———————— Update , Delete 처리

(상세보기에서 처리 할 예정)

1. 수정의 처리 과정

——— 데이터를 먼저 출력하고

수정할 수 있는 화면으로 이동

이동한 화면에서 수정하고

작업을 수행하고 상세보기로 이동

2. 삭제의 처리 과정

——— 삭제의 처리 과정은 삭제한 후 목록 보기로 이동

———————— ajax 로 처리하는 경우

수정 ——〉 put

삭제 ——〉 Delete

———————— 1. 수정과 삭제를 위한 링크를 read.html 파일에 추가

—— gno(글번호 : 수정을 위해서 수정할 데이터를 가져와야 하기 때문)

—— page(현재 페이지 번호 : 작업이 끝나면 현재 페이지로 돌아가야 해서)

—— 수정 전

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
  <th:block th:fragment="content">
    <h1 class="mt-4">방명록 상세보기</h1>

    <div class="form-group">
      <label>Gno</label>
      <input type="text" class="form-control" name="gno" th:value="${dto.gno}" readonly >
    </div>

    <div class="form-group">
      <label >Title</label>>
      <input type="text" class="form-control" name="title" th:value="${dto.title}" readonly >
    </div>
    <div class="form-group">
      <label >Content</label>
      <textarea class="form-control" rows="5" name="content" readonly>[[${dto.content}]]</textarea>
    </div>
    <div class="form-group">
      <label >Writer</label>
      <input type="text" class="form-control" name="writer" th:value="${dto.writer}" readonly>
    </div>
    <div class="form-group">
      <label >RegDate</label>
      <input type="text" class="form-control" name="regDate" th:value="${#temporals.format(dto.regDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>
    <div class="form-group">
      <label >ModDate</label>
      <input type="text" class="form-control" name="modDate" th:value="${#temporals.format(dto.modDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>

    <a th:href="@{/guestbook/list(page=${requestDto.page})}"><button type="button" class="btn btn-info">목록</button></a>

  </th:block>
</th:block>
```

—— 수정 후

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
  <th:block th:fragment="content">
    <h1 class="mt-4">방명록 상세보기</h1>

    <div class="form-group">
      <label>Gno</label>
      <input type="text" class="form-control" name="gno" th:value="${dto.gno}" readonly >
    </div>

    <div class="form-group">
      <label >Title</label>>
      <input type="text" class="form-control" name="title" th:value="${dto.title}" readonly >
    </div>
    <div class="form-group">
      <label >Content</label>
      <textarea class="form-control" rows="5" name="content" readonly>[[${dto.content}]]</textarea>
    </div>
    <div class="form-group">
      <label >Writer</label>
      <input type="text" class="form-control" name="writer" th:value="${dto.writer}" readonly>
    </div>
    <div class="form-group">
      <label >RegDate</label>
      <input type="text" class="form-control" name="regDate" th:value="${#temporals.format(dto.regDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>
    <div class="form-group">
      <label >ModDate</label>
      <input type="text" class="form-control" name="modDate" th:value="${#temporals.format(dto.modDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>
```

```
    <a th:href="@{/guestbook/modify(gno=${dto.gno},page=${requestDto.page})}"><button type="button" class="btn btn-info">목록</button></a>
```

```
    <a th:href="@{/guestbook/list(page=${requestDto.page})}"><button type="button" class="btn btn-info">목록</button></a>

  </th:block>
</th:block>
```

———————— GuestBookController 클래스에
수정 및 삭제를 위한 보기 요청을
처리할 코드를 생성
수정화면으로 이동하는 처리는 상세보기와 동일
(하나의 데이터를 가져와서 수정할 수 있도록 화면을 변경해서 출력해주면 된다)
——〉 적절하게 데이터를 hidden과 readOnly를 적용 시키는 것이 방법이다.

——〉 수정 전

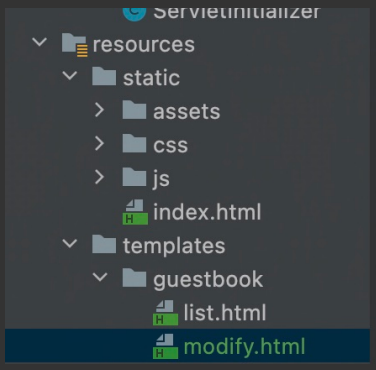
```
package com.singsiuk.guestbook.controller;

import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;
import com.singsiuk.guestbook.service.GuestBookService;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

// 로그 추력을 위한 Annotation
@Log4j2
//PageController 를 만들기 위한 Annotation
@Controller
@RequiredArgsConstructor
public class GuestBookController {
    // Service 주입
    private final GuestBookService service;

    @GetMapping("/")
    public String main(){
        // 배포시 지워진다
        log.info("시작요청");
        //Template 에 있는 Guestbook 디렉토리의 list.html 을 출력


    return "redirect:/guestbook/list";
    }

    @GetMapping("/guestbook/list")
    public void list(PageRequestDto pageRequestDto, Model model){
        log.info("Show List");
        PageResponseDto result = service.getList(pageRequestDto);
        model.addAttribute("result",result);
    }

    @GetMapping("/guestbook/register")
    public void register(){
        log.info("삽입 요청 page 로 이동 ");
    }

    @PostMapping("/guestbook/register")
    public String register(GuestBookDto dto, RedirectAttributes redirectAttributes){
        log.info("Insert Request");
        // 삽입 처리
        Long gno = service.register(dto);
        // 리다이렉트 할 때 한번만 사용하는 데이터 생성
        redirectAttributes.addFlashAttribute(
                "msg", gno+"Insert");
        // 작업 후 목록보기로 redirect
        return "redierct:/guestbook/list";
    }

    @GetMapping("/guestbook/read")
    // 파라미터 중 gno는 gno에 대입
    // 나머지는 requestDto 에 대입
    // 다음과 같은 페이지를 전송
    public void read(long gno,
                     @ModelAttribute("requestDto")
                             PageRequestDto requestDto,
                                     Model model){
        GuestBookDto dto = service.read(gno);
        model.addAttribute("dto",dto);
    }
```

——— 수정 후

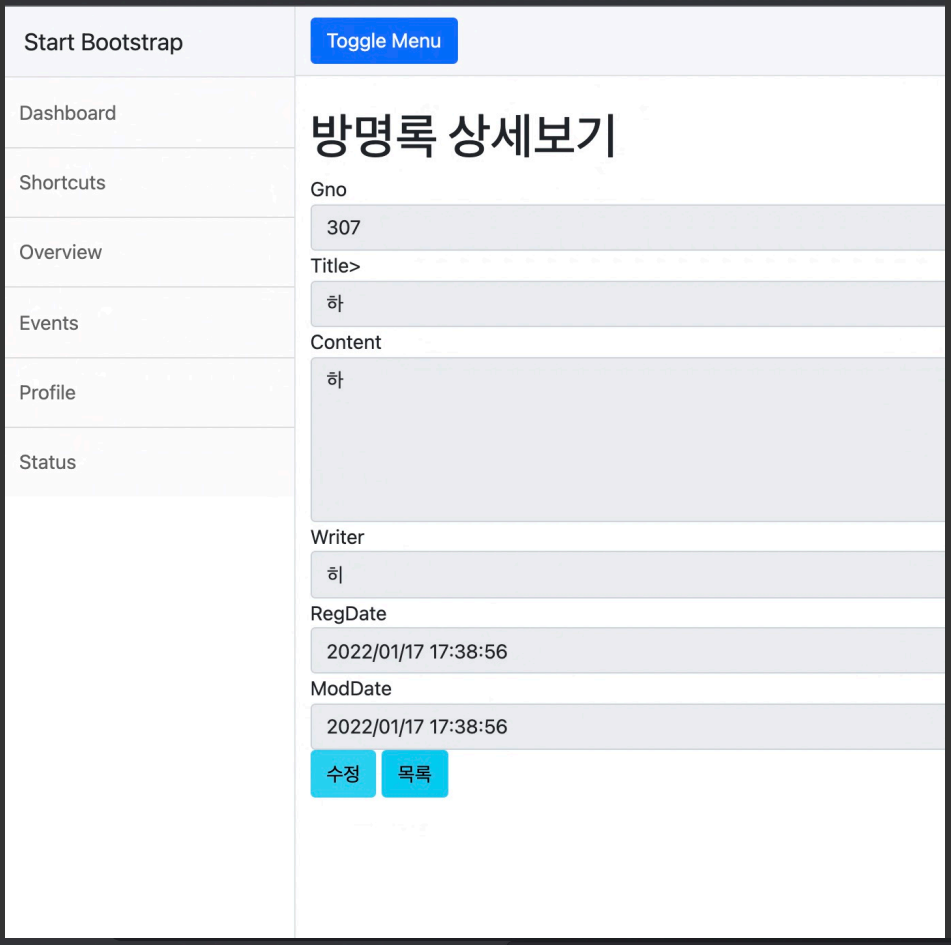
```
package com.singsiuk.guestbook.controller;

import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;
import com.singsiuk.guestbook.service.GuestBookService;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

// 로그 추력을 위한 Annotation
@Log4j2
//PageController 를 만들기 위한 Annotation
@Controller
@RequiredArgsConstructor
public class GuestBookController {
    // Service 주입
    private final GuestBookService service;

    @GetMapping("/")
    public String main(){
        // 배포시 지워진다
        log.info("시작요청");
        //Template 에 있는 Guestbook 디렉토리의 list.html 을 출력

    return "redirect:/guestbook/list";
    }

    @GetMapping("/guestbook/list")
    public void list(PageRequestDto pageRequestDto, Model model){
        log.info("Show List");
        PageResponseDto result = service.getList(pageRequestDto);
        model.addAttribute("result",result);
    }

    @GetMapping("/guestbook/register")
    public void register(){
        log.info("삽입 요청 page 로 이동 ");
    }

    @PostMapping("/guestbook/register")
    public String register(GuestBookDto dto, RedirectAttributes redirectAttributes){
        log.info("Insert Request");
        // 삽입 처리
        Long gno = service.register(dto);
        // 리다이렉트 할 때 한번만 사용하는 데이터 생성
        redirectAttributes.addFlashAttribute(
                "msg", gno+"Insert");
        // 작업 후 목록보기로 redirect
        return "redierct:/guestbook/list";
    }
    @GetMapping({"/guestbook/read", "guestbook/modify"})
    // 파라미터 중 gno는 gno에 대입
    // 나머지는 requestDto 에 대입
    // 다음과 같은 페이지를 전송
    public void read(long gno,
                     @ModelAttribute("requestDto")
                             PageRequestDto requestDto,
                                             Model model){
        GuestBookDto dto = service.read(gno);
        model.addAttribute("dto",dto);
    }
}
```

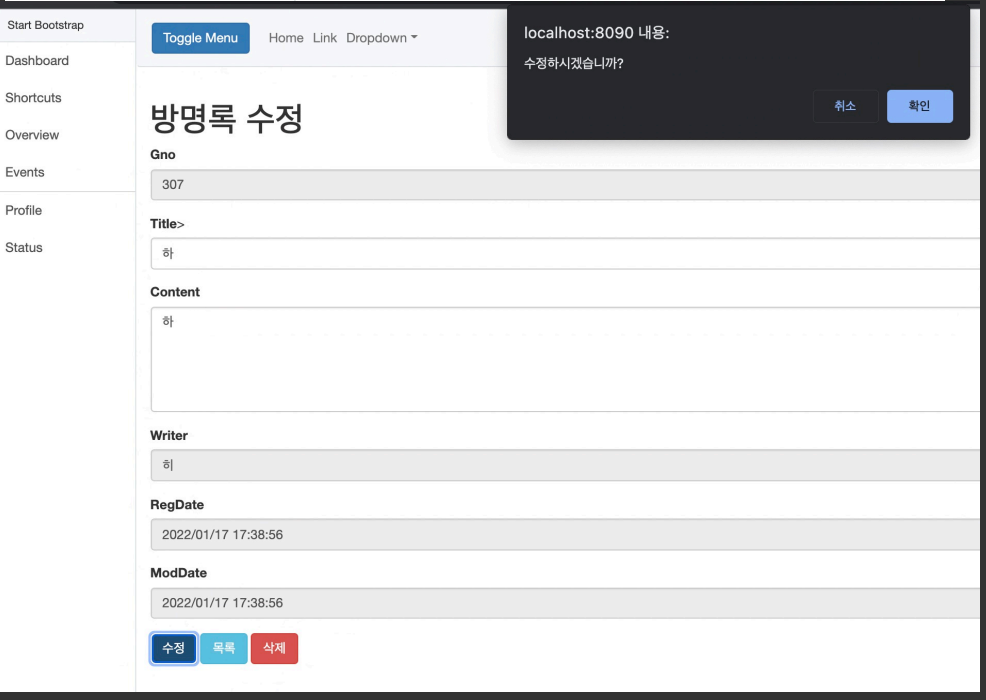
——————— GuestBookService Interface 에 수정과 삭제를 위한 method 생성 (수정은 삽입과 모양이 같다)

———————— 수정 화면을 guestbook 디렉토리에 modify.html 을 생성




```
<!DOCTYPE html>
Thymeleaf 의 th를 사용하기 위한 설정
최신 버전에서는 생략해도 상관 없다
<html lang="en" xmlns:th="http://www.thymeleaf.org">

이 파일의 내용을 출력할 레이아웃 파일의 경로를 설정
출력할 때 layout/basic 에다가 출력하겠다

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">

기본 틀을 출력한 상태에서 변경되는 내용
  <th:block th:fragment="content">
    <h1 class="mt-4">방명록 수정</h1>

수정을 하기 위한 폼
    <form action="/guestbook/modify" method="post">
     페이지 번호
     페이지 번호는 필요한데 출력할 필요는 없어서 숨김
      <input type="hidden" name="page" th:value="${requestDto.page}">

     글번호는 출력은 하지만 수정할 수 없도록 읽기 전용
      <div class="form-group">
        <label >Gno</label>
        <input type="text" class="form-control" name="gno" th:value="${dto.gno}" readonly="readonly" >
      </div>
     title 은 한줄 입력도구에 출력
      <div class="form-group">
        <label >Title</label>>
        <input type="text" class="form-control" name="title" th:value="${dto.title}" >
      </div>
     여러줄 입력도구에 출력 / textarea
      <div class="form-group">
        <label >Content</label>
        <textarea class="form-control" rows="5" name="content">[[${dto.content}]]</textarea>
      </div>
     작성자도 readonly
      <div class="form-group">
        <label >Writer</label>
        <input type="text" class="form-control" name="writer" th:value="${dto.writer}" readonly="readonly">
      </div>
<!--       보여줘도 되고 안보여줘도 된다. 사용자가 입력하는게 아니기 때문-->
      <div class="form-group">
        <label >RegDate</label>
        <input type="text" class="form-control" th:value="${#temporals.format(dto.regDate, 'yyyy/MM/dd HH:mm:ss')}"
readonly="readonly">
      </div>
<!--       보여줘도 되고 안보여줘도 된다. 사용자가 입력하는게 아니기 때문-->
      <div class="form-group">
        <label >ModDate</label>
        <input type="text" class="form-control" th:value="${#temporals.format(dto.modDate, 'yyyy/MM/dd HH:mm:ss')}"
readonly="readonly">
      </div>

    </form>

    <button type="button" class="btn btn-primary modifyBtn">수정</button>
    <button type="button" class="btn btn-info listBtn">목록</button>
    <button type="button" class="btn btn-danger removeBtn">삭제</button>
<!--    boot strap 을 사용할 때 들어가는 문장-->
<!--    1. bootstrap 은 event 처리를 jQuery로 한다-->
jQuery가 먼저 나와야한다.
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

    <script th:inline="javascript">
       // $는 Jquery에서 선택자를 의미한다. ---> $is not found 발생 ---> 링크가 없다
      var actionForm = $("form"); //form 태그 객체

       삭제 버튼을 누를 때
      $(".removeBtn").click(function(){
        actionForm
                .attr("action", "/guestbook/remove")
                .attr("method","post");
        actionForm.submit();
      });

       수정 버튼을 누를 때
      $(".modifyBtn").click(function() {
        if(!confirm("수정하시겠습니까?")){
          return ;
        }
        actionForm
                .attr("action", "/guestbook/modify")
                .attr("method","post")
                .submit();
      });
       목록 버튼을 눌렀을 때
      $(".listBtn").click(function() {
        //var pageInfo = $("input[name='page']");
        var page = $("input[name='page']");
        actionForm.empty(); //form 태그의 모든 내용을 지우고
        actionForm.append(page);
        actionForm
                .attr("action", "/guestbook/list")
                .attr("method","get");
        actionForm.submit();
      })
    </script>
  </th:block>
</th:block>
```


Start Bootstrap
Toggle Menu
Dashboard
Shortcuts
Overview
Events
Profile
Status
방명록 상세보기
Gno
307
Title>
하
Content
하
Writer
히
RegDate
2022/01/17 17:38:56
ModDate
2022/01/17 17:38:56
수정
목록


Start Bootstrap
Toggle Menu
Home
Link
Dropdown
Dashboard
Shortcuts
Overview
Events
Profile
Status
localhost:8090 내용:
수정하시겠습니까?
취소
확인
방명록 수정
Gno
307
Title>
하
Content
하
Writer
히
RegDate
2022/01/17 17:38:56
ModDate
2022/01/17 17:38:56
수정
목록
삭제

———————— GuestBookService 에 수정 삭제 관련 method 를 선언

```
package com.singsiuk.guestbook.service;

import com.singsiuk.guestbook.Entity.GuestBook;
import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;

public interface GuestBookService {
    //DTO 클래스의 인스턴스를 Entity 인스턴스로 변환해주는 method
    default GuestBook dtoToEntity(GuestBookDto dto){
    GuestBook entity = GuestBook.builder()
            .gno(dto.getGno())
            .title(dto.getTitle())
            .content(dto.getContent())
            .writer(dto.getWriter())
            .build();
    return entity;
    }
    // Entity 클래스의 Instance를 DTO 클래스의 Instance로 변환해 주는 method
    default GuestBookDto entityDto(GuestBook guestBook){
        GuestBookDto dto= GuestBookDto.builder()
                .gno(guestBook.getGno())
                .title(guestBook.getTitle())
                .content(guestBook.getContent())
                .writer(guestBook.getWriter())
                .regDate(guestBook.getRegDate())
                .modDate(guestBook.getModDate())
                .build();
        return dto;
    }

    // 데이터 삽입을 위한 method
    public Long register(GuestBookDto dto);


    // 데이터 목록보기를 위한 method
    public PageResponseDto<GuestBookDto,GuestBook>
            getList(PageRequestDto requestDto);


    // 상세보기를 위한 method
    public GuestBookDto read(Long gno);

    // 데이터 수정을 위한 method
    public void modify(GuestBookDto dto);

    // 데이터 삭제를 위한 method
    public void remove(Long gno);
}
```

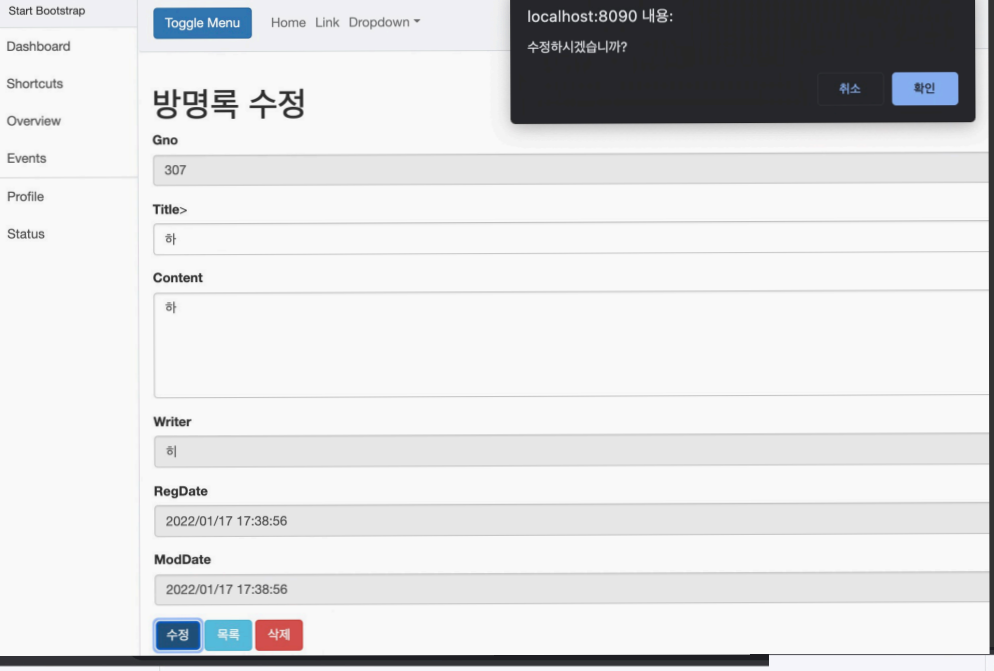
———————— GuestBookServiceImpl에 구현

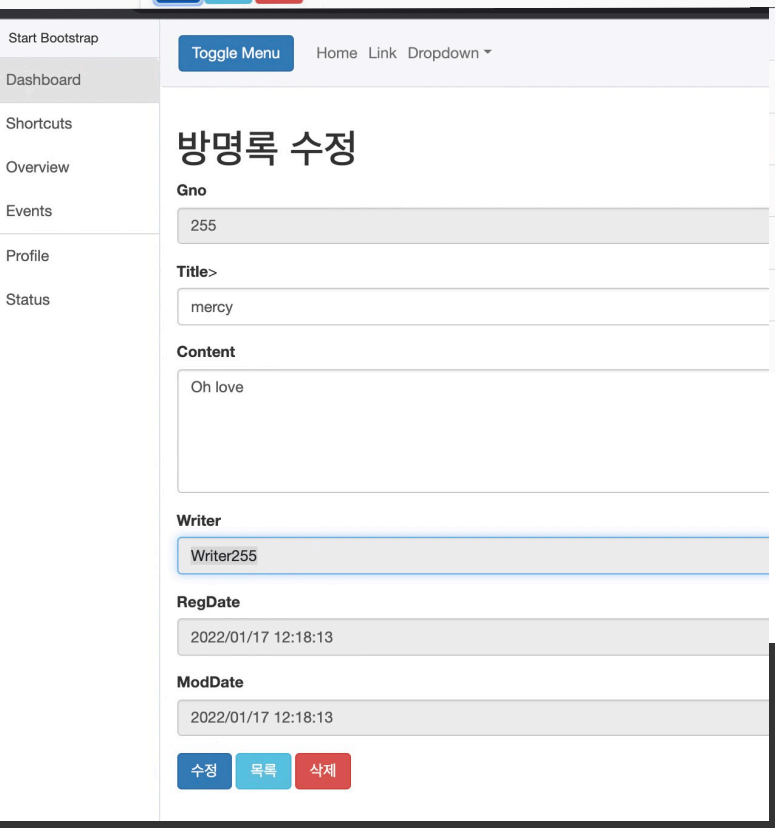
```
package com.singsiuk.guestbook.service;


import com.singsiuk.guestbook.Entity.GuestBook;
import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;
import com.singsiuk.guestbook.repository.GuestBookRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import java.util.Optional;
import java.util.function.Function;

@Service
@Log4j2
@RequiredArgsConstructor
public class GuestServiceImpl implements GuestBookService{
    // 자동 주입 받기 위해서 final 로 선언해야 한다 .
    private final GuestBookRepository repository;

    @Override
    public Long register(GuestBookDto dto) {
        log.info(dto);
                // Dto를 Entity 로 변환
        GuestBook entity = dtoToEntity(dto);
        log.info(entity);
        // 데이터 삽입
        repository.save(entity);
        // 삽입한 데이터의 gno 리턴
        return entity.getGno();
        }

    @Override
    public PageResponseDto<GuestBookDto, GuestBook> getList(PageRequestDto requestDto) {
        // Pageable 객체 생성
        Pageable pageable = requestDto.getPageable(
                Sort.by("gno").descending());
        // 결과를 가져오기
        Page<GuestBook> result = repository.findAll(pageable);

        //Function 생성
        Function<GuestBook,GuestBookDto> fn
                // entity 를 dto로 바꾸는 요청
                =(entity ->entityDto(entity));
        return new PageResponseDto<>(result, fn);
    }

    @Override
    public GuestBookDto read(Long gno) {
        // return type 이 Optional
        Optional<GuestBook> guestbook= repository.findById(gno);
         return guestbook.isPresent()?entityDto(guestbook.get()):null;
    }

    @Override
    public void modify(GuestBookDto dto) {
        // 수정할 데이터 찾아오기
        // 굳이 데이터가 없는 것을 실행할 필요는 없기 때문에
        Optional<GuestBook> result = repository.findById(dto.getGno());
        if(result.isPresent()){
            GuestBook entity =result.get();
            entity.changeTitle(dto.getTitle());
            entity.changeContent(dto.getContent());
            repository.save(entity);
        }
    }

    @Override
    public void remove(Long gno) {
        // 삭제할 데이터를 찾아오기
        Optional<GuestBook> result = repository.findById(gno);
        if(result.isPresent()){
            repository.deleteById(gno);
        }

    }
}
```

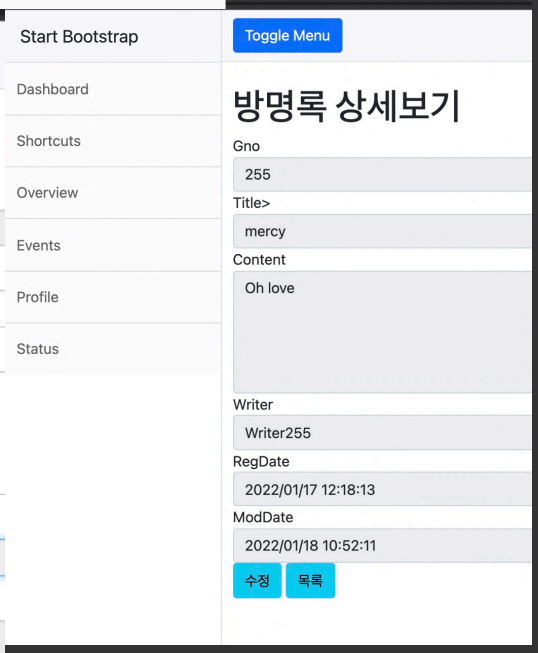
GuestBookController 에서 처리

```
package com.singsiuk.guestbook.controller;

import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;
import com.singsiuk.guestbook.service.GuestBookService;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

// 로그 추적을 위한 Annotation
@Log4j2
//PageController 를 만들기 위한 Annotation
@Controller
@RequiredArgsConstructor
public class GuestBookController {
    // Service 주입
    private final GuestBookService service;

    @GetMapping("/")
    public String main(){
        // 배포시 지워진다
        log.info("서작성중");
        //Template 에 있는 Guestbook 디렉토리의 list.html 을 출력

    return "redirect:/guestbook/list";
    }

    @GetMapping("/guestbook/list")
    public void list(PageRequestDto pageRequestDto, Model model){
        log.info("Show list");
        PageResponseDto result = service.getList(pageRequestDto);
        model.addAttribute("result",result);
    }

    @GetMapping("/guestbook/register")
    public void register(){
        log.info("등록 요청 page 로 이동 ");
    }

    @PostMapping("/guestbook/register")
    public String register(GuestBookDto dto, RedirectAttributes redirectAttributes){
        log.info("Insert Request ");
        // 성공 처리
        Long gno = service.register(dto);
        // 리다이렉트 할 때 한번만 사용하는 데이터 생성
        redirectAttributes.addFlashAttribute(
                "msg", gno+"insert");
        // 작업 후 목록보기로 redirect
        return "redirect:/guestbook/list";
    }



    @GetMapping({"/guestbook/read", "guestbook/modify"})
    // 파라미터 중 gno는 gno에 대입
    // 나머지는 requestDto 에 대입
    // 다음과 같은 페이지를 전송
    public void read(long gno,
                     @ModelAttribute("requestDto")
                             PageRequestDto requestDto,
                                     Model model){

        GuestBookDto dto = service.read(gno);
        model.addAttribute("dto",dto);
    }
```

```
    @PostMapping("/guestbook/modify")
    public String modify(GuestBookDto dto, @ModelAttribute("requestDto")
            PageRequestDto requestDto,RedirectAttributes redirectAttributes){
        //수정 method 수행
        service.modify(dto);
        //전달할 데이터 생성
        redirectAttributes.addAttribute("page", requestDto.getPage());
        redirectAttributes.addAttribute("gno",dto.getGno());

        //상세보기로 redirect
        return "redirect:/guestbook/read";
    }

    @PostMapping("/guestbook/remove")
    public String modify(Long gno, RedirectAttributes redirectAttributes){
        //삭제 method 수행
        service.remove(gno);
        //전달할 데이터 생성
        redirectAttributes.addFlashAttribute("msg", gno+"삭제");

        //상세보기로 redirect
        return "redirect:/guestbook/list";
    }
}
```


Start Bootstrap
Dashboard
Shortcuts
Overview
Events
Profile
Status
Toggle Menu
Home Link Dropdown
localhost:8090 내용:
수정하시겠습니까?
취소
확인
방명록 수정
Gno
307
Title>
하
Content
하
Writer
히
RegDate
2022/01/17 17:38:56
ModDate
2022/01/17 17:38:56
수정
목록
삭제


Start Bootstrap
Dashboard
Shortcuts
Overview
Events
Profile
Status
Toggle Menu
Home Link Dropdown
방명록 수정
Gno
255
Title>
mercy
Content
Oh love
Writer
Writer255
RegDate
2022/01/17 12:18:13
ModDate
2022/01/17 12:18:13
수정
목록
삭제


Start Bootstrap
Dashboard
Shortcuts
Overview
Events
Profile
Status
Toggle Menu
방명록 상세보기
Gno
255
Title>
mercy
Content
Oh love
Writer
Writer255
RegDate
2022/01/17 12:18:13
ModDate
2022/01/18 10:52:11
수정
목록

방명록 수정

Gno

308

Title>

dfvadsf

Content

sdafasdf

Writer

asfa

RegDate

2022/01/17 18:10:38

ModDate

2022/01/17 18:10:38

수정 목록 삭제

#	Title
307	하
306	gk
305	ak
304	할머니
303	할머니
302	제목
301	제목
300	Title300
299	Title299
298	Title298

———————— 검색 기능 추가

기본키와 일치하는 data를 검색하는 것은.
JPA 에서는 별도의 구현이 필요 없다.

기본키와 일치하는 데이터를 제외한 조회는
JPQL 이나 @Query등을 이용해야 한다 ..

1. 조회 요청을 할 때 사용하는 PageRequestDTO 클래스의
검색항목과 실제 키워드를 의미하는
2개의 속성을 추가

———————— PageRequestDto 수정

```
package com.singsiuk.guestbook.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;

@Builder
@Data
@AllArgsConstructor
public class PageRequestDto {
    // 현재 페이지 번호
    private int page;

    // 페이지당 출력 개수
    private int size;

        // 검색 항목
        private String type;

        // 검색할 키워드
        private String keyword;

    // Parameter가 없는 생성자 ( default constructor) 생성
    public PageRequestDto(){
        page=1;
        size=10;
    }

    // Page 와 Size 를 이용한 Pageable 객체를 생성해서 return하는 method

    public Pageable getPageable(Sort sort){
        // 페이지 번호는 -1 을 해서 생성
        // 사람은 1 부터
        // 컴퓨터는 0 부터
        return PageRequest.of(page-1,size,sort);
    }
}
```

——— 〉 title, content, writer 그리고 세가지의 조합으로 검색이 가능하도록 작성

——— 〉 select로 type을 만들고
각각의 value를 t,c,w로 하고 조합은 tcw를 합친형태로 생성 할 예정

GuestBookServiceImpl 클래스에 검색 조건을 만들어주는 method를 작성

```
    private BooleanBuilder getSearch(PageRequestDto requestDto){
        String type= requestDto.getType();
        String keyword = requestDto.getKeyword();
        BooleanBuilder booleanBuilder= new BooleanBuilder();
        QGuestBook qGuestBook = QGuestBook.guestBook;


        if(type==null||type.trim().length()==0){
            // 좌우 공백 제거
            return booleanBuilder;
        }
        // 나머지 조건들 나열
        BooleanBuilder conditionBuilder = new BooleanBuilder();
        // t c w 는 검색 화면에서 select의 option들의 value가 되어야한다.
        if(type.contains("t")){
            conditionBuilder.or(qGuestBook.title.contains(keyword));
        }
        if(type.contains("c")) {
            conditionBuilder.or(qGuestBook.content.contains(keyword));
        }
        if(type.contains("w")) {
            conditionBuilder.or(qGuestBook.writer.contains(keyword));
        }
        //모든 조건 통합
        booleanBuilder.and(conditionBuilder);
        return booleanBuilder;
    }
}
```

GuestBookServiceImpl 에서

PageResponseDto method 수정

수정 전

```java
package com.singsiuk.guestbook.service;


import com.querydsl.core.BooleanBuilder;
import com.singsiuk.guestbook.Entity.GuestBook;
import com.singsiuk.guestbook.Entity.QGuestBook;
import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;
import com.singsiuk.guestbook.repository.GuestBookRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import java.util.Optional;
import java.util.function.Function;

@Service
@Log4j2
@RequiredArgsConstructor
public class GuestServiceImpl implements GuestBookService{
    // 자동 주입 받기 위해서 final 로 선언해야 한다 .
    private final GuestBookRepository repository;

    @Override
    public Long register(GuestBookDto dto) {
        log.info(dto);
                // Dto를 Entity 로 변환
        GuestBook entity = dtoToEntity(dto);
        log.info(entity);
        // 데이터 삽입
        repository.save(entity);
        // 삽입한 데이터의 gno 리턴
        return entity.getGno();
        }

    @Override
    public PageResponseDto<GuestBookDto, GuestBook> getList(PageRequestDto requestDto) {
        // Pageable 객체 생성
        Pageable pageable = requestDto.getPageable(
                Sort.by("gno").descending());
        // 결과를 가져오기
        Page<GuestBook> result = repository.findAll(pageable);

        //Function 생성
        Function<GuestBook,GuestBookDto> fn
                // entity 를 dto로 바꾸는 요청
                =(entity ->entityDto(entity));
        return new PageResponseDto<>(result, fn);
    }

    @Override
    public GuestBookDto read(Long gno) {
        // return type 이 Optional
        Optional<GuestBook> guestbook= repository.findById(gno);
         return guestbook.isPresent()?entityDto(guestbook.get()):null;
    }

    @Override
    public void modify(GuestBookDto dto) {
        // 수정할 데이터 찾아오기
        // 굳이 데이터가 없는 것을 실행할 필요는 없기 때문에
        Optional<GuestBook> result = repository.findById(dto.getGno());
        if(result.isPresent()){
            GuestBook entity =result.get();
            entity.changeTitle(dto.getTitle());
            entity.changeContent(dto.getContent());
            repository.save(entity);
        }
    }

    @Override
    public void remove(Long gno) {
        // 삭제할 데이터를 찾아오기
        Optional<GuestBook> result = repository.findById(gno);
        if(result.isPresent()){
            repository.deleteById(gno);
        }
    }
    private BooleanBuilder getSearch(PageRequestDto requestDto){
        String type= requestDto.getType();
        String keyword = requestDto.getKeyword();
        BooleanBuilder booleanBuilder= new BooleanBuilder();
        QGuestBook qGuestBook = QGuestBook.guestBook;


        if(type==null||type.trim().length()==0){
            // 좌우 공백 제거
            return booleanBuilder;
        }
        // 나머지 조건들 나열
        BooleanBuilder conditionBuilder = new BooleanBuilder();
        // t c w 는 검색 화면에서 select의 option들의 value가 되어야한다.
        if(type.contains("t")){
            conditionBuilder.or(qGuestBook.title.contains(keyword));
        }
        if(type.contains("c")) {
            conditionBuilder.or(qGuestBook.content.contains(keyword));
        }
        if(type.contains("w")) {
            conditionBuilder.or(qGuestBook.writer.contains(keyword));
        }
        //모든 조건 통합
        booleanBuilder.and(conditionBuilder);
        return booleanBuilder;
    }
}
```

수정 후

```java
package com.singsiuk.guestbook.service;


import com.querydsl.core.BooleanBuilder;
import com.singsiuk.guestbook.Entity.GuestBook;
import com.singsiuk.guestbook.Entity.QGuestBook;
import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;
import com.singsiuk.guestbook.repository.GuestBookRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import java.util.Optional;
import java.util.function.Function;

@Service
@Log4j2
@RequiredArgsConstructor
public class GuestServiceImpl implements GuestBookService{
    // 자동 주입 받기 위해서 final 로 선언해야 한다 .
    private final GuestBookRepository repository;

    @Override
    public Long register(GuestBookDto dto) {
        log.info(dto);
                // Dto를 Entity 로 변환
        GuestBook entity = dtoToEntity(dto);
        log.info(entity);
        // 데이터 삽입
        repository.save(entity);
        // 삽입한 데이터의 gno 리턴
        return entity.getGno();
        }

    @Override
    public PageResponseDto<GuestBookDto, GuestBook> getList(PageRequestDto requestDto) {
        // Pageable 객체 생성
        Pageable pageable = requestDto.getPageable(
                Sort.by("gno").descending());
        // 결과를 가져오기
        BooleanBuilder booleanBuilder = getSearch(requestDto);
        Page<GuestBook> result = repository.findAll(booleanBuilder,pageable);


        //Function 생성
        Function<GuestBook,GuestBookDto> fn
                // entity 를 dto로 바꾸는 요청
                =(entity ->entityDto(entity));
        return new PageResponseDto<>(result, fn);
    }

    @Override
    public GuestBookDto read(Long gno) {
        // return type 이 Optional
        Optional<GuestBook> guestbook= repository.findById(gno);
         return guestbook.isPresent()?entityDto(guestbook.get()):null;
    }

    @Override
    public void modify(GuestBookDto dto) {
        // 수정할 데이터 찾아오기
        // 굳이 데이터가 없는 것을 실행할 필요는 없기 때문에
        Optional<GuestBook> result = repository.findById(dto.getGno());
        if(result.isPresent()){
            GuestBook entity =result.get();
            entity.changeTitle(dto.getTitle());
            entity.changeContent(dto.getContent());
            repository.save(entity);
        }
    }

    @Override
    public void remove(Long gno) {
        // 삭제할 데이터를 찾아오기
        Optional<GuestBook> result = repository.findById(gno);
        if(result.isPresent()){
            repository.deleteById(gno);
        }
    }
    private BooleanBuilder getSearch(PageRequestDto requestDto){
        String type= requestDto.getType();
        String keyword = requestDto.getKeyword();
        BooleanBuilder booleanBuilder= new BooleanBuilder();
        QGuestBook qGuestBook = QGuestBook.guestBook;


        if(type==null||type.trim().length()==0){
            // 좌우 공백 제거
            return booleanBuilder;
        }
        // 나머지 조건들 나열
        BooleanBuilder conditionBuilder = new BooleanBuilder();
        // t c w 는 검색 화면에서 select의 option들의 value가 되어야한다.
        if(type.contains("t")){
            conditionBuilder.or(qGuestBook.title.contains(keyword));
        }
        if(type.contains("c")) {
            conditionBuilder.or(qGuestBook.content.contains(keyword));
        }
        if(type.contains("w")) {
            conditionBuilder.or(qGuestBook.writer.contains(keyword));
        }
        //모든 조건 통합
        booleanBuilder.and(conditionBuilder);
        return booleanBuilder;
    }
}
```

———— TEST class 에서 TEST
: title 에 연습이 포함된 데이터 조회

```java
@Test
    public void listTest(){
        PageRequestDto pageRequestDto=
                PageRequestDto.builder()
                        .page(1)
                        .size(10)
                        .type("t")
                        .keyword("할머니")
                        .build();
        PageResponseDto<GuestBookDto, GuestBook>
                pageResponseDto=service.getList(pageRequestDto);
        for(GuestBookDto dto:pageResponseDto.getDtoList()){
            System.out.println(dto);
        }
        //이전 과 다음 링크 여부와 전체 페이지 개수 확인
        System.out.println("==================================");
        System.out.println("이전:"+pageResponseDto.isPrev());
        System.out.println("다음:"+pageResponseDto.isNext());
        System.out.println("다음:"+pageResponseDto.getTotalPage());

        // 페이지 번호 목록 추력
        System.out.println("==================================");
        for(Integer i:pageResponseDto.getPageList()){
            System.out.print(i+"\t");
        }
```

```
    from
        guest_book guestbook0_
    where
        guestbook0_.title like ? escape '!'
    order by
        guestbook0_.gno desc limit ?
GuestBookDto(gno=304, title=할머니, content=할아버지, writer=삼촌, regDate=2022-01-17T17:19:41.042365, modDate=2022-01-17T17:19:41.042365)
GuestBookDto(gno=303, title=할머니, content=할아버지, writer=삼촌, regDate=2022-01-17T17:16:43.911413, modDate=2022-01-17T17:16:43.911413)
==================================
이전:false
다음:false
다음:1
==================================
```

——————— list.html 파일에 검색 form 을 추가

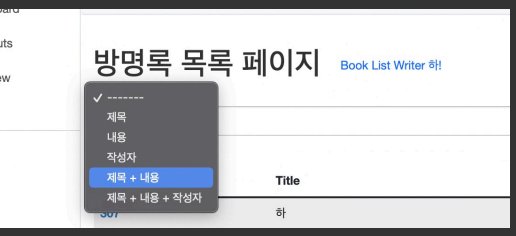
— 요즘에는 검색을 상단에 넣는경우가 많음

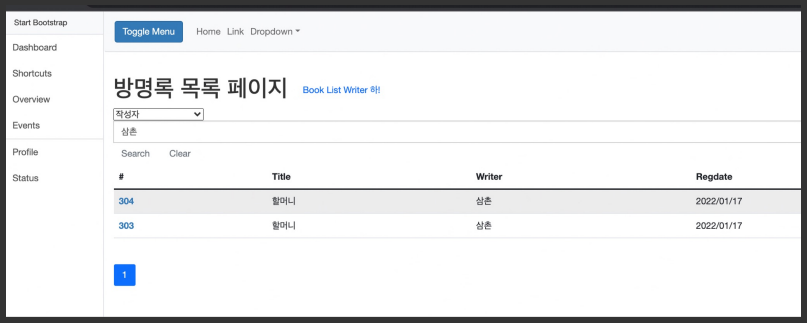
```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
    <th:block th:fragment="content">
        <h1 class="mt-4">방명록 목록 페이지
            <span>
                <a th:href="@{/guestbook/register}">
                    <button type="button" class="btn btn-outline-primary">
                        Book List Writer 하!
                    </button>
                </a>
            </span>
        </h1>

        <form action="/guestbook/list" method="get" id="searchForm">
            <div class="input-group">
                <input type="hidden" name="page" value = "1">
                <div class="input-group-prepend">
                    <select class="custom-select" name="type">
                        <option th:selected="${pageRequestDto.type == null}">-------</option>
                        <option value="t" th:selected="${pageRequestDto.type =='t'}" >제목</option>
                        <option value="c" th:selected="${pageRequestDto.type =='c'}"  >내용</option>
                        <option value="w"  th:selected="${pageRequestDto.type =='w'}" >작성자</option>
                        <option value="tc"  th:selected="${pageRequestDto.type =='tc'}" >제목 + 내용</option>
                        <option value="tcw"  th:selected="${pageRequestDto.type =='tcw'}" >제목 + 내용 + 작성자</option>
                    </select>
                </div>
<!--                keyword 입력하는 란을 만들어준것-->
                <input class="form-control" name="keyword" th:value="${pageRequestDto.keyword}">
                <div class="input-group-append" id="button-addon4">
                    <button class="btn btn-outline-secondary btn-search" type="button">Search</button>
                    <button class="btn btn-outline-secondary btn-clear" type="button">Clear</button>
                </div>
            </div>
        </form>


        <table class="table table-striped">
            <thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col">Title</th>
                <th scope="col">Writer</th>
                <th scope="col">Regdate</th>
            </tr>
            </thead>
            <tbody>
            <tr th:each="dto : ${result.dtoList}" >
<!--                상세 보기시 검색 부분도 같이 들어가야한다-->
                <th scope="row"><a th:href="@{/guestbook/read(gno = ${dto.gno}, page=${result.page})}">
                    [[${dto.gno}]]
                </a>
                </th>
                <td>[[${dto.title}]]</td>
                <td>[[${dto.writer}]]</td>
                <td>[[${#temporals.format(dto.regDate, 'yyyy/MM/dd')}]]</td>
            </tr>
            </tbody>

        </table>
        <ul class="pagination h-100 justify-content-center align-items-center">
            <!--        이전 링크 출력-->
            <li class="page-item " th:if="${result.prev}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.start -1})}" tabindex="-1">이전</a>
            </li>
            <!--        페이지 번호 목록 출력-->
            <!--        현재 페이지 번호는 active클래스를 추가해서 비활성화-->
            <li th:class=" 'page-item ' + ${result.page == page?'active':''} " th:each="page: ${result.pageList}">
                <a class="page-link" th:href="@{/guestbook/list(page = ${page})}">
                    [[${page}]]
                </a>
            </li>
            <!--        다음 링크 출력-->
            <li class="page-item" th:if="${result.next}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.end + 1})}">다음</a>
            </li>

        </ul>
        <div class="modal" tabindex="-1" role="dialog">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Modal title</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <p>Modal body text goes here.</p>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                        <button type="button" class="btn btn-primary">Save changes</button>
                    </div>
                </div>
            </div>
        </div>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>


<!--        msg내용을 msg 변수에 저장 -->
        <script th:inline="javascript">
            // msg 에 내용을 msg 변수에 저장
            var msg = [[${msg}]];
            console.log(msg);
            // msg 가 존재하면 modal 창을 출력
            if(msg){
                $(".modal").modal();
            }
        </script>


    </th:block>
</th:block>
```

list.html 에 Script Code를 추가

```
//검색 폼 가져오기
var searchForm = $("#searchForm");

$('.btn-search').click(function(e){
    searchForm.submit();
});

$('.btn-clear').click(function(e){
    searchForm.empty().submit();
});
```

```
</script>


</th:block>
</th:block>
```





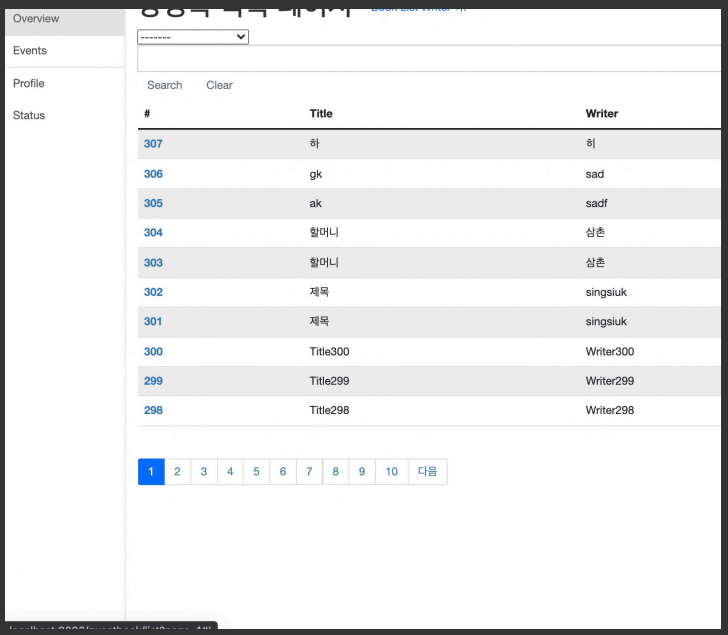
list.html 수정

수정 전

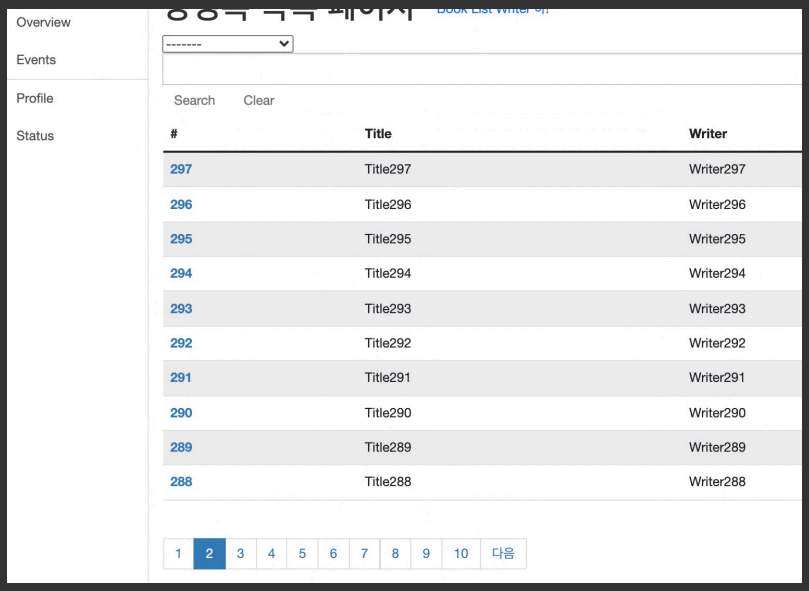
```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
    <th:block th:fragment="content">
        <h1 class="mt-4">방명록 목록 페이지
            <span>
                <a th:href="@{/guestbook/register}">
                    <button type="button" class="btn btn-outline-primary">
                        Book List Writer 하!
                    </button>
                </a>
            </span>
        </h1>

        <form action="/guestbook/list" method="get" id="searchForm">
            <div class="input-group">
                <input type="hidden" name="page" value = "1">
                <div class="input-group-prepend">
                    <select class="custom-select" name="type">
                        <option th:selected="${pageRequestDto.type == null}">-------</option>
                        <option value="t" th:selected="${pageRequestDto.type =='t'}" >제목</option>
                        <option value="c" th:selected="${pageRequestDto.type =='c'}"  >내용</option>
                        <option value="w"  th:selected="${pageRequestDto.type =='w'}" >작성자</option>
                        <option value="tc"  th:selected="${pageRequestDto.type =='tc'}" >제목 + 내용</option>
                        <option value="tcw"  th:selected="${pageRequestDto.type =='tcw'}" >제목 + 내용 + 작성자</option>
                    </select>
                </div>
<!--                keyword 입력하는 란을 만들어준것-->
                <input class="form-control" name="keyword" th:value="${pageRequestDto.keyword}">
                <div class="input-group-append" id="button-addon4">
                    <button class="btn btn-outline-secondary btn-search" type="button">Search</button>
                    <button class="btn btn-outline-secondary btn-clear" type="button">Clear</button>
                </div>
            </div>
        </form>


        <table class="table table-striped">
            <thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col">Title</th>
                <th scope="col">Writer</th>
                <th scope="col">Regdate</th>
            </tr>
            </thead>
            <tbody>
            <tr th:each="dto : ${result.dtoList}" >
<!--                상세 보기시 검색 부분도 같이 들어가야한다-->
                <th scope="row"><a th:href="@{/guestbook/read(gno = ${dto.gno}, page=${result.page})}">
                    [[${dto.gno}]]
                </a>
                </th>
                <td>[[${dto.title}]]</td>
                <td>[[${dto.writer}]]</td>
                <td>[[${#temporals.format(dto.regDate, 'yyyy/MM/dd')}]]</td>
            </tr>
            </tbody>

        </table>
        <ul class="pagination h-100 justify-content-center align-items-center">
            <!--        이전 링크 출력-->
            <li class="page-item " th:if="${result.prev}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.start -1})}" tabindex="-1">이전</a>
            </li>
            <!--        페이지 번호 목록 출력-->
            <!--        현재 페이지 번호는 active클래스를 추가해서 비활성화-->
            <li th:class=" 'page-item ' + ${result.page == page?'active':''} " th:each="page: ${result.pageList}">
                <a class="page-link" th:href="@{/guestbook/list(page = ${page})}">
                    [[${page}]]
                </a>
            </li>
            <!--        다음 링크 출력-->
            <li class="page-item" th:if="${result.next}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.end + 1})}">다음</a>
            </li>

        </ul>

                <div class="modal" tabindex="-1" role="dialog">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Modal title</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <p>Modal body text goes here.</p>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                        <button type="button" class="btn btn-primary">Save changes</button>
                    </div>
                </div>
            </div>
        </div>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>


<!--        msg내용을 msg 변수에 저장 -->
        <script th:inline="javascript">
            // msg 에 내용을 msg 변수에 저장
            var msg = [[${msg}]];
            console.log(msg);
            // msg 가 존재하면 modal 창을 출력
            if(msg){
                $(".modal").modal();
            }
            //검색 폼 가져오기
            var searchForm = $("#searchForm");

            $('.btn-search').click(function(e){
                searchForm.submit();
            });

            $('.btn-clear').click(function(e){
                searchForm.empty().submit();
            });

        </script>

    </th:block>
</th:block>
```

수정 후

```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
    <th:block th:fragment="content">
        <h1 class="mt-4">방명록 목록 페이지
            <span>
                <a th:href="@{/guestbook/register}">
                    <button type="button" class="btn btn-outline-primary">
                        Book List Writer 하!
                    </button>
                </a>
            </span>
        </h1>

        <form action="/guestbook/list" method="get" id="searchForm">
            <div class="input-group">
                <input type="hidden" name="page" value = "1">
                <div class="input-group-prepend">
                    <select class="custom-select" name="type">
                        <option th:selected="${pageRequestDto.type == null}">-------</option>
                        <option value="t" th:selected="${pageRequestDto.type =='t'}" >제목</option>
                        <option value="c" th:selected="${pageRequestDto.type =='c'}"  >내용</option>
                        <option value="w"  th:selected="${pageRequestDto.type =='w'}" >작성자</option>
                        <option value="tc"  th:selected="${pageRequestDto.type =='tc'}" >제목 + 내용</option>
                        <option value="tcw"  th:selected="${pageRequestDto.type =='tcw'}" >제목 + 내용 + 작성자</option>
                    </select>
                </div>
<!--                keyword 입력하는 란을 만들어준것-->
                <input class="form-control" name="keyword" th:value="${pageRequestDto.keyword}">
                <div class="input-group-append" id="button-addon4">
                    <button class="btn btn-outline-secondary btn-search" type="button">Search</button>
                    <button class="btn btn-outline-secondary btn-clear" type="button">Clear</button>
                </div>
            </div>
        </form>


        <table class="table table-striped">
            <thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col">Title</th>
                <th scope="col">Writer</th>
                <th scope="col">Regdate</th>
            </tr>
            </thead>
            <tbody>
            <tr th:each="dto : ${result.dtoList}" >
<!--                상세 보기시 검색 부분도 같이 들어가야한다-->
                <th scope="row"><a th:href="@{/guestbook/read(gno = ${dto.gno}, page=${result.page})}">
                    [[${dto.gno}]]
                </a>
                </th>
                <td>[[${dto.title}]]</td>
                <td>[[${dto.writer}]]</td>
                <td>[[${#temporals.format(dto.regDate, 'yyyy/MM/dd')}]]</td>
            </tr>
            </tbody>

        </table>
        <ul class="pagination h-100 justify-content-center align-items-center">

            <li class="page-item " th:if="${result.prev}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.start -1}, type=${pageRequestDTO.type},
keyword=${pageRequestDTO.keyword})}" tabindex="-1">이전</a>
            </li>

            <li th:class=" 'page-item ' + ${result.page == page?'active':''} " th:each="page: ${result.pageList}">
                <a class="page-link" th:href="@{/guestbook/list(page = ${page}, type=${pageRequestDTO.type}, keyword=$
{pageRequestDTO.keyword})}">
                    [[${page}]]
                </a>
            </li>

            <li class="page-item" th:if="${result.next}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.end + 1}, type=${pageRequestDTO.type},
keyword=${pageRequestDTO.keyword})}">다음</a>
            </li>

        </ul>
        <div class="modal" tabindex="-1" role="dialog">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Modal title</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <p>Modal body text goes here.</p>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                        <button type="button" class="btn btn-primary">Save changes</button>
                    </div>
                </div>
```


Overview
Events
Profile
Status

Search
Clear
Title Writer
307 하 하
306 gk sad
305 ak sadf
304 할머니 삼촌
303 할머니 삼촌
302 제목 singsiuk
301 제목 singsiuk
300 Title300 Writer300
299 Title299 Writer299
298 Title298 Writer298
1 2 3 4 5 6 7 8 9 10 다음


Overview
Events
Profile
Status

Search
Clear
Title Writer
297 Title297 Writer297
296 Title296 Writer296
295 Title295 Writer295
294 Title294 Writer294
293 Title293 Writer293
292 Title292 Writer292
291 Title291 Writer291
290 Title290 Writer290
289 Title289 Writer289
288 Title288 Writer288
1 2 3 4 5 6 7 8 9 10 다음

———— list.html 에서 상세보기 링크 수정

——— 수정 전

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
    <th:block th:fragment="content">
        <h1 class="mt-4">방명록 목록 페이지
            <span>
                <a th:href="@{/guestbook/register}">
                    <button type="button" class="btn btn-outline-primary">
                        Book List Writer 하!
                    </button>
                </a>
            </span>
        </h1>

        <form action="/guestbook/list" method="get" id="searchForm">
            <div class="input-group">
                <input type="hidden" name="page" value = "1">
                <div class="input-group-prepend">
                    <select class="custom-select" name="type">
                        <option th:selected="${pageRequestDto.type == null}">--------</option>
                        <option value="t" th:selected="${pageRequestDto.type =='t'}" >제목</option>
                        <option value="c" th:selected="${pageRequestDto.type =='c'}"  >내용</option>
                        <option value="w"  th:selected="${pageRequestDto.type =='w'}" >작성자</option>
                        <option value="tc"  th:selected="${pageRequestDto.type =='tc'}" >제목 + 내용</option>
                        <option value="tcw"  th:selected="${pageRequestDto.type =='tcw'}" >제목 + 내용 + 작성자</option>
                    </select>
                </div>
<!--                keyword 입력하는 란을 만들어준것-->
                <input class="form-control" name="keyword" th:value="${pageRequestDto.keyword}">
                <div class="input-group-append" id="button-addon4">
                    <button class="btn btn-outline-secondary btn-search" type="button">Search</button>
                    <button class="btn btn-outline-secondary btn-clear" type="button">Clear</button>
                </div>
            </div>
        </form>


        <table class="table table-striped">
            <thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col">Title</th>
                <th scope="col">Writer</th>
                <th scope="col">Regdate</th>
            </tr>
            </thead>
            <tbody>
            <tr th:each="dto : ${result.dtoList}" >
<!--                상세 보기시 검색 유형도 같이 들어가게끔!-->
                                    <th scope="row"><a th:href="@{/guestbook/read(gno = ${dto.gno}, page=${result.page})}">
                                        [[${dto.gno}]]
                                    </a>

                </th>
                <td>[[${dto.title}]]</td>
                <td>[[${dto.writer}]]</td>
                <td>[[${#temporals.format(dto.regDate, 'yyyy/MM/dd')}]]</td>
            </tr>
            </tbody>

        </table>
        <ul class="pagination h-100 justify-content-center align-items-center">

            <li class="page-item " th:if="${result.prev}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.start -1}, type=${pageRequestDto.type}, keyword=${pageRequestDto.keyword})}" tabindex="-1">이전</a>
            </li>

            <li th:class=" 'page-item ' + ${result.page == page?'active':''} " th:each="page: ${result.pageList}">
                <a class="page-link" th:href="@{/guestbook/list(page = ${page}, type=${pageRequestDto.type}, keyword=${pageRequestDto.keyword})}">
                    [[${page}]]
                </a>
            </li>

            <li class="page-item" th:if="${result.next}">
                <a class="page-link" th:href="@{/guestbook/list(page= ${result.end + 1}, type=${pageRequestDto.type}, keyword=${pageRequestDto.keyword})}">다음</a>
            </li>

        </ul>

        <div class="modal" tabindex="-1" role="dialog">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Modal title</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <p>Modal body text goes here.</p>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                        <button type="button" class="btn btn-primary">Save changes</button>
                    </div>
                </div>
            </div>
        </div>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.min.css">
        <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/js/bootstrap.bundle.min.js"></script>


<!--        msg내용을 msg 변수에 저장 -->
        <script th:inline="javascript">
            // msg 에 내용을 msg 변수에 저장
            var msg = [[${msg}]];
            console.log(msg);
            // msg 가 존재하면 modal 창을 출력
            if(msg){
                $('.modal').modal();
            }
            //검색 폼 가져오기
            var searchForm = $("#searchForm");

            $('.btn-search').click(function(e){
                searchForm.submit();
            });

            $('.btn-clear').click(function(e){
                searchForm.empty().submit();
            });

        </script>


    </th:block>
</th:block>
```

——— 수정 후

```
<a th:href="@{/guestbook/read(gno = ${dto.gno}, page=${result.page} , type=${pageRequestDTO.type}, keyword=${pageRequestDTO.keyword})}">
    [[${dto.gno}]]
</a>
```

—— read.html 수정

— 수정 전

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
  <th:block th:fragment="content">
    <h1 class="mt-4">방명록 상세보기</h1>

    <div class="form-group">
      <label>Gno</label>
      <input type="text" class="form-control" name="gno" th:value="${dto.gno}" readonly >
    </div>

    <div class="form-group">
      <label >Title</label>
      <input type="text" class="form-control" name="title" th:value="${dto.title}" readonly >
    </div>
    <div class="form-group">
      <label >Content</label>
      <textarea class="form-control" rows="5" name="content" readonly>[[${dto.content}]]</textarea>
    </div>
    <div class="form-group">
      <label >Writer</label>
      <input type="text" class="form-control" name="writer" th:value="${dto.writer}" readonly>
    </div>
    <div class="form-group">
      <label >RegDate</label>
      <input type="text" class="form-control" name="regDate" th:value="${#temporals.format(dto.regDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>
    <div class="form-group">
      <label >ModDate</label>
      <input type="text" class="form-control" name="modDate" th:value="${#temporals.format(dto.modDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>


    <a th:href="@{/guestbook/modify(gno=${dto.gno},page=${requestDto.page})}"><button type="button" class="btn btn-info">수정</button></
a>
    <a th:href="@{/guestbook/list(page=${requestDto.page})}"><button type="button" class="btn btn-info">목록</button></a>

  </th:block>
</th:block>
```

— 수정 후

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">
  <th:block th:fragment="content">
    <h1 class="mt-4">방명록 상세보기</h1>

    <div class="form-group">
      <label>Gno</label>
      <input type="text" class="form-control" name="gno" th:value="${dto.gno}" readonly >
    </div>

    <div class="form-group">
      <label >Title</label>
      <input type="text" class="form-control" name="title" th:value="${dto.title}" readonly >
    </div>
    <div class="form-group">
      <label >Content</label>
      <textarea class="form-control" rows="5" name="content" readonly>[[${dto.content}]]</textarea>
    </div>
    <div class="form-group">
      <label >Writer</label>
      <input type="text" class="form-control" name="writer" th:value="${dto.writer}" readonly>
    </div>
    <div class="form-group">
      <label >RegDate</label>
      <input type="text" class="form-control" name="regDate" th:value="${#temporals.format(dto.regDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>
    <div class="form-group">
      <label >ModDate</label>
      <input type="text" class="form-control" name="modDate" th:value="${#temporals.format(dto.modDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
    </div>


    <!--         <a th:href="@{/guestbook/modify(gno = ${dto.gno}, page=${requestDTO.page})}"><button type="button" class="btn btn-
primary">Modify</button></a>-->

    <!--         <a th:href="@{/guestbook/list(page=${requestDTO.page})}"><button type="button" class="btn btn-info">List</button></a>-->

    <a th:href="@{/guestbook/modify(gno = ${dto.gno}, page=${requestDTO.page}, type=${requestDTO.type}, keyword =$
{requestDTO.keyword})}">
      <button type="button" class="btn btn-primary">수정</button>
    </a>

    <a th:href="@{/guestbook/list(page=${requestDTO.page} , type=${requestDTO.type}, keyword =${requestDTO.keyword})}">
      <button type="button" class="btn btn-info">목록</button>
    </a>

  </th:block>

</th:block>
```

——— Modify.html 파일의 form 에 type 과 form 을 추가

```
<!DOCTYPE html>
<!--Thymeleaf 의 th를 사용하기 위한 설정 -->
<!--정식 버전에서는 생략해도 상관 없다 -->
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<!--이 파일의 내용을 출력할 레이아웃 파일의 경로를 설정 -->
<!--// 출력할 때 layout/basic 에서가 출력하겠다 -->

<th:block th:replace="~{/layout/basic :: setContent(~{this::content} )}">

<!--  기본 틀을 출력한 상태에서 변경되는 내용-->
    <th:block th:fragment="content">
        <h1 class="mt-4">방명록 수정</h1>
        <input type="hidden" name="type" th:value="${requestDTO.type}" >
        <input type="hidden" name="keyword" th:value="${requestDTO.keyword}" >
```

```
            <!--        수정을 하기 위한 폼-->
    <form action="/guestbook/modify" method="post">
        <!--페이지 번호 -->
<!--        페이지 번호는 입력하게 출력할 필요는 없어서 숨김-->
        <input type="hidden" name="page" th:value="${requestDto.page}">

<!--        갱신하는 출력은 하지만 수정할 수 없도록 읽기 전용-->
        <div class="form-group">
          <label >Gno</label>
          <input type="text" class="form-control" name="gno" th:value="${dto.gno}" readonly="readonly" >
        </div>
<!--        title 은 수정 입력도구로 출력-->
        <div class="form-group">
          <label >Title</label>
          <input type="text" class="form-control" name="title" th:value="${dto.title}" >
        </div>
<!--        내용을 입력도구로 출력 / textarea-->
        <div class="form-group">
          <label >Content</label>
          <textarea class="form-control" row="5" name="content">[[${dto.content}]]</textarea>
        </div>
<!--        작성자도 readonly -->
        <div class="form-group">
          <label >Writer</label>
          <input type="text" class="form-control" name="writer" th:value="${dto.writer}" readonly="readonly">
        </div>
<!--        등록일도 읽고 수정하도록 함수, 사용자가 입력하는게 아니기 때문-->
        <div class="form-group">
          <label >RegDate</label>
          <input type="text" class="form-control" th:value="${#temporals.format(dto.regDate, 'yyyy/MM/dd HH:mm:ss')}" readonly="readonly">
        </div>
<!--        수정일 역시 수정하도록 함수, 사용자가 입력하는게 아니기 때문-->
        <div class="form-group">
          <label >ModDate</label>
          <input type="text" class="form-control" th:value="${#temporals.format(dto.modDate, 'yyyy/MM/dd HH:mm:ss')}" readonly="readonly">
        </div>

    </form>

    <button type="button" class="btn btn-primary modifyBtn">수정</button>
    <button type="button" class="btn btn-info listBtn">목록</button>
    <button type="button" class="btn btn-danger removeBtn">삭제</button>
<!--    bootstrap 을 사용할 때 필요하는 요소-->
<!--    1. bootstrap 을 export 하려면 jquery를 먼저-->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>

    <script th:inline="javascript">
     // 만 jquery에서 인식하를 못하면, ----> $is not found 발생 ---> 원인가 없다
     var actionForm = $("form"); //form 태그 객체

     $(".removeBtn").click(function(){
       actionForm
             .attr('action', '/guestbook/remove')
             .attr('method','post');

       actionForm.submit();
     });
     $(".modifyBtn").click(function() {
       if(!confirm('수정하시겠습니까?')){
         return ;
       }
       actionForm
             .attr('action', '/guestbook/modify')
             .attr('method', 'post')
             .submit();
     });
```

```
    $(".listBtn").click(function() {
            //var pageInfo = $("input[name='page']");
            var page = $("input[name='page']");
            var type = $("input[name='type']");
            var keyword = $("input[name='keyword']");
            actionForm.empty(); //form 태그의 모든 내용을 지우고
            actionForm.append(page);
            actionForm
                    .attr("action", "/guestbook/list")
                    .attr("method","get");
            actionForm.submit();
        })
    </script>
  </th:block>
</th:block>
```

GuestBookController 수정

```java
package com.singsiuk.guestbook.controller;

import com.singsiuk.guestbook.dto.GuestBookDto;
import com.singsiuk.guestbook.dto.PageRequestDto;
import com.singsiuk.guestbook.dto.PageResponseDto;
import com.singsiuk.guestbook.service.GuestBookService;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

// 로그 추력을 위한 Annotation
@Log4j2
//PageController 를 만들기 위한 Annotation
@Controller
@RequiredArgsConstructor
public class GuestBookController {
    // Service 주입
    private final GuestBookService service;

    @GetMapping("/")
    public String main(){
        // 배포시 지워진다
        log.info("시작요청");
        //Template 에 있는 Guestbook 디렉토리의 list.html 을 출력

    return "redirect:/guestbook/list";
    }

    @GetMapping("/guestbook/list")
    public void list(PageRequestDto pageRequestDto, Model model){
        log.info("Show List");
        PageResponseDto result = service.getList(pageRequestDto);
        model.addAttribute("result",result);
    }

    @GetMapping("/guestbook/register")
    public void register(){
        log.info("삽입 요청 page 로 이동 ");
    }

    @PostMapping("/guestbook/register")
    public String register(GuestBookDto dto, RedirectAttributes redirectAttributes){
        log.info("Insert Request");
        // 삽입 처리
        Long gno = service.register(dto);
        // 리다이렉트 할 때 한번만 사용하는 데이터 생성
        redirectAttributes.addFlashAttribute(
                "msg", gno+"Insert");
        // 작업 후 목록보기로 redirect
        return "redierct:/guestbook/list";
    }

    @GetMapping({"/guestbook/read", "guestbook/modify"})
    // 파라미터 중 gno는 gno에 대입
    // 나머지는 requestDto 에 대입
    // 다음과 같은 페이지를 전송
    public void read(long gno,
                     @ModelAttribute("requestDto")
                             PageRequestDto requestDto,
                                       Model model){
        GuestBookDto dto = service.read(gno);
        model.addAttribute("dto",dto);
    }

    @PostMapping("/guestbook/modify")
    public String modify(GuestBookDto dto, @ModelAttribute("requestDto")
            PageRequestDto requestDto,RedirectAttributes redirectAttributes){
        //수정 method 수행
        service.modify(dto);
        //전달할 데이터 생성
        redirectAttributes.addAttribute("page", requestDto.getPage());
        redirectAttributes.addAttribute("gno",dto.getGno());
        redirectAttributes.addAttribute("type",requestDto.getType());
        redirectAttributes.addAttribute("keyword",requestDto.getKeyword());

        //상세보기로 redirect
        return "redirect:/guestbook/read";
    }

    @PostMapping("/guestbook/remove")
    public String modify(Long gno, RedirectAttributes redirectAttributes){
        //삭제 method 수행
        service.remove(gno);
        //전달할 데이터 생성
        redirectAttributes.addFlashAttribute("msg", gno+"삭제");
        //상세보기로 redirect
        return "redirect:/guestbook/list";
    }
}
```