**API更新日志**

**API v1：**

**重构表结构，重建十张基本数据表**

**API v2：**

**新加入ItemSubCategory与Banner两张表，分别描述项目的子类与轮播图相关信息。**

**在Item中加入新字段item\_sub\_category表示项目所属子类。**

**在Item中加入新字段item\_off表示商品降价打折信息。**

**在Item中加入新字段item\_PrePrice表示商品打折前的原价信息。**

**更新排序检索API。**

**API v3：**

**新增对字段的设计理念说明**

**更新Web API：加入根据用户user\_id获取该用户购物车内的所有商品信息的接口**

**更新Web API：加入根据用户user\_id获取该用户所关注的所有商店信息**

**更新Web API：加入根据用户user\_id获取该用户所关注的所有商品信息**

**更新Web API：加入根据商品id，添加该商品的多张详情图片或返回该商品的所有详情图片路径**

**更新Web API：加入根据商品id，添加该商品的多张轮播图片或返回该商品的所有轮播图片路径**

**新加入SystemMessage表，描述系统消息，比如某某商品打折信息、发货信息等等系统消息。**

**新加入ChatMessage表，描述用户或组内聊天消息。**

**新加入Coupon表，描述优惠券信息。**

**新加入DailyOffItem表，描述每日特价商品信息**

**新加入ItemDetailImage表，存储商品图片信息**

**新加入ItemBannerImage表，存储商品轮播图信息**

**新加入Address表，存储用户收货地址信息**

**新加入Brand表，表示商品品牌**

**在Item中加入新字段item\_sale表示某商品的销量。**

**在Item中加入新字段item\_create\_time表示商品上架时间。**

**在Item中加入新字段item\_VIPOff表示该商品VIP特惠部分的价格。**

**在Item中加入新字段item\_brand表示该商品的品牌**

**在User中加入新字段user\_coupon表示用户所拥有的优惠券**

**在User中加入新字段user\_TJUCarat描述用户所具有的“洋克拉“数。**

**在User中加入新字段user\_isVIP描述该用户是否为会员。**

**在Trade中加入新字段trade\_TJUCarat表示该笔交易获得的洋克拉。**

**在ShoppingCart中加入新字段cart\_item\_amount表示该商品的数量。**

**在ItemCategory中新加入item\_category\_isHaveSub字段，表示是否有子类**

**在ItemCategory中新加入item\_category\_own\_sub字段，表示该大类拥有的子类id集合。**

**修改Item中的item\_category和item\_sub\_category字段，均外键到对应表的标识id上，且为item\_sub\_category设置缺省值1**

**修改Item中的item\_ image字段，改名为item\_preview\_image，表示该商品的单张预览图。**

**删除ItemSubCategory中的item\_subCategory\_belong字段**

**API v3.1：**

**删除ItemCategory的item\_category\_own\_sub字段**

**在ItemSubCategory中加入item\_subCategory\_belong字段**

**API v4：**

**修复了若干BUG**

**更新Web API：加入当前端浏览商品时自动更新浏览记录并合并记录、自动给商品浏览量+1、并返回该商品的信息的功能**

**更新Web API：加入当前端搜索某字符串时后端自动更新搜索记录并合并。**

**更新Web API：加入根据用户id返回该用户所有的优惠券信息**

**更新Web API：加入根据用户id、商品id直接删除收藏夹的记录**

**更新Web API：加入根据用户id返回该用户收藏夹中所有商品信息**

**更新Web API：加入根据商店id返回该商店的综合评价信息并更新该商店store\_evaluate字段**

**更新Web API：加入根据用户id和商品id判断该用户是否收藏了该商品**

**更新Web API：加入根据评价id，可以添加该评价的多张图片或返回该评价的图片路径**

**更新Web API：加入根据商品id返回该商品有多少条评价信息**

**更新Web API：加入根据商品id返回该商品的综合评价信息**

**更新Web API：加入根据用户id和商店id返回该用户是否在该店购买过商品**

**更新Web API：加入根据商品id返回该商品被多少人收藏**

**更新Web API：加入根据商品id返回“推荐商品”的详细信息（智能推荐算法）**

**更新Web API：加入根据商品id返回该商品的好评数、中评数、差评数以及有图的评价数并返回评价信息**

**更新Web API：加入根据用户id返回待收货的商品信息以及数量**

**更新Web API：加入根据用户id返回待评价的商品信息以及数量**

**更新Web API：加入根据用户id返回该用户所有已完成的订单数及商品详情**

**更新Web API：加入根据用户id、商品id、商店id、购买数量，处理前端点击“立即购买”后的逻辑**

**更新Web API：加入根据用户id、商品id、购买数量，处理前端在购物车页面进行购买的逻辑**

**更新Web API：加入根据用户id、商品id、商店id、购买数量，处理前端点击加入购物车的逻辑**

**新加入Collection表，表示收藏夹。**

**新加入BrowseRecord表，表示浏览记录**

**新加入SearchRecord表，表示搜索记录**

**新加入Evaluate表，表示评价**

**新加入EvaluateImage表，表示评价的图片**

**在Store表中新加入字段store\_evaluate表示商店的评价**

**在Banner表中新加入字段banner\_belong**

**修改Trade表中的trade\_evaluate字段，改为trade\_isEvaluate表示该交易是否评价。**

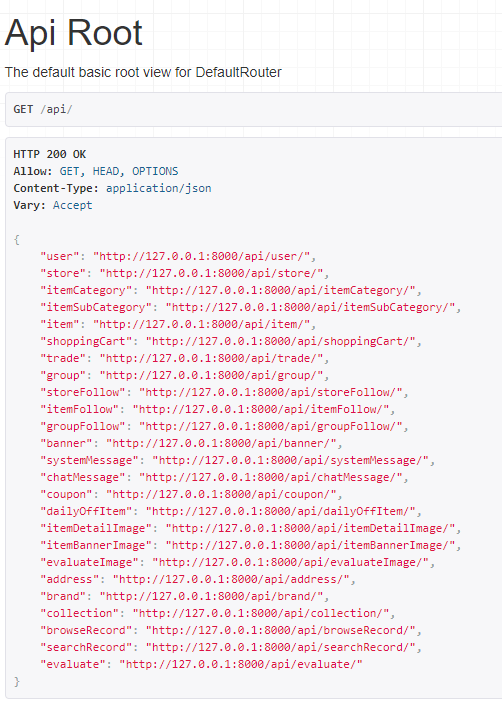
**为User表中user\_nickname增加unique属性**

**删除Banner表中的banner\_clickUrl**

**E-commerce项目后端数据库数据类型以及Web API介绍**

**Web API图形化主页：**172.23.175.252:8000/api/

**主页中有所有数据的信息，包括返回的JSON格式等等信息。**



**共有二十五张表**

**所有表都会返回一个唯一标识一条数据的id**

**所有表的主键都是自增列**

**Tips：所有外键指向的字段均指向对应表的标识id字段。**

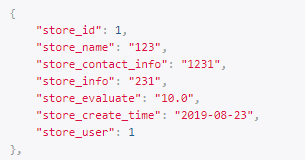
**User：用户表**

|  |  |  |
| --- | --- | --- |
| **User** | **类型** | **描述** |
| **User\_id** | 自增列 | 用户ID |
| **User\_nickname** | 字符串 | 用户昵称 |
| **User\_sex** | 字符串 | 用户性别 |
| **User\_tel** | 字符串 | 用户手机号 |
| **User\_headImage** | 可点击的URL | 用户头像 |
| **User\_isAdmin** | 布尔 | 用户是否为管理员。系统预先内置一些管理员账户，前端提交用户注册表单时此项默认置0.当用户登陆时，检查此项是否为1，为1开启管理员权限。 |
| **User\_isBindQQ** | 布尔 | 用户是否与QQ绑定 |
| **User\_isVIP** | 布尔 | 用户是否为VIP |
| **User\_location** | 字符串 | 用户位置 |
| **User\_location\_x** | 十进制小数 | 用户经度 |
| **User\_location\_y** | 十进制小数 | 用户纬度 |
| **User\_coupon** | 多对多字段 | 表示用户拥有的优惠券的id的集合 |
| **User\_TJUCarat** | 整数 | “洋克拉”，类似于京东京豆的一种优惠机制。 |



**Store：商店表**

|  |  |  |
| --- | --- | --- |
| **Store** | **类型** | **描述** |
| **Store\_id** | 自增列 | 商店ID |
| **Store\_name** | 字符串 | 商店名 |
| **Store\_user** | 外键：引用User中的user\_id，CASCADE。 | 店主的user\_id |
| **Store\_contact\_info** | 字符串 | 商店联系信息 |
| **Store\_info** | 字符串 | 商店简介 |
| **Store\_create\_time** | 标准年-月-日的日期格式 | 商店创建时间，此项在行记录创建时自动生成，后续对其他列记录的修改不影响此项。 |
| **Store\_evaluate** | 十进制小数 | 商店的整体评价，10代表未有评价信息 |



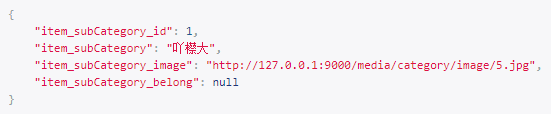
**ItemCategory：商品类表**

|  |  |  |
| --- | --- | --- |
| **ItemCategory** | **类型** | **描述** |
| **Item\_category\_id** | 自增列 | 大类ID，唯一标识一个大类 |
| **Item\_category** | 字符串 | 大类名 |
| **Item\_category\_isHaveSub** | 布尔 | 是否有子类 |



**ItemSubCategory：商品子类表**

|  |  |  |
| --- | --- | --- |
| **ItemSubCategory** | **类型** | **描述** |
| **Item\_subCategory\_id** | 自增列 | 商品子类ID |
| **Item\_subCategory** | 字符串 | 子类名 |
| **Item\_subCategory\_image** | 可点击的URL | 子类图片 |
| **Item\_subCategory\_belong** | 外键：指向ItemCategory的item\_category\_id，SET\_NULL | 该子类属于哪个大类 |



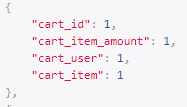
**Item：商品表**

|  |  |  |
| --- | --- | --- |
| **Item** | **类型** | **描述** |
| **Item\_id** | 自增列 | 商品ID |
| **Item\_name** | 字符串 | 商品名 |
| **Item\_price** | 十进制小数 | 商品价格 |
| **Item\_stoke** | 整数 | 商品库存 |
| **Item\_preview\_image** | 可点击的URL | 商品预览图片，可不上传，有缺省图片 |
| **Item\_category** | 外键：引用ItemCategory中的item\_category\_id，SET\_NULL | 商品大类 |
| **Item\_sub\_category** | 外键：引用ItemSubCategory中的item\_SubCategory\_id，SET（1） | 商品子类，缺省为1 |
| **Item\_browse** | 整数 | 商品浏览量 |
| **Item\_store** | 外键：引用Store中的store\_id，CASCADE | 商品所属的店铺 |
| **Item\_off** | 字符串 | 商品打折降价的说明，比如“4折”、“9折” |
| **Item\_prePrice** | 十进制小数 | 商品原价 |
| **Item\_create\_time** | 标准日期格式 | 商品上架时间 |
| **Item\_VIPOff** | 整数 | VIP特惠部分的价格，比如优惠20块 |
| **Item\_sale** | 整数 | 商品销量 |
| **Item\_brand** | 外键：引用Brand中的brand\_id，SET\_NULL | 商品品牌 |



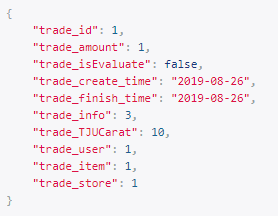
**ShoppingCart：购物车表，主要记录用户添加到购物车的商品信息**

|  |  |  |
| --- | --- | --- |
| **ShoppingCart** | **类型** | **描述** |
| **Cart\_id** | 自增列 | 购物车ID |
| **Cart\_user** | 外键：引用User中的user\_id，CASCADE | 描述改行记录的购物车添加记录是由哪个用户发起的 |
| **Cart\_item** | 外键：引用Item中的item\_id，CASCADE | 描述加入购物车的商品 |
| **Cart\_item\_amount** | 整数 | 加入购物车的某商品的数量 |



**Trade：交易表，描述用户下单的商品的相关信息。**

|  |  |  |
| --- | --- | --- |
| **Trade** | **类型** | **描述** |
| **Trade\_id** | 自增列 | 交易ID |
| **Trade\_user** | 外键：引用User中的user\_id，CASCADE | 描述改交易由哪个用户发起 |
| **Trade\_item** | 外键：引用Item中的item\_id，CASCADE | 描述交易的商品 |
| **Trade\_amount** | 整数 | 描述交易商品的数量 |
| **Trade\_store** | 外键：引用Store中的store\_id，CASCADE | 描述该交易是在哪个店铺发起的 |
| **Trade\_isEvaluate** | 布尔 | 表示该次交易是否评价 |
| **Trade\_create\_time** | 标准日期格式 | 当行记录创建时自动生成时间，后每次对行数据修改不会对其产生影响。 |
| **Trade\_finish\_time** | 标准日期格式 | 每次修改该行任意数据都会自动变更为最后修改时间。 |
| **Trade\_info** | 整数 | 暂定表示发货信息。0代表未发货，1代表正在运输中，2代表已到达目的地，3代表已确认收货，交易完成 |
| **Trade\_TJUCarat** | 整数 | 表示本次交易获得洋克拉数，默认为10 |



**Group：组表**

|  |  |  |
| --- | --- | --- |
| **Group** | **类型** | **描述** |
| **Group\_id** | 自增列 | 组ID |
| **Group\_name** | 字符串 | 组名 |
| **Group\_info** | 字符串 | 组简介 |
| **Group\_create\_time** | 标准日期格式 | 组创建时间，首次创建自动生成时间，后续修改不影响。 |



**StoreFollow：商店关注表，该表表示用户关注商店的信息**

|  |  |  |
| --- | --- | --- |
| **StoreFollow** | **类型** | **描述** |
| **Follow\_id** | 自增列 | 关注ID |
| **Follow\_user** | 外键：引用User中的user\_id，CASCADE | 表示关注发起用户的id |
| **Follow\_store** | 外键：引用Store中的store\_id，CASCADE | 表示用户关注的店id |



**ItemFollow：商品关注表，表示用户关注商品的信息**

|  |  |  |
| --- | --- | --- |
| **ItemFollow** | **类型** | **描述** |
| **Follow\_id** | 自增列 | 关注ID |
| **Follow\_user** | 外键：引用User中的user\_id，CASCADE | 表示关注发起用户的id |
| **Follow\_item** | 外键：引用Item中的item\_id，CASCADE | 表示用户关注的商品id |



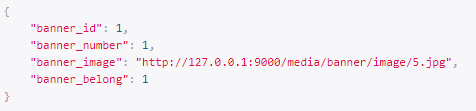
**GroupFollow：组关注表，表示用户所在群组的信息**

|  |  |  |
| --- | --- | --- |
| **GroupFollow** | **类型** | **描述** |
| **Follow\_id** | 自增列 | 关注ID |
| **Follow\_group** | 外键：引用Group中的group\_id，CASCADE | 表示用户所处的组的id |
| **Follow\_user** | 外键：引用User中的user\_id，CASCADE | 表示是哪个用户处于某组 |



**Banner：轮播图表**

|  |  |  |
| --- | --- | --- |
| **Banner** | **类型** | **描述** |
| **Banner\_id** | 自增列 | 轮播图ID |
| **Banner\_number** | 整数 | 轮播图序号 |
| **Banner\_image** | 可点击的URL | 轮播图的访问路径 |
| **Banner\_belong** | 外键：引用Item中的item\_id，CASCADE | 该轮播图属于哪个item |



**SystemMessage：系统消息**

|  |  |  |
| --- | --- | --- |
| **SystemMessage** | **类型** | **描述** |
| **Message\_id** | 自增列 | 消息ID |
| **Message\_content** | 字符串 | 消息内容 |
| **Message\_type** | 整数 | 消息类型。暂定0代表系统消息，优先级最高；1代表一些优惠促销消息等等,2代表交易物流信息等等...... |
| **Message\_create\_time** | 标准日期格式 | 消息创建时间，首次创建自动生成时间，后续修改不影响。 |



**ChatMessage：聊天消息表**

|  |  |  |
| --- | --- | --- |
| **ChatMessage** | **类型** | **描述** |
| **Message\_id** | 自增列 | 消息ID |
| **Message\_type** | 整数 | 消息类型。暂定0代表用户之间的消息记录，1代表组内消息记录 |
| **Message\_ori** | 外键：引用User中的user\_id，SET\_NULL | 消息发起者的user\_id |
| **Message\_rec** | 外键：引用User中的user\_id，SET\_NULL | 消息接收者的user\_id |
| **Message\_create\_time** | 标准日期格式 | 消息创建时间，首次创建自动生成时间，后续修改不影响。 |
| **Message\_content** | 字符串 | 消息内容 |
| **Message\_group** | 外键：引用Group中的group\_id，SET\_NULL | 该消息属于哪个群组 |



**Coupon：优惠券表**

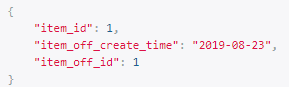
|  |  |  |
| --- | --- | --- |
| **Coupon** | **类型** | **描述** |
| **Coupon\_id** | 自增列 | 优惠券ID |
| **Coupon\_off** | 整数 | 优惠券优惠的价格 |
| **Coupon\_require** | 整数 | 使用优惠券需达到的总购买价 |
| **Coupon\_category** | 外键：引用ItemCategory中的item\_category\_id，CASCADE | 优惠券所适用的商品大类。与上两个字段合起来表示：该优惠券满XX元减XX元，只能购买XX类的商品。 |
| **Coupon\_last** | DurationField：表示一段时间，具体参考api网站 | 优惠券的有效期，该字段表示一段时间，并非日期。比如表示1小时。可精确到秒。 |



**DailyOffItem：每日特价商品表**

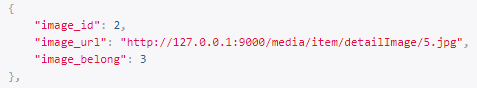
|  |  |  |
| --- | --- | --- |
| **DailyOffItem** | **类型** | **描述** |
| **Item\_id** | 自增列 | 特价ID |
| **Item\_off\_id** | 外键：引用Item中的item\_id，CASCADE | 特价商品的item\_id |
| **Item\_off\_create\_time** | 标准日期格式 | 该特价商品的创建时间 |

**Tips：该表可每天由后端人员负责修改，前端只需适配好数据格式和UI界面即可。**



**ItemDetailImage：商品详情图片表**

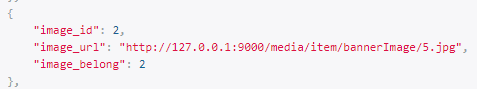
|  |  |  |
| --- | --- | --- |
| **ItemDetailImage** | **类型** | **描述** |
| **Image\_id** | 自增列 | 详情图ID |
| **Image\_url** | 可点击URL | 详情图 |
| **Image\_belong** | 外键：引用Item中的item\_id，CASCADE | 该图属于哪个商品 |



**Tips：该表在Web API部分有专门的接口来实现上传图片和查询图片**

**ItemBannerImage：商品轮播图片表**

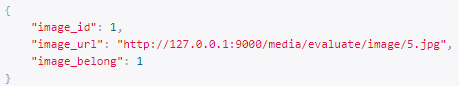
|  |  |  |
| --- | --- | --- |
| **ItemBannerImage** | **类型** | **描述** |
| **Image\_id** | 自增列 | 轮播图ID |
| **Image\_url** | 可点击URL | 轮播图 |
| **Image\_belong** | 外键：引用Item中的item\_id，CASCADE | 该图属于哪个商品 |



**Tips：该表在Web API部分有专门的接口来实现上传图片和查询图片**

**EvaluateImage：评价图片表**

|  |  |  |
| --- | --- | --- |
| **EvaluateImage** | **类型** | **描述** |
| **Image\_id** | 自增列 | 评价图ID |
| **Image\_url** | 可点击URL | 评价图 |
| **Image\_belong** | 外键：引用Evaluate中的evaluate\_id，CASCADE | 该图属于哪个评价 |



**Tips：该表在Web API部分有专门的接口来实现上传图片和查询图片**

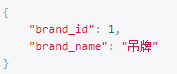
**Address：用户收货地址表**

|  |  |  |
| --- | --- | --- |
| **Address** | **类型** | **描述** |
| **Address\_id** | 自增列 | 地址ID |
| **Address\_username** | 字符串 | 用户的收货名字 |
| **Address\_tel** | 字符串 | 用户的收货手机 |
| **Address\_detail** | 字符串 | 用户的收获详细地址 |
| **Address\_type** | 字符串 | 用户的收货地址类型，比如学校、家庭等等 |
| **Address\_isDefault** | 布尔 | 该地址是否为默认地址 |
| **Address\_user** | 外键：引用User中的user\_id字段，CASCADE | 该地址所属的用户 |



**Brand：品牌表**

|  |  |  |
| --- | --- | --- |
| **Brand** | **类型** | **描述** |
| **Brand\_id** | 自增列 | 品牌ID |
| **Brand\_name** | 字符串 | 品牌名 |



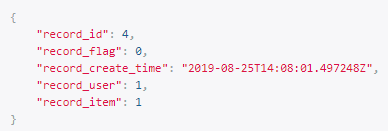
**Collection：收藏夹表**

|  |  |  |
| --- | --- | --- |
| **Collection** | **类型** | **描述** |
| **Collection\_id** | 自增列 | 收藏夹记录ID |
| **Collection\_user** | 外键：引用User中的user\_id，CASCADE | 该记录所属用户 |
| **Collection\_item** | 外键：引用Item中的item\_id，CASCADE | 关注的商品 |



**BrowseRecord：浏览记录**

|  |  |  |
| --- | --- | --- |
| **BrowseRecord** | **类型** | **描述** |
| **Record\_id** | 自增列 | 记录ID |
| **Record\_user** | 外键：引用User中的user\_id，CASCADE | 产生该浏览记录的用户 |
| **Record\_item** | 外键：引用Item中的item\_id，CASCADE | 浏览的商品 |
| **Record\_flag** | 整数 | 记录修改标志位，前端无需了解该字段作用，只需在浏览商品时返回后端用户id和商品id即可，后端已封装好API。 |
| **Record\_create\_time** | 标准日期+时间格式 | 记录的创建时间，精确到毫秒级 |



**SearchRecord：搜索记录**

|  |  |  |
| --- | --- | --- |
| **SearchRecord** | **类型** | **描述** |
| **Record\_id** | 自增列 | 记录ID |
| **Record\_user** | 外键：引用User中的user\_id，CASCADE | 产生该搜索记录的用户 |
| **Record\_content** | 字符串 | 搜索的内容 |
| **Record\_flag** | 整数 | 记录修改标志位，前端无需了解该字段作用，只需在搜索商品时返回后端用户id和搜索内容即可，后端已封装好API。 |
| **Record\_create\_time** | 标准日期+时间格式 | 记录的创建时间，精确到毫秒级 |



**Evaluate：评价表**

|  |  |  |
| --- | --- | --- |
| **Evaluate** | **类型** | **描述** |
| **Evaluate\_id** | 自增列 | 评价ID |
| **Evaluate\_user** | 外键：引用User中的user\_id，CASCADE | 评价者 |
| **Evaluate\_item** | 外键：引用Item中的item\_id，CASCADE | 评价的商品 |
| **Evaluate\_TJUCarat** | 整数 | 完成该次评价获得的洋克拉数，默认为10 |
| **Evaluate\_isAnonymous** | 布尔 | 评价是否匿名 |
| **Evaluate\_content** | 字符串 | 评价内容 |
| **Evaluate\_item\_score** | 整数 | 评价商品的分数，默认为5，表示满评，下同 |
| **Evaluate\_express\_package** | 整数 | 评价的快递包裹分数 |
| **Evaluate\_express\_speed** | 整数 | 评价的快递速度分数 |
| **Evaluate\_express\_service** | 整数 | 评价的快递服务分数 |
| **Evaluate\_create\_time** | 标准日期格式 | 评价时间 |



**Web API**

**Web API提供基本的“增删改查“、排序、以及单独编写的API接口。**

**这些功能由前端发起Request不同Method的请求来实现。**

**例如：“POST”增加数据**

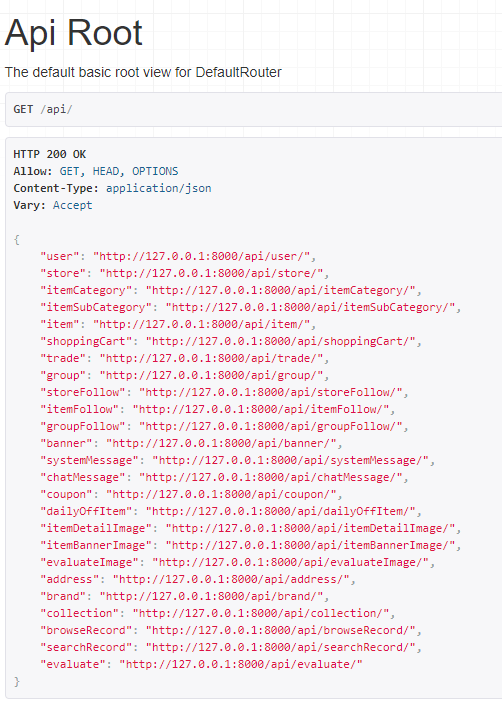
**“PUT”修改数据**

**“DELETE”删除数据**

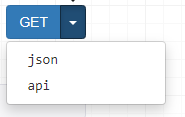
**“GET”查询数据**

**注意所在API网址是否支持你想用的请求方法。**

**先看API主页：**



**HEAD处描述该网址可以进行的请求，GET、HEAD、OPTIONS。**

**花括号内为JSON格式数据，如需直接返回JSON格式数据，需在网址后追加一些信息，具体追加什么视不同情况而定。具体网址参考**

**以下说明大部分基于API图形化界面，如需返回JSON格式，选择一下就行。**

**访问某张表的所有行信息**

在网址后追加表名即可，注意大小写敏感。

例如172.23.175.252:8000/api/user/访问user表，网址最后最好加上/防止错误。

**访问某张表中某一行数据的信息**

在网址后追加行表示id即可

例如172.23.175.252:8000/api/user/1/ 表示访问user表中id为1的行数据

另外如需删除或修改某行信息，需在类似172.23.175.252:8000/api/user/1/下进行修改，针对某一行数据的访问才具备DELETE与PUT的能力。对表访问只具备GET和POST的能力。这在界面上都有写。

**查询某张表中符合特定列条件的行信息**

以user表为例，看一个例子：

172.23.175.252:8000/api/user/?user\_id=1&user\_nickname\_\_icontains=&user\_sex=&user\_tel=&user\_isAdmin=&user\_location\_\_icontains=

每个表支持搜索的字段都不同，模式都是这样。多字段联合搜索用&连接起来即可。例如访问id为2，昵称为xx的用户信息：

172.23.175.252:8000/api/user/?user\_id=2&user\_nickname\_\_icontains=xx即可。

Icontains表示该字段是模糊搜索的，没有icontains标识的就是完全匹配搜索。注意icontains前是两个连续的下划线

**每个字段（除了外键）都可以同时提供模糊搜索和完全匹配两种搜索模式，如有特别需要说一下，我这边适配**

**查询某张表中某列符合特定范围的行信息**

有一些字段的含义可以进行范围搜索，比如某商品的价格，某店的开张日期。**像这种字段都同时提供了精确匹配、范围搜索两种模式。**

以Store表查询开店日期为例：

精确搜索

172.23.175.252:8000/api/store/?create\_date=2019-08-22

日期精确等于2019-08-22

范围搜索

172.23.175.252:8000/api/store/?min\_date=2019-08-21

日期大于2019-08-21

**日期搜索必须给出精确到日的格式，比如：2019-08-06**

**排序**

关键字段：ordering=xxx

例如：访问user表中所有行的信息，并以降序返回。

http://127.0.0.1:8000/api/user/?ordering=-user\_id

**以下为单独提供的API接口**

**不成文规定：需要1个参数的请求，只需传单个值如api/1这样即可。如果有两个或两个以上的参数，那么需要按键值对的方式给出具体参数以避免调用顺序错误，如api/?id=1&name=x这种方式，参数是可以乱序的，而用第一种方式参数必须定序。**

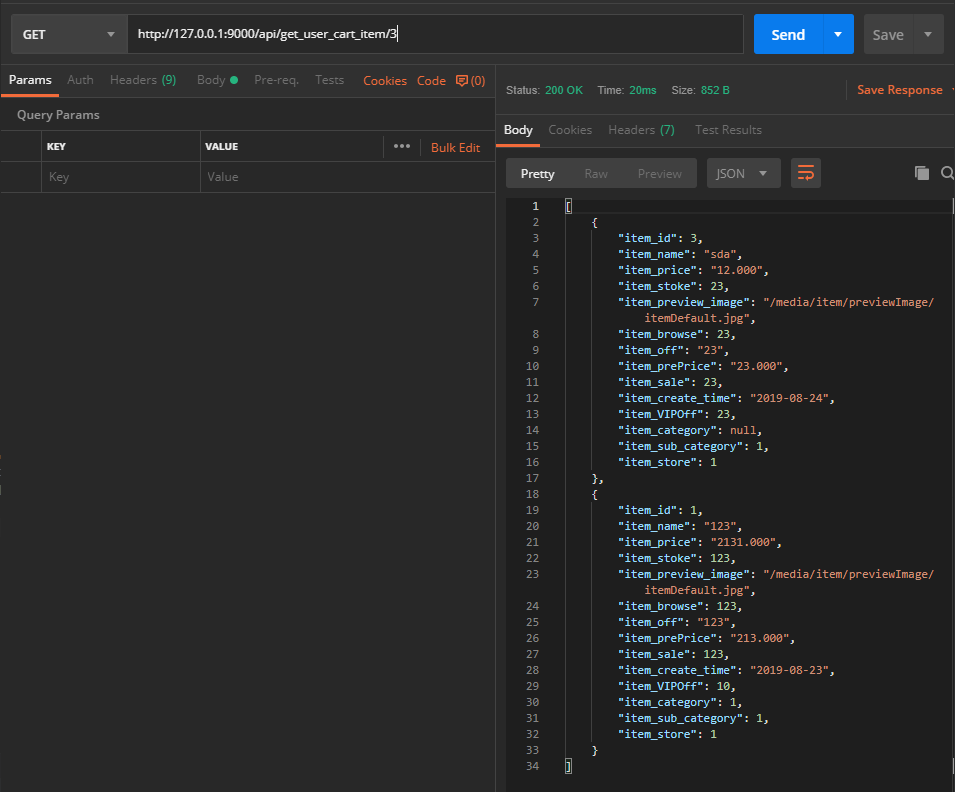
**根据用户user\_id获取该用户购物车内的所有商品信息**

接口：http://127.0.0.1:8000/api/get\_user\_cart\_item/pk

网址最后的参数pk是用户的user\_id。此接口只允许GET请求。

示例：获取用户id为3的购物车商品信息：

http://127.0.0.1:8000/api/get\_user\_cart\_item/3



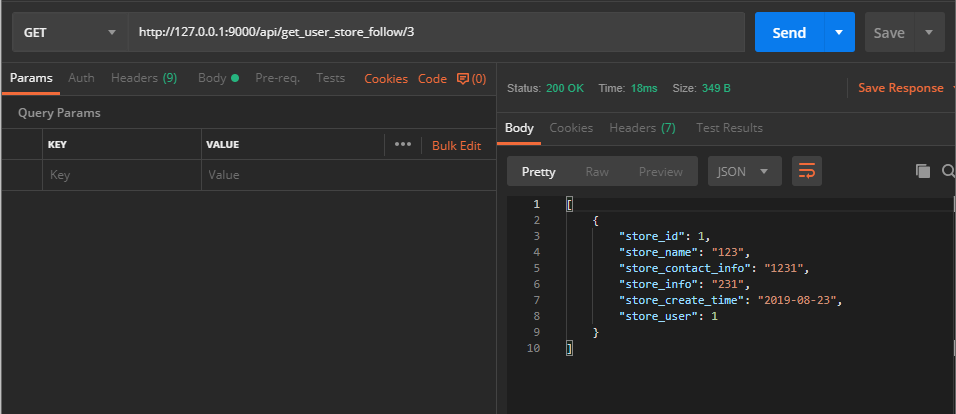
**根据用户user\_id获取该用户关注的所有商店的信息**

接口：http://127.0.0.1:8000/api/get\_user\_store\_follow/pk

网址最后的参数pk是用户的user\_id。此接口只允许GET请求。

示例：获取用户id为3的关注商店信息

http://127.0.0.1:8000/api/get\_user\_store\_follow/3



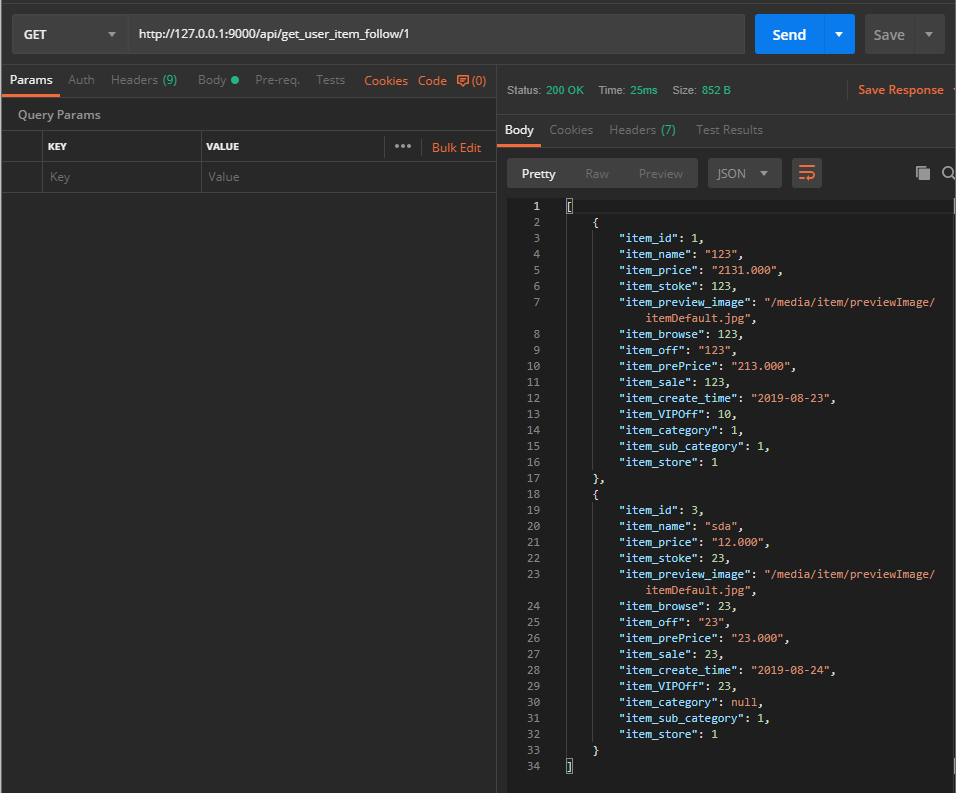
**根据用户user\_id获取该用户关注的所有商品的信息**

接口：http://127.0.0.1:8000/api/get\_user\_item\_follow/pk

网址最后的参数pk是用户的user\_id。此接口只允许GET请求。

示例：获取用户id为1的关注商品信息

http://127.0.0.1:8000/api/get\_user\_item\_follow/1



**根据item\_id，添加该商品的详情图或返回该商品的详情图路径**

接口：http://127.0.0.1:8000/api/item\_detail\_image/pk

网址最后的参数pk是商品的item\_id。此接口允许GET和POST请求。

**上传图片和查询图片使用同一个接口，支持同时上传多图。**

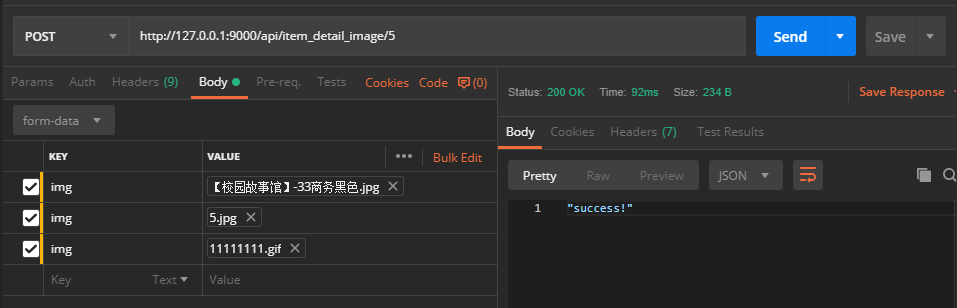
**上传图片发POST请求，查询图片发GET请求。**

**上传图片时POST参数name需设为img，这样接口才能取到文件数据。**

**后端获取文件列表，根据前端name参数来获取。**

示例：为id为5的商品上传多张图片

http://127.0.0.1:8000/api/item\_detail\_image/5

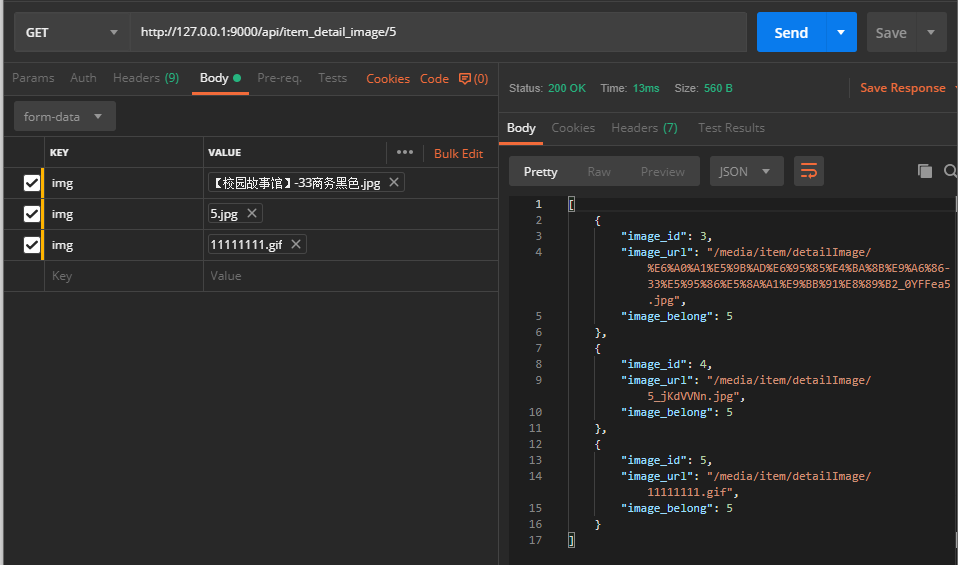


**注意前端POST请求的name参数一定要设置成img，不然出错。**

**上传成功返回success**

示例：查询id为5的商品的所有详情图

http://127.0.0.1:8000/api/item\_detail\_image/5



**上传查询网址都一样，只是请求方法不同。**

**根据item\_id，添加该商品的轮播图或返回该商品的轮播图路径**

接口：http://127.0.0.1:8000/api/item\_banner\_image/pk

网址最后的参数pk是商品的item\_id。此接口允许GET和POST请求。

**上传图片和查询图片使用同一个接口，支持同时上传多图。**

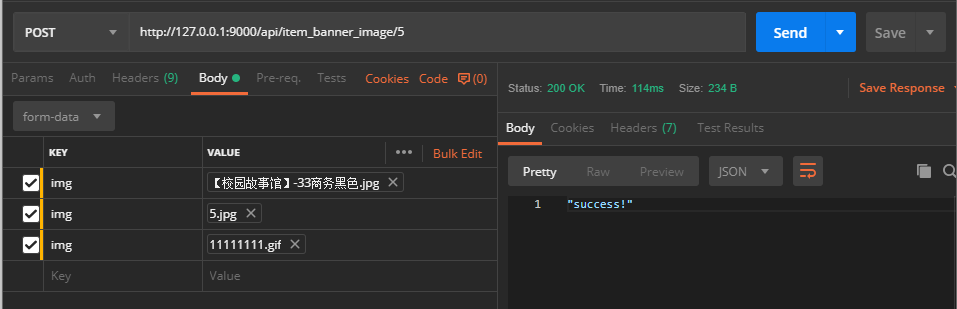
**上传图片发POST请求，查询图片发GET请求。**

**上传图片时POST参数name需设为img，这样接口才能取到文件数据。**

**后端获取文件列表，根据前端name参数来获取。**

示例：为id为5的商品上传多张图片

http://127.0.0.1:8000/api/item\_banner\_image/5



**注意前端POST请求的name参数一定要设置成img，不然出错。**

**上传成功返回success**

示例：查询id为5的商品的所有轮播图

http://127.0.0.1:8000/api/item\_banner\_image/5

**根据评价id，添加该评价的多张图片或返回该评价的图片路径**

接口：http://127.0.0.1:8000/api/evaluate\_image/pk

网址最后的参数pk是评价的evaluate\_id。此接口允许GET和POST请求。

**上传图片和查询图片使用同一个接口，支持同时上传多图。**

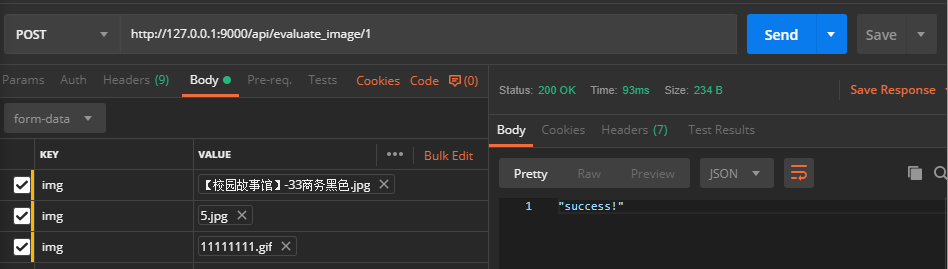
**上传图片发POST请求，查询图片发GET请求。**

**上传图片时POST参数name需设为img，这样接口才能取到文件数据。**

**后端获取文件列表，根据前端name参数来获取。**

示例：为id为1的评价上传多张图片

http://127.0.0.1:8000/api/evaluate\_image/1

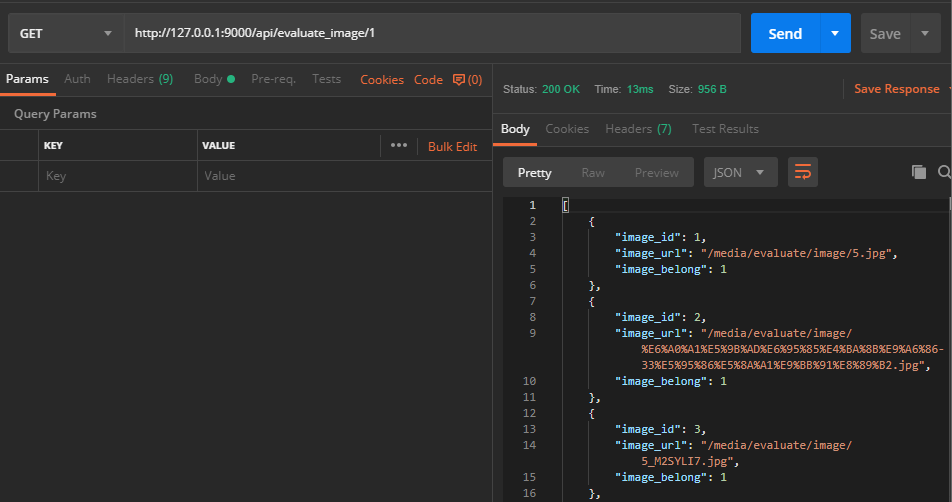


**注意前端POST请求的name参数一定要设置成img，不然出错。**

**上传成功返回success**

示例：查询id为1的评价的所有评价图

http://127.0.0.1:8000/api/evaluate\_image/1



**上传查询网址都一样，只是请求方法不同。**

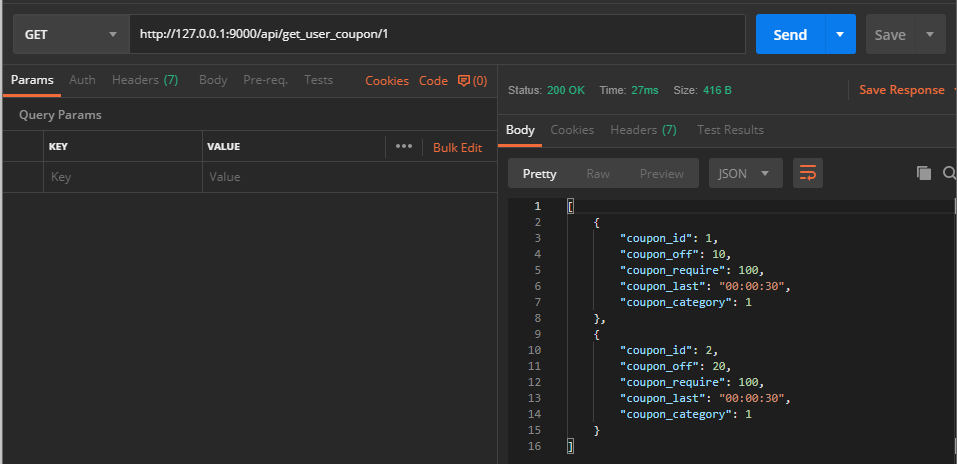
**根据user\_id，返回该用户拥有的优惠券信息**

接口：http://127.0.0.1:8000/api/get\_user\_coupon/pk

网址最后的参数pk是用户的user\_id。此接口只允许GET请求。

示例：获取用户id为1的所有优惠券信息：

http://127.0.0.1:8000/api/get\_user\_coupon/1



**根据user\_id、item\_id删除用户收藏夹的信息**

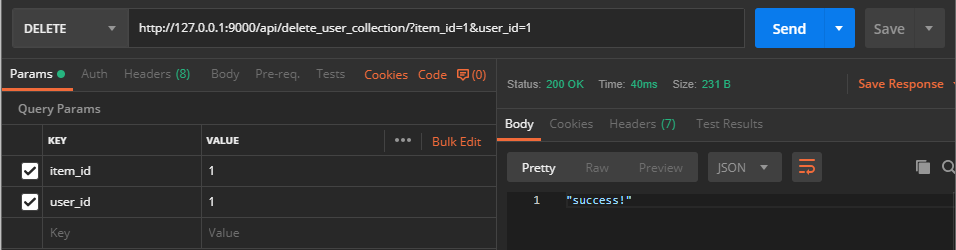
接口：http://127.0.0.1:8000/api/delete\_user\_collection/?item\_id=x&user\_id=x

访问接口的基本格式如上，只支持DELETE方法。

Item\_id和user\_id的位置可以任意互换，只需保证变量名对应就好。

示例：删除用户id为1，商品id为1的收藏夹信息：

http://127.0.0.1:8000/api/delete\_user\_collection/?item\_id=1&user\_id=1



**成功返回success**

**根据用户id和商品id判断该用户是否收藏了该商品**

接口：

http://127.0.0.1:8000/api/whether\_user\_collect\_item/?item\_id=x&user\_id=x

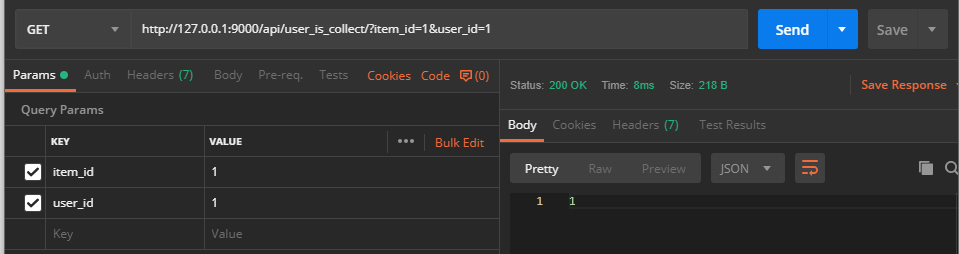
基本请求格式如上所示，参数顺序可随意，但参数名一定要匹配。

此接口只允许GET请求。

示例：

**判断用户id为1的是否收藏了商品id为1的**

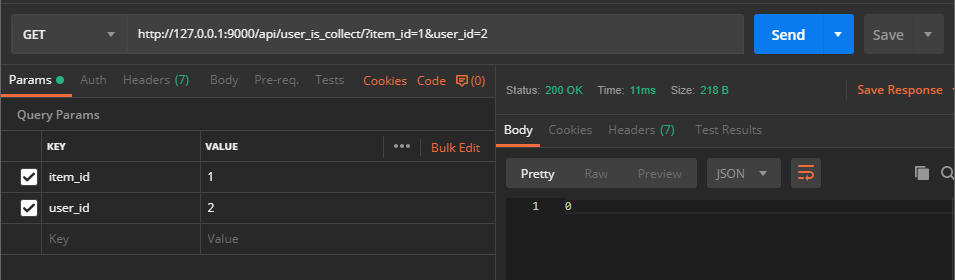
http://127.0.0.1:8000/api/whether\_user\_collect\_item/?item\_id=1&user\_id=1



**返回数字1表示收藏了**

**判断用户id为2的是否收藏了商品id为1的**

http://127.0.0.1:8000/api/whether\_user\_collect\_item/?item\_id=1&user\_id=2



**返回数字0表示没收藏**

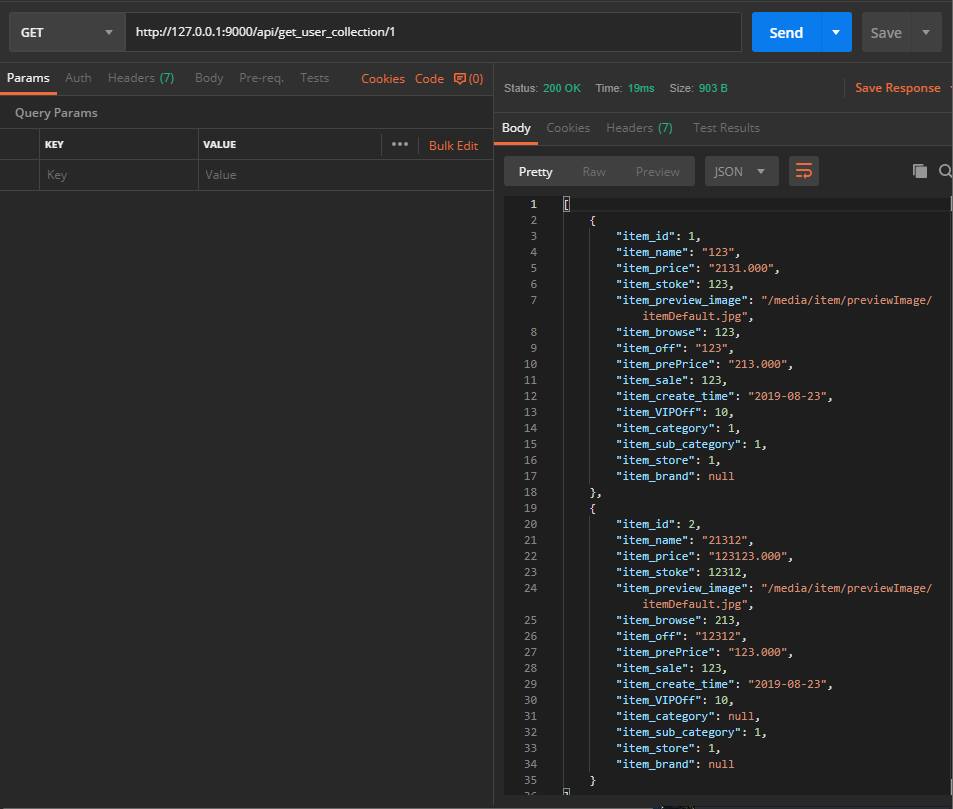
**根据user\_id返回用户收藏夹的所有商品信息**

接口：http://127.0.0.1:8000/api/get\_user\_collection/pk

网址最后的参数pk是用户的user\_id。此接口只允许GET请求。

示例：返回用户id为1的收藏夹商品全部信息：

http://127.0.0.1:8000/api/get\_user\_collection/1



**根据user\_id、item\_id，当前端浏览商品时自动更新浏览记录并合并记录、自动给商品浏览量+1、并返回该商品的信息**

接口：http://127.0.0.1:8000/api/browse\_item/?user\_id=1&item\_id=2

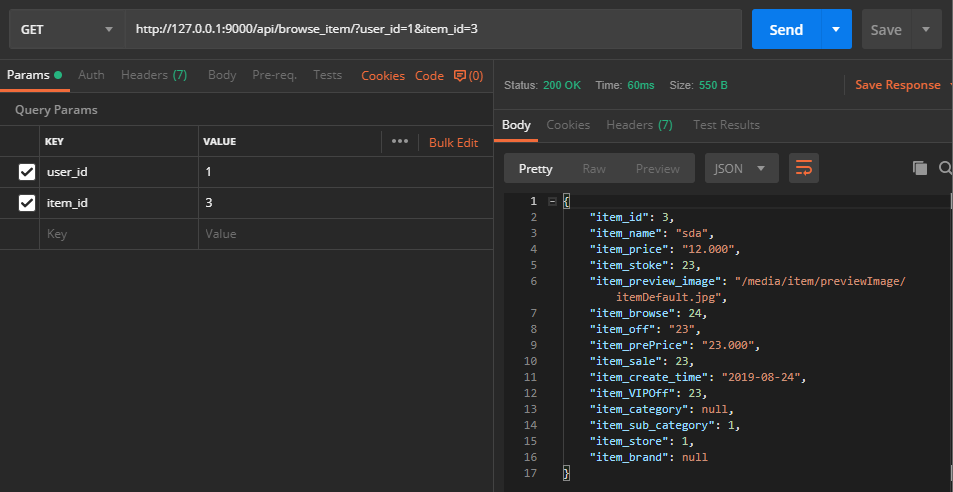
基本请求格式如上所示，各个参数可以乱序。

此接口只允许GET请求。

示例：以id为1的用户浏览商品id为3为例

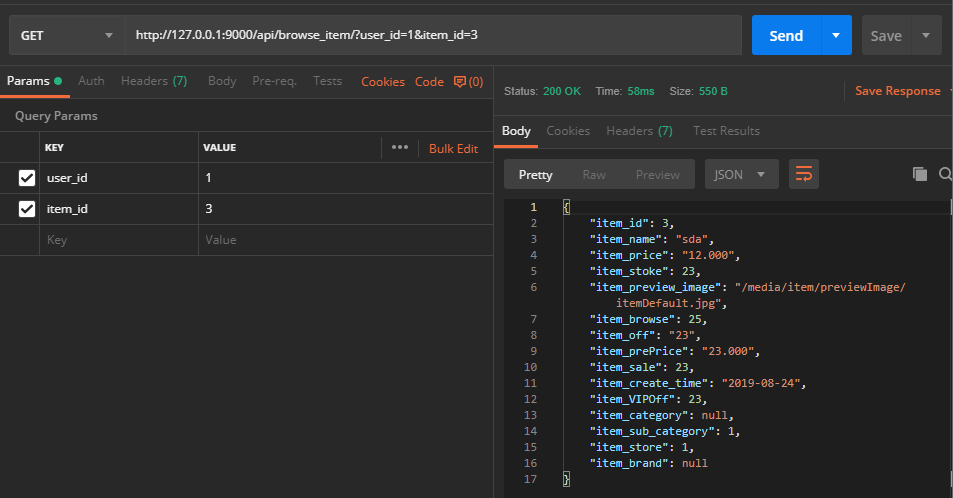
**浏览一个曾经未浏览过的商品：**

http://127.0.0.1:8000/api/browse\_item/?user\_id=1&item\_id=3



**浏览一个曾经浏览过的商品：**

http://127.0.0.1:8000/api/browse\_item/?user\_id=1&item\_id=3



**每次访问某商品，只要向这个接口发送请求，record\_create\_time就会自动更新到当前请求的时间，前端就可以利用原生接口提供的排序功能来构造历史记录列表。**

**根据用户id和搜索内容更新搜索记录并合并**

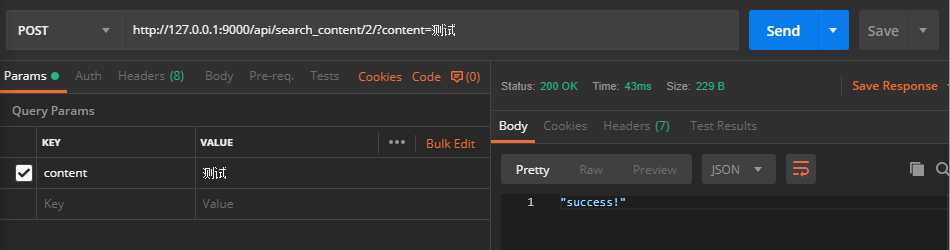
接口：http://127.0.0.1:8000/api/search\_content/pk/?content=xx

网址中的pk表示用户id。此接口只允许POST请求。

示例：用户id为2的人搜索内容为 测试 的字符串：

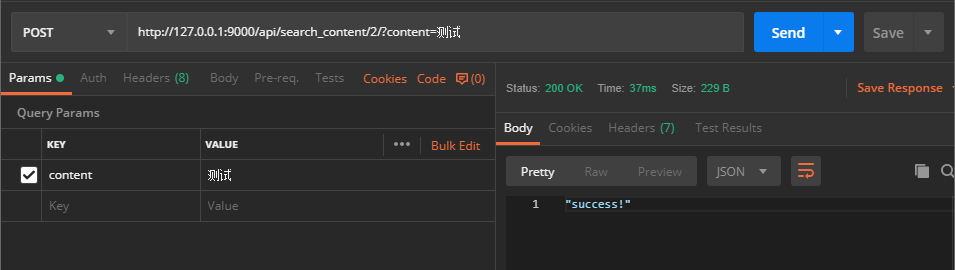
**第一次搜索：**

http://127.0.0.1:8000/api/search\_content/2/?content=测试



**隔一段时间，第二次再搜索同样的内容：**

http://127.0.0.1:8000/api/search\_content/2/?content=测试



**每次访问某商品，只要向这个接口发送请求，record\_create\_time就会自动更新到当前请求的时间，前端就可以利用原生接口提供的排序功能来构造历史记录列表。**

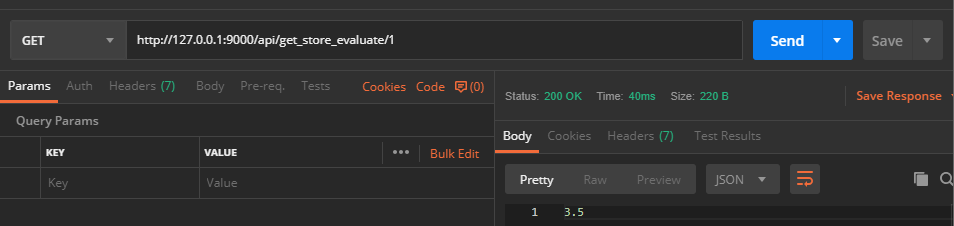
**根据商店id返回该商店的综合评价信息并更新该商店store\_evaluate字段**

接口：http://127.0.0.1:8000/api/get\_store\_evaluate/pk

网址最后的参数pk是用户的user\_id。此接口只允许GET请求。

示例：获取商店id为1的商店综合评价

http://127.0.0.1:8000/api/get\_store\_evaluate/1



**返回一位小数并自动更新store\_evaluate字段**

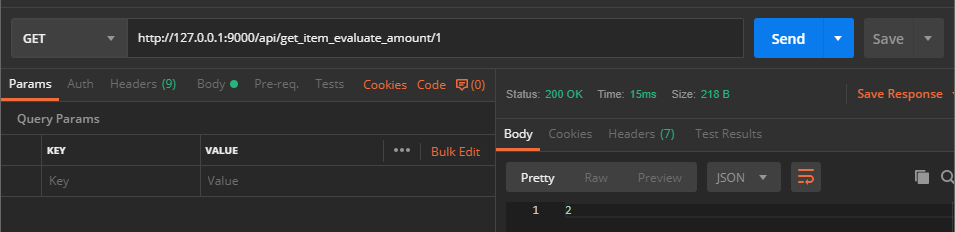
**根据商品id返回该商品有多少条评价信息**

接口：http://127.0.0.1:8000/api/get\_item\_evaluate\_amount/pk

网址最后的参数pk是商品的item\_id。此接口只允许GET请求。

示例：获取商品id为1的评价数

http://127.0.0.1:8000/api/get\_item\_evaluate\_amount/1



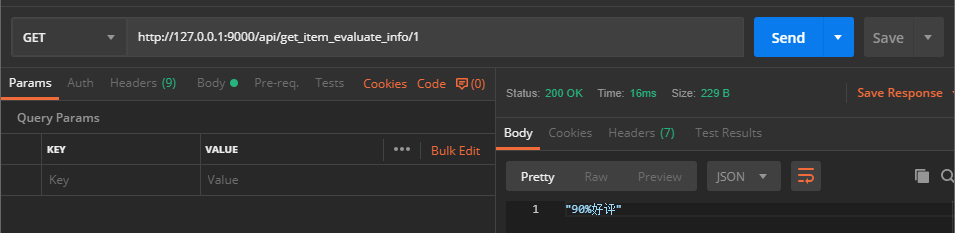
**根据商品id返回该商品的综合评价信息。默认以好评率字符串返回，例如：98%好评**

接口：http://127.0.0.1:8000/api/get\_item\_evaluate\_info/pk

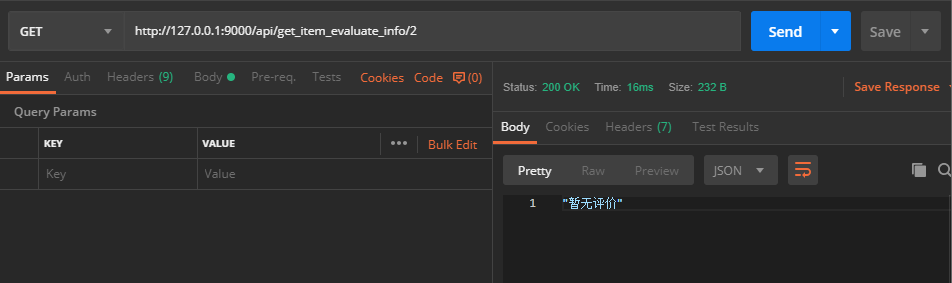
网址最后的参数pk是商品的item\_id。此接口只允许GET请求。

示例：获取id为1的商品的综合评价

http://127.0.0.1:8000/api/get\_item\_evaluate\_info/1



**若该商品未有评价信息，则返回“暂无评价”**



**根据用户id和商店id返回该用户是否在该店购买过商品**

接口：

http://127.0.0.1:8000/api/whether\_user\_buy\_item\_in\_store/?user\_id=x&store\_id=x

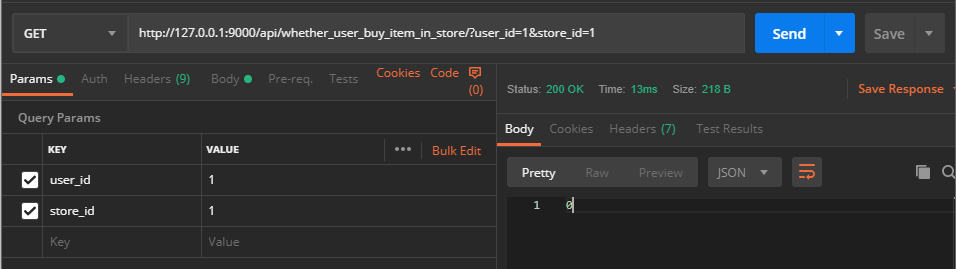
基本请求格式如上，参数可乱序但参数名一定要对应，只允许GET

**返回0代表未买过，返回1代表买过**

**此接口主要根据Trade表中是否有数据以及trade\_info进行判断，当trade表中有数据且有至少一条记录的trade\_info为3（3代表已完成交易）时，返回1。否则返回0**

示例：用户1是否在商店1买过商品

http://127.0.0.1:8000/api/whether\_user\_buy\_item\_in\_store/?user\_id=1&store\_id=1



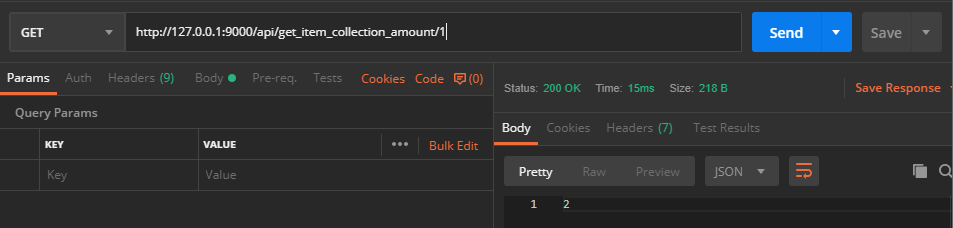
**根据商品id返回该商品被多少人收藏**

接口：http://127.0.0.1:8000/api/get\_item\_collection\_amount/pk

网址最后的参数pk是商品的item\_id。此接口只允许GET请求。

示例：获取id为1的商品收藏数

http://127.0.0.1:8000/api/get\_item\_collection\_amount/1



**根据商品id返回“推荐商品”的详细信息（智能推荐算法）**

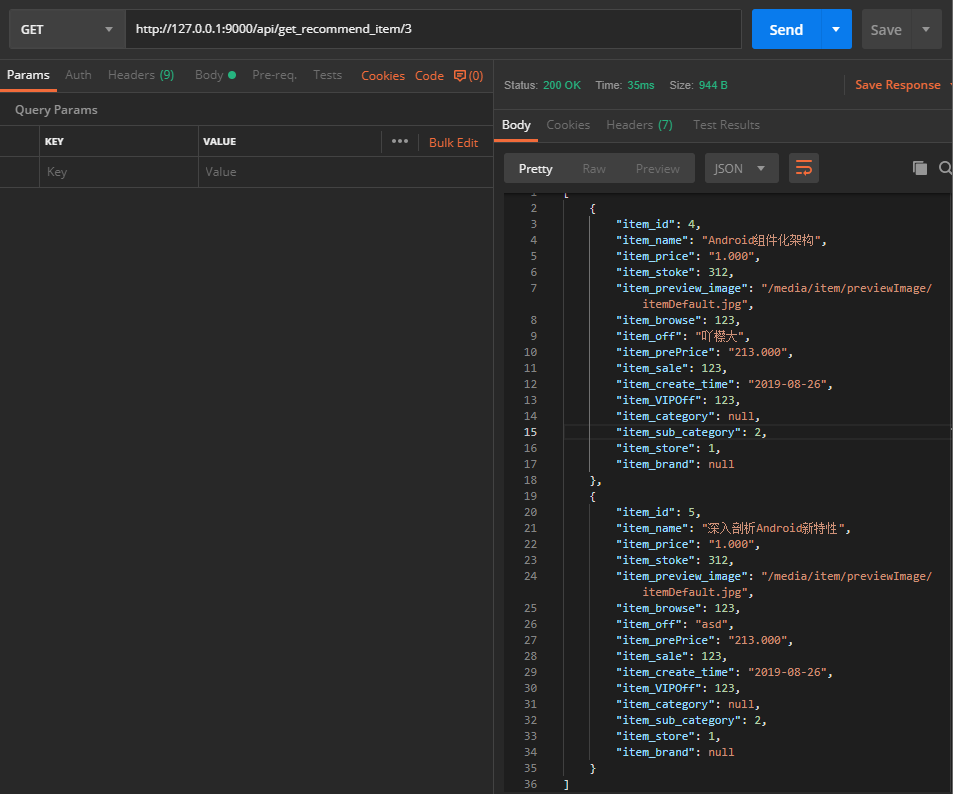
接口：http://127.0.0.1:8000/api/get\_recommend\_item/pk

网址最后的参数pk是商品的item\_id。此接口只允许GET请求。

**后端内部是先筛选出与商品属于同一子类的所有商品，然后根据商品名字的相似度进行推荐**

示例：获取id为3的商品的推荐商品

http://127.0.0.1:8000/api/get\_recommend\_item/3



**根据商品id返回该商品的好评数、中评数、差评数和有图的评价数并返回评价信息**

接口：

http://127.0.0.1:8000/api/get\_evaluate\_type/?item\_id=x&evaluate\_type=xxxx&return\_type=xxxx

基本请求格式如上所示，参数可以乱序但参数名一定要对应。

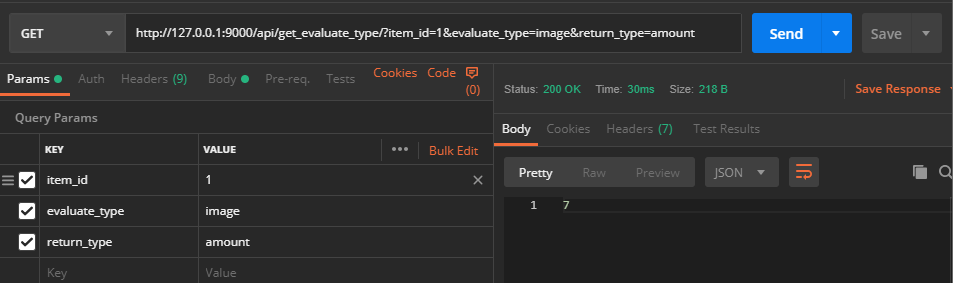
此接口只允许GET

**Evaluate\_type表示评价类型，good代表好评、normal代表中评、bad代表差评、image代表有图**

**Return\_type代表返回类型，amount表示返回数字、info表示返回具体信息**

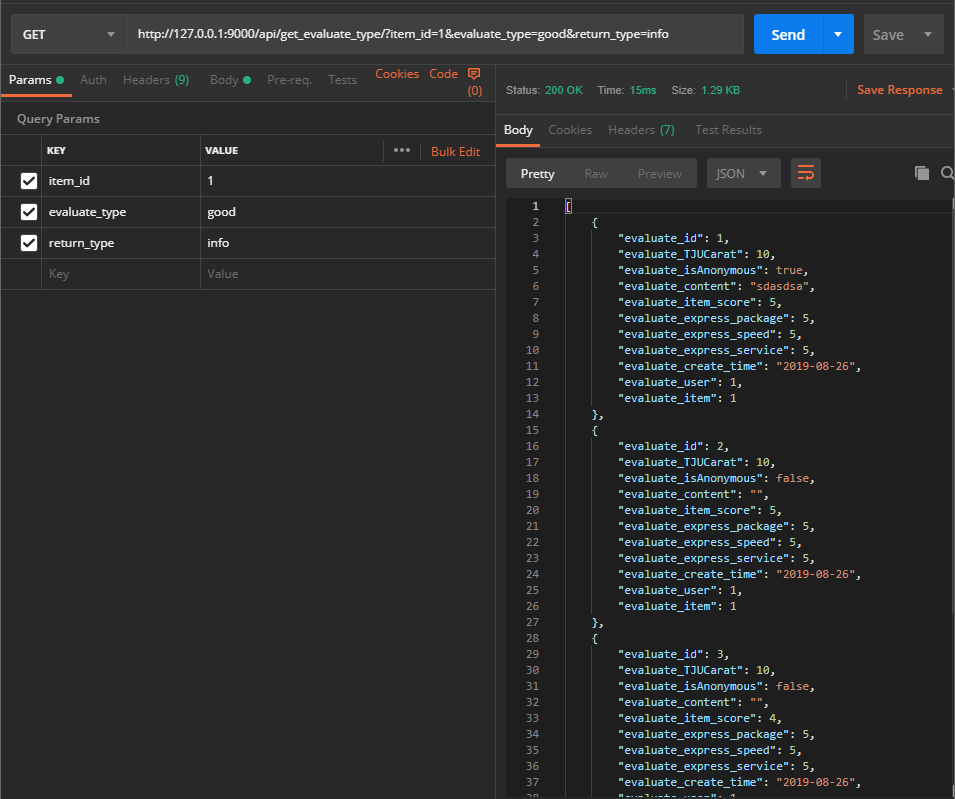
示例：获取id为1的商品的有图评价数

http://127.0.0.1:8000/api/get\_evaluate\_type/?item\_id=1&evaluate\_type=image&return\_type=amount



示例：获取id为1的商品的好评信息

http://127.0.0.1:8000/api/get\_evaluate\_type/?item\_id=1&evaluate\_type=good&return\_type=info



**根据用户id返回待收货的商品信息以及数量**

接口：http://127.0.0.1:8000/api/get\_wait\_receive/?user\_id=x&return\_type=xxx

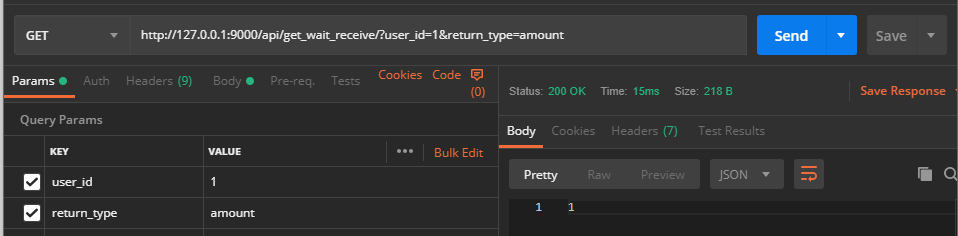
请求格式如上所示，参数可以乱序但参数名一定要对应

此接口只允许GET

**Return\_type表示返回类型，info代表返回商品详情、amount返回数量**

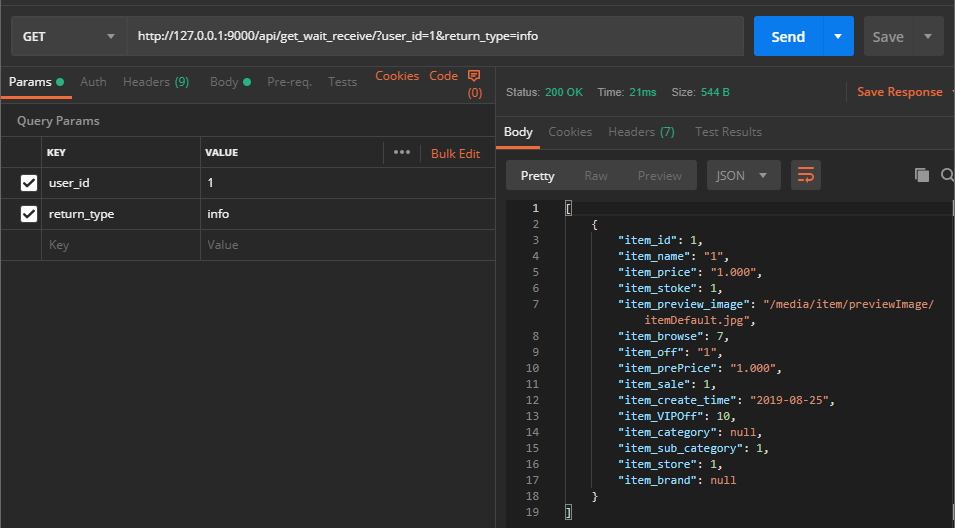
示例：返回id为1的用户的待收货数量

http://127.0.0.1:8000/api/get\_wait\_receive/?user\_id=1&return\_type=amount



示例：返回id为1的用户的待收货的商品详情

http://127.0.0.1:8000/api/get\_wait\_receive/?user\_id=1&return\_type=info



**根据用户id返回待评价的商品信息以及数量**

接口：http://127.0.0.1:8000/api/get\_wait\_evaluate/?user\_id=x&return\_type=xxxx

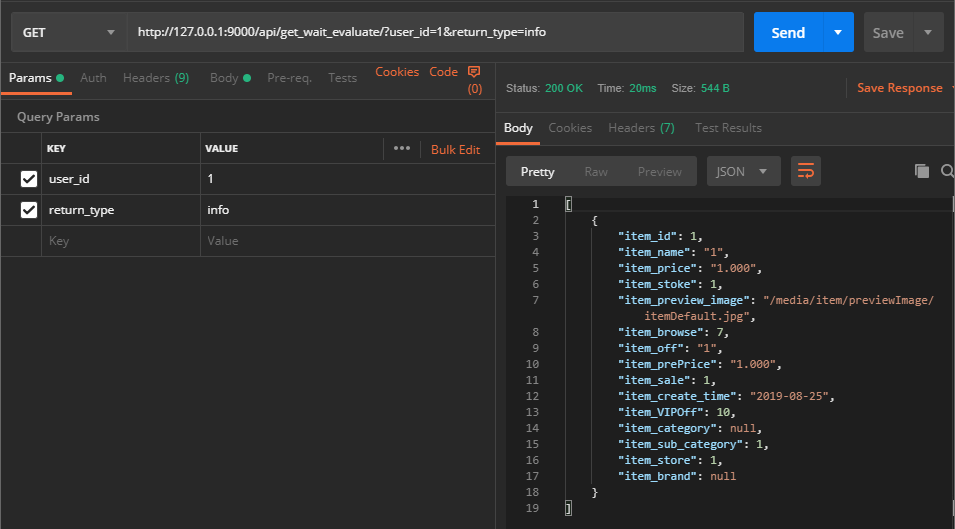
请求格式如上所示，参数可以乱序但参数名一定要对应

接口只允许GET

**Return\_type为返回类型，amount表示返回数量，info表示返回商品详情**

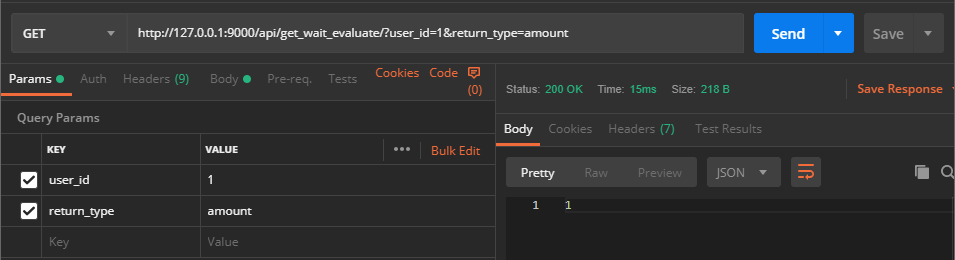
示例：返回id为1的用户的待评价商品信息

http://127.0.0.1:8000/api/get\_wait\_evaluate/?user\_id=1&return\_type=info



示例：返回id为1的用户的待评价数

http://127.0.0.1:8000/api/get\_wait\_evaluate/?user\_id=1&return\_type=amount



**根据用户id返回该用户所有已完成的订单数及商品详情**

接口：

http://127.0.0.1:8000/api/get\_complete\_trade/?user\_id=x&return\_type=xxxx

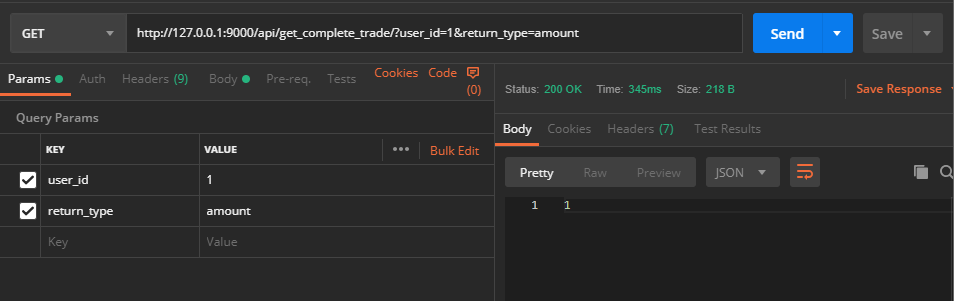
请求格式如上所示，参数可以乱序但参数名一定要对应

接口只允许GET

**Return\_type为返回类型，amount表示返回数量，info表示返回商品详情**

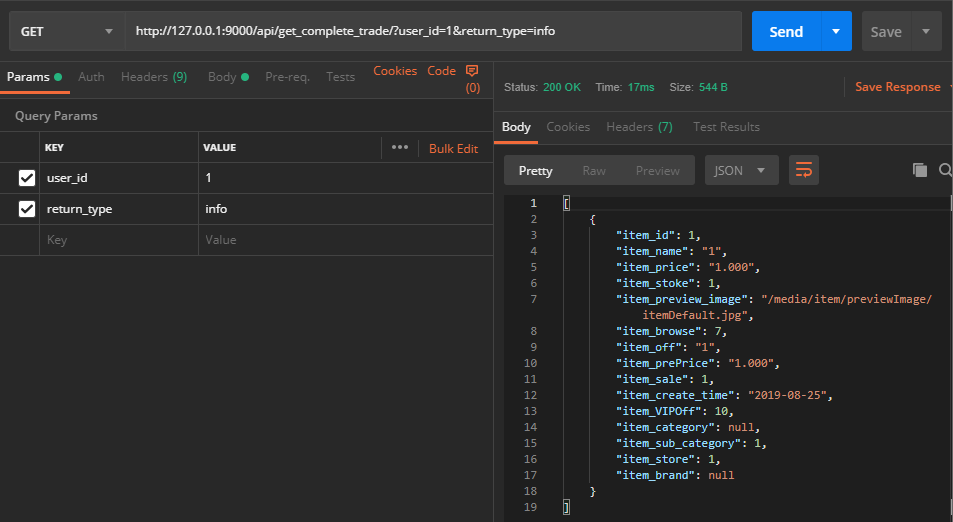
示例：返回id为1的用户的所有已完成的订单数

http://127.0.0.1:8000/api/get\_complete\_trade/?user\_id=1&return\_type=amount



示例：返回id为1的用户的所有已完成的订单的商品详情

http://127.0.0.1:8000/api/get\_complete\_trade/?user\_id=1&return\_type=info



**根据用户id、商品id、商店id、购买数量，处理前端点击“立即购买”后的逻辑**

接口：

http://127.0.0.1:8000/api/buy\_now/?user\_id=x&item\_id=x&store\_id=x&amount=xx

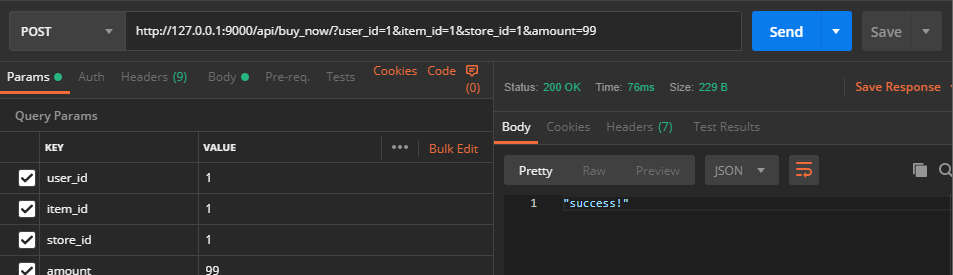
请求格式如上，参数可以乱序但参数名必须对应。

**该接口只允许POST。**

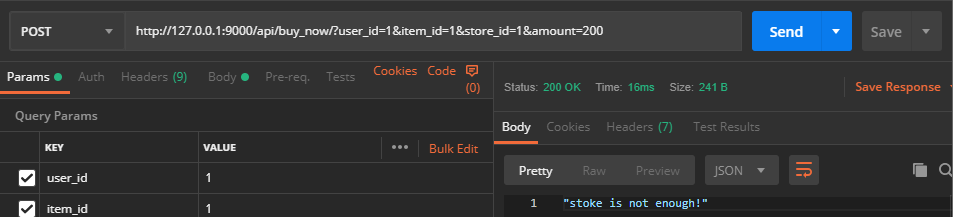
**后端封装的功能：先检测商品的库存够不够，不够的话直接返回字符串“stoke is not enough”。若足够，则给商品的库存减去购买量，销量加上购买量。然后在Trade中加入新记录，trade\_info置0，表示未发货。上述过程成功后返回success。**

示例：在商店1中立即购买商品1 99件

http://127.0.0.1:8000/api/buy\_now/?user\_id=1&item\_id=1&store\_id=1&amount=99



再买一次200件：



**根据用户id、商品id、购买数量，处理前端点击加入购物车的逻辑**

接口：

http://127.0.0.1:8000/api/add\_into\_cart/?user\_id=x&item\_id=x&amount=xx

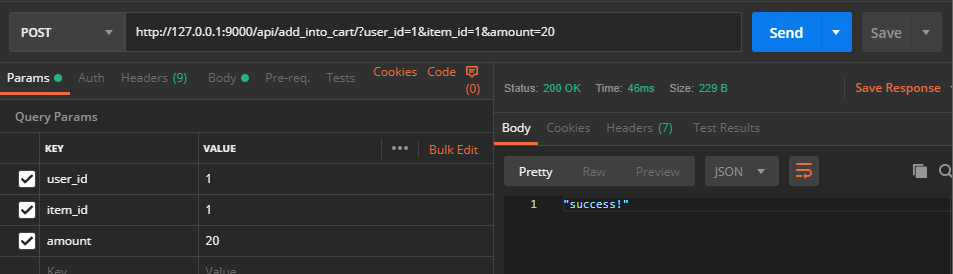
请求格式如上，参数可以乱序但参数名必须对应。

**该接口只允许POST。**

**后端封装的功能：先检测商品的库存够不够，不够的话直接返回字符串“stoke is not enough”。若足够，在购物车中加记录（不会减该商品的库存）。上述成功后返回success。**

示例：用户1对商品1立即加入购物车20件

http://127.0.0.1:8000/api/add\_into\_cart/?user\_id=1&item\_id=1&amount=20



**根据用户id、商品id、商店id、购买数量，处理前端在购物车页面进行购买的逻辑**

接口

http://127.0.0.1:8000/api/buy\_in\_cart/?user\_id=x&item\_id=x&store\_id=x&amount=xx

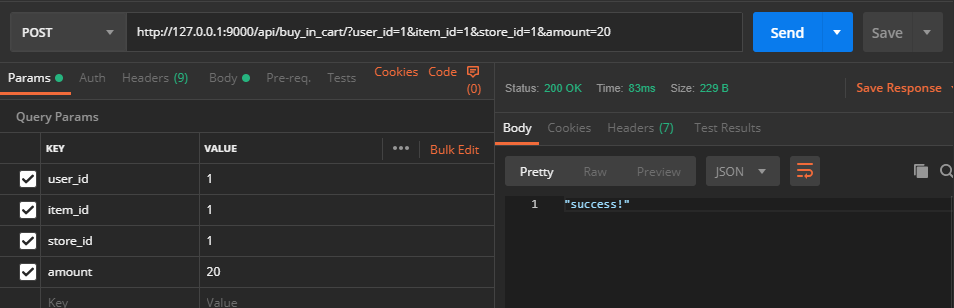
请求格式如上，参数可以乱序但参数名必须对应。

**该接口只允许POST。**

**后端封装的功能：先检测商品的库存够不够，不够的话直接返回字符串“stoke is not enough”。若足够，则给商品的库存减去购买量，销量加上购买量。然后在Trade中加入新记录，trade\_info置0，表示未发货。购物车删除对应记录。上述过程成功后返回success。**

示例：用户1对商品1商店1在购物车下单20件

http://127.0.0.1:8000/api/buy\_in\_cart/?user\_id=1&item\_id=1&store\_id=1&amount=20



**类似的搜索模式以及写法具体参照api网站**