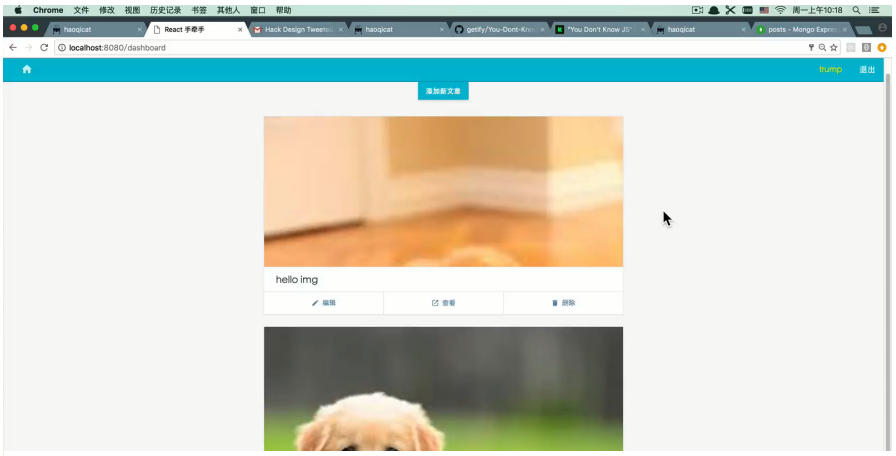

React 组件连接
React Store 分发
Redux Action

前端导航栏体现用户登录状态

前端实现用户退出登录功能

显示文章详情

服务器端编写获取单篇文章的接口

修改 `server/routes.js` 文件，编写 GET `/posts/:post_id` 接口，如下：

```
app.get('/posts/:post_id', function(req, res)
  Post.findById({_id: req.params.post_id}, fu
    if (err) return res.status(422).json({err
    res.json({ post: post })
  })
})
```

HTTP 状态码422（不可处理实体）意思是说服务器不能处理客户端的请求，虽然客户端的请求无误，适

合服务器端数据验证失败的情况。

前端添加 ShowPost 组件路由

打开 `src/routes.js` 文件，导入 ShowPost 组件：

```
import ShowPost from './ui/posts/ShowPost';
```

添加组件对应的路由：

```
<Route path='/posts/:post_id' component={Show
```

前端编写 ShowPost 组件

新建 `ui/posts/ShowPost.js` 文件，用来展示文章详情，代码如下：

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import { getPost } from '../../redux/actions/
import { Settings } from '../../settings';
import isEmpty from 'lodash/fp/isEmpty';

class ShowPost extends Component {
  componentWillMount() {
    this.props.getPost(this.props.params.post
  }

  render() {
    const styles = {
      cover: {
```

```
      backgroundImage: isEmpty(this.props.
      height: '500px',
      position: 'relative',
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat',
      backgroundPosition: 'center center'
    },
    container: {
      maxWidth: '960px',
      margin: '56px auto 0'
    },
    name: {
      fontSize: '28px',
      lineHeight: '28px',
      color: '#2e4453',
      paddingBottom: '48px'
    },
    content: {
      color: '#666'
    }
  }

  return (
    <div>
      <div style={styles.cover}></div>
      <div style={styles.container}>
        <div style={styles.name}>{this.prop
        <div style={styles.content}>{this.p
      </div>
    </div>
  );
 }
}

ShowPost.propTypes = {
  post: React.PropTypes.object.isRequired
```

```
}

const mapStateToProps = (state) => ({
  post: state.post
})

export default connect(mapStateToProps, { get
```

因为从服务器端获取的单篇文章的信息只是在 ShowPost 组件中使用，所以把单篇文章的信息保存到 Redux store 中，或者直接在 ShowPost 组件内部定义一个 state 存储文章信息都是可以的。不过，使用 store 保存文章信息，可以把数据和展示组件分离开，代码结构更清晰。另外，当更新文章的时候，也可以调用 `getPost` 创建函数请求服务器端的 GET `/posts/:post_id` 接口，这样代码就能复用。

编写异步 Action 创建函数 getPost

修改 `redux/actions/postActions.js` 文件，添加代码：

```
export function getPost(id) {
  return (dispatch) => {
    axios.get(`${Settings.host}/posts/${id}`)
      dispatch({ type: 'LOAD_POST', post: res
    }).catch(error => {
      handleError(error);
    });
  }
}
```

定义与操作单篇文章相关的 Reducer

新建 `redux/reducers/post.js` 文件，添加代码：

```
export default (state = {}, action = {}) => {
  switch(action.type) {
    case 'LOAD_POST':
      return action.post;
    default:
      return state;
  }
}
```

合并 Reducer

修改 `redux/rootReducer.js` 文件，把刚才定义的匿名 reducer 导入为名为 `post` 的 reducer:

```
import post from './reducers/post';
```

然后，把 `post` reducer 和其它的 reducer 一起合并为一个根级 reducer，构建一个状态树：

```
export default combineReducers({
  ...
  post
});
```

添加文章详情页入口

修改 PostActionList 组件，更改 查看 标签的链接：

```
<Link to={`/posts/${this.props.post._id}`} st
  ...
  <span>查看</span>
</Link>
```

另外，需要让普通用户也可以浏览文章详情页，修改 PostItem 组件，导入 Link 组件：

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

给文章标题添加链接，让其指向文章详情页：

```
<Link to={`/posts/${this.props.post._id}`} st
  {this.props.post.name}
</Link>
```

至此，就可以到浏览器中查看文章详细信息了。

欢迎添加 Peter 的微信：happypeter1983

冀ICP备15007992号-3