1.使用babel-node在后端支持jsx环境

Node并不完全支持es6和jsx，所以我们可以利用babel

（1）安装babel-cli 和 cross-env

npm i babel-cli --save 这个是babel命令行工具，我们要用到它里面的babel-node，让node支持es6,jsx

npm i cross-env --save 跨平台地设置及使用环境变量

（2）配置全局babel参数

在根目录下创建.babelrc文件，把package.json文件的babel配置复制过去，如下

{  
 **"presets"**: [  
 **"react-app"** ],  
 **"plugins"**: [  
 [  
 **"import"**,  
 {  
 **"libraryName"**: **"antd-mobile"**,  
 **"style"**: **"css"** }  
 ]  
 ]  
}

（3）package.json文件下设置脚本命令行

**"scripts"**: {  
 **"server"**: **"cross-env NODE\_ENV=test nodemon --exec babel-node server/server.js"**}

设置node环境为test，默认为node 修改为nodemon --exec babel-node

2.后端渲染首屏

（1）npm  i  css-modules-require-hook 处理css请求

a.后端引入import csshook from 'css-modules-require-hook/preset'

必须要在渲染的内容之前，不然不会生效

b.在项目根目录下新建cmrh.conf.js文件

module.exports = {

// Same scope name as in webpack build

generateScopedName: '[name]\_\_[local]\_\_\_[hash:base64:5]',

}

（2）npm i --save asset-require-hook 处理图片等其他资源

a.后端引入

import assethook from 'asset-require-hook';

// 配置参数

assethook({

extensions: ['jpg','png']

});

（3）后端引入

import React from 'react';

// renderToString把react组件渲染成浏览器可以识别的div，这样可以加快渲染速度，因为返回react组件给前端，前端又调用react去识别他，然后在进行渲染，这样速度会变慢

import {renderToString} from 'react-dom/server';

// function App(){

// return (

// <h1>server render</h1>

// )

// }

// console.log(renderToString(<App></App>));

（4）渲染

// 利用服务器渲染首屏，有利于解决SEO  
import App from '../src/app.js';  
import {**Provider**} from 'react-redux';  
// 服务端用StaticRouter代替BrowserRouter  
import {**StaticRouter**} from 'react-router-dom';  
import {applyMiddleware, compose, createStore} from "redux";  
import reducers from "../src/reducer.js";  
import thunk from "redux-thunk";  
// 引入打包生成的asset-manifest文件，里面有打包后的文件路径  
import staticPath from '../build/asset-manifest.json';

app.use((req,res,next)=>{  
 // 设置白名单，以/user/开头的说明是发送网络请求，以/static/开头的说明是请求静态资源文件，这里不可能是请求网络资源，因为请求网络资源在上面  
 if(req.**url**.startsWith('/user/')||req.**url**.startsWith('/static/')){  
 return next()  
 }  
 // 前端路由没有处理其他的路由情况，所以要在这里处理  
 // 返回主页  
 // return res.sendFile(path.resolve('build/index.html'));  
  
 // 创建store  
 const store = createStore(reducers,compose(  
 // 处理异步，需要用到thunk中间件  
 applyMiddleware(thunk),  
 ));  
  
 // react-router4官方推荐写法  
 var context={}  
 var markup=renderToString((  
 <Provider store={store}>  
 <StaticRouter  
 location={req.**url**}  
 context={context} //路由跳转相关  
 >  
 <App></App>  
 </StaticRouter>  
 </Provider>  
 ));  
  
 var obj={  
 '/msg':'消息列表',  
 '/boss':'boss列表'  
 }  
 var pagehtml=`<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  
 <meta name="theme-color" content="#000000">  
 <meta name="keywords" content="react16 redux react-router4">  
 <!--动态SEO-->  
 <meta name="description" content="${obj[req.**url**]}">  
 <link rel="manifest" href="%PUBLIC\_URL%/manifest.json">  
 <link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico">  
 <title>React App</title>  
 <link rel="stylesheet" href="/${staticPath['main.css']}">  
 </head>  
 <body>  
 <noscript>  
 You need to enable JavaScript to run this app.  
 </noscript>  
 <div id="root">${markup}</div>  
 <script src="${staticPath['main.js']}"></script>  
 </body>  
</html>`  
  
  
 res.send(pagehtml);  
})