axios를 이용한 서버 통신

HTTP 서버와 통신하기 위한 라이브러리

- axios, fetch, superagent, vue-resource
- 이중 axios가 가장 범용적이고 많이 사용됨.

서비스 API 소개

- http://sample.bmaster.kro.kr 또는
- https://github.com/stepanowon/contactsvc 내려받아 실행

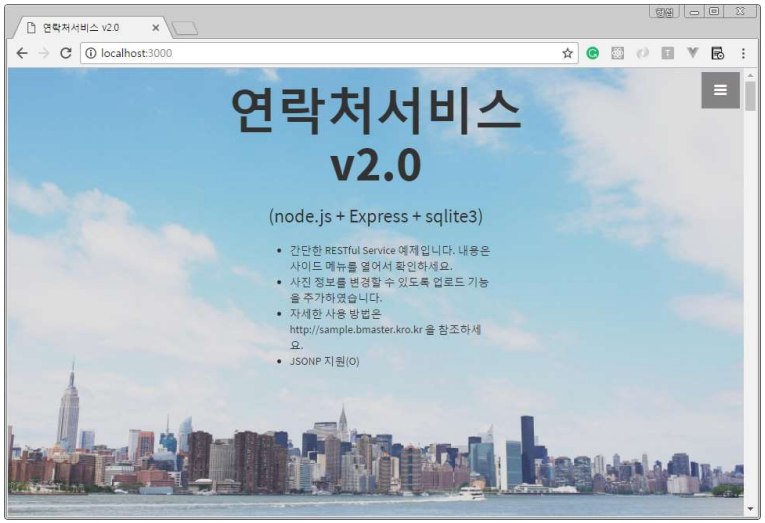


표 10-01 연락처 서비스 API 목록

GET /contacts	연락처 목록을 조회하며 특정 페이지 데이터를 조회할 수 있습니다
GET /contacts/:no	특정 일련번호 한 건의 연락처 정보를 조회합니다
GET /contacts/search/:name	이름의 일부를 이용해 연락처를 검색합니다. 2글자 이상부터 검색이 가능합니다.
POST /contacts	새로운 연락처를 추가합니다
PUT /contacts/:no	기존 연락처 정보를 수정합니다.
DELETE /contacts/:no	일련번호 정보를 이용해 연락처 정보를 삭제합니다.
POST /contacts/batchinsert	한 번에 여러 건의 연락처 정보를 추가합니다.
POST /contacts/:no/photo	연락처에 저장하는 사람의 사진 정보를 등록합니다.
GET /contacts_long	GET /contacts와 동일한 기능이지만 1초의 의도적 지연 시간 후에 응답합니다.
GET /contacts_long/search/:name	GET /contacts/search/:name와 동일한 기능이지만 1초의 의도적 지연 시간 후에 응답합니다.

axios 기능 테스트

- **서비스 측에서 CORS를 지원하면**
  - http proxy를 이용할 필요가 없음
- **프로젝트 템플릿 설정**

```
vue create contactsapp
cd contactsapp
yarn add axios
```

axios 기능 테스트

http 프록시 설정

- SOP : Same Origin Policy, 브라우저의 기본 보안 정책

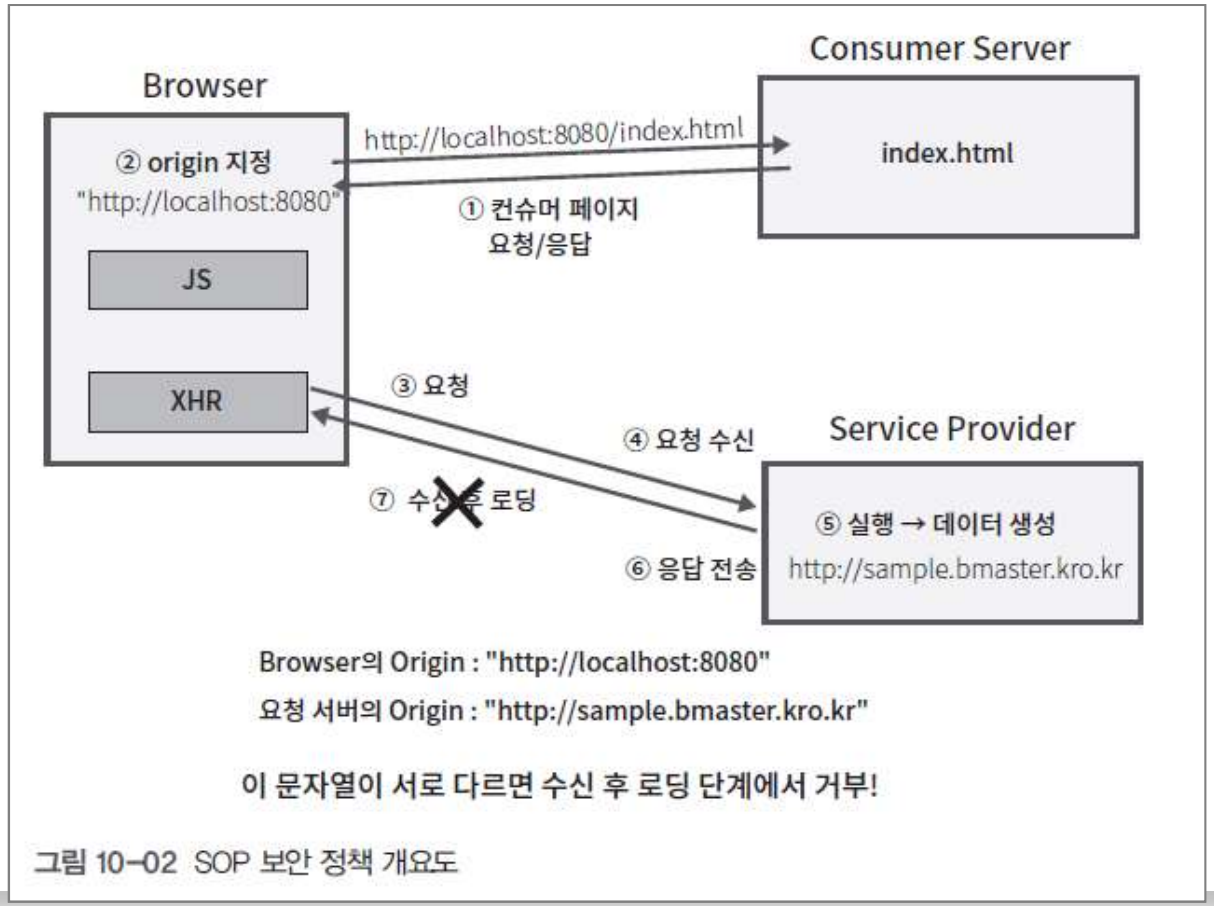


그림 10-02 SOP 보안 정책 개요도

axios 기능 테스트

- 크로스 오리진 문제 해결 방법
  - 컨슈머 서버측에 프록시 요소 생성
  - 서비스 제공자측에서 CORS기능을 제공
  - 서비스 제공자측에서 JSONP기능을 제공
- 컨슈머 서버 프록시

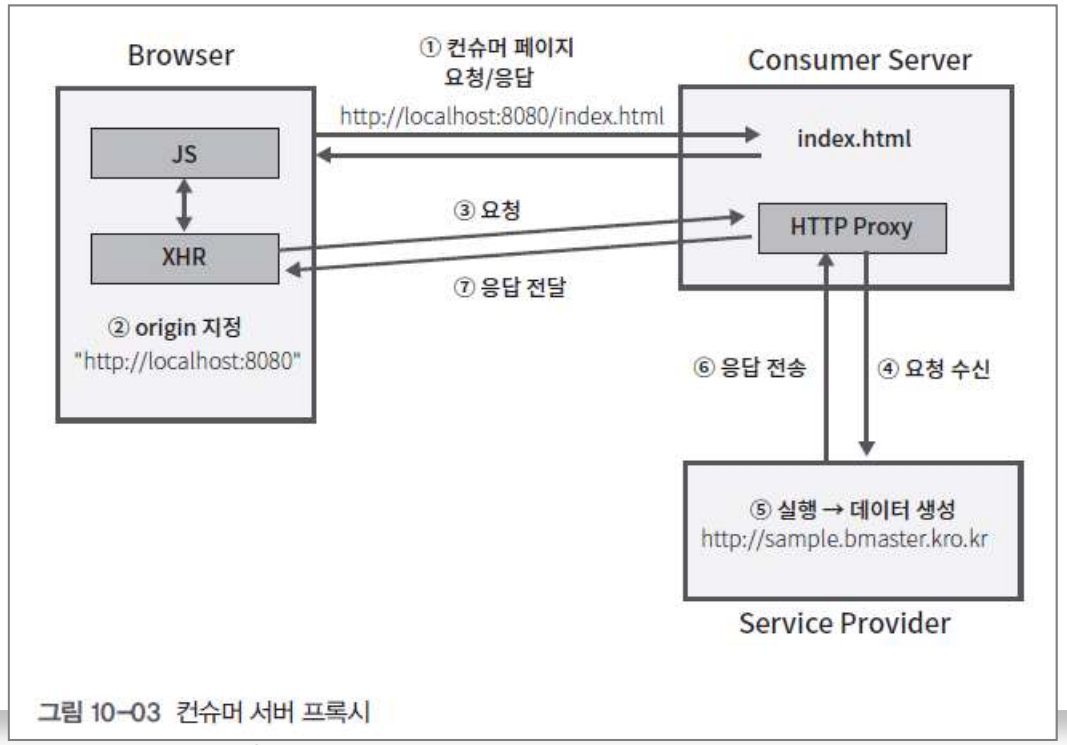


그림 10-03 컨슈머 서버 프록시

axios 기능 테스트

- HTTP 프록시 설정 기능
  - vue.config.js에 proxy 설정

```
module.exports = {
    devServer: {
        proxy: {
            '/api': {
                target: 'http://localhost:3000',
                changeOrigin: true,
                pathRewrite: {
                    '^/api' : ''
                }
            }
        }
    }
}
```

/api/contacts 로 요청
--> http://localhost:3000/contacts로 전달

axios 기능 테스트

axios 사용

- 라이브러리 참조
  - npm install --save axios
  - <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
- API 이용 방법

```
[저수준 API]
axios(config)
axios(url, config)

[각 메소드별 별칭]
axios.get(url[, config])
axios.delete(url[, config])
axios.post(url[, data[, config]])
axios.put(url[, data[, config]])
axios.head(url[, config])
axios.options(url[, config])
```

axios 기능 테스트

- axios 저수준 메서드 & get 메서드

```
axios({
    method : 'GET',
    url : '/api/contacts',
    params : { pageno : 1, pagesize:5 }
})
.then((response) => {
    console.log(response);
    this.result = response.data;
})
.catch((ex)=> {
    console.log("ERROR!!!! : ", ex);
})
```

```
axios.get('/api/contacts', {
  params : { pageno:1, pagesize:5 }
})
.then(...)
.catch(...)
```

```
axios.get('/api/contacts/'+this.no)
.then((response) => {
    console.log(response);
    this.result = response.data;
})
```

axios 기능 테스트

- axios 응답 형식

```
▼Object {data: Object, status: 200, statusText: "OK", headers: Object, config: Object…}
  ▶config: Object
  ▶data: Object
  ▶headers: Object
  ▶request: XMLHttpRequest
   status: 200
   statusText: "OK"
  ▶__proto__: Object
```

- config : 요청 시에 사용된 config 옵션 정보입니다.
- headers : 응답 헤더 정보입니다.
- request : 서버와 통신 시에 사용된 XMLHttpRequest 객체 정보입니다.
- status : HTTP 상태 코드(HTTP Status Code)
- statusText : 서버 상태를 나타내는 문자열 정보입니다.

- status 의 의미

- 2XX : 성공
- 3XX : 리다이렉션
- 4XX : 요청 오류(클라이언트측 오류)
- 5XX : 서버 오류

axios 기능 테스트

- 기타 메서드
  - post 메서드

```
axios.post('/api/contacts',
    { name:this.name, tel:this.tel, address:this.address })
.then((response) => {
    console.log(response);
    this.result = response.data;
    this.no = response.data.no;
})
.catch((ex)=> {
    console.log("ERROR!!!! : ", ex);
})
```

  - put 메서드

```
axios.put('/api/contacts/'+this.no, { name:this.name, tel:this.tel, address:this.address })
.then((response) => {
    console.log(response);
    this.name = '';   this.tel = '';    this.address='';
    this.result = response.data;
})
.catch((ex)=> {
    console.log("ERROR!!!! : ", ex);
})
```

  - delete 메서드 : 생략

axios 기능 테스트

- 파일 업로드 처리
  - 다음 폼을 이용해 파일 업로드를 수행하는 기능이 존재한다고 가정!!

```
<form method="post" enctype="multipart/form-data" action="/contacts/1491586656774/photo">
    <input type="file" name="photo">
    <input type="submit">
</form>
```

  - 파일 업로드 기능 구현을 위해 <input type="file" /> 필드의 실제 DOM 요소를 참조해야 함. (ref 특성 이용)

```
<input type="file" ref="photofile" name="photo" />
```

```
var data = new FormData();
var file = this.$refs.photofile.files[0];
data.append('photo', file);

axios.post('/api/contacts/' +this.no + '/photo', data)
.then((response) => {
    this.result = response.data;
})
.catch((ex) => {
    console.log('updatePhoto failed', ex);
});
```

axios 기능 테스트

- axios 요청과 config 옵션
  - baseURL : 이 옵션을 이용해 공통적인 URL의 앞부분을 미리 등록해두면 요청 시 나머지 부분만을 요청 URL로 전달하면 됩니다. 가능하다면 axios.defaults.baseURL 값을 미리 바꾸는 편이 좋습니다.
  - transformRequest : 요청 데이터를 서버로 전송하기 전에 데이터를 변환하기 위한 함수를 등록합니다.
  - transformResponse : 응답 데이터를 수신한 직후에 데이터를 변환하기 위한 함수를 등록합니다.
  - headers : 요청시에 서버로 전달하고자 하는 HTTP 헤더 정보를 설정합니다.

- Vue 인스턴스에서 axios 이용하기

```
.....
import App from './AppAxiosTest'
import axios from 'axios';

Vue.prototype.$axios = axios;
Vue.config.productionTip = false
```

```
this.$axios.get('/api/contacts/'+this.no)
.then((response) => {
    console.log(response);
    this.result = response.data;
})
```

axios 기능 테스트

axios 사용시 주의사항

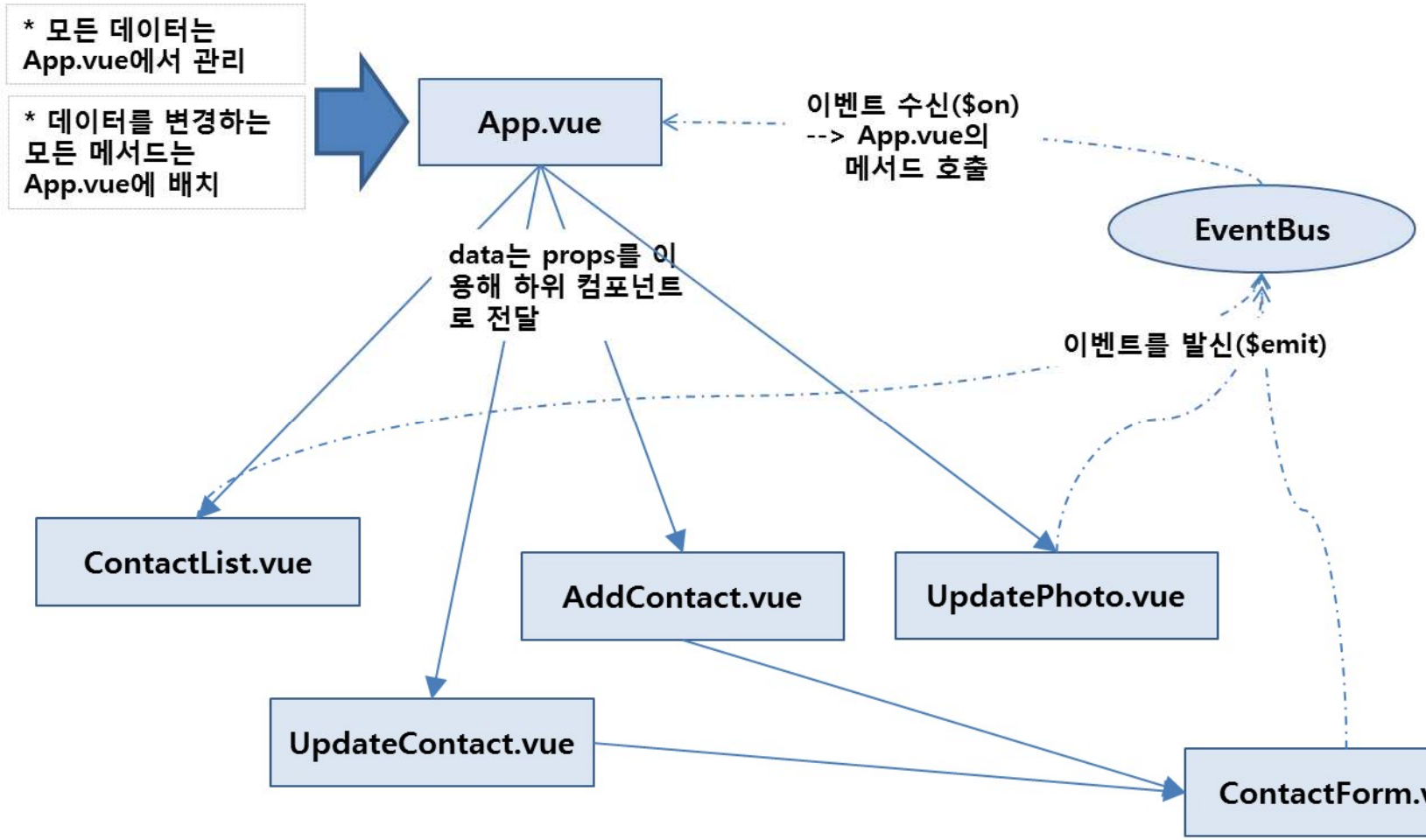
- then() 내부에서 화살표 함수를 사용하지 않을 경우
  - 바깥쪽 스코프에서의 this인 Vue 인스턴스가 then() 내부함수의 this로 전달되지 않음
  - 이 문제를 해결하려면 바깥쪽 스코프의 this를 다른 변수로 저장해둔 다음 클로저 형태로 접근해야 함 --> 번거롭다!!
  - 화살표 함수를 사용하면 Vue 인스턴스가 this로 전달될 수 있음

```
this.$axios.get('/api/contacts/'+this.no)
.then((response) => {
    console.log(response);
    this.result = response.data;
})
```

```
var vm = this;
this.$axios.get('/api/contacts/'+this.no)
.then(function(response) {
    console.log(response);
    this.result = response.data;
})
```

연락처 애플리케이션 예제

전체 구조



연락처 애플리케이션 예제

기초 작업

- src/main.js

```
import Vue from 'vue'
import App from './App'
//import App from './AppAxiosTest'
import axios from 'axios';

Vue.prototype.$axios = axios;
.....(나머지 코드 생략)
```

- src/Config.js, src/EventBus.js

```
var BASE_URL = "/api";

export default {
    PAGESIZE : 5,
    FETCH : BASE_URL + "/contacts",
    ADD : BASE_URL + "/contacts",
    UPDATE : BASE_URL + "/contacts/${no}",
    FETCH_ONE : BASE_URL + "/contacts/${no}",
    DELETE : BASE_URL + "/contacts/${no}",
    UPDATE_PHOTO : BASE_URL + "/contacts/${no}/photo"
}
```

```
import Vue from 'vue';

var vm = new Vue({
    name : "EventBus"
});

export default vm;
```

연락처 애플리케이션 예제

5개의 컴포넌트

표 10-01 컴포넌트별 필요 데이터

컴포넌트	필요 데이터
App.vue	▪ currentView : 동적 컴포넌트로 보여줄 컴포넌트를 지정
ContactList.vue	▪ contactlist : 연락처 목록 데이터
AddContact.vue	
UpdateContact.vue	▪ contact : 연락처 한 건 데이터
ContactForm.vue	▪ mode : 쓰기/수정 여부('add' 또는 'update')
UpdatePhoto.vue	▪ contact : 연락처 한 건 데이터

- 하향식으로 컴포넌트 작성
  - AddContact.vue와 UpdateContact.vue는 ContactForm.vue를 공유

연락처 애플리케이션 예제

관련 라이브러리 추가 설치 & 설정

- 페이징 기능 지원 : vuejs-paginate
- Promise polyfill 설치 : es6-promise
  - IE는 promise 객체를 지원하지 않으므로 polyfill을 설치해야 함.
- 패키지 추가
  - yarn add vuejs-paginate@1.x.x bootstrap@3.x.x es6-promise
  - npm install --save vuejs-paginate@1.x.x bootstrap@3.x.x es6-promise
- src/main.js 변경

```
import Vue from 'vue'
import App from './App.vue'
//import App from './AppAxiosTest.vue'
import axios from 'axios'
import 'bootstrap/dist/css/bootstrap.css'
import ES6Promise from 'es6-promise'
ES6Promise.polyfill()

Vue.prototype.$axios = axios;
```

연락처 애플리케이션 예제

App.vue 작성

- 앱 전체에서 사용하는 모든 상태(데이터)와 메서드(axios 통신 기능 포함)를 App.vue에 배치
- 동적 컴포넌트 사용하여 화면 전환

```
<template>
  <div id="container">
      <div class="page-header">
        <h1 class="text-center">연락처 관리 애플리케이션</h1>
        <p>(Dynamic Component + EventBus + Axios) </p>
      </div>
      <component :is="currentView" :contact="contact"></component>
      <contactList :contactlist="contactlist"></contactList>
  </div>
</template>
```

연락처 애플리케이션 예제

App.vue(이어서)

```
<script>
import ContactList from './components/ContactList';
import AddContact from './components/AddContact';
import UpdateContact from './components/UpdateContact';
import UpdatePhoto from './components/UpdatePhoto';
import CONF from './Config.js';
import eventBus from './EventBus.js';
export default {
    name: 'app',
    components : { ContactList, AddContact, UpdateContact, UpdatePhoto },
    data: function() {
        return {
            currentView : null,
            contact : { no:0, name:'', tel:'', address:'', photo:'' },
            contactlist : {
                pageno:1, pagesize: CONF.PAGESIZE, totalcount:0, contacts:[]
            }
        }
    },
    mounted : function() {

    },
    methods : {

    }
}
</script>
```

연락처 애플리케이션 예제

- App.vue의 메서드에 추가할 내용 : 예제 10-17

표 10-02 필요 기능

메서드명	필요 인자	메서드 기능
pageChanged	page	보여줄 페이지를 변경함. data 속성의 contactlist 정보를 변경 후 fetchContacts 호출하도록 작성. Paginate 컴포넌트에서 이 함수를 바인딩함.
fetchContacts	pageno pagesize	전체 연락처 데이터를 페이징하여 조회함. pageno, pagesize는 data 속성의 contactlist 정보를 활용함.
fetchContactOne	no	일련번호를 이용해 특정 연락처 한 건을 조회함.
addContact	contact	연락처 한 건을 추가함. contact는 객체임
updateContact	contact	연락처 한 건을 수정함. contact는 객체임
deleteContact	no	일련번호를 이용해 연락처 한 건을 삭제함.
updatePhoto	no file	일련번호와 파일 요소 정보를 이용해 사진 파일을 변경함.

연락처 애플리케이션 예제

- mounted 이벤트 훅 : 예제 10-18
  - Event Bus 객체를 통한 이벤트 수신 기능 작성

표 10-03 App.vue에서의 수신 이벤트

이벤트명	전달 인자	설명
addContactForm		연락처 추가 폼이 나타날 수 있도록 currentView를 addContact로 변경함.
editContactForm	no	변경폼에 기존 연락처 데이터가 나타날 수 있도록 no 인자를 이용해 fetchContactOne 메서드를 호출하고, 연락처 변경 폼이 나타날 수 있도록 currentView를 updateContact로 변경함.
editPhoto	no	editContatForm 이벤트와 유사하게 no 인자를 이용해 fetchContactOne 메서드를 호출하고 currentView를 updatePhoto로 변경함.
cancel		모든 입력폼에서 취소 버튼을 클릭했을 때 발생되는 이벤트. currentView를 null로 변경함.
addSubmit	contact	연락처가 추가되는 이벤트. contact 객체를 받아서 addContact 메서드를 호출함. 연락처가 추가되면 입력폼은 사라져야 하므로 currentView를 null로 설정함
updateSubmit	contact	연락처가 수정되는 이벤트. updateContact 메서드를 호출함. 수정폼은 사라지도록 currentView를 null로 설정함.
updatePhoto	no file	파일 정보가 존재할 때 updatePhoto 메서드를 호출하고 사진 변경 폼이 사라질 수 있도록 currentView를 null로 설정함.
deleteContact	no	no를 이용해 deleteContact 메서드를 호출함.

연락처 애플리케이션 예제

ContactList.vue 작성

- App.vue로부터 contactlist 데이터를 props로 전달받아 화면 구성
- Event Bus 객체의 $emit 을 이용해 App.vue로 이벤트 정보 전달

이벤트	전달 인자	설명
addContactForm		'새 연락처 추가' 버튼을 클릭했을 때 입력폼을 나타내기 위한 이벤트
editContactForm	no	조회하고 있는 연락처 리스트 중에서 편집 버튼을 누른 연락처의 no 필드값을 인자로 전달하여 연락처 수정 폼을 나타내기 위한 이벤트
deleteContact	no	일련 번호를 이용해 삭제하기 위한 이벤트
editPhoto	no	조회하고 있는 연락처 리스트에서 사진을 클릭했을 때 no 필드 값을 전달하여 사진 변경 폼을 나타내기 위한 이벤트
pageChanged	page	ContactList.vue 컴포넌트에서 사용하는 vuejs-paginate 컴포넌트에서 페이지가 바뀌면 App.vue로 알려서 처리하기 위한 이벤트

연락처 애플리케이션 예제

- 예제 10-21 : src/components/ContactList.vue

```
<template>
    <div>
    <p class="addnew">
        <button class="btn btn-primary" @click="addContact()">
            새로운 연락처 추가하기</button>
    </p>
    <div id="example">
    <table id="list" class="table table-striped table-bordered table-hover">
        <thead>
            <tr>
                <th>이름</th><th>전화번호</th><th>주소</th><th>사진</th><th>편집/삭제</th>
            </tr>
        </thead>
        <tbody id="contacts" >
            <tr v-for="contact in contactlist.contacts" :key="contact.no">
                <td>{{contact.name}}</td>
                <td>{{contact.tel}}</td>
                <td>{{contact.address}}</td>
                <td><img class="thumbnail" :src="contact.photo"
                    @click="editPhoto(contact.no)" /></td>
                <td>
                    <button class="btn btn-primary"  @click="editContact(contact.no)">편집</button>
                    <button class="btn btn-primary"  @click="deleteContact(contact.no)">삭제</button>
                </td>
            </tr>
        </tbody>
    </table>
```

연락처 애플리케이션 예제

- 예제 10-21 : src/components/ContactList.vue(이어서)

```
    </div>
        <paginate ref="pagebuttons"
            :page-count="totalpage"
            :page-range="7"
            :margin-pages="3"
            :click-handler="pageChanged"
            :prev-text="'이전'"
            :next-text="'다음'"
            :container-class="'pagination'"
            :page-class="'page-item'">
        </paginate>
    </div>
</template>

<script>
import eventBus from '../EventBus';
import Paginate from 'vuejs-paginate';

export default {
    name : 'contactList',
    components : { Paginate },
    props : [ 'contactlist' ],
    computed : {
        totalpage : function() {
            return Math.floor((this.contactlist.totalcount-1) / this.contactlist.pagesize)+1;
        }
    },
```

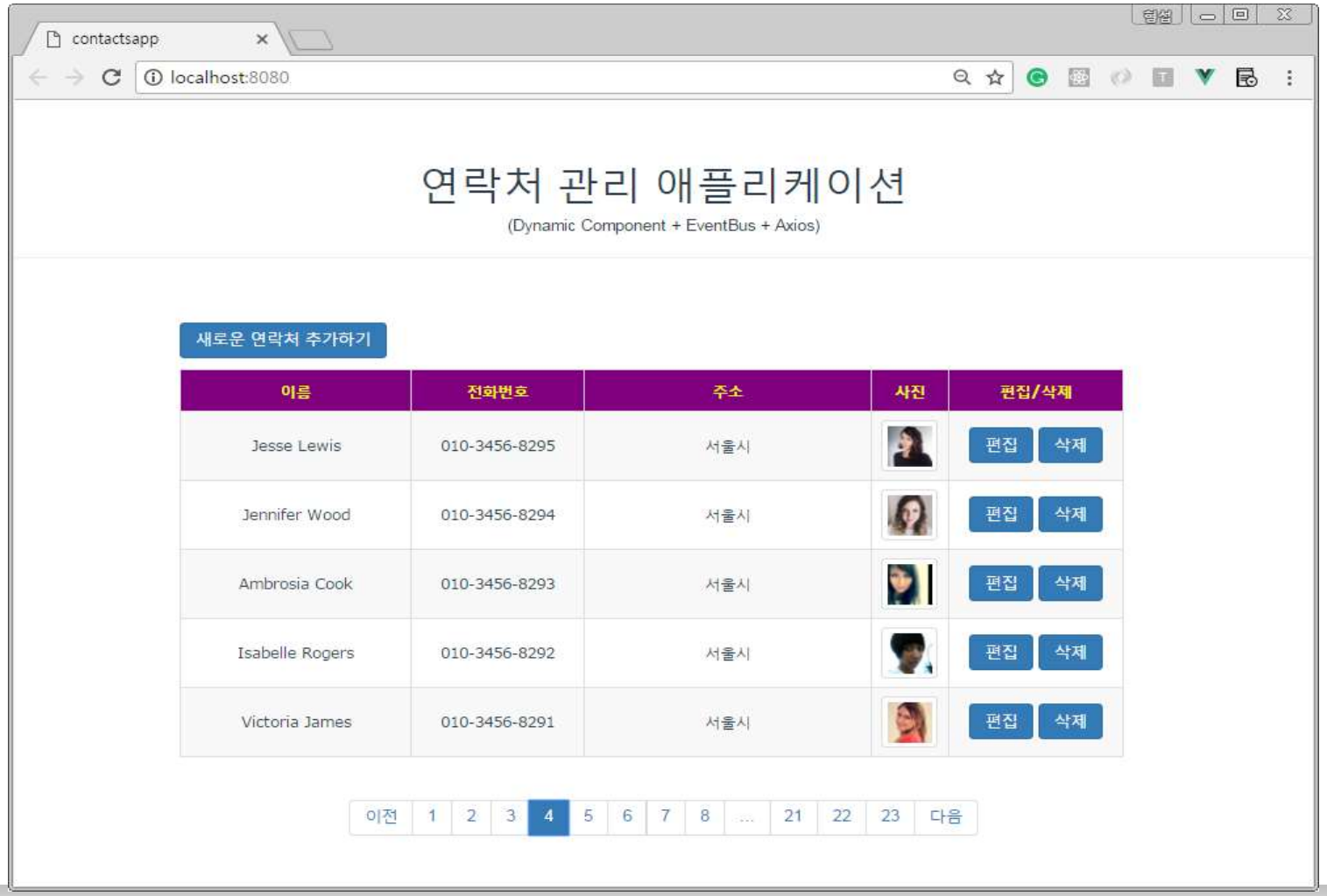
연락처 애플리케이션 예제

- 예제 10-21 : src/components/ContactList.vue(이어서)

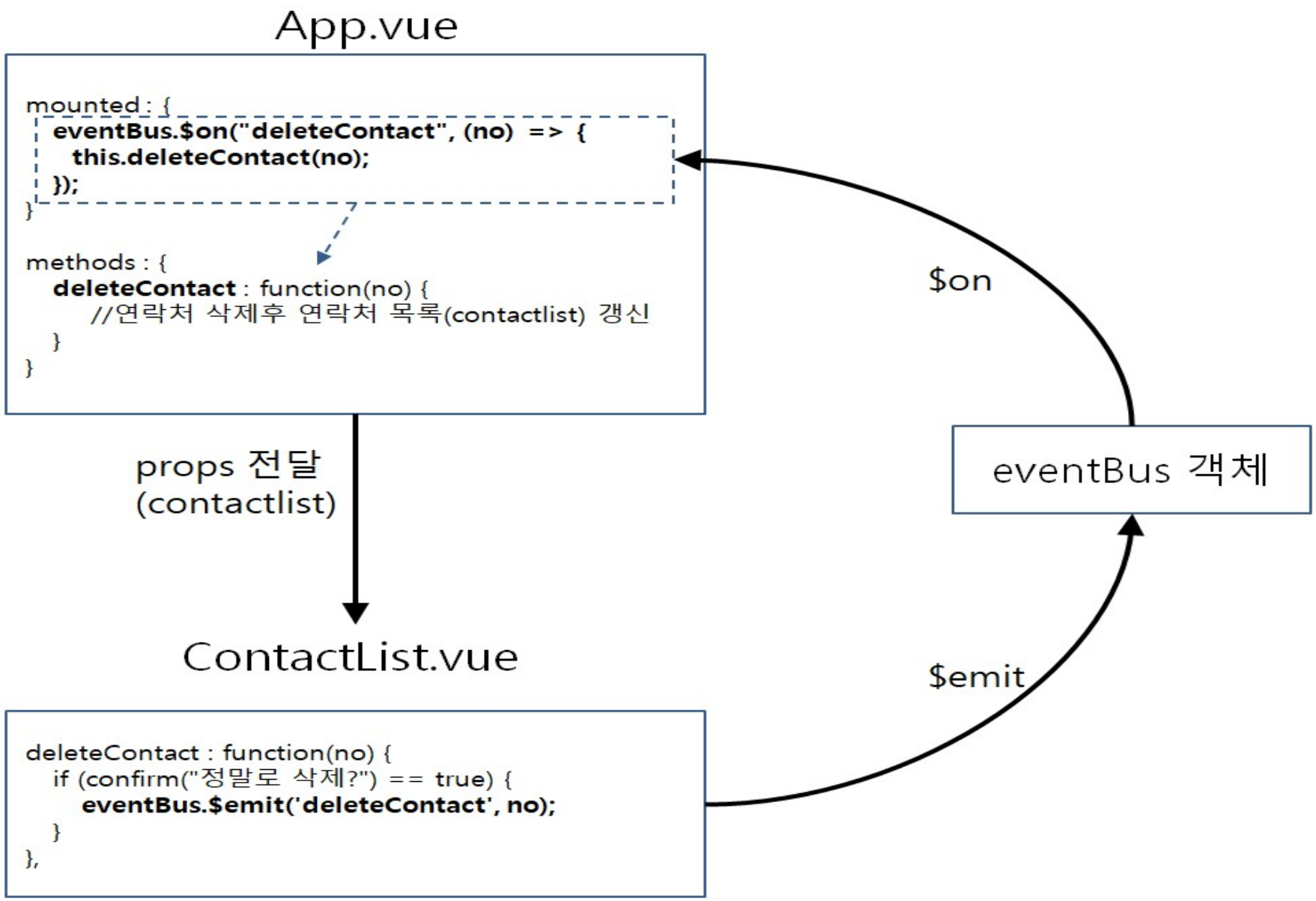
```
    watch : {
        ['contactlist.pageno'] : function() {
            this.$refs.pagebuttons.selected = this.contactlist.pageno-1;
        }
    },
    methods : {
        pageChanged : function(page) {
            eventBus.$emit("pageChanged", page);
        },
        addContact : function() {
            eventBus.$emit("addContactForm");
        },
        editContact : function(no) {
            eventBus.$emit("editContactForm", no)
        },
        deleteContact : function(no) {
            if (confirm("정말로 삭제하시겠습니까?") == true) {
                eventBus.$emit('deleteContact', no);
            }
        },
        editPhoto : function(no) {
            eventBus.$emit("editPhoto", no);
        }
    }
}
</script>
<style scoped>...(생략)</style>
```

연락처 애플리케이션 예제

- 여기까지 실행 결과!!



연락처 애플리케이션 예제



연락처 애플리케이션 예제

입력폼, 수정폼 작성

- 입력, 수정 폼은 유사한 UI이므로 공통부분을 처리할 수 있는 ContactForm.vue를 자식컴포넌트로 사용
- ContactForm.vue에서는 mode props를 전달받아 어느 폼을 보여줄지를 결정하도록 함.
- src/components/AddContact.vue : 예제 10-21

```
<template>
    <contactForm mode="add" />
</template>

<script>
import ContactForm from './ContactForm.vue';

export default {
    name : "addContact",
    components : { ContactForm }
}
</script>
```

연락처 애플리케이션 예제

- src/components/UpdateContact.vue : 예제 10-22

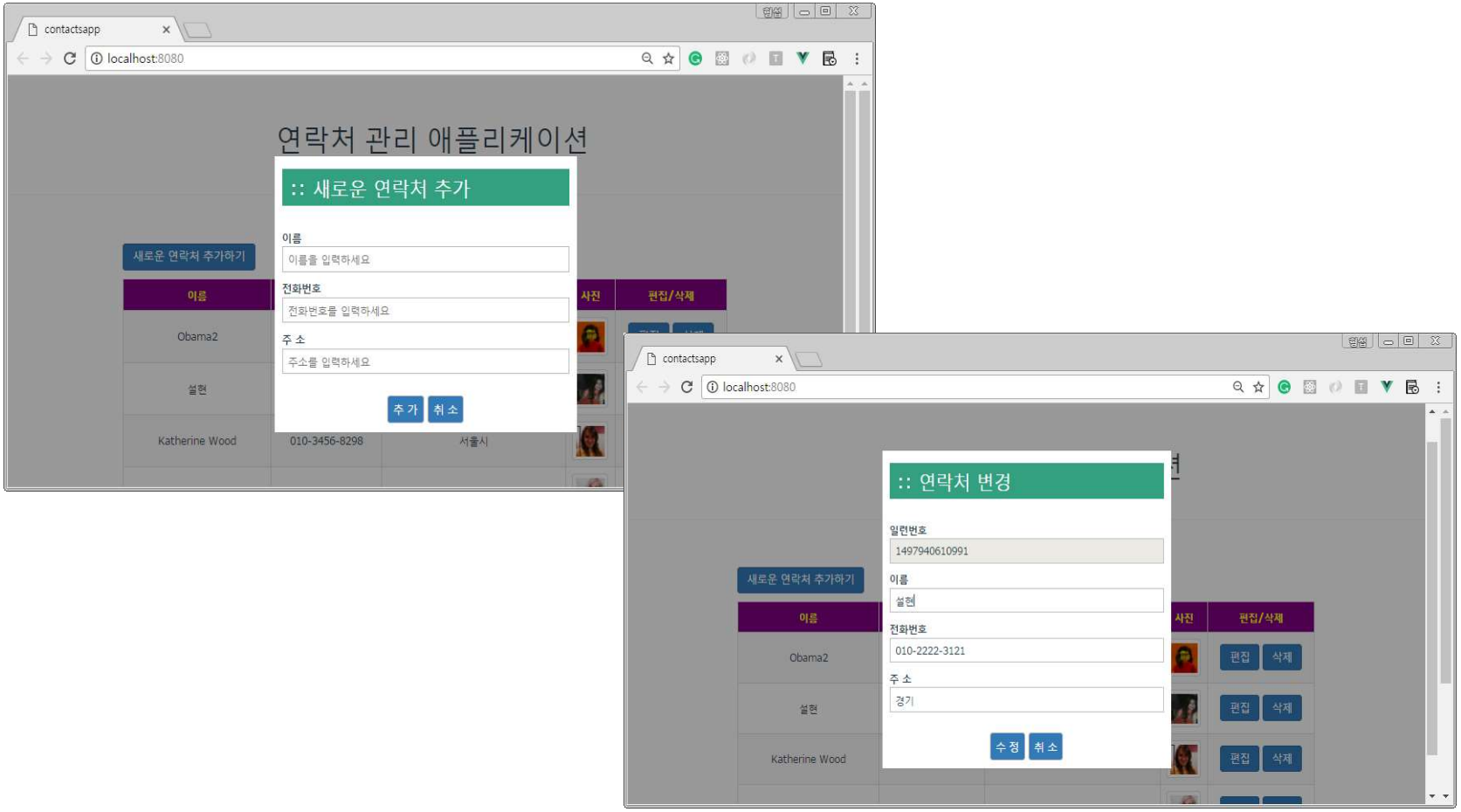
```
<template>
    <contactForm mode="update" :contact="contact" />
</template>

<script>
import ContactForm from './ContactForm.vue';

export default {
    name : "updateContact",
    components : { ContactForm },
    props : [ 'contact' ]
}
</script>
```

연락처 애플리케이션 예제

- src/components/ContactForm.vue : 예제 10-23
  - mode props 값이 add일 때와 update 일때 서로 다른 UI가 나타나도록 함.
    - update : 전달된 contact props 값이 화면에 나타나도록 하고 타이틀과 버튼을 '수정'의 경우로 나타나도록 설정
    - add : 타이틀과 버튼을 '추가'의 경우로 나타나도록 설정
    - 계산형 속성을 적용하여 설정함.
  - 버튼을 클릭했을 때 Event Bus 객체로 $emit 메서드로 이벤트를 발신하여 App.vue의 메서드가 호출될 수 있도록 함.
  - 모달 폼을 구현하기 위해 입력/수정 폼이 나타나고 있는 동안 연락처 목록 화면이 비활성화될 수 있도록 .modal css 클래스를 적용
  - ESC 키를 눌렀을 때도 폼이 사라질 수 있도록 @keyup.esc="cancelEvent" 와 같이 이벤트를 처리함,.

연락처 애플리케이션 예제

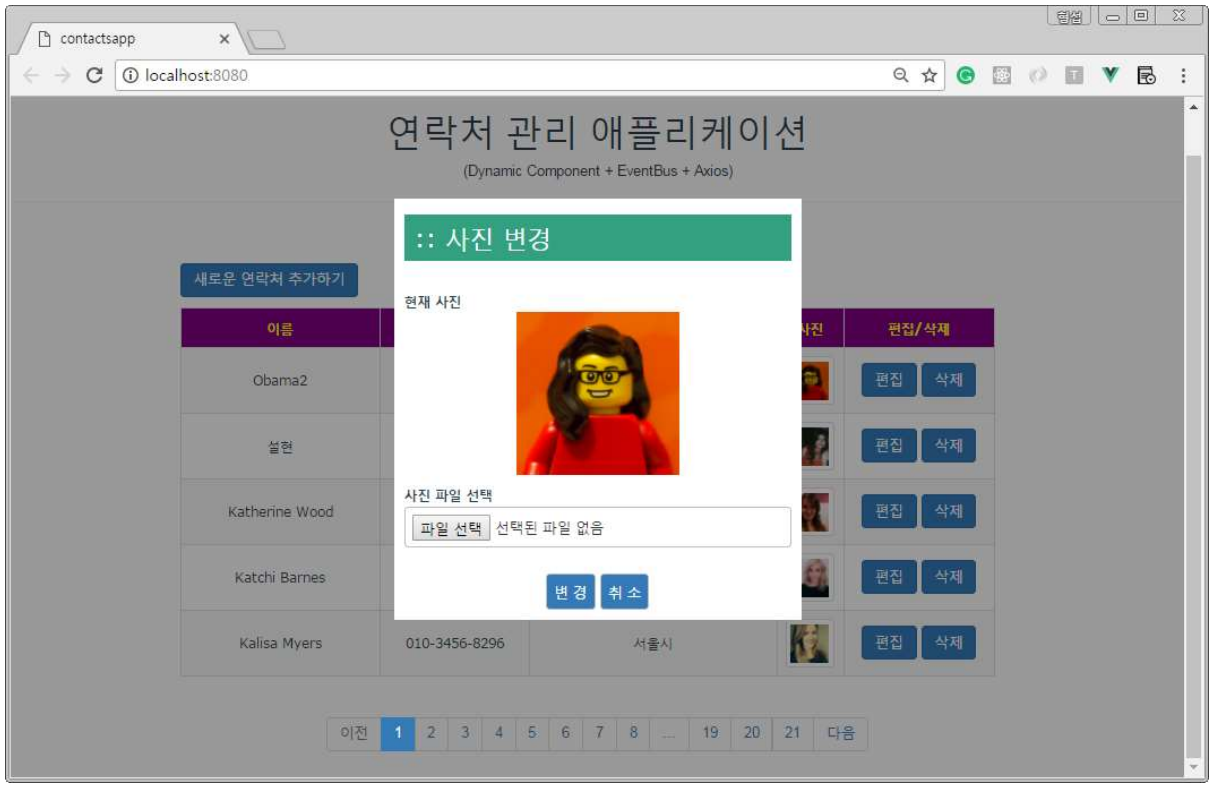
여기까지 실행 결과



연락처 애플리케이션 예제

사진 변경 폼 작성 : 예제 10-24

- 연락처 목록의 섬네일 사진을 클릭하면 현재 사진을 확인하고 사진 파일을 업로드하여 변경할 수 있는 폼을 제공



정리

- axios를 이용하여 HTTP 통신을 수행하는 방법 학습
- 이밖에도 이제까지 배웠던 내용을 적용하여 예제 작성

  - 상위 컴포넌트에서 하위 컴포넌트로 데이터를 전달하기 위해 props를 사용합니다.
  - 데이터가 위치한 곳에 데이터를 변경하는 메서드를 중앙집중화하여 배치합니다.
  - 다른 컴포넌트로 이벤트 정보를 전달하기 위해 이벤트 버스 객체를 사용합니다.
  - 동적 컴포넌트를 이용해 <component> 위치에 여러 컴포넌트를 나타낼 수 있도록 합니다.