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=78 sentences=126, words=745

## 1.UmiJS #

- UmiJS (https://umijs.org/zh/guide/) 是一个类 Next.JS 的 react 开发框架。
- 他基于一个约定，即 pages 目录下的文件即路由，而文件则导出 react 组件
- 然后打通从源码到产物的每个阶段，并配以完善的插件体系，让我们能把 umi 的产物部署到各种场景里。

## 2.安装 #

- umi源码 (https://github.com/umijs/umi)
- create-umi (https://github.com/umijs/create-umi)
- umi-plugin-react文档 (https://umijs.org/zh/plugin/umi-plugin-react.html)
- umi-plugin-react源码 (https://github.com/umijs/umi/tree/master/packages/umi-plugin-react)
- umi-plugin-dva (https://github.com/umijs/umi/tree/master/packages/umi-plugin-dva)
- dva-immer (https://github.com/dvajs/dva/blob/master/packages/dva-immer/src/index.js)
- immer (https://immerjs.github.io/immer/docs/introduction)
- umi-blocks (https://github.com/umijs/umi-blocks)
- pro-blocks (https://github.com/ant-design/pro-blocks)

```
$ cnpm install -g umi
$ yarn global add umi
$ umi -v
umi@3.2.15
$ yarn global bin
$ npm root -g
```

### 2.1 目录约定 #

```
.
├── dist/
├── mock/
├── config/
    ├── config.js
└── src/
    ├── layouts/index.js
    ├── pages/
        ├── .umi/
        ├── .umi-production/
        ├── document.ejs
        ├── 404.js
        ├── page1.js
        ├── page1.test.js
        └── page2.js
    ├── global.css
    ├── global.js
├── .umirc.js
├── .env
└── package.json
```

## 3. 新建项目 #

### 3.1 新建项目目录 #

```
mkdir zhufeng-umi
cd zhufeng-umi
cnpm init -y
```

### 3.2 新建pages目录 #

```
mkdir src/pages
```

### 3.3 新建页面 #

#### 3.3.1 Home组件 #

```
umi g page index
```

pages\index.js

```
import React, { Component } from 'react'
import { Link } from 'umi';
import styles from './index.css';
export default class componentName extends Component {
  render() {
    return (
      <div>
        <h1 className={styles.title}>首页h1>
        <Link to="/profile">个人中心Link>
      div>
    )
  }
}
```

#### 3.3.2 用户管理 #

```
umi g page user
```

```
import React from 'react';
import styles from './user.css';

export default () => {
  return (
    <div>
      <h1 className={styles.title}>Page userh1>
    div>
  );
}
```

#### 3.3.2 个人中心 #

```
umi g page profile
```

pages\profile.js

```
import React, { Component } from 'react'
import { history } from 'umi';
import styles from './profile.css';
export default class componentName extends Component {
  render() {
    return (
      <div>
        <h1 className={styles.title}>个人中心h1>
        <button onClick={()=>history.goBack()}>返回button>
      div>
    )
  }
}
```

#### 3.3.3 启动服务器 #

##### 3.3.3.1 启动配置 #

```
  "scripts": {
    "dev": "umi dev",
    "build": "umi build"
  }
```

##### 3.3.3.2 启动项目 #

```
npm run dev
```

##### 3.3.3.3 部署发布 #

- 

```
npm run build
```

## 4. 全局 layout #

- 约定 `src/layouts/index.js` 为全局路由，返回一个 React 组件，通过 `props.children` 渲染子组件

src/layouts/index.js

```
import React,{Component} from 'react';
import {Link} from 'umi';
export default class Layout extends Component{
    render() {
        return (
            <div>
                <ul>
          <li><Link to="/" >首页Link>li>
          <li><Link to="/user">用户管理Link>li>
                    <li><Link to="/profile">个人设置Link>li>
                ul>
                <div>
            {this.props.children}
                div>
            div>
        )
    }
}
```

## 5. 用户管理 #

### 5.1 嵌套路由 #

- umi 里约定目录下有 _layout.js 时会生成嵌套路由，以 _layout.js 为该目录的 layout pages/user/_layout.js

```
import React,{Component,Fragment} from 'react';
import {Link} from 'umi';
export default class User extends Component{
    render() {
        return (
            <div >
                <ul>
                    <li><Link to="/user/list">用户列表Link>li>
                    <li><Link to="/user/add">新增用户Link>li>
                ul>
                <div>
                    {this.props.children}
                div>
            div>
        )
    }
}
```

### 5.2 user/list.js #

/pages/user/list.js

```
import React,{Component,Fragment} from 'react';
import {Link} from 'umi';
export default class List extends Component{
    render() {
        return (
            <ul>
        <li><Link to="/user/detail/1">张三Link>li>
        <li><Link to="/user/detail/2">李四Link>li>
            ul>
        )
    }
}
```

#### 5.3 pages/user/add.js #

pages/user/add.js

```
import React,{Component} from 'react';
export default class Add extends Component{
    render() {
        return (
            <form >
                  <input/>
         <input type="submit" />
            form>
        )
    }
}
```

#### 5.3 动态路由 #

- 约定 [] 包裹的文件或文件夹为动态路由

src\pages\user\detail[id].js

```
import React,{Component} from 'react';
export default class  extends Component{
    render() {
        console.log(this.props);
        return (
            <table>
                <thead>
                    <tr>
                        <td>字段td>
                        <td>值td>
                    tr>
                thead>
                <tbody>
                    <tr>
                        <td>1td>
                        <td>张三td>
                    tr>
                tbody>
            table>
        )
    }
}
```

### 6. 权限路由 #

- 通过指定高阶组件 wrappers 达成效果

#### 6.1 profile.js #

```
import React, { Component } from 'react'
import { history } from 'umi';
import styles from './profile.css';
class Profile extends Component {
  render() {
    return (

        个人中心
        history.goBack()}>返回

    )
  }
}

+Profile.wrappers = ['@/wrappers/auth']
+export default Profile;
```

#### 6.2 auth.js #

src\wrappers\auth.js

```
import { Redirect } from 'umi';

export default (props) => {
  const isLogin = localStorage.getItem('isLogin');
  if (isLogin) {
    return <div>{ props.children }div>;
  } else {
    return <Redirect to={{pathname:"/login",state:{from:'/profile'}}} />;
  }
}
```

#### 6.3 pages/login.js #

pages/login.js

```
import React from 'react';
import styles from './login.css';
import { history } from 'umi';
export default (props) => {
  return (
    <div>
      <h1 className={styles.title}>Page loginh1>
      <button onClick={()=>{
        localStorage.setItem('isLogin','true');
        if(props.location.state&&props.location.state.from){
          history.push(props.location.state.from);
        }
      }}>登录button>
    div>
  );
}
```

## 7. 动态路由 #

### 7.1 前台运行时 #

src\app.js

```
export function patchRoutes({ routes }) {
    routes.unshift({
        path: '/foo',
        exact: true,
        component: require('./Foo').default,
    });
}
```

src\Foo.js

```
import React, { Component } from 'react'
export default class componentName extends Component {
  render() {
    return (
      <div>
      Foo
      div>
    )
  }
}
```

### 7.2 接口返回 #

src\app.js

```
let extraRoutes;
export  function modifyClientRenderOpts(memo) {
    memo.routes.unshift(...extraRoutes);
    return memo;
}
export function render(oldRender) {
    fetch('/api/routes').then(res => res.json()).then((res) => {
        extraRoutes = res.map(item=>{
            let component = item.component;
            component = require(`./components/${component}`).default;
            return { ...item, component};
        });
        oldRender();
    })
}
```

mock\routes.js

```
export default {
    'GET /api/routes': [
        {
            path:'/foo',
            component:'Foo.js'
        }
    ]
}
```