Reusable "components" with AngularJS

General part

Explain generally about:

- Mechanisms for reuse in Java.
- Mechanisms for reuse with Angular
- Mechanisms for reuse in general
- Mechanisms for reusing ideas

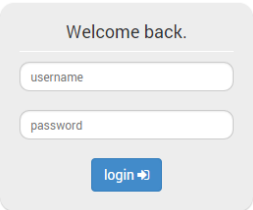
Practical part

Start Code for this exercise: *You should use the project found in **reausableControlsAngular.zip** as the starting point for this exercise. It is a basic HTML-5 project (since there is no backend code involved in this exercise), with support for Karma/Jasmine-testing[1]. If you place all your JavaScript in app.js (or in the same folder) and all your Test-code in tests.js (or the same folder) your test-cases should be recognized.*

In angular we have some very useful possibilities for creating reusable source code. Here you must create examples of reusable code using filters, directives, and services/factories.

1. Create a filter that will produce the name of a person in the form `"lastname, firstname"`.
   Given a person object on the form `{ firstName: 'Peter', lastName: 'Smith' }`
   it should output `"Smith, Peter"` when used like this `{{ person | name }}`
   Create an angular application that shows the filter in use.

2. Create a directive named login-form that encapsulate the following:

   You do not need to implement the same style as in the figure, just the functionality. It must be possible to use the directive with the following html snippet:

   ```
   <login-form></login-form>
   ```

   You should provide an angular application that shows the directive in use. If you have time provide a header attribute so the header can be defined in html:

   ```
   <login-form header="Welcome back."></login-form>
   ```

3. Create a service or factory which provide three functions
   - titleCase("my example service") should return: "My Example Service",
   - camelCase("my example service") should return: "MyExampleService",
   - dashCase("my example service") should return: "my-example-service".

4. Create a controller that uses the service and an angular application that shows all functions in use.

5. To verify the behaviour of the Controls implemented above you should write the following unit tests, using Jasmine and Karma:
   - Write a unit test that tests the filter implemented in 1)
   - Write a unit test that tests the service implemented in 3)
   - Write a unit test that test the controller implemented in 4

---

[1] Requires you to do "resolve project dependencies" which again require that you have installed node.JS (https://nodejs.org/en/download/)