

SUPFile-PROJECT

***Class* Group Project**

***4-PJT-SUPFILE***

Delivery

Technical Documentation & Web API Documentation

Table of contents

[1 Group summary 3](#_Toc516252172)

[1.1 Group members 3](#_Toc516252173)

[2 Techinal Introduction 3](#_Toc516252174)

[2.1 java 3](#_Toc516252175)

[2.2 Server -Glassfish 3](#_Toc516252176)

[2.3 FrameWork –JSF 4](#_Toc516252177)

[2.4 UI - Primefaces 4](#_Toc516252178)

[2.5 DataBase –MySql 4](#_Toc516252179)

[2.6 Client-Android 4](#_Toc516252180)

[3 technical Document 5](#_Toc516252181)

[3.1 Web 5](#_Toc516252182)

[3.1.1 Project strcture 5](#_Toc516252183)

[3.1.2 Project packages 5](#_Toc516252184)

[3.1.3 User Interface 6](#_Toc516252185)

[3.2 Android 7](#_Toc516252186)

[3.2.1 Packages explain 7](#_Toc516252187)

[3.2.2 Function and Interface 9](#_Toc516252188)

[4 Database structure 12](#_Toc516252189)

[5 Web API documentation 13](#_Toc516252190)

[5.1.1 User API 14](#_Toc516252191)

[5.1.2 File API 14](#_Toc516252192)

[6 Storage and backup solution 16](#_Toc516252193)

# Group summary

## Group members

|  |  |  |
| --- | --- | --- |
| ID | Last Name | First Name |
| 210725 | LiangYi | Sun |
| 210733 | HongYang | Liu |
| 210700 | Xue | Cui |
| 210703 | Siqi | Du |

# Techinal Introduction

## java

**Java** is a general-purpose computer-programming language that is concurrent, class-based, object-oriented,and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use,particularly for client-server web applications, with a reported 9 million developers

## Server -Glassfish

GlassFish is an open-source application server project started by Sun Microsystems for the Java EE platform and now sponsored by Oracle Corporation. The supported version is called Oracle GlassFish Server. GlassFish is free software, dual-licensed under two free software licences: the Common Development and Distribution License (CDDL) and the GNU General Public License (GPL) with the classpath exception.

## FrameWork –JSF

JavaServer Faces is a Java specification for building component-based user interfaces for web applications and was formalized as a standard through the Java Community Process being part of the Java Platform, Enterprise Edition. It is also a MVC web framework that simplifies construction of user interfaces (UI) for server-based applications by using reusable UI components in a page.

JSF 2 uses Facelets as its default templating system. Other view technologies such as XUL or plain Java can also be employed. In contrast, JSF 1.x uses JavaServer Pages (JSP) as its default templating system.

JSF life cycle:

**Retrieve view :** Build the tree components of the page

**Apply application parameters :**Retrieve all the values in the request parameter

**Validation of the user entries :**Validate all the user entries.

If the validation succeed the lifecycle continue. Otherwise an exception is thrown.

**Update model :**Update all the models if the validation and the conversion succeed.

Convert thanks to converter the String parameter to Object values. If conversion fail an exception is thrown.

**Invocation of the application :**Invoke the requested method of the managed bean. Can eventually choose the next page.

**Response rendering :**Create the page rendering.

## UI - Primefaces

PrimeFaces is a popular open source framework for JavaServer Faces featuring over 100 components, touch optimized mobilekit, client side validation, theme engine and more.

## DataBase –MySql

MySQL is an open-source relational database management system (RDBMS). MySQL is a central component of the LAMP open-source web application software stack . LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python".

## Client-Android

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets.We develop Android side as client to access server.

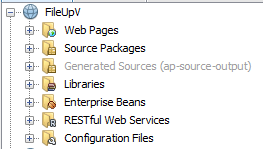
# technical Document

## Web

We use jsf framework for the backend and primefaces + html + css + jacascript for the frontend.

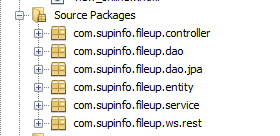
### Project strcture

This is the project code structure.



### Project packages

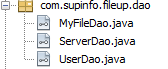
These are all the 6 package of the project



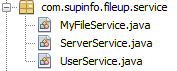
Entity package is POJO classes represent database tables with the same name.



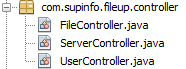
Dao is database interfaces , JpaDao is the implementation of Dao.



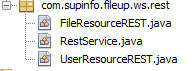
Service: inject the Dao interfaces to service sothat the controller can use the service remotely.



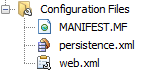
Controller: this package is used to handle the frontend request. Usercontroller userd to handle use requests like register, login logout; Filecontroller used to handle file operations like rename, upload, download, delete, copy, add folder; serverController used to connect the storage servers , when a user registered, he will have 30 GB of storage on the least used storage server.



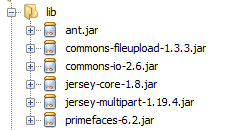
Ws.rest : this is the rest service package, userResourceREST used to retirve the user resourcces and FileResourceREST used to retrieve file resources.



This is the configuration files: persistence.xml used to connect the db and web.xml is the web application configuration file.



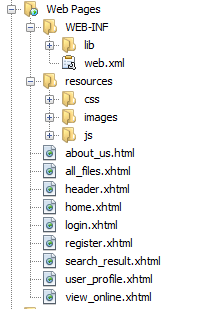
These are the external libraries we used. Commons-fileupload and commons-io are used to upload file, jersey is used for create web apis, primeface is used to ease the frontend work.



### User Interface

We primeraly use Primefaces to build out user interface.

To use primefaces, we add a xmlns for primefaces: “xmlns:p=<http://primefaces.org/ui> «and we rewrite some component of primefaces using css and javascript for improve user experience.

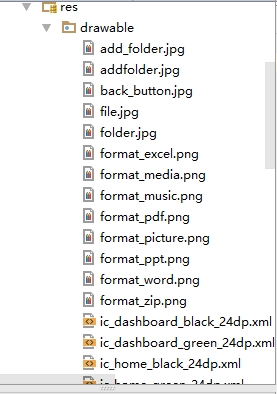


## Android

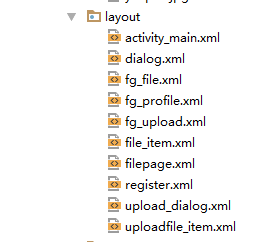
### 3.2.1 Packages explain

This part will explain the packages we created and what are they.

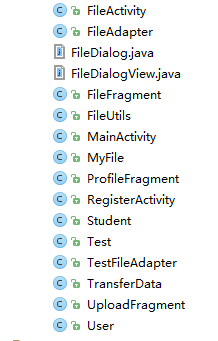
Drawable:



This part stores all the UI resources which will be used in the interface.



This part stores all the xml layout file.



In Java folder, there are all java class we have. They are used for layout events and implement logical functions.

### Function and Interface

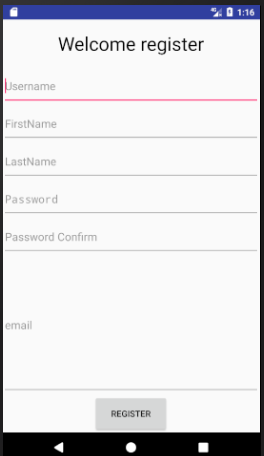
Our application use web api to access the server resources. And it achieve following functions:

(1)User can login logout register

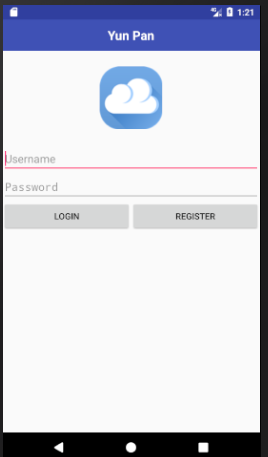
(2) Some operations about the files: such as add the folder,display files , rename filename download files.

These are our interaces.

Register interface:



Login interface:

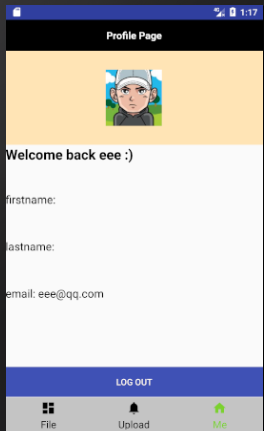


File display page:



In this page, user can see all their files and add folder or rename filename.

User profile page:



In this page, users can see all their personal information.

# Database structure

We have 3 tables in total : server, user and myfile



Table1 : **server**



Represent the storage server :

Id : generated in increase pattern.

Serverpath : used to store the path of the storage server

Servername : name of the storage server

Size : each time a user registered and storage his info on this server, the size increase by 1.

Table2 : **user**

Id : generated in increase pattern.

Email : email of the user , can not be the same as other user.

isLoggedin : 1 if the user is logged in, to prevent login in multi place.

Username : username. Can not be null.

Firstname : user firstname.

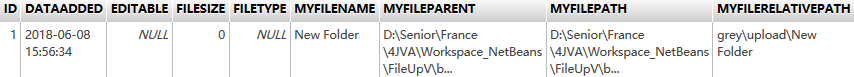
Lastname : user lastname

Password : password of the user

serverId : the id of the storage server that the user info is stored.

Usedspace : space that user has used. If upto 30GB, the user can not upload files anymore

Table3 : **myfile**





Id : generated in increase pattern

Myfilename : name

myfileParent : paretn full path of the file or folder

myfilePath : file Full Path

myFileRelativePath : relative path of file

type : is file or folder

dateadded : date that the file is uploaded

userId : the user that the file belongs

fileSize : size of the file

# Web API documentation

Welcome to the SupFile Web API! You can use the API to access SupFile resources, which can get information on users and all your files. All the web api are RESTful and were designed with Java EE Technologies (Servlet, JAX-RS, JPA). The exchanges of this API is in JSON. Users must log in with this API to use our file service.

All the web services can be accessed in the /rest path

resources related to user service can be accessed through /rest/auth

resources related to files and folders can be accessed through /rest/files

Error Code

| Code | Name | Description |
| --- | --- | --- |
| 400 | Bad Request | We could not process that action |
| 404 | Not Found | The requested resource could not be found |
| 500 | Internal Server Error | We had a problem with our server. |
| 503 | Service Unavailable | We are temporarily offline for maintenance. |

**API**

### User API

**User services: register, login, logout**

1. **Register : POST /rest/auth/register**

This call will return a user that has been registered successfully, if the registration is not successful, this call will return null. All parameters should be inside a form.

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User to register, must not be null. |
| password | Not null |
| passwordConfirm | Not null and the same as password |
| firsrname | Can be null |
| Lastname | Can be null |
| Email | Not null, must restricted to email pattern |

1. **Login : POST /rest/auth/login**

This call will return the user that is logged in, if login unsuccessful, return null. One user can not login twice in the same time. All parameters should be inside a form.

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User to login, must not be null, must be a registered username |
| password | Not null |

1. **Logout : GET /rest/auth/logout/{username}**

This call will return true if the user is logout, user must be currently login to logout. Parameter should in the url path.

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User to logout. |

### File API

**File services: getFileByPath, getAllFiles, upload, download, addFolder, rename**

1. **getFileByPath : GET rest/files/{username}/getFile**

this call will return a myfile instance with the file path, so that file id can be retrived. The file path can be seen in the website when user is logged in. if the path is not exist, this call will return null. Parameter username should in the url path and parameter filePath should in query of the request.

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User has logged in. |
| filePath | Path of the file. |

Example : <http://127.0.0.1:8080/FileUpV/rest/files/grey/getFile?filePath=\upload>

1. **getAllFiles : GET /rest/files/{username} /getFiles**

this call will return a list of myfile instances in json. If the folder of the files not exists, this call will return null.

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User has logged in. |
| folderId | Id of the folder to list all files, can be retrieved first by a getFileByPath request. |

Example: <http://127.0.0.1:8080/FileUpV/rest/files/grey/getFiles?folderId=2>

1. **Upload File : POST /rest/files/{username} /uploadTo/{folderId}**

This call will upload a file to the storage server in the directory with folder id. If success, this call will return a MyFile instance. If not , return null. Content type of this request should be multipart/form-data

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User has logged in |
| Filename | Should be a header |
| file | File to upload |
| folderId | The id of the folder this file should be uploaded to, can be retrieved first by a getFileByPath request. |

Example: <http://127.0.0.1:8080/FileUpV/rest/files/grey/uploadTo/2>

1. **Download : GET /rest/files/{username} /download/ {fileId}**

This call will return a byte[] that stores the file to download .

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User has logged in |
| fileId | The id of the file to download, can be retrieved first by a getFileByPath request. |

Example : <http://127.0.0.1:8080/FileUpV/rest/files/grey/download/666>

1. **addFolder : POST /rest/files/{username}/addFolder**

this call return the folder instance of the folder just added. Param username should in the url path and params currentFolderId and addFolderName should in a form.

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User has logged in, should in the url path |
| currentFolderId | The id of the parent folder of the folder to add, can be retrieved first by a getFileByPath request. |
| addFolderName | The folder to add. |

Example: [http://127.0.0.1:8080/FileUpV /rest/files/grey/addFolder](http://127.0.0.1:8080/FileUpV%20/rest/files/grey/addFolder/)

1. **Rename : POST /rest/files/{username}/rename**

This call will return the file or folder instance that has been renamed. If the renamed instance is exists, this call will return null.

|  |  |
| --- | --- |
| **Parameters** | |
| **Name** | **Description** |
| username | User has logged in, should in the url path |
| fileId | The id of the file or folder to rename, can be retrieved first by a getFileByPath request. |
| renameFileName | The file or folder to rename. |

Example: <http://127.0.0.1:8080/FileUpV/rest/files/grey/rename>

# Storage and backup solution

Our web client is high available and can survive losing nodes

1. For the storage , we used 3 storage servers using linux system , configure samba on each of them, then we can access them on our web server. When a user registered, he wil assign 30GB of storage on the least used storage server.
2. For the High availability, we use DRBD and ISCSI for the storage, and Glassfish Cluster for HA of glassfish.

* In case of disks failure and any other unexpected situation, we store all user files and data in there storage servers. We use DRBD and ISCSI to back up those data into 2 nodes, one primary, and one secondary.
* Clustering and HA support includes the creation of multiple clusters per domain and multiple instances per cluster with up to 100 instances per domain using the web-based Admin Console or the command line interface.

The deploy steps can be accessed in our Architecture storage and HA deployment manual.