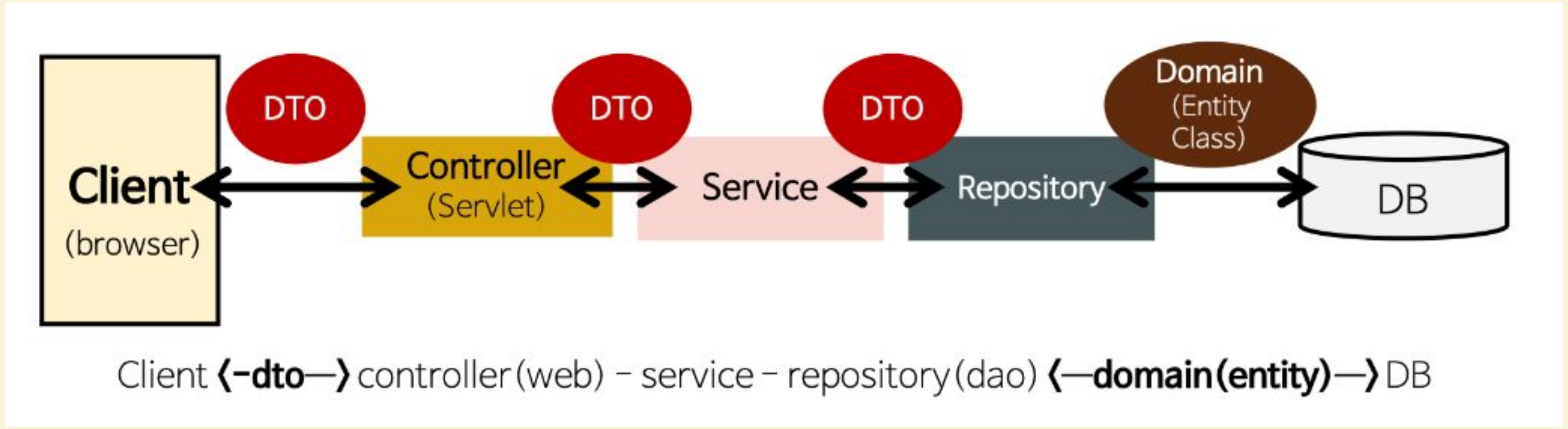
스프링부트로 RestFulAPI 구현하기

8장 자유게시판 구현하기

- 글조회 get /freeboards/1
- 글저장 post /freeboards
- 글수정 put /freeboards/1
- 글삭제 delete /freeboard/1

박명회

8장 controller service respository entity



컨트롤러 : @Controller (프레젠테이션 레이어, 웹 요청과 응답을 처리함)

로직 처리 : @Service (서비스 레이어, 내부에서 자바 로직을 처리함)

외부I/O 처리 : @Repository (퍼시스턴스 레이어, DB나 파일같은 외부 I/O 작업을 처리함)

8장 @EnableJpaAudting

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@SpringBootApplication
@EnableJpaAuditing
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

8장 BoardRequestDto

```
@Getter
@Setter
public class BoardRequestDto {

    @NotBlank(message = "Title is mandatory")
    @Size(min = 3, max = 100, message = "Title must be between 3 and 100 characters")
    private String title;

    @NotBlank(message = "Content is mandatory")
    @Size(min = 10, max = 1000, message = "Content must be between 10 and 1000 
characters")
    private String content;

    @NotBlank(message = "Author is mandatory")
    @Size(min = 3, max = 50, message = "Author must be between 3 and 50 characters")
    private String author;
}
```

8장 BoardResponseDto

```
@Getter
@Setter
public class BoardResponseDto {
    private Long id;
    private String title;
    private String content;
    private String author;
    private LocalDateTime createdDate;
    private LocalDateTime updatedDate;
}
```

8장 Board entity 작성하기

```
@Entity
@Table(name = "boards")
@EntityListeners(AuditingEntityListener.class)
@Getter
@Setter
@NoArgsConstructor
public class Board {

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

    @Column(nullable = false, length = 100)
    private String title;

    @Column(nullable = false, length = 1000)
    private String content;

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

8장 Board entity 작성하기

```
    @CreatedDate
    @Column(updatable = false)
    private LocalDateTime createdDate;

    @LastModifiedDate
    private LocalDateTime updatedDate;

    @CreatedBy
    @Column(updatable = false)
    private String createdBy;

    @LastModifiedBy
    private String modifiedBy;

    @Builder
    public Board(String title, String content, String author) {
        this.title = title;
        this.content = content;
        this.author = author;
    }
}
```

8장 BoardRepository 생성

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface BoardRepository extends JpaRepository<Board, Long> {

}
```

8장 BoardService

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

@Configuration
public class AppConfig {

    @Bean
    public ModelMapper modelMapper() {
        return new ModelMapper();
    }
}
```

```
implementation 'org.modelmapper:modelmapper:3.2.1'
```

8장 BoardService

```
import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

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

@Service
public class BoardService {

    private final BoardRepository boardRepository;
    private final ModelMapper modelMapper;

    public List<BoardResponseDto> findAll() {
        return boardRepository.findAll().stream()
                .map(board -> modelMapper.map(board, BoardResponseDto.class))
                .collect(Collectors.toList());
    }
```

8장 BoardService

```
    public Optional<BoardResponseDto> findById(Long id) {
        return boardRepository.findById(id)
                .map(board -> modelMapper.map(board, BoardResponseDto.class));
    }

    public BoardResponseDto save(BoardRequestDto boardRequestDto) {
        Board board = modelMapper.map(boardRequestDto, Board.class);
        Board savedBoard = boardRepository.save(board);
        return modelMapper.map(savedBoard, BoardResponseDto.class);
    }

    public void deleteById(Long id) {
        boardRepository.deleteById(id);
    }
}
```

8장 BoardController

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/boards")
@Validated
public class BoardController {

    private final BoardService boardService;

    @Autowired
    public BoardController(BoardService boardService) {
        this.boardService = boardService;
    }
```

8장 BoardController

```
    @GetMapping
    public List<BoardResponseDto> getAllBoards() {
        return boardService.findAll();
    }

    @GetMapping("/{id}")
    public ResponseEntity<BoardResponseDto> getBoardById(@PathVariable Long id) {
        return boardService.findById(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public ResponseEntity<BoardResponseDto> createBoard(@Valid @RequestBody
BoardRequestDto boardRequestDto) {
        BoardResponseDto createdBoard = boardService.save(boardRequestDto);
        return ResponseEntity.ok(createdBoard);
    }
```

8장 BoardController

```
    @PutMapping("/{id}")
    public ResponseEntity<BoardResponseDto> updateBoard(@PathVariable Long id, @Valid
@RequestBody BoardRequestDto boardRequestDto) {
        return boardService.findById(id)
                .map(board -> {
                    Board updatedBoard = Board.builder()
                            .id(id)
                            .title(boardRequestDto.getTitle())
                            .content(boardRequestDto.getContent())
                            .author(boardRequestDto.getAuthor())
                            .build();
                    BoardResponseDto updatedBoardDto = boardService.save(boardRequestDto);
                    return ResponseEntity.ok(updatedBoardDto);
                })
                .orElse(ResponseEntity.notFound().build());
    }
```

8장 BoardController

```
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBoard(@PathVariable Long id) {
        return boardService.findById(id)
                .map(board -> {
                    boardService.deleteById(id);
                    return ResponseEntity.noContent().build();
                })
                .orElse(ResponseEntity.notFound().build());
    }
}
```

감사합니다