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>
      <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

/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 返回博文列表

业务层

创建service/post.js文件，新建PostService类。

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

export default 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' //+ 信息显示
```

```
            }
        ).catch(
            error => {
                console.log(error);
                this.msg = '登陆失败'; //+ 信息显示
            }
        )
    }
}
```

发布组件

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

```
import React from 'react';
import { Link, Redirect } from 'react-router-dom';
import { observer } from 'mobx-react';
import { message } from 'antd';
import { inject } from '../utils';
import { Form, Input, Button } from 'antd';
import FormItem from 'antd/lib/form/FormItem'; // 不在antd中单独导
import PostService from '../service/post';

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

const { TextArea } = Input;

const service = new PostService();

@inject({ service })
@observer
export default class Pub extends React.Component {
    handleSubmit(event) {
        event.preventDefault();
        console.log(event.target)
        let fm = event.target;
        console.log(fm[0].value);
        console.log(fm[1].value);
        this.props.service.pub(fm[0].value, fm[1].value);
    }

    render() {
        if (this.props.service.msg) {
            message.info(this.props.service.msg, 3,
            () => setTimeout(()=>this.props.service.msg = '',1000);
        }

        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>
        );
    }
}
```

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。
Input 输入框，placeholder提示字符
TextArea 文本框，rows行数
Button 按钮，htmlType使用HTML中的type值，submit是提交按钮会触发提交行为，但是handleSubmit中要阻止默认行为。

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

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

export default class PostService {
    constructor() {
        // 创建自定义实例，可以增加请求header
        this.axios = axios.create({
            baseURL : '/api/post/'
        });
    }

    @observable msg = "";

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

    pub (title, content) {
        console.log(title);

        this.axios.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' //+ 信息显示
            }
        ).catch(
            error => {
                console.log(error);
                this.msg = '登陆失败'; //+ 信息显示
            }
        )
    }
}
```

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。

```
// 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`，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，有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。

component/list.js代码如下

```
import React from 'react';
import { observer } from 'mobx-react';
import { message } from 'antd';
import { inject } from '../utils';
import {List} from 'antd';
import {Link} from 'react-router-dom';

import PostService from '../service/post';

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

const service = new PostService();

@inject({service})
@observer
export default class L extends React.Component {
    constructor(props) {
        super(props);
        // 将查询字符串向后传
        props.service.list(props.location.search);
    }
    render () {
        let data = this.props.service.posts;
        if (data.length) {
            return (
                <List bordered={true} dataSource={data} renderItem={
                    item => (<List.Item>{item.title}</List.Item>)
                } />
```

```
            );
        } else {
            return (<div></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**部分代码**如下

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

export default class PostService {
    constructor() {
        // 创建自定义实例，可以增加请求header
        this.axios = axios.create({
            baseURL : '/api/post/'
        });
    }

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

```
    pub (title, content) {/*省略*/}

    list (search) {
        this.axios.get(search)
        .then(
            response => { // 此函数要注意this的问题
                console.log(response.data);
                console.log(response.status);
                this.posts = response.data.posts;
                this.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
            }
        )
    }
}
```

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

```
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L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。

component/list.js代码修改如下

```
import React from 'react';
import { observer } from 'mobx-react';
import { message } from 'antd';
import { inject, parse_qs } from '../utils';
import {List} from 'antd';
import {Link} from 'react-router-dom';

import PostService from '../service/post';

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

const service = new PostService();

@inject({service})
@observer
export default class L extends React.Component {
    constructor(props) {
        super(props);
        // 将查询字符串向后传
        props.service.list(props.location.search);
    }

    handleChange(pageNo, pageSize) {
        console.log(pageNo, 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=' + pageNo + '&size=' + pageSize;
        this.props.service.list(search);
    }

    render () {
        let data = this.props.service.posts;
        if (data.length) {
            const pagination = this.props.service.pagination;
            return (
                <List bordered={true} dataSource={data} renderItem={
                    item => (<List.Item>
                        <List.Item.Meta title={<Link to={'/post/' + item.post_id}>{item.title}
</Link>} />
                     </List.Item>)
                }
                pagination={{
                    current:pagination.page,
                    pageSize:pagination.size,
                    total:pagination.count,
                    onChange:this.handleChange.bind(this)
                }}
                />
            );
        } else {
            return (<div></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，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

```
import React from 'react';
import { observer } from 'mobx-react';
import { message } from 'antd';
import { inject, parse_qs } from '../utils';
import {List} from 'antd';
import {Link} from 'react-router-dom';

import PostService from '../service/post';

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

const service = new PostService();

@inject({service})
@observer
export default class L extends React.Component {
    constructor(props) {
        super(props);
        // 将查询字符串向后传
        props.service.list(props.location.search);
    }
```

```
    handleChange(pageNo, pageSize) {
        console.log(pageNo, 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=' + pageNo + '&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(current, type, originalElement) {
        if (type === 'page')
            return <Link to={this.geturl(current)}>{current}</Link>;
        return originalElement;
    }

    render () {
        let data = this.props.service.posts;
        if (data.length) {
            const pagination = this.props.service.pagination;
            return (
                <List bordered={true} dataSource={data} renderItem={
                    item => (<List.Item>
                        <List.Item.Meta title={<Link to={'/post/' + item.post_id}>{item.title}
</Link>} />
                     </List.Item>)
                }
                pagination={{
                    current:pagination.page,
                    pageSize:pagination.size,
                    total:pagination.count,
                    onChange:this.handleChange.bind(this),
                    itemRender:this.itemRender.bind(this),
                }}
                />
            );
        } else {
            return (<div></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 { observer } from 'mobx-react';
import { message } from 'antd';
import { inject, parse_qs } from '../utils';
import {List} from 'antd';
```

```
import {Link} from 'react-router-dom';

import PostService from '../service/post';

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

const service = new PostService();

@inject({service})
@observer
export default class L extends React.Component {
    constructor(props) {
        super(props);
        // 将查询字符串向后传
        props.service.list(props.location.search);
    }

    handleChange(pageNo, pageSize) {
        console.log(pageNo, 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=' + pageNo + '&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(current, type, originalElement) {
        if (current == 0) return originalElement; // 竟然返回0，只能屏蔽它
        if (type === 'page')
            return <Link to={this.geturl(current)}>{current}</Link>;
        if (type === 'next')
            return <Link to={this.geturl(current)} className='ant-pagination-item-link'></Link>;
        if (type === 'prev')
            return <Link to={this.geturl(current)} className='ant-pagination-item-link'></Link>;
        return originalElement;
    }

    render () {
        let data = this.props.service.posts;
        if (data.length) {
            const pagination = this.props.service.pagination;
            return (
                <List bordered={true} dataSource={data} renderItem={
                    item => (<List.Item>
                        <List.Item.Meta title={<Link to={'/post/' + item.post_id}>{item.title}
</Link>} />
                     </List.Item>)
                }
```

```
                pagination={{
                    current:pagination.page,
                    pageSize:pagination.size,
                    total:pagination.count,
                    onChange:this.handleChange.bind(this),
                    itemRender:this.itemRender.bind(this),
                }}
                />
            );
        } else {
            return (<div></div>);
        }
    }
}
```

至此，分页问题解决。

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 { observer } from 'mobx-react';
import { message } from 'antd';
import { inject } from '../utils';
import { Card } from 'antd';

import PostService from '../service/post';

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

const service = new PostService();

@inject({ service })
@observer
export default class L extends React.Component {
    constructor(props) {
        super(props);
        // 匹配
        let { id = -1 } = props.match.params; // {id:'3'}
        props.service.getpost(id);
    }
    render() {
```

```
        let s = this.props.service;
        if (s.msg) {
            message.info(s.msg, 3, () => setTimeout(()=>s.msg='', 1000));
        }
        let post = s.post;
        if (post.title) {
            return <Card title={post.title} bordered={true} style={{ width: 600 }}>
                <p>{post.author} {new Date(post.postdate*1000).toLocaleDateString()}</p>
                <p>{post.content}</p>
            </Card>
        } else
            return (<div></div>);
    }
}
```

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。