Exercise

Introduction

You have been given the task to add features to a prototype for people searching which was originally built as a proof of concept.

Technical details

The application you will be working on is a gradle project and can be found in the zipped file. It is a spring-boot app. Since you don't have access to our internal maven repository some dependencies are loaded from the external_dependencies folder to the project. Feel free to add any other dependencies you may need. You can already run the project from command line with: gradle clean bootRun then head on to http://localhost:8080 and you should see a page with a form.

- You are asked to follow best coding practices (e.g. separation of concerns).
- You only need to build some UI for “Requirement 2”. No need to make it look good.
- Use Spock for writing your unit tests (there are very simple tests in HardcodedListOfCustomersImplTest already). If you’ve never used Spock before, the following documentation should be enough for this exercise: http://spockframework.org/spock/docs/1.1/spock_primer.html
- Only unit tests are expected here but all functionalities should be covered

Current functionality

- The system has the concept of a customer. A customer is either a premium customer or a non
- paying customer.
- The system searches people data based on surname + postcode as search criteria
- The system return search results as a collection of records. Each record consist of:
  - Person details such as forename, middle name, surname, telephone
  - Address details such as build number, street, postcode and town
  - Indicator where the source data come from. A record can be sourced from 1 or many datasources (source types)
- The system return data from the following data sources (source types):
  - BT (British Telecom)
  - DNB (Duns & Bradstreet)
  - ELECTORAL_ROLL (UK Electoral Roll)
- There are currently no restrictions on searching or displaying results
- Searching is currently free

New functionality

Requirement 1

Implement the sign-in form, submitting the form should put the email in session if customer exists otherwise it should return an appropriate exception/error message. If sign-in was successful it should redirect to search form.

Hint: The following interface have been provided for you to query existing customer database net.icdpublishing.exercise2.myapp.customers.dao.CustomerDao

Requirement 2

Add a controller with 3 mappings. One will return a HTML page with a search form (that’s where it should redirect to after submitting sign-in form). This form should have two input fields: surname + postcode. When submitted it will trigger a POST request to the second mapping which will display the results in a table (and only that, no need for any other UI related element in this exercise). Finally the last mapping should perform a search (same input parameters) which will return the results in json format with an application/json response type.

Requirement 3

- Ensure that the system only return BT specific records for non paying customers. In other words, non paying customers are only allowed to view records where BT is the exclusive data source.
- Premium customers can view data from all three data sources (BT, DNB, ELECTORAL_ROLL).

Hint: Treat the search engine as a black box. The focus of this task is to manipulate what comes back from the search engine.

Requirement 4

Premium customers have purchased 192 credits in advance which they intend to use. Your task is to implement a mechanism to charge premium customers based on the data coming back from the search engine.
The system should charge premium customers 1 credit per record returned with the following exception. We don't charge for individual records which exclusively contain data sourced from "BT".

Hint: A legacy charging service has been provided for you. Your task is to integrate with the following service: net.icdpublishing.exercise2.myapp.charging.services.ChargingService