CSCI318 - Task 1

Contents

CSCI318 - Task 1

Dependencies

- Spring Boot 2.5.3
- Java 8
- Spring Web
- Spring Data JPA
- H2

Building

```
cd csci318-project && mvn clean build
```

Running

```
java -jar ./target/online-ordering-0.0.1-SNAPSHOT.jar
```

Description of the project

Structure

```
$ tree -A ./src/
./src/
├── main
│   ├── java
│   │   └── com
│   │       └── csci318project
│   │           └── onlineordering
│   │               ├── api
│   │               │   ├── ContactController.java
│   │               │   └── CustomerController.java
│   │               ├── model
│   │               │   ├── Contact.java
│   │               │   └── Customer.java
│   │               ├── OnlineOrderingApplication.java
│   │               └── repository
│   │                   ├── ContactRepository.java
│   │                   └── CustomerRepository.java
│   └── resources
│       ├── application.properties
│       ├── static
│       └── templates
└── test
    └── java
        └── com
            └── csci318project
            └── onlineordering
                └── OnlineOrderingApplicationTests.java
```

The onlineordering application is split into api, model and repository.

The API Layer consists of the ContactController and the CustomerController class which interfaces with the REST Requests (GET, POST, PUT and DELETE). This API Layer interacts with the Service Layer with domain models underneath it. The project has two models, Customer and Contact. In addition, the Service Layer interacts with the repository interfaces which is the Datat Access Layer for this task.

API Layer

There are two classes in the API Layer, ContactController and CustomerController. ContactController accepts REST Requests with the path set to '/contact'. This class handles REST Requests and has the following functionality:

- getAllContact()
  - Returns all the contacts stored in the database.
- getContactById()
  - Returns the contact stored in the database which matches the "{id}" specified in the request.
- addContact()
  - Returns the new contact after accepting a JSON body which takes these following keys:
    - name
    - phone
    - email
    - position
- deleteContactById()
  - Deletes the contact specified by "{id}" in the database.
- updateContactById()
  - Accepts a JSON body which is similar to addContact() and updates the contact in the database which matches the specified "{id}". The function returns the updated contact.

The CustomerController class accepts REST requests with the path '/customer' and has the following functionality:

- getAllCustomer()
  - Returns all the customers stored in the database.
- addCustomer()
  - Returns the new customer after accepting a JSON body which takes in the following keys:
    - companyName
    - address
    - country
- getCustomerById()
  - Returns the customer which matches the "{id}" specified.
- getCustomerByCompanyName()
  - Requires the path '/customer/company' and the search param 'name'.
  - Returns the customer which matches the search parameter.
- updateCustomer()
  - Accepts a JSON body similar to addCustomer() and updates the customer in the database which matches the "{id}" specified in the path.
- addCustomerContact()
  - Links an existing customer with a contact which exists in the database.
  - Requires specifying with '/customer/{id}/contact/{contactId}'
  - Returns the updated customer.

Service Layer

In the Customer Domain, two models created for this task, Customer and Contact.

A Customer model has the following variables:

- companyName
- address
- country
- contact

A Contact model has the following variables:

- name
- phone
- email
- position
- customer

Data Access Layer

The repositories for this task extends on the JpaRepository from the Spring framework. Thus, the default function such as save(), findById() and delete() are used.

The CustomerRepository has an extra definition which enables the application to search the database when provided the search term. This is implemented as the function findByCompanyName() and returns a List of the matches.

Database

The project currently uses the H2 in memory database.

Examples of Input and Output

Rest Requests

Request 1: Creating a new customer entry.

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "companyName":"Microsoft",
        "address" : "Redmond",
        "country" : "USA"
    }' http://localhost:8080/customer
```

- Output:

```
"id":1,"companyName":"Microsoft","address":"Redmond","country":"USA","contact":null}
```

Request 2: Creating a new contact entry.

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "name" : "Bill Gates",
        "phone" : "04555555555",
        "email" : "thisisgates@gmail.com",
        "position" : "CEO"
    }' http://localhost:8080/contact
```

- Output:

```
{"id":2,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}
```

Request 3: Linking a Customer and a Contact.

- Input:

```
curl -i -X PUT http://localhost:8080/customer/1/contact/2
```

- Output:

```
{"id":1,"companyName":"Microsoft","address":"Redmond","country":"USA",
"contact":{"id":2,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}}
```

Request 4: Updating a specific contact by ID.

- Input:

```
curl -i -X PUT -H "Content-Type:application/json" -d \
    '{
        "name" : "Tim Cook",
        "phone" : "04555555557",
        "email" : "thisistim@hotmail.com",
        "position" : "CEO",
        "customer" : "Apple"
    }' http://localhost:8080/contact/2
```

- Output:

```
{"id":2,"name":"Tim Cook","phone":"04555555557","email":"thisistim@hotmail.com",
"position":"CEO"}
```

Request 5: Get all customers in the database.

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "companyName":"Apple",
        "address" : "California",
        "country" : "USA"
    }' http://localhost:8080/customer

curl -X GET http://localhost:8080/customer
```

- Output:

```
{"id":3,"companyName":"Apple","address":"California","country":"USA","contact":null}
[{"id":1,"companyName":"Microsoft","address":"Redmond","country":"USA",
"contact":{"id":2,"name":"Tim Cook","phone":"04555555557","email":"thisistim@hotmail.com",
"position":"CEO"}},{"id":3,"companyName":"Apple","address":"California",
"country":"USA","contact":null}]
```

Request 6: Get a specific customer in the database.

- Input:

```
curl -X GET http://localhost:8080/customer/3
```

- Output:

```
{"id":3,"companyName":"Apple","address":"California","country":"USA","contact":null}
```

Request 7: Search for a customer by company name.

- Input:

```
curl -X GET http://localhost:8080/customer/company?name=Apple
```

- Output:

```
[{"id":3,"companyName":"Apple","address":"California","country":"USA","contact":null}]
```

Request 8: Get all contacts in the database.

- Input:

```
curl -X POST -H "Content-Type:application/json" -d \
    '{
        "name" : "Bill Gates",
        "phone" : "04555555555",
        "email" : "thisisgates@gmail.com",
        "position" : "CEO"
    }' http://localhost:8080/contact

curl -X GET http://localhost:8080/contact
```

- Output:

```
{"id":4,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}[{"id":2,"name":"Tim Cook","phone":"04555555557",
"email":"thisistim@hotmail.com","position":"CEO"},{"id":4,"name":"Bill Gates",
"phone":"04555555555","email":"thisisgates@gmail.com","position":"CEO"}]
```

Request 9: Get a specific contact in the database.

- Input:

```
curl -X GET http://localhost:8080/contact/4
```

- Output:

```
{"id":4,"name":"Bill Gates","phone":"04555555555","email":"thisisgates@gmail.com",
"position":"CEO"}
```