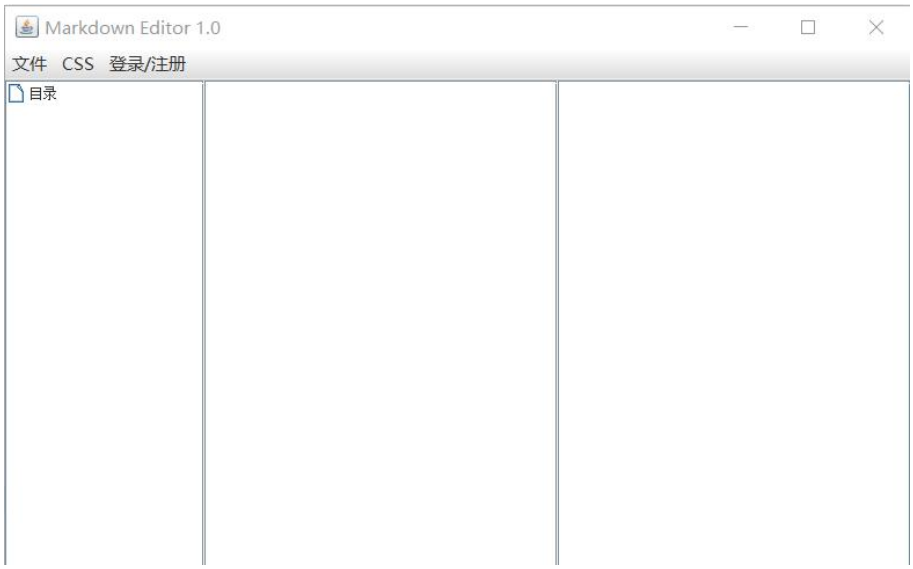
cscw markdown

一、 功能简介

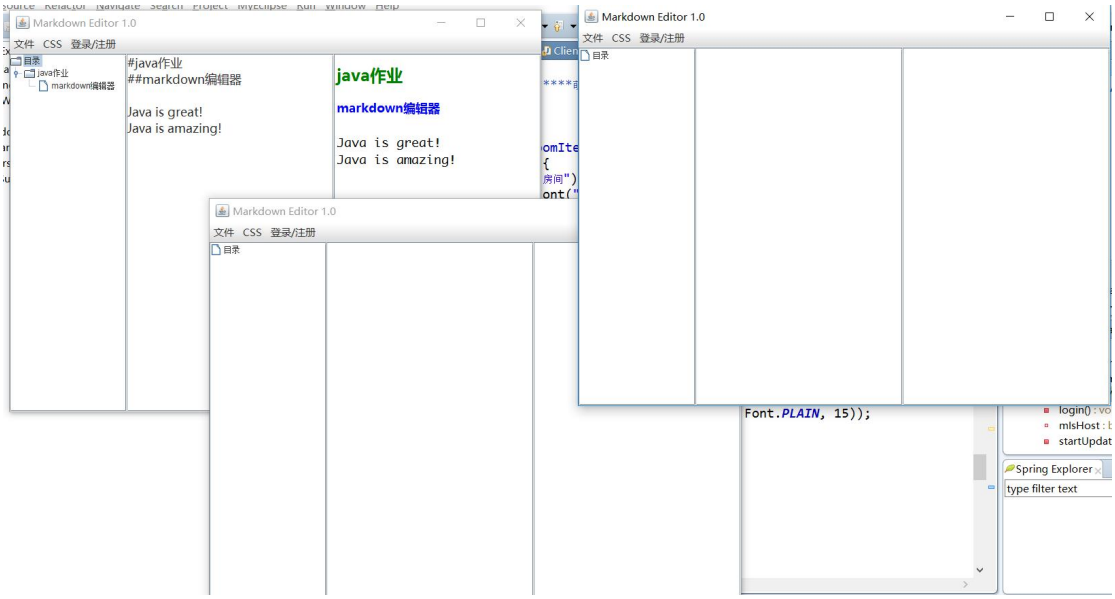


1. **离线功能**。支持本地文件导入、保存，导出 HTML、Word 文档，添加 CSS 样式、导入外部 CSS 文件，大文件处理。如果用户一直输入，没有停下超过 1 秒，右边的 HTML 视图就不会更新，这样可以减少无用的计算。左侧的导航栏用树形结构表示，点击标题可以跳转到对应行。

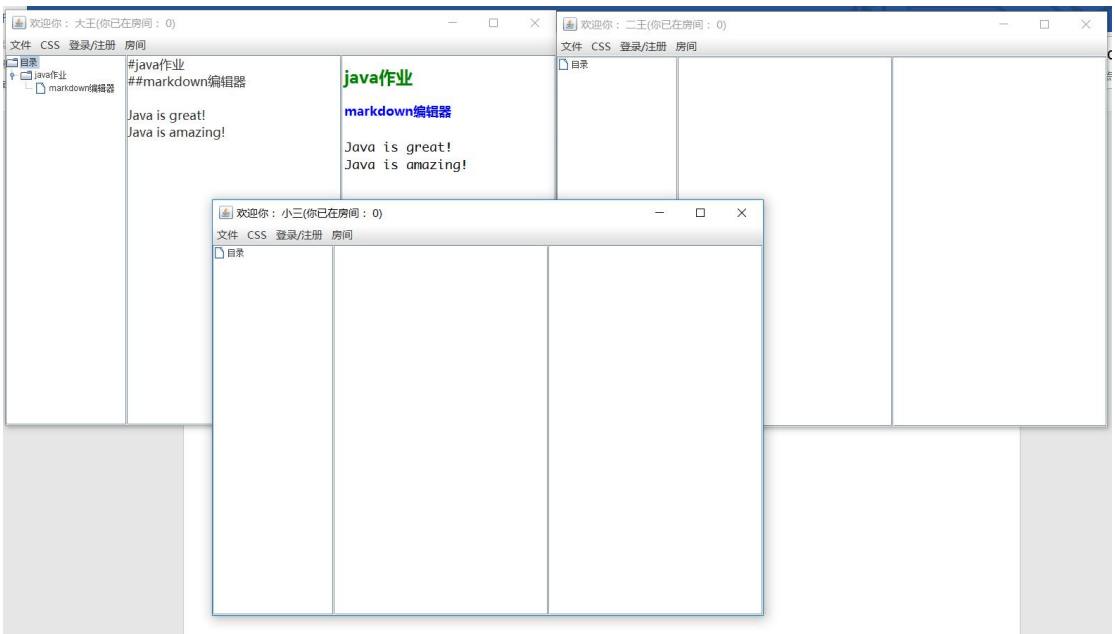
2. **在线编辑**。模拟多人在线编辑的效果，可以用 IDE 运行多个当前程序，除了第一个正常，之后的会提示无法创建本地服务器等等好几个错误，这是正常的，因为服务

器已经由第一个进程创建，之后的就不能再次创建。忽略即可。

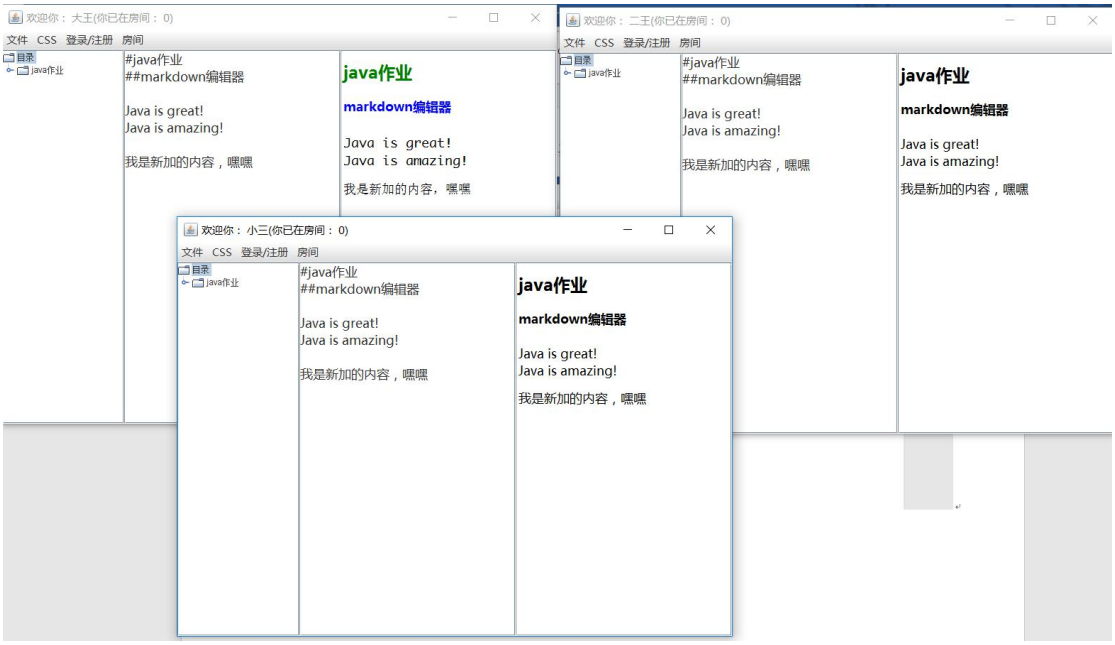
如图，运行 3 个进程。



每一个都注册。然后让一个人创建一个房间，另外两个加入进去。



他们三个现在在 0 号房间内，并且用户“大王”是房间主。目前设计的模式是只有房间主人可以进行文本的修改，其他成员无权修改。



如图。

二、 技术实现

Markdown 解析：

1. **Markdown 导出：**主要用了开源的 markdown4j 包。
2. **实时显示：**为了减少不必要的计算，这里用线程阻塞的方式，结合 SwingUtilities.InvokeLater()实现高效地更新。每次，想要更新的线程先 sleep 1000 毫秒，如果之后又有想要更新的线程，就将上一个阻塞，这样的话上一个就直接返回，放弃更新的操作。这样的效果就是，只要持续输入或删除内容，对文本内容做出改变的间隔不超过 1 秒钟，就一直不会更新，当用户停止 1 秒后，才会执行更新操作。大大减少了无用的计算。

```
private void updateUIEfficiently() {
    new Thread(() -> {
        Thread last = lastThread;
        lastThread = Thread.currentThread();
        //if(isUpdating) return;
        //isUpdating = true;

        try {
            //阻塞上一个更新的线程
            if(last != null) {
                last.interrupt();
            }
            Thread.sleep(1000);
        } catch(InterruptedException exc) {
            return;
        }

        if(Thread.currentThread().isInterrupted()) return;
        SwingUtilities.invokeLater(() -> {update();});

        if(mIsHost) {
            String updation = mTextArea.getText();
            try {
                mClient.disposeRequest(RequestType.UPLOAD_UPDATION, updation);
            } catch (Exception e) {
                e.printStackTrace();
                Utility.error("与服务器端连接出现错误！");
            }
        }

        //isUpdating = false;
    }).start();
```

3. 导航栏：用了 JTree，生成关于目录的树状结构，并对每个节点添加点击事件。

Socket 编程：

1. 创建一个本地服务器，开放 2 个端口，8080 和 8081。总共创建了 2 个类 **Server** 和 **Client**。Server 处理客户端发来的所有请求，并返回回复；Client 用于在客户端帮助用户发送所有请求。工程建立在 **C/S** 架构上。

2. 所有客户端一旦试图进行登录或注册，就会创建一个 Client，并建立相应的端口进行通信。服务器也会新开一个线程对这个客户端进行服务。客户端进行的一切操作都是在向服务器发送请求，然后得到服务器的回复。当客户端退出登录或是退出程序，就断开连接的端口，释放资源，防止内存浪费。

3. 一旦一个用户创建或是加入了一个房间，那么服务器就必须通过另一个端口——8081，来进行房间内内容的更新等操作。所以加入房间后，客户端会试图连接服务器的 8081 端口。对应的服务器上 8081 端口专门有一个线程来进行处理，依然是每有一个客户端连接过来就新开一个线程进行服务。这个端口做的事就是专门把房间主人发过来的文本内容发送给房间其他所有成员让他们进行同步。所以，每个客户端也必须开一个线程，不断尝试从对应 8081 端口的 socket 读取内容，然后更新。

4. 关于请求的类型定义在 RequestType.java 中：

```
public void disposeRequest(RequestType type, String ... args) throws Exception {
    //传输请求
    switch (type) {
    case LOGIN:
        disposeLogin(type, args[0], args[1]);
        break;

        //注册和登录在客户端的处理不区分
    case REGISTER:
        disposeLogin(type, args[0], args[1]);
        break;

    case CUT_CONNECT:
        disposeCut(type);
        break;

    case CREATE_ROOM:
        disposeCreateRoom(type);
        break;

    case JOIN_ROOM:
        disposeJoinRoom(type, args[0]);
        break;

    case UPLOAD_UPDATION:
        disposeUpdation(type, args[0]);
        break;

    default:
        break;
    }
}
```

一共定义了这么几种类型。

同样，服务器端也根据某些特征判断发来的请求是什么类型的，然后执行相应的处理方法。

目前设计的模式是只有房间主人可以对文本内容进行更改。由于时间有限，没有添加申请成为房间主人的功能，不过这个只不过是在 RequestType 中加入了一个新的请求。