MOTC 風景React App

**姓名**: 許鐸鐙

**學校**: 虎尾科技大學

**手機**: 0909050645

目錄

**一、建置開始2**

安裝套件2

整體架構2

頁面component2

數據state2

無限滾動渲染頁面3

**二、react-router-dom理解介紹5**

react-router-dom使用5

**三、redux理解介紹6**

react-redux使用6

**四、axios理解介紹8**

axios使用8

**五、gh-pages配置10**

**六、結論11**

**七、參考文獻12**

1. **建置開始**

**create-react-app motc**

**套件安裝**

**Npm install axios immutable react-immutable react-redux react-router-dom react-thunk redux redux-immutable redux-thunk styled-components gh-pages**

**整體架構**

|  |
| --- |
| * **Src**   + **Common(共用的UI)**     - **Header**   + **Pages(會有變化的UI)**     - **Home**     - **ScenicSpot**       * **Components**       * **store(數據分部)**     - **ScenicSpotCity**       * **Components**       * **Store(數據分部)**   + **Store(數據總部)**   + **App** |

**頁面component**

|  |
| --- |
| * **Home** * **ScenicSpot**   + **SpotList** * **ScenicSpotCity**   + **CityNavbar**   + **SpotCityList** * **common/header** |

**數據state**

|  |
| --- |
| * **state** * **scenicSpot**   + **scenicSpotList:[]**   + **loading**   + **hasMore** * **spotCity**   + **City**     - **{CityName,CityNameEN,CityList:[],hasMore,loading} 全部縣市** |

**無限滾動渲染頁面**

**實作步驟:**

1. **預設loading: false, hasMore: true**
2. **選取全部或各縣市按鈕拿取資料axios.get(url,$top:30)前30筆**
3. **紀錄資料的最後一筆的ref**
4. **判斷是否滾到最後一筆資料(loading)以及是否還有資料(hasMore)**
5. **再次呼叫axios.get(url,$top:30,$skip:index)跳過已有的資料讀取30筆**

**實作程式:**

|  |
| --- |
| **import { useRef, useCallback } from 'react';**  **調用useRef,useCallback方法**  **function SpotList(props) {**  **將父親的state Object 的值解賦出來三個一個數組兩個布林值兩個方法**  **const { scenicSpotList, hasMore, loading, handleloading, handlehasMoreList } = props;**  **scenicSpotList是immutable數組所以要轉成JS數組**  **const list = scenicSpotList.toJS();**  **使用react 裡的API觀察者使用的ref在開始render的時候**  **const observer = useRef()**  **紀錄最後一個ref值並回調它**  **const lastElementRef = useCallback((node) => {**  **如果觀察者目前有值，取消連接觀察者**  **if(observer.current) observer.current.disconnect()**  **使用web API IntersectionObserver 建立一個物件**  **observer.current = new IntersectionObserver(entries => {**  **判斷是否有ref進入**  **if(entries[0].isIntersecting){**  **將scenicSpot 裡的loading變成true**  **handleloading()**  **拿取後30筆資料**  **handlehasMoreList(list, hasMore, list.length, loading);**  **}**  **})**  **如果有最後一個ref ，觀察者觀察最後一個ref**  **if(node) observer.current.observe(node)**  **})**  **UI部分**  **return (**  **list.map((item, index)=>{**  **if(list.length === index + 1){**  **return (**  **<SpotListCard ref={lastElementRef}>**  **</SpotListCard>**  **)**  **} else{**  **return (**  **<SpotListCard>**  **</SpotListCard>**  **)**  **}**  **})**  **)**  **}**  **資料讀取路口**  **const mapState = (state) => {**  **return {**  **使用immutable getIn()方法拿取資料，而不使用get()方法一層一層的拿取**  **scenicSpotList: state.getIn(['scenicSpot','scenicSpotList']),**  **loading: state.getIn(['scenicSpot','loading']),**  **hasMore: state.getIn(['scenicSpot','hasMore'])**  **}**  **}**  **資料改變出口**  **const mapDispatch = (dispatch) => {**  **return {**  **handleloading(){**  **dispatch(ActionCreators.loading())**  **},**  **handlehasMoreList(prevState, hasMore, index, loading){**  **dispatch(ActionCreators.getMoreSpotList(prevState, hasMore, index, loading))**  **}**  **}**  **}**  **export default connect(mapState, mapDispatch)(SpotList)** |

1. **react-router-dom理解介紹**

**BrowserRouter:包裹在所有component外圍**

**Link:放置在各別component的需要轉址按鈕或a連結to=’網址街口’**

**Route :各路由配置轉址，path=’網址街口’，exact完全吻合街口，component配置渲染{component}**

**useParams():回傳一個Object(key,value) path:’/:key’value=網址上的值**

**react-router-dom使用**

|  |
| --- |
| **<BrowserRouter>**  **<Header />**  **<Route path='/' exact component={home}></Route>**  **<Route path='/scenicSpot' exact component={ScenicSpot}></Route>**  **<Route path='/scenicSpot/:cityname' exact component={ScenicSpotCity}></Route>**  **</BrowserRouter>** |

1. **redux理解介紹**

**簡介:一個資料處理方式，重要名詞store、state、action、reducer**

**store:整個資料結構的入口**

**state:資料**

**action:描述發生什麼事的物件**

**reducer:pure function，它取得先前的state和一個action，並回傳一個新的state物件。**

**(這次的作業我使用了react-redux更快速方便實作)**

**react-redux使用**

**App.js**

|  |
| --- |
| **<Provider store={store}>**  **</Provider>** |

**Store/index.js**

|  |
| --- |
| **import { createStore, applyMiddleware, compose } from 'redux';**  **import thunk from 'redux-thunk';**  **import reducer from './reducer';**  **redux除錯工具**  **const composeEnhancers = window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose;**  **創建一個store，並且使用thunk中間件**  **const store = createStore(reducer, composeEnhancers(applyMiddleware(thunk)));**  **export default store** |

**Store/reducer.js**

|  |
| --- |
| **import { combineReducers } from 'redux-immutable';**  **import { reducer as scenicSpotReducer } from '../pages/ScenicSpot/store';**  **import { reducer as SpotCityReducer } from '../pages/ScenicSpotCity/store';**  **將兩個各別的reducer結合起來成為一個**  **const reducer = combineReducers({**  **scenicSpot: scenicSpotReducer,**  **spotCity: SpotCityReducer**  **})**  **export default reducer** |

**Pages/XXX/store/reducer.js**

|  |
| --- |
| **import { fromJS } from 'immutable';**  **import \* as ActionTypes from './ActionTypes';**  **初始值並改成immutable格式**  **const defaultState = fromJS({**  **XXX**  **})**  **const reducer = (state=defaultState,action) =>{**  **判斷發生事件的類型，符合就用新數據覆蓋舊數據**  **switch(action.type) {**  **case ActionTypes.XXX:**  **return state.set('XXX',action.data)**  **default:**  **return state**  **}**  **}**  **export default reducer** |

**Component與react-redux使用大致配置**

|  |
| --- |
| **import React, { Component } from 'react';**  **連接store**  **import { connect } from 'react-redux';**  **class Name extends Component {**    **render(){**  **const { componentParmeter, componentMethod } = this.props;**  **return (**  **<>**  **<div onClick={ componentMethod }>{ componentParmeter }</div>**  **</>**  **)**  **}**  **}**  **const mapState = (state) =>{**  **return {**  **componentParmeter: state.getIn([' storeParentParmeter ', ‘storeChildParmeter’])**  **}**  **}**  **Const mapDispatch = (dispatch)=>{**  **Return {**  **ComponentMethod(){**  **dispatch(ActionCreators.ReducerMethod())**  **}**  **}**  **}**  **export default connect(mapState, mapdispatch)(Name)** |

1. **axios理解介紹**

**簡介:方便做到get,put,post,patch…的請求**

* 1. **axios使用**

**全部沒有數據時**

|  |
| --- |
| **if(prevState.length === 0){**  **axios.get('https://ptx.transportdata.tw/MOTC/v2/Tourism/ScenicSpot',{params: {$top: 30, $format:'JSON'}})**  **.then((res)=>{**  **dispatch(changeList(res.data));**  **}).catch(()=>{**  **console.log('error');**  **})**  **}** |

**全部加載更多數據**

|  |
| --- |
| **if(hasMore && loading){**  **axios.get('https://ptx.transportdata.tw/MOTC/v2/Tourism/ScenicSpot',{params: {$top: 30, $skip: index, $format:'JSON'}})**  **.then((res)=>{**  **const moreData = [...prevState,...res.data]**  **if(res.data.length === 0 && hasMore){**  **dispatch(hasNoMore())**  **}else{**  **dispatch(changeList(moreData));**  **}**  **dispatch(loaded());**  **}).catch(()=>{**  **console.log('error');**  **})**  **}** |

**縣市沒有數據**

|  |
| --- |
| **if(prevState.length === 0){**  **axios.get(`https://ptx.transportdata.tw/MOTC/v2/Tourism/ScenicSpot/${city}`**  **,{ params: {$top: 30, $format:'JSON'}})**  **.then((res)=>{**  **dispatch(changeList(res.data, cityIndex));**  **}).catch(()=>{**  **console.log('error');**  **})**  **}** |

**縣市加載更多數據**

|  |
| --- |
| **if(hasMore && loading){**  **axios.get(`https://ptx.transportdata.tw/MOTC/v2/Tourism/ScenicSpot/${city}` ,{params: {$top: 30, $skip: index, $format:'JSON'}})**  **.then((res)=>{**  **const moreData = [...prevState,...res.data]**  **if(res.data.length === 0){**  **dispatch(hasNoMore(cityIndex))**  **}else{**  **dispatch(changeList(moreData, cityIndex));**  **}**  **dispatch(loaded(cityIndex));**  **}).catch(()=>{**  **console.log('error');**  **})**  **}** |

1. **gh-pages配置**

**package.json**

|  |
| --- |
| **{**  **"homepage": "http://vincent19961112.github.io/MOTC",**  **"dependencies": {**  **……**  **"gh-pages": "^3.1.0",**  **},**  **"scripts": {**  **…..**  **"predeploy": "npm run build",**  **"deploy":"gh-pages -d build"**  **},**  **}** |

**App.js**

|  |
| --- |
| **<BrowserRouter basename={ process.env.PUBLIC\_URL }>**  **<Header />**  **<Route path='/' exact component={home}></Route>**  **<Route path='/scenicSpot' exact component={ScenicSpot}></Route>**  **<Route path='/scenicSpot/:cityname' exact component={ScenicSpotCity}></Route>**  **</BrowserRouter>** |

1. **結論**

**這次的作業能做出來，其實自己挺意外的，因為我對react不是很熟練，慶幸的是我從中學會了很多新觀念redux，如果沒有這次的活動消息，我或許又會沒有方向的亂做練習，因為自己常常會被一些酷炫的資源吸引，但我發現如果我沒有好的底子，再好的資源，我也沒辦法很好的應用，藉由這次的活動，我跟著(Jomy King) youtube影片做練習才了解使用react與redux的，雖然不知道自己還有多少東西需要補強，但這次有不少的收穫。**

1. **參考文獻**

**React 16.x教程React系列课程从零基础到项目开发实战**

[**https://www.youtube.com/watch?v=ZXg4W7qm-uw&list=PL9nxfq1tlKKnza3MPogWqaYIPtdW\_G2lF**](https://www.youtube.com/watch?v=ZXg4W7qm-uw&list=PL9nxfq1tlKKnza3MPogWqaYIPtdW_G2lF)

**Web Dev Simplified-Infinite Scrolling With React – Tutorial**

<https://www.youtube.com/watch?v=NZKUirTtxcg>

**麥克的半路出家筆記**

<https://medium.com/麥克的半路出家筆記/認識-intersection-observer-api-實作-lazy-loading-和-infinite-scroll-c8d434ad218c>

**手寫筆記**

<https://medium.com/手寫筆記/a-little-bit-of-react-router-dom-e5b809fcb127>

阮一峰的網絡日誌

<https://www.ruanyifeng.com/blog/2016/09/redux_tutorial_part_three_react-redux.html>

React 官網

<https://zh-hant.reactjs.org/>

immutable 官網

<https://immutable-js.github.io/immutable-js/>

redux 官網

<https://chentsulin.github.io/redux/>