## RACS前端开发指南

### 1. 阅读设计文档

\* 前端设计文档：[设计文档示例](http://128.96.99.110:8080/RACS/%E5%89%8D%E7%AB%AF%E8%AE%BE%E8%AE%A1/RACS010101-%E5%9C%BA%E6%89%80%E4%BF%A1%E6%81%AF%E7%AE%A1%E7%90%86/)

\* RESTfulAPI接口设计文档：[设计文档示例](http://128.96.99.110:8080/RACS/%E8%81%94%E6%9C%BA%E6%9C%8D%E5%8A%A1/A30714001%E9%99%A2%E6%A0%A1%E5%88%97%E8%A1%A8%E6%9F%A5%E8%AF%A2/)

### 2. 文件创建规范

\* 界面主体文件：基于React的jsx文件，遵从dva框架。用于定义页面主体Component

\* 界面样式文件(非必须)：基于less语法编写，运用css语言描述页面样式

\* 界面模型文件：遵从dva框架。用于定义界面数据模型，编写界面与后台交互的业务处理逻辑

`示例` 信息管理模块，用例RACS010101-场所信息管理

```

// 界面主体文件

pages/IM/site/SiteInfoAdmin.js

// 界面主体文件

pages/IM/site/SiteInfoAdmin.less

// 界面模型文件

models/IM/site.js

```

功能模块规划

```

信息管理 IM

资金管理 FM

安全管理 SM

政务管理 GM

```

### 3. 开发实施步骤

#### 3.1 配置页面路由

src/common/router.js

```javascript

const routerConfig = {

'/IM/gather/site': {

component: dynamicWrapper(app, ['IM/site'], () => import('../pages/site/SiteInfoAdmin')),

},

'/IM/gather/site-new': {

component: dynamicWrapper(app, ['IM/site'], () => import('../pages/site/SiteInfoEnter')),

},

'/IM/gather/site-modify/:id': {

component: dynamicWrapper(app, ['IM/site'], () => import('../pages/site/SiteInfoEnter')),

},

'/IM/site-detail/:id': {

component: dynamicWrapper(app, ['IM/site'], () => import('../pages/site/SiteInfoDetail')),

},

};

```

`说明`

> dynamicWrapper方法为遵从dva框架的包装方法，用于建立界面模型文件(规定位于models目录中)与界面主体文件(一般位于pages目录)之间的关系

> 菜单功能的界面组件都基于BasicLayout

#### 3.2 定义界面数据结构

界面模型文件中定义界面数据结构，此类数据一般和后台数据交互相关

```javascript

export default {

namespace: 'site',

// 定义界面数据结构

state: {

pageSearchResult: {

list: [],

pagination: {},

},

pageSearching: false,

enterSubmitting: false,

},

subscriptions: {

},

effects: {

},

reducers: {

},

};

```

界面主体文件里Component构造方法中定义数据结构，此类数据一边和界面交互状态相关

```javascript

export default class SiteInfoAdmin extends PureComponent {

state = {

modalVisible: false,

expandForm: false,

formValues: {},

};

}

```

#### 3.2 编写界面布局和样式

```javascript

// 引入React和dva框架

import React, { PureComponent } from 'react';

import { connect } from 'dva';

// 引入antd组件

// eslint-disable-next-line

import { Form, Input } from 'antd';

// 引入自定义组件

import PageHeaderLayout from '../../../layouts/PageHeaderLayout';

// 引入样式文件

// import styles from './SiteInfoAdmin.less';

const FormItem = Form.Item;

// 关联models中state属性和Component的props属性，关联变量名为model文件中定义的命名空间

@connect(state => ({

site: state.site,

}))

// Form工具方法props属性引入语法糖

@Form.create()

// 定义页面主体Component，命名和文件名保持一致

export default class SiteInfoAdmin extends PureComponent {

// 定义界面数据结构

state = {

formValues: {},

};

// 页面渲染入口方法

render() {

const { getFieldDecorator } = this.props.form;

// 组织展示数据

const { } = this.props;

const { } = this.state;

return (

<PageHeaderLayout>

<Card bordered={false}>

<Form onSubmit={this.handleSubmit}>

<FormItem label="输入测试">

{getFieldDecorator('inputTest')(

<Input placeholder="请输入" />

)}

</FormItem>

</Form>

</Card>

</PageHeaderLayout>

);

}

}

// 文件结尾必须含空行

```

#### 3.3 编写界面交互与业务处理

界面主体文件中编写界面交互事件处理逻辑

```javascript

export default class APIDetail extends Component {

// 定义事件处理方法

handleSubmit = (e) => {

e.preventDefault();

const { dispatch, form } = this.props;

form.validateFields((err, fieldsValue) => {

if (err) return;

const values = {

...fieldsValue,

};

this.setState({

formValues: values,

});

dispatch({

type: 'site/new',

payload: values,

});

});

}

componentWillMount() {

}

componentDidMount() {

const { dispatch } = this.props;

dispatch({

type: 'site/fetch',

});

}

componentWillReceiveProps(nextProps) {

}

shouldComponentUpdate(nextProps, nextState) {

}

componentWillUpdate(nextProps, nextState) {

}

componentDidUpdate(prevProps, prevState) {

}

componentWillUnmount() {

}

componentDidCatch(error, info) {

}

render() {

return (

<PageHeaderLayout>

<Card bordered={false}>

<Form onSubmit={this.handleSubmit}>

<FormItem label="输入测试">

{getFieldDecorator('inputTest')(

<Input placeholder="请输入" />

)}

</FormItem>

</Form>

</Card>

</PageHeaderLayout>

);

}

}

```

界面模型文件中编写与后台服务交互的处理逻辑

```javascript

import { stringify } from 'qs';

import request from '../../utils/request';

export async function querySite(params) {

return request('/api/im/site/search', {

method: 'POST',

body: {

...params,

method: 'post',

},

});

}

export default {

namespace: 'site',

state: {

pageSearchResult: {

list: [],

pagination: {},

},

pageSearching: false,

enterSubmitting: false,

},

effects: {

\*search({ payload }, { call, put }) {

yield put({

type: 'changePageSearching',

payload: true,

});

const response = yield call(querySite, payload);

yield put({

type: 'updateSearchResult',

payload: response,

});

yield put({

type: 'changePageSearching',

payload: false,

});

},

},

reducers: {

updateSearchResult(state, action) {

return {

...state,

pageSearchResult: {

list: action.payload.site\_grp,

pagination: {},

},

};

},

changePageSearching(state, action) {

return {

...state,

pageSearching: action.payload,

};

},

},

};

```

### 4. 编码规范

#### 4.1 Sublime配置

```

{

"font\_size": 14,

"ignored\_packages":

[

"Vintage"

],

"show\_encoding": true,

"show\_line\_endings": true,

"translate\_tabs\_to\_spaces": true,

"word\_wrap": true,

"tab\_size": 2,

"default\_line\_ending": "unix"

}

```

#### 4.2 ESLint常见语法校验

### 5. 调试与问题诊断