React全家桶

课堂目标

1. 掌握redux
2. 掌握redux中间件
3. 掌握react-router4
4. 理解redux及其中间件原理

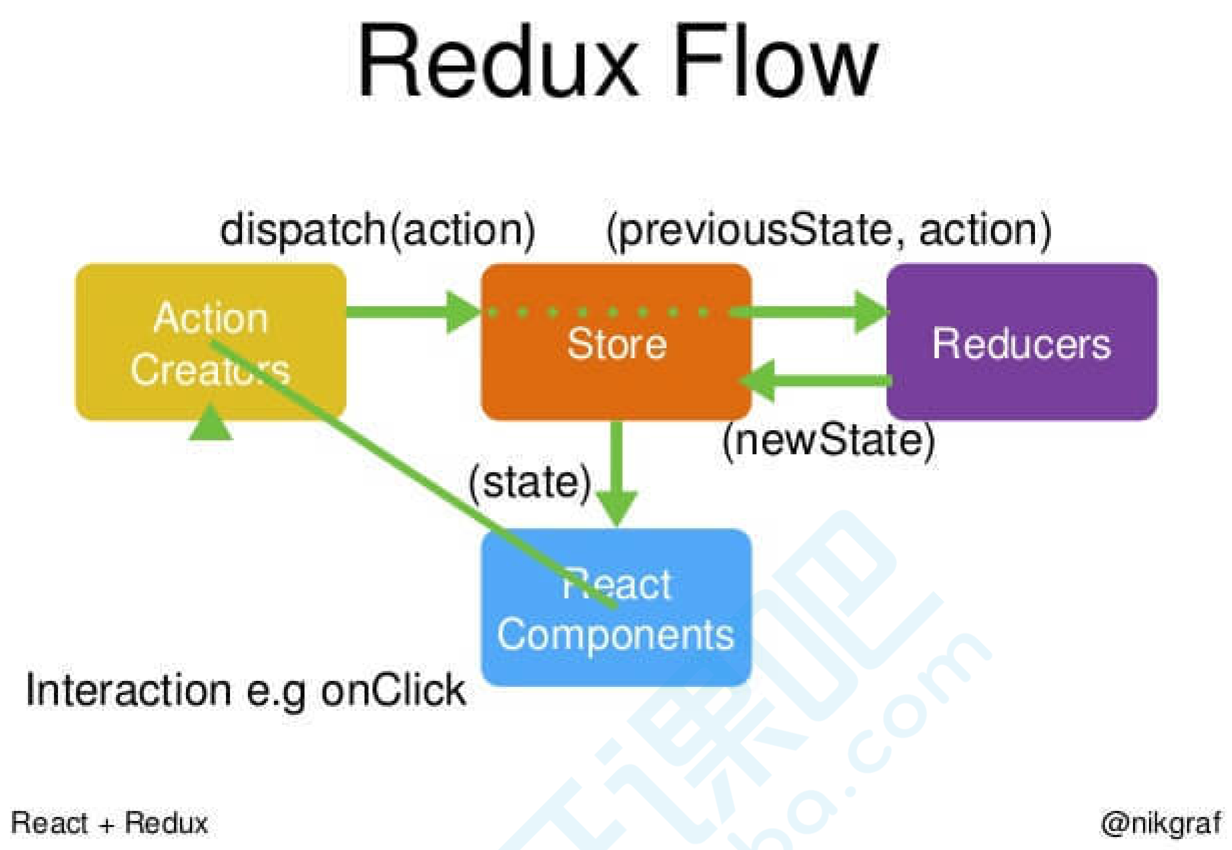
知识要点

1. 思考应用状态管理的模式
2. redux全局状态管理
3. react路由管理

预习资源

1. [redux]
2. [react-redux]
3. [react-router]

起步



redux

安装

`npm install redux --save`

使用

创建store.js

```
import {createStore} from 'redux'

const counterReducer = (state = 0, action) => {
    switch (action.type) {
      case 'add':
        return state + 1
      case 'minus':
        return state - 1
      default:
        return state
    }
  }
const store = createStore(counterReducer)

export default store
```

应用

app.js

```
import React from 'react'
import store from './store'

class App extends React.Component{
    render(){
        return <div>
            <p>{store.getState()}</p>
            <div>
                <button onClick={()=>store.dispatch({type:"add"})}>+</button>
                <button onClick={()=>store.dispatch({type:"minus"})}>-</button>
            </div>
        </div>
    }
}
export default App
```

订阅

index.js

```
import React from 'react'
import ReactDom from 'react-dom'
import App from './App'
import store from './store'
const render = ()=>{
  ReactDom.render(
    <App/>,
    document.querySelector('#root')
  )
}
render()

store.subscribe(render)
```

react-redux

react整合redux，简化使用难度，需要react-redux的支持

安装

```
npm install react-redux --save
```

注入store

index.js

```
import React from 'react'
import ReactDom from 'react-dom'
import App from './App'
import store from './store'

import { Provider } from 'react-redux'
ReactDom.render(
  <Provider store={store}>
    <App/>
  </Provider>,
  document.querySelector('#root')
)
```

使用状态

app.js

```
import React from 'react'
import {connect} from 'react-redux'
const mapStateToProps = (state)=>{
    return {
        num:state
    }
}
const mapDispatchToProps = dispatch=>{
    return {
        add: ()=>dispatch({type:"add"}),
        minus: ()=>dispatch({type:"add"})
    }
}

class App extends React.Component{
    render(){
        return <div>
            <p>{this.props.num}</p>
            <div>
                <button onClick={()=>this.props.add()}>+</button>
                <button onClick={()=>this.props.minus()}>-</button>
            </div>
        </div>
    }
}
export default connect(mapStateToProps,mapDispatchToProps)(App)
```

装饰器写法

```
import React from 'react'
import {connect} from 'react-redux'
```

```
@connect(
    state=>({num:state}),
    dispatch=>({
        add: ()=>dispatch({type:"add"}),
        minus: ()=>dispatch({type:"add"})
    })
)
class App extends React.Component{
    render(){
        return <div>
            <p>{this.props.num}</p>
            <div>
                <button onClick={()=>this.props.add()}>+</button>
                <button onClick={()=>this.props.minus()}>-</button>
            </div>
        </div>
    }
}
export default App
```

异步

redux只支持同步，实现异步任务需要中间件支持

安装

```
npm install redux-thunk redux-logger --save
```

试用redux-logger

store.js

```
import { applyMiddleware, createStore } from 'redux';
import logger from 'redux-logger'

const counterReducer = (state = 0, action) => {...}
const store = createStore(
    counterReducer,
    applyMiddleware(logger)
  );

export default store
```

试用redux-thunk

配置

store.js

```
import { applyMiddleware, createStore } from 'redux';
import logger from 'redux-logger'
import thunk from 'redux-thunk';

const counterReducer = (state = 0, action) => {...}
const store = createStore(
    counterReducer,
    applyMiddleware(logger,thunk)
  );

export default store
```

应用

App.js

```
import React from 'react'
import { connect } from 'react-redux'

@connect(
  state => ({ num: state }),
  {
    add: () => ({ type: "add" }),
    minus: () => ({ type: "minus" }),
    asyncAdd: () => dispatch => {
      setTimeout(() => {
        // 异步结束后，手动执行dispatch
        dispatch({ type: "add" });
      }, 1000);
    }
  }
)
class App extends React.Component {
  render() {
    return <div>
      <p>{this.props.num}</p>
      <div>
        <button onClick={() => this.props.add()}>+</button>
        <button onClick={() => this.props.minus()}>-</button>
        <button onClick={() => this.props.asyncAdd()}>延迟添加</button>
      </div>
    </div>
  }
}
export default App
```

> 代码需重构：抽离reducer和action

react-router-4

安装

`npm install --save react-router-dom`

应用路由

index.js

```
import React from 'react'
import ReactDom from 'react-dom'
import { Provider } from 'react-redux'

import { applyMiddleware, createStore } from 'redux';
import logger from 'redux-logger'
import thunk from 'redux-thunk';
import {counterReducer} from './counter.redux'
import App from './App'

import { BrowserRouter} from "react-router-dom";

const store = createStore(
  counterReducer,
  applyMiddleware(logger,thunk)
);

ReactDom.render(
  <BrowserRouter>
    <Provider store={store}>
      <App/>
    </Provider>
  </BrowserRouter>,
  document.querySelector('#root')
)
```

配置、导航

app.js

```
import React from 'react'
import { connect } from 'react-redux'
import {add, minus,asyncAdd} from './counter.redux'
import {Route,Link} from 'react-router-dom'

function About(){
  return <div>About</div>
}
function Detail(){
  return <div>Detail</div>
}

@connect(
  state => ({ num: state }),
  {add, minus,asyncAdd}
)
class Counter extends React.Component {
  render() {
    return <div>
      <p>{this.props.num}</p>
      <div>
        <button onClick={() => this.props.add()}>+</button>
        <button onClick={() => this.props.minus()}>-</button>
        <button onClick={() => this.props.asyncAdd()}>延迟添加</button>
      </div>
    </div>
  }
}

class App extends React.Component{
  render(){
    return <div>
      <ul>
        <Link to="/">累加器</Link>
        <Link to="/about">About</Link>
        <Link to="/detail">Detail</Link>
      </ul>
      <div>
      <Route exact path="/" component={Counter} />
      <Route path="/about" component={About} />
      <Route path="/detail" component={Detail} />
      </div>
    </div>
  }
}

export default App
```

动态路由

使用 `:id` 的形式定义参数

```
<Route path="/detail/:id" component={Detail} />

function Detail(props){
  return <div>Detail :{props.match.params.id}</div>
}
```

redux原理

```
export function createStore(reducer, enhancer){
    if (enhancer) {
        return enhancer(createStore)(reducer)
    }
    let currentState = {}
    let currentListeners = []

    function getState(){
        return currentState
    }
    function subscribe(listener){
        currentListeners.push(listener)
    }
    function dispatch(action){
        currentState = reducer(currentState, action)
        currentListeners.forEach(v=>v())
        return action
    }
    dispatch({type:'@IMOOC/WONIU-REDUX'})
    return { getState, subscribe, dispatch}
}

export function applyMiddleware(...middlewares){
    return createStore=>(...args)=>{
        const store = createStore(...args)
        let dispatch = store.dispatch

        const midApi = {
            getState:store.getState,
            dispatch:(...args)=>dispatch(...args)
        }
        const middlewareChain = middlewares.map(middleware=>middleware(midApi))
        dispatch = compose(...middlewareChain)(store.dispatch)
        return {
            ...store,
            dispatch
        }

    }
}
export function compose(...funcs){
    if (funcs.length==0) {
        return arg=>arg
    }
    if (funcs.length==1) {
        return funcs[0]
    }
    return funcs.reduce((ret,item)=> (...args)=>ret(item(...args)))
}
function bindActionCreator(creator, dispatch){
    return (...args) => dispatch(creator(...args))
}
export function bindActionCreators(creators,dispatch){
    return Object.keys(creators).reduce((ret,item)=>{
        ret[item] = bindActionCreator(creators[item],dispatch)
        return ret
    },{})
}
```

react-redux原理

```
import React from 'react'
import PropTypes from 'prop-types'
import {bindActionCreators} from './woniu-redux'

export const connect = (mapStateToProps=state=>state,mapDispatchToProps={})=>
(WrapComponent)=>{
    return class ConnectComponent extends React.Component{
        static contextTypes = {
            store:PropTypes.object
        }
        constructor(props, context){
            super(props, context)
            this.state = {
                props:{}
            }
        }
        componentDidMount(){
            const {store} = this.context
            store.subscribe(()=>this.update())
            this.update()
        }
        update(){
            const {store} = this.context
            const stateProps = mapStateToProps(store.getState())
            const dispatchProps = bindActionCreators(mapDispatchToProps,
store.dispatch)
            this.setState({
                props:{
                    ...this.state.props,
                    ...stateProps,
                    ...dispatchProps
                }
            })
        }
        render(){
            return <WrapComponent {...this.state.props}></WrapComponent>
        }
    }
}

export class Provider extends React.Component{
    static childContextTypes = {
        store: PropTypes.object
    }
    getChildContext(){
        return {store:this.store}
    }
    constructor(props, context){
        super(props, context)
        this.store = props.store
    }
    render(){
        return this.props.children
    }
}
```

redux-thunk原理

```
const thunk = ({dispatch,getState})=>next=>action=>{
    if (typeof action=='function') {
        return action(dispatch,getState)
    }
    return next(action)
}
export default thunk
```

回顾