React全家桶2

课堂目标

1. 掌握umi
2. 掌握redux解决方案--dva
3. 掌握generator
4. 掌握redux-saga

资源

1. umi
2. dva
3. redux-saga：中文、英文
4. generator

知识要点

1. umi架构思想和用法
2. 状态管理解决方案dva

起步

redux-saga使用

- 安装：**npm install --save redux-saga**
- 使用：用户登录redux-saga实现

创建./store/sagas.js

```
import { call, put, takeEvery } from "redux-saga/effects";

// 模拟登录
const UserService = {
  login(uname) {
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        if (uname === "Jerry") {
          resolve({ id: 1, name: "Jerry", age: 20 });
        } else {
          reject("用户名或密码错误");
        }
      }, 1000);
```

```
    });
  }
};

// worker Saga
function* login(action) {
  try {
    yield put({ type: "requestLogin" });
    const result = yield call(UserService.login, action.uname);
    yield put({ type: "loginSuccess", result });
  } catch (message) {
    yield put({ type: "loginFailure", message });
  }
}

function* mySaga() {
  yield takeEvery("login", login);
}

export default mySaga;
```

修改user.redux.js

```
export const user = (
  state = { isLogin: false, loading: false, error: "" },
  action
) => {
  switch (action.type) {
    case "requestLogin":
      return { isLogin: false, loading: true, error: "" };
    case "loginSuccess":
      return { isLogin: true, loading: false, error: "" };
    case "loginFailure":
      return { isLogin: false, loading: false, error: action.message };
    default:
      return state;
  }
};
export function login(uname) {
  return { type: "login", uname };
}
// export function login() {
//   return dispatch => {
//     dispatch({ type: "requestLogin" });
//     setTimeout(() => {
//       dispatch({ type: "login" });
//     }, 2000);
//   };
// }
```

注册redux-saga，./store/index.js

```
import { user } from "./user.redux";
import createSagaMiddleware from "redux-saga";
import mySaga from "./sagas";

const sagaMiddleware = createSagaMiddleware();
const store = createStore(
  combineReducers({ user }),
  applyMiddleware(logger, sagaMiddleware)
);

sagaMiddleware.run(mySaga);
export default store;
```

使用状态，RouteSample.js

```
const Login = connect(
  state => ({
    isLogin: state.user.isLogin,
    loading: state.user.loading,
    error: state.user.error // 登录错误信息
  }),
  { login }
)(({ location, isLogin, login, loading, error }) => {// 登录错误信息
  const redirect = location.state.redirect || "/";
  const [uname, setUname] = useState(""); // 用户名输入状态
  if (isLogin) return <Redirect to={redirect} />;

  return (
    <div>
      <p>用户登录</p>
      <hr />
      {/* 显示错误信息 */}
      {error && <p>{error}</p>}
      {/* 输入用户名 */}
      <input
        type="text"
        onChange={e => setUname(e.target.value)}
        value={uname}
      />
      {/* 登录传参 */}
      <button onClick={() => login(uname)} disabled={loading}>
        {loading ? "登录中..." : "登录"}
      </button>
    </div>
  );
});
```

###

React企业级应用程序框架 - Umi

全局安装umi：

```
npm install umi -g
```

新建 index页

```
umi g page index
umi g page about
```

起服务看效果

```
umi dev
```

动态路由：以$开头的文件或目录

```
// 创建users/$id.js，内容和其他页面相同，显示一下传参
export default function(props) {
  return (
    <div>
      <h1>user id: {props.match.params.id}</h1>
    </div>
  );
}
```

嵌套路由

```
// 创建父组件 umi g page users/_layout
export default function(props) {
  return (
    <div>
      <h1>Page _layout</h1>
      <div>{props.children}</div>
    </div>
  )
}
// 创建兄弟组件 umi g page users/index
```

页面跳转

```
// 用户列表跳转至用户详情页，users/index.js
import Link from "umi/link";
import router from "umi/router";

export default function() {
  // 模拟数据
  const users = [{ id: 1, name: "tom" }, { id: 2, name: "jerry" }];
  return (
```

```
    <div className={styles.normal}>
      <h1>用户列表</h1>
      <ul>
        {users.map(u => (
          // 声明式
          // <li key={u.id}>
          //   <Link to={`/users/${u.id}`}>{u.name}</Link>
          // </li>
          // 命令式
          <li key={u.id} onClick={()=>router.push(`/users/${u.id}`)}>{u.name}</li>
        ))}
      </ul>
    </div>
  );
}
```

配置式路由：默认路由为声明式，根据pages下面内容自动生成路由，业务复杂后仍需配置路由

```
// 创建config/config.js
export default {
  routes: [
    { path: "/", component: "./index" },
    {
      path: "/users",
      component: "./users/_layout",
      routes: [
        { path: "/users/", component: "./users/index" },
        { path: "/users/:id", component: "./users/$id" }
      ]
    }
  ]
};
```

404页面：

- 创建404页面：`umi g page NotFound`
- 添加不带path的路由配置项：`{component: './NotFound'}`

权限路由：

- 通过配置路由的 `Routes` 属性来实现

```
{
    path: "/about",
    component: "./about",
    Routes: ["./routes/PrivateRoute.js"] // 这里相对根目录，文件名后缀不能少
}
```

- 创建./routes/PrivateRoute.js

```
import Redirect from "umi/redirect";

export default props => {
  // 50%概率需要去登录页面
  if (Math.random()>0.5) {
    return <Redirect to="/login" />;
  }
  return (
    <div>
      <div>PrivateRoute (routes/PrivateRoute.js)</div>
      {props.children}
    </div>
  );
};
```

- 创建登录页面：`umi g page login`，并配置路由：`{ path: "/login", component: "./login" }`

```
export default function() {
  return (
    <div className={styles.normal}>
      <h1>Page login</h1>
    </div>
  );
}
```

引入antd

- 添加antd：`npm install antd -S`
- 添加 umi-plugin-react: `npm install umi-plugin-react -D`

> Win10有权限错误，通过管理员权限打开vscode

- 修改./config/config.js

```
plugins: [
    ['umi-plugin-react', {
      antd: true
    }],
],
```

```
import {Button} from 'antd'
export default () => {
  return <div>
    <Button>登录</Button>
  </div>
}
```

引入dva

配置dva

```
export default {
  plugins: [
    ['umi-plugin-react', {
      antd: true,
      dva: true,
    }],
  ],
  // ...
}
```

创建model：维护页面数据状态

- 新建./models/goods.js

```
export default {
  namespace: 'goods', // model的命名空间，区分多个model
  state: [{ title: "web全栈" },{ title: "java架构师" }], // 初始状态
  effects:{}, // 异步操作
  reducers: { // 更新状态 }
}
```

使用状态:

- 创建页面goods.js：`umi g page goods`，并配置路由：`{ path: "/goods", component: "./goods" }`

```
import React, { Component } from "react";
import { Button, Card } from "antd";
import { connect } from "dva";

@connect(
  state => ({
    goodsList: state.goods // 获取指定命名空间的模型状态
  }),
  {
    addGood: title => ({
      type: "goods/addGood", // action的type需要以命名空间为前缀+reducer名称
      payload: { title }
    })
  }
)
class Goods extends Component {
  render() {
    return (
      <div>
        {/* 商品列表 */}
        <div>
          {this.props.goodsList.map(good => {
            return (
              <Card key={good.title}>
```

```
                <div>{good.title}</div>
              </Card>
            );
          })}
          <div>
            <Button
              onClick={() =>
                this.props.addGood("商品" + new Date().getTime())
              }
            >
              添加商品
            </Button>
          </div>
        </div>
      </div>
    );
  }
}
export default Goods;
```

- 更新模型src/models/goods.js

```
export default {
  reducers: {
    addGood(state, action) {
      return [...state, {title: action.payload.title}];
    }
  }
}
```

数据mock：模拟数据接口

mock目录和src平级，新建mock/goods.js

```
let data = [
  {title:"web全栈"},
  {title:"java架构师"}
];

export default {
  'get /api/goods': function (req, res) {
    setTimeout(() => {
      res.json({ result: data })
    }, 250)
  },
}
```

effect处理异步：基于redux-saga，使用generator函数来控制异步流程

- 请求接口，models/goods.js

```
// 首先安装axios
import axios from 'axios';

// api
function getGoods(){
  return axios.get('/api/goods')
}

export default {
  state: [
    // {title:"web全栈"},
    // {title:"java架构师"},
    // {title:"百万年薪"}
  ],
  effects: { // 副作用操作，action-动作、参数等，saga-接口对象
    *getList(action, {call, put}){
      const res = yield call(getGoods)
      yield put({ type: 'initGoods', payload: res.data.result })
    }
  },
  reducers: {
    initGoods(state,{payload}){
      return payload
    }
  }
}
```

- 组件调用，goods.js

```
@connect(
  state => ({...}),
  {
    ...,
    getList: () => ({ // 映射getList动作
      type: 'goods/getList'
    })
  }
)
class Goods extends Component {
  componentDidMount(){ // 调用getList动作
    this.props.getList();
  }
}
```