link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=244 sentences=675, words=3733

1. 什么是同构

- 同构的项目支持客户端渲染和服务器端渲染
- 客户端渲染缺点
    - 首屏速度加载慢
    - 不支持SEO和搜索引擎优化
    - 首页需要通过请求初始化数据

2.Next.js

- Next.js英文文档 (https://nextjs.org/docs),Next.js中文文档 (https://nextjs.frontendx.cn/docs/) 是一个轻量级的 React 服务端渲染应用框架
- 默认支持服务端渲染
- 自动根据页面进行代码分割
- 基于页面的客户端路由方案
- 基于 Webpack 的开发环境，支持热模块替换
- 可以跟 Koa或者其它 Node.js服务器进行集成
- 支持 Babel 和 Webpack 的配置项定制
- 静态文件服务 public

3.项目初始化

3.1 创建项目

```
mkdir zhufengnextjs
cd zhufengnextjs
npm init -y
yarn add --dev react  react-dom  next axios redux react-redux express  body-parser  cors express-session connect-mongo mongoose koa koa-router
mkdir pages
```

3.2 添加脚本

package.json

```
{
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start"
  }
}
```

.gitignore (https://github.com/zeit/next.js/blob/canary/.gitignore)

3.3 访问服务

- 以 ./pages作为服务端的渲染和索引的根目录
-
    - pages 是 next.js 中非常重要的一个目录，其中每一个 js 文件就代表一个页面，但是有两个例外， _app.js 和 _document.js
- next.js会将pages下的js文件根据其路径名和文件名自动生成对应的路由
- pages组件代码自动分割
- next.js项目运行之后会自动生成 .next目录

3.3.1 index.js

```
export default function () {
    return (
        <div>Homediv>
    )
}
```

3.3.2 访问

```
curl http:
```

4.跑通路由和样式

4.1 知识点

- 绑定 styled-jsx 来生成独立作用域的 CSS
- 如何支持本地和全局 css
- 路由的使用和两种跳转路径的方法

4.2 pages_app.js

- App组件是每个页面的根组件，页面切换时 App 不会销毁，但是里面的页面组件会销毁，因此可以利用这个来设置全局属性和样式
- 全局 css 只能写在这里，否则会报错
    - 当页面变化时保持页面布局
    - 当路由变化时保持页面状态
    - 注入额外数据到页面里

pages_app.js

```
import App, { Container } from 'next/app';
import Link from 'next/link';
import _appStyle from './_app.module.css';
import '../styles/global.css';
class LayoutApp extends App{
    render() {
        let { Component } = this.props;
        return (
            <div>
                <style jsx>
                    {
                    `li{
                        display:inline-block;
                        margin-left:10px;
                        line-height:31px;
                      }`
                    }
                style>
                <div>
                    <header>
                        <img src="/images/jglogo.png" className={_appStyle.logo}/>
                        <ul>
                            <li><Link href="/"><a>首页a>Link>li>
                            <li><Link href="/user" ><a>用户管理a>Link>li>
                            <li><Link href="/profile"><a>个人中心a>Link>li>
                        ul>
                    header>
                    <Component />
                    <footer style={{ textAlign: 'center' }} >@copyright 珠峰架构footer>
                div>
            div>
        )
    }
}
export default LayoutApp;
```

4.3 pages_app.module.css

pages_app.module.css

```
.logo{
    width: 120px;
    height: 31px;
    float: left;
}
```

4.4 global.css

styles\global.css

```
html,
body {
  padding: 0;
  margin: 0;
}
```

4.5 pages\index.js

pages\index.js

```
export default function () {
    return (
        <div>
            <p>Homep>
        div>
    )
}
```

4.6 pages\user.js

pages\user.js

```
import Link from 'next/link'
export default function () {
    return (
        <div>
            <p>Userp>
            <Link href="/">首页Link>
        div>
    )
}
```

4.7 pages\profile.js

pages\profile.js

```
import router from 'next/router'
export default function () {
    return (
        <div>
            <p>Userp>
            <button onClick={() => router.back()}>返回button>
        div>
    )
}
```

5.二级路由

5.1 知识点

- 支持二级路由
- 实现二级布局组件
- 路由跳转传递参数
- 页面组件通过 getInitialProps获取数据

5.2 执行顺序

5.2.1 后台顺序

- LayoutApp getInitialProps
- UseList getInitialProps
- LayoutApp constructor
- UseList constructor **5.2.2 前台顺序#**
- 初次渲染
    - LayoutApp constructor
    - UseList constructor
- 路由切换
    - LayoutApp getInitialProps
    - UseList getInitialProps
    - UseList constructor

** 5.2.3 pages_app.js #**

pages_app.js

```
import App from 'next/app';
import Link from 'next/link';
import _appStyle from './_app.module.css';
import '../styles/global.css';
class LayoutApp extends App{
+   static async getInitialProps({ Component, ctx }) {
+       let pageProps = {};
+       if (Component.getInitialProps)
+           pageProps = await Component.getInitialProps(ctx);
+       return { pageProps };
+   }
    render() {
+       let { Component,pageProps } = this.props;
        return (

                    {
                    `li{
                        display:inline-block;
                        margin-left:10px;
                        line-height:31px;
                      }`
                    }

                            首页
+                           用户管理
                            个人中心

+
                    @copyright 珠峰架构

        )
    }
}
export default LayoutApp;
```

** 5.2.4 user\index.js #**

pages\user\index.js

```
import Link from 'next/link';

function UserLayout(props) {
    return (
        <div>
            <div>
                 <ul>
                    <li><Link href="/user/list"><a>用户列表a>Link>li>
                    <li><Link href="/user/add"><a>添加用户a>Link>li>
                ul>
                <div>
                  {props.children}
                div>
            div>
        div>
    )
}
export default UserLayout;
```

** 5.2.5 user\list.js #**

pages\user\list.js

```
import Link from 'next/link';
import UserLayout from './';
function UseList(props) {
    return (
        <UserLayout>
            <ul>
                {
                    props.list.map((user)=>(
                        <li key={user.id}>
                            <Link href={`/user/detail/${user.id}`}>{user.name}Link>
                        li>
                    ))
                }
            ul>
        UserLayout>
    )
}
UseList.getInitialProps = async (ctx) => {
    let list = [{ id: 1, name: '张三'}, { id: 2, name: '李四'}];
    return { list };
}
export default UseList;
```

** 5.2.6 user\add.js #**

pages\user\add.js

```
import Link from 'next/link';
import UserLayout from './';
import React from 'react';
function UserAdd(props) {
    let nameRef = React.useRef();
    let passwordRef = React.useRef();
    let handleSubmit  = (event)=>{
        event.preventDefault();
        let user = {name:nameRef.current.value,password:passwordRef.current.value};
        console.log('添加',user);
    }
    return (
        <UserLayout>
            <form onSubmit={handleSubmit}>
                用户名:<input  ref={nameRef}/>
                密码:<input  ref={passwordRef}/>
                <button type="submit">添加button>
            form>
        UserLayout>
    )
}

export default UserAdd;
```

** 5.2.7 user\detail[id].js #**

pages\user\detail[id].js

```
import React from 'react';
import UserLayout from '../';

function UserDetail(props) {
  return (
    <UserLayout>
      <p>ID:{props.user.id}p>
    UserLayout>
  )
}
UserDetail.getInitialProps = async (ctx) => {
  return { user:{id:ctx.query.id}};
}
export default UserDetail;
```

6.调用接口

- 当服务渲染时, getInitialProps将会把数据序列化，就像 JSON.stringify
- 所以确保 getInitialProps返回的是一个普通 JS 对象，而不是Date, Map 或 Set类型
- 当页面初始化加载时, getInitialProps只会加载在服务端。只有当路由跳转（Link组件跳转或 API 方法跳转）时，客户端才会执行 getInitialProps
- getInitialProps将不能使用在子组件中。只能使用在 pages页面中

6.1 utils\axios.js

utils\axios.js

```
import axios from 'axios';
axios.defaults.withCredentials = true
const instance = axios.create({
    baseURL: 'http://localhost:4000'
})
export default instance;
```

6.2 pages\user\add.js

pages\user\add.js

```
import Link from 'next/link';
import UserLayout from './';
import React from 'react';
+import axios from '../../utils/axios';
+import router from 'next/router'
function UserAdd() {
    let nameRef = React.useRef();
    let passwordRef = React.useRef();
    let handleSubmit  = async (event)=>{
        event.preventDefault();
        let user = {name:nameRef.current.value,password:passwordRef.current.value};
+        let response = await axios.post('/api/register', user).then(res=>res.data);
+        if (response.success) {
+            router.push('/user/list');
+        } else {
+            alert('添加用户失败');
+        }
    }
    return (

                用户名:
                密码:
                添加

    )
}

export default UserAdd;
```

6.3 user\list.js

pages\user\list.js

```
import Link from 'next/link';
import UserLayout from './';
+import axios from '../../utils/axios';
function UseList(props) {
    return (

                {
                    props.list.map((user)=>(

                            {user.name}

                    ))
                }

    )
}
UseList.getInitialProps = async (ctx) => {
+   let response = await axios({ url: '/api/users', method: 'GET' }).then(res=>res.data);
+   return { list:response.data };
}
export default UseList;
```

6.4 [id].js

pages\user\detail[id].js

```
import React from 'react';
import UserLayout from '../';
import axios from '../../../utils/axios';
function UserDetail(props) {
  let {user} = props;
  return (
    <UserLayout>
      <p>ID:{user.id}p>
      <p>ID:{user.name}p>
    UserLayout>
  )
}
UserDetail.getInitialProps = async (ctx) => {
  let response = await axios({ url: `/api/users/${ctx.query.id}`, method: 'GET' }).then(res=>res.data);
  return { user: response.data };
}
export default UserDetail;
```

7.懒加载

7.1 知识点

- 组件懒加载
- 模块懒加载

7.2 pages\user\detail[id].js

pages\user\detail[id].js

```
import React from 'react';
import UserLayout from '../';
import axios from '../../../utils/axios';
import dynamic from 'next/dynamic';
const UserInfo = dynamic(import('../../../components/UserInfo'))
function UserDetail(props) {
  let {user} = props;
  let [show, setShow] = React.useState(false);
  return (
    <UserLayout>
      <p>ID:{user.id}p>
      <button onClick={() => setShow(!show)}>显示/隐藏button>
      {
        show && <UserInfo user={user} />
      }
    UserLayout>
  )
}
UserDetail.getInitialProps = async (ctx) => {
  let response = await axios({ url: `/api/users/${ctx.query.id}`, method: 'GET' }).then(res=>res.data);
  return { user: response.data };
}
export default UserDetail;
```

7.3 components\UserInfo.js

components\UserInfo.js

```
import React from 'react';
function UserInfo(props) {
    let {user} = props;
    let [created, setCreated] = React.useState(props.user.created);
    async function changeFormat() {
        let moment = await import('moment');
        setCreated(moment.default(user.created).fromNow());
    }
    return (
        <div>
            <p>用户名:{user.name}p>
            <p>创建时间:{created}<button onClick={changeFormat}>切换为相对时间button>p>
        div>
    )
}
export default UserInfo;
```

8.用户登录

8.1 pages_app.js

pages_app.js

```
        首页
        用户管理
        个人中心
+       登录
```

8.2 pages\login.js

```
import React from 'react';
import axios from '../utils/axios';
import router from 'next/router';
import {connect} from 'react-redux';
import * as types from '../store/action-types';
function Login() {
    let nameRef = React.useRef();
    let passwordRef = React.useRef();
    let handleSubmit  = async (event)=>{
        event.preventDefault();
        let user = {name:nameRef.current.value,password:passwordRef.current.value};
        let response = await axios.post('/api/login', user).then(res=>res.data);
        if (response.success) {
             props.dispatch({type:types.SET_USER_INFO,payload:response.data});
            router.push('/');
        } else {
            alert('登录失败');
        }
    }
    return (
        <form onSubmit={handleSubmit}>
                用户名:<input  ref={nameRef}/>
                密码:<input  ref={passwordRef}/>
                <button type="submit">登录button>
        form>
    )
}

export default connect(state=>state)(Login);
```

9.集成redux

- 初次渲染的执行顺序是
    - 后台 LayoutApp.getInitialProps LayoutApp.constructor
    - 前台 LayoutApp.getInitialProps
- 切换路由时的执行顺序
    - 前台 LayoutApp.getInitialProps

9.1 安装依赖

```
yarn add redux react-redux
```

9.2 pages_app.js

pages_app.js

```
import App from 'next/app';
import Link from 'next/link';
import _appStyle from './_app.module.css';
import '../styles/global.css';
+import { Provider } from 'react-redux';
+import axios from '../utils/axios';
+import createStore from '../store';
+import * as types from '../store/action-types';
+function getStore(initialState) {
+    if (typeof window == 'undefined') {
+        return createStore(initialState);//如果是服务器端,每次都返回新的仓库
+    } else {
+        if (!window._REDUX_STORE_) {
+            window._REDUX_STORE_ = createStore(initialState);
+        }
+        return window._REDUX_STORE_;
+    }
+}
+class LayoutApp extends App{
+    constructor(props){
+        super(props);
+        this.store = getStore(props.initialState);//3.后台用新状重新创建新仓库
+        //4.初始状态序列化后发给了客户端，在客户端重新创建新的仓库
+    }
+   static async getInitialProps({ Component, ctx }) {
+       let store = getStore();//1.后台创建新仓库  5.每次切换路由都会执行此方法获取老仓库
+       if (typeof window == 'undefined') {//2.后台获取用户信息
+           let options = { url: '/api/currentUser' };
+           if (ctx.req&&ctx.req.headers.cookie) {
+               options.headers = options.headers || {};
+               options.headers.cookie = ctx.req.headers.cookie;
+           }
+           let response = await axios(options).then(res=>res.data);
+           if (response.success) {
+               store.dispatch({ type: types.SET_USER_INFO, payload: response.data });
+           }
+       }
+      let pageProps = {};
+      if (Component.getInitialProps)
+         pageProps = await Component.getInitialProps(ctx);
+      let props =  { pageProps};
+      if (typeof window == 'undefined') {//后台获取用赋值状态
+       props.initialState=store.getState();
+      }
+      return props;
+   }
    render() {
        let { Component,pageProps} = this.props;
+       let state = this.store.getState();
        return (
+

                    {
                    `li{
                        display:inline-block;
                        margin-left:10px;
                        line-height:31px;
                      }`
                    }

                        首页
                        用户管理
                        个人中心
+
+                            {
+                             state.currentUser ? {state.currentUser.name}:登录
+                            }
+


                       {

                       }

                  @copyright 珠峰架构

+
       )
    }
}
export default LayoutApp;
```

9.3 store\action-types.js

store\action-types.js

```
export const SET_USER_INFO = 'SET_USER_INFO';
```

9.4 store\reducer.js

store\reducer.js

```
import * as types from './action-types';
let initState = {
    currentUser: null
}
const reducer = (state = initState, action) => {
    switch (action.type) {
        case types.SET_USER_INFO:
            return { currentUser: action.payload }
        default:
            return state;
    }
}
export default reducer;
```

9.5 store\index.js

store\index.js

```
import { createStore } from 'redux';
import reducer from './reducer';

export default function (initialState) {
    return createStore(reducer, initialState);
}
```

10.loading效果

- 路由事件 (https://nextjs.frontendx.cn/docs/#%E8%B7%AF%E7%94%B1)

事件 触发时机 routeChangeStart(url) 路由开始切换时触发 routeChangeComplete(url) 完成路由切换时触发 routeChangeError(err, url) 路由切换报错时触发 beforeHistoryChange(url) 浏览器 history 模式开始切换时触发 hashChangeStart(url) 开始切换 hash 值但是没有切换页面路由时触发 hashChangeComplete(url) 完成切换 hash 值但是没有切换页面路由时触发

11.1 pages_app.js

```
import App from 'next/app';
import Link from 'next/link';
import _appStyle from './_app.module.css';
import '../styles/global.css';
import { Provider } from 'react-redux';
import axios from '../utils/axios';
import createStore from '../store';
import * as types from '../store/action-types';
+import router from 'next/router';
function getStore(initialState) {
    if (typeof window == 'undefined') {
        return createStore(initialState);//如果是服务器端,每次都返回新的仓库
    } else {
        if (!window._REDUX_STORE_) {
            window._REDUX_STORE_ = createStore(initialState);
        }
        return window._REDUX_STORE_;
    }
}
class LayoutApp extends App{
+    state = { loading: false }
    constructor(props){
        super(props);
        this.store = getStore(props.initialState);//3.后台用新状重新创建新仓库
        //4.初始状态序列化后发给了客户端，在客户端重新创建新的仓库
    }
    static async getInitialProps({ Component, ctx }) {
        let store = getStore();//1.后台创建新仓库  5.每次切换路由都会执行此方法获取老仓库
        if (typeof window == 'undefined') {//2.后台获取用户信息
            let options = { url: '/api/currentUser' };
            if (ctx.req&&ctx.req.headers.cookie) {
                options.headers = options.headers || {};
                options.headers.cookie = ctx.req.headers.cookie;
            }
            let response = await axios(options).then(res=>res.data);
            if (response.success) {
                store.dispatch({ type: types.SET_USER_INFO, payload: response.data });
            }
        }
       let pageProps = {};
       if (Component.getInitialProps)
          pageProps = await Component.getInitialProps(ctx);
       let props =  { pageProps};
       if (typeof window == 'undefined') {//后台获取用赋值状态
        props.initialState=store.getState();
       }
       return props;
    }
+   componentDidMount() {
+       this.routeChangeStart = (url) => {
+           this.setState({ loading: true });
+       };
+       router.events.on('routeChangeStart', this.routeChangeStart);
+       this.routeChangeComplete = (url) => {
+           this.setState({ loading: false });
+       };
+       router.events.on('routeChangeComplete', this.routeChangeComplete);
+   }
+   componentWillUnmount() {
+       router.events.off('routeChangeStart', this.routeChangeStart)
+       router.events.off('routeChangeStart', this.routeChangeComplete)
+   }
    render() {
        let { Component,pageProps} = this.props;
        let state = this.store.getState();
        return (

                    {
                    `li{
                        display:inline-block;
                        margin-left:10px;
                        line-height:31px;
                      }`
                    }

                            首页
                            用户管理
                            个人中心

                                {
                                 state.currentUser ? {state.currentUser.name}:登录
                                }

                        {

                        }

+                   {
+                       this.state.loading ? 切换中...... :
+                   }
                   @copyright 珠峰架构

        )
    }
}
export default LayoutApp;
```

11.受保护路由

11.1 pages\profile.js

pages\profile.js

```
+import router from 'next/router';
+import { connect } from 'react-redux';
+import axios from '../utils/axios';
+function Profile(props) {
+    let { currentUser } = props;
+    return (
+
+            当前登录用户:{currentUser.name}
+             router.back()}>返回
+
+    )
+}
+Profile.getInitialProps = async function (ctx) {
+    let options = { url: '/api/currentUser' };
+    if (ctx.req && ctx.req.headers.cookie) {
+        options.headers = options.headers || {};
+        options.headers.cookie = ctx.req.headers.cookie;
+    }
+    let response = await axios(options).then(res=>res.data);
+    if (response.success) {
+        return {currentUser:response.data};
+    } else {
+        if (ctx.req) {
+            ctx.res.writeHead(303, { Location: '/login' })
+            ctx.res.end()
+        } else {
+            router.push('/login');
+        }
+        return {};
+    }
+}
+
+const WrappedProfile = connect(
+    state => state
+)(Profile);
+export default WrappedProfile;
```

12.自定义Document

12.1 pages_document.js

pages_document.js

```
import Document, { Html, Head, Main, NextScript } from 'next/document';
class CustomDocument extends Document {
    static async getInitialProps(ctx) {
        const props = await Document.getInitialProps(ctx);
        return { ...props };
    }
    render() {
        return (
            <Html>
                <Head>
                    <style>
                        {
                            `
                            *{
                                padding:0;
                                margin:0;
                            }
                            `
                        }
                    style>
                Head>
                <body>
                    <Main />
                    <NextScript />
                body>
            Html>
        )
    }
}
export default CustomDocument;
```

13.2 pages\index.js

pages\index.js

```
import Head from 'next/head'
export default function (props) {
    return (
        <div>
              <Head>
                <title>首页title>
                <meta name="description" content="这是首页" />
             Head>
            <p>Homep>
        div>
    )
}
```

13. 新版初始化函数

- data-fetching (https://nextjs.org/docs/basic-features/data-fetching)
    - getStaticProps (Static Generation): Fetch data at build time
    - getStaticPaths (Static Generation): Specify dynamic routes to pre-render based on data
    - getServerSideProps (Server-side Rendering): Fetch data on each request

13.1 pages\user.js

pages\user.js

```
import Link from 'next/link';
export default function ({users}) {
    return (
        <ul>
              {
                  users.map(user=>(
                      <li key={user.id}><Link href={`/user/detail/${user.id}`}>{user.name}Link>li>
                  ))
              }
        ul>
    )
}

export async function getServerSideProps() {
    const res = await fetch('http://localhost:4000/api/users').then(res=>res.json());
    return {
      props: {
        users:res.data
      },
    }
  }
```

13.2 pages\user\list.js

pages\user\list.js

```
import Link from 'next/link';
export default function ({users}) {
    return (
        <ul>
              {
                  users.map(user=>(
                      <li key={user.id}><Link href={`/user/detail/${user.id}`}>{user.name}Link>li>
                  ))
              }
        ul>
    )
}

export async function getStaticProps() {
    const res = await fetch('http://localhost:4000/api/users').then(res=>res.json());
    return {
      props: {
        users:res.data
      },
    }
}
```

13.3 pages\user\detail[id].js

pages\user\detail[id].js

```
export default function ({user}) {
    return (
       <div>
           <p>ID:{user.id}p>
           <p>name:{user.name}p>
       div>
    )
}
export async function getStaticPaths() {
    const res = await fetch('http://localhost:4000/api/users').then(res=>res.json());
    const users = res.data;
    const paths = users.map(user=>`/user/detail/${user.id}`);
    return {paths,fallback:false}
}
export async function getStaticProps({params}) {
    console.log('params',params,new Date().toTimeString());
    const res = await fetch(`http://localhost:4000/api/users/${params.id}`).then(res=>res.json());
    return {
      props: {
        user:res.data
      }
    }
}
```

14. 布署

14.1 直接布署

```
npm run build
npm run start
```

14.2 集成express布署

```
npm run build
```

** 14.2.1 start.js #**

start.js

```
const next = require('next');
const app = next({ dev:false });
const handler = app.getRequestHandler();
app.prepare().then(() => {
    let express = require("express");
    let bodyParser = require("body-parser");
    let {UserModel} = require('./model');
    let session = require("express-session");
    let config = require('./config');
    let MongoStore = require('connect-mongo')(session);
    let app = express();
    app.use(bodyParser.urlencoded({ extended: false }));
    app.use(bodyParser.json());
    app.use(
        session({
            secret: config.secret,
            resave: false,
            saveUninitialized: true,
            store: new MongoStore({
                url: config.dbUrl,
                mongoOptions: {
                    useNewUrlParser: true,
                    useUnifiedTopology: true
                }
            })
        })
    );
    app.get('/api/users', async (req, res) => {
        let users = await UserModel.find();
        users = users.map(user=>user.toJSON());
        res.send({ success:true, data: users });
    });
    app.get('/api/users/:id', async (req, res) => {
        let user = await UserModel.findById(req.params.id);
        res.send({ success:true, data: user.toJSON() });
    });
    app.post('/api/register', async (req, res) => {
        let user = req.body;
        user = await UserModel.create(user);
        res.send({ success:true, data: user.toJSON() });
    });
    app.post('/api/login', async (req, res) => {
        let user = req.body;
        let dbUser = await UserModel.findOne(user);
        if (dbUser) {
            req.session.currentUser = dbUser.toJSON();
            res.send({ success:true, data: dbUser.toJSON() });
        } else {
            res.send({ success:false, error: '登录失败' });
        }
    });

    app.get('/api/currentUser', async (req, res) => {
        let currentUser = req.session.currentUser;
        if (currentUser) {
            res.send({ success:true, data: currentUser });
        } else {
            res.send({ success:false, error: '当前用户未登录' });
        }
    });
    app.get('*', async (req, res) => {
        await handler(req, res);
    })
    app.listen(3000, () => {
        console.log('服务器在3000端口启动!');
    });
});
```

** 14.2.2 config.js#**

config.js

```
module.exports = {
    secret: 'zhufeng',
    dbUrl: "mongodb://127.0.0.1/zhufengnext2"
}
```

** 14.2.3 model.js#**

model.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const config = require('./config');
const conn = mongoose.createConnection(config.dbUrl, { useNewUrlParser: true, useUnifiedTopology: true });
const UserModel = conn.model('User', new Schema({
    name: { type: String },
    password: { type: String },
},{ timestamps: true ,    toJSON: {
    transform: function (_doc, result) {
        result.id = result._id;
        delete result._id;
        return result;
    }
}}));
module.exports = {
    UserModel
}
```

13.服务端

13.1 安装

```
yarn add express  body-parser  cors express-session connect-mongo mongoose
```

13.2 接口文档

** 13.2.1 注册 #**

13.2.1.1 接口地址

```
POST /api/register
```

13.2.1.2 参数

** JSON请求体参数 #**

变量名 类型 说明 name 字符串 用户名 password 字符串 密码

13.2.1.3 返回值

13.2.1.4 返回结果

变量名 类型 说明 success 布尔值 是否成功 data 对象 注册成功之后的对象

示例

```
{
    "success": true,
    "data": {
            "id": 1,
            "name": "张三",
            "password": "123456"
    }
}
```

** 13.2.2 登录 #**

13.2.2.1 接口地址

```
POST /api/login
```

13.2.2.2 参数

** JSON请求体参数 #**

变量名 类型 说明 name 字符串 用户名 password 字符串 密码

13.2.2.3 返回值

13.2.2.4 返回结果

变量名 类型 说明 success 布尔值 是否成功 data 对象 登录成功之后的对象

示例

```
{
    "success": true,
    "data": {
            "id": 1,
            "name": "张三",
            "password": "123456"
    }
}
```

** 13.2.3 查看当前登录的用户信息 #**

13.2.3.1 接口地址

```
GET /api/currentUser
```

13.2.3.2 参数

13.2.3.3 返回值

13.2.3.4 返回结果

变量名 类型 说明 success 布尔值 是否成功 data 对象 当前用户对象

示例

```
{
    "success": true,
    "data": {
            "id": 1,
            "name": "张三",
            "password": "123456"
    }
}
```

** 13.2.4 查看用户列表 #**

13.2.4.1 接口地址

```
GET /api/users
```

13.2.4.2 参数

13.2.4.3 返回值

13.2.4.4 返回结果

变量名 类型 说明 success 布尔值 是否成功 data 对象数组 用户对象数组

示例

```
{
    "success": true,
    "data": [
        {
            "id": 1,
            "name": "张三",
            "password": "123456"
        },
         {
            "id": 2,
            "name": "李四",
            "password": "123456"
        }
    ]
}
```

** 13.2.5 查看某个用户信息 #**

13.2.5.1 接口地址

```
GET /api/users/:id
```

13.2.5.2 参数

13.2.5.3 返回值

13.2.5.4 返回结果

变量名 类型 说明 success 布尔值 是否成功 data 对象 用户对象

示例

```
{
    "success": true,
    "data": {
            "id": 1,
            "name": "张三",
            "password": "123456"
    }
}
```

13.1 api\index.js

```
let express = require("express");
let bodyParser = require("body-parser");
let cors = require("cors");
let {UserModel} = require('./model');
let session = require("express-session");
let config = require('./config');
let MongoStore = require('connect-mongo')(session);
let app = express();
app.use(
    cors({
        origin: ['http://localhost:3000'],
        credentials: true,
        allowedHeaders: "Content-Type,Authorization",
        methods: "GET,HEAD,PUT,PATCH,POST,DELETE,OPTIONS"
    })
);
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use(
    session({
        secret: config.secret,
        resave: false,
        saveUninitialized: true,
        store: new MongoStore({
            url: config.dbUrl,
            mongoOptions: {
                useNewUrlParser: true,
                useUnifiedTopology: true
            }
        })
    })
);
app.get('/api/users', async (req, res) => {
    let users = await UserModel.find();
    users = users.map(user=>user.toJSON());
    res.send({ code: 0, data: users });
});
app.get('/api/users/:id', async (req, res) => {
    let user = await UserModel.findById(req.params.id);
    res.send({ code: 0, data: user.toJSON() });
});
app.post('/api/register', async (req, res) => {
    let user = req.body;
    user = await UserModel.create(user);
    res.send({ code: 0, data: user.toJSON() });
});
app.post('/api/login', async (req, res) => {
    let user = req.body;
    let dbUser = await UserModel.findOne(user);
    if (dbUser) {
        req.session.currentUser = dbUser.toJSON();
        res.send({ code: 0, data: dbUser.toJSON() });
    } else {
        res.send({ code: 1, error: '登录失败' });
    }
});

app.get('/api/currentUser', async (req, res) => {
    let currentUser = req.session.currentUser;
    if (currentUser) {
        res.send({ code: 0, data: currentUser });
    } else {
        res.send({ code: 1, error: '当前用户未登录' });
    }
});
app.listen(4000, () => {
    console.log('服务器在4000端口启动!');
});
```

13.2 api\config.js

api\config.js

```
module.exports = {
    secret: 'zhufeng',
    dbUrl: "mongodb://127.0.0.1/zhufengnext2"
}
```

13.3 server\model.js

server\model.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const config = require('./config');
const conn = mongoose.createConnection(config.dbUrl, { useNewUrlParser: true, useUnifiedTopology: true });
const UserModel = conn.model('User', new Schema({
    name: { type: String },
    password: { type: String }
},{ timestamps: true ,    toJSON: {
    transform: function (_doc, result) {
        result.id = result._id;
        delete result._id;
        return result;
    }
}}));

module.exports = {
    UserModel
}
```