webpack 템플릿을 이용한 테스트

기본 기능 테스트

- 예제 A-01 : src/sum.js

```
let sum = (a, b) => {
  return a + b
}
export default sum
```

- 예제 A-02 : test/unit/specs/sum.spec.js

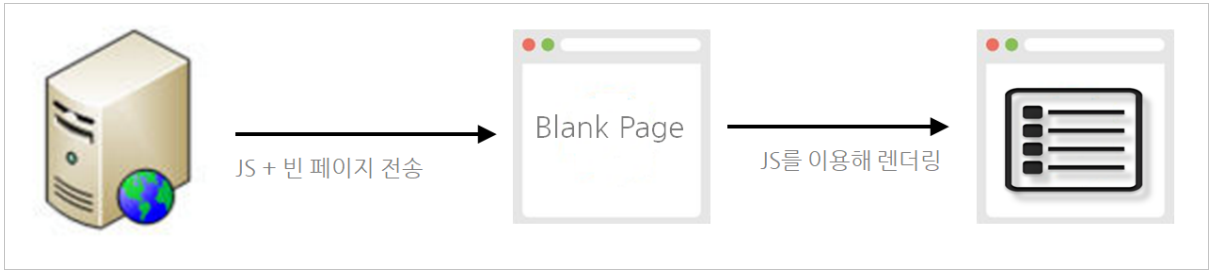
```
import sum from '@/sum.js'
describe('sum', () => {
  it('add 2+3 equals 5', () => {
    expect(sum(2, 3)).to.equal(5)
  })
})
```

- npm run test
  - karma 테스트 러너를 이용함

서버 사이드 렌더링

vue.js는 클라이언트 측 애플리케이션 프레임워크

- 클라이언트 사이드 렌더링!!
- 빈 HTML 페이지와 JS 코드를 내려받아 브라우저에서 화면을 렌더링



클라이언트 사이드 렌더링의 문제점

- SEO : Search Engine Optimization
- 느린 화면 로딩 속도

서버 사이드 렌더링

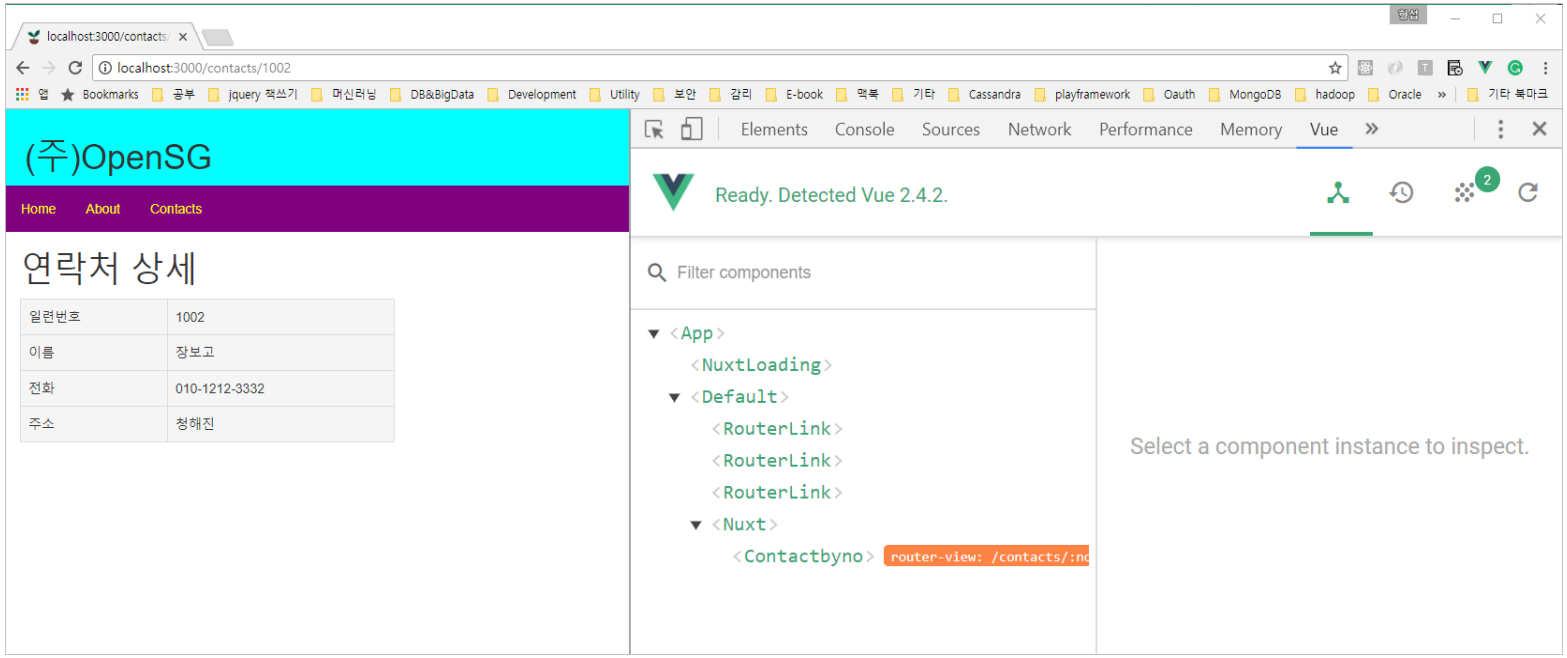
SSR(Server Side Rendering)이 반드시 필요한가?

- 그렇지는 않다.
  - 초기화면 로딩 속도가 중요하지 않은 경우도 있고
  - SEO가 요구되지 않는 경우도 많음
  - 브라우저와 서버와의 라운드트립이 더 빈번하게 발생하므로 서버 부하 가중
  - 설치와 배포가 까다로움
- 자세한 SSR에 대한 문서
  - http://ssr.vuejs.org/ko/ 참조하세요
- 조금 더 쉽게 SSR을 구현하려면 Nuxt.js 사용
  - 잘 정리된 한글 문서 : http://ko.nuxtjs.org/guide/

서버 사이드 렌더링

Nuxt.js를 이용해 샘플 작성

- 12장에서 작성했던 routertest 프로젝트를 Nuxt.js 버전으로 작성



프로젝트 생성과 초기화

생성

- mkdir nuxttest
- cd nuxttest
- npm init
- npm install --save nuxt
  - 이것으로 관련된 패키지(vue, vue-router, vuex 등)가 모두 설치됨
- package.json 에 scripts 요소 추가

```
{
  ……
  "scripts": {
    "dev": "nuxt",
    "build": "nuxt build",
    "start": "nuxt start",
    "generate": "nuxt generate"
  },
  ……
  "dependencies": {
    "nuxt": "^1.0.0-rc11"
  }
}
```

디렉터리 구조

디렉터리, 파일 구조

- layouts, pages, store 디렉터리 사용
  - 라우트와 Vuex 상태 관리 적용

표 B-01 디렉터리, 파일 정보

디렉터리명	설명
assets	less, sass, javascript와 같은 컴파일되지 않는 자원들을 포함합니다.
components	Vue.js 컴포넌트를 포함합니다.
layouts	애플리케이션의 레이아웃을 포함합니다. 레이아웃은 .vue 파일로 작성하면 됩니다.
middleware	애플리케이션의 미들웨어를 포함합니다. 이 디렉터리 아래에는 페이지나 레이아웃이 렌더링되기 전에 실행할 함수들을 등록할 수 있습니다.
pages	애플리케이션의 .vue 파일을 작성합니다. 이 파일이 페이지이며, 단지 화면 뷰만 제공하는 것이 아니라 하위 디렉터리 구성을 포함해 라우트 정보를 자동 생성합니다.
plugins	Vue.js 애플리케이션이 생성되기 전 실행하고 싶은 자바스크립트 플러그인을 포함합니다.
static	이곳에는 이미지, 텍스트 파일 등의 정적 파일을 배치합니다.
store	이곳에는 Vuex 저장소(store) 파일을 배치합니다. 이 디렉터리의 파일은 Nuxt.js 애플리케이션이 실행되면서 자동으로 로딩합니다.
nuxt.config.js	이 파일에는 사용자 정의 설정을 작성합니다.

저장소 기능 작성

예제 B-02 : constant.js

```
export default {
    CHANGE_NO : "changeNo",
}
```

예제 B-03 : store/state.js

```
export default {
    no : 0,
    contacts : [
        { no:1001, name:'김유신', tel:'010-1212-3331', address:'경주' },
        { no:1002, name:'장보고', tel:'010-1212-3332', address:'청해진' },
        { no:1003, name:'관창', tel:'010-1212-3333', address:'황산벌' },
        { no:1004, name:'안중근', tel:'010-1212-3334', address:'해주' },
        { no:1005, name:'강감찬', tel:'010-1212-3335', address:'귀주' },
        { no:1006, name:'정몽주', tel:'010-1212-3336', address:'개성' },
        { no:1007, name:'이순신', tel:'010-1212-3337', address:'통제영' },
        { no:1008, name:'김시민', tel:'010-1212-3338', address:'진주' },
        { no:1009, name:'정약용', tel:'010-1212-3339', address:'남양주' }
    ]
}
```

저장소 기능 작성

예제 B-04 : store/mutations.js

```
import Constant from '~/constant';

export default {
    [Constant.CHANGE_NO] : (state, payload) => {
        if (payload.no !== "") {
            state.no = payload.no;
        }
    }
}
```

예제 B-05 : store/getters.js

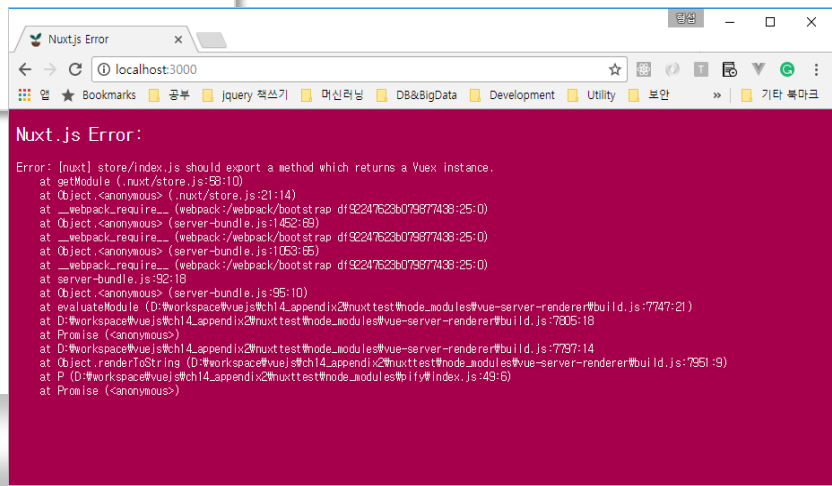
```
export default {
    getContactOne(state) {
        var no = state.no;
        var arr = state.contacts.filter(function(item, index) {
            return item.no == no;
        });
        if (arr.length == 1)   return arr[0];
        else   return {};
    },
    getContacts(state) {
        return state.contacts;
    }
}
```

저장소 기능 작성

예제 B-06 : store/index.js

- Vuex Store 객체를 만들 때 반드시 함수를 통해 리턴하도록 작성해야 함.

```
import Vue from 'vue';
import Vuex from 'vuex';
import state from './state.js';
import getters from './getters.js';
import mutations from './mutations.js';

const store = () => {
    return new Vuex.Store({
        state,
        getters,
        mutations
    })
}

export default store
```



Vuex 인스턴스를 함수를 이용해 리턴하지 않았을 때의 오류

nuxt.config.js 파일 작성

nuxt.config.js 파일

- Nuxt.js 앱의 기능을 설정하는 역할
- 이 예제에서는 css 옵션을 사용해 볼 것임
  - 모든 페이지에서 사용할 CSS 파일, 라이브러리를 등록할 수 있는 기능 제공

예제 B-07 : nuxt.config.js

```
module.exports = {
  head: {
    link: [
      {
        rel: 'stylesheet',
        href: 'https://cdn.bootcss.com/bootstrap/3.3.5/css/bootstrap.css'
      }
    ]
  }
}
```

레이아웃 작성

레이아웃을 작성하지 않았을 때의 기본 템플릿

```
<!DOCTYPE html>
<html {{ HTML_ATTRS }}>
  <head>
    {{ HEAD }}
  </head>
  <body {{ BODY_ATTRS }}>
    {{ APP }}
  </body>
</html>
```

추가적인 레이아웃을 사용하고 싶다면...

- layouts 디렉터리에 레이아웃 파일 작성

레이아웃 작성

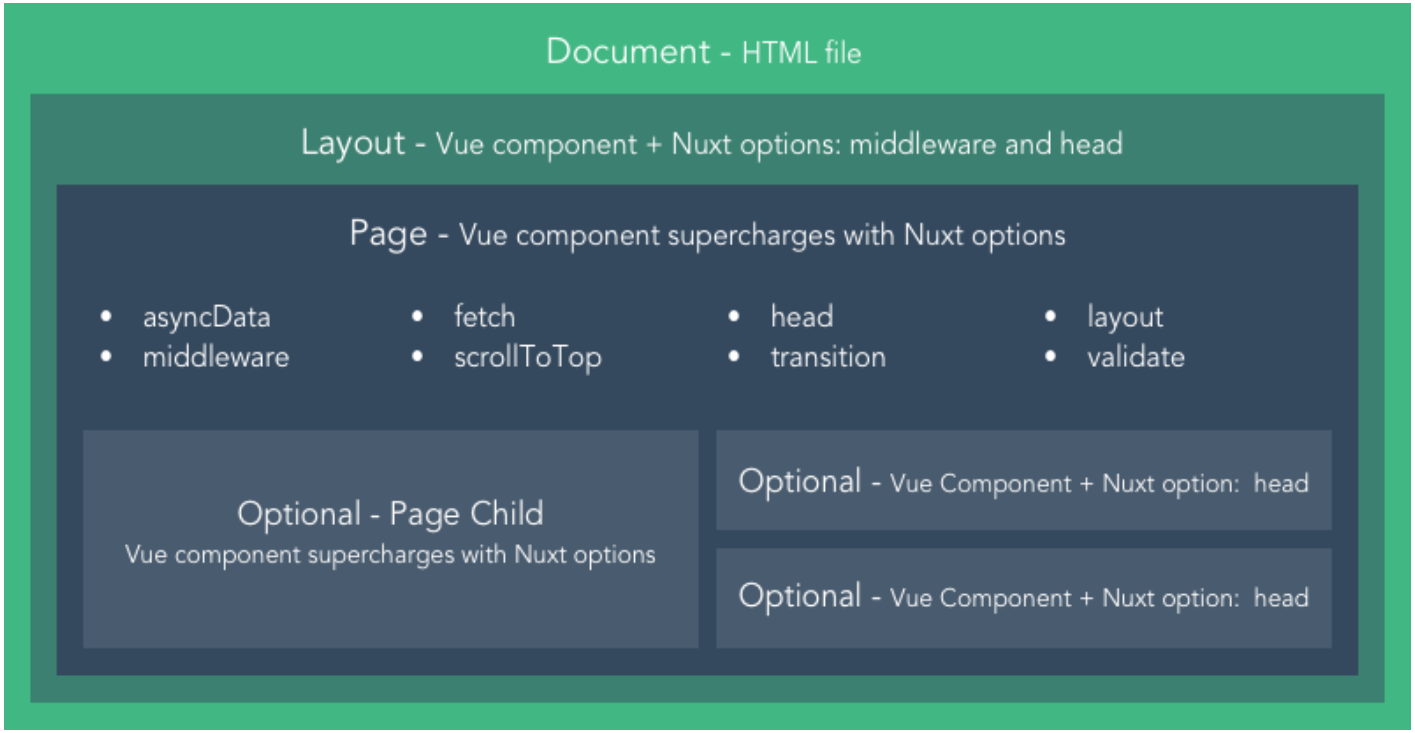
예제 B-08 : layouts/default.vue

```
<template>
<div>
  <div class="header">
    <h1 class="headerText">(주)OpenSG</h1>
    <nav>
      <ul>
        <li>
          <nuxt-link to="/">Home</nuxt-link>
        </li>
        <li>
          <nuxt-link to="/about">About</nuxt-link>
        </li>
        <li>
          <nuxt-link to="/contacts">Contacts</nuxt-link>
        </li>
      </ul>
    </nav>
  </div>

  <div class="container">
    <nuxt></nuxt>
  </div>
</div>
</template>
```

```
<script>
export default {
  name : 'default'
}
</script>
<style>
   .header { background-color:aqua;
      padding: 10px 0px 0px 0px; }
   .headerText { padding: 0px 20px 0px 20px; }
   ul { list-style-type: none; margin: 0; padding: 0;
      overflow: hidden; background-color: purple;  }
   li { float: left; }
   li a { display: block; color: yellow; text-align: center;
      padding: 14px 16px; text-decoration: none;  }
   li a:hover { background-color: aqua; color:black; }
</style>
```

페이지 작성

- pages 디렉터리에 .vue 파일을 작성하면 적절히 조합되어 뷰를 생성함.
- Nuxt.js에서의 뷰 조합 방식



페이지 작성

.vue 페이지의 추가적인 옵션

표 B-02 추가적인 .vue 페이지 옵션

옵션명	설명
asyncData	페이지 컴포넌트가 로딩되기 전에 매번 호출되는 함수입니다. 페이지 컴포넌트마다 비동기로 외부 데이터를 가져와야할 경우에 사용하면 유용합니다.
fetch	페이지가 렌더링되기전에 Vuex 저장소(store)에 데이터를 생성하기 위해 사용합니다.
head	페이지에 대한 특정 메타 태그를 설정하기 위해 사용합니다.
layout	layouts 디렉터리에 정의된 특정 레이아웃을 사용하도록 지정합니다.
transition	페이지에 대한 트랜지션 효과 기능을 설정합니다.
validate	동적 라우트에 대한 유효성을 검증하는 기능을 설정합니다.
middleware	페이지나 레이아웃이 렌더링되기 전에 실행할 사용자 함수를 설정합니다.

페이지 작성

작성할 페이지의 경로와 생성되는 라우트 정보

표 B-03 정적 라우트 경로

라우트 경로	디렉터리 및 파일 경로
/	~/pages/index.vue
/about	~/pages/about.vue
/contacts	~/pages/contacts/index.vue
/contacts/:no	~/pages/contacts/_no.vue

- 하위 경로를 나타내기 위해 디렉터리를 작성함.
- 기본 경로는 index.vue로 작성함.
- 동적 라우트 정보는 _no.vue와 같이 언더스코어 기호 사용

페이지 작성

예제 B-09 : pages/index.vue

```
<template>
  <div>
      <h1>Home</h1>
  </div>
</template>
<script>
export default {
  name : "home"
}
</script>
```

- pages/about.vue 는 예제 B-09와 볼드체 표현 부분만 다르게 작성

페이지 작성

예제 B-10 : pages/contacts/index.vue

```
<template>
  <div>
    <h1>연락처</h1>
    <div class="wrapper">
        <div class="box" v-for="c in contacts" :key="c.no">
            <nuxt-link v-bind:to="'/contacts/'+c.no">{{c.name}}</nuxt-link>
        </div>
    </div>
  </div>
</template>
<script>
import { mapGetters } from 'vuex'

export default {
  name : "contacts",
  computed : mapGetters({
    contacts: 'getContacts'
  })
}
</script>
<style>
.....(생략)
</style>
```

페이지 작성

예제 B-11 : pages/contacts/_no.vue

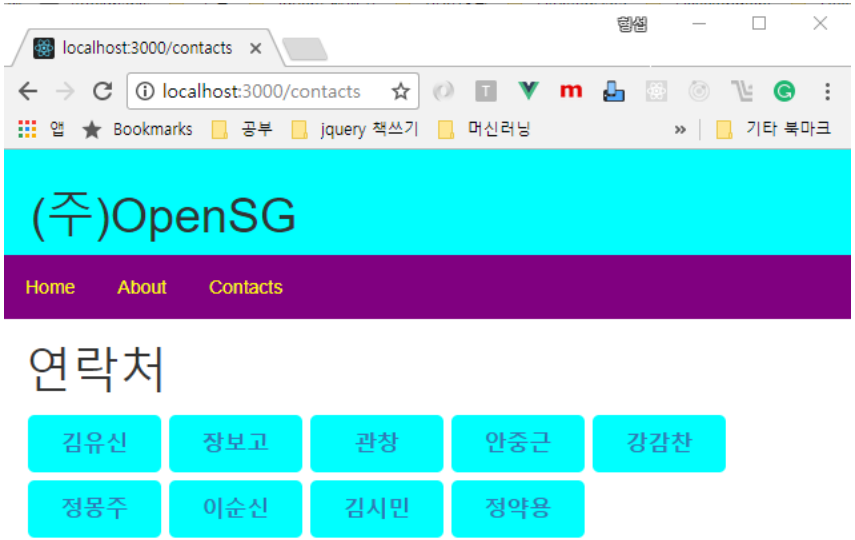
```
<template>
  <div>
    <h1>연락처 상세</h1>
    <div >
        <table class="detail table table-bordered">
            <tbody>
            <tr class="active">
                <td>일련번호</td>
                <td>{{contact.no}}</td>
            </tr>
            <tr class="active">
                <td>이름</td>
                <td>{{contact.name}}</td>
            </tr>
            <tr class="active">
                <td>전화</td>
                <td>{{contact.tel}}</td>
            </tr>
            <tr class="active">
                <td>주소</td>
                <td>{{contact.address}}</td>
            </tr>
            </tbody>
        </table>
    </div>
  </div>
</template>
```

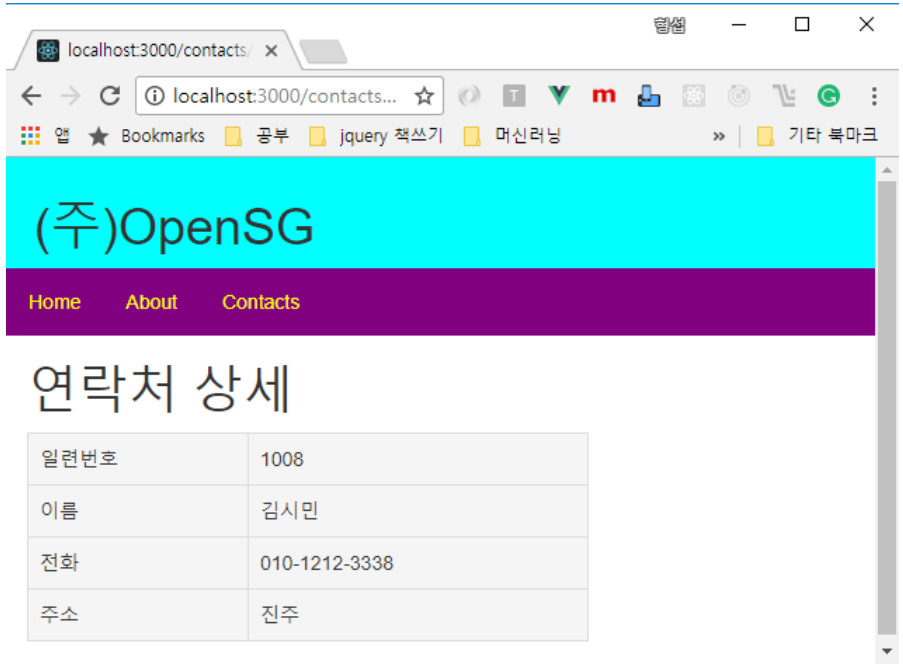
```
<script>
import { mapGetters } from 'vuex'
import Constant from '~/constant'

export default {
    name : 'contactbyno',
    computed : mapGetters({
        contact : 'getContactOne'
    }),
    created : function() {
        var no = this.$route.params.no;
        this.$store.commit(Constant.CHANGE_NO, { no : no })
    }
}
</script>
<style>
table.detail { width:400px; }
</style>
```

페이지 작성

실행

- npm run dev
  - 페이지를 전환한 후 페이지 소스보기를 해보자!1
  - 렌더링된 HTML을 서버로부터 응답받았음을 알 수 있음.

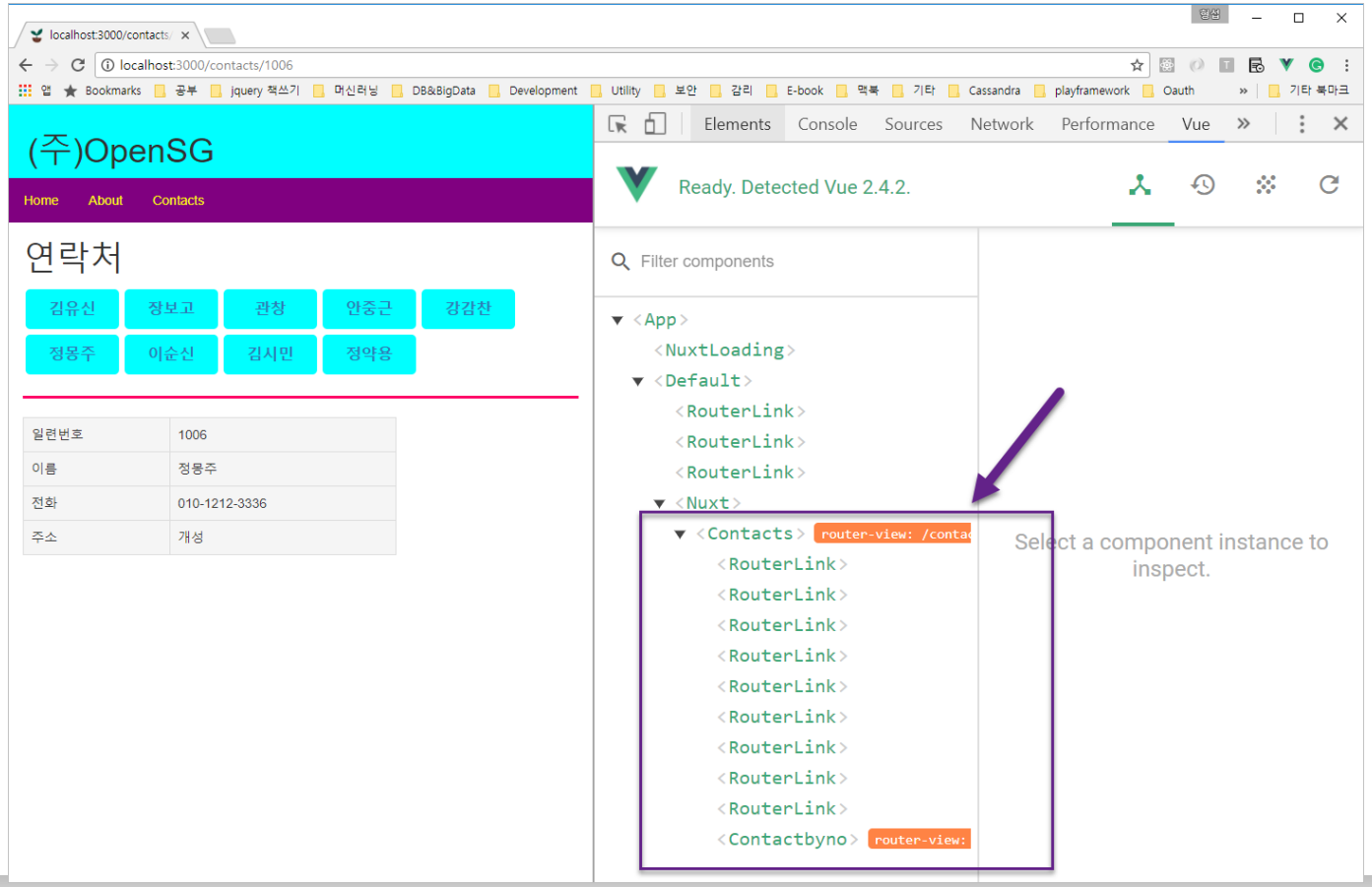




중첩 라우트 적용

미리 실행 결과 확인

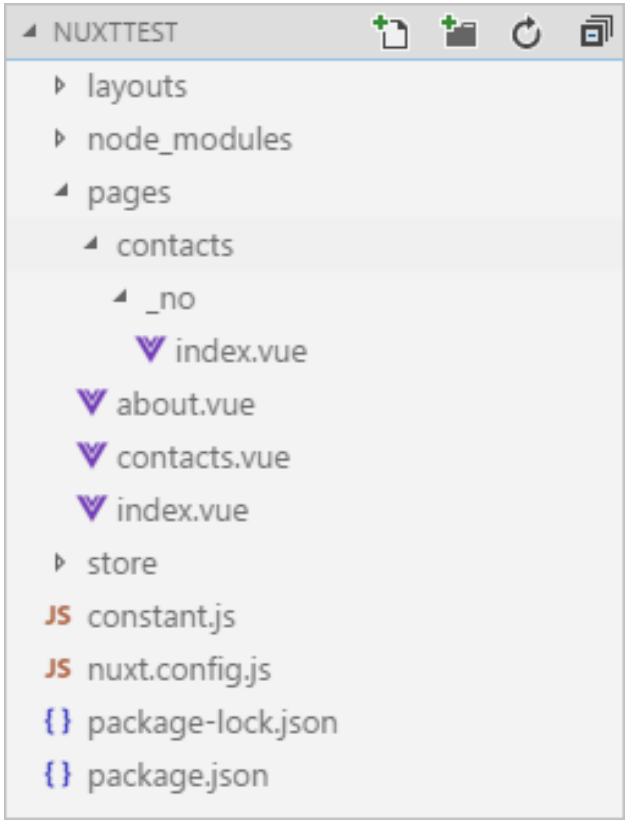
- contacts 화면 내부에 상세 정보 화면이 포함됨



중첩 라우트 적용

중첩 라우트 적용을 위한 디렉터리 구조

- 디렉터리 경로가 바뀜
- pages/contacts/index.vue
  - pages/contacts.vue로 변경
- pages/contacts/_no.vue
  - pages/contacts/_no/index.vue로 변경



중첩 라우트 적용

예제 B-12 : pages/contacts.vue

```
<template>
  <div>
    <h1>연락처</h1>
    <div class="wrapper">
        <div class="box" v-for="c in contacts" :key="c.no">
            <nuxt-link v-bind:to="'/contacts/'+c.no">{{c.name}}</nuxt-link>
        </div>
    </div>
    <nuxt-child></nuxt-child>
  </div>
</template>
.....(생략)
```

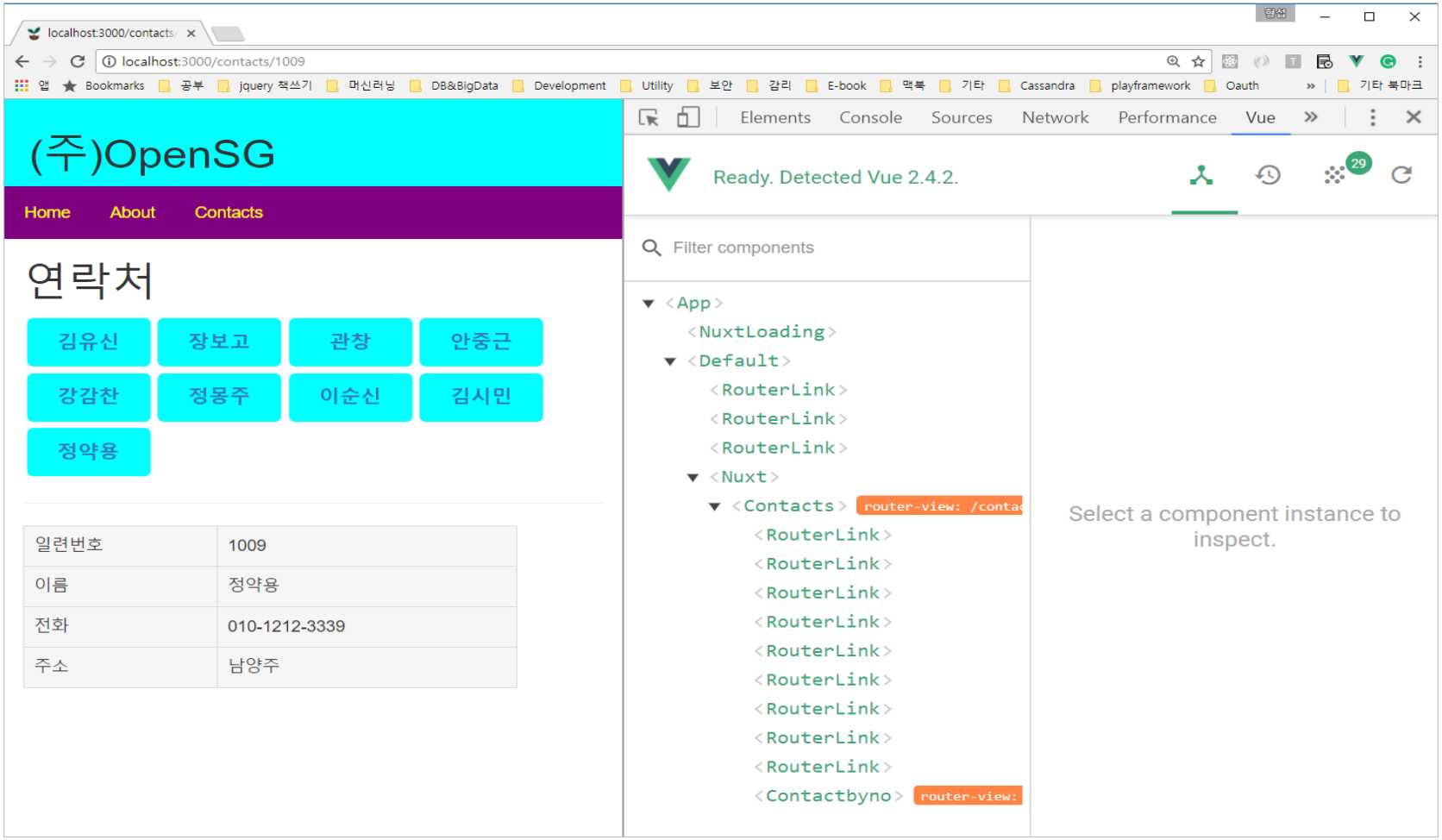
중첩 라우트 적용

예제 B-13 : pages/contacts/_no/index.vue

```
<template>
  <div>
    <hr class="divider"></hr>
    <div >
        <table class="detail table table-bordered">
            ......(생략)
        </table>
    </div>
  </div>
</template>
<script>
.....(생략)
export default {
    ......(생략)
    beforeRouteUpdate(to,from,next) {
        var no = to.params.no;
        this.$store.commit(Constant.CHANGE_NO, { no : no })
        next()
    }
}
</script>
......
```

중첩 라우트 적용

실행 결과



트랜지션 효과 적용

페이지 단위의 트랜지션 효과를 잘 지원함

- 기본 트랜지션 효과 이름 : page
- 프로젝트 디렉터리에 assets/transition.css 파일을 생성

```
.page-enter-active, .page-leave-active {
  transition: opacity .3s;
}
.page-enter, .page-leave-to {
  opacity: 0;
}
```

- nuxt.config.js 파일 변경

```
module.exports = {
  head: {
    link: [
      {
        rel: 'stylesheet',
        href: 'https://cdn.bootcss.com/bootstrap/3.3.5/css/bootstrap.css'
      }
    ]
  },
  css: [ 'assets/transition.css' ]
}
```

트랜지션 효과 적용

- 여기까지 실행
  - 화면 전환이 부드러워졌다!!
- elastic 효과 적용
  - assets/transition.css 파일 변경

```
.page-enter-active, .page-leave-active {
  transition: opacity .5s;
}
.page-enter, .page-leave-to {
  opacity: 0;
}

.elastic-enter-active {
    animation: elastic-in 0.5s;
}
.elastic-leave-active {
    animation: elastic-in 0.5s reverse;
}
@keyframes elastic-in {
    0% { transform: scale(0); opacity:0;}
    70% { transform: scale(1.2); opacity:0.5; }
    100% { transform: scale(1); opacity:1; }
}
```

트랜지션 효과 적용

- pages/contacts.vue 파일 변경

```
<template>
   ……(생략)
</template>
<script>
import { mapGetters } from 'vuex'

export default {
  name : "contacts",
  transition : "elastic",
  computed : mapGetters({
    contacts: 'getContacts'
  })
}
</script>
<style>
 ……(생략)
</style>
```

- 다시 실행!!
  - 이제 elastic 효과까지 적용되었음.