首页 目录 退出

00:00 -14:27

展示从 API 服务器获取的所有文章

添加 App 组件

新建 'src/ui/App.js' 文件，添加如下代码：

```
import React, { Component } from 'react';
import {Link} from 'react-router';

class App extends Component {
  getStyles() {
    return {
      header: {
        height: '64px',
        width: '100%',
        backgroundColor: '#00bcd4',
        textAlign: 'center',
```

```
        lineHeight: '64px',
      },
      link: {
        fontSize: '1.5em',
        color: '#fff',
        textDecoration: 'none'
      }
    };
  }

  render() {
    let styles = this.getStyles();
    return (
      <div>
        <header style={styles.header}>
          <div style={styles.link}>BORN TO C(
        </header>
      </div>
    );
  }
}

export default App;
```

显示 App 组件

修改 `src/index.js` 文件，导入刚才定义的路由组件，并在页面中渲染出来：

```
import App from './ui/App';

render(<App />, document.getElementById('root
```

这时，在浏览器中访问地址 'http://localhost:8080'，会在页面中看到一个蓝底白字的条幅

定义 PostList 组件

新建文件 `src/ui/posts/PostList.js` 文件，用来显示所有文章标题的列表：

```
import React, { Component } from 'react';

class PostList extends Component {
  getStyles() {
    return {
      content: {
        position: 'relative',
        width: '100%',
        height: '60px',
        maxWidth: '600px',
        margin: '20px auto',
        backgroundColor: '#fff',
        borderRadius: '5px',
        padding: '16px',
        boxShadow: 'rgba(0, 0, 0, 0.12) 0px 1
      },
      title: {
        fontSize: '1.2em'
      }
    }
  }

  render() {
    const styles = this.getStyles();
    return (
      <div>
```

```
          <div style={styles.content}>
            <div style={styles.title}>文章标题</
          </div>
        </div>
      );
    }
  }

  export default PostList;
```

显示 PostList 组件

打开 `src/ui/App.js` 文件，导入 PostList 组件：

```
import PostList from './posts/PostList';
```

并在 App 组件中加载 PostList 组件：

```
return (
  <div>
    ...
    <PostList />
  </div>
);
```

这时，再浏览页面，会展示文章列表中的一个条目的样式

从 API 服务器获取所有文章

打开 PostList 组件所在文件，首先添加代码：

```
import axios from 'axios';
```

axios 用来向服务器发送 API 请求。

然后再添加代码:

```
constructor() {
  super();
  this.state = {
    posts: []
  }
}
```

定义一个 state 变量 `posts`，其初始值为一个空数组，目的是为了存储从服务器返回的所有文章。接下来使用 React 组件的生命周期函数

`componentWillMount()`

```
componentWillMount() {
  axios.get(`${Settings.host}/posts`).then(re
    this.setState({
      posts: res.data.posts
    })
  })
  .catch(error => {
    if (error.response) {
      // 服务器响应了客户端发送的请求，但服务器返回
      console.log(error.response.data.error);
    } else {
      // 比如 API 服务器宕机的时候，则打印 'Netwo
      console.log(error.message);
    }
```

```
    });
  }
```

在 PostList 组件将要挂载的时候，通过 `axios` 请求服务器端的 `http://localhost:3000/posts` 接口，若请求成功，则把 state 变量 `posts` 设置为 `res.data.posts`，代表所有文章；若请求失败，则把错误信息在浏览器控制台中打印出来。对于处理请求错误信息的方式，借鉴于 `axios` 官方文档提供的范例，可以参考这里

不过，这会儿打开浏览器控制台会看到这样的报错信息：

```
XMLHttpRequest cannot load http://localhost:3
```

上述报错信息是由浏览器的同源策略安全机制引发的。比如现在 PostList 组件所在页面的网址是 `http://localhost:8080`，在这个页面中通过 XMLHttpRequest 对象向后端 `http://localhost:3000/posts` 接口请求资源的时候，因为前端页面网址和后端 API 使用了不同的端口号，导致它们不属于同一个域，浏览器无法获取后端接口的响应信息，所以浏览器报告错误。

为了让浏览器安全的支持跨域请求，就涉及到 CORS（跨域资源共享）机制了。想了解更多关于 CORS 的知识点，请阅读 MDN 网站上的文档 HTTP

访问控制(CORS)。

在服务器端安装 cors 中间件

了解了一些关于 CORS 的知识之后，我们就需要在服务器端做些事情了，安装 cors 的中间件，它可以响应前端发送过来的 API 请求，并且让浏览器接受响应信息。

```
npm install --save cors
```

打开 `index.js` 文件，导入 `cors` 中间件

```
var cors = require('cors');
```

然后使用它的最简单方式，如下所示：

```
app.use(cors());
```

这行代码让所有的请求源都能访问服务器提供的接口，当服务器端响应 API 请求的时候，会在响应信息中添加一些响应头标识（response header），如下所示：

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,HEAD,PUT,P
```

添加配置文件

注意： 在本地开发环境下，获得所有文章的 API 地址是 `http://localhost:3000/posts`，若部署到服务器上，域名改变了，API 地址也会发生变化，若项目中多处请求 API，就得修改多处代码，很不方便。因此新建一个配置文件，在配置文件中定义一个域名变量存放域名地址，而 API 地址中的域名使用配置文件中的域名变量，这样当域名改变的时候，我们只需要修改配置文件中的域名就可以了。

新建一个配置文件 `src/settings.js`，添加代码：

```
const Settings = {
  host: 'http://localhost:3000'
}

export default Settings;
```

接下来，在 PostList 组件中导入刚刚定义的 `Settings` 对象常量：

```
import Settings from '../../settings';
```

更改 PostList 组件中使用的 API 地址：

```
axios.get(`${Settings.host}/posts`).then(res
```

显示所有文章

首先从 JS 库 Lodash 的 FP 模块中导入 `map` 方法

```
import map from 'lodash/fp/map';
```

使用 `map` 方法，把数组 `this.state.posts` 中的每篇文章编辑成一个显示文章标题的 React 小组件，最终生成一个新的组件数组 `postList` 挂载到页面中渲染出来。

```
render() {
  const styles = this.getStyles();
  const postList = map((post) => {
    return (
      <div style={styles.content} key={post._
        <div style={styles.title}>{post.title
      </div>
    )
  }, this.state.posts);

  return (
    <div>
      { postList }
    </div>
  );
}
```

保证本地 API 服务器运行的状态下，若 MongoDB 数据库中的 `posts` 集合中有数据的话，在浏览器中刷新 http://localhost:8080 页面，则把 `posts` 集合中 `title` 字段的数据都能显示出来。