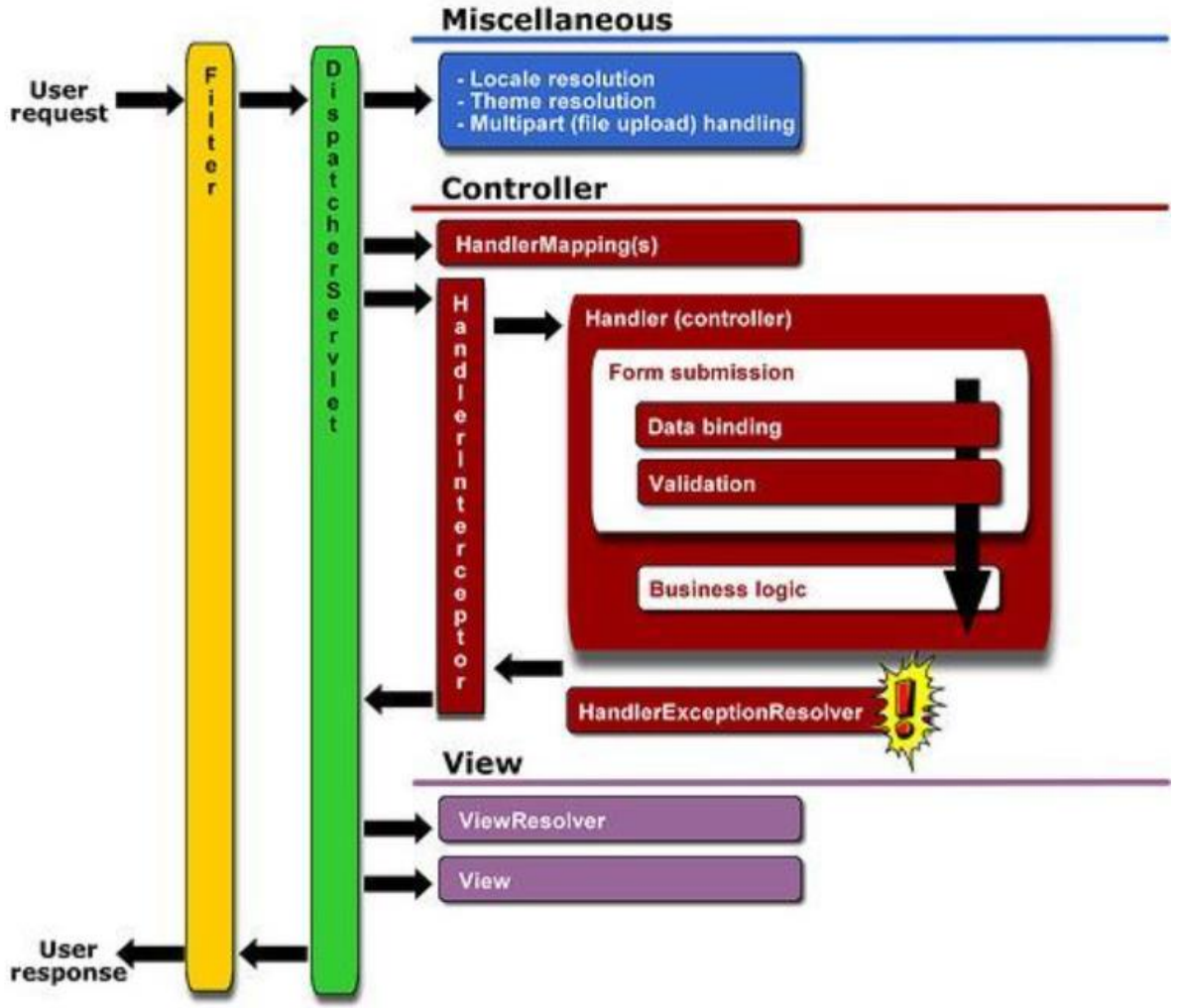

spring

필터와 인터셉터

- 어떤 URI를 호출했을 때, 해당 요청의 컨트롤러가 처리되기 전과 후에 특정 작업을 하기 위해서는 필터 또는 인터셉터를 사용할 수 있다.
- 기능적인 면에서 이 둘은 비슷해보이나, 차이가 있다.



필터와 인터셉터

- 팔터는 DispatcherServlet 앞 단에서 동작하지만 인터셉터는 DispatcherServlet에서 컨트롤러로 가기 전에 동작한다.
- 필터는 서블릿의 기능 중 일부이지만, 인터셉터는 스프링 프레임워크에서 제공되는 기능이다. 즉, 인터셉터에서만 스프링 빈을 사용할 수 있다.

- 일반적으로 웹 전반에서 사용되는 기능(문자열 인코딩 등)은 필터로 구현하고,
  클라이언트 요청과 관련이 있는 처리(로그인 인증, 권한 처리 등)은 인터셉터로 처리한다.

- 필터 : 서버와 클라이언트가 주고 받는 HTTP 요청과 응답에 대해 일괄적인 전처리/후처리
- 인터셉터 : Controller와 View page 사이에 오가는 신호를 처리

인터셉터 구현하기

- 스프링에서 인터셉터는 HandlerInterceptor 인터페이스를 구현하여 사용할 수 있다.

- HandlerInterceptor 는 세 가지 메서드가 제공된다.

메소드	역할
preHandle	컨트롤러 실행 전에 수행
postHandle	컨트롤러 수행 후 결과를 뷰로 보내기 전에 수행
afterCompletion	뷰의 작업까지 완료된 후 수행

인터셉터 사용하기

- 각 요청의 시작과 끝을 보여주는 로그를 출력해주는 인터셉터를 작성해보자.

```
package com.board.interceptor;

@Slf4j
public class LoggerInterceptor implements HandlerInterceptor {
	@Override
	public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
	throws Exception {
		log.info("============================ 시작 ============================");
		log.info(" Request URI \t: {}", request.getRequestURI());
		return HandlerInterceptor.super.preHandle(request, response, handler);
	}

	@Override
	public void postHandle(HttpServletRequest request, HttpServletResponse response,
	Object handler, ModelAndView modelAndView) throws Exception {
		log.info("============================ 끝 ============================\n");
		HandlerInterceptor.super.postHandle(request, response, handler, modelAndView);
	}
}
```

인터셉터 사용하기

- 이제 앞에서 만든 LoggerInterceptor를 스프링 빈으로 등록해주기 위한 설정을 해보자.

```
package com.board.configuration;

@Configuration
public class WebMvcConfiguration implements WebMvcConfigurer {
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new LoggerInterceptor());
    }
}
```

트랜잭션 사용하기

- 트랜잭션이란 데이터베이스의 상태를 변화시킬 때, 더 이상 분리할 수 없는 작은 작업의 단위를 의미한다.

- 하나의 트랜잭션에서 일련의 작업이 처리되어야 한다.
- 즉, 되려면 모두 다 되어야 하고, 하나라도 안 된다면 모두 다 안되어야 한다.

```
@Override
public BoardDTO selectBoardById(int id) throws Exception {
    boardMapper.updateHit(id);
    int i = 10 / 0;
    return boardMapper.selectBoardById(id);
}
```

```
@Service
@Transactional
public class BoardServiceImpl ...
```

트랜잭션 사용하기

- 트랜잭션이란 데이터베이스의 상태를 변화시킬 때, 더 이상 분리할 수 없는 작은 작업의 단위를 의미한다.

- 하나의 트랜잭션에서 일련의 작업이 처리되어야 한다.
- 즉, 되려면 모두 다 되어야 하고, 하나라도 안 된다면 모두 다 안되어야 한다.

예외 처리하기

- 스프링 프레임워크에서 예외를 처리하는 방법은 두 가지로 나눌 수 있다.

1. 각각의 컨트롤러에서 @ExceptionHandler를 이용한 예외처리
2. @ControllerAdvice를 이용한 전역 예외처리

예외 처리하기

- @ExceptionHandler의 경우는 컨트롤러별로 각 예외 처리를 추가해야하기 때문에 코드의 중복이 생길 수 있다.

```
@ExceptionHandler(Exception.class)
public ModelAndView handleException(Exception e) {
    Log.error("예외 발생: ", e);
    // src/main/resources/templates/board/error.html
    ModelAndView mv = new ModelAndView("board/error");
    mv.addObject("errorMessage", e.getMessage());
    return mv;
}
```

예외 처리하기

- @ControllerAdvice를 이용한 전역 예외처리를 해보자.

```
package com.board.common;

@Slf4j
@ControllerAdvice // 예외처리 클래스임을 명시
public class ExceptionHandlers {
    @ExceptionHandler(Exception.class)
    public ModelAndView defaultExceptionHandler(Exception e) {
        log.error("예외 발생: ", e);
        // src/main/resources/templates/board/error.html
        ModelAndView mv = new ModelAndView("board/error");
        mv.addObject("errorMessage", e.getMessage());
    return mv;
}
```

파일 업로드와 다운로드

- 파일 첨부를 위한 DB 설정

```sql
CREATE TABLE file_tbl (
    id INT PRIMARY KEY AUTO_INCREMENT COMMENT '파일 번호',
    board_id INT NOT NULL COMMENT '게시글 번호',
    origin_file_name VARCHAR(255) NOT NULL COMMENT '원본 파일명',
    stored_file_path VARCHAR(255) NOT NULL COMMENT '파일 저장 경로',
    file_size INT NOT NULL COMMENT '파일 크기',
    creator VARCHAR(50) NOT NULL COMMENT '작성자',
    created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '작성일',
    updated_at DATETIME DEFAULT NULL ON UPDATE CURRENT TIMESTAMP COMMENT '수정일',
    is_deleted CHAR(1) NOT NULL DEFAULT 'n' COMMENT '삭제 여부'
);
```

파일 업로드와 다운로드 - 업로드

- 파일 첨부를 위한 글쓰기 html 변경

```
<form action="insert" method="post" enctype="multipart/form-data">


<div class="mb-3">
    <label for="files" class="form-label">파일</label>
    <input type="file" class="form-control" name="files" id="files" multiple />
</div>
```

파일 업로드와 다운로드 - 업로드

- Controller 변경

```
@PostMapping("/insert")
public String boardInsert(BoardDTO boardDTO, @RequestParam("files")
List<MultipartFile> files) throws Exception {
    bs.insertBoard(boardDTO, files);
    return "redirect:/board/list";
}
```

- Service 인터페이스 메소드 변경

```
void insertBoard(BoardDTO boardDTO, List<MultipartFile> files) throws Exception;
```

파일 업로드와 다운로드 - 업로드

- ServiceImpl 메소드 변경

```
@Override
public void insertBoard(BoardDTO boardDTO, List<MultipartFile> files) throws
Exception {
    boardMapper.insertBoard(boardDTO);
    System.out.println(boardDTO.getId());
}
```

파일 업로드와 다운로드 - 업로드

- Mapper파일 변경

```
<insert id="insertBoard" parameterType="com.board.dto.BoardDTO" useGeneratedKeys="true" keyProperty="id">
     INSERT INTO
          board_tbl
     (title, content, created_at, creator)
          VALUES
     (#{title}, #{content}, NOW(), "최인규")
</insert>
```

```java
boardMapper.insertBoard(boardDTO);
if (files != null && !files.isEmpty()) {
    List<FileDTO> fileList = new ArrayList<>();
    for (MultipartFile file : files) {
        if (!file.isEmpty()) {
            String originalFilename = file.getOriginalFilename();
            String storedFileName = UUID.randomUUID().toString() + "_" + originalFilename;
            String storedFilePath = "C:\\Users\\Administrator\\" + storedFileName;
            long fileSize = file.getSize();

            FileDTO fileDTO = new FileDTO();
            fileDTO.setBoardId(boardDTO.getId());
            fileDTO.setFileSize(fileSize);
            fileDTO.setOriginFileName(originalFilename);
            fileDTO.setStoredFilePath(storedFilePath);
            fileList.add(fileDTO);
            try {
                File dest = new File(storedFilePath);
                file.transferTo(dest); // 파일 저장
            } catch (IOException e) {
                throw new Exception("파일 업로드 중 오류가 발생했습니다.");
            }
        }
    }

    if (!fileList.isEmpty()) {
        boardMapper.insertBoardFile(fileList);
    }
}
```

파일 업로드와 다운로드 - 업로드

- Mapper 인터페이스 메소드 추가

```
void insertBoardFile(List<FileDTO> fileList) throws Exception;
```

- 매퍼 파일에 insert 구문 추가

```
<insert id="insertBoardFile" parameterType="com.board.dto.FileDTO">
    INSERT INTO
        file_tbl (board_id, origin_file_name, stored_file_path, file_size, creator)
    VALUES
        <foreach collection="list" item="item" separator=",">
            (#{item.boardId}, #{item.originFileName}, #{item.storedFilePath}, #{item.fileSize}, "최인규")
        </foreach>
</insert>
```

파일 업로드와 다운로드 - 다운로드

- BoardDTO 변경

```
@Data
public class BoardDTO {
    private int id;
    private String title, content;
    private int hit;
    private LocalDateTime createdAt;
    private Simplementation 'commons-io:commons-io:2.16.1'
    private LocalDateTime updatedAt;
    private List<FileDTO> fileList;
}
```

파일 업로드와 다운로드 - 다운로드

- ServiceImpl 메소드 변경

```
@Override
public BoardDTO selectBoardById(int id) throws Exception {
    BoardDTO boardDTO = boardMapper.selectBoardById(id);
    List<FileDTO> fileList = boardMapper.selectFilesByBoardId(id);
    boardDTO.setFileList(fileList);
    boardMapper.updateHit(id);
    return boardDTO;
}
```

- Mapper 인터페이스 추가

```
List<FileDTO> selectFilesByBoardId(int id) throws Exception;
```

파일 업로드와 다운로드 - 다운로드

- 매퍼 파일에 추가

```
<select id="selectFilesByBoardId" parameterType="int" resultType="com.board.dto.FileDTO">
    SELECT
        id, board_id, origin_file_name, FORMAT(ROUND(file_size / 1024), 0) AS file_size
    FROM
        file_tbl
    WHERE
        board_id = #{id}
        AND
        is_deleted = 'n'
</select>
```

파일 업로드와 다운로드 - 다운로드

- detail 뷰 추가

```
<div class="mb-3">
    <a
        class="mx-2"
        th:each="list : ${board.fileList}"
        th:href="@{download(id=${list.id}, boardId=${list.boardId})}"
        th:text="|${list.originFileName}(${list.fileSize} kb)|">
    </a>
</div>
```

파일 업로드와 다운로드 - 다운로드

- 다운로드 컨트롤러

```
@RequestMapping("/download")
public ResponseEntity<Resource> downloadFile(@RequestParam("id") int id,
@RequestParam("boardId") int boardId, HttpServletResponse response) throws Exception {
     FileDTO fileDTO = bs.selectFileById(id, boardId);
     String fileName = fileDTO.getOriginFileName();
     UrlResource resource;

     try{
          resource = new UrlResource("file:"+ fileDTO.getStoredFilePath());
     } catch (Exception e){
          throw new Exception("파일 다운로드 에러");

     }
     String encodedOriginalFileName = UriUtils.encode(fileName, StandardCharsets.UTF_8);
     String contentDisposition = "attachment; filename=\"" + encodedOriginalFileName + "\"";

     return ResponseEntity
          .ok()
          .header(HttpHeaders.CONTENT_DISPOSITION,contentDisposition)
          .body(resource);
}
```

파일 업로드와 다운로드 - 다운로드

- 서비스 인터페이스

```
FileDTO selectFileById(int id, int boardId) throws Exception;
```

- 서비스 구현 클래스

```
@Override
public FileDTO selectFileById(int id, int boardId) throws Exception {
    return boardMapper.selectFileById(id, boardId);
}
```

- 매퍼 인터페이스

```
FileDTO selectFileById(@Param("id") int id, @Param("boardId") int boardId) throws Exception;
```

파일 업로드와 다운로드 - 다운로드

- 매퍼파일

```
<select id="selectFileById" parameterType="map" resultType="com.board.dto.FileDTO">
    SELECT
        origin_file_name, stored_file_path, file_size
    FROM
        file_tbl
    WHERE
        id = #{id}
        AND
        board_id = #{boardId}
        AND
        is_deleted = 'n'
</select>
```