Design Document

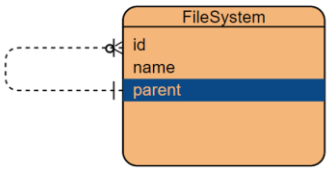
Abstract

This document describes the design choices I have made, along with the assumptions and why. I will show the ERD. What are the other ways I could have done this and limitation of the current implementation and how to improve it.

Technology Stack:

I have used java with spring boot/MVC for the task1. UI part I have used jsp along with bootstrap. I have taken H2 database as the RDBMS of choice as it is easy to setup and in-memory.

Entity relationship diagram:



[ERD diagram]

In the table file_system each directory stores the parent reference as foreign key.

Research

I have explored few ways and implemented the above design.

I) Using parent id

This is what I have implemented. Advantage of this design is moving one node from one to another is simple and easy. Just change the parent of that node, everything else will be same as before. Limitation of this design is while back track to get the full path I have to fetch iteratively. A possible solution is to use cache so that for a parented if already full path is extracted get it from cache.

II) Storing full path

This method has a major drawback when on node needs update like rename/ move from one folder to another all child need to be updated. At the same time benefit of this design is if there is no update or very few updates, this design can be used speed up the search ang get the full path of the object.

III) Storing the boundary values

This a unique approach, where each folder will be stored with two index left-start and right-end boundaries. In this way it will be easy to get all the children of a folder. But mobbing object from one folder to other or adding new object will be costly operation as a lot up update is required.

Reading the text file and storing into database

Here I am reading a tab delimited text file and automatically stored in database. Where the indent decides the parent chile relationship. The file is read and converted to a tree data structure, as I thought a tree will best represent a file system. Then I am passing the tree (root node) to DAO layer and storing into database.

What can be done to improve? (More research)

If I get more time then, I would like to change this logic. Instead of the tree data structure, I would like to read line by line using buffer reader and send the processed lines (name, parent) to database for storing the line in proper manner. In this way I can reduce the memory requirement of the program.

Another way is to, use multithreading. Read the parent node, then save all child nodes at a time using multithreading apis like ExecutorService of java with reference to the parent node. This can be recursively, and parallel insertion can be done for sibling nodes.

I the file size is really large say like 10 GB for example. Then I will try to decouple the reading and processing the file and storing into nosql database (es/mongo). Some messaging queue can be used like rabbitmq to create a push-pull system. A little more research can be done in this are if that is falls inside the scope of the project.

Error handling

Various error handling has been used.

I) Custom exception

A custom exception is written, it can be used as business exception or converting a checked exception to and unchecked exception if needed.

II) Model validation

Spring model validation framework is used to validate the input form for checking the input name should not be empty while search. Ui message is incorporated for error message.

III) Global exception handler

A global exception handler is written to catch any exception and redirect to error.jsp page.

OO design

I have done using OO design. All classes like controller, service, dao, models, utils are logically separated and encapsulated by class. I have used design patterns like composite pattern, façade pattern while design the model and delegating the flow from controller to dao layer.

Unit testing

I have used spring test to write Junits. Here I have written 2-unit tests, if passed then only build will happen. One is to check the txt file parsing logic and other is to check the logic of search mechanism.