[GSHMedia] Java Interview Test

Requirements

Develop a Java application to handle Orders and Stocks.

The application receives Stocks as files in XML format.

- xml files are placed in the input folder *(../stocks_new)*
- file is moved to processed folder after processing *(../stocks_processed)*

The application receives Orders requests as JSON messages on a RabbitMQ queue *(ORDERS queue).*

- The application must be able to handle 5 concurrent Orders requests!

The application sends the Orders response on a different RabbitMQ queue *(ORDERS_RESPONSE queue):*

- Orders response values: RESERVED or INSUFFICIENT_STOCKS

Technical Details

Create entities:

- Product (name, id, stock)
- Order (products, name, client, status)

Entities must be persisted in a mysql database.

Handle project dependencies with Maven or Gradle.

Use Java15.

Install/Use a local database server and RabbitMQ server.

> **Do not use any Spring Framework libraries !**

Deliverables

1. Dump of the database.
2. Source code.
3. application.properties file containing:
    a. database connection details
    b. RabbitMQ configuration details
    c. path to folders (/stock_new and /stock_processed)

Evaluation methodology

1. Variables, method and class naming.
2. Package structure.
3. Correct use of interfaces.
4. Application lifecycle:
    a. application initialization
        i. establish connections
    b. graceful stop
    c. exceptions handling
    d. proper threads management
5. Proper sharing of resources (connections, channels etc.)
6. Use of design patterns.
7. Following coding industry standards.

Testing methodology

We will:

- use the sql dump to create the database.
- import source code in Intellij
- setup the correct configuration in application.properties
- load initial stocks via xml
- send Orders requests on the corresponding queue
- verify Orders response

Example of XML file containing Stocks

```
<?xml version="1.0" encoding="UTF-8"?>
<stocks>
   <stock>
      <product_id>1</product_id>
      <quantity>10</quantity>
   </stock>
   <stock>
      <id_produs>2</id_produs>
      <quantity>20</quantity>
   </stock>
   <stock>
      <id_produs>3</id_produs>
      <quantity>-30</quantity>
   </stock>
</stocks>
```

Example of Order request

```
{
   "client_name":"John",
   "items":[
      {
         "product_id":1,
         "quantity":3
      },
      {
         "product_id":4,
         "quantity":6
      }
   ]
}
```

Example of Order response

```
{
   "order_id":1,
   "order_status":"RESERVED",
   "error_message":"(error here if applies)"
}
```