张皓明 Haoming ZHANG

[公司名称] | [公司地址]

[文档标题]

[文档副标题]

目录

[一、 用户界面 4](#_Toc56070639)

[现状 4](#_Toc56070640)

[简介 4](#_Toc56070641)

[问题 6](#_Toc56070642)

[预期 6](#_Toc56070643)

[方法列表 7](#_Toc56070644)

[二、 数据中心 9](#_Toc56070645)

[现状 9](#_Toc56070646)

[问题 9](#_Toc56070647)

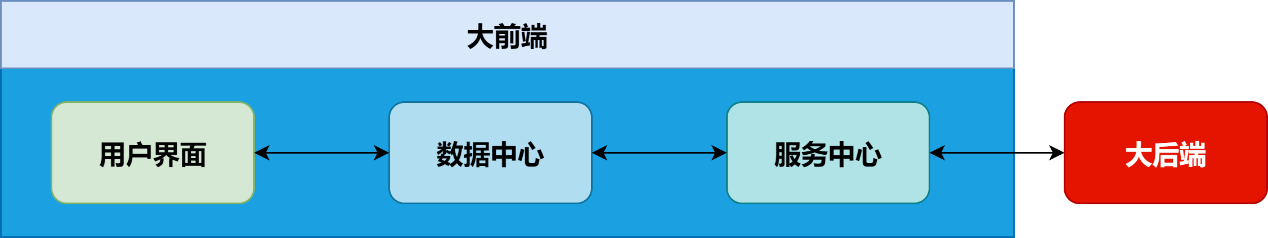
[预期 9](#_Toc56070648)

[三、 服务中心 10](#_Toc56070649)

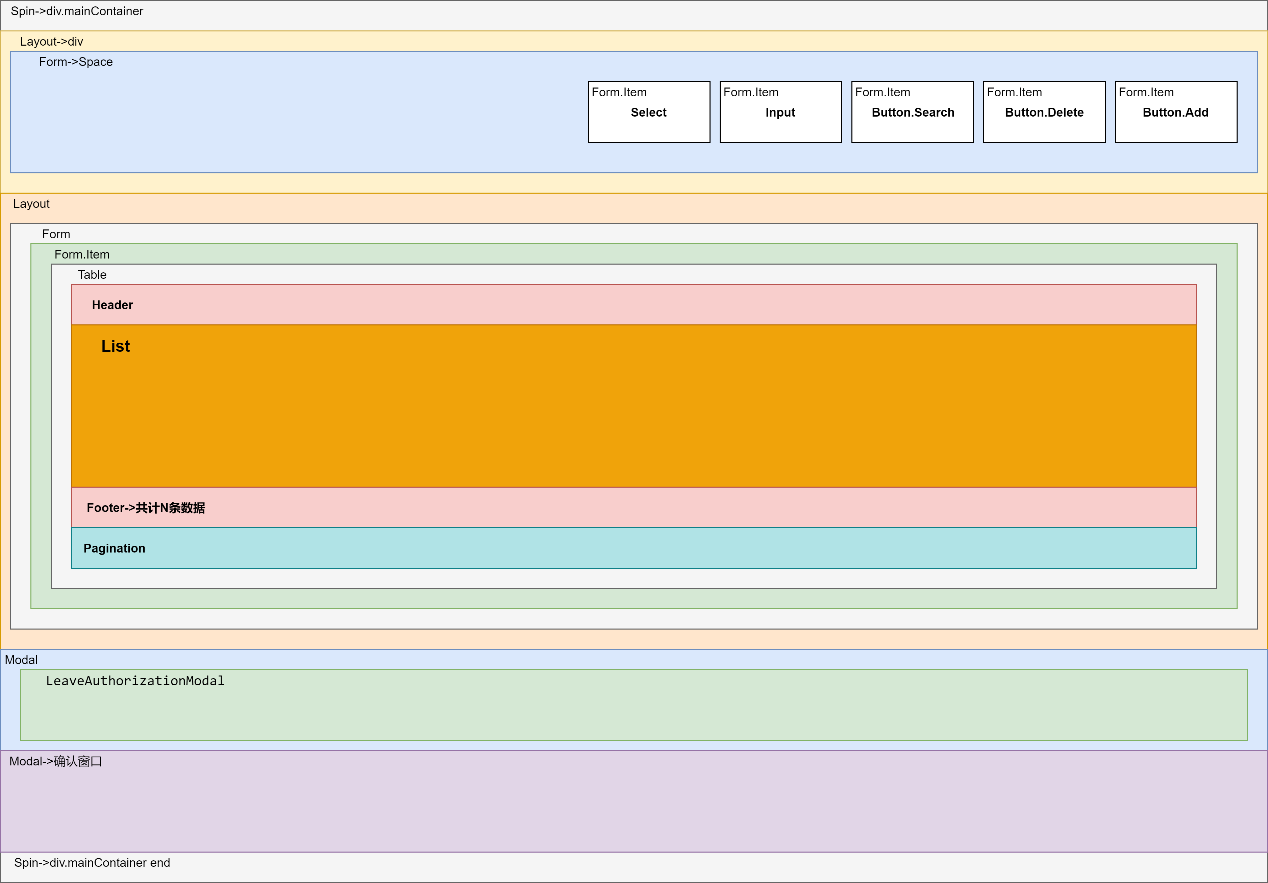
[现状 10](#_Toc56070650)

[问题 10](#_Toc56070651)

[预期 10](#_Toc56070652)



以LeaveAuthrization节点为例：



# 用户界面

## 现状

1. 文件路径：pages/BPM\_RM/LeaveAuthrization.js
2. 已经和【数据中心】建立对接关系，包括数据源、加载状态、查询条件、当前编辑数据项；
3. 页面更新仍旧通过state触发，model中数据更新不能触发页面更新；
4. 生命周期：
   1. componentDidMount:实例化this.form和this.formSearch表单对象，调用this.loadData [异步方法]以加载数据；
   2. this.loadData: 初始化查询条件、修改加载状态[[1]](#加载状态)、清空已选列表[state.selectedRowKeys]、调用dispatch远程加载数据，通过payload传递参数[分页信息：页面行数，当前页码、查询条件、回调方法：queryCallBack]；
   3. queryCallBack: 解析参数得到：PaginationTotal、dataSource、allCount、pagi\_total、spinning，用allCount[~~PaginationTotal~~]更新this.pagination.total，更新state中的dataSource**[[更新table数据]](#更新table)**、allCount、~~pagi\_total~~、spinning。
5. 数据操作：
   1. 新增：this.handleAddRecord:初始化数据对象newRecord、设置state对象[editingRecord:newRecord、operation:insertion、modalShow: true] end；
   2. 编辑：单击行编辑按钮或者双击行即可打开编辑窗口，调用this.handleEditRecord(record): 设置state对象[editingRecord:record、operation:update、modalShow: false]，dispatch到数据中心中end；
   3. 删除，理论上分为单项删除和多项删除，删除后通过回调函数重新加载数据：
      1. 单项删除：点击行删除按钮，弹出PopConfirm组件，onConfirm事件触发this.handleDeleteRecord(record)，确认state.dataSource中包含record记录（否则不执行后续操作），清空state.selectedRowKeys，组织待删除ID数组，作为dispatch参数，触发数据中心的**批量**删除方法(deleteItems)；
      2. 多项删除：先确认已选择需要删除的数据项[selectedRowKeys.length>0]，>0则设置state对象[notificationModalShow: true]，在弹出的确认窗口中点击[确认删除]以后，设置state.notificationModalShow为false，组织待删除ID数组，dispatch数据中心的**批量**删除方法(deleteItems)；
   4. 搜索：this.handleSearch，触发this.loadData；
   5. 分页：构造函数中定义this.pagination，结构参考antd默认项：
      1. pageSize: default 10
      2. total: default 0
      3. current: default 1
      4. showQuickJumper: default true
      5. onChange(page, pageSize): 更新：this.pagination.pageSize、this.pagination.current、state.pageSize、state.current，回调加载数据(this.loadData)；
   6. 表格：把table相关的配置信息整合后作为options传给table展开，集中管理，注意：因为是通过state维护的，所以不能通过tableOptions传递优化点；

## 问题

* 1. 更新有问题，更新的时候会更新错数据；
  2. 系统级问题：刷新界面后没有自动定义到菜单中并自动展开，之前已在研发中，因其他事情延误；
  3. 发起查询前应修改loading（spinning）状态为true；
  4. 【已解决】简化state中的变量，PaginationTotal、allCount、pagi\_total感觉很累赘；
  5. 系统级问题：容器内部的滚动条显示问题；
  6. SearchSquare尚未拆分成为独立组件；

## 预期

1. 新增按钮显隐控制作为配置项；
2. 单击行自动选中/反选；
3. [**FAILED**]弹窗编辑：打开窗口前，将数据中心中的当前编辑项设置好，打开弹窗后，获取该数据进行处理；
4. **将该节点进一步拆分为多个组件或者方法，高度组件化和解耦合**；
   * 1. [**ONGOING**]将SearchSquare成功拆分成为独立组件；
     2. 。。。

## 注意

1. 在通过点击SearchSquare获取数据的时候，查询条件必须先到达数据中心；
2. 查询结束后返回结果必须通过回调函数更新页面中的变量，否则界面不会渲染（遵守state触发页面渲染的原则）；

## 方法列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 参数 | 逻辑 | 用途 | 备注 |
| handleAddRecord | - |  | 新增数据事件 | 点击新增按钮触发 |
| handleEditRecord | record | 将record dispatch到数据中心，设置currentEditingRecord  为record。 | 编辑行事件 | 编辑按钮或者双击行触发 |
| handleDeleteRecord | record |  | 删除单项 |  |
| handleDeleteSelectedRecords | - |  | 删除选中项 |  |
| loadData | - |  | 加载数据 | async |
| queryCallBack | e |  | 加载数据回调函数 |  |
| ~~handleSearch~~ | ~~e~~ |  | ~~搜索按钮事件~~ | 弃用 |
| updateOkButtonAvailable | value |  | 编辑窗口OK按钮可用性控制 |  |
| updateCancelButtonAvailable | value |  | 编辑窗口CANCEL按钮可用性控制 |  |
| getUpdateColumns | record |  | 获取需要更新的列的信息 |  |
| handleOkModal | - |  | 编辑窗口OK按钮点击事件 |  |
| handleCancelModal | - |  | 编辑窗口CANCEL按钮点击事件 |  |
| handleTableRowSelectedChange | selectedRowKeys |  | 更新选中行信息 |  |
| updateEditingRecordState | record |  | LeaveAuthrizationModal中的updateParentState方法，更新主页编辑行 |  |
| handleFilterSelectChange | e |  | 暂不明 | 未启用 |

# 数据中心

## 现状

1. 文件路径：models/leaveauth.js
2. 根据UMIJS规范格式定义结构内容namespace、state、effects、reducers；
3. Effects定义方式：

\*fetchData ({payload}, {put, call, select}) {

yield call(外部定义函数)

yield put(reducers)

yield select(state=>state数据筛选)

}

Effects定义方式

1. Reducers定义方式：

‘saveData’: (state, action 或者 {payload}) => {

// action 为type或者payload, payload中可以定义任何参数

return {

…state // 返回一个新的state对象

}

}

Reducers定义方式

## 问题

暂无

## 预期

1. 批量处理dispatch，很有可能出现的连续操作，如：在SearchSquare的文本框中输入内容后直接点击回车开始查询，而此时需要先把条件内容传输到数据中心后才能触发查询的dispatch。想法是：effects中对批量dispatch进行顺序处理，结构为{id, dispatchName, dispatchPayloads}，dispatchPayloads中可能传入回调函数，对dispatchPayloads类型进行解析，如果是基础类型或者数组直接传给dispatchName为参数用；如果是对象类型，进行一次解构，每个解构子对象再进行一次判定，如果包含函数类型的则作为回调顺序执行；如果是函数类型，作为dispatchName的回调对象传入。
2. 。。。

## 方法列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 参数 | Effects(A)、Reducers(B) | 用途 | 备注 |
| fetchData | {payload}, {put} | A | 获取远程数据 | 调用service  方法 |
| insertItem | {payload} | A | 插入数据 |
| updateItem | {payload} | A | 更新数据 |
| deleteItems | {payload} | A | 删除数据 |
| saveDataWithRemoteResponse | state, {payload} | B | 获取远程数据后，更新本中心的state对象 |  |
| setSearchCondition | state, {payload} | B | 设定远程查询条件(state中的对象) |  |
| setCurrentEditingRecord | state, {payload} | B | 设定当前编辑项(state中的对象) | 弹出编辑时的对象 |

# 服务中心

## 现状

## 问题

## 预期