Sunday, 10 December 2017

Java Challenge

Current Implementation highlight

- Support parallel execution on transfer between multiple account at same time.
- No Dead-lock
- No locking on the Service and Repository, but lock on the account object which is currently in transfer mode. Other accounts are eligible to perform transfer.
- Well written unit test for repository(AccountsRepositoryTest.java), Service (BankingServiceTest.java) and controller(BankingControllerTest.java)
    - Well Written Unit test for parallel execution scenario.
    - Used Mock implementation of Notification Service in BankingController unit test. See *BankingTestConfiguration.java* .
- Proper exception handling and well written unit test for each case.
- 100% code coverage of transfer functionality.

Future Enhancement

- If I talk about bringing new functionality I wants to bring following functionalities:
    - Every transfer (regardless of fail or successful) ***should must have a transaction reference***, without that, it would be hard to trace any transfer in future. The transaction reference would have a unique id, transfer information and transaction status.
    - The transaction reference should be provided with the result response.
    - Authentication and Authorization: Current implementation has does not have any authentication and authorization mechanism. We should bring at-least this to provide user a secure transfer. I would prefer Spring Security
    - Enable schedule transfer between accounts.
    - There is no information whom holds the account, So user is important.
    - I should implement accountRepository communicating with actual database.
    - Notification via other media like sms.
    - Transfer to other banks(or system like payTM etc)

Extra work I would consider Important to do before Go Live

- **Continuous integration and Delivery pipeline:** This is the most important enhancement I would prefer this application should have before go live. It enables the code intact and provide the working software with ease to the customer. Teamcity, builfordge, Jenkins and many more option available to enable the delivery pipeline. API testing is also a biggest challenge in destributed environment. Cucumber and some other framework enable end to end BDD.
- Bring code modularity: Banking itself involve several other operation and it is better to split account and banking into two separate microservice.
- If we split the project into multi micro service we need a API gateway that will encapsulate all the AIP's and context and will provide ease of access of the exposed API's. I would certainly prefer Zuul proxy.
- When we split our application Into micro service a reliable communication between these are a biggest challenge. it is better to implement a common interface that would keep track of all running microservice like banking and accounts. I would prefer eureka to introduce this capabilities.
- There would be many properties that I would require to change at run time when the server would be running. A configserver gives environment to achieve that.
- It may require to monitor the application health check. Actuator enable application to do that without going to the actual server tracing.
- If Say one server (either banking or account) is down, bringing fault tolerant mechanism is a biggest challenge. Histrix enable circuit breaker within the application.
- Logging is also a biggest cross cutting concern, Without proper logging we can not fix production issues and defects. Sleuth enable distributed tracing and, Kibana etc provide a nice way to logging and monitoring.
- The consumer able to know about the api's, Swagger provide a nice dashboard with all the exposed api with inbuilt documentation.
-