

迭代二计划书

1. 阶段目标定义

1.1. 系统测试

1.1.1. 客户端测试

1.1.2. 接口测试

1.1.3. 压力性能测试

1.2. 用户培训

1.3. 版本发行准备

1.3.1. 编译打包

1.3.2. 部署环境准备

2. 系统测试方案

2.1. 客户端测试

2.1.1. 测试目标

2.1.2. 环境准备

2.1.3 测试流程

1. 阶段目标定义

1.1. 系统测试

1.1.1. 客户端测试

处于用户视角（司机、乘客、管理员），采用客户端界面与系统交互，主要检查各个界面交互逻辑是否正常。

1.1.2. 接口测试

采用Postman工具，测试http连接以及websocket连接是否正常。测试各接口行为是否正常，返回参数是否符合前后端联调协议，返回格式是否符合接口文档设计。

1.1.3. 压力性能测试

采用Jmeter压力测试工具，测试流量高峰期系统能否抗住高并发量。测试是否存在线程并发问题。记录系统面对突发流量处理速率以及异常率，并进行性能评估。

1.2. 用户培训

出具内测用户培训方案，提供面向不同群体的培训内容。于内网发布beta版以供内测用户使用。收集用户反馈与意见，与预期计划进行对比、调整。

1.3. 版本发行准备

1.3.1. 编译打包

准备打包编译工具，测试产物是否能够部署到测试机与服务器上。初步测试系统行为是否有异常。

1.3.2. 部署环境准备

准备部署容器，商议服务器采购方案，计划维护成本，挑选维护团队。

2. 系统测试方案

2.1. 客户端测试

2.1.1. 测试目标

通过客户端界面进行司乘约车以及完成订单的核心流程(主要演示一个订单的完整生命周期)

2.1.2. 环境准备

1. 可以通过uniapp框架提供的官方编辑器HbuilderX编译运行客户端
2. 或者可以通过cli的方式编译运行整个客户端项目

具体详情请见uniapp官网:[uni-app官网 \(dcloud.net.cn\)](http://uni-app.dcloud.net.cn)

2.1.3 测试流程

整个测试流程涉及两个(或至少两个)客户端，请确保您复现该测试时拥有足够的设备。

1. 事先声明

本项目**仅关注安卓app端**，如果在其他端(ios,微信小程序,浏览器)出现任何异常、错误或无问题，本项目概不负责。

2. 司乘登陆

这建立在已经注册的前提下，如果还没有为司机/乘客注册账号，可以通过客户端界面，或sql脚本，注册账号。

这里假设已经注册好了两个账号，在登陆时，请选择正确的用户类型，否则将无法登陆。

本系统规定，登陆时，只要验证码和密码有一个正确即可，当然，你同时填写两个，且均正确，也是可以登陆的。

13:29 95

N 蓝牙 HD 5G 92



17349742869

.....

验证码

当前选择 司机

获取验证码

登 录





13372100020

.....

验证码

当前选择 乘客

获取验证码

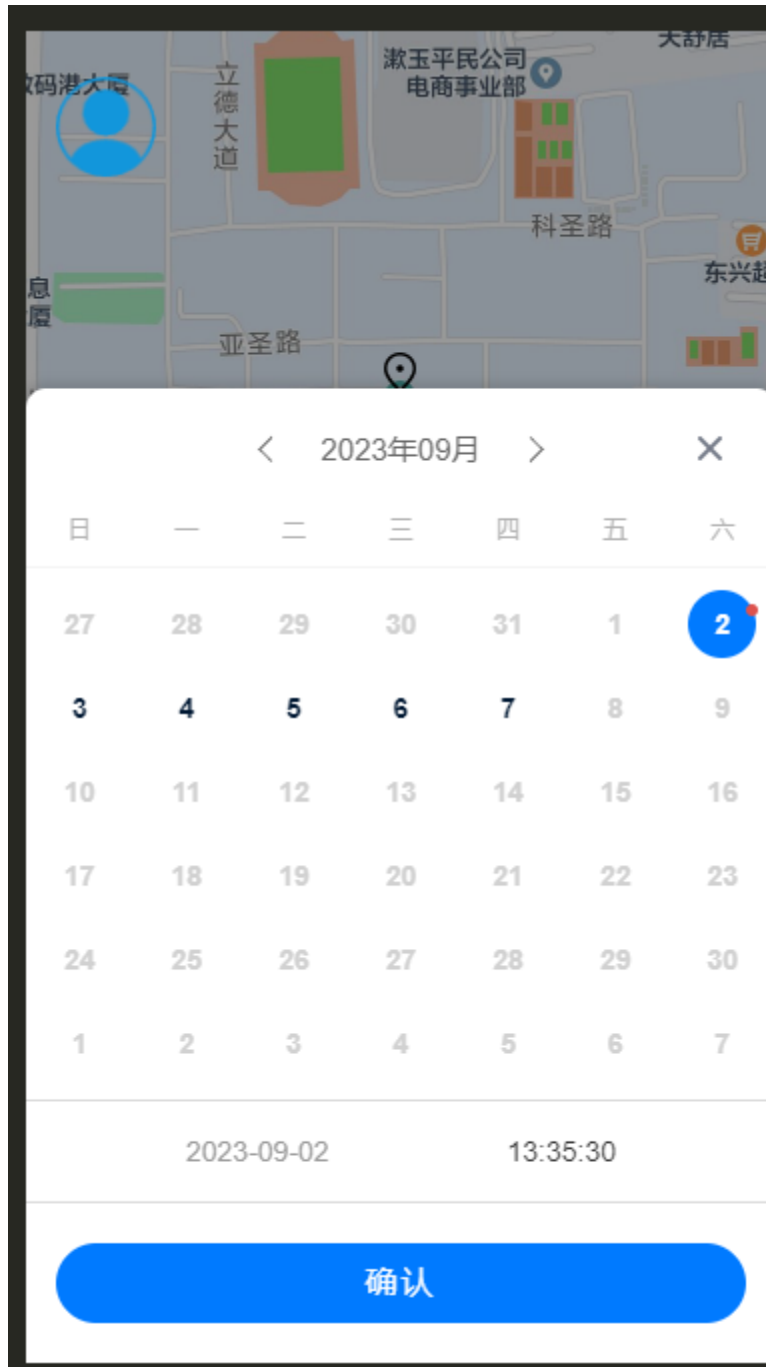
登录

3. 乘客发起约车请求

- 请选择合法的上车点与下车点。可以在搜索栏输入地点，高德地图/腾讯地图的api将为您搜索相关地点。
- 您也可以选择是否代叫或预约，这全按照用户的需求决定。
- 点击开始约车即可发送约车请求。







4. 司机开始听单

- a. 司机点击开始听单，即可表达自己准备听单的意愿



5. 系统分配订单

a. 这个对于用户来说是黑盒的，具体实现参照其他文档与压测环节。

6. 司机接到订单

- a. 司机可以选择接单或不接，如果不接，之后他将不会再收到这个订单。



7. 司机开始导航

- 在已经接单列表中，点击某一订单的导航图标，即可开始导航
- 点击后，地图上将出现导航路线与乘客上下车点





8. 司机接到乘客

- 司机点击接到乘客，确认手机尾号正确后，即可开始接送乘客。





9. 司机提交订单

- 注意！如果司机距离目的地太远，将无法提交订单！
- 提交订单后，乘客端将收到评价提醒

13:42



取消所有订单

关闭



订单已完成





10. 乘客提交评价

- 填写评分级别
- 填写评价内容
- 点击提交即可

评分

★★★★★

评论

司机: driver1

非常好!

提交评价

提交成功

2.2. 接口测试

本接口测试采用的工具为Postman。官网地址:[Postman API Platform | Sign Up for Free](#)

Postman最新版本提供了webSocket连接的接口，这也是我选择该应用作为接口测试工具的原因之一。

2.2.1 测试目标

一个司机和一个乘客登陆，完成核心业务流程中除了乘客提交评价的任何接口的测试。

核心业务流程具体参照系统演示文档。

2.2.2. 环境准备

1. 下载安装Postman软件，注册账户并登陆，创建所需的connection文件
2. 可以通过sql脚本，也可以通过客户端，或者可以通过Postman本身向负责注册业务的接口发送请求。总之，注册一个司机和一个乘客。
3. 通过客户端或Postman向负责登陆业务的接口发送请求。总之，获取这两个账号登陆的token。
4. 在Postman中所有需要鉴权的接口的Header中添加请求头字段

```
authorization:${token}
```

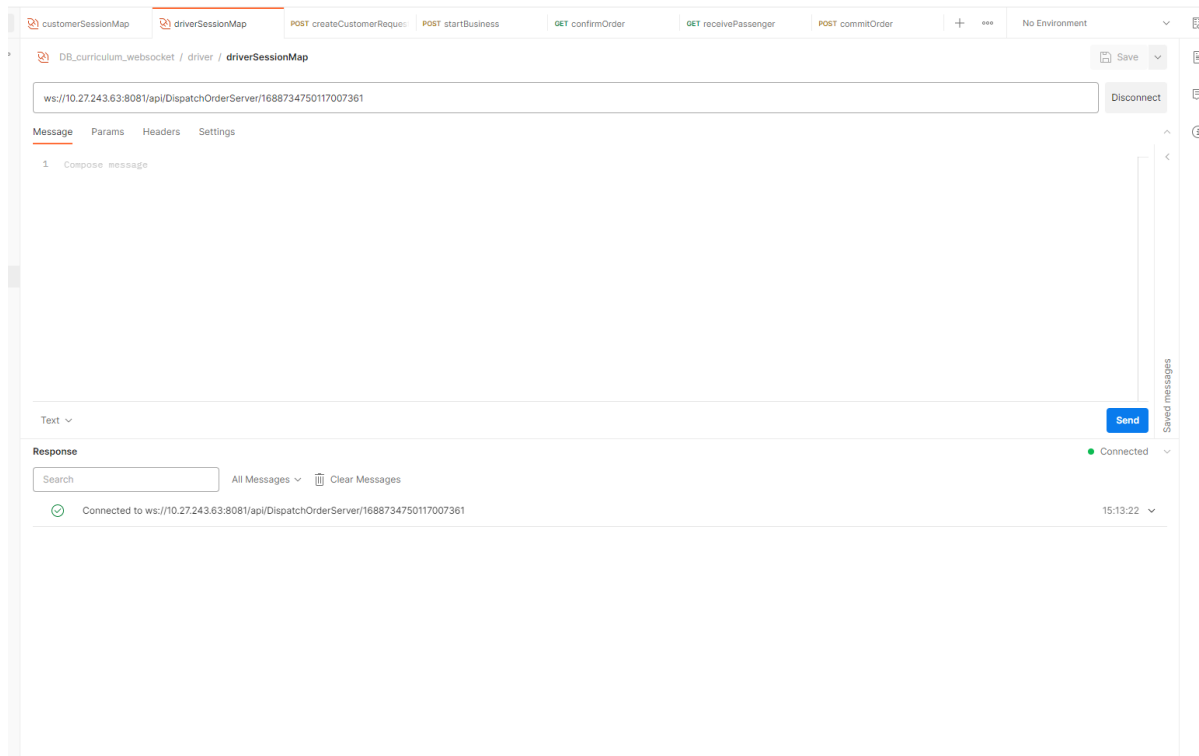
<input checked="" type="checkbox"/>	Postman-Token	③	<calculated when request is sent>
<input checked="" type="checkbox"/>	Content-Type	③	application/json
<input checked="" type="checkbox"/>	Content-Length	③	<calculated when request is sent>
<input checked="" type="checkbox"/>	Host	③	<calculated when request is sent>
<input checked="" type="checkbox"/>	User-Agent	③	PostmanRuntime/7.32.3
<input checked="" type="checkbox"/>	Accept	③	/*/*
<input checked="" type="checkbox"/>	Accept-Encoding	③	gzip, deflate, br
<input checked="" type="checkbox"/>	Connection	③	keep-alive
<input checked="" type="checkbox"/>	authorization		7b38cbe7749744e9a25b93c12fdd19bc

6. 确保服务端处于运行状态

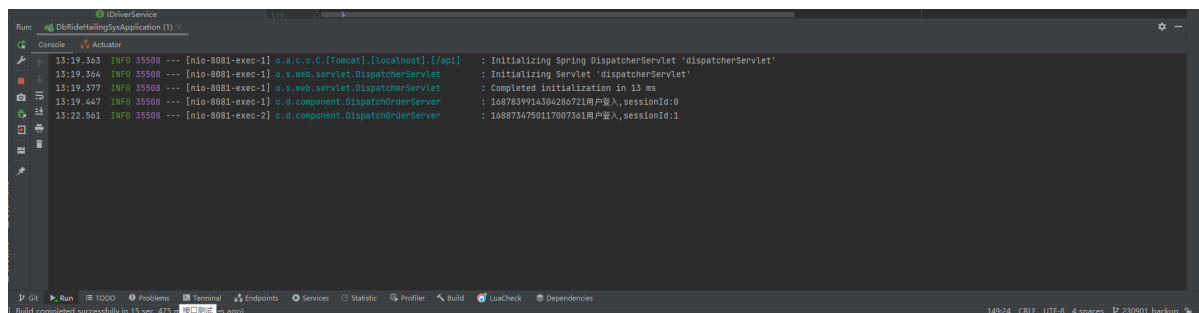
2.2.3. 测试流程

1. 建立WebSocket连接

向服务端的指定路径分别建立与司机、乘客客户端（Postman模拟)的ws连接。

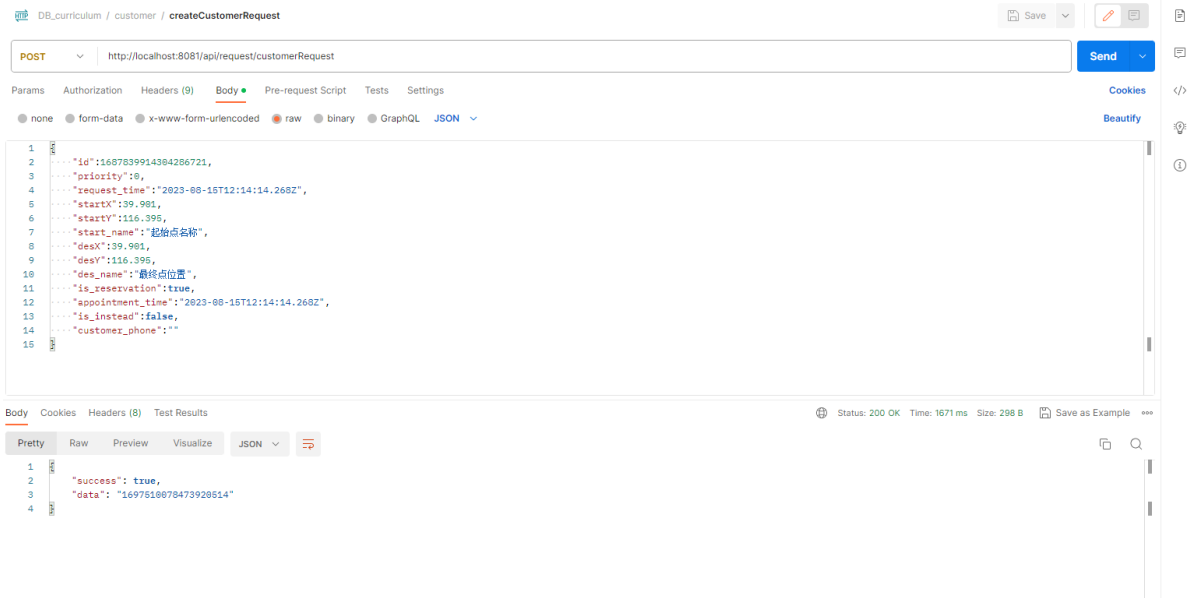


服务端我配置了控制台日志，可以看到用户已经连接上了。



2. 乘客用户发起约车请求

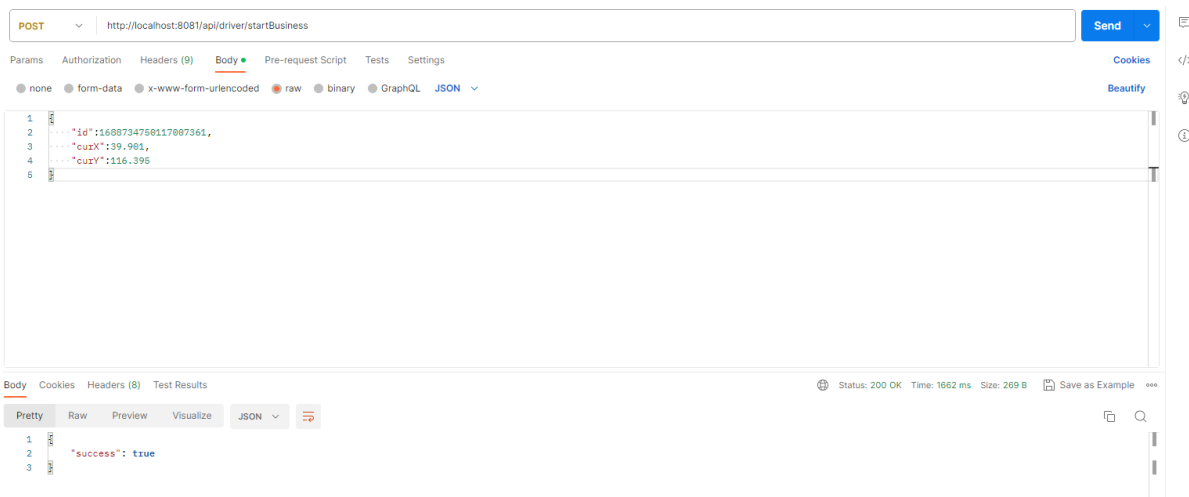
你可以自己创建这个文件，或者导入我附录中的文件。请求的参数可以自己修改。



发出请求后，获得带有请求id的响应。请注意保留这个请求id，这是为了测试后续的接口。

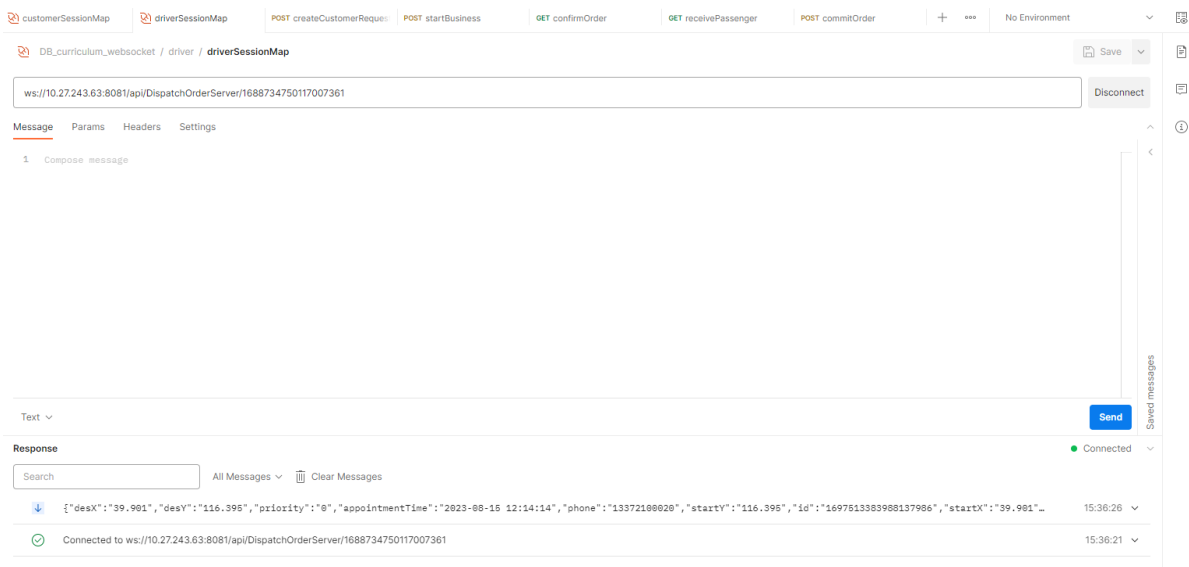
3. 司机开始听单

参数可以自己修改



由于当前系统中只有这一个刚开始听单的空闲司机，于是刚刚发出的约车请求一定会派发给该司机。

我们回到ws的连接中查看，果然，分配给了该司机。

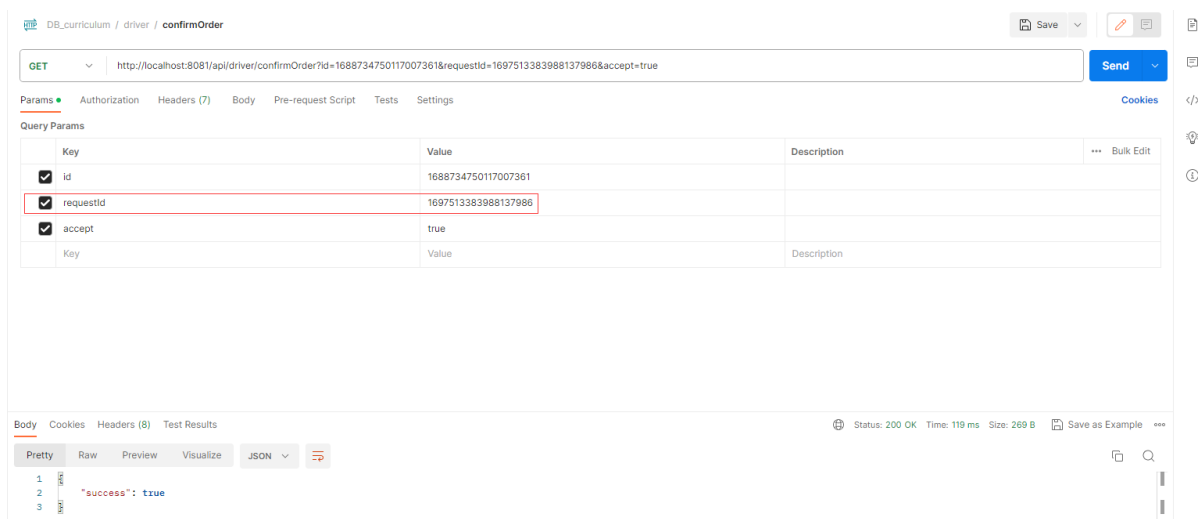


随后，控制台的日志中也打印了将订单分配给该司机的过程。

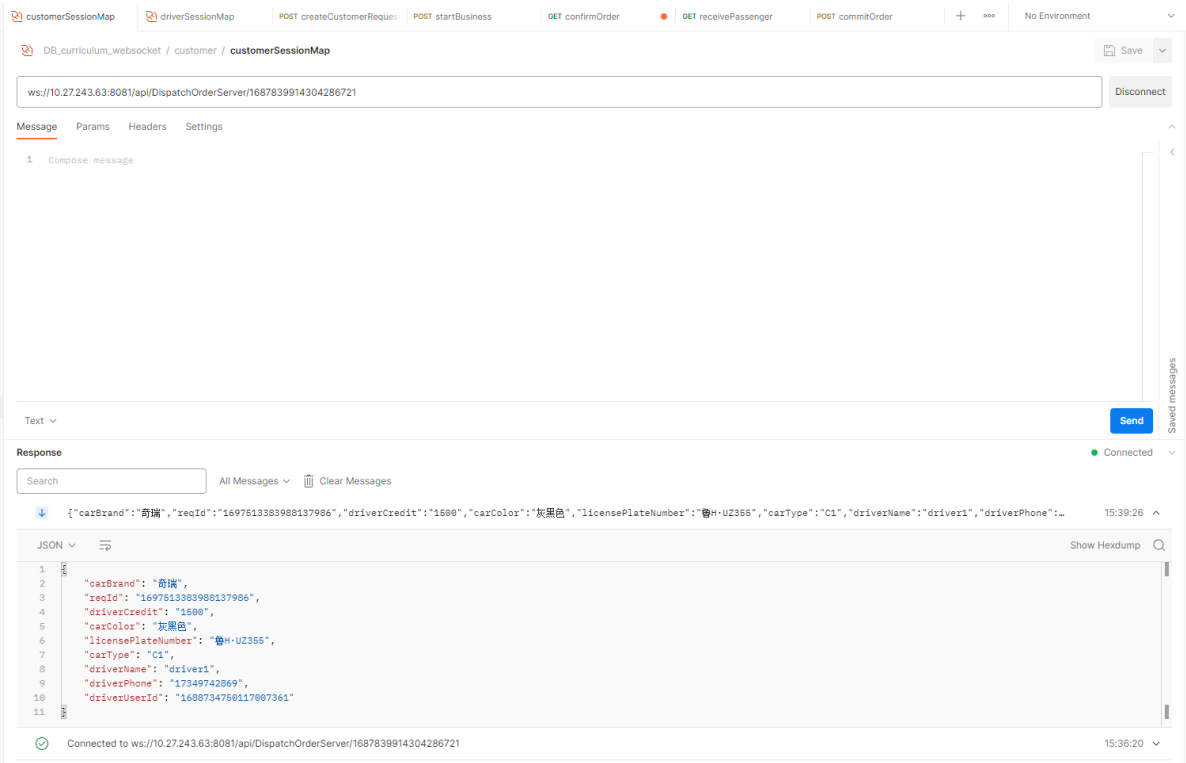
```
36:20.242 [WARN] 35088 --- [pool-1-thread-1] c.d.R.RequestDao.queryRequestByIdByReqId : ==> Parameters: 1697513383988137986(Long)
36:20.251 [DEBUG] 35088 --- [pool-1-thread-1] c.d.R.RequestDao.queryRequestByIdByReqId : <== Total: 1
36:20.252 [DEBUG] 35088 --- [pool-1-thread-1] c.d.service.impl.IndentServiceImpl : 正在准备分配订单, 订单id: 1697513383988137986
36:20.253 [DEBUG] 35088 --- [pool-1-thread-1] c.d.service.impl.IndentServiceImpl : 当前排队中司机数量: 1
36:20.259 [DEBUG] 35088 --- [pool-1-thread-1] c.d.service.impl.IndentServiceImpl : 司机1688734750117007361匹配的列表为:
36:20.259 [DEBUG] 35088 --- [pool-1-thread-1] c.d.service.impl.IndentServiceImpl : 分配司机id: 1688734750117007361
36:20.260 [DEBUG] 35088 --- [pool-1-thread-1] c.d.dao.UserDao.queryPhoneByUID : ==> Preparing: select phone from user where id = ? and deleted = 0
36:20.260 [DEBUG] 35088 --- [pool-1-thread-1] c.d.dao.UserDao.queryPhoneByUID : ==> Parameters: 1687839914304280721(Long)
36:20.260 [DEBUG] 35088 --- [pool-1-thread-1] c.d.dao.UserDao.queryPhoneByUID : <== Total: 1
36:20.264 [DEBUG] 35088 --- [pool-1-thread-1] c.d.d.R.queryReservationByRequestId : ==> Preparing: select * from reservation where request_id = ? and deleted=0
36:20.264 [DEBUG] 35088 --- [pool-1-thread-1] c.d.d.R.queryReservationByRequestId : ==> Parameters: 1697513383988137986(Long)
36:20.270 [DEBUG] 35088 --- [pool-1-thread-1] c.d.d.R.queryReservationByRequestId : <== Total: 1
36:20.270 [DEBUG] 35088 --- [pool-1-thread-1] c.d.service.impl.IndentServiceImpl : Reservation(id=1697513384080001201, requestId=1697513383988137986, reserveTime=2023-08-15T12:14:14, deleted=0, version=1)
36:20.271 [DEBUG] 35088 --- [pool-1-thread-1] c.d.service.impl.IndentServiceImpl : 将订单: 1697513383988137986分配给司机: 1688734750117007361
36:20.301 [INFO] 35088 --- [pool-1-thread-1] c.d.b.w.Broadcast.websocketBroadcast : 向客户端3发送订单结果StringRequestMessageForDriver(id=1697513383988137986, priority=0, startX=39.901, startY=116.395, startName=赵德成名称, de
```

4. 司机接受订单

把ws中的请求id复制到这个测试文件中的requestId参数中。

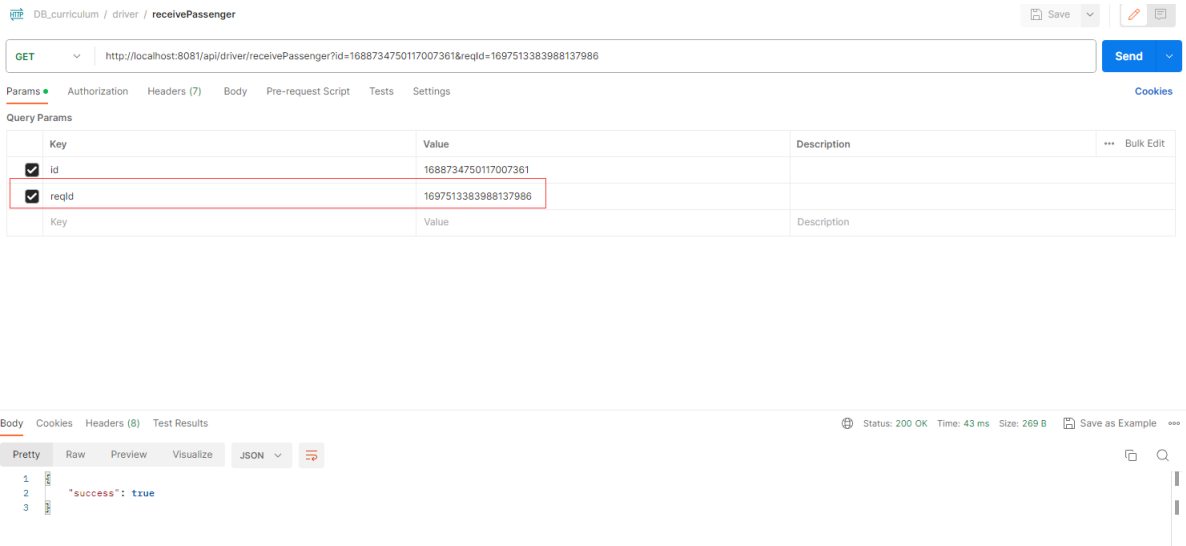


回到乘客的ws连接中，发现已经将司机和车辆信息发送给了乘客客户端



5. 司机接到乘客

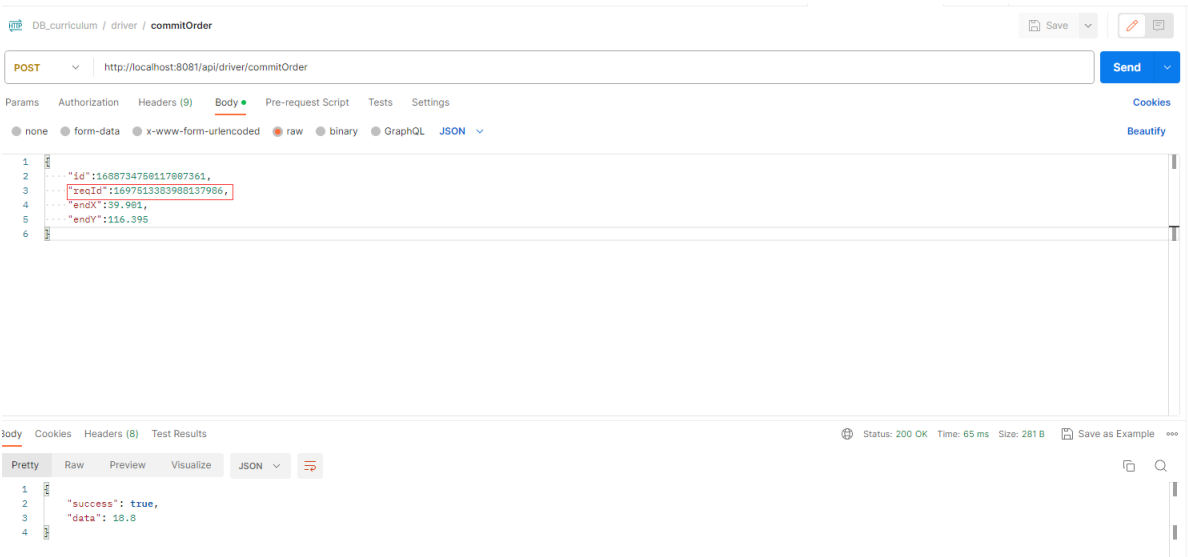
把请求id复制到reqId字段中。



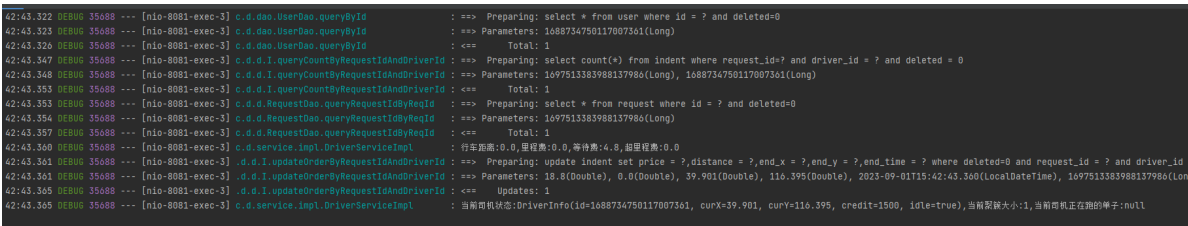
这个环节的业务比较简单，我就没有在控制台打日志。

6. 司机到达目的地后提交订单

把请求id复制到reqId字段中。



在服务端控制台打印了日志，至此，该订单结束。



注：如果你想测试别的接口，请自己编写测试文件，系统庞大、接口繁杂，在有限的时间内仅凭一个人很难完成所有测试工作。

2.3. 压力测试

2.3.1. 测试目标

数据量：1000乘客发起约车请求、1000司机开始听单

具体服务：派单业务，服务端为这1000个约车请求以及司机分配订单

2.3.2. 环境准备

环境准备分为三个部分：MySQL环境、Redis环境以及Jmeter环境

2.3.2.1. MySQL环境准备

派单业务需要同时有两部分数据才能进行：首先要有乘客约车请求，然后要有向服务端表达过听单意愿的司机，因此我们要先在数据库中准备1000个乘客与1000个司机。注意相关的表也要对应的插入数据。

执行附录中SQL脚本: `createBatchUser.sql`。该SQL脚本中创建了4个Procedure，分别用来循环向相关的表中插入：乘客实体，司机实体，车辆实体，司机所属车辆的联系。后二者是由于后端有校验，没有注册过车辆的司机不能接单。

至此MySQL环境准备完毕。

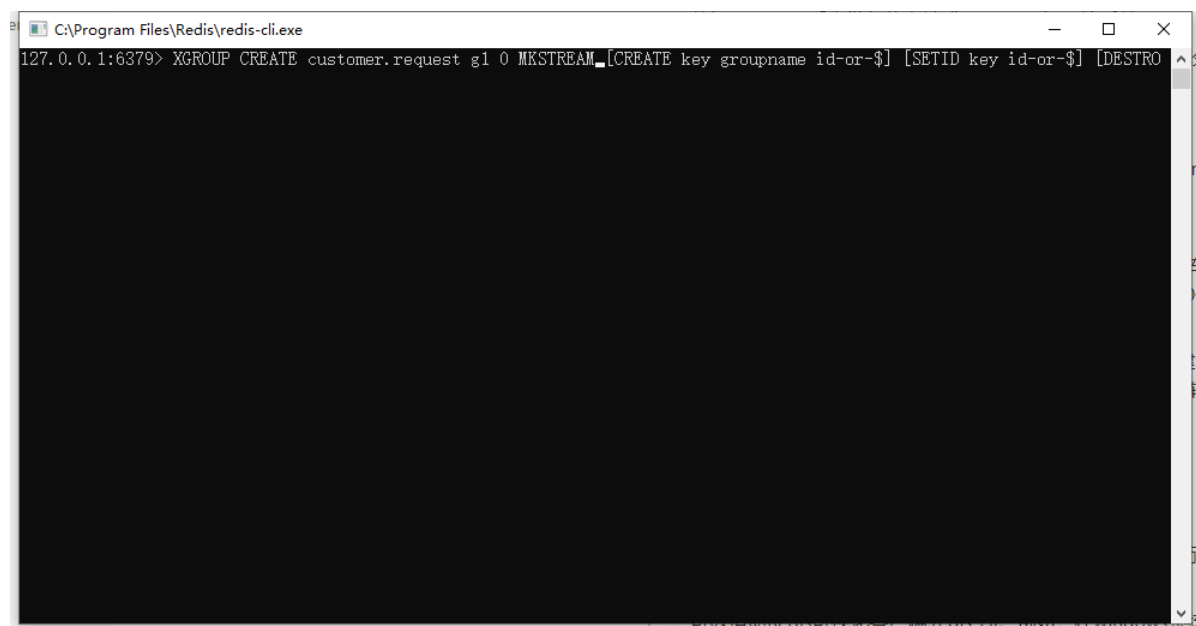
2.3.2.2. Redis环境准备

由于派单业务的实现需要借助Redis中的Stream消息队列，而消息队列的创建是在服务端部署上线前就在服务器上完成的。所以需要手动创建。

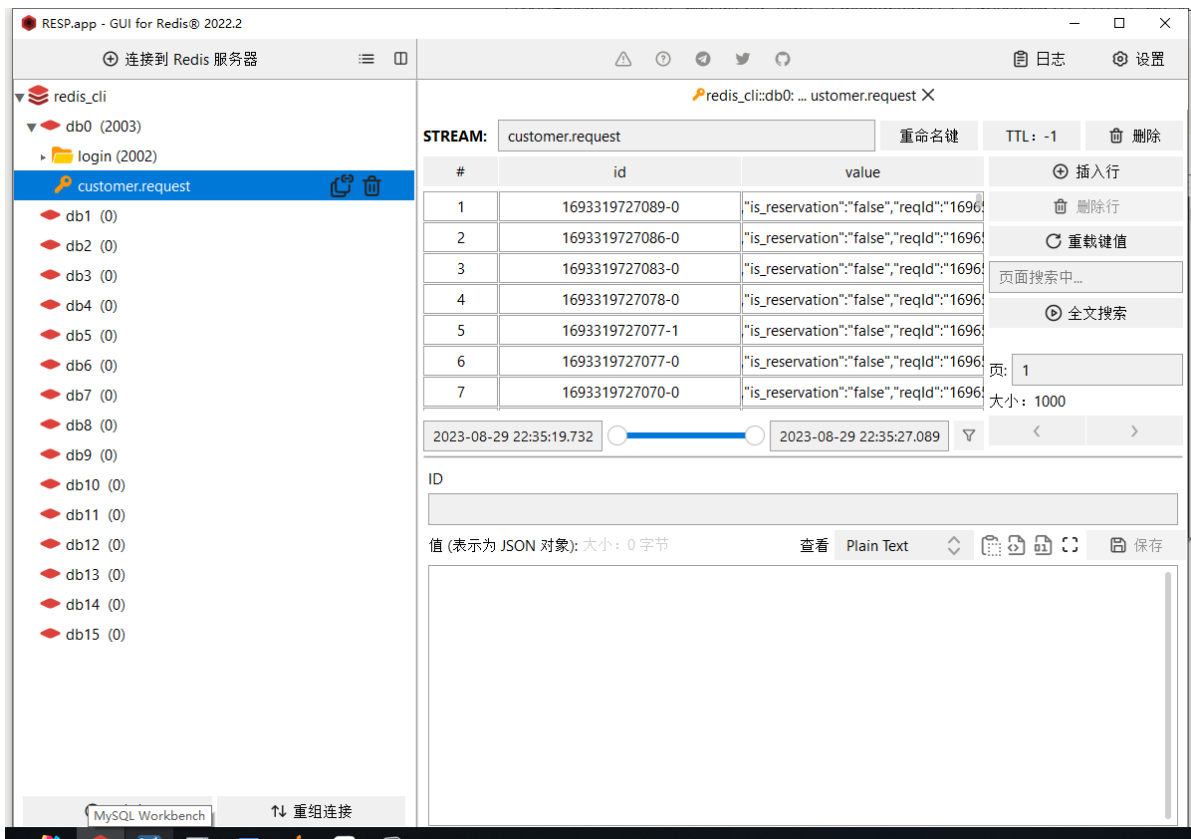
可以借助Redis可视化客户端:redis-cli。例如，在windows系统下，启动Redis服务，打开客户端，键入以下命令：

```
XGROUP CREATE customer.request g1 0 MKSTREAM
```

这里的消息队列名称，包括服务端的消费者组名称，都是写死的常量。



之后可以在Redis的图形化客户端RESP中看到这个消息队列



至此Redis的环境准备就绪。

2.3.2.3. Jmeter的环境准备

这里假设从零运行这个项目，以此为基础展示如何配置Jmeter数据源。

首先打开Jmeter，导入附录中的脚本：`RideHailing-DispatchOrder-test.jmx`。随后可以看到已经创建了4个线程组，这4个线程组均向服务器指定端口的指定路径发送Http请求，分别是:乘客登陆，司机登陆，乘客发起约车请求，司机发起开始听单。

由于后两个线程组发送的Http请求附带有用户个人的信息，也就是需要鉴权的接口，因此要在请求头中携带token，所以Jmeter环境的准备第一步就是获取Token。但获取方式的Token的方式仅有一种：即请求对应的登陆业务接口，然后才能拿到服务端生成的token。因此在此之前我们要配置首个数据源：用户登陆的数据源。

用户登陆时需要以下参数:

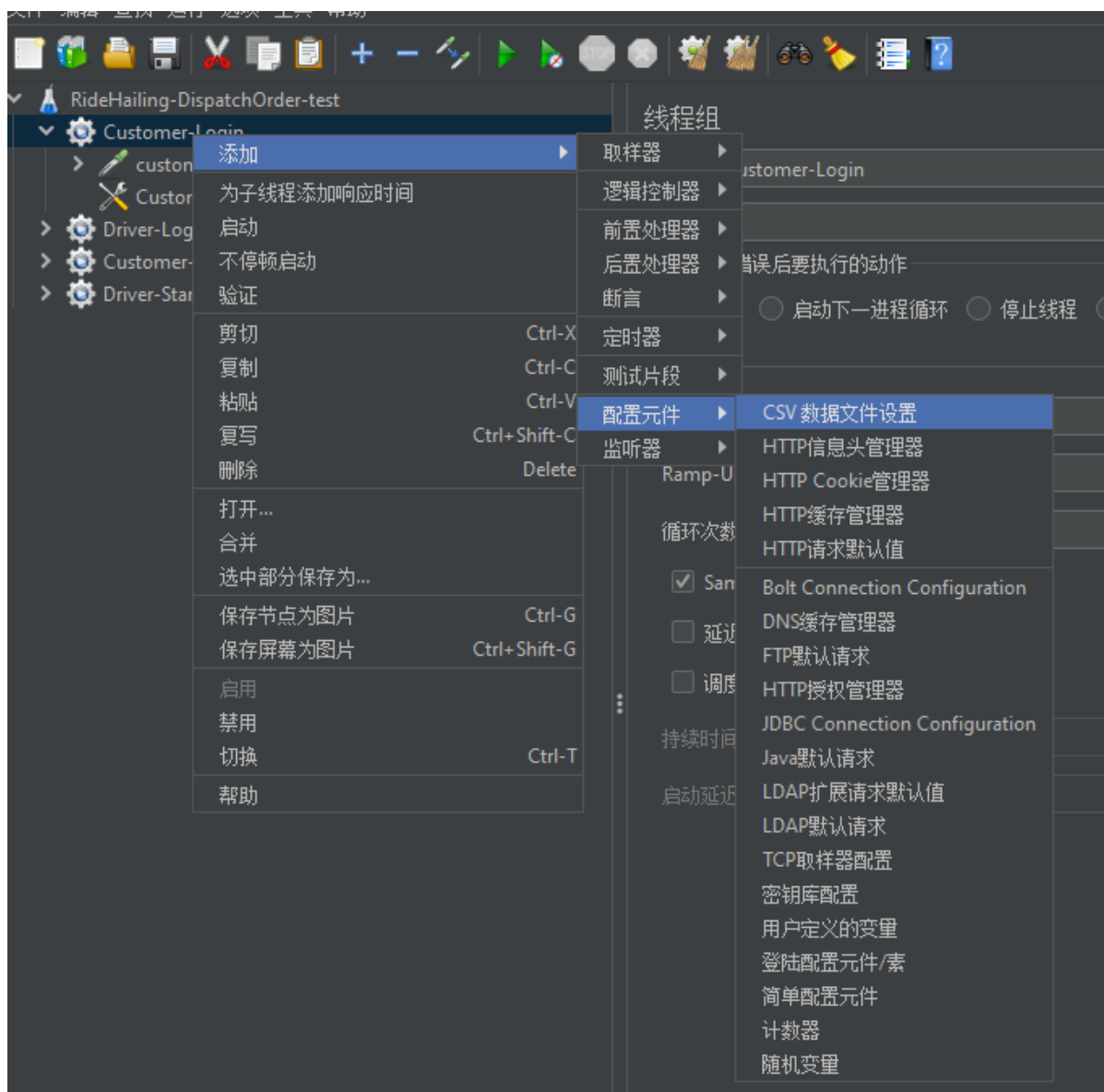
```
{
  "phone": phone, //用户手机号
  "password": password, //用户密码
  "code": code, //用户验证码
}
```

```
"role_id":role_id//用户类型,1为司机,2为乘客
}
```

由于服务端的逻辑是，登陆时只要提供密码或验证码中的至少一个即可，因此我们配置的数据源中仅选择提供密码，置空验证码。

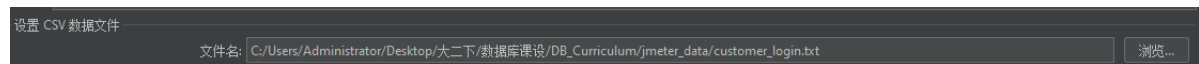
因此最终的请求中，需要动态的获取三个来自数据源的数据：手机号，密码，用户类型。

对对应线程组添加配置元件：[CSV数据文件设置](#)。这个数据文件不一定要是.csv格式的，这里我选择了.txt的文本，也是可以的。



随后在变量名称中配置变量名，以逗号分隔，**注意不要使用中文**，没有对应编码，识别不出来的。

最后在文件路径处选择对应数据源文件。

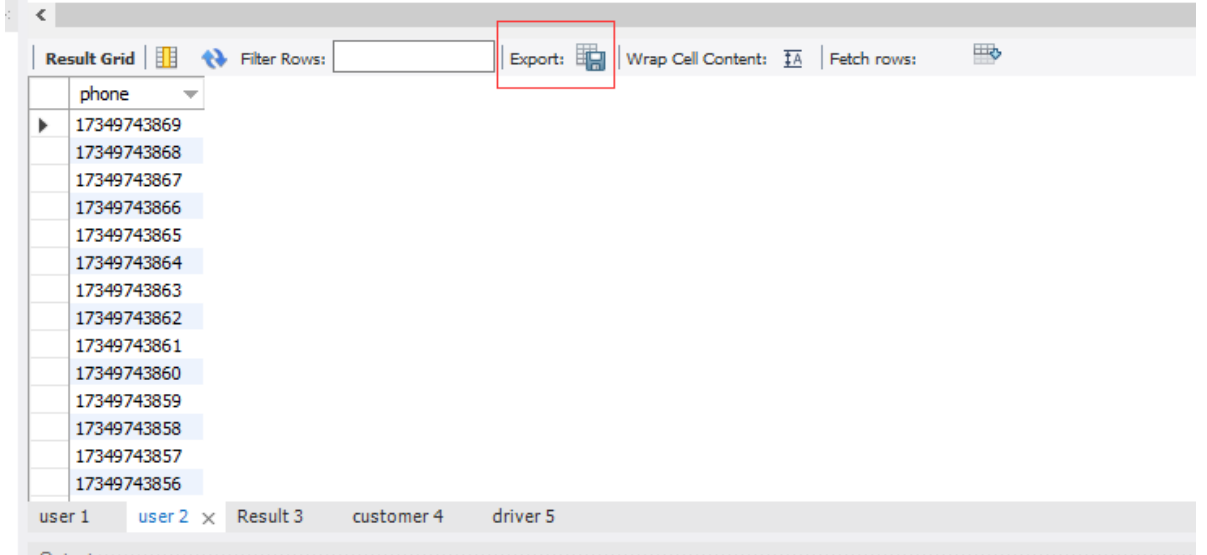


这个路径不一定适配所有电脑，最好在你本机创建一个专门存放jmeter数据源的文件夹，把需要的文件放进去。

那么这个文件从何来？我使用了MySQLWorkBench中的Export导出功能，将选择的数据导出为.txt文件。

例如，我们之前执行的SQL脚本：`createBatchUser.sql` 最终查询了用户的手机号，我们可以直接在这个查询结果的界面将数据导出。格式可以自己选择。

```
69
70 select * from user;
71 select phone from user;
72 select count(*) from user;
73 select * from customer;
74 select * from driver;
75
76 ## insert into user(id,name,phone,password,credit,role_id,deleted,version) values (1688734750
77 ## delete from user where id = 1688734750117008362
78
79 ##INSERT INTO driver(id,comment_rate,income,user_id,deleted,version,license) VALUES (16887347
80 ##delete from driver where user_id = 1688734750117008362
81
82
83
```



The screenshot shows a database query tool interface. At the top, there is a SQL query editor with lines 69 through 83. Below the editor is a toolbar with buttons for 'Result Grid', 'Filter Rows', 'Export', 'Wrap Cell Content', and 'Fetch rows'. The 'Export' button is highlighted with a red box. Below the toolbar is a table with one column labeled 'phone' and 15 rows of phone numbers. At the bottom, there is a tab bar with tabs for 'user 1', 'user 2', 'Result 3', 'customer 4', and 'driver 5'.

phone
17349743869
17349743868
17349743867
17349743866
17349743865
17349743864
17349743863
17349743862
17349743861
17349743860
17349743859
17349743858
17349743857
17349743856

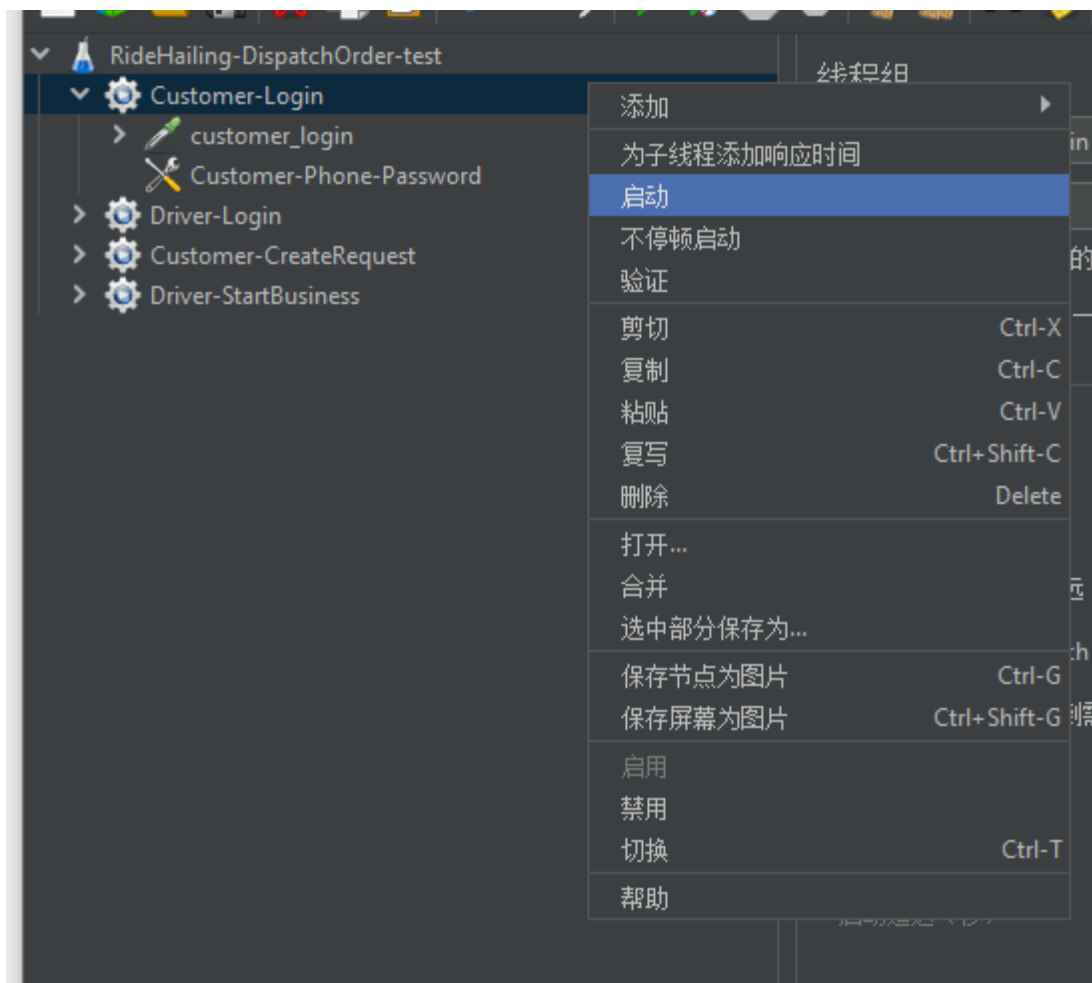
然后将导出文件的所有行适配成jmeter软件要求的格式:不同变量用逗号分隔，一组变量结束换行开始下一行变量，这里以 `customer_login.txt` 为例：

customer_login.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
17349742870,1234567,1
17349742871,1234567,1
17349742872,1234567,1
17349742873,1234567,1
17349742874,1234567,1
17349742875,1234567,1
17349742876,1234567,1
17349742877,1234567,1
17349742878,1234567,1
17349742879,1234567,1
17349742880,1234567,1
17349742881,1234567,1
17349742882,1234567,1
17349742883,1234567,1
17349742884,1234567,1
17349742885,1234567,1
17349742886,1234567,1
17349742887,1234567,1
17349742888,1234567,1
17349742889,1234567,1
17349742890,1234567,1
```

由于我为所有用户注册的密码的对应明文都为 `"1234567"`，因此密码可以写死，如果是乘客，那么用户类型字段role_id应该为1。

如此配置两份登陆信息数据源（司机与乘客），然后准备使用jmeter批量请求登陆。



分别启动 `Customer-Login` 和 `Driver-Login` 线程组，这将分别向服务器发送1000条登陆请求。随后token将存入Redis。服务端配置了将日志输出到指定文件的功能，因此我选择将token以日志的方式输出到某个文件中。

```
//记录token
if(roleId==Customer_Role){
    recordLog( log: token+'\n',CustomerTokenFilePath, append: true);
}
else {
    recordLog( log: token + '\n', DriverTokenFilePath, append: true);
}
```

注意我输出的路径可能不适配你的计算机，因此最好将服务端输出路径的常量配置修改为你的对应路径。

```

public static String CustomerTokenFilePath = "C:\\Users\\Administrator\\Desktop\\大二下\\数据库课设\\DB_Curriculum\\jmeter_data\\CustomerToken.txt";

2 usages
public static String DriverTokenFilePath = "C:\\Users\\Administrator\\Desktop\\大二下\\数据库课设\\DB_Curriculum\\jmeter_data\\DriverToken.txt";

```

(修改上图配置)

这样将获取登陆用户的token，作为请求头中 `authorization` 字段的数据源。

随后是发起约车请求和开始听单的请求。这里需要动态配置的字段为用户id，地理位置坐标（约车开始位置，目标位置，司机当前位置等），以及请求头中的token。

用户id方面，运行附录脚本：`genDriverIdAndCustomerId.sql`，然后通过上文描述的MySQLWorkBench的Export功能导出对应数据源文件。

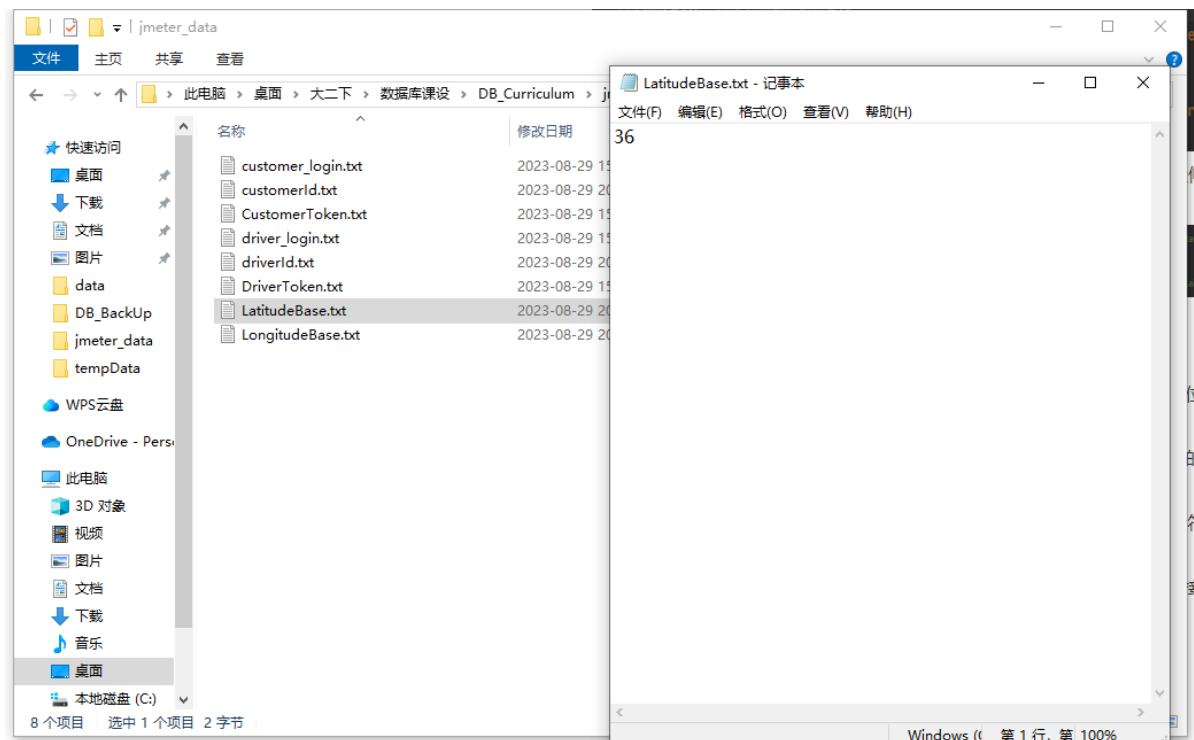
地理位置坐标方便，可以新建txt文件设置基准位置，然后在后方拼接jmeter随机字符串函数RandomString返回的随机字符串来实现随机均匀地理坐标。

例如我身处济南，就将济南的大致地理坐标：36，117设置为基准，在此基础上拼接随机字符串。

```

"startX": "${longitude}.${__RandomString(4,1234567890,,)}",
"startY": "${latitude}.${__RandomString(4,1234567890,,)}",

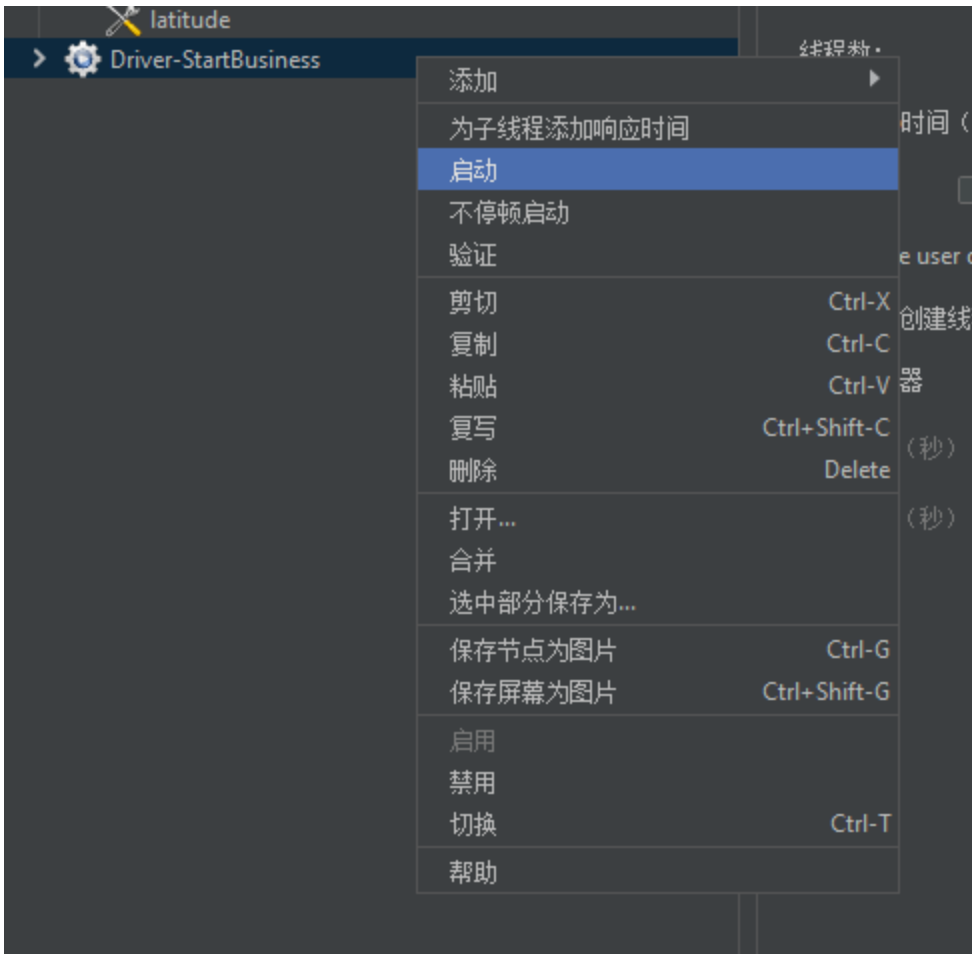
```



至此Jmeter的环境准备就绪。

2.3.3. 压力测试流程以及观测压测结果

由于在环境准备中我们已经运行了两类用户登陆的线程组，因此这里不再需要了。



运行司机开始听单线程组，将所有司机列入服务端维护的空闲司机的数据结构中。

但只有空闲司机还不够，再运行乘客发起约车请求的线程组，请求约车。

然后观察 `C:\Users\Administrator\RideHailingDispatchOrder.txt` 文件，可以看到1000条派发订单的日志。

2023-08-29T22:35:22.643 将订单:1696531980911718401分配给司机:1688734750117008518
2023-08-29T22:35:23.775 将订单:1696531984044863491分配给司机:1688734750117008378
2023-08-29T22:35:24.064 将订单:1696531984044863490分配给司机:1688734750117008491
2023-08-29T22:35:24.525 将订单:1696531983981948935分配给司机:1688734750117008491
2023-08-29T22:35:26.150 将订单:1696531984061640707分配给司机:1688734750117008491
2023-08-29T22:35:26.203 将订单:1696531983973560322分配给司机:1688734750117008378
2023-08-29T22:35:26.470 将订单:1696531984111972355分配给司机:1688734750117008378
2023-08-29T22:35:26.507 将订单:1696531983981948932分配给司机:1688734750117008378
2023-08-29T22:35:26.540 将订单:1696531983981948937分配给司机:1688734750117008513
2023-08-29T22:35:26.579 将订单:1696531983981948931分配给司机:1688734750117008513
2023-08-29T22:35:26.603 将订单:1696531983981948933分配给司机:1688734750117008491
2023-08-29T22:35:26.845 将订单:1696531983344414723分配给司机:1688734750117008565
2023-08-29T22:35:26.932 将订单:1696531983981948934分配给司机:1688734750117008513
2023-08-29T22:35:26.962 将订单:1696531983973560324分配给司机:1688734750117008565
2023-08-29T22:35:27.046 将订单:1696531983960977411分配给司机:1688734750117008378
2023-08-29T22:35:27.087 将订单:1696531983960977410分配给司机:1688734750117008565
2023-08-29T22:35:27.108 将订单:1696531983872897028分配给司机:1688734750117008476
2023-08-29T22:35:27.118 将订单:1696531983830953986分配给司机:1688734750117008513
2023-08-29T22:35:27.132 将订单:1696531983868702721分配给司机:1688734750117008378
2023-08-29T22:35:27.166 将订单:1696531983809982466分配给司机:1688734750117008513
2023-08-29T22:35:27.175 将订单:1696531984111972354分配给司机:1688734750117008378
2023-08-29T22:35:27.192 将订单:1696531983776428033分配给司机:1688734750117008491
2023-08-29T22:35:27.202 将订单:1696531983872897026分配给司机:1688734750117008378
2023-08-29T22:35:27.211 将订单:1696531983675764741分配给司机:1688734750117008513
2023-08-29T22:35:27.228 将订单:1696531983872897027分配给司机:1688734750117008513
2023-08-29T22:35:27.237 将订单:1696531983872897029分配给司机:1688734750117008513
2023-08-29T22:35:27.246 将订单:1696531983302471684分配给司机:1688734750117008491
2023-08-29T22:35:27.266 将订单:1696531983927422978分配给司机:1688734750117008491
2023-08-29T22:35:27.277 将订单:1696531983302471683分配给司机:1688734750117008518
2023-08-29T22:35:27.288 将订单:1696531984061640706分配给司机:1688734750117008513
2023-08-29T22:35:27.298 将订单:1696531983260528641分配给司机:1688734750117008513
2023-08-29T22:35:27.312 将订单:1696531984044863492分配给司机:1688734750117008378
2023-08-29T22:35:27.330 将订单:1696531981037547525分配给司机:1688734750117008549
2023-08-29T22:35:27.338 将订单:1696531980588756995分配给司机:1688734750117008491
2023-08-29T22:35:27.347 将订单:1696531984061640708分配给司机:1688734750117008476
2023-08-29T22:35:27.362 将订单:1696531983004676097分配给司机:1688734750117008513
2023-08-29T22:35:27.373 将订单:1696531983960977409分配给司机:1688734750117008491
2023-08-29T22:35:27.388 将订单:1696531983369580547分配给司机:1688734750117008549
2023-08-29T22:35:27.396 将订单:1696531983461855234分配给司机:1688734750117008476
2023-08-29T22:35:27.410 将订单:1696531980081246212分配给司机:1688734750117008378
2023-08-29T22:35:27.427 将订单:1696531983507992578分配给司机:1688734750117008513
2023-08-29T22:35:27.435 将订单:1696531983507992579分配给司机:1688734750117008378

也可以观察Jmeter的汇总报告进行性能评估。

至此完成整个派发订单业务的压测。