尚硅谷电商数仓项目--实时计算

版本：V 1.5

张晨

# 实时需求概览

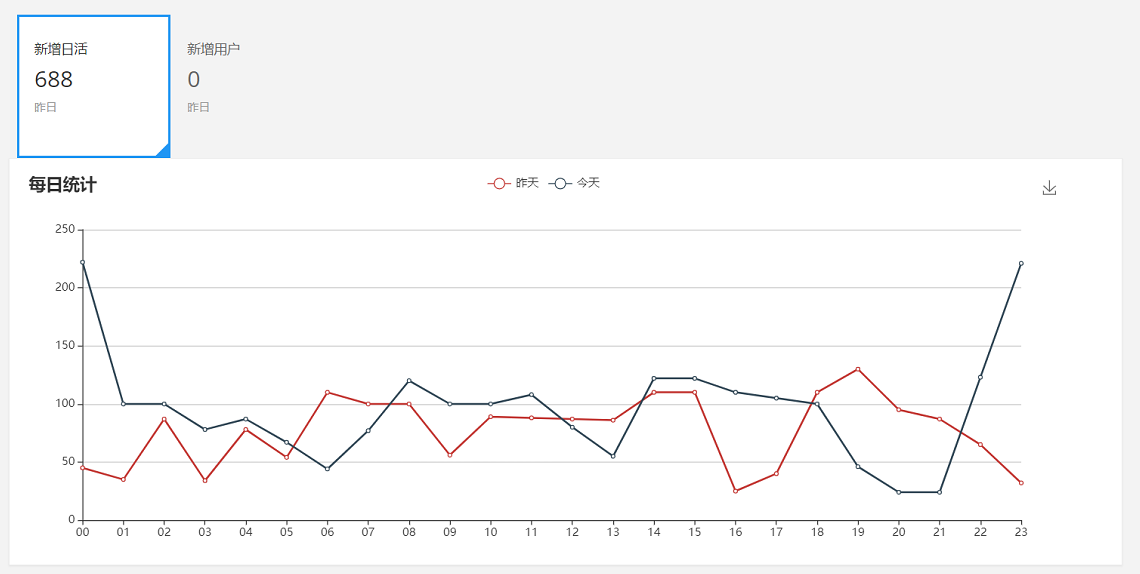
## 1 实时需求与离线需求的比较

**离线需求**，一般是根据前一日的数据生成报表等数据，虽然统计指标、报表繁多，但是对时效性不敏感。

**实时需求**主要侧重于对当日数据的实时监控，通常业务逻辑相对离线需求简单一下，统计指标也少一些，但是更注重数据的时效性，以及用户的交互性。

## 2 需求明细

### 2.1当日用户首次登录（日活）分时趋势图，昨日对比(主讲)



### 2.2当日新增用户及分时趋势图，昨日对比

### 2.3当日交易额及分时趋势图，昨日对比 (主讲)

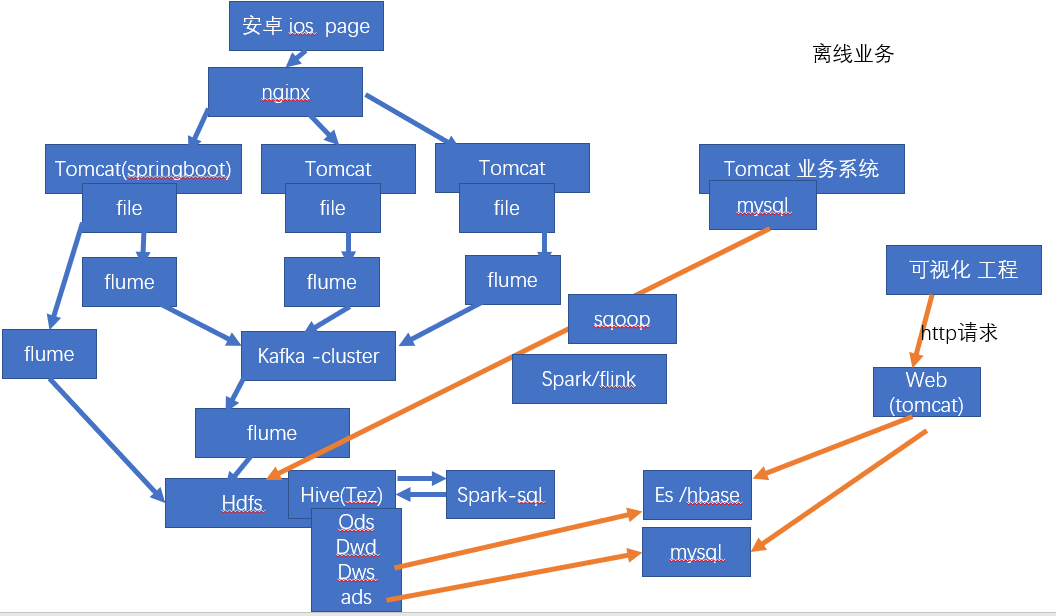
### 2.4当日订单数及分时趋势图，昨日对比

### 2.5 购物券功能风险预警 (主讲)

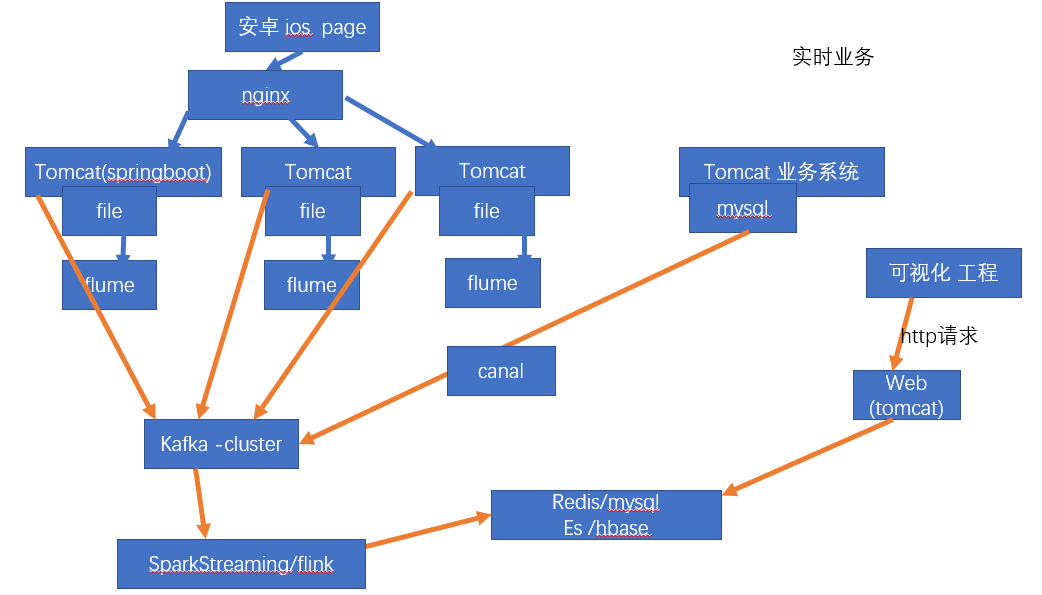
### 2.5 用户购买明细灵活分析功能 (主讲)

# 第二章 实时统计架构

## 1 离线：

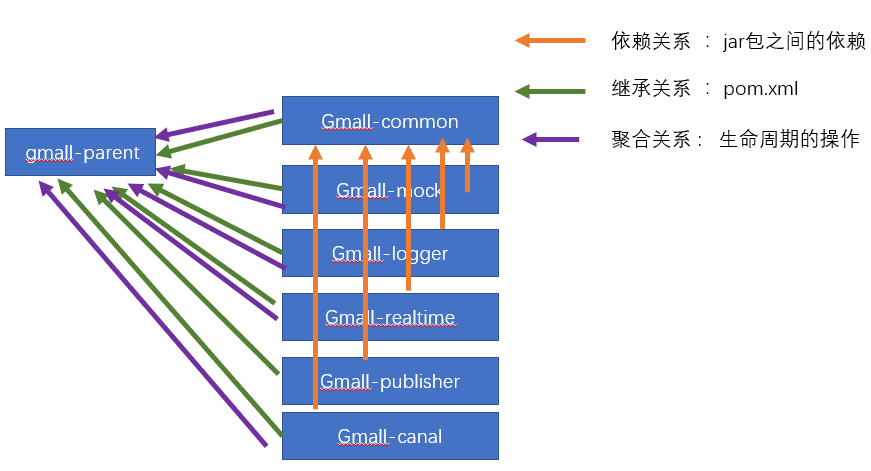


## 2 实时：



# 第三章 基础工程搭建

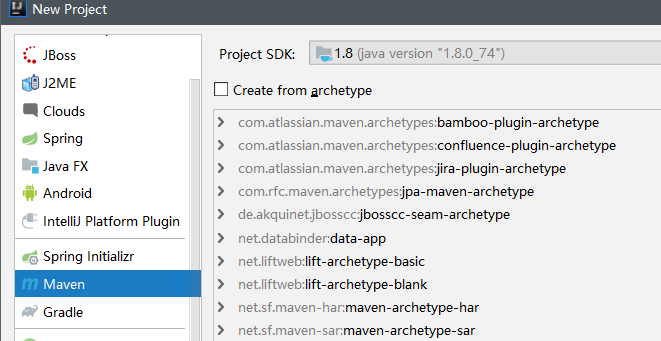
## 1、maven结构图：

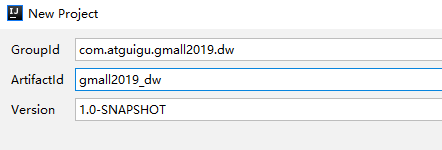


## 2、父工程：gmall2019-dw

### 2.1 建立工程

(课堂中命名可能与课件中不一致)

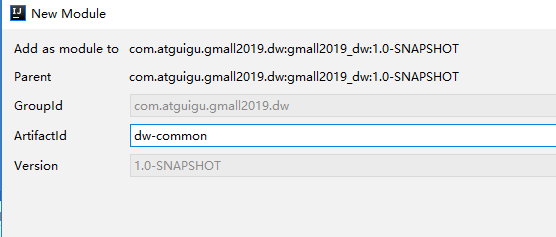




### 2.2 pom.xml

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>com.atguigu.gmall2019.dw</groupId>  <artifactId>gmall2019-dw</artifactId>  <version>1.0-SNAPSHOT</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.10.RELEASE</version>  <relativePath/> *<!-- lookup parent from repository -->* </parent>  <properties>  <spark.version>2.1.1</spark.version>  <scala.version>2.11.8</scala.version>  <log4j.version>1.2.17</log4j.version>  <slf4j.version>1.7.22</slf4j.version>   <fastjson.version>1.2.47</fastjson.version>  <httpclient.version>4.5.5</httpclient.version>  <httpmime.version>4.3.6</httpmime.version>   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version> </properties>  <dependencies>  *<!--此处放日志包，所有项目都要引用-->  <!-- 所有子项目的日志框架 -->* <dependency>  <groupId>org.slf4j</groupId>  <artifactId>jcl-over-slf4j</artifactId>  <version>${slf4j.version}</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>${slf4j.version}</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>${slf4j.version}</version>  </dependency>  *<!-- 具体的日志实现 -->* <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>${log4j.version}</version>  </dependency> </dependencies>  <dependencyManagement>  <dependencies>  *<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->* <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  <version>${httpclient.version}</version>  </dependency>  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpmime</artifactId>  <version>${httpmime.version}</version>  </dependency>   <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>${fastjson.version}</version>  </dependency>   <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-core\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>   <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-hive\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>   <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-sql\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>   <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-streaming\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>   <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-streaming-kafka-0-10\_2.11</artifactId>  <version>${spark.version}</version>  </dependency>  </dependencies> </dependencyManagement>  </project> |

## 3 子模块：公共模块

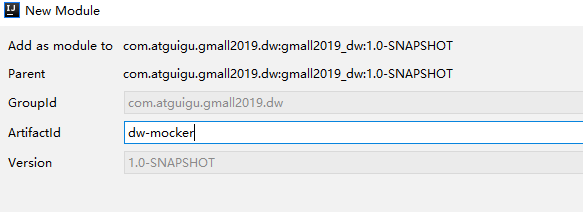


### 3.1 pom.xml

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <parent>  <artifactId>gmall2019-dw</artifactId>  <groupId>com.atguigu.gmall2019.dw</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>   <artifactId>gmall2019-common</artifactId>  <dependencies>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  </dependency>   <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  </dependency>  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpmime</artifactId>  </dependency>   </dependencies>  </project> |

## 4 子模块：模拟数据模块 mock

### 4.1 pom.xml



|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <parent>  <artifactId>gmall2019\_dw</artifactId>  <groupId>com.atguigu.gmall2019.dw</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>   <artifactId>dw-mocker</artifactId>   <dependencies>  *<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->* <dependency>  <groupId>com.atguigu.gmall2019.dw</groupId>  <artifactId>dw-common</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>   </dependencies>  </project> |

### 4.2 工具类

#### RandomDate

|  |
| --- |
| **import** java.util.Date; **import** java.util.Random;  **public class** RandomDate {  Long **logDateTime** =0L;*//* **int maxTimeStep**=0 ;    **public** RandomDate (Date startDate , Date endDate,**int** num) {   Long avgStepTime = (endDate.getTime()- startDate.getTime())/num;  **this**.**maxTimeStep**=avgStepTime.intValue()\*2;  **this**.**logDateTime**=startDate.getTime();   }    **public** Date getRandomDate() {  **int** timeStep = **new** Random().nextInt(**maxTimeStep**);  **logDateTime** = **logDateTime**+timeStep;  **return new** Date( **logDateTime**);  }  } |

#### RanOpt

|  |
| --- |
| **public class** RanOpt<T>{  T **value** ;  **int weight**;   **public** RanOpt ( T value, **int** weight ){  **this**.**value**=value ;  **this**.**weight**=weight;  }   **public** T getValue() {  **return value**;  }   **public int** getWeight() {  **return weight**;  }  } |

#### RandomOptionGroup

|  |
| --- |
| **import** java.util.ArrayList; **import** java.util.List; **import** java.util.Random;  **public class** RandomOptionGroup<T> {   **int totalWeight**=0;   List<RanOpt> **optList**=**new** ArrayList();   **public** RandomOptionGroup(RanOpt<T>... opts) {  **for** (RanOpt opt : opts) {  **totalWeight**+=opt.getWeight();  **for** (**int** i = 0; i <opt.getWeight() ; i++) {  **optList**.add(opt);  }   }  }   **public** RanOpt<T> getRandomOpt() {  **int** i = **new** Random().nextInt(**totalWeight**);  **return optList**.get(i);  }    **public static void** main(String[] args) {  RanOpt[] opts= {**new** RanOpt(**"zhang3"**,20),**new** RanOpt(**"li4"**,30),**new** RanOpt(**"wang5"**,50)};  RandomOptionGroup randomOptionGroup = **new** RandomOptionGroup(opts);  **for** (**int** i = 0; i <10 ; i++) {  System.***out***.println(randomOptionGroup.getRandomOpt().getValue());  }   }   } |

#### RandomNum

|  |
| --- |
| **import** java.util.Random;  **public class** RandomNum {  **public static final int** getRandInt(**int** fromNum,**int** toNum){  **return** fromNum+ **new** Random().nextInt(toNum-fromNum+1);  } } |

### 4.3 发送日志工具类 ：LogUploader

通过http方法发送到采集系统的web端口

|  |
| --- |
| **import** org.apache.http.HttpEntity; **import** org.apache.http.NameValuePair; **import** org.apache.http.client.ClientProtocolException; **import** org.apache.http.client.config.RequestConfig; **import** org.apache.http.client.entity.UrlEncodedFormEntity; **import** org.apache.http.client.methods.CloseableHttpResponse; **import** org.apache.http.client.methods.HttpPost; **import** org.apache.http.entity.mime.MultipartEntityBuilder; **import** org.apache.http.entity.mime.content.FileBody; **import** org.apache.http.impl.client.CloseableHttpClient; **import** org.apache.http.impl.client.HttpClients; **import** org.apache.http.message.BasicNameValuePair; **import** org.apache.http.util.EntityUtils;  **import** java.io.File; **import** java.io.IOException; **import** java.io.OutputStream; **import** java.io.UnsupportedEncodingException; **import** java.net.HttpURLConnection; **import** java.net.URL; **import** java.util.ArrayList; **import** java.util.List;  **public class** LogUploader {       **public static void** sendLogStream(String log){  **try**{  *//不同的日志类型对应不同的URL* URL url =**new** URL(**"http://logserver/log"**);   HttpURLConnection conn = (HttpURLConnection) url.openConnection();  *//设置请求方式为post* conn.setRequestMethod(**"POST"**);   *//时间头用来供server进行时钟校对的* conn.setRequestProperty(**"clientTime"**,System.*currentTimeMillis*() + **""**);  *//允许上传数据* conn.setDoOutput(**true**);  *//设置请求的头信息,设置内容类型为JSON* conn.setRequestProperty(**"Content-Type"**, **"application/x-www-form-urlencoded"**);   System.***out***.println(**"upload"** + log);   *//输出流* OutputStream out = conn.getOutputStream();  out.write((**"logString="**+log).getBytes());  out.flush();  out.close();  **int** code = conn.getResponseCode();  System.***out***.println(code);  }  **catch** (Exception e){  e.printStackTrace();  }  }  } |

### 4.4 日志生成类

|  |
| --- |
| **import** com.alibaba.fastjson.JSON; **import** com.alibaba.fastjson.JSONObject; **import** com.atguigu.gmall.dw.mock.utils.\*;  **import** java.text.ParseException; **import** java.text.SimpleDateFormat; **import** java.util.Date; **import** java.util.Random;  **public class** JsonMocker {   **int startupNum**=100000;  **int eventNum**=200000 ;   RandomDate **logDateUtil**= **null**;    RanOpt[] **osOpts**= {**new** RanOpt(**"ios"**,3),**new** RanOpt(**"andriod"**,7) };  RandomOptionGroup<String> **osOptionGroup**= **new** RandomOptionGroup(**osOpts**);  Date **startTime**= **null**;  Date **endTime**= **null**;   RanOpt[] **areaOpts**= {**new** RanOpt(**"beijing"**,10),  **new** RanOpt(**"shanghai"**,10),**new** RanOpt(**"guangdong"**,20),**new** RanOpt(**"hebei"**,5),  **new** RanOpt(**"heilongjiang"**,5),**new** RanOpt(**"shandong"**,5),**new** RanOpt(**"tianjin"**,5),  **new** RanOpt(**"shan3xi"**,5),**new** RanOpt(**"shan1xi"**,5),**new** RanOpt(**"sichuan"**,5)  };  RandomOptionGroup<String> **areaOptionGroup**= **new** RandomOptionGroup(**areaOpts**);   String **appId**=**"gmall2019"**;   RanOpt[] **vsOpts**= {**new** RanOpt(**"1.2.0"**,50),**new** RanOpt(**"1.1.2"**,15),  **new** RanOpt(**"1.1.3"**,30),  **new** RanOpt(**"1.1.1"**,5)  };   RandomOptionGroup<String> **vsOptionGroup**= **new** RandomOptionGroup(**vsOpts**);   RanOpt[] **eventOpts**= {**new** RanOpt(**"addFavor"**,10),**new** RanOpt(**"addComment"**,30),  **new** RanOpt(**"addCart"**,20), **new** RanOpt(**"clickItem"**,40)  };   RandomOptionGroup<String> **eventOptionGroup**= **new** RandomOptionGroup(**eventOpts**);   RanOpt[] **channelOpts**= {**new** RanOpt(**"xiaomi"**,10),**new** RanOpt(**"huawei"**,20),  **new** RanOpt(**"wandoujia"**,30), **new** RanOpt(**"360"**,20), **new** RanOpt(**"tencent"**,20)  , **new** RanOpt(**"baidu"**,10), **new** RanOpt(**"website"**,10)  };   RandomOptionGroup<String> **channelOptionGroup**= **new** RandomOptionGroup(**channelOpts**);   RanOpt[] **quitOpts**= { **new** RanOpt(**true**,20),**new** RanOpt(**false**,80)};   RandomOptionGroup<Boolean> **isQuitGroup**= **new** RandomOptionGroup(**quitOpts**);   **public** JsonMocker( ) {   }   **public** JsonMocker(String startTimeString ,String endTimeString,**int** startupNum,**int** eventNum) {  **try** {  **startTime**= **new** SimpleDateFormat(**"yyyy-MM-dd"**).parse(startTimeString);  **endTime**= **new** SimpleDateFormat(**"yyyy-MM-dd"**).parse(endTimeString);  } **catch** (ParseException e) {  e.printStackTrace();  }  **logDateUtil**= **new** RandomDate(**startTime**,**endTime**,startupNum+eventNum);  }   String initEventLog(String startLogJson){  */\*`type` string COMMENT '日志类型',  `mid` string COMMENT '设备唯一 表示',  `uid` string COMMENT '用户标识',  `os` string COMMENT '操作系统',  `appid` string COMMENT '应用id',  `area` string COMMENT '地区' ,  `evid` string COMMENT '事件id',  `pgid` string COMMENT '当前页',  `npgid` string COMMENT '跳转页',  `itemid` string COMMENT '商品编号',  `ts` bigint COMMENT '时间',\*/* JSONObject startLog = JSON.*parseObject*(startLogJson);  String mid= startLog.getString(**"mid"**);  String uid= startLog.getString(**"uid"**);  String os= startLog.getString(**"os"**);  String appid=**this**.**appId**;  String area=startLog.getString(**"area"**);  String evid = **eventOptionGroup**.getRandomOpt().getValue();  **int** pgid = **new** Random().nextInt(50)+1;  **int** npgid = **new** Random().nextInt(50)+1;  **int** itemid = **new** Random().nextInt(50);  *// long ts= logDateUtil.getRandomDate().getTime();* JSONObject jsonObject = **new** JSONObject();  jsonObject.put(**"type"**,**"event"**);  jsonObject.put(**"mid"**,mid);  jsonObject.put(**"uid"**,uid);  jsonObject.put(**"os"**,os);  jsonObject.put(**"appid"**,appid);  jsonObject.put(**"area"**,area);  jsonObject.put(**"evid"**,evid);  jsonObject.put(**"pgid"**,pgid);  jsonObject.put(**"npgid"**,npgid);  jsonObject.put(**"itemid"**,itemid);  **return** jsonObject.toJSONString();  }    String initStartupLog( ){  */\*`type` string COMMENT '日志类型',  `mid` string COMMENT '设备唯一标识',  `uid` string COMMENT '用户标识',  `os` string COMMENT '操作系统', ,  `appId` string COMMENT '应用id', ,  `vs` string COMMENT '版本号',  `ts` bigint COMMENT '启动时间', ,  `area` string COMMENT '城市' \*/* String mid= **"mid\_"**+ RandomNum.*getRandInt*(1,500);  String uid=**""**+ RandomNum.*getRandInt*(1,500);  String os=**osOptionGroup**.getRandomOpt().getValue();  String appid=**this**.**appId**;  String area=**areaOptionGroup**.getRandomOpt().getValue();  String vs = **vsOptionGroup**.getRandomOpt().getValue();  *//long ts= logDateUtil.getRandomDate().getTime();* String ch=os.equals(**"ios"**)?**"appstore"**: **channelOptionGroup**.getRandomOpt().getValue();     JSONObject jsonObject = **new** JSONObject();  jsonObject.put(**"type"**,**"startup"**);  jsonObject.put(**"mid"**,mid);  jsonObject.put(**"uid"**,uid);  jsonObject.put(**"os"**,os);  jsonObject.put(**"appid"**,appid);  jsonObject.put(**"area"**,area);  jsonObject.put(**"ch"**,ch);  jsonObject.put(**"vs"**,vs);  **return** jsonObject.toJSONString();  }  **public static void** genLog() {  JsonMocker jsonMocker = **new** JsonMocker();  jsonMocker.**startupNum** = 1000000;  **for** (**int** i = 0; i < jsonMocker.**startupNum**; i++) {  String startupLog = jsonMocker.initStartupLog();  jsonMocker.sendLog(startupLog);  **while** (!jsonMocker.**isQuitGroup**.getRandomOpt().getValue()) {  String eventLog = jsonMocker.initEventLog(startupLog);  jsonMocker.sendLog(eventLog);  }  **try** {  Thread.*sleep*(20);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }   }  **public void** sendLog(String log) {  LogUploader.*sendLogStream*(log);  }   **public static void** main(String[] args) {  *genLog*();  }  } |

# 第四章 搭建日志采集系统的集群

## 1、子模块：日志采集模块 logger --（单机开发调试）

### 1.1 springboot简介

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。 该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

#### 1.1.1有了springboot 我们就可以…

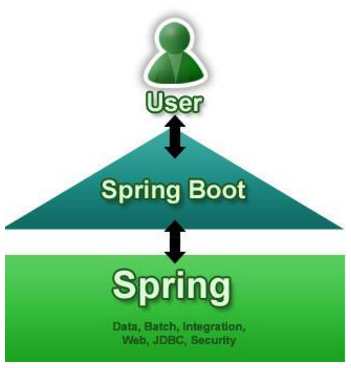
1 不再需要那些千篇一律，繁琐的xml文件。

2 内嵌Tomcat,不再需要外部的Tomcat

3 更方便的和各个第三方工具（mysql,redis,elasticsearch,dubbo,kafka等等整合），而只要维护一个配置文件即可。

#### 1.1.2 springboot和ssm的关系

springboot整合了springmvc ，spring等核心功能。也就是说本质上实现功能的还是原有的spring ,springmvc的包，但是springboot单独包装了一层，这样用户就不必直接对springmvc， spring等，在xml中配置。



#### 1.1.3 没有xml，我们要去哪配置

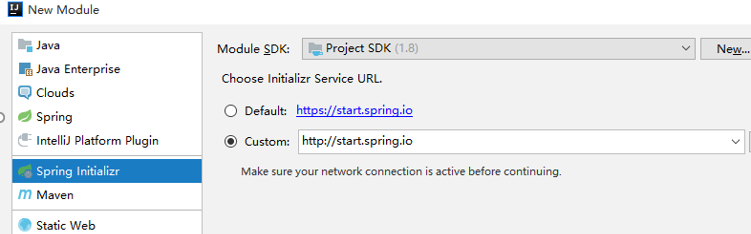
1) springboot实际上就是把以前需要用户手工配置的部分，全部作为默认项。除非用户需要额外更改不然不用配置。这就是所谓的：“约定大于配置”

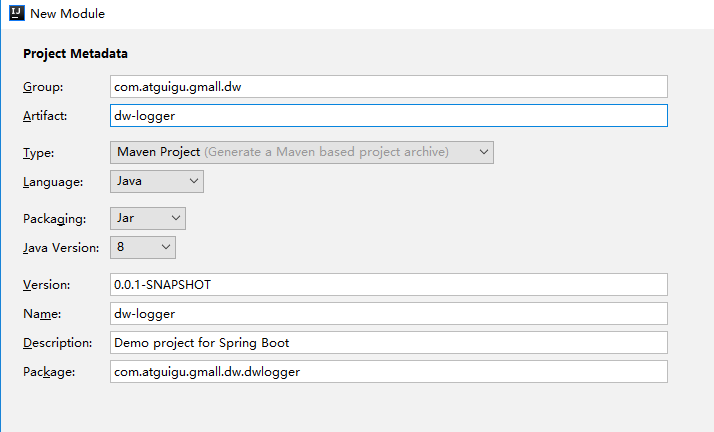
2) 如果需要特别配置的时候，去修改application.properties （application.yml）

### 1.2 快速搭建

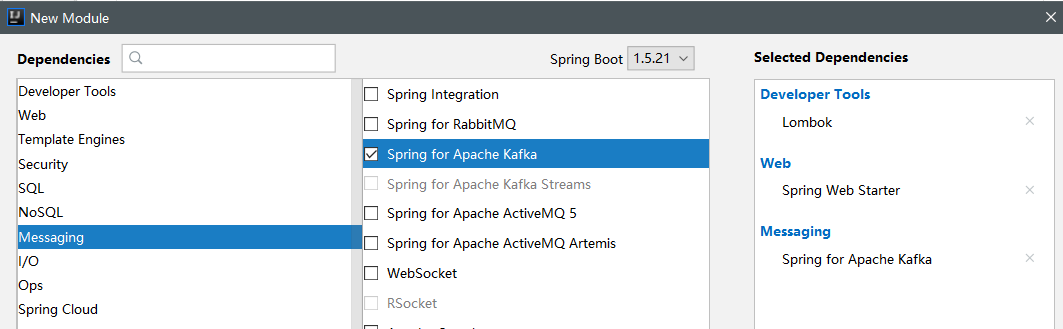
#### 1.2.1 新建module

在project下增加一个Module，选择Spring Initializr





目前企业中普遍选择1.5.x，不推荐选择2.x.x



#### 1.2.2 pom.xml

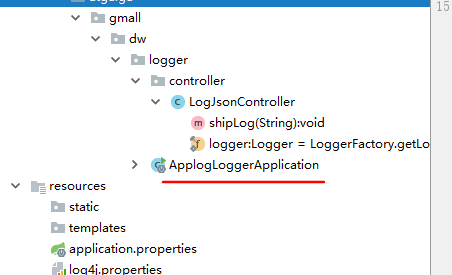
|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <parent>  <artifactId>gmall2019\_dw</artifactId>  <groupId>com.atguigu.gmall2019.dw</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <groupId>com.atguigu.gmall2019.dw</groupId>  <artifactId>dw-logger</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>dw-logger</name>  <description>Demo project for Spring Boot</description>   <properties>  <java.version>1.8</java.version>  </properties>   <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>   <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>   *<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->* <dependency>  <groupId>commons-io</groupId>  <artifactId>commons-io</artifactId>  <version>2.4</version>  </dependency>    <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  </dependency>    <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter</artifactId>  <exclusions>  <exclusion>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-logging</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-log4j</artifactId>  <version>1.3.8.RELEASE</version>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

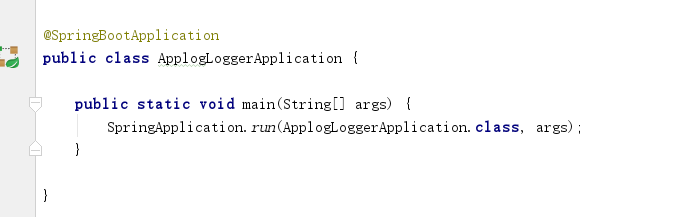
#### 1.2.3 controller

|  |
| --- |
| @Controller **public class** Demo1Controller {  @ResponseBody  @RequestMapping(**"testDemo"**)  **public** String testDemo(){  **return "hello demo"**;  } } |

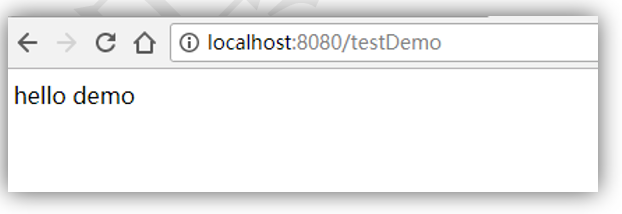
#### 1.2.4 用main方法启动tomcat

在程序中直接找到XXXXXApplication这个类



这个类只有一个main函数 ，直接执行就可以

#### 1.2.5 用浏览器测试



#### 1.2.6 修改端口号

在 resources目录下的application.properties 加入

|  |
| --- |
| server.port=80 |

### 1.3 springboot整合kafka

#### 1.3.1 application.propeties

|  |
| --- |
| *#============== kafka =================== # 指定kafka 代理地址，可以多个* spring.kafka.bootstrap-servers=hadoop1:9092   *# 指定消息key和消息体的编解码方式* spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer |

#### 1.3.2 LogJsonController

|  |
| --- |
| **import** com.alibaba.fastjson.JSON; **import** com.alibaba.fastjson.JSONObject; **import** lombok.extern.slf4j.Slf4j; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.kafka.core.KafkaTemplate; **import** org.springframework.web.bind.annotation.\*;  **import** com.atguigu.gmall.constant.GmallConstants;  @Slf4j @RestController *//Controller+responsebody* **public class** LoggerController {   @Autowired  KafkaTemplate<String,String> **kafkaTemplate**;    @PostMapping(**"log"**)  **public** String doLog(@RequestParam(**"logString"**) String logString ){   *// 0 补充时间戳* JSONObject jsonObject = JSON.*parseObject*(logString);  jsonObject.put(**"ts"**,System.*currentTimeMillis*());  *// 1 落盘 file* String jsonString = jsonObject.toJSONString();  ***log***.info(jsonObject.toJSONString());    *// 2 推送到kafka* **if**( **"startup"**.equals( jsonObject.getString(**"type"**))){  **kafkaTemplate**.send(GmallConstants.***KAFKA\_TOPIC\_STARTUP***,jsonString);  }**else**{  **kafkaTemplate**.send(GmallConstants.***KAFKA\_TOPIC\_EVENT***,jsonString);  }   **return "success"**;  }  } |

#### 1.3.3 pom.xml调整

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter</**artifactId**>  <**exclusions**>  <**exclusion**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-logging</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-log4j</**artifactId**>  <**version**>1.3.8.RELEASE</**version**>  </**dependency**> |

#### 1.3.4 在common模块

|  |
| --- |
| public class GmallConstants {   public static final String *KAFKA\_TOPIC\_STARTUP*="GMALL\_STARTUP";  public static final String *KAFKA\_TOPIC\_EVENT*="GMALL\_EVENT";  public static final String *KAFKA\_TOPIC\_NEW\_ORDER*="GMALL\_NEW\_ORDER";  public static final String *KAFKA\_TOPIC\_ORDER\_DETAIL*="GMALL\_ORDER\_DETAIL";   public static final String *ES\_INDEX\_DAU*="gmall2019\_dau";  public static final String *ES\_INDEX\_NEW\_MID*="gmall2019\_new\_mid";  public static final String *ES\_INDEX\_NEW\_ORDER*="gmall2019\_new\_order";  public static final String *ES\_INDEX\_SALE\_DETAIL*="gmall2019\_sale\_detail";  } |

#### 1.3.5 增加log4j.properties

|  |
| --- |
| log4j.appender.atguigu.MyConsole=org.apache.log4j.ConsoleAppender log4j.appender.atguigu.MyConsole.target=System.err log4j.appender.atguigu.MyConsole.layout=org.apache.log4j.PatternLayout  log4j.appender.atguigu.MyConsole.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %10p (%c:%M) - %m%n   log4j.appender.atguigu.File=org.apache.log4j.DailyRollingFileAppender log4j.appender.atguigu.File.file=d:/applog/gmall2019/log/app.log log4j.appender.atguigu.File.DatePattern='.'yyyy-MM-dd log4j.appender.atguigu.File.layout=org.apache.log4j.PatternLayout log4j.appender.atguigu.File.layout.ConversionPattern=%m%n  log4j.logger.com.atguigu.xxxxxx.XXXXcontroller=info,atguigu.File,atguigu.MyConsole |

#### 1.3.6 启动LoggerApplication ， 启动日志Mock的生成类

用kafka进行测试

|  |
| --- |
| /bigdata/kafka\_2.11-0.11.0.2/bin/kafka-console-consumer.sh --bootstrap-server hadoop1:9092,hadoop2:9092,hadoop3:9092 --topic **GMALL\_STARTUP** |

## 2、 日志采集模块打包部署--- (部署到服务器)

### 2.1 修改log4j中的配置文件

|  |
| --- |
| log4j.appender.atguigu.MyConsole=org.apache.log4j.ConsoleAppender log4j.appender.atguigu.MyConsole.target=System.error log4j.appender.atguigu.MyConsole.layout=org.apache.log4j.PatternLayout  log4j.appender.atguigu.MyConsole.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %10p (%c:%M) - %m%n   log4j.appender.atguigu.File=org.apache.log4j.DailyRollingFileAppender log4j.appender.atguigu.File.file=/applog/gmall2019/log/app.log log4j.appender.atguigu.File.DatePattern='.'yyyy-MM-dd log4j.appender.atguigu.File.layout=org.apache.log4j.PatternLayout log4j.appender.atguigu.File.layout.ConversionPattern=%m%n  log4j.logger.com.atguigu.gmall2019.xxxx.XXrController=info,atguigu.File |

### 2.2 把打好的jar包拷贝到Linux 路径下

### 2.3 启动jar包

|  |
| --- |
| java -jar /app/gmall2019/dw-logger-0.0.1-SNAPSHOT.jar |

如果出现权限问题，是因为Linux默认不允许非root账号使用1024以下的端口，所以改换为8080端口

|  |
| --- |
| java -jar /app/gmall2019/dw-logger-0.0.1-SNAPSHOT.jar –server.port=8080 >/dev/null 2>&1 & |

### 2.4 再次测试kafka消费

|  |
| --- |
| /bigdata/kafka\_2.11-0.11.0.2/bin/kafka-console-consumer.sh --bootstrap-server hadoop1:9092,hadoop2:9092,hadoop3:9092 --topic **GMALL\_STARTUP** |

## 3 搭建日志采集集群---(集群部署)

### 3.1 Nginx 入门

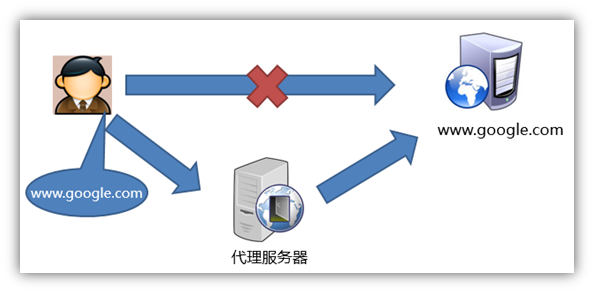
#### 3.1.1简介

*Nginx* ("engine x") 是一个**高性能的HTTP和反向代理服务器**,特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好，中国大陆使用nginx网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。

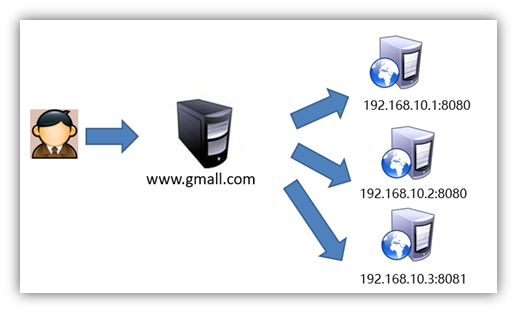
#### 3.1.2 Nginx 功能

1. 反向代理

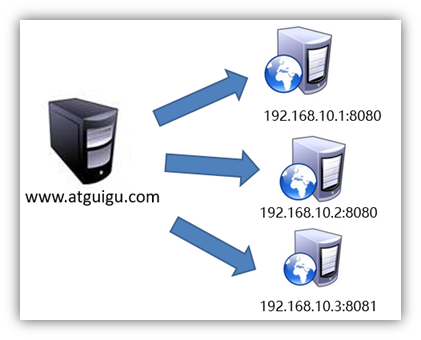
什么是反向代理？先看什么是正向代理



再看什么是反向代理



1. 负载均衡

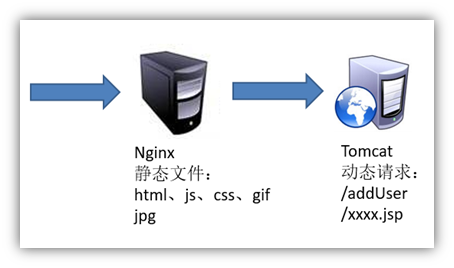


负载均衡策略： 轮询

权重

备机

1. 动静分离



#### 3.1.3 安装

##### 1） yum安装依赖包

|  |
| --- |
| sudo yum -y install openssl openssl-devel pcre pcre-devel zlib zlib-devel gcc gcc-c++ |

##### 2） 安装依赖包

|  |
| --- |
| 解压缩nginx-xx.tar.gz包。  进入解压缩目录，执行  ./configure --prefix=/opt/module/nginx  make && make install |

--prefix=要安装到的目录

##### 3） 启动、关闭命令nginx

|  |
| --- |
| 启动命令: 在/usr/local/nginx/sbin目录下执行 ./nginx  关闭命令: 在/usr/local/nginx/sbin目录下执行 ./nginx -s stop  重新加载命令: 在/usr/local/nginx/sbin目录下执行 ./nginx -s reload |

**如果启动时报错：**

|  |
| --- |
| ln -s /usr/local/lib/libpcre.so.1 /lib64 |

#### 3.1.4 赋权限

nginx占用80端口，默认情况下非root用户不允许使用1024以下端口

|  |
| --- |
| sudo setcap cap\_net\_bind\_service=+eip /bigdata/nginx/sbin/nginx |

#### 3.1.5 修改/bigdata/nginx/conf/nginx.conf

|  |
| --- |
| http{  ..........  upstream logserver{  server hadoop1:8080 weight=1;  server hadoop2:8080 weight=1;  server hadoop3:8080 weight=1;    }  server {  listen 80;  server\_name logserver;    location / {  root html;  index index.html index.htm;  proxy\_pass http://logserver;  proxy\_connect\_timeout 10;    }  ..........  } |

### 3.2 集群脚本

|  |
| --- |
| #!/bin/bash  JAVA\_BIN=/bigdata/jdk1.8.0\_152/bin/java  PROJECT=gmall2019  APPNAME=xxxxx.jar  SERVER\_PORT=8080    case $1 in  "start")  {    for i in hadoop1 hadoop2 hadoop3  do  echo "========: $i==============="  ssh $i "$JAVA\_BIN -Xms32m -Xmx64m -jar /applog/$PROJECT/$APPNAME --server.port=$SERVER\_PORT >/dev/null 2>&1 &"  done  echo "========NGINX==============="  /usr/local/nginx/sbin/nginx  };;  "stop")  {  echo "======== NGINX==============="  /usr/local/nginx/sbin/nginx -s stop  for i in hadoop1 hadoop2 hadoop3  do  echo "========: $i==============="  ssh $i "ps -ef|grep $APPNAME |grep -v grep|awk '{print \$2}'|xargs kill" >/dev/null 2>&1  done    };;  esac |