## ****Documentation for Director Components****

| **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /api/directors | Create a new movie director |
| PUT | /api/directors/{id} | Update an existing director by ID |
| GET | /api/directors/{id} | Get a specific director by UUID |
| GET | /api/directors | Retrieve a list of all directors |
| DELETE | /api/directors/{id} | Delete a director by UUID |

### DirectorController.java

**Purpose:** DirectorController is the REST layer that handles all director-related operations and maps HTTP requests to DirectorService methods.

**Responsibilities:**

Validate and route client input using @Valid, @RequestBody, and @PathVariable.

Wrap responses in ResponseModel<T> format.

Translate service-layer responses into HTTP responses with appropriate status codes.

**Endpoints Exposed:**

POST /api/directors – Create a new director

PUT /api/directors/{id} – Update an existing director

GET /api/directors/{id} – Fetch director by ID

GET /api/directors – Fetch all directors

DELETE /api/directors/{id} – Delete director by ID

**Spring Annotations:**

@RestController, @RequestMapping, @PostMapping, etc.

@Validated, @Valid

@Autowired for service injection

**Test Coverage Connection:**

Each controller method is directly covered in DirectorControllerTest via HTTP simulations and JSON validation.

### DirectorService.java

**Purpose:**  
This is a **service interface** that defines the contract for all business operations related to the **Director** entity. It declares method signatures for creating, updating, retrieving, and deleting directors, ensuring a clean separation of concerns between the controller and service layers. This interface promotes **loose coupling** and enables easy unit testing and future extensibility of the application.

### DirectorServiceImpl.java

**Purpose:**  
This is the **concrete implementation** of the DirectorService interface. It contains the actual **business logic** for handling operations on the **Director** entity, such as interacting with the data access layer (e.g., repositories), performing validations, and managing transactions. By implementing the service interface, it provides a modular and testable structure that adheres to the **Service Layer** pattern in enterprise application architecture.

### DirectorControllerTest.java

**Purpose:**  
This unit test class verifies the behavior of the DirectorController by isolating it from other layers using Spring’s @WebMvcTest and MockMvc. It uses mocked responses from DirectorService to focus solely on controller logic.

**Tools Used:**

@WebMvcTest(DirectorController.class) – to restrict context to only DirectorController

MockMvc – to simulate HTTP requests

Mockito – to mock DirectorService

ObjectMapper – for JSON conversion

**Tested Endpoints:**

| **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /api/directors | Create a new director |
| PUT | /api/directors/{id} | Update an existing director |
| GET | /api/directors/{id} | Retrieve a director by ID |
| GET | /api/directors | List all directors |
| DELETE | /api/directors/{id} | Delete a director by ID |

**Test Coverage Provided:**

Validates successful creation and update with correct JSON responses.

Verifies UUID-based retrieval, listing, and deletion.

Confirms correct HTTP statuses (200 OK, 201 Created).

Uses jsonPath assertions to verify response fields.

Ensures service method calls were triggered correctly using verify.

### DirectorDTO.java

**Purpose:**  
A DTO class that acts as a bridge between the API and internal entity layers for handling director data.

**Fields:**

UUID id – Unique identifier for the director

String name – Name of the director (required)

String bio – Biography of the director (optional)

**Validation Rules:**

@NotBlank(message = "Name is mandatory") on name to ensure non-empty input

**Utility Methods:**

convertToDirectorDTO(Director) – Maps from entity to DTO

convertToDirector(DirectorDTO) – Maps from DTO to entity

**Test Coverage Connection:**

DTO fields are populated in DirectorControllerTest as JSON request/response bodies.

Validation of the name field is indirectly confirmed in test cases through successful requests.

Summary Table

| **Class** | **Type** | **Tested Components** | **Mocked Dependencies** | **Validations Tested** | **Responsibilities** |
| --- | --- | --- | --- | --- | --- |
| DirectorController | Controller | All REST endpoints | Yes (DirectorService) | Yes (@Valid, @NotBlank) | API request processing and delegation |
| DirectorControllerTest | Test Class | All controller methods | Yes | Partially | Verifies controller behavior |
| DirectorDTO | DTO | Used in requests/responses | N/A | Yes (@NotBlank) | Data encapsulation and conversion |