- Duplicate yesterday's project, and continue here
- Add various types of field validation to the `Movie` class
  - The validation should be similar, even better, than in database table. For example, if the column `title` in database is `VARCHAR(120)` and `NOT NULL`, you should use `@Size(min = 2, max = 120)` and `@NotNull` on the field in the class
    - Minimum length of 2 does not exist in the database column `title`, but that does not mean you should not add it to the `Movie` class `title` field - this ensures no faulty data gets to the database
  - Add validation of your liking to the `director` field, but you need to incorporate `@Pattern` regular expression validation, too
    - Ensure that the user can only send a `director` if it starts with an uppercase letter, and does not contain any numbers
  - What about validating the fields that contain a list of objects? Is it possible? Try it out! Use validation annotations that make sense to you
- End result:
  - Test out the `POST` and `PUT` controllers, to see whether or not the validation actives. Do not worry if the validation message is a bit ugly, this will get fixed in the upcoming exercise
    - Also, don't forget to remove the `if` statements from these controllers, if the annotations already validate the same thing