基于React Hooks与Parse实现用户鉴权的最佳实践（useAuth）



所有React App最常见也是最难处理的场景就是如何把状态从父节点传递到子节点，一个常见的解决方案是Redux，但总觉得在React Native上使用Redux有点重，而React16.8以后提供的Hooks API无疑成了更好的选择。

看代码：

核心的Hook代码use-auth.js，这里封装了Parse登陆注册等一系列方法，如果你是自建Server或其他服务，直接替换相关代码即可：

// Hook(use-auth.js)

import React , { useState, useEffect, useContext, createContext } from ‘react’;

import Parse from ‘parse/react-native’;

import AsyncStorage from ‘@react-native-community/async-storage’;

const setupParse = () => {

Parse.initialize(‘your-app-id’);

Parse.serverURL = ‘http://your-serverurl/parse’;

Parse.setAsyncStorage(AsyncStorage);

};

setupParse();

const authContext = createContext();

// Provider component that wraps your app and makes auth object…

// …available to any child component that calls useAuth().

export function ProvideAuth({children}){

const auth = useProvideAuth();

return <authContext.Provider value={auth}>{children}</authContext.Provider>;

}

// Hook for child components to get the auth object…

// …and re-render when it changes.

export const useAuth = () => {

return useContext(authContext);

};

// Provider hook that creates auth object and handles state

function useProvideAuth(){

const [user, setUser] = useState(null);

// Wrap any Parse methods we want to use making sure …

// … to save the user to state.

const signin = (username, password) => {

return Parse.User.logIn(username, password)

.then(currentUser);

return currentUser;

};

const signup = () => {

return Parse.User

.logOut()

.then(()=>setUser(false));

};

}