React.js 小书

<-- 返回首页

动手实现 Redux（六）：Redux 总结

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson35

（本文未审核）

不知不觉地，到这里大家不仅仅已经掌握了 Redux，而且还自己动手写了一个 Redux。我们从一个非常原始的代码开始，不停地在发现问题、解决问题、优化代码的过程中进行推演，最后把 Redux 模式自己总结出来了。这就是所谓的 Redux 模式，我们再来回顾一下这几节我们到底干了什么事情。

我们从一个简单的例子的代码中发现了共享的状态如果可以被任意修改的话，那么程序的行为将非常不可预料，所以我们提高了修改数据的门槛：你必须通过 `dispatch` 执行某些允许的修改操作，而且必须大张旗鼓的在 `action` 里面声明。

这种模式挺好用的，我们就把它抽象出来一个 `createStore`，它可以产生 `store`，里面包含 `getState` 和 `dispatch` 函数，方便我们使用。

后来发现每次修改数据都需要手动重新渲染非常麻烦，我们希望自动重新渲染视图。所以后来加入了订阅者模式，可以通过 `store.subscribe` 订阅数据修改事件，每次数据更新的时候自动重新渲染视图。

接下来我们发现了原来的“重新渲染视图”有比较严重的性能问题，我们引入了“共享结构的对象”来帮我们解决问题，这样就可以在每个渲染函数的开头进行简单的判断避免没有被修改过的数据重新渲染。

我们优化了 `stateChanger` 为 reducer，定义了 reducer 只能是纯函数，功能就是负责初始 `state`，和根据 `state` 和 `action` 计算具有共享结构的新的 `state`。

`createStore` 现在可以直接拿来用了，套路就是：

```
// 定一个 reducer
function reducer (state, action) {
  /* 初始化 state 和 switch case */
}

// 生成 store
const store = createStore(reducer)

// 监听数据变化重新渲染页面
```

```
store.subscribe(() => renderApp(store.getState()))

// 首次渲染页面
renderApp(store.getState())

// 后面可以随意 dispatch 了，页面自动更新
store.dispatch(...)
```

现在的代码跟 React.js 一点关系都没有，接下来我们要把 React.js 和 Redux 结合起来，用 Redux 模式帮助管理 React.js 的应用状态。

课后练习

-
- 实现 Users Reducer

> 因为第三方评论工具有问题，对本章节有任何疑问的朋友可以移步到 React.js 小书的论坛 发帖，我会回答大家的疑问。

下一节：动手实现 React-redux（一）：初始化工程

上一节：动手实现 Redux（五）：不要问为什么的 reducer

2 条评论，2 人参与。

2

我有话说...

使用社交帐号登录 发布前先点击左边的按钮登录

最新评论

时光荏苒~ • 7月3日 13:40

套路那里是不是少了新旧数据的对比

顶 • 回复 • 分享»

librAquarius • 4月19日 14:48

赞

顶 • 回复 • 分享»

友言?