
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
好客租房移动Web（下）

目录 Contents

- 登录模块
- 我的收藏模块
- 发布房源模块
- 项目打包和优化

1. 登录模块

1.1 功能分析

业务：通过账号和密码完成登录，实现登录访问控制等。

功能：

- 用户登录
- 我的页面
- 封装路由访问控制组件

难点： 登录访问控制、表单验证。

1. 登录模块

1.2 用户登录

1. 分析页面结构和样式

① 复用 NavHeader 组件设置顶部导航栏。

② WingBlank 两翼留白组件。

③ WhiteSpace 上下留白组件。

1. 登录模块

1.2 用户登录

2. 登录功能

① 添加状态：username（账号） 和 password（密码）。
② 使用受控组件方式获取表单元素值。
③ 给 form 表单添加 onSubmit。
④ 创建方法 handleSubmit，实现表单提交。
⑤ 在方法中，通过 username 和 password 获取到账号和密码。
⑥ 使用 API 调用登录接口，将 username 和 password 作为参数。
⑦ 判断返回值 status 为 200 时，表示登录成功。
⑧ 登录成功后，将 token 保存到本地存储中（hkzf_token）。
⑨ 返回登录前的页面。

1. 登录模块

1.2 用户登录

3. 表单验证说明

- 表单提交前，需要先进行表单验证，验证通过后再提交表单。
- 方式一：antd-mobile 组件库的方式。（需要 InputItem 文本输入组件）。
- 推荐：使用更通用的 formik ， React 中专门用来进行表单处理和表单校验的库。

```
handleSubmit = () => {
  // 手动处理表单校验：
  // 1 非空校验
  // 2 账号格式校验
  // 3 密码格式校验
  // 4 表单校验失败样式处理
  // ...
}
```

1.3 formik 表单处理

1. 介绍

- 场景：表单处理，表单验证（比如，登录功能等）。
- Github 地址：formik（构建 React 表单，没有眼泪）。
- 优势：轻松处理 React 中的复杂表单，包括：获取表单元素的值、表单验证和错误信息、处理表单提交。并且将这些内容放在一起统一处理，有利于代码阅读、重构、测试等。
- 两种使用方式：1 高阶组件（withFormik） 2 render-props（<Formik render={() => {}} />）。

1.3 formik 表单处理

2. 使用 formik 重构登录功能

① 安装：yarn add formik。

② 导入 withFormik，使用 withFormik 高阶组件包裹 Login 组件。

③ 为 withFormik 提供配置对象：mapPropsToValues / handleSubmit。

④ 在 Login 组件中，通过 props 获取到 values（表单元素值对象）、 handleSubmit、handleChange。

⑤ 使用 values 提供的值，设置为表单元素的 value，使用 handleChange 设置为表单元素的 onChange。

⑥ 使用 handleSubmit 设置为表单的 onSubmit 。

⑦ 在 handleSubmit 中，通过 values 获取到表单元素值。

⑧ 在 handleSubmit 中，完成登录逻辑。

```
Login = withFormik({
  mapPropsToValues: () => ({ username: '' }),   // 提供表单项的值
  handleSubmit: (values, { props }) => {},      // 提供表单提交事件
})(Login)
```

1.3 formik 表单处理

3. 两种表单验证方式

- 两种方式：1 通过 validate 配置项手动校验 2 通过 validationSchema 配置项配合 Yup 来校验。
- 推荐： validationSchema 配合 Yup 的方式进行表单校验。

```
const validate = (values, props) => {
  let errors = {}
  if (!values.username) { errors.username = '账号为必填项' }
  // ...
  return errors
}
```

```
Yup.object().shape({
  username: Yup.string().required('账号为必填项')
})
```

1.3 formik 表单处理

4. 给登录功能添加表单验证

① 安装：yarn add yup （Yup 文档），导入 Yup。

② 在 withFormik 中添加配置项 validationSchema，使用 Yup 添加表单校验规则。

③ 在 Login 组件中，通过 props 获取到 errors（错误信息）和 touched（是否访问过，注意：需要给表单元素添加 handleBlur 处理失焦点事件才生效！）。

④ 在表单元素中通过这两个对象展示表单校验错误信息。

```
validationSchema: Yup.object().shape({
  username: Yup.string().required('账号为必填项')
    .matches(REG_UNAME, '长度为5到8位，只能出现数字、字母、下划线')
})
```

```
errors.username && touched.username && (
  <div className={styles.error}>{errors.username}</div>
)
```

1.3 formik 表单处理

5. 简化表单处理

① 导入 Form 组件，替换 form 元素，去掉 onSubmit。

```
<Form>...省略表单结构</Form>
```

② 导入 Field 组件，替换 input 表单元素，去掉 onChange、onBlur、value。

```
<Field type="text" name="username" placeholder="" className="" />
```

③ 导入 ErrorMessage 组件，替换原来的错误消息逻辑代码。

```
<ErrorMessage name="username" className="" component="div"  />
```

④ 去掉所有 props。

1. 登录模块

1.4 我的页面

1. 页面结构和样式

- Button 按钮组件（文档）。
- Grid 宫格组件（文档）。

1.4 我的页面

2. 功能分析

① 判断是否登录（本地缓存中是否有 hkzf_token，直接调用 isAuth() 方法即可）。

② 如果登录了，就发送请求获取个人资料，并且在页面中展示个人资料。

③ 如果没有登录，则不获取个人资料，只在页面中展示未登录信息。

④ 在页面中展示登录或未登录信息，就要通过 state 变化来体现，因此，需要一个表示是否登录的状态。

```
state = {
  isLogin: isAuth(),
  userInfo: {...}
}
```

1.4 我的页面

3. 功能实现

① 在 state 中添加两个状态：isLogin（是否登录）和 userInfo（用户信息）。
② 从 utils 中导入 isAuth（登录状态）、getToken（获取token）。
③ 创建方法 getUserInfo，用来获取个人资料。
④ 在方法中，通过 isLogin 判断用户是否登录。
⑤ 如果没有登录，则不发送请求，渲染未登录信息。
⑥ 如果已登录，就根据接口发送请求，获取用户个人资料。
⑦ 渲染个人资料数据。

1.4 我的页面

4. 退出功能

① 给退出按钮绑定单击事件，创建方法 logout 作为事件处理程序。

② 导入 Modal 对话框组件（文档）。

③ 在方法中，拷贝 Modal 组件文档中确认对话框的示例代码。

④ 修改对话框的文字提示。

⑤ 在退出按钮的事件处理程序中，先调用退出接口（让服务端退出），再移除本地token（本地退出）。

⑥ 将登陆状态 isLogin 设置为 false。

⑦ 清空用户状态对象。

1.5 登录访问控制

1. 概述

项目中的两种类型的功能和两种类型的页面：

两种功能：

- 登录后才能进行操作（比如：获取个人资料）
- 不需要登录就可以操作（比如：获取房屋列表）

两种页面：

- 需要登录才能访问 （比如：发布房源页）
- 不需要登录即可访问（比如：首页）

对于需要登录才能操作的功能使用 axios 拦截器进行处理（比如：统一添加请求头 authorization 等）

对于需要登录才能访问的页面使用路由控制

1.5 登录访问控制

2. 使用 axios 拦截器统一处理 token

① 在 api.js 中，添加请求拦截器。
② 获取到当前请求的接口路径（url）。
③ 判断接口路径，是否是以 /user 开头，并且不是登录或注册接口（只给需要的接口添加请求头）。
④ 如果是，就添加请求头 Authorization。
⑤ 添加响应拦截器。
⑥ 判断返回值中的状态码。
⑦ 如果是 400，表示 token 超时或异常，直接移除 token。

1.5 登录访问控制

3. 分析 AuthRoute 鉴权路由组件

- 场景：限制某个页面只能在登录的情况下访问。
- 说明：在 React 路由中并没有直接提供该组件，需要手动封装，来实现登录访问控制（类似于 Vue 路由的导航守卫）。
- 如何封装？参考 react-router-dom 文档中提供的鉴权示例 。
- 如何使用？使用 AuthRoute 组件代替默认的 Route 组件，来配置路由规则。
- AuthRoute 组件实际上就是对原来的 Route 组件做了一次包装，来实现了一些额外的功能。
- render 方法：render props 模式，指定该路由要渲染的组件内容（类似于 component 属性）。
- Redirect 组件：重定向组件，通过 to 属性，指定要跳转到的路由信息。
- state 属性：表示给路由附加一些额外信息，此处，用于指定登录成功后要进入的页面地址。

```
// 使用方式：
<AuthRoute path="/rent/add" component={Rent} />
```

1.5 登录访问控制

4. 封装 AuthRoute 鉴权路由组件

① 在 components 目录中创建 AuthRoute/index.js 文件。

② 创建组件 AuthRoute 并导出。

③ 在 AuthRoute 组件中返回 Route 组件（在 Route 基础上做了一层包装，用于实现自定义功能）。

④ 给 Route 组件，添加 render 方法，指定该组件要渲染的内容（类似于 component 属性）。

⑤ 在 render 方法中，调用 isAuth() 判断是否登录。

⑥ 如果登录了，就渲染当前组件（通过参数 component 获取到要渲染的组件，需要重命名）。

⑦ 如果没有登录，就重定向到登录页面，并且指定登录成功后要跳转到的页面路径。

⑧ 将 AuthRoute 组件接收到的 props 原样传递给 Route 组件（保证与 Route 组件使用方式相同）。

⑨ 使用 AuthRoute 组件配置路由规则，验证能否实现页面的登录访问控制。

1.5 登录访问控制

5. 修改登录成功跳转

① 登录成功后，判断是否需要跳转到用户想要访问的页面（判断 props.location.state 是否有值）。

② 如果不需要（没有值），则直接调用 history.go(-1) 返回上一页。

③ 如果需要，就跳转到 from.pathname 指定的页面（推荐使用 replace 方法模式，而不是 push）。

目录 Contents

- 登录模块
- 我的收藏模块
- 发布房源模块
- 项目打包和优化

2. 我的收藏模块

2.1 功能分析

业务：收藏房源

功能：

- 检查房源是否收藏
- 收藏房源

2. 我的收藏模块

2.2 检查房源是否收藏

实现步骤

① 在 state 中添加状态：isFavorite （表示是否收藏），默认值为 false。

② 创建方法 checkFavorite，在进入房源详情页面时调用该方法。

③ 先调用 isAuth 方法，来判断是否已登录。

④ 如果未登录，直接 return，不再检查是否收藏。

⑤ 如果已登录，从路由参数中，获取到当前房屋id。

⑥ 使用 API 调用接口，查询该房源是否收藏。

⑦ 如果返回状态码为 200 ，就更新 isFavorite；否则，不做任何处理（token过期）。

⑧ 在页面结构中，通过状态 isFavorite 修改收藏按钮的文字和图片内容。

2.3 收藏房源

实现步骤

1. 给收藏按钮绑定单击事件，创建方法 handleFavorite 作为事件处理程序。
2. 调用 isAuth 方法，判断是否登录。
3. 如果未登录，则使用 Modal.alert 提示用户是否去登录。
4. 如果点击取消，则不做任何操作。
5. 如果点击去登录，就跳转到登录页面，同时传递 state（登录后，再回到房源收藏页面）。
6. 根据 isFavorite 判断，当前房源是否收藏。
7. 如果未收藏，就调用添加收藏接口，添加收藏。
8. 如果已收藏，就调用删除收藏接口，删除收藏。

```
// push 方法第二个参数为：state，用于指定登录后要返回的页面
props.history.push('/login', {
  from: props.location
})
```

目录 Contents

- 登录模块
- 我的收藏模块
- 发布房源模块
- 项目打包和优化

3. 发布房源模块

3.1 功能演示和介绍

- 功能：获取房源的小区信息 、 房源图片上传 、房源发布等

3. 发布房源模块

3.2 模板改动说明

1. 修改首页（Index）去出租链接为：/rent/add。
2. 修改公共组件 NoHouse 的 children 属性校验为：node（任何可以渲染的内容）。
3. 修改公共组件 HousePackage，添加 onSelect 属性的默认值。
4. 添加 utils/city.js，封装当前定位城市 localStorage 的操作。
5. 创建了三个页面组件：Rent（已发布房源列表）、Rent/Add（发布房源）、Rent/Search（关键词搜索小区信息）。

3. 发布房源模块

3.3 配置三个页面的路由规则

实现步骤

1. 在 App.js 中导入 Rent 已发布房源列表页面。
2. 在 App.js 中导入 AuthRoute 组件。
3. 使用 AuthRoute 组件，配置路由规则。
4. 使用同样的方式，配置 Rent/Add 房源发布页面、Rent/Search 关键词搜索小区信息页面。
5. 给 Rent 组件的路由规则，添加 exact 属性（表示精确匹配模式）。

3.4 关键词搜索小区信息

1. 实现思路

- 获取 SearchBar 搜索栏组件的值（searchTxt）。
- 在搜索栏的 change 事件中，判断当前值是为空。
- 如果为空，直接 return，不做任何处理。
- 如果不为空，就根据当前输入的值以及当前城市id，获取该关键词对应的小区信息。
- 问题：搜索栏中每输入一个值，就发一次请求，如何解决？（对服务器压力大、用户体验不好）
- 解决方式：使用定时器（setTimeout）延迟执行（关键词：JS 文本框输入防抖）。

```
// 先清除定时器
clearTimeout(this.timerId)
// 开启定时器，延迟 500 毫秒发送请求。如果输入间隔小于 500 毫秒，就不会发送请求。
this.timerId = setTimeout(async () => {
  await API.get('url...')
}, 500)
```

3.4 关键词搜索小区信息

2. 实现步骤

① 给 SearchBar 组件，添加 onChange 配置项，获取文本框的值。

② 判断当前文本框的值是否为空。

③ 如果为空，清空列表，然后 return，不再发送请求。

④ 如果不为空，使用 API 发送请求，获取小区数据。

⑤ 使用定时器 setTimeout 来延迟搜索，提升性能。

3. 发布房源模块

3.4 关键词搜索小区信息

3. 传递小区信息给发布房源页面

① 给搜索列表项添加单击事件。

② 在事件处理程序中，调用 history.replace() 方法跳转到发布房源页面。

③ 将被点击的小区信息作为数据一起传递过去。

④ 在发布房源页面，判断 history.loaction.state 是否为空。

⑤ 如果为空，不做任何处理。

⑥ 如果不为空，则将小区信息存储到发布房源页面的状态中。

3. 发布房源模块

3.5 发布房源

1. 页面结构分析

① List 列表组件。
② InputItem 文本输入组件。
③ TextareaItem 多行输入组件。
④ Picker 选择器组件。
⑤ ImagePicker 图片选择器组件。

3.5 发布房源

2. 获取房源数据分析

- InputItem、TextareaItem、Picker 组件，都使用 onChange 配置项，来获取当前值。
- 处理方式：封装一个事件处理程序 getValue 来统一获取三种组件的值。

```
// name 表示要更新的状态
// value 表示当前输入值或选中值
getValue = (name, value) => {
  this.setState({
    [name]: value
  })
}
```

3.5 发布房源

2. 获取房源数据步骤

① 创建方法 getValue 作为三个组件的事件处理程序。

② 该方法接收两个参数：1 name 当前状态名 2 value 当前输入值或选中值。

③ 分别给 InputItem / TextareaItem / Picker 组件，添加 onChange 配置项。

④ 分别调用 getValue 并传递 name 和 value 两个参数（注意：Picker 组件选中值为数组，而接口需要字符串，所以，取索引号为 0 的值即可）。

```
this.getValue('roomType', value[0])
```

3.5 发布房源

3. 获取房屋配置数据

① 给 HousePackge 组件，添加 onSelect 属性。

② 在 onSelect 处理方法中，通过参数获取到当前选中项的值。

③ 根据发布房源接口的参数说明，将获取到的数组类型的选中值，转化为字符串类型。

④ 调用 setState() 更新状态。

3. 发布房源模块

3.5 发布房源

4. 房屋图片上传分析

- 根据发布房源接口，最终需要的是：房屋图片路径。
- 两个步骤：1 获取房屋图片 2 上传图片获取到图片路径（接口返回）。
- 如何获取房屋图片？ImagePicker 图片选择器组件，通过 onChange 配置项来获取。
- 如何上传房屋图片？根据图片上传接口，将图片转化为 FormData 数据后再上传，由接口返回图片路径。

3. 发布房源模块

3.5 发布房源

5. 获取房屋图片

① 给 ImagePicker 组件添加 onChange 配置项。

② 通过 onChange 的参数，获取到上传的图片，并存储到状态 tempSlides 中。

3.5 发布房源

6. 上传房屋图片

① 给提交按钮，绑定单击事件。
② 在事件处理程序中，判断是否有房屋图片。
③ 如果没有，不做任何处理。
④ 如果有，就创建 FormData 的实例对象（form）。
⑤ 遍历 tempSlides 数组，分别将每一个图片对象，添加到 form 中（键为： file，根据接口文档获得）。
⑥ 调用图片上传接口，传递form参数，并设置请求头 Content-Type 为 multipart/form-data。
⑦ 通过接口返回值获取到的图片路径。

3.5 发布房源

7. 发布房源

① 在 addHouse 方法中，从 state 里面获取到所有房屋数据。

② 使用 API 调用发布房源接口，传递所有房屋数据。

③ 根据接口返回值中的状态码，判断是否发布成功。

④ 如果状态码是 200，表示发布成功，就提示：发布成功，并跳转到已发布房源页面。

⑤ 否则，就提示：服务器偷懒了，请稍后再试~。

目录 Contents

- 登录模块
- 我的收藏模块
- 发布房源模块
- 项目打包和优化

4. 项目打包和优化

4.1 项目打包

① 打开 create-react-app 脚手架文档中的部署。
② 在根目录创建 .env.production 文件，配置生产环境的接口基础路径。
③ 在项目根目录中，打开终端。
④ 输入命令：yarn build，进行项目打包，生成 build 文件夹（打包好的项目内容）。
⑤ 将 build 目录中的文件内容，部署到服务器中即可。
⑥ 可以通过终端中的提示，使用 serve –s build 来本地查看（需要全局安装工具包 serve）。

4. 项目打包和优化

4.2 修改脚手架配置说明

create-react-app 中隐藏了 webpack 的配置，隐藏在 react-scripts 包中。

修改脚手架的 webpack 配置有两种方式：

- 运行命令 npm run eject 释放 webpack 配置（注意：不可逆操作）
- 通过第三方包重写 webpack 配置（比如：react-app-rewired 等）

4. 项目打包和优化

4.3 项目优化

1. antd-mobile 按需加载

① 打开 antd-mobile 在 create-react-app 中使用的文档。
② 安装 yarn add react-app-rewired customize-cra（用于重写脚手架配置）。
③ 修改 package.json 中的 scripts。
④ 在项目根目录创建文件：config-overrides.js（用于覆盖脚手架默认配置）。
⑤ 安装 yarn add babel-plugin-import 插件（用于按需加载组件代码和样式）。
⑥ 修改 config-overrides.js 文件，配置按需加载功能。
⑦ 重启项目（yarn start）。
⑧ 移除 index.js 中导入的 antd-mobile 样式文件。
⑨ 将 index.css 移动到 App 后面，让 index.css 中的页面背景色生效。

4. 项目打包和优化

4.3 项目优化

2. 基于路由的代码分割

- 目的：将代码按照路由进行分割，只在访问该路由时才加载该组件内容，提高首屏加载速度。
- 如何实现？React.lazy() 方法 + import() 方法 、Suspense 组件（React Code-Splitting文档）。
- React.lazy() 作用：处理动态导入的组件，让其像普通组件一样使用。
- import('组件路径') 作用： 告诉 webpack ，这是一个代码分割点，进行代码分割。
- Suspense 组件：用来在动态组件加载完成之前，显示一些 loading 内容，需要包裹动态组件内容。

```
const CityList = React.lazy(() => import('./pages/CityList'))
```

```
<Suspense fallback={<div>loading...</div>}>
  <Route path="/citylist" component={CityList} />
</Suspense>
```

4. 项目打包和优化

4.3 项目优化

3. 其他优化和说明

- React.js 优化性能文档。
- react-virtualized 只加载用到的组件（文档）。
- 脚手架配置代理解决跨域问题文档。

好客租房移动 Web（下）

1. 登录模块：使用 Fomik 组件实现了表单处理和表单校验、封装鉴权路由 AuthRoute 和 axios 拦截器实现登录访问控制。
2. 我的收藏模块：添加、取消收藏。
3. 发布房源模块：小区关键词搜索、图片上传、发布房源信息。
4. 项目打包和优化：antd-mobile 组件库按需加载、基于路由的代码分割实现组件的按需加载，提高了首屏加载速度。

传智播客旗下高端IT教育品牌