lesson2 导航器使用与鉴权流程

课堂目标

1. 掌握React Navigation
2. 掌握RN不同导航类型：stack、tab、drawer
3. 掌握鉴权

资源

1. 课堂代码地址(不要忘记切分支)
2. react native知识图谱
3. react-native
4. react-native中文
5. react native elements
6. React Navigation

知识要点

开始

```
npx react-native init lesson2
cd lesson2
yarn ios
yarn android
```

介绍

在web端，我们可以使用a标签导向不同的页面。当用户点击一个link，对应的URL就被push到浏览器的历史栈中，当用户点击后退按钮，浏览器再把历史栈中的这个URL弹出(pop)。这个时候，当前活跃的页面就是上次访问的页面了。不过React Native并没有浏览器那样的历史栈，所以React Navigation就来做这件事了。

- `react-native` >= 0.63.0

```
yarn add @react-navigation/native react-native-screens
react-native-safe-area-context
```

> 注意：执行yarn之后，直接执行yarn ios会提示原生相关的一些错误，这个时候需要进入ios文件夹，再次执行`pod install`安装ios相关的依赖：
>
> ```
> cd ios
> pod install
> ```
>
> 然后再在lesson2下执行yarn ios。

导航方式

React Navigation包含以下功能来帮助你创建导航器:

- StackNavigator
- TabNavigator
- DrawerNavigator

Stack navigation

```
yarn add @react-navigation/native-stack
```

基本使用

`createNativeStackNavigator`是一个函数，返回一个对象，这个对象里有两个都是React组件的属性：`Screen` and `Navigator`。注意它们的包裹关系：NavigationContainer->Navigator->Screen。

Navigator属性：

- `initialRouteName`：第一个要渲染的页面的name。
- `screenOptions`：定义子Screen的默认option。

Screen属性：

- `name`：string，标记页面，此值须唯一。
- `component`：要渲染的组件。(同react-router中Route的path以及component)
- `options`: object。比如默认情况下header显示的是name，也可以通过options的title修改。
- `initialParams`给组件参数

App.js

```
import React from 'react';
import {View, Text} from 'react-native';
import {NavigationContainer} from '@react-
navigation/native';
import RootRouter from './RootRouter';

export default function App() {
  return (
    <NavigationContainer>
      <RootRouter />
    </NavigationContainer>
  );
}
```

RootRouter

```
import React from 'react';
import {View, Text} from 'react-native';
import {createNativeStackNavigator} from '@react-
navigation/native-stack';
import HomeScreen from '@/screens/HomeScreen';
import UserScreen from '@/screens/UserScreen';

const {Screen, Navigator} =
createNativeStackNavigator();

export default function RootRouter() {
  return (
    //   ! SafeAreaView不能写这里
    // <SafeAreaView>
    <Navigator>
      <Screen name="home" component={HomeScreen} />
      <Screen
```

```
          name="user"
          component={UserScreen}
          options={{title: '用户中心'}}
          initialParams={{name: '我是小明'}}
        />
      </Navigator>
      // </SafeAreaView>
    );
}
```

> 注意: Screen的 `component` 是组件。不要传一个内联函数 (e.g. `component={() => <HomeScreen />}`), 不然会导致组件无法复用。

页面跳转

Link跳转

```
<Link to={{screen: 'user', params: {id: 'jane'}}}>
  Go to Jane's profile
</Link>
```

事件跳转

navigation属性：https://reactnavigation.org/docs/navigation-prop/

每一个`Screen`组件都有一个`navigation`的属性。使用navigation.navigate跳转的时候，注意这个跳转的name一定得是Navigator里声明过的，不然开发环境会报错，而生产环境则没反应。

```
<Button
  title="go home"
  onPress={() => {
    // navigation.push('home');
    // navigation.navigate('home');
  }}
  />
```

注意：页面是否已经存在，执行navigation.navigate导致的结果是不同的。如果页面已经存在，执行navigation.navigate则会回到原先的页面，并且移除原先页面在历史栈中的记录。如果页面不存在，则push一个新的记录到历史栈中。

但是如果你执行navigation.navigate的时候，给页面添加了参数，那对与历史栈来说，这个记录对于历史栈就是个新的记录了。

快速回到首页

```
<Button title="回到首页" onPress={() =>
navigation.popToTop()} />
```

页面参数修改

`navigation.setParams` 允许我们更新当前页面的`route.params`，类似`React.setState`。

```
<Button
  onPress={() => navigation.setParams({username: '小
花'})}
  title="更改params"
 />
```

useNavigation

`useNavigation`是一个传回`navigation`的自定义hook。当你不想从组件拿到`navigation`或者因为组件嵌套层级过多不方便的时候，这个api就非常有用了

```
import {useNavigation} from '@react-navigation/native';
import Section from '@/components/Section';
import {Link} from '@react-navigation/native';
import {Button} from 'react-native-elements';

// export default function SettingScreen({navigation})
{

export default function SettingScreen() {
  const navigation = useNavigation();

  return (
    <View>
      <Section>SettingScreen</Section>
      <Link to={{screen: 'home'}}>go home</Link>
      <Button
        title="go home"
        onPress={() => {
          // navigation.push('home');
          // navigation.navigate('home')
          navigation.popToTop();
        }}
      />

      <Button
        title="back修改UserScreen的params"
```

```
        onPress={() => navigation.navigate('user',
{username: '小白'})}
      />
    </View>
  );
}
```

给组件传其他属性

有时候我们可能想在Screen这里给页面传参数，这里有以下两种办法：

1. 用React Context，但是注意这个时候的Provider得包在Navigator外，而不是Screen。因为Navigator的子组件只能是Screen。(推荐使用)
2. 不用`component`，而是使用函数，以此给渲染组件添加props属性。但是不要忘记上一个注意事项，这个时候我们需要给HomeScreen加上`React.memo` or `React.PureComponent` 以优化，不然会导致不必要的渲染。

   ```
   <Screen name="Home">
     {props => <HomeScreen {...props} extraData=
   {someData} />}
   </Screen>
   ```

Tab navigation

文档：https://reactnavigation.org/docs/tab-based-navigation

大部分手机app里都有的tabs，位置自己定义，一般位于底部。

```
yarn add @react-navigation/bottom-tabs
```

实例：

```
import React from 'react';
import {View, Text} from 'react-native';
import {createBottomTabNavigator} from '@react-
navigation/bottom-tabs';
import {createNativeStackNavigator} from '@react-
navigation/native-stack';
import HomeScreen from '@/screens/HomeScreen';
import UserScreen from '@/screens/UserScreen';
import SettingScreen from '@/screens/SettingScreen';

const {Screen, Navigator} = createBottomTabNavigator();
// const {Screen, Navigator} =
createNativeStackNavigator();

export default function RootRouter() {
  return (
    <Navigator
      initialRouteName="home"
      screenOptions={{
        title: '默认设置所有的title',
        // headerBackTitle: '返回',
        headerShown: true,
        headerStyle: {
          backgroundColor: 'orange',
        },
      }}>
      <Screen
        name="user"
        component={UserScreen}
```

```
        options={{title: '用户中心', headerBackTitle: '用户中心的back'}}
      />
      <Screen name="home" component={HomeScreen} options={{title: '首页'}} />
      <Screen
        name="setting"
        component={SettingScreen}
        options={{title: '设置'}}
      />
    </Navigator>
  );
}
```

Drawer navigation

鉴权

结合redux

创建store，建立状态仓库

```
import {createStore, combineReducers, applyMiddleware}
from 'redux';
import {loginReducer} from './loginReducer';

// 创建一个状态管理库
const store = createStore(
  combineReducers({user: loginReducer}),
);

export default store;
```

创建reducer，定义修改规则

loginReducer

```
import {REQUEST, LOGIN_SUCCESS, LOGOUT_SUCCESS,
LOGIN_FAILURE} from './const';

const userInit = {
  isLogin: false,
  userInfo: {id: null, name: '', score: 0},
  loading: false, // loading
  err: {msg: ''},
};

// 定义用户基本信息修改规则
export const loginReducer = (state = {...userInit},
{type, payload}) => {
  switch (type) {
    case REQUEST:
      return {...state, loading: true};
    case LOGIN_SUCCESS:
```

```
      return {...state, isLogin: true, loading: false,
userInfo: {...payload}};
    case LOGIN_FAILURE:
      return {...state, ...userInit, ...payload};
    case LOGOUT_SUCCESS:
      return {...state, isLogin: false, loading:
false};
    default:
      return state;
  }
};
```

const

```
export const REQUEST = 'REQUEST';
export const LOGIN_SUCCESS = 'LOGIN_SUCCESS';
export const LOGIN_FAILURE = 'LOGIN_FAILURE';
export const LOGOUT_SUCCESS = 'LOGOUT_SUCCESS';

export const LOGIN_SAGA = 'LOGIN_SAGA';
```

action/user

```
import {LOGIN_SUCCESS, LOGOUT_SUCCESS} from
'@/store/const';

export const login = () => ({type: LOGIN_SUCCESS});
export const logout = () => ({type: LOGOUT_SUCCESS});
```

把store放在组件最顶层，通过Provider跨层级传递

```
import store from '@/store/';

export default function App() {
  return (
    <Provider store={store}>
      <NavigationContainer>
        <RootRouter />
      </NavigationContainer>
    </Provider>
  );
}
```

登录

```
import React from 'react';
import {View, Text} from 'react-native';
import Section from '@/components/Section';
import {Button} from 'react-native-elements';
import {useDispatch, useSelector} from 'react-redux';
import {login} from '@/action/user';

export default function LoginScreen() {
  const user = useSelector(({user}) => user);
  const {isLogin, userInfo} = user;

  const dispatch = useDispatch();

  return (
    <View>
      <Section>LoginScreen</Section>
      <Text>{JSON.stringify(isLogin)}</Text>
```

```
      <Button
        onPress={() =>
          dispatch({type: 'LOGIN_SUCCESS', payload:
{name: '小明'}})
        }
        title="login"
      />
    </View>
  );
}
```

显示用户信息

```
import React, {useEffect} from 'react';
import {View, Text} from 'react-native';
import Section from '@/components/Section';
import {Link} from '@react-navigation/native';
import {Button} from 'react-native-elements';
import {login} from '@/action/user';
import {useDispatch, useSelector} from 'react-redux';

export default function UserScreen({navigation, route})
{
  const user = useSelector(({user}) => user);
  const {isLogin, userInfo} = user;
  const dispatch = useDispatch();

  return (
    <View>
      <Section>UserScreen</Section>
      {/* <Section>Home传过来的params：{route?.params.?
username}</Section> */}
```

```
      {/* <Button
        onPress={() => navigation.setParams({username:
'小花'})}
        title="更改params"
      />
      <Link to={{screen: 'setting'}}>go setting</Link>
*/}

      {isLogin ? (
        <Button
          onPress={() => dispatch({type:
'LOGOUT_SUCCESS'})}
          title={userInfo.name + 'logout'}
        />
      ) : (
        <Button
          onPress={() =>
            dispatch({type: 'LOGIN_SUCCESS', payload:
{name: '小明'}})
          }
          title="login"
        />
      )}
    </View>
  );
}
```

所有页面均需要登录权限

```
import React from 'react';
import {View, Text} from 'react-native';
import Section from '@/components/Section';
```

```
import {createNativeStackNavigator} from '@react-
navigation/native-stack';
import {createBottomTabNavigator} from '@react-
navigation/bottom-tabs';
import HomeScreen from '../screens/HomeScreen';
import UserScreen from '@/screens/UserScreen';
import SettingScreen from '@/screens/SettingScreen';
import {useSelector} from 'react-redux';
import LoginScreen from '@/screens/LoginScreen';
import HomeStackScreen from './HomeStackScreen';

// const {Navigator, Screen, Group} =
createNativeStackNavigator();
const {Navigator, Screen, Group} =
createBottomTabNavigator();

export default function RootRouter() {
  const user = useSelector(({user}) => user);
  const {isLogin} = user;

  return (
    <Navigator
      initialRouteName="home"
      screenOptions={{
        headerStyle: {backgroundColor: 'orange'},
        headerBackTitle: '返回',
        // title: '开课吧',
      }}>
      {isLogin ? (
        <Group>
          <Screen
            name="home"
            // component={HomeScreen}
            component={HomeStackScreen}
            options={{headerShown: false}}
```

```
            />
            <Screen
              name="user"
              component={UserScreen}
              options={{title: '用户中心'}}
            />
            <Screen
              name="setting"
              component={SettingScreen}
              options={{title: '设置'}}
            />
          </Group>
        ) : (
          <Group>
            <Screen
              name="home"
              component={HomeScreen}
              options={{title: '首页'}}
            />
            <Screen
              name="login"
              component={LoginScreen}
              options={{title: '登录'}}
            />
          </Group>
        )}
      </Navigator>
    );
}
```

部分页面需要登录权限

```
<Group>
  <Screen
    name="home"
    component={HomeScreen}
    options={{title: '首页'}}
    />
  <Screen
    name="login"
    component={LoginScreen}
    options={{title: '登录'}}
    />
</Group>
```

导航嵌套

```
import React from 'react';
import {createNativeStackNavigator} from '@react-
navigation/native-stack';
import HomeScreen from '@/screens/HomeScreen';
import MovieScreen from '@/screens/MovieScreen';

const HomeStack = createNativeStackNavigator();

export default function HomeStackScreen() {
  return (
    <HomeStack.Navigator>
      <HomeStack.Screen
        name="home"
        component={HomeScreen}
        options={{title: '首页'}}
      />
```

```
      <HomeStack.Screen name="movie" component=
{MovieScreen} />
    </HomeStack.Navigator>
  );
}
```

最佳实践

少点嵌套。

回顾

作业

"隐藏"MovieScreen的TabBar:

文档地址：https://reactnavigation.org/docs/hiding-tabbar-in-screens