



YouTube

NAVER 카페

구멍가게 코딩단

코드로 배우는 리액트

3. 스프링 부트와 API 서버

3장. 스프링 부트 RESTful

스프링 부트 RESTful

- 최신 애플리케이션 개발 추세: 프론트엔드와 백엔드 분리, API 서버로 데이터 전달.
- API 서버 역할: 데이터 전달에 중점, 화면 구성은 아님.
- 다양한 전송 방식과 데이터 형식 활용: GET/POST/PUT/DELETE/OPTIONS, XML/JSON 등.
- RESTful 서비스 구성: 다양한 방식으로 특정 자원 처리, 스프링 부트를 활용한 예시.

개발목표

1. Spring Data JPA를 이용한 데이터 처리
2. DTO와 VO의 처리
3. 스프링 부트 RESTful 서비스 개발

Index

3.1 프로젝트 설정

3.2 TodoRepository 테스트

3.3 서비스 계층과 DTO 처리

3.4 서비스 계층의 구현

3.5 목록 처리와 DTO

3.6 @RestControllerAdvice

3.7 REST관련 툴을 이용한 POST/PUT/DELETE

프로젝트 설정

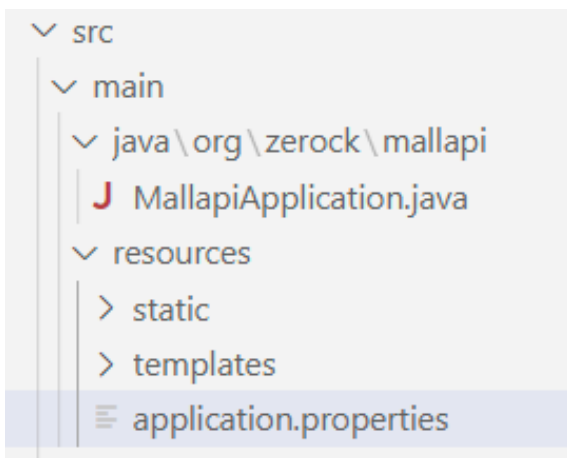
- VSCode의 STS(Spring Tool Suites) 플러그인으로 프로젝트 생성
- 스프링 부트는 <https://start.spring.io/> 웹 사이트를 통해 프로젝트 생성 후 다운로드
 - 개발 도구에서 스프링 부트 지원이 없을 때 유용하게 활용됨

The screenshot displays the Spring Initializr web interface. At the top, there's a green banner with the text "Meet the Spring team this August at SpringOne." Below this, the "spring initializr" logo is visible. The main content area is divided into several sections:

- Project:** Includes radio buttons for "Gradle - Groovy" (selected), "Gradle - Kotlin", and "Maven".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for "3.1.1 (SNAPSHOT)", "3.1.0" (selected), "3.0.8 (SNAPSHOT)", and "3.0.7". Below these are "2.7.13 (SNAPSHOT)" and "2.7.12".
- Project Metadata:** A form with fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo). There's also a "Packaging" section with "Jar" (selected) and "War" options.
- Dependencies:** A section with a button "ADD DEPENDENCIES... CTRL + B". It lists several dependencies: "Lombok" (DEVELOPER TOOLS), "Spring Boot DevTools" (DEVELOPER TOOLS), and "Spring Reactive Web" (WEB).

프로젝트 설정

→ 프로젝트의 src/main/resources 폴더에 있는 application.properties

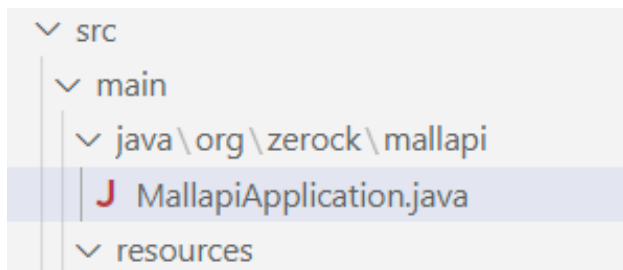


```
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver  
spring.datasource.url=jdbc:mariadb://localhost:3306/malldb  
spring.datasource.username=malldbuser  
spring.datasource.password=malldbuser
```

```
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.properties.hibernate.format_sql=true  
spring.jpa.show-sql=true
```

프로젝트 설정

→ 프로젝트 생성시에 자동으로 만들어진 'xxxApplication.java'를 실행해서 확인



```
@SpringBootApplication
public class MallapiApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(primarySource:MallapiApplication.class, args);
    }

}
```

Spring Data JPA

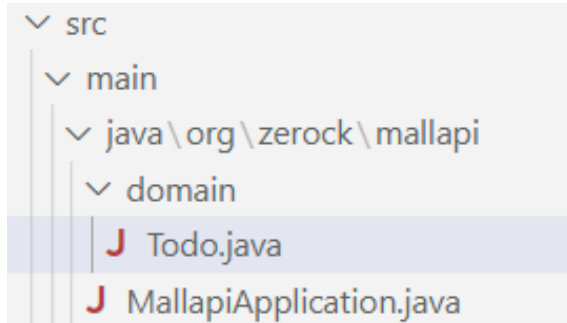
→ application.properties 확인

```
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.properties.hibernate.format_sql=true  
spring.jpa.show-sql=true
```

- **spring.jpa.hibernate.ddl-auto** : SQL의 DDL(테이블 생성이나 객체 생성시 사용)기능
속성값 : create / create-drop / update / validate / none
- **spring.jpa.properties.hibernate.format_sql=true** : SQL들의 포매팅(줄 바꿈) 설정
- **spring.jpa.show-sql=true** : 실행과정에서 만들어지는 SQL의 출력 여부 설정

엔티티 클래스 작성

→ domain 패키지 생성 후 Todo 클래스 생성



```
@Entity
@Table(name = "tbl_todo")
@Getter
@ToString
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class Todo {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long tno;

    private String title;
    private String writer;
    private boolean complete;
    private LocalDate dueDate;

}
```


엔티티 클래스 작성

→ 스프링 부트 프로젝트를 실행 후 테이블을 생성하는 D이 실행 확인

Hibernate:

```
create table tbl_todo (  
    tno bigint not null auto_increment,  
    complete bit not null,  
    due_date date,  
    title varchar(255),  
    writer varchar(255),  
    primary key (tno)  
) engine=InnoDB
```

Repository 설정과 테스트

→ repository 패키지를 생성 후 TodoRepository 인터페이스 생성



```
package org.zerock.mallapi.repository;

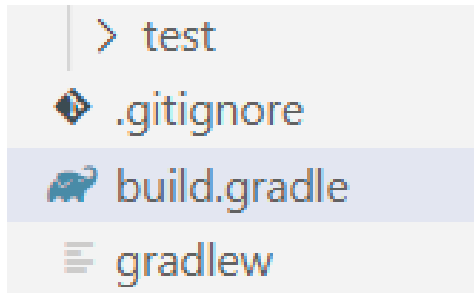
import org.springframework.data.jpa.repository.JpaRepository;
import org.zerock.mallapi.domain.TODO;

public interface TodoRepository
    extends JpaRepository<TODO, Long>{

}
```

TodoRepository 테스트

→ 테스트를 위한 환경 설정

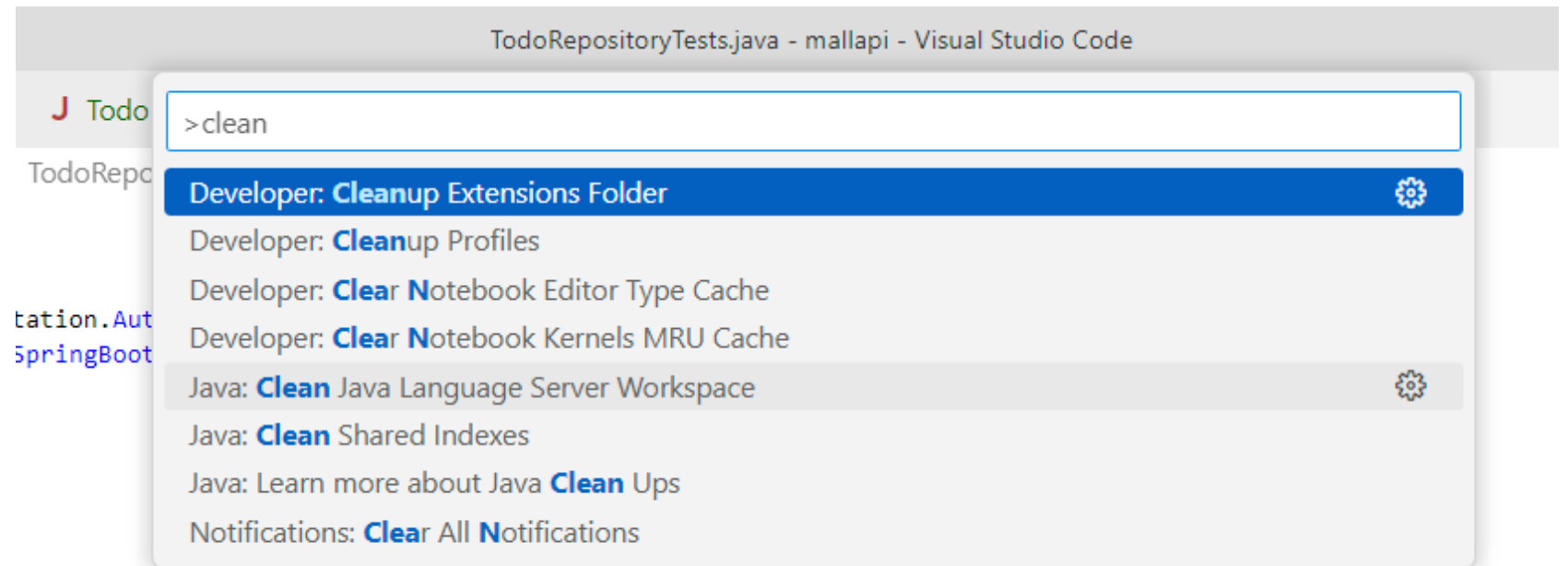


```
dependencies {  
    ...생략  
  
    testCompileOnly 'org.projectlombok:lombok'  
    testAnnotationProcessor 'org.projectlombok:lombok'  
}
```

TodoRepository 테스트

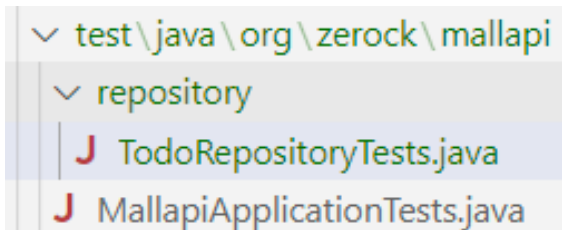
→ 작업영역을 초기화

'View -> Command Palette' 메뉴에서 'clean' 키워드로 검색해서 'Server Workspace' 항목을 선택하고 VSCode 재시작



TodoRepository 테스트

→ Test 폴더에 repository 패키지를 생성하고 TodoRepositoryTests 클래스 생성



```
test\java\org\zerock\mallapi
├── repository
│   ├── TodoRepositoryTests.java
│   └── MallapiApplicationTests.java
```

```
package org.zerock.mallapi.repository;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

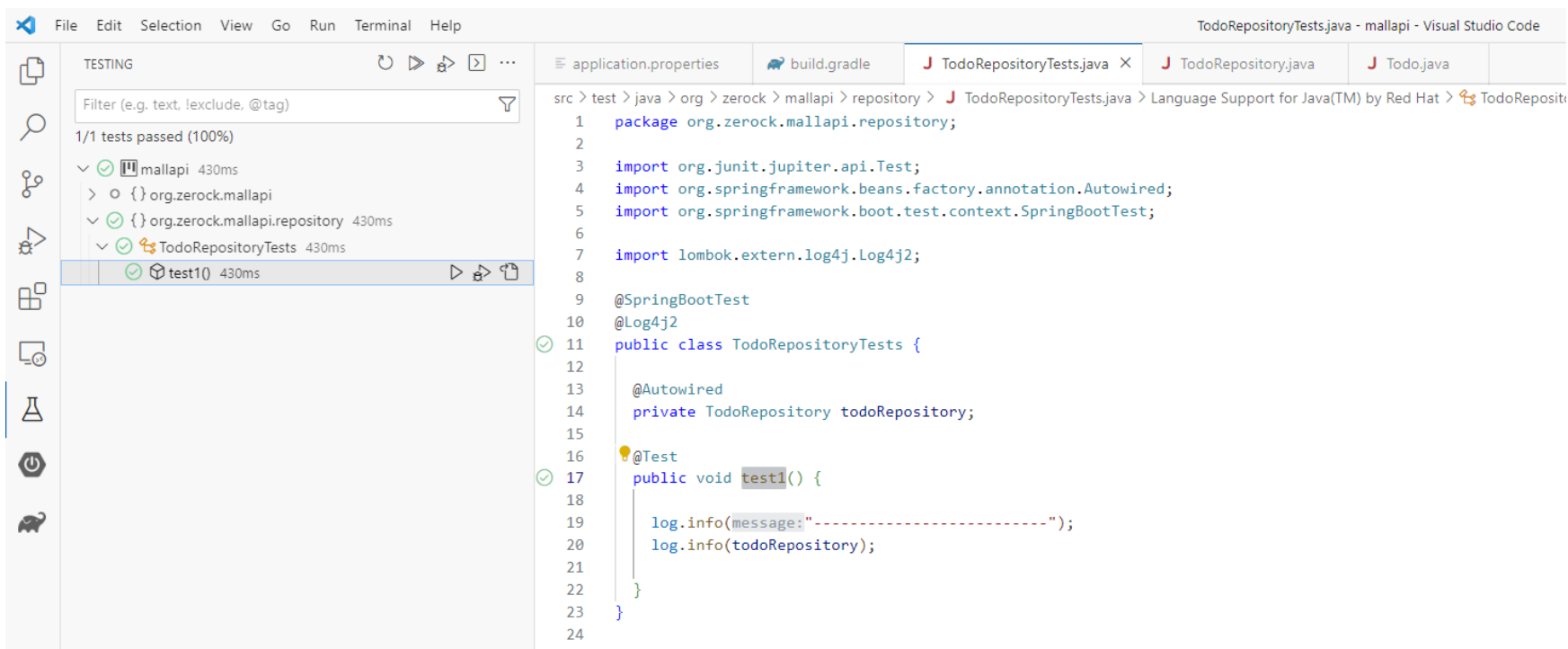
import lombok.extern.log4j.Log4j2;

@SpringBootTest
@Log4j2
public class TodoRepositoryTests {

    @Autowired
    private TodoRepository todoRepository;

    @Test
    public void test1() {
        log.info("-----");
        log.info(todoRepository);
    }
}
```

TodoRepository 테스트



```

o.z.m.repository.TODORepositoryTests : Started TODORepositoryTests in 3.069 seconds (process running for 4.655)
o.z.m.repository.TODORepositoryTests : -----
o.z.m.repository.TODORepositoryTests : org.springframework.data.jpa.repository.support.SimpleJpaRepository@278cbf5a
j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
  
```

데이터 추가

test\java\org\zerock\mallapi
repository
J TodoRepositoryTests.java
J MallapiApplicationTests.java

```
@SpringBootTest
@Log4j2
public class TodoRepositoryTests {

    @Autowired
    private TodoRepository todoRepository;

    @Test
    public void testInsert() {
        for (int i = 1; i <= 100; i++) {
            Todo todo = Todo.builder()
                .title("Title..." + i)
                .dueDate(LocalDate.of(2023, 12, 31))
                .writer("user00")
                .build();
            todoRepository.save(todo);
        }
    }
}
```

```
Hibernate:
insert
into
    tbl_todo
    (complete,due_date,title,writer)
values
    (?, ?, ?, ?)
Hibernate:
insert
into
    tbl_todo
    (complete,due_date,title,writer)
values
    (?, ?, ?, ?)
```

데이터 조회

```
▼ test\java\org\zerock\mallapi
  ▼ repository
    J TodoRepositoryTests.java
    J MallapiApplicationTests.java
```

~~~ 생략

```
@Test
public void testRead() {
    //존재하는 번호로 확인
    Long tno = 33L;

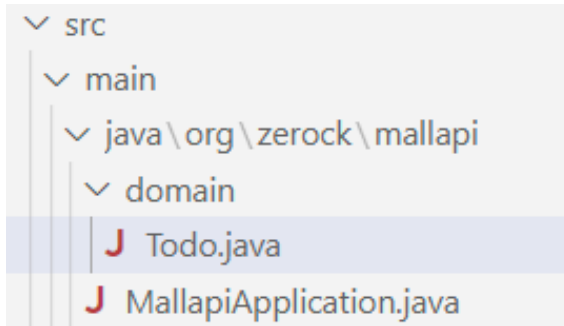
    java.util.Optional<Todo> result = todoRepository.findById(tno);
    Todo todo = result.orElseThrow();
    log.info(todo);
}
```

Hibernate:

```
select
    t1_0.tno,
    t1_0.complete,
    t1_0.due_date,
    t1_0.title,
    t1_0.writer
from
    tbl_todo t1_0
where
    t1_0.tno=?
```



## 데이터 수정



```
public class Todo {

    ...생략

    public void changeTitle(String title){
        this.title = title;
    }

    public void changeComplete(boolean complete){
        this.complete = complete;
    }

    public void changeDueDate(LocalDate dueDate){
        this.dueDate = dueDate;
    }

}
```

## 데이터 수정

```
test\java\org\zerock\mallapi
  repository
    J TodoRepositoryTests.java
    J MallapiApplicationTests.java
```

```
@Test
public void testModify() {
    Long tno = 33L;
    Optional<Todo> result = todoRepository.findById(tno);
    Todo todo = result.orElseThrow();
    todo.changeTitle("Modified 33...");
    todo.changeComplete(true);
    todo.changeDueDate(LocalDate.of(2023,10,10));
    todoRepository.save(todo);
}
```

```
Hibernate:
select
  t1_0.tno,
  t1_0.complete,
  t1_0.due_date,
  t1_0.title,
  t1_0.writer
from
  tbl_todo t1_0
where
  t1_0.tno=?
```



```
Hibernate:
select
  t1_0.tno,
  t1_0.complete,
  t1_0.due_date,
  t1_0.title,
  t1_0.writer
from
  tbl_todo t1_0
where
  t1_0.tno=?
```



```
Hibernate:
update
  tbl_todo
set
  complete=?,
  due_date=?,
  title=?,
  writer=?
where
  tno=?
```

## 데이터 삭제

```
▼ test\java\org\zerock\mallapi
  ▼ repository
    J TodoRepositoryTests.java
    J MallapiApplicationTests.java
```



```
@Test
public void testDelete() {
    Long tno = 1L;

    todoRepository.deleteById(tno);
}
```

```
Hibernate:
select
    t1_0.tno,
    t1_0.complete,
    t1_0.due_date,
    t1_0.title,
    t1_0.writer
from
    tbl_todo t1_0
where
    t1_0.tno=?
Hibernate:
delete
from
    tbl_todo
where
    tno=?
```

## 페이징 처리

```
test\java\org\zerock\mallapi
  repository
    J TodoRepositoryTests.java
    J MallapiApplicationTests.java
```

```
Hibernate:
  select
    t1_0.tno,
    t1_0.complete,
    t1_0.due_date,
    t1_0.title,
    t1_0.writer
  from
    tbl_todo t1_0
  order by
    t1_0.tno desc limit ?,
    ?
```

```
@SpringBootTest
@Log4j2
public class TodoRepositoryTests {
    ...
    @Test
    public void testPaging() {

        //import org.springframework.data.domain.Pageable;
        Pageable pageable = PageRequest.of(0,10, Sort.by("tno").descending());

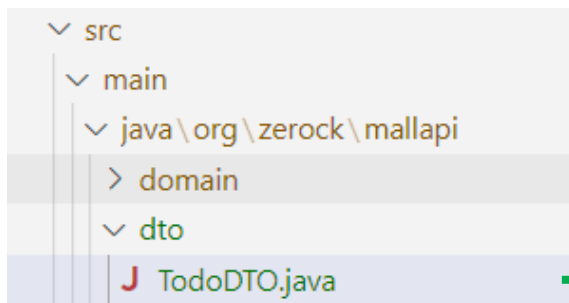
        Page<Todo> result = todoRepository.findAll(pageable);

        log.info(result.getTotalElements());

        result.getContent().stream().forEach(todo -> log.info(todo));
    }
}
```

## 서비스 계층과 DTO 처리

→ dto 패키지를 생성하고 TodoDTO 클래스를 생성



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

    private Long tno;

    private String title;

    private String writer;

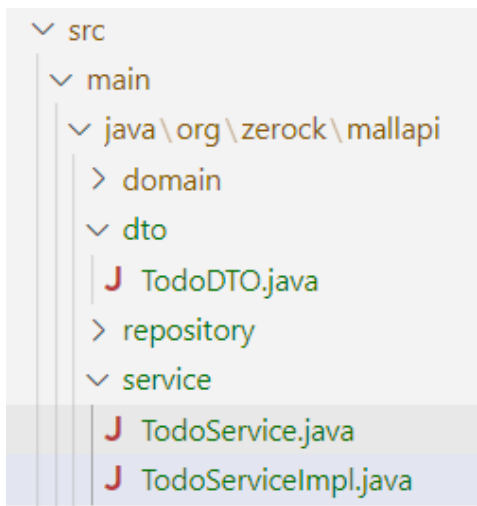
    private boolean complete;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")
    private LocalDate dueDate;

}
```

## 서비스 선언

→ service 패키지 생성하고 TodoService 인터페이스와 TodoServiceImpl 클래스를 생성



```
package org.zerock.mallapi.service;

import org.zerock.mallapi.dto.TODO;

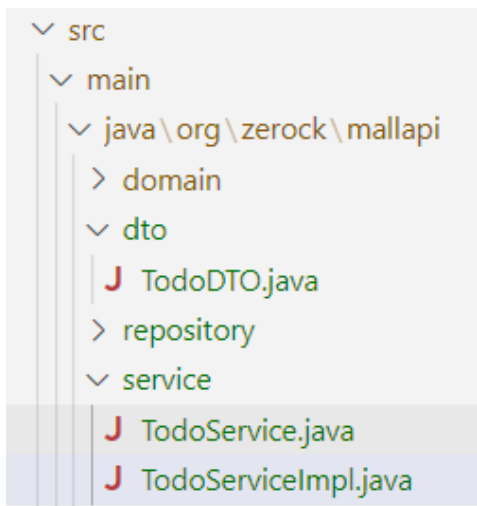
public interface TodoService {

    Long register(TODO todo);

}
```

## 서비스 선언

→ service 패키지 생성하고 TodoService 인터페이스와 TodoServiceImpl 클래스를 생성



```
package org.zerock.mallapi.service;

import org.zerock.mallapi.dto.TODO;

public interface TodoService {

    Long register(TODO todo);

}
```

## 서비스 선언

→ service 패키지 생성하고 TodoService 인터페이스와 TodoServiceImpl 클래스를 생성



```
package org.zerock.mallapi.service;

import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.zerock.mallapi.dto.TODO;

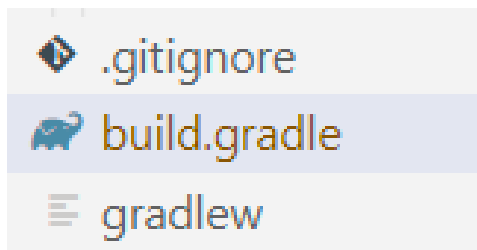
import lombok.extern.log4j.Log4j2;

@Service
@Transactional
@Log4j2
public class TodoServiceImpl implements TodoService {
    @Override
    public Long register(TODO todo) {
        log.info(".....");
        return null;
    }
}
```



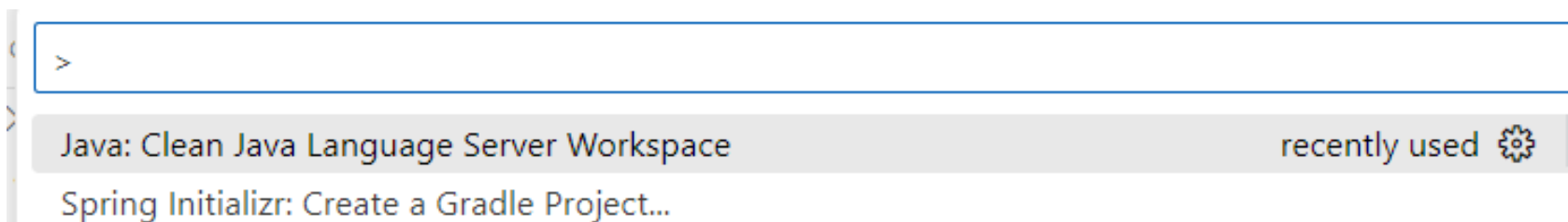
## ModelMapper 라이브러리

→ build.gradle 파일에 ModelMapper 라이브러리 추가



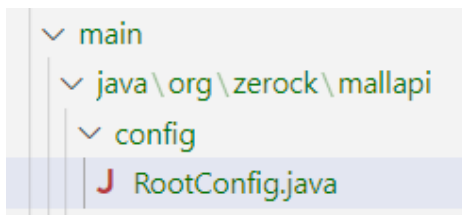
```
dependencies {  
    ...생략  
    implementation 'org.modelmapper:modelmapper:3.1.1'  
}
```

→ 라이브러리를 추가한 후 'Clean..Server Workspace'



## ModelMapper 라이브러리

→ config 패키지 추가 후 RootConfig 클래스 생성



```
@Configuration
public class RootConfig {

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

## 등록 기능의 구현




```
@Service
@Transactional
@Log4j2
@RequiredArgsConstructor // 생성자 자동 주입
public class TodoServiceImpl implements TodoService {

    //자동주입 대상은 final로
    private final ModelMapper modelMapper;
    private final TodoRepository todoRepository;

    @Override
    public Long register(TodoDTO todoDTO) {
        log.info(".....");
        Todo todo = modelMapper.map(todoDTO, Todo.class);
        Todo savedTodo = todoRepository.save(todo);
        return savedTodo.getId();
    }
}
```

## 등록 기능의 구현

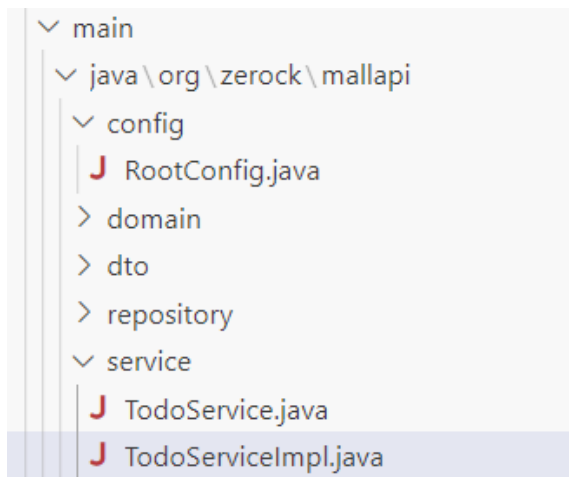
```
✓ test\java\org\zerock\mallapi  
  > repository  
  ✓ service  
    J TodoServiceTests.java
```



```
@SpringBootTest  
@Log4j2  
public class TodoServiceTests {  
  
    @Autowired  
    private TodoService todoService;  
  
    @Test  
    public void testRegister() {  
        TodoDTO todoDTO = TodoDTO.builder()  
            .title("서비스 테스트")  
            .writer("tester")  
            .dueDate(LocalDate.of(2023, 10, 10))  
            .build();  
        Long tno = todoService.register(todoDTO);  
        log.info("TNO: " + tno);  
    }  
}
```

## 조회 기능의 구현

→ 조회용 메서드 추가



```
public interface TodoService {
    Long register(TodoDTO todoDTO);
    TodoDTO get(Long tno);
}
```

```
@Override
public TodoDTO get(Long tno) {
    Optional<Todo> result = todoRepository.findById(tno);
    Todo todo = result.orElseThrow();
    TodoDTO dto = modelMapper.map(todo, TodoDTO.class);
    return dto;
}
```

## 조회 기능의 구현

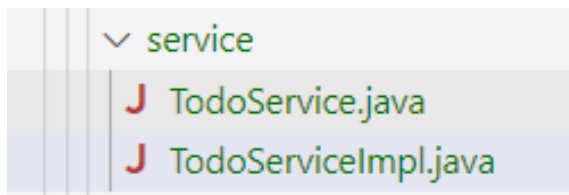
→ 조회용 메서드 추가

```
test\java\org\zerock\mallapi  
  > repository  
  > service  
    J TodoServiceTests.java
```

```
@Test  
public void testGet() {  
  
    Long tno = 101L;  
  
    TodoDTO todoDTO = todoService.get(tno);  
  
    log.info(todoDTO);  
  
}
```

## 수정/삭제 기능의 구현

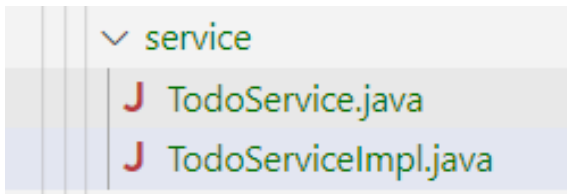
→ 수정 / 삭제 메서드 추가



```
public interface TodoService {  
    Long register(TodoDTO todoDTO);  
    TodoDTO get(Long tno);  
    void modify(TodoDTO todoDTO);  
    void remove(Long tno);  
}
```

## 수정/삭제 기능의 구현

→ 수정 / 삭제 메서드 추가



```
@Override
public void modify(TodoDTO todoDTO) {

    Optional<Todo> result = todoRepository.findById(todoDTO.getTno());

    Todo todo = result.orElseThrow();
    todo.changeTitle(todoDTO.getTitle());
    todo.changeDueDate(todoDTO.getDueDate());
    todo.changeComplete(todoDTO.isComplete());

    todoRepository.save(todo);
}

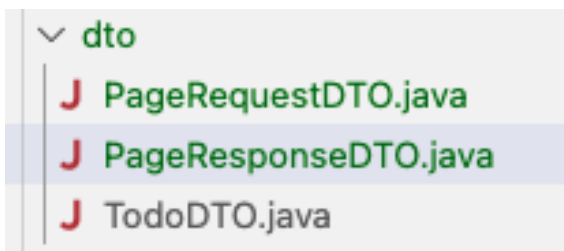
@Override
public void remove(Long tno) {
    todoRepository.deleteById(tno);
}
```



## 페이징 처리된 목록 데이터

→ TodoDTO 리스트 (해당 페이지의 데이터) :

페이지 번호, 전체 데이터 수, 이전/다음 페이지 등 부가 정보



```
@Data
@SuperBuilder
@AllArgsConstructor
@NoArgsConstructor
public class PageRequestDTO {

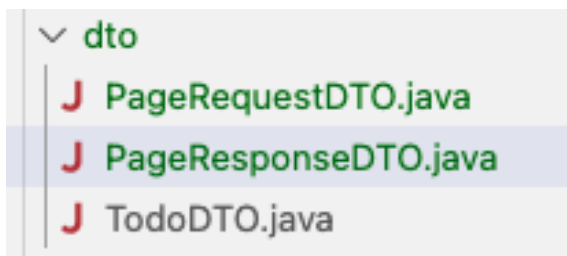
    @Builder.Default
    private int page = 1;

    @Builder.Default
    private int size = 10;

}
```

## 페이징 처리된 목록 데이터

→ TodoDTO 리스트 (해당 페이지의 데이터)



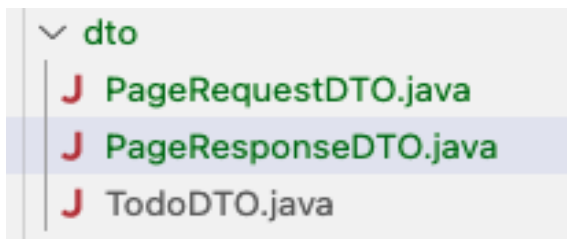
```
package org.zerock.mallapi.dto;

import lombok.Builder;
import lombok.Data;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

@Data
public class PageResponseDTO<E> {
    private List<E> dtoList;
    private List<Integer> pageNumList;
    private PageRequestDTO pageRequestDTO;
    private boolean prev, next;
    private int totalCount, prevPage, nextPage, totalPages, current;
```

## 페이징 처리된 목록 데이터



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

    this.dtoList = dtoList;
    this.pageRequestDTO = pageRequestDTO;
    this.totalCount = (int)totalCount;

    int end = (int)(Math.ceil( pageRequestDTO.getPage() / 10.0 )) * 10;
    int start = end - 9;
    int last = (int)(Math.ceil((totalCount/(double)pageRequestDTO.getSize())));
    end = end > last ? last: end;
    this.prev = start > 1;
    this.next = totalCount > end * pageRequestDTO.getSize();

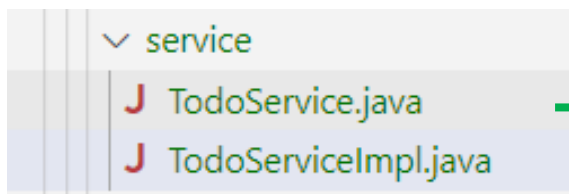
    this.pageNumList = IntStream.rangeClosed(start,end).boxed().collect(Collectors.toList());

    if(prev)
        this.prevPage = start - 1;

    if(next)
        this.nextPage = end + 1;

    this.totalPage = this.pageNumList.size();
    this.current = pageRequestDTO.getPage();
}
}
```

## 목록(페이징) 처리 구현



```
package org.zerock.mallapi.service;

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

public interface TodoService {

    ...생략

    PageResponseDTO<TODO> list(PageRequestDTO pageRequestDTO);
}
```

## 목록(페이징) 처리 구현



```
package org.zerock.mallapi.service;

import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import org.modelmapper.ModelMapper;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.zerock.mallapi.domain.TODO;
import org.zerock.mallapi.dto.PageRequestDTO;
import org.zerock.mallapi.dto.PageResponseDTO;
import org.zerock.mallapi.dto.TODODTO;
import org.zerock.mallapi.repository.TODORepository;

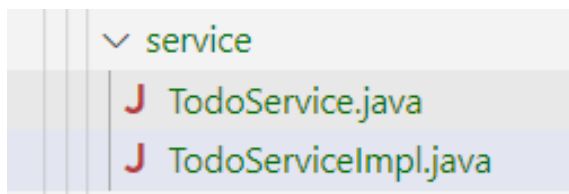
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;

@Service
@Transactional
@Log4j2
@RequiredArgsConstructor // 생성자 자동 주입
public class TodoServiceImpl implements TodoService {

    //자동주입 대상은 final로
    private final ModelMapper modelMapper;

    private final TODORepository todoRepository;
```

## 목록(페이징) 처리 구현



...생략

```
@Override
public PageResponseDTO<TodoDTO> list(PageRequestDTO pageRequestDTO) {
    Pageable pageable =
        PageRequest.of(
            pageRequestDTO.getPage() - 1, // 1페이지가 0이므로 주의
            pageRequestDTO.getSize(),
            Sort.by("tno").descending());

    Page<Todo> result = todoRepository.findAll(pageable);

    List<TodoDTO> dtoList = result.getContent().stream()
        .map(todo -> modelMapper.map(todo, TodoDTO.class))
        .collect(Collectors.toList());


    long totalCount = result.getTotalElements();

    PageResponseDTO<TodoDTO> responseDTO = PageResponseDTO.<TodoDTO>withAll()
        .dtoList(dtoList)
        .pageRequestDTO(pageRequestDTO)
        .totalCount(totalCount)
        .build();

    return responseDTO;
}
```

## 목록(페이징) 처리 구현

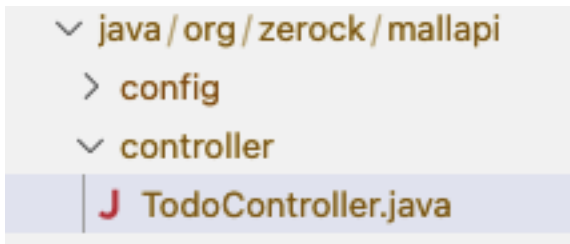
```
✓ test\java\org\zerock\mallapi  
  > repository  
  ✓ service  
    J TodoServiceTests.java
```



```
@Test  
public void testList() {  
    PageRequestDTO pageRequestDTO = PageRequestDTO.builder()  
        .page(2)  
        .size(10)  
        .build();  
  
    PageResponseDTO<TodoDTO> response = todoService.list(pageRequestDTO);  
    log.info(response);  
}
```

## RESTful 서비스를 위한 컨트롤러

→ controller 패키지 구성 후 TodoController 클래스 추가



```
@RestController
@RequiredArgsConstructor
@Log4j2
@RequestMapping("/api/todo")
public class TodoController {
    private final TodoService service;

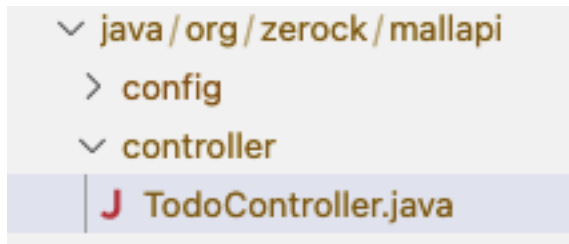
    @GetMapping("/{tno}")
    public TodoDTO get(@PathVariable(name = "tno") Long tno) {
        return service.get(tno);
    }

    @GetMapping("/list")
    public PageResponseDTO<TodoDTO> list(PageRequestDTO pageRequestDTO) {
        log.info(pageRequestDTO);
        return service.list(pageRequestDTO);
    }
}
```



## RESTful 서비스를 위한 컨트롤러

→ controller 패키지 구성 후 TodoController 클래스 추가

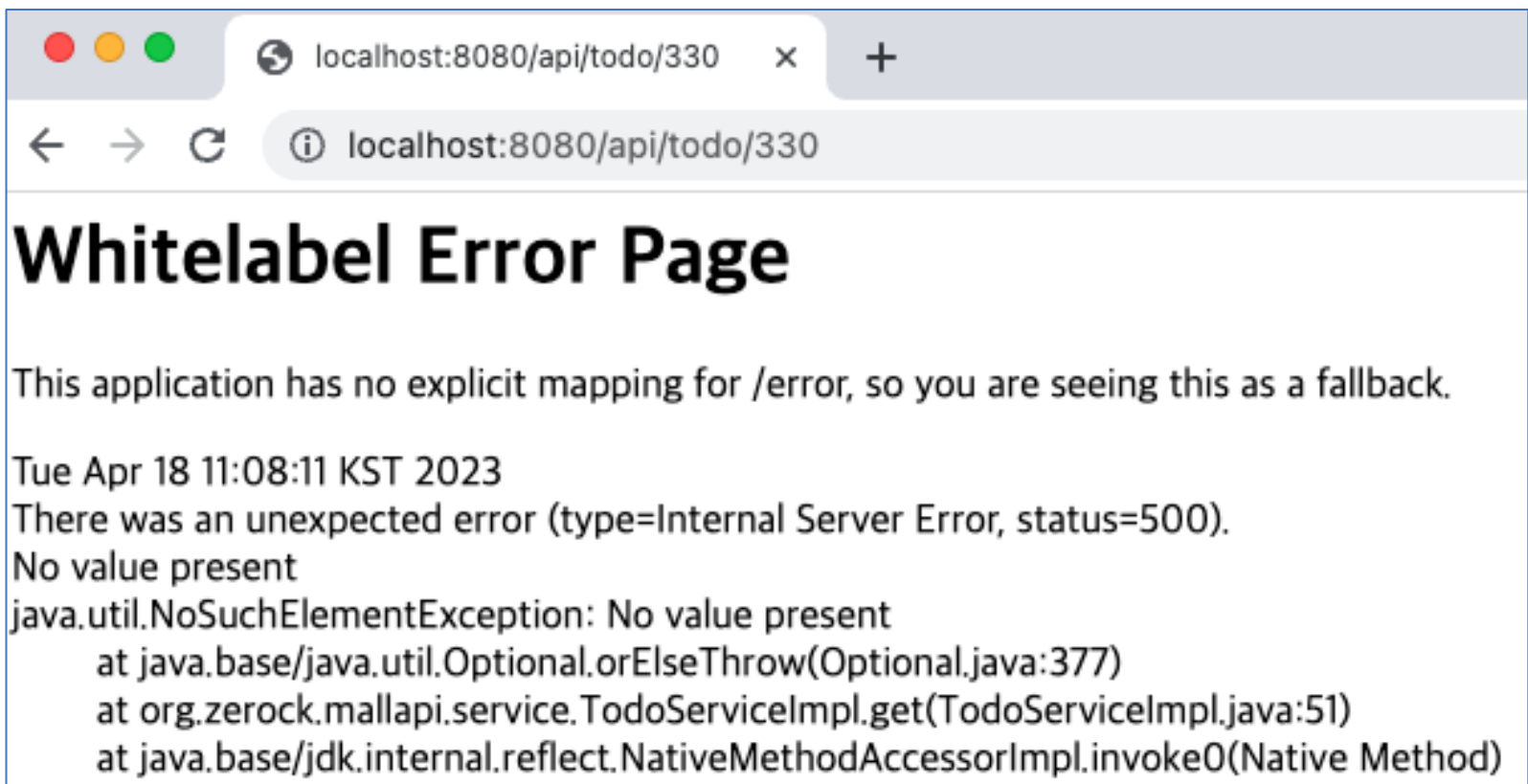


```
@RestController
@RequiredArgsConstructor
@Log4j2
@RequestMapping("/api/todo")
public class TodoController {
    private final TodoService service;

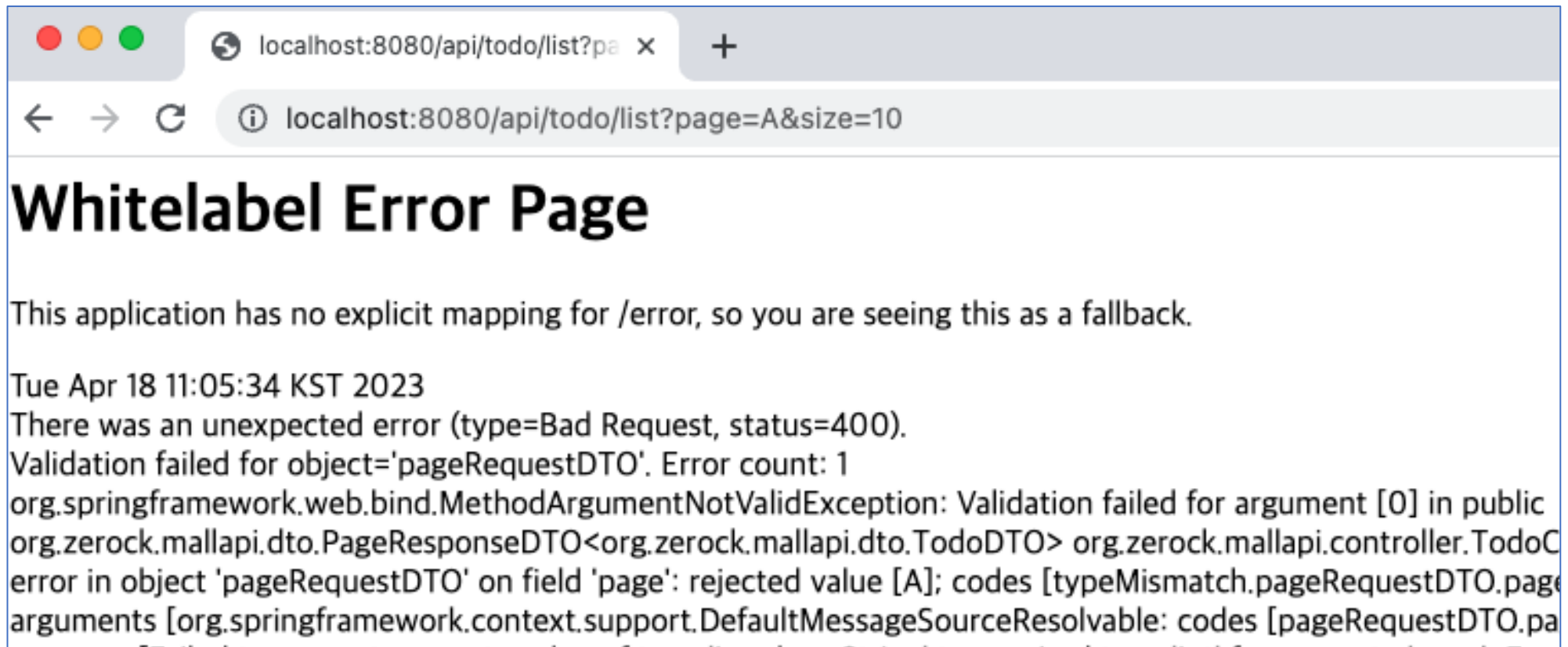
    @GetMapping("/{tno}")
    public TodoDTO get(@PathVariable(name = "tno") Long tno) {
        return service.get(tno);
    }

    @GetMapping("/list")
    public PageResponseDTO<TodoDTO> list(PageRequestDTO pageRequestDTO) {
        log.info(pageRequestDTO);
        return service.list(pageRequestDTO);
    }
}
```

## NoSuchElementException

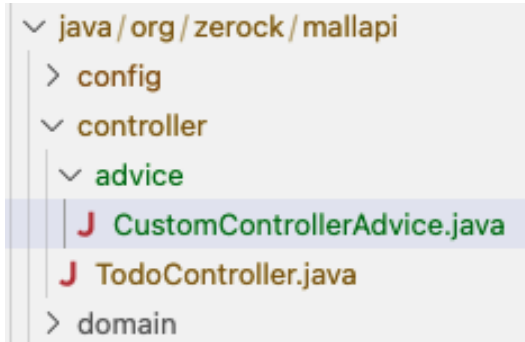


## MethodArgumentNotValidException



## 예외를 처리

→ controller 패키지 내에 advice 패키지를 추가 후 CustomControllerAdvice 클래스 생성



```
@RestControllerAdvice
public class CustomControllerAdvice {

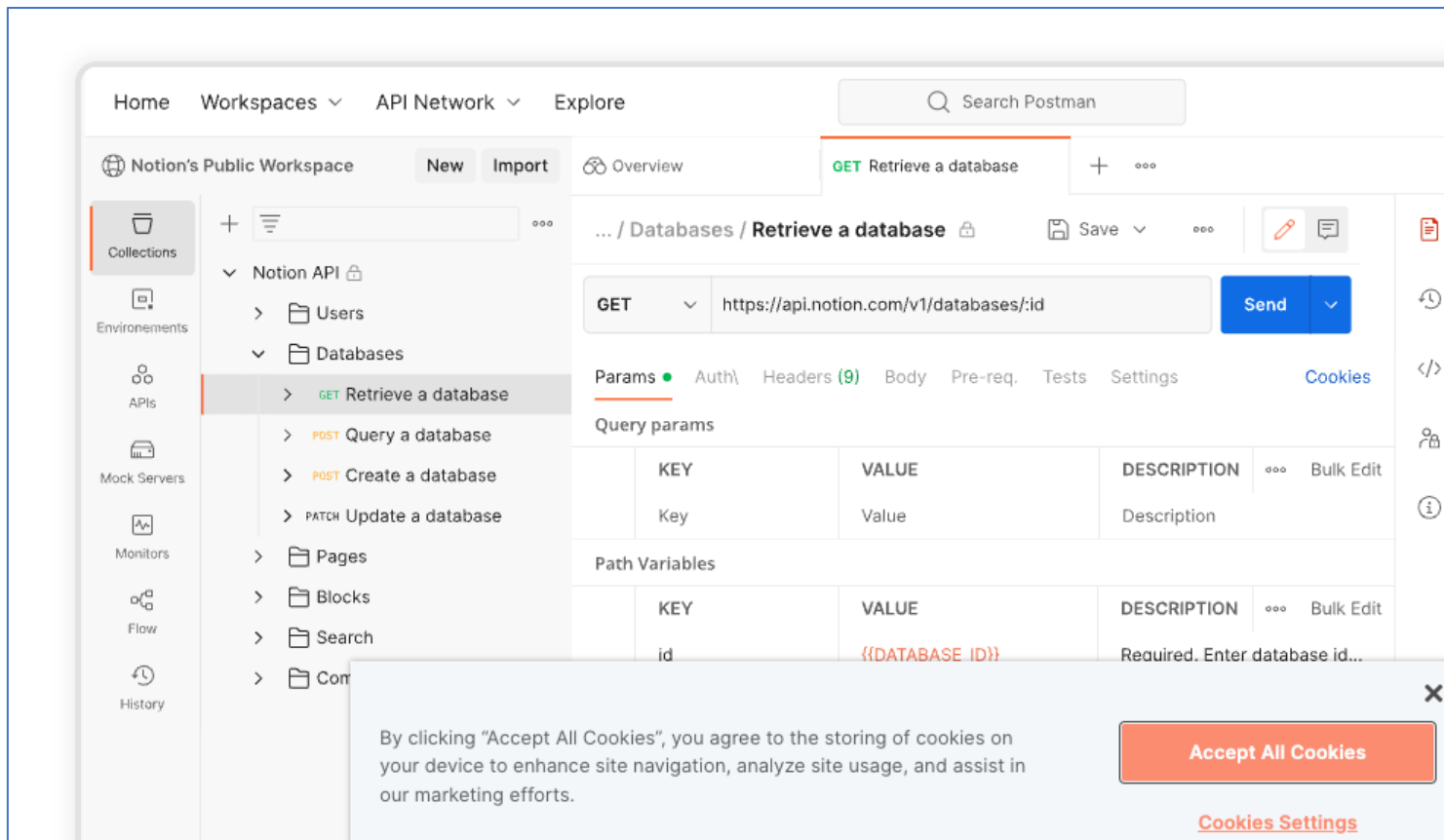
    @ExceptionHandler({NoSuchElementException.class})
    protected ResponseEntity<?> notExist(NoSuchElementException e) {

        String msg = e.getMessage();
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(Map.of("msg", msg));
    }

    @ExceptionHandler({MethodArgumentNotValidException.class})
    protected ResponseEntity<?>
    handleIllegalArgumentException(MethodArgumentNotValidException e) {

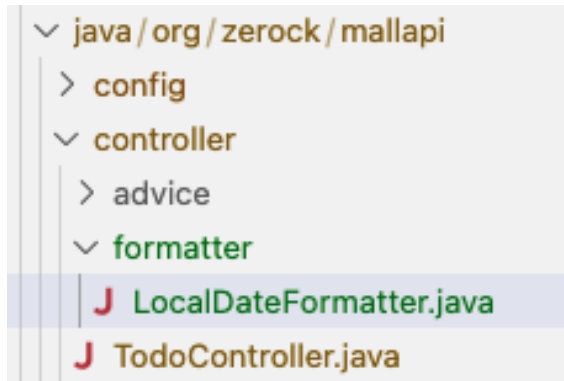
        String msg = e.getMessage();
        return ResponseEntity.status(HttpStatus.NOT_ACCEPTABLE).body(Map.of("msg", msg));
    }
}
```

## REST API 테스트 프로그램 : Psotman



## Formatter를 이용한 LocalDate 처리

→ controller 패키지 하위로 formatter 패키지 생성 후 LocalDateFormatter 클래스 생성



```
package org.zerock.mallapi.controller.formatter;

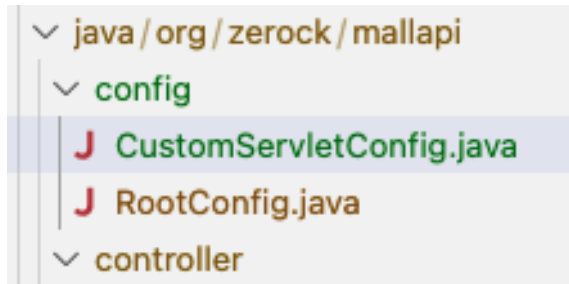
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Locale;

import org.springframework.format.Formatter;

public class LocalDateFormatter implements Formatter<LocalDate>{
    @Override
    public LocalDate parse(String text, Locale locale) {
        return LocalDate.parse(text, DateTimeFormatter.ofPattern("yyyy-MM-dd"));
    }
    @Override
    public String print(LocalDate object, Locale locale) {
        return DateTimeFormatter.ofPattern("yyyy-MM-dd").format(object);
    }
}
```

## Formatter를 이용한 LocalDate 처리

→ config 패키지에 CustomServletConfig 클래스를 생성하고 설정 추가



```
package org.zerock.mallapi.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.format.FormatterRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.zerock.mallapi.controller.formatter.LocalDateFormatter;

@Configuration
public class CustomServletConfig implements WebMvcConfigurer{

    @Override
    public void addFormatters(FormatterRegistry registry) {

        registry.addFormatter(new LocalDateFormatter());
    }
}
```

## POST 방식의 등록 처리

→ TodoController 클래스에 기능 메소드 추가

```
▼ java\org\zerock\mallapi
  > config
  ▼ controller
    > advice
    > formatter
    J TodoController.java
```

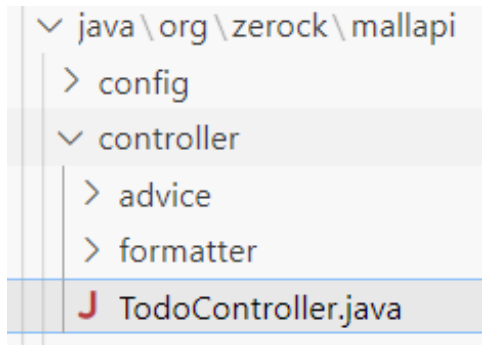
```
@PostMapping("/")
public Map<String, Long> register(@RequestBody TodoDTO todoDTO){
    log.info("TodoDTO: " + todoDTO);
    Long tno = service.register(todoDTO);
    return Map.of("TNO", tno);
}
```

```
{
    "title": "Sample Title",
    "writer": "user1",
    "dueDate": "2023-10-10"
}
```

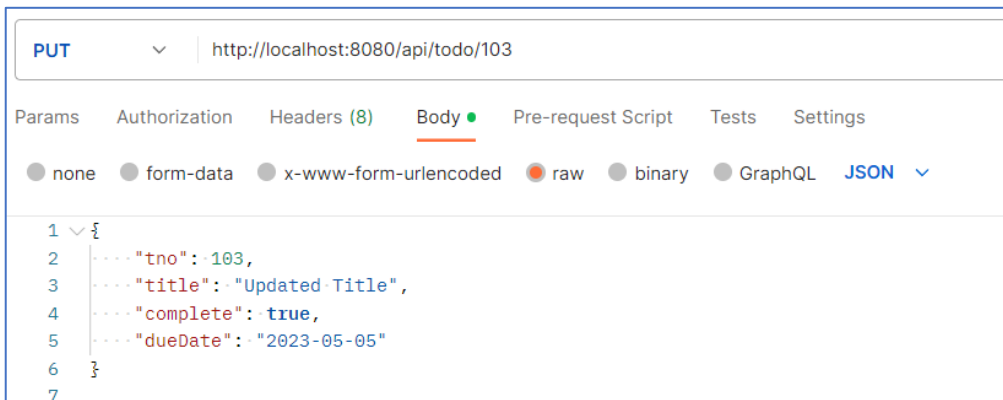


## PUT 방식의 수정 처리

→ TodoController 클래스에 기능 메소드 추가



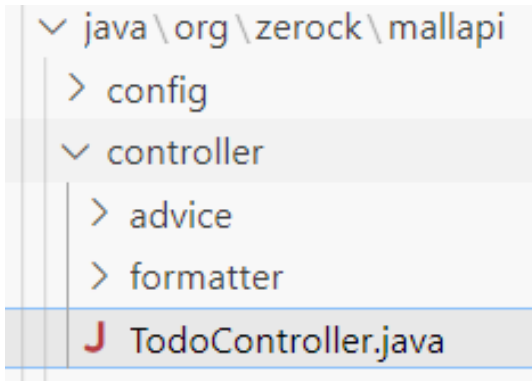
```
@PutMapping("/{tno}")
public Map<String, String> modify(
    @PathVariable(name="tno") Long tno,
    @RequestBody TodoDTO todoDTO) {
    todoDTO.setTno(tno);
    log.info("Modify: " + todoDTO);
    service.modify(todoDTO);
    return Map.of("RESULT", "SUCCESS");
}
```



```
{
  "tno": 103,
  "title": "Updated Title",
  "complete": true,
  "dueDate": "2023-05-05"
}
```

## DELETE 방식의 삭제 처리

→ TodoController 클래스에 기능 메소드 추가



```
@DeleteMapping("/{tno}")
public Map<String, String> remove( @PathVariable(name="tno") Long tno ){

    log.info("Remove: " + tno);

    service.remove(tno);

    return Map.of("RESULT", "SUCCESS");

}
```

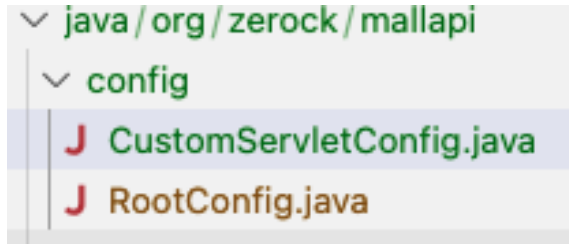
## 결과 확인

The screenshot displays a REST client interface. At the top, a dropdown menu is set to 'DELETE' and the URL is 'http://localhost:8080/api/todo/3'. Below this, a tabbed interface shows 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is selected, and the content type is set to 'JSON'. The response body is shown in a 'Pretty' view, displaying a JSON object: 

```
{  "RESULT": "SUCCESS"}
```

## CORS관련 설정

→ CustomServletConfig에 추가적인 설정



```
@Configuration
public class CustomServletConfig implements WebMvcConfigurer{

    @Override
    public void addFormatters(FormatterRegistry registry) {
        registry.addFormatter(new LocalDateFormatter());
    }

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**").allowedOrigins("*")
            .allowedMethods("HEAD", "GET", "POST", "PUT", "DELETE", "OPTIONS")
            .maxAge(300)
            .allowedHeaders("Authorization", "Cache-Control", "Content-Type");
    }
}
```

감사합니다.