1 설치 환경

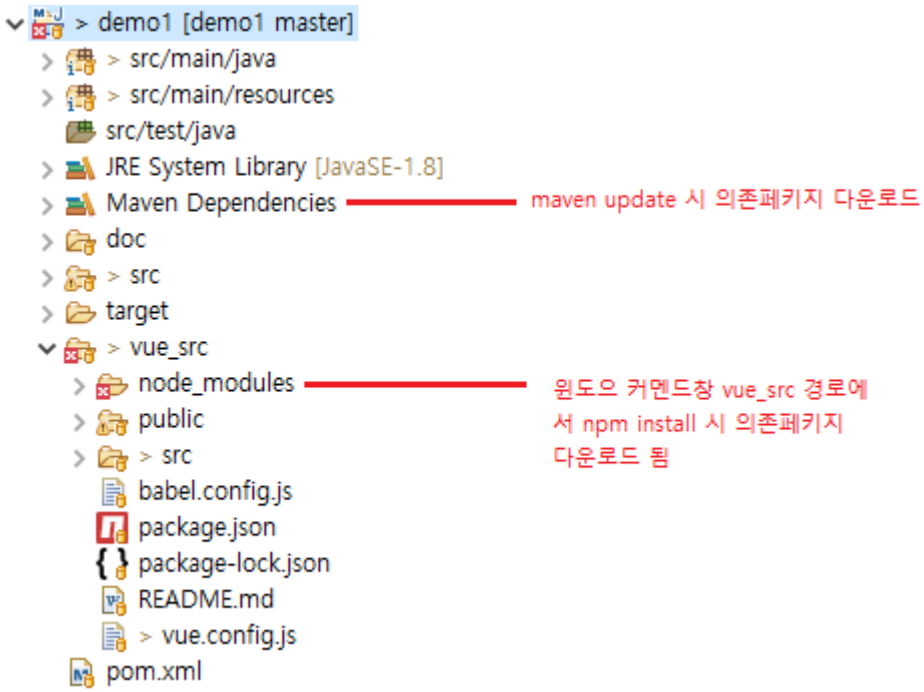
프로그램	버전	비고
Java	1.8.0_291	Oracle Java
eGovFrameDev	3.10.0-64bit	Java IDE
Node.JS	V14.17.5	
VueCLI	@vue/cli 4.5.15	
Visual Studio Code	1.63.2	VueJS IDE
GIT	2.32.0.windows.2	개발자 PC에 설치 형상관리 프로그램

2 설치 순서

① git clone 으로 프로젝트 복사
② eGovFrame 실행 후 복사한 프로젝트 import
③ eGovFrame(Eclipse) > Package Explorer 에서 마우스 오른쪽 클리 > Maven Update Project 수행
④ domo1(프로젝트 루트 폴더)/vue_src 경로를 윈도우 커멘드창으로 실행하여 npm install 수행(vue_src 폴더 밑에 node_modules가 생성되며 의존 페키지를 다운로드하게 된다.)

3 설치 구성

① 디렉토리 구성



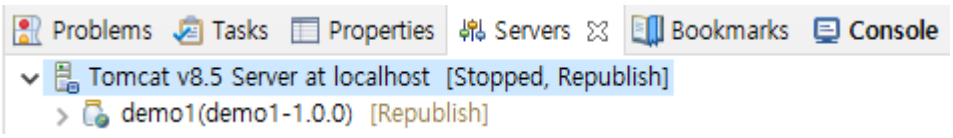
② Java Maven 프로젝트 생성시 부여한 특성
i. eGovFrame Web Project(Generate Example 소스로 시작하여 eGovFrame 기본 기술 최대한 활용)

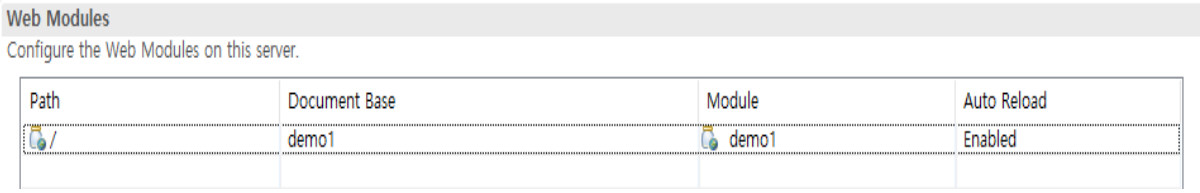
③ Vue 프로젝트 생성시 부여한 특성
i. Vue version 2.X

ii. Bable, Router(history mode), Vuex, CSS Pre-processors(node-sass), Linter/Formatter(ESLinter + Prettier)

4 실행 방법

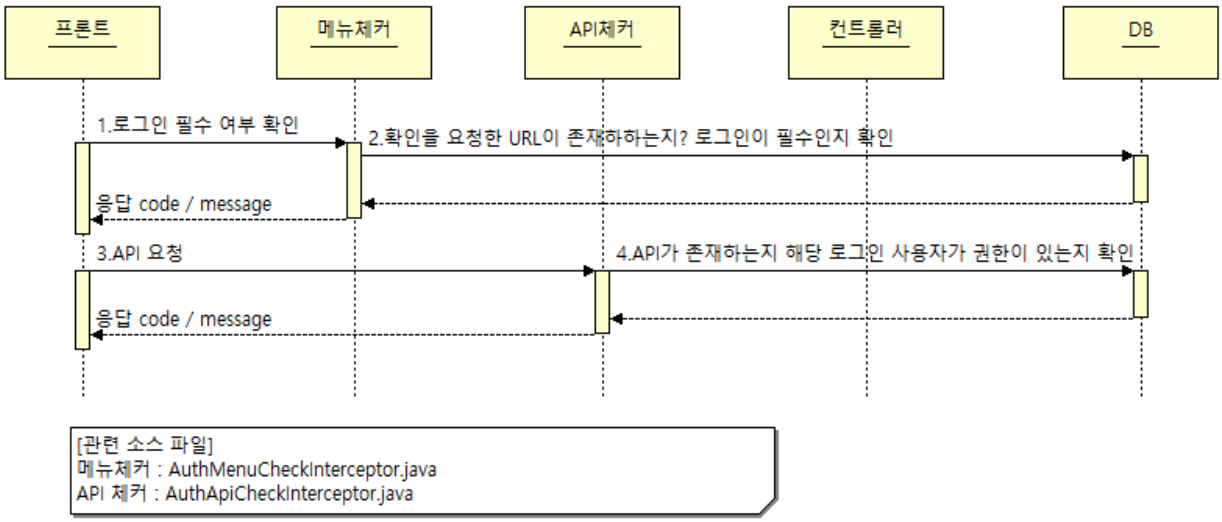
① 프론트 : 윈도우 커멘드 vue_src 폴더에서 npm run serve 실행(8888포트)
② 백엔드 : demo1 프로젝트를 톰캣으로 실행(8080포트)



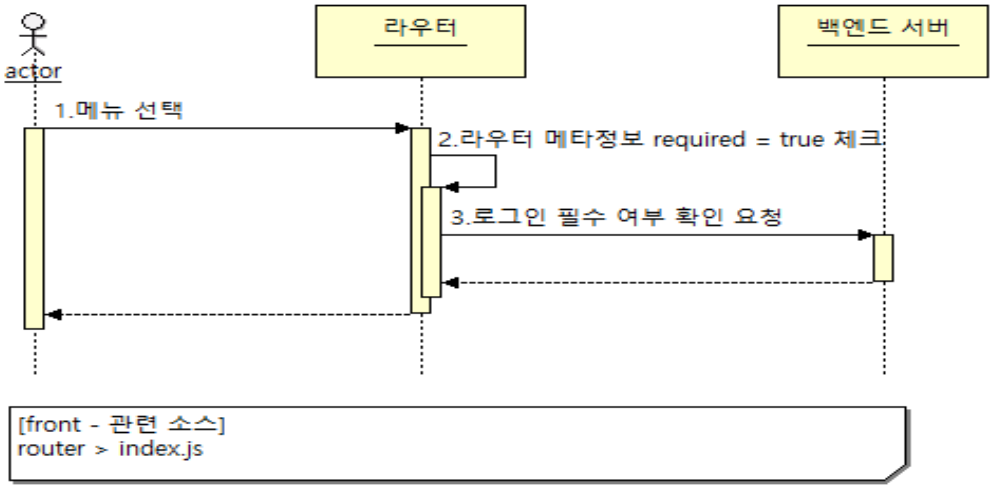


5 기반 기술 설명

① 접근권한 체크를 위한 구조 설명

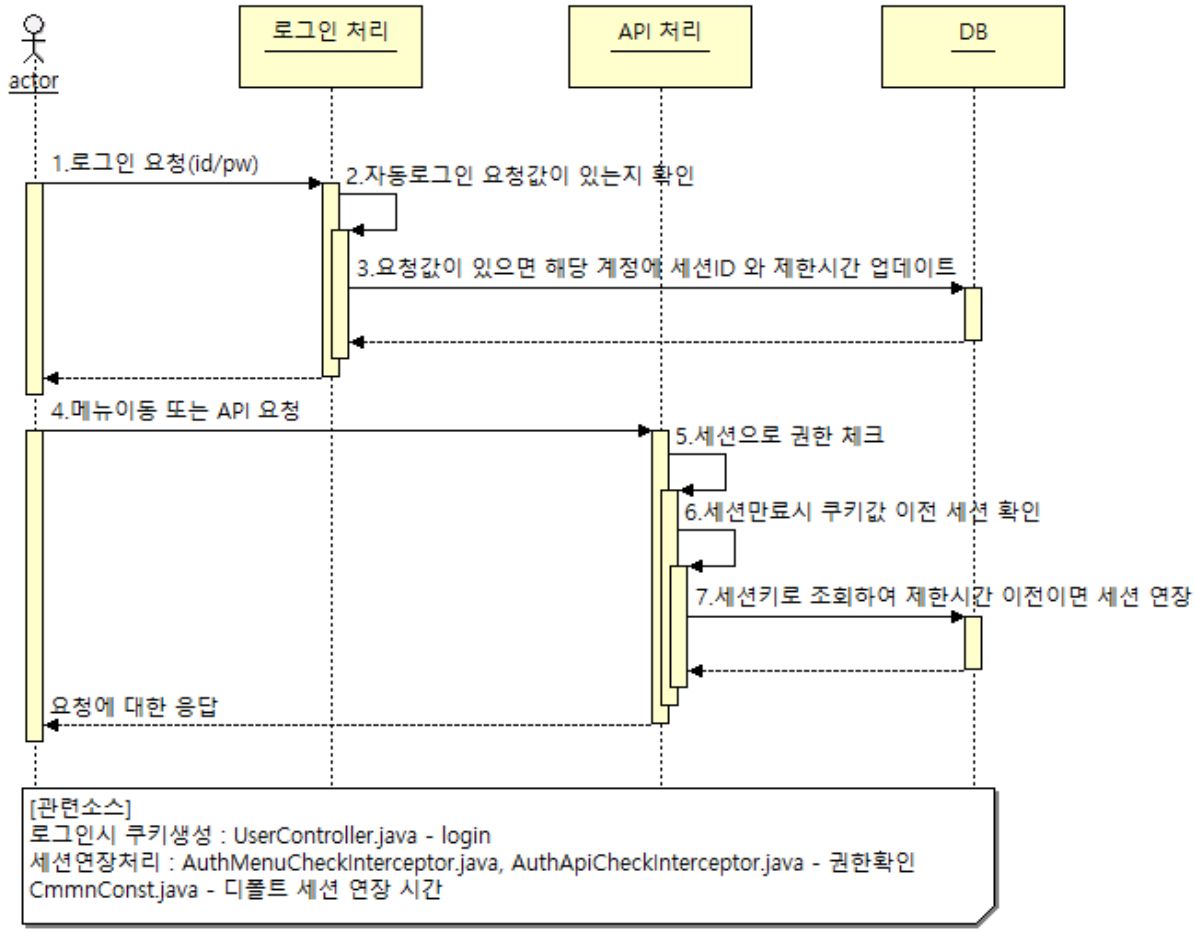


i. 확인방법 : 데모프로젝트 실행 후 상단 메뉴에서 나의정보 클릭
ii. 하단에 “토스트메시지로 로그인 후 이동가능합니다.” 메시지 출력
iii. 이번에는 로그인 메뉴를 선택하고(참고로 로그인 메뉴는 누구나 접근할 수 있는 메뉴임) 아이디 및 비밀번호에 demo1 / 1234 입력 후 로그인 하면 정상 로그인 후 나의정보 페이지로 자동 이동함



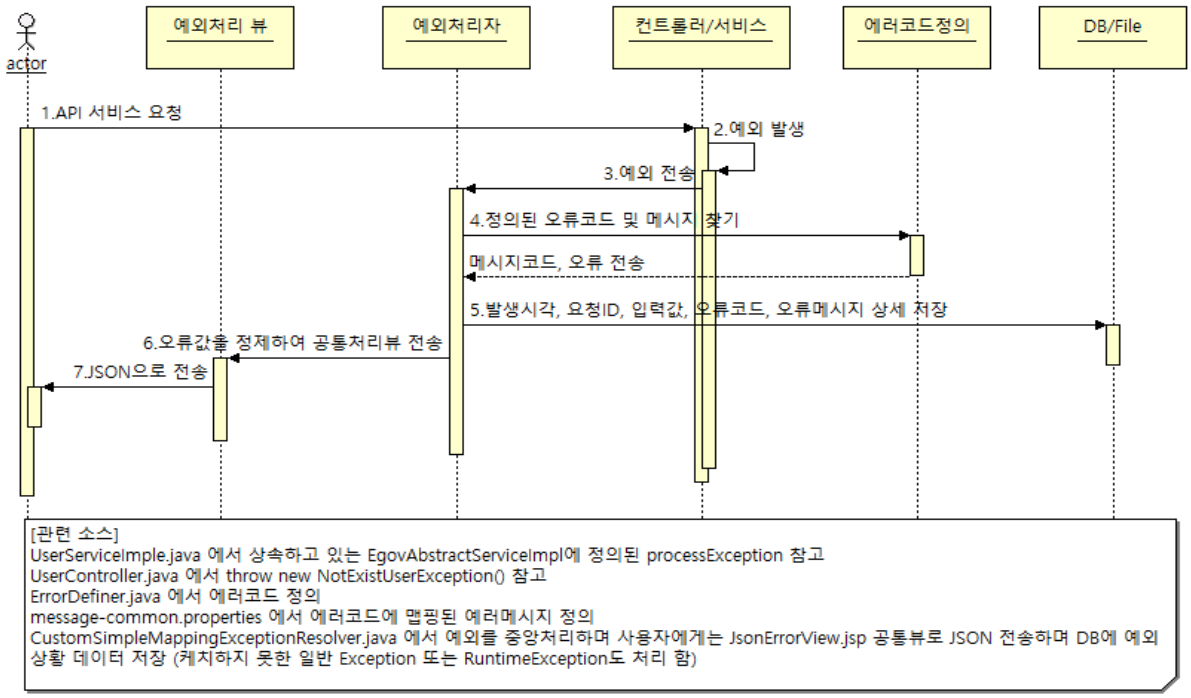
② 세션연장(자동 로그인)

i. 로그인시 자동로그인 체크 후 확인

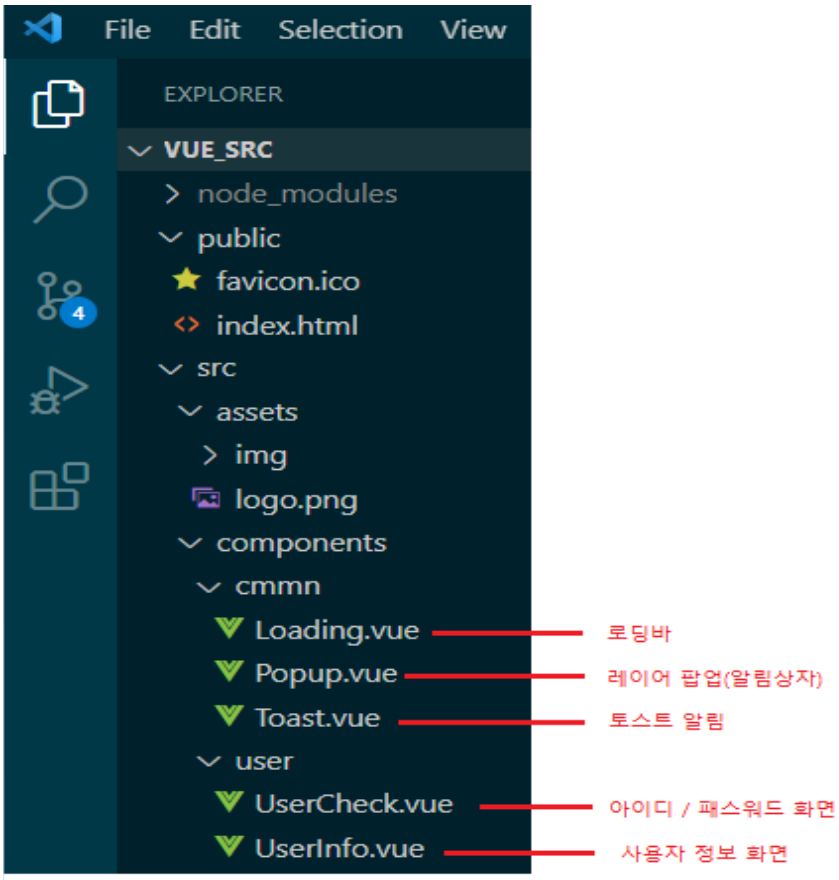


③ 에러, 안내처리 기능

i. 확인방법 : 회원가입시 동일ID로 두번(중복) 등록하면 에러코드/에러메시지 전송되며 DB에 자세한 에러메시지 저장됨



④ 화면에서 많이 사용하는 Component 모듈화



i. 로딩바 : http 통신 전 보이고, http 통신 완료 후 가림 처리
ii. 레이어 팝업 : 오류메시지 등 자세하 알림
iii.토스트 알림 : 간단한 메시지 표시(자동으로 보이고 잠시 후 사라짐)
iv.아이디 패스워드 화면 : 사용자 / 관리자 등 재사용 목적

v. 사용자 정보 화면 : 회원가입 / 나의 정보 등 재사용 목적

A. 사용하는 곳 : 선언부(App.vue – template에 공통 컴포넌트 설정)

```
<template>
  <div id="app">
    <div class="menu-container">
      <router-link class="menu" to="/">HOME</router-link>
      <router-link class="menu" to="/api/user/memberJoin">회원가입</router-link>
      <router-link class="menu" to="/api/user/login">로그인</router-link>
      <router-link class="menu" to="/api/user/myInfo">나의정보</router-link>
    </div>
    <router-view></router-view>
    <loading ref="loading"></loading>
    <toast ref="toast"></toast>
    <popup ref="popup" :popupWidth="40"></popup>
  </div>
</template>
```

App.vue 상단에 전역으로 설정

(App.vue – Vue 최상위 인스턴스 생성시 이벤트 수신기 설정)

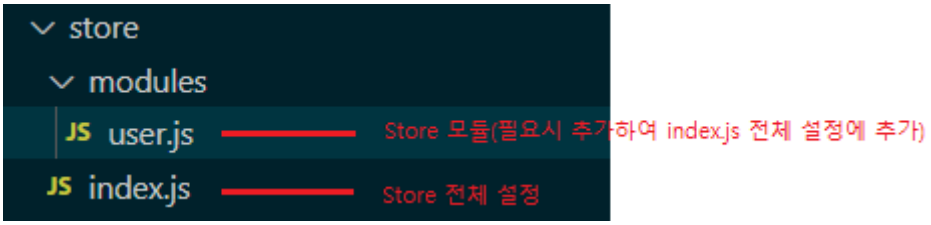
```
created() {
  // 부모객체에서 공통 수신함수 등록
  this.$EventBus.$on("showToast", (msg) => {
    this.showToast(msg);
  });
  this.$EventBus.$on("showLoading", () => {
    this.showLoading();
  });
  this.$EventBus.$on("showAlert", (title, msg, callback) => {
    this.showAlert(title, msg, callback);
  });
```

B. 사용하는 곳 : 이용부(MainView.vue)

```
<script>
export default {
  components: {  },
  methods: {
    showAlert() {
      this.$EventBus.$emit("showAlert", "제목", "내용 : 하단 확인버튼을 클릭하면 파라미터
      함수 실행이 필요할 경우 적절하게 사용하시기 바랍니다.", (data) => { alert(data) })
    }
  }
};
</script>
```

⑤ 로그인 및 세션정보(store) 설정

i. 관련 소스



⑥ 유틸성 함수(팝업, 메시지 처리, 화면이동, 기타 등등)

i. 관련 소스

A. src>mixins>cmmn.js : 뷰 인스턴스 생성시 공통 메서드 정의
B. src>mixins>filter.js : 뷰 인스턴스 생성시 공통 필터 함수(예 천단위 콤마처리)
C. src>mixins>lifeCycle.js : 뷰 생성시 자동 호출되는 함수의 공통 로직 정의
D. src>cmmn>validator.js : 사용자 입력값 유효성 검증처리
E. src>components>cmmn>Loading.vue : http 비동기 통신 로딩뷰
F. src>components>cmmn>Paging.vue : 목록요청시 페이징바 뷰
G. src>components>cmmn>Popup.vue : 오류 및 기타 공통 팝업뷰
H. src>components>cmmn>Toast.vue : 간단한 메시지 뷰

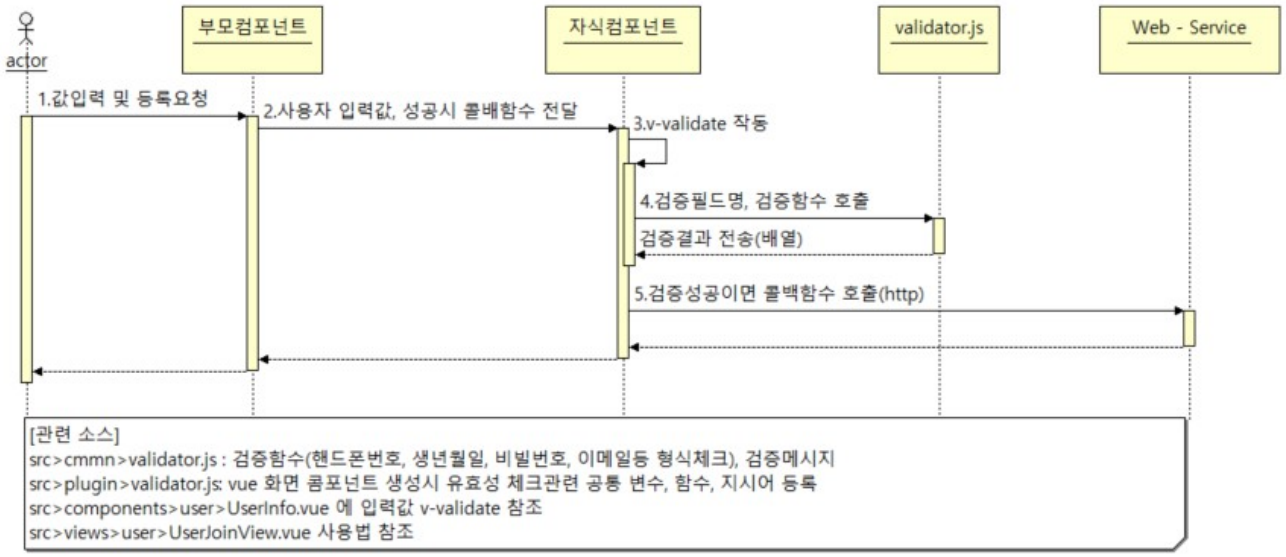
⑦ Component별 코딩 샘플

i. 관련 소스

A. 회원가입, 로그인에서 주요 컴포넌트가 모두 사용되고 있으니 참조 부탁드립니다.

⑧ 필수값 및 정합성 체크 코드

i. 프론트 체크



A. src>cmmn>validator.js

```
const validateFns = {
  required(key, val) {
    if (!val) {
      return `${key}(은)는 필수항목 입니다.`;
    }
  },
  onlyNumber(key, val) {
    if (!/^[0-9]*$/.test(val)) {
      return `${key}(은)는 숫자만 가능합니다.`;
    }
  },
```

B. src>plugins>validator.js

```
import { directives, validator } from "@/cmmn/validator.js";
import Vue from "vue";

const plugin = {
  install(Vue) {
    Vue.directive("validate", directives.validate);

    Vue.mixin({
      data() {
        return { errorBag: {} };
      },
      computed: {
        $errors() {…
        },
        $validator() {…
        },
      },
    });
  },
};

Vue.use(plugin);
export default plugin;
```

C. src>components>user>UserInfo.vue

```
<template>
  <div class="userinfo-container">
    <slot name="header"></slot>
    <div>
      <div class="row">
        <label class="item text-label">아이디 :</label
        ><input
          class="item text-field"
          type="text"
          :disabled="isDisabled === 1"
          v-model="user.id"
          name="user.id"
          v-validate="'required|alphaNum'"
        />

<p class="input-err" v-if="$errors.has('user.id')">
  {{ $errors.first("user.id") }}
</p>
<p class="input-err" v-if="$errors.has('user.passwd')">
  {{ $errors.first("user.passwd") }}
</p>
```

유효성 체크 필드

유효성 체크 함수

D. src>views>UserJoinView.vue

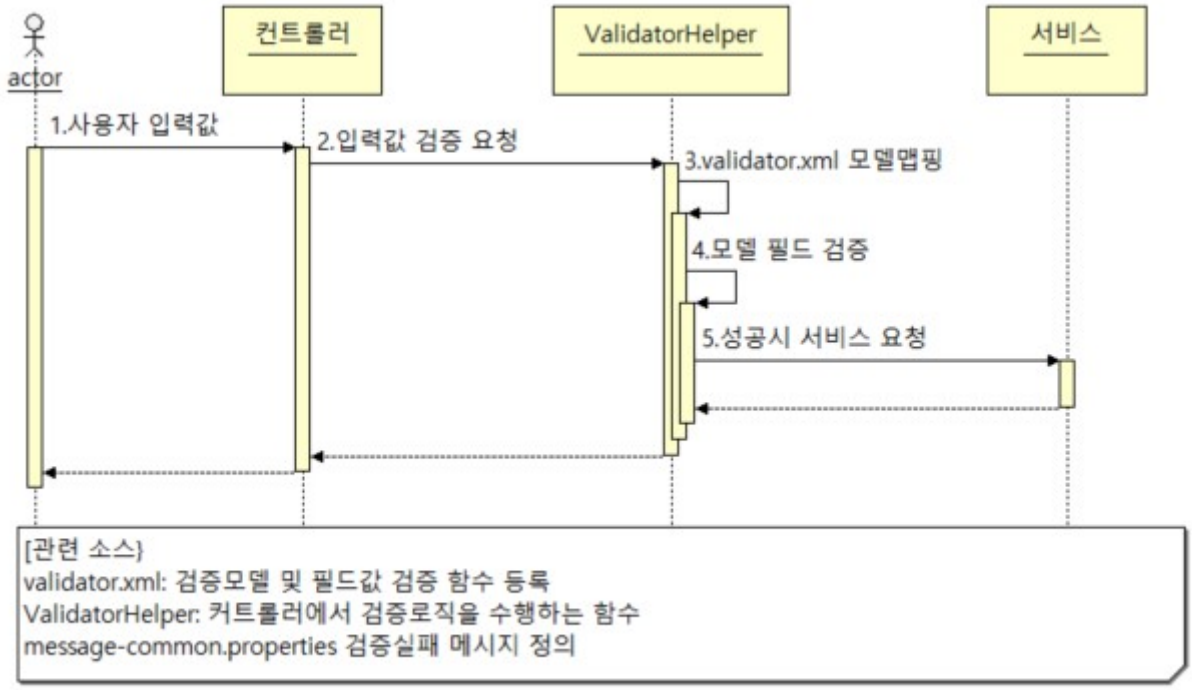
```
joinUserWrap() {
  this.$refs.userInfo.validate(() => {
    this.joinUser({ user: this.$refs.userInfo.getUserInfo });
  });
},
```

유효성 체크 동작

검증 성공시 콜백호출

ii. 서버 체크

A. 유효성 체크 흐름



B. validator.xml

```
<formset>
    <form name="userVO">
        <field property="id" depends="required, mask">
            <arg0 key="user.id" />
            <var>
                <var-name>mask</var-name>
                <var-value>^[0-9a-zA-Z]*$</var-value>
            </var>
        </field>
        <field property="passwd" depends="required, mask">
        <field property="name" depends="required, korean">
        <field property="birthday" depends="date">
        <field property="cellPhoneNo" depends="required, mask">
        <field property="email" depends="required, email">
    </form>
</formset>
```

C. UserController.java

```java
@RequestMapping("/api/user/joinUser.do")
@ResponseBody
public PackingVO joinUser(@RequestBody UserVO userVO, BindingResult bindingResult) throws Exception {

    Logger.debug("userVO : {}", userVO);

    PackingVO pack = getPack("0000", "회원가입완료", userVO);
    pack.setId(userVO.getId());

    /* 입력값 유효성 체크
     * 실패시 에러코드 1000에 유효성 체크에 실패한 항목(필드)관련 오류 메시지 전송
     */
    if (!ValidatorHelper.validate(messageSource, beanValidator, userVO, bindingResult, pack)) {
        return pack;
    } else {
        // 유효성 체크 성공시 정상 프로세스 진행
        userService.insertUser(userVO);
    }

    return pack;

}
```

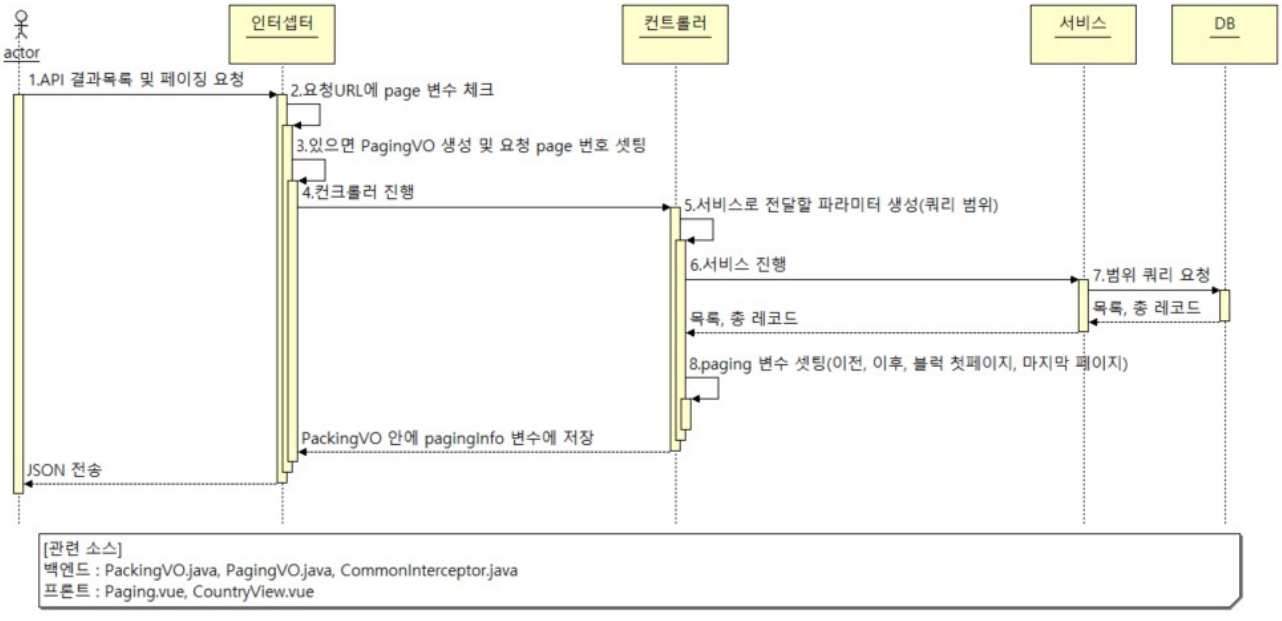
⑨ 트랜잭션(CRUD)처리(동기, 비동기) 및 결과처리(팝업알림, 오류처리)

i. 관련 소스

A. 회원가입, 로그인, 나의정보 에서 주요 컴포넌트가 모두 사용되고 있으니 참조 부탁드립니다.

⑩ 페이징 처리 샘플

i. 백엔드 처리과정



ii. 프론트 관련 소스

A. CountryView.vue

```vue
<template>
  <div class="box">
    <div class="search-bar">
      <label class="item" for="countryNm">국가명:</label>
      <input class="item" id="countryNm" type="text" v-model="countryNm" />
      <button class="item" @click="search(1)">조회</button>
    </div>
    <table>…
    </table>
    <paging ref="paging"></paging>
  </div>
</template>
```

```vue
methods: {
  search(pageNo) {
    window
      .axios({
        method: "POST",
        url: "/api/etc/countryList.do?page=" + pageNo,
        data: { countryNm: this.countryNm },
      })
      .then((res) => {
        console.log(res);
        this.countries = res.data.output;
        this.$refs.paging.setPagingInfo(res.data.pagingInfo);

      })
      .catch((error) => {
        console.error(error);
      });
  },
},
```

화면에 보여줄 범위 목록

화면에 보여줄 페이징 처리

B. Paging.vue

```vue
methods: {
  setPagingInfo(pagingInfo) {
    Object.assign(this.pagingInfo, pagingInfo);
  },
  search(n) {
    this.$EventBus.$emit("search", n);
  },
```

부모에서 조회 후 페이징 변수 셋업

페이징 버튼 선택시 부모뷰에 정의한 조회함수 콜

⑪ computed와 watch기능 샘플

i. 관련 소스

A. 회원가입 사용자 입력값 유효성 체크에서 computed가 사용되고 있습니다.

B. watch기능은 MTS 관련 뷰에서 데이터 변경시 실시간 변경 감지 및 처리 관련하여 작성 예정입니다.