**1 团队成员**

游婕-硕9073-3119305625

何静-硕9073-3119305648

肖温-硕9074-3119305689

吴多贤-硕9081-3119405033

**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：前端代码编写，负责删除和增加功能的编写

肖温：前端代码编写，负责修改和编辑功能的编写

吴多贤：e2e测试

**4 知识点列表**

注：1-6游婕总结，7-11肖温总结，12-20吴多贤总结，21-25何静总结

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 jar包内部的资源文件读取（游婕）

·**出处：**后端项目在IDE中能正常运行，但打包成jar包供前端测试时无法读取src/java/resources/static/task.json文件

·**原因：**RandomAccessFile和FileOutputStream无法读取jar包中的资源文件

·**解决方案：**只能通过ClassPathResource等方式读取，但没有找到能向jar包中的文件写入数据的较好方法，所以将在工程路径下读写文件改成在本地创建文件进行读写

4.7.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)

}

4.8.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)

})

4.9.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；

4.10.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。

4.11.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);

}

})

});

4.12.通过AJAX做前后端分离时，要创建XHR对象（XML Http Request）XHR对象是一个javascript对象，它可以在用户没有感觉的情况下，就像背后运行的一根小线程一般，悄悄的和服务器进行数据交互。AJAX就是通过它做到无刷新效果的。

4.13.在jsp中当一个变量被声明了但是没有赋值时就叫做undefined。例如：var x；

伪元素before和after的用法，在css中采用双冒号写法，在::before和::after下特有的content，用于在css渲染中向元素逻辑上的头部或尾部添加内容。

这些添加不会出现在DOM中，不会改变文档内容，不可复制，仅仅是在css渲染层加入。

4.14document.getElementById()是通过id获取节点是BOM的一种方法，即把html里面的各种数据当作对象进行操作，可以通过document.getElementById 获取某个元素对应的元素节点对象。

4.15jQuery 代码中 $(function(){})是由$()和function(){}组成，这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

4.16prop和attr的区别：**attr**也可以用来获取与设置元素的属性，区别在于，对于**自定义属性**和**选中属性**的处理。选中属性指的是 checked,selected 这2种属性  
（1）对于自定义属性 attr能够获取，prop不能获取  
（2） 对于选中属性  
attr **只能获取初始值**， 无论是否变化，prop 能够访问变化后的值，并且以**true|false**的布尔型返回。

4.17.原生BOM对象和JQuery对象的转化：JQuery对象可以通过get[0]或[0]转化为BOM对象 ；BOM对象通过$()转化为JQuery对象。

1. JSON  JavaScript 对象表示法（**J**ava**S**cript **O**bject **N**otation） 是一种存储数据的方式。JSON对象由**名称/值对**组成，名称和值之间用冒号:隔开，名称必须用**双引号**包含起来。

4.18.在 JavaScript中async和await的用法：async用于声明一个function是异步的，而await 用于等待一个异步方法执行完成，并且await 只能出现在 async 函数中，async 函数返回的是一个 Promise 对象。

4.19敏捷实践包括：**精益软件开发（LSD）、SCRUM、极限编程、看板方法、水晶方法、功能驱动开发（FDD）、动态系统开发方法DSDM 、AUP敏捷统一过程、OpenUP。**

4.20在设置发送请求时，通过open函数设置背后的这个小线程，将要访问的页面url ，xmlhttp.open("GET",url,true);通过send函数进行实际的访问，xmlhttp.send(null);

4.21 要修改数据，就要使用setState方法，它可以有两个参数，第一个参数又有两种情况。

第一种情况是一个对象：

this.setState({

            isLiked:!this.state.isLiked

        })

第二种情况是一个方法：

this.setState(()=>{

            return{

                isLiked:!this.state.isLiked

            }

        })

第二个参数是一个回调函数，该函数会在setSatate函数调用完成并组件开始重新渲染的时候被调用，我们可以用该函数来监听渲染是否完成，也可以用来获取修改后的counter值

4.22 在前端连接后端的时候，使用的是fetch从远程读取数据

fetch() 函数的基本格式：

fetch(input,init).then(function(response){ ... });

其中：

input是要获取的资源，包含要获取资源的URL或一个Request对象

init包括所有请求的设置，可选的参数有：

 method：请求的方法，如GET、POST。

headers：请求的头信息，形式为Headers对象或者ByteString等等

一般的用法：

fetch(input)

.then( (response) => response.json() )

.then( (response) => {具体的设置或者是要改变的属性以及值} )

.catch( (err) => {加载数据出错执行函数} )

4.23 state与props

（1）state：通过用户交互，改变state的状态，然后更新UI，实现数据与界面一致

（2）props:用来给子组件传递数据，不可以变动，是只读的，而state对象内容可以根据用户交互改变

constructor(props) {

super(props);

this.state = {date: new Date()};

}

4.24 React里的key帮助React 识别哪些元素改变了，比如被添加或删除。因此应当给数组中的每一个元素赋予一个确定的标识。 key 只是在兄弟节点之间必须唯一。

4.25 分析功能状况；

获取所有的todo:使用GET method，这部分是每次做了动作就要获取所有的资料，然后render

删除和新增操作：新建按钮，执行动作，然后串API，与后台连接，添加的method是POST。

onAddBtnClick = () => {

if (this.state.inputContent.trim() === '') {

window.alert("请填入list内容");

} else {

var url = '/api/tasks/';

var data = { content: this.state.inputContent };

console.log(data)

fetch(url, {

method: 'POST',

body: JSON.stringify(data),

headers: new Headers({

'Content-Type': 'application/json'

})

}).then(res => res.json())

.catch(error => console.error('Error:', error))

.then(data => {

console.log(data)

if (data.status === false) {

window.alert(data.message);

} else {

window.location.reload();

}

this.setState({

inputContent: ''

})

});

}

}