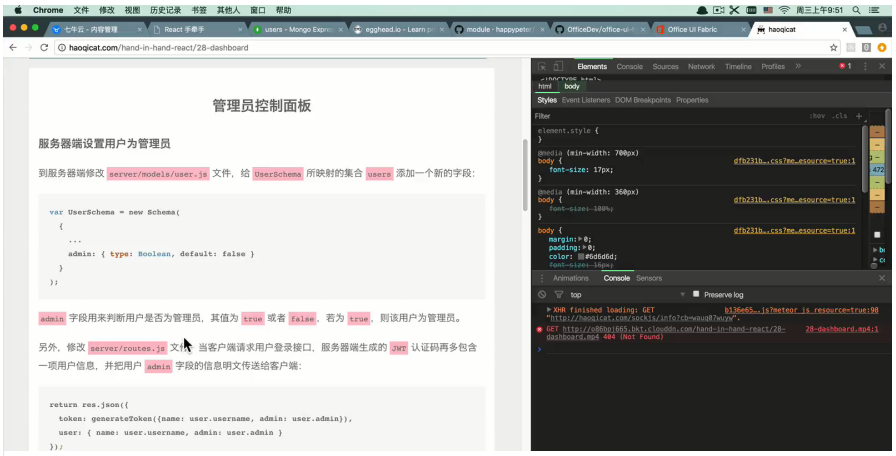




# 管理员功能实现

## 服务器端设置用户为管理员

到服务器端修改 `server/models/user.js` 文件，给 `UserSchema` 所映射的集合 `users` 添加一个新的字段：

```
var UserSchema = new Schema(
  {
    ...
    admin: { type: Boolean, default: false }
  }
);
```

`admin` 字段用来判断用户是否为管理员，其值为

`true` 或者 `false`，若为 `true`，则该用户为管理员。

另外，修改 `server/routes.js` 文件，当客户端请求用户登录接口，服务器端生成的 `JWT` 认证码再多包含一项用户信息，并把用户 `admin` 字段的信息明文传送给客户端：

```
return res.json({
  token: generateToken({name: user.username,
  user: { name: user.username, admin: user.ad
});
```

服务器端更改完毕之后，可以到客户端新注册一个用户名 'trump'，然后通过 `mongo` 或者 `mongo-express` 工具把这个新注册用户设置为管理员，若使用 `mongo` 实现，命令如下：

```
use react-hand-in-hand
db.users.update({username: 'trump'}, {$set: {
```

上述命令，把用户 `trump` 设置为管理员。

## 前端管理员登录之后渲染 DashBoard 组件

修改 `client/src/redux/acitons/authActions.js` 文件，当前端请求用户登录接口成功之后，若用户为管理员，则跳转到 `/dashboard` 页面：

```
user.admin === true ? browserHistory.push(`/
```

## 编写 DashBoard 组件

新建文件 `src/ui/DashBoard.js`，添加代码：

```
import React, { Component } from 'react';
class DashBoard extends Component {
  render() {
    const styles = {
      root: {
        maxWidth: '720px',
        margin: '30px auto'
      },
      actions: {
        marginTop: '32px',
        marginBottom: '32px',
        textAlign: 'center'
      }
    }
    return (
      <div style={styles.root}>
        <div style={styles.actions}>
          管理员控制面板
        </div>
      </div>
    );
  }
}
export default DashBoard;
```

## 添加 DashBoard 组件对应的路由

修改路由文件 `src/routes.js`，首先导入 DashBoard 组件：

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

添加 DashBoard 组件对应的路由：

```
<Route path='/' component={App}>
  <Route path='/DashBoard' component={DashBoa
</Route>
);
```

## 限制 DashBoard 组件的访问权限

DashBoard 组件只有管理员才能访问，所以用户在浏览器中访问 `/dashboard` 路径的时候，需要先验证用户的身份是否为管理员，把 DashBoard 组件对应的路由更改为：

```
<Route path='/DashBoard' component={DashBoard
```

添加了一个 `onEnter` 属性，在访问页面之前，先执行 `onEnter` 钩子函数，判断登录用户是否为管理员。

onEnter

## 编写 onEnter 钩子函数

```
function requireAuth(nextState, replace) {
  if (!isAdmin()) {
    replace('/login')
  }
}
```

当用户要访问 `/dashboard` 页面的时候，若用户不是管理员则重定向到 `/login` 页面。

## 定义 isAdmin 函数

`isAdmin` 函数用来判断用户是否为管理员：

```
function isAdmin() {
  if (!sessionStorage.getItem('jwtToken') &&
  const user = JSON.parse(sessionStorage.getI
  return user.admin === true ? true : false
}
```

首先判断用户是否登录，若没有登录，则函数返回值为 `false`；若用户登录了，则读取保存在 `sessionStorage` 中的 `currentUser` 键名对应的数值，然后再判断用户是否是管理员，若是管理员则返回 `true`，若不是则返回 `false`。

`user` 键名所对应的用户信息是一个对象类型，但是通过下面一行代码：

```
sessionStorage.getItem('user')
```

返回的用户信息是一个字符串类型，所以还需要借助 `JSON.parse()` 接口把字符串类型的数据解析成对象类型，这样才能访问用户信息中的各个属性值。