|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **版本日期** | **修改人** | **变更描述** |
| 0.1 | 2017-10-27 | 王见雄 | 创建文档 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **Events定义**

## **事件类型和模板**

### **使用类事件**

Events in PredicitonIO are sent to the EventSever in the following form:

{

"event" : "purchase",

"entityType" : "user",

"entityId" : "1243617",

"targetEntityType" : "item",

"targetEntityId" : "iPad",

"properties" : {},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

Rules for usage events (non-PIO-reserved events like $set) are:

* event: the value must be one the the "eventName" array in engine.json
* entityType: This is always "user", do not use any other type for usage events.
* entityId: This is whatever string you use to identify a user.
* targetEntityType: This is always "item", do not use any other type for usage events.
* targetEntityId: The id for items that correspond to the "eventName". May be a product-id, category-id, a tag-id, anything that the event was connected with.
* properties: always empty and can be omitted for input but if you export the EventServer it will be output as blank.
* eventTime: the ISO8601 formatted string for the time the event occurred

### **实体属性变更类事件**

使用$set事件实现商品的属性变更:

{

"event" : "$set",

"entityType" : "item",

"entityId" : "商品ID",

"properties" : {

"category": ["商品一级分类", "商品二级分类", “商品品牌”],

"expireDate": "2016-10-05T21:02:49.228Z"

},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

商品属性一般（除非在engine.json中特别定义）为字符串数组，作为商品分类的标记。可以增加诸如"tier": ["free", "premium"]类型的属性，而通过向推荐引擎传入"fields"参数来指定非注册用户可以仅查询free类型的商品。

### **批量事件**

通过REST接口，最大一次可以导入50个Events。

## **User preference indicators**

### **Primary events定义**

本次迭代关注的关键用户行为事件：

1. 购买(生成订单)

事件列表：

|  |  |  |  |
| --- | --- | --- | --- |
| # | 事件名 | 事件说明 | 备注 |
| 1 | purchase | 用户购买 | 用户购买商品。  本次迭代仅从订单表shop\_order定期批量获取。非实时事件。 |
| 2 |  |  |  |

#### **Event：购买[purchase]**

{

"event" : "purchase",

"entityType" : "user",

"entityId" : "会员ID",

"targetEntityType" : "item",

"targetEntityId" : "商品ID",

"properties" : {},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

### **Secondary events定义**

本次迭代关注的**次**关键用户行为事件：

1. 浏览（可以从商品访问统计表获取批量历史记录），实时记录获取通过JS实现
2. 收藏
3. 加入购物车
4. 用户属性变更（创建，变更）？

本次迭代暂不考虑用户行为（放入下一迭代）：

1. 搜索（需要关联商品tag, 放到下一个迭代里）
2. 退换货

事件列表：

|  |  |  |  |
| --- | --- | --- | --- |
| # | 事件名 | 事件说明 | 备注 |
| 1 | visit | 用户访问商品 | 用户访问商品。页面获取实时数据。  测试用批量训练数据获取从goods\_click表获取 |
| 2 | bookmark | 用户收藏商品 | 历史数据从shop\_favorites获取 |
| 3 | shopping-cart | 加购物车 | 页面获取实时数据 |
| 4 | user-profile | 用户资料变更 |  |

#### **Event：浏览[visit]**

{

"event" : "visit",

"entityType" : "user",

"entityId" : "会员ID",

"targetEntityType" : "item",

"targetEntityId" : "商品ID",

"properties" : {},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

#### **Event：收藏[bookmark]**

{

"event" : "bookmark",

"entityType" : "user",

"entityId" : "会员ID",

"targetEntityType" : "item",

"targetEntityId" : "商品ID",

"properties" : {},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

#### **Event：加入购物车[shopping-cart]**

{

"event" : "shopping-cart",

"entityType" : "user",

"entityId" : "会员ID",

"targetEntityType" : "item",

"targetEntityId" : "商品ID",

"properties" : {},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

#### **Event：用户资料变更[user-profile]**

本次迭代假设影响推荐的用户属性如下：

* 用户id
* 会员等级ID
* 会员类型
* 用户性别
* 用户生日
* 登录次数
* 会员注册时间
* 上次登录时间
* 会员消费积分
* 会员定级里程
* 会员定级航段
* 可用里程余额
* 省ID
* 市ID

Event定义：

{

"event" : "$set",

"entityType" : "user",

"entityId" : "用户ID",

"properties" : {

"member-grade": ["会员级别id"],

"member-type": ["会员类型id"],

"gender": ["性别"],

"birthday": ["生日"],

"login-num": ["登录次数"],

"create-date": "2016-10-05T21:02:49.228Z" --["注册时间"],

"last-login-date": "2016-10-05T21:02:49.228Z" --["上次登录时间"],

"consumed-points": ["会员消费积分"],

"mileage": ["定级里程"],

"segment": ["定级航段"],

"balance": ["里程余额"],

"location-prvc": ["省ID"],

"location-city": ["市ID"]

},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

## **Item property changes**

本次迭代关注的次关键商品属性变更（创建，变更）事件：

1. 商品创建

本次迭代暂不考虑商品属性变更事件（放入下一迭代）：

1. 商品资料变更

### **Events定义**

事件列表：

|  |  |  |  |
| --- | --- | --- | --- |
| # | 事件名 | 事件说明 | 备注 |
| 1 | item-profile | 商品属性变更 | 商品属性变更 |
| 2 |  |  |  |

#### **Event：商品属性变更[item-profile]**

本次迭代假设影响推荐的商品属性如下：

* 商品ID
* 商品分类：gc\_id, e.g. 汽车装饰
* 商品类型: type\_id, e.g. 汽车用品吸尘器
* 商品品牌：brand\_id
* 店铺ID
* 商品关键字
* 商品使用类型（全新/二手）： goods\_form
* 商品所在地（省）
* 商品所在地（市）
* 评论次数
* 售出数量
* 收藏数量
* 是否允许退货
* 是否允许内购
* 商品价格

Event定义：

{

"event" : "$set",

"entityType" : "item",

"entityId" : "商品ID",

"properties" : {

"category": ["商品类型id", "商品分类id", “商品品牌id”],

"shop-id": ["店铺id"],

"key-words": ["商品关键字"],

"used": ["全新/二手"],

"location-prvc": ["省ID"],

"location-city": ["市ID"],

"comments": ["评论次数"],

"sold": ["售出次数"],

"bookmarked": ["收藏次数"],

"refundable": ["YES/NO"],

"emp-price": ["YES/NO"],

"item-price": ["销售价格"],

"expireDate": "2016-10-05T21:02:49.228Z"

},

"eventTime" : "2015-10-05T21:02:49.228Z"

}

# **Events配置**

TODO：导入events是否要配置engine.json中的定义？

# **批量Events导入**

## **批量文件导入Events**

对于大量的历史数据，可通过文件批量导入而不是通过HTTP协议导入。文件可以保存在本地文件系统或者HDFS上。文件每一行都是一个JSON对象字符串。例如（注：文件不能存在空行，否则会导致导入出错）：

{"event":"$set","entityType":"item","entityId":"73","properties":{"category":["13114","20044","318473"],"shop-id":["38"],"key-words":[""],"used":["1"],"location-prvc":["1"],"location-city":["36"],"comments":["0"],"sold":["0"],"bookmarked":["0"],"refundable":["0"],"emp-price":["1"],"item-price":["198"],"expireDate":"2018-10-05T21:02:49.228Z"},"eventTime":"2017-10-05T21:02:49.228Z"}

执行：

[root@master data]# pio import --appid 16 --input import\_data\_items\_inline.json

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/mnt/share/PredictionIO-0.11.0-incubating/lib/spark/pio-data-hdfs-assembly-0.11.0-incubating.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/mnt/share/PredictionIO-0.11.0-incubating/lib/pio-assembly-0.11.0-incubating.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

[INFO] [Runner$] Submission command: /mnt/share/PredictionIO-0.11.0-incubating/vendors/spark-1.6.3-bin-hadoop2.6/bin/spark-submit --class org.apache.predictionio.tools.imprt.FileToEvents --jars file:/mnt/share/PredictionIO-0.11.0-incubating/lib/spark/pio-data-elasticsearch1-assembly-0.11.0-incubating.jar,file:/mnt/share/PredictionIO-0.11.0-incubating/lib/spark/pio-data-hbase-assembly-0.11.0-incubating.jar,file:/mnt/share/PredictionIO-0.11.0-incubating/lib/spark/pio-data-hdfs-assembly-0.11.0-incubating.jar,file:/mnt/share/PredictionIO-0.11.0-incubating/lib/spark/pio-data-jdbc-assembly-0.11.0-incubating.jar,file:/mnt/share/PredictionIO-0.11.0-incubating/lib/spark/pio-data-localfs-assembly-0.11.0-incubating.jar --files file:/mnt/share/PredictionIO-0.11.0-incubating/conf/log4j.properties,file:/mnt/share/PredictionIO-0.11.0-incubating/vendors/hbase-1.2.6/conf/hbase-site.xml --driver-class-path /mnt/share/PredictionIO-0.11.0-incubating//conf:/mnt/share/hadoop-2.6.5:/mnt/share/PredictionIO-0.11.0-incubating/vendors/hbase-1.2.6/conf --driver-java-options -Dpio.log.dir=/root file:/mnt/share/PredictionIO-0.11.0-incubating/lib/pio-assembly-0.11.0-incubating.jar --appid 16 --input file:/mnt/share/PredictionIO-0.11.0-incubating/ur1/data/import\_data\_items\_inline.json --env PIO\_STORAGE\_SOURCES\_HBASE\_TYPE=hbase,PIO\_ENV\_LOADED=1,PIO\_STORAGE\_REPOSITORIES\_METADATA\_NAME=pio\_meta,PIO\_STORAGE\_SOURCES\_ELASTICSEARCH\_HOSTS=master,PIO\_STORAGE\_SOURCES\_HBASE\_HOME=/mnt/share/PredictionIO-0.11.0-incubating/vendors/hbase-1.2.6,PIO\_HOME=/mnt/share/PredictionIO-0.11.0-incubating,PIO\_STORAGE\_SOURCES\_LOCALFS\_PATH=/models,PIO\_STORAGE\_SOURCES\_ELASTICSEARCH\_TYPE=elasticsearch,PIO\_STORAGE\_REPOSITORIES\_METADATA\_SOURCE=ELASTICSEARCH,PIO\_STORAGE\_REPOSITORIES\_MODELDATA\_SOURCE=LOCALFS,PIO\_STORAGE\_REPOSITORIES\_EVENTDATA\_NAME=pio\_event,PIO\_STORAGE\_SOURCES\_ELASTICSEARCH\_HOME=/mnt/share/PredictionIO-0.11.0-incubating/vendors/elasticsearch-1.7.6,PIO\_STORAGE\_REPOSITORIES\_MODELDATA\_NAME=pio\_model,PIO\_STORAGE\_REPOSITORIES\_EVENTDATA\_SOURCE=HBASE,PIO\_CONF\_DIR=/mnt/share/PredictionIO-0.11.0-incubating//conf,PIO\_STORAGE\_SOURCES\_ELASTICSEARCH\_PORTS=9300,PIO\_STORAGE\_SOURCES\_LOCALFS\_TYPE=localfs

[INFO] [Remoting] Starting remoting

[INFO] [Remoting] Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSystem@10.10.181.127:39108]

[INFO] [FileToEvents$] Events are imported.

[INFO] [FileToEvents$] Done.

## **测试数据准备**