Ask Company

리액트와 함께 장고 시작하기 / 리액트

라우팅 처리

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

2가지 라우팅 방식

1. 브라우저에 의한 라우팅

- 웹페이지에서 주로 사용되던 전통적인 방법
- <a> 태그를 활용하거나, JavaScript에서 location.href 속성을 활용하여, 페이지 이동

2. JavaScript 단에서 라우팅을 흉내내기

- SPA(Single Page Application)에서 사용하는 방법
- 리액트는 SPA 방식으로 개발하는 것이 보다 적합
- 브라우저 히스토리 API 활용 → react-router-dom 라이브러리를 통한 보다 손쉬운 처리

react-router-dom

설치> yarn add react-router-dom

5.0.1 버전에서 테스트되었습니다.

```
import { BrowserRouter, Route, Link } from 'react-router-dom';

const App = () => (
  <BrowserRouter>
    <div>
      <Link to="/">Home</Link>
      <Link to="/posts">포스팅</Link>
      <hr/>
      <Route exact path="/" component={Home} />
      <Route path="/posts" component={PostList} />
    </div>
  </BrowserRouter>
);
```

현재 Route에 매칭되는 Route의 Component가 **모두** 렌더링됩니다.
단일 Route만 선택되게 할려면, Switch를 사용합니다.

NavLink

Link와 유사

activeStyle과 activeClassName 속성을 지원

```
import { BrowserRouter, Route, NavLink, Link } from 'react-router-dom';

const App = () => {
  const activeStyle = {
    fontWeight: 'bold',
    backgroundColor: 'yellow',
  };

  return (
    <BrowserRouter>
      <div>
        <NavLink exact to="/" activeStyle={activeStyle}>Home</NavLink>
        <NavLink to="/posts" activeStyle={activeStyle}>포스팅</NavLink>
        <NavLink to="/tags" activeStyle={activeStyle}>태그</NavLink>
```

Route로 설정된 컴포넌트가 받는 3가지 props

history : 히스토리 조작

.location, .push(...), .replace(...), .goBack(), goForward() 등

location : 현재 경로 정보

.hash, .pathname, .search, .state 속성

match : Router 매칭 정보

.isExact, .url, .path, .params 속성

Switch와 No Match 처리

src/App.js

```
import { BrowserRouter, Route, NavLink, Link, Switch } from 'react-router-dom';

const App = () => {
  return (
    <BrowserRouter>
      중략 ...
      <Switch>
        <Route exact path="/" component={Home} />
        <Route exact path="/posts" component={PostList} />
        <Route component={RouteNoMatch} />
      </Switch>
```

단일 Route만 처리되도록 Switch 지정

순서대로 매칭을 시도

path를 지정하지 않기에, 모든 path에 매칭

src/RouteNoMatch.js

```
import React from 'react';

const RouteNoMatch = ({ location, match }) => (
  <div>
    {location.pathname}에 매칭된 Route가 없습니다.
  </div>
);

export default RouteNoMatch;
```

match ➔ .url, .params 속성

src/App.js

```
<Route exact path="/posts/:post_id" component={PostDetail} />
<Route exact path="/posts" component={PostList} />
```

src/PostList.js

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

const PostList = ({ match }) => (
  <div>
    <h2>PostList</h2>
    <ul>
      <li>
        <Link to={`${match.url}/100`}>100번 포스팅</Link>
      </li>
      <li>
        <Link to={`${match.url}/200`}>200번 포스팅</Link>
      </li>
    </ul>
  </div>
);

export default PostList;
```

src/PostDetail.js

```
import React from 'react';

const PostDetail = ({ match }) => (
  <div>
    <h2>PostDetail #{match.params.post_id}</h2>
    Lorem Ipsum is simply dummy text of the printing
    and typesetting industry.
  </div>
);

export default PostDetail;
```

QueryString 처리

location.search 속성에서 문자열로 제공 → "?foo=bar"

query-string 라이브러리를 통해, 객체로 변환 가능

설치> yarn add query-string

모든 값은 문자열 타입.
숫자 등은 별도 변환이 필요

```
import queryString from "query-string";

const About = ({history, location, match}) => {
  const qs1 = queryString.parse("?foo=bar");
  const qs2 = queryString.parse(location.search)
```

https://www.npmjs.com/package/query-string

Life is short.
You need Python and Django.

I will be your pacemaker.