ACS Programming Assignment 1

Kai Arne S. Myklebust, Silvan Adrian

Handed in: November 26, 2018

Contents

1 Questions for Discussion on Architecture

1.1 Question 1

1.1.1 (a)

Implementation

- **rateBooks:** We use an additional validate method. Which takes a book rating and validates it by checking if the rating is between (0 - 5) and use the ISBN validation method. If there is a validation error we throw an exception (BookStoreException), this way we either save all or none of the ratings.
- **getTopRatedBooks:** We only check if the number of Books is a positive number (≥ 0).
- **getBooksInDemand:** No Parameter validation. We used streams filtered by missed sales and return an immutable book.

Tests

- **rateBooks:** We created the test case **shouldRateNoBook**, which rates the same book 2 times but one of the ratings is not valid. So no books are rated (total rating: 0, number of times rated: 0)
- **getTopRatedBooks:** We have the test case **shouldReturnAllTopRated-Books** which just returns all top rated books (according to the parameter number of books).
- **getBooksInDemand:** We have a test case **shouldGetBooksInDemand** which just gets all the books in demand.

1.1.2 (b)

We did that by asserting the results which we get back from the service, so in case we set the local flag true or false. The tests will run either way (stay green).

1.2 Question 2

1.2.1 (a)

By using a Client/Server architecture, we achieve modularity even though technically the server saves all the books only in the memory. This doesn't allow for multiple servers using the same data (having not a single Database).

1.2.2 (b)

It distinguishes between mutable and immutable books, so that the bookstore clients are not allowed to mutate data only get immutable books (no writing is allowed).

1.2.3 (c)

If an error occurs in either a client or the service it would mean that both of them will shutdown/fail since they run on the same JVM.

1.3 Question 3

1.3.1 (a)

The message Handler provides a naming service, so that we are able to use message tags which then executes the corresponding method.

1.3.2 (b)

The client calls the server by its server address + a message tag. So the message tag sent by the client decides on which operation gets run on the server.

1.4 Question 4

In the Architecture ”at-most-once” RPC semantics is implemented, this we see since there are operations which have side-effects (operations which change the state on the server). In case of failure the operation won't be completed.

1.5 Question 5

1.5.1 (a)

Yes it is safe to use web proxies with the architecture.

1.5.2 (b)

Since the communication between the clients and the Server is already HTTP based it is safe to place web proxies in between the clients and the server. To the communication itself it doesn't matter if it get gets passed through a proxy before it arrives at the server.

Possible placement of proxies:

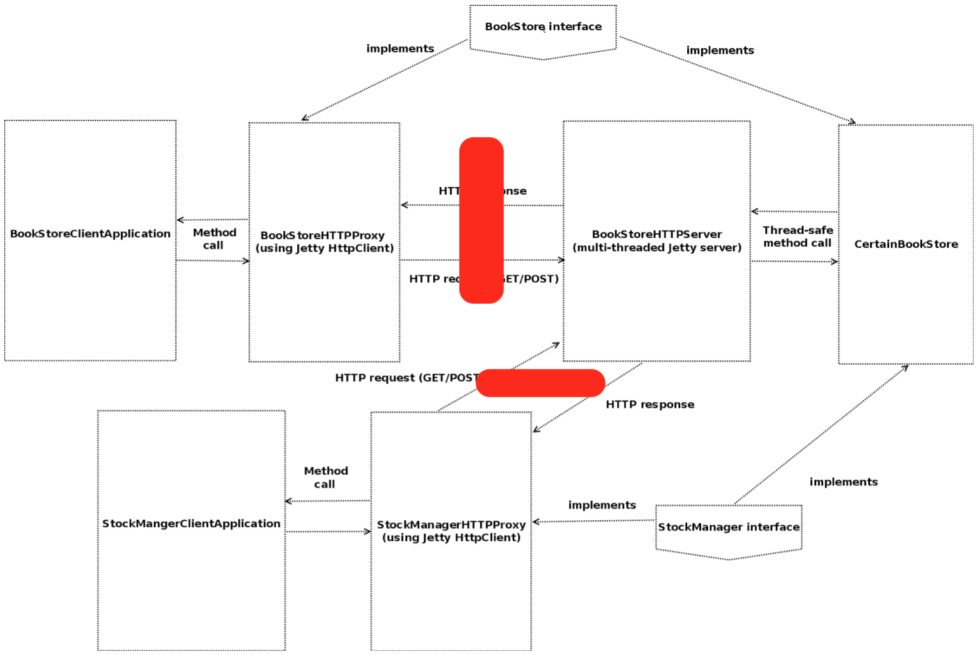

Figure 1: Placement of proxies in architecture

1.6 Question 6

1.6.1 (a)

Yes there is a bottleneck in the architecture.

1.6.2 (b)

The bottleneck is the server, since its state is hold in the memory, so it can't be scaled. If there are too many clients the server could run out of memory. Same for the amount of request the server can process, since it can't be scaled horizontally only vertically.

1.7 Question 7

1.7.1 (a)

Yes failures would be experienced differently since the proxy can handle failures of communicating with the server itself. The client could then get a failure response from the proxy.

1.7.2 (b)

Yes it could, by responding to requests from clients by using it's cache (so the server doesn't even need to be contacted).

1.7.3 (c)

Using a web cache won't affect the semantics, since the semantics depends on the state on the server. For example when buying a book, the state on the server still needs to be updated.