文件系统

2025 年 5 月 30 日

一、主要任务：

1. 重命名文件/tea/win/makefile.vc 为/tea/win/mymakefile.vc
2. 将/tea/win 移动到/tea/doc
3. 将/tea/doc/rules.vc 重命名为 /tea/doc/myrules.vc
4. 获取至少有两个文件的目录
5. 删除/tea/tclconfig

二、实现方法：

1. 创建表 dir 表示路径
2. 创建表 dir_dir 表示目录层级关系
3. 创建表 file 表示目录中的文件
4. 使用 mybatis 框架完成各个操作

三、成果展示：

1. 建表

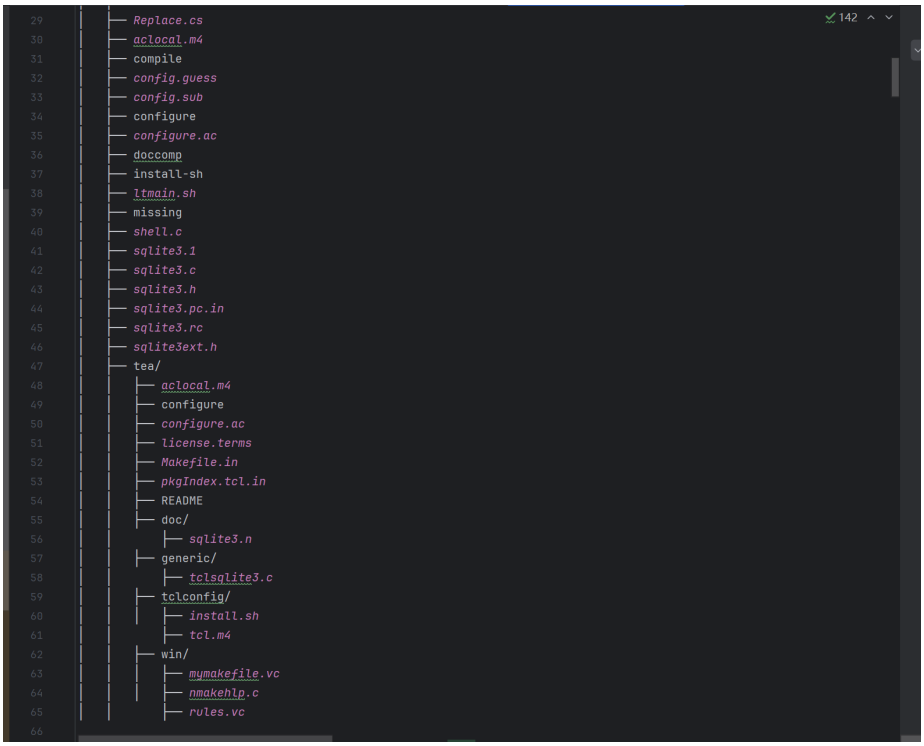
```sql
-- 创建目录表
CREATE TABLE dir (
    dir_id INTEGER PRIMARY KEY,
    dirname TEXT NOT NULL
);

-- 创建目录关系表
CREATE TABLE dir_dir (
    parent_id INTEGER,
    child_id INTEGER,
    PRIMARY KEY (parent_id, child_id),
    FOREIGN KEY (parent_id) REFERENCES dir(dir_id),
    FOREIGN KEY (child_id) REFERENCES dir(dir_id)
);

-- 创建文件表
CREATE TABLE file (
    file_id INTEGER PRIMARY KEY,
    filename TEXT NOT NULL,
    dir_id INTEGER,
    FOREIGN KEY (dir_id) REFERENCES dir(dir_id)
);
```

以上是三个表的建表语句，dir 表：dir 的 id 和目录名，dir_dir: 父目录和子目录的 id，file 表：fileid，文件名，文件所属目录

2. 重命名 mapper

```java
@Update("UPDATE file SET filename = #{newName} " +  2 个用法
        "WHERE filename = #{oldName} AND dir_id = " +
        "(SELECT dir_id FROM dir WHERE dirname = #{dirName})")
void renameFile(@Param("oldName") String oldName,
                @Param("newName") String newName,
                @Param("dirName") String dirName);
```

函数需要传入新文件名，旧文件名，文件所属目录名，通过 update 语句将文件名更新好

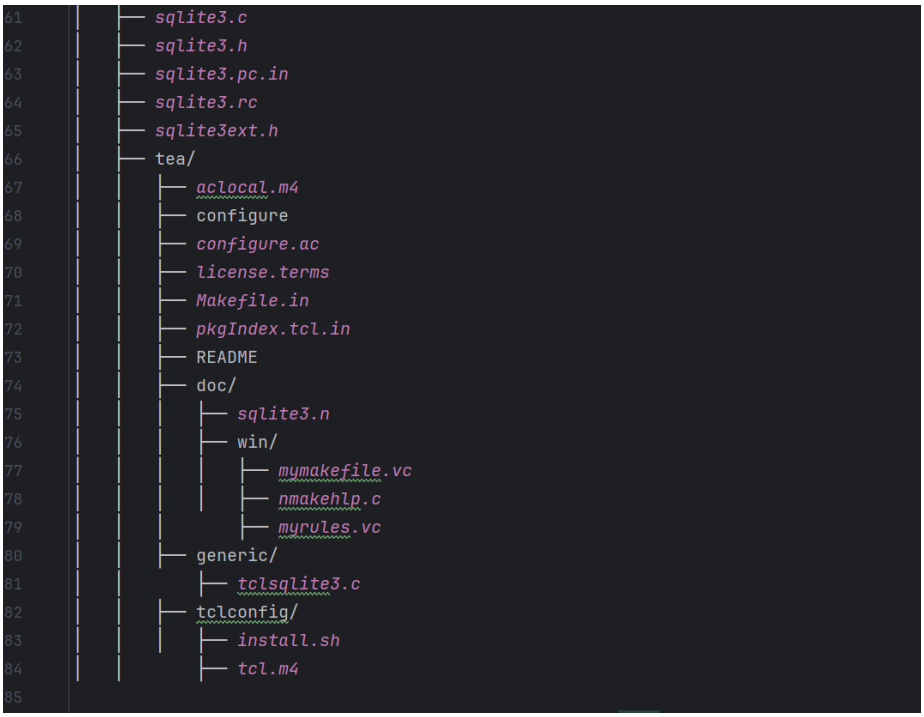

142
Replace.cs
aclocal.m4
compile
config.guess
config.sub
configure
configure.ac
doccomp
install-sh
ltmain.sh
missing
shell.c
sqlite3.1
sqlite3.c
sqlite3.h
sqlite3.pc.in
sqlite3.rc
sqlite3ext.h
tea/
aclocal.m4
configure
configure.ac
license.terms
Makefile.in
pkgIndex.tcl.in
README
doc/
sqlite3.n
generic/
tclsqlite3.c
tclconfig/
install.sh
tcl.m4
win/
mymakefile.vc
nmakehlp.c
rules.vc


sqlite3.c
sqlite3.h
sqlite3.pc.in
sqlite3.rc
sqlite3ext.h
tea/
aclocal.m4
configure
configure.ac
license.terms
Makefile.in
pkgIndex.tcl.in
README
doc/
sqlite3.n
win/
mymakefile.vc
nmakehlp.c
myrules.vc
generic/
tclsqlite3.c
tclconfig/
install.sh
tcl.m4

可见 makefile 和 rule 都已经重命名完成了

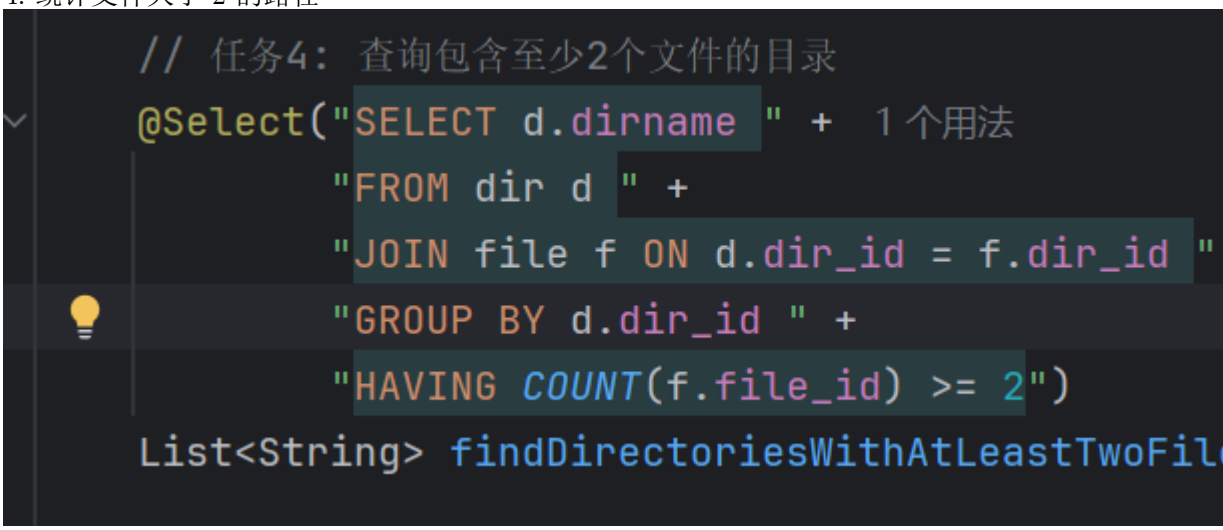
3. 修改文件路径

先从 dir_dir 中删除原本的父子目录关系，再插入新的关系

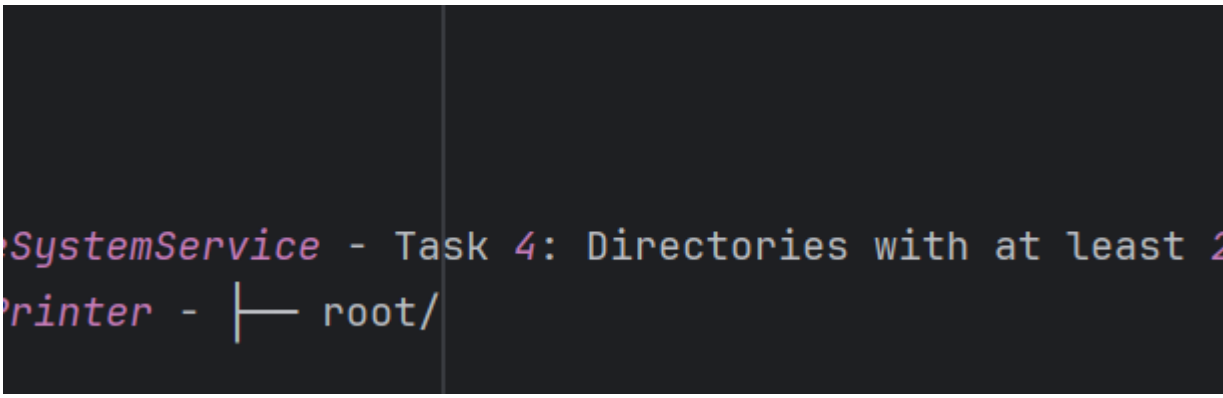


可见目录已经移动完毕

4. 统计文件大于 2 的路径



用 dir_id 给文件分组，查询出分组的个数大于 2 的路径名

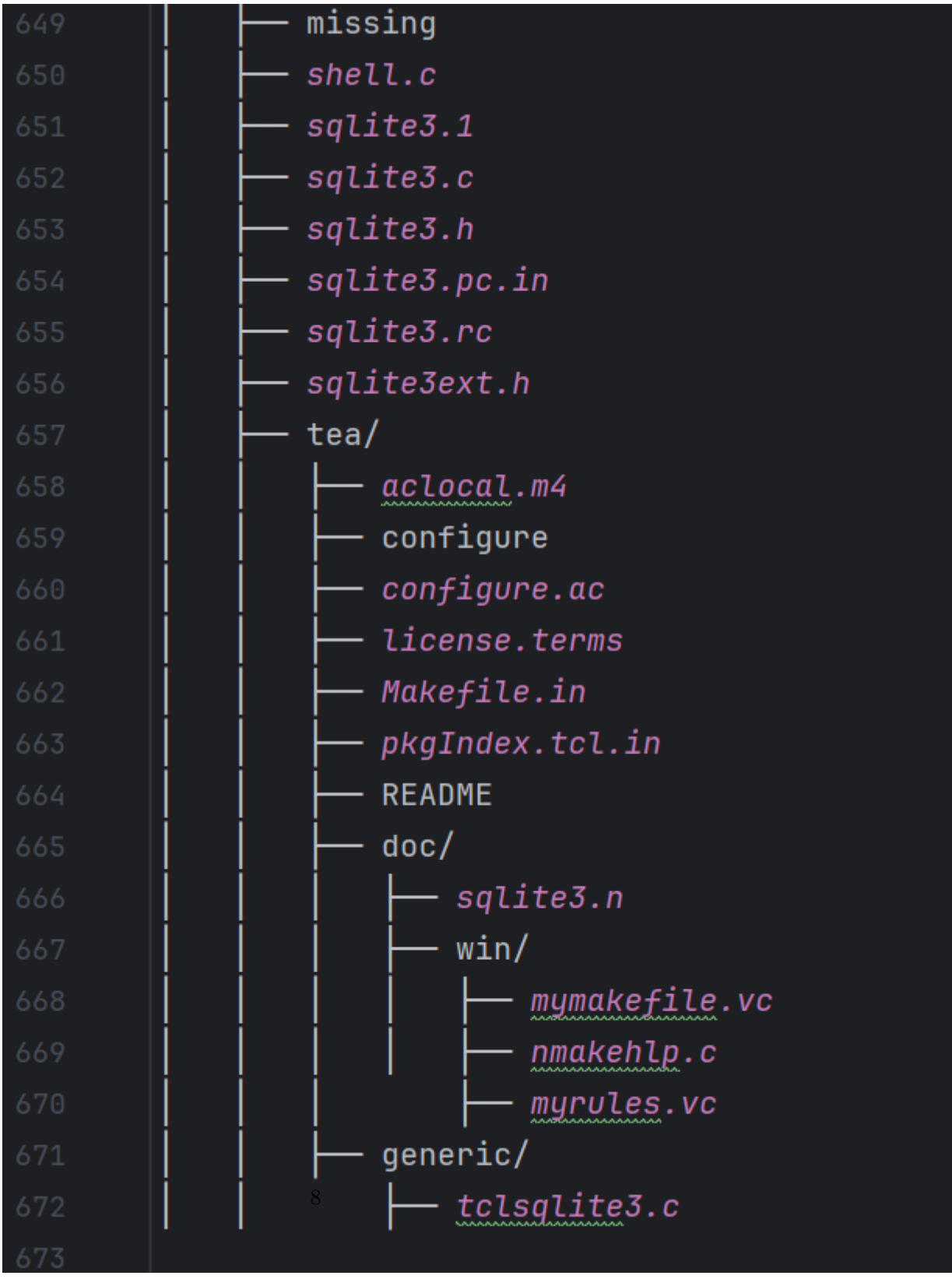


可见已经统计出来了

5. 删除路径

```
@Delete("DELETE FROM file WHERE dir_id = " +  1 个用法
        "(SELECT dir_id FROM dir WHERE dirname = #{dirName})")
void deleteFilesInDirectory(@Param("dirName") String dirName);

@Delete("DELETE FROM dir_dir WHERE child_id = " +  1 个用法
        "(SELECT dir_id FROM dir WHERE dirname = #{dirName})")
void deleteDirectoryRelations(@Param("dirName") String dirName);

@Delete("DELETE FROM dir WHERE dirname = #{dirName}")  1 个用法
void deleteDirectory(@Param("dirName") String dirName);
```

在三个表中分别把有关该路径的内容全部删除


missing
shell.c
sqlite3.1
sqlite3.c
sqlite3.h
sqlite3.pc.in
sqlite3.rc
sqlite3ext.h
tea/
aclocal.m4
configure
configure.ac
license.terms
Makefile.in
pkgIndex.tcl.in
README
doc/
sqlite3.n
win/
mymakefile.vc
nmakehlp.c
myrules.vc
generic/
tclsqlite3.c

可见已经成功删除

四、结论：

1. 可以用数据库模拟文件系统，要学会自己设计表，使表尽量满足范式

2. 使用 springboot 集成的 mybatis 框架操作数据库是十分简单便捷的

五. 源代码：

```
package homework.databasehomework.mapper;

import homework.databasehomework.pojo.Directory;
import homework.databasehomework.pojo.File;
import org.apache.ibatis.annotations.*;

import java.util.List;

@Mapper
public interface FileSystemMapper {

    @Update("UPDATE file SET filename = #{newName} " +
            "WHERE filename = #{oldName} AND dir_id = " +
            "(SELECT dir_id FROM dir WHERE dirname = #{dirName})")
    void renameFile(@Param("oldName") String oldName,
                    @Param("newName") String newName,
                    @Param("dirName") String dirName);

    // 任务2: 移动目录
    @Delete("DELETE FROM dir_dir " +
            "WHERE child_id = (SELECT dir_id FROM dir WHERE dirname = #{dirName})")
    void removeDirectoryRelation(@Param("dirName") String dirName);

    @Insert("INSERT INTO dir_dir (parent_id, child_id) " +
            "VALUES ((SELECT dir_id FROM dir WHERE dirname = #{parentDir}), " +
            "(SELECT dir_id FROM dir WHERE dirname = #{childDir}))")
    void addDirectoryRelation(@Param("parentDir") String
```

```
    parentDir,
                                @Param("childDir") String
                                    childDir);

    // 任务3: 重命名文件（同任务1，可直接复用）

    // 任务4: 查询包含至少2个文件的目录
    @Select("SELECT d.dirname " +
            "FROM dir d " +
            "JOIN file f ON d.dir_id = f.dir_id " +
            "GROUP BY d.dir_id " +
            "HAVING COUNT(f.file_id) >= 2")
    List<String> findDirectoriesWithAtLeastTwoFiles();

    // 任务5: 删除目录及其内容
    @Delete("DELETE FROM file WHERE dir_id = " +
            "(SELECT dir_id FROM dir WHERE dirname = #{dirName
                })")
    void deleteFilesInDirectory(@Param("dirName") String
        dirName);

    @Delete("DELETE FROM dir_dir WHERE child_id = " +
            "(SELECT dir_id FROM dir WHERE dirname = #{dirName
                })")
    void deleteDirectoryRelations(@Param("dirName") String
        dirName);

    @Delete("DELETE FROM dir WHERE dirname = #{dirName}")
    void deleteDirectory(@Param("dirName") String dirName);
    @Select("SELECT * FROM dir WHERE dirname = #{dirname}")
    Directory getDirectoryByName(String dirname);

    // 根据父目录ID获取子目录列表
    @Select("SELECT d.* FROM dir d JOIN dir_dir dd ON d.dir_id
        = dd.child_id " +
            "WHERE dd.parent_id = #{parentId}")
    List<Directory> getSubDirectoriesByParentId(Integer
        parentId);

```

```
    // 根据目录ID获取文件列表
    @Select("SELECT * FROM file WHERE dir_id = #{dirId}")
    List<File> getFilesByDirectoryId(Integer dirId);

}
```

```
package homework.databasehomework.service.impl;

import homework.databasehomework.mapper.FileSystemMapper;
import homework.databasehomework.print.FileSystemPrinter;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional
    ;

import java.util.List;

@Service
@Slf4j
public class FileSystemService {
    @Autowired
    private FileSystemMapper fileSystemMapper;

    // 任务1: 重命名文件 /tea/win/makefile.vc → /tea/win/
        mymakefile.vc
    public void renameMakefile() {
        fileSystemMapper.renameFile("makefile.vc", "mymakefile.
            vc", "win");
    }

    // 任务2: 移动 /tea/win 到 /tea/doc
    public void moveWinToDoc() {
        // 1. 删除原有的目录关系
        fileSystemMapper.removeDirectoryRelation("win");

        // 2. 添加新的目录关系
        fileSystemMapper.addDirectoryRelation("doc", "win");
    }
```

```
31
32      // 任务3: 重命名文件 /tea/doc/rules.vc → /tea/doc/myrules.
            vc
33      public void renameRules() {
34          fileSystemMapper.renameFile("rules.vc", "myrules.vc", "
                win");
35      }
36
37      // 任务4: 查询包含至少2个文件的目录
38      public List<String> getDirectoriesWithAtLeastTwoFiles() {
39          return fileSystemMapper.
                findDirectoriesWithAtLeastTwoFiles();
40      }
41
42      // 任务5: 删除 /tea/tclconfig
43      @Transactional
44      public void deleteTclconfig() {
45          // 1. 删除目录下的文件
46          fileSystemMapper.deleteFilesInDirectory("tclconfig");
47
48          // 2. 删除目录关系
49          fileSystemMapper.deleteDirectoryRelations("tclconfig");
50
51          // 3. 删除目录本身
52          fileSystemMapper.deleteDirectory("tclconfig");
53      }
54
55      // 执行所有任务
56      @Transactional
57      public void performAllTasks() {
58          // 任务1
59          renameMakefile();
60          FileSystemPrinter fileSystemPrinter=new
                FileSystemPrinter(fileSystemMapper);
61
62          log.info("Task␣1:␣Renamed␣makefile.vc␣to␣mymakefile.vc"
                );
63          fileSystemPrinter.printFileSystem();
64
```

```
65          // 任务2
66          moveWinToDoc();
67          log.info("Task 2: Moved /tea/win to /tea/doc");
68          fileSystemPrinter.printFileSystem();
69
70          // 任务3
71          renameRules();
72          log.info("Task 3: Renamed rules.vc to myrules.vc");
73          fileSystemPrinter.printFileSystem();
74
75          // 任务4
76          List<String> directories =
                getDirectoriesWithAtLeastTwoFiles();
77          log.info("Task 4: Directories with at least 2 files: "
                + directories);
78          fileSystemPrinter.printFileSystem();
79          // 任务5
80          deleteTclconfig();
81          log.info("Task 5: Deleted /tea/tclconfig");
82          fileSystemPrinter.printFileSystem();
83      }
84  }
```

递归调用，打印文件目录

```
1   package homework.databasehomework.print;
2
3   import homework.databasehomework.mapper.FileSystemMapper;
4   import homework.databasehomework.pojo.Directory;
5   import homework.databasehomework.pojo.File;
6   import lombok.extern.slf4j.Slf4j;
7
8   import java.util.*;
9   @Slf4j
10  public class FileSystemPrinter {
11      private final FileSystemMapper fileSystemMapper;
12      private StringBuilder stringBuilder = new StringBuilder();
13      public FileSystemPrinter(FileSystemMapper fileSystemMapper)
            {
14          this.fileSystemMapper = fileSystemMapper;
```

```
15      }
16
17      // 打印整个文件系统结构
18      public void printFileSystem() {
19          // 获取根目录（假设根目录名为"root"）
20          Directory rootDir = fileSystemMapper.getDirectoryByName
                ("root");
21
22          if (rootDir == null) {
23              System.out.println("Root␣directory␣not␣found!");
24              return;
25          }
26
27          // 递归构建文件系统树
28          TreeNode root = buildTree(rootDir, 0);
29
30          // 打印树形结构
31          printTree(root, "");
32          log.info(stringBuilder.toString());
33      }
34
35      // 递归构建文件系统树
36      private TreeNode buildTree(Directory directory, int level)
            {
37          TreeNode node = new TreeNode(directory.getDirname(),
                level);
38
39          // 获取当前目录下的文件
40          List<File> files = fileSystemMapper.
                getFilesByDirectoryId(directory.getDirId());
41          for (File file : files) {
42              node.addFile(file.getFilename());
43          }
44
45          // 获取子目录
46          List<Directory> subDirs = fileSystemMapper.
                getSubDirectoriesByParentId(directory.getDirId());
47          for (Directory subDir : subDirs) {
48              TreeNode childNode = buildTree(subDir, level + 1);
```

```
49              node.addChild(childNode);
50          }
51
52          return node;
53      }
54
55      // 递归打印树形结构
56      private void printTree(TreeNode node, String prefix) {
57          // 打印当前目录
58          stringBuilder.append(prefix + "    ␣" + node.getName() +
                "/"+"\n");
59
60          // 打印文件
61          List<String> files = node.getFiles();
62          for (int i = 0; i < files.size(); i++) {
63              String filePrefix = prefix + (node.hasChildren() ||
                    i < files.size() - 1 ? " ␣␣␣" : "␣␣␣␣");
64              stringBuilder.append(filePrefix + "    ␣" + files.
                    get(i)+"\n");
65          }
66
67          // 递归打印子目录
68          List<TreeNode> children = node.getChildren();
69          for (int i = 0; i < children.size(); i++) {
70              String childPrefix = prefix + (i < children.size()
                    - 1 || !files.isEmpty() ? " ␣␣␣" : "␣␣␣␣");
71              printTree(children.get(i), childPrefix);
72          }
73      }
74
75      // 树节点内部类
76      private static class TreeNode {
77          private final String name;
78          private final int level;
79          private final List<String> files = new ArrayList<>();
80          private final List<TreeNode> children = new ArrayList
                <>();
81
82          public TreeNode(String name, int level) {
```

```
83              this.name = name;
84              this.level = level;
85          }
86
87          public void addFile(String fileName) {
88              files.add(fileName);
89          }
90
91          public void addChild(TreeNode child) {
92              children.add(child);
93          }
94
95          public String getName() {
96              return name;
97          }
98
99          public List<String> getFiles() {
100             return files;
101         }
102
103         public List<TreeNode> getChildren() {
104             return children;
105         }
106
107         public boolean hasChildren() {
108             return !children.isEmpty();
109         }
110     }
111 }
```