SYSC 4806 – Lab 6: the Client Side

Now it's time to provide a functional frontend for your AddressBook app!

But before we go to the instructions, some quick advice based on what I've noticed so far: the best use of the Spring tutorials that I've linked to so far is to read them thoroughly first (they're not that long anyway), and then to try to apply and reproduce on your own projects. Don't just download their code and try to modify it to suit your own purpose in a scattershot approach! You'll end up with a mess of irrelevant code on your hands and hard to debug problems. For example, I've seen students who had both a Maven project AND a Gradle project, and they were running one while changing the other.

Part 1: the more traditional client-side

(note: you may have already done some of this in past labs!)

Using HTML static pages and/or template views (with Thymeleaf for example), create web pages and/or forms to allow the user to create an address book, to populate it with buddy infos, and to list the current content of your address book. This means that you need to provide controllers on the server-side to handle the client requests and to render the relevant views. This tutorial may again be helpful.

Part 2: the Single Page Application (SPA) version

(note: again, you may already have done some of this already)

If you have provided Spring Data repositories for your AddressBook and BuddyInfo JPA models, (see this tutorial for example), then you already have a set of RESTful controllers for your application[1]. You can test them using a

[1] To keep things DRY, the controllers you wrote in Part 1 should probably just be delegating to the RESTful ones. The only added value should be to select and populate the view to render.

command line tool like `curl`, or, at least in the case of GET methods, directly in the browser by entering the full formed URL in the URL bar. The JSON objects returned by these controllers can be used by a Single Page Application on the client side.

So now you should be able to write the JavaScript functions that will make an AJAX call to the appropriate controller on the server, with a corresponding JavaScript callback function that will use the returned JSON response to modify and render the current HTML page. These functions can then be registered to the relevant submit action (a button press or other). This tutorial may or may not prove useful.

Ideally, this registration process should be done on load time. This way, if the browser doesn't support JavaScript, your client-side will still behave correctly as per Part 1. This is the "progressive" way!

Done? Are you a frontend wizard now? Show your work to the TA!