# Testing REST-endpoints

This exercise continues with the movie database you started day-1.

If you're not done yet, take a look at this [project](https://github.com/HartmannSolution/movie_service_start) for hints on how to complete it to a stage where you can start to test. The sample Movie Class inserted at the end of yesterday's exercise is taken from this project. Don’t clone the project but use it as inspiration to complete your own solution

Make sure to include this endpoint: **api/movie/all**

It should return a JSON-response like in the example given below (if you “invented” your own fields for the Movie class, just change the response to match your own ideas)

[

{

"id": 1,

"year": 1934,

"name": "De døde heste",

"actors": [

"Ulla Tørnæse",

"Pia Køl",

"Freddy Frøstrup"

]

},...]

Hint: Actors was made with a String array of names on the Movie entity.

Setup a few test movies before each test (using the @BeforeEach annotation), similar to how it was done in the initial version of the start code.

Create the following tests using RestAssured and Hamcrest:

1. Create a test that verifies that the server is up (similar to the “Hello World” response)
2. Create a test for the endpoint: **api/movie/count** (expected result, depends on how many movies you created before each test ).
3. Create a test for the endpoint **api/movie/all** and assert that the body contains an actor named Freddy Frøstrup (or just an actor added by your BEFORE code)
4. Create a test for an endpoint: **api/movie/name/{name}**. Use a name you know exists, and (for red students) also try with a name that does not exist (obviously this requires that you know what you return in such a case)
5. Create a test for the an endpoint: **api/movie/{id}** and verify that you get the expected Movie