7/20 ~ 7/24 3주차 - 3

Vue-Router 살펴보기

7/20일 오늘 뷰 라우터를 적용하려했는데 잘 되지 않았다. 기본 개념을 다시 공부하자.

뷰JS의 간단한 라우팅

- 단일 페이지 어플리케이션(SPA)에서 가장 먼저 처리해야 할 것이 바로 라우팅이다.
- 서버에서 라우팅은 URI에 따라 해당하는 정적파일을 내려주는 방식이다.
- 이를 브라우져에서 구현해야 하는 것이 SPA 개발의 핵심이다.

아이디어는 간단하다! 요청 URI에 따라 브라우져에서 돔을 변경하는 방식이다.

Vue.js 공식 문서의 Scaling Up 섹션에 구현된 코드가 있다.

라우팅 별로 세개의 컴포넌트를 만든다.

```
const Home = { template: '<p>home page</p>' }
const About = { template: '<p>about page</p>' }
const NotFound = { template: '<p>Page not found</p>' }
```

경로에 따라 위 세개 컴포넌트를 바꿔가면서 렌더링 해주려고 한다. 그럼 경로와 컴포넌트 딕셔너리가 필요하다.

```
const routes = {
  '/': Home,
  '/about': About
}
```

루트('/') 경로와 어바웃('/about') 경로에 대한 각각의 컴포넌트 딕셔너리를 만들었다.

- 뷰 인스턴스 설정 객체에는 el, data 뿐만아니라 render 속성도 있다.
- 사실 함수다.
- createElement같은 돔 생성 함수를 인자로 주입 받는 함수가 render 함수이다.
- 함수 h()에 컴포넌트 설정 객체를 넘겨주면 뷰JS는 컴포넌트를 화면에 그린다. (컴포넌트를 마운팅 포인트에 갖다 붙인다.)

요청 경로에 따라 render() 함수가 동적으로 동작하도록 만들면 라우팅에 따라 다른 화면을 구현할 수 있는 셈이다.

```
new Vue({
  /* 생략 */

  render (h) { return h(this.ViewComponent) }
})
```

- render() 함수는 ViewComponent라는 계산된 속성을 인자로 전달해서 h() 함수를 호출한다.

```
new Vue({
  /* 생략 */

  computed: {
    ViewComponent () {
      return routes[window.location.pathname] || NotFound
    }
```

```
  },

  /* 생략 */
})
```

- VueComponent는 현재 경로로 routes 디셔너리에서 컴포넌트를 찾아 반환한다. 정의되지 않은 경로일 경우 NotFound 컴포넌트를 반환한다.

라우터 라이브러리: vue-router

이렇게 단순한 라우팅을 직접 구현할 수도 있겠지만, SPA를 개발하려면 라우터 전용 라이브러리를 사용하는 것이 좋다.

왜?

→ 중첩 라우팅, 네비게이션 가드 등 SPA 개발시 다양한 상황을 수월하게 처리할 수 있기 때문이고 공식 문서에도 해당 라이브러리 사용을 추천함.

- Github: https://github.com/vuejs/vue-router
- 문서: https://router.vuejs.org/kr/

간단한 스케폴딩 위에서 뷰 라우터의 기본적인 사용법을 간단히 학습해 보자.

(vue-cli의 webpack-simple 템플릿으로 시작한다.)

라우터 객체 생성

- 뷰 인스턴스 생성 객체에는 router 속성이 있다. 뷰 라우터를 사용하려면 이 속성으로 뷰 라우터 객체를 넘겨줘야 한다.

```
// main.js

import Vue from 'vue'
import App from './App.vue'
import router from './router'
```

```
new Vue({
  el: '#app',
  render: h => h(App),

  // 라우터 객체를 넘겨준다
  router
})
```

그럼 실제 router/index.js 파일에서 라우터 객체를 생성해 보자.

→ 뷰 라우터는 플러그인 형태이기 때문에 Vue.use() 함수를 이용해서 등록한다.

```
// router/index.js

import Vue from 'vue'
import VueRouter from 'vue-router'

// 뷰 어플리케이션에 라우터 플러그인을 추가한다.
Vue.use(VueRouter)
```

라우팅에 따라 렌더링할 컴포넌트 설정 객체를 정의한다.

template 속성만 정의했다.

```
// router/index.js

/* 생략 */
const Home = { template: '<div>Home</div>'}
const NotFound = { template: '<div>Not Found</div>'}
```

마지막으로 VueRouter 클래스 함수로 라우터 객체를 생성한다.

```
// router/index.js

/* 생략 */
const router = new VueRouter({
  mode: 'history',
  routes: [
    { path: '/', component: Home },
```

```
    { path: '*', component: NotFound }
  ]
})

export default router
```

- mode 속성 기본값은 “해쉬뱅 모드”다.
- 브라우져 history 객체의 pushState() API를 사용하기 위해서는 “history”로 모드값을 설정한다.
- routes는 경로와 컴포넌트으로 이뤄진 컬렉션이다. 실제 이 맵을 보고 라우터가 경로에 따라 그에 맞는 컴포넌트를 렌더링한다.
- 이 컬렉션은 순서가 중요한데 정의한 순서대로 경로를 매칭해서 컴포넌트를 출력한다.
- 홈(/) 경로로 들어올 경우 { path: '/', component: Home } 설정에 의해 Home 컴포넌트가 렌더링 될 것이다.
- 정의되지 않은 경로(예: /NotFound) 로 들어올결우 { path: '*', component: NotFound } 설정에 의해 NotFound 컴포넌트가 그려질 것이다.

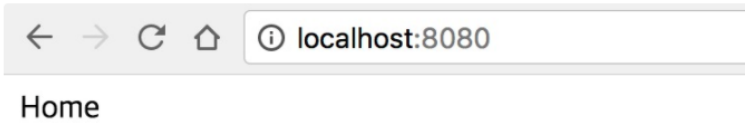
라우터 뷰

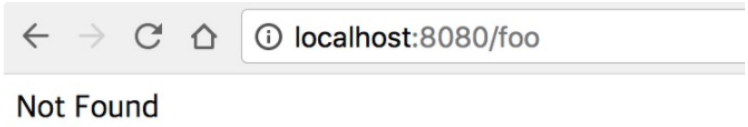
```
<!-- App.vue --><template>
  <div>
    <!-- 여기에 라우터가 컴포넌트를 그린다 --><router-view></router-view>
  </div>
</template>
```

루트 컴포넌트에 라우터 뷰를 추가한다.

라우팅이 경로에 따라 컴포넌트를 바꿔치기해서 렌더링한다고 했는데...

렌더링해주는 부분이 <router-view> 태그 부분이다. 결과를 확인해 보면 Home과 Notfound가 경로에 따라 출력되는 것을 확인할 수 있다.


localhost:8080
Home


localhost:8080/foo
Not Found