1 4、基于 Spring Cloud Alibaba + Vue

前端：Vue + Element UI + Mint UI + MUI

买家端（移动端）： Vue + Mint UI + MUI

卖家端（PC 端）： Vue + Element UI

后端：

Spring Boot + Spring Cloud Alibaba + MyBatis Plus + MySQL + Rocket MQ + Redis + Gateway + 第三方短信服务接口

2 项目启动步骤

1、启动虚拟机 RocketMQ

2、启动 Redis

3、启动 Nacos

4、启动 Product-service，提供商品服务

5、启动 Order-service，提供订单服务

6、启动 Sms-service，短信服务

7、启动 Account-service，账户服务

8、启动 Mq-service，MQ 服务

9、启动 Gateway，完成统一路径

10、启动买家端

11、启动卖家端

3 前端原型

前端原型（不需要对接后台，使用假数据完成的页面）

> 买家端

1、创建 Vue 工程

2、添加 axios 插件

3、安装 Mint UI

```
cnpm install mint-ui -S
```

4、main.js 中引入 Mint UI 组件

```
import Vue from 'vue'
import './plugins/axios'
import App from './App.vue'
import router from './router'
import store from './store'
import Mint from 'mint-ui'
import 'mint-ui/lib/style.css'

Vue.config.productionTip = false

Vue.use(Mint)

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

5、测试，App.vue 使用 Mint UI 组件

```
<template>
  <div id="app">
    <mt-button @click.native="test">按钮</mt-
button>
    <div id="nav">
      <router-link to="/">Home</router-link> |
      <router-link to="/about">About</router-
link>
    </div>
```

```
    <router-view/>
  </div>
</template>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-
serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}

#nav {
  padding: 30px;
}

#nav a {
  font-weight: bold;
  color: #2c3e50;
}

#nav a.router-link-exact-active {
  color: #42b983;
}
</style>

<script>
  export default {
    methods: {
      test:function () {
        this.$toast('Hello World');
```

```
        }
      }
    }
</script>
```

安装成功

6、测试 axios

```
test:function () {

axios.get('http://localhost:8181/index').then(f
unction (resp) {
      console.log(resp)
  })
}
```

```
@GetMapping("/index")
public String index(){
    return "index";
}
```

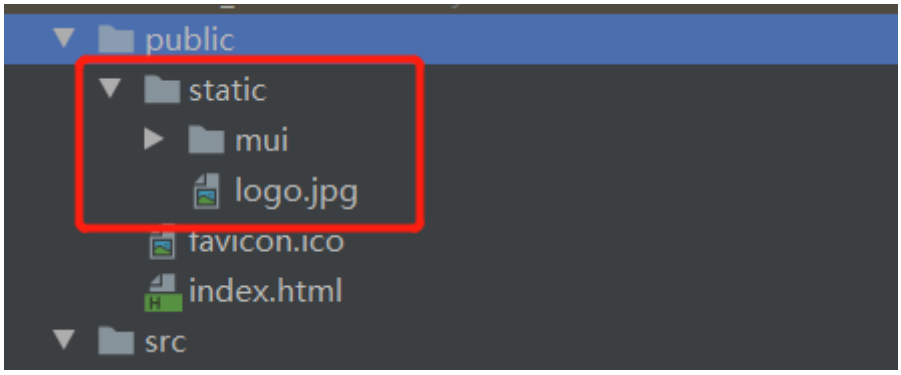
```
package com.southwind.configuration;

import
org.springframework.context.annotation.Configur
ation;
import
org.springframework.web.servlet.config.annotati
on.CorsRegistry;
import
org.springframework.web.servlet.config.annotati
on.WebMvcConfigurer;

@Configuration
public class CrosConfiguration implements
WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry
registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("GET", "HEAD",
"POST", "PUT", "DELETE", "OPTIONS")
                .allowCredentials(true)
                .maxAge(3600)
                .allowedHeaders("*");
    }
}
```

App.vue?234e:40

```
{data: "index", status: 200, statusText: "", headers: {…}, config: {…}, …}
  config: {transformRequest: {…}, transformResponse: {…}, timeout: 0, xsrfCookieName: "XS…
  data: "index"
  headers: {content-length: "5", content-type: "application/json"}
  request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XML…
  status: 200
  statusText: ""
  __proto__: Object
```

测试成功

7、引入 mui，导入静态文件，注意位置 public/static/



8、main.js 引入相关依赖

```
import Vue from 'vue'
import './plugins/axios'
import App from './App.vue'
import router from './router'
import store from './store'
import Mint from 'mint-ui'
import 'mint-ui/lib/style.css'
import '../public/static/mui/css/mui.min.css'

Vue.config.productionTip = false

Vue.use(Mint)

new Vue({
  router,
```

```
  store,
  render: h => h(App)
}).$mount('#app')
```

9、测试，App.vue 使用 mui 组件

```
<template>
  <div id="app">
    <mt-button @click.native="test">按钮</mt-
button>
    <button type="button" class="mui-btn mui-
btn-danger">红色</button>
    <div id="nav">
      <router-link to="/">Home</router-link> |
      <router-link to="/about">About</router-
link>
    </div>
    <router-view/>
  </div>
</template>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-
serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}

#nav {
  padding: 30px;
```

```
}

#nav a {
  font-weight: bold;
  color: #2c3e50;
}

#nav a.router-link-exact-active {
  color: #42b983;
}
</style>

<script>
  export default {
    methods: {
      test:function () {

 axios.get('http://localhost:8181/index').then(
function (resp) {
            console.log(resp)
        })
      }
    }
  }
</script>
```

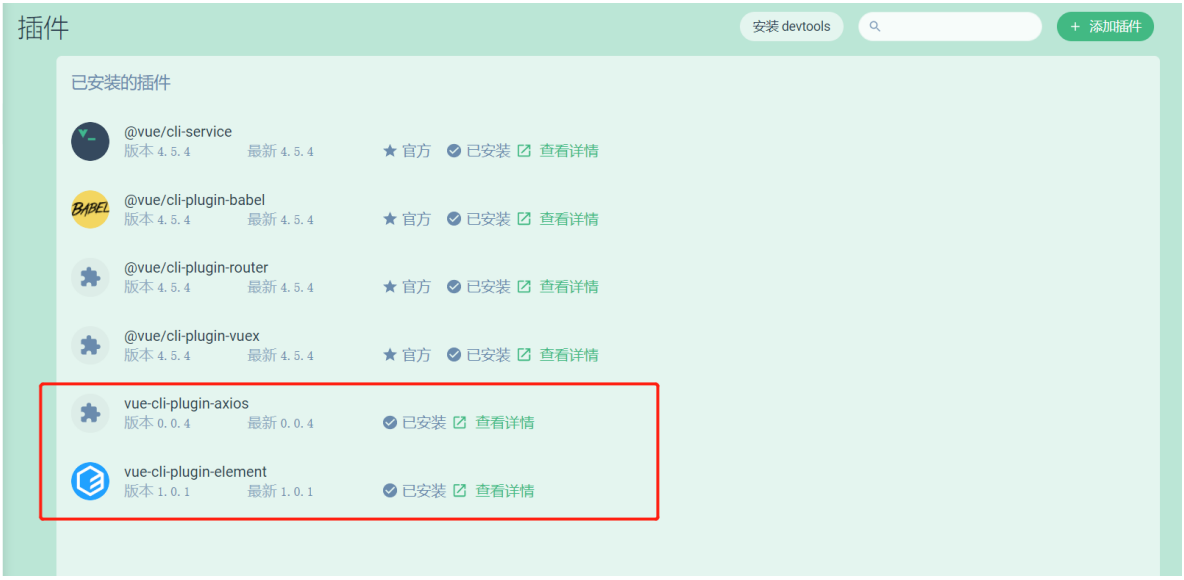
添加成功，takeout_buyer 基于移动端的买家前端工程 Vue + Mint UI + mui + axios 环境搭建成功。

接下来完成前端页面原型的开发，**使用假数据把用户交互界面做出来。**

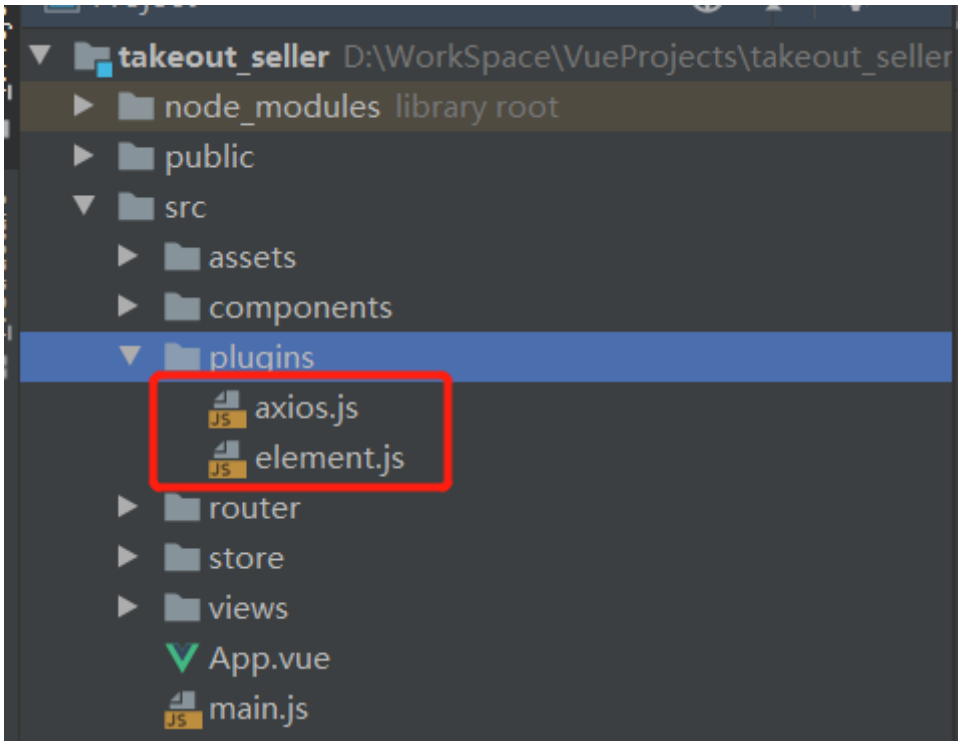
> 卖家端

1、创建 Vue 工程

2、添加 Element UI、axios 插件



3、将工程导入 IDEA，启动



4、用假数据写静态原型页面

4 MyBatis 和 MyBatis Plus 整合开发

单独使用 MyBatis 或者 MyBatis Plus，Mapper 接口只需要在启动类进行注册即可，配置文件不需要进行任何配置。

```
package com.southwind;

import
org.mybatis.spring.annotation.MapperScan;
import
org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBo
otApplication;

@SpringBootApplication
@MapperScan("com.southwind.mapper")
public class OrderServiceApplication {

    public static void main(String[] args) {

 SpringApplication.run(OrderServiceApplication.
class, args);
    }

}
```

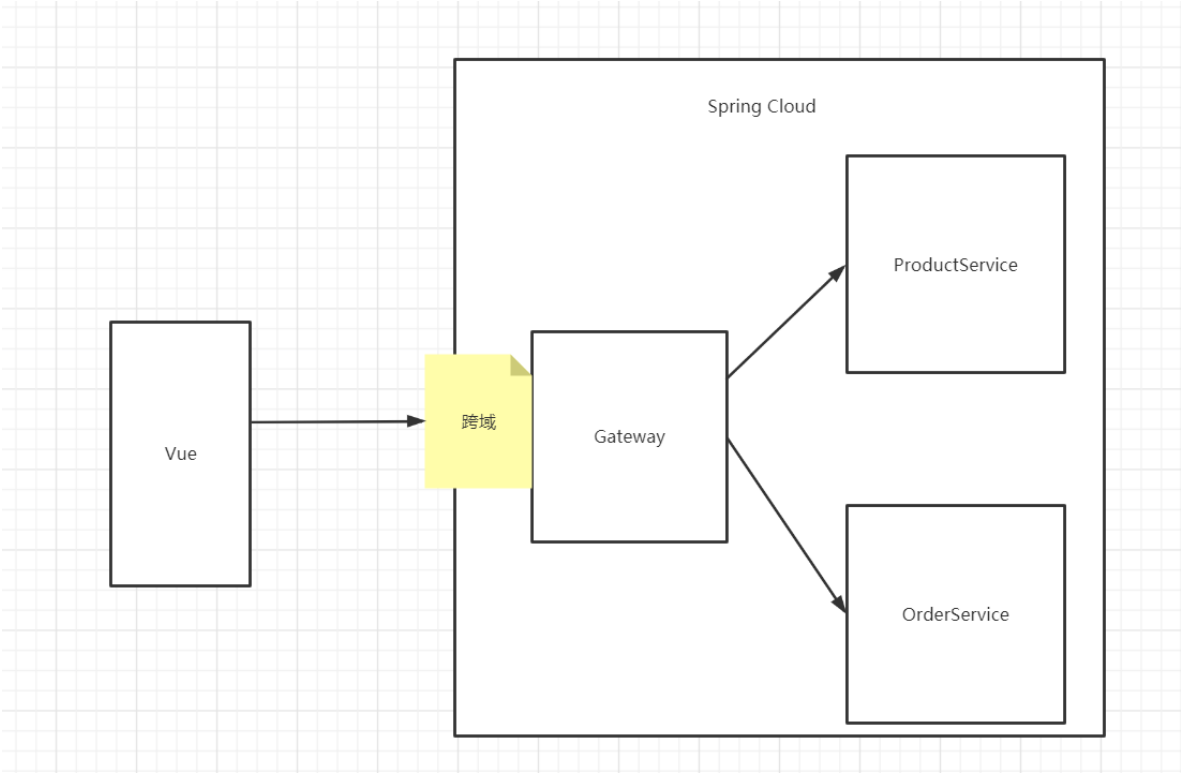
但是如果要整合 MyBatis 和 MyBatis Plus，就需要额外在 application.yml 中配置 Mapper.xml 文件的路径（用 MyBatis Plus 来配置）。

```
mybatis-plus:
  mapper-locations:
classpath:com/southwind/mapper/xml/*.xml
```

5 网关跨域

网关上游配置跨域，下游服务就不用配了

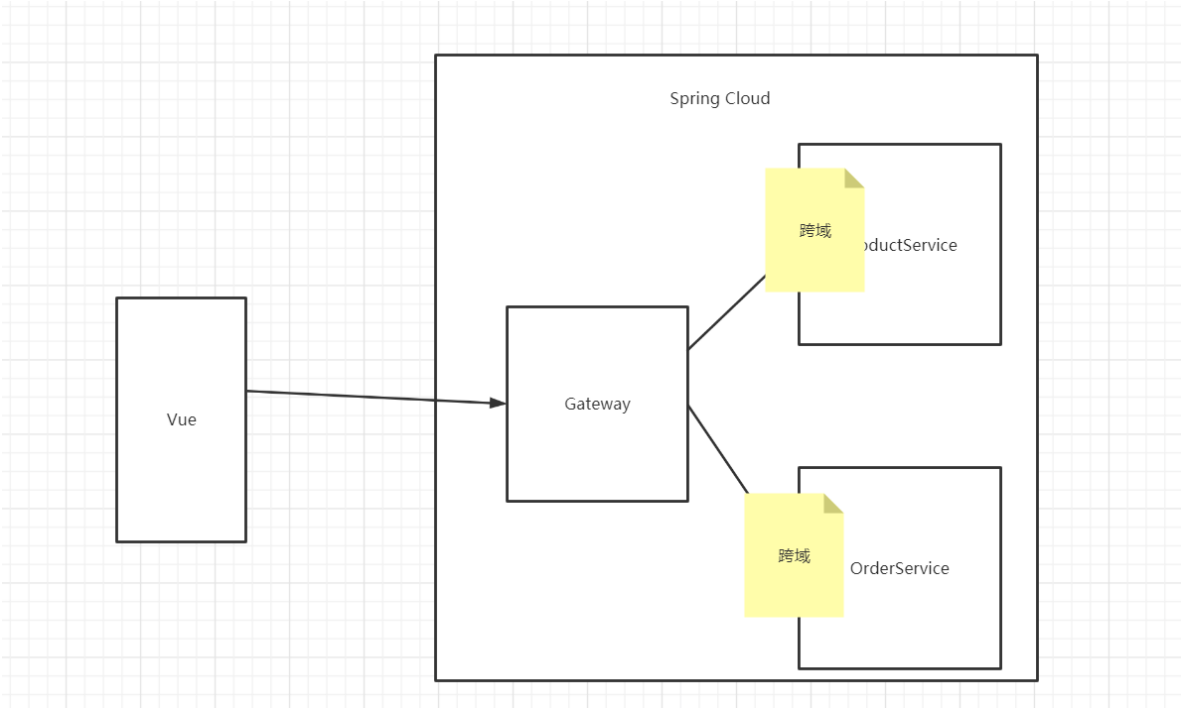
```
server:
  port: 8180
spring:
  application:
    name: gateway
  cloud:
    gateway:
      globalcors:
        cors-configurations:
          '[/**]':
            allowed-origins:
            - "http://localhost:8080"  #客户端（买家uri）
            - "http://localhost:8082"  #客户端（卖家uri）
            allowed-headers: "*"
            allowed-methods: "*"
            max-age: 3600
      discovery:
        locator:
          enabled: true
product-service:
  ribbon:
    NFLoadBalancerRuleClassName: com.southwind.configuration.NacosWeightedRule
```



网关上游不配，下游服务通过配置类进行配置

```
package com.southwind.configuration;

import
org.springframework.context.annotation.Configur
ation;
import
org.springframework.web.servlet.config.annotati
on.CorsRegistry;
import
org.springframework.web.servlet.config.annotati
on.WebMvcConfigurer;

@Configuration
public class CorsConfiguration implements
WebMvcConfigurer {

    @Override
```

```
    public void addCorsMappings(CorsRegistry
registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("GET", "HEAD",
"POST", "PUT", "DELETE", "OPTIONS")
                .allowCredentials(true)
                .maxAge(3600)
                .allowedHeaders("*");
    }

}
```



6 WebSocket

实现 Order-Service 和前端工程的实时通信，Order-Service 创建订单的时候，通过 WebSocket 向前端发送消息，进行通知，让前端完成相关的业务逻辑处理（播放提示音乐，完成页面跳转）

> 后端

1、pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-
websocket</artifactId>
</dependency>
```

2、创建 WebSocket

```
package com.southwind.service.impl;

import lombok.extern.slf4j.Slf4j;
import
org.springframework.stereotype.Component;

import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.ServerEndpoint;
import java.io.IOException;
import
java.util.concurrent.CopyOnWriteArraySet;

@Component
@ServerEndpoint("/webSocket")
```

```
@Slf4j
public class WebSocket {

    private Session session;

    private static
CopyOnWriteArraySet<WebSocket> webSocketSet =
new CopyOnWriteArraySet<>();

    @OnOpen
    public void onOpen(Session session){
        this.session = session;
        webSocketSet.add(this);
        log.info("【websocket消息】有新的连接，总
数：{}",webSocketSet.size());
    }

    @OnClose
    public void onClose(){
        webSocketSet.remove(this);
        log.info("【websocket】连接断开，总数：
{}",webSocketSet.size());
    }

    @OnMessage
    public void onMessage(String message){
        log.info("【websocket】收到客户端发来的消
息：{}",message);
    }

    public void sendMessage(String message){
        for(WebSocket webSocket:webSocketSet){
```

```
            log.info("【websocket消息】广播消息,
message={}",message);
            try {

 webSocket.session.getBasicRemote().sendText(me
ssage);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

3、配置 WebSocket

```
package com.southwind.configuration;

import
org.springframework.context.annotation.Bean;
import
org.springframework.stereotype.Component;
import
org.springframework.web.socket.server.standard.
ServerEndpointExporter;

@Component
public class WebSocketConfiguration {

    @Bean
    public ServerEndpointExporter
serverEndpointExporter(){
        return new ServerEndpointExporter();
```

```
    }
}
```

4、创建订单的时候，调用 WebSocket 发消息

```
//通知后台管理系统
this.webSocket.sendMessage("有新的订单");
```

> 前端

App.vue

```
<template>
  <div id="app">
    <router-view/>
    <audio hidden id="notice" :src="music">
</audio>
  </div>
</template>

<script>
import HelloWorld from
'./components/HelloWorld.vue'

export default {
  name: 'app',
  components: {
    HelloWorld
  },

  data() {
    return {
      websock: null,
      music:'alert.mp3',
```

```
    }
  },

  methods:{
    initWebSocket(){
      this.websock = new
WebSocket('ws://localhost:8180/order-
service/webSocket');
      this.websock.onmessage =
this.webSocketOnMessage;
      this.websock.onopen =
this.webSocketOnOpen;
      this.websock.onerror =
this.websocketonerror;
      this.websock.onclose =
this.webSocketClose;
    },
    webSocketOnOpen(event){
      console.log('建立连接')
    },
    webSocketOnMessage(event){
      document.getElementById('notice').play();
      const _this = this
      this.$alert('有新的订单', '消息', {
        confirmButtonText: '确定',
        callback: action => {
          _this.$router.push('/orderManage')
        }
      });
    },
    webSocketClose(event){
      console.log('连接关闭');
    }
```

```
  },
  created() {
    this.initWebSocket();
  },
  destroyed() {
    this.websock.close()
  }
}
</script>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial,
sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

7 ECharts 开发流程

7.1 前端原型搭建

安装 Echarts

1、执行命令

```
cnpm install echarts@4.9.0 --save
```

2、main.js 中引入

```
import echarts from 'echarts'
Vue.prototype.$echarts = echarts
```

3、代码

```
<template>
    <div id="myChart" :style="{width: '300px',
height: '300px'}"></div>
</template>
<script>
    export default {
        name: 'Echarts',
        data () {
            return {
                msg: 'Welcome use Echarts'
            }
        },
        mounted(){
            this.drawLine();
        },
        methods: {
            drawLine(){
                // 基于准备好的dom，初始化echarts实
例
                let myChart =
this.$echarts.init(document.getElementById('myC
hart'))
                // 绘制图表
                myChart.setOption({
```

```
                    title: { text: '在Vue中使用
echarts' },
                    tooltip: {},
                    xAxis: {
                        data: ["衬衫","羊毛
衫","雪纺衫","裤子","高跟鞋","袜子"]
                    },
                    yAxis: {},
                    series: [{
                        name: '销量',
                        type: 'bar',
                        data: [5, 20, 36, 10,
10, 20]
                    }]
                });
            }
        }
    }
</script>
```

7.2 后端项目开发

7.2.1 创建工程结构

ECharts 数据库

```
1、第一步统计有销量的日期
select distinct
DATE_FORMAT(order_detail.create_time, '%Y-%m-
%d') as dd from order_detail
```

2、将第1步的结果作为虚拟表查询商品在这些日期的数据

```
select pi.product_name,mm.dd
from product_info pi,
     (select distinct
DATE_FORMAT(order_detail.create_time, '%Y-%m-
%d') as dd from order_detail)
          as mm
where pi.product_name = '鸡汤米线';
```

3、在第2步基础上查询商品在当天的销量

```
select product_name as name,mm.dd as date,(
    select COALESCE(sum(product_quantity),0)
    from order_detail where
    pi.product_id = order_detail.product_id and
    DATE_FORMAT(order_detail.create_time, '%Y-
%m-%d') = mm.dd
    ) as count
from product_info pi,
     (select distinct
DATE_FORMAT(order_detail.create_time, '%Y-%m-
%d') as dd from order_detail)
          as mm
where pi.product_name = '皮蛋瘦肉粥'order by
mm.dd;
```