电商实时数仓（数据接口大屏展示）

版本：V3.0

# 第1章 数据可视化接口

## 1.1 设计思路

DWS 层把轻度聚合的结果保存到 ClickHouse 中，主要的目的就是提供即时的数据查询、统计、分析服务。这些统计服务一般会以两种形式呈现，一种是面向专业数据分析人员准备的 BI 工具，一种是面向非专业人员的更加直观的数据大屏。

本项目将面向 sugar 数据大屏开发数据接口服务。

## 1.2 需求梳理

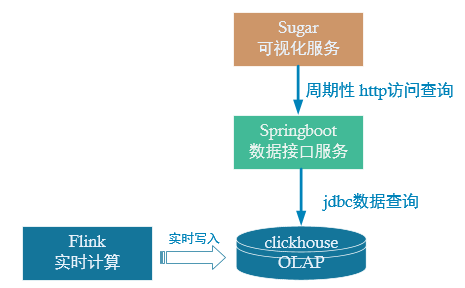
### 1.2.1 最终显示效果图







### 1.2.3 接口执行过程



DWS 层计算结果存储在 ClickHouse，本项目将开发数据接口查询ClickHouse中的数据，提供给 Sugar 进行大屏展示。这里主要有两项工作：

* 配置可视化大屏服务。
* 编写数据查询接口以供可视化大屏进行访问。

# 第2章 环境准备

## 2.1 sugar 简介

### 2.1.1 产品介绍

Sugar是百度云推出的敏捷 BI 和数据可视化平台，目标是解决报表和大屏的数据 BI 分析和可视化问题，解放数据可视化系统的开发人力。

### 2.1.2 使用入口

<https://cloud.baidu.com/product/sugar.html>



### 2.1.3 创建数据大屏

**1）点击【立即使用】后，登录百度账号**

**2）然后首先创建组织**



**3）创建中选择产品【大屏尝鲜版】，首次使用有一个月的试用期**

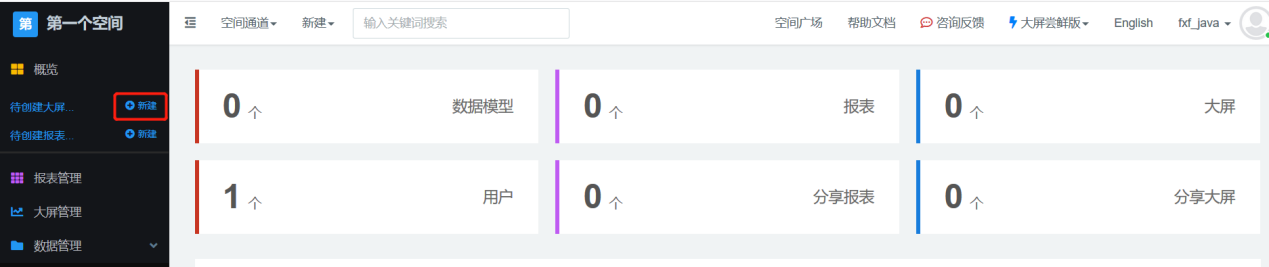


**4）新建好组织后选择【进入组织】**

 **5）然后进入默认的【第一个空间】**

****

**6）在空间中选择【待创建大屏】后的【新建】**

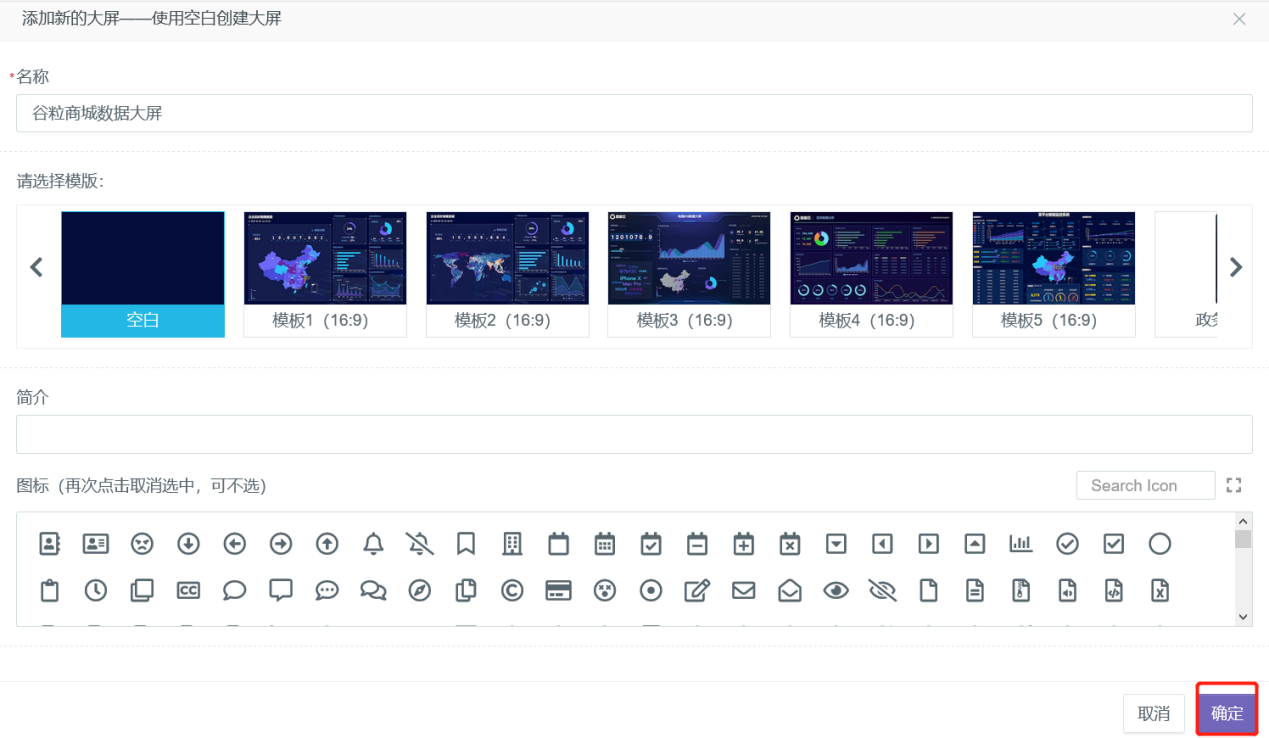


**7）选择大屏的模板**

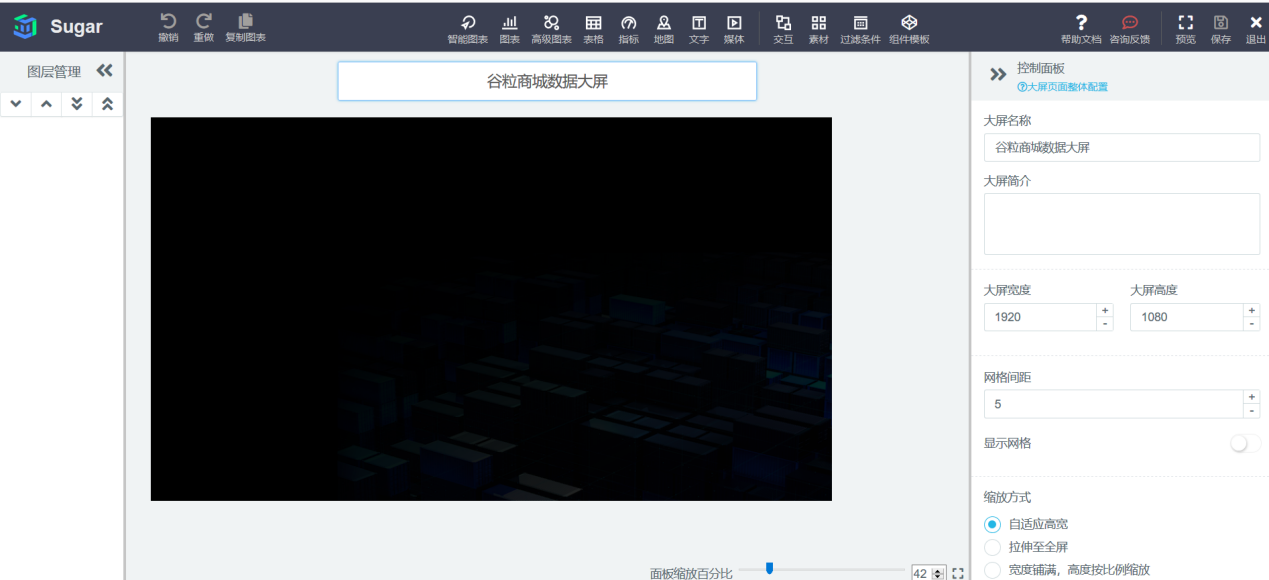


**8）可以选空模板，也可以根据现有的模板进行修改**

我们这里选择空白模板，并指定大屏的名称



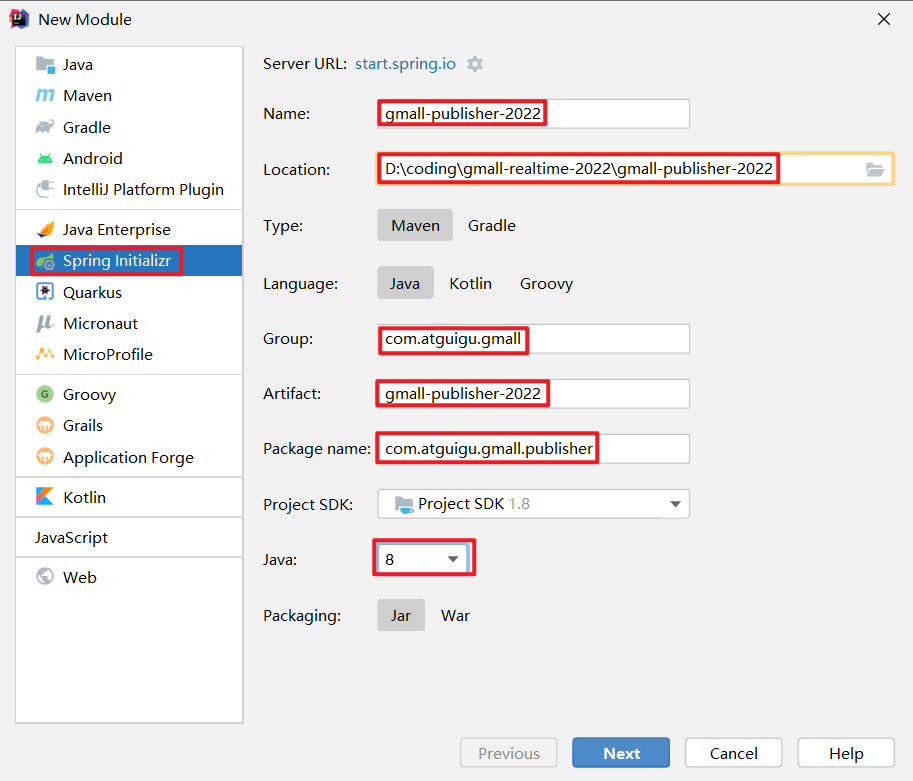
**9）进入大屏的编辑窗口**



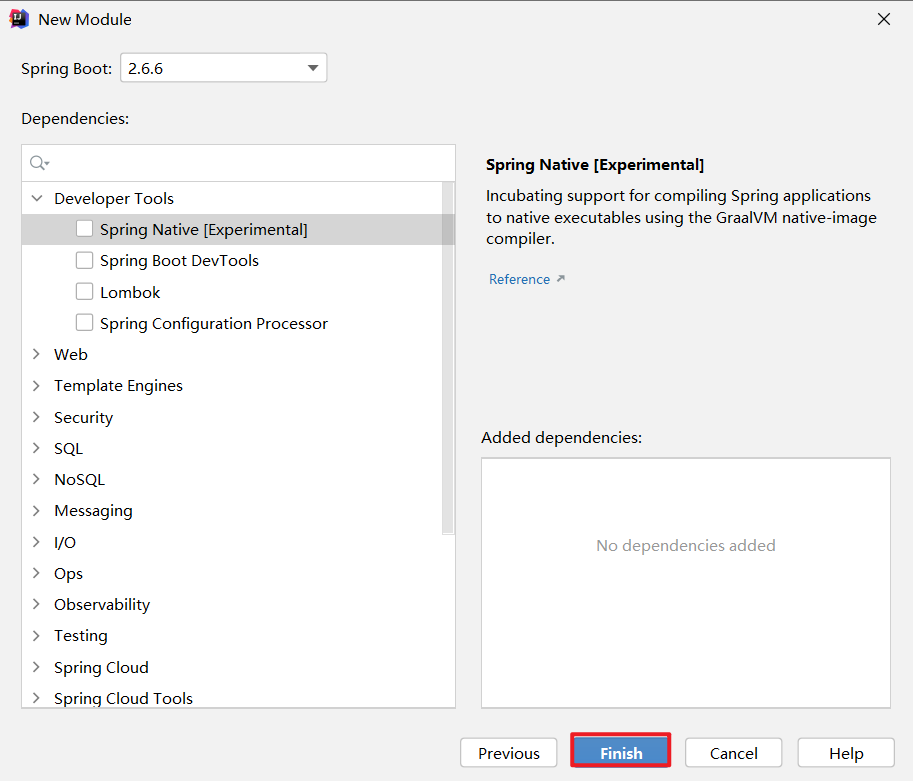
## 2.2 SpringBoot 开发环境构建

### 2.2.1 步骤

**1）在 gmall-realtime-2022 项目下新建模块 gmall-2022-publisher**



此处不勾选，在 pom 文件中添加依赖。



**2）添加依赖**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>2.1.3</version>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.11</version>

</dependency>

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

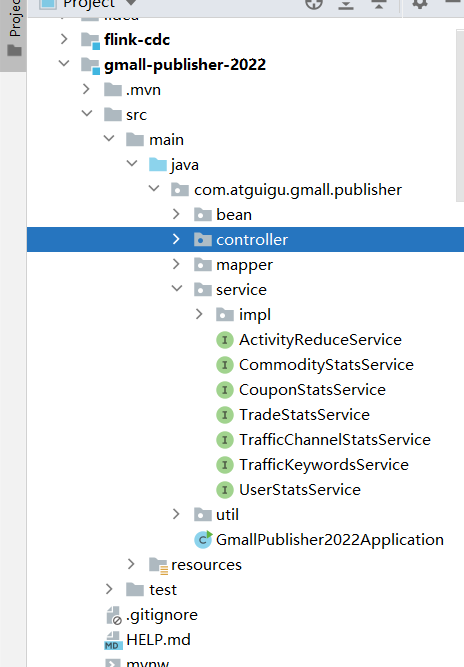
<artifactId>clickhouse-jdbc</artifactId>

<version>0.1.55</version>

</dependency>

</dependencies>

**3）目录结构如下**



**4）在 application.properties 内添加如下内容**

SpringBoot 内嵌了 Tomact，默认端口为 8080。集群 Zookeeper 版本为 3.5.7，该版本提供的 AdminServer 服务端口号也是 8080，为了避免端口冲突，此处将 SpringBoot 内嵌的 Tomcat 容器端口号修改为 8070。

接口对接的数据库为 ClickHouse，需要指定驱动及 url。

server.port=8070

#配置ClickHouse驱动以及URL

spring.datasource.driver-class-name=ru.yandex.clickhouse.ClickHouseDriver

spring.datasource.url=jdbc:clickhouse://hadoop102:8123/gmall\_rebuild

### 2.2.2 SpringBoot 项目分层

**1）表示层（也叫控制层）**

主要任务是拦截并处理请求。表示层代码通常在 controller 包下。

**2）业务层**

业务层代码通常在 service 包下。

service 下会有个名为 impl 的子包，里面放置 service 层接口的实现类。

实现类类名规范：接口名后面加Impl

**3）持久层**

和数据库交互，最常用的框架是 Mybatis，所以也叫 Mapper 层。

持久层代码通常在 mapper 包下。

## 2.3 内网穿透

### 2.3.1 内网穿透简介

内网即局域网。假设局域网中有一台电脑部署了web服务，现在希望所有人都能访问它。很显然，这台电脑只有一个局域网ip，没有公网ip。同一局域网内的设备可以通过局域网ip访问此电脑，而局域网之外的设备无法访问。

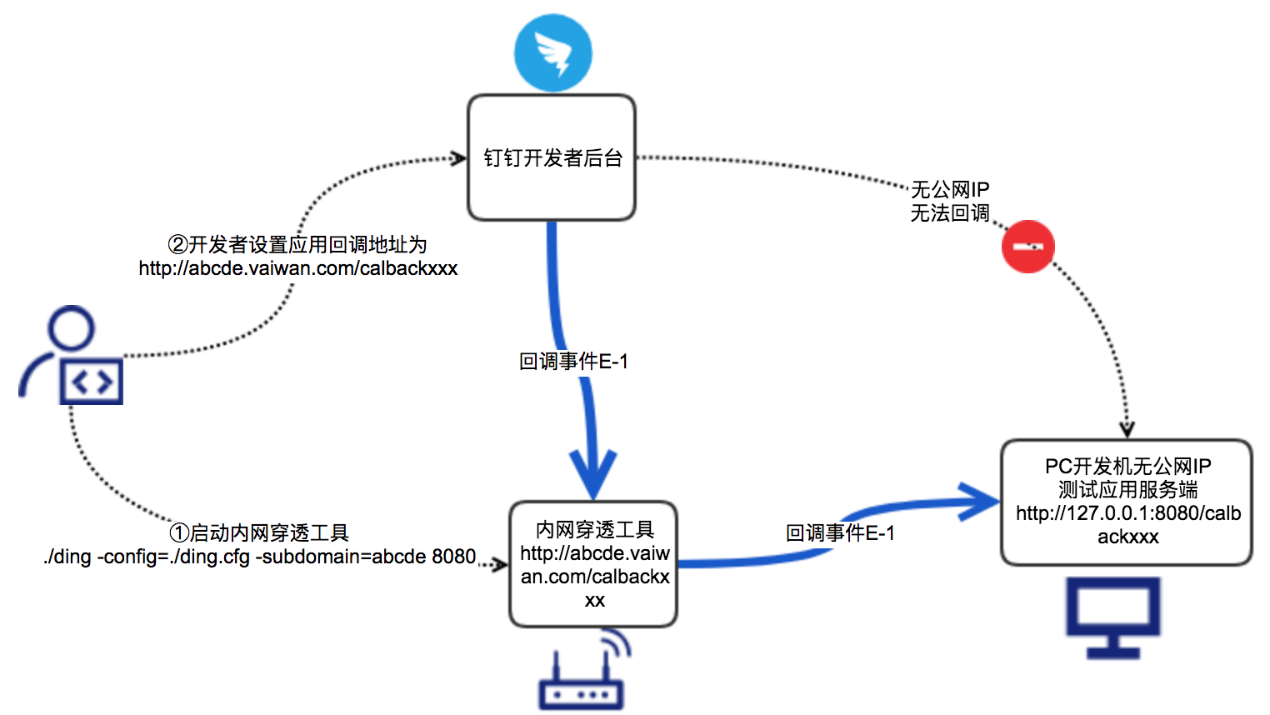
内网穿透可以将 ip + 端口唯一标识的本机服务映射为公网域名，局域网之外的设备可以通过该域名访问本机服务。

### 2.3.2 实现步骤

本项目将使用钉钉提供的内网穿透工具。

Github 地址为 <https://github.com/open-dingtalk/dingtalk-pierced-client>

1）原理

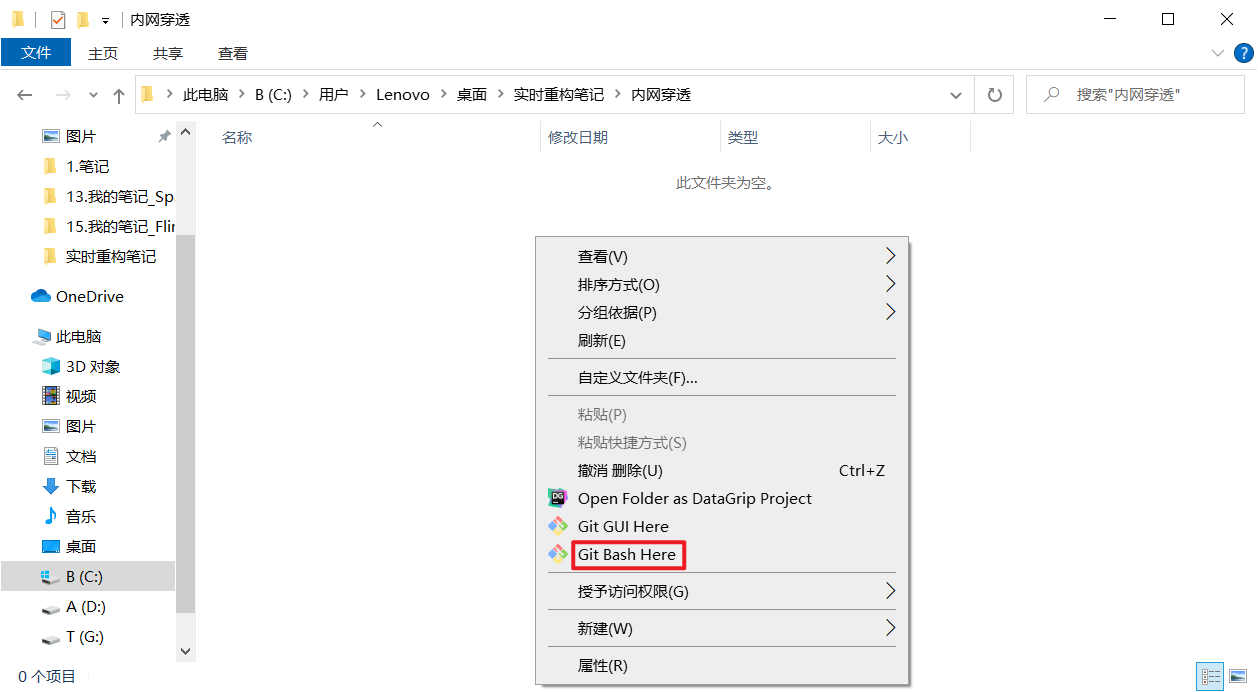


2）步骤

（1）下载工具

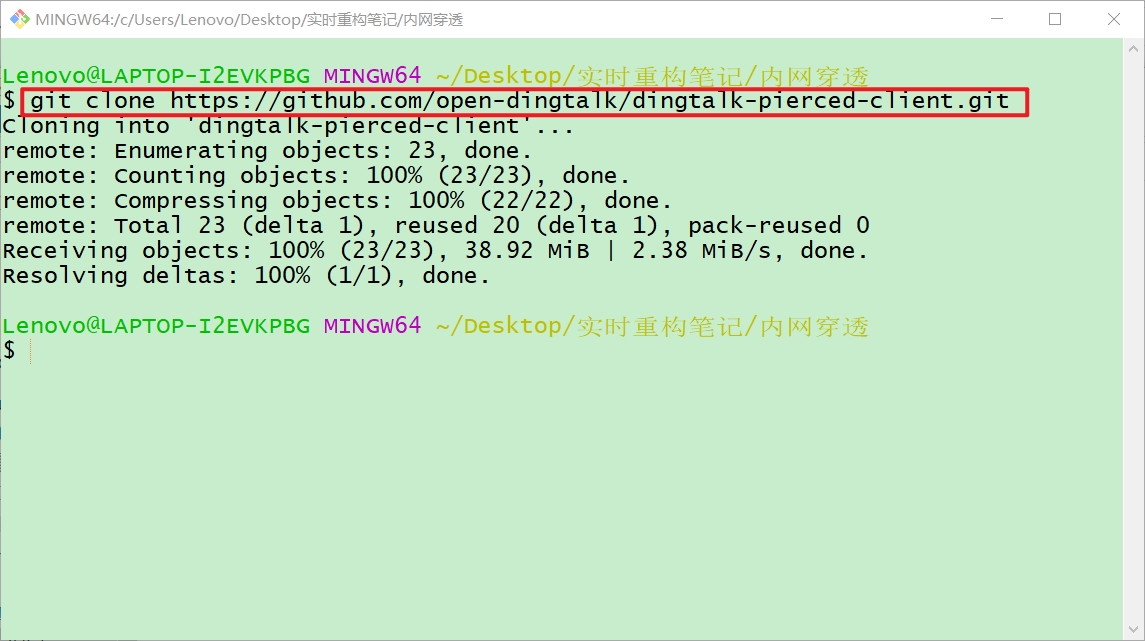
确保本机已安装 git。

① 在任意目录右键空白处，单击 Git Bash Here。

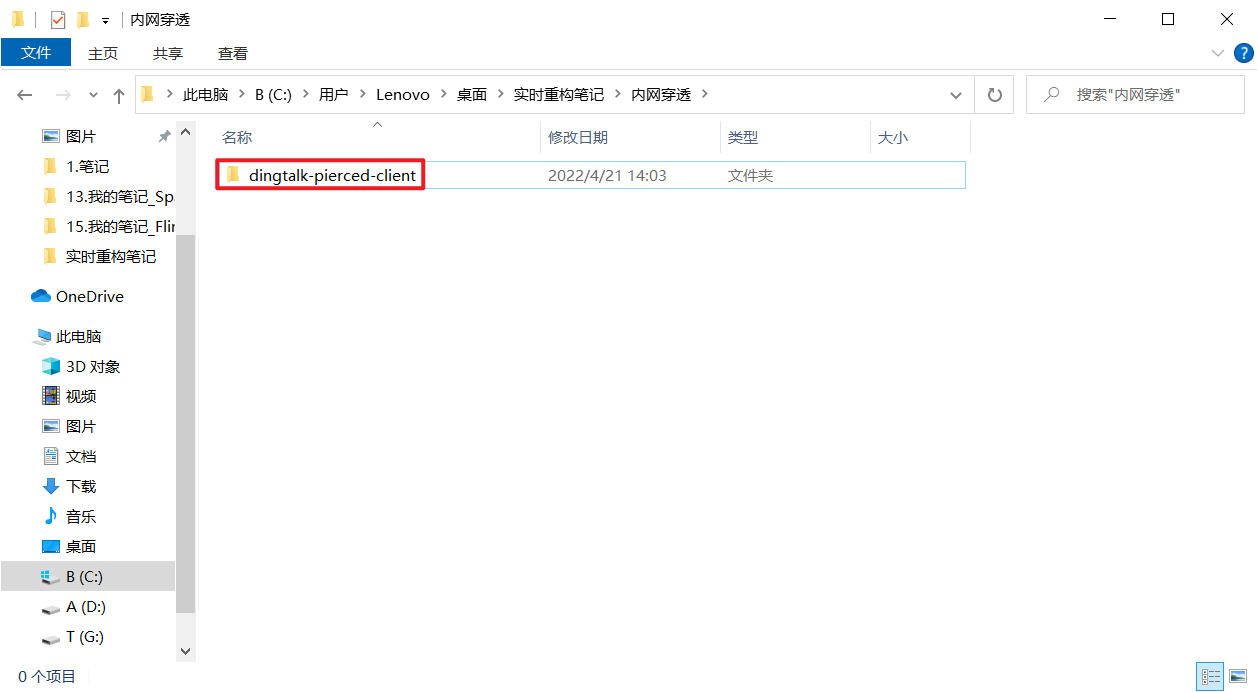


② 在弹出的窗口中执行以下命令

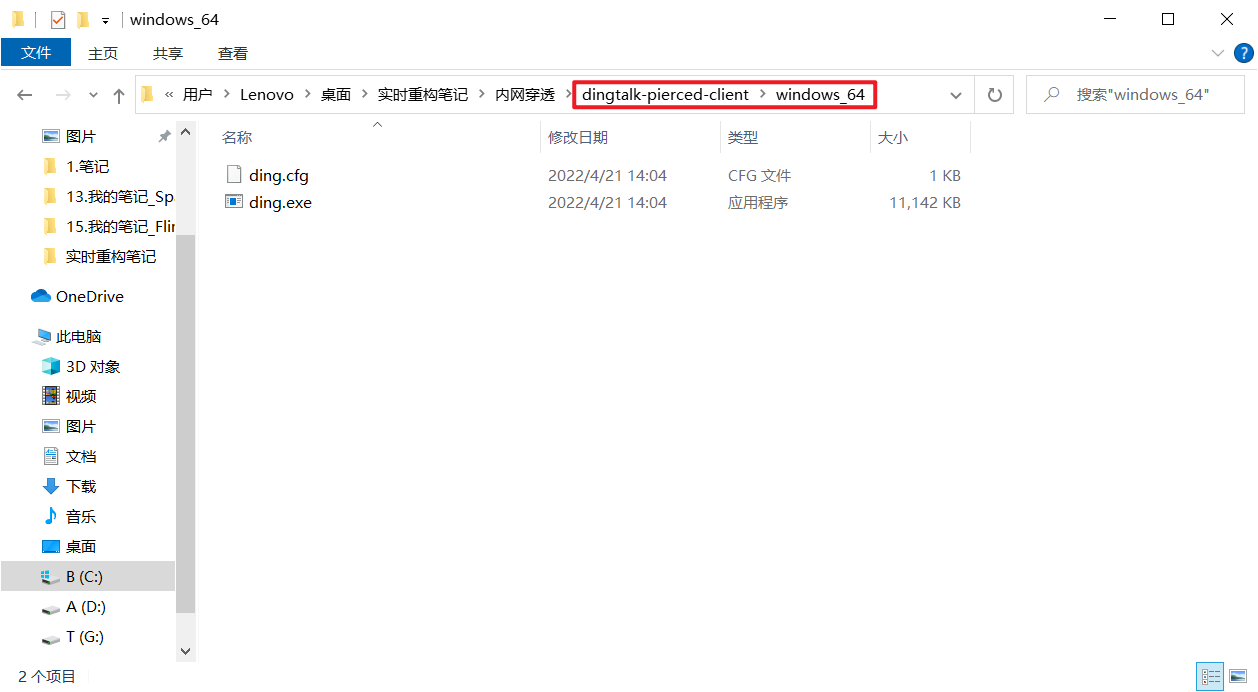
git clone https://github.com/open-dingtalk/dingtalk-pierced-client.git



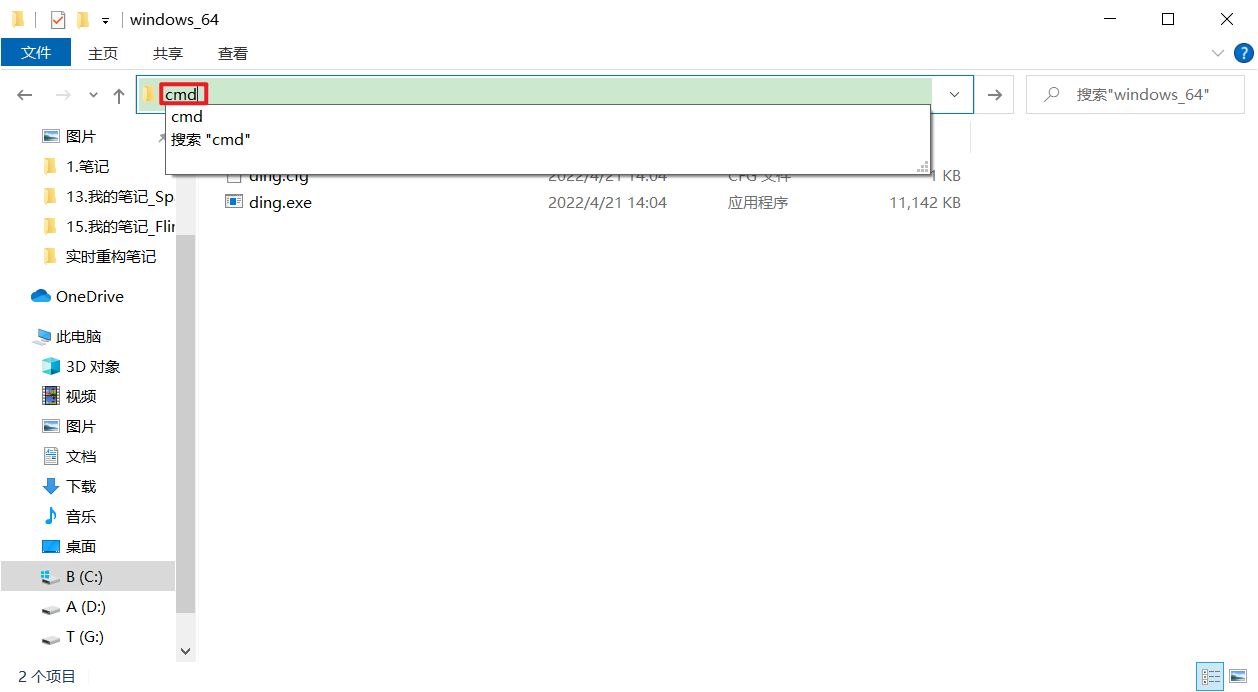
③ 查看目录，多了 dingtalk-pierced-client 文件。



④ 进入 windows\_64 目录



⑤ 在地址栏输入 cmd，回车



⑥ 在弹出的 cmd 窗口中执行以下命令

ding --config ding.cfg --subdomain dinghhh 8070

* + --config 指定内网穿透的配置文件，固定为钉钉提供的./ding.cfg，无需修改。
  + --subdomain 指定需要使用的域名前缀，该前缀将会匹配到“vaiwan.cn”前面，此处 subdomain 是 dinghhh，启动工具后会将 dinghhh.vaiwan.cn 映射到本地。
  + 8070 为需要代理的本地服务 http-server 端口。

执行完毕后，局域网之外的设备可以通过 <http://dinghhh.vaiwan.cn> 访问本地 8070 端口的 web 服务。

⑦ 启动客户端后http://dinghhh.vaiwan.cn/xxx 的请求会映射到 http://localhost:8070/xxx。

# 第3章 数仓开发之ADS层

## 3.1 流量主题

### 3.1.1 各渠道流量统计

**1）需求说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 渠道 | 独立访客数 | 统计访问人数 |
| 当日 | 渠道 | 会话总数 | 统计会话总数 |
| 当日 | 渠道 | 会话平均浏览页面数 | 统计每个会话平均浏览页面数 |
| 当日 | 渠道 | 会话平均停留时长 | 统计每个会话平均停留时长 |
| 当日 | 渠道 | 跳出率 | 只有一个页面的会话的比例 |

**2）需求分析**

柱状图可以直观展示不同渠道的度量值，本节将为上述五个指标各生成一张柱状图。

**3）数据结构**

图表所需数据结构的获取详见本节 **7）Sugar 配置**。

{

"status": 0,

"msg": "",

"data": {

"categories": [

"Appstore",

"xiaomi",

...

],

"series": [

{

"name": "独立访客数",

"data": [

"99",

"74",

...

]

}

]

}

}

（1）categories

数组中存储的元素为柱状图横轴所有取值。

（2）data

数组中存储的元素为与横轴对应的纵轴取值。

**4）Mapper 层**

（1）实体类

① TrafficUvCt

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TrafficUvCt {

// 渠道

String ch;

// 独立访客数

Integer uvCt;

}

② TrafficSvCt

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

public class TrafficSvCt {

// 渠道

String ch;

// 会话数

Integer svCt;

}

③ TrafficPvPerSession

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

public class TrafficPvPerSession {

// 渠道

String ch;

// 各会话页面浏览数

Double pvPerSession;

}

④ TrafficDurPerSession

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

public class TrafficDurPerSession {

// 渠道

String ch;

// 各会话页面访问时长

Double durPerSession;

}

⑤ TrafficUjRate

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

public class TrafficUjRate {

// 渠道

String ch;

// 跳出率

Double ujRate;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.\*;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface TrafficChannelStatsMapper {

**// 1. 获取各渠道独立访客数**

@Select("select ch,\n" +

" sum(uv\_ct) uv\_ct\n" +

"from dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt), ch\n" +

"order by uv\_ct desc;")

List<TrafficUvCt> selectUvCt(@Param("date")Integer date);

**// 2. 获取各渠道会话数**

@Select("select ch,\n" +

" sum(sv\_ct) sv\_ct\n" +

"from dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt), ch\n" +

"order by sv\_ct desc;")

List<TrafficSvCt> selectSvCt(@Param("date")Integer date);

**// 3. 获取各渠道会话平均页面浏览数**

@Select("select ch,\n" +

" sum(pv\_ct) / sum(sv\_ct) pv\_per\_session\n" +

"from dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt), ch\n" +

"order by pv\_per\_session desc;")

List<TrafficPvPerSession> selectPvPerSession(@Param("date")Integer date);

**// 4. 获取各渠道会话平均页面访问时长**

@Select("select ch,\n" +

" sum(dur\_sum) / sum(sv\_ct) dur\_per\_session\n" +

"from dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt), ch\n" +

"order by dur\_per\_session desc;")

List<TrafficDurPerSession> selectDurPerSession(@Param("date")Integer date);

**// 5. 获取各渠道跳出率**

@Select("select ch,\n" +

" sum(uj\_ct) / sum(sv\_ct) uj\_rate\n" +

"from dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt), ch\n" +

"order by uj\_rate desc;")

List<TrafficUjRate> selectUjRate(@Param("date")Integer date);

}

（3）注解

① @Select 注解

位于方法定义语句上方，修饰方法。

添加该注解后， Mybatis 会自动实现 JDBC 的环境准备，实现对应的 Mapper 层接口，并在 Tomcat 容器启动时将该实现类加载到容器中，包含 @Service 注解的功能。可以在服务层的 Impl 类中通过 @Autowired 自动装载该实现类。该实现类是单例的。

注意：要将该类加载到容器中，需要在启动类上方添加 @MapperScan 注解，指明 mapper 包路径，如下。

package com.atguigu.gmall.publisher;

import org.mybatis.spring.annotation.MapperScan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**@MapperScan(basePackages = "com.atguigu.gmall.publisher.mapper")**

public class GmallPublisher2022Application {

public static void main(String[] args) {

SpringApplication.run(GmallPublisher2022Application.class, args);

}

}

② @Param 注解

位于方法参数数据类型之前，修饰参数。

该注解可以将方法参数接收的值赋给注解参数中的变量，该变量可以在方法上方 @Select 注解的 SQL 语句中使用，调用方式为 #{变量名}。

如 @Param("paramDate")Integer date，将参数 date 接收到的数据赋值给 paramDate，可以在 SQL 中通过 #{paramDate} 获取 paramDate 参数的值。

**5）Service 层**

（1）service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.\*;

import java.util.List;

public interface TrafficChannelStatsService {

**// 1. 获取各渠道独立访客数**

List<TrafficUvCt> getUvCt(Integer date);

**// 2. 获取各渠道会话数**

List<TrafficSvCt> getSvCt(Integer date);

**// 3. 获取各渠道会话平均页面浏览数**

List<TrafficPvPerSession> getPvPerSession(Integer date);

**// 4. 获取各渠道会话平均页面访问时长**

List<TrafficDurPerSession> getDurPerSession(Integer date);

**// 5. 获取各渠道跳出率**

List<TrafficUjRate> getUjRate(Integer date);

}

（2）service 实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.\*;

import com.atguigu.gmall.publisher.mapper.TrafficChannelStatsMapper;

import com.atguigu.gmall.publisher.service.TrafficChannelStatsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class TrafficChannelStatsServiceImpl implements TrafficChannelStatsService {

**// 自动装载 Mapper 接口实现类**

@Autowired

TrafficChannelStatsMapper trafficChannelStatsMapper;

**// 1. 获取各渠道独立访客数**

@Override

public List<TrafficUvCt> getUvCt(Integer date) {

return trafficChannelStatsMapper.selectUvCt(date);

}

**// 2. 获取各渠道会话数**

@Override

public List<TrafficSvCt> getSvCt(Integer date) {

return trafficChannelStatsMapper.selectSvCt(date);

}

**// 3. 获取各渠道会话平均页面浏览数**

@Override

public List<TrafficPvPerSession> getPvPerSession(Integer date) {

return trafficChannelStatsMapper.selectPvPerSession(date);

}

**// 4. 获取各渠道会话平均页面访问时长**

@Override

public List<TrafficDurPerSession> getDurPerSession(Integer date) {

return trafficChannelStatsMapper.selectDurPerSession(date);

}

**// 5. 获取各渠道跳出率**

@Override

public List<TrafficUjRate> getUjRate(Integer date) {

return trafficChannelStatsMapper.selectUjRate(date);

}

}

（3）注解

① @Service 注解

位于实现类定义语句上方，修饰类。

把该实现类注册成一个组件，这个组件常驻内存，而且默认为单态（即单例，整个服务器进程，该类对象只有一个）。启动时会扫描所有被 @Service 注解修饰的实现类，将它们加载到内存。

② @Autowired 注解

位于成员变量（属性）定义语句上方，修饰类的属性。

① 控制器组件会扫描接口名上方添加了 @Autowired（自动装配）注解的接口，然后去内存的常驻组件中寻找适配组件（实现类），令此处的引用指向适配组件。

② 如果一个接口有多个实现类注册成为了组件，可以通过 @ Qualifier("组件名称") 注解指定不同的适配组件。此处的组件名称在 @Service 注解的参数中指定，如 @Service("组件名称")，只有一个实现类时可以不起名（不传参），直接使用 @Service 注解。

**6）controller 层**

（1）代码

package com.atguigu.gmall.publisher.controller;

import com.atguigu.gmall.publisher.bean.\*;

import com.atguigu.gmall.publisher.service.TrafficChannelStatsService;

import com.atguigu.gmall.publisher.service.TrafficKeywordsService;

import com.atguigu.gmall.publisher.service.TrafficVisitorStatsService;

import com.atguigu.gmall.publisher.util.DateUtil;

import org.apache.commons.lang3.StringUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/gmall/realtime/traffic")

public class TrafficController {

**// 自动装载渠道流量统计服务实现类**

@Autowired

private TrafficChannelStatsService trafficChannelStatsService;

**// 1. 独立访客请求拦截方法**

@RequestMapping("/uvCt")

public String getUvCt(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficUvCt> trafficUvCtList = trafficChannelStatsService.getUvCt(date);

if (trafficUvCtList == null) {

return "";

}

StringBuilder categories = new StringBuilder("[");

StringBuilder uvCtValues = new StringBuilder("[");

for (int i = 0; i < trafficUvCtList.size(); i++) {

TrafficUvCt trafficUvCt = trafficUvCtList.get(i);

String ch = trafficUvCt.getCh();

Integer uvCt = trafficUvCt.getUvCt();

categories.append("\"").append(ch).append("\"");

uvCtValues.append("\"").append(uvCt).append("\"");

if (i < trafficUvCtList.size() - 1) {

categories.append(",");

uvCtValues.append(",");

} else {

categories.append("]");

uvCtValues.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"categories\":" + categories + ",\n" +

" \"series\": [\n" +

" {\n" +

" \"name\": \"独立访客数\",\n" +

" \"data\": " + uvCtValues + "\n" +

" }\n" +

" ]\n" +

" }\n" +

"}";

}

**// 2. 会话数请求拦截方法**

@RequestMapping("/svCt")

public String getPvCt(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficSvCt> trafficSvCtList = trafficChannelStatsService.getSvCt(date);

if (trafficSvCtList == null) {

return "";

}

StringBuilder categories = new StringBuilder("[");

StringBuilder svCtValues = new StringBuilder("[");

for (int i = 0; i < trafficSvCtList.size(); i++) {

TrafficSvCt trafficSvCt = trafficSvCtList.get(i);

String ch = trafficSvCt.getCh();

Integer svCt = trafficSvCt.getSvCt();

categories.append("\"").append(ch).append("\"");

svCtValues.append("\"").append(svCt).append("\"");

if (i < trafficSvCtList.size() - 1) {

categories.append(",");

svCtValues.append(",");

} else {

categories.append("]");

svCtValues.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"categories\":" + categories + ",\n" +

" \"series\": [\n" +

" {\n" +

" \"name\": \"会话数\",\n" +

" \"data\": " + svCtValues + "\n" +

" }\n" +

" ]\n" +

" }\n" +

"}";

}

**// 3. 各会话浏览页面数请求拦截方法**

@RequestMapping("/pvPerSession")

public String getPvPerSession(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficPvPerSession> trafficPvPerSessionList = trafficChannelStatsService.getPvPerSession(date);

if (trafficPvPerSessionList == null) {

return "";

}

StringBuilder categories = new StringBuilder("[");

StringBuilder pvPerSessionValues = new StringBuilder("[");

for (int i = 0; i < trafficPvPerSessionList.size(); i++) {

TrafficPvPerSession trafficPvPerSession = trafficPvPerSessionList.get(i);

String ch = trafficPvPerSession.getCh();

Double pvPerSession = trafficPvPerSession.getPvPerSession();

categories.append("\"").append(ch).append("\"");

pvPerSessionValues.append("\"").append(pvPerSession).append("\"");

if (i < trafficPvPerSessionList.size() - 1) {

categories.append(",");

pvPerSessionValues.append(",");

} else {

categories.append("]");

pvPerSessionValues.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"categories\":" + categories + ",\n" +

" \"series\": [\n" +

" {\n" +

" \"name\": \"会话平均页面浏览数\",\n" +

" \"data\": " + pvPerSessionValues + "\n" +

" }\n" +

" ]\n" +

" }\n" +

"}";

}

**// 4. 各会话累计访问时长请求拦截方法**

@RequestMapping("/durPerSession")

public String getDurPerSession(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficDurPerSession> trafficDurPerSessionList = trafficChannelStatsService.getDurPerSession(date);

if (trafficDurPerSessionList == null) {

return "";

}

StringBuilder categories = new StringBuilder("[");

StringBuilder durPerSessionValues = new StringBuilder("[");

for (int i = 0; i < trafficDurPerSessionList.size(); i++) {

TrafficDurPerSession trafficDurPerSession = trafficDurPerSessionList.get(i);

String ch = trafficDurPerSession.getCh();

Double durPerSession = trafficDurPerSession.getDurPerSession();

categories.append("\"").append(ch).append("\"");

durPerSessionValues.append("\"").append(durPerSession).append("\"");

if (i < trafficDurPerSessionList.size() - 1) {

categories.append(",");

durPerSessionValues.append(",");

} else {

categories.append("]");

durPerSessionValues.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"categories\":" + categories + ",\n" +

" \"series\": [\n" +

" {\n" +

" \"name\": \"会话平均页面访问时长\",\n" +

" \"data\": " + durPerSessionValues + "\n" +

" }\n" +

" ]\n" +

" }\n" +

"}";

}

**// 5. 跳出率请求拦截方法**

@RequestMapping("/ujRate")

public String getUjRate(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficUjRate> trafficUjRateList = trafficChannelStatsService.getUjRate(date);

if (trafficUjRateList == null) {

return "";

}

StringBuilder categories = new StringBuilder("[");

StringBuilder ujRateValues = new StringBuilder("[");

for (int i = 0; i < trafficUjRateList.size(); i++) {

TrafficUjRate trafficUjRate = trafficUjRateList.get(i);

String ch = trafficUjRate.getCh();

Double ujRate = trafficUjRate.getUjRate();

categories.append("\"").append(ch).append("\"");

ujRateValues.append("\"").append(ujRate).append("\"");

if (i < trafficUjRateList.size() - 1) {

categories.append(",");

ujRateValues.append(",");

} else {

categories.append("]");

ujRateValues.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"categories\":" + categories + ",\n" +

" \"series\": [\n" +

" {\n" +

" \"name\": \"跳出率\",\n" +

" \"data\": " + ujRateValues + "\n" +

" }\n" +

" ]\n" +

" }\n" +

"}";

}

}

（2）注解

① @RestController

位于控制类定义语句上方，修饰控制类。

标注方法为控制类，可以将外部请求引入该类的方法中。当方法返回值类型为 String 时，不会发生页面跳转，相当于 @Controller + @ResponseBody。

② @RequestMapping

可以在控制类定义语句上方添加，也可以在方法定义语句上方添加。

示例：@RequestMapping(value = "/traffic", method = RequestMethod.GET)。

该注解用于建立请求URL和请求处理方法之间的对应关系。value 用于指定请求 URL。请求处理方法最终拦截的 URL为类注解指定的路径与方法注解指定的路径拼接所得的 URL。如当前 SpringBoot 服务占用了本机的 8080 端口，类上注解为 @RequestMapping("/gmall ")，方法上方注解为 @RequestMapping("/uvCt")，则最终拦截的 url 为 localhost:8080/gmall/uvCt。控制类上方可以没有该注解。

如果拦截的是 GET 请求，method 参数赋值语句可以省略。此外，SpringBoot 的注解只有一个参数，且参数名称为 value时，可以省略参数名称和等号。

所以，@RequestMapping(value = "/traffic", method = RequestMethod.GET) 可简写为 @RequestMapping("/traffic")。

该注解相当于 @GetMapping("/traffic")

③ @RequestParma("name")

位于方法参数数据类型之前。接收请求路径中键值对(名值对)的参数(通常用于查询条件、辅助参数) 并传递给所修饰的参数。

如 URL 为 localhost:8080/test?paramDt=20220221，注解为 @RequestParam("paramDt") Integer dt，则 20220221 会被解析为 Integer 类型数据赋值给 dt 参数。

**7）Sugar 配置**

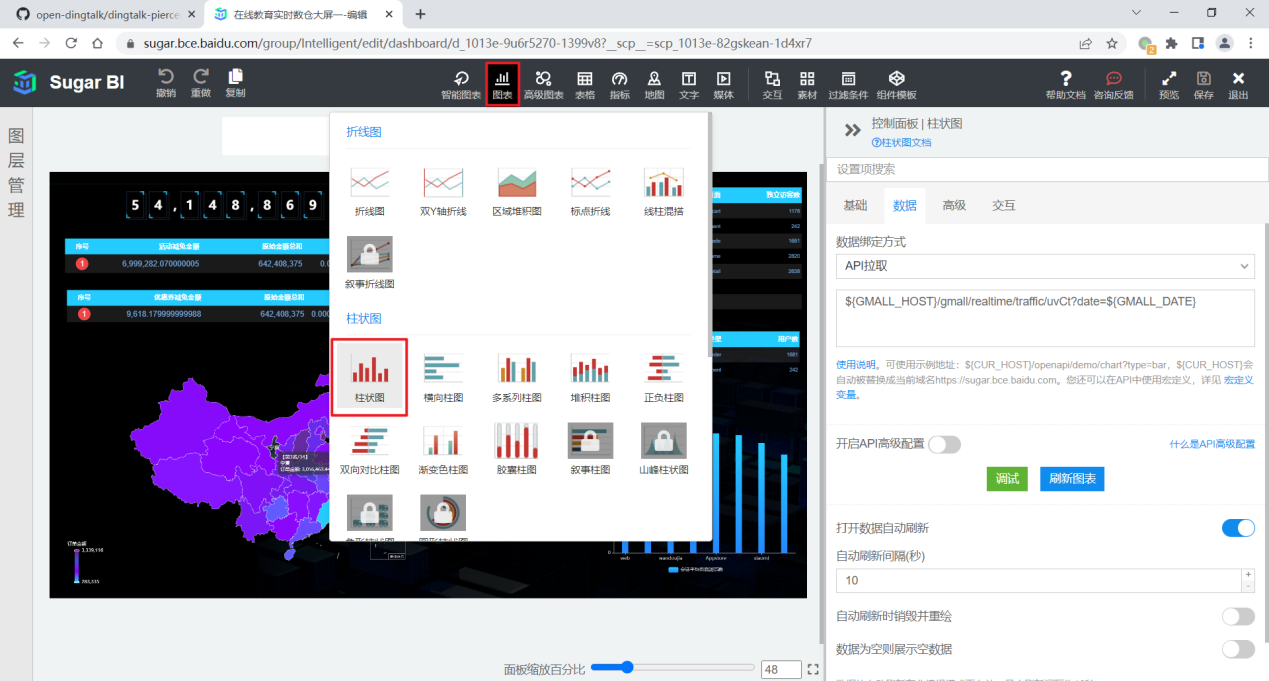
（1）页面宏定义变量

单击大屏空白处，在右侧“控制面板”中点击“页面宏定义变量”，定义 GMALL\_HOST 和 GMALL\_DATE。

* + GMALL\_HOST：本地 8070 端口服务映射的公网域名。
  + GMALL\_DATE：当日日期。本项目的“当日”为模拟数据的业务日期。



（2）在“图表”中选择“柱状图”



（3）在弹出的控制面板中选择 API 拉取

* + API 拉取：通过给定的数据接口获取数据。我们选择这种方式。
  + 静态 JSON：通过给定的静态 JSON 字符串获取数据。

API 拉取返回的数据格式与静态 JSON 相同，因此可以通过静态 JSON 的示例数据查看数据格式。

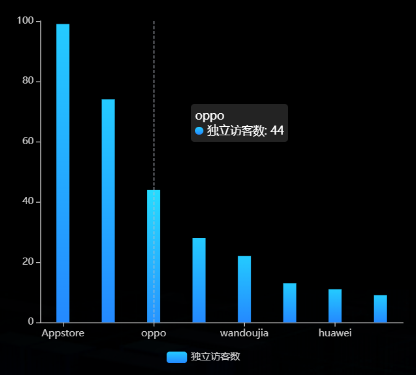




（4）输入数据接口的 URL

${GMALL\_HOST}/gmall/realtime/traffic/uvCt?date=${GMALL\_DATE}

（5）刷新图表，查看效果



另外四张柱状图的配置同理，不再赘述。

所有图表的 URL 如下。

**# 独立访客**

${GMALL\_HOST}/gmall/realtime/traffic/uvCt?date=${GMALL\_DATE}

**# 会话数**

${GMALL\_HOST}/gmall/realtime/traffic/svCt?date=${GMALL\_DATE}

**# 会话平均页面浏览数**

${GMALL\_HOST}/gmall/realtime/traffic/pvPerSession?date=${GMALL\_DATE}

**# 会话平均访问时长**

${GMALL\_HOST}/gmall/realtime/traffic/durPerSession?date=${GMALL\_DATE}

**# 跳出率**

${GMALL\_HOST}/gmall/realtime/traffic/ujRate?date=${GMALL\_DATE}

### 3.1.2 流量分时统计

**1）需求说明**

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 1 小时 | 独立访客数 | 统计当日各小时独立访客数 |
| 1 小时 | 页面浏览数 | 统计当日各小时页面浏览数 |
| 1 小时 | 新访客数 | 统计当日各小时新访客数 |

**2）需求分析**

本节统计分时指标，选用折线图进行展示。三个指标将体现在同一张折线图中。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": {

"categories": [

"00",

"01",

...

],

"series": [

{

"name": "独立访客数",

"data": [

0,

300,

...

]

},

{

"name": "页面浏览数",

"data": [

0,

988,

...

]

},

{

"name": "新访客数",

"data": [

0,

155,

...

]

}

]

}

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TrafficVisitorStatsPerHour {

// 小时

Integer hr;

// 独立访客数

Long uvCt;

// 页面浏览数

Long pvCt;

// 新访客数

Long newUvCt;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.TrafficVisitorTypeStats;

import com.atguigu.gmall.publisher.bean.TrafficVisitorStatsPerHour;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface TrafficVisitorStatsMapper {

**// 分时流量数据查询**

@Select("select\n" +

"toHour(stt) hr,\n" +

"sum(uv\_ct) uv\_ct,\n" +

"sum(pv\_ct) pv\_ct,\n" +

"sum(if(is\_new = '1', dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window.uv\_ct, 0)) new\_uv\_ct\n" +

"from dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by hr")

List<TrafficVisitorStatsPerHour> selectVisitorStatsPerHr(Integer date);

}

**5）Service 层**

（1）Service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.TrafficVisitorStatsPerHour;

import com.atguigu.gmall.publisher.bean.TrafficVisitorTypeStats;

import java.util.List;

public interface TrafficVisitorStatsService {

**// 获取分时流量数据**

List<TrafficVisitorStatsPerHour> getVisitorPerHrStats(Integer date);

}

（2）实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.TrafficVisitorStatsPerHour;

import com.atguigu.gmall.publisher.bean.TrafficVisitorTypeStats;

import com.atguigu.gmall.publisher.mapper.TrafficVisitorStatsMapper;

import com.atguigu.gmall.publisher.service.TrafficVisitorStatsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class TrafficVisitorStatsServiceImpl implements TrafficVisitorStatsService {

@Autowired

private TrafficVisitorStatsMapper trafficVisitorStatsMapper;

**// 获取分时流量统计数据**

@Override

public List<TrafficVisitorStatsPerHour> getVisitorPerHrStats(Integer date) {

return trafficVisitorStatsMapper.selectVisitorStatsPerHr(date);

}

}

**6）Controller 层**

在 TrafficController 类中补充成员变量和请求拦截方法。

**// 自动装载访客状态统计服务实现类**

@Autowired

private TrafficVisitorStatsService trafficVisitorStatsService;

**// 访客状态分时统计请求拦截方法**

@RequestMapping("/visitorPerHr")

public String getVisitorPerHr(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficVisitorStatsPerHour> visitorPerHrStatsList = trafficVisitorStatsService.getVisitorPerHrStats(date);

if (visitorPerHrStatsList == null || visitorPerHrStatsList.size() == 0) {

return "";

}

TrafficVisitorStatsPerHour[] perHrArr = new TrafficVisitorStatsPerHour[24];

for (TrafficVisitorStatsPerHour trafficVisitorStatsPerHour : visitorPerHrStatsList) {

Integer hr = trafficVisitorStatsPerHour.getHr();

perHrArr[hr] = trafficVisitorStatsPerHour;

}

String[] hrs = new String[24];

Long[] uvArr = new Long[24];

Long[] pvArr = new Long[24];

Long[] newUvArr = new Long[24];

for (int hr = 0; hr < 24; hr++) {

hrs[hr] = String.format("%02d", hr);

TrafficVisitorStatsPerHour trafficVisitorStatsPerHour = perHrArr[hr];

if (trafficVisitorStatsPerHour != null) {

uvArr[hr] = trafficVisitorStatsPerHour.getUvCt();

pvArr[hr] = trafficVisitorStatsPerHour.getPvCt();

newUvArr[hr] = trafficVisitorStatsPerHour.getNewUvCt();

} else{

uvArr[hr] = 0L;

pvArr[hr] = 0L;

newUvArr[hr] = 0L;

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"categories\": [\n\"" +

StringUtils.join(hrs, "\",\"") + "\"\n" +

" ],\n" +

" \"series\": [\n" +

" {\n" +

" \"name\": \"独立访客数\",\n" +

" \"data\": [\n" +

StringUtils.join(uvArr, ",") + "\n" +

" ]\n" +

" },\n" +

" {\n" +

" \"name\": \"页面浏览数\",\n" +

" \"data\": [\n" +

StringUtils.join(pvArr, ",") + "\n" +

" ]\n" +

" },\n" +

" {\n" +

" \"name\": \"新访客数\",\n" +

" \"data\": [\n" +

StringUtils.join(newUvArr, ",") + "\n" +

" ]\n" +

" }\n" +

" ]\n" +

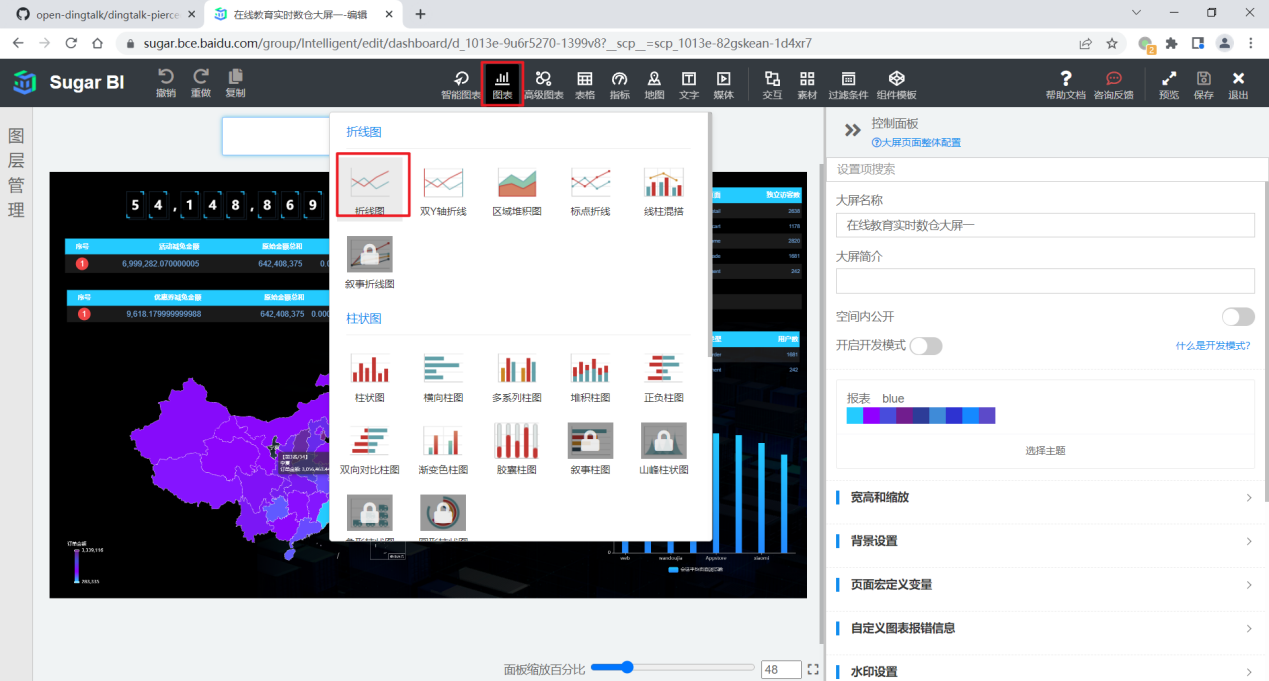
" }\n" +

"}";

}

**7）Sugar 配置**

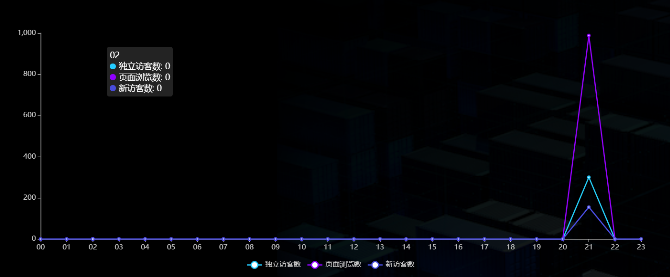
① 选择折线图。



② 数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/traffic/visitorPerHr?date=${GMALL\_DATE}

③ 效果如下。



### 3.1.3 新老访客流量统计

**1）需求介绍**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 访客类型 | 访客数 | 分别统计新老访客数 |
| 当日 | 访客类型 | 页面浏览数 | 分别统计新老访客页面浏览数 |
| 当日 | 访客类型 | 跳出率 | 分别统计新老访客跳出率 |
| 当日 | 访客类型 | 平均在线时长 | 分别统计新老访客平均在线时长 |
| 当日 | 访客类型 | 平均访问页面数 | 分别统计新老访客平均访问页面数 |

**2）需求分析**

本节将通过表格对数据进行展示。所有指标将在同一张表格中体现。

**3）数据结构**

{

"status": 0,

"data": {

"total": 5,

"columns": [

{

"name": "类别",

"id": "type"

},

{

"name": "新访客",

"id": "new"

},

{

"name": "老访客",

"id": "old"

}

],

"rows": [

{

"type": "访客数(人)",

"new": 155,

"old": 145

},

{

"type": "总访问页面数(次)",

"new": 507,

"old": 481

},

{

"type": "跳出率(%)",

"new": 0,

"old": 0

},

{

"type": "平均在线时长(秒)",

"new": 33.659716129032255,

"old": 35.37985517241379

},

{

"type": "平均访问页面数(人次)",

"new": 3.270967741935484,

"old": 3.317241379310345

}

]

}

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import org.springframework.web.bind.annotation.RequestMapping;

@Data

@AllArgsConstructor

public class TrafficVisitorTypeStats {

// 新老访客状态标记

String isNew;

// 独立访客数

Long uvCt;

// 页面浏览数

Long pvCt;

// 会话数

Long svCt;

// 跳出会话数

Long ujCt;

// 累计访问时长

Long durSum;

// 跳出率

public Double getUjRate(){

if(svCt == 0) {

return 0.0;

}

return (double)ujCt/(double)svCt;

}

// 会话平均在线时长（秒）

public Double getAvgDurSum() {

if(svCt == 0) {

return 0.0;

}

return (double)durSum/(double) svCt / 1000;

}

// 会话平均访问页面数

public Double getAvgPvCt(){

if(svCt == 0) {

return 0.0;

}

return (double)pvCt / (double) svCt;

}

}

（2）Mapper 接口

在 TrafficVisitorStatsMapper 接口中补充方法。

@Select("select\n" +

"is\_new,\n" +

"sum(uv\_ct) uv\_ct,\n" +

"sum(pv\_ct) pv\_ct,\n" +

"sum(sv\_ct) sv\_ct,\n" +

"sum(uj\_ct) uj\_ct,\n" +

"sum(dur\_sum) dur\_sum\n" +

"from dws\_traffic\_vc\_ch\_ar\_is\_new\_page\_view\_window\n" +

"where toYYYYMMDD(stt) =#{date}\n" +

"group by is\_new")

List<TrafficVisitorTypeStats> selectVisitorTypeStats(@Param("date")Integer date);

**5）Service 层**

（1）Service 接口

在 TrafficVisitorStatsService 接口中补充方法。

List<TrafficVisitorTypeStats> getVisitorTypeStats(Integer date);

（2）Service 实现类

在 TrafficVisitorStatsServiceImpl 实现类中补充 getVisitorTypeStats 方法的实现，

@Override

public List<TrafficVisitorTypeStats> getVisitorTypeStats(Integer date) {

return trafficVisitorStatsMapper.selectVisitorTypeStats(date);

}

**6）Controller 层**

在 TrafficController 类中补充请求拦截方法。

@RequestMapping("/visitorPerType")

public String getVisitorPerType(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficVisitorTypeStats> visitorTypeStatsList = trafficVisitorStatsService.getVisitorTypeStats(date);

if (visitorTypeStatsList == null || visitorTypeStatsList.size() == 0) {

return "";

}

// 方法一：通过循环的方式拼接字符串，较为繁琐，不推荐

// StringBuilder columns = new StringBuilder("[\n" +

// " {\n" +

// " \"name\": \"指标\",\n" +

// " \"id\": \"indicators\"\n" +

// " },");

//

// StringBuilder uvRow = new StringBuilder("{\n" +

// " \"indicators\": \"独立访客数\",\n");

//

// StringBuilder pvRow = new StringBuilder("{\n" +

// " \"indicators\": \"页面浏览数\",\n");

//

// StringBuilder ujRow = new StringBuilder("{\n" +

// " \"indicators\": \"跳出率\",\n");

//

// StringBuilder avgDurRow = new StringBuilder("{\n" +

// " \"indicators\": \"会话平均访问时长\",\n");

//

// StringBuilder avgPvRow = new StringBuilder("{\n" +

// " \"indicators\": \"会话平均页面浏览数\",\n");

//

// for (int i = 0; i < visitorTypeStatsList.size(); i++) {

// TrafficVisitorTypeStats trafficVisitorTypeStats = visitorTypeStatsList.get(i);

// String isNew = trafficVisitorTypeStats.getIsNew();

// Integer uvCt = trafficVisitorTypeStats.getUvCt();

// Integer pvCt = trafficVisitorTypeStats.getPvCt();

// Double ujRate = trafficVisitorTypeStats.getUjRate();

// Double avgDurSum = trafficVisitorTypeStats.getAvgDurSum();

// Double avgPvCt = trafficVisitorTypeStats.getAvgPvCt();

// if (isNew.equals("1")) {

// columns.append("{\n" +

// " \"name\": \"新访客\",\n" +

// " \"id\": \"newVisitor\"\n" +

// " }");

// uvRow.append("\"newVisitor\": " + uvCt);

// pvRow.append("\"newVisitor\": " + pvCt);

// ujRow.append("\"newVisitor\": " + ujRate);

// avgDurRow.append("\"newVisitor\": " + avgDurSum);

// avgPvRow.append("\"newVisitor\": " + avgPvCt);

// } else {

// columns.append("{\n" +

// " \"name\": \"老访客\",\n" +

// " \"id\": \"oldVisitor\"\n" +

// " }");

//

// uvRow.append("\"oldVisitor\": " + uvCt + "\n");

// pvRow.append("\"oldVisitor\": " + pvCt + "\n");

// ujRow.append("\"oldVisitor\": " + ujRate + "\n");

// avgDurRow.append("\"oldVisitor\": " + avgDurSum + "\n");

// avgPvRow.append("\"oldVisitor\": " + avgPvCt + "\n");

// }

// if (i == 0) {

// columns.append(",\n");

// uvRow.append(",\n");

// pvRow.append(",\n");

// ujRow.append(",\n");

// avgDurRow.append(",\n");

// avgPvRow.append(",\n");

// } else {

// columns.append("\n]");

// uvRow.append("\n}");

// pvRow.append("\n}");

// ujRow.append("\n}");

// avgDurRow.append("\n}");

// avgPvRow.append("\n}");

// }

// }

// return "{\n" +

// " \"status\": 0,\n" +

// " \"msg\": \"\",\n" +

// " \"data\": {\n" +

// " \"columns\": "+ columns +",\n" +

// " \"rows\": [\n" +

// " "+ uvRow +",\n" +

// " "+ pvRow +",\n" +

// " "+ ujRow +",\n" +

// " "+ avgDurRow +",\n" +

// " "+ avgPvRow +"\n" +

// " ]\n" +

// " }\n" +

// "}";

// 方法二，直接拼接字符串，简单明了

TrafficVisitorTypeStats newVisitorStats = null;

TrafficVisitorTypeStats oldVisitorStats = null;

for (TrafficVisitorTypeStats visitorStats : visitorTypeStatsList) {

// System.out.println(visitorStats);

if ("1".equals(visitorStats.getIsNew())) {

// 新访客

newVisitorStats = visitorStats;

} else {

// 老访客

oldVisitorStats = visitorStats;

}

}

//拼接json字符串

String json = "{\"status\":0,\"data\":{\"total\":5," +

"\"columns\":[" +

"{\"name\":\"类别\",\"id\":\"type\"}," +

"{\"name\":\"新访客\",\"id\":\"new\"}," +

"{\"name\":\"老访客\",\"id\":\"old\"}]," +

"\"rows\":[" +

"{\"type\":\"访客数(人)\",\"new\":" + newVisitorStats.getUvCt() + ",\"old\":" + oldVisitorStats.getUvCt() + "}," +

"{\"type\":\"总访问页面数(次)\",\"new\":" + newVisitorStats.getPvCt() + ",\"old\":" + oldVisitorStats.getPvCt() + "}," +

"{\"type\":\"跳出率(%)\",\"new\":" + newVisitorStats.getUjRate() + ",\"old\":" + oldVisitorStats.getUjRate() + "}," +

"{\"type\":\"平均在线时长(秒)\",\"new\":" + newVisitorStats.getAvgDurSum() + ",\"old\":" + oldVisitorStats.getAvgDurSum() + "}," +

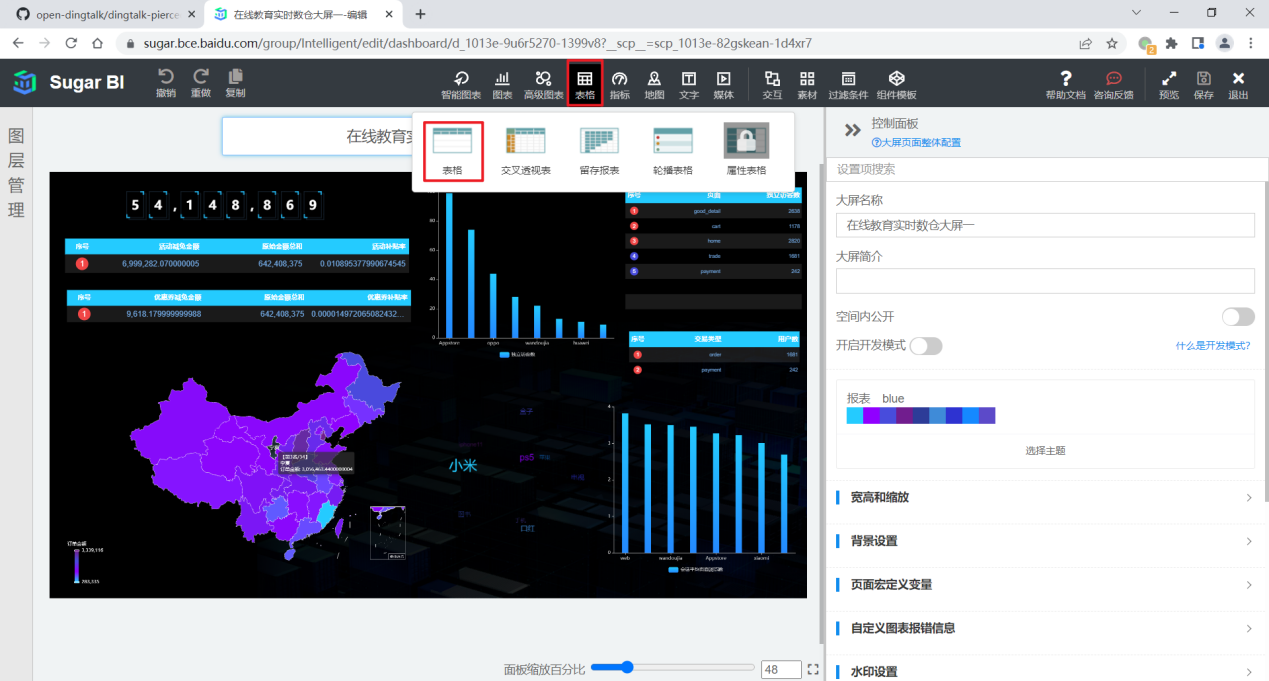
"{\"type\":\"平均访问页面数(人次)\",\"new\":" + newVisitorStats.getAvgPvCt() + ",\"old\":" + oldVisitorStats.getAvgPvCt() + "}]}}";

return json;

}

**7）Sugar 配置**

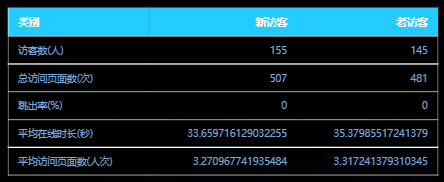
① 选择表格。



② 数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/traffic/visitorPerType?date=${GMALL\_DATE}

③ 效果如下。



### 3.1.4 关键词统计

**1）需求介绍**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 关键词 | 关键词评分 | 根据不同来源和频次计算得分 |

**2）需求分析**

关键词的展示使用 3D 词云实现。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": [

{

"name": "小米",

"value": 2370

},

{

"name": "ps5",

"value": 1290

},

...

]

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TrafficKeywords {

// 关键词

String keyword;

// 关键词评分

Integer keywordScore;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.TrafficKeywords;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface TrafficKeywordsMapper {

@Select("select keyword,\n" +

" sum(keyword\_count \* multiIf(\n" +

" source = 'SEARCH', 10,\n" +

" source = 'ORDER', 5,\n" +

" source = 'CART', 2,\n" +

" source = 'CLICK', 1, 0\n" +

" )) keyword\_score\n" +

"from dws\_traffic\_source\_keyword\_page\_view\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt), keyword\n" +

"order by keyword\_score desc;")

List<TrafficKeywords> selectKeywords(@Param(value = "date") Integer date);

}

**5）Service 层**

（1）Service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.TrafficKeywords;

import java.util.List;

public interface TrafficKeywordsService {

List<TrafficKeywords> getKeywords(Integer date);

}

（2）Service 实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.TrafficKeywords;

import com.atguigu.gmall.publisher.mapper.TrafficKeywordsMapper;

import com.atguigu.gmall.publisher.service.TrafficKeywordsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class TrafficKeywordsServiceImpl implements TrafficKeywordsService {

@Autowired

TrafficKeywordsMapper trafficKeywordsMapper;

@Override

public List<TrafficKeywords> getKeywords(Integer date) {

return trafficKeywordsMapper.selectKeywords(date);

}

}

**6）Controller 层**

在 TrafficController 中补充关键词服务接口类型成员变量和请求拦截方法。

@Autowired

private TrafficKeywordsService trafficKeywordsService;

@RequestMapping("/keywords")

public String getKeywords(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrafficKeywords> keywordsList = trafficKeywordsService.getKeywords(date);

if (keywordsList == null) {

return "";

}

StringBuilder data = new StringBuilder("[");

for (int i = 0; i < keywordsList.size(); i++) {

TrafficKeywords trafficKeywords = keywordsList.get(i);

String keyword = trafficKeywords.getKeyword();

Integer keywordScore = trafficKeywords.getKeywordScore();

data.append("" +

"{\n" +

" \"name\": \"" + keyword + "\",\n" +

" \"value\": " + keywordScore + "\n" +

" }");

if (i < keywordsList.size() - 1) {

data.append(",");

} else {

data.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

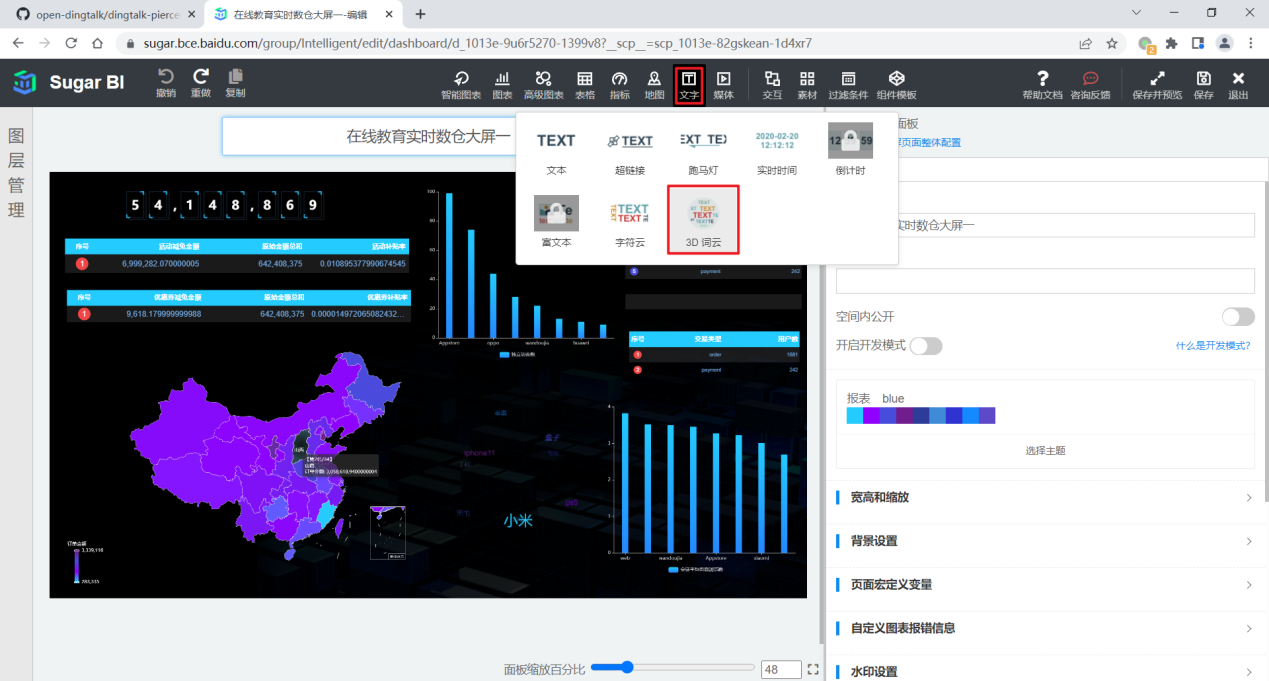
" \"data\": " + data + "\n" +

"}";

}

**7）Sugar 配置**

① 选择 3D 词云



② 数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/traffic/keywords?date=${GMALL\_DATE}

③ 效果如下



## 3.2 用户主题

### 3.2.1 用户变动统计

**1）需求介绍**

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 当日 | 回流用户数 | 之前的活跃用户，一段时间未活跃（流失），今日又活跃了，就称为回流用户。此处要求统计回流用户总数。 |

本节需求与 3.2.2 节需求合并展示。

### 3.2.2 用户新增活跃统计

需求说明如下

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **指标说明** |
| 当日 | 新增用户数 | 略 |
| 当日 | 活跃用户数 | 略 |

**2）需求分析**

本节和 3.2.1 节指标展示将使用表格展示。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": {

"columns": [

{

"name": "变动类型",

"id": "type"

},

{

"name": "用户数",

"id": "user\_ct"

}

],

"rows": [

{

"type": "activeUserCt",

"user\_ct": "915"

},

{

"type": "newUserCt",

"user\_ct": "20"

},

{

"type": "backCt",

"user\_ct": "0"

}

]

}

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class UserChangeCtPerType {

// 变动类型

String type;

// 用户数

Integer userCt;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.UserChangeCtPerType;

import com.atguigu.gmall.publisher.bean.UserPageCt;

import com.atguigu.gmall.publisher.bean.UserTradeCt;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface UserStatsMapper {

@Select("select 'backCt' type,\n" +

" sum(back\_ct) back\_ct\n" +

"from dws\_user\_user\_login\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"union all\n" +

"select 'activeUserCt' type,\n" +

" sum(uu\_ct) uu\_ct\n" +

"from dws\_user\_user\_login\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"union all\n" +

"select 'newUserCt' type,\n" +

" sum(register\_ct) register\_ct\n" +

"from dws\_user\_user\_register\_window\n" +

"where toYYYYMMDD(stt) = #{date};")

List<UserChangeCtPerType> selectUserChangeCtPerType(@Param("date")Integer date);

}

**5）Service 层**

（1）Service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.UserChangeCtPerType;

import com.atguigu.gmall.publisher.bean.UserPageCt;

import com.atguigu.gmall.publisher.bean.UserTradeCt;

import java.util.List;

public interface UserStatsService {

List<UserChangeCtPerType> getUserChangeCt(Integer date);

}

（2）Service 实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.UserChangeCtPerType;

import com.atguigu.gmall.publisher.bean.UserPageCt;

import com.atguigu.gmall.publisher.bean.UserTradeCt;

import com.atguigu.gmall.publisher.mapper.UserStatsMapper;

import com.atguigu.gmall.publisher.service.UserStatsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.autoconfigure.AutoConfigureOrder;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class UserStatsServiceImpl implements UserStatsService {

@Autowired

UserStatsMapper userStatsMapper;

@Override

public List<UserChangeCtPerType> getUserChangeCt(Integer date) {

return userStatsMapper.selectUserChangeCtPerType(date);

}

}

**6）Controller 层**

package com.atguigu.gmall.publisher.controller;

import com.atguigu.gmall.publisher.bean.UserChangeCtPerType;

import com.atguigu.gmall.publisher.bean.UserPageCt;

import com.atguigu.gmall.publisher.bean.UserTradeCt;

import com.atguigu.gmall.publisher.service.UserStatsService;

import com.atguigu.gmall.publisher.util.DateUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/gmall/realtime/user")

public class UserStatsController {

@Autowired

private UserStatsService userStatsService;

@RequestMapping("/userChangeCt")

public String getUserChange(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<UserChangeCtPerType> userChangeCtList = userStatsService.getUserChangeCt(date);

if (userChangeCtList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < userChangeCtList.size(); i++) {

UserChangeCtPerType userChangeCt = userChangeCtList.get(i);

String type = userChangeCt.getType();

Integer userCt = userChangeCt.getUserCt();

rows.append("{\n" +

"\t\"type\": \"" + type + "\",\n" +

"\t\"user\_ct\": \"" + userCt + "\"\n" +

"}");

if (i < userChangeCtList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"变动类型\",\n" +

" \"id\": \"type\"\n" +

" },\n" +

" {\n" +

" \"name\": \"用户数\",\n" +

" \"id\": \"user\_ct\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

" }\n" +

"}";

}

}

**7）Sugar 配置**

① 选择表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/user/userChangeCt?date=${GMALL\_DATE}

② 效果如下



### 3.2.3 用户行为漏斗分析

**1）需求介绍**

漏斗分析是一个数据分析模型，它能够科学反映一个业务过程从起点到终点各阶段用户转化情况。由于其能将各阶段环节都展示出来，故哪个阶段存在问题，就能一目了然。

图表, 漏斗图

描述已自动生成

该需求要求统计一个完整的购物流程各个阶段的人数，具体说明如下：

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 当日 | 首页浏览人数 | 略 |
| 当日 | 商品详情页浏览人数 | 略 |
| 当日 | 加购人数 | 略 |
| 当日 | 下单人数 | 略 |
| 当日 | 支付人数 | 支付成功人数 |

**2）需求分析**

本节指标展示使用轮播表格实现。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": {

"columns": [

{

"name": "页面",

"id": "page\_id"

},

{

"name": "独立访客数",

"id": "uv\_ct"

}

],

"rows": [

{

"page\_id": "home",

"uv\_ct": "2820"

},

{

"page\_id": "good\_detail",

"uv\_ct": "2638"

},

{

"page\_id": "payment",

"uv\_ct": "242"

},

{

"page\_id": "cart",

"uv\_ct": "1178"

},

{

"page\_id": "trade",

"uv\_ct": "1681"

}

]

}

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class UserPageCt {

// 页面 id

String pageId;

//独立访客数

Integer uvCt;

}

（2）Mapper 接口

在 UserStatsMapper 接口中补充查询方法。

@Select("select 'home' page\_id,\n" +

" sum(home\_uv\_ct) uvCt\n" +

" from dws\_traffic\_page\_view\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" union all\n" +

"select 'good\_detail' page\_id,\n" +

" sum(good\_detail\_uv\_ct) uvCt\n" +

" from dws\_traffic\_page\_view\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" union all\n" +

"select 'cart' page\_id,\n" +

" sum(cart\_add\_uu\_ct) uvCt\n" +

" from dws\_trade\_cart\_add\_uu\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" union all\n" +

"select 'trade' page\_id,\n" +

" sum(order\_unique\_user\_count) uvCt\n" +

" from dws\_trade\_order\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" union all\n" +

"select 'payment' page\_id,\n" +

" sum(payment\_suc\_unique\_user\_count) uvCt\n" +

" from dws\_trade\_payment\_suc\_window\n" +

" where toYYYYMMDD(stt) = #{date};")

List<UserPageCt> selectUvByPage(@Param("date") Integer date);

**5）Service 层**

（1）Service 接口

在 UserStatsService 接口中补充方法。

List<UserPageCt> getUvByPage(Integer date);

（2）Service 实现类

在实现类 UserStatsServiceImpl 中补充方法

@Override

public List<UserPageCt> getUvByPage(Integer date) {

return userStatsMapper.selectUvByPage(date);

}

**6）Controller 层**

在 UserStatsController 控制类中补充方法

@RequestMapping("/uvPerPage")

public String getUvPerPage(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<UserPageCt> uvByPageList = userStatsService.getUvByPage(date);

if (uvByPageList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < uvByPageList.size(); i++) {

UserPageCt userPageCt = uvByPageList.get(i);

String pageId = userPageCt.getPageId();

Integer uvCt = userPageCt.getUvCt();

rows.append("{\n" +

" \"page\_id\": \"" + pageId + "\",\n" +

" \"uv\_ct\": \"" + uvCt + "\"\n" +

" }");

if (i < uvByPageList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"页面\",\n" +

" \"id\": \"page\_id\"\n" +

" },\n" +

" {\n" +

" \"name\": \"独立访客数\",\n" +

" \"id\": \"uv\_ct\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

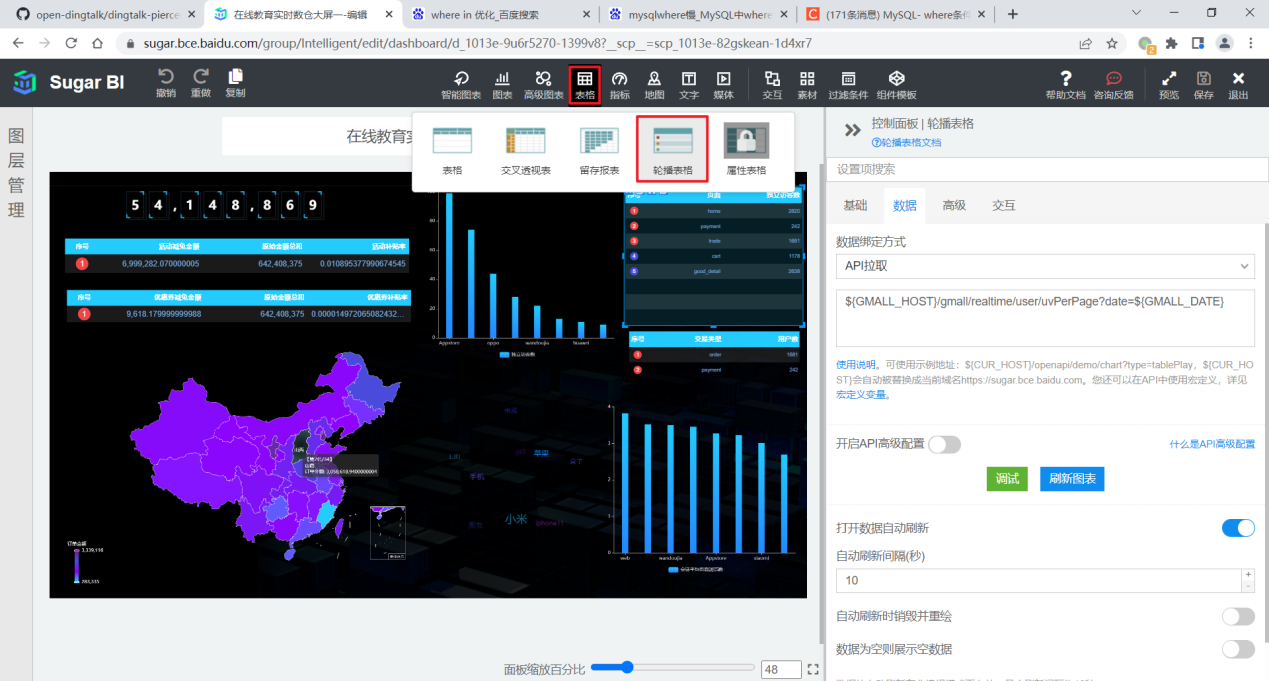
" }\n" +

"}";

}

**7）Sugar 配置**

① 选择轮播表格



② 数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/user/uvPerPage?date=${GMALL\_DATE}

③ 效果如下



### 3.2.4 新增交易用户统计

**1）需求说明**

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 当日 | 新增下单人数 | 略 |
| 当日 | 新增支付人数 | 略 |

**2）需求分析**

本节指标展示使用轮播表格实现。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": {

"columns": [

{

"name": "交易类型",

"id": "type"

},

{

"name": "新增用户数",

"id": "user\_ct"

}

],

"rows": [

{

"type": "order",

"user\_ct": "1681"

},

{

"type": "payment",

"user\_ct": "242"

}

]

}

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class UserTradeCt {

// 交易类型

String type;

// 用户数

Integer userCt;

}

（2）Mapper 接口

在 UserStatsMapper 类中补充方法。

@Select("select 'order' trade\_type,\n" +

" sum(order\_new\_user\_count) order\_new\_user\_count\n" +

" from dws\_trade\_order\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" union all\n" +

"select 'payment' trade\_type,\n" +

" sum(payment\_new\_user\_count) pay\_suc\_new\_user\_count\n" +

" from dws\_trade\_payment\_suc\_window\n" +

" where toYYYYMMDD(stt) = #{date};")

List<UserTradeCt> selectTradeUserCt(@Param("date")Integer date);

**5）Service 层**

（1）Service 接口

在 UserStatsService 接口中补充方法。

List<UserTradeCt> getTradeUserCt(Integer date);

（2）Service 实现类

在 UserStatsServiceImpl 实现类中补充方法。

@Override

public List<UserTradeCt> getTradeUserCt(Integer date) {

return userStatsMapper.selectTradeUserCt(date);

}

**6）Controller 层**

在 UserStatsController 控制类中补充方法。

@RequestMapping("/userTradeCt")

public String getUserTradeCt(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<UserTradeCt> tradeUserCtList = userStatsService.getTradeUserCt(date);

if (tradeUserCtList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < tradeUserCtList.size(); i++) {

UserTradeCt userTradeCt = tradeUserCtList.get(i);

String type = userTradeCt.getType();

Integer userCt = userTradeCt.getUserCt();

rows.append("{\n" +

"\t\"type\": \"" + type + "\",\n" +

"\t\"user\_ct\": \"" + userCt + "\"\n" +

"}");

if (i < tradeUserCtList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"交易类型\",\n" +

" \"id\": \"type\"\n" +

" },\n" +

" {\n" +

" \"name\": \"新增用户数\",\n" +

" \"id\": \"user\_ct\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

" }\n" +

"}";

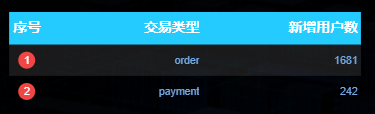
}

**7）Sugar 配置**

① 选择轮播表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/user/userTradeCt?date=${GMALL\_DATE}

② 效果如下



## 3.3 商品主题

### 3.3.1 各品牌商品交易统计

**1）需求说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 品牌 | 订单数 | 略 |
| 当日 | 品牌 | 订单人数 | 略 |
| 当日 | 品牌 | 订单金额 | 略 |
| 当日 | 品牌 | 退单数 | 略 |
| 当日 | 品牌 | 退单人数 | 略 |

**2）需求分析**

本节指标的展示使用轮播表格和环形饼图实现。其中轮播表格展示所有指标，环形饼图仅展示订单金额。

**3）数据结构**

（1）轮播表格数据结构

{

"status": 0,

"msg": "",

"data": {

"columns": [

{

"name": "品牌名称",

"id": "trademark"

},

{

"name": "订单数",

"id": "order\_count"

},

{

"name": "订单人数",

"id": "uu\_count"

},

{

"name": "订单金额",

"id": "order\_amount"

},

{

"name": "退单数",

"id": "refund\_count"

},

{

"name": "退单人数",

"id": "refund\_uu\_count"

}

],

"rows": [

{

"trademark": "索芙特",

"order\_count": "734",

"uu\_count": "628",

"order\_amount": "1075887.91",

"refund\_count": "37",

"refund\_uu\_count": "36"

},

{

"trademark": "苹果",

"order\_count": "995",

"uu\_count": "826",

"order\_amount": "1.2911363511E8",

"refund\_count": "52",

"refund\_uu\_count": "49"

},

...

]

}

}

（2）环形饼图数据结构

{

"status": 0,

"msg": "",

"data": [

{

"name": "索芙特",

"value": 1075887.91

},

{

"name": "苹果",

"value": 129113635.11

},

...

]

}

**4）Mapper 层**

（1）实体类

① 轮播表格实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TrademarkCommodityStats {

// 品牌名称

String trademarkName;

// 订单数

Integer orderCt;

// 订单人数

Integer uuCt;

// 订单金额

Double orderAmount;

// 退单数

Integer refundCt;

// 退单人数

Integer refundUuCt;

}

② 环形饼图实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TrademarkOrderAmountPieGraph {

// 品牌名称

String trademarkName;

// 销售额

Double orderAmount;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.CategoryCommodityStats;

import com.atguigu.gmall.publisher.bean.SpuCommodityStats;

import com.atguigu.gmall.publisher.bean.TrademarkOrderAmountPieGraph;

import com.atguigu.gmall.publisher.bean.TrademarkCommodityStats;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface CommodityStatsMapper {

@Select("select trademark\_name,\n" +

" order\_count,\n" +

" uu\_count,\n" +

" order\_amount,\n" +

" refund\_count,\n" +

" refund\_uu\_count\n" +

"from (select trademark\_id,\n" +

" trademark\_name,\n" +

" sum(order\_count) order\_count,\n" +

" count(distinct user\_id) uu\_count,\n" +

" sum(order\_amount) order\_amount\n" +

" from dws\_trade\_trademark\_category\_user\_spu\_order\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" group by trademark\_id, trademark\_name) oct\n" +

" full outer join\n" +

" (select trademark\_id,\n" +

" trademark\_name,\n" +

" sum(refund\_count) refund\_count,\n" +

" count(distinct user\_id) refund\_uu\_count\n" +

" from dws\_trade\_trademark\_category\_user\_refund\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" group by trademark\_id, trademark\_name) rct\n" +

" on oct.trademark\_id = rct.trademark\_id;\n")

List<TrademarkCommodityStats> selectTrademarkStats(@Param("date") Integer date);

@Select("select trademark\_name,\n" +

" sum(order\_amount) order\_amount\n" +

"from dws\_trade\_trademark\_category\_user\_spu\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by trademark\_id, trademark\_name;")

List<TrademarkOrderAmountPieGraph> selectTmOrderAmtPieGra(@Param("date")Integer date);

}

**5）Service 层**

（1）Service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.CategoryCommodityStats;

import com.atguigu.gmall.publisher.bean.SpuCommodityStats;

import com.atguigu.gmall.publisher.bean.TrademarkCommodityStats;

import com.atguigu.gmall.publisher.bean.TrademarkOrderAmountPieGraph;

import java.util.List;

public interface CommodityStatsService {

List<TrademarkCommodityStats> getTrademarkCommodityStatsService(Integer date);

List<TrademarkOrderAmountPieGraph> getTmOrderAmtPieGra(Integer date);

}

（2）Service 实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.CategoryCommodityStats;

import com.atguigu.gmall.publisher.bean.SpuCommodityStats;

import com.atguigu.gmall.publisher.bean.TrademarkCommodityStats;

import com.atguigu.gmall.publisher.bean.TrademarkOrderAmountPieGraph;

import com.atguigu.gmall.publisher.mapper.CommodityStatsMapper;

import com.atguigu.gmall.publisher.service.CommodityStatsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CommodityStatsServiceImpl implements CommodityStatsService {

@Autowired

private CommodityStatsMapper commodityStatsMapper;

@Override

public List<TrademarkCommodityStats> getTrademarkCommodityStatsService(Integer date) {

return commodityStatsMapper.selectTrademarkStats(date);

}

@Override

public List<TrademarkOrderAmountPieGraph> getTmOrderAmtPieGra(Integer date) {

return commodityStatsMapper.selectTmOrderAmtPieGra(date);

}

}

**6）Controller 层**

package com.atguigu.gmall.publisher.controller;

import com.atguigu.gmall.publisher.bean.CategoryCommodityStats;

import com.atguigu.gmall.publisher.bean.SpuCommodityStats;

import com.atguigu.gmall.publisher.bean.TrademarkCommodityStats;

import com.atguigu.gmall.publisher.bean.TrademarkOrderAmountPieGraph;

import com.atguigu.gmall.publisher.service.CommodityStatsService;

import com.atguigu.gmall.publisher.util.DateUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/gmall/realtime/commodity")

public class CommodityStatsController {

@Autowired

private CommodityStatsService commodityStatsService;

@RequestMapping("/trademark")

public String getTrademarkCommodityStats(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TrademarkCommodityStats> trademarkCommodityStatsList = commodityStatsService.getTrademarkCommodityStatsService(date);

if (trademarkCommodityStatsList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < trademarkCommodityStatsList.size(); i++) {

TrademarkCommodityStats trademarkCommodityStats = trademarkCommodityStatsList.get(i);

String trademarkName = trademarkCommodityStats.getTrademarkName();

Integer orderCt = trademarkCommodityStats.getOrderCt();

Integer uuCt = trademarkCommodityStats.getUuCt();

Double orderAmount = trademarkCommodityStats.getOrderAmount();

Integer refundCt = trademarkCommodityStats.getRefundCt();

Integer refundUuCt = trademarkCommodityStats.getRefundUuCt();

rows.append("{\n" +

"\t\"trademark\": \"" + trademarkName + "\",\n" +

"\t\"order\_count\": \"" + orderCt + "\",\n" +

"\t\"uu\_count\": \"" + uuCt + "\",\n" +

"\t\"order\_amount\": \"" + orderAmount + "\",\n" +

"\t\"refund\_count\": \"" + refundCt + "\",\n" +

"\t\"refund\_uu\_count\": \"" + refundUuCt + "\"\n" +

"}");

if (i < trademarkCommodityStatsList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"品牌名称\",\n" +

" \"id\": \"trademark\"\n" +

" },\n" +

" {\n" +

" \"name\": \"订单数\",\n" +

" \"id\": \"order\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"订单人数\",\n" +

" \"id\": \"uu\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"订单金额\",\n" +

" \"id\": \"order\_amount\"\n" +

" },\n" +

" {\n" +

" \"name\": \"退单数\",\n" +

" \"id\": \"refund\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"退单人数\",\n" +

" \"id\": \"refund\_uu\_count\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

" }\n" +

"}";

}

@RequestMapping("/tmPieGraph")

public String getTmOrderAmtPieGra(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if(date == 1) {

date = DateUtil.now();

}

List<TrademarkOrderAmountPieGraph> tmOrderAmtPieGraList = commodityStatsService.getTmOrderAmtPieGra(date);

if(tmOrderAmtPieGraList == null || tmOrderAmtPieGraList.size() == 0) {

return "";

}

StringBuilder dataSet = new StringBuilder("[");

for (int i = 0; i < tmOrderAmtPieGraList.size(); i++) {

TrademarkOrderAmountPieGraph trademarkOrderAmountPieGraph = tmOrderAmtPieGraList.get(i);

String trademarkName = trademarkOrderAmountPieGraph.getTrademarkName();

Double orderAmount = trademarkOrderAmountPieGraph.getOrderAmount();

dataSet.append("{\n" +

" \"name\": \""+ trademarkName +"\",\n" +

" \"value\": "+ orderAmount +"\n" +

" }");

if(i < tmOrderAmtPieGraList.size() - 1) {

dataSet.append(",");

} else {

dataSet.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": "+ dataSet +"\n" +

"}";

}

}

**7）Sugar 配置**

① 选择轮播表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/commodity/trademark?date=${GMALL\_DATE}

② 效果如下



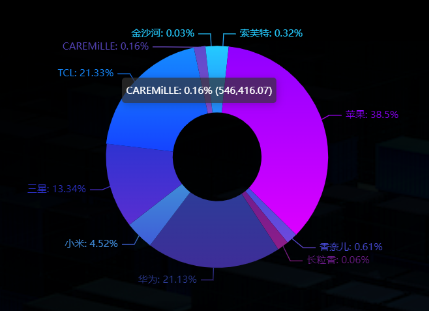
③ 选择环形饼图。



④ 数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/commodity/tmPieGraph?date=${GMALL\_DATE}

⑤ 效果如下。



### 3.3.2 各品类商品交易统计

**1）需求说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 品类 | 订单数 | 略 |
| 当日 | 品类 | 订单人数 | 略 |
| 当日 | 品类 | 订单金额 | 略 |
| 当日 | 品类 | 退单数 | 略 |
| 当日 | 品类 | 退单人数 | 略 |

**2）需求分析**

本节指标的展示使用轮播表格实现。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": {

"columns": [

{

"name": "一级品类名称",

"id": "category1\_name"

},

{

"name": "二级品类名称",

"id": "category2\_name"

},

{

"name": "三级品类名称",

"id": "category3\_name"

},

{

"name": "订单数",

"id": "order\_count"

},

{

"name": "订单人数",

"id": "uu\_count"

},

{

"name": "订单金额",

"id": "order\_amount"

},

{

"name": "退单数",

"id": "refund\_count"

},

{

"name": "退单人数",

"id": "refund\_uu\_count"

}

],

"rows": [

{

"category1\_name": "家用电器",

"category2\_name": "大 家 电",

"category3\_name": "平板电视",

"order\_count": "1834",

"uu\_count": "1000",

"order\_amount": "1.10715225E8",

"refund\_count": "71",

"refund\_uu\_count": "70.0"

},

...

]

}

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class CategoryCommodityStats {

// 一级品类名称

String category1\_name;

// 二级品类名称

String category2\_name;

// 三级品类名称

String category3\_name;

// 订单数

Integer orderCt;

// 下单用户数

Integer uuCt;

// 订单金额

Double orderAmount;

// 退单数

Integer refundCt;

// 退单金额

Double refundUcCt;

}

（2）Mapper 接口

在 CommodityStatsMapper 接口中补充方法。

@Select("select category1\_name,\n" +

" category2\_name,\n" +

" category3\_name,\n" +

" order\_count,\n" +

" uu\_count,\n" +

" order\_amount,\n" +

" refund\_count,\n" +

" refund\_uu\_count\n" +

"from (select category1\_id,\n" +

" category1\_name,\n" +

" category2\_id,\n" +

" category2\_name,\n" +

" category3\_id,\n" +

" category3\_name,\n" +

" sum(order\_count) order\_count,\n" +

" count(distinct user\_id) uu\_count,\n" +

" sum(order\_amount) order\_amount\n" +

" from dws\_trade\_trademark\_category\_user\_spu\_order\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" group by category1\_id,\n" +

" category1\_name,\n" +

" category2\_id,\n" +

" category2\_name,\n" +

" category3\_id,\n" +

" category3\_name) oct\n" +

" full outer join\n" +

" (select category1\_id,\n" +

" category1\_name,\n" +

" category2\_id,\n" +

" category2\_name,\n" +

" category3\_id,\n" +

" category3\_name,\n" +

" sum(refund\_count) refund\_count,\n" +

" count(distinct user\_id) refund\_uu\_count\n" +

" from dws\_trade\_trademark\_category\_user\_refund\_window\n" +

" where toYYYYMMDD(stt) = #{date}\n" +

" group by category1\_id,\n" +

" category1\_name,\n" +

" category2\_id,\n" +

" category2\_name,\n" +

" category3\_id,\n" +

" category3\_name) rct\n" +

" on oct.category1\_id = rct.category1\_id\n" +

" and oct.category2\_id = rct.category2\_id\n" +

" and oct.category3\_id = rct.category3\_id;")

List<CategoryCommodityStats> selectCategoryStats(@Param("date") Integer date);

**5）Service 层**

（1）Service 接口

在 CommodityStatsService 接口中补充方法。

List<CategoryCommodityStats> getCategoryStatsService(Integer date);

（2）Service 实现类

在 CommodityStatsServiceImpl 实现类中补充方法。

@Override

public List<CategoryCommodityStats> getCategoryStatsService(Integer date) {

return commodityStatsMapper.selectCategoryStats(date);

}

**6）Controller 层**

在 CommodityStatsController 类中补充方法。

@RequestMapping("/category")

public String getCategoryStats(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<CategoryCommodityStats> categoryStatsServiceList = commodityStatsService.getCategoryStatsService(date);

if (categoryStatsServiceList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < categoryStatsServiceList.size(); i++) {

CategoryCommodityStats categoryCommodityStats = categoryStatsServiceList.get(i);

String category1Name = categoryCommodityStats.getCategory1\_name();

String category2Name = categoryCommodityStats.getCategory2\_name();

String category3Name = categoryCommodityStats.getCategory3\_name();

Integer orderCt = categoryCommodityStats.getOrderCt();

Integer uuCt = categoryCommodityStats.getUuCt();

Double orderAmount = categoryCommodityStats.getOrderAmount();

Integer refundCt = categoryCommodityStats.getRefundCt();

Double refundUcCt = categoryCommodityStats.getRefundUcCt();

rows.append("{\n" +

"\t\"category1\_name\": \"" + category1Name + "\",\n" +

"\t\"category2\_name\": \"" + category2Name + "\",\n" +

"\t\"category3\_name\": \"" + category3Name + "\",\n" +

"\t\"order\_count\": \"" + orderCt + "\",\n" +

"\t\"uu\_count\": \"" + uuCt + "\",\n" +

"\t\"order\_amount\": \"" + orderAmount + "\",\n" +

"\t\"refund\_count\": \"" + refundCt + "\",\n" +

"\t\"refund\_uu\_count\": \"" + refundUcCt + "\"\n" +

"}");

if (i < categoryStatsServiceList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"一级品类名称\",\n" +

" \"id\": \"category1\_name\"\n" +

" },\n" +

" {\n" +

" \"name\": \"二级品类名称\",\n" +

" \"id\": \"category2\_name\"\n" +

" },\n" +

" {\n" +

" \"name\": \"三级品类名称\",\n" +

" \"id\": \"category3\_name\"\n" +

" },\n" +

" {\n" +

" \"name\": \"订单数\",\n" +

" \"id\": \"order\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"订单人数\",\n" +

" \"id\": \"uu\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"订单金额\",\n" +

" \"id\": \"order\_amount\"\n" +

" },\n" +

" {\n" +

" \"name\": \"退单数\",\n" +

" \"id\": \"refund\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"退单人数\",\n" +

" \"id\": \"refund\_uu\_count\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

" }\n" +

"}";

}

**7）Sugar 配置**

① 选择轮播表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/commodity/category?date=${GMALL\_DATE}

② 效果如下



### 3.3.3 各 SPU 商品交易统计

**1）需求说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | SPU | 订单数 | 略 |
| 当日 | SPU | 订单人数 | 略 |
| 当日 | SPU | 订单金额 | 略 |

**2）需求分析**

关键词的展示使用轮播表格实现。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": {

"columns": [

{

"name": "SPU 名称",

"id": "spu\_name"

},

{

"name": "下单次数",

"id": "order\_count"

},

{

"name": "下单人数",

"id": "uu\_count"

},

{

"name": "订单金额",

"id": "order\_amount"

}

],

"rows": [

{

"spu\_name": "香奈儿（Chanel）女士香水5号香水 粉邂逅柔情淡香水EDT ",

"order\_count": "565",

"uu\_count": "485",

"order\_amount": "2060032.0"

},

...

]

}

}

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class SpuCommodityStats {

// SPU 名称

String spuName;

// 下单数

Integer orderCt;

// 下单用户数

Integer uuCt;

// 下单金额

Double orderAmount;

}

（2）Mapper 接口

在 CommodityStatsMapper 接口中补充方法。

@Select("select spu\_name,\n" +

" sum(order\_count) order\_count,\n" +

" count(distinct user\_id) uu\_count,\n" +

" sum(order\_amount) order\_amount\n" +

"from dws\_trade\_trademark\_category\_user\_spu\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by spu\_id, spu\_name;")

List<SpuCommodityStats> selectSpuStats(@Param("date") Integer date);

**5）Service 层**

（1）Service 接口

在 CommodityStatsService 中补充方法。

List<SpuCommodityStats> getSpuCommodityStats(Integer date);

（2）Service 实现类

在 CommodityStatsServiceImpl 实现类中补充方法。

@Override

public List<SpuCommodityStats> getSpuCommodityStats(Integer date) {

return commodityStatsMapper.selectSpuStats(date);

}

**6）Controller 层**

在 CommodityStatsController 中补充方法。

@RequestMapping("/spu")

public String getSpuStats(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<SpuCommodityStats> spuCommodityStatsList = commodityStatsService.getSpuCommodityStats(date);

if (spuCommodityStatsList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < spuCommodityStatsList.size(); i++) {

SpuCommodityStats spuCommodityStats = spuCommodityStatsList.get(i);

String spuName = spuCommodityStats.getSpuName();

Integer orderCt = spuCommodityStats.getOrderCt();

Integer uuCt = spuCommodityStats.getUuCt();

Double orderAmount = spuCommodityStats.getOrderAmount();

rows.append("{\n" +

"\t\"spu\_name\": \"" + spuName + "\",\n" +

"\t\"order\_count\": \"" + orderCt + "\",\n" +

"\t\"uu\_count\": \"" + uuCt + "\",\n" +

"\t\"order\_amount\": \"" + orderAmount + "\"\n" +

"}");

if (i < spuCommodityStatsList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"SPU 名称\",\n" +

" \"id\": \"spu\_name\"\n" +

" },\n" +

" {\n" +

" \"name\": \"下单次数\",\n" +

" \"id\": \"order\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"下单人数\",\n" +

" \"id\": \"uu\_count\"\n" +

" },\n" +

" {\n" +

" \"name\": \"订单金额\",\n" +

" \"id\": \"order\_amount\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

" }\n" +

"}";

}

**7）Sugar 配置**

① 选择轮播表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/commodity/spu?date=${GMALL\_DATE}

② 效果如下



## 3.4 交易主题

### 3.4.1 交易综合统计

**1）需求说明**

|  |  |  |
| --- | --- | --- |
| **统计周期** | **指标** | **说明** |
| 当日 | 订单总额 | 订单最终金额 |
| 当日 | 订单数 | 略 |
| 当日 | 订单人数 | 略 |
| 当日 | 退单数 | 略 |
| 当日 | 退单人数 | 略 |

**2）需求分析**

订单总额用数字翻牌器展示，其余指标用轮播表格展示。

**3）数据结构**

（1）数字翻牌器

{

"status": 0,

"msg": "",

"data": 54148869.14

}

（2）轮播表格

同上，不再展示。

**4）Mapper 层**

（1）实体类

数字翻牌器无须实体类。

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TradeStats {

// 指标类型

String type;

// 度量值

Integer orderCt;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderAmount;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderCt;

import com.atguigu.gmall.publisher.bean.TradeStats;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface TradeStatsMapper {

**// 交易总金额**

@Select("select sum(order\_amount) order\_total\_amount\n" +

"from dws\_trade\_province\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt);")

Double selectTotalAmount(@Param("date")Integer date);

**// 下单情况统计**

@Select("select '下单数' type,\n" +

" sum(order\_count) value\n" +

"from dws\_trade\_trademark\_category\_user\_spu\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"union all\n" +

"select '下单人数' type,\n" +

" count(distinct user\_id) value\n" +

"from dws\_trade\_trademark\_category\_user\_spu\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"union all\n" +

"select '退单数' type,\n" +

" sum(refund\_count) value\n" +

"from dws\_trade\_trademark\_category\_user\_refund\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"union all\n" +

"select '退单人数' type,\n" +

" count(distinct user\_id) value\n" +

"from dws\_trade\_trademark\_category\_user\_refund\_window\n" +

"where toYYYYMMDD(stt) = #{date};")

List<TradeStats> selectTradeStats(@Param("date")Integer date);

}

**5）Service 层**

（1）Service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderAmount;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderCt;

import com.atguigu.gmall.publisher.bean.TradeStats;

import java.util.List;

public interface TradeStatsService {

Double getTotalAmount(Integer date);

List<TradeStats> getTradeStats(Integer date);

}

（2）Service 实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderAmount;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderCt;

import com.atguigu.gmall.publisher.bean.TradeStats;

import com.atguigu.gmall.publisher.mapper.TradeStatsMapper;

import com.atguigu.gmall.publisher.service.TradeStatsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class TradeStatsServiceImpl implements TradeStatsService {

@Autowired

TradeStatsMapper tradeStatsMapper;

@Override

public Double getTotalAmount(Integer date) {

return tradeStatsMapper.selectTotalAmount(date);

}

@Override

public List<TradeStats> getTradeStats(Integer date) {

return tradeStatsMapper.selectTradeStats(date);

}

}

**6）Controller 层**

package com.atguigu.gmall.publisher.controller;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderAmount;

import com.atguigu.gmall.publisher.bean.TradeProvinceOrderCt;

import com.atguigu.gmall.publisher.bean.TradeStats;

import com.atguigu.gmall.publisher.service.TradeStatsService;

import com.atguigu.gmall.publisher.util.DateUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/gmall/realtime/trade")

public class TradeController {

@Autowired

private TradeStatsService tradeStatsService;

@RequestMapping("/total")

public String getTotalAmount(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

Double totalAmount = tradeStatsService.getTotalAmount(date);

if (totalAmount == null) {

return "";

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": " + totalAmount + "\n" +

"}";

}

@RequestMapping("/stats")

public String getStats(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TradeStats> tradeStatsList = tradeStatsService.getTradeStats(date);

if (tradeStatsList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < tradeStatsList.size(); i++) {

TradeStats tradeStats = tradeStatsList.get(i);

String type = tradeStats.getType();

Integer orderCt = tradeStats.getOrderCt();

rows.append("{\n" +

"\t\"type\": \"" + type + "\",\n" +

"\t\"value\": " + orderCt + "\n" +

"}\n");

if (i < tradeStatsList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"指标类型\",\n" +

" \"id\": \"type\"\n" +

" },\n" +

" {\n" +

" \"name\": \"度量值\",\n" +

" \"id\": \"value\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

" }\n" +

"}\n";

}

}

**7）Sugar 配置**

① 选择数字翻牌器



② 数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/trade/total?date=${GMALL\_DATE}

③ 效果如下



④ 选择轮播表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/trade/stats?date=${GMALL\_DATE}

⑤ 效果如下



### 3.4.2 各省份交易统计

**1）需求说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 省份 | 订单数 | 略 |
| 当日 | 省份 | 订单金额 | 略 |

**2）需求分析**

本节将为两个指标各生成一张中国省份色彩地图。

**3）数据结构**

{

"status": 0,

"msg": "",

"data": {

"mapData": [

{

"name": "内蒙古",

"value": 12

},

{

"name": "上海",

"value": 10

},

...

],

"valueName": "订单数"

}

}

**4）Mapper 层**

（1）实体类

① 订单数实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TradeProvinceOrderCt {

// 省份名称

String provinceName;

// 订单数

Integer orderCt;

}

② 订单金额实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class TradeProvinceOrderAmount {

// 省份名称

String provinceName;

// 下单金额

Double orderAmount;

}

（2）Mapper 接口

在 TradeStatsMapper 接口中补充方法。

@Select("select province\_name,\n" +

" sum(order\_count) order\_count\n" +

"from dws\_trade\_province\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by province\_id, province\_name;")

List<TradeProvinceOrderCt> selectTradeProvinceOrderCt(@Param("date")Integer date);

@Select("select province\_name,\n" +

" sum(order\_amount) order\_amount\n" +

"from dws\_trade\_province\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by province\_id, province\_name;")

List<TradeProvinceOrderAmount> selectTradeProvinceOrderAmount(@Param("date")Integer date);

**5）Service 层**

（1）Service 接口

在 TradeStatsService 接口中补充方法。

List<TradeProvinceOrderCt> getTradeProvinceOrderCt(Integer date);

List<TradeProvinceOrderAmount> getTradeProvinceOrderAmount(Integer date);

（2）Service 实现类

在 TradeStatsServiceImpl 实现类中补充方法。

@Override

public List<TradeProvinceOrderCt> getTradeProvinceOrderCt(Integer date) {

return tradeStatsMapper.selectTradeProvinceOrderCt(date);

}

@Override

public List<TradeProvinceOrderAmount> getTradeProvinceOrderAmount(Integer date) {

return tradeStatsMapper.selectTradeProvinceOrderAmount(date);

}

**6）Controller 层**

在 TradeController 控制类中补充方法。

@RequestMapping("/provinceOrderCt")

public String getProvinceOrderCt(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TradeProvinceOrderCt> tradeProvinceOrderCtList = tradeStatsService.getTradeProvinceOrderCt(date);

if (tradeProvinceOrderCtList == null) {

return "";

}

StringBuilder mapData = new StringBuilder("[");

for (int i = 0; i < tradeProvinceOrderCtList.size(); i++) {

TradeProvinceOrderCt tradeProvinceOrderCt = tradeProvinceOrderCtList.get(i);

String provinceName = tradeProvinceOrderCt.getProvinceName();

Integer orderCt = tradeProvinceOrderCt.getOrderCt();

mapData.append("{\n" +

" \"name\": \"" + provinceName + "\",\n" +

" \"value\": " + orderCt + "\n" +

" }");

if (i < tradeProvinceOrderCtList.size() - 1) {

mapData.append(",");

} else {

mapData.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"mapData\": " + mapData + ",\n" +

" \"valueName\": \"订单数\"\n" +

" }\n" +

"}";

}

@RequestMapping("/provinceOrderAmount")

public String getProvinceOrderAmount(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<TradeProvinceOrderAmount> tradeProvinceOrderAmountList = tradeStatsService.getTradeProvinceOrderAmount(date);

if (tradeProvinceOrderAmountList == null) {

return "";

}

StringBuilder mapData = new StringBuilder("[");

for (int i = 0; i < tradeProvinceOrderAmountList.size(); i++) {

TradeProvinceOrderAmount tradeProvinceOrderAmount = tradeProvinceOrderAmountList.get(i);

String provinceName = tradeProvinceOrderAmount.getProvinceName();

Double orderAmount = tradeProvinceOrderAmount.getOrderAmount();

mapData.append("{\n" +

" \"name\": \"" + provinceName + "\",\n" +

" \"value\": " + orderAmount + "\n" +

" }");

if (i < tradeProvinceOrderAmountList.size() - 1) {

mapData.append(",");

} else {

mapData.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"mapData\": " + mapData + ",\n" +

" \"valueName\": \"订单金额\"\n" +

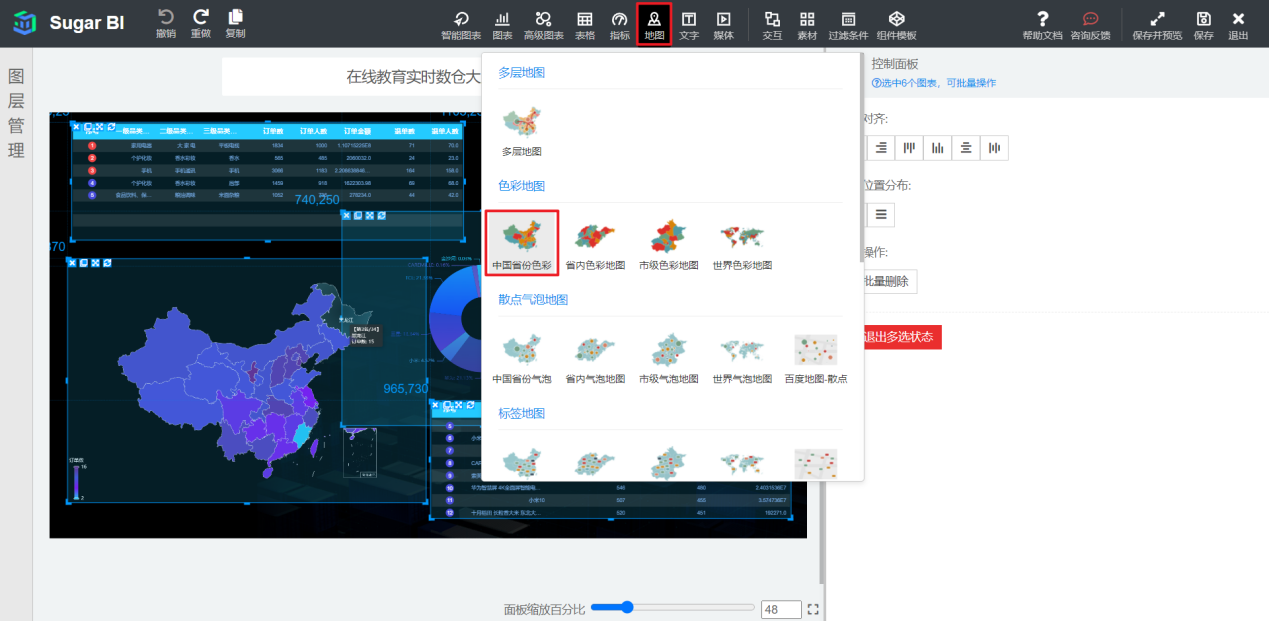
" }\n" +

"}";

}

**7）Sugar 配置**

① 选择中国省份色彩地图



② 数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

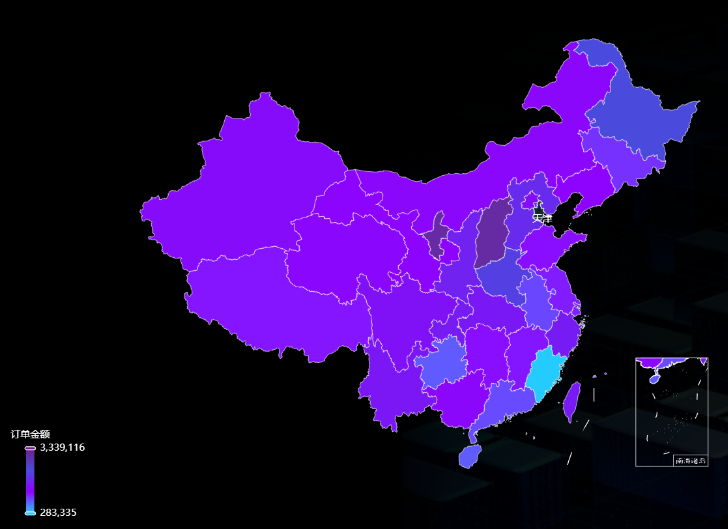
**# 订单数 URL**

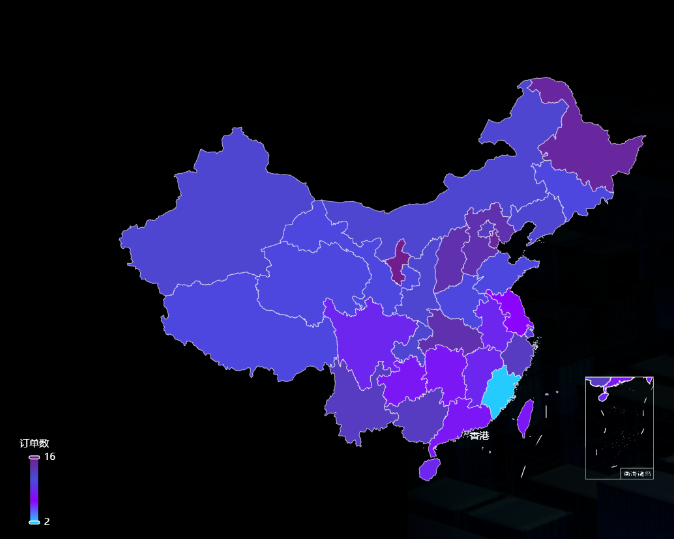
${GMALL\_HOST}/gmall/realtime/trade/provinceOrderCt?date=${GMALL\_DATE}

**# 订单金额 URL**

${GMALL\_HOST}/gmall/realtime/trade/provinceOrderAmount?date=${GMALL\_DATE}

③ 效果如下





## 3.5 优惠券主题

### 3.5.1 当日优惠券补贴率

**1）需求说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 优惠券 | 补贴率 | 用券的订单明细优惠券减免金额总和/原始金额总和 |

**2）需求分析**

本节指标使用轮播表格展示。

**3）数据结构**

前文已有展示，略。

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class CouponReduceStats {

// 优惠券减免金额

Double couponReduceAmount;

// 原始金额

Double originTotalAmount;

// 优惠券补贴率

Double couponSubsidyRate;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.CouponReduceStats;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface CouponStatsMapper {

@Select("select sum(order\_coupon\_reduce\_amount) coupon\_reduce\_amount,\n" +

" sum(order\_origin\_total\_amount) origin\_total\_amount,\n" +

" round(round(toFloat64(coupon\_reduce\_amount), 5) /\n" +

" round(toFloat64(origin\_total\_amount), 5), 20) coupon\_subsidy\_rate\n" +

"from dws\_trade\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt);")

List<CouponReduceStats> selectCouponStats(@Param("date")Integer date);

}

**5）Service 层**

（1）Service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.CouponReduceStats;

import java.util.List;

public interface CouponStatsService {

List<CouponReduceStats> getCouponStats(Integer date);

}

（2）Service 实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.CouponReduceStats;

import com.atguigu.gmall.publisher.mapper.CouponStatsMapper;

import com.atguigu.gmall.publisher.service.CouponStatsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CouponStatsServiceImpl implements CouponStatsService {

@Autowired

private CouponStatsMapper couponStatsMapper;

@Override

public List<CouponReduceStats> getCouponStats(Integer date) {

return couponStatsMapper.selectCouponStats(date);

}

}

**6）Controller 层**

package com.atguigu.gmall.publisher.controller;

import com.atguigu.gmall.publisher.bean.CouponReduceStats;

import com.atguigu.gmall.publisher.service.CouponStatsService;

import com.atguigu.gmall.publisher.util.DateUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

/\*\*

\* description:

\* Created by 铁盾 on 2022/4/19

\*/

@RestController

@RequestMapping("gmall/realtime/coupon")

public class CouponStatsController {

@Autowired

private CouponStatsService couponStatsService;

@RequestMapping("/stats")

public String getCouponStats(

@RequestParam(value = "date", defaultValue = "1") Integer date){

if(date == 1) {

date = DateUtil.now();

}

List<CouponReduceStats> couponStatsList = couponStatsService.getCouponStats(date);

if(couponStatsList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < couponStatsList.size(); i++) {

CouponReduceStats couponReduceStats = couponStatsList.get(i);

Double couponReduceAmount = couponReduceStats.getCouponReduceAmount();

Double originTotalAmount = couponReduceStats.getOriginTotalAmount();

Double couponSubsidyRate = couponReduceStats.getCouponSubsidyRate();

rows.append("{\n" +

" \"couponReduceAmount\": "+ couponReduceAmount +",\n" +

" \"originTotalAmount\": "+ originTotalAmount +",\n" +

" \"couponSubsidyRate\": "+ couponSubsidyRate +"\n" +

" }");

if(i < couponStatsList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"优惠券减免金额\",\n" +

" \"id\": \"couponReduceAmount\"\n" +

" },\n" +

" {\n" +

" \"name\": \"原始金额总和\",\n" +

" \"id\": \"originTotalAmount\"\n" +

" },\n" +

" {\n" +

" \"name\": \"优惠券补贴率\",\n" +

" \"id\": \"couponSubsidyRate\"\n" +

" }\n" +

" ],\n" +

" \"rows\": "+ rows +"\n" +

" }\n" +

"}";

}

}

**7）Sugar 配置**

① 选择轮播表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/coupon/stats?date=${GMALL\_DATE}

② 效果如下



## 3.6 活动主题

### 3.6.1 当日活动补贴率

**1）需求说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **统计周期** | **统计粒度** | **指标** | **说明** |
| 当日 | 活动 | 补贴率 | 参与促销活动的订单明细活动减免金额总和/原始金额总和 |

**2）需求分析**

本节指标使用轮播表格展示。

**3）数据结构**

略。

**4）Mapper 层**

（1）实体类

package com.atguigu.gmall.publisher.bean;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class ActivityReduceStats {

// 活动减免金额

Double activityReduceAmount;

// 原始金额

Double originTotalAmount;

// 活动补贴率

Double activitySubsidyRate;

}

（2）Mapper 接口

package com.atguigu.gmall.publisher.mapper;

import com.atguigu.gmall.publisher.bean.ActivityReduceStats;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

public interface ActivityStatsMapper {

@Select("select sum(order\_activity\_reduce\_amount) activity\_reduce\_amount,\n" +

" sum(order\_origin\_total\_amount) origin\_total\_amount,\n" +

" round(round(toFloat64(activity\_reduce\_amount), 5) /\n" +

" round(toFloat64(origin\_total\_amount), 5), 20) subsidyRate\n" +

"from dws\_trade\_order\_window\n" +

"where toYYYYMMDD(stt) = #{date}\n" +

"group by toYYYYMMDD(stt);")

List<ActivityReduceStats> selectActivityStats(@Param(value = "date")Integer date);

}

**5）Service 层**

（1）Service 接口

package com.atguigu.gmall.publisher.service;

import com.atguigu.gmall.publisher.bean.ActivityReduceStats;

import java.util.List;

public interface ActivityReduceService {

List<ActivityReduceStats> getActivityStats(Integer date);

}

（2）Service 实现类

package com.atguigu.gmall.publisher.service.impl;

import com.atguigu.gmall.publisher.bean.ActivityReduceStats;

import com.atguigu.gmall.publisher.mapper.ActivityStatsMapper;

import com.atguigu.gmall.publisher.service.ActivityReduceService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class ActivityReduceServiceImpl implements ActivityReduceService {

@Autowired

private ActivityStatsMapper activityStatsMapper;

@Override

public List<ActivityReduceStats> getActivityStats(Integer date) {

return activityStatsMapper.selectActivityStats(date);

}

}

**6）Controller 层**

package com.atguigu.gmall.publisher.controller;

import com.atguigu.gmall.publisher.bean.ActivityReduceStats;

import com.atguigu.gmall.publisher.service.ActivityReduceService;

import com.atguigu.gmall.publisher.util.DateUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/gmall/realtime/activity")

public class ActivityStatsController {

@Autowired

private ActivityReduceService activityReduceService;

@RequestMapping("/stats")

public String getActivityStats(

@RequestParam(value = "date", defaultValue = "1") Integer date) {

if (date == 1) {

date = DateUtil.now();

}

List<ActivityReduceStats> activityStatsList = activityReduceService.getActivityStats(date);

if (activityStatsList == null) {

return "";

}

StringBuilder rows = new StringBuilder("[");

for (int i = 0; i < activityStatsList.size(); i++) {

ActivityReduceStats activityReduceStats = activityStatsList.get(i);

Double activityReduceAmount = activityReduceStats.getActivityReduceAmount();

Double originTotalAmount = activityReduceStats.getOriginTotalAmount();

Double activitySubsidyRate = activityReduceStats.getActivitySubsidyRate();

rows.append("{\n" +

" \t\"activityReduceAmount\": " + activityReduceAmount + ",\n" +

" \t\"originTotalAmount\": " + originTotalAmount + ",\n" +

" \t\"activitySubsidyRate\": " + activitySubsidyRate + "\n" +

" }");

if (i < activityStatsList.size() - 1) {

rows.append(",");

} else {

rows.append("]");

}

}

return "{\n" +

" \"status\": 0,\n" +

" \"msg\": \"\",\n" +

" \"data\": {\n" +

" \"columns\": [\n" +

" {\n" +

" \"name\": \"活动减免金额\",\n" +

" \"id\": \"activityReduceAmount\"\n" +

" },\n" +

" {\n" +

" \"name\": \"原始金额总和\",\n" +

" \"id\": \"originTotalAmount\"\n" +

" },\n" +

" {\n" +

" \"name\": \"活动补贴率\",\n" +

" \"id\": \"activitySubsidyRate\"\n" +

" }\n" +

" ],\n" +

" \"rows\": " + rows + "\n" +

" }\n" +

"}";

}

}

**7）Sugar 配置**

① 选择轮播表格，数据绑定方式为 “API 拉取”，输入数据接口的 URL，如下。

${GMALL\_HOST}/gmall/realtime/activity/stats?date=${GMALL\_DATE}

② 效果如下

