Create a `Movie` Entity

Objective:

Familiarize yourself with creating an entity and mapping to a database table.

1. Create a new Spring Boot project with `Spring Web`, `Spring Data JPA` and `MySQL` database dependencies. Use Spring Initializr
2. Inside of phpMyAdmin, create a database called `movie_studio`. The database `movie_studio` should contain a table called `movies`. The `movies` table has to have these columns:
    - `id` (`BIGINT`, `PRIMARY KEY`, `AUTO_INCREMENT`)
    - `title` (`VARCHAR(100)`, `NOT NULL`)
    - `director` (`VARCHAR(50)`, `NOT NULL`)
3. Define an entity `Movie` in the Spring Boot app, with the following fields (class should be created in `model` package):
    - Look at the database table `movies` and convert the columns to class fields in Java, by yourself
        - Use this link for reference
    - Don't forget constructor, getters and setters
        - You should also add a parameterless constructor
4. Annotate the `Movie` class with `@Entity`, as well as `@Table(name = "movies")`, and configure the primary key using `@Id` and `@GeneratedValue(strategy = GenerationType.IDENTITY)`
5. Create a `repository` package. In it, create a `MovieRepository` interface. It should extend `JpaRepository<Movie, Long>`
6. Create a `service` package. In it, create a `MovieService` class. The class should have a field `MovieRepository`, it should be final. Use constructor injection to wire it with the class, don't forget `@Autowired` annotation
7. Inside `MovieService` class, create a `public List<Movie> findAllMovies()` method. It should call the `findAll()` method of `MovieRepository`, which is injected as a field
8. Run the application and verify that Spring Boot starts successfully
9. Manually add some rows to the table `movies`, in phpMyAdmin
10. In `MovieController` class, add a field `MovieService`, it should be final. Use constructor injection to wire it with the class, don't forget `@Autowired` annotation

11. Add a `getMovies` controller with `GET` endpoint in `MovieController` class:
    - The method is similar to what you have created yesterday
    - Don't forget to use `ResponseEntity<List<Movie>>` as return type
    - The controller should call the `findAllMovies()` method of `MovieService` class object, that is, field
12. If successful, you should get a list of movies from the database, when calling the endpoint from a client