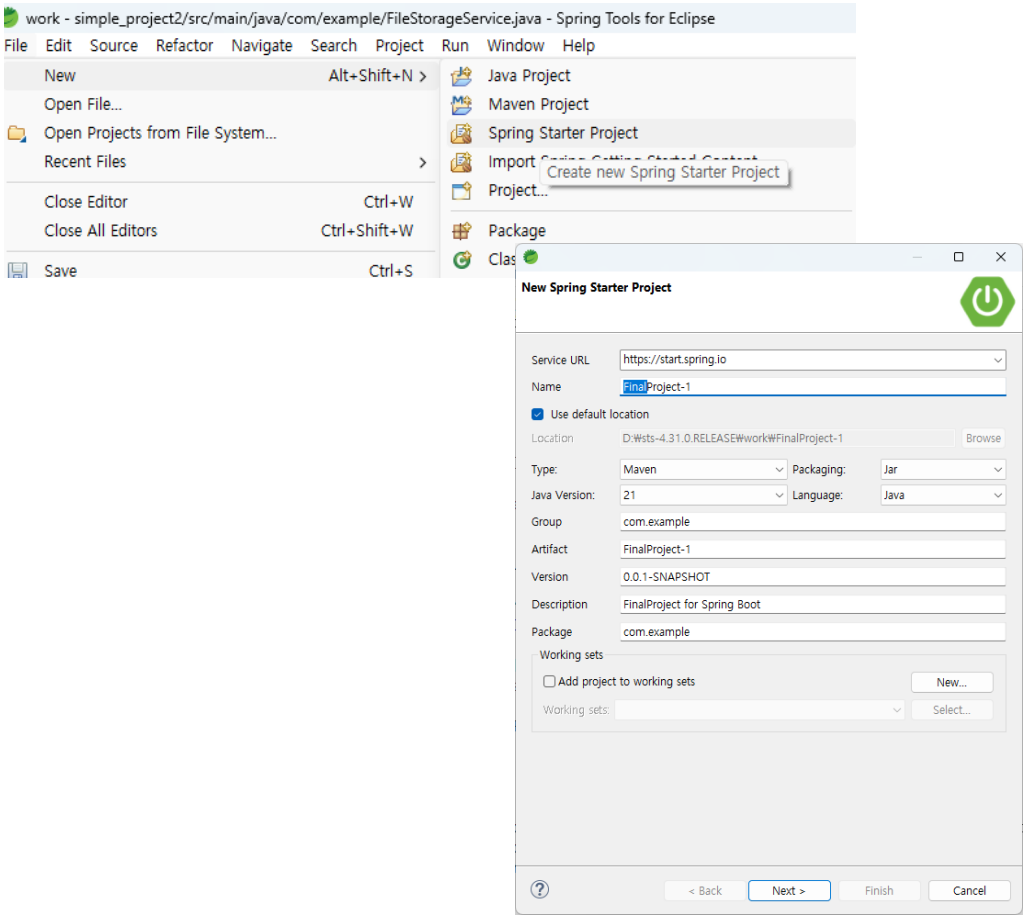
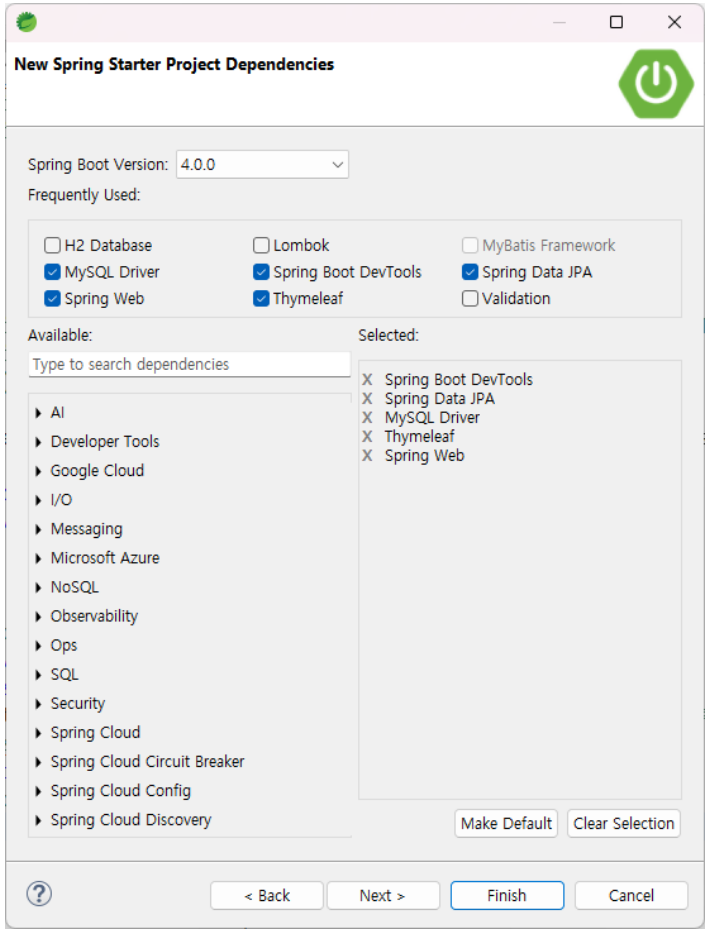
# 15주차 대면수업

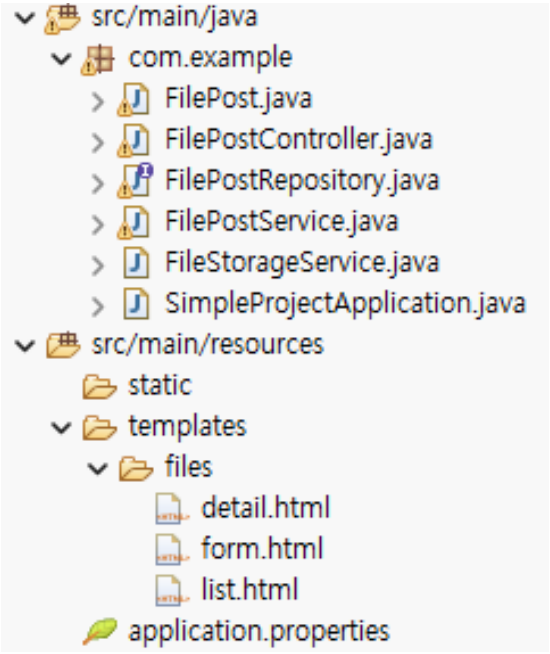
## AI를 활용한 자료실 만들기

# 프로젝트 생성


work - simple_project2/src/main/java/com/example/FileStorageService.java - Spring Tools for Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Java Project
Maven Project
Spring Starter Project
Import
Project...
Package
Create new Spring Starter Project
New Spring Starter Project
Service URL https://start.spring.io
Name FinalProject-1
Use default location
Location D:₩sts-4.31.0.RELEASE₩work₩FinalProject-1
Browse
Type: Maven
Packaging: Jar
Java Version: 21
Language: Java
Group com.example
Artifact FinalProject-1
Version 0.0.1-SNAPSHOT
Description FinalProject for Spring Boot
Package com.example
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel


New Spring Starter Project Dependencies
Spring Boot Version: 4.0.0
Frequently Used:
H2 Database
Lombok
MyBatis Framework
MySQL Driver
Spring Boot DevTools
Spring Data JPA
Spring Web
Thymeleaf
Validation
Available:
Type to search dependencies
AI
Developer Tools
Google Cloud
I/O
Messaging
Microsoft Azure
NoSQL
Observability
Ops
SQL
Security
Spring Cloud
Spring Cloud Circuit Breaker
Spring Cloud Config
Spring Cloud Discovery
Selected:
X Spring Boot DevTools
X Spring Data JPA
X MySQL Driver
X Thymeleaf
X Spring Web
Make Default
Clear Selection
< Back
Next >
Finish
Cancel

# 파일 계층 구조




[기능]

- 자료 목록 보기
- 자료 등록 (파일 업로드 1개)
- 자료 상세보기
- 자료 수정
- 자료 삭제
- 파일 다운로드

[URL 예시]

- 목록: GET /files
- 등록 폼: GET /files/new
- 등록 처리: POST /files
- 상세: GET /files/{id}
- 수정 폼: GET /files/{id}/edit
- 수정 처리: POST /files/{id}
- 삭제: POST /files/{id}/delete
- 다운로드: GET /files/{id}/download

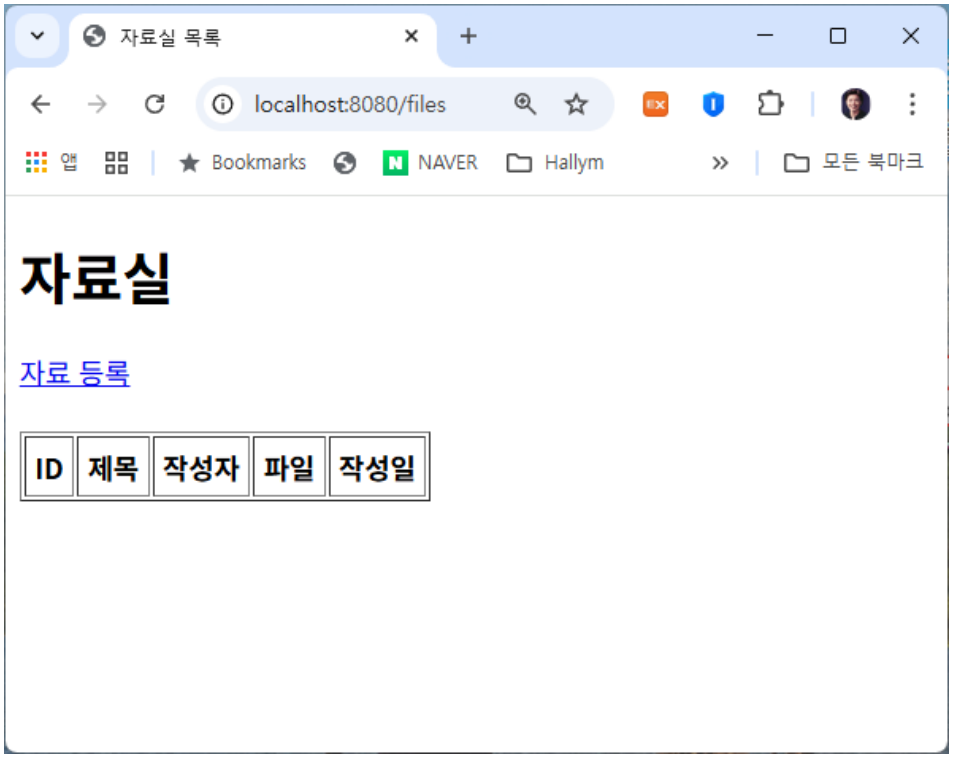
# application.properties

```
# ========================
# MySQL Database Configuration
# ========================
spring.datasource.url=jdbc:mysql://localhost:3306/board?serverTimezone=Asia/Seoul&characterEncoding=UTF-8&useSSL=false&allowPublicKeyRetrieval=true

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=kim1234

# ========================
# JPA / Hibernate 설정
# ========================
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true


# ========================
# Server 설정
# ========================
server.port=8080

spring.servlet.multipart.max-file-size=20MB
spring.servlet.multipart.max-request-size=20MB

# 파일 업로드 저장 경로 (원하는 경로로 변경 가능)
file.upload-dir=E:/upload
```

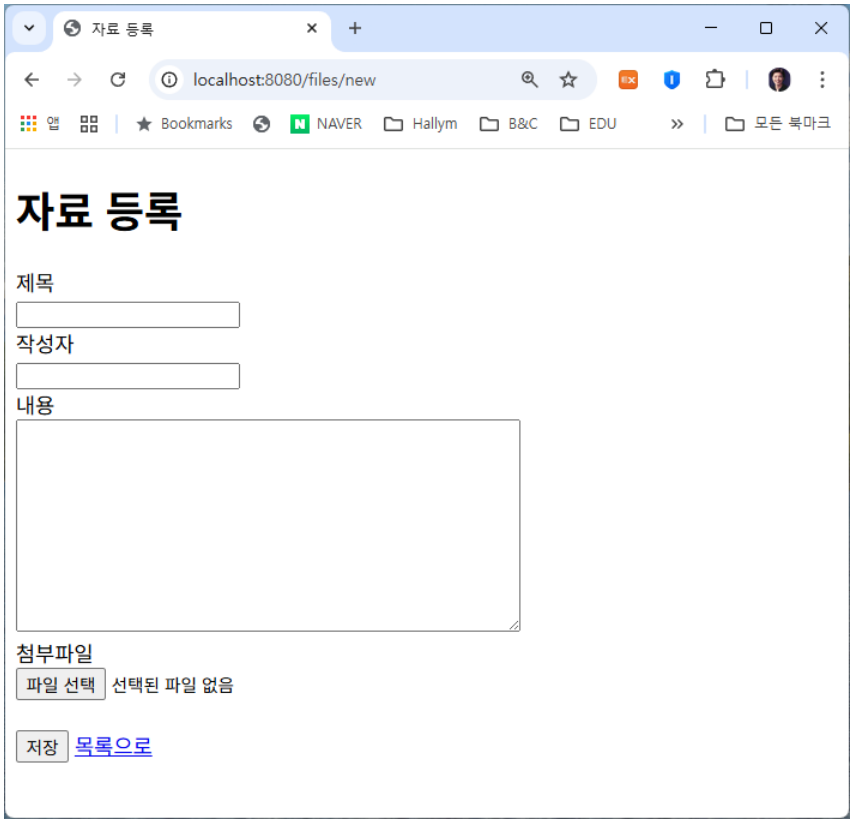
| 옵션 | 값 | 역할 / 의미 |
|---|---|---|
| serverTimezone | Asia/Seoul | DB 연결 시 사용하는 서버 시간대를 서울로 설정 (시간 오류 방지) |
| characterEncoding | UTF-8 | 문자 인코딩을 UTF-8로 지정 (한글 깨짐 방지) |
| useSSL | false | SSL 연결 비활성화 |
| allowPublicKeyRetrieval | true | MySQL 8 인증 방식에서 public key 요청 허용 |

| 속성 | 값 | 역할 / 의미 |
|---|---|---|
| spring.servlet.multipart.max-file-size | 20MB | 업로드 가능한 단일 파일의 최대 크기를 20MB로 제한 |
| spring.servlet.multipart.max-request-size | 20MB | 한 번의 업로드 요청에서 허용되는 전체 데이터의 최대 크기를 20MB로 제한 |

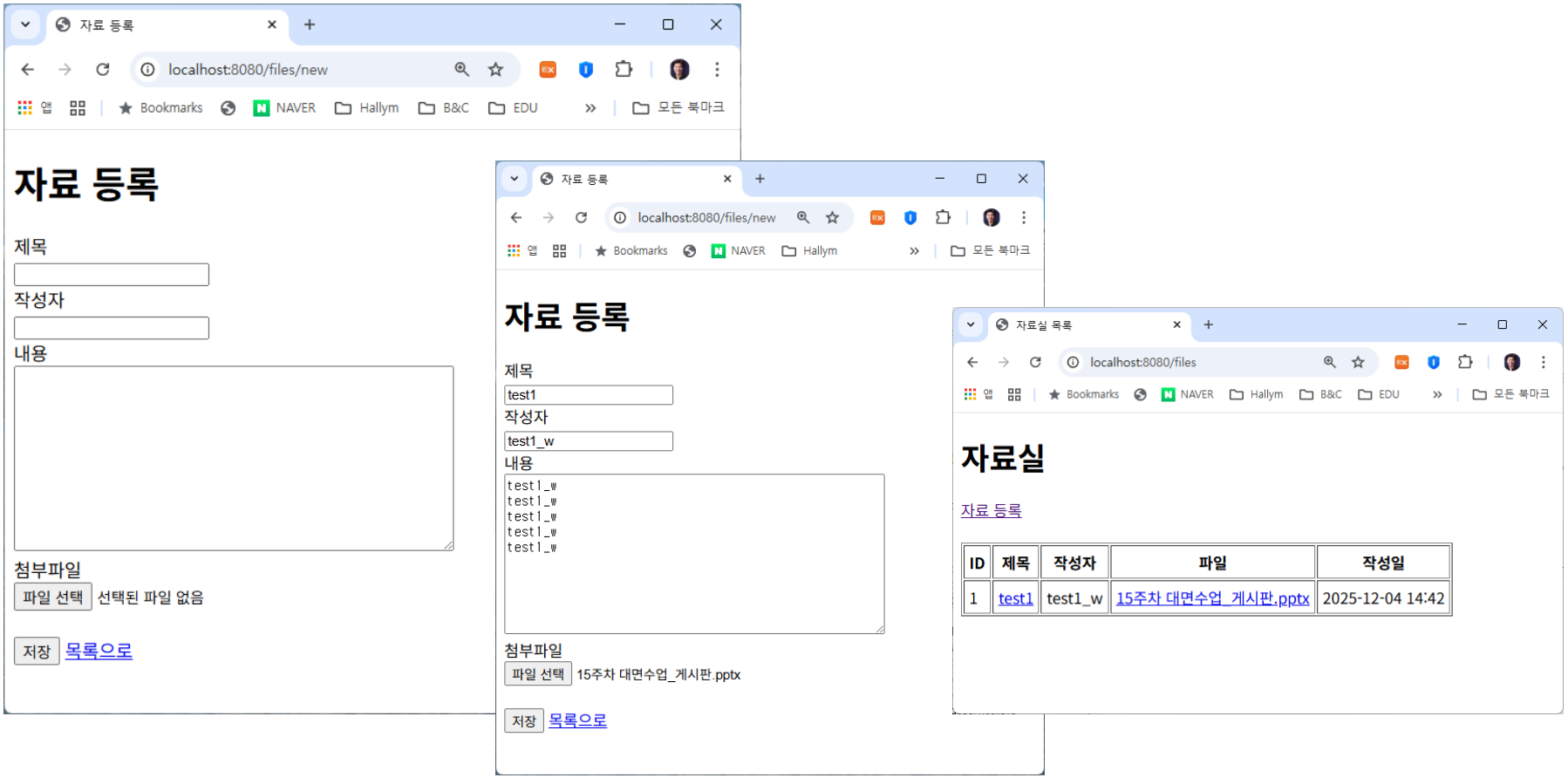
# 자료실 리스트 첫 화면


자료실 목록
localhost:8080/files
Bookmarks
NAVER
Hallym
모든 북마크
자료실
자료 등록
ID
제목
작성자
파일
작성일

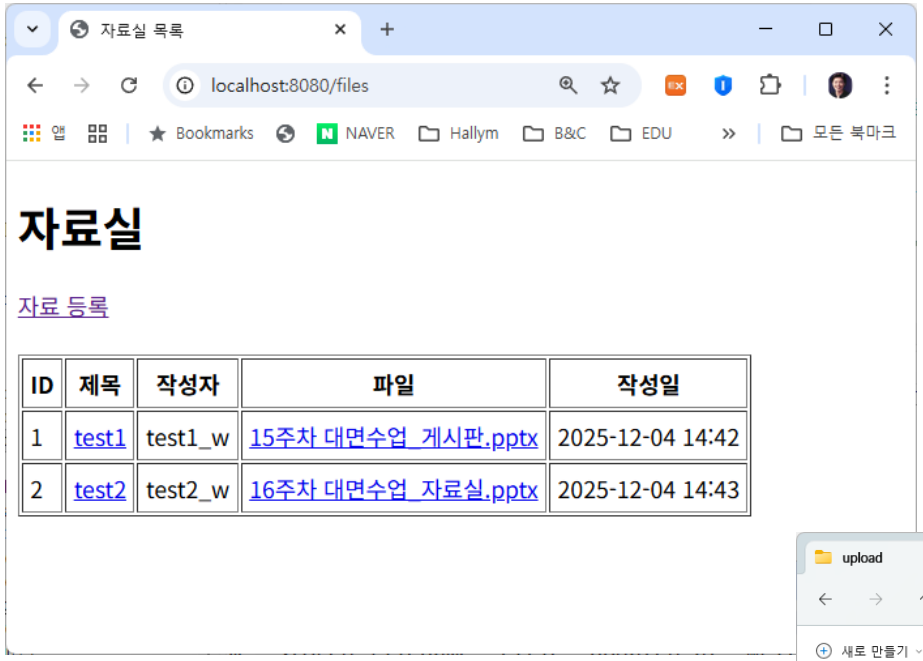
# 자료실 등록 화면

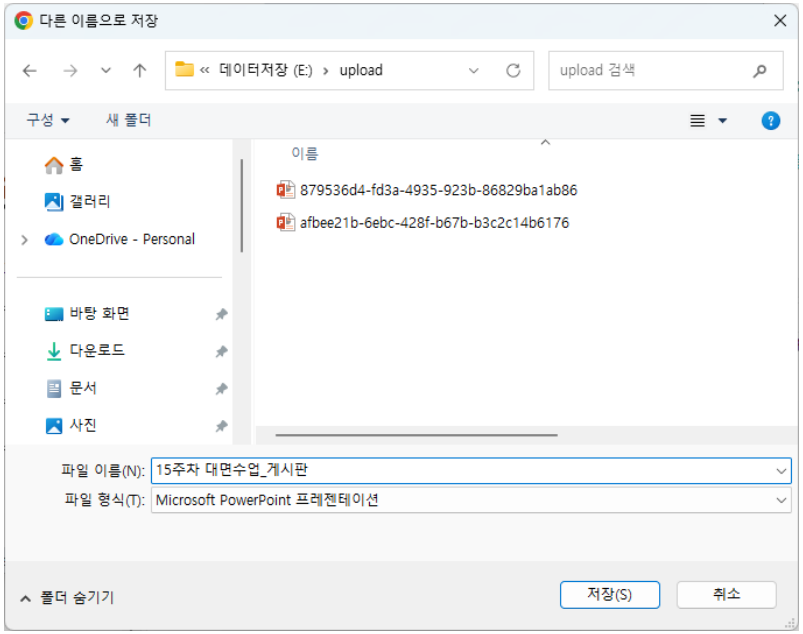

자료 등록
localhost:8080/files/new
앱
Bookmarks
NAVER
Hallym
B&C
EDU
모든 북마크
자료 등록
제목
작성자
내용
첨부파일
파일 선택
선택된 파일 없음
저장
목록으로

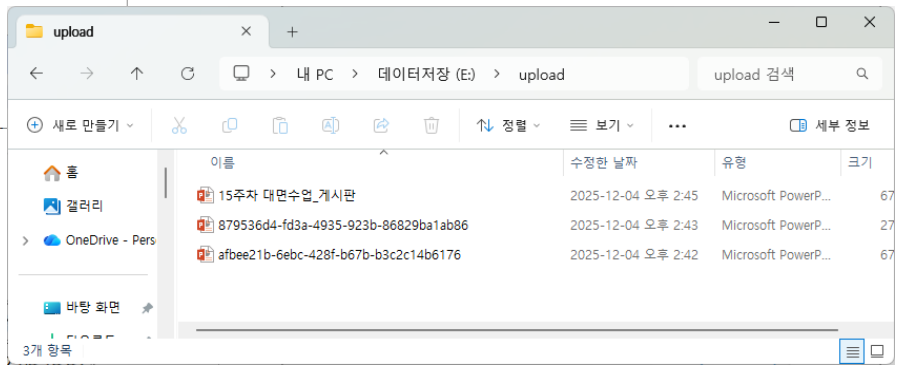
# 자료실 등록 화면-계속


자료 등록
localhost:8080/files/new
Bookmarks
NAVER
Hallym
B&C
EDU
모든 북마크
자료 등록
제목
작성자
내용
첨부파일
파일 선택
선택된 파일 없음
저장
목록으로
자료 등록
localhost:8080/files/new
자료 등록
제목
test1
작성자
test1_w
내용
test1_w
test1_w
test1_w
test1_w
test1_w
첨부파일
파일 선택
15주차 대면수업_게시판.pptx
저장
목록으로
자료실 목록
localhost:8080/files
자료실
자료 등록
ID
제목
작성자
파일
작성일
1
test1
test1_w
15주차 대면수업_게시판.pptx
2025-12-04 14:42

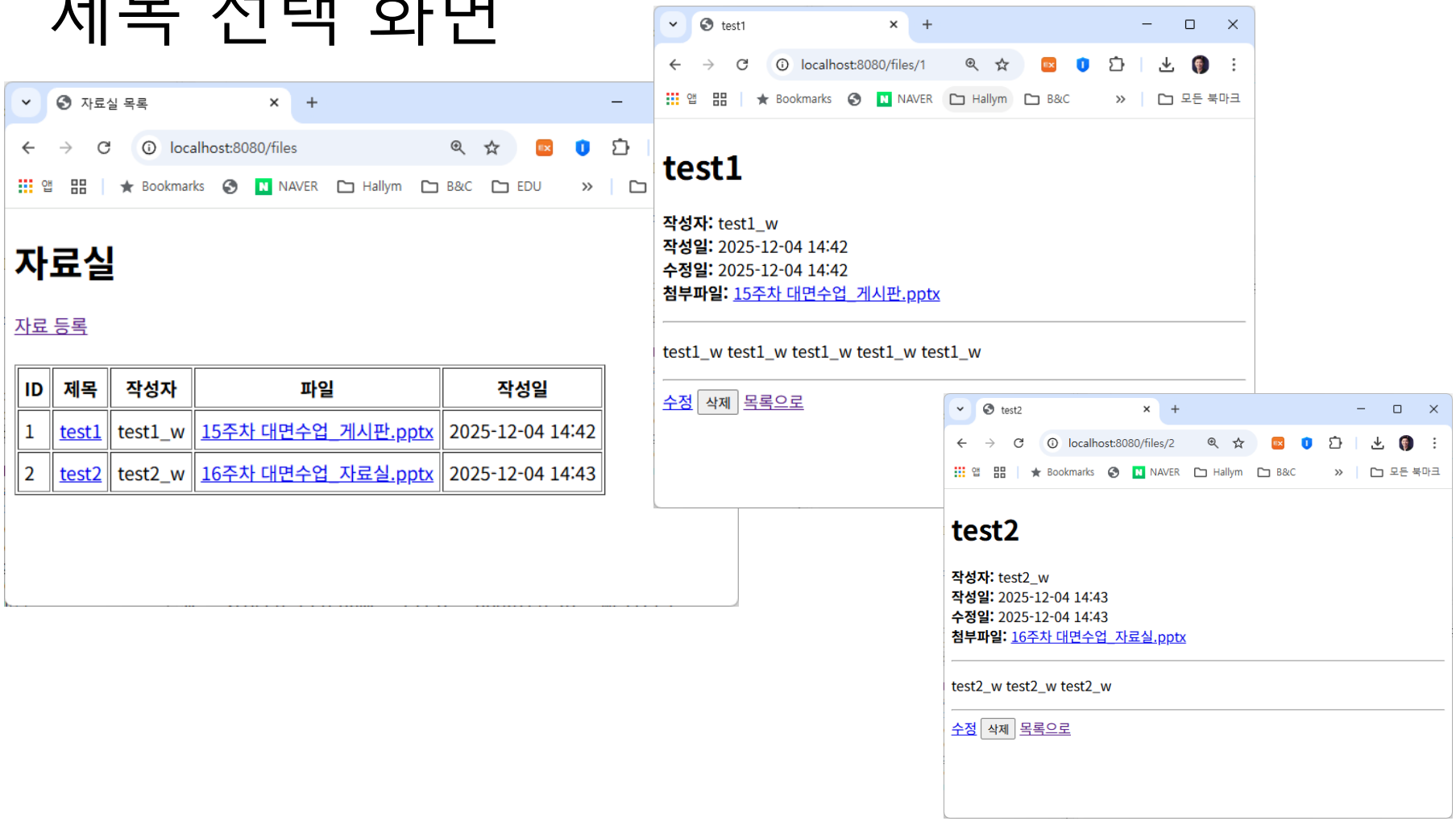
# 파일 다운로드


자료실 목록
localhost:8080/files
Bookmarks
NAVER
Hallym
B&C
EDU
모든 북마크
자료실
자료 등록
ID
제목
작성자
파일
작성일
1
test1
test1_w
15주차 대면수업_게시판.pptx
2025-12-04 14:42
2
test2
test2_w
16주차 대면수업_자료실.pptx
2025-12-04 14:43

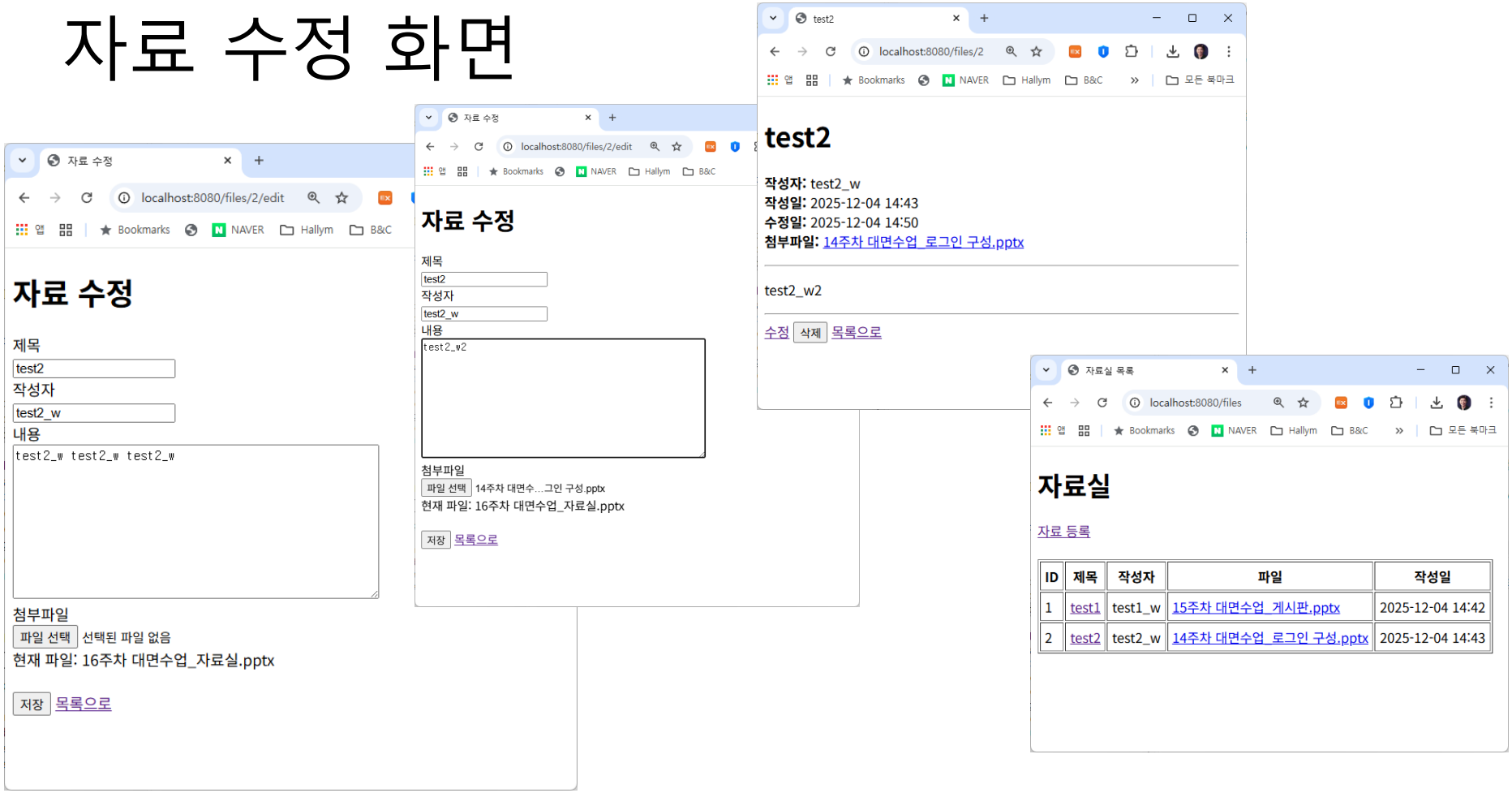

다른 이름으로 저장
« 데이터저장 (E:) › upload
upload 검색
구성
새 폴더
이름
879536d4-fd3a-4935-923b-86829ba1ab86
afbee21b-6ebc-428f-b67b-b3c2c14b6176
홈
갤러리
OneDrive - Personal
바탕 화면
다운로드
문서
사진
파일 이름(N): 15주차 대면수업_게시판
파일 형식(T): Microsoft PowerPoint 프레젠테이션
폴더 숨기기
저장(S)
취소


upload
내 PC › 데이터저장 (E:) › upload
upload 검색
새로 만들기
정렬
보기
세부 정보
홈
갤러리
OneDrive - Pers
바탕 화면
이름
수정한 날짜
유형
크기
15주차 대면수업_게시판
2025-12-04 오후 2:45
Microsoft PowerP...
879536d4-fd3a-4935-923b-86829ba1ab86
2025-12-04 오후 2:43
Microsoft PowerP...
afbee21b-6ebc-428f-b67b-b3c2c14b6176
2025-12-04 오후 2:42
Microsoft PowerP...
3개 항목

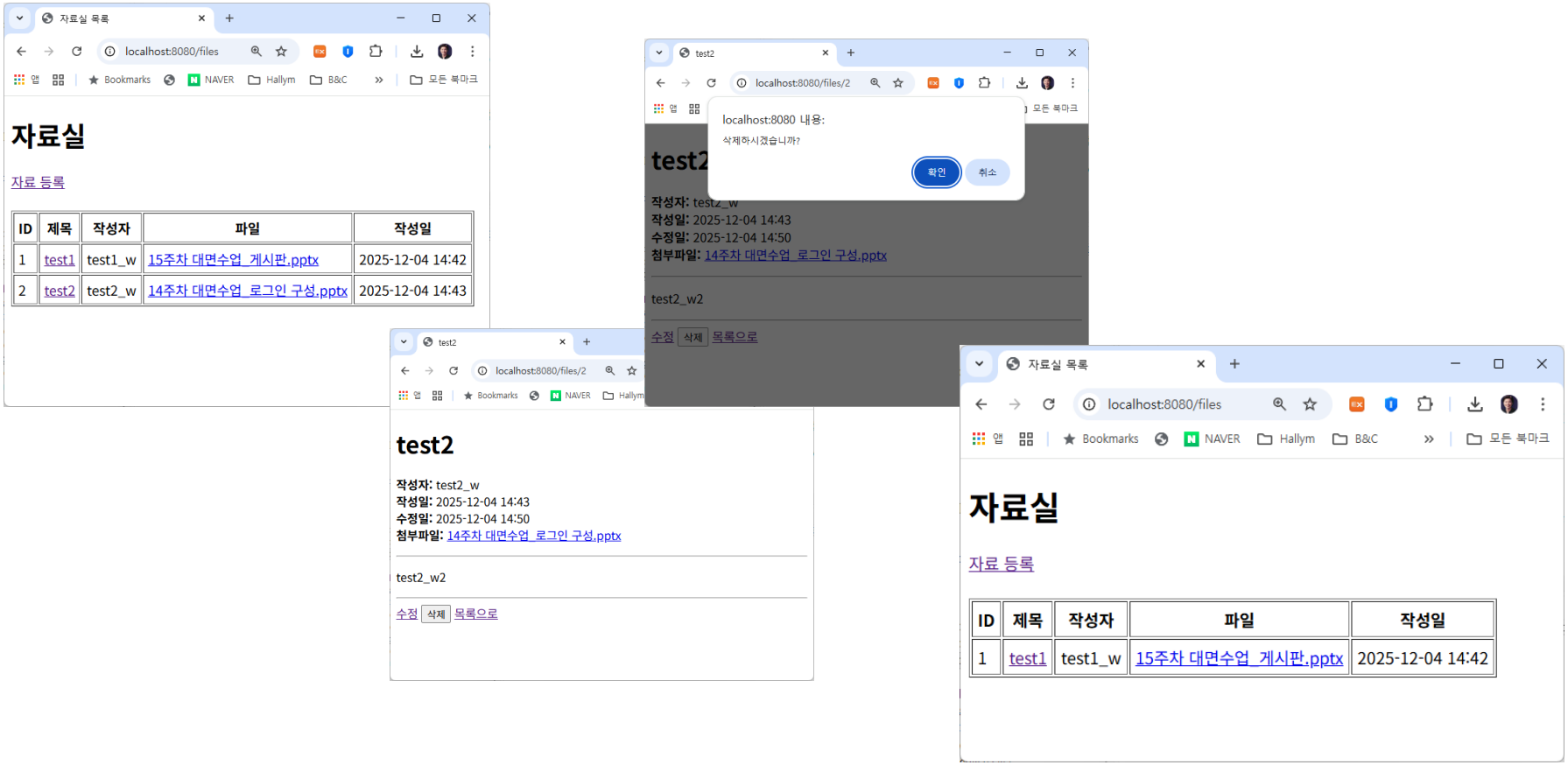
# 제목 선택 화면


자료실 목록
localhost:8080/files
Bookmarks
NAVER
Hallym
B&C
EDU
자료실
자료 등록
ID
제목
작성자
파일
작성일
1
test1
test1_w
15주차 대면수업_게시판.pptx
2025-12-04 14:42
2
test2
test2_w
16주차 대면수업_자료실.pptx
2025-12-04 14:43
test1
localhost:8080/files/1
test1
작성자: test1_w
작성일: 2025-12-04 14:42
수정일: 2025-12-04 14:42
첨부파일: 15주차 대면수업_게시판.pptx
test1_w test1_w test1_w test1_w test1_w
수정
삭제
목록으로
test2
localhost:8080/files/2
test2
작성자: test2_w
작성일: 2025-12-04 14:43
수정일: 2025-12-04 14:43
첨부파일: 16주차 대면수업_자료실.pptx
test2_w test2_w test2_w
수정
삭제
목록으로

# 자료 수정 화면


자료 수정
localhost:8080/files/2/edit
Bookmarks
NAVER
Hallym
B&C
자료 수정
제목
test2
작성자
test2_w
내용
test2_w test2_w test2_w
첨부파일
파일 선택
선택된 파일 없음
현재 파일: 16주차 대면수업_자료실.pptx
저장
목록으로
자료 수정
localhost:8080/files/2/edit
제목
test2
작성자
test2_w
내용
test2_w2
첨부파일
파일 선택
14주차 대면수...그인 구성.pptx
현재 파일: 16주차 대면수업_자료실.pptx
저장
목록으로
test2
localhost:8080/files/2
test2
작성자: test2_w
작성일: 2025-12-04 14:43
수정일: 2025-12-04 14:50
첨부파일: 14주차 대면수업_로그인 구성.pptx
test2_w2
수정
삭제
목록으로
자료실 목록
localhost:8080/files
자료실
자료 등록
ID
제목
작성자
파일
작성일
1
test1
test1_w
15주차 대면수업_게시판.pptx
2025-12-04 14:42
2
test2
test2_w
14주차 대면수업_로그인 구성.pptx
2025-12-04 14:43

# 자료 삭제 화면


자료실 목록
localhost:8080/files
자료실
자료 등록
ID 제목 작성자 파일 작성일
1 test1 test1_w 15주차 대면수업_게시판.pptx 2025-12-04 14:42
2 test2 test2_w 14주차 대면수업_로그인 구성.pptx 2025-12-04 14:43
test2
localhost:8080/files/2
localhost:8080 내용:
삭제하시겠습니까?
확인
취소
작성자: test2_w
작성일: 2025-12-04 14:43
수정일: 2025-12-04 14:50
첨부파일: 14주차 대면수업_로그인 구성.pptx
test2_w2
수정 삭제 목록으로

# templates/files/list.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>자료실 목록</title>
</head>
<body>
<h1>자료실</h1>

<a th:href="@{/files/new}">자료 등록</a>
<br/><br/>

<table border="1" cellpadding="5">
<thead>
<tr>
<th>ID</th>
<th>제목</th>
<th>작성자</th>
<th>파일</th>
<th>작성일</th>
</tr>
</thead>
<tbody>
<tr th:each="post : ${posts}">
<td th:text="${post.id}">1</td>
<td>
<a th:href="@{|/files/${post.id}|}" th:text="${post.title}">제목</a>
</td>
<td th:text="${post.writer}">작성자</td>
<td>
<span th:if="${post.originalFilename != null}">
<a th:href="@{|/files/${post.id}/download|}"
th:text="${post.originalFilename}">파일</a>
</span>
</td>
<td th:text="${#temporals.format(post.createdAt, 'yyyy-MM-dd HH:mm')}">2025-01-01</td>
</tr>
</tbody>
</table>
</body>
</html>
```

# templates/files/form.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title th:text="${post.id} != null ? '자료 수정' : '자료 등록'">자료 등록</title>
</head>
<body>
<h1 th:text="${post.id} != null ? '자료 수정' : '자료 등록'">자료 등록</h1>

<form th:action="${post.id} != null ? @{'/files/' + ${post.id}} : @{/files}"
method="post"
enctype="multipart/form-data">
<div>
<label>제목</label><br/>
<input type="text" name="title" th:value="${post.title}">
</div>
<div>
<label>작성자</label><br/>
<input type="text" name="writer" th:value="${post.writer}">
</div>
<div>
<label>내용</label><br/>
<textarea name="content" rows="10" cols="50" th:text="${post.content}"></textarea>
</div>
<div>
<label>첨부파일</label><br/>
<!-- 새로 업로드할 파일 -->
<input type="file" name="file"><br/>
<!-- 기존 파일 정보 표시 -->
<span th:if="${post.originalFilename != null}">
현재 파일: <span th:text="${post.originalFilename}"></span>
</span>
</div>
<br/>
<button type="submit">저장</button>
<a th:href="@{/files}">목록으로</a>
</form>
</body>
</html>
```

# templates/files/detail.html

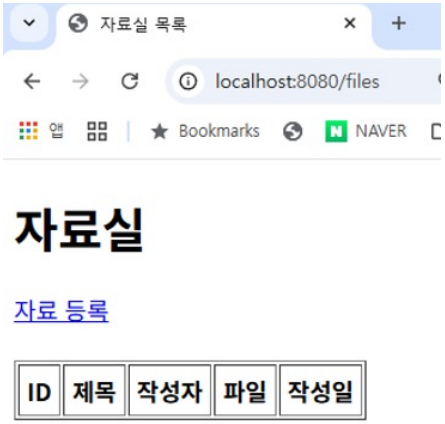
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title th:text="${post.title}">자료 상세</title>
</head>
<body>
<h1 th:text="${post.title}">제목</h1>
<p>
<strong>작성자:</strong> <span th:text="${post.writer}"></span><br/>
<strong>작성일:</strong>
<span th:text="${#temporals.format(post.createdAt, 'yyyy-MM-dd HH:mm')}"></span><br/>
<strong>수정일:</strong>
<span th:text="${#temporals.format(post.updatedAt, 'yyyy-MM-dd HH:mm')}"></span><br/>
<strong>첨부파일:</strong>
<span th:if="${post.originalFilename != null}">
<a th:href="@{|/files/${post.id}/download|}"
th:text="${post.originalFilename}">다운로드</a>
</span>
<span th:if="${post.originalFilename == null}">
(없음)
</span>
</p>
<hr/>

<p th:text="${post.content}">내용</p>

<hr/>
<a th:href="@{|/files/${post.id}/edit|}">수정</a>
<form th:action="@{|/files/${post.id}/delete|}" method="post" style="display:inline;">
<button type="submit" onclick="return confirm('삭제하시겠습니까?')">삭제</button>
</form>

<a th:href="@{/files}">목록으로</a>
</body>
</html>
```

# SimpleProjectApplication.java

```
package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SimpleProjectApplication {
	//http://localhost:8080/lists
	public static void main(String[] args) {
		SpringApplication.run(SimpleProjectApplication.class, args);
	}

}
```

# FilePost.java

```java
@Entity
@Table(name = "file_posts")
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class FilePost {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, length = 200)
    private String title;

    @Column(nullable = false, columnDefinition = "TEXT")
    private String content;

    @Column(nullable = false, length = 50)
    private String writer;

    // 파일 원래 이름
    private String originalFilename;

    // 서버에 저장된 파일 이름 (UUID 등)
    private String storedFilename;

    private Long fileSize;

    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;

    @PrePersist
    public void prePersist() {
        this.createdAt = LocalDateTime.now();
        this.updatedAt = LocalDateTime.now();
    }

    @PreUpdate
    public void preUpdate() {
        this.updatedAt = LocalDateTime.now();
    }

    public Long getId() {   return id;  }
    public void setId(Long id) { this.id = id;  }
    public String getTitle() {  return title;   }
    public void setTitle(String title) {    this.title = title; }
    public String getContent() {    return content; }
    public void setContent(String content) {    this.content = content; }
    public String getWriter() { return writer;  }
    public void setWriter(String writer) {      this.writer = writer;   }
    public String getOriginalFilename() {       return originalFilename;    }
    public void setOriginalFilename(String originalFilename) {  this.originalFilename = originalFilename;   }
    public String getStoredFilename() { return storedFilename;  }
    public void setStoredFilename(String storedFilename) {  this.storedFilename = storedFilename; }
    public Long getFileSize() {     return fileSize;    }
    public void setFileSize(Long fileSize) {    this.fileSize = fileSize;   }
    public LocalDateTime getCreatedAt() {   return createdAt;   }
    public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }
    public LocalDateTime getUpdatedAt() {   return updatedAt;   }
    public void setUpdatedAt(LocalDateTime updatedAt) { this.updatedAt = updatedAt; }
}
```



# FilePostController.java

```java
@Controller
@RequestMapping("/files")
@RequiredArgsConstructor
public class FilePostController {

    private final FilePostService filePostService;
    private final FileStorageService fileStorageService;

    public FilePostController(FilePostService filePostService, FileStorageService fileStorageService) {
        this.filePostService = filePostService;
        this.fileStorageService = fileStorageService;
    }

    // 목록
    @GetMapping
    public String list(Model model) {
        model.addAttribute("posts", filePostService.findAll());
        return "files/list";
    }

    // 등록 폼
    @GetMapping("/new")
    public String createForm(Model model) {
        model.addAttribute("post", new FilePost());
        return "files/form";
    }

    // 등록 처리
    @PostMapping
    public String create(@ModelAttribute FilePost post,
                         @RequestParam("file") MultipartFile file) throws IOException {
        filePostService.save(post, file);
        return "redirect:/files";
    }
```

# FilePostController.java-계속

```java
// 상세보기
@GetMapping("/{id}")
public String detail(@PathVariable Long id, Model model) {
    FilePost post = filePostService.findById(id);
    model.addAttribute("post", post);
    return "files/detail";
}

// 수정 폼
@GetMapping("/{id}/edit")
public String editForm(@PathVariable Long id, Model model) {
    FilePost post = filePostService.findById(id);
    model.addAttribute("post", post);
    return "files/form";
}

// 수정 처리 (파일도 다시 업로드 가능)
@PostMapping("/{id}")
public String update(@PathVariable Long id,
                     @ModelAttribute FilePost formPost,
                     @RequestParam(value = "file", required = false) MultipartFile file) throws IOException {

    FilePost post = filePostService.findById(id);

    post.setTitle(formPost.getTitle());
    post.setContent(formPost.getContent());
    post.setWriter(formPost.getWriter());

    filePostService.save(post, file);
    return "redirect:/files/" + id;
}
```

# FilePostController.java-계속

```
    // 삭제
    @PostMapping("/{id}/delete")
    public String delete(@PathVariable Long id) {
        filePostService.deleteById(id);
        return "redirect:/files";
    }

    // 파일 다운로드
    @GetMapping("/{id}/download")
    public ResponseEntity<Resource> download(@PathVariable Long id) throws MalformedURLException {
        FilePost post = filePostService.findById(id);

        if (post.getStoredFilename() == null) {
            throw new IllegalArgumentException("첨부파일이 없습니다.");
        }

        Resource resource = fileStorageService.loadFileAsResource(post.getStoredFilename());

        String encodedFileName = URLEncoder.encode(post.getOriginalFilename(), StandardCharsets.UTF_8)
                .replaceAll("\\+", "%20");

        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION,
                        "attachment; filename=\"" + encodedFileName + "\"")
                .body(resource);
    }
}
```

| 필드명 | 타입 | 역할 / 의미 |
|---|---|---|
| originalFilename | String | 사용자가 업로드한 원본 파일 이름 (다운로드 시 사용자에게 보여줄 이름) |
| storedFilename | String | 서버에 저장된 실제 파일 이름(보통 UUID) 다운로드 시 서버가 찾는 파일 |
| fileSize | Long | 파일의 크기(Byte) 다운로드 시 표시 또는 검증 용도 |

| 필드 | 예시 값 |
|---|---|
| originalFilename | "보고서.pptx" |
| storedFilename | "a09e4f88-9221-43c1-bfef-e6bd6cbab491.pptx" |
| fileSize | 1532490 |

* 다운로드 동작 흐름 요약
1) 사용자가 파일 업로드
2) 서버가 저장하면서
   - 원본 이름: originalFilename
   - UUID 파일명: storedFilename
   - 파일 크기: fileSize
3) 다운로드 요청 시
   - storedFilename 으로 파일 조회
   - originalFilename 으로 사용자에게 제공

# FilePostRepository.java

```
package com.example;

import com.example.FilePost;
import org.springframework.data.jpa.repository.JpaRepository;

public interface FilePostRepository extends JpaRepository<FilePost, Long> {

}
```

| 항목 | 내용 |
|---|---|
| 파일명 / 클래스명 | FilePostRepository |
| 유형 | 인터페이스 |
| 상속 | JpaRepository<FilePost, Long> |
| 관리 엔티티 | FilePost |
| PK 타입 | Long |
| 역할 | FilePost 엔티티에 대한 데이터베이스 접근(DAO) |
| 스프링 빈 등록 | 별도 어노테이션 없어도 자동 등록 (Spring Data JPA가 처리) |
| 자동 제공 기능 | CRUD 기능 자동 제공 (save, findById, findAll, deleteById, count, existsById) |
| 구현 방식 | 개발자가 구현하지 않고, Spring Data JPA가 런타임에 구현체 생성 |
| 장점 | 코드 간결, 유지보수 용이, SQL 작성 불필요, 확장 쉬움 |
| 확장 방법 | 네이밍 기반 쿼리 메서드 추가 가능 (예: findByWriter(String writer)) |
| 사용 목적 | 자료 저장, 조회, 수정, 삭제를 DB와 연결하여 처리 |

파일을 저장하고, 그 정보를 포함한 FilePost 데이터를 DB에 관리하는 서비스 계층

# FilePostService.java

```java
package com.example;

import com.example.FilePost;
import com.example.FilePostReposit
import lombok.RequiredArgsConstru
import org.springframework.stereoty
import org.springframework.web.mu

import java.io.IOException;
import java.util.List;

@Service
@RequiredArgsConstructor
public class FilePostService {

private final FilePostRepository filePost
private final FileStorageService fileSt

public FilePostService(FilePostReposi
this.filePostRepository = filePostRepo
this.fileStorageService = fileStorageS
}

public List<FilePost> findAll() {
return filePostRepository.findAll();
}

public FilePost findById(Long id) {
return filePostRepository.findById(id)
.orElseThrow(() -> new IllegalArgume
}

public FilePost save(FilePost filePost,
if (file != null && !file.isEmpty()) {
String storedFilename = fileStorageS
filePost.setOriginalFilename(file.getO
filePost.setStoredFilename(storedFile
filePost.setFileSize(file.getSize());
}
return filePostRepository.save(filePos
}

public void deleteById(Long id) {
filePostRepository.deleteById(id);
// 실제 파일 삭제는 옵션 (필요하면 파
}
}
```

```java
@Service
@RequiredArgsConstructor
public class FilePostService {
    private final FilePostRepository filePostRepository;
    private final FileStorageService fileStorageService;

    public FilePostService(FilePostRepository filePostRepository, FileStorageService fileStorageService) {
        this.filePostRepository = filePostRepository;
        this.fileStorageService = fileStorageService;
    }

    public List<FilePost> findAll() {
        return filePostRepository.findAll();
    }

    public FilePost findById(Long id) {
        return filePostRepository.findById(id)
                .orElseThrow(() -> new IllegalArgumentException("자료가 없습니다. id=" + id));
    }

    public FilePost save(FilePost filePost, MultipartFile file) throws IOException {
        if (file != null && !file.isEmpty()) {
            String storedFilename = fileStorageService.storeFile(file);
            filePost.setOriginalFilename(file.getOriginalFilename());
            filePost.setStoredFilename(storedFilename);
            filePost.setFileSize(file.getSize());
        }
        return filePostRepository.save(filePost);
    }

    public void deleteById(Long id) {
        filePostRepository.deleteById(id);
        // 실제 파일 삭제는 옵션 (필요하면 파일도 삭제 로직 추가)
    }
```

| 필드명 | 타입 | 역할 |
|---|---|---|
| filePostRepository | FilePostRepository | DB CRUD 수행 |
| fileStorageService | FileStorageService | 파일 저장 및 로딩 처리 |

| 필드명 | 타입 | 역할 / 의미 |
|---|---|---|
| originalFilename | String | 사용자가 업로드한 원본 파일 이름 (다운로드 시 사용자에게 보여줄 이름) |
| storedFilename | String | 서버에 저장된 실제 파일 이름(보통 UUID) 다운로드 시 서버가 찾는 파일 |
| fileSize | Long | 파일의 크기(Byte) 다운로드 시 표시 또는 검증 용도 |

| save() 메소드 단계 | 동작 |
|---|---|
| 파일 존재 확인 | 업로드가 있는지 점검 |
| 파일 저장 | fileStorageService.storeFile(file) 호출 |
| DB에 저장될 파일 정보 설정 | originalFilename, storedFilename, fileSize 저장 |
| 게시물 저장 | filePostRepository.save(filePost) |

* 흐름 요약
1) 사용자가 업로드 요청
2) storeFile()로 파일 저장
3) FilePost 엔티티 값 설정
4) Repository로 DB 저장
5) 조회/수정/삭제 기능 지원

설정된 디렉토리에 파일을 저장하고, 저장된 파일을 Resource로 읽어오는 파일 업로드/다운로드 핵심 서비스 클래스

# FileStorageService.java

```java
@Service
public class FileStorageService {
    private final Path uploadDir;

    // application.properties 의 file.upload-dir 값을 주입
    public FileStorageService(@Value("${file.upload-dir}") String uploadDir) throws IOException {
        this.uploadDir = Paths.get(uploadDir).toAbsolutePath().normalize();

        //폴더가 없으면 자동 생성
        Files.createDirectories(this.uploadDir);
    }
    public String storeFile(MultipartFile file) throws IOException {
        if (file == null || file.isEmpty()) {
            return null;
        }

        String originalFilename = StringUtils.cleanPath(file.getOriginalFilename());

        String ext = "";
        int dotIndex = originalFilename.lastIndexOf(".");
        if (dotIndex != -1) {
            ext = originalFilename.substring(dotIndex);
        }

        String storedFilename = UUID.randomUUID().toString() + ext;

        // 항상 우리가 정한 uploadDir 밑에 저장
        Path target = this.uploadDir.resolve(storedFilename);

        // 혹시 모를 상위 폴더 생성 (대부분은 이미 있음)
        Files.createDirectories(target.getParent());

        file.transferTo(target.toFile());

        return storedFilename;
    }
```

| 코드 | 설명 |
|---|---|
| this.uploadDir = Paths.get(uploadDir)... | 설정값을 Path 객체로 변환 |
| Files.createDirectories(...) | 저장 폴더가 없으면 자동 생성 |

| 메서드명 | 입력 | 출력 | 역할 |
|---|---|---|---|
| storeFile() | MultipartFile file | String storedFilename | 파일 저장 후 저장 파일명 반환 |
| loadFileAsResource() | String storedFilename | Resource | 저장된 파일을 읽어 Resource로 반환 |

```java
    public Resource loadFileAsResource(String storedFilename) throws MalformedURLException {
        Path file = uploadDir.resolve(storedFilename).normalize();
        Resource resource = new UrlResource(file.toUri());
        if (resource.exists()) {
            return resource;
        } else {
            throw new RuntimeException("파일을 찾을 수 없습니다: " + storedFilename);
        }
    }
}
```

# FileStorageService.java-계속

* storeFile() : 업로드 파일 저장 및 DB에 저장될 파일명 제공

| 단계 | 기능 | 사용 필드/값 |
|---|---|---|
| 파일 검증 | null 또는 empty 체크 | file |
| 원본 파일명 정리 | 안전한 파일명으로 변경 | StringUtils.cleanPath() |
| 확장자 추출 | 마지막 "." 기준 | ext |
| 저장 파일명 생성 | UUID + 확장자 | storedFilename |
| 저장 경로 계산 | uploadDir + 파일명 | Path target |
| 디렉터리 확인 | 필요시 폴더 생성 | Files.createDirectories() |
| 파일 저장 | 실제 파일 저장 | file.transferTo() |
| 반환 값 | 저장된 파일명 반환 | storedFilename |

* loadFileAsResource() :다운로드 시 파일 읽기

| 단계 | 기능 | 설명 |
|---|---|---|
| 경로 계산 | 저장 폴더 + 파일명 | uploadDir.resolve() |
| 리소스 생성 | UrlResource 생성 | new UrlResource(file.toUri()) |
| 파일 존재 확인 | 존재하면 반환 | 성공 시 Resource 반환 |
| 예외 처리 | 없으면 오류 발생 | RuntimeException |

# 문제해결

- 게시판과 자료실이 병합된 커뮤니티 만들기
- AI도구 활용 및 제출물
  - AI 도구명
  - 질문 프롬프트
  - 실행소스와 결과