Car Rental Booking project specification

1.Instructions

This is a simple web client rental car booking API service for customers to reserve a car for a period of time. Users can log in and register for the system and can book and return vehicles within a certain time frame.

For time reasons and because I had personal matters to attend to before the deadline, I have simply built a framework here to implement the basic operation of a user renting and returning a car and some simple test cases were also implemented.

Design thinking

1.Database table design
Based on the scenario that users rent and return a car, for a simple demo, I need three database tables.
a. user: contains basic information about the user and the role, basic information for login display, user role for permission control, divided into: admin (0) and ordinary user (1), permission control through Spring Security
b.car_info: contains basic information about the car, name, inventory, rent, type, up/down status. The rent, type and up/down status are all for possible use in later extensions. In my project, the home page will only display information about cars that are on the shelf and have stock greater than 0
c.rental_car_order: the order information of the user's rental car, including orderNo, userId, carId, order creation time, order end time, order expiry time, order status, etc. The status of the order is divided into

```
0 order in progress
1 order placed
2 cancelled
3 car return has ended
4 no car return timeout.
```

In the specific implementation, a pre-deducted stock method is used, the user will have 10 minutes to decide when to place an order, then the status of the order is in progress, if the user refuses, the order status changes to cancelled, otherwise it changes to placed. If the user has not made a choice, there is a timed task to change the status of the order to cancelled. When the user returns the car at the before of the rental period, the order status will change to Ended, if the car is not returned after the time limit, the order status will change to Timeout.

pre-deducted stock: once the order is placed, the stock is reduced and no action is taken if the user confirms it later, otherwise the stock is returned.

Both the cancellation and the return of the car will result in a return of stock. In **Getting Started** you will see table design.

2.System design
(1) Basic data storage using Mysql database
(2) The deduction and increase of inventory and the generation of order numbers are controlled using Redis distributed locks
(3) Orders that are not selected by the user are cancelled by a timed task, similar to the order

countdown that is cancelled if not paid for
(4) The user type is divided into 0 admin, 1 user, where admin can enter the background to complete a series of operations such as adding, modifying, deleting, etc. Currently, the background code functions have been implemented, the front-end display only does the ordinary user can operate the function, such as login, registration, order rental, return the car, log out, etc. The permission control here is Spring Security.
(5) ORM framework to choose Beetlsql, the performance is relatively high.

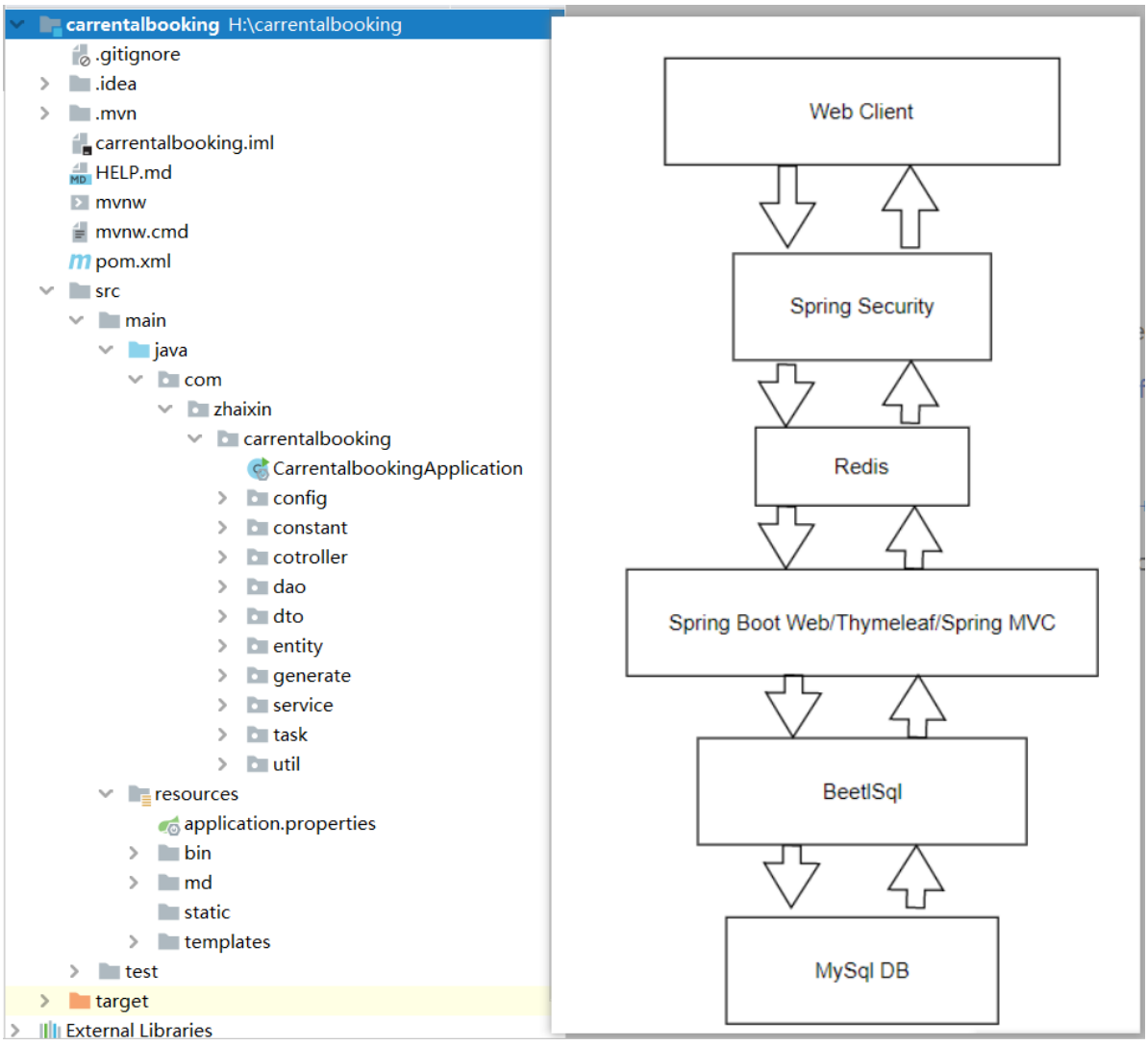
(6)For rapid development, the OrderNo has not been designed and will have to be developed specifically for the business. OrderNo use UUID.

Basic implementation process.

1. user registration after login
2. Login successfully, you can see the existing inventory of cars greater than 0 information
3. rent a car, deduct the inventory, generate an order, cancel the rental car then increase the inventory, modify the order status
4. Return the car, add inventory, modify the order status
5. View your own orders
6. Log out

The specific implementation can be viewed in the source code

2.Project Structure



- config: RedisConfig, BeetlSqlConfig, DataSourceConfig, SecurityConfig etc.
- constant: Generic result splitting and return code

- task: time task
- generate: beetlsql code generate

3.Tech Stack

- Java JDK8
- Spring Boot Web
- Spring Boot Thymeleaf
- Spring Boot Time Task
- Spring Security
- BeetlSql
- Redis & Redis distributed lock
- MySql5.7
- Druid Data Source
- JUint Test
- lombok
- HTML/CSS/JS

  .....

4.Getting Started

1.clone code from github

https://github.com/Zxnaruto/carrentalbooking

If you want to run the project locally, please note that the application. properties file in the mysql and redis port and your local consistent, this project in the mysql port = 8989, redis port = 6379, if not consistent, change to and your local the same.

2.advance preparation:

(1) you should install mysql5.7 or other version

If you don not want install Mysql to your PC, you can use Docker like me. follow those step:

```
docker pull mysql:5.7
docker run -itd --name mysql-test -p 3306:3306 -e MYSQL_ROOT_PASSWORD=123456
mysql:5.7
```

when the image has start, you can config you data source in application.properties.

(2) you should install redis for redis distributed lock

If you don not want install Mysql to your PC, you can use Docker like me. follow those step:

```
docker pull redis
docker run -itd --name redis-test -p 6379:6379 redis
```

config redis in application.properties

(3) create database and create sql table，it has three tables(user, car_info, car_rental_order)

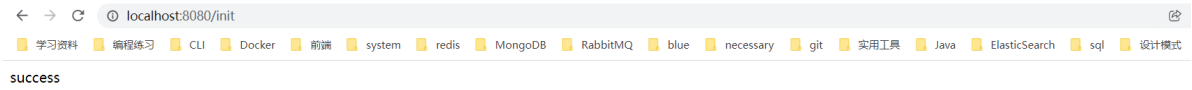
```
CREATE TABLE `user`  (
  `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键id',
  `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  `phone` varchar(20) NOT NULL DEFAULT '' COMMENT '用户手机号',
  `username` varchar(64) NOT NULL DEFAULT '' COMMENT '用户名',
  `email` varchar(50) NOT NULL DEFAULT '' COMMENT '用户邮箱',
  `password` varchar(256) NOT NULL DEFAULT '' COMMENT '密码',
  `role` int(2) NOT NULL DEFAULT 0 COMMENT '用户角色：0:user , 1:admin',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uniq_phone`(`phone`) USING BTREE,
  UNIQUE INDEX `uniq_email`(`email`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT = '用户信息表';

CREATE TABLE `car_info`  (
  `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键id',
  `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  `car_name` varchar(50) NOT NULL DEFAULT 0 COMMENT '汽车名称',
  `car_type` int(2) NOT NULL DEFAULT 0 COMMENT '汽车类型：1:BMW , 2:Toyota',
  `state` int(2) NOT NULL DEFAULT 0 COMMENT '汽车上下架状态：0:上架 , 1:下架',
  `car_stock` int(10) NOT NULL DEFAULT 0 COMMENT '汽车库存',
  `rent` bigint(20) NOT NULL DEFAULT 0 COMMENT '租金/天',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uniq_car_name`(`car_name`) USING BTREE,
  INDEX `idx_car_stock`(`car_stock`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT = '汽车信息表';

CREATE TABLE `car_rental_order`  (
  `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键',
  `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  `order_no` varchar(50) NOT NULL DEFAULT 0 COMMENT '订单编号',
  `user_id` bigint(20) NOT NULL DEFAULT 0 COMMENT '用户id',
  `car_id` bigint(20) NOT NULL DEFAULT 0 COMMENT '汽车id',
  `state` int(2) NOT NULL DEFAULT 0 COMMENT '订单状态：0:下单中，1:已下单，2:已取消，
3:已结束，4:超时',
  `start_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建订单时间',
  `end_time` timestamp NULL DEFAULT NULL COMMENT '订单结束时间',
  `expire_time` timestamp NULL DEFAULT NULL COMMENT '订单过期时间',
  `count` int(5) NOT NULL DEFAULT 0 COMMENT '租车天数',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uniq_order_no`(`order_no`) USING BTREE,
  INDEX `idx_user_id`(`user_id`) USING BTREE,
  INDEX `idx_car_id`(`car_id`) USING BTREE,
  INDEX `idx_start_time`(`start_time`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT = '租车订单表';
```

(4) when all dependence are Ok, include Maven dependence. You can run this project and init data

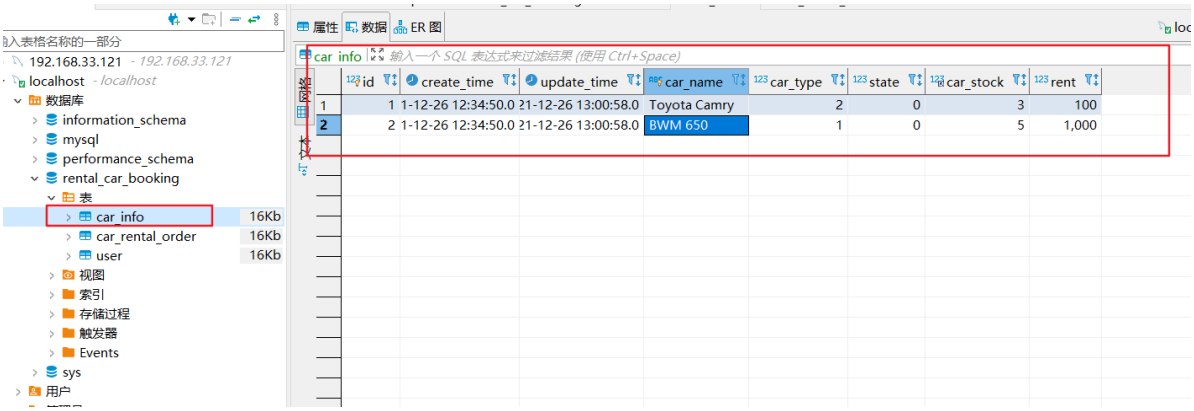
In you browse input: localhost:8080/init, and execute. You will see return "success" .

5.Running demo

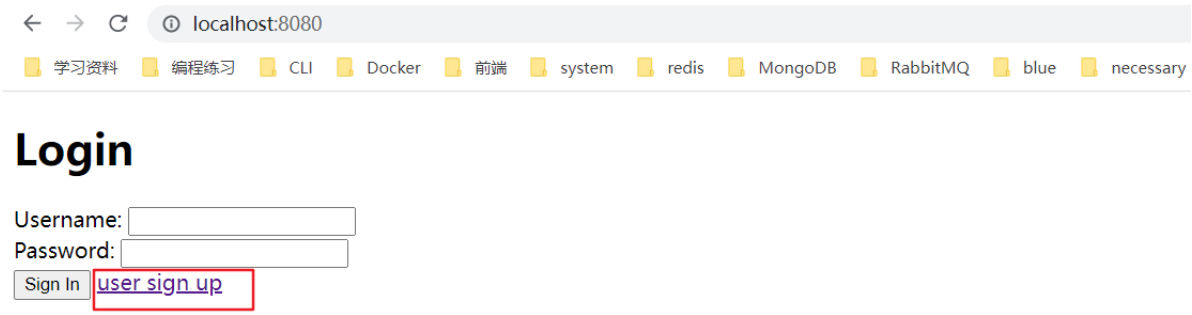
init data



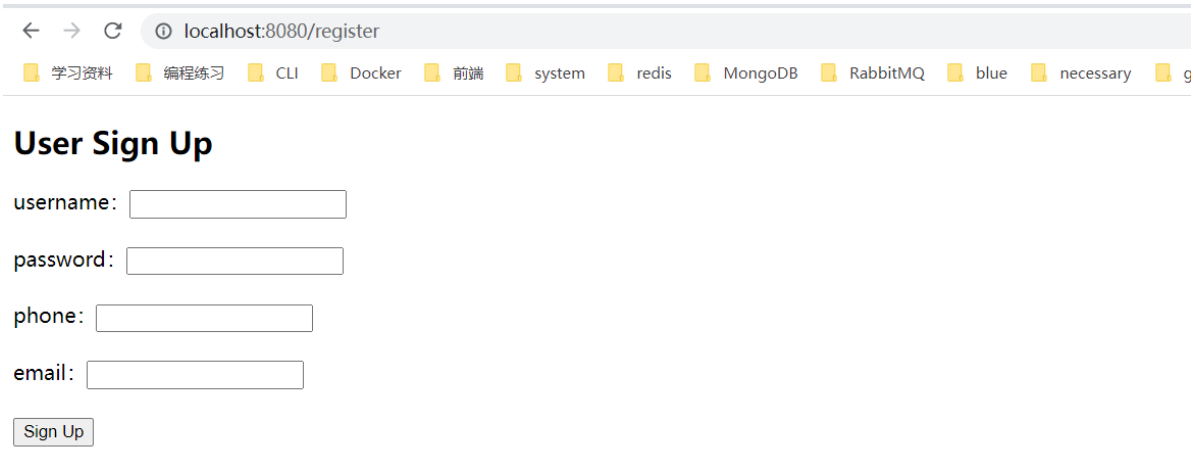
After that, you database car_info table has already initialized



(1) In you browse input:http://localhost:8080/ or http://localhost:8080/login you will see login page. When you first time use this project, you should sign up.

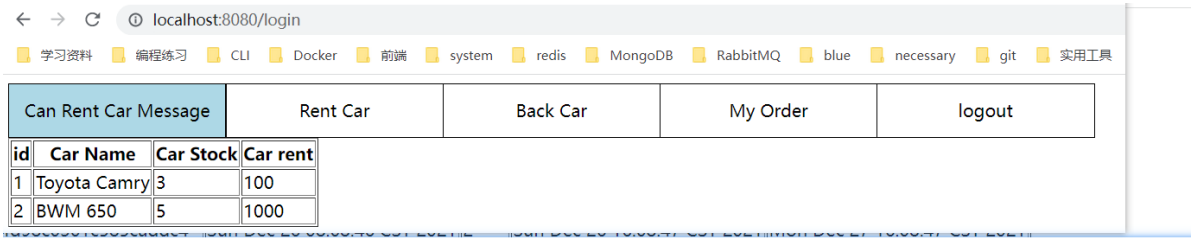


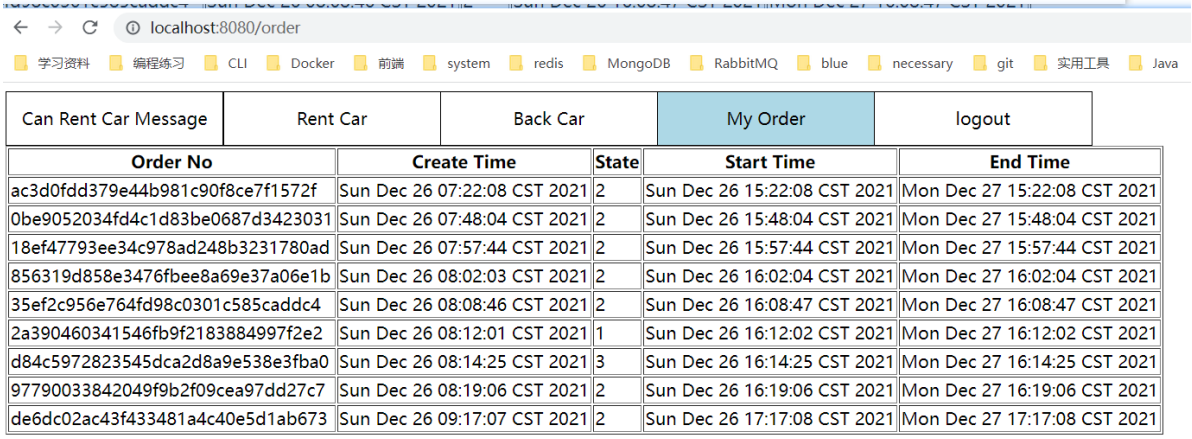
(2) You will see register page



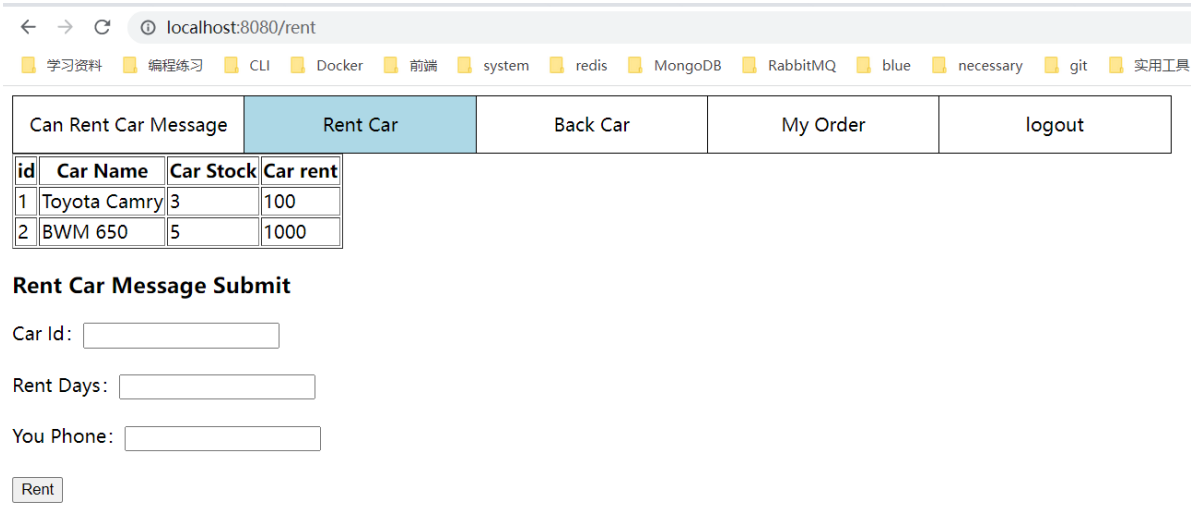
and write any message to sign up, **the mail should correct format**.

(3) finished sign up, you can sign in and see the home page, where show you rental car message. You can choose button to choose operation.(**Please use phone as username to sign in**)

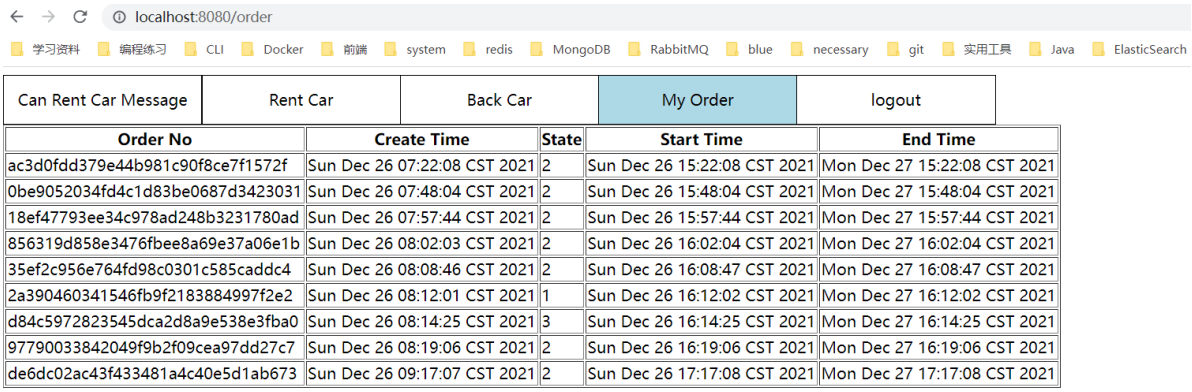




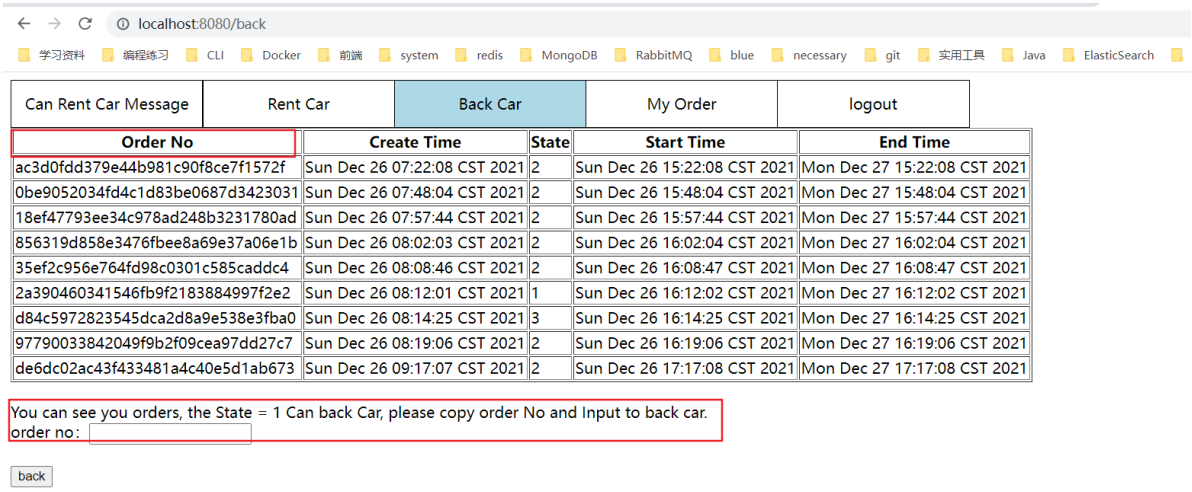
(4) rent car: you can rent a car according to the car rental information on the home page



and the car stock will reduce one. You can see My Order and you will see the order.



(5) Back car: you can return the car according to the order number and state=1 (order placed), after returning the car stock plus one. and order state update equal 3(finished).



(6) logout: return the login page

6.Important interface descriptions

Since I'm a Java back-end, I'm not very familiar with the front-end, but to show how it works, I went with Spring Boot Thymeleaf.

1.InitDataController

http://localhost:8080/init

There is only one initialize initData() method with no parameters, which returns a String

2.UserController

This interface includes a range of user-related actions such as logging in, logging out, renting and returning a car.

http://localhost:8080/login

http://localhost:8080/logout

http://localhost:8080/register

http://localhost:8080/doregister

http://localhost:8080/rent

http://localhost:8080/dorent

http://localhost:8080/rentconfirm

http://localhost:8080/rentcancel

http://localhost:8080/back

http://localhost:8080/doback

etc.

3.CarInfoController

Home page car information display

http://localhost:8080/home

4.CarRentalOrderController

order display

http://localhost:8080/order

8.Deploy to the cloud

As I don't have access to a free cloud, and as I've been delayed by some personal matters, I can only give an overview of the deployment steps for now.
Steps.

1. Use Xshell, login to the server according to the server ip and login password
2. Install jdk8, and configure environment variables
3. Install Mysql:5.7 and Redis on the server and run them, or install docker and use a docker image to install mysql and reids quickly and easily
4. Use the FTP tool to upload the local project jar package to the server
5. Because I am using spring boot web with tomcat embedded, I can run the project directly on the server using the java -jar command
6. you may need to open a port or turn off the firewall
7. Access the project in the browser by using ip+port

9.Optimisation

- 1.Complete operations related to the administrator's webpage, including adding, deleting and changing in the backend of the webpage
- 2.Order timeout logic implementation
- 3.Do not use Spring boot web this form, but use the form of front-end and back-end separation, do better front-end page.
- 4.Improve the test cases and increase the test coverage.
- 5.Streamline the code to achieve better clean code
- 6.Add log , exception handling and Parameter validation
- 7.According to the business separation project, such as orders have an order system, inventory has an inventory system, automotive goods have automotive goods system, and then distributed deployment project, the use of Redis Cluster to ensure that the system is highly available, Seata distributed transaction framework to solve the problem of distributed transactions, message queues to do system decoupling, the use of Cannl to ensure that the cache and database consistency, timing tasks The framework uses Elastic-job and so on.