Single Origin Policy, and CORS with Java and JAX-RS

1) CORS with Java and Jax-rs

The Backend

Create a new Netbeans Maven web project, and use the wizard to create a new REST-endpoint (remember to include the necessary dependencies). The API should be the simplest you can come up with, for example, you could use a single static variable to hold an object like this: { "name" : "Peter Pan"} and then build a "full CRUD API" around this.

Test the API, using POSTMAN. This should not cause any CORS-related problems, WHY?

The Frontend

Create (again, the simplest you can come up with) a JavaScript frontend that uses this API. DO NOT, add your HTML+JavaScript code to the same project, as above since it will then be hosted on the same Origin. One way to ensure it runs on a separate origin could be to use our simple SPA-start code project (clone from here: https://github.com/Cphdat3sem2018f/code_simple_SPA.git). If you go for this suggestion, deployment of the front-end can be done in less than 30 seconds :-)

Does this work?, can you use the API, from your new "SPA-client"? If not explain why (the errors you get).

Using CORS with Jax-rs

Add the necessary CORS headers to your backend to solve the problem. You can do this using JAX-RS filters as demonstrated here (three slides). *Note: this example sets CORS headers for all your endpoints, which is not always what you want, and it probably also relaxes the SOP, more than what is necessary. You are however free to use this template for the rest of the semester to solve SOP-problems*.

Verify that you can use the full API. Monitor all request, especially POST, and make sure you can explain all the request made by the browser, also those which are not a GET, POST, PUT or DELETE request.

2) CORS with Java and Jax-rs for a "real" project

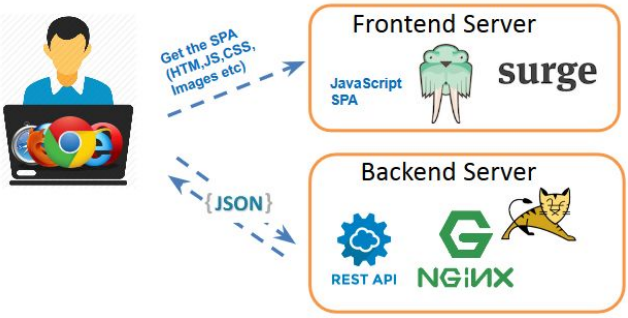
Use one of your projects from last week (the most mature one) for this exercise.
If you already have created a JavaScript frontend, move this into another server (for example the one use for exercise 1) to ensure different Origins.
If not, create a simple JavaScript frontend that must use your API.

Solve SOP-problems using CORS, and monitor and explain the CORS-headers used in the communication between the two ends.

3) Deploy a SPA on separate Servers

To simplify “matters”, we suggest this deployment strategy:

- Deploy your backend (REST-API) as usually on Tomcat on a droplet on Digitalocean.
- For the frontend, use Surge, because it’s simple. Once installed, you can deploy your frontend in less than 30 seconds :-)



Backend

Deploy the backend you used for part-2 on a Digitalocean, and verify that you can access the API using POSTMAN. Make sure that your backend set’s the necessary CORS header before you deploy.

Deploy the frontend.

Note: this assumes you have used our suggested simple SPA-start code project for your client code.

Change the URL, used in your frontend to point to your DEPLOYED backend and verify that you can execute it locally, using your deployed backend.

In a terminal, in the root of your Client project, type **`npm run build`**. This will "build" your project into a folder ***`build`*** (verify this)

If not done in the class, open a terminal and type **`npm install -g surge`**
This will install the command line interface to surge, a free hosting server for static code

Still in the root of your project, type: **`surge --project ./build --domain A_DOMAIN_NAME.surge.sh`**

If this was your first time running surge, you'll be prompted to create an account. Add email and password, then hit enter. You’ll then see an output similar this:

```
# surge
    Surge - surge.sh
            email: <email>@<domain>.com
            token: **********************
      project path: path/to/my-project
```

That’s it. Your SPA Client is now hosted like
`A_DOMAIN_NAME.surge.sh` :-)

Verify that you can start the hosted client, and use the backend, using this deployed client.

Note: This way of deploying will be acceptable for the rest of the semester, but you will be provided with an alternative, which allows you to use your own domain name :-)