1 团队成员

游婕-硕9073-3119305625

何静

肖温-硕9074-3119305689

吴多贤

2 代码的github地址

<https://github.com/Yyyyyyj/FinalHomework>

3 问题分析与任务拆分

由于需要前后端分离且各自部署，所以将前后端拆开，考虑到团队内同学熟悉前端代码编写的较少，将前端划分给三人共同开发分担任务，剩下的后端由比较熟悉后端的人开发，具体分工如下

游婕：后端开发部署

何静：

肖温：前端代码编写

吴多贤：

4 知识点列表

4.1 文件（数据库）并发操作中的一致性问题（游婕）

文件（数据库）并发操作中的一致性问题包括：脏读，不可重复读，幻读。

脏读：一个事务读取到了其他事务没有提交的数据。

不可重复读：一个事务内多次根据同一个查询条件查询出来的同一行记录的值不一样，原因是在多次搜索期间查询条件覆盖的数据被其他事务修改了。

幻读：一个事务内多次根据同个条件查出来的记录行数不一样。幻读的原因也是另外一个事务新增或者删除或者修改了第一个事务结果集里面的数据，不同在于不可重复读是同一个记录的数据内容被修改了，幻读是数据行记录变多了或者少了。

·**出处：**后端新增todo时的读写冲突问题。

后端在新增todo时不允许添加相同内容的待办任务，需要先从文件中读取原有数据并检查新增的todo任务内容是否在文件中已存在，满足条件时才将新增todo写入文件，否则返回提示任务已存在。

但在测试时发现能添加相同todo，原因是在第一次添加的数据还未写入完毕时就读出了文件数据来检查是否重复。

·**使用场景：**文件/数据库的并发操作

4.2 Java FileChannel文件通道（游婕）

FileChannel是java NIO库中提供的一个类，它可以通过可以通过文件通道读写文件。FileChannel的特点包括：

它是线程安全的，可以锁定文件的某一部分来限制其他线程访问。

它不能设置为非阻塞模式，总是运行在阻塞模式。

它将文件的一部分直接加载到内存中来提高效率。

它能在文件中的特定位置读取和写入。

它能强制将更新立即写入文件以进行存储，来避免数据丢失。

不可重复读：一个事务内多次根据同一个查询条件查询出来的同一行记录的值不一样，原因是在多次搜索期间查询条件覆盖的

·**使用场景：**利用其特性解决新增todo时的读写冲突问题

4.3 阻塞/非阻塞和同步/异步（游婕）

阻塞：A调用B，A被挂起直到B返回结果给A，才能继续执行

非阻塞：A调用B，A不会被挂起，A可以执行其他操作

同步：A调用B，B处理直到获得结果，才返回给A

异步：A调用B，无需等待结果，B以消息或回调的方式来通知A

两者的修饰对象不同，同步异步是一种消息通信机制，阻塞非阻塞是线程的一种状态。同步异步指的是被调用者结果返回时通知线程的一种机制，阻塞非阻塞指的是调用结果返回进程前的状态，是挂起还是继续处理其他任务。

·**出处：** FileChannel总是运行在阻塞模式

4.4 四种常见的POST提交数据方式以及对应的接收方法（游婕）

@RequestParam

Content-Type：application/x-www-form-urlencoded或form-data

该注解有两个属性:value，required；value用来指定要传入值的id名称，required用来指示参数是否必须绑定

@RequestBody

Content-Type：不是application/x-www-form-urlencoded编码的内容，如application/json，application/xml等

·**出处：**后端新增todo时用RequestParam无法接收到数据

·**使用场景：**根据前端传值的Content-Type用对应的注解

4.5 前后端交互测试时，后端项目如何提供外网访问（游婕）

方法一：将项目部署在有公网ip的服务器上

方法二：将java编写的后端springboot项目经过maven clean，maven install打包成jar包，在前端项目所在机器上运行jar包即可启动后端项目，从而避免外网访问问题

·**出处：**前后端交互测试时，外网无法访问后端项目

·**使用场景：**前后端交互

4.6.input点击删除所在父div:（**出处**：删除事件）

onDelBtnClick(){

var odiv = document.getElementById('div1');//每次添加返回一个div,虽然div的id相同，但此方法可实现仅删除当前div

var obt = document.getElementById("bt");

odiv.parentNode.removeChild(odiv)

}

4.7.fetch 请求：前后端交互（**出处**：交互事件）

fetch("url",{

method:"post",

headers:{

'Accept':'application/json,text/plain,\*/\*',

'Content-Type':'application/x-www-form-urlencoded'

},、/headers默认参数

body://传值

})

.then(res =>{

return res.json()

})

.then(data =>{//data 后端返回数据

console.log(data)

})

4.8.componentWillMount每一个组件render之前立即调用；componentDidMount render之后并不会立即调用，而是所有的子组件都render完之后才可以调用；componentWillMount可以在服务端被调用，也可以在浏览器端被调用；

4.9.当fetch中格式为

.then(function(res) {

return res.json();

})

.then(function(data){

console.log(data);

that.setState({//注意此处that

todoList:data

})

fetch前要加var that = this,用that调用，直接用this会报错。

4.10.修改input内容并保存：（**出处**：修改事件）

$(".data").bind("change",function(){

//input id=data

var data= $("#datatable\_id tbody").find(".data").val();

var url = "basic/data/changedata?id="+$(this).attr("name");//绑定id

$.post(url,{"data":data},function(rd){

if(rd.flag){

layer.msg("修改成功");

window.location.reload();

}else{

layer.alert(rd.msg);

}

})

});