## ****Documentation for Movie Components****

### MovieController.java

**Purpose:**  
MovieController handles HTTP operations related to movies, interacting with the MovieService and returning a ResponseModel<T>.

**Responsibilities:**

Accept validated input via @Valid DTOs.

Use path variables to fetch or modify specific records.

Handle creation, update, retrieval, listing, and deletion of movie data.

**Defined Endpoints:**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/movies | POST | Create new movie |
| /api/movies/{id} | PUT | Update existing movie by ID |
| /api/movies/{id} | GET | Get movie by ID |
| /api/movies | GET | Get all movies |
| /api/movies/{id} | DELETE | Delete movie by ID |

**Key Annotations:**

@RestController, @RequestMapping, @PostMapping, etc.

@Validated, @Valid, @PathVariable

@Autowired to inject MovieService

**Test Coverage Connection:**  
Each public method in this controller is directly and fully covered in MovieControllerTest via HTTP simulation and service mocking.

### MovieControllerTest.java

**Purpose:**  
Unit test class to verify MovieController behavior using Spring Boot’s @WebMvcTest and MockMvc. It mocks the service layer to test only the controller's logic and response handling.

**Tools & Techniques Used:**

@WebMvcTest(MovieController.class) – Loads only the MovieController bean.

MockMvc – Simulates HTTP requests to controller endpoints.

Mockito – Mocks the MovieService.

ObjectMapper – Converts Java objects to and from JSON.

**Tested Endpoints:**

| **HTTP Method** | **Endpoint** | **Purpose** |
| --- | --- | --- |
| POST | /api/movies | Create a new movie |
| PUT | /api/movies/{id} | Update an existing movie |
| GET | /api/movies/{id} | Retrieve movie by ID |
| GET | /api/movies | Retrieve all movies |
| DELETE | /api/movies/{id} | Delete movie by ID |

**Coverage Provided:**

Confirms correct HTTP status responses (201, 200).

Verifies request/response JSON structure using jsonPath.

Confirms controller correctly delegates to the mocked MovieService.

Ensures endpoint-specific logic (UUID path params, body validation).

Uses verify(...) to confirm service methods were invoked.

### MovieDTO.java

**Purpose:**  
A Data Transfer Object that encapsulates movie data for API communication, including conversion utilities for entity mapping.

**Fields:**

UUID id

String title – Movie title (required)

String description – Short movie summary

String genre – Genre of the movie

LocalDate releaseDate – Movie release date

UUID directorId – Associated director ID (required)

DirectorDTO directorDTO – Nested DTO for full info

**Validation Rules:**

@NotBlank on title

@NotNull on releaseDate and directorId

**Conversion Utilities:**

convertToMovieDTO(Movie) – Maps Movie entity → DTO

convertToMovie(MovieDTO, Director) – Maps DTO + Director → Entity

**Test Coverage Connection:**  
DTO fields are included in JSON request/response content and validated within controller test cases.

## Summary Table

| **Class** | **Type** | **Tested Components** | **Mocked Dependencies** | **Validations Tested** | **Responsibilities** |
| --- | --- | --- | --- | --- | --- |
| MovieController | Controller | All REST endpoints for movies | Yes (MovieService) | Yes (@Valid, etc.) | Handles API requests and response building |
| MovieControllerTest | Test Class | All endpoints and paths tested | Yes | Yes (implicit) | Verifies HTTP status, payloads, interactions |
| MovieDTO | DTO | Used in requests/responses | N/A | Yes (@NotNull) | Data transfer and conversion logic |