INPUT VALIDATION

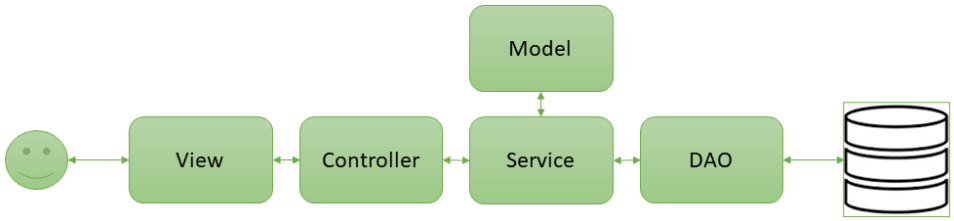
Project Report

CSE-5382-001, Secure Programming

Yash Rajanish Karanje
1002088995

Secure Programming

The project “Input Validation” has been implemented using Spring Boot. Like any n-tier architecture, this project has several layers including View, Controller, Service, Model, DAO, Persistence Layer.



View

We do not have UI implementations available. However, our view layer is being leveraged with help from Postman. We hit the calls to our server through postman collections provided.

Controller

Controller is the entry gate where external calls hit the server. We have exposed 4 such routes.

- GET /PhoneBook/list – Produces a list of the members of the database.
- POST /PhoneBook/add – Add a new person to the database. Argument is an object with name and phone number string elements as request body represented in JSON.
- PUT /PhoneBook/deleteByName – Removes entry from the database by name. Argument is the name as a string request parameter.
- PUT /PhoneBook/deleteByNumber – Remove entry by phone number. Argument is the phone number as a string request parameter.

Model

Model layer is a representation of the blueprint of the Phone Book entity. Here, we have specified the name and phone number attributes as an instance of Phone Book type.

Service

This is the most crucial layer of the project which handles the business logic. Controller layer redirects to the request specific utility of Service layer.

Regular Expression

1. Phone Number:

First Part: International Phone Numbers, Second Part: Postal Code Format, Third Part: Hyphenated Area Code, Fourth Part: Basic 5-Digit Number, Fifth Part: International Number with Spaces. This regex is quite comprehensive and aims to match a variety of phone number formats.

2. Name

Names can contain alphabetical characters (both uppercase and lowercase), may be separated by a hyphen, comma, or single quote, and the entire pattern should repeat maximum 3 times.

Logging

- This project uses Apache Log4j framework to capture system logs.
- The resources folder contains *log4j.properties* file that contains logging configuration.
- The file /logs/dump.logs contains all the logs. Timestamp follows UTC time zone
- Root logger is set to DEBUG mode. It will capture Debug < Info < Warn < Error < Fatal < Off.

Adding Entry Functionality Cases

- True case: Response 200 OK
- Database error: Response 400 BAD REQUEST
- Exception: Response 500 INTERNAL SERVER ERROR
- Invalid Input: Response 400 BAD REQUEST

View Phone Book Functionality Cases

- True case: Response 200 OK
- Exception: Response 500 INTERNAL SERVER ERROR

Delete Entry By Name Case

- True case: Response 200 OK
- Not found: Response 404 NOT FOUND
- Database error: Response 400 BAD REQUEST
- Exception: Response 500 INTERNAL SERVER ERROR
- Invalid Input: Response 400 BAD REQUEST

Delete Entry By Number Case

- True case: Response 200 OK
- Not found: Response 404 NOT FOUND
- Database error: Response 400 BAD REQUEST
- Exception: Response 500 INTERNAL SERVER ERROR
- Invalid Input: Response 400 BAD REQUEST

Data Access Object (DAO) & Database

- This project uses MySQL database hosted on UTA cloud. UTA cloud allows manipulation of databases through php scripting that allows prepared statements.
- This layer makes calls to php files over UTA cloud which further deals with MySQL database and manipulates the data.
- This layer also has 4 basic functions corresponding to the calls we receive.

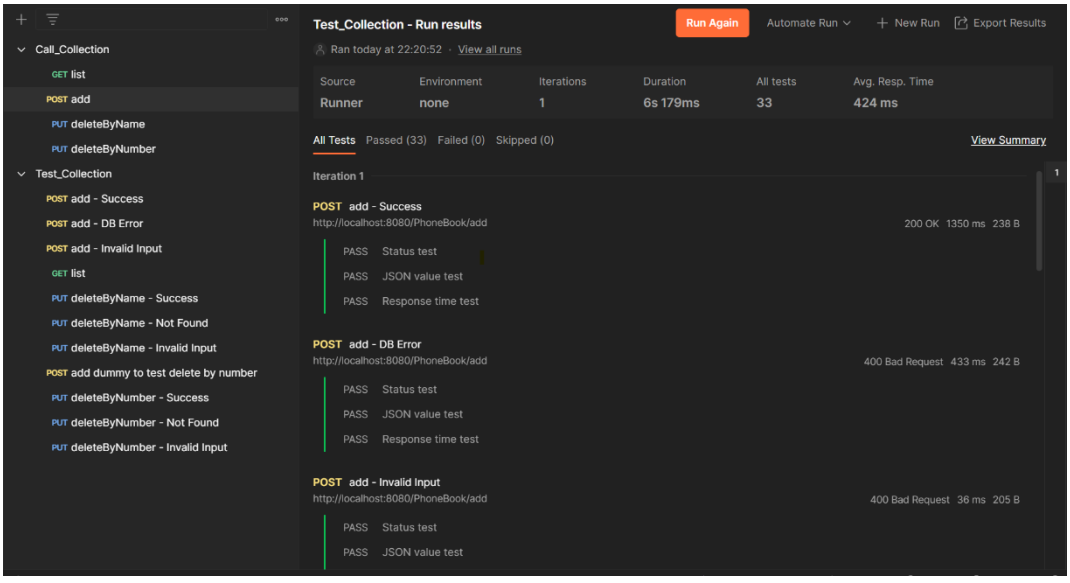
- The response received from UTA cloud is encapsulated into JSON object and returned through preceding layers till it reached requesting user.
- The config.php is a configuration file which contains database connectivity credentials. The conn.php file establishes the connection with mysql database. This connection is used further by rest of the php files.
- All the database related files have been placed in “db” folder. Those are the exact same files that have been uploaded on UTA cloud.

Screenshots

Logs

```
bash-4.4# cat /logs/dump.logs
2023-11-18 01:01:57,567 INFO  PhoneBookService:94 - Phone Book listed successfully
2023-11-18 01:02:30,901 INFO  PhoneBookService:72 - Entry added successfully, Add Phone Book Entry, name: Sanjana Karanje, number: (8
14) 580-4579
2023-11-18 01:03:37,098 INFO  PhoneBookService:109 - Entry deleted successfully, Deleting entry by name Sanjana Karanje
2023-11-18 01:04:08,694 INFO  PhoneBookService:139 - Entry deleted successfully, Deleting entry by number (738) 580-4579
2023-11-18 01:08:41,471 INFO  PhoneBookService:94 - Phone Book listed successfully
2023-11-18 01:27:55,244 INFO  PhoneBookService:72 - Entry added successfully, Add Phone Book Entry, name: Sanjana Karanje, number: (8
14) 580-4579
2023-11-18 01:28:02,039 INFO  PhoneBookService:94 - Phone Book listed successfully
2023-11-18 01:28:19,129 INFO  PhoneBookService:109 - Entry deleted successfully, Deleting entry by name Sanjana Karanje
2023-11-18 01:28:23,343 DEBUG PhoneBookService:143 - Not Found, Deleting entry by number (738) 580-4579
2023-11-18 01:28:45,270 INFO  PhoneBookService:139 - Entry deleted successfully, Deleting entry by number (992) 384-6321
2023-11-18 01:39:51,062 INFO  PhoneBookService:72 - Entry added successfully, Add Phone Book Entry, name: Jay Karanje, number: (992)
384-6321
2023-11-18 01:39:52,246 DEBUG PhoneBookService:85 - Invalid Input, Add Phone Book Entry, name: 1234 Karanje, number: yash 384-6321
2023-11-18 01:39:53,784 INFO  PhoneBookService:72 - Entry added successfully, Add Phone Book Entry, name: Yash Karanje, number: (682)
374-6321
2023-11-18 01:39:55,352 INFO  PhoneBookService:94 - Phone Book listed successfully
2023-11-18 01:39:56,875 DEBUG PhoneBookService:113 - Not Found, Deleting entry by name Noah Karanje
2023-11-18 01:39:58,382 DEBUG PhoneBookService:113 - Not Found, Deleting entry by name Noah Karanje
2023-11-18 01:39:59,499 DEBUG PhoneBookService:127 - Invalid Input, Deleting entry by name 1234 Karanje
2023-11-18 01:40:01,059 INFO  PhoneBookService:139 - Entry deleted successfully, Deleting entry by number (682)374-6321
2023-11-18 01:40:02,569 DEBUG PhoneBookService:143 - Not Found, Deleting entry by number (123) 374-6321
2023-11-18 01:40:03,658 DEBUG PhoneBookService:157 - Invalid Input, Deleting entry by number (yash)374-6321
2023-11-18 01:41:43,594 DEBUG PhoneBookService:75 - Duplicate entry '(992) 384-6321' for key 'PRIMARY', Add Phone Book Entry, name: J
ay Karanje, number: (992) 384-6321
2023-11-18 01:42:41,971 DEBUG PhoneBookService:75 - Duplicate entry '(992) 384-6321' for key 'PRIMARY', Add Phone Book Entry, name: J
ay Karanje, number: (992) 384-6321
2023-11-18 01:42:43,073 DEBUG PhoneBookService:85 - Invalid Input, Add Phone Book Entry, name: 1234 Karanje, number: yash 384-6321
2023-11-18 01:42:44,580 INFO  PhoneBookService:72 - Entry added successfully, Add Phone Book Entry, name: Yash Karanje, number: (682)
374-6321
2023-11-18 01:42:46,085 INFO  PhoneBookService:94 - Phone Book listed successfully
2023-11-18 01:42:47,612 DEBUG PhoneBookService:113 - Not Found, Deleting entry by name Noah Karanje
2023-11-18 01:42:49,184 DEBUG PhoneBookService:113 - Not Found, Deleting entry by name Noah Karanje
```

Test cases



Assumptions

- In the database, phone_number is the primary key. A user can have multiple phone_numbers but a phone_number will belong to only one user.
- Postman is replacing + sign's %2B with %20. Hence, + sign is not being recognized correctly. To be specific, the character "+" must be substituted with "%2B" when checking for a phone number beginning with "+" especially while passing it as a request parameter. (https://github.com/postmanlabs/postman-app-support/issues/2517)
- All special characters are removed by the regular expressions except for a dot, hyphen, apostrophe (for name only), and space. Both the phone number and the name cannot contain any numbers or letters.

Pros

- Use of UTA cloud made it easy to manage database on cloud and php connectivity is also seamless.
- No need to programmatically write JUnit test cases as same can be achieved through Postman in a much efficient manner.
- Use of n-tiered architecture, following coding standards, and the generic approach toward implementation made it easy to maintain such a multiple framework integrated project.

Cons

- Integrating local development of project into docker image and its execution could have been more seamless. However, it is good from a portability perspective.
- If a high number of requests are hit to UTA cloud from same user, it will block the user for a while.