개발자 매뉴얼

-[기말 최종과제] 프로젝트 명: OurDiary

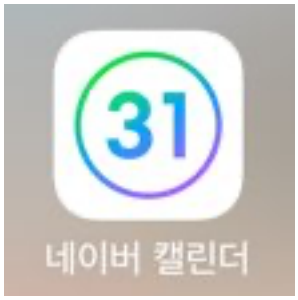
2017301040
컴퓨터과학과
신가은

목차

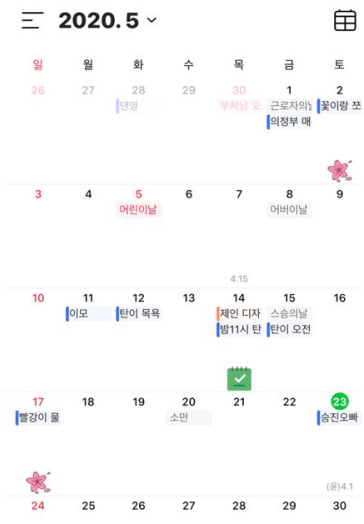
1. 프로젝트 소개 및 목적

Our Diary 프로젝트는 기존의 다이어리와 유사하게 날짜별 일기를 기록하는 기능을 제공합니다.
기존의 다이어리와 달리, 컴퓨터 내 파일을 별도 저장하여 관리하기 때문에 유실가능성이 적고, 키워드 검색을 통해 편의성면에서 효율적이라는 장점이 있습니다.
스프링부트 서버를 이용하여, 소스코드 외부 스프링 컨테이너의 의존성 주입을 통해 프로그램의 유지보수에 용이하도록 작성하였습니다.
또한, 회원과 일기의 데이터 관리는 MySQL 데이터베이스를 이용하여 관리하였습니다.

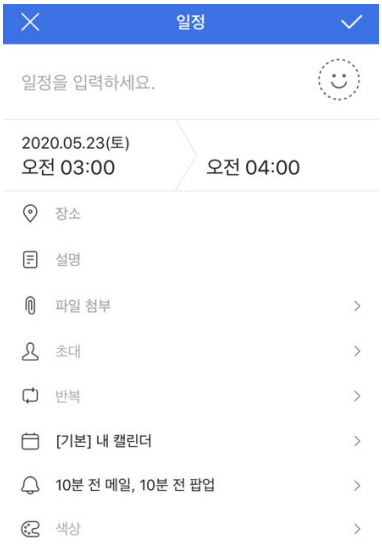
2. 유사시스템에 대한 벤치마킹



네이버 캘린더 앱



네이버 캘린더 앱의 월 캘린더



네이버 캘린더 앱의 날짜별 일정 관리

Our Diary 프로젝트는 네이버 캘린더 앱을 벤치마킹하였습니다.
네이버 캘린더 앱에서 제공하는 한 면에 월별로 일정을 기록할 수 있는 “월 캘린더”와 일 별로 일정, 날씨를 기록할 수 있는 “날짜별 일정 관리”에서 착안하여, Our Diary 프로젝트의 “일기 구간 검색기능”과 “오늘의 일기기능”을 구현하였습니다.
또한, 네이버 캘린더 앱과는 달리, 일기 키워드 검색하기 기능을 통해 사용자가 이전에 작성한 일기의 제목, 내용에 해당 키워드를 포함하는 경우 그 리스트를 조회할 수 있는 기능을 추가 구현하였습니다.

3. 주요기능

(1) 로그인/회원가입기능을 통한 Our Diary프로젝트의 다중 사용자의 이용
(2) 사용자 별 회원정보(email, password, name, phone, profile) 생성, 수정 기능
(3) 일기 정보(날짜, 제목, 날씨, 내용, 등록일, 수정일) 생성, 수정, 삭제, 조회 기능
(4) 키워드, 월별 검색을 통한 일기 리스트 조회 기능

4. 데이터베이스 내부 구조

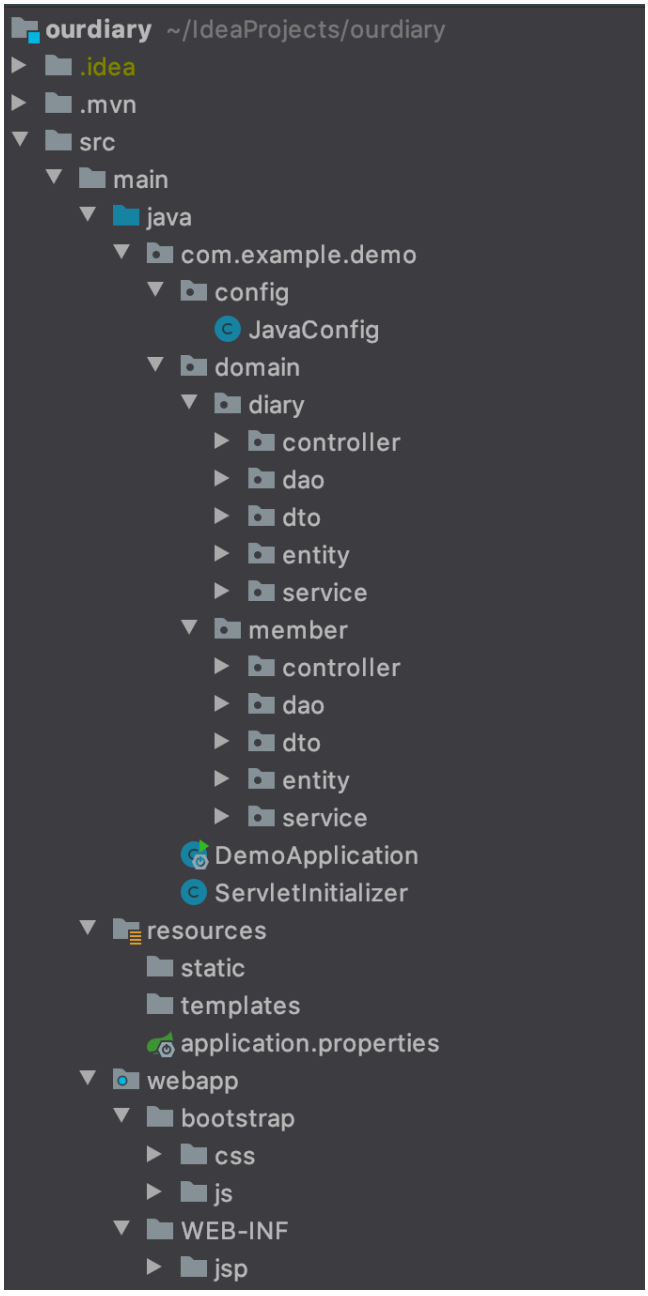
(1) Member 테이블

Column	Type	Nullable	Indexes
id	int(11)	NO	PRIMARY
email	varchar(45)	NO	
password	varchar(45)	NO	
name	varchar(45)	YES	
phone	varchar(45)	YES	
profile	varchar(200)	YES	

(2) Diary 테이블

Column	Type	Nullable	Indexes
id	int(11)	NO	PRIMARY
writer	varchar(45)	NO	
date	datetime	NO	
title	varchar(45)	YES	
content	varchar(1000)	YES	
weather	varchar(45)	YES	
create_at	datetime	YES	
modify_at	datetime	YES	

5. 내부 폴더 구조


ourdiary ~/IdeaProjects/ourdiary
.idea
.mvn
src
main
java
com.example.demo
config
JavaConfig
domain
diary
controller
dao
dto
entity
service
member
controller
dao
dto
entity
service
DemoApplication
ServletInitializer
resources
static
templates
application.properties
webapp
bootstrap
css
js
WEB-INF
jsp

6. 시스템 구조 및 설명

6.1. main/java폴더

6.1.1 config 폴더
JavaConfig.java

```java
@Configuration
public class JavaConfig {
    @Bean(destroyMethod="close")
    public DataSource dataSource() {
        DataSource ds = new DataSource();
        ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
        ds.setUrl("jdbc:mysql://localhost/mydiary?characterEncoding=utf8&serverTimezone=UTC");
        ds.setUsername("root");
        ds.setPassword("1234");
        ds.setMaxActive(10);
        ds.setTestWhileIdle(true);
        ds.setMinEvictableIdleTimeMillis(60000*3);
        ds.setTimeBetweenEvictionRunsMillis(10*1000);
        return ds;
    }

    @Bean
    public MemberDao memberDao() { return new MemberDao(dataSource()); }
    @Bean
    public LoginService loginService() { return new LoginService(); }
    @Bean
    public SignUpService signUpService() { return new SignUpService(); }
    @Bean
    public GetMemberService getMemberService() { return new GetMemberService(); }
    @Bean
    public UpdateMemberService updateMemberService() { return new UpdateMemberService(); }
    @Bean
    public DeleteMemberService memberService() { return new DeleteMemberService(); }
    @Bean
    public DiaryDao diaryDao() { return new DiaryDao(dataSource()); }
    @Bean
    public WriteDiaryService writeDiaryService() { return new WriteDiaryService(); }
    @Bean
    public GetDiaryService getDiaryService() { return new GetDiaryService(); }
    @Bean
    public UpdateDiaryService updateDiaryService() { return new UpdateDiaryService(); }
    @Bean
    public DeleteDiaryService deleteDiaryService() { return new DeleteDiaryService(); }

}
```

- Spring JDBC를 이용해서 데이터베이스 연결하는 DataSource빈 등록
-데이터베이스를 직접 접근하는 클래스를 빈 등록
-서비스단을 빈 등록

6.1.2 domain/diary
(1) controller폴더

- DiaryController.java
:다이어리 관리 기능을 수행하기 위한 Controller

매핑정보

@GetMapping("/diary/detail")	다이어리 id(PK)로 조회해서 GetDiaryService에서 다이어리 정보를 GetDiaryDto에 반환받고, jsp 페이지에 전달한다. 다이어리 상세보기 페이지로 이동한다.
@GetMapping("/diary/detail/info")	다이어리 id(PK)로 조회해서 GetDiaryService에서 다이어리 정보를 GetDiaryDto에 반환받고, jsp 페이지에 전달한다. 다이어리를 수정하는 페이지로 이동한다.
@PostMapping("/diaryCheck")	다이어리 id(PK)로 조회한 다이어리를 수정 후, UpdateDiaryService 로 보내서 데이터베이스에 수정한다. jsp에 수정에 성공하면, UPDATE_OK 속성에 success를 보낸다.
@GetMapping("/diary")	다이어리 작성하는 폼 페이지로 이동
@PostMapping("/diary")	작성된 다이어리를 WriteDiaryService에 보내서 데이터베이스에 삽입을 성공한다. jsp에 WRITE_OK 속성에 success를 보낸다. 실패하면 WRITE_OK 속성에 IsAlreadyDiary를 보낸다.
@GetMapping("/diary/detail/delete")	다이어리 id(PK)를 DeleteDiaryService에 보내서 데이터베이스 값을 찾아 해당 일기를 삭제한후, 성공하면 jsp에 DELETE_OK속성에 success를 보낸다.
@PostMapping("/diary/list")	session에 저장된 email값을 GetDiaryService에 보내서 해당 멤버의 일기리스트를 가져온다. 가져온 리스트와 입력된 키워드를 jsp에 속성값으로 보낸다.

- MainHomeController.java
:메인홈 네비게이터 기능을 수행하기 위한 Controller

매핑정보

@GetMapping("/mainhome")	session에 저장된 email값을 GetDiaryService로 보내서 일기 리스트를 반환받는다. 리스트에 값이 있으면 jsp에 보낸다.
@GetMapping("/logout")	jsp로 이동한다.

(2) dao폴더

- DiaryDao.java
  :데이터베이스에 직접접근하는 기능을 수행하기 위한 Dao
  :JdbcTemplate 의존관계를 주입

public GetDiaryDto selectById (Long id){
"select * from diary where id=? order by date asc" 쿼리문으로 id로 다이어리 정보를 반환한다.
public GetDiaryDto selectByWriterAndDate (String writer, LocalDate date){
"select * from diary where writer=? and date=? order by date asc" 쿼리문으로 wirter와 date로 다이어리 정보를 반환한다.
public List<GetDiaryDto> selectAllByWriter(String writer) {
"select id, date, title, content, weather, create_at, modify_at from diary where writer=?" 쿼리문으로 writer로 다이어리를 찾아 다이어리 리스트를 반환한다.
public List<GetDiaryDto> selectAllByWriterAndKeyword (String writer, String keyword){
"select id, date, title from diary where writer=? and title like ? or content like ? or weather like ?"로 wrtier와 keyword로 다이어리를 찾아 다이어리 리스트를 반환한다.
public List<GetDiaryDto> selectAllByWriterAndSection (SearchDiaryDto searchDiaryDto) {
"select date, title from diary where writer=? and date(date) >= ? and date(date) < ? order by date asc"로 writer와 section에 일치하는 다이어리 리스트를 반환한다. section값을 year,month만 받기때문에 day를 1로 지정해주었다.
public Boolean insert (WriteDiaryDto writeDiaryDto){
"insert diary (writer,date,title,content,weather, create_at, modify_at) values (?,?,?,?,?,?,?)" 쿼리문으로 다이어리를 데이터베이스에 삽입한다.
public void update (Long id, UpdateDiaryDto updateDiaryDto) {
"update diary set title=?, content=?, weather=?, modify_at=? where id=?" 쿼리문으로 id로 다이어리를 찾아 데이터베이스의 해당다이어리의 정보를 수정한다.
public void delete (Long id) {
"delete from diary where id=?" 쿼리문으로 id로 다이어리를 찾아 데이터베이스의 해당 다이어리 정보를 삭제한다.
public LocalDate stringToLocalDate (String str){
String 형식으로 입력된 날짜값(2020-06-26)값을 LocalDate형식으로 바꾸어 반환한다.

(3) dto폴더

- GetDiaryDto.java
  :다이어리 조회 시 사용되는 DTO

id	다이어리 id (PK)
date	다이어리에 작성된 날짜
title	다이어리에 작성된 제목
content	다이어리에 작성된 내용
weather	다이어리에 작성된 날씨
create_at	다이어리를 작성한 등록일
modify_at	다이어를 마지막으로 수정한 등록일 (default: create_at과 같음)

- SearchDiaryDto.java
  :다이어리 검색 시 사용되는 DTO

writer	다이어리를 찾는 계정아이디
date	검색할 다이어리에 작성된 날짜

- UpdateDiaryDto.java
  :다이어리 수정 시 사용되는 DTO

id	수정할 다이어리 id (PK)
date	다이어리 날짜 수정 내용
title	다이어리 제목 수정 내용
content	다이어리내용 수정 내용
weather	다이어리 날씨 수정 내용
create_at	다이어리를 작성한 등록일
modify_at	다이어를 마지막으로 수정한 등록일

- WriteDiaryDto.java
  :다이어리 작성 시 사용되는 DTO

writer	다이어리를 작성하는 계정아이디
date	다이어리에 작성된 날짜
title	다이어리에 작성된 제목
content	다이어리에 작성된 내용
weather	다이어리에 작성된 날씨

(4) entity폴더

- Diary.java
  : 데이터베이스의 Diary 테이블 필드

```
public class Diary {
    private long id;
    private String writer;
    private Date date;
    private String title;
    private String content;
    private String weather;
    private Date create_at;
    private Date modify_at;
```

(5) service폴더

- DeleteDiaryService.java

public void delete (Long id)	다이어리의 Dao클래스를 이용하여 삭제서비스 구현

- GetDiaryService.java

public GetDiaryDto getDiaryDetail (Long id){	다이어리의 Dao클래스를 이용하여 다이어리를 반환
public List<GetDiaryDto> getAllByWriter (String writer)	다이어리의 Dao클래스를 이용하여 다이어리 리스트를 반환
public List<GetDiaryDto> getAllByWriterAndKeyword (String writer, String keyword){	다이어리의 Dao클래스를 이용하여 다이어리 리스트를 반환

- UpdateDiaryService.java

public void update (Long id, UpdateDiaryDto updateDiaryDto)	다이어리의 Dao클래스를 이용하여 다이어리 수정

- WriteDiaryService.java

public Boolean IsAlreadyDiary (WriteDiaryDto writeDiaryDto){	다이어리의 Dao클래스를 이용하여 다이어리를 조회하고, 다이어리 반환값이 있으면 true반환
public void insert (WriteDiaryDto writeDiaryDto)	다이어리의 Dao클래스를 이용하여 다이어리 삽입

6.1.3 domain/member

(1) controller 폴더

- LoginController.java

매핑관계

@GetMapping("/")	jsp로 이동한다.
@PostMapping("/login")	LoginDto를 받아서 LoginService로 보낸다. 만약 LoginDto값이 null이거나 LoginService의 반환값이 null이라면 jsp에 LOGIN_OK속성에 null을 보낸다. 만약 loginDto의 이메일, 패스워드값과 LoginService의 반환값의 이메일, 패스워드값이 일치하면 jsp에 LOGIN_OK속성에 success를 보낸다. 만약 비밀번호가 일치하지 않으면 wrongPassword값을 보낸다.

- MemberController.java

매핑관계

@GetMapping("/memberInfo")	session에 저장된 email값을 GetMemberService에 보내서 해당 멤버정보를 가져온다. 가져온 멤버정보를 jsp에 속성값으로 보낸다.
@GetMapping("/mypage")	session에 저장된 email값을 GetMemberService에 보내서 해당 멤버정보를 가져온다. 가져온 멤버정보를 jsp에 속성값으로 보낸다. 수정페이지로 이동한다.
@PostMapping("/mypage")	session에 저장된 email값과 UpdateMemberDto를 UpdateMemeberService에 보낸다. 반환값이 false이면 jsp에 UPDATE_OK속성에 WrongPassword를 보낸다. 만약 true이면, 서비스단에서 데이터베이스 수정을 하고, jsp에 UPDATE_OK 속성에 success를 보낸다. 그리고 멤버의 이름을 UPDATE_USERNAME값으로 보낸다.
@GetMapping("/leave")	jsp로 이동한다.
@PostMapping("/leaveCheck")	session에 저장된 email값과 패스워드를 DeleteMemberService보낸다. 반환값이 false이면 jsp에 LEAVE_OK 속성에 wrongPassword를 보낸다. 만약 true이면, success를 보내고, 서비스단에서 데이터베이스에서 멤버 정보를 삭제한다.

- SignUpController.java

매핑관계

@GetMapping("/signUp")	jsp로 이동한다.
@PostMapping("/signUpCheck")	SignUpDto를 받아서 SignUpService로 보낸다. 만약 SignUpDto값이 null이라면 jsp에 SIGNUP_OK속성에 null을 보낸다. 만약 반환값이 false라면 jsp에 SIGNUP_OK속성에 IsAlreadyMember를 보낸다. 만약 반환값이 true라면 jsp에 success를 보낸다.

(2) dao 폴더

- MemberDao.java
  :데이터베이스에 직접접근하는 기능을 수행하기 위한 Dao
  :JdbcTemplate 의존관계를 주입

public List<Member> selectAll() {
"select * from member" 쿼리문으로 모든 멤버 리스트를 반환한다.
public Member selectByEmail(String email) {
"select * from member where email=?" 쿼리문으로 email으로 멤버정보를 반환한다.
public void insert (SignUpDto signUpDto){
"insert member (email,password,name,phone,profile) values (?,?,?,?,?)" 쿼리문으로 멤버를 데이터베이스에 삽입한다.
public void update (UpdateMemberDto updateMemberDto){
"update member set password=?, name=?, phone=?, profile=? where email=?" 쿼리문으로 email으로 멤버를 찾아 데이터베이스의 해당 멤버 정보를 수정한다.
public void delete (String email){
"delete from member where email=?"로 email로 멤버를 찾아 데이터베이스의 해당 멤버 정보를 삭제한다.

(3) dto폴더

- GetMemberDto.java
  :멤버 조회 시 사용되는 DTO

email	조회할 멤버 이메일
password	조회할 멤버 패스워드
name	조회할 멤버 이름
phone	조회할 멤버 핸드폰번호
profile	조회할 멤버 프로필

- LoginDto.java
  :로그인 시 사용되는 DTO

email	로그인 이메일
password	로그인 패스워드

- SignUpDto.java
  :회원가입 시 사용되는 DTO

email	회원가입 멤버 이메일
password	회원가입 멤버 패스워드
name	회원가입 멤버 이름
phone	회원가입 멤버 핸드폰번호
profile	회원가입 멤버 프로필

- UpdateMemberDto.java
  :멤버 수정 시 사용되는 DTO

email	수정할 멤버 이메일
password	수정할 멤버 패스워드
name	수정할 멤버 이름
phone	수정할 멤버 핸드폰번호
profile	수정할 멤버 프로필

(4) entity폴더

- Member.java

  : 데이터베이스의 Member 테이블 필드

```java
@Getter
@Setter
public class Member {
    private Long id;
    private String email;
    private String password;
    private String name;
    private String phone;
    private String profile;
```

(5) service폴더

- DeleteMemberService.java

public Boolean CorrectPassword (String email, String password){	멤버의 Dao클래스를 이용하여 멤버를 조회하고, 비밀번호가 일치하면 true를 반환
public void delete (String email)	멤버의 Dao클래스를 이용하여 멤버 삭제

- GetMemberService.java

public GetMemberDto info(String email) {	멤버의 Dao클래스를 이용하여 멤버 정보 반환

-

- LoginService.java

public Member login (LoginDto loginDto){	멤버의 Dao클래스를 이용하여 멤버를 조회하고, 멤버 값 반환

- SignUpService.java

public Boolean IsAlreadyMember (SignUpDto signUpDto){	멤버의 Dao클래스를 이용하여 멤버를 조회하고, 멤버 반환값이 null이 아니면 true반환
public void insert (SignUpDto signUpDto)	멤버의 Dao클래스를 이용하여 멤버 삽입

- UpdateMemberService.java

public Boolean CorrectPassword (String email, UpdateMemberDto updateMemberDto){	멤버의 Dao클래스를 이용하여 멤버를 조회하고, 멤버의 비밀번호와 UpdateMemberDto의 비밀번호와 일치하면 true 반환
public void update (UpdateMemberDto updateMemberDto)	멤버의 Dao클래스를 이용하여 멤버 정보 수정

6.2. main/resources폴더
application.properties
:jsp 경로 설정

6.3 main/webapp폴더
6.3.1 bootstrap폴더
:부트스트랩에서 지원하는 css파일과 js파일

6.3.2 WEB-INF/jsp 폴더

body.jsp	메인홈의 바디 부분 일기 키워드 검색 폼 일기 쓰기 버튼 클릭 시, 일기 쓰기 페이지로 이동 Calendar.jsp
calendar.jsp	달력 년,월 이동 네비게이터를 통해 년,월값을 /mainhome으로 반환한다. 반환된 값은 하단의 달력에 다시 로딩된다.
diary.jsp	일기 작성 페이지 작성 완료시 /diary로 반환된다. 메인홈으로 이동 버튼 클릭 시 메인홈 페이지로 이동
diaryDeleteCheck.jsp	일기삭제시, 알림창을 띄우는 페이지
diaryDetail.jsp	다이어리 상세정보를 볼 수 있는 페이지 다이어리 수정페이지나 메인홈페이지로 이동할 수 있음
diaryList.jsp	키워드 검색 폼과 검색결과 리스트를 출력함
diaryUpdate.jsp	다이어리 수정 전, 이전 정보를 조회할 수 있고, 수정할 수 있음
diaryUpdateCheck.jsp	일기수정시, 알림창을 띄우는 페이지
footer.jsp	Our Diary 서비스 소개와 만든사람이름이 표기됨
header.jsp	메인홈 헤더 프로젝트 이름과 환영인사말로 구성됨 메인홈 이동, 메인페이지 이동, 로그아웃 버튼으로 구성됨
leave.jsp	탈퇴 페이지 비밀번호 입력란으로 구성됨
leaveCheck.jsp	회원탈퇴시, 알림창을 띄우는 페이지
login.jsp	프로젝트 시작화면 페이지 Our Diary 서비스 소개와 로그인창으로 구성됨 회원가입페이지로 이동할 수 있음
loginCheck.jsp	로그인시, 알림창을 띄우는 페이지 비밀번호 오류 알림, 회원정보 없음 알림으로 구성됨
logout.jsp	로그아웃 시 알림창을 띄우는 페이지 기존 세션값을 제거함
mainhome.jsp	메인홈페이지 헤더, 바디, 푸터로 구성되어 있음

mypage.jsp	마이페이지 헤더, 마이페이지 인포, 푸터로 구성되어 있음
mypageInfo.jsp	마이페이지 조회하는 페이지 회원 정보를 조회할 수 있음 회원정보수정페이지, 탈퇴페이지로 이동할 수 있음
mypageUpdate.jsp	회원정보를 수정하는 페이지 이전 정보를 조회할 수 있고, 수정할 수 있음
mypageUpdateCheck.jsp	마이페이지 수정시, 알림창을 띄우는 페이지
signUp.jsp	회원가입 폼
signUpCheck.jsp	회원가입시 알림창을 띄우는 페이지
writeDiaryCheck.jsp	일기작성시 알림창을 띄우는 페이지