# 1. 项目背景

网站、app的运营者需要知道自己的产品或服务的运营状况，就需要对使用自己产品的用户进行各种角度的数据分析，比如：

用户数量

新增用户

留存用户

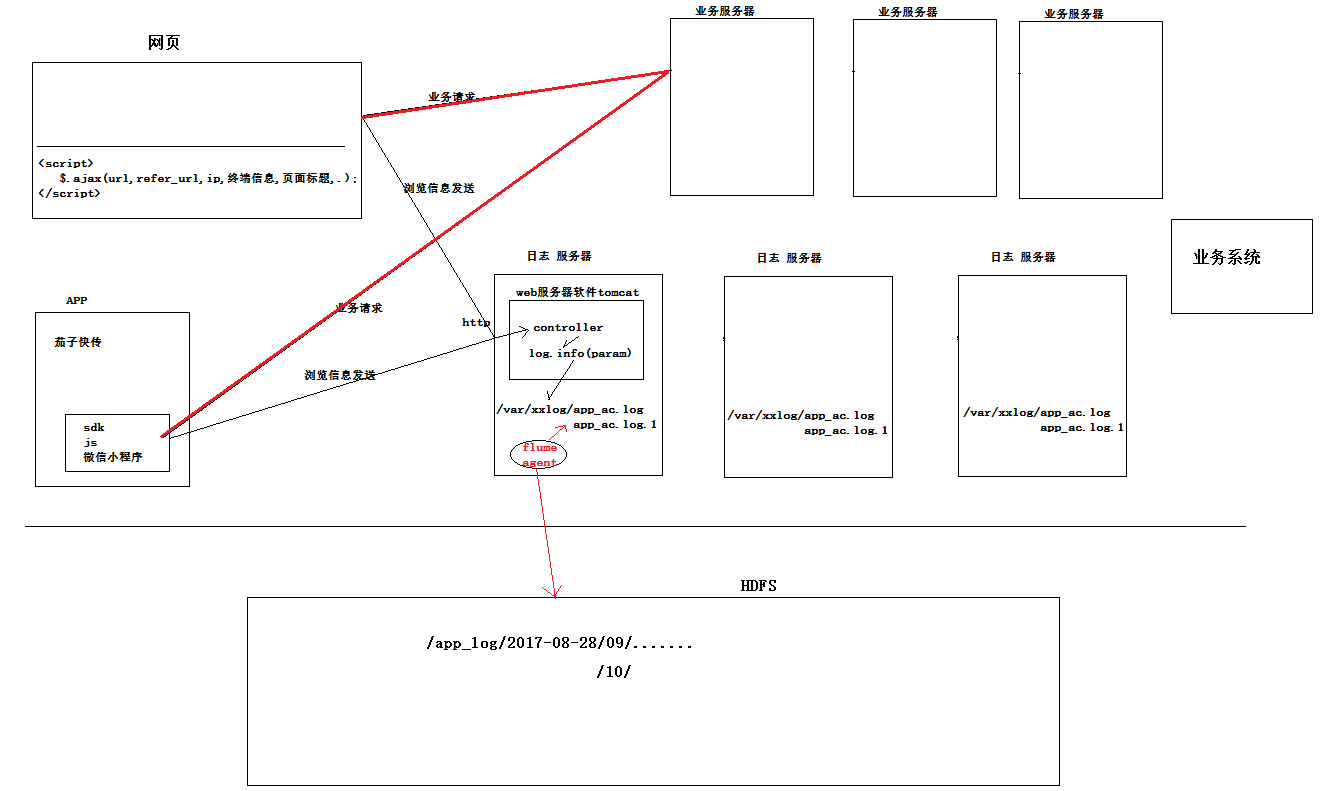
活跃用户

地域分析

渠道分析

.......

要做这样的分析，数据来源应该是用户的产品使用的行为日志，行为日志是由app或者网站的页面获取用户相关信息后，发送给后台服务器记录下来的：



# 项目开发--模块1（数据预处理）

**需求：**

1. 检查每条日志的必选字段是否完整，不完整的日志应该滤除

(

cid\_sn ;

mobile\_data\_type

os\_ver

mac ;

resolution

commit\_time ;

sdk\_ver ;

device\_id\_type ;

city ;

device\_model ;

android\_id

carrier

promotion\_channel

app\_ver\_name ;

imei ;

app\_ver\_code ;

pid

net\_type

device\_id ;

app\_device\_id

release\_channel ; ## 用户下载该app时所用的app应用市场：360，安智市场，

country ;

time\_zone ;

os\_name ; ## 操作系统名称

manufacture ## 手机生产厂家

commit\_id ; ## 提交请求的序号

account ## 账号

app\_token ; ## app名称：

app\_id ; ## app的id标识(所属的公司事业部)

language ; ## 用户的操作系统语言（）

build\_num

)

2、为每条日志添加一个用户唯一标识字段：user\_id

user\_id的取值逻辑：

如果是ios设备，user\_id=device\_id

如果是android设备， user\_id = android\_id

如果android\_id为空，则user\_id = device\_id

1. 将event字段抛弃，将header中的各字段解析成普通文本行

主要技术点：json解析 gson/fastjson/jackson/......

公司有一个变态需求：

需要将清洗后的结果数据，分ios和android和其他 三种类别，输出到3个不同的文件夹；

/app\_clean\_log/2017-09-20/ios/part-r-00000

/ios/part-r-00000

/ios/part-r-00001

/android/part-r-00000

/android/part-r-00001

/other/part-r-00000

/other/part-r-00001

百度：multipleOutputs

# 补充：MAVEN

maven是一个项目管理工具：依赖管理，编译管理，打包管理，项目结构管理等

最核心的是：依赖管理、打包管理功能

依赖管理：

maven的出发点——解放开发人员导依赖jar包的麻烦（开发人员需要一个a.jar，导入后，运行发现，a.jar中的某些类依赖另一个b.jar中的类）

maven如何解决？

——maven有一个对jar包进行描述的统一规则：任何人开发的jar包要想纳入maven的依赖管理体系，那么，这个jar包就必须有一个全球唯一的坐标：groupid、artifactId、version

——maven提供一个中央仓库，符合条件的公司或个人，可以将其开发的jar包提交到maven的中央仓库

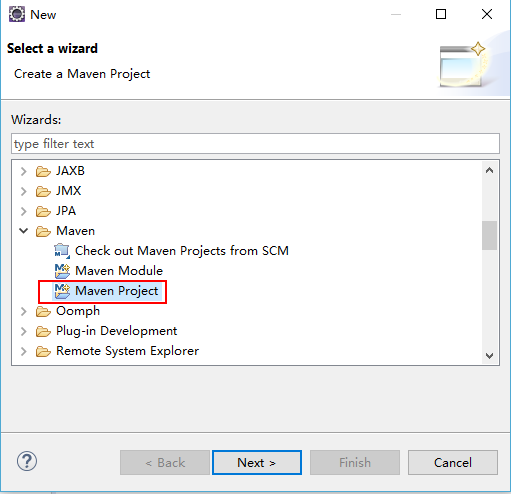
——有了上述两个机制，开发人员在开发自己的项目，需要引入第三方jar包时，只需要在maven的pom.xml文件中配置所需要的第三方jar包的坐标即可

如何使用maven？

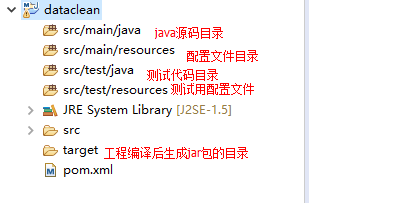
1. 先安装一个maven软件，可以独立运行，对我们的源码工程进行编译、打包
2. 在eclipse中安装一个maven插件，就可以在eclipse中无缝使用maven的功能

(eclipse中已经内置)

1. 新建一个项目时，选择——》maven 工程

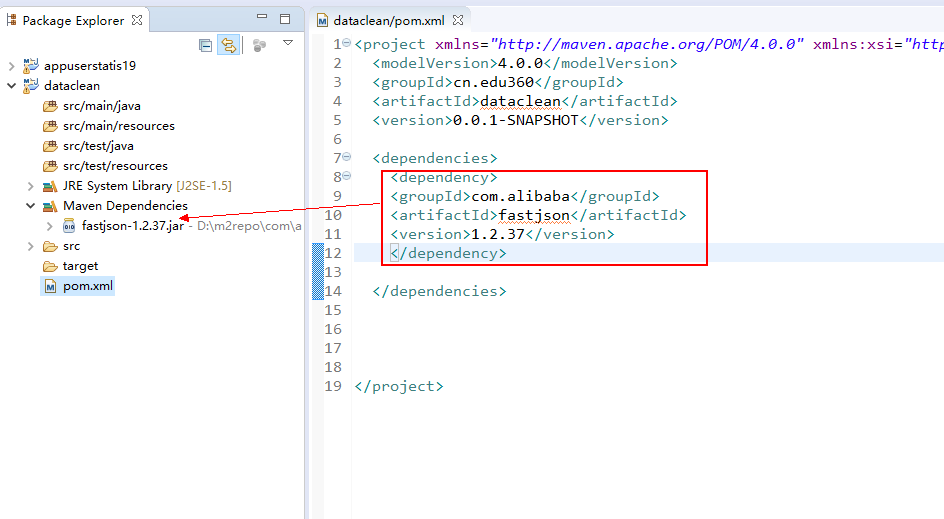


1. maven工程的结构



1. 引入依赖

可以在maven提供的中央仓库搜索网站中找自己所要的依赖：<http://search.maven.org/>



maven下载过的jar包会存在本地磁盘上的一个maven本地库目录中，下次还需要引用该jar时，maven就不需要再从中央仓库下载了

1. 开发代码

跟没有maven时一模一样

1. 构建

maven对项目构建时是按“生命周期”分目标的：

clean：清除上次构建生成的结果文件，本质上就是清空target目录

compile： 只编译

package： 编译+ 打包

install： 编译 + 打包 + 安装到本地库

操作：在项目上点右键--》Run as--》 Maven Buid...

在弹出框中，输入你想要实现的目标，比如package

1. maven中配置插件示例：

管理maven项目jdk版本的插件

在pom.xml中，与<dependencies>同级，粘贴如下配置

|  |
| --- |
| <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin>  </plugins>  </build> |

# 数据入库

提示：

1. 在hive中建表ods\_app\_log（外部表、分区表）映射预处理阶段生成的清洗数据
2. 每天定时将预处理之后的天数据 导入到 ods\_app\_log的天分区 中

# 新增用户分析

新增用户的定义：

比如，在2017-08-28日出现了一些以前从没出现过的用户，则这些用户就是2017-08-28日的新增用户

需求：

1. 将每日的新增用户从ods\_app\_log表中抽取出来，存入一个新用户信息表：

dw\_new\_user\_day的日分区中

2、统计如下报表：

某日 城市 渠道 版本 新增用户数

2017-08-28 all all all ?

2017-08-28 具体城市 all all ?

2017-08-28 all 具体渠道 all ?

2017-08-28 all all 具体版本 ?

2017-08-28 all 具体渠道 具体版本 ?

2017-08-28 具体城市 all 具体版本 ?

2017-08-28 具体城市 具体渠道 all ?

2017-08-28 具体城市 具体渠道 具体版本 ?

# 5. 活跃用户（日活）

概念：某一天使用过app的用户就是活跃用户

计算过程：

* 源表——ods\_app\_log
* 目标表：日活表

create table etl\_user\_active\_day like ods\_app\_log;

计算逻辑：

# 6. 留存用户统计

概念：

比如，15号的新增用户，在16号又活跃了，这些用户就是次日留存用户;

比如，12号的新增用户，在15号又活跃了，这些用户就是3日留存用户;

做次日留存的分析：

思路

——ETL：先抽取出次日留存用户，存入一个次日留存用户信息表；

——维度分析：统计各种维度下的留存用户数、留存用户比例

# 8. 沉默用户分析

概念：一段时间内（连续7天）没有使用过app的用户

思路：

比如，现在运算的是20号的报表，

用13号的新增用户 left join 活跃用户表的（14-20号分区）

取右表join后结果为null的用户

from etl\_user\_new\_day a left join etl\_user\_active\_day b

on a.user\_id = b.user\_id

where a.day='2017-09-13' and b.day >='2017-09-14' and b.day<='2017-09-20'

# 9. 版本升级轨迹

需求：每天统计出如下报表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 日期 | user\_id | app\_token | channel | city | source\_ver | curr\_ver |

*示例：*

*2017-08-14,许老师,共享女友,360应用,北京,v1.0*

*2017-08-14,赵老师,共享女友,安智市场,北京,v1.2*

*2017-08-14,许老师,共享女友,360应用,天津,v1.2*

*2017-08-14,许老师,共享女友,小米应用,天津,v2.0*

*2017-08-15,许老师,共享女友,360应用,北京,v2.0*

*2017-08-15,赵老师,共享女友,安智市场,北京,v1.2*

*2017-08-15,赵老师,共享女友,安智市场,北京,v1.5*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *2017-08-14* | *许老师* | *共享女友* | *360应用* | *天津* | *v1.0* | *v1.2* |
| *2017-08-14* | *许老师* | *共享女友* | *小米应用* | *天津* | *v1.2* | *v2.0* |

# 补充：hive的窗口分析函数

上述例子中的需求，就可以按如下步骤解决：

|  |
| --- |
| 2017-08-14,赵老师,共享女友,安智市场,北京,v1.2  2017-08-14,赵老师,共享女友,安智市场,北京,v1.2  2017-08-14,赵老师,共享女友,安智市场,北京,v1.2  2017-08-14,许老师,共享女友,360应用,天津,v1.2  2017-08-14,许老师,共享女友,360应用,天津,v1.2  2017-08-14,许老师,共享女友,360应用,天津,v1.2  2017-08-14,赵老师,共享女友,安智市场,北京,v1.2  2017-08-14,许老师,共享女友,360应用,天津,v1.2  2017-08-14,许老师,共享女友,360应用,天津,v1.2  2017-08-14,许老师,共享女友,360应用,天津,v1.2  2017-08-14,许老师,共享女友,360应用,天津,v1.2  2017-08-14,许老师,共享女友,小米应用,天津,v2.0  解决方案：  create table t\_lag\_test(day string,user\_id string,app\_token string,release\_channel string,city string,app\_ver\_name string)  row format delimited fields terminated by ',';  load data local inpath '/root/hivetest/ver.test' into table t\_lag\_test;  select  day,user\_id,app\_token,release\_channel,city,ver\_2,app\_ver\_name  from  (  select  day,user\_id,app\_token,release\_channel,city,app\_ver\_name,  lag(app\_ver\_name,1,null) over(partition by user\_id order by app\_ver\_name) as ver\_2  from t\_lag\_test) tmp  where ver\_2 is not null and app\_ver\_name>ver\_2  ; |

更多窗口分析函数使用方法，请参考《HIVE窗口分析函数.sql》