

前端开发

导航菜单

菜单, <https://ant.design/components/menu-cn/>

Menu 菜单组件

mode有水平、垂直、内嵌

Menu.Item 菜单项

key 菜单项item的唯一标识

```
// src/index.js
import React from 'react';
import ReactDOM from 'react-dom';
import { Route, Link, BrowserRouter as Router } from 'react-router-dom';
import { Menu, Icon } from 'antd';
import Login from './component/login';
import Reg from './component/reg';
import Pub from './component/pub'; // 发布页
//import L from './component/list'; // 列表页
//import Post from './component/post'; // 详情页

import 'antd/lib/menu/style';
import 'antd/lib/icon/style';

const Home = () => (
  <div>
    <h2>Home</h2>
  </div>
);

const About = () => (
  <div>
    <h2>About</h2>
  </div>
);

const App = () => (
  <Router>
    <div>
      <div>
        <Menu mode="horizontal">
          <Menu.Item key="home"><Link to="/"><Icon type="home" />主页</Link></Menu.Item>
          <Menu.Item key="login"><Link to="/login"><Icon type="login" />登录</Link></Menu.Item>
          <Menu.Item key="reg"><Link to="/reg">注册</Link></Menu.Item>
          <Menu.Item key="pub"><Link to="/">发布</Link></Menu.Item>
          <Menu.Item key="list"><Link to="/"><Icon type="bars" />文章列表</Link></Menu.Item>
          <Menu.Item key="about"><Link to="/about">关于</Link></Menu.Item>
        </Menu>
      </div>
    </div>
  </Router>
);
```

```

    </div>
    <Route path="/about" component={About} />
    <Route path="/login" component={Login} />
    <Route path="/reg" component={Reg} />
    <Route exact path="/" component={Home} />
  </div>
</Router>
);

ReactDOM.render(<App />, document.getElementById('root'));

```

布局

采用上中下布局, 参考 <https://ant.design/components/layout-cn/>

```

import React from 'react';
import ReactDOM from 'react-dom';
import { Route, Link, BrowserRouter as Router } from 'react-router-dom';
import { Layout, Menu, Icon } from 'antd';
import Login from './component/login';
import Reg from './component/reg';
import Pub from './component/pub'; // 发布页
// import L from './component/list'; // 列表页
// import Post from './component/post'; // 详情页

import 'antd/lib/layout/style';
import 'antd/lib/menu/style';
import 'antd/lib/icon/style';

const { Header, Content, Footer } = Layout; // 上中下

const Home = () => (
  <div>
    <h2>Home</h2>
  </div>
);

const About = () => (
  <div>
    <h2>About</h2>
  </div>
);

const App = () => (
  <Router>
    <Layout>
      <Header>
        <Menu mode="horizontal" theme="dark">
          <Menu.Item key="home"><Link to="/"><Icon type="home" />主页</Link></Menu.Item>
          <Menu.Item key="login"><Link to="/login"><Icon type="login" />登录</Link></Menu.Item>
          <Menu.Item key="reg"><Link to="/reg">注册</Link></Menu.Item>
          <Menu.Item key="pub"><Link to="/pub">发布</Link></Menu.Item>
          <Menu.Item key="list"><Link to="/"><Icon type="bars" />文章列表</Link></Menu.Item>

```

```

        <Menu.Item key="about"><Link to="/about">关于</Link></Menu.Item>
      </Menu>
    </Header>
    <Content style={{ padding: '8px 50px' }}>
      <div style={{ background: '#fff', padding: 24, minHeight: 280 }}>
        <Route path="/about" component={About} />
        <Route path="/login" component={Login} />
        <Route path="/reg" component={Reg} />
        <Route path="/pub" component={Pub} />
        <Route exact path="/" component={Home} />
      </div>
    </Content>
    <Footer style={{ textAlign: 'center' }}>
      马哥教育©2008-2018
    </Footer>
  </Layout>
</Router>
);

ReactDOM.render(<App />, document.getElementById('root'));

```

注意，`<Menu.Item key="list"><Link to="/"><Icon type="bars" />文章列表</Link></Menu.Item>` 这里Link需要包着Icon，否则会错位。

博文业务

url	method	说明
/post/pub	POST	提交博文的title、content，成功返回Json，post_id
/post/id	GET	返回博文详情，返回Json，post_id、title、author、author_id、postdate（时间戳）、content
/post/	GET	返回博文标题列表

业务层

创建service/post.js文件，新建PostService类。

```

import axios from 'axios';
import { observable } from 'mobx';

class PostService{
  @observable msg = '';

  pub(title, content) {
    console.log(title);
    axios.post('/api/post/pub', {
      title, content
    });/* dev server会代理 */
  }
}

```

```

        .then(
            response => { // 此函数要注意this的问题
                console.log(response.data);
                console.log(response.status);
                this.msg = '博文提交成功';
            }
        ).catch(
            error=> {
                console.log(error);
                this.msg = '博文提交失败'; //+ 信息显示
            }
        )
    }
}

const postService = new PostService();
export {postService};

```

发布组件

使用Form组件, <https://ant.design/components/form-cn/>

```

// component/pub.js
import React from 'react';
import { observer } from 'mobx-react';
import { Form, Icon, Input, Button, message } from 'antd';
import { inject } from '../utils';
import { postService as service } from '../service/post';

import 'antd/lib/form/style';
import 'antd/lib/icon/style';
import 'antd/lib/input/style';
import 'antd/lib/button/style';
import 'antd/lib/message/style';

const FormItem = Form.Item;
const { TextArea } = Input;

@Inject({service})
@observer
export default class Pub extends React.Component {
    handleSubmit(event){
        event.preventDefault();
        const [title, content] = event.target; // event.target返回form, 而form是表单控件的数组
        this.props.service.pub(title.value, content.value);
    }

    render() {
        let msg = this.props.service.msg;
        return (
            <Form layout="vertical" onSubmit={this.handleSubmit.bind(this)}>

```

```

    <FormItem label="标题" labelCol={{ span: 4 }} wrapperCol={{ span: 14 }}>
      <Input placeholder="标题" />
    </FormItem>
    <FormItem label="内容" labelCol={{ span: 4 }} wrapperCol={{ span: 14 }}>
      <TextArea rows={10} />
    </FormItem>
    <FormItem wrapperCol={{ span: 14, offset: 4 }}>
      <Button type="primary" htmlType="submit">提交</Button>
    </FormItem>
  </Form>
);
}

componentDidUpdate(prevProps, prevState) { //+ 渲染后显示消息组件
  if (prevProps.service.msg) {
    message.info(prevProps.service.msg, 3, ()=>prevProps.service.msg='');
  }
}
}

```

- Form 表单组件，layout是垂直，onSubmit提交，注意这个提交的this就是表单自己
- FormItem 表单项
 - label设置控件前的标题
 - labelCol设置label的宽度，wrapperCol是label后占用的宽度，这些单位都是栅格系统的宽度
 - 参考 Antd/Form表单/表单布局 <https://ant.design/components/form-cn/#components-form-demo-layout>
 - 栅格系统：AntD提供了一个类似于Bootstrap的栅格系统，它将页面分成了24等分的列
 - span代表占几个格子；offset表示左边空出多少个格子
- Input 输入框，placeholder提示字符
- TextArea 文本框，rows行数
- Button 按钮
 - type是按钮的类型，也决定它的颜色
 - htmlType使用HTML中的type值，submit是提交按钮会触发提交行为，一定要写，但是handleSubmit中要阻止默认行为。

富文本编辑器(可选)

<https://github.com/margox/braft-editor>

使用yarn安装

```
$ yarn add braft-editor
```

使用npm安装

```
$ npm install braft-editor --save
```

组件使用

```
import BraftEditor from 'braft-editor';
import 'braft-editor/dist/index.css';
```

业务层改进

header中的Jwt

由于与后台Django Server通信，身份认证需要Jwt，这个要放到request header中。使用axios的API

`axios.post(url[, data[, config]])` 可以使用第三个参数config。config是一个对象，字典中设置headers字段，该字段的值依然是对象，都是键值对形式。

```
// service/post.js
import axios from 'axios';
import { observable } from 'mobx';
import store from 'store';

class PostService {
  constructor() {
    this.axios = axios.create({
      baseURL: '/api/post/'
    });
  }

  @observable msg = '';

  getJwt(){
    return store.get('token', null);
  }

  pub(title, content) {
    console.log(title);
    axios.post('/api/post/pub', {
      title, content
    }, {
      headers: {'Jwt': this.getJwt()}
    })/* dev server会代理 */
    .then(
      response => { // 此函数要注意this的问题
        console.log(response.data);
        console.log(response.status);
        this.msg = '博文提交成功';
      }
    ).catch(
      error => {
        console.log(error);
        this.msg = '博文提交失败'; //+ 信息显示
      }
    )
  }
}

const postService = new PostService();
export { postService };
```

编写文章内容并提交，注意jwt过期问题。

文章列表页组件

创建component/list.js, 创建List组件。在index.js中提交菜单项和路由。

使用L是为了避免和AntD的List冲突。

```
// index.js
import L from './component/list'; // 列表页

const App = () => (
  <Router>
    <Layout>
      <Header>
        <Menu mode="horizontal" theme="dark">
          <Menu.Item key="home"><Link to="/"><Icon type="home" />主页</Link></Menu.Item>
          <Menu.Item key="login"><Link to="/login"><Icon type="login" />登录</Link></Menu.Item>
          <Menu.Item key="reg"><Link to="/reg">注册</Link></Menu.Item>
          <Menu.Item key="pub"><Link to="/pub">发布</Link></Menu.Item>
          <Menu.Item key="list"><Link to="/list"><Icon type="bars" />文章列表</Link></Menu.Item>
          <Menu.Item key="about"><Link to="/about">关于</Link></Menu.Item>
        </Menu>
      </Header>
      <Content style={{ padding: '8px 50px' }}>
        <div style={{ background: '#fff', padding: 24, minHeight: 280 }}>
          <Route path="/about" component={About} />
          <Route path="/login" component={Login} />
          <Route path="/reg" component={Reg} />
          <Route path="/list" component={L} />
          <Route path="/pub" component={Pub} />
          <Route exact path="/" component={Home} />
        </div>
      </Content>
      <Footer style={{ textAlign: 'center' }}>
        马哥教育@2008-2018
      </Footer>
    </Layout>
  </Router>
);
```

查询字符串处理

用户请求的URL是 `http://127.0.0.1:3000/list?page=2`, url要被转换成 `/api/post/?page=2`, 如何提取查询字符串?

现在前端路由有react-router管理, 它匹配路径后, 才会路由。它提供了匹配项, 它将匹配的数据注入组件的props中, 也可以使用解构提取 `const { match, location } = this.props`。

location也是一个对象, pathname表示路径, search表示查询字符串。 `{pathname: "/list", search: "?page=2"}`。拿到查询字符串后, 可以使用URLSearchParams解析它, 但是它是实验性的, 不建议用在生产环境中。本次将查询字符串直接拼接发往后端, 由Django服务器端判断。

```
var params = new URLSearchParams(url.search);
console.log(params.get('page'), params.get('size'))
```

参考 <https://reacttraining.com/react-router/core/api/location>

List组件

Ant design的List，需要使用3.x版本，修改package.json的版本信息 "antd": "^3.1.5"。

然后 `$ npm update`，更新成功后，就可以使用List组件了。

或者 `$ yarn upgrade antd@^3.1.5`。

注意：npm安装和yarn安装不要混用。

component/list.js代码如下

```
import React from 'react';
import { inject } from '../utils';
import { postService as service } from '../service/post';
import { List } from 'antd';
import { observer } from 'mobx-react';

import 'antd/lib/list/style';

@inject({ service })
@observer
export default class L extends React.Component {
  constructor(props) {
    super(props);
    // 不处理查询字符串，传到后台服务提取处理
    props.service.list(props.location.search); // ?page=1
  }

  render() {
    const {posts:data, pagination} = this.props.service.posts;
    console.log(data)
    if (data.length)
      return (
        <List
          header="博文列表"
          footer={<div>Footer</div>}
          bordered
          dataSource={data}
          renderItem={item => (<List.Item>{item.title}</List.Item>)}
        />
      );
    else
      return (<div>无数据</div>);
  }
}
```


List 列表组件

bordered 有边线

dataSource 给定数据源

renderItem 渲染每一行，给定一个一参函数迭代每一行

List.Item 每一行的组件

使用Link组件增加链接

```
<List bordered={true} dataSource={data} renderItem={
  item => (<List.Item>
    <Link to={'/post/' + item.post_id}>{item.title}</Link>
  </List.Item>)
} />
```

如果需要根据复杂的效果可以这样

```
<List bordered={true} dataSource={data} renderItem={
  item => (<List.Item>
    <List.Item.Meta title={<Link to={'/post/' + item.post_id}>{item.title}</Link>} />
  </List.Item>)
} />
```

```
<Link to={'/post/' + item.post_id}>{item.title}</Link> 这是详情页的链接
```

PostService代码如下

```
import axios from 'axios';
import { observable } from 'mobx';
import store from 'store';

class PostService {
  constructor() {
    this.axios = axios.create({
      baseURL: '/api/post/'
    });
  }

  @observable msg = '';
  // 博文列表,分页信息
  @observable posts = {'posts':[], 'pagination':{'page:1, size:20, count:0, pages:0}}

  getJwt(){
    return store.get('token', null);
  }

  pub(title, content) {
    console.log(title);
    axios.post('/api/post/pub', {
      title, content
    });
  }
}
```

```

    }, {
      headers: { 'Jwt': this.getJwt() }
    })/* dev server会代理 */
    .then(
      response => { // 此函数要注意this的问题
        console.log(response.data);
        console.log(response.status);
        this.msg = '博文提交成功';
      }
    ).catch(
      error => {
        console.log(error);
        this.msg = '博文提交失败'; //+ 信息显示
      }
    )
  }

  list(search) {
    this.axios.get(search).then(response => {
      this.posts = {
        'posts': response.data.posts,
        'pagination': response.data.pagination // 分页信息
      };
    }).catch(error => {
      console.log(error);
      this.msg = '文章列表加载失败'; //+ 信息显示
    });
  }
}

const postService = new PostService();
export { postService };

```

测试 <http://localhost:3000/list?page=2&size=2>

分页功能

分页还是需要解析查询字符串的，因此写一个解析函数，把这个函数放入utils.js中，需要时调用。

```

let url = '?id=5&page=1&size=20&id=&age=20&name=abc&name=汤姆=&测试=1'

function parse_qs(qs, re=/(\w+)=(\[^\&]+\))/){
  let obj = {};
  if (qs.startsWith('?'))
    qs = qs.substr(1)
  console.log(qs);
  qs.split('&').forEach(element => {
    let match = re.exec(element);
    //console.log(match)
    if (match) obj[match[1]] = match[2];
  });
  return obj;
}

```

```
}

console.log(parse_qs(url))
```

分页使用了Pagination组件，在List组件的render函数的List组件中使用pagination属性，这个属性内放入一个pagination对象，有如下属性

- current，当前页
- pageSize，页面内行数
- total，记录总数
- onChange，页码切换时调用，回调函数为(pageNo, pageSize) => {}，即切换是获得当前页码和页内行数。

可以参考 <https://ant.design/components/list-cn/#components-list-demo-vertical>

component/list.js代码修改如下

```
import React from 'react';
import { inject } from '../utils';
import { postService as service } from '../service/post';
import { List, Pagination } from 'antd';
import { observer } from 'mobx-react';
import { Link } from 'react-router-dom';

import 'antd/lib/list/style';
import 'antd/lib/pagination/style';

@inject({ service })
@observer
export default class L extends React.Component {
  constructor(props) {
    super(props);
    // 不处理查询字符串，传到后台服务提取处理
    props.service.list(props.location.search); // ?page=1
  }

  handleChange(page, pageSize) {
    // 不管以前查询字符串是什么，重新拼接 查询字符串 向后传
    let search = `?page=${page}&size=${pageSize}`;
    this.props.service.list(search);
  }

  render() {
    const {posts:data=[], pagination={}} = this.props.service.posts;

    if (data.length) {
      const {page:current=1, count:total=0, size:pageSize=20} = pagination;

      return (
        <List
          header="博文列表"
          bordered

```

```

        dataSource={data}
        renderItem={item => (
          <List.Item>
            <Link to={'/post/' + item.post_id}>{item.title}</Link>
          </List.Item>
        )}
        pagination={{
          current:current,
          total:total,
          pageSize:pageSize,
          onChange:this.handleChange.bind(this)
        }}
      />
    );
  }
  else
    return (<div>无数据</div>);
}
}

```

测试可以切换页面。但是鼠标放到左右两端发现上一页、下一页是英文，如何修改？国际化。

国际化

index.js修改如下(部分代码)

```

import { LocaleProvider } from 'antd';
import zhCN from 'antd/lib/locale-provider/zh_CN';

ReactDOM.render(
  <LocaleProvider locale={zhCN}>
    <App />
  </LocaleProvider>
, document.getElementById('root'));

```

将App这个根组件包裹住就行了，再看分页组件就显示中文了。

浏览器地址不变的问题（可选）

基本上没有什么问题了，但是，如果在地址栏里面输入 `http://127.0.0.1:3000/list?size=2&page=2` 后，再切换分页，地址栏URL不动，不能和当前页一致。

这个问题的解决有一定的难度。需要定义Pagination的itemRender属性，定义一个函数，这个函数有3个参数

- current，当前pageNo
- type，当前类型，有三种可能，上一页为prev，下一页为next，页码为page
- originalElement，不要动这个参数，直接返回就行了

```
const itemRender = (page, type: 'page' | 'prev' | 'next', originalElement) => React.ReactNode
```

```

// component/list.js
import React from 'react';
import { inject } from '../utils';

```

```

import { postService as service } from '../service/post';
import { List, Pagination } from 'antd';
import { observer } from 'mobx-react';
import { Link } from 'react-router-dom';

import 'antd/lib/list/style';
import 'antd/lib/pagination/style';

@Inject({ service })
@observer
export default class L extends React.Component {
  constructor(props) {
    super(props);
    // 不处理查询字符串, 传到后台服务提取处理
    props.service.list(props.location.search); // ?page=1
  }

  handleChange(page, pageSize) {
    // 不管以前查询字符串是什么, 重新拼接 查询字符串 向后传
    let search = `?page=${page}&size=${pageSize}`;
    this.props.service.list(search);
  }

  getUrl(c){
    let obj = parse_qs(this.props.location.search);
    let {size=20} = obj;
    return `/list?page=${c}&size=${size}`;
  }

  renderItem(page, type, originalElement){
    console.log(originalElement)
    if (type === 'page')
      return <Link to={this.getUrl(page)}>{page}</Link>;
    return originalElement;
  }

  render() {
    const {posts:data=[], pagination={}} = this.props.service.posts;

    if (data.length) {
      const {page:current=1, count:total=0, size:pageSize=20} = pagination;

      return (
        <List
          header="博文列表"
          bordered
          dataSource={data}
          renderItem={item => (
            <List.Item>
              <Link to={'/post/' + item.post_id}>{item.title}</Link>
            </List.Item>
          )}
          pagination={{
            current:current,
            total:total,
          }}
        />
      )
    }
  }
}

```

```

        pageSize: pageSize,
        onChange: this.handleChange.bind(this),
        itemRender: this.itemRender.bind(this)
      }}
    />
  );
}
else
  return (<div>无数据</div>);
}
}

```

基本解决问题。但是，上一页、下一页点击不能改变浏览器地址栏。

```

import React from 'react';
import { inject } from '../utils';
import { postService as service } from '../service/post';
import { List, Pagination } from 'antd';
import { observer } from 'mobx-react';
import { Link } from 'react-router-dom';

import 'antd/lib/list/style';

@Inject({ service })
@observer
export default class L extends React.Component {
  constructor(props) {
    super(props);
    // 不处理查询字符串，传到后台服务提取处理
    props.service.list(props.location.search); // ?page=1
  }

  handleChange(page, pageSize) {
    // 不管以前查询字符串是什么，重新拼接 查询字符串 向后传
    let search = `?page=${page}&size=${pageSize}`;
    this.props.service.list(search);
  }

  getUrl(c){
    let obj = parse_qs(this.props.location.search);
    let {size=20} = obj;
    return `/list?page=${c}&size=${size}`;
  }

  itemRender(page, type, originalElement){
    console.log(originalElement)
    if (page == 0) return originalElement;
    if (type === 'page')
      return <Link to={this.getUrl(page)}>{page}</Link>;
    if (type === 'prev')

```

```

        return <Link to={this.getUrl(page)} className="ant-pagination-item-link">&lt;
</Link>;
        if (type === 'next')
            return <Link to={this.getUrl(page)} className="ant-pagination-item-link">&lt;
</Link>;
    }

    render() {
        const {posts:data=[], pagination={}} = this.props.service.posts;

        if (data.length) {
            const {page:current=1, count:total=0, size:pageSize=20} = pagination;

            return (
                <List
                    header="博文列表"
                    bordered
                    dataSource={data}
                    renderItem={item => (<List.Item>
                        <Link to={'/post/' + item.post_id}>{item.title}</Link>
                    </List.Item>)}
                    pagination={{
                        current:current,
                        total:total,
                        pageSize:pageSize,
                        onChange:this.handleChange.bind(this),
                        itemRender:this.itemRender.bind(this)
                    }}
                />
            );
        }
        else
            return (<div>无数据</div>);
    }
}

```

至此，分页问题基本解决。

详情页组件

index.jsp

```

import Post from './component/post'; // 详情页

<Route exact path="/post/:id" component={Post} />

```

新建component/post.js，创建Post组件。使用antd Card布局。

```

import React from 'react';

```

```

import { inject } from '../utils';
import { postService as service } from '../service/post';
import { observer } from 'mobx-react';
import { message, Card } from 'antd';

import 'antd/lib/message/style';
import 'antd/lib/card/style';

@Inject({ service })
@observer
export default class Post extends React.Component {
  constructor(props) {
    super(props);
    let { id = -1 } = props.match.params;
    this.props.service.getPost(id);
  }
  render() {
    let msg = this.props.service.msg;
    const { title="", content="", author, postdate } = this.props.service.post;
    if (title) {
      return (<Card title={title} bordered={false} style={{ width: 300 }}>
        <p>{author}</p>
        <p>{new Date(postdate*1000).toLocaleDateString()}</p>
        <p>{content}</p>
      </Card>);
    }
    else
      return (<div>无数据</div>);
  }
  componentDidUpdate(prevProps, prevState) { //+ 渲染后显示消息组件
    if (prevProps.service.msg) {
      message.info(prevProps.service.msg, 3, ()=>prevProps.service.msg='');
    }
  }
}

```

至此，前后端分离的博客系统基本框架搭好了，去看看页面的成果。

service/post.js代码如下

```

import axios from 'axios';
import { observable } from 'mobx';
import store from 'store';

class PostService {
  constructor() {
    this.axios = axios.create({
      baseURL: '/api/post/'
    });
  }

  @observable msg = '';
  // 博文列表,分页信息

```



```

@observable posts = {'posts':[], 'pagination':{'page:1, size:20, count:0, pages:0}}

@observable post = {}; // 文章

getJwt(){
    return store.get('token', null);
}

pub(title, content) {
    console.log(title);
    axios.post('/api/post/pub', {
        title, content
    }, {
        headers: {'Jwt': this.getJwt()}
    })/* dev server会代理 */
    .then(
        response => { // 此函数要注意this的问题
            console.log(response.data);
            console.log(response.status);
            this.msg = '博文提交成功';
        }
    ).catch(
        error => {
            console.log(error);
            this.msg = '博文提交失败'; //+ 信息显示
        }
    )
}

list(search) {
    this.axios.get(search).then(response => {
        this.posts = {
            'posts':response.data.posts,
            'pagination':response.data.pagination // 分页信息
        };
    }).catch(error => {
        console.log(error);
        this.msg = '文章列表加载失败'; //+ 信息显示
    });
}

getPost(id) {
    this.axios.get(id).then(response => {
        this.post = response.data.post;
    }).catch(error => {
        console.log(error);
        this.post = {}
        this.msg = '文章加载失败'; //+ 信息显示
    });
}

}

const postService = new PostService();

```

```
export { postService };
```

如果使用了富文本编辑器，那么显示的时候，发现不能按照网页标签显示。

原因是为了安全，防止XSS攻击，React不允许直接按照HTML显示。

使用dangerouslySetInnerHTML属性，这个名字提醒使用者很危险。

```
<p>{content}</p>
```

修改为

```
<p dangerouslySetInnerHTML={{__html:content}}></p>
```

