# 通用流量系统设计说明文档

## 设计目的

本系统旨在设计一套较为通用的流量观测系统，当新的需要观测流量的产品上线后，可以简介、方便、快速的观察它的实时流量情况。

## 设计思路

1. 将每个产品抽象为一个文件集合，根文件夹名称即为产品名，文件结构如下：

产品可以有多个，将放在WEB-INF/ classes/ products文件夹下。

1. 系统将根据products文件夹下的文件夹生成各个产品。

* menu.xml文件为菜单文件，示例如下：

<无线流量系统 dataSchemaFile=*"schema.xml"*>

<import>dimensionChannel.xml</import>

<import>dimensionBrowser.xml</import>

<import>dimensionProvince.xml</import>

<import>dimensionABtest.xml</import>

</无线流量系统>

\* dataSchemaFile关键字表示数据文件是dataSchemaXmls下的*schema.xml*文件；

\* “无线流量系统”即为本产品的展示名称；

\* import关键字指明需要引入的维度，示例中的四个文件在dimensionXmls文件夹下均可找到；

* VERSION文件起到标示版本的作用，起始数值为1，该文件夹中的内容只能是整数数字，每当需要对线上产品进行重部署（例如修改了菜单文件或者其他文件），将VERSION中的内容加一并保存即可。如果需要下线产品，将VERSION中的内容置为-1，就可以达到下线产品的目的。
* dimensionXmls文件夹中存储维度文件，文件均为并列关系且结构相同，为xml tree结构，示例如下：

<渠道 dimension=*"channel"* value=" *channel-all*">

<Sogou域名流量汇总>

<手机QQ浏览器/>

<手机腾讯网/>

<手机搜狗浏览器/>

<手机输入法APP/>

</Sogou域名流量汇总>

<SOSO域名流量汇总>

<SOSO-手机QQ浏览器/>

<SOSO-手机腾讯网/>

<SOSO-手机搜狗浏览器/>

<SOSO-手机输入法APP/>

</SOSO域名流量汇总>

</渠道>

\* dimension关键字为此维度的唯一标示，将会作为查询时的key；

\* value是所有节点默认具有的属性，如果不配置，将会和名字相同，value的值会作为request参数的值传递到后台，所以这个参数会与数据库中记录值匹配；

\* Dom节点的名称即为本维度的展示名称，同时也是菜单的描述结构；

\*可以看到节点之间的层级关系，将会反映在json对象中；

* dataSchemaXmls文件夹中存放当前产品的元数据描述信息，包括数据库信息和生成的json结构信息，一般不要修改。示例：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<schema>

<datasource name=*"wap.demo.tmp"* bigtable=*"true"*

address=*"default"* username=*"default"* password=*"default"*

columns=*"pv,uv,cl,timestamp"*/>

<!-- <datasource name="dbname" bigtable="false"

address="localhost:3306" username="root" password="root"/> -->

<json>

<list>

<struct key=*"$currentDimension"*>

<item type=*"string"* name=*"range"* value=*"$time"*/>

<item type=*"string"* name=*"count"* value=*"$count"*/>

<!-- The value pv,uv,cl should be the same order with datasource's columns-->

<columns type=*"string[]"* name=*"columns"* value=*"\*pv,uv,cl,timestamp"*/>

<points type=*"list"* name=*"points"* value=*"@TABLE"*/>

<sons type=*"list"* name=*"sons"* value=*"#INHERIT"*/>

</struct>

</list>

</json>

</schema>

\* datasource关键字描述了数据库信息，可通过bigtable区分hbase和关系型数据库；其中，columns非常重要，直接决定了从数据库读取信息的关键字和顺序；

\* json结构下的内容标示了返回给前台的数据的结构情况；

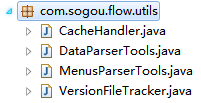
currentDimension以$号开头，代表是从request传递的参数中获得的值并把这个值作为json数据产生的入口参数，通过这个key制作数据；

struct元素内部的节点，item节点代表非必需节点，他们是从request中获得的值；其它节点都是必需的；其中，columns代表生成的json对象中的常量情况，\*符号开始的后面的值，将会按照常量输出到json中；points即为从数据库中查出的数据集；sons是层级关系的下一集合，具体请参考通用流量系统 参数说明.txt文档中的json示例对象；

## 代码结构

代码基本结构采用MVC经典模式：action、service、dao，以下采用事件驱动说明。

1. 启动Web Container，调用相关方法，向内存中加载产品信息并同时启动各个产品的版本文件追踪器：

* 加载产品信息由负责，继承自接口，加载后的信息存储在CacheHandler.menuMapper以及CacheHandler.dataSchemaMapper、CacheHandler.dimensionsMapper中。
* Map变量“menuMapper”存储菜单信息，一个产品一个菜单，key:产品名，value:MenuItemWrapper对象。
* Map变量“dataSchemaMapper”存储产品和产品相关后台数据结构描述，key:产品名，value:dom4j的Document对象。
* Map变量“dimensionsMapper”存储产品和产品中各个维度中的记录的层次关系，仅存储父子关系（即2级），key:产品名，value:list的儿子维度。
* 启动版本追踪器，由com.sogou.flow.utils.VersionFileTracker.java负责。定时扫描各个产品文件夹下的VERSION文件，按照文件最后修改时间以及VERSION文件中的版本号判断是否需要重载或者卸载产品（均由ProductLoader负责），VERSION使用见2.设计思路。涉及到使用的CacheHandler.fileTracker记录相关文件信息。

1. 触发URL，系统响应进入action包：



其中MenuServlet.java响应客户端展示菜单请求并输出json菜单对象；QueryServlet.java响应前端数据请求；具体内容均有注释，详见代码。

MenuServlet时序图如下：



QueryServlet时序图如下：

