# 之前的笔记链接

react全家桶学习笔记1：[路由基础配置](react全家桶学习笔记1-路由基础配置.docx)

react全家桶学习笔记2：[首页布局和菜单渲染](react全家桶学习笔记2-首页布局和菜单渲染.docx)

react全家桶学习笔记3：[二级路由配置](react全家桶学习笔记3-二级路由配置.docx)

react全家桶学习笔记4： [配置菜单默认项](react全家桶学习笔记4-设置菜单默认项.docx)

react全家桶学习笔记5： [axios配置网络请求](react全家桶学习笔记5-axios设置网络请求.docx)

react全家桶学习笔记6：[用户列表渲染](react全家桶学习笔记6-antd用户列表渲染.docx)

react全家桶学习笔记7：[路由传参](react全家桶学习笔记7-路由传参.docx)

react全家桶学习笔记8: [状态机初始化配置](react全家桶学习笔记8-状态机初始配置.docx)

react全家桶学习笔记9: [redux获取分类数据](react全家桶学习笔记9-redux获取分类数据.docx)

react全家桶学习笔记10：[渲染一二级分类数据以及优化](react全家桶学习笔记10-redux渲染一二级分类以及优化处理.docx)，

# react全家桶学习笔记11：redux获取用户数据

# 继续React-fam-bucketdemo项目，我们学习获取用户信息并且保存到仓库

# 我们在财务统计的销售金额菜单那里来完成用户功能的状态机版本

## 需要先修改菜单组件，找到财务统计的销售金额菜单给他添加一个Link组件

|  |
| --- |
|  |

## 2.在index目录下面新建一个finances文件夹，然后新建一个叫做FinamcesSale.jsx的组件

|  |
| --- |
|  |

## 3.在App.jsx中配置路由，注意：使用之前需要导入组件

|  |
| --- |
|  |

#### 效果：此时点击菜单就会进入对于的组件

|  |
| --- |
|  |

### 注意：因为后端没有销售金额的接口，所以我们在这里用用户数据接口来练习状态机的使用流程

## 4.在store文件夹里面新一个users文件夹，然后在这个文件夹里面新一个reducers.js和actions.js文件

|  |
| --- |
|  |

## 5.在reducers.js里面添加初始化代码，以后还需要修改

|  |
| --- |
| **export let** *usersReducers* = (state=[],{type,payload}) =>{  **switch** (type) {  **default**:  **return** state  } } |

## 6.需要在store/index.js里面合并usersReducers

|  |
| --- |
| **import** {*legacy\_createStore*,*combineReducers*,*applyMiddleware*} **from "redux"**; **import** {*rolesReducer*} **from "./roles/reducers"**; *//引入redux-thunk中间件* **import** {***thunk*** }**from 'redux-thunk' import** {*categoryReducer*} **from "./category/reducers"**; **import** {*usersReducers*} **from "./users/reducers"**;  *//合并所有模块的reducer函数* **let** allReducer = *combineReducers*({  roles:*rolesReducer*,  category:*categoryReducer*,  users:*usersReducers* });  *//应用中间件* **let** middleware = *applyMiddleware*(***thunk***); **let *store*** = *legacy\_createStore*(allReducer,middleware);   **export default *store*** |

## 7.可以在FinancesSale.jsx组件里面测试一下看看有没有users

|  |
| --- |
| **import *React* from 'react'**; **import** {***useSelector***} **from "react-redux"**;  **const** *FinancesSale* = () => {  ***useSelector***(state => {  ***console***.log(state)  })  **return** (  <**div**>  销售金额  </**div**>  ); };  **export default** *FinancesSale*; |

### 效果：

|  |
| --- |
|  |

#### 可见，初始值的没有问题的

## 8.需要获取数据，使用流程：组件调仓库，仓库调api，因为users的api我们已经有了（和老师的不一样），只需要编写actions中的异步方法来定义api，我们给store/users/actions.jx添加异步方法

|  |
| --- |
| **import** {*getUserApi*} **from "../../api/users"**;  **export let** *getUsersAsync* = () =>{  **return async** (dispatch) =>{  **let** res = **await** *getUserApi*()  ***console***.log(**"用户数据"**,res)  } } |

## 9.在组件里面调用actions的异步方法，看看是否能够获取到数据

|  |
| --- |
| **import** React from **'react'**; **import** {useSelector} from **"react-redux"**;  **const** FinancesSale = () => {  useSelector(state => {  console.log(state)  })   **return** (  <**div**>  销售金额  </**div**>  ); };  **export default** FinancesSale; |

### 效果：是能够看到初始值

|  |
| --- |
|  |

## 10.然后我们需要在组件里面调用actions的方法来获取数据

### src/pages/index/finances/FinancesSale.jsx

|  |
| --- |
| **import *React***,{*useEffect*} **from 'react'**; **import** {***useSelector***,***useDispatch***} **from "react-redux"**; **import** {*getUsersAsync*} **from "../../../store/users/actions"**;   **const** *FinancesSale* = () => {  **let** dispatch = ***useDispatch***()  ***useSelector***(state => {  ***console***.log(state)  })  *//定义组件自己的方法* **let** getUsers = () =>{  dispatch(*getUsersAsync*())  }   *useEffect*(()=>{  getUsers()  },[])  **return** (  <**div**>  销售金额  </**div**>  ); };  **export default** *FinancesSale*; |

#### 效果：

|  |
| --- |
|  |

## 11.继续完善actions.js里面的方法

### store/users/actions.js

|  |
| --- |
| **import** {*getUserApi*} **from "../../api/users"**;  *//封装方法来返回action对象* **let** setUserAction = (payload) =>{  **return** {**type**:**"SET\_USERS"**,payload} }  **export let** *getUsersAsync* = () =>{  **return async** (dispatch) =>{  **let** res = **await** *getUserApi*()  **if**(res){  ***console***.log(**"用户数据"**,res)  *//触发reducer方法并且传递数据给他，让他保存数据到仓库* dispatch(setUserAction(res)) *//需要封装方法来分号action对象* }   } } |

## 12.完善reducers.js的方法

### store/users/reducers.js

|  |
| --- |
| **export let** *usersReducers* = (state=[],{type,payload}) =>{  **switch** (type) {  **case 'SET\_USERS'**:  **return** payload  **default**:  **return** state  } } |

### 然后我们在组件里面测试一下，改变actions里面的log

|  |
| --- |
| **import *React***,{*useEffect*} **from 'react'**; **import** {***useSelector***,***useDispatch***} **from "react-redux"**; **import** {*getUsersAsync*} **from "../../../store/users/actions"**;   **const** *FinancesSale* = () => {  **let** dispatch = ***useDispatch***()  ***useSelector***(state => {  ***console***.log(**"仓库里面的用户数据:"**,state)  })  *//定义组件自己的方法* **let** getUsers = () =>{  dispatch(*getUsersAsync*())  }   *useEffect*(()=>{  getUsers()  },[])  **return** (  <**div**>  销售金额  </**div**>  ); };  **export default** *FinancesSale*; |

#### 效果：

|  |
| --- |
|  |

## 13.然后我们需要再做一次封装，把type封装为常量，在store/users文件夹下面新建一个actionTypes.js文件，内容如下

|  |
| --- |
| **export const *SET\_USERS*** = **"SET\_USERS"** |

## 然后修改actions.js的代码

### store/users/actions.js

|  |
| --- |
| **import** {*getUserApi*} **from "../../api/users"**; **import** {***SET\_USERS***} **from "./actionTypes"**;   *//封装方法来返回action对象* **let** setUserAction = (payload) =>{  **return** {**type**:***SET\_USERS***,payload} }  **export let** *getUsersAsync* = () =>{  **return async** (dispatch) =>{  **let** res = **await** *getUserApi*()  **if**(res){  *// console.log("用户数据",res)  //触发reducer方法并且传递数据给他，让他保存数据到仓库* dispatch(setUserAction(res)) *//需要封装方法来分号action对象* }   } } |

### store/users/reducers.js

|  |
| --- |
| **import** {***SET\_USERS***} **from "./actionTypes"**;  **export let** *usersReducers* = (state=[],{type,payload}) =>{  **switch** (type) {  **case *SET\_USERS***:  **return** payload  **default**:  **return** state  } } |

## 14，然后就可以获取仓库数据然后渲染表格

|  |
| --- |
|  |
|  |

### 这个笔记的主要作用是梳理状态机的完整使用流程，需要好好体会