

5. 스프링Web MVC구현하기(CRUD)

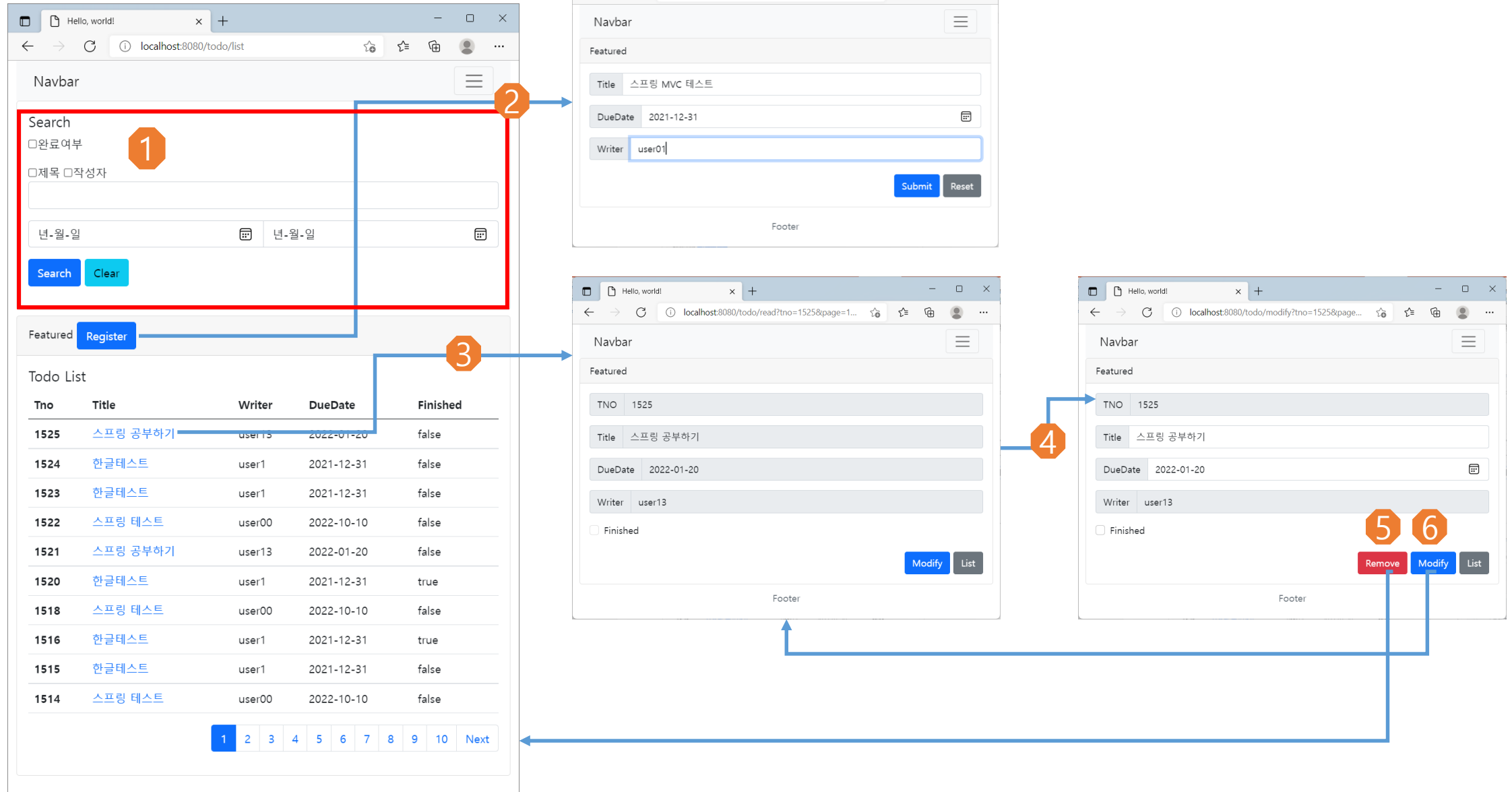
목차

1. 프로젝트 구현 목표
2. 데이터베이스 테이블 설계
3. ModelMapper 설정과 @Configuration
4. 화면디자인 -부트스트랩적용
5. Mybatis와 스프링 영속성 처리
6. 리스트, 글등록, 상세페이지, 수정, 삭제 구현
- 7.페이징처리
8. 검색/필터링

프로젝트의 구현 목표

- 1) 검색과 필터링을 적용할 수 있는 화면을 구성하고 MyBatis의 동적 쿼리를 이용해서 상황에 맞는 Todo들을 검색합니다.
- 2) 새로운 Todo를 등록할 때 문자열/boolean/LocalDate를 자동으로 처리하도록 합니다.
- 3) 목록에서 조회 화면으로 이동할 때 모든 검색/필터링/페이징 조건을 유지하도록 구성합니다.
- 4) 조회화면에서는 모든 조건을 유지한 채로 수정/삭제화면으로 이동하도록 구성합니다.
- 5) 삭제 시에는 다시 목록 화면으로 이동합니다.
- 6) 수정 시에는 다시 조회 화면으로 이동하지만 검색/필터링/페이징 조건은 초기화 합니다.

프로젝트 전체구조



데이터베이스 테이블

```
drop table tbl_board;
```

```
create table tbl_board (  
    bno int auto_increment primary key ,  
    title varchar(100) not null,  
    postdate date not null,  
    writer varchar(50) not null,  
    content text not null,  
    visitcount int not null default 0  
);
```

ModelMapper 설정과 @Configuration

스프링의 경우 XML파일을 통한 설정 외에도 @Configuration이 있는 클래스를 이용하여 설정을 지정할 수 있음

```
package org.zerock.springex.config;
```

```
import org.modelmapper.ModelMapper;  
import org.modelmapper.convention.MatchingStrategies;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;
```

@Configuration

```
public class ModelMapperConfig {
```

@Bean

```
public ModelMapper getMapper() {  
    ModelMapper modelMapper = new ModelMapper();  
    modelMapper.getConfiguration()  
        .setFieldMatchingEnabled(true)  
        .setFieldAccessLevel(org.modelmapper.config.Configuration.AccessLevel.PRIVATE)  
        .setMatchingStrategy(MatchingStrategies.LOOSE);  
  
    return modelMapper;  
}
```

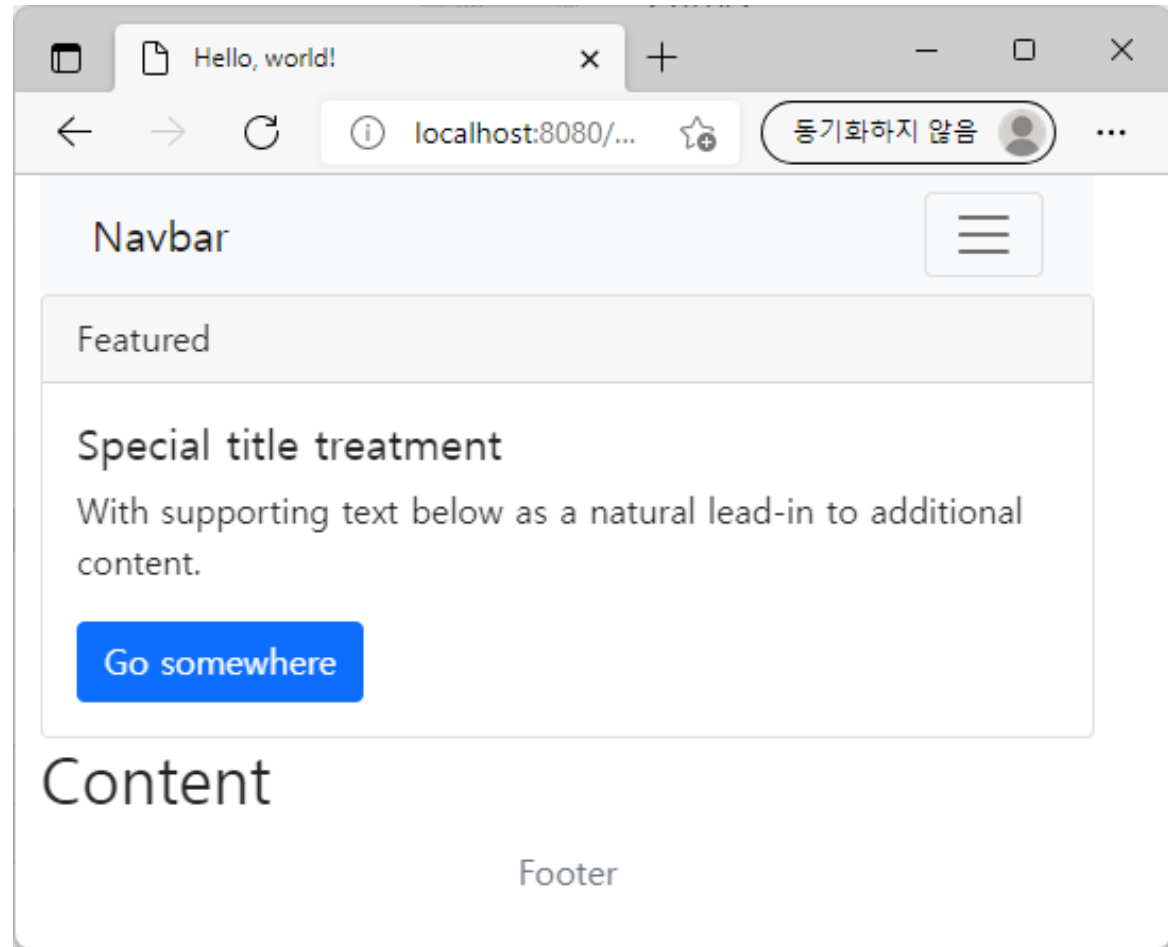
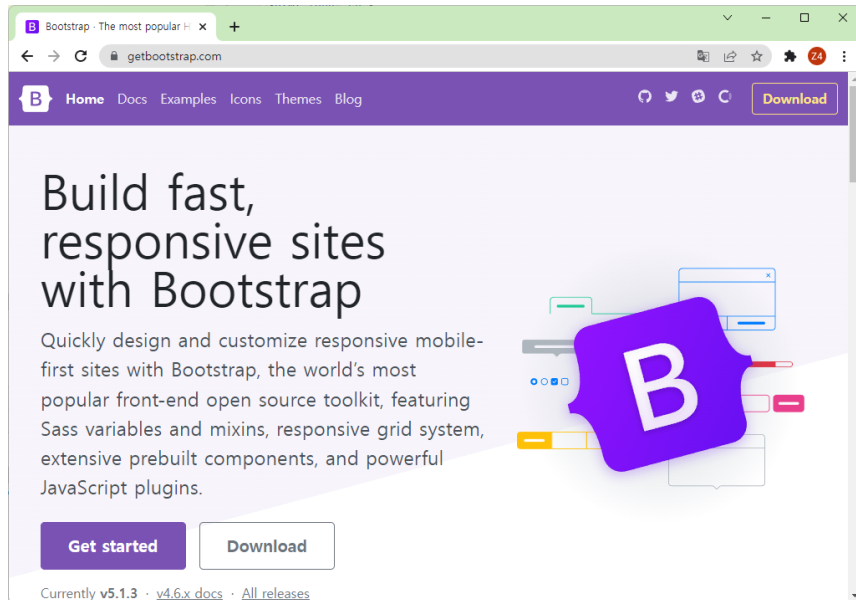
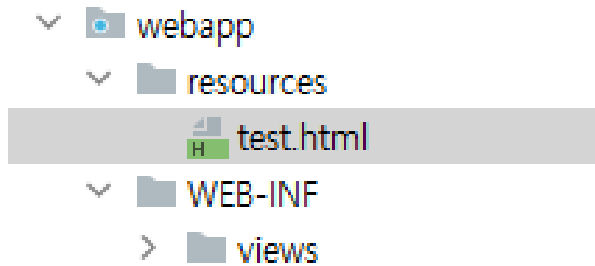
```
}
```



```
<context:component-scan base-package="org.zerock.springex.config"/>
```

화면디자인 - 부트스트랩 (<https://www.w3schools.com/>)

- 부트스트랩을 이용해서 간단한 화면 구성
- 컴포넌트를 이용해서 레이아웃이나 화면 디자인 가능



MyBatis와 스프링을 이용한 영속 처리

- MyBatis를 이용해서 SQL을 처리하고 테스트
- 개발의 단계
 - VO 클래스 개발
 - Mapper인터페이스 개발
 - XML을 이용해서 SQL 작성
 - 테스트 코드의 개발

▼ java
▼ org.zerock.springex
 > config
 > controller
 ▼ domain
 c TodoVO
 dto

```
package org.zerock.springex.domain;
```

```
import lombok.*;  
import java.time.LocalDate;
```

```
@Getter  
@ToString  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
public class TodoVO {  
    private Long tno;  
    private String title;  
    private LocalDate dueDate;  
    private String writer;  
    private boolean finished;  
}
```

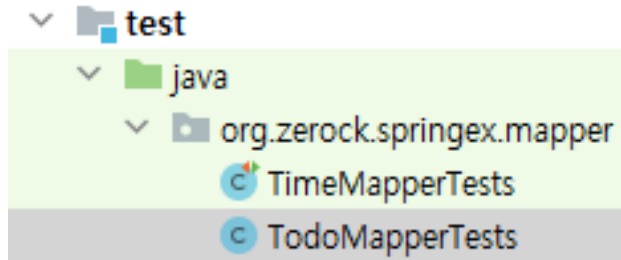
▼ mapper
 I TimeMapper
 I TimeMapper2
 I TodoMapper

▼ resources
 ▼ mappers
 TimeMapper2.xml
 TodoMapper.xml
 log4j2.xml

```
package org.zerock.springex.mapper;
```

```
public interface TodoMapper {  
    String getTime();  
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper  
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >  
<mapper namespace="org.zerock.springex.mapper.TodoMapper">  
    <select id="getTime" resultType="string">  
        select now()  
    </select>  
</mapper>
```



```
package org.zerock.springex.mapper;

import lombok.extern.log4j.Log4j2;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit.jupiter.SpringExtension;
```

```
@Log4j2
@ExtendWith(SpringExtension.class)
@ContextConfiguration(locations="file:src/main/webapp/WEB-INF/root-context.xml")
public class TodoMapperTests {

    @Autowired(required = false)
    private TodoMapper todoMapper;

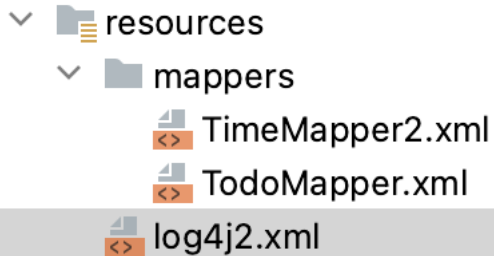
    @Test
    public void testGetTime() {

        log.info(todoMapper.getTime());
    }
}
```

```
09:14:34 INFO [org.springframework.test.context.support.DefaultTestContextBootstrapper] Loaded default TestExecutionLi
09:14:34 INFO [org.springframework.test.context.support.DefaultTestContextBootstrapper] Using TestExecutionListeners:
09:14:35 INFO [com.zaxxer.hikari.HikariDataSource] HikariPool-1 - Starting...
09:14:35 INFO [com.zaxxer.hikari.pool.HikariPool] HikariPool-1 - Added connection org.mariadb.jdbc.Connection@7e7f0216
09:14:35 INFO [com.zaxxer.hikari.HikariDataSource] HikariPool-1 - Start completed.
09:14:35 INFO [org.zerock.springex.mapper.TodoMapperTests] 2022-03-09 21:14:35
BUILD SUCCESSFUL in 2s
```

로그 레벨의 조정

- MyBatis와 JDBC의 상세 로그를 위한 로그 레벨 조정



```
<?xml version="1.0" encoding="UTF-8"?>

<configuration status="INFO">
  <Appenders>
    <!-- 콘솔 -->
    <Console name="console" target="SYSTEM_OUT">
      <PatternLayout charset="UTF-8" pattern="%d{hh:mm:ss} %5p [%c] %m%n"/>
    </Console>
  </Appenders>

  <loggers>
    <logger name="org.springframework" level="INFO" additivity="false">
      <appender-ref ref="console" />
    </logger>
    <logger name="org.zerock" level="INFO" additivity="false">
      <appender-ref ref="console" />
    </logger>
    <logger name="org.zerock.springex.mapper" level="TRACE" additivity="false">
      <appender-ref ref="console" />
    </logger>
    <root level="INFO" additivity="false">
      <AppenderRef ref="console"/>
    </root>
  </loggers>

</configuration>
```

Todo 기능 개발(insert)

- 개발 순서
 - TodoMapper -> TodoService -> TodoController -> JSP

```
package org.zerock.springex.mapper;

import org.zerock.springex.domain.TODOVO;

public interface TodoMapper {

    String getTime();

    void insert(TODOVO todoVO);
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="org.zerock.springex.mapper.TODOMapper">

    <select id="getTime" resultType="string">
        select now()
    </select>

    <insert id="insert">
        insert into tbl_todo (title, dueDate, writer) values ( #{title}, #{dueDate}, #{writer})
    </insert>

</mapper>
```

TodoService와 TodoServiceImpl

> mapper

▼ service

TodoService

```
package org.zerock.springex.service;

import org.zerock.springex.dto.TODO;

public interface TodoService {

    void register(TODO todo);

}
```

```
package org.zerock.springex.service;

import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.modelmapper.ModelMapper;
import org.springframework.stereotype.Service;
import org.zerock.springex.domain.TODO;
import org.zerock.springex.dto.TODO;
import org.zerock.springex.mapper.TODO;

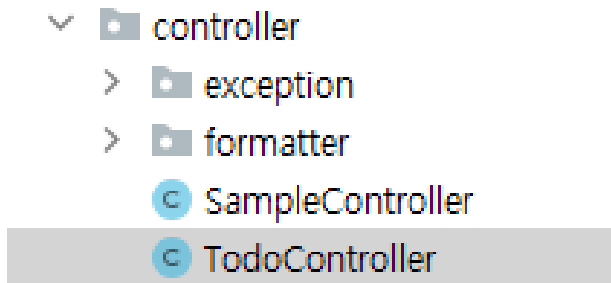
@Service
@Log4j2
@RequiredArgsConstructor
public class TodoServiceImpl implements TodoService{

    private final TODO todo;
    private final ModelMapper modelMapper;

    @Override
    public void register(TODO todo) {
        log.info(modelMapper);
        TODO todo = modelMapper.map(todo, TODO.class );
        log.info(todo);
        todoMapper.insert(todo);
    }

}
```

TodoController의 GET/POST 처리



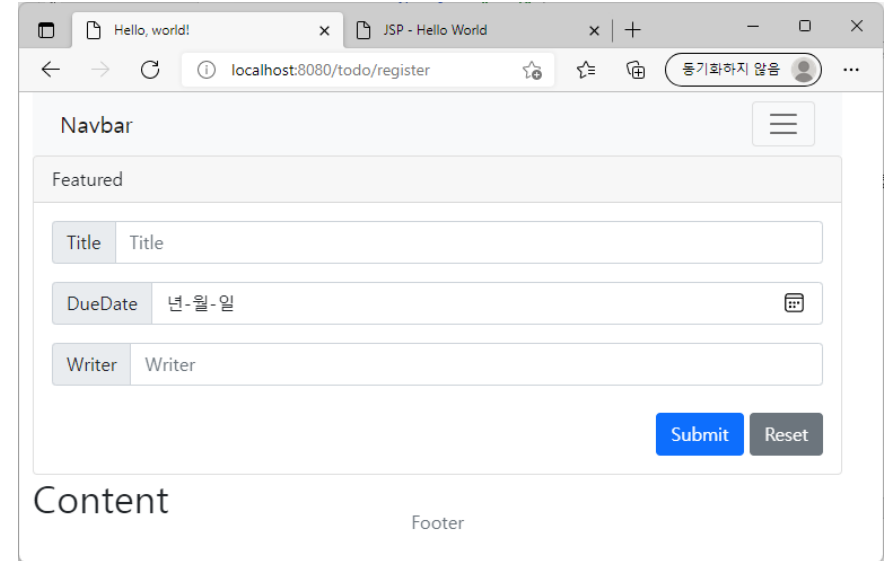
```
package org.zerock.springex.controller;

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.RequestMapping;
```

```
@Controller
@RequestMapping("/todo")
@Log4j2
public class TodoController {

    @RequestMapping("/list")
    public void list(Model model){
        log.info("todo list.....");
    }

    @GetMapping("/register")
    public void registerGET() {
        log.info("GET todo register.....");
    }
}
```

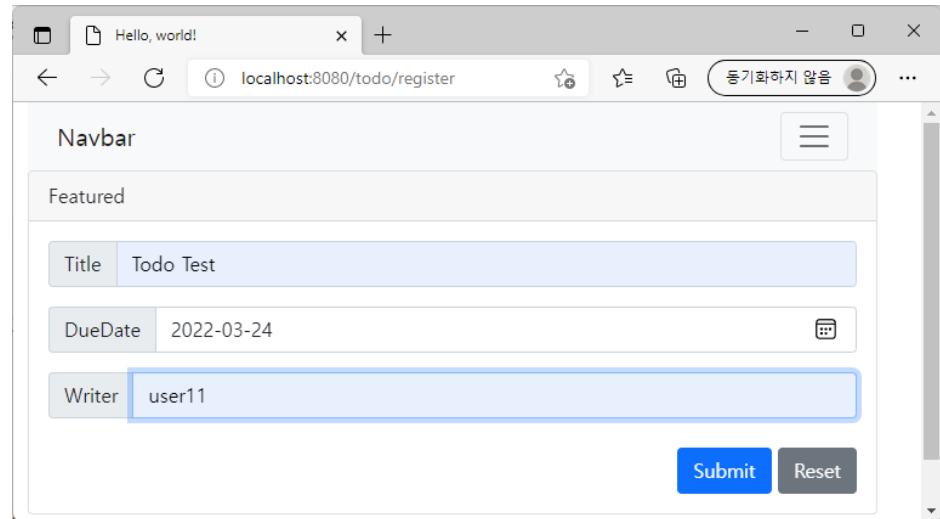


```
@PostMapping("/register")
public String registerPost(TodoDTO todoDTO, RedirectAttributes redirectAttributes) {
    log.info("POST todo register.....");

    log.info(todoDTO);

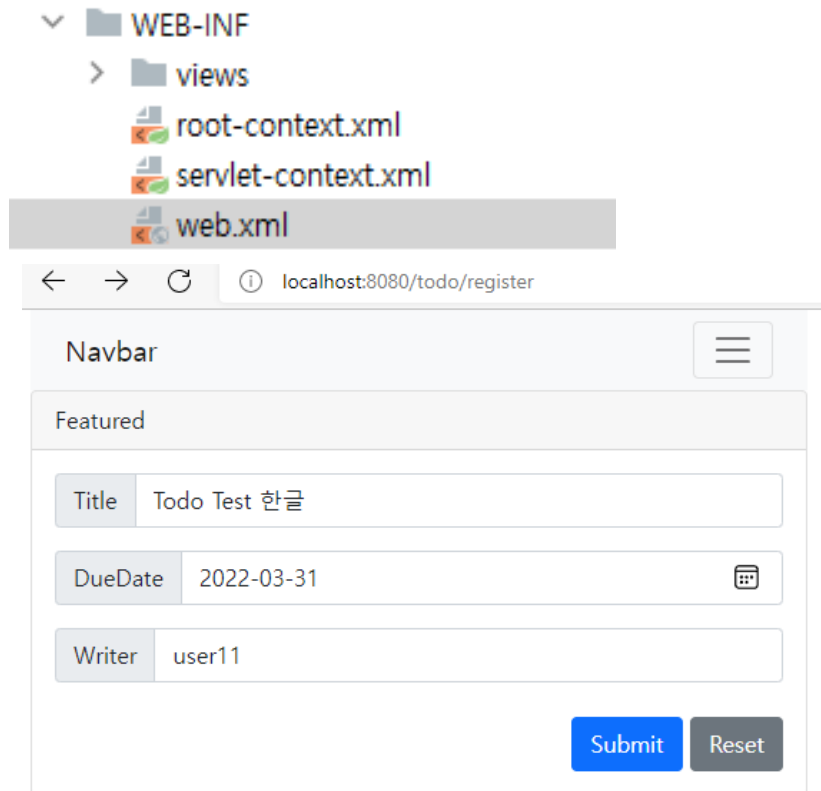
    return "redirect:/todo/list";
}
```

'/todo/list'에 대한 처리가 아직 이루어지지 않은 상황이지만
한글 깨짐등의 문제를 먼저 처리하도록 구성



The screenshot shows a web browser window with a single tab titled 'Hello, world!'. The address bar displays 'localhost:8080/todo/register'. The page content includes a 'Navbar' with a hamburger menu icon and a 'Featured' section. This section contains a form with three input fields: 'Title' (containing 'Todo Test'), 'DueDate' (containing '2022-03-24'), and 'Writer' (containing 'user11'). Below the form are two buttons: 'Submit' (blue) and 'Reset' (grey). The browser's user interface includes back, forward, and refresh buttons, as well as a user profile icon with the text '로그인하지 않음'.

UTF-8 필터 처리(Tomcat 10버전부터 필요 없음)



```
<filter>
  <filter-name>encoding</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</
filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>encoding</filter-name>
  <servlet-name>appServlet</servlet-name>
</filter-mapping>
```

POST todo register.....

TodoDTO(tno=null, title=Todo Test 한글, dueDate=2022-03-31, finished=false, writer=user11)

todo list.....

@Valid를 이용한 서버사이드 검증

- hibernate-validate 라이브러리를 이용해서 서버사이드에서 검증

> test
build.gradle
gradlew

```
@ToString
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class TodoDTO {

    private Long tno;

    @NotEmpty
    private String title;
    @Future
    private LocalDate dueDate;
    private boolean finished;
    @NotEmpty
    private String writer;

}
```

```
@PostMapping("/register")
public String registerPost(@Valid TodoDTO todoDTO,
                           BindingResult bindingResult,
                           RedirectAttributes redirectAttributes) {

    log.info("POST todo register.....");

    if(bindingResult.hasErrors()) {
        log.info("has errors.....");
        redirectAttributes.addFlashAttribute("errors", bindingResult.getAllErrors() );
        return "redirect:/todo/register";
    }

    log.info(todoDTO);
    return "redirect:/todo/list";
}
```

Featured

Title	Todo Test 한글
DueDate	2022-03-31
Writer	Writer

Submit Reset

```
[org.zerock.springex.controller.TODOController] has errors.....
[org.zerock.springex.controller.TODOController] GET todo register.....
```

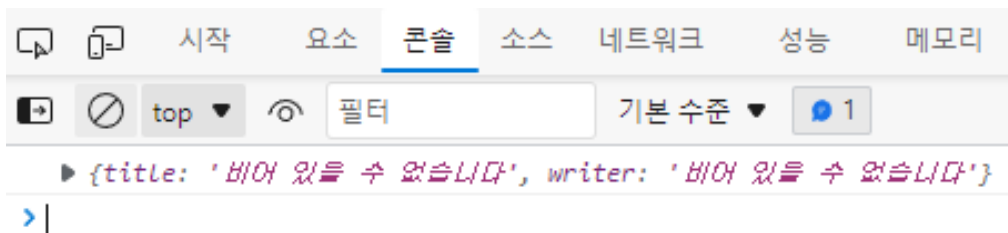
서버 검증 메시지 처리

- BindResult의 에러 메시지를 좀 더 편리하게 JavaScript 로 처리
- 상황에 따라서 처리가 가능하다는 장점

```
<script>

const serverValidResult = {}
<c:forEach items="${errors}" var="error">
serverValidResult['${error.getField()}'] = '${error.defaultMessage}'
</c:forEach>
console.log(serverValidResult)

</script>
```



```
<script>

const serverValidResult = {}

console.log(serverValidResult)

</script>

<script>

const serverValidResult = {}

serverValidResult['title'] = '비어 있을 수 없습니다'

serverValidResult['writer'] = '비어 있을 수 없습니다'

console.log(serverValidResult)

</script>
```

Todo 목록 기능 개발

TodoMapper 기능 개발

```
public interface TodoMapper {  
  
    String getTime();  
    void insert(TodoVO todoVO);  
    List<TodoVO> selectAll();  
  
}
```

```
<select id="selectAll" resultType="org.zerock.spring  
ex.domain.TODOVO">  
    select * from tbl_todo order by tno desc  
</select>
```

TodoService 기능 개발

```
package org.zerock.springex.service;  
  
import org.zerock.springex.dto.TODODTO;  
import java.util.List;  
  
public interface TodoService {  
  
    void register(TODODTO todoDTO);  
    List<TODODTO> getAll();  
  
}
```

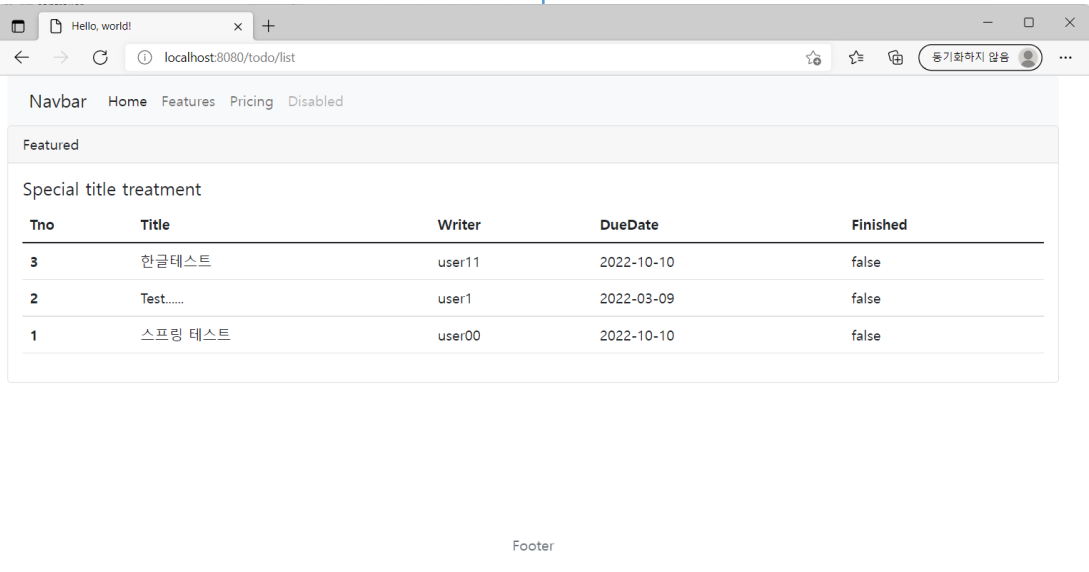
```
@Override  
public List<TODODTO> getAll() {  
  
    List<TODODTO> dtoList = todoMapper.selectAll().stream()  
        .map(vo -> modelMapper.map(vo, TODODTO.class))  
        .collect(Collectors.toList());  
  
    return dtoList;  
}
```

TodoController의 목록 처리

```
@RequestMapping("/list")
public void list(Model model){

    log.info("todo list.....");
    model.addAttribute("dtoList", todoService.getAll());
}
```

```
<div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">Tno</th>
                <th scope="col">Title</th>
                <th scope="col">Writer</th>
                <th scope="col">DueDate</th>
                <th scope="col">Finished</th>
            </tr>
        </thead>
        <tbody>
            <c:forEach items="${dtoList}" var="dto">
                <tr>
                    <th scope="row"><c:out value="${dto.tno}"/></th>
                    <td><c:out value="${dto.title}"/></td>
                    <td><c:out value="${dto.writer}"/></td>
                    <td><c:out value="${dto.dueDate}"/></td>
                    <td><c:out value="${dto.finished}"/></td>
                </tr>
            </c:forEach>
        </tbody>
    </table>
</div>
</div>
</div>
</div>
```



Tno	Title	Writer	DueDate	Finished
3	한글 테스트	user11	2022-10-10	false
2	Test.....	user1	2022-03-09	false
1	스프링 테스트	user00	2022-10-10	false

Todo 조회 기능 개발

- '/todo/read?tno=xx ' 와 같이 TodoController호출시 동작

TodoMapper 기능 개발

```
public interface TodoMapper {  
  
    String getTime();  
    void insert(TodoVO todoVO);  
    List<TodoVO> selectAll();  
    TodoVO selectOne(Long tno);  
}
```

```
<mapper namespace="org.zerock.springex.mapper.TodoMapper">  
  
    ...  
  
    <select id="selectOne" resultType="org.zerock.springex.domain.TodoVO">  
        select * from tbl_todo where tno = #{tno}  
    </select>  
</mapper>
```

TodoService 기능 개발

```
public interface TodoService {  
  
    void register(TodoDTO todoDTO);  
    List<TodoDTO> getAll();  
    TodoDTO getOne(Long tno);  
}
```

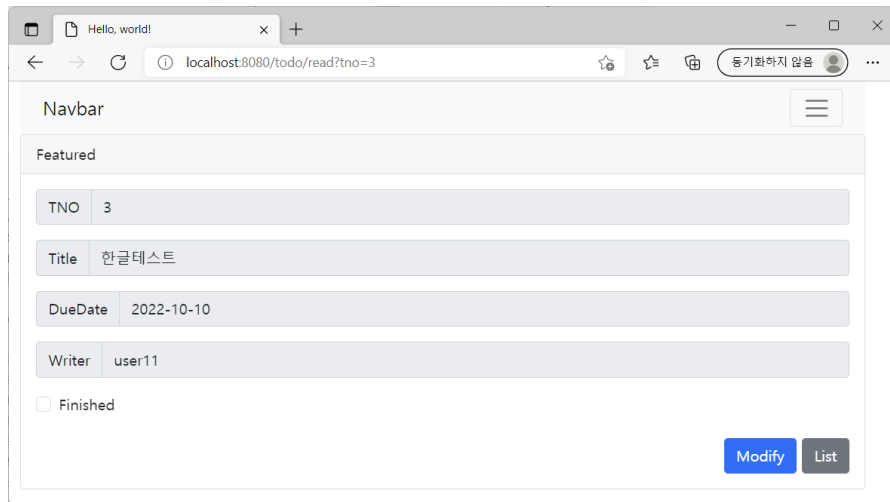
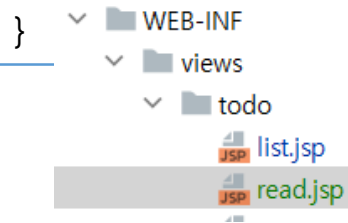
```
@Override  
public TodoDTO getOne(Long tno) {  
  
    TodoVO todoVO = todoMapper.selectOne(tno);  
    TodoDTO todoDTO = modelMapper.map(todoVO, TodoDTO.class);  
    return todoDTO;  
}
```

TodoController의 개발

```
@GetMapping("/read")
public void read(Long tno, Model model){
```

```
    TodoDTO todoDTO = todoService.getOne(tno);
    log.info(todoDTO);
```

```
    model.addAttribute("dto", todoDTO);
```



```
<div class="card-body">
  <div class="input-group mb-3">
    <span class="input-group-text">TNO</span>
    <input type="text" name="tno" class="form-control"
      value=<c:out value="{dto.tno}"></c:out> readonly>
  </div>
  <div class="input-group mb-3">
    <span class="input-group-text">Title</span>
    <input type="text" name="title" class="form-control"
      value='<c:out value="{dto.title}"></c:out>' readonly>
  </div>

  <div class="input-group mb-3">
    <span class="input-group-text">DueDate</span>
    <input type="date" name="dueDate" class="form-control"
      value=<c:out value="{dto.dueDate}"></c:out> readonly>
  </div>

  <div class="input-group mb-3">
    <span class="input-group-text">Writer</span>
    <input type="text" name="writer" class="form-control"
      value=<c:out value="{dto.writer}"></c:out> readonly>
  </div>

  <div class="form-check">
    <label class="form-check-label">
      Finished &nbsp;<input type="checkbox" name="finished"
        value=<c:out value="{dto.finished}"></c:out> checked="">
    </label>
  </div>

  <div class="my-4">
    <div class="float-end">
      <button type="button" class="btn btn-primary">Modify</button>
      <button type="button" class="btn btn-secondary">List</button>
    </div>
  </div>
</div>
```

수정/삭제를 위한 링크 처리

- 화면상에서 'Modify'버튼을 클릭해서 GET방식으로 이동

```
<div class="my-4">
  <div class="float-end">
    <button type="button" class="btn btn-primary">Modify</button>
    <button type="button" class="btn btn-secondary">List</button>
  </div>
</div>

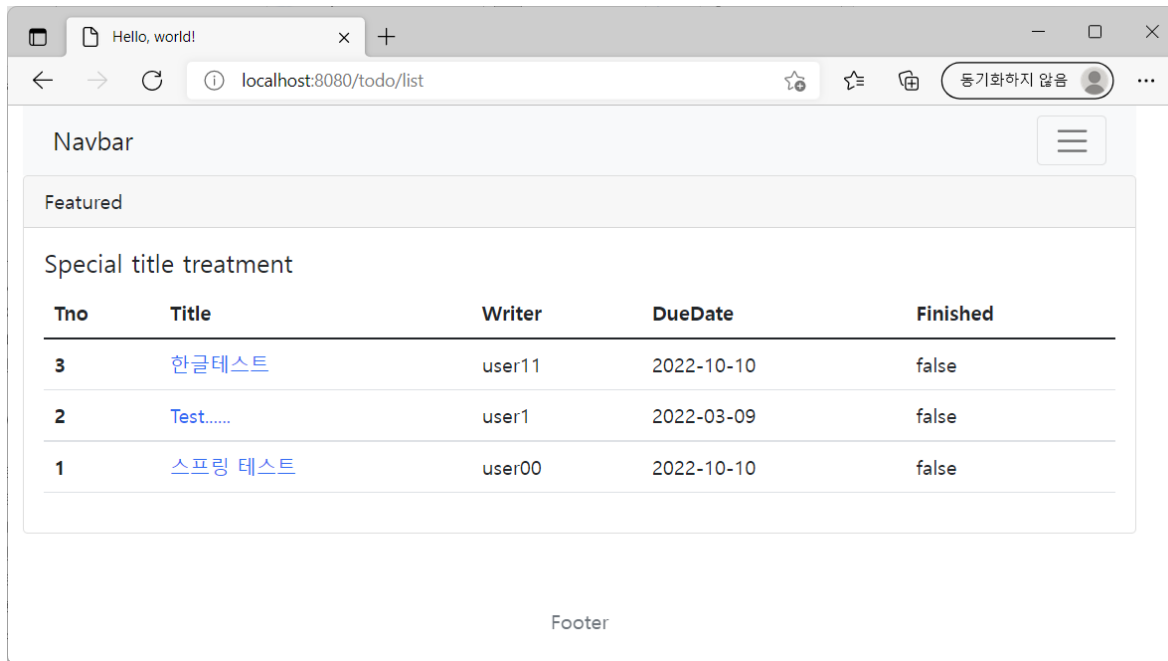
<script>
  document.querySelector(".btn-primary").addEventListener("click", function(e){
    self.location = "/todo/modify?tno="+${dto.tno}
  },false)

  document.querySelector(".btn-secondary").addEventListener("click", function(e){
    self.location = "/todo/list";
  },false)
</script>
```

list.jsp의 링크 처리

- list.jsp에서는 제목부분에 링크를 통해서 조회로 이동하도록 수정

```
<tr>
  <th scope="row"><c:out value="${dto.tno}"/></th>
  <td><a href="/todo/read?tno=${dto.tno}" class="text-decoration-none"><c:out value="${dto.title}"/></a></td>
  <td><c:out value="${dto.writer}"/></td>
  <td><c:out value="${dto.dueDate}"/></td>
  <td><c:out value="${dto.finished}"/></td>
</tr>
```



Todo의 삭제 기능 개발

- 수정 화면에서 POST방식을 통해서 삭제 처리

/todo/register(GET)

Navbar

Featured

Title Title

DueDate 년-월-일

Writer Writer

Submit Reset

Footer

/todo/list(GET)

Navbar

Featured

Special title treatment

Tno	Title	Writer	DueDate	Finished
3	등록 기능 테스트	user01	2021-12-31	false
2	Test.....	user1	2021-12-23	false
1	스프링 테스트	user00	2022-10-10	false

Footer

/todo/read?tno=xxx(GET)

Navbar

Featured

TNO 2

Title Test.....

DueDate 2021-12-23

Writer user1

Finished

Modify List

Footer

/todo/modify?tno=xxx(GET)

Navbar

Featured

TNO 2

Title Test.....

DueDate 2021-12-23

Writer user1

☐ Finished

Remove Modify List

Footer

/todo/remove
(POST)
tno=xxx

/todo/modify
(POST)
tno=xxx&title
=XXX....

수정/삭제 화면

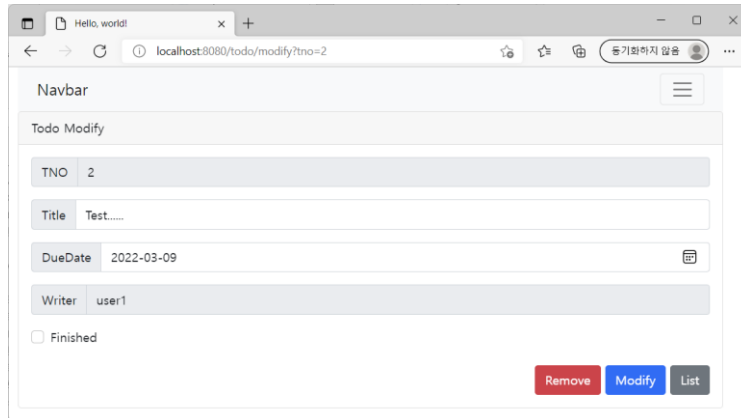
- controller
 - exception
 - formatter
 - SampleController
 - TodoController

- WEB-INF
 - views
 - todo
 - list.jsp
 - modify.jsp
 - read.jsp
 - register.jsp

```
@GetMapping("/{/read", "/modify"})
public void read(Long tno, Model model){
```

```
    TodoDTO todoDTO = todoService.getOne(tno);
    log.info(todoDTO);
    model.addAttribute("dto", todoDTO );
```

```
}
```



```
<div class="card-body">
  <form action="/todo/modify" method="post">
    <div class="input-group mb-3">
      <span class="input-group-text">TNO</span>
      <input type="text" name="tno" class="form-control"
        value=<c:out value="${dto.tno}"></c:out> readonly>
    </div>
    <div class="input-group mb-3">
      <span class="input-group-text">Title</span>
      <input type="text" name="title" class="form-control"
        value='<c:out value="${dto.title}"></c:out>' readonly>
    </div>
    <div class="input-group mb-3">
      <span class="input-group-text">DueDate</span>
      <input type="date" name="dueDate" class="form-control"
        value=<c:out value="${dto.dueDate}"></c:out> >
    </div>
    <div class="input-group mb-3">
      <span class="input-group-text">Writer</span>
      <input type="text" name="writer" class="form-control"
        value=<c:out value="${dto.writer}"></c:out> readonly>
    </div>
    <div class="form-check">
      <label class="form-check-label">
        Finished &nbsp;  
      </label>
      <input class="form-check-input" type="checkbox" name="finished"
        ${dto.finished?"checked":""} >
    </div>
    <div class="my-4">
      <div class="float-end">
        <button type="button" class="btn btn-danger">Remove</button>
        <button type="button" class="btn btn-primary">Modify</button>
        <button type="button" class="btn btn-secondary">List</button>
      </div>
    </div>
  </form>
</div>
```

Remove버튼의 처리

```
</form>
</div>

<script>

    const formObj = document.querySelector("form")

    document.querySelector(".btn-danger").addEventListener("click", function(e) {

        e.preventDefault()
        e.stopPropagation()

        formObj.action = "/todo/remove"
        formObj.method = "post"

        formObj.submit()

    }, false);

</script>
```

```
@PostMapping("/remove")
public String remove(Long tno, RedirectAttributes redirectAttributes){

    log.info("-----remove-----");
    log.info("tno: " + tno);
    return "redirect:/todo/list";
}
```

```
INFO [org.zerock.springex.controller.TODOController] -----remove-----
INFO [org.zerock.springex.controller.TODOController] tno: 2
INFO [org.zerock.springex.controller.TODOController] todo list.....
```

삭제 – TodoMapper/TodoService/TodoController

```
public interface TodoMapper {  
  
    String getTime();  
    void insert(TodoVO todoVO);  
    List<TodoVO> selectAll();  
    TodoVO selectOne(Long tno);  
    void delete(Long tno);  
}
```

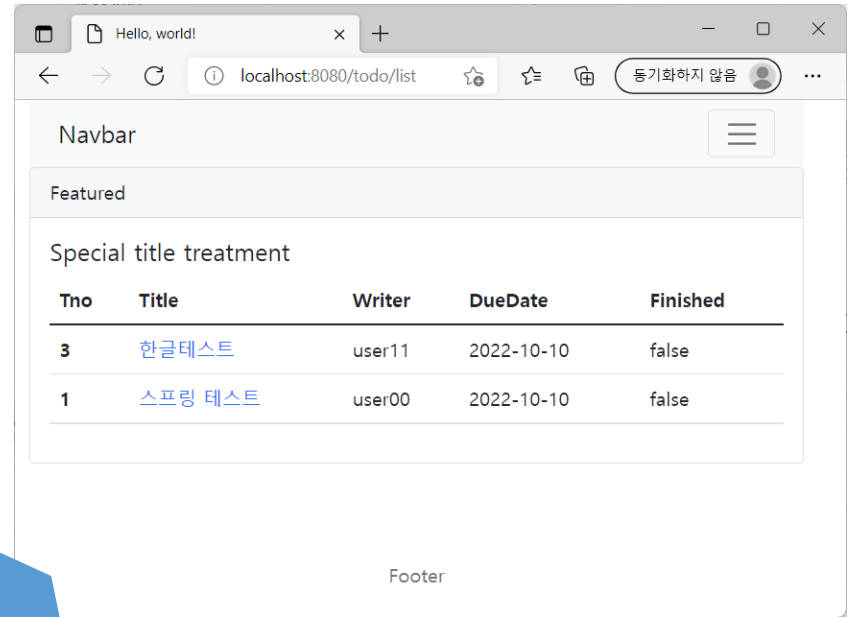
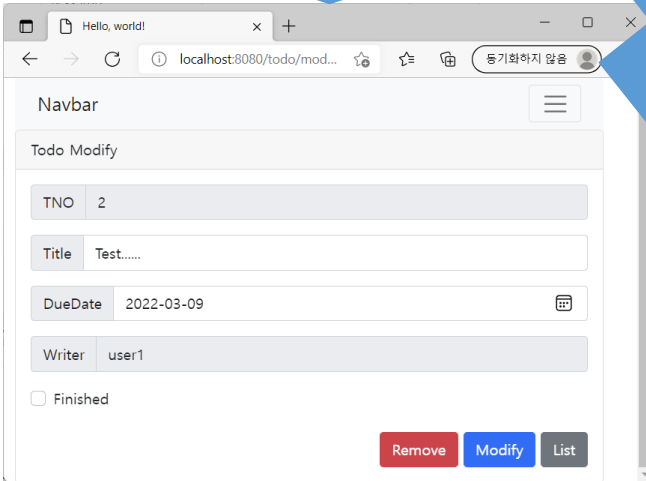
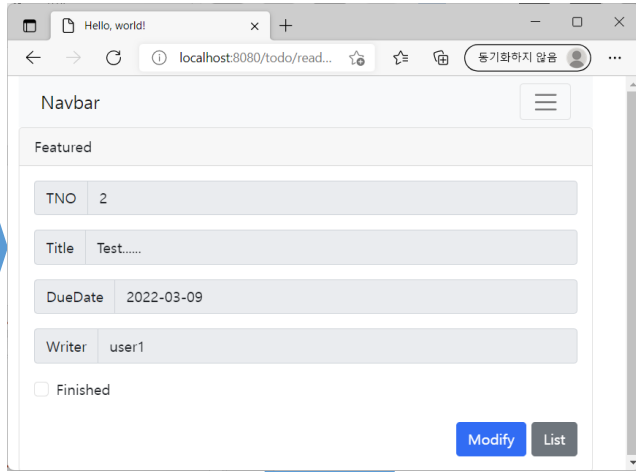
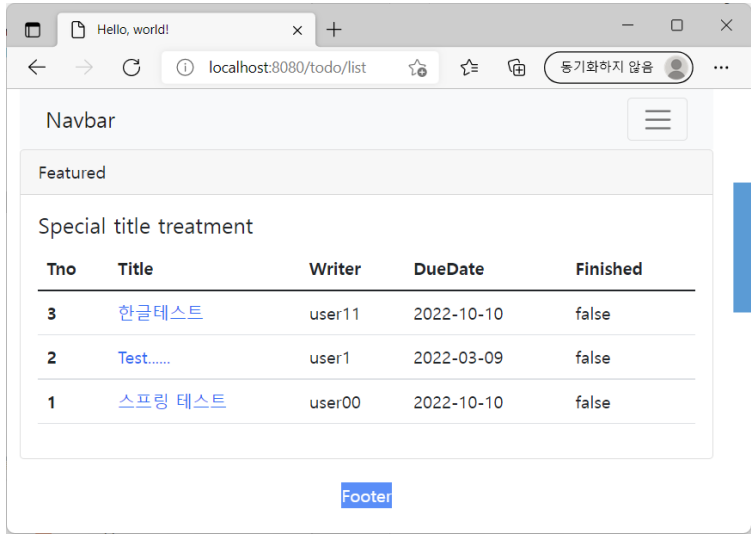
```
<mapper namespace="org.zerock.springex.mapper.TodoMapper">  
  
    ...  
  
    <delete id="delete">  
        delete from tbl_todo where tno = #{tno}  
    </delete>  
</mapper>
```

```
public interface TodoService {  
  
    void register(TodoDTO todoDTO);  
  
    List<TodoDTO> getAll();  
  
    TodoDTO getOne(Long tno);  
  
    void remove(Long tno);  
}
```

```
public class TodoServiceImpl implements TodoService{  
  
    ...  
    @Override  
    public void remove(Long tno) {  
  
        todoMapper.delete(tno);  
  
    }  
}
```

TodoController

```
@PostMapping("/remove")  
public String remove(Long tno, RedirectAttributes redirectAttributes){  
  
    log.info("-----remove-----");  
    log.info("tno: " + tno);  
  
    todoService.remove(tno);  
  
    return "redirect:/todo/list";  
}
```



Todo 수정 기능 개발

TodoMapper

```
public interface TodoMapper {  
  
    String getTime();  
    void insert(TodoVO todoVO);  
    List<TodoVO> selectAll();  
    TodoVO selectOne(Long tno);  
    void delete(Long tno);  
    void update(TodoVO todoVO);  
}
```

```
<update id="update">  
    update tbl_todo set title = #{title} , dueDate = #{dueDate},  
    finished= #{finished} where tno = #{tno}  
</update>
```

TodoService/TodoServiceImpl

```
public interface TodoService {  
  
    void register(TodoDTO todoDTO);  
    List<TodoDTO> getAll();  
    TodoDTO getOne(Long tno);  
    void remove(Long tno);  
    void modify(TodoDTO todoDTO);  
}
```

```
@Override  
public void modify(TodoDTO todoDTO) {  
  
    TodoVO todoVO = modelMapper.map(todoDTO, TodoVO.class );  
    todoMapper.update(todoVO);  
}
```

checkbox를 위한 Formatter

```
package org.zerock.springex.controller.formatter;
import org.springframework.format.Formatter;

import java.text.ParseException;
import java.util.Locale;

public class CheckboxFormatter implements Formatter<Boolean> {

    @Override
    public Boolean parse(String text, Locale locale) throws ParseException {
        if(text == null ) {
            return false;
        }
        return text.equals("on");
    }

    @Override
    public String print(Boolean object, Locale locale) {
        return object.toString();
    }
}
```

```
<bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
    <property name="formatters">
        <set>
            <bean class="org.zerock.springex.controller.formatter.LocalDateFormatter"/>
            <bean class="org.zerock.springex.controller.formatter.CheckboxFormatter"/>
        </set>
    </property>
</bean>
```

- controller
 - exception
 - formatter
 - CheckboxFormatter
 - LocalDateFormatter
 - SampleController
 - TodoController

- WEB-INF
 - views
 - root-context.xml
 - servlet-context.xml

TodoController

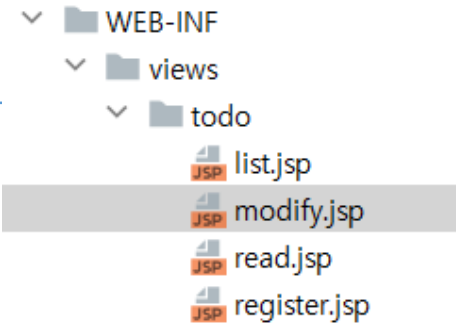
```
@PostMapping("/modify")
public String modify(@Valid TodoDTO todoDTO,
                    BindingResult bindingResult,
                    RedirectAttributes redirectAttributes){

    if(bindingResult.hasErrors()) {
        log.info("has errors.....");
        redirectAttributes.addFlashAttribute("errors", bindingResult.getAllErrors() );
        redirectAttributes.addAttribute("tno", todoDTO.getTno() );
        return "redirect:/todo/modify";
    }

    log.info(todoDTO);

    todoService.modify(todoDTO);

    return "redirect:/todo/list";
}
```



```
<script>
```

```
const serverValidResult = {}
```

```
<c:forEach items="${errors}" var="error">
serverValidResult[${error.getField()}] = '${error.defaultMessage}'
</c:forEach>
```

```
console.log(serverValidResult)
```

```
<script>
```

```
const formObj = document.querySelector("form")
document.querySelector(".btn-danger").addEventListener("click",function(e) {

    ...

},false);

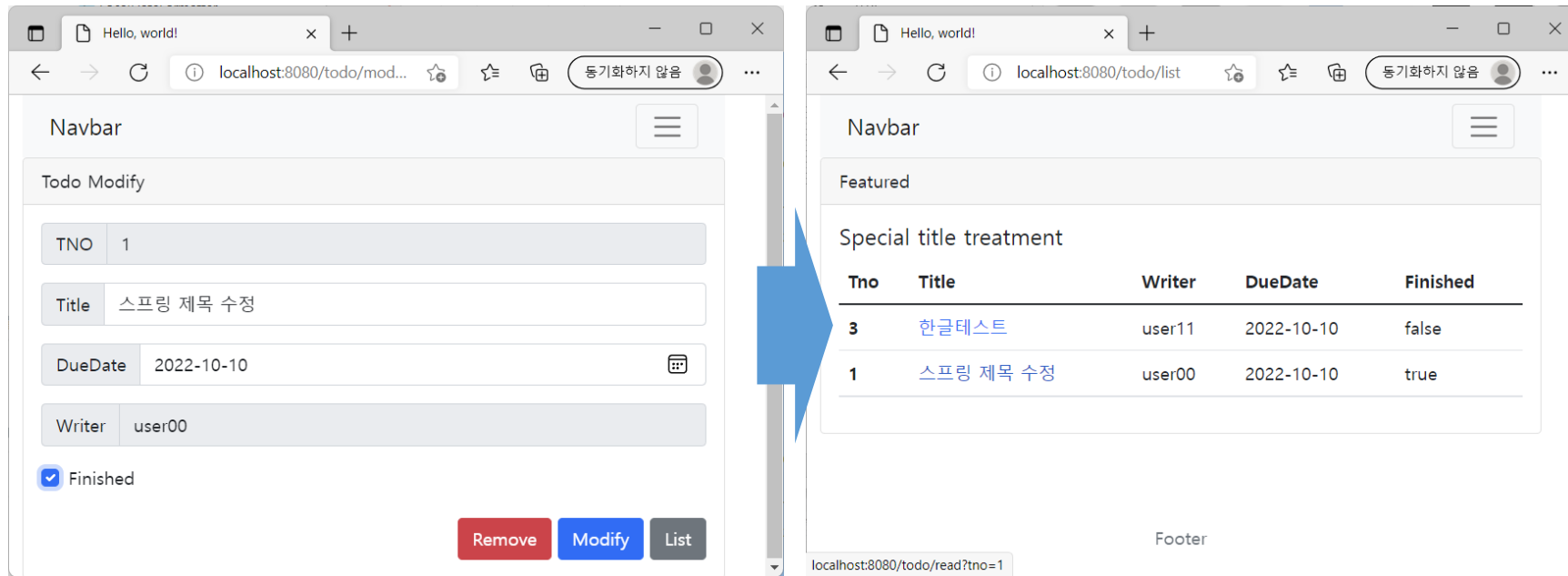
document.querySelector(".btn-primary").addEventListener("click",function(e) {

    e.preventDefault()
    e.stopPropagation()

    formObj.action = "/todo/modify"
    formObj.method = "post"
    formObj.submit()

},false);
```

```
</script>
```

```
<script>  
    const serverValidResult = {}  
  
    serverValidResult['dueDate'] = '미래 날짜여야 합니다'  
  
    console.log(serverValidResult)  
</script>
```

페이징처리를 위한 TodoMapper

- MariaDB/MySQL에서는 limit를 이용해서 페이징 쿼리 작성

```
select * from tbl_todo order by tno desc limit 10;
```

가져오는 데이터의 수(fetch)

```
select * from tbl_todo order by tno desc limit 10, 10;
```

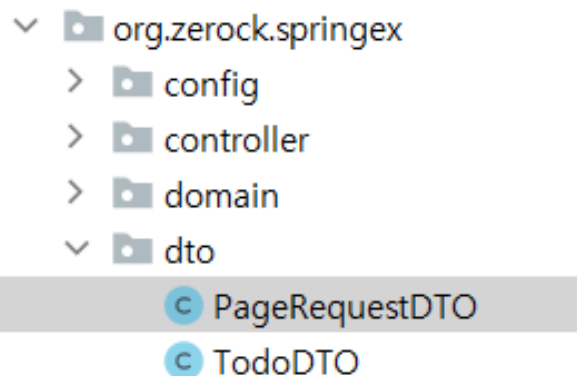
건너뛰는 데이터의 수(skip)

가져오는 데이터의 수(fetch)

limit기능은 뒤에 값만을 사용할 수 있고 식(expression)을 사용할 수 없다는 단점

페이지 처리를 위한 DTO

- 현재 페이지 번호(page)
- 한 페이지당 데이터의 수(size)



```
package org.zerock.springex.dto;

import lombok.*;

import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.Positive;

@Builder
@Data
@AllArgsConstructor
@NoArgsConstructor
public class PageRequestDTO {

    @Builder.Default
    @Min(value = 1)
    @Positive
    private int page = 1;

    @Builder.Default
    @Min(value = 10)
    @Max(value = 100)
    @Positive
    private int size = 10;

    public int getSkip(){

        return (page - 1) * 10;
    }
}
```

TodoMapper의 목록/count 처리

```
public interface TodoMapper {  
  
    ...  
  
    List<TodoVO> selectList(PageRequestDTO pageRequestDTO);  
}
```

```
<select id="selectList" resultType="org.zerock.springex.domain.TODOVO">  
    select * from tbl_todo order by tno desc limit #{skip}, #{size}  
</select>
```

```
public interface TodoMapper {  
  
    String getTime();  
  
    ...  
  
    List<TodoVO> selectList(PageRequestDTO pageRequestDTO);  
  
    int getCount(PageRequestDTO pageRequestDTO);  
}
```

```
<select id="getCount" resultType="int">  
    select count(tno) from tbl_todo  
</select>
```

목록 데이터를 위한 DTO/서비스 계층

- 목록 화면에서 필요한 데이터를 하나의 DTO객체로 만들어서 사용하면 나중에 재사용이 가능
 - TodoDTO의 목록
 - 전체 데이터의 수
 - 페이지 번호의 처리를 위한 데이터들 (시작 페이지 번호/ 끝 페이지 번호...)

> domain

dto

PageRequestDTO

PageResponseDTO

TodoDTO

```
package org.zerock.springex.dto;

import java.util.List;
public class PageResponseDTO<E> {

    private int page;
    private int size;
    private int total;
    //시작 페이지 번호
    private int start;
    //끝 페이지 번호
    private int end;
    //이전 페이지의 존재 여부
    private boolean prev;
    //다음 페이지의 존재 여부
    private boolean next;
    private List<E> dtoList;

}
```

PageResponseDTO의 생성자

```
@Builder(builderMethodName = "withAll")
public PageResponseDTO(PageRequestDTO
pageRequestDTO, List<E> dtoList, int total){

    this.page = pageRequestDTO.getPage();
    this.size = pageRequestDTO.getSize();
    this.total = total;
    this.dtoList = dtoList;

}
```

페이지 번호의 계산

- page가 1인 경우: 시작 페이지(start)는 1, 마지막 페이지(end)는 10
- page가 10인 경우: 시작 페이지(start)는 1, 마지막 페이지(end)는 10
- page가 11인 경우: 시작 페이지(start)는 11, 마지막 페이지(end)는 20

시작 페이지 번호/마지막 페이지 번호 계산

```
this.end = (int)(Math.ceil(this.page / 10.0 )) * 10;
```

page를 10으로 나눈 값을 올림 처리 한 후 * 10

1 / 10 => 0.1 => 1 => 10

11 / 10 => 1.1 => 2 => 20

10 / 10 => 1.0 => 1 => 10

```
this.end = (int)(Math.ceil(this.page / 10.0 )) * 10;
```

```
this.start = this.end - 9;
```

```
int last = (int)(Math.ceil((total/((double)size))));
```

이전/다음 페이지 계산

```
this.prev = this.start > 1;
```

```
this.next = total > this.end * this.size;
```

```
@Getter
@ToString
public class PageResponseDTO<E> {

    private int page;
    private int size;
    private int total;

    //시작 페이지 번호
    private int start;
    //끝 페이지 번호
    private int end;
    //이전 페이지의 존재 여부
    private boolean prev;
    //다음 페이지의 존재 여부
    private boolean next;
    private List<E> dtoList;

    @Builder(builderMethodName = "withAll")
    public PageResponseDTO(PageRequestDTO pageRequestDTO, List<E> dtoList, int total){

        this.page = pageRequestDTO.getPage();
        this.size = pageRequestDTO.getSize();

        this.total = total;
        this.dtoList = dtoList;
        this.end = (int) (Math.ceil(this.page / 10.0 )) * 10;
        this.start = this.end - 9;
        int last = (int) (Math.ceil((total/(double) size)));
        this.end = end > last ? last: end;
        this.prev = this.start > 1;
        this.next = total > this.end * this.size;

    }
}
```

TodoService/TodoServiceImpl에서의 목록 처리

```
package org.zerock.springex.service;

import org.zerock.springex.dto.PageRequestDTO;
import org.zerock.springex.dto.PageResponseDTO;
import org.zerock.springex.dto.TODO;

public interface TodoService {

    void register(TODO todo);

    //List<TODO> getAll();

    PageResponseDTO<TODO> getList(PageRequestDTO
pageRequestDTO);

    TODO getOne(Long tno);

    void remove(Long tno);

    void modify(TODO todo);
}
```

```
@Override
public PageResponseDTO<TODO> getList(PageRequestDTO pageRequestDTO) {

    List<TODOVO> voList = todoMapper.selectList(pageRequestDTO);
    List<TODO> dtoList = voList.stream()
        .map(vo -> modelMapper.map(vo, TODO.class))
        .collect(Collectors.toList());

    int total = todoMapper.getCount(pageRequestDTO);

    PageResponseDTO<TODO> pageResponseDTO = PageResponseDTO.<TODO>
O> withAll()
        .dtoList(dtoList)
        .total(total)
        .pageRequestDTO(pageRequestDTO)
        .build();

    return pageResponseDTO;
}
```


TodoController와 JSP처리

- controller
 - exception
 - formatter
 - SampleController
 - TodoController

```
@GetMapping("/list")
public void list(@Valid PageRequestDTO pageRequestDTO, BindingResult bindingResult, Model model){

    log.info(pageRequestDTO);

    if(bindingResult.hasErrors()){
        pageRequestDTO = PageRequestDTO.builder().build();
    }
    model.addAttribute("responseDTO", todoService.getList(pageRequestDTO));
}
```

- WEB-INF
 - views
 - todo
 - list.jsp
 - modify.jsp

```
<c:forEach items="${responseDTO.dtoList}" var="dto">
<tr>
<th scope="row"><c:out value="${dto.tno}"/></th>
<td><a href="/todo/read?tno=${dto.tno}" class="text-decoration-none"><c:out value="${dto.title}"/></a></td>
<td><c:out value="${dto.writer}"/></td>
<td><c:out value="${dto.dueDate}"/></td>
<td><c:out value="${dto.finished}"/></td>
</tr>
</c:forEach>
```

페이지 번호 출력과 이동

부트스트랩의 pagination 컴포넌트 활용

Disabled and active states

Pagination links are customizable for different circumstances. Use `.disabled` for links that appear un-clickable and `.active` to indicate the current page.

While the `.disabled` class uses `pointer-events: none` to try to disable the link functionality of `<a>`s, that CSS property is not yet standardized and doesn't account for keyboard navigation. As such, you should always add `tabindex="-1"` on disabled links and use custom JavaScript to fully disable their functionality.

Previous 1 2 3 Next

```
<div class="float-end">
  <ul class="pagination flex-wrap">
    <c:forEach begin="${responseDTO.start}" end="${responseDTO.end}" var="num">
      <li class="page-item"><a class="page-link" href="#">${num}</a></li>
    </c:forEach>
  </ul>
</div>
```

http://localhost:8080/todo/list

Featured				
Special title treatment				
Tno	Title	Writer	DueDate	Finished
1527	한글테스트	user11	2022-10-10	false
1526	스프링 제목 수정	user00	2022-10-10	false
1525	한글테스트	user11	2022-10-10	false
1524	스프링 제목 수정	user00	2022-10-10	false
1523	한글테스트	user11	2022-10-10	false
1522	스프링 제목 수정	user00	2022-10-10	false
1521	한글테스트	user11	2022-10-10	false
1520	스프링 제목 수정	user00	2022-10-10	false
1519	한글테스트	user11	2022-10-10	false
1518	스프링 제목 수정	user00	2022-10-10	false
1 2 3 4 5 6 7 8 9 10				

http://localhost:8080/todo/list?page=12

Featured				
Special title treatment				
Tno	Title	Writer	DueDate	Finished
1417	한글테스트	user11	2022-10-10	false
1416	스프링 제목 수정	user00	2022-10-10	false
1415	한글테스트	user11	2022-10-10	false
1414	스프링 제목 수정	user00	2022-10-10	false
1413	한글테스트	user11	2022-10-10	false
1412	스프링 제목 수정	user00	2022-10-10	false
1411	한글테스트	user11	2022-10-10	false
1410	스프링 제목 수정	user00	2022-10-10	false
1409	한글테스트	user11	2022-10-10	false
1408	스프링 제목 수정	user00	2022-10-10	false
11 12 13 14 15 16 17 18 19 20				

페이지 번호의 클릭 이벤트 처리를 위한 'data-num' 속성 추가

<pre><ul class="pagination flex-wrap"> <c:if test="\${responseDTO.prev}"> <li class="page-item"> Previous </c:if> <c:forEach begin="\${responseDTO.start}" end="\${responseDTO.end}" var="num"> <li class="page-item \${responseDTO.page == num? "active":""}"> \${num} </c:forEach> <c:if test="\${responseDTO.next}"> <li class="page-item"> Next </c:if> </pre>	<pre><script> document.querySelector(".pagination").addEventListener("click", function (e) { e.preventDefault() e.stopPropagation() const target = e.target if(target.tagName !== 'A') { return } const num = target.getAttribute("data-num") self.location = `/todo/list?page=\${ \${num} }` //백틱(` `)을 이용해서 템플릿 처리 },false) </script></pre>
--	--

목록에서 조회 페이지 이동

- 조회 페이지에서 다시 목록으로 돌아올 수 있도록 현재 페이지를 같이 쿼리스트링으로 유지할 필요가 있음
- PageResponseDTO를 이용해서 링크를 생성하는 기능을 미리 구성

```
private String link;

public int getSkip(){
    return (page - 1) * 10;
}

public String getLink() {
    if(link == null){
        StringBuilder builder = new StringBuilder();

        builder.append("page=" + this.page);

        builder.append("&size=" + this.size);
        link = builder.toString();
    }
    return link;
}
```

```
<c:forEach items="${responseDTO.dtoList}" var="dto">
    <tr>
        <th scope="row"><c:out value="${dto.tno}"/></th>
        <td>
            <a href="/todo/read?tno=${dto.tno}&${pageRequestDTO.link}" class="text-decoration-none" data-tno="${dto.tno}" >
                <c:out value="${dto.title}"/>
            </a>
        </td>
        <td><c:out value="${dto.writer}"/></td>
        <td><c:out value="${dto.dueDate}"/></td>
        <td><c:out value="${dto.finished}"/></td>
    </tr>
</c:forEach>
```

```
<tr>
  <th scope="row">1385</th>
  <td>
    <a href="/todo/read?tno=1385&page=15&size=10" class="text-decoration-none" data-tno="1385">...</a>
  </td>
  <td>user11</td>
  <td>2022-10-10</td>
  <td>false</td>
</tr>
<tr>
  <th scope="row">1384</th>
  <td>
    <a href="/todo/read?tno=1384&page=15&size=10" class="text-decoration-none" data-tno="1384">...</a>
  </td>
  <td>user00</td>
```

페이지 정보 유지를 위한 수정

- 조회 페이지 역시 추가적으로 PageRequestDTO를 이용하도록 수정
- 수정/삭제의 경우에도 페이지 정보 유지가 필요하다면 파라미터로 PageRequestDTO를 지정

검색/필터링 조건의 정의

완료여부와 기간은 AND 필터링

'제목/작성자' 는 OR 검색

Search

☒완료여부

☒제목 ☐작성자

년-월-일 년-월-일

Search Clear

- 제목(title)과 작성자(writer)는 키워드(keyword)를 이용하는 검색 처리
- 완료여부를 필터링 처리
- 특정한 기간을 지정 (from, to) 필터링 처리

검색과 필터링에 필요한 데이터는 다음과 같이 구분

- 완료 여부에 사용되는 boolean 타입 (finished)
- 제목, 작성자 검색에 사용하는 문자열(keyword)
- 특정 기간 검색을 위한 LocalDate 변수 2개 (from, to)

- > config
- > controller
- > domain
- ✓ dto

- PageRequestDTO
- PageResponseDTO
- TodoDTO

```
public class PageRequestDTO {
```

```
...
```

```
private String[] types;
```

```
private String keyword;
```

```
private boolean finished;
```

```
private LocalDate from;
```

```
private LocalDate to;
```

```
...
```

```
}
```

types에 따른 동적 쿼리

- MyBatis의 동적 쿼리(dynamic query)기능을 이용해서 상황에 따라서 다른 코드를 만들어 낼 수 있음
 - if
 - choose(when, otherwise)
 - trim(where, set)
 - foreach

```
<select id="selectList" resultType="org.zerock.springex.domain.TODOVO">
  select * from tbl_todo
  <foreach collection="types" item="type">
    #{type}
  </foreach>
  order by tno desc limit #{skip}, #{size}
</select>
```

```
<select id="selectList" resultType="org.zerock.springex.domain.TODOVO">
  select * from tbl_todo
  <foreach collection="types" item="type">
    <if test="type == 't.toString()'">
      title like concat('%', #{keyword}, '%')
    </if>
    <if test="type == 'w.toString()'">
      writer like concat('%', #{keyword}, '%')
    </if>
  </foreach>
  order by tno desc limit #{skip}, #{size}
</select>
```

```
<foreach collection="types" item="type" open="(" close=")" separator=" OR ">
  <if test="type == 't.toString()'">
    title like concat('%', #{keyword}, '%')
  </if>
  <if test="type == 'w.toString()'">
    writer like concat('%', #{keyword}, '%')
  </if>
</foreach>
```


<where> 처리

```
<select id="selectList" resultType="org.zerock.springex.domain.TODOVO">
  select * from tbl_todo
  <where>
    <if test="types != null and types.length > 0">
      <foreach collection="types" item="type" open="(" close=")" separator=" OR ">
        <if test="type == 't'.toString()">
          title like concat('%', #{keyword}, '%')
        </if>
        <if test="type == 'w'.toString()">
          writer like concat('%', #{keyword}, '%')
        </if>
      </foreach>
    </if>
  </where>
  order by tno desc limit #{skip}, #{size}
</select>
```

<trim>과 완료 여부/완료일 필터링

```
<where>
  <if test="types != null and types.length > 0">
    <foreach collection="types" item="type" open="(" close=")" separator=" OR ">
      <if test="type == 't'.toString()">
        title like concat('%', #{keyword}, '%')
      </if>
      <if test="type == 'w'.toString()">
        writer like concat('%', #{keyword}, '%')
      </if>
    </foreach>
  </if>

  <if test="finished">
    <trim prefix="and">
      finished = 1
    </trim>
  </if>
</where>
```

```
<where>
  <if test="types != null and types.length > 0">
    <foreach collection="types" item="type" open="(" close=")" separator=" OR ">
      <if test="type == 't'.toString()">
        title like concat('%', #{keyword}, '%')
      </if>
      <if test="type == 'w'.toString()">
        writer like concat('%', #{keyword}, '%')
      </if>
    </foreach>
  </if>

  <if test="finished">
    <trim prefix="and">
      finished = 1
    </trim>
  </if>

  <if test="from != null and to != null">
    <trim prefix="and">
      dueDate between #{from} and #{to}
    </trim>
  </if>

</where>
```

검색 조건을 위한 화면처리

<!--추가하는 코드-->

```
<div class="row content">
  <div class="col">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Search </h5>
        <form action="/todo/list" method="get">
          <input type="hidden" name="size" value="${pageRequestDTO.size}">
          <div class="mb-3">
            <input type="checkbox" name="finished" >완료 여부
          </div>
          <div class="mb-3">
            <input type="checkbox" name="types" value="t">제목
            <input type="checkbox" name="types" value="w">작성자
            <input type="text" name="keyword" class="form-control" >
          </div>
          <div class="input-group mb-3 dueDateDiv">
            <input type="date" name="from" class="form-control">
            <input type="date" name="to" class="form-control">
          </div>
          <div class="input-group mb-3">
            <div class="float-end">
              <button class="btn btn-primary" type="submit">Search</button>
              <button class="btn btn-info" type="reset">Clear</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
```

The screenshot shows a web browser window with the URL `localhost:8080/todo/list?page=4&size=10`. The page has a "Navbar" section with a search bar. Below the search bar, there are checkboxes for "완료 여부" (Completed) and "제목" (Title), and a text input for "keyword". There are also date pickers for "from" and "to" dates. Below the search bar, there are two buttons: "Search" and "Clear".

Below the search bar, there is a "Featured" section with a table titled "Special title treatment". The table has columns: "Tno", "Title", "Writer", "DueDate", and "Finished".

Tno	Title	Writer	DueDate	Finished
1497	한글테스트	user11	2022-10-10	false
1496	스프링 제목 수정	user00	2022-10-10	false
1495	한글테스트	user11	2022-10-10	false
1494	스프링 제목 수정	user00	2022-10-10	false
1493	한글테스트	user11	2022-10-10	false
1492	스프링 제목 수정	user00	2022-10-10	false
1491	한글테스트	user11	2022-10-10	false
1490	스프링 제목 수정	user00	2022-10-10	false
1489	한글테스트	user11	2022-10-10	false
1488	스프링 1488	user00	2022-10-10	true

Below the table, there is a pagination bar with buttons for "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", and "Next". The "4" button is highlighted.

At the bottom of the page, there is a "Footer" section.

Search

☒완료여부

☒제목 ☒작성자

테스트

2022-03-01 2022-12-30

Search Clear

PageRequestDTO로 수집된 결과

INFO [org.zerock.springex.controller.TODOController]
PageRequestDTO(page=1, size=10, link=page=1&size=10, types=[t, w], keyword=테스트, finished=true, from=2022-03-01, to=2022-12-30)

실행되는 SQL

select * from tbl_todo WHERE (title like concat('%', ?, '%') OR writer like concat('%', ?, '%')) and finished = 1 and dueDate between ? and ? order by tno desc limit ?, ?

select count(tno) from tbl_todo WHERE (title like concat('%', ?, '%') OR writer like concat('%', ?, '%'))) and finished = 1 and dueDate between ? and ?