2017年6月19日 星期一

redux详解

学习redux: containers这个文件是大关卡,理解它很总要！containers里面写的是容器组件；顾名思义它理应只是拿来承载props的；承载props;自然是为传给UI组件来显示；containers里面的组件并不应该拿来显示内容的；

那么想想我们怎么得到props呢?

props包含两种东西：一个是某个reducer对应的state的值；或者是多个reducer对应多个state的值；

另一个是：你想你的UI组件需要去触发action对吧？但是你不能直接在UI组件写action吧？所以使用mapDispatchToProps方法来加载出所有当前容器组件需要使用的action;把action当做容器组件的属性；这样你的UI组件不就可以拿到这个containers文件中容器组件的action了么?

reducer一般做两件事情：

第一件事情；将当前reducer文件里面要用到state转为props给显示组件用；；

connect(mapStateToProps,mapDispatchToProps)(MaterialApi);

这里connect方法里面的两个方法mapStateToProps会返回你想要的那几个reducer对应的几个state 的某个值；mapDispatchToProps会返回你想在当前这个 containers文件中需要用到的action;connect这个高阶函数会将这两个方法的返回值都传给MaterialApi这个容器组件；明白了吧；现在你的容器组件就可以

使用到；你想用的某个reducer的state的值了；你可以拿到action了；action可以让你当做容器组件的属性传给UI组件来触发了；

```
function mapStateToProps(state) {
return {
```

user: state.reducer1.user,

age: state.reducer2.age,

param: state.reducer3.param,

}

}

如上：你看到了mapStateToProps(state)方法的参数state了吗？它是什么？怎么来的?

const rootReducer = combineReducers({

reducer1,

reducer2,

reducer3,

reducer4,

});

在我们开发项目的时候；必定有多个reducer文件;使用combineEeducers方法将多个reducer合并成一个；

我们知道一个reducer就会返回一个新的state;现在多个reducer合并成了一个reducer;下面有将这个总的reducer

合并成了一个store;

const store = createStore(reducers);

我一直觉得store是个概念性的东西；形象点说；一个reducer对应一个state;然后通过combineReducers将多个state合并成了一个大的state对象; 这个大的state那就是mapStateToProps方法的参数state的来源了；

既然你知道了这个大的state就是由多个小的reducer的state的合并的对象！记住它是一个对象；可以用他逆向拿到某个reducer；拿到某个

reducer的某个值了；如上 user: state.reducer1.user,

这样就拿到名叫reducer1的reducer的state的user值了；

同时：mapStateToProps方法会将我们拿到的state.user转为props;

随后我们在容器组件 ； 在render方法中直接

{ user,age ,param} = this.props;

哈哈；容器组件就可以把它当做属性值来传给UI组件啦！！！

第二件事情：在mapDispatchToProps方法中加载出这个reducer要用到的action;这样拿到action之后；让action作为容器组件的属性；这样component的UI

组件就可以调用这个action了；

```
function mapDispatchToProps(dispatch) {
return {
dispatch,
actionName1: (params)=>dispatch(actions.actionName1(params)),
actionName2: (params)=>dispatch(actions.actionName2(params)),
actionName3: (params)=>dispatch(actions.actionName3(params)),
actionMame4: (params)=>dispatch(actions.actionName4(params)),
}
}
```

如上我们在mapDispatchToProps方法中加载出我们当前containers文件要调用到的action;(是写好需要用到的Action而不是所有的action) 和我们的容器组件需要传给自己的UI组件的action;

例如我们在当前页面有一个：componentWillMount方法；在里面初始化 一个action；直接actionName1（）；

这样就会自动去触发dispatch(action);

总结一下：redux的设计思想简单的说；

第一步：action :不管三七二之一；模拟出事件；

第二步：reducer：开发中自然有多个reducer;

新建一个reducer文件只做一件事；吧多个reducer合并成一个reducer.

使用combineReducers方法来将多个reducer合并成一个；每个reducer会生成一个新的state;将多个reducer合并成一个reducer;那也就等同于将多个小的state合并成一个大的state对象；

第三步： const store = createStore(reducers); 将state交给store管理；

第四步：这样想：action和reducer都是单独的东西；store将存储着所有的state;

可以认为以上种种都是为了容器组件服务的；在containers里面计算得到state并转为props;得到需要的action也传给显示组件！！

第五步：容器组件中mapStateToProps方法可以得到所有的reducer对应的state并转为props;

mapDispatchToProps方法可以访问到所有的action;写好action:传给UI组件来用。