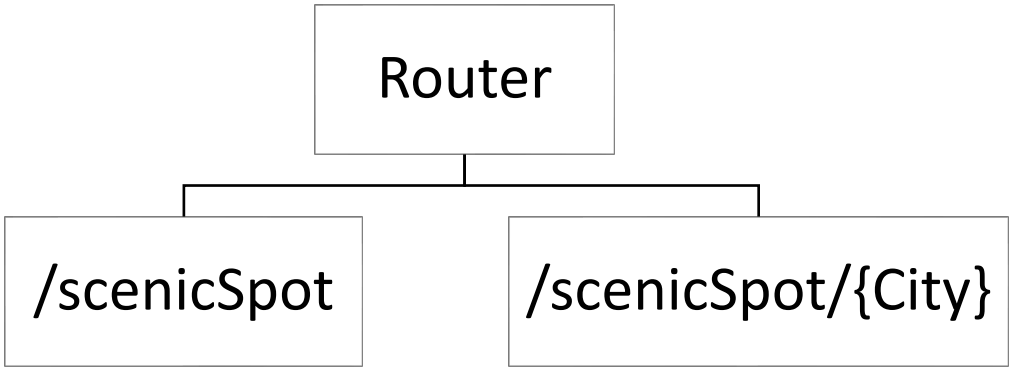
解釋文件

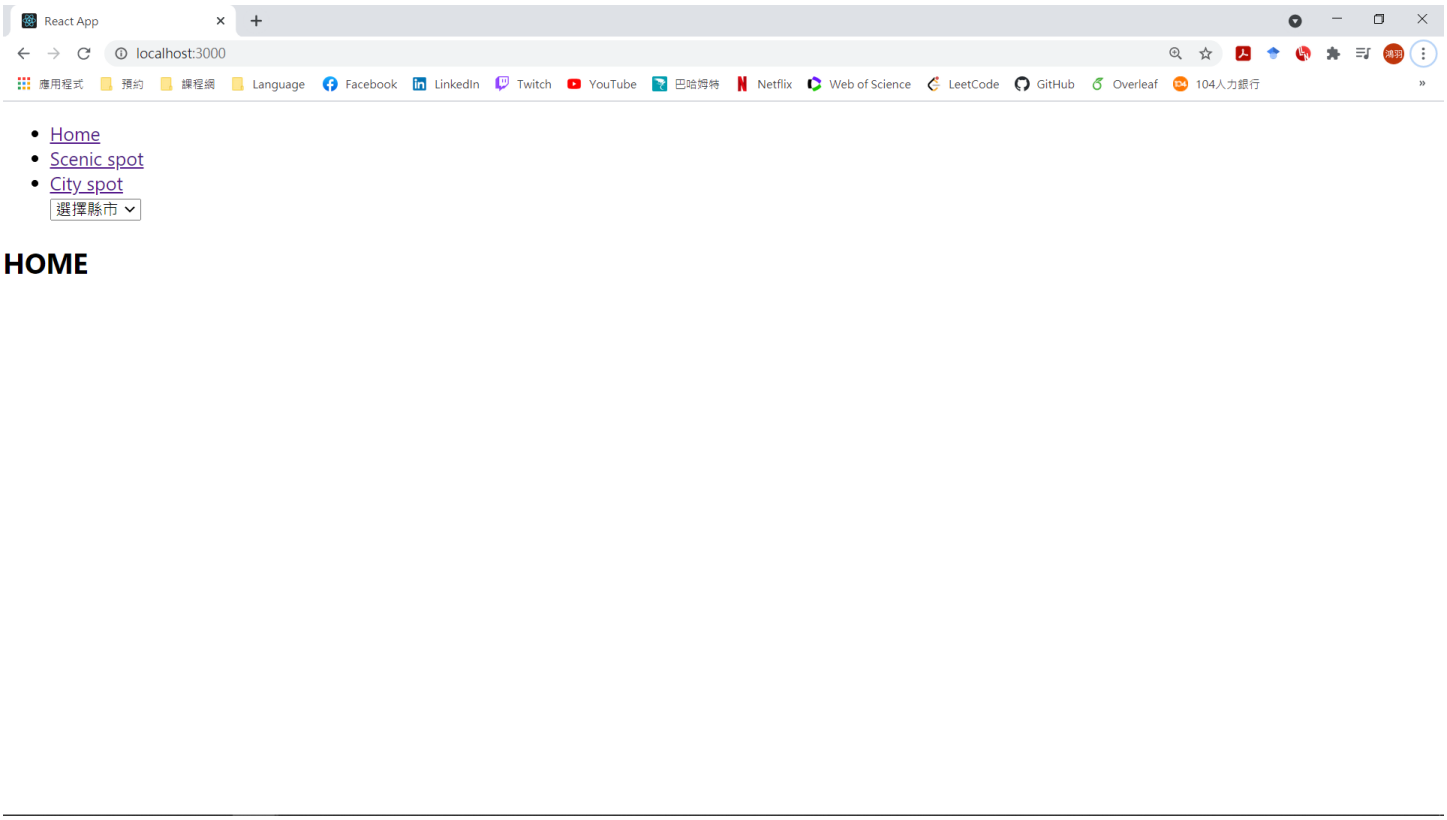
一、 大綱

依據題目要求，可以以下階層圖簡單表示。

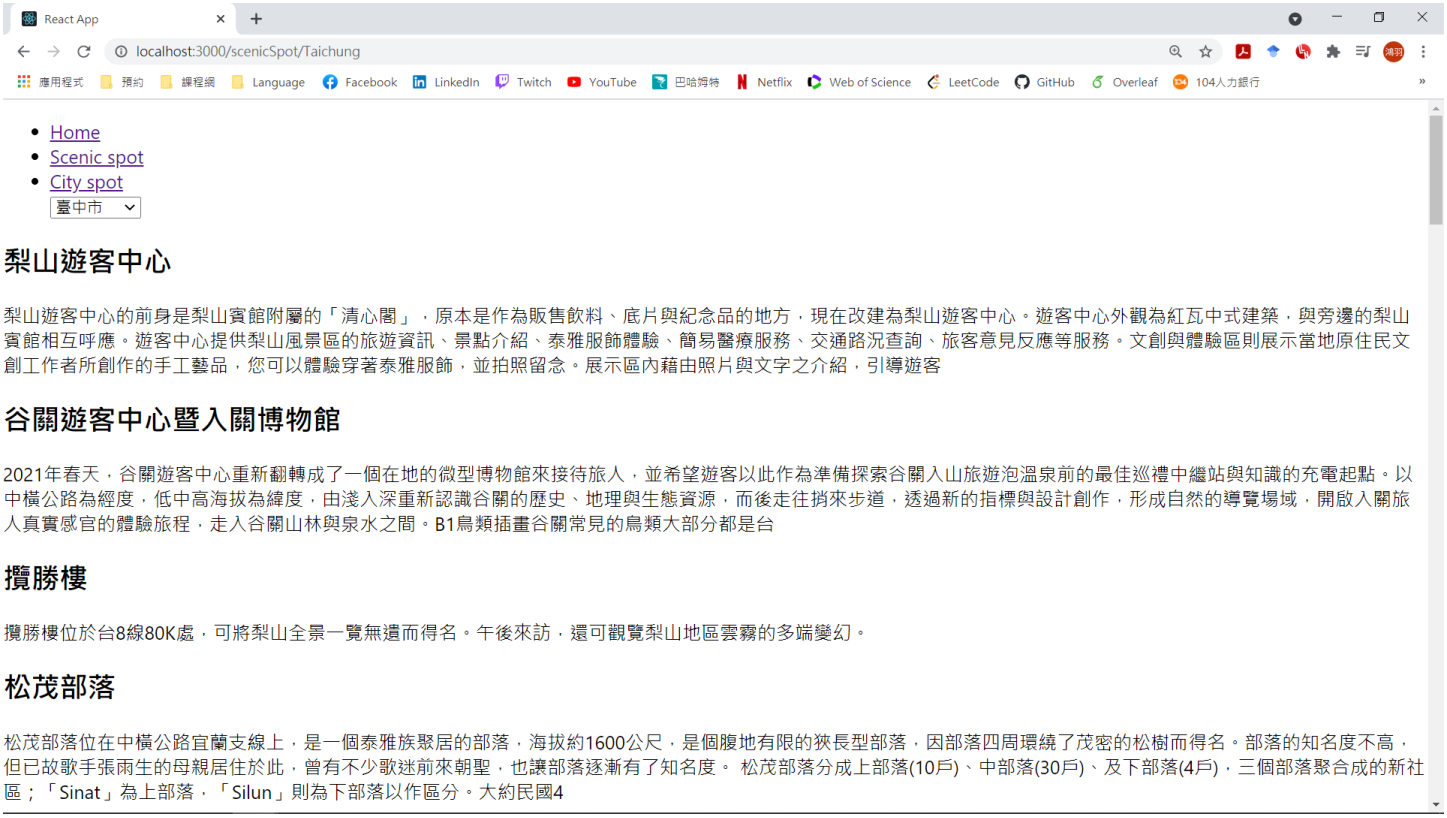


我們主要藉由 Router (作業要求三)來進行跳轉至不同頁面，兩個頁面分別是全部景點(作業要求一)及縣市景點(作業要求二)

二、 如何使用



上圖為介面，可點擊 Scenic Spot 或 City spot 進行頁面跳轉，若要點擊 City spot，請先在下方下拉選單選擇其中一個縣市，才能進行跳轉。



以上圖為例，選擇台中市後點擊 City spot，會出現三十個景點資訊，滑動至底端，會加載另外三十個景點。

三、 程式碼

1. 常用參數

```
let baseURL = 'https://ptx.transportdata.tw/MOTC/v2/Tourism/ScenicSpot';
let baseReqDemand = '?$top=30&$format=JSON'
let citys ={
  '臺北市':'Taipei','新北市':'NewTaipei','桃園市':'Taoyuan','臺中市':'Taichung','臺南市':'Tainan',
  '高雄市':'Kaohsiung','基隆市':'Keelung','新竹市':'Hsinchu','新竹縣':'HsinchuCounty','苗栗縣':'MiaoliCounty',
  '彰化縣':'ChanghuaCounty','南投縣':'NantouCounty','雲林縣':'YunlinCounty','嘉義縣':'ChiayiCounty',
  '嘉義市':'Chiayi','屏東縣':'PingtungCounty','宜蘭縣':'YilanCounty','花蓮縣':'HualienCounty',
  '臺東縣':'TaitungCounty','金門縣':'KinmenCounty','澎湖縣':'PenghuCounty','連江縣':'LienchiangCounty'
  };
```

首先我們先設定一些基本參數。由於我們是用 https://ptx.transportdata.tw/MOTC/v2/Tourism/ScenicSpot ?$top=30&$format=JSON 取得全部景點資料，及 https://ptx.transportdata.tw/MOTC/v2/Tourism/ScenicSpot/{City} ?$top=30&$format=JSON 取得縣市景點資料，根據這兩者相同的部分我們設定 baseURL 及 baseReqDemand。

然後，因應作業要求二的縣市景點，我們需要知道有哪些縣市以及其對應的英文名稱，用物件 citys 記錄下來。

2. Router

開始時做第一層 Router 部分

```
21  class App extends React.Component {
22    constructor(props){        Find related code in dcard
23      super(props);
24      this.state = {city_value: []};
25      this.handleChange = this.handleChange.bind(this);
26    }
27    handleChange(event) {
28      this.setState({city_value: event.target.value});
29    }
```

名稱是 App，繼承 React.Component。this.state 中記錄縣市，會用 handleChange 偵測下拉選單所選的縣市，並用 setState 記錄下當前所選的縣市值。

下方圖片則是 render 部分，主要是用 switch 及 route 進行頁面跳轉。應要求把 全部景點 path 設 /scenicSpot，縣市景點 path 設 `/scenicSpot/${this.state.city_value}`。

```
30  render(){
31    return(
32      <Router>       Find related code in dcard
33      <div>
34        <nav>
35          <ul>
36            <li><Link to="/">Home</Link></li>
37            <li><Link to="/scenicSpot">Scenic Spot</Link></li>
38            <li><Link to={`/scenicSpot/${this.state.city_value}`}>City spot</Link></li>
39            <select onChange={this.handleChange}>
40              <option>選擇縣市</option>
41              {Object.keys(citys).map(key=>(<option value = {citys[key]}>{key}</option>))}
42            </select>
43          </ul>
44        </nav>
45        <Switch>
46          <Route exact path="/scenicSpot" component={Scenicspot} />
47          <Route exact path={`/scenicSpot/${this.state.city_value}`}>
48            <Cityspot cityname={this.state.city_value} />
49          </Route>
50          <Route exact path="/">
51            <Home />
52          </Route>
53        </Switch>
54      </div>
55    </Router>
56    );
```

3. /scenicSpot and /scenicSpot/{City}

```
70  class Scenicspot extends App {
71    constructor(props){
72      super(props);
73      if (props.cityname === undefined){
74        this.url = baseURL+ baseReqDemand;
75      }
76      else{
77        this.url = baseURL+'/' +props.cityname +baseReqDemand;}
78      console.log("URL:",this.url);
79      this.state = {data:[],isLoading:false, count:0};
80      this.handleScroll = this.handleScroll.bind(this);
81    }
```

對作業要求一(全部景點)及作業要求二(縣市景點)的實作，因為兩者要求大同小異，只差在傳送 API 位址不同，這部分在 ln73-77 進行實作，若 props.cityname 未定義，則代表目前是要求全部景點，傳送 API 位址(this.url)為 baseURL+baseReqDemand；反之，則代表要求縣市景點，傳送 API 位址為 baseURL+props.cityname+baseReqDemand。 this.state.data=[] 準備儲存回傳回來的資料，isLoading 是判斷 render 的內容，count 則是傳送要求的次數，後面會更詳細說明。this.handleScroll 則是因應作業要求滑到底端執行特定動作，因此要附加偵測頁面滾動事件。

```
96    componentDidMount() {
97      window.addEventListener("scroll", this.handleScroll);
98      fetch(this.url)
99      .then(res=> res.json())
100     .then((result)=>{
101       this.setState({
102         data : result,
103         isLoading:true
104         })
105       })
106     }
107   componentWillUnmount() {
108     window.removeEventListener("scroll", this.handleScroll);
109     this.setState({
110       isLoading:false
111     });
112     }
```

接著，我們要用 componentDidMount() 來做 render 第一次後馬上進行的動作。ln 97 進行滾動事件監聽，接著傳送第一次 API 請求，把回傳的結果代入 this.state.data 中，並把 isLoading 設為 true。在 componentWillMount()移除監聽並把 isLoading 設為 false。

```
82    handleScroll(event){
83      if (detectBottom()) {
84        let request = new XMLHttpRequest();
85        this.state.count++;
86        request.open('GET', this.url+'&$skip=' + String(30*(this.state.count)))
87        let olddata= this.state.data;
88        const scenic = this;
89        request.onload = function () {
90          let newdata = olddata.concat(JSON.parse(this.response));
91          scenic.setState({data : newdata});
92        }
93        request.send();
94      };
95    }
```

在滾動事件中，我們首先用 detectBottom()來判斷是否以滾動至頁面底部，確認後則發送請求，根據之前的請求數 count 來判斷要 skip 多少個 30 筆資料，同時 count 也要加一以記錄已經發過幾次請求，取得回傳資料加入 this.state.data 中。

```
114   render() {
115     const { data , isLoading, count } = this.state;
116     if (!isLoading){
117       return <div>Loading...</div>;
118     } else{
119       return(
120       <div onScroll={this.handleScroll}>
121         {data.map(dat=><div><h2>{dat.Name}</h2><p>{dat.Description}</p></div>)}
122       </div>);
123     }
124     }
125   }
```

render 部分，我們主要用 map 方式 (ln121) ，將所有在 this.state.data 裡的資料印出來。但是在第一次請求收到回傳前，data 裡為空，因此最前面我們設定 isLoading，在第一次收到回傳前設為 false，這樣在 render 部分就不會因為 data 裡面資料為空而使得 .map 出問題。

縣市景點(/scenicSpot/{City})則繼承剛剛的全部景點 class。