备、数字化解决方案在家庭、企业、政府、学校中进一步普及，数据存储需求呈爆炸性增长。根据 IDC（Internet Data Center，互联网数据中心）的统计数据，从 2007 年到 2013 年，全球数据总量的年复合增长率高达 52%，尤其是基于文件数据的存储，每年的年复合增长率达到了 69.4%，如果采用传统的存储方式，那么所需的服务器，包括硬件软件和网络等将不可估量，传统的网络存储已不能满足目前庞大的数据存储市场需求，而云存储成为政府、企业、学校降低存储使用成本、提高存储效率、统一安全规范管理的重要选择。

云存储是在云计算概念上延伸和发展出来的一个新概念，它是指通过集群应用、网格技术或分布式文件系统等功能，将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作简单来说，云存储就是将储存资源放到云上供用户存取的一种新兴方案。使用者可以在任何时间、任何地方，透过任何可连网的装置连接到云上方便地存取数据。

云存储更适用于关键业务应用、在线事务处理，满足企业对数据安全性、系统可用性的要求，云存储是建设企业数据中心的首选方案。

在我国，云存储成为网络经济下一个新的增长点已经成为一种共识。随着高校信息化的不断发展，在教学、科研、行政等领域产生了大量的数据信息，数据信息共享交流需求快速增长，高校迫切需要建立安全统一的数据资源存储与共享平台。目前，大多高校校园网已提供了电子邮件和 FTP（File Transfer Protocol，文件传输协议）服务，电子邮件服务可以通过邮件附件的形式进行数据存储和传输，但没有提供统一的数据存储接口，数据使用和管理不方便，FTP 服务虽然能够提供数据上传和下载统一服务接口，但不提供个人存储服务接口。长期以来，在校师生主要通过携带 U 盘、移动硬盘和个人电脑完成数据存储和管理服务，这些存储方式存在着一些缺点：第一，数据携带不方便，必须随身携带存储设备；第二，在存储设备硬件损坏的情况下，会丢失数据；第三，数据存储没有统一的

安全保证机制；第四，同一资源会在不同设备上出现冗余，可能会导致资源的不一致。在数据管理方面，因数据存储容量需求的增加，电子邮件服务器和 FTP 服务器相关设备需要扩展，从而导致建设成本、管理成本、维护成本、能耗成本的快速增加。尽管目前大多高校已经购置大量的服务器设备和存储设备，但这些设备分布在多个机房，设备的硬件类型、软件类型以及安全管理模式不一定相同，导致管理起来十分复杂，不能通过集群模式整合所有的存储系统，并且当前服务器中的数据损坏时，用户便无法使用数据；同时存储服务无法提供数据副本的动态读取和复制功能，无法保证用户数据的可靠性。

正是在这样的背景下，我们小组提出，开发一款云存储系统平台，以满足高校数据量增长的需求，方便在校师生的数据存储，优化存储成本，同时提供一个云计算基础平台。云存储系统平台总体建设思路是采用 B/S 架构，基于 TCP/IP 协议体系实现客户端对云端存储池的文件管理。

（二）系统的研究意义

正如前文所述，云存储相对于传统的存储模式具有很多优势，其中包括扩容和管理更方便、成本更低廉。其中存储的数据相对来说也更安全，而且云存储提供的服务不容易中断。因此研究云存储的相关实现是当前的一大热点。

在次课设的研究当中，我们小组选择分布式文件系统 HDFS，它作为 Hadoop 很重要的一部分。Hadoop 作为 Apache 的开源项目，目的就是利用低廉的硬件设备搭建可扩展、稳定的分布式架构。目前为止，HDFS 在全球已经被广泛使用和研究。它为广大公司和各大科研机构搭建云存储服务提供了很好的参考。据我们所知，国内外现有的一些公司的云存储服务都采用 Hadoop 作为其基础系统，然后根据公司实际业务和需求进行二次开发。我们已知的包括阿里巴巴的云存储、百度的云盘和 Facebook（现更名为 meta）的云存储都是采用的 HDFS 作为基础来进行架设的。目前 HDFS 基本都是用来提供个人网盘的服务用来做研究和探索。针对于实际需求的存储系统还很少见，同时个人网盘虽然功能比较完善，但是并不完全符合高校实际应用的场景。因此对 Hadoop 当中的 HDFS 进行研究是具有很大的实际意义的，可以帮助我们实现云存储平台。我们的目标是搭建一个符合高校实际应用需求的云存储系统。

然而，HDFS 作为云计算的辅助系统，自身还存在着这样那样的不足，包括 NameNode 管理节点单点故障的问题，同时 NameNode 作为管理节点同客户端交互相当频繁，大规模的访问会造成 NameNode 节点不堪重负，从而成为整个系统的性能瓶颈。因此解决这一系列问题，显得具有很多现实意义和实用价值。同时访问 HDFS 也仅仅通过 API 实现，访问方式显得单调并且对使用者的技术门槛很高。因此提供丰富多样的客户端使用 HDFS 搭建的云存储服务显得尤其重要，同时对云存储的普及有很大的帮助。

因此，综上所述，本次课设研究基于 HDFS 的云存储平台具有如下意义：

1. 当前的研究重点领域是 Hadoop 中的 HDFS，因此研究 HDFS 对于云存储相关领域具有重要的指导意义。
2. 针对目前 HDFS 使用门槛高的弊端，提出多种使用方式，其中包括 Web 方式和网络文件系统挂载方式。这对普及 HDFS 分布式文件系统有很大的帮助。
3. 搭建 HDFS 为基础的云存储平台，对实现基于 HDFS 的相关应用具有指导意义和探索意义。

（三）系统的需求分析

1.3.1 用例分析

基于 HDFS 的高校云存储系统面向的用户是高校师生。在云存储系统中将系统的功能分为用户、文件管理、文件夹管理、笔记等模块。用户模块主要包含用户的注册、登录等功能，文件管理包含文件上传、下载、分享、重命名、预览、删除等功能，文件夹管理主要包含创建、删除、重命名等功能，笔记主要包含添加、编辑、删除等功能。

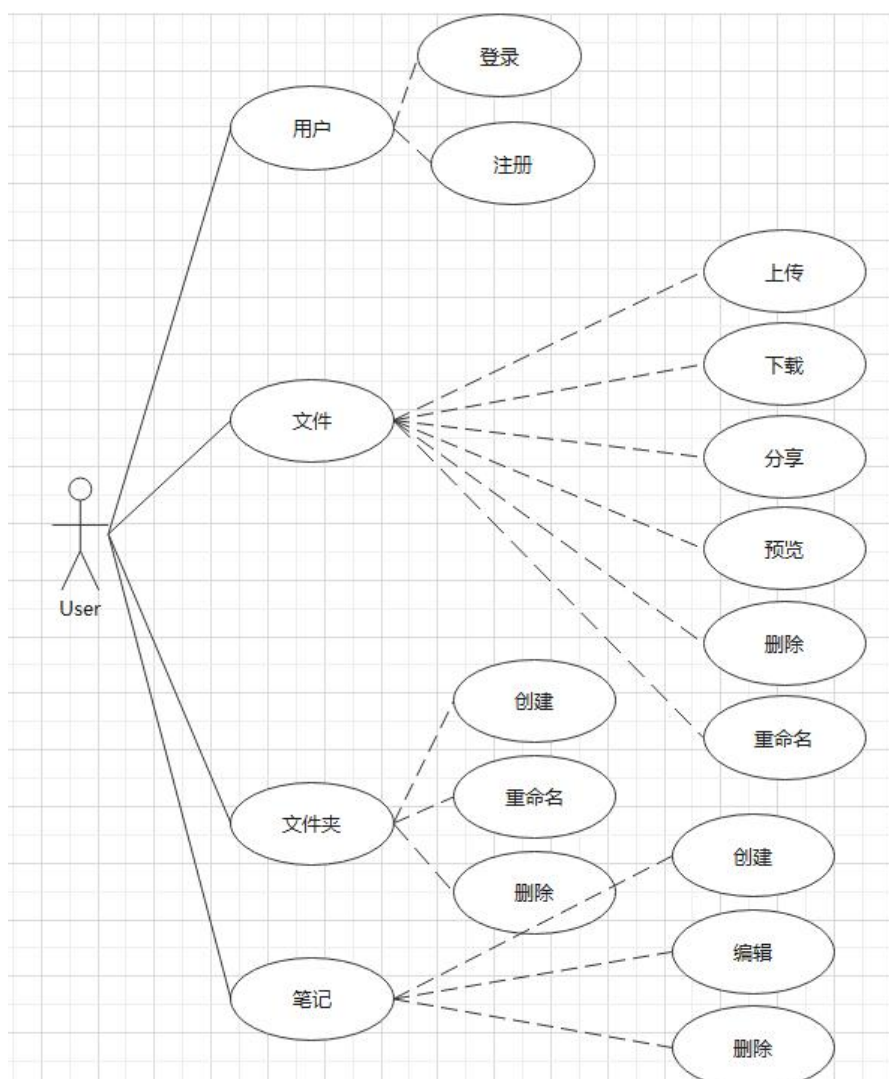


图 1.1

1.3.2 功能需求分析

基于 HDFS 的高校云存储系统主要功能分析如下：

(1) 用户管理功能

用户管理功能模块应该实现用户的登录和注册功能，利用 Hbase 对用户信息进行保存。在用户登录后能够对当前账号完成一些基本的账号管理功能

(2) 文件夹管理功能

文件夹管理功能应包含文件夹的创建、删除、重命名等基本功能，并能够简单查看文件夹的信息。应该能够将 web 页面的操作同步到 HDFS 中。

(3) 文件管理功能

文件管理功能模块包含文件的上传、下载、分享、删除、重命名、预览等

功能。上传功能实现将文件从本地上传到 HDFS 文件系统中，并能够在 web 页面中进行显示；下载功能实现将文件从 HDFS 中下载到本地；分享功能实现将用户保存在云存储系统中的文件分享给好友；删除功能实现将文件从 HDFS 中删除；重命名功能实现对保存在云存储系统中的文件进行重命名，同时将重命名同步到 HDFS 中；预览功能实现了对某些格式的文件进行预览，如 txt、pdf 等文档格式的文件。

（4）笔记功能模块

笔记功能模块实现了每个用户能够在导航栏中的笔记模块添加笔记、编辑已有笔记、以及删除笔记的功能。用户添加的笔记只能由自己进行查看和编辑。

1.3.3 非功能需求分析

（1）安全需求

在登录模块，系统设计了反暴力破解机制，防止用户密码被暴力破解，后台数据存储应该能够进行记录日志，系统支持通过 web 页面方式管理系统。未登录用户无法获取存储在系统中的文件、数据。

（2）性能需求

系统响应时间：访问页面的响应时间不超过 5 秒；对数据库特定检索的响应时间不超过 20 秒。操作响应时间：系统对于用户查询操作不得超过 20 秒。并发用户数达到门限时，系统应满足响应时间的要求，并可以正常操作，系统平均负荷应保持在 50%以下（CPU、内存占有率、I/O 负荷），系统峰值负荷应小于 70%。存储系统应达到电信级产品的可用性，正常工作时间达到 99.9%。系统 7*24 小时不间断工作，平均故障间隔（MTBF）应大于存储设备的平均故障时间。

（3）系统的可扩展性和可维护性

随着用户需求的发展，肯定会出现一些新的功能需求，那么就需要保证系统具有良好的扩展性，避免在下次开发时重复编写基础功能或者对系统重新架构，提高工作效率，同时便于多平台的代码调用和系统融合开发。使得系统不仅能够应用于 Web 端，也适用于 PC 客户端、移动客户端的开发。同时，应当考虑当需要加入新的节点时，系统应尽量在不修改代码的

情况下完成扩展，同时保证加入新节点不会影响系统的正常使用。可维护性指当系统的节点出现故障时，系统能提供恢复功能，在排查节点故障后，将节点恢复到故障出现前的某个时间的状态。

（4）系统的友好性

应为系统设计一个简洁、逻辑清晰、易于操作的 UI，能够让用户不需要耗费时间和精力来学习系统的使用方法，只要拥有账号就能保证能够使用系统的所有功能。

（5）系统的可靠性

系统对于搭建集群使用的机器没有严格的要求，这就需要设计合理的容灾机制，保证当任何一个节点出现故障时，系统都能保证文件数据不丢失，每个文件数据都有相应的备份。并通过备份策略存储到不同的节点中。

1.3.4 系统开发的必要性分析

目前校园网没有提供统一的存储方式，用户主要通过随身携带存储介质实现数据的移动和使用，使用不便、安全性不高、不易共享。本项目的目标是为在校学生和教师提供统一的远程存储池，用来存储用户的数据，通过浏览器方式获取数据存储管理服务。同时，通过云存储系统对文件进行管理，一定程度上降低了使用第三方存储介质时被感染木马的风险。

1.3.5 系统开发的可行性分析

①技术可行性：使用 Java 语言、HDFS 文件系统、HBase 数据库、Java Web 技术，开发云存储系统，用于满足高校师生的文件管理，该系统主要功能模块有用户注册登录模块、文件管理模块、文件夹管理模块、笔记管理等。每一个主要功能模块都由数据中心模块和业务模块两大部分组成。数据中心模块主要由 HBase 数据库的数据模型提供技术支撑，而业务模块主要由 Java 语言和 HDFS 的 API 提供技术支撑，前端由 html、css 和 JS 进行开发，后端利用 java 轻量级应用框架 SpringMVC 进行开发。

②法律可行性：遵守各项法律和各项规定；

③市场可行性：基于 HDFS 的高校云存储系统符合高校师生的文件管理需要。

二、系统设计

（一）系统中的数据定义

本云盘的数据采用分布式列族 nosql 数据库 Hbase 进行数据的存储。存储的信息包括各种事务的 id（包括分享事务，关注 id 以及用户的全局 id），用来唯一标识个体及其事务、按照用户 id 索引的用户信息（包括用户注册邮箱、用户名以及用户密码）、按照用户名进行索引的用户信息（包括用户 id）、用户存储的记事本（包括用户 id 以及用户的记事本内容）、用户关注关系（包括记录关注的用户信息）、用户被关注关系（包括记录该用户被那些用户关注的 id）、用户分享记录（包括用户分享了哪些文件给哪些用户）。将上述存储的信息设计为 hbase 中的表格，如下进行详细说明：

gid 表主要用来保存各种**事务**和个体的 id，用来唯一识别。这里的表分别存储记事本的 id，用户的 id，以及分享事务的 id，对应只在同名的列限定符上取值，所有的列限定符都在 gid 的列族之下。

表 2-1 gid 表

行键	列族	列限定符
bookid、gid、shareid	gid	bookid、gid、shareid

id_user 表主要以用户的 id 为索引的行键，列限定符包括 email、name、pwd，都在 user 的列族之下，主要用来按用户 id 索引信息使用。

表 2-2 id_user 表

行键	列族	列限定符
gid	user	email、name、pwd

user_id 表主要用来以用户的用户名为**索引**的行键，列限定符包括 id，都在 id 的列族之下，主要用来按用户名索引信息使用。

表 2-3 user_id 表

行键	列族	列限定符
username	id	id

email_user 表主要用来按照用户的邮箱进行索引的行键，列限定符 userid，在 user 的列族之下，主要用来按照用户邮箱进行索引使用。

表 2-4 email_user 表

行键	列族	列限定符
email	user	userid

share 表主要用来记录每个用户将文件分享给了哪个用户，并记录分享的用户名称、所分享文件的 hdfs 目录、所分享文件的路径以及分享文件的类型。主要用 gid 中的 shareid 来作为索引使用的行键列限定符包括 dir、path、ts 以及 type，都在列族 content 中。

表 2-5 share 表

行键	列族	列限定符
shareid	content	ir、path、ts、type

shared 表主要用来记录哪些用户是被分享的，行键为 shareid 加被分享的 id，列族为 shareid，存储的值就是 shareid。

表 2-6 shared 表

行键	列族	列限定符
shareid+gid	shareid	shareid

follow 表主要用来记录用户之间的关注关系，行键为关注人的 user_id，列限定符为关注人的姓名，值为被关注人的姓名。

表 2-7 follow 表

行键	列族	列限定符
shareid+gid	shareid	shareid

followed 表主要用来记录被关注关系，行键为两个用户 user_id 的组合，列限定符为关注人的 userid，在 userid 的列限定符下。

表 2-8 followed 表

行键	列族	列限定符
userid+userid	userid	

book 表主要用来记录用户的记事本信息，行键为用户 id 与 bookid 的组合，列族为 content，值为记事本的内容。

表 2-9 book 表

行键	列族	列限定符
userid+bookid	content	

文件信息主要以目录的形式存储在 hdfs 中，包含每个用户的文件数据存储结构以及相应的文件，即利用名称节点存储用户存储文件的元数据信息（网盘目录），利用数据节点存储用户存储的文件。

（二）系统的概要设计

2.2.1 设计目标

通过建设校园网盘服务系统，为学生和教师提供非结构化存储空间，统一管理海量存储空间，保障数据文件的安全性，便于教学资料的相互交流传输，实现以下目标：

通过设计海量存储系统架构，形成基于分布式云存储的海量存储架构，实现教务办公文件及师生个人文件海量存储。基于分布式云存储的海量存储架构分为应用系统层及员工访问层。云存储层由存储服务器集群组成，存储服务器集群由低成本 X86 服务器组成，存储用户的实际数据，是整个云硬盘系统的存储资源提供者。

师生访问使用云存储网盘服务系统主要通过网页浏览器访问模式，既适用于高带宽的城域网接入，也适用于 Internet 网接入。

实现随时随地访问，数据统一存放在云端，个人凭账号及密码随时随地访问，为用户提供一站式、安全可信、方便易用的分布式存储服务，并提供创新的应用客户端，方便用户发现、使用和管理各类数据文件，把存储在 PC、手机、网络的资产迁移到分布式存储空间，并实现多终端同步及多屏互动。

具有一键共享功能，按照实际需求，实现 " 好友 " 之间文件一键共享，突破原有移动存储、网络传输、移动设备等固有数据拷贝模式限制，一份云端数据，多 " 好友 " 共同访问，减少数据共享成本。

任何一个系统的建设都必须遵守一些设计原则，这些原则包括安全性原则、系统化原则、实用性原则、稳定性原则和可维护性原则等等。本项目根据设计需

要提出了以下设计原则：

安全性原则：应将系统和信息的安全保密作为系统建设的基础，系统建设采用网络隔离技术，在前端服务器与后端服务器之间提供严格的安全保密措施，巩固和加强系统整体安全性。

系统化原则：为适应业务需求的发展和变化，系统在设计上应基于先进的技术架构，采用系统化的思想开发，使系统能进行快速的功能扩展和集成完善。

实用性原则：系统必须面向最终用户，以最终用户为主，做到操作简单方便，界面友好，业务流程合理，真正符合最终用户的实际需要，且能经过短期试用和培训，可在应用单位内部署使用；

稳定性原则：系统应采用成熟的高可用和数据备份技术，来确保系统业务连贯、数据可靠。同时系统在底层设计上应降低各类应用服务间的耦合度，提供服务分离的部署机制，避免因单个服务的故障影响整个系统的正常运巧。

可维护性原则：系统具有良好的管理、监控、故障分析和处理能力，可集中统一维护，在功能划分和设计时，使各模块尽可能相对独立、减少相关性，便于维护。

2.2.2 系统总体架构设计

系统总体架构自顶向下主要包括 4 个层次，包括：客户界面层、web 前后端系统、业务逻辑层和数据存储层。



图 2-1 系统总体架构

客户界面层：用户可以通过自助注册，注册用户经系统授权具有对云存储访问的权限，同时具有个人文件上传、下载、文件夹共享等权限。用户注册通过浏览器登录使用云存储服务，完成基本文件存储管理、个性化背景设置等功能。

Web 前端系统：文件管提供统一的文件存取接口对文件进行操作，包括保存、删除、查询。共享管理；用户可在其中进行文件操作管理，包括创建文件夹、创建文件、删除文件夹、删除文件、多线程上传文件、多线程下载文件、文件夹重命名、文件重命名、文件夹共享、我的共享等管理功能。身份认证，对接入云存储平台的业务系统需提供系统身份合法性验证接口。

业务逻辑层：主要是提供分布式列族数据库的组件、在分布式存储系统中进行文件的各项操作；对于文件进行共享的组件以及用户之间相互关注的组件；用户登录验证的组件。

数据存储层：支持 TB 级的海量数据存储；满足超大用户量并发访问，突破性能瓶颈；部署在廉价的 P C 服务器集群上；具有商可靠性，通过存放于不同结点的副本来保证单个结点失败时的数据安全。

2.2.3 系统技术架构

系统采用 hadoop 的文件系统 hdfs 作为底层的分布式数据存储，Hbase 用于维护云盘系统中的用户管理信息，zookeeper 用于监视和管理 hbase 的活动，上层的 web 框架采用 SpringMVC 架构进行设计,分为 controller、model 以及 view，用于具体云盘文件操作、数据维护以及用户管理的逻辑的实现

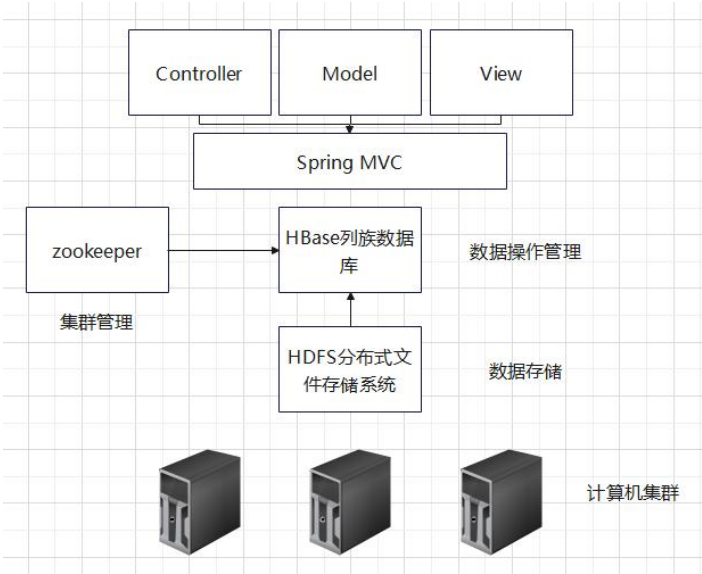


图 2-2 系统技术架构

2.2.3.1 Spring MVC 架构

Spring 的 web 框架围绕 DispatcherServlet 设计。DispatcherServlet 的作用是将请求分发到不同的处理器。Spring 的 web 框架包括可配置的处理器（handler）映射、视图（view）解析、本地化（local）解析、主题（theme）解析以及对文件上传的支持。Spring 的 Web 框架中缺省的处理器是 Controller 接口，这是一个非常简单的接口，仅包含 ModelAndView handleRequest(request, response) 方法。可以通过实现这个接口来创建自己的控制器（也可以称之为处理器），但是更推荐继承 Spring 提供的一系列控制器，比如 AbstractController、AbstractCommandController 和 SimpleFormController。注意，需要选择正确的基类：如果没有表单，就不需要一个 FormController。这是和 Struts 的一个主要区别。

Spring Web MVC 允许使用任何对象作为命令对象（或表单对象）- 不必实现某个特定于框架的接口或从某个基类继承。Spring 的数据绑定相当灵活，例如，它认为类型不匹配这样的错误应该是应用级的验证错误，而不是系统错误。所以你不需要为了保证表单内容的正确提交，而重复定义一个和业务对象有相同属性的表单对象来处理简单的无类型字符串或者对字符串进行转换。这也是和 Struts 相比的另一个重要区别，Struts 是围绕 Action 和 ActionForm 等基类构建的。

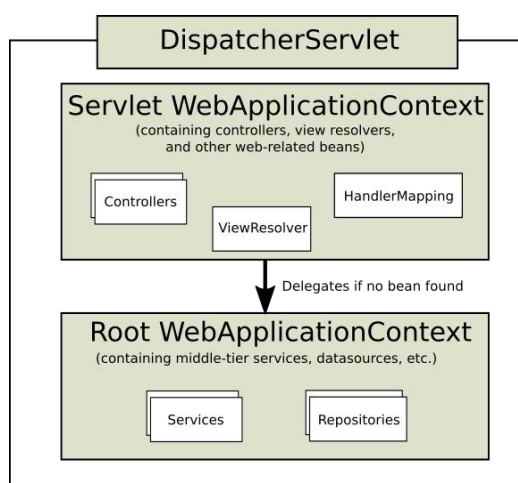


图 2-3 系统技术架构

和 WebWork 相比，Spring 将对象细分成更多不同的角色：控制器（Controller）、可选的命令对象（Command Object）或表单对象（Form Object），

以及传递到视图的模型（Model）。模型不仅包含命令对象或表单对象，而且也可以包含任何引用数据。相比之下，WebWork 的 Action 将所有的这些角色都合并在一个单独的对象里。虽然 WebWork 的确允许在表单中使用现有的业务对象，但是必须把它们定义成相应的 Action 类的 bean 属性。更重要的是，在进行视图层（View）运算和表单赋值时，WebWork 使用的是同一个处理请求的 Action 实例。因此，引用数据也需要被定义成 Action 的 bean 属性。这样一个对象就承担了太多的角色。

Spring 的视图解析相当灵活。一个控制器甚至可以直接向 response 输出一个视图（此时控制器返回 ModelAndView 的值必须是 null）。在一般的情况下，一个 ModelAndView 实例包含一个视图名字和一个类型为 Map 的 model，一个 model 是一些以 bean 的名字为 key，以 bean 对象（可以是命令或 form，也可以是其他的 JavaBean）为 value 的键值对。对视图名称的解析处理也是高度可配置的，可以通过 bean 的名字、属性文件或者自定义的 ViewResolver 实现来进行解析。实际上基于 Map 的 model（也就是 MVC 中的 M）是高度抽象的，适用于各种表现层技术。也就是说，任何表现层都可以直接和 Spring 集成，无论是 JSP、Velocity 还是其它表现层技术。Map model 可以被转换成合适的格式，比如 JSP request attribute 或者 Velocity template model。

2.2.3.2 校园云盘系统的 springmvc 架构设计

校园云盘的控制器（controller）包括基本 hdfs 与 hbase 实例的控制器（BaseController），负责生成 hdfs 和 hbase 的两个实例；继承 BaseController 的主要负责云存储的 CloudController 包括实现创建目录、上传文件、删除文件及文件夹、复制文件及文件夹、重命名文件及文件夹、查看文档、文件下载、文件分享、获取分享、获取被分享、记事本列表和新增记事本等功能；负责用户登录管理的 LoginController，负责用户的验证登录、注销、注册和初始化；负责用户管理继承 BaseController 的 UserController，包括验证用户注册并建立 hdfs 中属于该用户的目录、获取关注的用户、关注用户、取消关注的功能。

校园云盘的模型（Model）包括操作 HDFS 的 HdfsDB，负责在 hdfs 中进行创建目录、上传文件、删除文件及文件夹、复制文件及文件夹、重命名文件及文件夹、查看文档、文件下载、文件分享、获取分享、获取被分享、记事本列表和新

增记事本进行 Hdfs；操作 Hbase 的 HbaseDB，负责在 hbase 中获取所有表、删除所有表、创建表、删除表、删除表、根据 row 删除数据等操作；利用 openoffice 转换 swf 格式进行消息预览的 OpenOfficePDFConverter，将 txt 等文件转换为浏览器可以预览的格式；作为基本工具的 BaseUtils，负责规范文件尺寸、判断非空等功能；处理时间的基本工具 DateUtil 类，负责规范日期格式；负责处理文件的 FileUtils 类，用来获取文件的前后缀和复制文件；用于数据交互的 Json 类，用来进行网页之间的数据交互；用于视图层传输的 vo 对象，HdfsVo、FileSystemVo、HdfsVo、ShareVo、bookVo 等。

校园云盘的视图（View）包括云盘显示云文件列表的 list.jsp，显示分享界面的 share.jsp，显示上传界面的 upload.jsp，显示记事本的 book.jsp，显示关注页面的 follow.jsp 以及登录界面的 login.jsp 等。

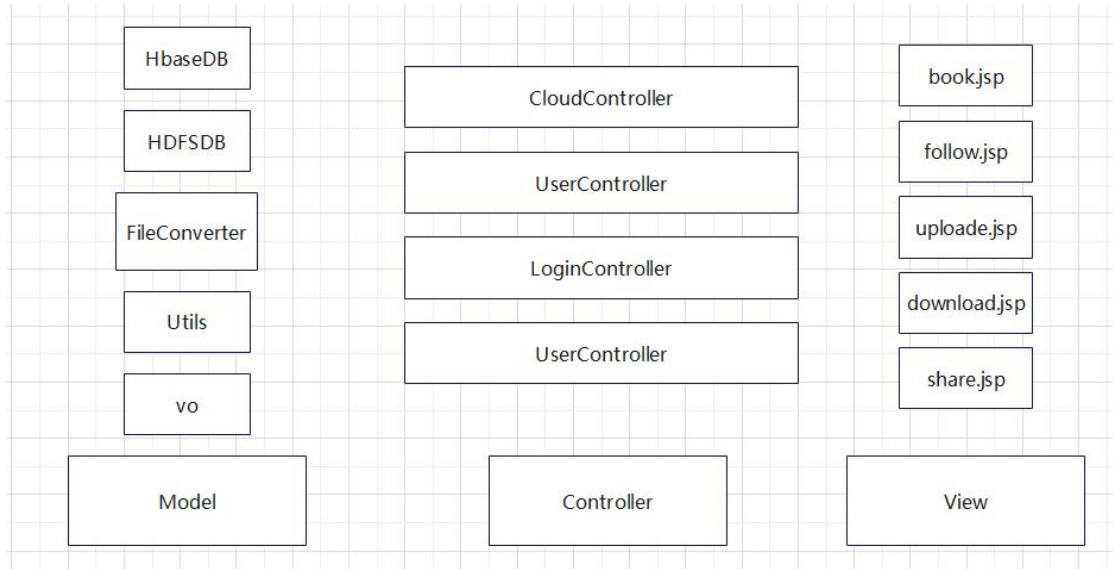


图 2-4 springmvc 架构设计

2.3 系统工作流程

用户关注模块直接在 Hbase 数据库中添加对应的用户关注关系即可，Hbase 直接将表分布式存储在 HDFS 中；文件分享模块直接在 Hbase 数据库中 添加分享关系,Hbase 直接将表分布式存储在 HDFS 中；文件上传模块通过 HDFS 文件操作模块将文件上传至 hdfs 中，并将路径保存至 Hbase 中；文件下载模块通过 HDFS 文件操作模块将文件下载至 hdfs 中。

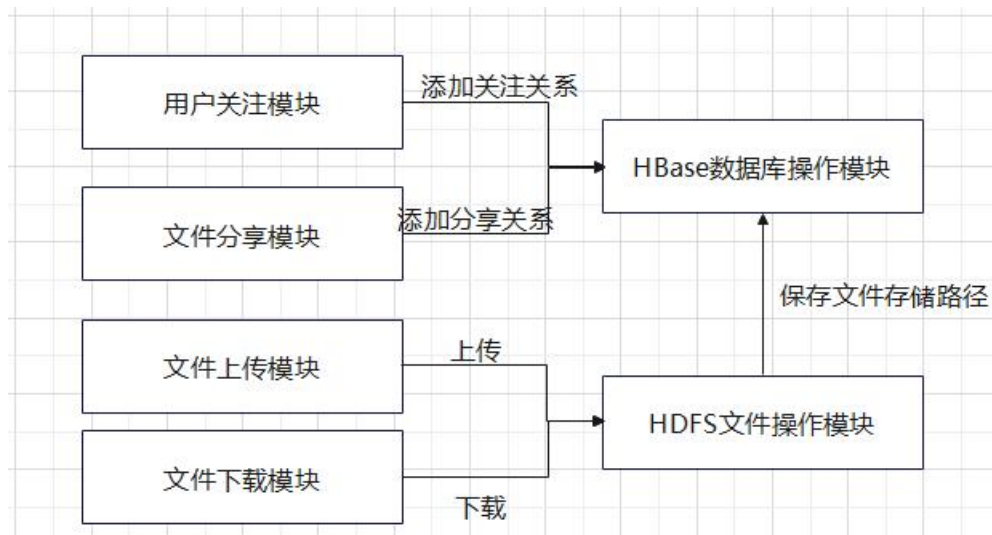


图 2-5 系统工作流程

（三）系统的详细设计

系统基于 HDFS 以及 Hbase 数据库，使用 SpringMVC 框架进行开发。系统创建了一个 BaseController 用于获取 HDFS 以及 Hbase 连接。另外还创建了 3 个控制器 CloudController、UserController 以及 LoginController 继承 BaseController 来完成文件管理、用户管理以及登录注册等的功能。

2.3.1 HDFS 文件操作模块；

HDFS 文件操作主要负责对上传云盘的文件或者文件夹进行上传、下载、移动、删除等操作，主要利用 java api 完成。

HDFS 操作模块在网盘中的体现就是网盘中的文件列表以及对应的对于各个文件的操作。文件列表部分需要先验证用户是否登录，如果未登录，则返回登录页面；如果登录则将用户的姓名传递给视图层，视图层转发对应用户显示网盘文件的列表的请求，会将 hdfs 中对应用户的文件列表以树形的方式展出会，具体的流程图如下图所示：

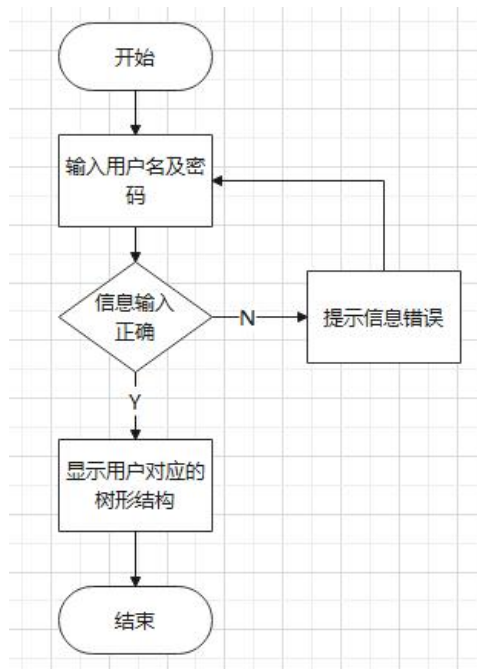


图 3-1 网盘文件展示流程图

Hdfs 对于文件的上传操作和下载操作需要解决的问题就是找到进行上传与下载对应文件夹，其他的操作也同理，用以下的数据流图来表示。

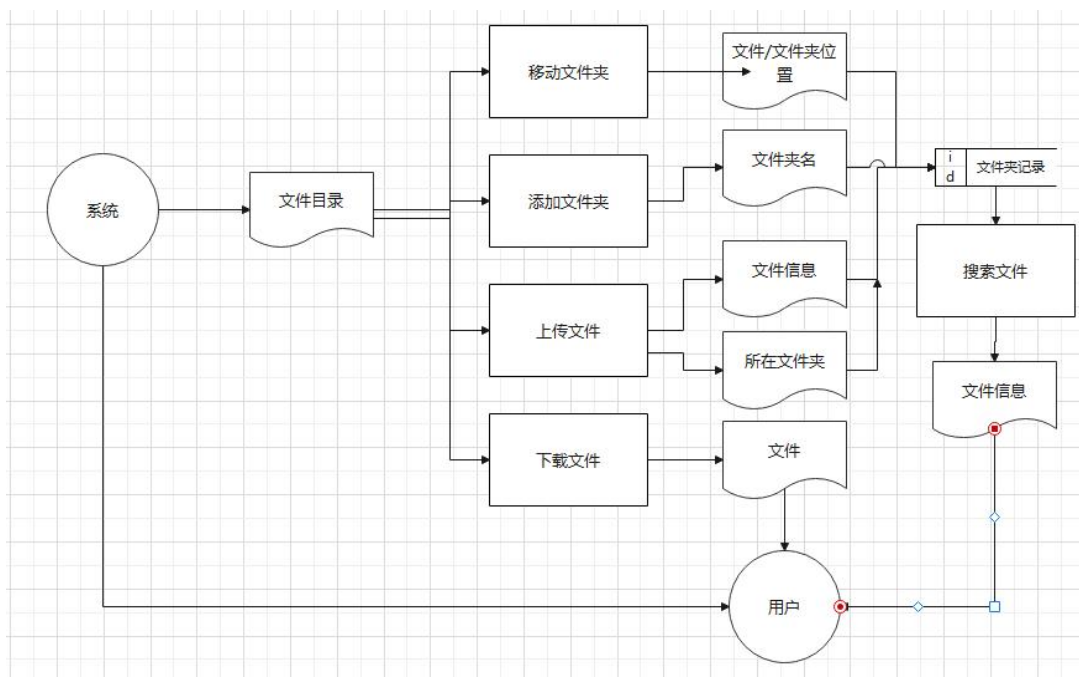


图 3-2 文件操作数据流图

2.3.2 Hbase 数据库操作模块

Hbase 数据库操作模块主要负责对数据定义中的表格进行数据的增删改查，主要利用 Hbase java api 完成，主要用到的 java 类如下图所示。

表 3-1 hbase java 类

java 类	HBase 数据模型
HBaseAdmin	数据库 (DataBase)
HBaseConfiguration	
HTable	表 (Table)
HTableDescriptor	列族 (Column Family)
Put	列修饰符 (Column Qualifier)
Get	
Scanner	

在云盘中主要涉及到 hbase 数据库的增删改查,实现的功能接口如下图所示:

表 3-2 hbase 操作模块实现接口

	实现接口
HbaseDB	<pre> public static TableName[] listTable() throws Exception; public static void deleteAllTable() throws Exception; public static void createTable(String tableName,String[] fams,int version) throws Exception; public static void delTable(String tableName) throws Exception; public static long getGid(String row) public static void add(String tableName, String rowKey, String family, String qualifier, String value) public static void deleteRow(String tableName, String[] rowKey) public static void deleteColumns(String tableName,Long rowKey,String family, Long qualifier) public static void deleteRow(String tableName,Long rowKey01,Long rowKey02) public static long getIdByUsername(String name) public boolean checkUsername(String name) public static String getUserNameById(long id) public boolean follow(String oname,String dname) public boolean unfollow(String oname,String dname) public Set<String> getFollow(String username) </pre>

2.3.3 用户关注模块

用户关注模块主要用来实现云盘中的用户互相关注功能，具体的实现核心代码如表所示，用户关注、关注列表获取以及取消关注如表 2-1 所示，

表 3-3 用户功能模块

UserController	负责控制根据用户模型出发用户试图实现注册、关注等功能
getFollowlist	<pre>public String getFollow(String ids,String dir,String types,HttpSession session,Model model) throws Exception { String name = (String) session.getAttribute("username"); if (name!=null) { model.addAttribute("follows", db.getFollow(name)); } model.addAttribute("dir", dir); model.addAttribute("ids", ids); model.addAttribute("types", types); return "cloud/share_getfollow"; }</pre>
Follow	<pre>public Json follow(String username,HttpSession session) throws Exception { Json json = new Json(); String name = (String) session.getAttribute("username"); if (name!=null && BaseUtils.isEmpty(username)) { try { db.follow(name, username); json.setSuccess(true); } catch (Exception e) { json.setMsg("关注失败"); e.printStackTrace(); } } return json; }</pre>
unFollow	<pre>public Json unfollow(String username,HttpSession session) throws Exception { Json json = new Json(); String name = (String) session.getAttribute("username"); if (name!=null && BaseUtils.isEmpty(username)) { try { db.unfollow(name, username); json.setSuccess(true); } catch (Exception e) { json.setMsg("取消失败"); e.printStackTrace(); } } return json; }</pre>

2.3.4 用户注册登录模块;

用户注册和登录模块主要实现用户的登录、注册。登录需要输入用户名和密码,如果信息输入正确则跳转到主页,如果信息输入错误则跳转到登录页面;注册需要填写用户信息,如果信息输入正确,则返回登录页面,如果错误则提示注册失败。

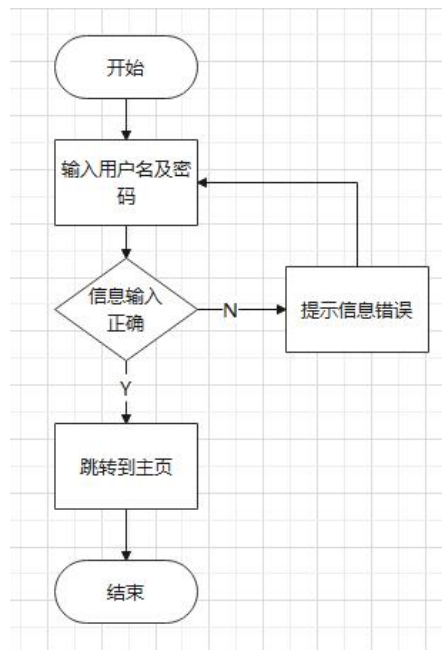


图 3-3 用户登录流程图

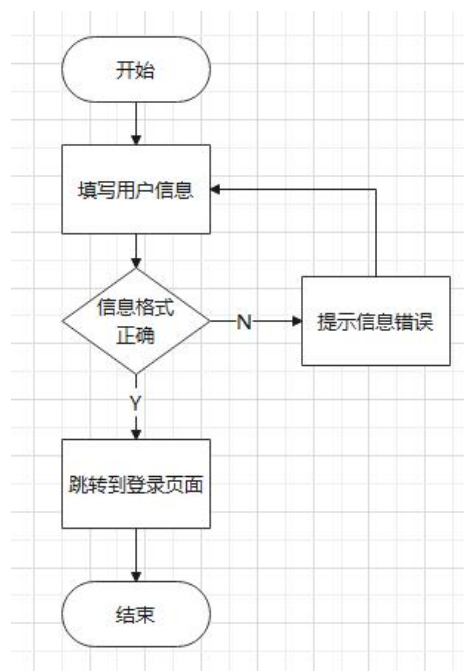


图 3-4 用户注册流程图

2.3.5 web 前后端模块

该模块采用 SpringMVC 架构。创建了 3 个控制器 CloudController、UserController 以及 LoginController 来完成文件管理、用户管理以及登录注册等的功能。

在 CloudController 中主要包含了文件上传、下载、分享、删除、移动、重命名等子控制模块。

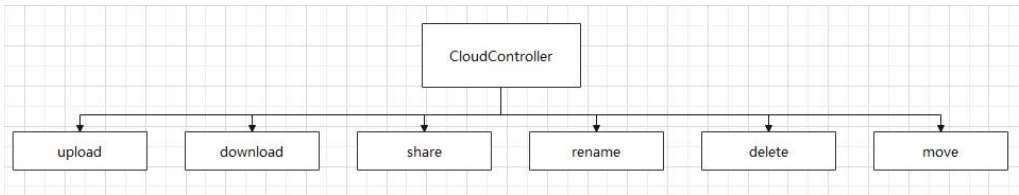


图 3-5 CloudController

表 2-4 CloudController 组成

模型层主要包含 HDFS 表示类、Menu 表示类、FileSystem 表示类、Share 表示类、HdfsDB 以及 HbaseDB。在 HdfsDB 类中封装了将用户对文件的操作同步到 HDFS 中的各种方法，包括上传、下载、分享、删除、重命名等。在 HbaseDB 类中主要封装了创建 Hbase 数据库连接、创建表、删除数据、添加数据等方法。四个表示类结构如表 2-5 所示：

表 3-4 vo 类结构

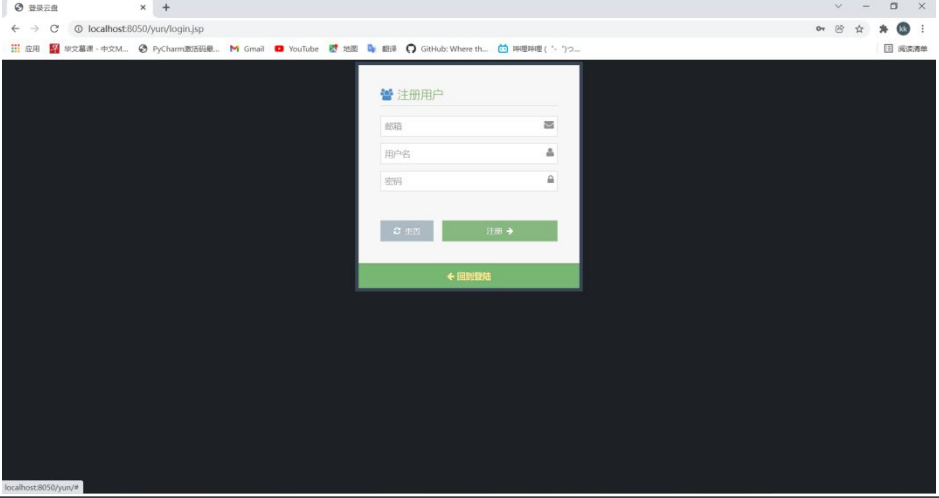
类名	封装属性
FileSystemVo	private Long id; private String name; private String dir; private String pdir; private String type; private String date; private String size; private String viewflag="N"; private String ids; private String names; private String dirs; private String namep;
HdfsVo	private String id; private String name; private String type; private String createDate;
Menu	private String id; private String text;

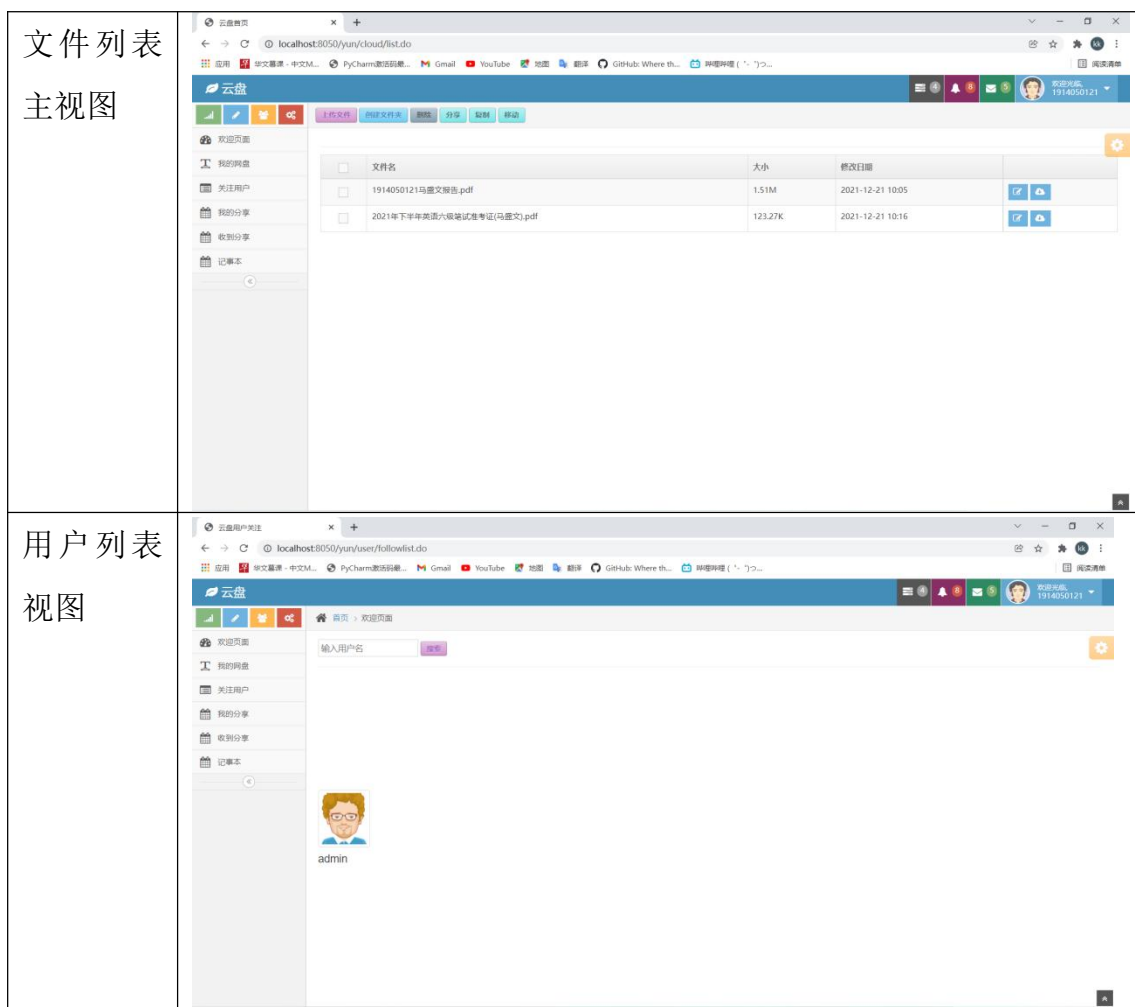
	<pre>private String name; private String iconcls; private String url; private String state = "closed"; private String pid; private String pname; private Map<String, Object> attributes; private String createtime;</pre>
ShareVo	<pre>private String shareid; private String path; private String ts; private String dir; private String type;</pre>

对于上述四个类中的每个属性，系统都实现了对应的 `get` 方法以及 `set` 方法方便调用以修改这些属性。

本次课设研究的云存储系统的视图层主要设计了注册/登录视图、云存储系统的文件列表、用户列表等。

表 3-5 视图层结构

功能名称	前端页面设计
登录/注册视图	



(四) 系统的核心算法

在显示网盘文件的时候采用了按照字典序排序的方法显示，因此采用了快速排序的算法，伪代码如下图所示：

```
quicksort(A, lo, hi)
  if lo < hi
    p = partition(A, lo, hi)
    quicksort(A, lo, p - 1)
    quicksort(A, p + 1, hi)
  partition(A, lo, hi)
    pivot = A[hi]
    i = lo //place for swapping
    for j = lo to hi - 1
      if A[j] <= pivot
        swap A[i] with A[j]
        i = i + 1
    swap A[i] with A[hi]
    return i
```

图 3-6 快速排序伪代码

三、系统编码及运行

（一）系统开发涉及的软件

表 4-1 开发软件

分布式数据存储	HDFS
列族数据库	HBase
服务器组件	Tomcat 8.5
开发工具	Eclipse jee

（二）系统运行界面及结果

要求有运行结果截图展示，可以分模块说明。如图 1 所示为系统登录界面。

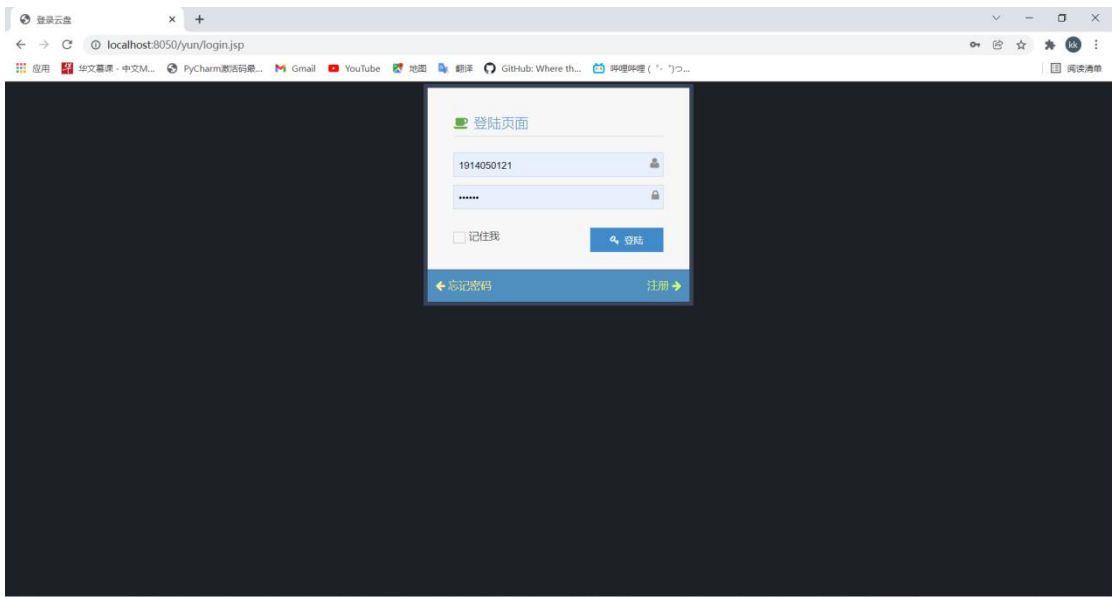


图 3-1 系统登录界面

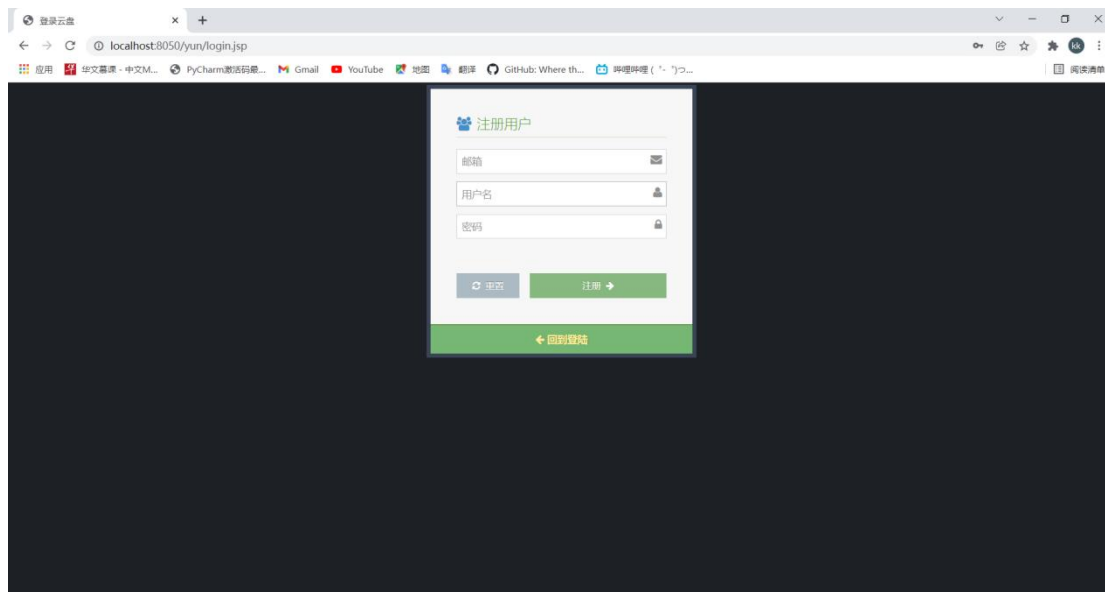


图 3-2 系统注册界面

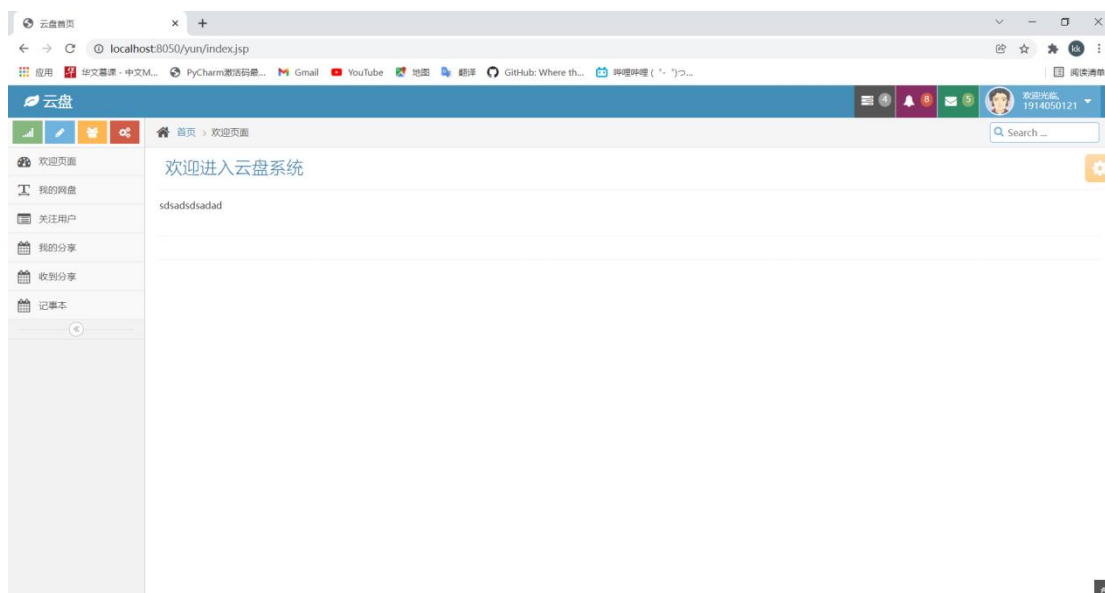


图 3-3 系统主页

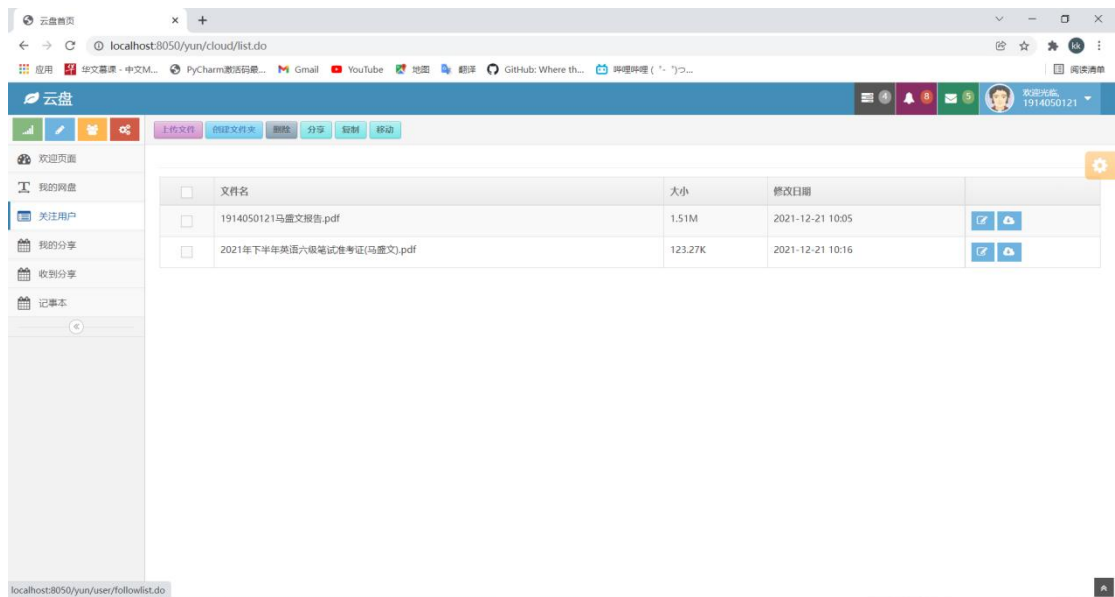


图 3-4 文件列表

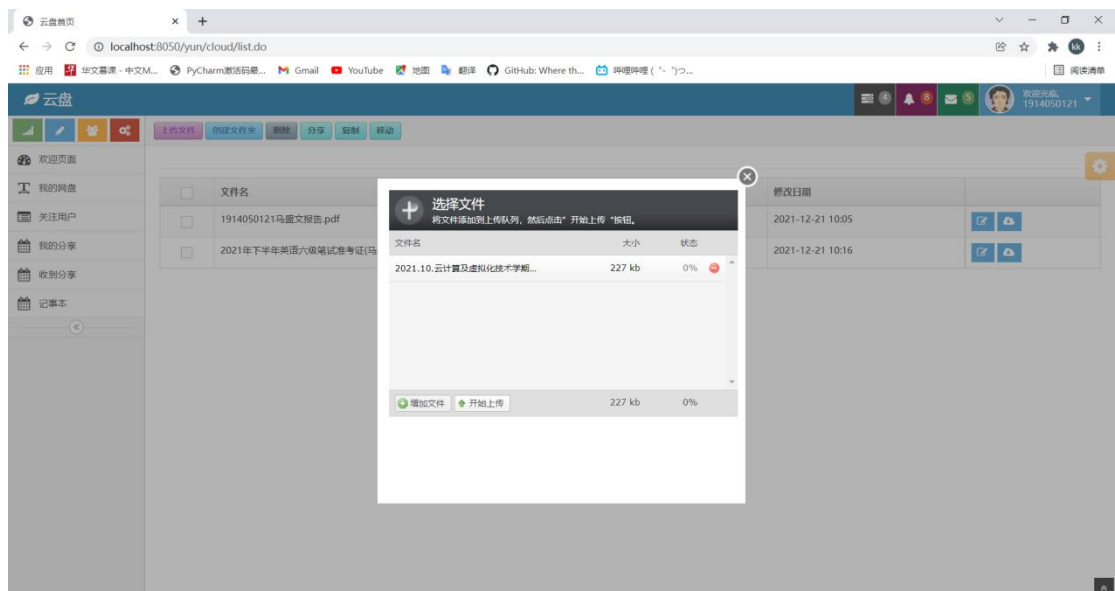


图 3-5 上传文件

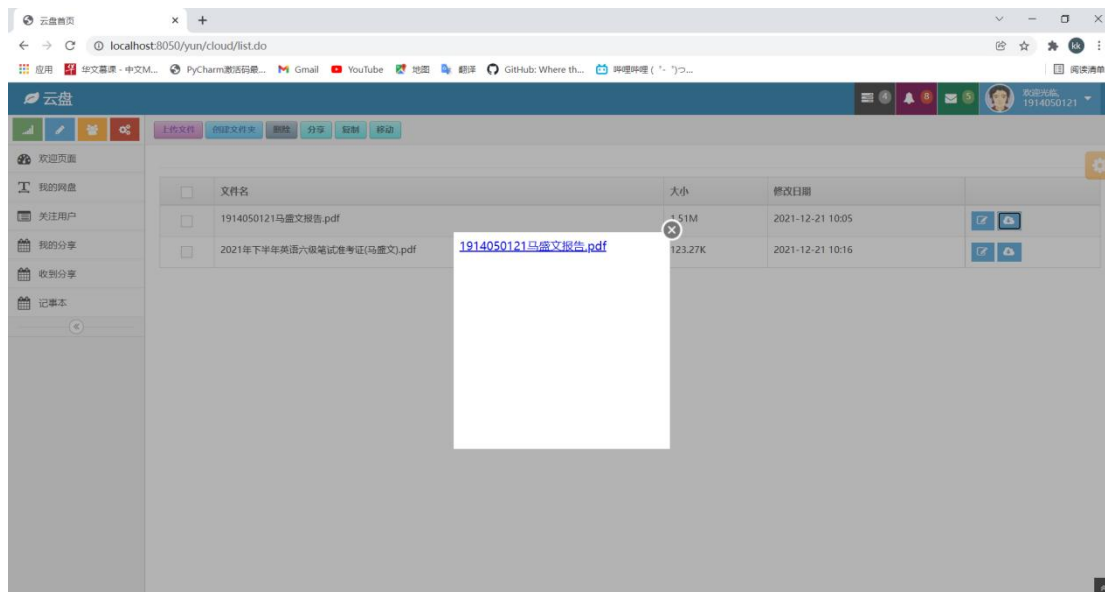


图 3-6 下载文件

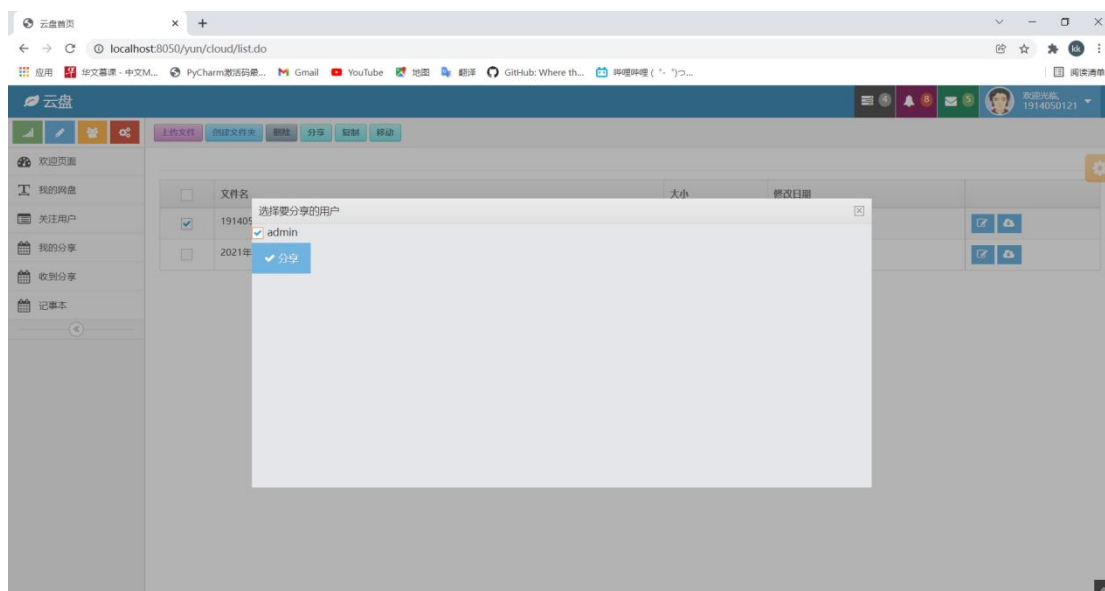


图 3-7 分享文件

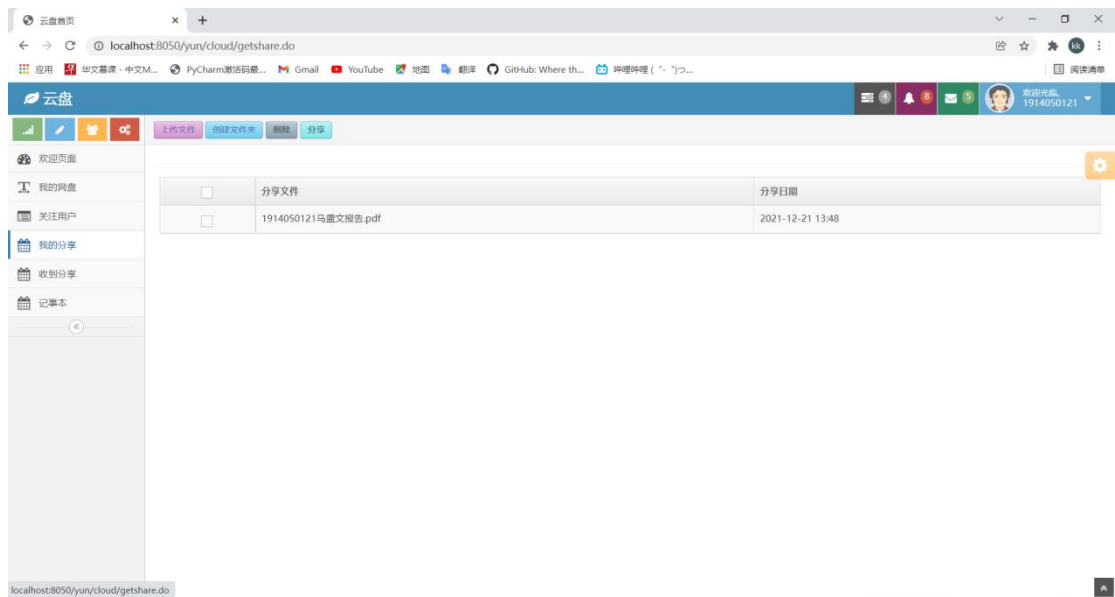


图 3-8 分享列表

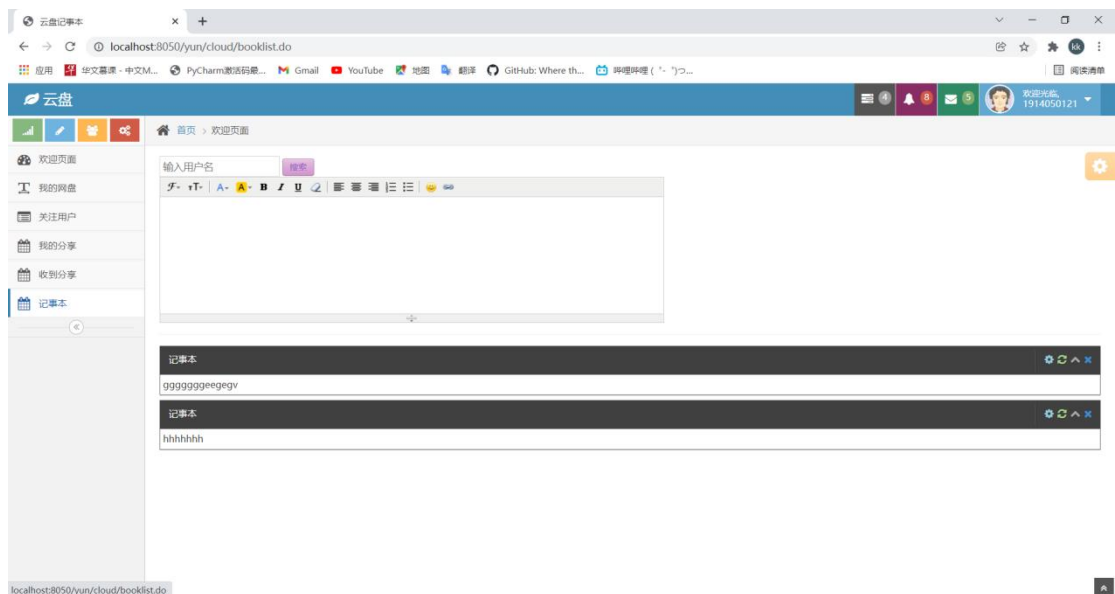


图 3-9 记事本

四、系统测试

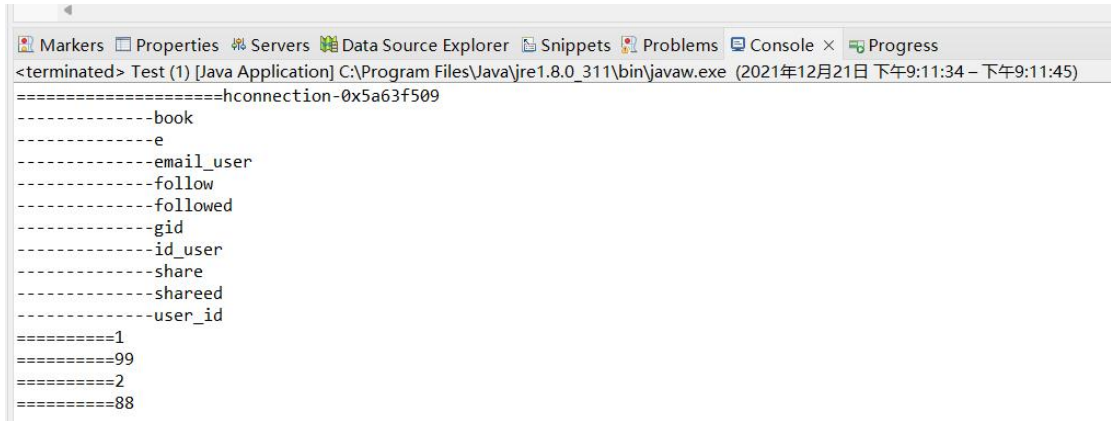
（一）测试方法

云存储系统的测试方法包括两种测试方法，包括黑盒测试法和白盒测试法。白盒测试以代码检查为主，黑盒测试主要以数据样例测试为主。

从程序执行的角度看，主要以运行程序的动态测试为主。

（二）样例数据

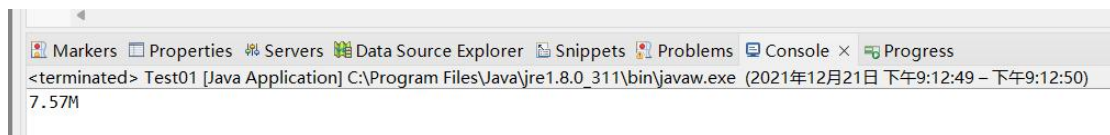
测试使用 java api 连接 hbase，运行结果出现 hbase 所有表，说明该程序运行正常。



```
<terminated> Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_311\bin\javaw.exe (2021年12月21日 下午9:11:34 - 下午9:11:45)
=====hconnection-0x5a63f509
-----book
-----e
-----email_user
-----follow
-----followed
-----gid
-----id_user
-----share
-----shareed
-----user_id
=====1
=====99
=====2
=====88
```

图 4-1 test 包中 test.java 运行结果

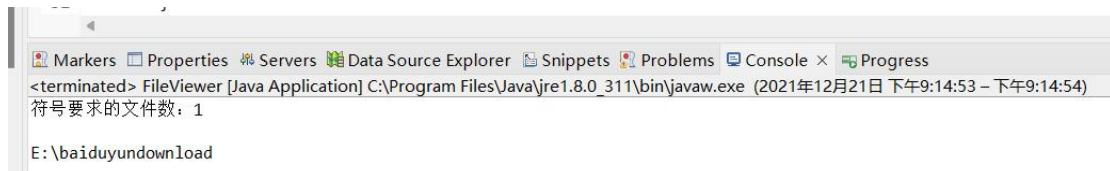
测试使用文件 utils 计算文件大小，运行结果出现文件尺寸，说明该程序运行正常。



```
<terminated> Test01 [Java Application] C:\Program Files\Java\jre1.8.0_311\bin\javaw.exe (2021年12月21日 下午9:12:49 - 下午9:12:50)
7.57M
```

图 4-2 test 包中 Test01.java 运行结果

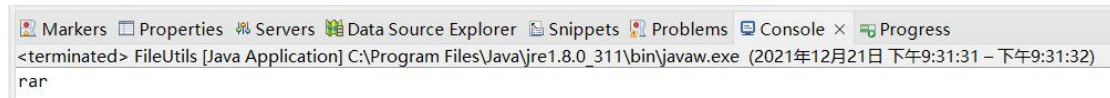
测试有多少文件满足预览条件，运行结果出现符合要求的文件数，说明该程序运行正常。



```
<terminated> FileViewer [Java Application] C:\Program Files\Java\jre1.8.0_311\bin\javaw.exe (2021年12月21日 下午9:14:53 - 下午9:14:54)
符号要求的文件数: 1
E:\baiduyundownload
```

图 4-3 FileViewer.java 运行结果

测试判断文件格式，运行结果出现的文件格式与实际一致，说明该程序运行正常。



```
<terminated> FileUtils [Java Application] C:\Program Files\Java\jre1.8.0_311\bin\javaw.exe (2021年12月21日 下午9:31:31 - 下午9:31:32)
rar
```

图 4-4 FileUtils.java 运行结果

(三) 前端测试

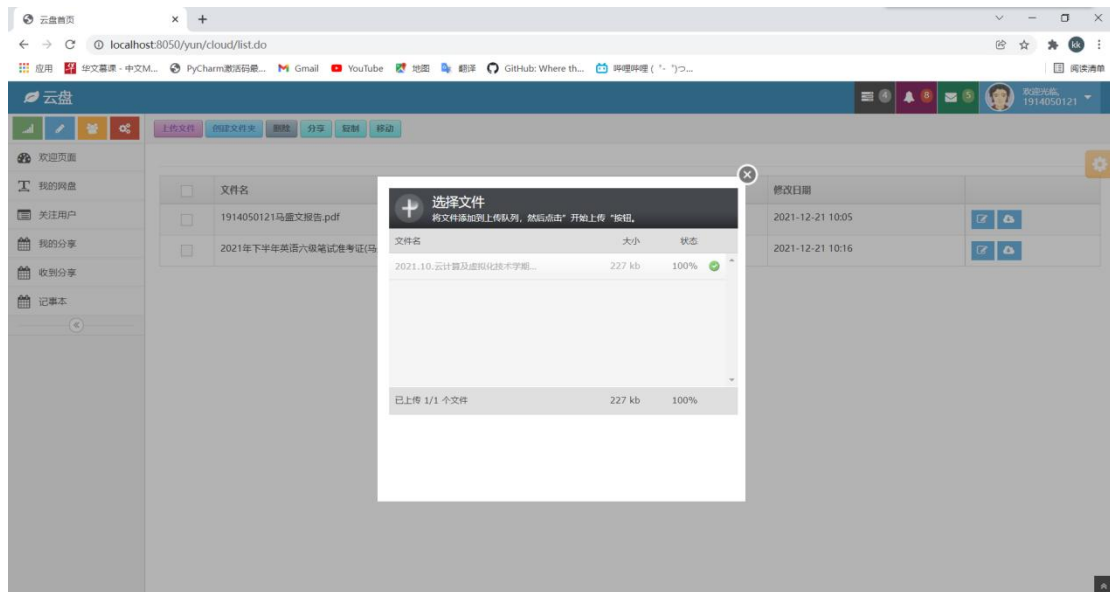


图 4-5 上传成功截图

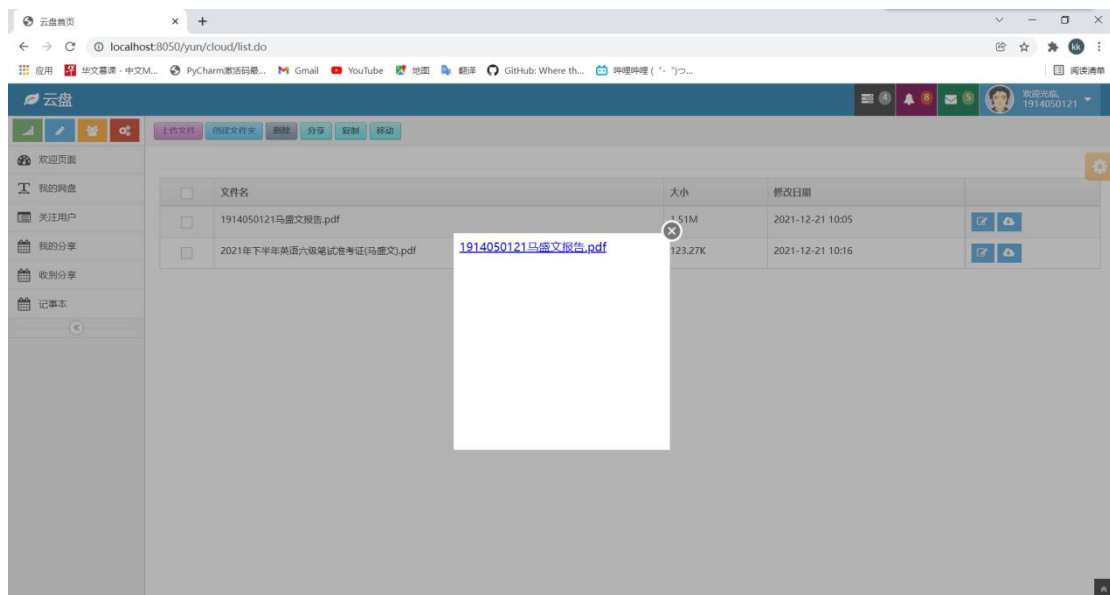


图 4-6 下载成功截图

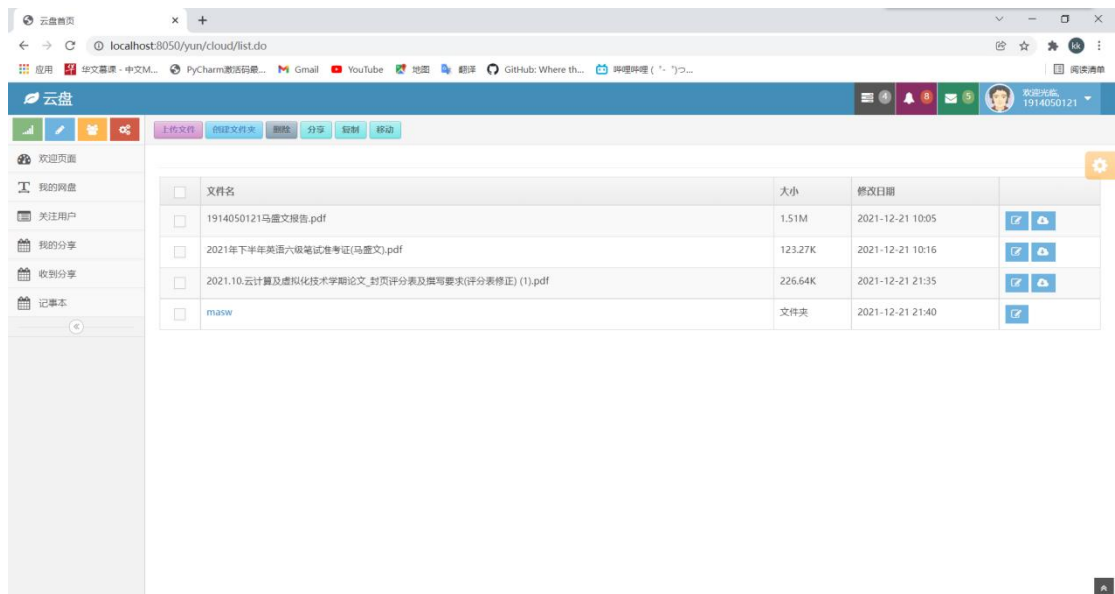


图 4-7 创建文件夹截图

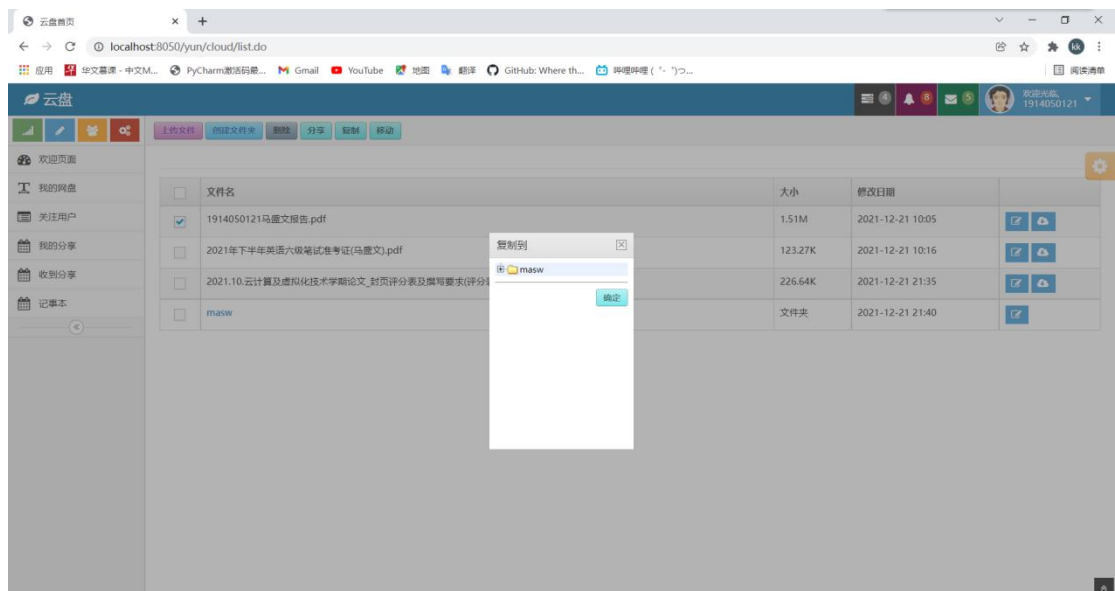


图 4-8 文件复制截图

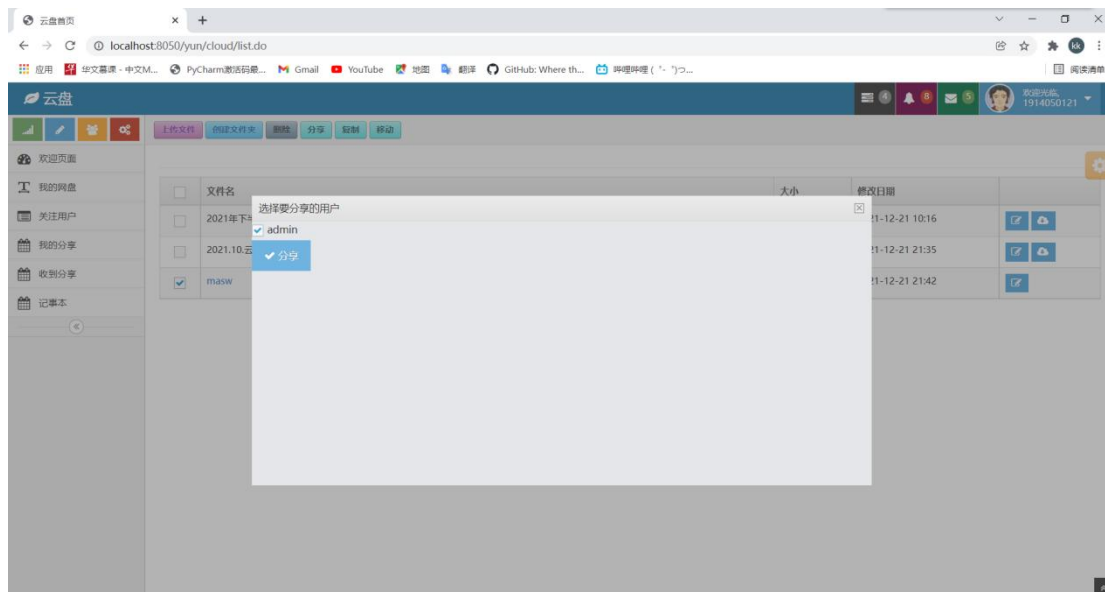


图 4-9 分享成功截图

五、总结

在本次课程设计中，我同小组其他成员对云存储系统进行探讨，从技术背景及研究现状入手，讨论了这一课题的研究意义，得出目前大多高校缺乏有效的云存储系统、普遍使用物理存储介质进行文件管理、传输导致病毒感染风险高、数据易丢失等结论。在此基础上，我们提出开发一个基于云计算技术的利用 HDFS 和 Hbase 结合 SpringMVC 框架的高校云存储系统，解决高校对文件存储、管理、传输等功能的需求。在系统的实际开发中，我们利用 Hbase 数据库对系统的用户数据和文件数据进行保存，利用 HDFS 进行管理，实现了用户的登录、注册功能。对于每个用户还实现了文件上传、下载、分享、移动、重命名、删除等的功能，并提供了创建文件夹的功能以方便用户管理自己的网盘。每个用户还可以关注其他用户，关注后可将自己存储在云盘中的文件分享给指定用户。

在系统的环境搭建中，先在 Windows 系统下部署 Hadoop 和 Hbase，调用 Hbase 提供的 java API 创建系统所需要的数据库表，最后使用 tomcat 进行前后端在服务器的部署。

在系统的测试中，云存储系统的测试方法包括两种测试方法，黑盒测试法和

白盒测试法。白盒测试以代码检查为主，黑盒测试主要以数据样例测试为主。从程序执行的角度看，主要以预先程序的动态测试为主，测试结果良好。

利用 SpringMVC 开发的云存储系统的有点在于可以为一个模型在运行是同时建立和使用多个视图变化-传播机制可以确保所有相关的视图及时得到模型数据变化，从而使所有关联的视图和控制器做到行为同步；视图与控制器的可接插性，允许更换视图和控制器对象，而且可以根据需求动态的打开或关闭、甚至在运行期间进行对象替换；模型的可移植性：因为模型是独立于视图的，所以可以把一个模型独立地移植到新的平台工作。需要做的只是在新平台上对视图和控制器进行新的修改；具有潜在的框架结构。可以基于此模型建立应用程序框架，不仅仅是用在设计界面的设计中。

缺点在于增加了系统结构和实现的复杂性。对于简单的界面，严格遵循 MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率；视图与控制器间的过于紧密的连接。视图与控制器是相互分离，但确实联系紧密的部件，视图没有控制器的存在，其应用是很有限的，反之亦然，这样就妨碍了他们的独立重用；视图对模型数据的低效率访问。依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问，也将损害操作性能。

在完成了整个系统的从设计到实现的所有工作后，我复习了云计算以及大数据技术基础两门课程的基础知识，对 HDFS 和 Hbase 有了更加深入的理解。在环境搭建过程中，碰到许多问题，通过查阅文献资料及阅读技术博客，了解了各种问题出现的原因以及问题解决方法原理。

附录（源代码，注意代码格式）

Book Model:

```
1. package com.fengdis.yun.vo;
2.
3. public class bookVo {
4.
5.     private String id;
6.     private String content;
7.     public String getId() {
8.         return id;
9.     }
10.    public void setId(String id) {
11.        this.id = id;
12.    }
13.    public String getContent() {
14.        return content;
15.    }
16.    public void setContent(String content) {
17.        this.content = content;
18.    }
19. }
```

FileSystem Model:

```
1. package com.fengdis.yun.vo;
2.
3. public class FileSystemVo {
4.
5.     private Long id;
6.     private String name;
7.     private String dir;
8.     private String pdir;
9.     private String type;
10.    private String date;
11.    private String size;
12.    private String viewflag="N";
13.
14.    private String ids;
15.    private String names;
16.    private String dirs;
```

```
17.  
18.     private String name;  
19.  
20.     public String getName() {  
21.         return name;  
22.     }  
23.     public void setName(String name) {  
24.         this.name = name;  
25.     }  
26.     public String getDir() {  
27.         return dir;  
28.     }  
29.     public void setDir(String dir) {  
30.         this.dir = dir;  
31.     }  
32.     public String getPdir() {  
33.         return pdir;  
34.     }  
35.     public void setPdir(String pdir) {  
36.         this.pdir = pdir;  
37.     }  
38.     public String getType() {  
39.         return type;  
40.     }  
41.     public void setType(String type) {  
42.         this.type = type;  
43.     }  
44.     public String getDate() {  
45.         return date;  
46.     }  
47.     public void setDate(String date) {  
48.         this.date = date;  
49.     }  
50.     public Long getId() {  
51.         return id;  
52.     }  
53.     public void setId(Long id) {  
54.         this.id = id;  
55.     }  
56.     public String getIds() {  
57.         return ids;  
58.     }  
59.     public void setIds(String ids) {  
60.         this.ids = ids;
```

```

61.     }
62.     public String getNames() {
63.         return names;
64.     }
65.     public void setNames(String names) {
66.         this.names = names;
67.     }
68.     public String getDirs() {
69.         return dirs;
70.     }
71.     public void setDirs(String dirs) {
72.         this.dirs = dirs;
73.     }
74.     public String getSize() {
75.         return size;
76.     }
77.     public void setSize(String size) {
78.         this.size = size;
79.     }
80.     public String getNamep() {
81.         return namep;
82.     }
83.     public void setNamep(String namep) {
84.         this.namep = namep;
85.     }
86.     public String getViewflag() {
87.         return viewflag;
88.     }
89.     public void setViewflag(String viewflag) {
90.         this.viewflag = viewflag;
91.     }
92. }

1. package com.fengdis.yun.vo;
2.
3. public class HdfsVo {
4.
5.     private String id;
6.     private String name;
7.     private String type;
8.     private String createDate;
9.     public HdfsVo() {}
10.    public HdfsVo(String name, String type) {
11.        this.name = name;
12.        this.type = type;

```

```

13.     }
14.     public String getName() {
15.         return name;
16.     }
17.     public void setName(String name) {
18.         this.name = name;
19.     }
20.     public String getType() {
21.         return type;
22.     }
23.     public void setType(String type) {
24.         this.type = type;
25.     }
26.     public String getCreateDate() {
27.         return createDate;
28.     }
29.     public void setCreateDate(String createDate) {
30.         this.createDate = createDate;
31.     }
32.     public String getId() {
33.         return id;
34.     }
35.     public void setId(String id) {
36.         this.id = id;
37.     }
38. }

```

Menu Model:

```

1. package com.fengdis.yun.vo;
2.
3. import java.util.Map;
4.
5. public class Menu {
6.
7.     private String id;
8.     private String text;
9.     private String name;
10.    private String iconcls;
11.    private String url;
12.
13.    private String state = "closed";
14.    private String pid;
15.    private String pname;

```

```

16.     private Map<String, Object> attributes;
17.     private String createtime;
18.     public Menu(){}
19.
20.     public Menu(String id, String text) {
21.         super();
22.         this.id = id;
23.         this.text = text;
24.     }
25.     public Map<String, Object> getAttributes() {
26.         return attributes;
27.     }
28.
29.     public void setAttributes(Map<String, Object> attributes) {
30.         this.attributes = attributes;
31.     }
32.
33.     public String getState() {
34.         return state;
35.     }
36.
37.     public void setState(String state) {
38.         this.state = state;
39.     }
40.
41.     public String getPid() {
42.         return pid;
43.     }
44.
45.     public void setPid(String pid) {
46.         this.pid = pid;
47.     }
48.
49.     public String getIconcls() {
50.         return iconcls;
51.     }
52.
53.     public void setIconcls(String iconcls) {
54.         this.iconcls = iconcls;
55.     }
56.
57.     public String getUrl() {
58.         return url;
59.     }

```

```
60.
61.     public void setUrl(String url) {
62.         this.url = url;
63.     }
64.
65.     public String getName() {
66.         return name;
67.     }
68.
69.     public void setName(String name) {
70.         this.name = name;
71.     }
72.
73.     public String getPname() {
74.         return pname;
75.     }
76.
77.     public void setPname(String pname) {
78.         this.pname = pname;
79.     }
80.
81.     public String getId() {
82.         return id;
83.     }
84.
85.     public void setId(String id) {
86.         this.id = id;
87.     }
88.
89.     public String getText() {
90.         return text;
91.     }
92.
93.     public void setText(String text) {
94.         this.text = text;
95.     }
96.
97.     public String getCreatetime() {
98.         return createtime;
99.     }
100.
101.     public void setCreatetime(String createtime) {
102.         this.createtime = createtime;
103.     }
```



```
104. }
```

Share Model:

```
1. package com.fengdis.yun.vo;
2.
3. public class ShareVo {
4.
5.     private String shareid;
6.     private String path;
7.     private String ts;
8.     private String dir;
9.     private String type;
10.    public String getShareid() {
11.        return shareid;
12.    }
13.    public void setShareid(String shareid) {
14.        this.shareid = shareid;
15.    }
16.    public String getPath() {
17.        return path;
18.    }
19.    public void setPath(String path) {
20.        this.path = path;
21.    }
22.    public String getTs() {
23.        return ts;
24.    }
25.    public void setTs(String ts) {
26.        this.ts = ts;
27.    }
28.    public String getDir() {
29.        return dir;
30.    }
31.    public void setDir(String dir) {
32.        this.dir = dir;
33.    }
34.    public String getType() {
35.        return type;
36.    }
37.    public void setType(String type) {
38.        this.type = type;
39.    }
40. }
```

Base Controller:

```
1. package com.fengdis.yun.base;
2.
3. import com.fengdis.yun.db.HbaseDB;
4. import com.fengdis.yun.db.HdfsDB;
5.
6. public class BaseController {
7.
8.     public HbaseDB db = HbaseDB.getInstance();
9.     public HdfsDB hdfsDB = HdfsDB.getInstance();
10. }
```

Cloud Controller:

```
1. package com.fengdis.yun.ctrl.cloud;
2.
3. import java.io.File;
4. import java.io.InputStream;
5. import java.util.Date;
6. import java.util.List;
7. import java.util.Map;
8.
9. import javax.servlet.http.HttpServletRequest;
10. import javax.servlet.http.HttpSession;
11.
12. import com.fengdis.yun.base.BaseController;
13. import com.fengdis.yun.vo.FileSystemVo;
14. import com.fengdis.yun.vo.Menu;
15. import org.springframework.stereotype.Controller;
16. import org.springframework.ui.Model;
17. import org.springframework.web.bind.annotation.RequestMapping;
18. import org.springframework.web.bind.annotation.ResponseBody;
19. import org.springframework.web.multipart.MultipartFile;
20. import org.springframework.web.multipart.MultipartHttpServletRequest;
21. import org.springframework.web.multipart.commons.CommonsMultipartResolver;
22.
23. import com.fengdis.yun.utils.BaseUtils;
24. import com.fengdis.yun.utils.DateUtil;
25. import com.fengdis.yun.utils.FileUtils;
26. import com.fengdis.yun.utils.Json;
27. import com.fengdis.yun.utils.OfficeToSwf;
28.
29. @Controller
```

```

30. @RequestMapping("/cloud")
31. public class CloudController extends BaseController {
32.
33.     @RequestMapping("/list")
34.     public String list(String name, HttpSession session, Model model) throws Exception {
35.         if (!BaseUtils.isEmpty(name)) {
36.             name = (String) session.getAttribute("username");
37.             name = "/" + name;
38.         }
39.         // model.addAttribute("fs", db.getFile(name));
40.         model.addAttribute("fs", hdfsDB.queryAll(name));
41.         model.addAttribute("dir", name);
42.         model.addAttribute("url", BaseUtils.getUrl(name));
43.         return "/cloud/list";
44.     }
45.     /**
46.      * 创建目录
47.      * @param mkdir
48.      * @param session
49.      * @return
50.      */
51.     @ResponseBody
52.     @RequestMapping("/mkdir")
53.     public Json mkdir(String mkdir, String dirName, HttpSession session) {
54.         Json json = new Json();
55.         if (!BaseUtils.isEmpty(mkdir)) {
56.             json.setMsg("空值无效");
57.             return json;
58.         }
59.         String name = (String) session.getAttribute("username");
60.         if (name == null) {
61.             json.setMsg("用户已注销, 请重新登陆");
62.             return json;
63.         }
64.         try {
65.             String dir = null;
66.             if (BaseUtils.isEmpty(dirName)) {
67.                 dir = dirName;
68.             } else {
69.                 dir = "/" + name;
70.             }
71.             //在该用户下创建目录
72.             hdfsDB.mkdir(dir + "/" + mkdir);

```

```

73.         /*long id = db.getGid();
74.         db.add("filesystem", id, "files", "name", mkdir);
75.         db.add("filesystem", id, "files", "dir", dir);
76.         db.add("filesystem", id, "files", "pdir", dir.substring(0, dir.lastIndexOf("/")));
77.         db.add("filesystem", id, "files", "type", "D");*/
78.
79.         FileSystemVo fs = new FileSystemVo();
80. //         fs.setId(id);
81.         fs.setName(mkdir);
82.         fs.setType("D");
83.         fs.setDate(DateUtil.DateToString("yyyy-MM-dd HH:mm", new Date()))
    ;
84.
85.         json.setObj(fs);
86.         json.setMsg("创建成功");
87.         json.setSuccess(true);
88.     } catch (Exception e) {
89.         e.printStackTrace();
90.         json.setMsg("创建失败");
91.     }
92.     return json;
93. }
94.
95. /**
96.  * 上传文件
97.  * @param dir
98.  * @param session
99.  * @param request
100.  * @return
101.  * @throws Exception
102.  */
103. @ResponseBody
104. @RequestMapping("/upload")
105. public Json upload(String dir,HttpSession session,HttpServletRequest request) throws Exception {
106.     Json json = new Json();
107.     String name = (String) session.getAttribute("username");
108.     if (name==null) {
109.         json.setMsg("用户已注销，请重新登陆");
110.         return json;
111.     }
112.     if(dir.equals("root")){
113.         dir = "/" + name;

```

```

114.     }
115.     CommonsMultipartResolver multipartResolver = new CommonsMultipartRe
        solver(request.getServletContext());
116.     if (multipartResolver.isMultipart(request)) {
117.         MultipartHttpServletRequest multipartRequest = (MultipartHttpSe
            rvletRequest) request;
118.         Map<String, MultipartFile> fms = multipartRequest.getFileMap();
119.         for (Map.Entry<String, MultipartFile> entity : fms.entrySet())
            {
120.             MultipartFile mf = entity.getValue();
121.             //System.out.println(mf.getSize());
122.             InputStream in = mf.getInputStream();
123.             hdfsDB.upload(in, dir+"/"+mf.getOriginalFilename());
124.             /*long id = db.getGid("gid");
125.             db.add("filesystem", id, "files", "name", mf.getOriginalFil
                ename());
126.             db.add("filesystem", id, "files", "dir", dir);
127.             db.add("filesystem", id, "files", "pdir", dir.substring(0,
                dir.lastIndexOf("/")));
128.             db.add("filesystem", id, "files", "type", "F");
129.             db.add("filesystem", id, "files", "size", BaseUtils.FormatF
                ileSize(mf.getSize()));*/
130.             in.close();
131.             json.setSuccess(true);
132.         }
133.     }
134.     return json;
135. }
136. /**
137.  * 删除文件及文件夹
138.  * @param ids
139.  * @param dir
140.  * @return
141.  * @throws Exception
142.  */
143. @ResponseBody
144. @RequestMapping("/delete")
145. public Json delete(String ids,String dir) throws Exception {
146.     Json json = new Json();
147.     try {
148.         String[] ns = ids.split(",");
149.         for (int i = 0; i < ns.length; i++) {
150.             hdfsDB.delete(dir+"/"+ns[i]);

```

```

151.         }
152.         json.setSuccess(true);
153.         json.setMsg("删除成功");
154.     } catch (Exception e) {
155.         json.setMsg("删除失败");
156.         e.printStackTrace();
157.     }
158.     return json;
159. }
160. /**
161.  * 复制文件及文件夹
162.  * @param ids
163.  * @param dir
164.  * @return
165.  * @throws Exception
166.  */
167. @ResponseBody
168. @RequestMapping("/copy")
169. public Json copy(String ids,String dir,String dst,boolean flag) throws
    Exception {
170.     Json json = new Json();
171.     String[] ns = ids.split(",");
172.     for (int i = 0; i < ns.length; i++) {
173.         ns[i] = dir+"/"+ns[i];
174.     }
175.     try {
176.         hdfsDB.copy(ns, dst, flag);
177.         json.setSuccess(true);
178.         json.setMsg("删除成功");
179.     } catch (Exception e) {
180.         json.setMsg("删除失败");
181.         e.printStackTrace();
182.     }
183.     return json;
184. }
185. /**
186.  * 重命名文件及文件夹
187.  * @param dir
188.  * @param name
189.  * @param rename
190.  * @param type
191.  * @return
192.  * @throws Exception
193.  */

```

```

194.     @ResponseBody
195.     @RequestMapping("/rename")
196.     public Json rename(String dir,String name,String rename,String type) throws Exception {
197.         Json json = new Json();
198.         try {
199.             if(type.equals("F")){
200.                 hdfsDB.rename(dir+"/"+name, dir+"/"+rename+name.substring(name.lastIndexOf(".")));
201.             }else if (type.equals("D")) {
202.                 hdfsDB.rename(dir+"/"+name, dir+"/"+rename);
203.             }
204.             json.setSuccess(true);
205.             json.setMsg("重命名成功");
206.         } catch (Exception e) {
207.             json.setMsg("重命名失败");
208.             e.printStackTrace();
209.         }
210.         return json;
211.     }
212.
213.     /**
214.      * 查看文档
215.      * @param dir
216.      * @param name
217.      * @param request
218.      * @param model
219.      * @return
220.      * @throws Exception
221.      */
222.     @RequestMapping("/view")
223.     public String view(String dir,String name,HttpServletRequest request,Model model) throws Exception {
224.         String local = request.getServletContext().getRealPath("/test");
225.         String swfFile = FileUtils.getFilePrefix(local+"\\ "+name)+".swf";
226.         File outFile = new File(swfFile);
227.         if(outFile.exists()){
228.             model.addAttribute("local", "test/"+name.substring(0,name.lastIndexOf("."))+".swf");
229.             return "cloud/view";
230.         }
231.         String pdfFile = FileUtils.getFilePrefix(local+"\\ "+name)+".pdf";
232.         File outFile01 = new File(pdfFile);
233.         if(!outFile01.exists()){

```

```

234.         hdfsDB.download(dir+"/"+name, local+"\\ "+name);
235.         OfficeToSwf.convert2PDF(local+"\\ "+name);
236.     }else{
237.         OfficeToSwf.pdf2swf(pdfFile);
238.     }
239.     model.addAttribute("local", "test/"+name.substring(0,name.lastIndexOf("."))+".swf");
240.     return "cloud/view";
241. }
242.
243. @RequestMapping("/download")
244. public String download(String dir,String name,HttpServletRequest request,Model model) throws Exception {
245.     String local = request.getServletContext().getRealPath("/test");
246.     File dirFile = new File(local);
247.     if(!dirFile.mkdirs()){
248.         dirFile.mkdirs();
249.     }
250.     File f = new File(local+"\\ "+name);
251.     if (!f.exists()) {
252.         hdfsDB.download(dir+"/"+name, local+"\\ "+name);
253.     }
254.     model.addAttribute("name", name);
255.     return "cloud/down";
256. }
257. /**
258.  * 分享
259.  * @param dir
260.  * @param names
261.  * @param usernames
262.  * @param session
263.  * @return
264.  * @throws Exception
265.  */
266. @ResponseBody
267. @RequestMapping("/share")
268. public Json share(String dir,String names,String usernames,String types,HttpSession session) throws Exception {
269.     Json json = new Json();
270.     String name = (String) session.getAttribute("username");
271.     if (name==null) {
272.         json.setMsg("用户已注销，请重新登陆");
273.         return json;
274.     }

```



```

275.         String[] n = names.split(",");
276.         String[] t = types.split(",");
277.         String[] u = usernames.split(",");
278.         for (int i = 0; i < u.length; i++) {
279.             try {
280.                 db.share(dir, name, n, t, u[i]);
281.             } catch (Exception e) {
282.                 json.setMsg("分享失败");
283.                 e.printStackTrace();
284.                 return json;
285.             }
286.         }
287.         json.setSuccess(true);
288.         return json;
289.     }
290.     /**
291.      * 获取分享
292.      * @param session
293.      * @param model
294.      * @return
295.      * @throws Exception
296.      */
297.     @RequestMapping("/getshare")
298.     public String getshare(HttpSession session, Model model) throws Exception {
299.         String name = (String) session.getAttribute("username");
300.         if (name == null) {
301.             return null;
302.         }
303.         model.addAttribute("shares", db.getshare(name));
304.         return "cloud/share";
305.     }
306.     /**
307.      * 获取被分享
308.      * @param session
309.      * @param model
310.      * @return
311.      * @throws Exception
312.      */
313.     @RequestMapping("/getshareed")
314.     public String getshareed(HttpSession session, Model model) throws Exception {
315.         String name = (String) session.getAttribute("username");
316.         if (name == null) {

```

```

317.         return null;
318.     }
319.     model.addAttribute("shares", db.getshareed(name));
320.     return "cloud/shareed";
321. }
322. /**
323.  * 查询被分享
324.  * @param dir
325.  * @param path
326.  * @param model
327.  * @return
328.  * @throws Exception
329.  */
330. @RequestMapping("/shareedList")
331. public String shareedList(String dir,String path,Model model) throws Ex
ception {
332.     if (dir==null) {
333.         dir=path;
334.     }else{
335.         dir=dir+"/"+path;
336.     }
337.     List<FileSystemVo> list = hdfsDB.queryAll(dir);
338.     model.addAttribute("shares", list);
339.     model.addAttribute("dir", dir);
340.     return "cloud/shareed_list";
341. }
342. /**
343.  * 记事本列表
344.  * @param session
345.  * @param model
346.  * @return
347.  * @throws Exception
348.  */
349. @RequestMapping("/booklist")
350. public String booklist(HttpSession session,Model model) throws Exceptio
n {
351.     String name = (String) session.getAttribute("username");
352.     if (name==null) {
353.         return null;
354.     }
355.     model.addAttribute("books", db.listbook(name));
356.     return "cloud/book";
357. }
358. /**

```

```

359.      * 新增记事本
360.      * @param content
361.      * @param session
362.      * @return
363.      * @throws Exception
364.      */
365.      @ResponseBody
366.      @RequestMapping("/bookadd")
367.      public Json bookadd(String content,HttpSession session) throws Exception
        n {
368.          Json json = new Json();
369.          String name = (String) session.getAttribute("username");
370.          if (name==null) {
371.              json.setMsg("用户已注销, 请重新登陆");
372.              return json;
373.          }
374.          try {
375.              db.addbook(name, content);
376.              json.setSuccess(true);
377.          } catch (Exception e) {
378.              json.setMsg("记事本添加失败");
379.              e.printStackTrace();
380.          }
381.          return json;
382.      }
383.      @ResponseBody
384.      @RequestMapping("/tree")
385.      public List<Menu> tree(String id, HttpSession session) throws Exception
        {
386.          List<Menu> menus = null;
387.          if (BaseUtils.isEmpty(id)) {
388.              menus = hdfsDB.tree(id);
389.          }else{
390.              String name = (String) session.getAttribute("username");
391.              if (name==null) {
392.                  return null;
393.              }
394.              menus = hdfsDB.tree("/"+name);
395.          }
396.          return menus;
397.      }
398. }

```

Login Controller:

```

1. package com.fengdis.yun.ctrl.login;
2.
3. import javax.servlet.http.HttpSession;
4.
5. import org.springframework.stereotype.Controller;
6. import org.springframework.web.bind.annotation.RequestMapping;
7.
8. import com.siyue.yun.base.BaseController;
9. import com.fengdis.yun.utils.BaseUtils;
10. import com.fengdis.yun.utils.Json;
11.
12. @Controller
13. @RequestMapping("/")
14. public class LoginController extends BaseController {
15.
16.     @RequestMapping("login")
17.     public String login(String userName,String pwd,HttpSession session) throws Exception {
18.         long userId = db.checkUser(userName, pwd);
19.         Json json = new Json();
20.         if (userId>0) {
21.             json.setSuccess(true);
22.             session.setAttribute("userid", userId);
23.             session.setAttribute("username", db.getUserNameById(userId));
24.         }else {
25.             json.setMsg("用户名或密码不正确");
26.         }
27.         return "redirect:index.jsp";
28.     }
29.
30.     @RequestMapping("logout")
31.     public String logout(HttpSession session) {
32.         if (session!=null) {
33.             session.invalidate();
34.         }
35.         return "redirect:login.jsp";
36.     }
37.
38.     @RequestMapping("reg")
39.     public String reg(String email,String username,String password) throws Exception {
40.         if(BaseUtils.isEmpty(email) && BaseUtils.isEmpty(username) &&
            BaseUtils.isEmpty(password)){
41.             long id = db.getGid("gid");

```

```

42.         db.add("user_id", username, "id", "id", id);
43.         db.add("id_user", id, "user", "name", username);
44.         db.add("id_user", id, "user", "pwd", password);
45.         db.add("id_user", id, "user", "email", email);
46.         db.add("email_user", email, "user", "userid", id);
47.         hdfsDB.mkdir("/"+username);
48.     }
49.     return "redirect:login.jsp";
50. }
51.
52. @RequestMapping("init")
53. public String init() throws Exception {
54.     String table_gid = "gid";
55.     String[] fam_gid = {"gid"};
56.     db.createTable(table_gid, fam_gid,1);
57.
58.     String table_id = "id_user";
59.     String[] fam_id = {"user"};
60.     db.createTable(table_id, fam_id,1);
61.
62.     String table_user = "user_id";
63.     String[] fam_user = {"id"};
64.     db.createTable(table_user, fam_user,1);
65.
66.     String table_email = "email_user";
67.     String[] fam_email = {"user"};
68.     db.createTable(table_email, fam_email,1);
69.
70.     db.add(table_gid, "gid", "gid", "gid", (long)0);
71.
72.     long id = db.getGid("gid");
73.     db.add("user_id", "admin", "id", "id", id);
74.     db.add("id_user", id, "user", "name", "admin");
75.     db.add("id_user", id, "user", "pwd", "admin");
76.     db.add("id_user", id, "user", "email", "923610744@qq.com");
77.     db.add("email_user", "923610744@qq.com", "user", "userid", id);
78.
79.     hdfsDB.mkdir("/admin");
80.
81.     String table_follow = "follow";
82.     String[] fam_follow_name = {"name"};
83.     db.createTable(table_follow, fam_follow_name,1);
84.
85.     String table_followed = "followed";

```

```

86.         String[] fam_followed_userid = {"userid"};
87.         db.createTable(table_followed, fam_followed_userid,1);
88.
89.         db.add("gid", "shareid", "gid", "shareid", (long)0);
90.         /*
91.          * tableName:share
92.          * rowkey:userid+shareid
93.          * content:path,content:ts
94.          */
95.         String table_share = "share";
96.         String[] fam_content = {"content"};
97.         db.createTable(table_share, fam_content,1);
98.
99.         /*
100.          * tableName:shareed
101.          * rowkey:userid+userid+shareid
102.          * shareid:
103.          */
104.         String table_shareed = "shareed";
105.         String[] fam_shareid = {"shareid"};
106.         db.createTable(table_shareed, fam_shareid,1);
107.
108.         db.add("gid", "bookid", "gid", "bookid", (long)0);
109.         //tableName:book
110.         //rowkey:userid+id
111.         //content:
112.         String table_book = "book";
113.         String[] fam_book_content = {"content"};
114.         db.createTable(table_book, fam_book_content,1);
115.
116.         return "redirect:login.jsp";
117.     }
118. }

```

EmunController:

```

1. package com.fengdis.yun.ctrl.sys;
2.
3. import java.util.List;
4.
5. import com.fengdis.yun.vo.Menu;
6. import org.springframework.stereotype.Controller;
7. import org.springframework.web.bind.annotation.RequestMapping;
8. import org.springframework.web.bind.annotation.ResponseBody;

```

```

9.
10. import com.siyue.yun.base.BaseController;
11.
12. @Controller
13. @RequestMapping("/emun")
14. public class EmunController extends BaseController{
15.
16.     @ResponseBody
17.     @RequestMapping("/list")
18.     public List<Menu> list() throws Exception {
19.         return db.queyAllEmun();
20.     }
21. }

```

User Controller:

```

1. package com.fengdis.yun.ctrl.sys;
2.
3. import javax.servlet.http.HttpSession;
4.
5. import org.springframework.stereotype.Controller;
6. import org.springframework.ui.Model;
7. import org.springframework.web.bind.annotation.RequestMapping;
8. import org.springframework.web.bind.annotation.ResponseBody;
9.
10. import com.siyue.yun.base.BaseController;
11. import com.fengdis.yun.utils.BaseUtils;
12. import com.fengdis.yun.utils.Json;
13.
14. @Controller
15. @RequestMapping("/user")
16. public class UserController extends BaseController{
17.
18.     @ResponseBody
19.     @RequestMapping("/reg")
20.     public Json reg(String userName,String pwd,String email) throws Exceptio
        n {
21.         Json json = new Json();
22.         if(db.checkUsername(userName)){
23.             json.setMsg("该用户名已存在, 请换一个");
24.             return json;
25.         }
26.         if (db.checkEmail(email)) {
27.             json.setMsg("该邮箱已存在");

```

```

28.         return json;
29.     }
30.     try {
31.         long id = db.getGid("gid");
32.         //创建用户
33.         db.add("user_id", userName, "id", "id", id);
34.         db.add("id_user", id, "user", "name", userName);
35.         db.add("id_user", id, "user", "pwd", pwd);
36.         db.add("id_user", id, "user", "email", email);
37.         db.add("email_user", email, "user", "userid", id);
38.         //在 hdfs 中创建用户根目录
39.         db.add("hdfs", id, "dir", "name", userName);
40.         db.add("hdfs", id, "dir", "type", "D");//D表示是目录,F表示是文件
41.         db.add("hdfs_name", userName, "id", "id", id);
42.         hdfsDB.mkdir(userName);
43.
44.         json.setSuccess(true);
45.         json.setMsg("用户注册成功");
46.     } catch (Exception e) {
47.         e.printStackTrace();
48.         json.setMsg("用户注册失败");
49.     }
50.     return json;
51. }
52. /**
53.  * 获取关注的用户
54.  * @param session
55.  * @param model
56.  * @return
57.  * @throws Exception
58.  */
59. @RequestMapping("/followlist")
60. public String followlist(HttpSession session, Model model) throws Exception {
61.     String name = (String) session.getAttribute("username");
62.     if (name != null) {
63.         model.addAttribute("follows", db.getFollow(name));
64.     }
65.     return "cloud/follow";
66. }
67. @RequestMapping("/getFollow")
68. public String getFollow(String ids, String dir, String types, HttpSession session, Model model) throws Exception {
69.     String name = (String) session.getAttribute("username");

```



```

70.         if (name!=null) {
71.             model.addAttribute("follows", db.getFollow(name));
72.         }
73.         model.addAttribute("dir", dir);
74.         model.addAttribute("ids", ids);
75.         model.addAttribute("types", types);
76.         return "cloud/share_getfollow";
77.     }
78.     /**
79.      * 关注用户
80.      * @param username
81.      * @param session
82.      * @throws Exception
83.      */
84.     @ResponseBody
85.     @RequestMapping("/follow")
86.     public Json follow(String username, HttpSession session) throws Exception
87.     {
88.         Json json = new Json();
89.         String name = (String) session.getAttribute("username");
90.         if (name!=null && BaseUtils.isEmpty(username)) {
91.             try {
92.                 db.follow(name, username);
93.                 json.setSuccess(true);
94.             } catch (Exception e) {
95.                 json.setMsg("关注失败");
96.                 e.printStackTrace();
97.             }
98.             return json;
99.         }
100.        /**
101.         * 取消关注
102.         * @param username
103.         * @param session
104.         * @throws Exception
105.         */
106.        @ResponseBody
107.        @RequestMapping("/unfollow")
108.        public Json unfollow(String username, HttpSession session) throws Except
109.        ion {
110.            Json json = new Json();
111.            String name = (String) session.getAttribute("username");
112.            if (name!=null && BaseUtils.isEmpty(username)) {

```

```

112.         try {
113.             db.unfollow(name, username);
114.             json.setSuccess(true);
115.         } catch (Exception e) {
116.             json.setMsg("取消失败");
117.             e.printStackTrace();
118.         }
119.     }
120.     return json;
121. }
122.
123. @ResponseBody
124. @RequestMapping("/getuser")
125. public Json getuser(String username) {
126.     Json json = new Json();
127.     if (db.checkUsername(username)) {
128.         json.setSuccess(true);
129.         json.setMsg(username);
130.     } else {
131.         json.setMsg("没有该用户");
132.     }
133.     return json;
134. }
135. }

```

HbaseDB:

```

1. package com.fengdis.yun.db;
2.
3. import java.io.IOException;
4. import java.io.Serializable;
5. import java.util.ArrayList;
6. import java.util.Date;
7. import java.util.HashSet;
8. import java.util.List;
9. import java.util.Set;
10.
11. import org.apache.hadoop.conf.Configuration;
12. import org.apache.hadoop.hbase.Cell;
13. import org.apache.hadoop.hbase.CellUtil;
14. import org.apache.hadoop.hbase.HBaseConfiguration;
15. import org.apache.hadoop.hbase.HColumnDescriptor;
16. import org.apache.hadoop.hbase.HTableDescriptor;
17. import org.apache.hadoop.hbase.TableName;

```

```

18. import org.apache.hadoop.hbase.client.Admin;
19. import org.apache.hadoop.hbase.client.Connection;
20. import org.apache.hadoop.hbase.client.ConnectionFactory;
21. import org.apache.hadoop.hbase.client.Delete;
22. import org.apache.hadoop.hbase.client.Get;
23. import org.apache.hadoop.hbase.client.Put;
24. import org.apache.hadoop.hbase.client.Result;
25. import org.apache.hadoop.hbase.client.ResultScanner;
26. import org.apache.hadoop.hbase.client.Scan;
27. import org.apache.hadoop.hbase.client.Table;
28. import org.apache.hadoop.hbase.filter.BinaryComparator;
29. import org.apache.hadoop.hbase.filter.CompareFilter.CompareOp;
30. import org.apache.hadoop.hbase.filter.Filter;
31. import org.apache.hadoop.hbase.filter.QualifierFilter;
32. import org.apache.hadoop.hbase.filter.SubstringComparator;
33. import org.apache.hadoop.hbase.util.Bytes;
34.
35. import com.fengdis.yun.utils.DateUtil;
36. import com.fengdis.yun.utils.SiteUrl;
37. import com.fengdis.yun.vo.FileSystemVo;
38. import com.fengdis.yun.vo.Menu;
39. import com.fengdis.yun.vo.ShareVo;
40. import com.fengdis.yun.vo.bookVo;
41.
42. public class HbaseDB implements Serializable{
43.     private static final long serialVersionUID = -7137236230164276653L;
44.     // static HConnection connection;
45.     static Connection connection;
46.
47.     private static class HbaseDBInstance{
48.         private static final HbaseDB instance = new HbaseDB();
49.     }
50.     public static HbaseDB getInstance() {
51.         return HbaseDBInstance.instance;
52.     }
53.     private HbaseDB() {
54.         Configuration conf = HBaseConfiguration.create();
55.         conf.set("hbase.zookeeper.quorum", SiteUrl.readUrl("host"));
56.         try {
57.             // connection = HConnectionManager.createConnection(conf);
58.             connection = ConnectionFactory.createConnection(conf);
59.         } catch (IOException e) {
60.             e.printStackTrace();
61.         }

```

```

62.     }
63.
64.     private Object readResolve(){
65.         return getInstance();
66.     }
67.     /**
68.      * 获取所有表
69.      * @return
70.      * @throws Exception
71.      */
72.     public static TableName[] listTable() throws Exception {
73. //         HBaseAdmin admin = new HBaseAdmin(connection);
74.         Admin admin = connection.getAdmin();
75.         TableName[] tableNames = admin.listTableNames();
76.         admin.close();
77.         return tableNames;
78.     }
79.     /**
80.      * 删除所有表
81.      */
82.     public static void deleteAllTable() throws Exception{
83. //         HBaseAdmin admin = new HBaseAdmin(connection);
84.         Admin admin = connection.getAdmin();
85.         TableName[] tableNames = admin.listTableNames();
86.         for (int i = 0; i < tableNames.length; i++) {
87. //             admin.disableTable(tableNames[i].getNameAsString());
88. //             admin.deleteTable(tableNames[i].getNameAsString());
89.             admin.disableTable(tableNames[i]);
90.             admin.deleteTable(tableNames[i]);
91.         }
92.         admin.close();
93.     }
94.     /**
95.      * 创建表
96.      * @param tableName
97.      * @param fams
98.      * @throws Exception
99.      */
100.    public static void createTable(String tableName,String[] fams,int versi
        on) throws Exception {
101. //         HBaseAdmin admin = new HBaseAdmin(connection);
102.         Admin admin = connection.getAdmin();
103.         TableName tn = TableName.valueOf(tableName);
104.         if (admin.tableExists(tn)) {

```

```

105.         admin.disableTable(tn);
106.         admin.deleteTable(tn);
107.     }
108.     /*if (admin.tableExists(tableName)) {
109.         admin.disableTable(tableName);
110.         admin.deleteTable(tableName);
111.     }*/
112.     HTableDescriptor tableDescriptor = null;
113.     HColumnDescriptor hd = null;
114.     for (int i = 0; i < fams.length; i++) {
115.         tableDescriptor = new HTableDescriptor(tn);
116.         hd = new HColumnDescriptor(fams[i]);
117.         hd.setMaxVersions(version);
118.         tableDescriptor.addFamily(hd);
119.         admin.createTable(tableDescriptor);
120.     }
121.     admin.close();
122. }
123.
124.
125. /**
126.  * 删除表
127.  * @param tableName
128.  * @throws Exception
129.  */
130. public static void delTable(String tableName) throws Exception {
131. //     HBaseAdmin admin = new HBaseAdmin(connection);
132.     Admin admin = connection.getAdmin();
133.     TableName tn = TableName.valueOf(tableName);
134.     /*if (admin.tableExists(tableName)) {
135.         admin.disableTable(tableName);
136.         admin.deleteTable(tableName);
137.     }*/
138.     if (admin.tableExists(tn)) {
139.         admin.disableTable(tn);
140.         admin.deleteTable(tn);
141.     }
142.     admin.close();
143. }
144.
145. /**
146.  * getGid
147.  * @param row
148.  * @throws Exception

```

```

149.     */
150.     public static long getGid(String row) throws Exception {
151.         Table table_gid = connection.getTable(TableName.valueOf("gid"));
152.         // HTable table_gid = new HTable(TableName.valueOf("gid"), connection);

153.         long id = table_gid.incrementColumnValue(Bytes.toBytes(row), Bytes.
            toBytes("gid"), Bytes.toBytes(row), 1);
154.         table_gid.close();
155.         return id;
156.     }
157.
158.     /**
159.      * 添加数据
160.      * @param tableName
161.      * @param rowKey
162.      * @param family
163.      * @param qualifier
164.      * @param value
165.      * @throws IOException
166.      */
167.     public static void add(String tableName, String rowKey, String family,
        String qualifier, String value) throws IOException {
168.         //连接到 table
169.         // HTable table = new HTable(TableName.valueOf(tableName), connection);

170.         Table table = connection.getTable(TableName.valueOf(tableName));
171.         Put put = new Put(Bytes.toBytes(rowKey));
172.         // put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBy
            tes(value));
173.         put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier), Byte
            s.toBytes(value));
174.         table.put(put);
175.         table.close();
176.     }
177.     /**
178.      * 添加数据
179.      * @param tableName
180.      * @param rowKey
181.      * @param family
182.      * @param qualifier
183.      * @param value
184.      * @throws IOException
185.      */

```

```

186.     public static void add(String tableName, Long rowKey, String family, Long
        qualifier, String value) throws IOException {
187.         //连接到 table
188.         //      HTable table = new HTable(tableName.valueOf(tableName), connection);

189.         Table table = connection.getTable(tableName.valueOf(tableName));
190.         Put put = new Put(Bytes.toBytes(rowKey));
191.         //      put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
192.         put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
193.         table.put(put);
194.         table.close();
195.     }
196.     /**
197.      * 添加数据
198.      * @param tableName
199.      * @param rowKey
200.      * @param family
201.      * @param qualifier
202.      * @param value
203.      * @throws IOException
204.      */
205.     public static void add(String tableName, Long rowKey01, Long rowKey02, String family, String qualifier, Long value) throws IOException {
206.         //连接到 table
207.         //      HTable table = new HTable(tableName.valueOf(tableName), connection);

208.         Table table = connection.getTable(tableName.valueOf(tableName));
209.         Put put = new Put(Bytes.add(Bytes.toBytes(rowKey01), Bytes.toBytes(rowKey02)));
210.         if (qualifier != null) {
211.             //      put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
212.             put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
213.         } else {
214.             //      put.add(Bytes.toBytes(family), null, Bytes.toBytes(value));
215.             put.addColumn(Bytes.toBytes(family), null, Bytes.toBytes(value));
216.         }
217.         table.put(put);
218.         table.close();
219.     }

```

```

220.    /**
221.     * 添加数据
222.     * @param tableName
223.     * @param rowKey
224.     * @param family
225.     * @param qualifier
226.     * @param value
227.     * @throws IOException
228.     */
229.    public static void add(String tableName, Long rowKey01, Long rowKey02, Long rowKey03, String family, String qualifier, Long value01, Long value02) throws IOException {
230.        //连接到 table
231.        // HTable table = new HTable(TableName.valueOf(tableName), connection);

232.        Table table = connection.getTable(TableName.valueOf(tableName));
233.        Put put = new Put(Bytes.add(Bytes.toBytes(rowKey01), Bytes.toBytes(rowKey02), Bytes.toBytes(rowKey03)));
234.        if (qualifier != null) {
235.            // put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.add(Bytes.toBytes(value01), Bytes.toBytes(value02)));
236.            put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.add(Bytes.toBytes(value01), Bytes.toBytes(value02)));
237.        } else {
238.            // put.add(Bytes.toBytes(family), null, Bytes.add(Bytes.toBytes(value01), Bytes.toBytes(value02)));
239.            put.addColumn(Bytes.toBytes(family), null, Bytes.add(Bytes.toBytes(value01), Bytes.toBytes(value02)));
240.        }
241.        table.put(put);
242.        table.close();
243.    }
244.    /**
245.     * 添加数据
246.     * @param tableName
247.     * @param rowKey
248.     * @param family
249.     * @param qualifier
250.     * @param value
251.     * @throws IOException
252.     */
253.    public static void add(String tableName, Long rowKey01, Long rowKey02, String family, String qualifier, String value) throws IOException {
254.        //连接到 table

```



```

255. //      HTable table = new HTable(TableName.valueOf(tableName), connection);

256.      Table table = connection.getTable(TableName.valueOf(tableName));
257.      Put put = new Put(Bytes.add(Bytes.toBytes(rowKey01), Bytes.toBytes(
rowKey02)));
258.      if (qualifier!=null) {
259. //          put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.
toBytes(value));
260.          put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier),
Bytes.toBytes(value));
261.      }else{
262. //          put.add(Bytes.toBytes(family), null, Bytes.toBytes(value));
263.          put.addColumn(Bytes.toBytes(family), null, Bytes.toBytes(value))
;
264.      }
265.      table.put(put);
266.      table.close();
267.  }
268.  /**
269.   * 添加数据
270.   * @param tableName
271.   * @param rowKey
272.   * @param family
273.   * @param qualifier
274.   * @param value
275.   * @throws IOException
276.   */
277.  public static void add(String tableName, Long rowKey, String family, St
ring qualifier, String value) throws IOException {
278.      //连接到 table
279. //      HTable table = new HTable(TableName.valueOf(tableName), connection);

280.      Table table = connection.getTable(TableName.valueOf(tableName));
281.      Put put = new Put(Bytes.toBytes(rowKey));
282. //      put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBy
tes(value));
283.      put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier), Byte
s.toBytes(value));
284.      table.put(put);
285.      table.close();
286.  }
287.  /**
288.   * 添加数据
289.   * @param tableName

```

```

290.     * @param rowKey
291.     * @param family
292.     * @param qualifier
293.     * @param value
294.     * @throws IOException
295.     */
296.     public static void add(String tableName, Long rowKey, String family, String qualifier, Long value) throws IOException {
297.         //连接到 table
298.         // HTable table = new HTable(TableName.valueOf(tableName), connection);

299.         Table table = connection.getTable(TableName.valueOf(tableName));
300.         Put put = new Put(Bytes.toBytes(rowKey));
301.         // put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
302.         put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
303.         table.put(put);
304.         table.close();
305.     }
306.     /**
307.     * 添加数据
308.     * @param tableName
309.     * @param rowKey
310.     * @param family
311.     * @param qualifier
312.     * @param value
313.     * @throws IOException
314.     */
315.     public static void add(String tableName, String rowKey, String family, String qualifier, Long value) throws IOException {
316.         //连接到 table
317.         // HTable table = new HTable(TableName.valueOf(tableName), connection);

318.         Table table = connection.getTable(TableName.valueOf(tableName));
319.         Put put = new Put(Bytes.toBytes(rowKey));
320.         // put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
321.         put.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier), Bytes.toBytes(value));
322.         table.put(put);
323.         table.close();
324.     }
325.     /**

```

```

326.      * 根据 row 删除数据
327.      * @param tableName
328.      * @param rowKey
329.      * @throws Exception
330.      */
331.      public static void deleteRow(String tableName, String[] rowKey) throws
        Exception {
332.          //      HTable table = new HTable(TableName.valueOf(tableName), connection);

333.          Table table = connection.getTable(TableName.valueOf(tableName));
334.          List<Delete> list = new ArrayList<Delete>();
335.          for (int i = 0; i < rowKey.length; i++) {
336.              Delete delete = new Delete(Bytes.toBytes(Long.valueOf(rowKey[i])
                ));
337.              list.add(delete);
338.          }
339.          table.delete(list);
340.          table.close();
341.      }
342.
343.      public static void deleteColumns(String tableName, Long rowKey, String fa
        mily, Long qualifier) throws Exception {
344.          //      HTable table = new HTable(TableName.valueOf(tableName), connection);

345.          Table table = connection.getTable(TableName.valueOf(tableName));
346.          Delete delete = new Delete(Bytes.toBytes(rowKey));
347.          //      delete.deleteColumns(Bytes.toBytes(family), Bytes.toBytes(qualifier)
                );
348.          delete.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier));

349.          table.delete(delete);
350.          table.close();
351.      }
352.      public static void deleteRow(String tableName, Long rowKey01, Long rowKey
        02) throws Exception {
353.          //      HTable table = new HTable(TableName.valueOf(tableName), connection);

354.          Table table = connection.getTable(TableName.valueOf(tableName));
355.          Delete delete = new Delete(Bytes.add(Bytes.toBytes(rowKey01), Bytes.
                toBytes(rowKey02)));
356.          table.delete(delete);
357.          table.close();
358.      }
359.

```

```

360.     public static long getIdByUsername(String name) {
361.         long id = 0;
362.         try {
363.             //      HTable table = new HTable(TableName.valueOf("user_id"), connect
                ion);
364.             Table table = connection.getTable(TableName.valueOf("user_id"));

365.             Get get = new Get(Bytes.toBytes(name));
366.             get.addColumn(Bytes.toBytes("id"), Bytes.toBytes("id"));
367.             Result rs = table.get(get);
368.             byte[] value = rs.getValue(Bytes.toBytes("id"), Bytes.toBytes("
                id"));
369.             id = Bytes.toLong(value);
370.             table.close();
371.         } catch (IOException e) {
372.             e.printStackTrace();
373.             return id;
374.         }
375.         return id;
376.     }
377.
378.     public boolean checkUsername(String name) {
379.         try {
380.             //      HTable table = new HTable(TableName.valueOf("user_id"), connect
                ion);
381.             Table table = connection.getTable(TableName.valueOf("user_id"));

382.             Get get = new Get(Bytes.toBytes(name));
383.             table.exists(get);
384.             if (table.exists(get)) {
385.                 table.close();
386.                 return true;
387.             }else{
388.                 table.close();
389.                 return false;
390.             }
391.         } catch (IOException e) {
392.             e.printStackTrace();
393.             return false;
394.         }
395.     }
396.
397.     public static String getUserByNameById(long id) {
398.         String name = null;

```

```

399.         try {
400. //             HTable table = new HTable(TableName.valueOf("id_user"), connect
                ion);
401.             Table table = connection.getTable(TableName.valueOf("id_user"));

402.             Get get = new Get(Bytes.toBytes(id));
403.             get.addColumn(Bytes.toBytes("user"), Bytes.toBytes("name"));
404.             Result rs = table.get(get);
405.             byte[] value = rs.getValue(Bytes.toBytes("user"), Bytes.toBytes
                ("name"));
406.             name = Bytes.toString(value);
407.             table.close();
408.         } catch (IOException e) {
409.             e.printStackTrace();
410.             return null;
411.         }
412.         return name;
413.     }
414.
415.     public static String getStringById(String tableName, Long rowKey, String
        family, String qualifier) {
416.         String name = null;
417.         try {
418. //             HTable table = new HTable(TableName.valueOf(tableName), connect
                ion);
419.             Table table = connection.getTable(TableName.valueOf(tableName));

420.             Get get = new Get(Bytes.toBytes(rowKey));
421.             get.addColumn(Bytes.toBytes(family), Bytes.toBytes(qualifier));

422.             Result rs = table.get(get);
423.             byte[] value = rs.getValue(Bytes.toBytes(family), Bytes.toBytes
                (qualifier));
424.             name = Bytes.toString(value);
425.             table.close();
426.         } catch (IOException e) {
427.             e.printStackTrace();
428.             return null;
429.         }
430.         return name;
431.     }
432.     /**
433.      * 通过目录名获取 ID
434.      * @param name

```

```

435.      * @return
436.      */
437.      public static long getIdByDirName(String name) {
438.          long id = 0;
439.          try {
440.              //      HTable table = new HTable(TableName.valueOf("hdfs_name"), connection);
441.              Table table = connection.getTable(TableName.valueOf("hdfs_name"));
442.              Get get = new Get(name.getBytes());
443.              get.addColumn(Bytes.toBytes("id"), Bytes.toBytes("id"));
444.              Result rs = table.get(get);
445.              byte[] value = rs.getValue(Bytes.toBytes("id"), Bytes.toBytes("id"));
446.              id = Bytes.toLong(value);
447.              table.close();
448.          } catch (IOException e) {
449.              e.printStackTrace();
450.              return id;
451.          }
452.          return id;
453.      }
454.
455.      public static boolean checkEmail(String email) throws Exception {
456.          //      HTable table = new HTable(TableName.valueOf("email_user"), connection);
457.          Table table = connection.getTable(TableName.valueOf("email_user"));
458.          Get get = new Get(Bytes.toBytes(email));
459.          get.addColumn(Bytes.toBytes("user"), Bytes.toBytes("userid"));
460.          Result rs = table.get(get);
461.          byte[] value = rs.getValue(Bytes.toBytes("user"), Bytes.toBytes("userid"));
462.          table.close();
463.          if(value!=null){
464.              return true;
465.          }else {
466.              return false;
467.          }
468.      }
469.
470.      public long checkUser(String userName,String pwd) throws Exception {
471.          long id = getIdByUsername(userName);
472.          if (id==0) {

```

```

473.         return 0;
474.     }
475. //     HTable table = new HTable(TableName.valueOf("id_user"), connection);

476.     Table table = connection.getTable(TableName.valueOf("id_user"));
477.     Get get = new Get(Bytes.toBytes(id));
478.     get.addColumn(Bytes.toBytes("user"), Bytes.toBytes("pwd"));
479.     Result rs = table.get(get);
480.     byte[] value = rs.getValue(Bytes.toBytes("user"), Bytes.toBytes("pw
        d"));
481.     if (pwd.equals(Bytes.toString(value))) {
482.         table.close();
483.         return id;
484.     }
485.     table.close();
486.     return 0;
487. }
488.
489. public void queryAll(String tableName) throws Exception {
490. //     HTable table = new HTable(TableName.valueOf(tableName), connection);

491.     Table table = connection.getTable(TableName.valueOf(tableName));
492.     ResultScanner rs = table.getScanner(new Scan());
493.     for (Result result : rs) {
494.         System.out.println("rowkey" +result.getRow());
495.         for (Cell cell : result.rawCells()) {
496.             System.out.println("family"+new String(cell.getFamilyArray()
                ));
497.             System.out.println("Qualifier"+new String(cell.getQualifier
                Array()));
498.             System.out.println("value"+new String(cell.getValueArray()))
                ;
499.         }
500.     }
501.     table.close();
502. }
503.
504. public void queryAllHDFS(String username) throws Exception {
505. //     HTable table = new HTable(TableName.valueOf("hdfs"), connection);
506.     Table table = connection.getTable(TableName.valueOf("hdfs"));
507.     ResultScanner rs = table.getScanner(new Scan());
508.     for (Result result : rs) {
509.         System.out.println("rowkey" +result.getRow());
510.         for (Cell cell : result.rawCells()) {

```

```

511.         System.out.println("family"+new String(cell.getFamilyArray()
    ));
512.         System.out.println("Qualifier"+new String(cell.getQualifier
    Array()));
513.         System.out.println("value"+new String(cell.getValueArray()))
    ;
514.     }
515. }
516.     table.close();
517. }
518.
519.     public static List<Menu> qureyAllEmun() throws Exception {
520. //         HTable table = new HTable(TableName.valueOf("emun"), connection);
521.         Table table = connection.getTable(TableName.valueOf("emun"));
522.         ResultScanner rs = table.getScanner(new Scan());
523.         List<Menu> menus = new ArrayList<Menu>();
524.         Menu m = null;
525.         for (Result r : rs) {
526.             m = new Menu();
527.             byte[] name = r.getValue(Bytes.toBytes("emun"), Bytes.toBytes("
    name"));
528.             byte[] url = r.getValue(Bytes.toBytes("emun"), Bytes.toBytes("u
    rl"));
529.             m.setName(Bytes.toString(name));
530.             m.setUrl(Bytes.toString(url));
531.             m.setText(Bytes.toString(name));
532.             menus.add(m);
533.         }
534.         table.close();
535.         return menus;
536.     }
537.
538.     public static void getAllUserTree(Long id) throws Exception {
539. //         HTable table_hdfs = new HTable(TableName.valueOf("hdfs"), connectio
    n);
540. //         HTable table = new HTable(TableName.valueOf("hdfs_cid"), connection)
    ;
541.         Table table_hdfs = connection.getTable(TableName.valueOf("hdfs"));
542.         Table table = connection.getTable(TableName.valueOf("hdfs_cid"));
543.         Get get = new Get(Bytes.toBytes(id));
544.         Result rs = table.get(get);
545. //         List<Menu> menus = new ArrayList<Menu>();
546.         Menu menu = null;

```



```

547.         for (Cell cell : rs.rawCells()) {
548.             Get get1 = new Get(CellUtil.cloneValue(cell));
549.             get1.addColumn(Bytes.toBytes("dir"), Bytes.toBytes("name"));
550.             Result rs1 = table_hdfs.get(get1);
551.             byte[] value = rs1.getValue(Bytes.toBytes("dir"), Bytes.toBytes(
                ("name")));
552.             String name = Bytes.toString(value);
553.
554.             get1.addColumn(Bytes.toBytes("dir"), Bytes.toBytes("type"));
555. //             Result rs2 = table_hdfs.get(get1);
556. //             byte[] type = rs2.getValue(Bytes.toBytes("dir"), Bytes.toBytes(
                "type"));
557. //             String y = Bytes.toString(type);
558.             menu = new Menu();
559.             menu.setId(Bytes.toString(CellUtil.cloneValue(cell)));
560.             menu.setName(name);
561.         }
562.         table.close();
563.     }
564.
565.     public static List<FileSystemVo> getFile(String dir) throws Exception {
566. //         HTable fileTable = new HTable(TableName.valueOf("filesystem"), conn
            ection);
567.         Table fileTable = connection.getTable(TableName.valueOf("filesystem
            "));
568.         Scan scan = new Scan();
569.         Filter filter = new QualifierFilter(CompareOp.LESS_OR_EQUAL, new Su
            bstringComparator(dir));
570.         scan.setFilter(filter);
571.         ResultScanner rs = fileTable.getScanner(scan);
572.         List<FileSystemVo> fs = new ArrayList<FileSystemVo>();
573.         FileSystemVo f = null;
574.         for (Result r : rs) {
575.             Cell cellName = r.getColumnLatestCell(Bytes.toBytes("files"), B
                ytes.toBytes("name"));
576.             Cell cellPdir = r.getColumnLatestCell(Bytes.toBytes("files"), B
                ytes.toBytes("pdir"));
577.             Cell cellType = r.getColumnLatestCell(Bytes.toBytes("files"), B
                ytes.toBytes("type"));
578.             Cell cellSize = r.getColumnLatestCell(Bytes.toBytes("files"), B
                ytes.toBytes("size"));
579.             f = new FileSystemVo();
580.             f.setId(Bytes.toLong(r.getRow()));

```

```

581.         f.setDir(dir);
582.         f.setName(Bytes.toString(CellUtil.cloneValue(cellName)));
583.         if (cellSize!=null) {
584.             f.setSize(Bytes.toString(CellUtil.cloneValue(cellSize)));
585.         }
586.         if(cellPdir!=null){
587.             f.setPdir(Bytes.toString(CellUtil.cloneValue(cellPdir)));
588.         }
589.         if (cellType!=null) {
590.             f.setType(Bytes.toString(CellUtil.cloneValue(cellType)));
591.         }
592.         f.setDate(DateUtil.longToString("yyyy-MM-dd HH:mm", cellName.ge
tTimestamp()));
593.         fs.add(f);
594.     }
595.     fileTable.close();
596.     return fs;
597. }
598.
599.     public static void delByDir(String dir) throws Exception {
600. //         HTable fileTable = new HTable(TableName.valueOf("filesystem"), conn
ection);
601.         Table fileTable = connection.getTable(TableName.valueOf("filesystem
"));
602.         Scan scan = new Scan();
603.         Filter filter = new QualifierFilter(CompareOp.LESS_OR_EQUAL, new Bi
naryComparator(Bytes.toBytes(dir)));
604.         scan.setFilter(filter);
605.         ResultScanner rs = fileTable.getScanner(scan);
606.         for (Result r : rs) {
607.             fileTable.delete(new Delete(r.getRow()));
608.         }
609.         fileTable.close();
610.     }
611.
612.     public boolean follow(String oname,String dname) throws Exception {
613.         long oid = this.getIdByUsername(oname);
614.         long did = this.getIdByUsername(dname);
615.         if (oid == 0 || did == 0 || oid == did){
616.             return false;
617.         }
618.         this.add("follow", oid, "name", did, dname);
619.
620.         this.add("followed", did, oid, "userid", null, oid);

```

```

621.         return true;
622.     }
623.     public boolean unfollow(String oname,String dname) throws Exception {
624.         long oid = this.getIdByUsername(oname);
625.         long did = this.getIdByUsername(dname);
626.         if (oid == 0 || did == 0 || oid == did){
627.             return false;
628.         }
629.         this.deleteColumns("follow", oid, "name", did);
630.
631.         this.deleteRow("followed", did, oid);
632.         return true;
633.     }
634.     /**
635.      * 获取关注的用户
636.      * @param username
637.      * @return
638.      * @throws Exception
639.      */
640.     public Set<String> getFollow(String username) throws Exception {
641.         Set<String> set = new HashSet<String>();
642.         long id = this.getIdByUsername(username);
643.         // HTable table = new HTable(TableName.valueOf("follow"), connection);
644.
645.         Table table = connection.getTable(TableName.valueOf("follow"));
646.         Get get = new Get(Bytes.toBytes(id));
647.         Result rs = table.get(get);
648.         for (Cell cell : rs.rawCells()) {
649.             set.add(Bytes.toString(CellUtil.cloneValue(cell)));
650.         }
651.         return set;
652.     }
653.     /**
654.      * 分享文件及文件夹
655.      * @param username
656.      * @param path
657.      * @param shareusername
658.      * @throws Exception
659.      */
660.     public void share(String dir,String username,String[] path,String[] typ
        e,String shareusername) throws Exception {
661.         long uid = getIdByUsername(username);
662.         for (int i = 0; i < path.length; i++) {
663.             long id = getGid("shareid");

```

```

663.         add("share", uid,id, "content", "dir", dir);
664.         add("share", uid,id, "content", "type", type[i]);
665.         add("share", uid,id, "content", "path", path[i]);
666.         add("share", uid,id, "content", "ts", DateUtil.DateToString("yy
yy-MM-dd HH:mm", new Date()));
667.
668.         long suid = getIdByUsername(shareusername);
669.         add("shareed", suid,uid,id, "shareid", null, uid,id);
670.     }
671. }
672. /**
673.  * 分享列表
674.  * @param name
675.  * @return
676.  * @throws Exception
677.  */
678. public List<ShareVo> getshare(String name) throws Exception {
679.     long uid = getIdByUsername(name);
680.     Scan scan = new Scan();
681.     scan.setStartRow(Bytes.toBytes(uid));
682.     scan.setStopRow(Bytes.toBytes(uid+1));
683. //     HTable share_table = new HTable(TableName.valueOf("share"), connect
ion);
684.     Table share_table = connection.getTable(TableName.valueOf("share"));
685.     ResultScanner rs = share_table.getScanner(scan);
686.     List<ShareVo> shareVos = new ArrayList<ShareVo>();
687.     ShareVo share = null;
688.     for (Result r : rs) {
689.         Cell cellPath = r.getColumnLatestCell(Bytes.toBytes("content"),
Bytes.toBytes("path"));
690.         Cell cellTs = r.getColumnLatestCell(Bytes.toBytes("content"), B
ytes.toBytes("ts"));
691.         Cell cellType = r.getColumnLatestCell(Bytes.toBytes("content"),
Bytes.toBytes("type"));
692.         Cell cellDir = r.getColumnLatestCell(Bytes.toBytes("content"),
Bytes.toBytes("dir"));
693.         share = new ShareVo();
694.         share.setShareid(Bytes.toString(r.getRow()));
695.         share.setPath(Bytes.toString(CellUtil.cloneValue(cellPath)));
696.         share.setTs(Bytes.toString(CellUtil.cloneValue(cellTs)));
697.         share.setType(Bytes.toString(CellUtil.cloneValue(cellType)));
698.         share.setDir(Bytes.toString(CellUtil.cloneValue(cellDir)));
699.         shareVos.add(share);

```

```

700.     }
701.     share_table.close();
702.     return shareVos;
703. }
704. /**
705.  * 被分享
706.  * @param username
707.  * @return
708.  * @throws Exception
709.  */
710. public List<FileSystemVo> getshareed(String username) throws Exception
711. {
712.     long uid = getIdByUsername(username);
713.     Scan scan = new Scan();
714.     scan.setStartRow(Bytes.toBytes(uid));
715.     scan.setStopRow(Bytes.toBytes(uid+1));
716.     // HTable shareed_table = new HTable(TableName.valueOf("shareed"), con
717.     nection);
718.     Table shareed_table = connection.getTable(TableName.valueOf("sharee
719.     d"));
720.     ResultScanner rs = shareed_table.getScanner(scan);
721.     // HTable share_table = new HTable(TableName.valueOf("share"), connect
722.     ion);
723.     Table share_table = connection.getTable(TableName.valueOf("share"));
724.
725.     List<FileSystemVo> fs = new ArrayList<FileSystemVo>();
726.     FileSystemVo f = null;
727.     for (Result r : rs) {
728.         Result shareRs = share_table.get(new Get(r.getValue(Bytes.toByt
729.         es("shareid"), null)));
730.         Cell cellPath = shareRs.getColumnLatestCell(Bytes.toBytes("cont
731.         ent"), Bytes.toBytes("path"));
732.         Cell cellTs = shareRs.getColumnLatestCell(Bytes.toBytes("conten
733.         t"), Bytes.toBytes("ts"));
734.         Cell cellType = shareRs.getColumnLatestCell(Bytes.toBytes("cont
735.         ent"), Bytes.toBytes("type"));
736.         Cell cellDir = shareRs.getColumnLatestCell(Bytes.toBytes("conte
737.         nt"), Bytes.toBytes("dir"));
738.         f = new FileSystemVo();
739.         // f.setShareid(Bytes.toString(shareRs.getRow()));
740.         f.setName(Bytes.toString(CellUtil.cloneValue(cellPath)));
741.         f.setDate(Bytes.toString(CellUtil.cloneValue(cellTs)));
742.         f.setType(Bytes.toString(CellUtil.cloneValue(cellType)));
743.         f.setDir(Bytes.toString(CellUtil.cloneValue(cellDir)));

```

```

734.         fs.add(f);
735.     }
736.     share_table.close();
737.     shareed_table.close();
738.     return fs;
739. }
740. /**
741.  * 新增记事本
742.  * @param username
743.  * @param content
744.  * @throws Exception
745.  */
746. public void addbook(String username,String content) throws Exception {
747.     long uid = getIdByUsername(username);
748.     long id = getGid("bookid");
749.     add("book", uid, id, "content", null, content);
750. }
751. /**
752.  * 查询记事本
753.  * @param username
754.  * @return
755.  * @throws Exception
756.  */
757. public List<bookVo> listbook(String username) throws Exception {
758.     long uid = getIdByUsername(username);
759.     Scan scan = new Scan();
760.     scan.setStartRow(Bytes.toBytes(uid));
761.     scan.setStopRow(Bytes.toBytes(uid+1));
762.     // HTable table = new HTable(TableName.valueOf("book"), connection);
763.     Table table = connection.getTable(TableName.valueOf("book"));
764.     ResultScanner rs = table.getScanner(scan);
765.     List<bookVo> books = new ArrayList<bookVo>();
766.     bookVo book = null;
767.     for (Result r : rs) {
768.         book = new bookVo();
769.         book.setId(Bytes.toString(r.getRow()));
770.         book.setContent(Bytes.toString(r.getValue(Bytes.toBytes("content"), null)));
771.         books.add(book);
772.     }
773.     table.close();
774.     return books;
775. }

```

```

776.
777.     public static void main(String[] args) throws Exception {
778. //         HbaseDB db = new HbaseDB();
779.
780.         System.out.println("ok");
781.     }
782. }

```

HDFSDB:

```

1. package com.fengdis.yun.db;
2.
3. import java.io.BufferedInputStream;
4. import java.io.FileInputStream;
5. import java.io.FileOutputStream;
6. import java.io.IOException;
7. import java.io.InputStream;
8. import java.io.OutputStream;
9. import java.util.ArrayList;
10. import java.util.List;
11.
12. import org.apache.hadoop.conf.Configuration;
13. import org.apache.hadoop.fs.FSDataInputStream;
14. import org.apache.hadoop.fs.FileStatus;
15. import org.apache.hadoop.fs.FileSystem;
16. import org.apache.hadoop.fs.FileUtil;
17. import org.apache.hadoop.fs.Path;
18. import org.apache.hadoop.io.IOUtils;
19. import org.apache.hadoop.util.Progressable;
20.
21. import com.fengdis.yun.utils.BaseUtils;
22. import com.fengdis.yun.utils.DateUtil;
23. import com.fengdis.yun.utils.FileUtils;
24. import com.fengdis.yun.utils.SiteUrl;
25. import com.fengdis.yun.vo.FileSystemVo;
26. import com.fengdis.yun.vo.Menu;
27.
28. public class HdfsDB {
29.
30.     private static String[] suf = {"csv", "txt", "doc", "docx", "xls", "xlsx", "ppt", "pptx"};
31.     private static final String ROOT = "/";
32.     static FileSystem fs;
33.     static Configuration conf;

```

```

34.
35.     private static class HdfsDBInstance {
36.         private static final HdfsDB instance = new HdfsDB();
37.     }
38.
39.     public static HdfsDB getInstance() {
40.         return HdfsDBInstance.instance;
41.     }
42.
43.     private HdfsDB() {
44.         conf = new Configuration();
45.         conf.set("fs.defaultFS", SiteUrl.readUrl("hdfs"));
46.         try {
47.             fs = FileSystem.get(conf);
48.         } catch (IOException e) {
49.             e.printStackTrace();
50.         }
51.     }
52.
53.     /**
54.      * 上传文件
55.      * @param filePath
56.      * @param dir
57.      * @throws Exception
58.      */
59.     public void upload(String filePath, String dir) throws Exception {
60.         InputStream in = new BufferedInputStream(new FileInputStream(filePath));
61.         OutputStream out = fs.create(new Path(ROOT + dir), new Progressable()
62.         {
63.             @Override
64.             public void progress() {
65.                 //System.out.println("ok");
66.             }
67.         });
68.         IOUtils.copyBytes(in, out, 4096, true);
69.     }
70.     /**
71.      * 已流形式上传
72.      * @param in
73.      * @param dir
74.      * @throws Exception
75.      */

```



```

76.     public void upload(InputStream in, String dir) throws Exception {
77.         OutputStream out = fs.create(new Path(dir), new Progressable() {
78.             @Override
79.             public void progress() {
80.                 //System.out.println("ok");
81.             }
82.         });
83.         IOUtils.copyBytes(in, out, 4096, true);
84.     }
85.     /**
86.      * 下载文件
87.      * @param path
88.      * @param local
89.      * @throws Exception
90.      */
91.     public void downLoad(String path,String local) throws Exception {
92.         FSDataInputStream in = fs.open(new Path(path));
93.         OutputStream out = new FileOutputStream(local);
94.         IOUtils.copyBytes(in, out, 4096, true);
95.     }
96.     /**
97.      * 重命名文件
98.      * @param src
99.      * @param dst
100.     * @throws Exception
101.     */
102.     public void rename(String src,String dst) throws Exception {
103.         fs.rename(new Path(src), new Path(dst));
104.     }
105.
106.     /**
107.      * 创建文件夹
108.      * @param dir
109.      * @throws Exception
110.     */
111.     public void mkdir(String dir) throws Exception {
112.         if (!fs.exists(new Path(dir))) {
113.             fs.mkdirs(new Path(dir));
114.         }
115.     }
116.     /**
117.      * 删除文件及文件夹
118.      * @param name
119.      * @throws Exception

```

```

120.     */
121.     public void delete(String name) throws Exception {
122.         fs.delete(new Path(name), true);
123.     }
124.
125.     /**
126.      * 查询文件夹
127.      * @param dir
128.      * @return
129.      * @throws Exception
130.      */
131.     public List<FileSystemVo> queryAll(String dir) throws Exception {
132.         FileStatus[] files = fs.listStatus(new Path(dir));
133.         List<FileSystemVo> fileVos = new ArrayList<FileSystemVo>();
134.         FileSystemVo f = null;
135.         for (int i = 0; i < files.length; i++) {
136.             f = new FileSystemVo();
137.             if (files[i].isDir()) {
138.                 f.setName(files[i].getPath().getName());
139.                 f.setType("D");
140.                 f.setDate(DateUtil.longToString("yyyy-MM-dd HH:mm", files[i]
141.                     .getModificationTime()));
142.                 f.setNamep(files[i].getPath().getName());
143.             } else {
144.                 f.setName(files[i].getPath().getName());
145.                 f.setType("F");
146.                 f.setDate(DateUtil.longToString("yyyy-MM-dd HH:mm", files[i]
147.                     .getModificationTime()));
148.                 f.setSize(BaseUtils.FormatFileSize(files[i].getLen()));
149.                 f.setNamep(f.getName().substring(0, f.getName().lastIndexOf
150.                     (".")));
151.                 String s=FileUtils.getFileSufix(f.getName());
152.                 for (int j = 0; j < suf.length; j++) {
153.                     if (s.equals(suf[j])) {
154.                         f.setViewflag("Y");
155.                         break;
156.                     }
157.                 }
158.                 fileVos.add(f);
159.             }
160.             return fileVos;
161.         }
162.     }
163.     /**

```

```

161.      * 移动或复制文件
162.      * @param path
163.      * @param dst
164.      * @param src true 移动文件;false 复制文件
165.      * @throws Exception
166.      */
167.      public void copy(String[] path, String dst,boolean src) throws Exceptio
n {
168.          Path[] paths = new Path[path.length];
169.          for (int i = 0; i < path.length; i++) {
170.              paths[i]=new Path(path[i]);
171.          }
172.          FileUtil.copy(fs, paths, fs, new Path(dst), src, true, conf);
173.      }
174.
175.      public List<Menu> tree(String dir) throws Exception {
176.          FileStatus[] files = fs.listStatus(new Path(dir));
177.          List<Menu> menus = new ArrayList<Menu>();
178.          for (int i = 0; i < files.length; i++) {
179.              if (files[i].isDir()) {
180.                  menus.add(new Menu(files[i].getPath().toString(), files[i].
getPath().getName()));
181.              }
182.          }
183.          return menus;
184.      }
185.
186.      public static void main(String[] args) throws Exception {
187.          HdfsDB hdfsDB = new HdfsDB();
188.          hdfsDB.mkdir(ROOT+"weir33/qq");
189.
190.          // String path = "C://Users//Administrator//Desktop//jeeshop-jeesho
p-master.zip";
191.          // hdfsDB.upload(path, "weir/"+jeeshop.zip");
192.          // hdfsDB.queryAll(ROOT);
193.          // hdfsDB.visitPath("hdfs://h1:9000/weir");
194.          // for (Menu menu : menus) {
195.          //     System.out.println(menu.getName());
196.          //     System.out.println(menu.getPname());
197.          // }
198.          // hdfsDB.delete("weirqq");
199.          // hdfsDB.mkdir("/weirqq");
200.          // hdfsDB.tree("/admin");
201.          System.out.println("ok");

```

```
202.    }
203. }
```

OpenOfficePDFConverter:

```
1. package com.fengdis.yun.pdf;
2.
3. import java.io.File;
4. import java.io.FileNotFoundException;
5.
6. import org.artofsolving.jodconverter.OfficeDocumentConverter;
7. import org.artofsolving.jodconverter.office.DefaultOfficeManagerConfiguratio
    n;
8. import org.artofsolving.jodconverter.office.OfficeManager;
9.
10. import com.fengdis.yun.utils.FileUtils;
11. import com.fengdis.yun.utils.SiteUrl;
12.
13. public class OpenOfficePDFConverter implements PDFConverter{
14.
15.     private static OfficeManager officeManager;
16.     private static int port[] = {8100};
17.
18.     public void convert2PDF(String inputFile, String pdfFile) {
19.
20.         if(inputFile.endsWith(".txt")){
21.             String odtFile = FileUtils.getFilePrefix(inputFile)+".odt";
22.             if(new File(odtFile).exists()){
23.                 //System.out.println("odt 文件已存在! ");
24.                 inputFile = odtFile;
25.             }else{
26.                 try {
27.                     FileUtils.copyFile(inputFile,odtFile);
28.                     inputFile = odtFile;
29.                 } catch (FileNotFoundException e) {
30.                     //System.out.println("文档不存在! ");
31.                     e.printStackTrace();
32.                 }
33.             }
34.         }
35.
36.         startService();
37.         System.out.println("进行文档转换转
    换:" + inputFile + " --> " + pdfFile);
```

```

38.         OfficeDocumentConverter converter = new OfficeDocumentConverter(offi
ceManager);
39.         converter.convert(new File(inputFile),new File(pdfFile));
40.         stopService();
41.         //System.out.println();
42.     }
43.
44.
45.     public void convert2PDF(String inputFile) {
46.         String pdfFile = FileUtils.getFilePrefix(inputFile)+".pdf";
47.         convert2PDF(inputFile,pdfFile);
48.     }
49.
50.     public static void startService(){
51.         DefaultOfficeManagerConfiguration configuration = new DefaultOfficeM
anagerConfiguration();
52.         try {
53.             System.out.println("准备启动服务....");
54.             configuration.setOfficeHome(SiteUrl.readUrl("openoffice")); //设置
OpenOffice.org 安装目录
55.             configuration.setPortNumbers(port); //设置转换端口，默认为 8100
56.             configuration.setTaskExecutionTimeout(1000 * 60 * 5L); //设置任务
执行超时为 5 分钟
57.             configuration.setTaskQueueTimeout(1000 * 60 * 60 * 24L); //设置任
务队列超时为 24 小时
58.
59.             officeManager = configuration.buildOfficeManager();
60.             officeManager.start(); //启动服务
61.             System.out.println("office 转换服务启动成功!");
62.         } catch (Exception ce) {
63.             System.out.println("office 转换服务启动失败!详细信息:" + ce);
64.             ce.printStackTrace();
65.         }
66.     }
67.
68.     public static void stopService(){
69.         System.out.println("关闭 office 转换服务....");
70.         if (officeManager != null) {
71.             officeManager.stop();
72.         }
73.         System.out.println("关闭 office 转换成功!");
74.     }
75.
76.     public static void main(String[] args) {

```

```

77.         OpenOfficePDFConverter op = new OpenOfficePDFConverter();
78.         op.convert2PDF("E:\\工作\\文档.docx");
79.     }
80. }

```

PDFConverter:

```

1. package com.fengdis.yun.pdf;
2.
3. public interface PDFConverter {
4.     public void convert2PDF(String inputFile,String pdfFile);
5.     public void convert2PDF(String inputFile);
6.
7. }

```

SWFConverter:

```

1. package com.fengdis.yun.pdf;
2.
3. public interface SWFConverter {
4.     public void convert2SWF(String inputFile,String swfFile);
5.     public void convert2SWF(String inputFile);
6. }

```

SWFToolsSWFConverter:

```

1. package com.fengdis.yun.pdf;
2.
3. import java.io.File;
4. import java.io.IOException;
5.
6. import com.fengdis.yun.utils.FileUtils;
7. import com.fengdis.yun.utils.SiteUrl;
8.
9. public class SWFToolsSWFConverter implements SWFConverter{
10.
11.     @Override
12.     public void convert2SWF(String inputFile, String swfFile) {
13.         File pdfFile = new File(inputFile);
14.         File outFile = new File(swfFile);
15.         if(!inputFile.endsWith(".pdf")){
16.             //System.out.println("文件格式非 PDF! ");
17.             return ;
18.         }

```

```

19.         if(!pdfFile.exists()){
20.             //System.out.println("PDF 文件不存在! ");
21.             return ;
22.         }
23.         if(outFile.exists()){
24.             //System.out.println("SWF 文件已存在! ");
25.             return ;
26.         }
27.         String command = SiteUrl.readUrl("pdf2swf") + " \""+inputFile+"\" -o
        "+swfFile+" -T 9 -f";
28.         try {
29.             //System.out.println("开始转换文档: "+inputFile);
30.             Runtime.getRuntime().exec(command);
31.             //System.out.println("成功转换为 SWF 文件! ");
32.         } catch (IOException e) {
33.             e.printStackTrace();
34.             //System.out.println("转换文档为 swf 文件失败! ");
35.         }
36.
37.     }
38.
39.     @Override
40.     public void convert2SWF(String inputFile) {
41.         String swfFile = FileUtils.getFilePrefix(inputFile)+".swf";
42.         convert2SWF(inputFile,swfFile);
43.     }
44.
45.     public static void main(String[] args) {
46.         SWFToolsSWFConverter swf = new SWFToolsSWFConverter();
47.         swf.convert2SWF("D:\\activiti.pdf");
48.     }
49. }

```

FileViewer:

```

1. package com.fengdis.yun.test;
2.
3. import java.io.*;
4. import java.util.ArrayList;
5. import java.util.Iterator;
6. import java.util.List;
7.
8. /**
9.  * 读取目录及子目录下指定文件名的路径 并放到一个数组里面返回遍历

```

```

10. */
11. public class FileViewer {
12.     public static void main(String[] args) {
13.         List arrayList = FileViewer.getListFiles("E:/baiduyundownload", null,
            true);
14.
15.         if (arrayList.isEmpty()) {
16.             System.out.println("没有符号要求的文件");
17.         } else {
18.             String message = "";
19.             message += "符号要求的文件数: " + arrayList.size() + "\r\n";
20.             System.out.println(message);
21.
22.             for (Iterator i = arrayList.iterator(); i.hasNext(); ) {
23.                 String temp = (String) i.next();
24.                 System.out.println(temp);
25.                 message += temp + "\r\n";
26.             }
27.
28.             // appendMethod("d:/ajax/menu.txt",message);
29.         }
30.     }
31.
32.     public static List<String> fileList = new ArrayList<String>();
33.
34.     /**
35.      *
36.      * @param path
37.      *      文件路径
38.      * @param suffix
39.      *      后缀名
40.      * @param isdepth
41.      *      是否遍历子目录
42.      * @return
43.      */
44.     public static List getListFiles(String path, String suffix, boolean isde
        pth) {
45.         File file = new File(path);
46.         return FileViewer.listFiles(file, suffix, isdepth);
47.     }
48.
49.     public static List listFile(File f, String suffix, boolean isdepth) {
50.         // 是目录, 同时需要遍历子目录
51.         if (f.isDirectory() && isdepth == true) {

```



```

52.         File[] t = f.listFiles();
53.         for (int i = 0; i < t.length; i++) {
54.             listFile(t[i], suffix, isdepth);
55.         }
56.     } else {
57.         String filePath = f.getAbsolutePath();
58.
59.         if (suffix != null) {
60.             int begIndex = filePath.lastIndexOf(".");// 最后一个.(即后缀名
           前面的.)的索引
61.             String tempsuffix = "";
62.
63.             if (begIndex != -1)// 防止是文件但却没有后缀名结束的文件
64.             {
65.                 tempsuffix = filePath.substring(begIndex + 1, filePath.l
           ength());
66.             }
67.
68.             if (tempsuffix.equals(suffix)) {
69.                 fileList.add(filePath);
70.             }
71.         } else {
72.             // 后缀名为 null 则为所有文件
73.             fileList.add(filePath);
74.         }
75.
76.     }
77.
78.     return fileList;
79. }
80.
81. /**
82.  * 方法追加文件: 使用 FileWriter
83.  *
84.  * @param fileName
85.  * @param content
86.  */
87. public static void appendMethod(String fileName, String content) {
88.     try {
89.         // 打开一个写文件器, 构造函数中的第二个参数 true 表示以追加形式写文件
90.         FileWriter writer = new FileWriter(fileName, true);
91.         writer.write(content + "\r\n");
92.         writer.close();
93.     } catch (IOException e) {

```

```

94.         e.printStackTrace();
95.     }
96. }
97. }

```

BaseUtils:

```

1. package com.fengdis.yun.utils;
2.
3. import java.text.DecimalFormat;
4.
5. public class BaseUtils {
6.     public static String FormetFileSize(long fileS) {
7.         DecimalFormat df = new DecimalFormat("#.00");
8.         String fileSizeString = "";
9.         if (fileS < 1024) {
10.            fileSizeString = df.format((double) fileS) + "B";
11.        } else if (fileS < 1048576) {
12.            fileSizeString = df.format((double) fileS / 1024) + "K";
13.        } else if (fileS < 1073741824) {
14.            fileSizeString = df.format((double) fileS / 1048576) + "M";
15.        } else {
16.            fileSizeString = df.format((double) fileS / 1073741824) + "G";
17.        }
18.        return fileSizeString;
19.    }
20.
21.    public static boolean isEmpty(String str) {
22.        if(str!=null && !"".equals(str.trim())){
23.            return true;
24.        }else{
25.            return false;
26.        }
27.    }
28.
29.    public static String getUrl(String str) {
30.        String[] ss = str.split("/");
31.        if(ss.length>2){
32.            StringBuffer sb = new StringBuffer();
33.            for (int i = 2; i < ss.length; i++) {
34.                sb.append("<a href='cloud/list.do?name="+str.substring(0, str.
                    indexOf(ss[i])-1)+"'>"+ss[i-1]+"</a>").append("|");
35.            }
36.            return sb.deleteCharAt(sb.length()-1).toString();

```

```

37.     }
38.     return null;
39. }
40.
41. public static void main(String[] args) {
42.     // System.out.println(isNotEmpty("null"));
43.     // System.out.println(getUrl("/admin/sss/asdsd"));
44.     // String s = "dsdsda.zip";
45.     String s = "dsdsdawwzip";
46.     System.out.println(s.lastIndexOf("."));
47. }
48. }

```

DateUtil:

```

1. package com.fengdis.yun.utils;
2.
3. import java.text.SimpleDateFormat;
4. import java.util.Date;
5.
6. public class DateUtil {
7.
8.     public static String longToString(String dateFormat, Long millSec) {
9.         SimpleDateFormat sdf = new SimpleDateFormat(dateFormat);
10.        Date date= new Date(millSec);
11.        return sdf.format(date);
12.    }
13.    public static String DateToString(String dateFormat, Date date) {
14.        SimpleDateFormat sdf = new SimpleDateFormat(dateFormat);
15.        return sdf.format(date);
16.    }
17.
18.    public static void main(String[] args) {
19.        System.out.println(DateUtil.DateToString("yyyy-MM-dd HH:mm:ss", new
        Date()));
20.    }
21. }

```

FileUtils:

```

1. package com.fengdis.yun.utils;
2.
3. import java.io.File;
4. import java.io.FileInputStream;

```

```

5. import java.io.FileNotFoundException;
6. import java.io.FileOutputStream;
7. import java.io.IOException;
8.
9. public class FileUtils {
10.
11.     public static String getFilePrefix(String fileName){
12.         int splitIndex = fileName.lastIndexOf(".");
13.         return fileName.substring(0, splitIndex);
14.     }
15.
16.     public static String getFileSufix(String fileName){
17.         int splitIndex = fileName.lastIndexOf(".");
18.         return fileName.substring(splitIndex + 1);
19.     }
20.
21.     public static void copyFile(String inputFile,String outputFile) throws F
        ileNotFoundException{
22.         File sFile = new File(inputFile);
23.         File tFile = new File(outputFile);
24.         FileInputStream fis = new FileInputStream(sFile);
25.         FileOutputStream fos = new FileOutputStream(tFile);
26.         int temp = 0;
27.         try {
28.             while ((temp = fis.read()) != -1) {
29.                 fos.write(temp);
30.             }
31.         } catch (IOException e) {
32.             e.printStackTrace();
33.         } finally{
34.             try {
35.                 fis.close();
36.                 fos.close();
37.             } catch (IOException e) {
38.                 e.printStackTrace();
39.             }
40.         }
41.     }
42.     public static void main(String[] args) {
43.         String s ="dsdsdsd.rar";
44.         System.out.println(getFileSufix(s));
45.     }
46. }

```

JSON:

```
1. package com.fengdis.yun.utils;
2.
3. public class Json{
4.
5.     private boolean success = false;
6.
7.     private String msg = "";
8.
9.     private Object obj = null;
10.
11.     public boolean isSuccess() {
12.         return success;
13.     }
14.
15.     public void setSuccess(boolean success) {
16.         this.success = success;
17.     }
18.
19.     public String getMsg() {
20.         return msg;
21.     }
22.
23.     public void setMsg(String msg) {
24.         this.msg = msg;
25.     }
26.
27.     public Object getObj() {
28.         return obj;
29.     }
30.
31.     public void setObj(Object obj) {
32.         this.obj = obj;
33.     }
34.
35. }
```

SiteUrl:

```
1. package com.fengdis.yun.utils;
2.
3. import java.io.IOException;
4. import java.util.Properties;
```

```

5.
6. public class SiteUrl {
7.     private static Properties properties = new Properties();
8.     static{
9.         try {
10.             properties.load(SiteUrl.class.getClassLoader().getResourceAsStream("db.properties"));
11.         } catch (IOException e) {
12.             e.printStackTrace();
13.         }
14.     }
15.
16.     public static String readUrl(String key){
17.         return (String)properties.get(key);
18.     }
19. }

```

Index View:

```

1. <%@ page language="java" pageEncoding="UTF-8"%>
2. <%@ include file="/public/taglib.jsp" %>
3. <%
4. String path = request.getContextPath();
5. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
6. %>
7. <!DOCTYPE HTML>
8. <html>
9.     <head><base href="<%=basePath%>">
10.         <title>云盘首页</title>
11.         <jsp:include page="/public/pub.jsp"></jsp:include>
12.     </head>
13.     <body>
14.         <jsp:include page="/public/top.jsp"></jsp:include>
15.
16.         <div class="main-container" id="main-container">
17.             <script type="text/javascript">
18.                 try{ace.settings.check('main-container' , 'fixed')}catch(e){}
19.             </script>
20.
21.             <div class="main-container-inner">
22.                 <a class="menu-toggler" id="menu-toggler" href="#">
23.                     <span class="menu-text"></span>

```

```

24.         </a>
25.
26.         <jsp:include page="/public/left.jsp"></jsp:include>
27.
28.         <div class="main-content">
29.             <div class="breadcrumbs" id="breadcrumbs">
30.                 <script type="text/javascript">
31.                     try{ace.settings.check('breadcrumbs' , 'fixed')}
catch(e){}
32.                 </script>
33.
34.                 <ul class="breadcrumb">
35.                     <li>
36.                         <i class="icon-home home-icon"></i>
37.                         <a href="#">首页</a>
38.                     </li>
39.                     <li class="active">欢迎页面</li>
40.                 </ul><!-- .breadcrumb -->
41.
42.                 <div class="nav-search" id="nav-search">
43.                     <form class="form-search">
44.                         <span class="input-icon">
45.                             <input type="text" placeholder="Search ..
46.                             ." class="nav-search-input" id="nav-search-input" autocomplete="off" />
47.                             <i class="icon-search nav-search-icon"><
/i>
48.                         </span>
49.                     </form>
50.                 </div><!-- #nav-search -->
51.
52.                 <div class="page-content">
53.                     <div class="page-header">
54.                         <h1>
55.                             欢迎进入云盘系统
56.                         </h1>
57.                     </div><!-- /.page-header -->
58.
59.                     <div class="row">
60.                         <div class="col-xs-12">
61.                             <!-- PAGE CONTENT BEGINS -->
62.
63.
64.                             sdsadsdsadad

```

```

65.         <div class="hr hr32 hr-dotted"></div>
66.
67.
68.         <div class="hr hr32 hr-dotted"></div>
69.
70.
71.         <!-- PAGE CONTENT ENDS -->
72.         </div><!-- /.col -->
73.         </div><!-- /.row -->
74.         </div><!-- /.page-content -->
75.         </div><!-- /.main-content -->
76.
77.         <jsp:include page="/public/container.jsp"></jsp:include>
78.     </div><!-- /.main-container-inner -->
79.
80.     <a href="#" id="btn-scroll-up" class="btn-scroll-up btn btn-sm b
tn-inverse">
81.         <i class="icon-double-angle-up icon-only bigger-110"></i>
82.     </a>
83. </div><!-- /.main-container -->
84. <script type="text/javascript">
85.     jQuery(function($) {
86.         $('#tasks').sortable({
87.             opacity:0.8,
88.             revert:true,
89.             forceHelperSize:true,
90.             placeholder: 'draggable-placeholder',
91.             forcePlaceholderSize:true,
92.             tolerance:'pointer',
93.             stop: function( event, ui ) {//just for Chrome!!!! so th
at dropdowns on items don't appear below other items after being moved
94.                 $(ui.item).css('z-index', 'auto');
95.             }
96.         }
97.     );
98.     $('#tasks').disableSelection();
99.     $('#tasks input:checkbox').removeAttr('checked').on('click',
function(){
100.         if(this.checked) $(this).closest('li').addClass('select
ed');
101.         else $(this).closest('li').removeClass('selected');
102.     });
103. })
104. </script>

```



```
105.     </body>
106. </html>
```

Login View:

```
1. <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
2. <%
3. String path = request.getContextPath();
4. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
5. %>
6.
7. <!DOCTYPE HTML>
8. <html>
9. <head>
10. <base href="<%=basePath%%">
11.
12. <title>登录云盘</title>
13. <!-- basic styles -->
14.
15. <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
16. <link rel="stylesheet" href="assets/css/font-awesome.min.css" />
17.
18. <!--[if IE 7]>
19.     <link rel="stylesheet" href="assets/css/font-awesome-ie7.min.css"
20.     />
21. <![endif]-->
22. <!-- page specific plugin styles -->
23.
24. <!-- ace styles -->
25.
26. <link rel="stylesheet" href="assets/css/ace.min.css" />
27. <link rel="stylesheet" href="assets/css/ace-rtl.min.css" />
28.
29. <!--[if lte IE 8]>
30.     <link rel="stylesheet" href="assets/css/ace-ie.min.css" />
31. <![endif]-->
32.
33. <!-- inline styles related to this page -->
34.
35. <!-- HTML5 shim and Respond.js IE8 support of HTML5 elements and media queries -->
36.
```

```

37. <!--[if lt IE 9]>
38.     <script src="assets/js/html5shiv.js"></script>
39.     <script src="assets/js/respond.min.js"></script>
40. <![endif]-->
41. </head>
42.
43. <body class="login-layout">
44.     <div class="main-container">
45.         <div class="main-content">
46.             <div class="row">
47.                 <div class="col-sm-10 col-sm-offset-1">
48.                     <div class="login-container">
49.
50.                         <div class="position-relative">
51.                             <div id="login-box"
52.                                 class="login-box visible widget-box no-borde
53.                                     r">
54.                                 <div class="widget-body">
55.                                     <div class="widget-main">
56.                                         <h4 class="header blue lighter bigge
57.                                             r">
58.                                             <i class="icon-coffee green"></i>
59.                                             > 登陆页面
60.
61.                                             </h4>
62.
63.                                             <div class="space-6"></div>
64.
65.                                             <form action="login.do" method="post
66.                                                 ">
67.
68.                                                 <fieldset>
69.                                                     <label class="block clearfix
70.                                                         "> <span
71.
72.                                                         class="block input-icon
73.                                                         input-icon-right"> <input
74.
75.                                                         type="text" name="us
76.                                                         erName" class="form-control" placeholder="用户名" /> <i
77.
78.                                                         class="icon-user"></i>
79.
80.                                                         <i>
81.
82.                                                         </span>
83.
84.                                                         </label> <label class="block
85.
86.                                                         clearfix"> <span
87.
88.                                                         class="block input-icon
89.                                                         input-icon-right"> <input

```

```

70.                                     type="password" name
    ="pwd" class="form-control" placeholder="密码" />
71.                                     <i class="icon-lock"
    ></i>
72.                                     </span>
73.                                     </label>
74.
75.                                     <div class="space"></div>
76.
77.                                     <div class="clearfix">
78.                                     <label class="inline"> <
    input type="checkbox"
79.                                     class="ace" /> <span
    class="lbl"> 记住我</span>
80.                                     </label>
81.
82.                                     <button type="submit"
83.                                     class="width-35 pull
    -right btn btn-sm btn-primary">
84.                                     <i class="icon-key">
    </i> 登陆
85.                                     </button>
86.                                     </div>
87.
88.                                     <div class="space-4"></div>
89.                                     </fieldset>
90.                                     </form>
91.
92.                                </div>
93.                                <!-- /widget-main -->
94.
95.                                <div class="toolbar clearfix">
96.                                <div>
97.                                <a href="#" onclick="show_box('f
    orgot-box'); return false;"
98.                                class="forgot-password-link"
    > <i class="icon-arrow-left"></i>
99.                                忘记密码
100.                                </a>
101.                                </div>
102.
103.                                <div>

```

```

104.             <a href="#" onclick="show_box('
    signup-box'); return false;"
105.                 class="user-signup-link">
    注册 <i class="icon-arrow-right"></i>
106.             </a>
107.         </div>
108.     </div>
109. </div>
110. <!-- /widget-body -->
111. </div>
112. <!-- /login-box -->
113.
114.     <div id="forgot-box" class="forgot-box widget-b
    ox no-border">
115.         <div class="widget-body">
116.             <div class="widget-main">
117.                 <h4 class="header red lighter bigge
    r">
118.                     <i class="icon-key"></i> 找回密码
    码
119.                 </h4>
120.
121.                 <div class="space-6"></div>
122.                 <p>输入你的注册邮箱</p>
123.
124.                 <form>
125.                     <fieldset>
126.                         <label class="block clearfi
    x"> <span
127.                             class="block input-icon
    input-icon-right"> <input
128.                                 type="email" class=
    "form-control" placeholder="邮箱" /> <i
129.                                     class="icon-envelop
    e"></i>
130.                         </span>
131.                         </label>
132.
133.                         <div class="clearfix">
134.                             <button type="button"
135.                                 class="width-35 pul
    l-right btn btn-sm btn-danger">
136.                                 <i class="icon-ligh
    tbulb"></i> 发送

```

```

137.                                     </button>
138.                                 </div>
139.                             </fieldset>
140.                         </form>
141.                    </div>
142.                <!-- /widget-main -->
143.
144.                <div class="toolbar center">
145.                    <a href="#" onclick="show_box('logi
n-box'); return false;"
146.                        class="back-to-login-link"> 回
到登陆 <i
147.                            class="icon-arrow-right"></i>
148.                        </a>
149.                    </div>
150.                </div>
151.            <!-- /widget-body -->
152.        </div>
153.    <!-- /forgot-box -->
154.
155.    <div id="signup-box" class="signup-box widget-b
ox no-border">
156.        <div class="widget-body">
157.            <div class="widget-main">
158.                <h4 class="header green lighter big
ger">
159.                    <i class="icon-group blue"></i>
注册用户
160.                </h4>
161.
162.                <form action="reg.do" method="post"
>
163.                    <fieldset>
164.                        <label class="block clearfi
x"> <span
165.                            class="block input-icon
input-icon-right"> <input
166.                                type="email" name="
email" class="form-control" placeholder="邮箱" /> <i
167.                                    class="icon-envelop
e"></i>
168.                        </span>
169.                        </label> <label class="bloc
k clearfix"> <span

```

```

170.                                     class="block input-icon
    input-icon-right"> <input
171.                                     type="text" name="u
    sername" class="form-control" placeholder="用户名" /> <i
172.                                     class="icon-user"><
    /i>
173.                                     </span>
174.                                     </label> <label class="bloc
    k clearfix"> <span
175.                                     class="block input-icon
    input-icon-right"> <input
176.                                     type="password" nam
    e="password" class="form-control" placeholder="密码" />
177.                                     <i class="icon-lock
    "></i>
178.                                     </span>
179.                                     </label>
180.
181.                                     <div class="space-24"></div
    >
182.
183.                                     <div class="clearfix">
184.                                     <button type="reset" cl
    ass="width-30 pull-left btn btn-sm">
185.                                     <i class="icon-refr
    esh"></i> 重置
186.                                     </button>
187.
188.                                     <button type="submit"
189.                                     class="width-65 pul
    l-right btn btn-sm btn-success">
190.                                     注
    册 <i class="icon-arrow-right icon-on-right"></i>
191.                                     </button>
192.                                     </div>
193.                                     </fieldset>
194.                                     </form>
195.                                     </div>
196.
197.                                     <div class="toolbar center">
198.                                     <a href="#" onclick="show_box('logi
    n-box'); return false;"
199.                                     class="back-to-login-link"> <i
    class="icon-arrow-left"></i>

```

```

200.             回到登陆
201.             </a>
202.         </div>
203.     </div>
204.     <!-- /widget-body -->
205. </div>
206. <!-- /signup-box -->
207. </div>
208. <!-- /position-relative -->
209. </div>
210. </div>
211. <!-- /.col -->
212. </div>
213. <!-- /.row -->
214. </div>
215. </div>
216. <!-- /.main-container -->
217.
218. <!-- basic scripts -->
219.
220. <!--[if !IE]> -->
221.
222.     <script src="assets/js/jquery-2.0.3.min.js"></script>
223.
224. <!-- <![endif]-->
225.
226. <!--[if IE]>
227. <script src="assets/js/jquery-1.10.2.min.js"></script>
228. <![endif]-->
229.
230. <!--[if !IE]> -->
231.
232.     <script type="text/javascript">
233.         window.jQuery || document.write("<script src='assets/js/jquery-
234.         2.0.3.min.js'>"+"/script>");
235.     </script>
236. <!-- <![endif]-->
237.
238. <!--[if IE]>
239. <script type="text/javascript">
240.     window.jQuery || document.write("<script src='assets/js/jquery-1.10.2.min.
241.     js'>"+"/script>");
242. </script>

```

```

242. <![endif]-->
243.
244.     <script type="text/javascript">
245.         if("ontouchend" in document) document.write("<script src='asset
s/js/jquery.mobile.custom.min.js'>"+ "<"+"/script>");
246.     </script>
247.
248.     <!-- inline scripts related to this page -->
249.
250.     <script type="text/javascript">
251.         function show_box(id) {
252.             jQuery('.widget-box.visible').removeClass('visible');
253.             jQuery('#'+id).addClass('visible');
254.         }
255.     </script>
256. </body>
257. </html>

```

Down View:

```

1. <%@ page language="java" pageEncoding="UTF-8"%>
2. <%@ include file="/public/taglib.jsp"%>
3. <%
4. String path = request.getContextPath();
5. String basePath = request.getScheme()+ "://" + request.getServerName() + ":" + request.getServerPort() + path + "/";
6. %>
7. <!DOCTYPE html>
8. <html>
9. <head><base href="<%=basePath%%">
10. <title>下载文件</title>
11. <meta name="content-type" content="text/html; charset=UTF-8">
12. </head>
13.
14. <body>
15.     <a href="test/${name}">${name}</a>
16. </body>
17. </html>

```

Share View:

```

1. <%@ page language="java" pageEncoding="UTF-8"%>
2. <%@ include file="/public/taglib.jsp" %>
3. <%

```



```

4. String path = request.getContextPath();
5. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
6. %>
7.
8. <!DOCTYPE HTML>
9. <html>
10. <head>
11. <base href="%=basePath%">
12.
13. <title>云盘首页</title>
14. <jsp:include page="/public/pub.jsp"></jsp:include>
15. </head>
16.
17. <body>
18. <jsp:include page="/public/top.jsp"></jsp:include>
19.     <div class="main-container" id="main-container">
20.         <script type="text/javascript">
21.             try{ace.settings.check('main-container' , 'fixed')}catch(e){}
22.
23.         </script>
24.
25.         <div class="main-container-inner">
26.             <a class="menu-toggler" id="menu-toggler" href="#"> <span class="menu-text"></span>
27.             </a>
28.             <jsp:include page="/public/left.jsp"></jsp:include>
29.             <div class="main-content">
30.                 <div class="breadcrumbs" id="breadcrumbs">
31.                     <script type="text/javascript">
32.                         try{ace.settings.check('breadcrumbs' , 'fixed')}
33.                         catch(e){}
34.                     </script>
35.
36.                     <ul class="breadcrumb">
37.                         <input type="button" class="button purple" value="上传文件" onclick="upload()" />
38.                         <input type="button" class="button blue" value="创建文件夹" onclick="mkdir()" />
39.                         <input type="button" class="button darkblue" value="删除" onclick="deldir()" />
40.                         <input type="button" class="button teal" value="分享" onclick="share()" />

```

```

40.             <!-- <li><i class="icon-home home-icon"></i> <a href
    =="#">首页</a></li>
41.             <li class="active">我的网盘</li> -->
42.         </ul>
43.         <!-- .breadcrumb -->
44.
45.     </div>
46.
47.     <div class="page-content">
48.         <script type="text/javascript">
49.             function mkdir(){
50.                 var str = '';
51.                 str += "<div><input type='text' id='mkdir' nam
    e='mkdir' placeholder='文件名' />";
52.                 str += "<input class='button purple' type='but
    ton' onclick='addDir()' value='提交' /> ";
53.                 str += "<input class='button black' type='butt
    on' onclick='removeDir()' value='取消' /></div>";
54.                 $("#mkdirForm").append(str);
55.             }
56.             function addDir(){
57.                 var mkdir = $.trim($("#mkdir").val());
58.                 if(mkdir==null || mkdir==''){
59.                     alert("请输入文件夹名称");
60.                     return false;
61.                 }
62.                 var data = $("#mkdirForm").formToArray();
63.                 $.ajax({
64.                     type:"POST",
65.                     url:"cloud/mkdir.do",
66.                     data:data,
67.                     dataType:"json",
68.                     success:function(json){
69.                         if(json.success){
70.                             $("#mkdirForm").empty();
71.                             /* var str = '';
72.                             str += "<tr>";
73.                             str += "<td><input type='checkbox
    ' value='"+json.obj.id+"' />"+json.obj.name+"</td>";
74.                             str += "<td>文件夹</td>";
75.                             str += "<td>"+json.obj.date+"</td
    >";
76.                             str += "</tr>";

```

```

77.                                     $("#listdir tr:eq(0)").before(st
    r); */
78.                                     location.reload();
79.                                     }else{
80.                                     alert(json.msg);
81.                                     }
82.
83.                                     }
84.                                     });
85.                                     }
86.                                     function removeDir(){
87.                                     $("#mkdirForm").empty();
88.                                     }
89.                                     function selectBox(){
90.                                     $("#listdir input[type=checkbox]").each(func
    tion(){
91.                                     if(this.checked==true){
92.                                     this.checked=false;
93.                                     }else{
94.                                     this.checked=true;
95.                                     }
96.                                     });
97.                                     }
98.                                     function deldir(){
99.                                     var dir = $("#dir").val();
100.                                    var ids = [];
101.                                    $("#listdir input[type=checkbox]").each(func
    tion(){
102.                                    if(this.checked==true){
103.                                    ids.push(this.value);
104.                                    }
105.                                    });
106.                                    if(ids.length>0){
107.                                    $( "#dialog-confirm" ).removeClass('hide
        ').dialog({
108.                                    resizable: false,
109.                                    modal: true,
110.                                    title: "删除提醒",
111.                                    title_html: true,
112.                                    buttons: [
113.                                    {
114.                                    text: "取消",
115.                                    "class" : "btn btn-xs",
116.                                    click: function() {

```

```

117.                                     $( this ).dialog( "clos
    e" );
118.                                     }
119.                                     },
120.                                     {
121.                                         text: "确定",
122.                                         "class" : "btn btn-primary
        btn-xs",
123.                                         click: function() {
124.                                             $.post('${pageContext.r
        equest.contextPath}/cloud/delete.do', {ids:ids.join(',') ,dir:dir}, function(
        j) {
125.                                                 if (j.success) {
126.                                                     location.reload
        ();
127.                                                 }else{
128.                                                     alert(json.msg);
129.                                                 }
130.                                             }, 'json');
131.                                         }
132.                                     }
133.                                 ]
134.                            });
135.                        }else{
136.                            alert("你没有选择");
137.                        }
138.                    }
139.                    $(document).ready(function(){
140.                        $('table th input:checkbox').on('click' , fu
        nction(){
141.                            var that = this;
142.                            $(this).closest('table').find('tr > td:
        first-child input:checkbox')
143.                                .each(function(){
144.                                    this.checked = that.checked;
145.                                    $(this).closest('tr').toggleClass('
        selected');
146.                                });
147.
148.                            });
149.                        });
150.                        function upload(){
151.                            var dir = $("#dir").val();

```

```

152.                //console.info(dir);
153.                $.layer({
154.                    type: 2,
155.                    border: [0],
156.                    title: false,
157.                    closeBtn: [0, true],
158.                    iframe: {src : 'cloud/upload11.jsp?dir=
'+dir},
159.                    area: ['510px', '450px']
160.                });
161.            }
162.            function editName(index){
163.                $("#edit01"+index).hide();
164.                $("#edit02"+index).show();
165.            }
166.            function removeBut(index){
167.                $("#edit02"+index).hide();
168.                $("#edit01"+index).show();
169.            }
170.            function renameBut(index,name,type){
171.                var dir = $("#dir").val();
172.                var newname = $.trim($("#rename"+index).val()
);
173.                if(newname==null || newname==''){
174.                    alert("请输入名称");
175.                    return false;
176.                }
177.                $.post('${pageContext.request.contextPath}/c
loud/rename.do', {dir:dir,name:name,rename:newname,type:type}, function(j) {
178.                    if (j.success) {
179.                        location.reload();
180.                    }else{
181.                        alert(json.msg);
182.                    }
183.                }, 'json');
184.                //$("#edit02"+index).hide();
185.                //$("#edit01"+index).show();
186.            }
187.            function viewName(index,name){
188.                var dir = $("#dir").val();
189.                $.layer({
190.                    type: 2,
191.                    border: [0],

```

```

192.                                     title: false,
193.                                     closeBtn: [0, true],
194.                                     iframe: {src : 'cloud/view.do?dir='+dir
    + '&name='+name},
195.                                     area: ['950px', '650px']
196.                                 });
197.                            }
198.                            function share(){
199.                                var dir = $("#dir").val();
200.                                var ids = [];
201.                                $("#listdir input[type=checkbox]").each(func
    tion(){
202.                                    if(this.checked==true){
203.                                        ids.push(this.value);
204.                                    }
205.                                });
206.                                if(ids.length>0){
207.                                    $.layer({
208.                                        type: 2,
209.                                        border: [0],
210.                                        title: '选择要分享的用户',
211.                                        closeBtn: [0, true],
212.                                        iframe: {src : 'user/getFollow.do?i
    ds='+ids+'&dir='+dir},
213.                                        area: ['860px', '400px']
214.                                    });
215.                                }else{
216.                                    alert("你没有选择");
217.                                }
218.                            }
219.                            </script>
220.                            <div class="page-header">
221.                                <h1>${url}</h1>
222.                            </div>
223.                            <!-- /.page-header -->
224.
225.                            <div class="row">
226.                                <div class="col-xs-12">
227.
228.                                    <div id="dialog-confirm" class="hide">
229.
230.                                        <p class="bigger-110 bolder center grey">
231.                                            <i class="icon-hand-right blue bigger-1
    20"></i> 你确定要删除么?

```

```

232.             </p>
233.         </div>
234.         <!-- PAGE CONTENT BEGINS -->
235.         <div>
236.             <form id="mkdirForm">
237.                 <input type="hidden" id="dir" name="dir
Name" value="\${dir}" />
238.             </form>
239.         </div>
240.         <table id="sample-table-1" class="table table-s
triped table-bordered table-hover">
241.             <thead>
242.                 <tr>
243.                     <th class="center"><label><input ty
pe="checkbox" class="ace" autocomplete="off"/> <span class="lb1" ></span> </
label></th>
244.                     <th>分享文件</th>
245.                     <th>分享日期</th>
246.                 </tr>
247.             </thead>
248.             <tbody id="listdir">
249.                 <c:forEach items="\${shares}" var="entry
" varStatus="sta">
250.                     <tr>
251.                         <td class="center"><label> <inp
ut type="checkbox" class="ace" value="" autocomplete="off"/> <span class="lb
1"></span>
252.                         </label></td>
253.                         <td><div id="edit01\${sta.index}
">
254.                             \${entry.path}
255.                         </div>
256.                         </td>
257.                         <td>\${entry.ts}</td>
258.                     </tr>
259.                 </c:forEach>
260.             </tbody>
261.         </table>
262.         <!-- PAGE CONTENT ENDS -->
263.     </div>
264. <!-- /.col -->
265. </div>
266. <!-- /.row -->
267. </div>

```

```

268.         <!-- /.page-content -->
269.     </div>
270.     <!-- /.main-content -->
271.
272.     <jsp:include page="/public/container.jsp"></jsp:include>
273. </div>
274. <!-- /.main-container-inner -->
275.
276.     <a href="#" id="btn-scroll-up" class="btn-scroll-up btn btn-sm btn-
inverse"> <i class="icon-double-angle-up icon-only bigger-110"></i>
277.     </a>
278. </div>
279. <!-- /.main-container -->
280. <script type="text/javascript">
281.     jQuery(function($) {
282.         $('#tasks').sortable({
283.             opacity:0.8,
284.             revert:true,
285.             forceHelperSize:true,
286.             placeholder: 'draggable-placeholder',
287.             forcePlaceholderSize:true,
288.             tolerance:'pointer',
289.             stop: function( event, ui ) {///just for Chrome!!!! so t
hat dropdowns on items don't appear below other items after being moved
290.                 $(ui.item).css('z-index', 'auto');
291.             }
292.         });
293.     });
294.     $('#tasks').disableSelection();
295.     $('#tasks input:checkbox').removeAttr('checked').on('click',
function(){
296.         if(this.checked) $(this).closest('li').addClass('select
ed');
297.         else $(this).closest('li').removeClass('selected');
298.     });
299. })
300. </script>
301. </body>
302. </html>

```

Upload View:

```

1. <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
2. <%

```



```

3. String path = request.getContextPath();
4. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
5. %>
6. <!DOCTYPE HTML>
7. <html>
8.     <head><base href="<%=basePath%>">
9.         <title>文件上传</title>
10.    <link rel="stylesheet" href="js/plupload-2.1.2/js/jquery.plupload.queue/css/jquery.plupload.queue.css" type="text/css" media="screen" />
11.    <script src="js/jquery.min.js"></script>
12.    <script type="text/javascript" src="js/plupload-2.1.2/js/plupload.full.min.js"></script>
13.    <script type="text/javascript" src="js/plupload-2.1.2/js/jquery.plupload.queue/jquery.plupload.queue.min.js"></script>
14.    <script type="text/javascript" src="js/plupload-2.1.2/js/i18n/zh_CN.js"></script>
15.    <body style="padding: 0;margin: 0;">
16.        <div id="uploader"> </div>
17.    <script type="text/javascript">
18. var files = [];
19. var errors = [];
20. //var type = 'file';
21. //var chunk = eval('${param.chunk}');
22. //var dir = eval('${param.dir}');
23. //var max_file_size = '5mb';
24. //var filters = {title : "文档", extensions : "zip,doc,docx,xls,xlsx,ppt,pptx"};
25. $("#uploader").pluploadQueue($.extend({
26.     runtimes : 'flash,html4,html5',
27.     url : '${pageContext.request.contextPath}/cloud/upload.do',
28.     //max_file_size : max_file_size,
29.     //file_data_name:'file',
30.     unique_names:true,
31.     //filters : [],
32.     flash_swf_url : 'js/Moxie.swf',
33.     init:{
34.         BeforeUpload: function(uploader, file) {
35.             uploader.setOption('url','${pageContext.request.contextPath}/cloud/upload.do?dir=${param.dir}');
36.         },
37.         FileUploaded:function(uploader,file,response){
38.             if(response.response){
39.                 var rs = $.parseJSON(response.response);

```

```

40.         if(rs.status){
41.             files.push(file.name);
42.         }else{
43.             errors.push(file.name);
44.         }
45.     }
46. },
47.     UploadComplete:function(uploader,fs){
48.         var e= errors.length ? ",失败"+errors.length+"个("+errors.join(",
            ")+" )。 " : "。 ";
49.         alert("上传完成! 共"+fs.length+"个。成功"+files.length+e);
50.         target.window("close");
51.     }
52. }
53. }));
54.
55. </script>
56. </body>
57. </html>

```

Tree View:

```

1. <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
2. <%
3. String path = request.getContextPath();
4. String basePath = request.getScheme()+ "://" +request.getServerName()+":"+requ
    est.getServerPort()+path+"/";
5. %>
6.
7. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
8. <html>
9.     <head>
10.         <base href="<%=basePath%%">
11.
12.         <title>My JSP '22.jsp' starting page</title>
13.         <script type="text/javascript" src="js/jquery-2.1.1.min.js"></script>
14.         <link rel="stylesheet" type="text/css" href="js/easyui1.4/themes/default
            /easyui.css">
15.         <link rel="stylesheet" type="text/css" href="js/easyui1.4/themes/icon.cs
            s">
16.         <script type="text/javascript" src="js/easyui1.4/jquery.easyui.min.js"><
            /script>
17.         <script type="text/javascript" src="js/easyui1.4/locale/easyui-lang-zh_C
            N.js"></script>

```

```

18.     <link rel="stylesheet" type="text/css" href="css/buttons.css">
19. <SCRIPT type="text/javascript">
20. var index = parent.layer.getFrameIndex(window.name); //获取当前窗体索引
21. $(function (){
22.     $('#tt').tree({
23.         url:'cloud/tree.do',
24.         lines:true,
25.         onClick: function(node){
26.             //alert(node.id);
27.             //alert(node.text);
28.             $("#path").val(node.id);
29.         }
30.     });
31. });
32. function copyOrMove(){
33.     //console.info(${param.dir});
34.     //alert(${param.dir}+${param.ids});
35.     $.post('${pageContext.request.contextPath}/cloud/copy.do',
36.         {dir:$("#dir").val(),ids:$("#ids").val(),dst:$("#path").val(),fl
37.         ag:$("#flag").val()}),
38.         function(j) {
39.             if (j.success) {
40.                 parent.layer.close(index);
41.             }else{
42.                 alert(json.msg);
43.             }
44.         }, 'json');
45. </SCRIPT>
46. </head>
47.
48. <body>
49.     <ul id="tt"></ul>
50.     <hr><input type="hidden" id="dir" value="${param.dir}"/>
51.     <input type="hidden" id="ids" value="${param.ids}"/>
52.     <input type="hidden" id="flag" value="${param.flag}"/>
53.     <input type="hidden" id="path"/>
54.     <div align="right"><input type="button" class="button teal" value="确定
55.         " onclick="copyOrMove()" /></div>
56. </body>
57. </html>

```

