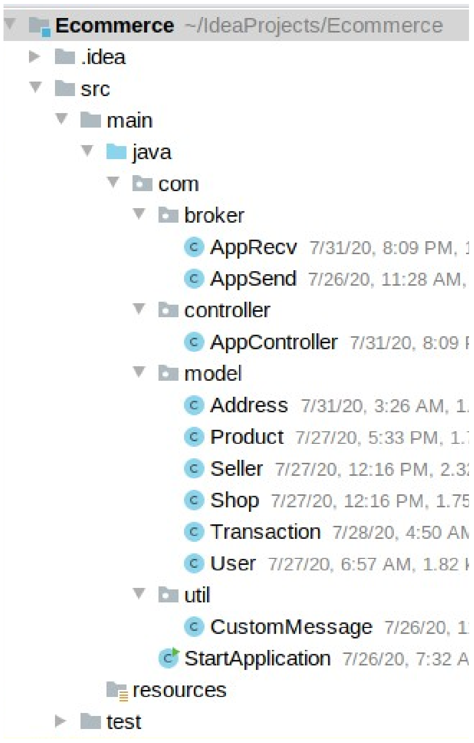
Program Documentation
Ecommerce Shopping & Payment

Purpose : Explaining the structure of the Project

This project is divided into 3 Folder:



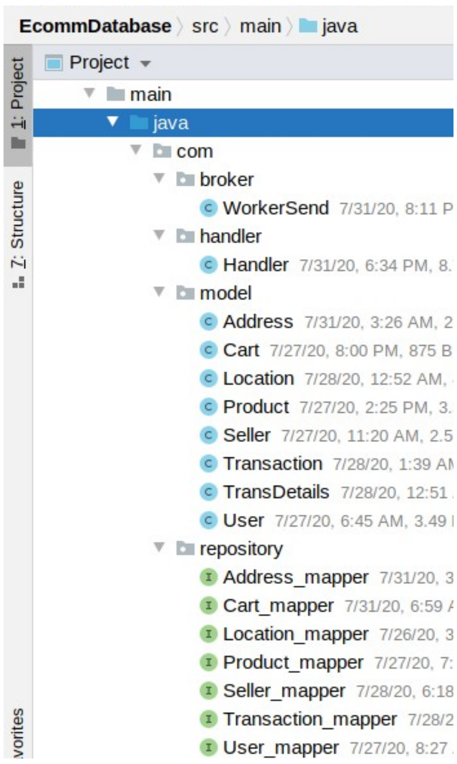
```
Ecommerce ~/IdeaProjects/Ecommerce
  .idea
  src
    main
      java
        com
          broker
            AppRecv 7/31/20, 8:09 PM,
            AppSend 7/26/20, 11:28 AM,
          controller
            AppController 7/31/20, 8:09
          model
            Address 7/31/20, 3:26 AM, 1.
            Product 7/27/20, 5:33 PM, 1.
            Seller 7/27/20, 12:16 PM, 2.3
            Shop 7/27/20, 12:16 PM, 1.75
            Transaction 7/28/20, 4:50 A
            User 7/27/20, 6:57 AM, 1.82
          util
            CustomMessage 7/26/20, 1
          StartApplication 7/26/20, 7:32 A
      resources
    test
```

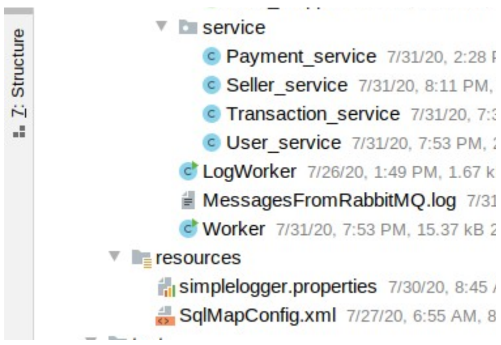
1. **Ecommerce : The Web API**

- Handles incoming request and send response back with json format.
- Send and receive messages with RabbitMQ.

Mappings:

- shopleen/profile
  - register
  - login
  - logout
  - confirm account
  - edit password
  - see profile
- shopleen/address
  - input
  - edit
  - delete
  - show all address
- shopleen/cart
  - add to cart
  - view cart
  - remove product from cart
  - empty cart
- shopleen/shop
  - view shop
  - view product
  - make a shop
  - view shop as seller
  - input product
  - edit product
  - delete product
- shopleen/buy
  - buy product
  - choose payment method
- shopleen/transaction
  - view transaction
  - confirm order received
  - confirm order shipped
  - check order arrived
  - confirm order done


```
EcommDatabase 〉 src 〉 main 〉 java
Project ▾
  ▼ main
    ▼ java
      ▼ com
        ▼ broker
            WorkerSend  7/31/20, 8:11 P
        ▼ handler
            Handler  7/31/20, 6:34 PM, 8.
        ▼ model
            Address  7/31/20, 3:26 AM, 2
            Cart  7/27/20, 8:00 PM, 875 B
            Location  7/28/20, 12:52 AM,
            Product  7/27/20, 2:25 PM, 3.
            Seller  7/27/20, 11:20 AM, 2.5
            Transaction  7/28/20, 1:39 AI
            TransDetails  7/28/20, 12:51
            User  7/27/20, 6:45 AM, 3.49 I
        ▼ repository
            Address_mapper  7/31/20, 3
            Cart_mapper  7/31/20, 6:59
            Location_mapper  7/26/20, 3
            Product_mapper  7/27/20, 7:
            Seller_mapper  7/28/20, 6:18
            Transaction_mapper  7/28/2
            User_mapper  7/27/20, 8:27
```


```
        ▼ service
            Payment_service  7/31/20, 2:28 I
            Seller_service  7/31/20, 8:11 PM,
            Transaction_service  7/31/20, 7:
            User_service  7/31/20, 7:53 PM,
          LogWorker  7/26/20, 1:49 PM, 1.67 k
          MessagesFromRabbitMQ.log  7/31
          Worker  7/31/20, 7:53 PM, 15.37 kB
    ▼ resources
        simplelogger.properties  7/30/20, 8:45
        SqlMapConfig.xml  7/27/20, 6:55 AM, 8
```

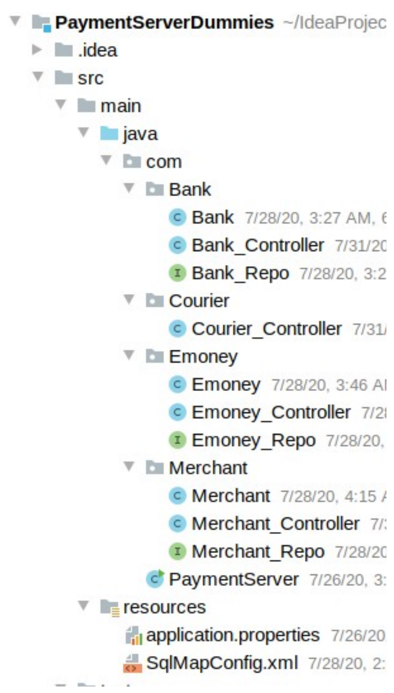
2. EcommDatabase: Services and Repository

- Receive message with RabbitMQ. Message received by 2 worker with the same exchange name. One worker write into log, the other execute business logic.
- Based on the value of "activity" key, the program reads the data received and call the corresponding method.
- Every method has different handlers to manage exceptions and illegal actions (for example: a user is not allowed to do anything if not logged in, a user cannot buy product from its own shop, etc). Handler class has 22 methods to check these exceptions.
- If all conditions are met, services execute business logics, manage databases and return response in json string format back to the Web API Controller with RabbitMQ.
- A user can do CRUD on address and cart.
- A seller can do CRUD on products
- Payment_service act as a gateaway to do choosing payment method logic and send information to payment server dummies.
- Transaction_service constructs a transaction and store it into database.

Explanation of User and Seller roles:

- User is the first basic role. User then can become a Seller.
- Seller is also a user, therefore it can also do user's activity, such as buying staffs.
- Seller has balance variable.
- User cannot buy from its own Seller's shop.

Transaction flow:
- User make a transaction
- User choose a payment method
- User check the transaction status → this action triggers the program to check the payment status in third-party payment API
- If the payment is done, seller can confirm to receive the order
- Seller insert order tracking number and confirm shipment
- Seller check the shipment status → this action triggers the program to check shipment status in third-party courier API
- User confirm order is done (User can confirm this even before order shipment status arrived, this is to simulate real-world situations where order has arrived but tracking doesn't say order has arrived). When order is done, balance is forwarded to Seller's account.



```
▼ PaymentServerDummies ~/IdeaProjec
  ▶ .idea
  ▼ src
    ▼ main
      ▼ java
        ▼ com
          ▼ Bank
              Bank 7/28/20, 3:27 AM, 6
              Bank_Controller 7/31/20
              Bank_Repo 7/28/20, 3:2
          ▼ Courier
              Courier_Controller 7/31/
          ▼ Emoney
              Emoney 7/28/20, 3:46 AI
              Emoney_Controller 7/28
              Emoney_Repo 7/28/20,
          ▼ Merchant
              Merchant 7/28/20, 4:15 A
              Merchant_Controller 7/
              Merchant_Repo 7/28/20
            PaymentServer 7/26/20, 3:
      ▼ resources
          application.properties 7/26/20
          SqlMapConfig.xml 7/28/20, 2:
```

3. PaymentServerDummies

- Payment (Bank, Emoney, and Merchant) and Courier are third-party entities and located outside the program, therefore these are called server dummies.
- Payment by user to the third-party are done manually by changing the data payment status to true in MySql Workbench.
- When user chooses payment method “Bank Transfer” and “Bank VA” , payment_service send user_id and the bill to the Bank. Bank stores them in the database.
- When user chooses payment method “Emoney”, Emoney checks if the user has an account and if the balance is enough. Payment instantly done if all checks are true. If not, user has to choose another payment method.
- When user chooses payment method “Merchant”, payment_service send user_id and the bill to the Merchant. Merchant stores them in the database.
- When user checks transaction status, HTTP POST request with trans_id as request is send to the corresponding Payment Server. Payment server checks the status of the transaction and returns true or false.
- When user checks shipment status, HTTP POST request is send. For this dummy, shipment status always returns true.