Next.js。项目实战

刘军 liujun

目录 content

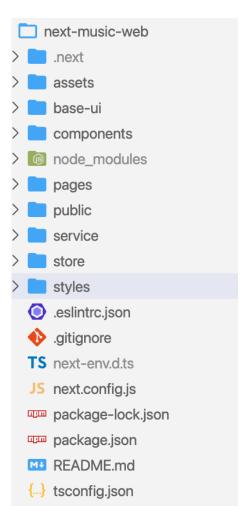


- **Next.js项目介绍**
- 2 Next.js项目实战
- **集成Redux Toolkit**
- 集成Ant Design 5
- 5 购买阿里云服务器
- 6 项目打包和部署

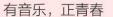


项目介绍

■ 网页云音乐-商城







戴上耳机,音乐旋律瞬间流进心里。只要心 灵还能与音乐共鸣,就不怕青春小鸟的远去。



亲爱的,晚安吧

晚安故事珍藏套装,每一只晚安熊都是我们的守护天使。





项目需安装的依赖

■样式

- □ npm i normalize.css --save
- □ npm i sass --save
- □ npm i classnames --save
- Redux And Toolkit
 - □ npm i next-redux-wrapper --save
 - □ npm i @reduxjs/toolkit react-redux --save
- Axios(最新的版本发现有bug)
 - □ npm i axios@1.1.3 --save
- AntDesign
 - □ npm i antd –save
 - □ npm i -D @types/antd

```
"dependencies": {
"@reduxjs/toolkit": "^1.9.0",
"@types/node": "18.11.9",
"@types/react": "18.0.25",
"@types/react-dom": "18.0.9",
"antd": "^4.24.3",
"axios": "^1.1.3",
"classnames": "^2.3.2",
"eslint": "8.27.0",
"eslint-config-next": "13.0.3",
"next": "13.0.3",
"next-redux-wrapper": "^8.0.0",
"normalize.css": "^8.0.1",
"react": "18.2.0",
"react-dom": "18.2.0",
"react-redux": "^8.0.5",
"sass": "^1.56.1".
"typescript": "4.9.3"
```



Next.js 集成 Redux

- 安装依赖库
 - □ npm i next-redux-wrapper --save
 - ✓ 可以避免在访问服务器端渲染页面时store的重置
 - ✓ 该库可以将服务器端redux存的数据,同步一份到客户端上
 - ✓ 该库提供了HYDRATE调度操作
 - ▶ 当用户访问动态路由或后端渲染的页面时,会执行Hydration来保持两端数据状态一致
 - ▶ 比如:每次当用户打开使用了getStaticProps或getServerSideProps函数生成的页面时,HYDRATE将执行调度操作。
 - □ npm i @reduxjs/toolkit react-redux --save



创建counter模块的reducer

- 我们先创建counter模块的reducer: 通过createSlice创建一个slice。
- createSlice主要包含如下几个参数:
 - □ name: 用来标记slice的名词
 - ✓ redux-devtool中会显示对应的名词;
 - □ initialState: 第一次初始化时的值;
 - □ reducers: 相当于之前的reducer函数
 - ✓ 对象类型,并且可以添加很多的函数;
 - ✓ 函数类似于 redux 原来reducer中的一个case语句;
 - □ extraReducer: 添加更多额外reducer处理other action
 - □ createSlice 返回值是一个对象
 - ✓ 对象包含所有的 actions 和 reducer;

```
import { HYDRATE } from "next-redux-wrapper";
const counterSlice = createSlice({
  name: "counter",
  initialState: {
    count: 100,
  reducers: {
    changeCountAction(state, action) {
     console.log("changeCountAction=>", action.type, action.payload);
     state.count += action.payload;
  extraReducers: (builder) => {
   builder.addCase(HYDRATE, (state, action: any) => {
     console.log(state.count); // 当前模块的state
     console.log(action.type); // action types: ___NEXT_REDUX_WRAPPER_HYDRATE__
     console.log("HYDRATE", action.payload); // payload 拿到整个rootState
     return {
       · . . state.
       ...action.payload.counter, // 更新模块数据
   · · · };
  ··});
// 异步的action
export const fetchHomeDataAction = createAsyncThunk(
```



store的创建

■ configureStore用于创建store对象,常见参数如下:

```
□ reducer,将slice中的reducer可以组成一个对象传入此处;
□ middleware: 可以使用参数, 传入其他的中间件(自行了解);
□ devTools: 是否配置devTools工具, 默认为true;
import { configureStore } from "@reduxjs/toolkit";
import { createWrapper } from "next-redux-wrapper";
import counterReducer from "./modules/counter";
const store = configureStore({

· reducer: -{
 counter: counterReducer,
 ⋅},
  devTools: true,
}):
const makeStore = () => store;
export const wrapper = createWrapper(makeStore);
```



store接入应用

■ 在app.js中将store接入应用:

```
□ Provider,内容提供者,给所有的子或孙子组件提供store对象;
□ store: 使用useWrappedStore函数导出的store对象;
import { wrapper } from "../store";
export default function App({ Component, ...rest }: AppProp
const { store, props } = wrapper.useWrappedStore(rest);
··return (
  <Provider store={store}>
     <<Component {...props.pageProps} />
  </Provider>
```



开始使用store

- 在函数式组件中可以使用 react-redux 提供的 Hooks API 连接、操作 store。
 - □ useSelector 允许你使用 selector 函数从 store 中获取数据 (root state)。
 - □ useDispatch 返回 redux store 的 dispatch 引用。你可以使用它来 dispatch actions。
 - □ useStore 返回一个 store 引用,和 Provider 组件引用完全一致。



Redux Toolkit异步Action操作

- 在之前的开发中,我们通过redux-thunk中间件让dispatch中可以进行异步操作。
- Redux Toolkit默认已经给我们继承了Thunk相关的功能: createAsyncThunk

```
export const getHomeMultidataAction = createAsyncThunk("home/multidata", async (payload, extraInfo) =>
{
    const res = await axios.get("http://123.207.32.32:8000/home/multidata")
    return res.data.data
})
```

■ 当createAsyncThunk创建出来的action被dispatch时,会存在三种状态:

□ pending: action被发出,但是还没有最终的结果;

□ fulfilled: 获取到最终的结果 (有返回值的结果);

□ rejected:执行过程中有错误或者抛出了异常;

■ 我们可以在createSlice的entraReducer中监听这些结果: 见右图

```
extraReducers: {
    [getHomeMultidataAction.pending](state, action) {
        console.log("getHomeMultidataAction pending", action)
},
    [getHomeMultidataAction.fulfilled](state, { payload }) {
        console.log("getHomeMultidataAction fuilfilled", payload)
        state.banners = payload.banner.list
        state.recommends = payload.recommend.list
},
    [getHomeMultidataAction.rejected](state, action) {
}
```



安装 Ant Design 5

- Ant Design 官网: https://ant.design/docs/react/use-in-typescript-cn
- Next.js 应用安装Ant Design5
 - □第一步:
 - ✓ npm i antd
 - ✓ npm i -D @types/antd
 - □ 第二步:
 - ✓ 统一样式风格
 - □ 第三步:
 - ✓ 就可以开始使用了
- 注意事项:
 - antd 默认支持基于 ES modules 的 tree shaking
 - □ 直接引入 import { Button } from 'antd' 就会有按需加载的效果

```
import React, { FC } from 'react';
import { Button } from 'antd';
import 'antd/dist/reset.css':
import './App.css';
const App: FC = () => (
  <div className="App">
    <Button type="primary">Button</Button>
  </div>
);
export default App;
```



项目打包和部署

- 项目打包
 - □ 执行 npm run build (整个项目是部署产物)
 - □ 执行 npm run start 本地预览效果
- 使用Node部署
 - □运行: npm run start
 - □ 指定端口: PORT=9090 npm run start
- 使用PM2部署 (推荐)
 - 项目根目录运行: pm2 start npm --name " music-mall" -- run start
 - ✓ pm2 start npm : 在当前目录执行npm
 - ✓ --name: 指定应用程序的名称
 - ✓ -- : 后面所有参数会传递给 npm 程序
 - □ OR: pm2 start "PORT=9090 npm run start" —-name music-mall

All option passed after — will be passed as argument to the app:

\$ pm2 start api.js -- arg1 arg2



PM2 常用命令

■ PM2 常用命令

```
# 命名进程
pm2 start app.js --name my-api
# 显示所有进程状态
pm2 list
# 停止指定的进程,
pm2 stop 0
# 停止所有进程
pm2 stop all
# 重启所有进程
pm2 restart all
# 重启指定的进程( id )
pm2 restart 0
# 杀死指定的进程
pm2 delete 0
# 杀死全部进程
pm2 delete all
#后台运行pm2, 启动4个app.js, 实现负载均衡
pm2 start app.js -i 4
```

```
module.exports = {
 apps: [
    name: "music-mall",
    exec_mode: "fork", // fork 模式, 创建一个 process_child 子进程
    increment_var: "PORT", // 端口自增长
·····instances: 3, // 三个实例,分别运行在: 8888,8889,8890端口
 script: "npm run start",
 ···env: {
 PORT: 8888,
 . . . } ,
};
module.exports = {
 apps: [
name: "music-next",
····· cwd: ·"./",
script: "node_modules/next/dist/bin/next",
· args: "start",
instances: "max",
exec_mode: "cluster",
 · · · },
```