# 一、实验目的

1.了解文档类数据库，并且能够做出合理的数据库设计

2.熟练使用mongoDB数据库和pymongo的增删改查

# 二、实验内容

考虑以下游戏场景：

1. 每个游戏玩家都有一定数量的金币、宝物。有一个市场供玩家们买卖宝物。玩家可以将宝物放到市场上挂牌，自己确定价格。其他玩家支付足够的金币，可购买宝物。
2. 宝物分为两类：一类为工具，它决定持有玩家的工作能力；一类为配饰，它决定持有玩家的运气。
3. 每位玩家每天可以通过寻宝获得一件宝物，宝物的价值由玩家的运气决定。每位玩家每天可以通过劳动赚取金币，赚得多少由玩家的工作能力决定。（游戏中的一天可以是现实中的1分钟、5分钟、10分钟。自主设定。）
4. 每个宝物都有一个自己的名字（尽量不重复）。每位玩家能够佩戴的宝物是有限的（比如一个玩家只能佩戴一个工具和两个配饰）。多余的宝物被放在存储箱中，不起作用，但可以拿到市场出售。
5. 在市场上挂牌的宝物必须在存储箱中并仍然在存储箱中，直到宝物被卖出。挂牌的宝物可以被收回，并以新的价格重新挂牌。当存储箱装不下时，运气或工作能力值最低的宝物将被系统自动回收。
6. 假设游戏永不停止而玩家的最终目的是获得最好的宝物。

请根据以上场景构建一个假想的Web游戏，可供多人在线上玩耍。界面尽可能简单（简单文字和链接即可，不需要style）。后台的数据库使用mongodb。对游戏玩家提供以下几种操作：寻宝（可以自动每天一次）、赚钱（可以自动每天一次）、佩戴宝物、浏览市场、买宝物、挂牌宝物、收回宝物。

# 三、实验环境

python 3.7

pycharm

flask框架 pytest框架

MongoDB

postman

# 四、数据库设计

在数据库中用到**三个集合**，存储玩家的数据，宝物的数据和每个玩家的session 记录

**玩家集合包括姓名、密码、金钱、工具、配饰、宝物箱（容量初始为3，最大容量是10）**

**宝物集合包括宝物名、能力值、运气值、是否可用（在宝物箱中为1，不在为0）、是否售卖（在市场上为1，不在为0）、所有者、价格**

**session集合包括用户名和session**

初始化的文档集合如下：

player：(

{ name: "Joe",

password: "123456",

money: 1000,

tool:"sword",

jewelry:"necklace",

treasure: ["dagger", "basketball", "magicbook"]

},

{ name: "Kate",

password: "123",

money: 500,

tool:"shovel",

jewelry:"earrings",

treasure: ["bag", "computer", "bracelet"]

} )

treasure：(

{ name: "dagger",  ability: 80, luck: 0, available:1, onsale:0, player:"Joe", "price":80},

{ name: "basketball", ability: 30, luck: 0, available:1, onsale:1, player:"Joe", "price":30}, { name: "magicbook",  ability: 100, luck: 0, available:1, onsale:0, player:"Joe", "price":100},

{ name: "shovel",  ability: 90, luck: 0, available:0, onsale:0, player:"Kate", "price":90},

{ name: "earrings",  ability: 0, luck: 30, available:0, onsale:0, player:"Kate", "price":30},

{ name: "bag",  ability: 0, luck: 10, available:1, onsale:1, player:"Kate", "price":10},

{ name: "computer",  ability: 99, luck: 0, available:1, onsale:1, player:"Kate", "price":99},

{ name: "bracelet",  ability: 0, luck: 20, available:1, onsale:1, player:"Kate", "price":20}, { name: "sword",  ability: 70, luck: 0, available:0, onsale:0, player:"Joe", "price":70},

{ name: "necklace", ability: 0, luck: 50, available:0, onsale:0, player:"Joe", "price":50}

)

session(

{username:"Joe",session:"eyJwYXNzd29yZCI6IjEyMzQ1NiIsInVzZXJuYW1lIjoiSm9lIn0.X4\_qzQ.JRW4eHJnmpMzoR\_2HJ3c9ylMkqI" })

# 五、功能实现

为实现游戏，功能模块有八个，分别是**注册、登陆、玩家每天自动工作赚钱、玩家每天自动寻宝、浏览市场并买宝物、****挂牌宝物或收回宝物、****更换佩戴工具、更换佩戴配饰**。

下面是八个功能模块的简要概述：

(1) **登陆**：在浏览器输入http://127.0.0.1:8081/时，返回的是登陆页面，这时输入用户名和密码，相当于post {“username”:XX, “password”:XXXXXX}。拿到request的用户名和密码后，使用find\_one()函数查找数据库player集合并比较，如果用户名和密码匹配了，进入个人的home页面，记录session；如果用户名正确密码错误，重新进入登陆页面；如果用户名不存在，进入注册页面。

(2) **注册**：在http://127.0.0.1:8081/register下进行注册操作。用户post {“username”:XX, “password”:XXXXXX}, 后端生成字典dict = {"name":username, "password":password, "money":500, "tool":None, "jewelry":None, "treasure":["bag","bracelet","basketball"]}，这是新用户的初始状态，将其insert\_one进数据库的player集合。

(3) **玩家每天自动工作赚钱**：在http://127.0.0.1:8081/<string:username>/work下，每个玩家每隔300秒自动赚钱一次。定时程序写在视图函数外，另开一个线程。视图函数内，如果work\_flag==1，表示今天已经赚过钱了，返回jsonify(“ok”:0)；如果work\_flag==0，将其设置为1，查找该用户佩戴的工具的价值，由于赚钱多少由玩家的工作能力决定，这里设置赚得的金钱为tool的ability，即toolon = myplayer.find\_one({"name":username})["tool"]，workability = mytreasure.find\_one({"name":toolon})["ability"]。最后，更新玩家的金钱myplayer.update({"name": username}, {"$set":{"money": money}})。函数的返回值是这一系列操作后用户的各种状态。

(4) **玩家每天自动寻宝**：在http://127.0.0.1:8081/<string:username>/find下，每个玩家每隔300秒自动寻宝一次。定时程序写在视图函数外，另开一个线程。视图函数内，如果find\_flag==1，表示今天已经寻过宝了，返回jsonify(“ok”:0)；如果find\_flag==0，将其设置为1，查找该用户佩戴的配饰的运气，因为寻到宝物的价值由玩家的运气决定。设置一个将宝物按价值从高到底排序的全局列表list1 = ["magicbook", "computer", "shovel", "dagger", "sword", "necklace", "basketball", "earrings", "bracelet", "bag"]。这里另写了一个random\_pick函数，以不同的概率随机在list1中取值，最好的宝物”magicbook”被random\_pick到的概率最小，为0.01，剩下的被选择到的概率根据玩家的运气不同而不同。将用户寻到的宝物放入宝物箱myplayer.update({"name":username}, {"$push":{"treasure":get}})，并将这个新的宝物的数据插入treasure集合，dict = {"name":get, "ability":ability\_, "luck":luck\_, "available":1, "onsale":0, "player":username, "price":price\_}，mytreasure.insert\_one(dict)，注意这里新宝物和原来就在数据库中的宝物只有“\_id”和”player”字段不同。接下来注意控制宝物箱中宝物的数量，如果超过了10个，选择价值最小的宝物从player的treasure中$pull出来，并设置treasure集合中这个宝物的拥有者为root。函数的返回值是这一系列操作后用户的各种状态。

(5) **浏览市场并买宝物**：在http://127.0.0.1:8081/<string:username>/market下，如果request == “GET”, 查找数据库中所有{“onsale”:1}的宝物，返回这些宝物的名称和价格。如果request == “POST”，用户post的内容是想要购买的宝物的名称。如果查询数据库后宝物的onsale不是1，不能购买(注：在我的数据库设计中，只有在宝物箱中的宝物才能被出售，即若available为0，onsale不可能为1)。若购买宝物操作发生，更新数据库，将宝物从卖家的treasure中$pull出来，$push进买家的数据库，买卖双方金钱都减少增加相应的值，并更新宝物在treasure集合中的“所有者”字段，具体操作见代码。接下来注意控制买家宝物箱中宝物的数量，如果超过了10个，选择价值最小的宝物从player的treasure中$pull出来，并设置treasure集合中这个宝物的拥有者为root。root是所有被丢弃的宝物的拥有者，玩家可以付出相应的金钱购买”root”的宝物，但root在player中并不存在。函数的返回值是这一系列操作后用户即买家的状态信息。

(6) **挂牌宝物或收回宝物**：在http://127.0.0.1:8081/<string:username>/sell下，如果request == “GET”，查找当前用户的数据状态并返回；如果request == “POST”，用户post的内容是{“item”:宝物的名称，”task”:任务类型，“price”:宝物价格}。如果task == 1 ,表示挂牌宝物，在treasure中查找到一个宝物名称相符，”available”为1且拥有者为session[“username”]的文档（只有在宝物箱中的宝物才能出售），修改其”onsale”字段为1，”price”字段为post的值。

如果task == 0 ,表示收回宝物，在treasure中查找到一个宝物名称相符，”onsale”为1且拥有者为session[“username”]的文档，修改其”onsale”字段为0，post的”price”值不管。最后，查找到该用户所有正在出售的宝物，返回宝物名和价格的json数据结构。

(7) **更换佩戴工具**：在http://127.0.0.1:8081/<string:username>/changetool下，如果request == “GET”，查找当前用户的数据状态并返回；如果request == “POST”，用户post的内容是想要更换的工具的名称。先查找到该宝物，确定它是工具（工具的运气值为0，配饰的能力值为0）且在该用户的宝物箱中，否则返回jsonify({“ok”:0})。查找到用户之前佩戴的工具，记为old，如果old不是None, 将tool设为该用户的tool，”available”置为0并且从宝物箱中$pull出来，将old的”available”置为1”onsale”置为0并且$push进宝物箱。如果old是None，不需要操作old。函数的返回值是这一系列操作后用户的状态信息。

(8) **更换佩戴配饰**：在http://127.0.0.1:8081/<string:username>/changejewelry下，如果request == “GET”，查找当前用户的数据状态并返回；如果request == “POST”，用户post的内容是想要更换的配饰的名称。先查找到该宝物，确定它是配饰（工具的运气值为0，配饰的能力值为0）且在该用户的宝物箱中，否则返回jsonify({“ok”:0})。查找到用户之前佩戴的配饰，记为old，如果old不是None, 将jewelry设为该用户的jewelry，”available”置为0并且从宝物箱中$pull出来，将old的”available”置为1”onsale”置为0并且$push进宝物箱。如果old是None，不需要操作old。函数的返回值是这一系列操作后用户的状态信息。

# 六、每个功能涉及的CRUD操作

(1) **登陆**：

if mysession.find\_one({"username":session["username"]}) != None:  
 mysession.delete\_one({"username":session["username"]})  
mysession.insert\_one({"username":session["username"],"session":request.cookies["session"]})  
temp = myplayer.find\_one({"name":session["username"]})

(2) **注册**：

dict = {"name":username, "password":password, "money":500, "tool":None, "jewelry":None, "treasure":["bag","bracelet","basketball"]}  
myplayer.insert\_one(dict)

(3) **玩家每天自动工作赚钱**：

money = myplayer.find\_one({"name":username})["money"]  
toolon = myplayer.find\_one({"name":username})["tool"]  
workability = mytreasure.find\_one({"name":toolon})["ability"]  
money= money + workability  
myplayer.update\_one({"name": username}, {"$set":{"money": money}})

(4) **玩家每天自动寻宝**：

jewelry = myplayer.find\_one({"name":username})["jewelry"]  
luck = mytreasure.find\_one({"name":jewelry})["luck"]  
myplayer.update\_one({"name":username}, {"$push":{"treasure":get}})  
temp1 = mytreasure.find\_one({"name":get})  
ability\_ = temp1["ability"]  
luck\_ = temp1["luck"]  
price\_ = temp1["price"]  
dict = {"name":get, "ability":ability\_, "luck":luck\_, "available":1, "onsale":0, "player":username, "price":price\_}

mytreasure.insert\_one(dict)  
res = myplayer.find\_one({"name": username})  
cnt = len(res["treasure"])

if (cnt > 10): #宝物箱中的宝物大于10个，扔掉价值最小的

temp1 = mytreasure.find\_one({"player": username, "available": 1}, sort=[("price", 1)])  
 print(temp1)  
 myplayer.update\_one({"name": username}, {"$pull": {"treasure": temp1["name"]}})  
 mytreasure.update\_one({"player":username,"name": temp1["name"]}, {"$set": {"player": "root"}})

(5) **浏览市场并买宝物**：

res = mytreasure.find\_one({"name":item,"onsale":1})  
if(res == None):  
 return jsonify({"ok":0})  
elif(res["player"] == username):  
 return jsonify({"ok": 0})  
else:  
 seller = res["player"]  
 temp = res["price"]  
 mytreasure.update\_one({"name": item}, {"$set":{"player":buyer}})  
 myplayer.update\_one({"name":buyer}, {"$push":{"treasure":item}})  
 myplayer.update\_one({"name": buyer}, {"$inc": {"money": -temp}}) # 买家金钱减少  
 if seller != "root":  
 myplayer.update\_one({"name": seller}, {"$pull": {"treasure": item}})  
 myplayer.update\_one({"name": seller}, {"$inc": {"money": temp}}) # 卖家金钱增加  
  
 res = myplayer.find\_one({"name": buyer})

(6) **挂牌宝物或收回宝物**：

if task == 1: #挂牌宝物  
 res1 = mytreasure.find\_one({"name":item, "player":player, "available":1, "onsale":0}) #正在使用的宝物不能出售  
 print(res1)  
 if res1 != None:  
 mytreasure.update\_one({"name":item, "player":player}, {"$set":{"onsale":1}})  
 mytreasure.update\_one({"name": item, "player": player}, {"$set": {"price": price}})  
 else:  
 return jsonify({"ok":0})  
else: #收回宝物  
 res2 = mytreasure.find\_one({"name":item, "player":player, "onsale":1})  
 print(res2)  
 if res2 != None:  
 mytreasure.update\_one({"name":item, "player":player}, {"$set":{"onsale":0}})  
 else:  
 return jsonify({"ok":0})

(7) **更换佩戴工具**：

if mytreasure.find\_one({"name":tool, "player":player}) == None:  
 return jsonify({"ok":0})  
elif mytreasure.find\_one({"name":tool, "player":player})["ability"] == 0:  
 print("this is jewelry , cannot change")  
 return jsonify({"ok": 0})  
elif myplayer.find\_one({"name":player, "treasure":tool}) == None:  
 return jsonify({"ok": 0})  
else:  
 old = myplayer.find\_one({"name": player})["tool"]  
 myplayer.update\_one({"name": player}, {"$pull": {"treasure": tool}})  
 myplayer.update\_one({"name": player},{"$set":{"tool":tool}})  
 if old != None:  
 myplayer.update\_one({"name": player}, {"$push": {"treasure": old}})  
 mytreasure.update\_one({"name": old}, {"$set":{"available":1}})  
 mytreasure.update\_one({"name": tool}, {"$set": {"available": 0}, "$set":{"onsale":0}})

(8) **更换佩戴配饰**：

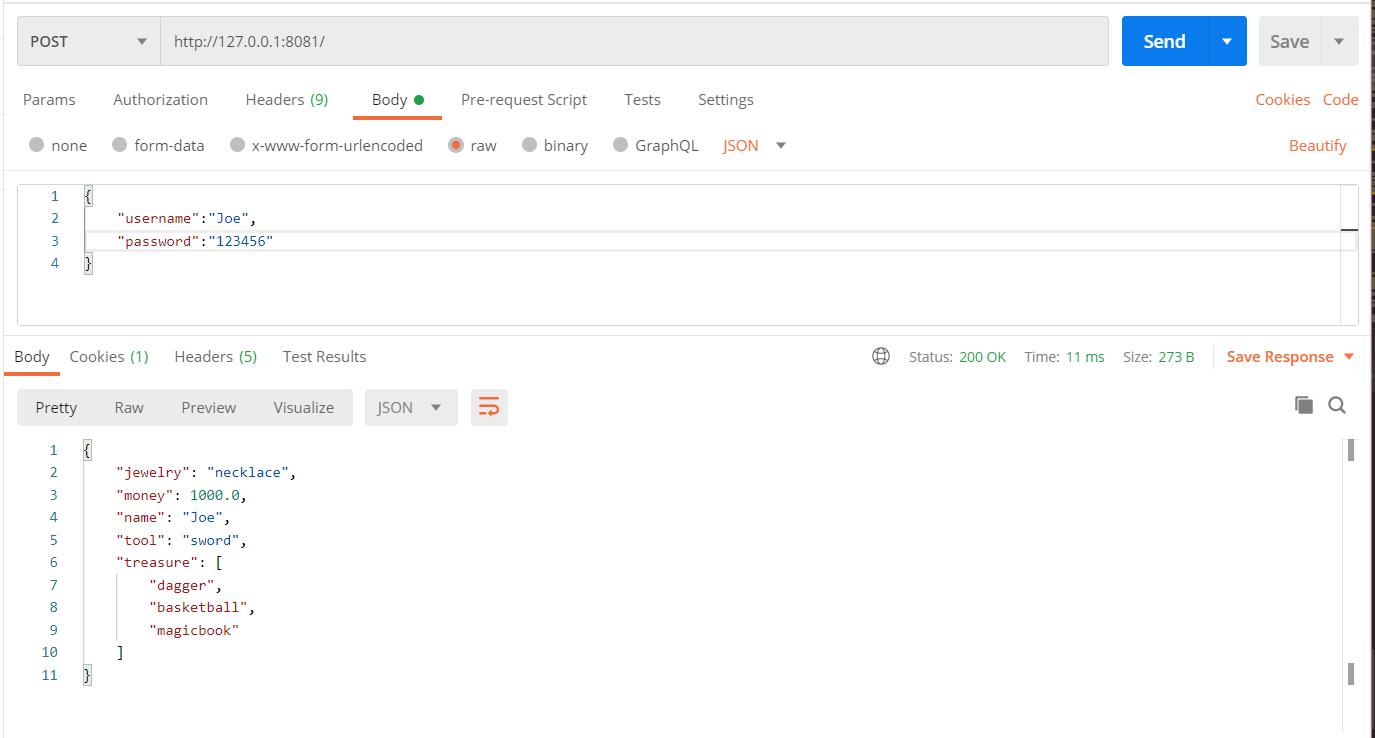
if mytreasure.find\_one({"name":jewelry, "player":player})["luck"] == 0:  
 print("this is tool , cannot change")  
 return jsonify({"ok": 0})  
elif myplayer.find\_one({"name":player, "treasure":jewelry}) == None:  
 return jsonify({"ok": 0})  
else :  
 old = myplayer.find\_one({"name": player})["jewelry"]  
 myplayer.update\_one({"name": player}, {"$pull": {"treasure": jewelry}})  
 myplayer.update\_one({"name": player},{"$set":{"jewelry":jewelry}})  
 if old != None:  
 myplayer.update\_one({"name": player}, {"$push": {"treasure": old}})  
 mytreasure.update\_one({"name": old}, {"$set":{"available":1}})  
 mytreasure.update\_one({"name": jewelry}, {"$set": {"available": 0}, "$set":{"onsale":0}})

# 七、postman测试

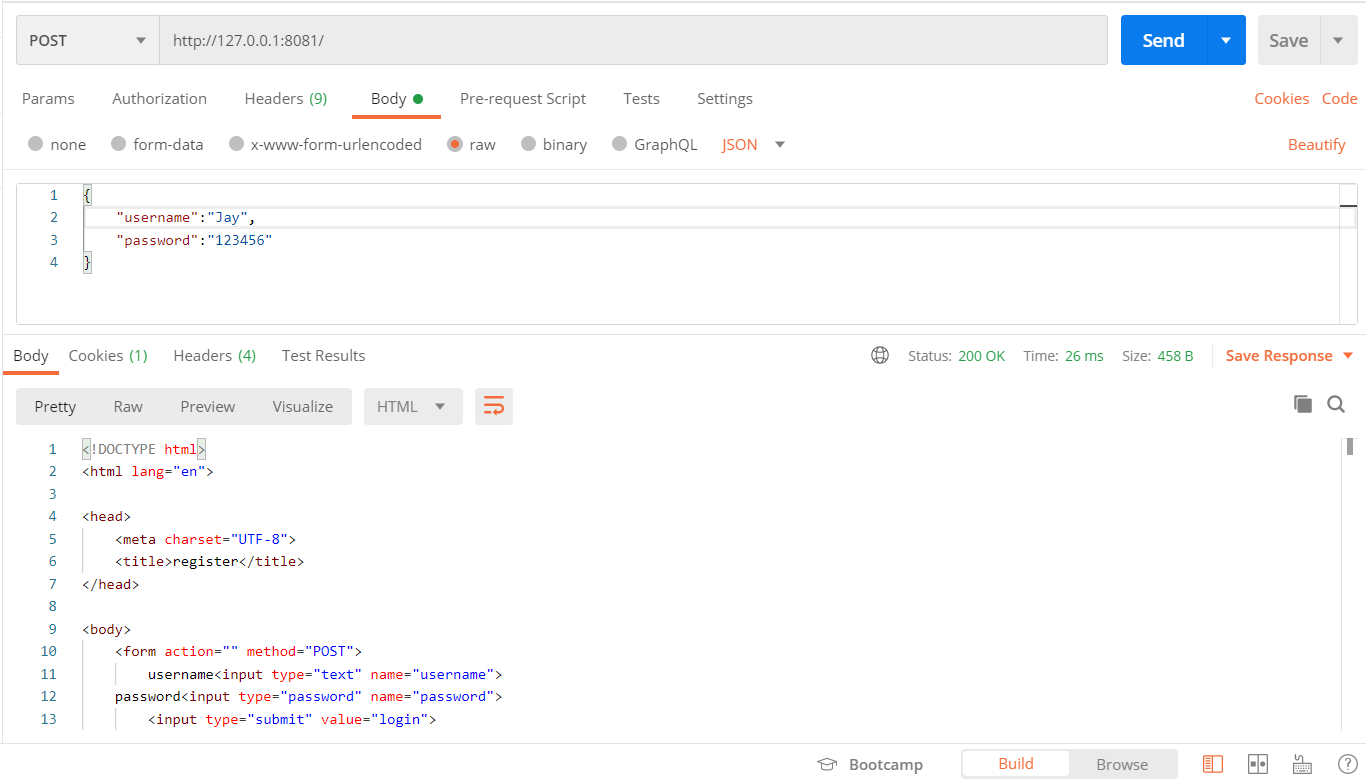
上述功能偏后端，视图函数接收和返回的都是json数据类型，可以直接用postman进行测试。对上述八种功能分别模拟多种情况提交表单，测试结果如下：

(1) 登陆：

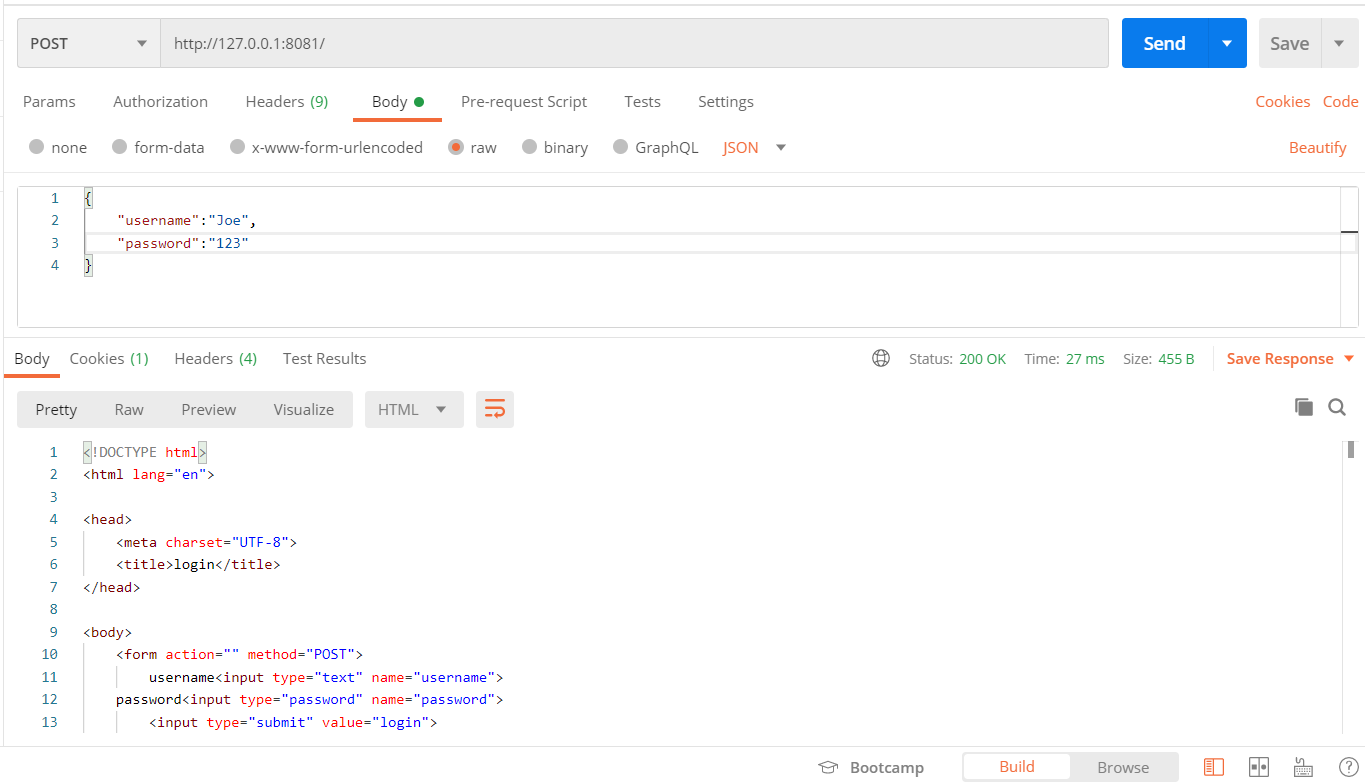
输入正确的用户名和密码，跳转到/<string:username>/home，返回正常的用户信息



输入不存在的用户名，跳转到注册页面

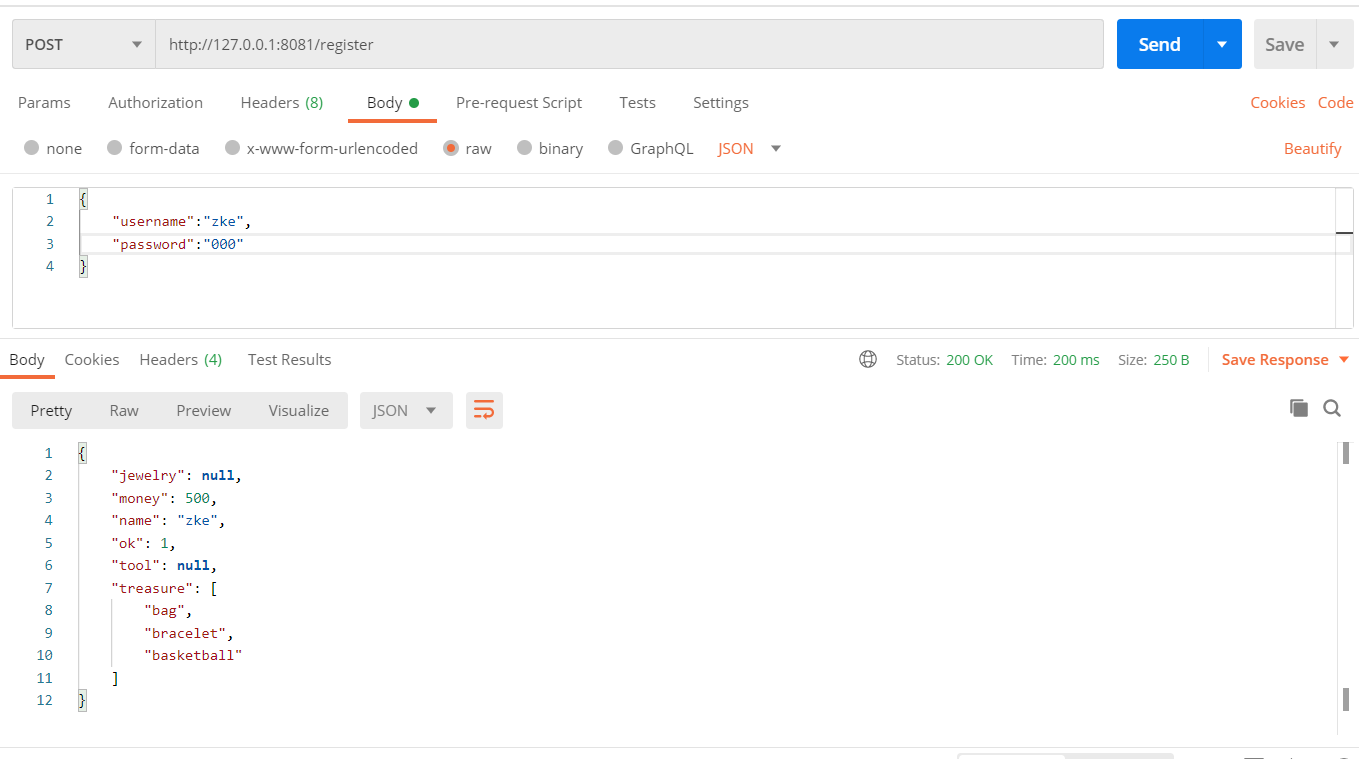


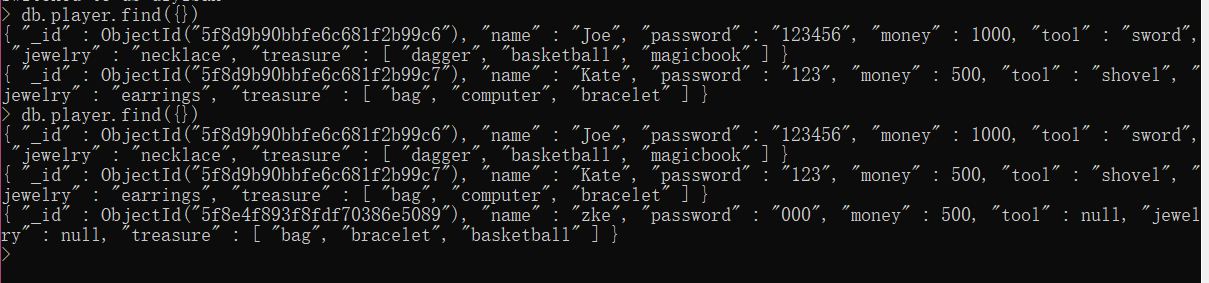
输入不正确的密码，仍然跳转到登陆页面（这时是”GET”方法），可以重新输入用户名和密码：



(2) **注册：**

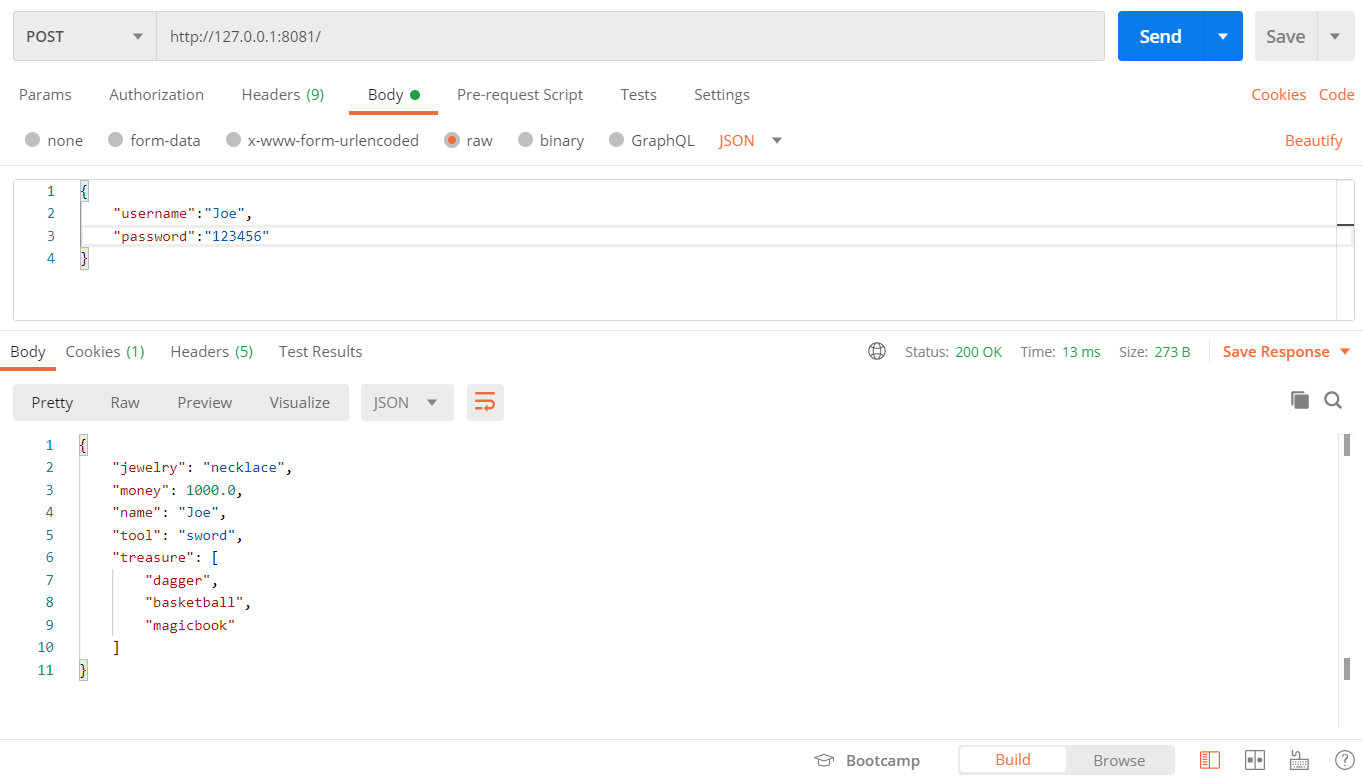
这一功能的实现较简单，没有对密码实现秘文处理。只要post提交了用户名和密码都可以注册。postman测试结果返回了创建玩家后玩家数据的初始状态，可以看到数据库的player集合也有更新。



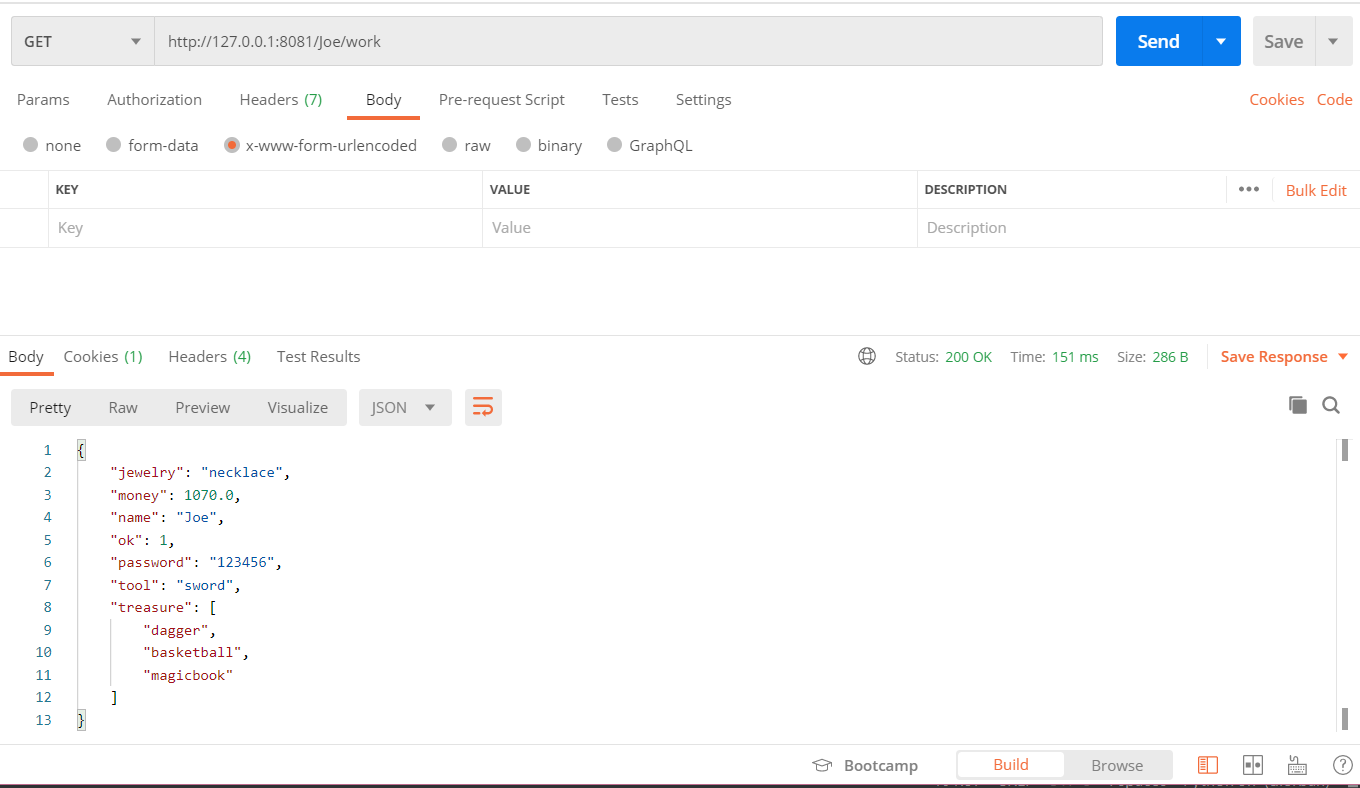


(3) 玩家每天自动赚钱：

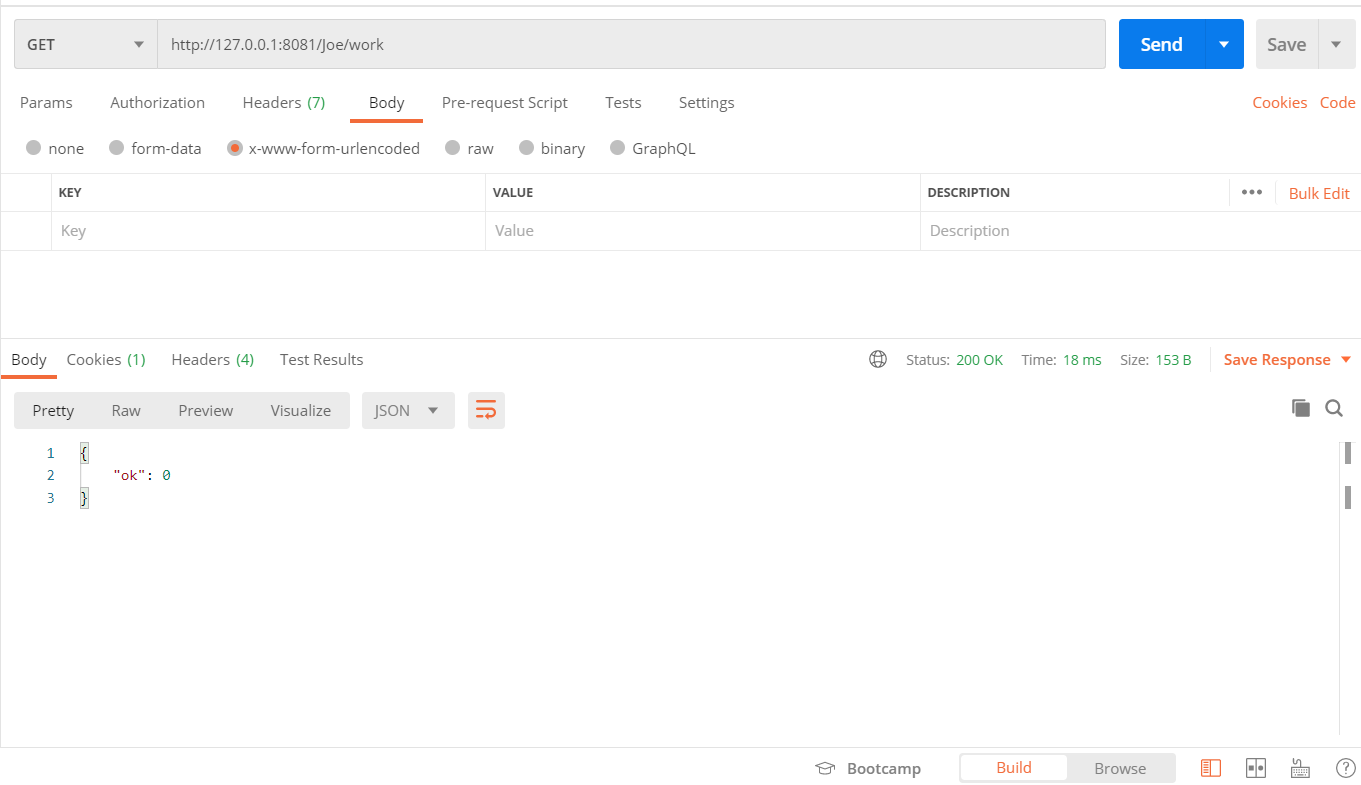
先登录，返回玩家初始状态：



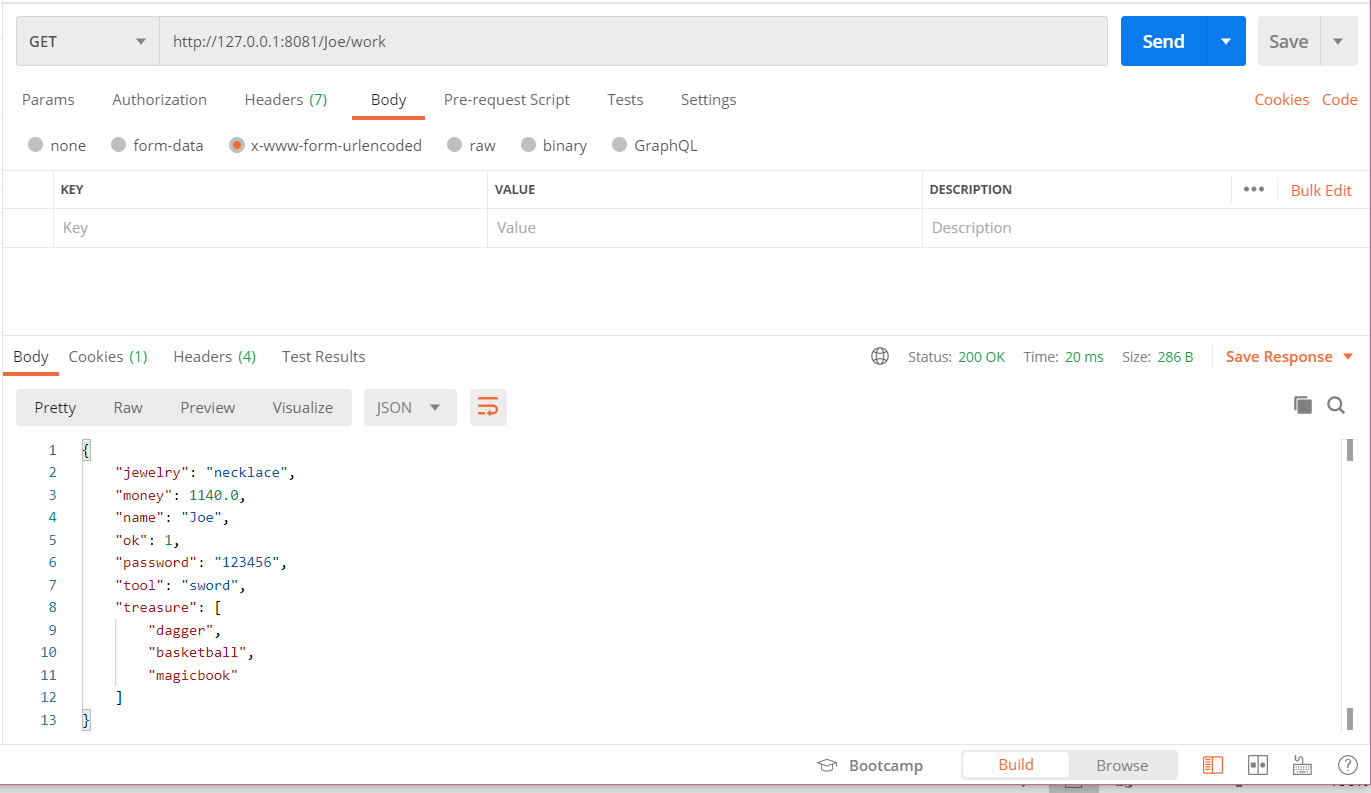
进入玩家的工作页面，金钱增加工作能力的值，即70：



在5分钟内再GET一次，返回{“ok”:0}，表示”今天已经工作过了，金钱没有增加”：



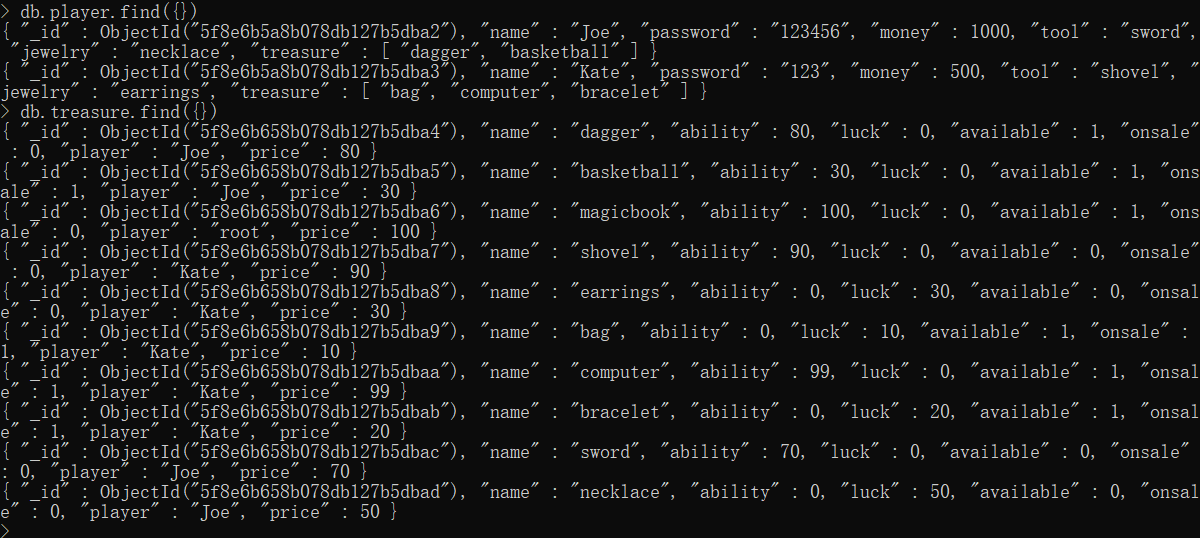
5分钟后再GET一次，返回用户状态，可以看到金钱增加了140：



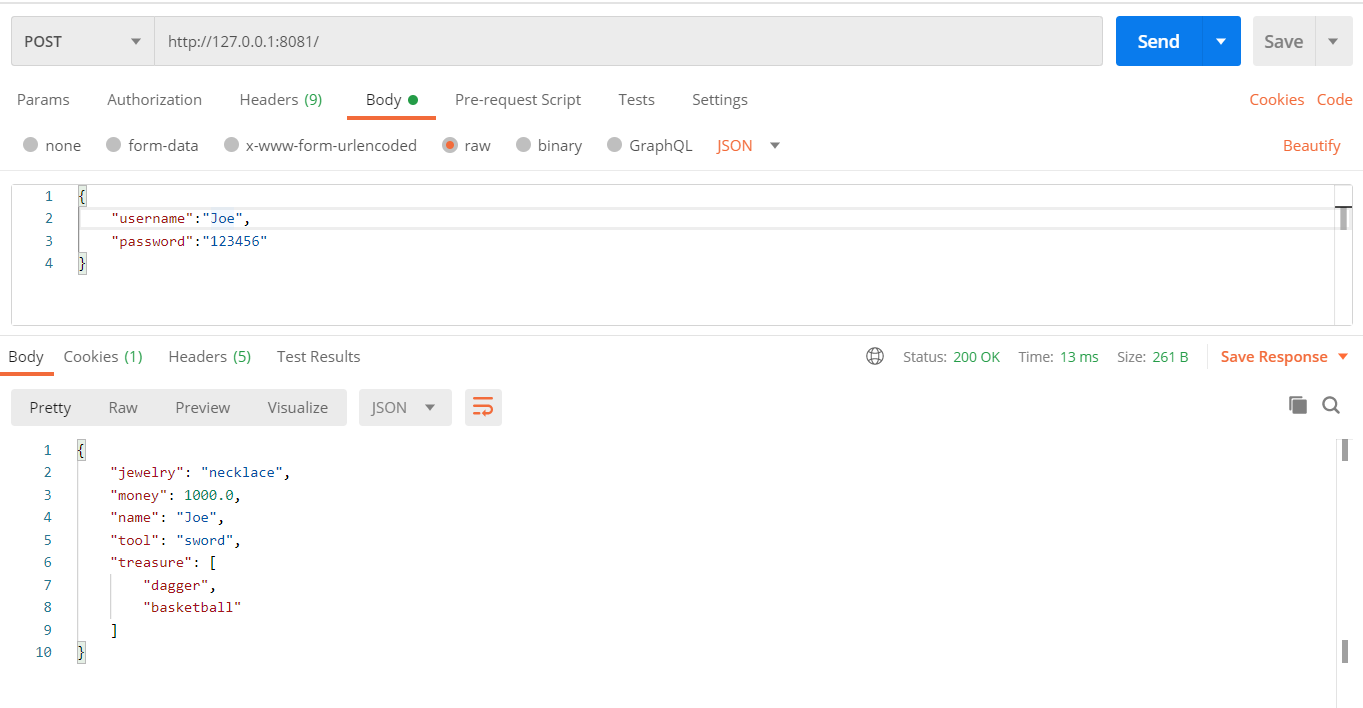
以上四次postman测试说明work视图函数可以正常运行。

(4)玩家每天自动寻宝：

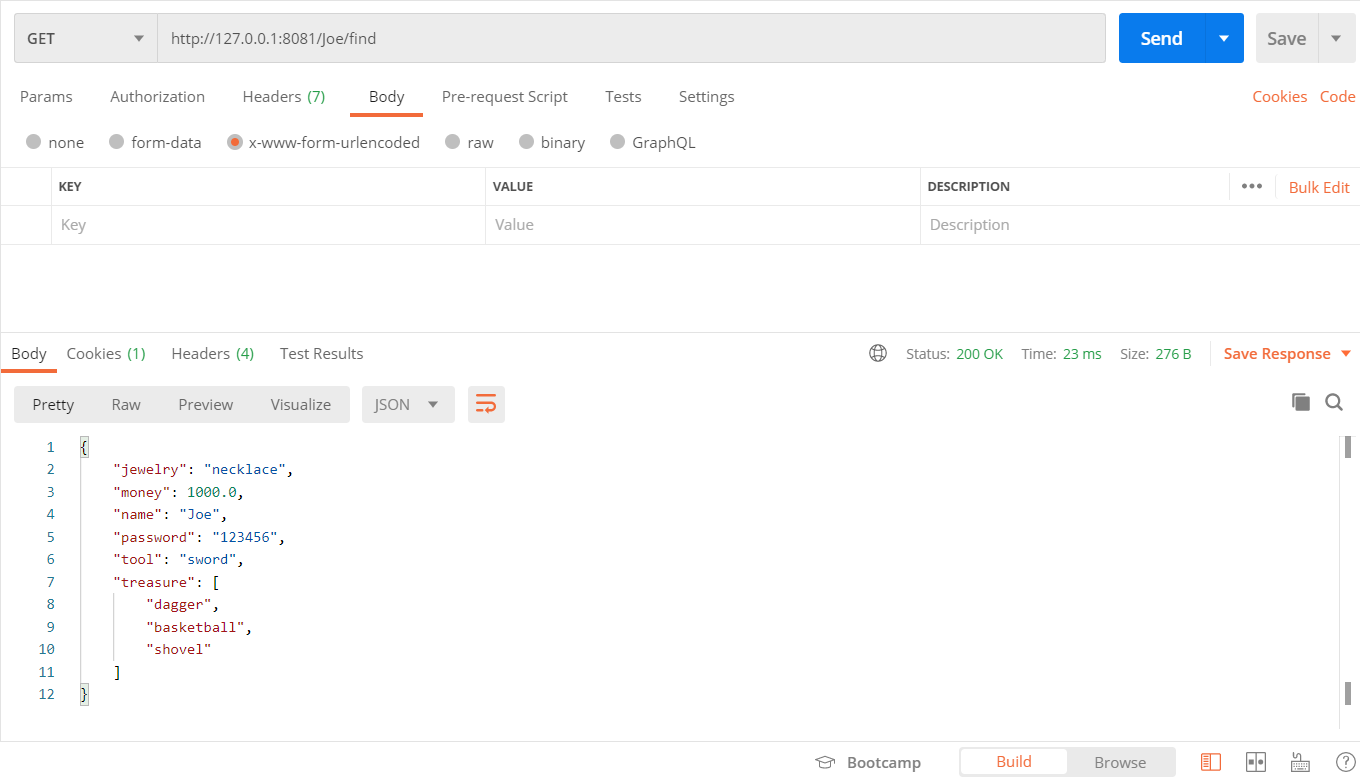
数据库初始状态：



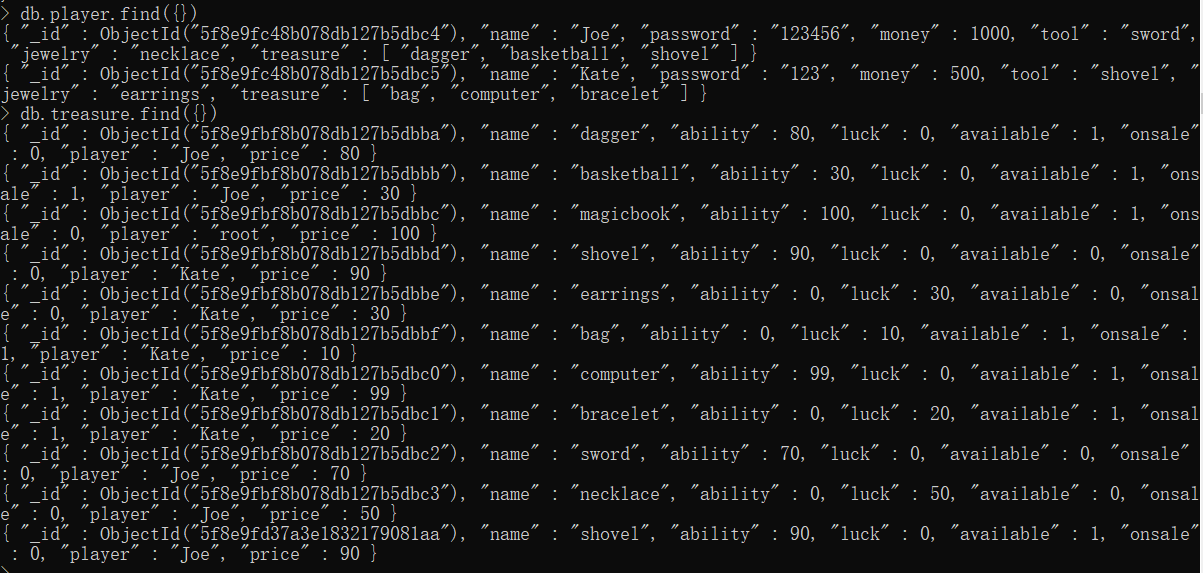
玩家先登陆，返回的数据信息：



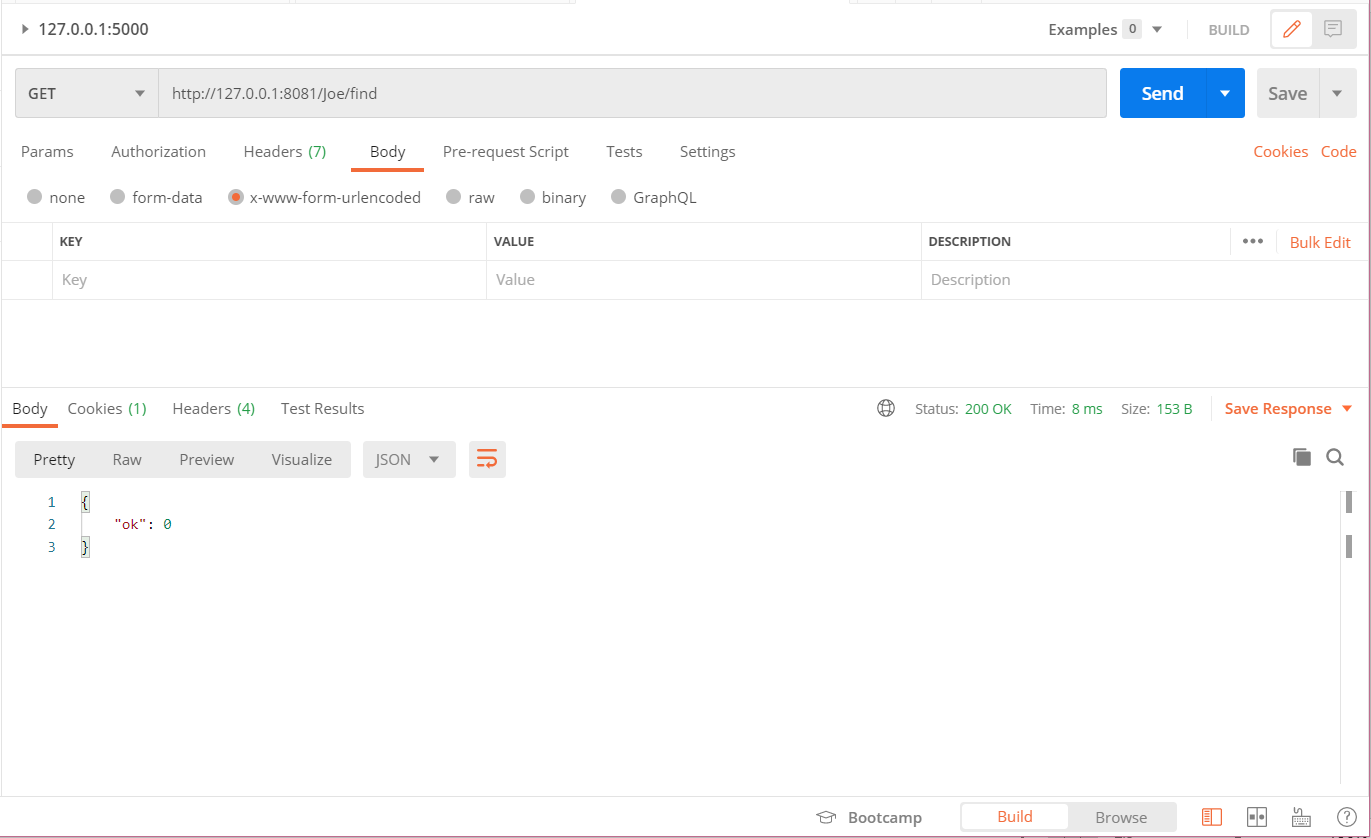
输入/Joe/find一次寻宝后(可以看到系统随机选择出了”shovel”)：



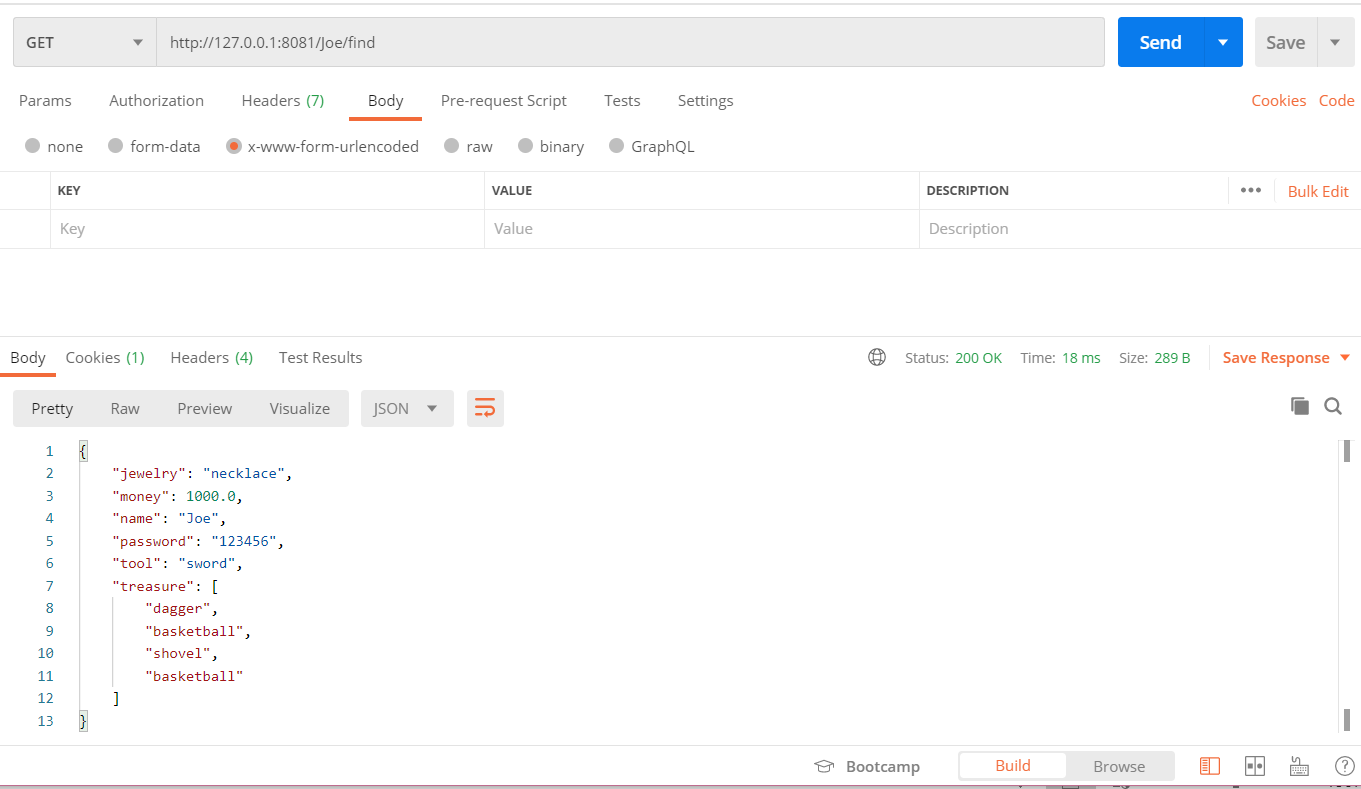
数据库的treasure集合也增加一个”shovel”，且与初始状态的”shovel”的拥有者不同：



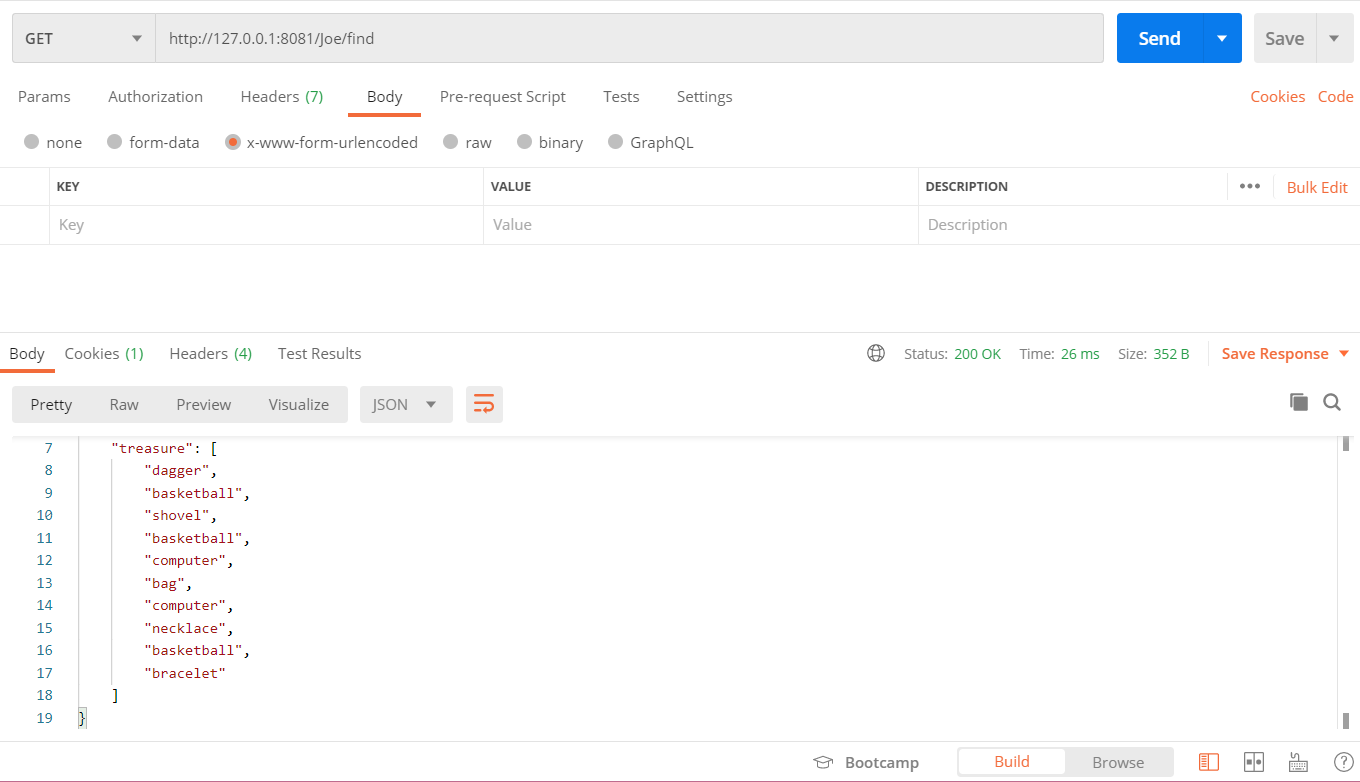
在5分钟内再GET一次，返回{“ok”:0}，表示”今天已经寻宝过了，宝物没有增加”：



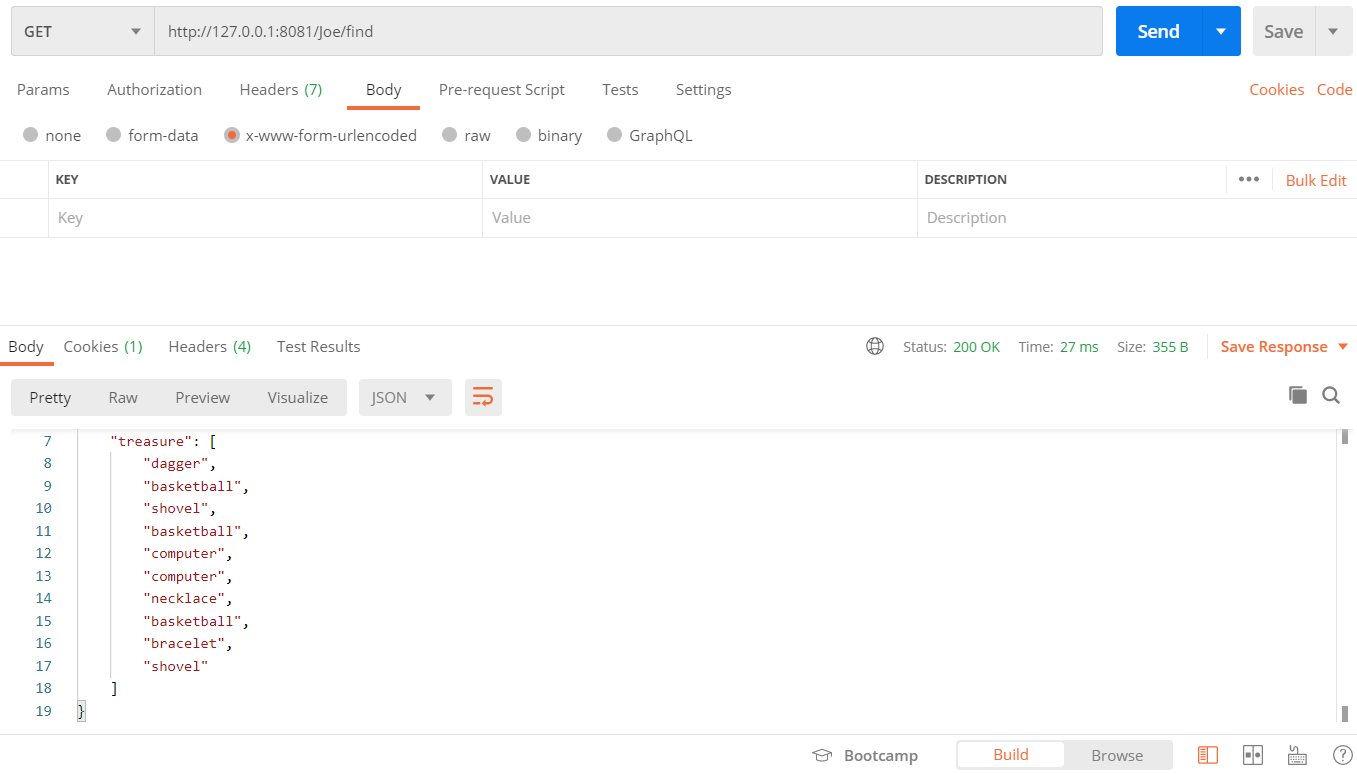
5分钟后再GET一次，返回用户状态，可以看到寻找到了宝物”basketball”：



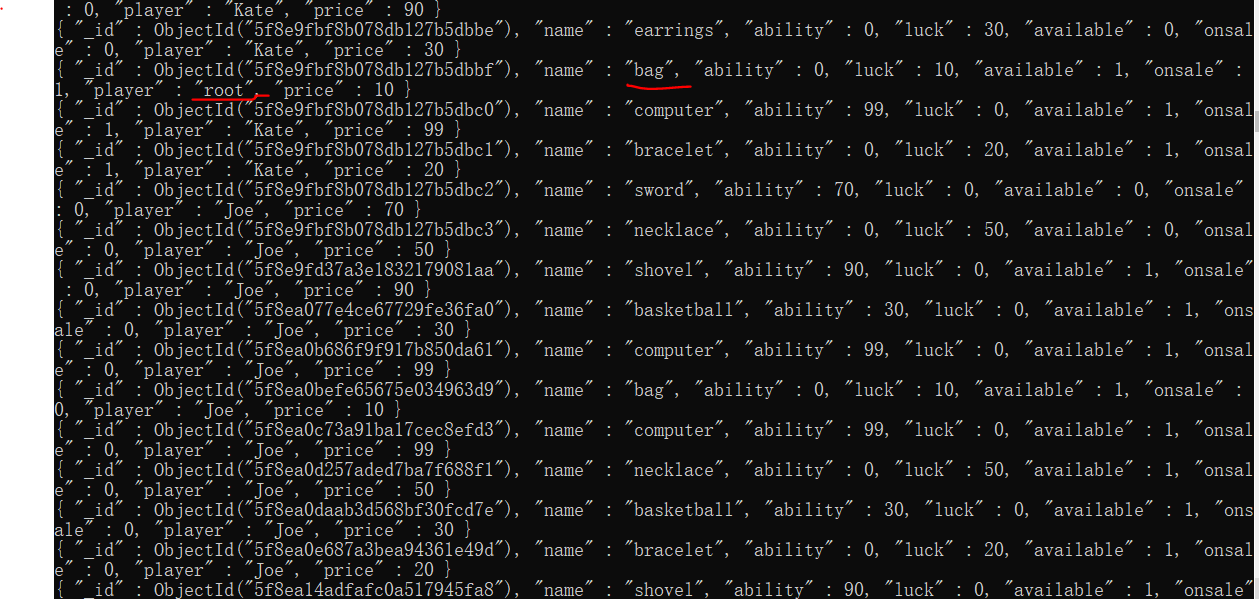
多次”GET”之后宝物箱的宝物个数达到10：(因为最好的宝物magicbook被选择到的概率非常低，因此游戏不会轻易结束)



再”GET”一次，宝物个数仍为10，寻宝得到”shovel”，”bag”被丢弃：

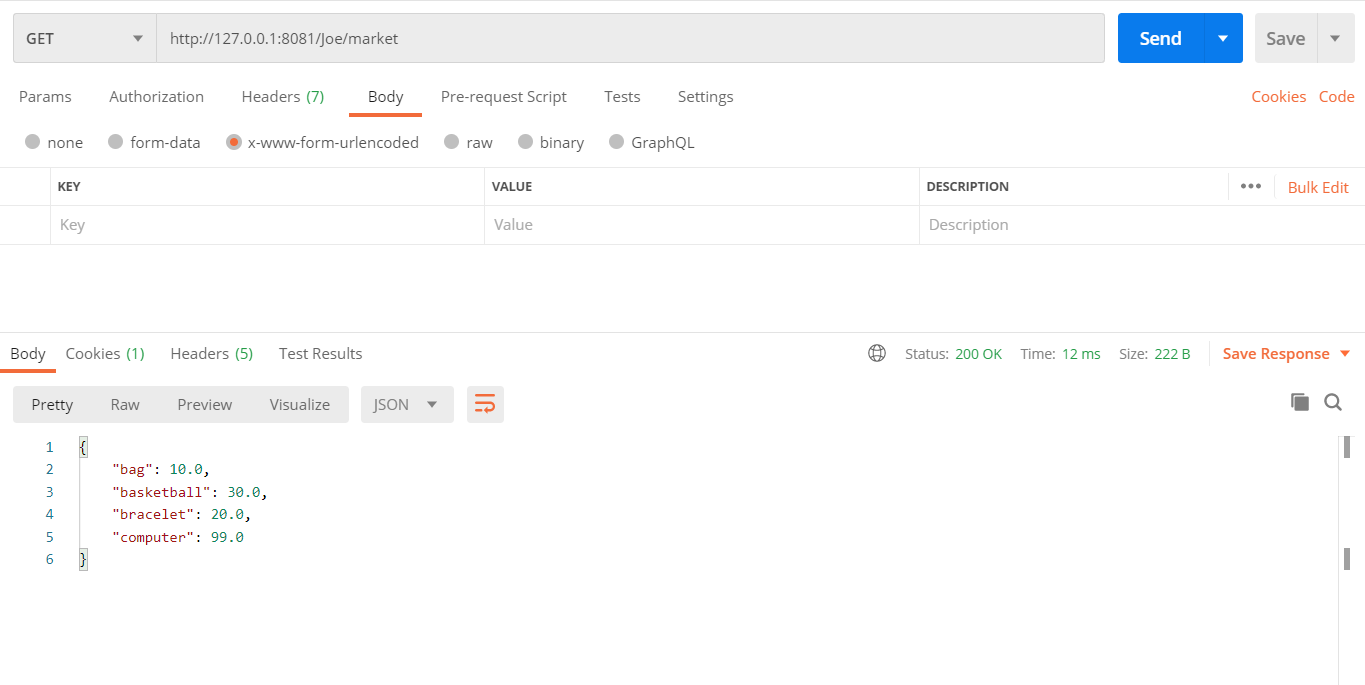


可以看到数据库中被丢弃的”bag”拥有者被置为”root”：



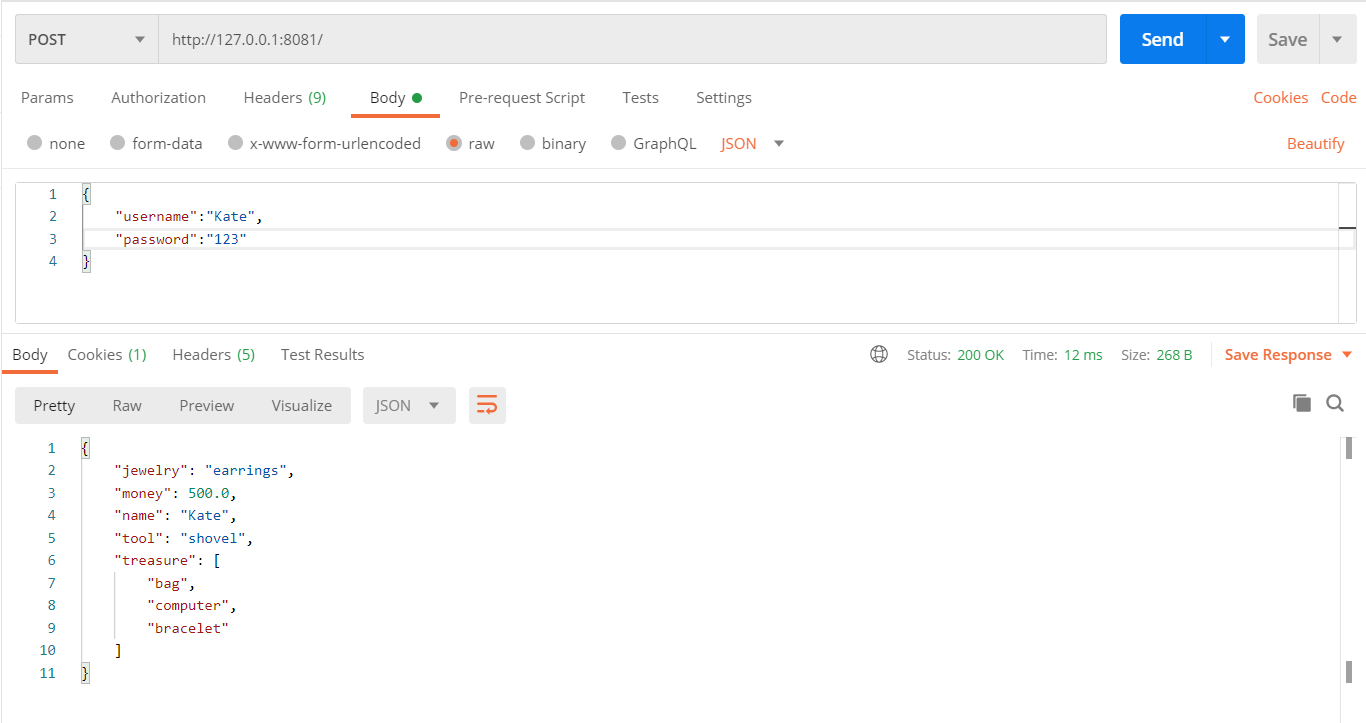
(5) 浏览市场并买宝物

GET http://127.0.0.1:8081/Joe/market，用户可以浏览所有可以购买的宝物。

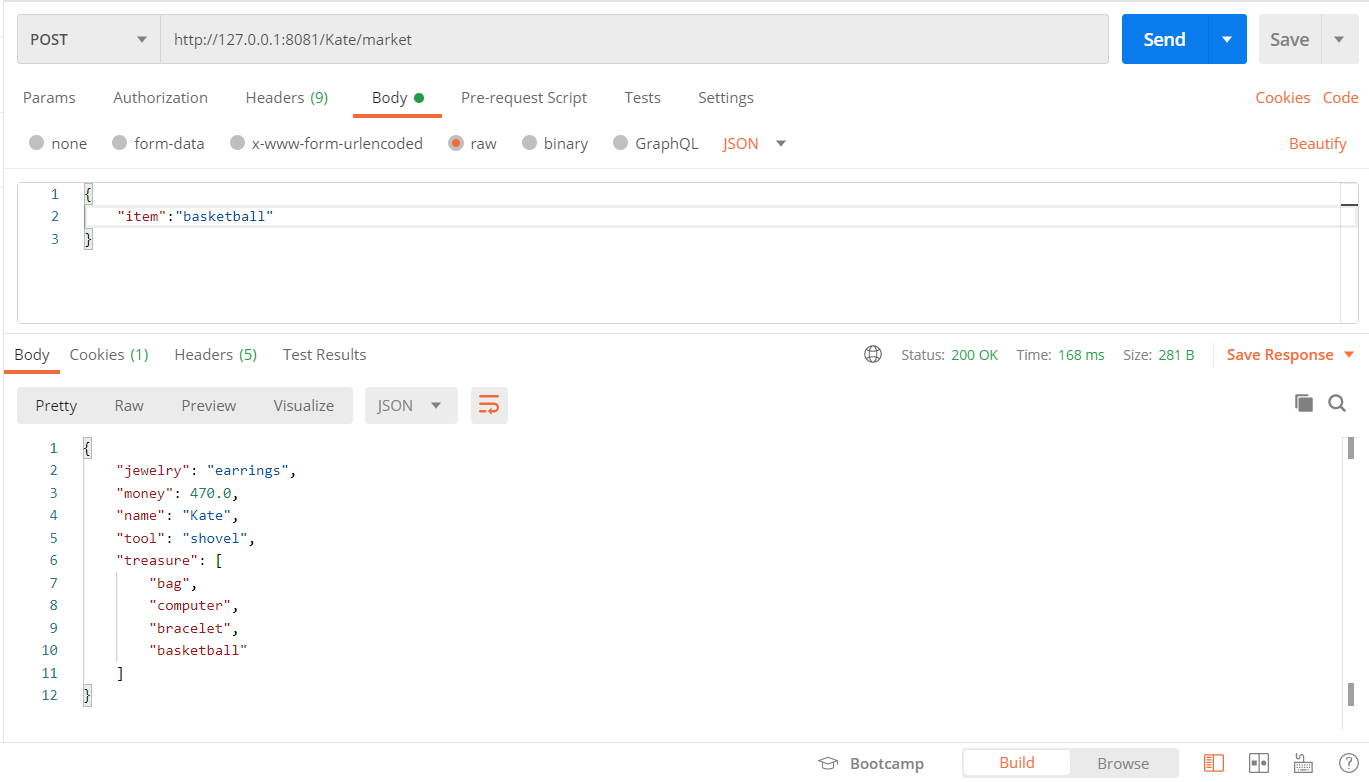


提交post表单进行购买操作：

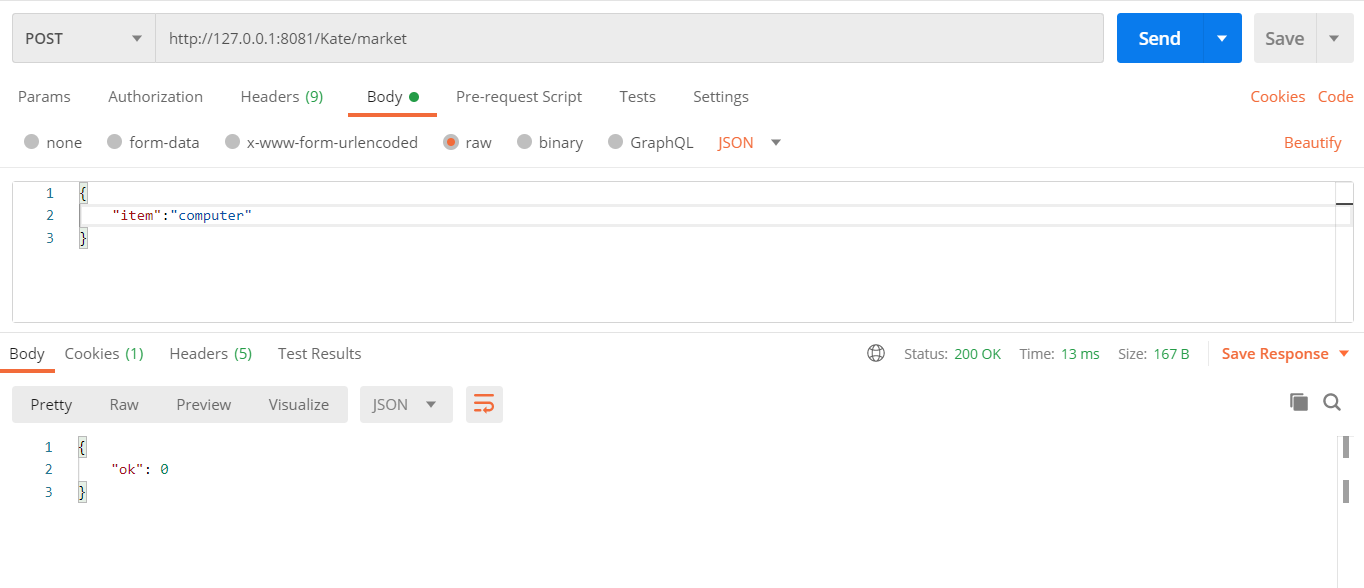
Kate原本有三件宝物：



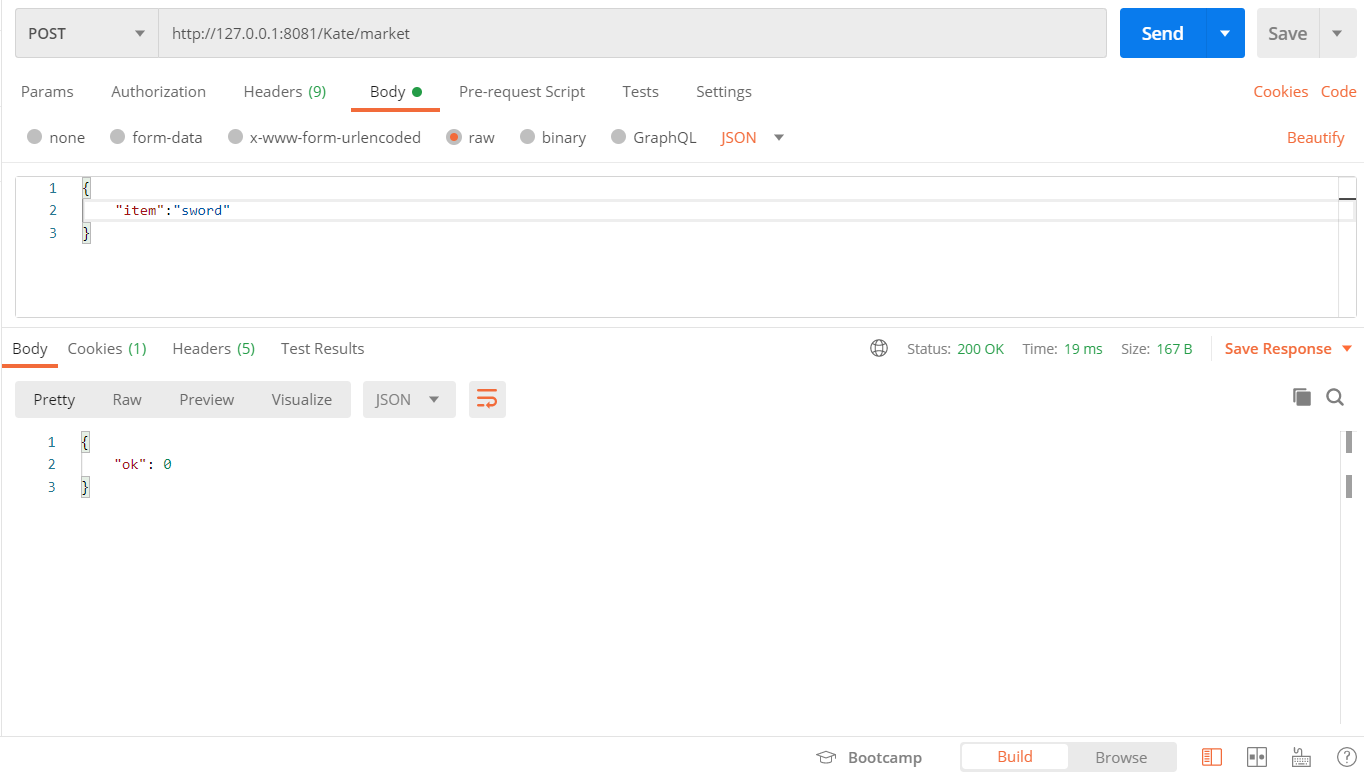
购买Joe的”basketball”之后，宝物增加到四件，金钱-30：



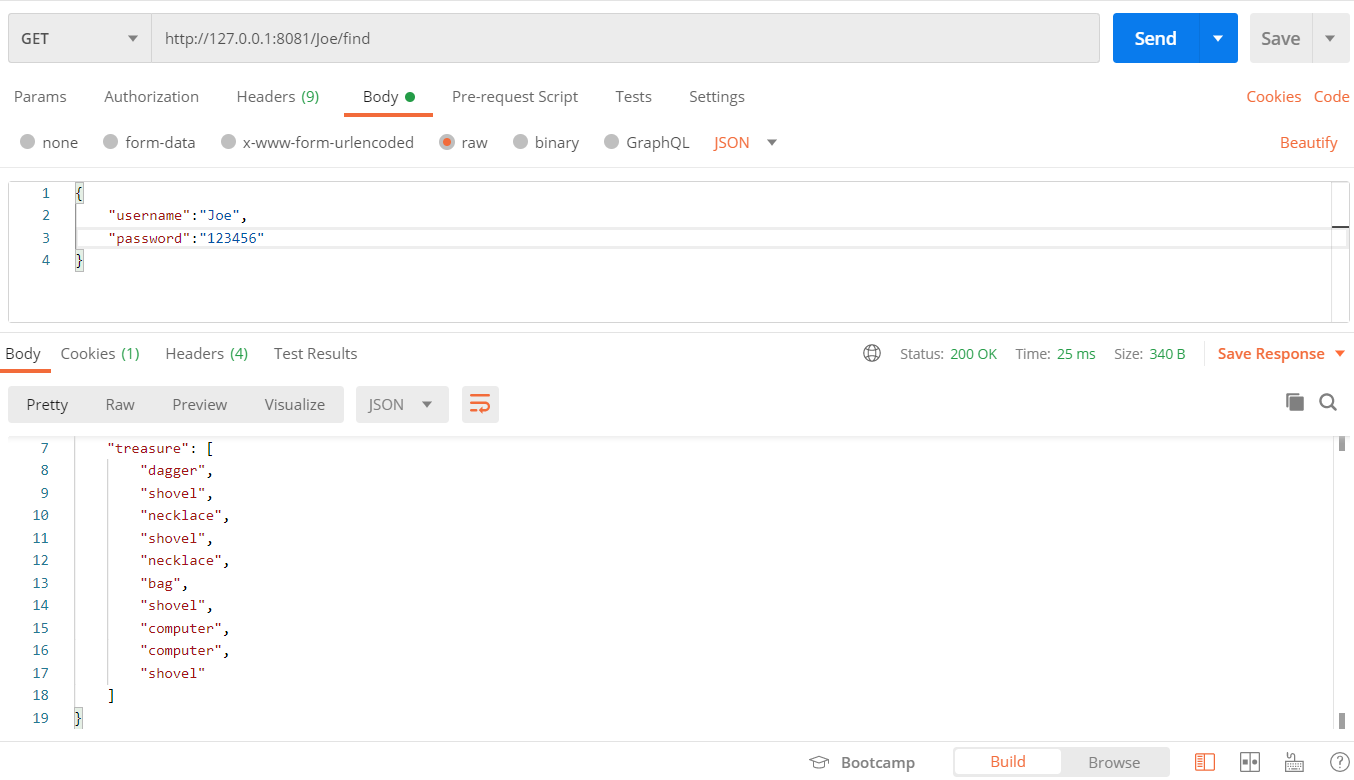
尝试购买自己的宝物，不被允许，返回{“ok”:0}：

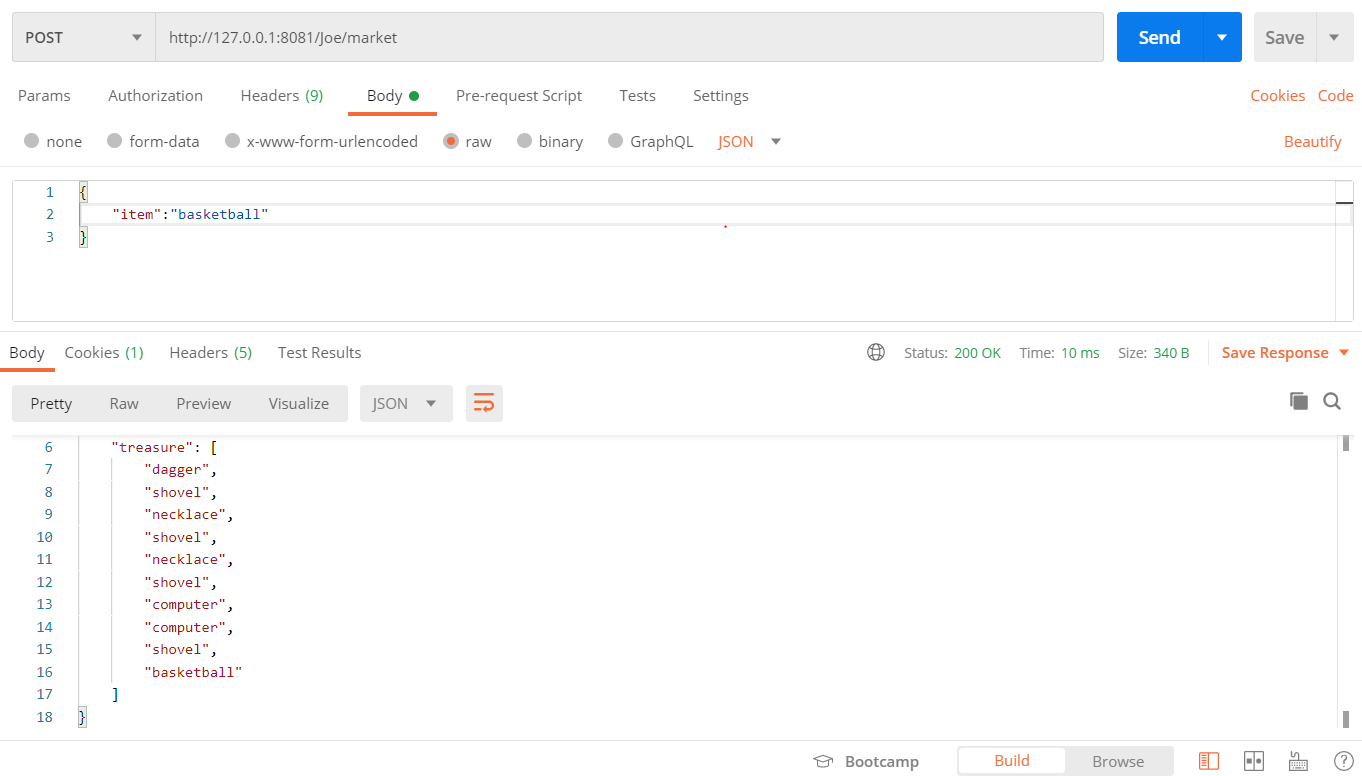


尝试购买不在市场上的宝物，不被允许，返回{“ok”:0}：



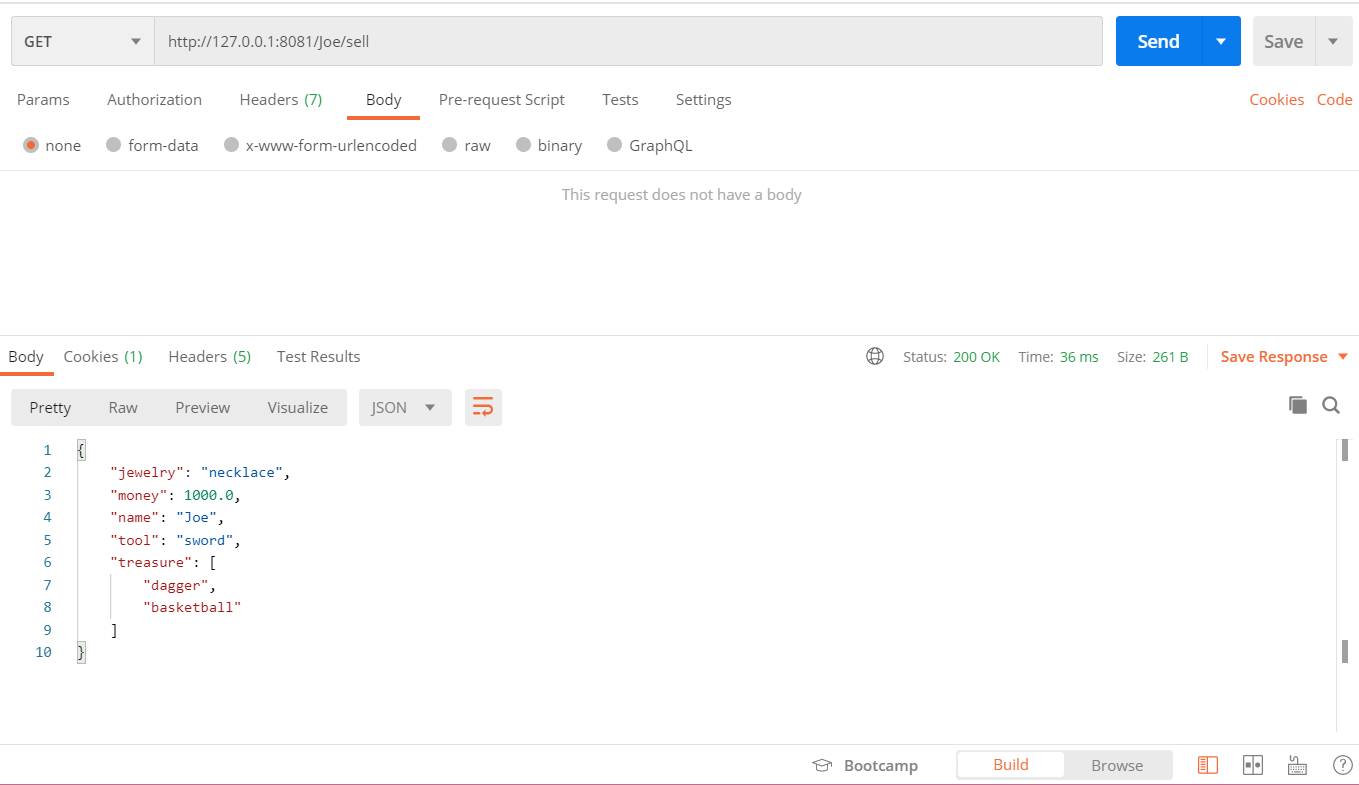
宝物箱已满的Joe购买宝物（会自动将购买后宝物箱中价值最小的宝物丢弃）：



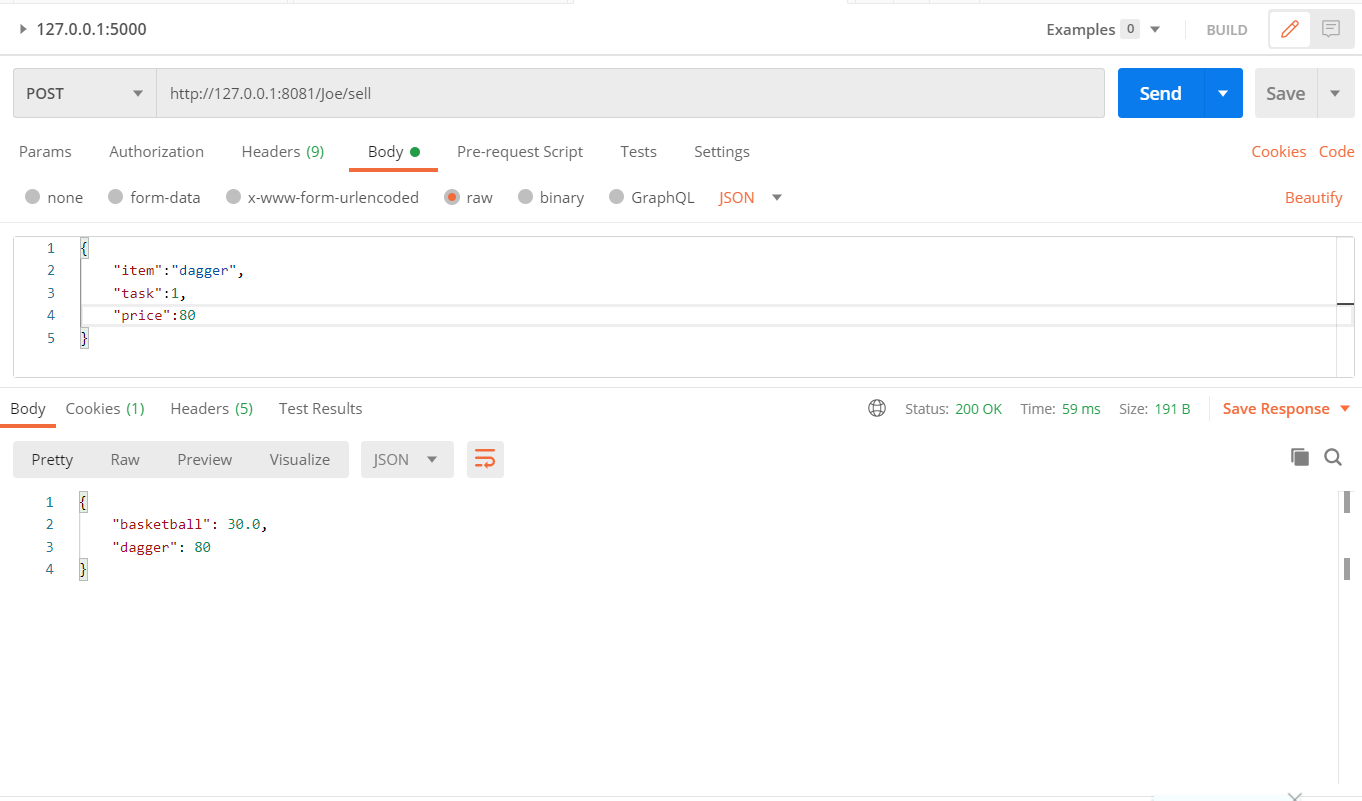


(6) 挂牌宝物或收回宝物：

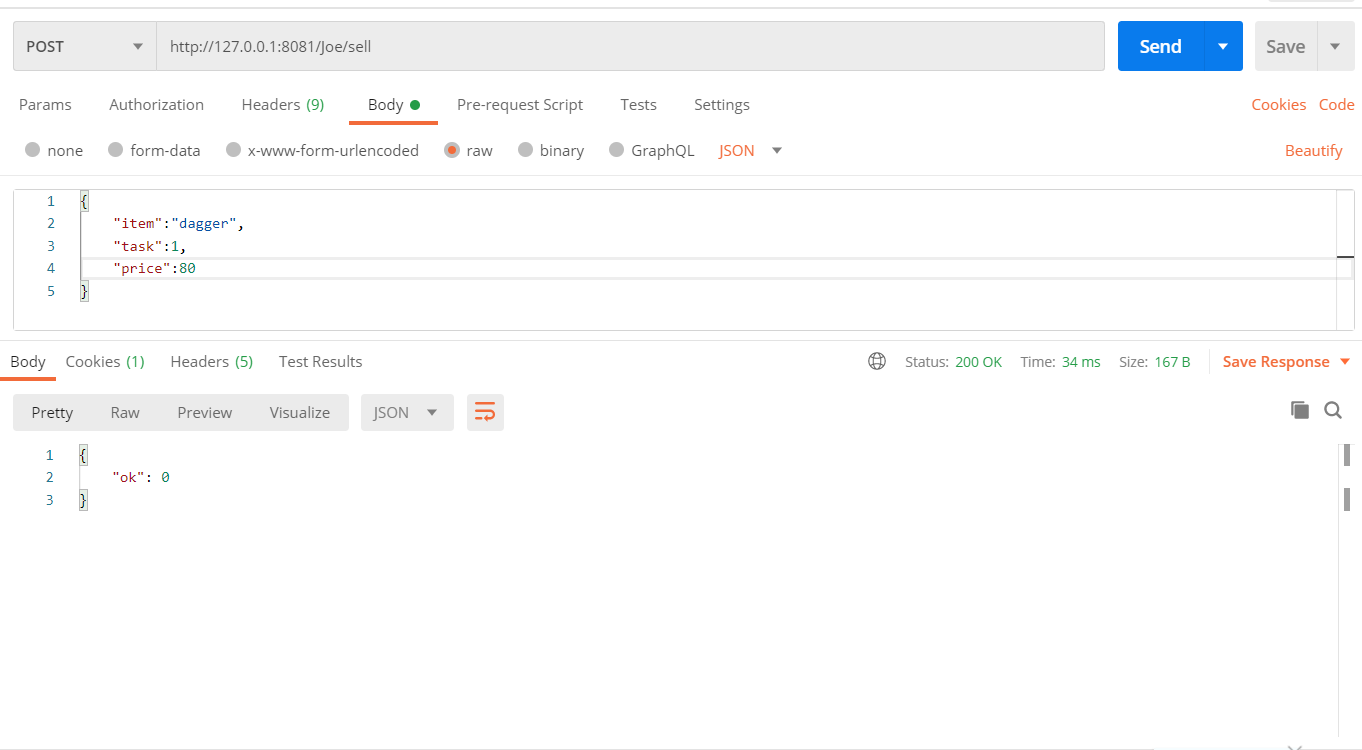
GET http://127.0.0.1:8081/Joe/market，返回用户当前的数据信息



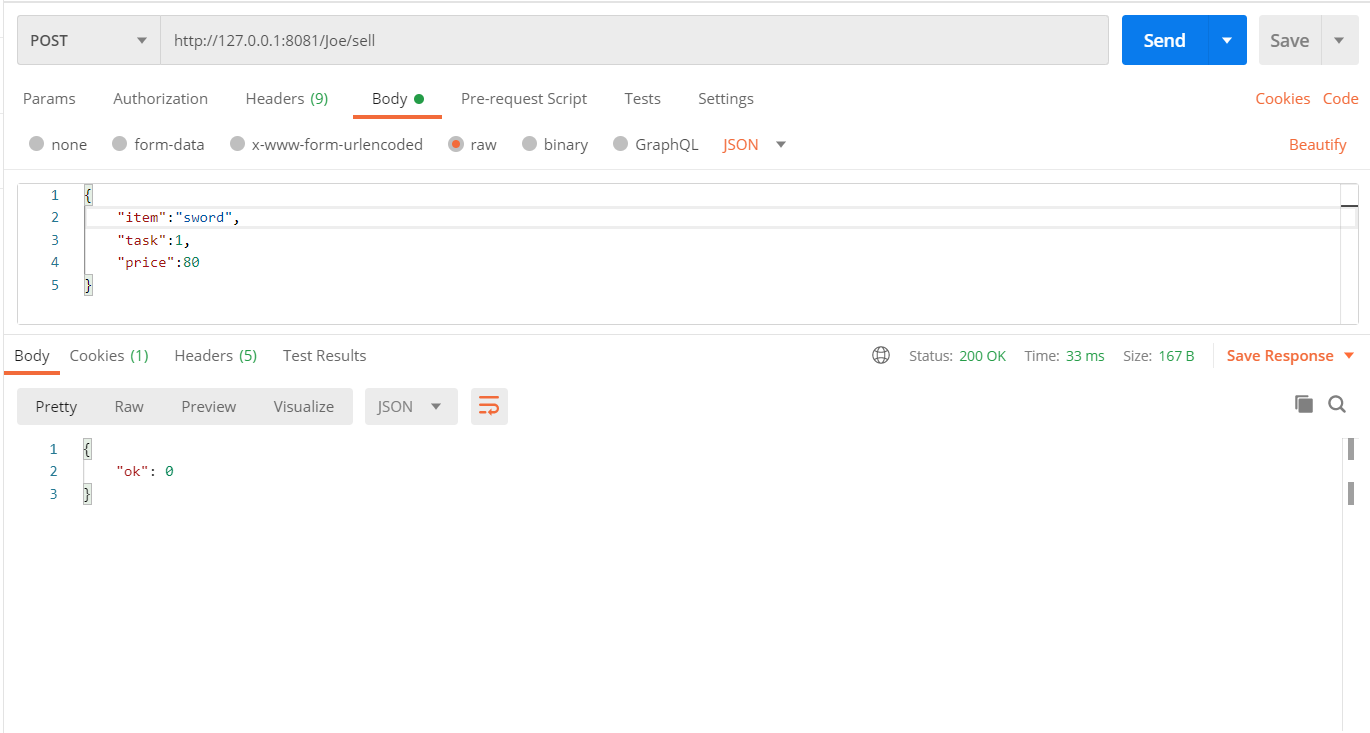
将Joe宝物箱中未出售的宝物挂牌，成功返回Joe所有出售的宝物：



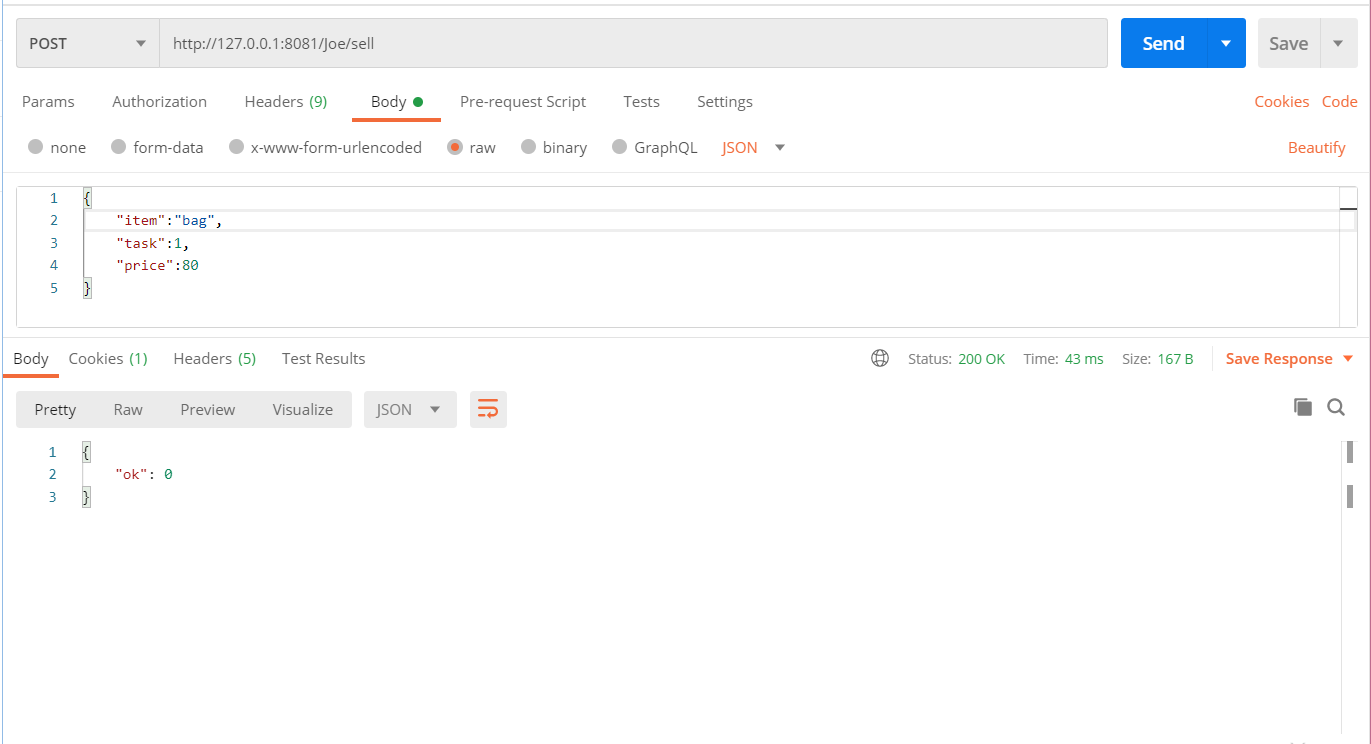
已挂牌没有收回的宝物不能进行挂牌操作：



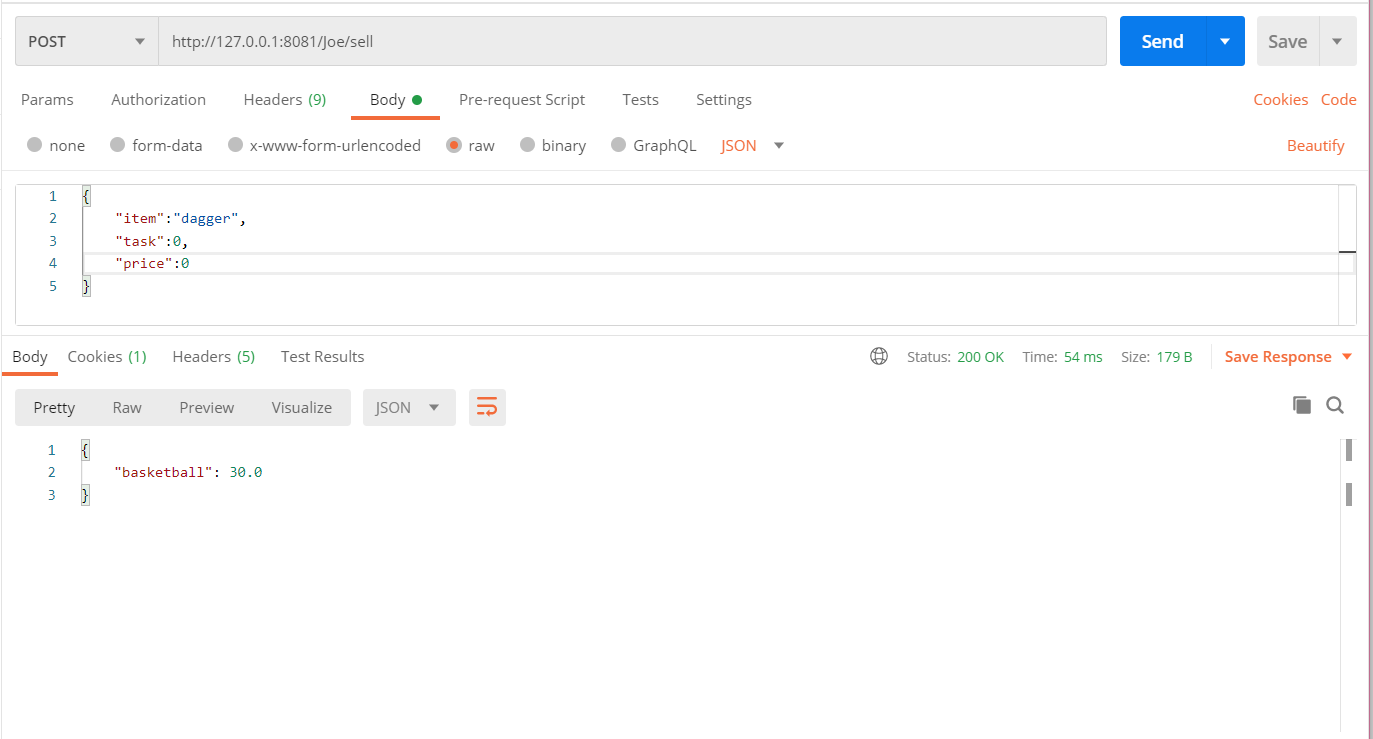
已佩戴的宝物(“available”:0)不能进行挂牌操作：



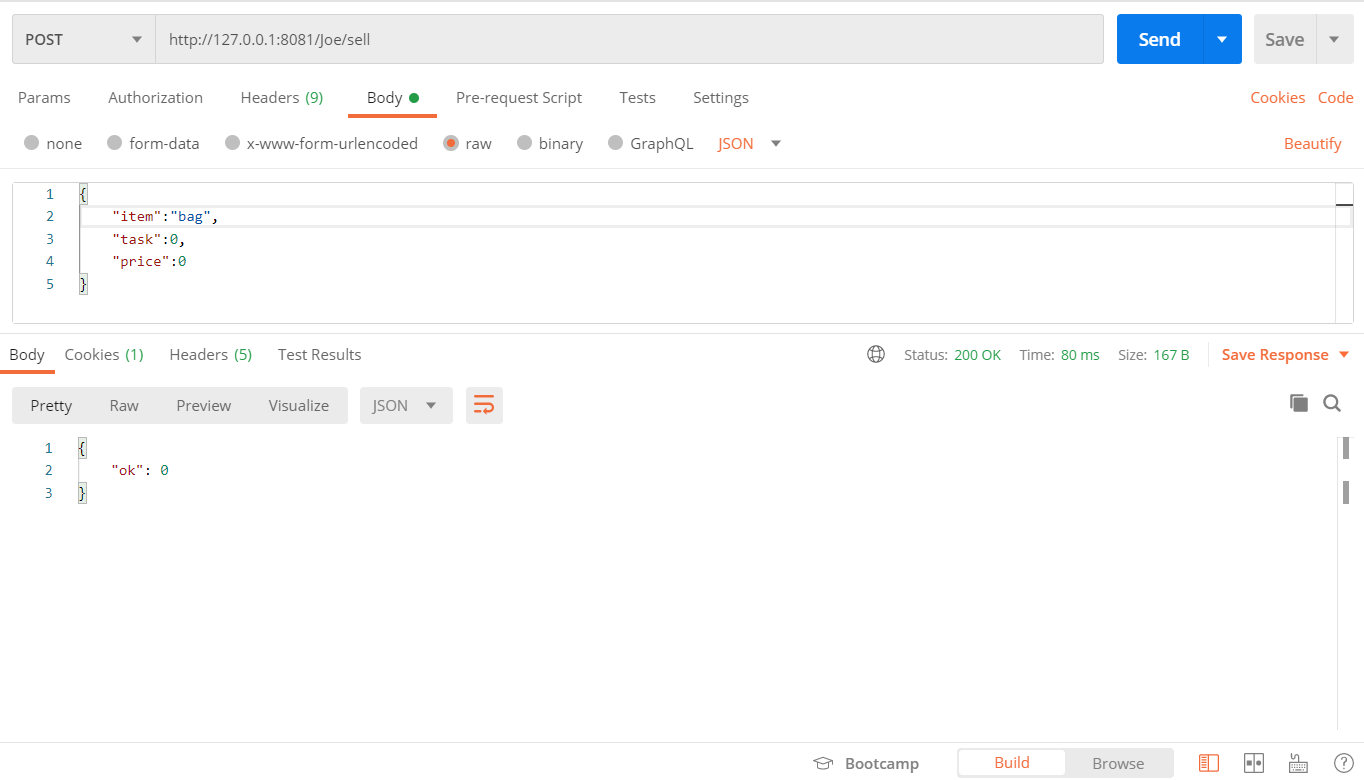
不存在的宝物不能进行挂牌操作：



将挂牌的宝物收回：

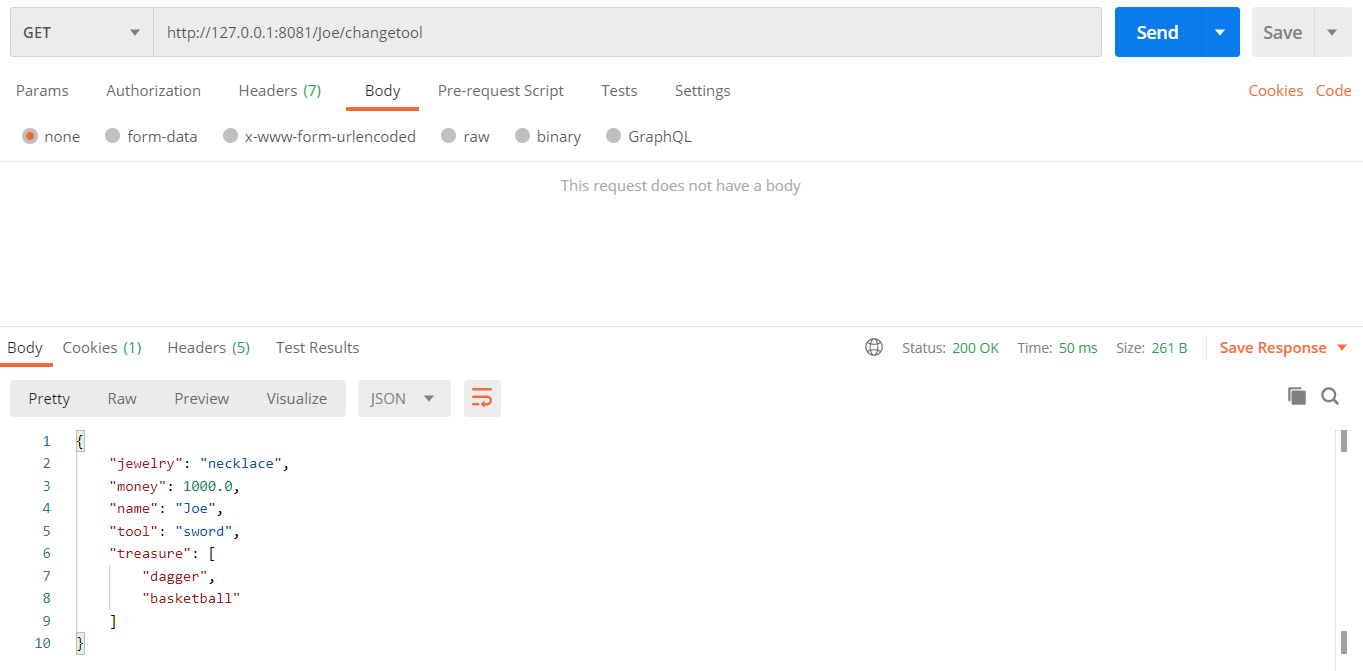


不能将不在出售的宝物收回：

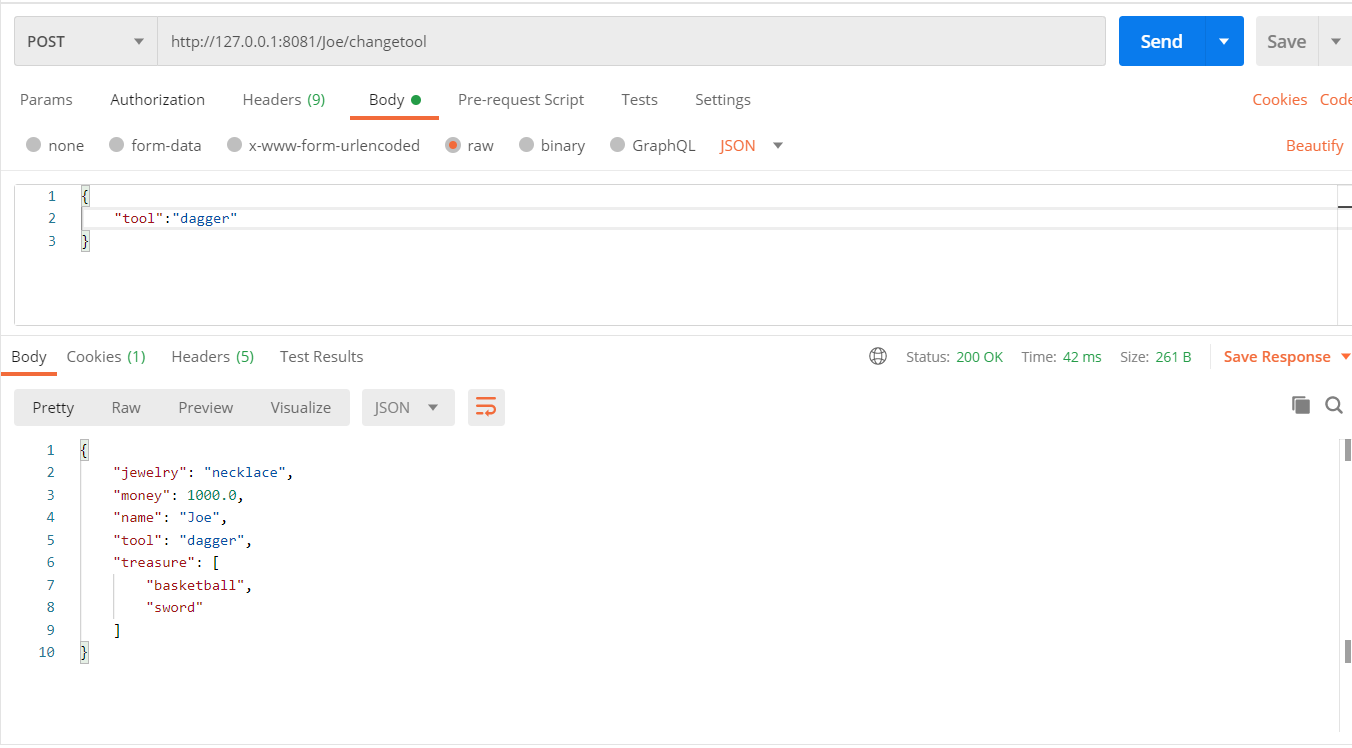


(7) 更换佩戴工具

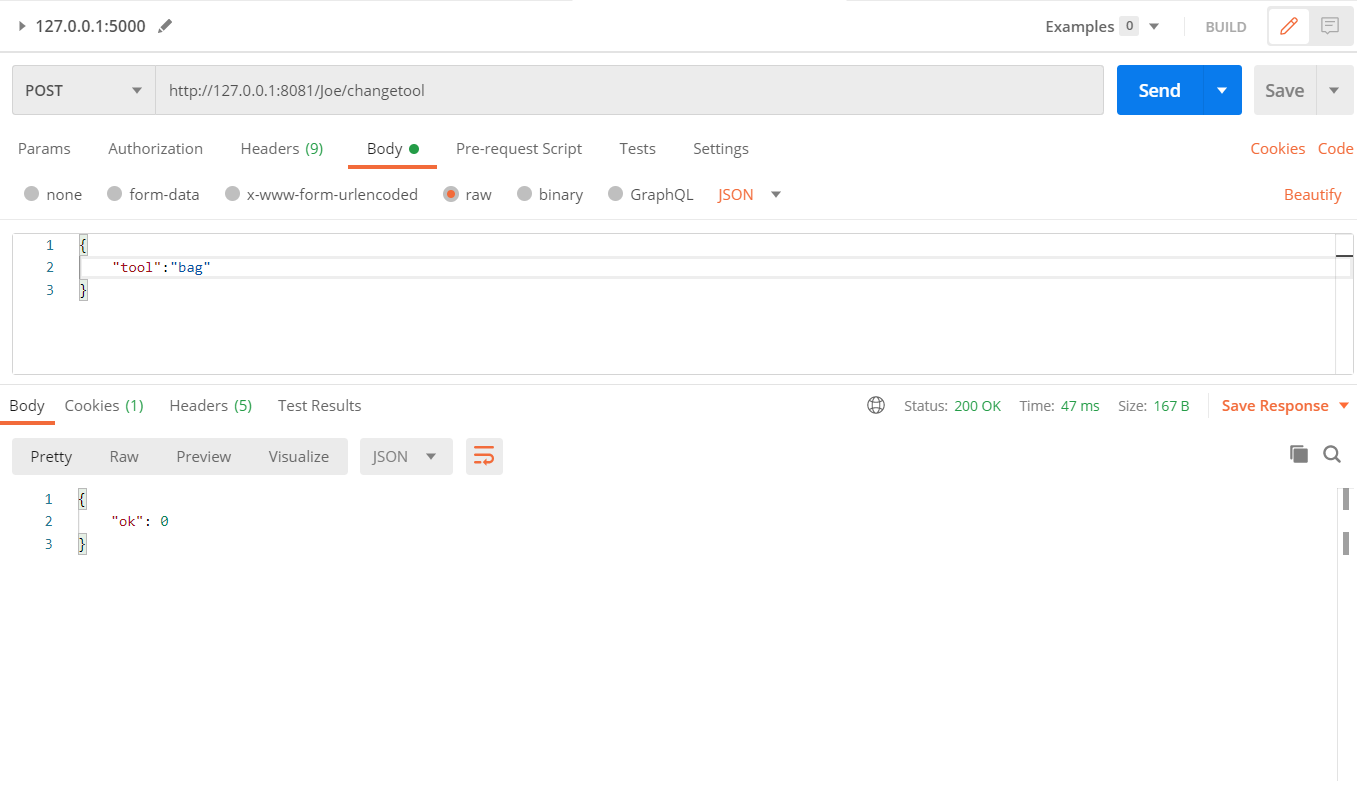
GET http://127.0.0.1:8081/Joe/changetool，返回用户当前的数据信息



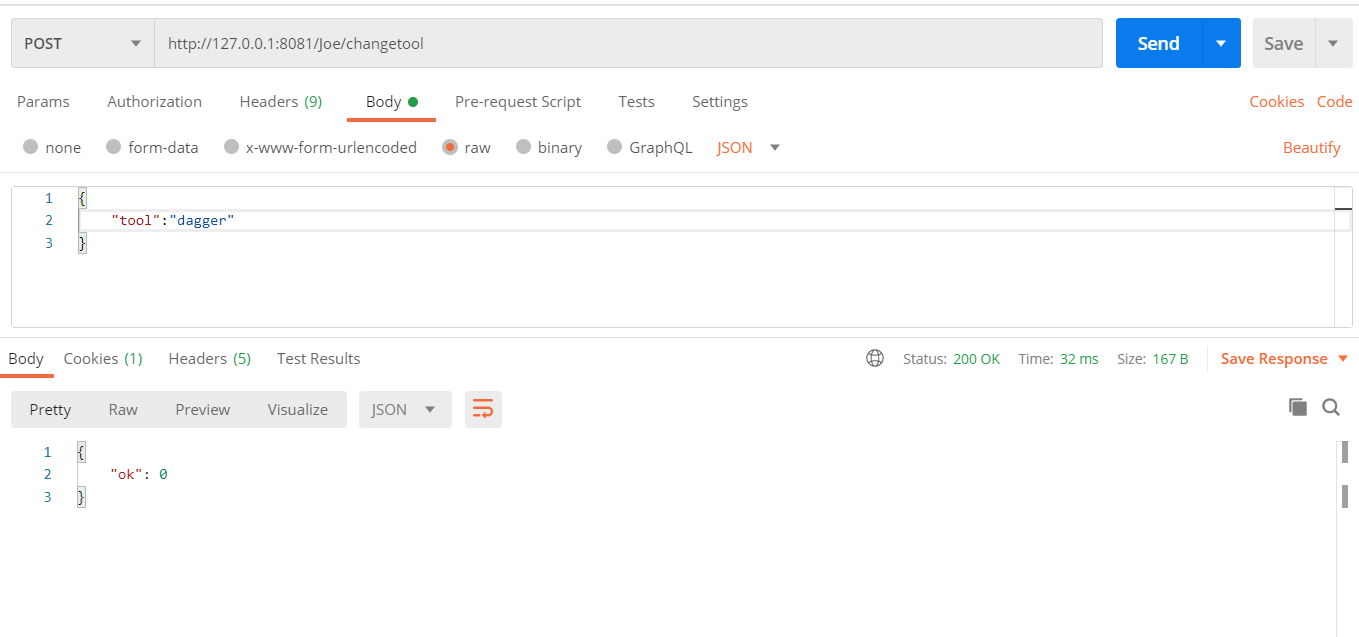
将佩戴的sword更换为dagger, sword放入宝物箱：



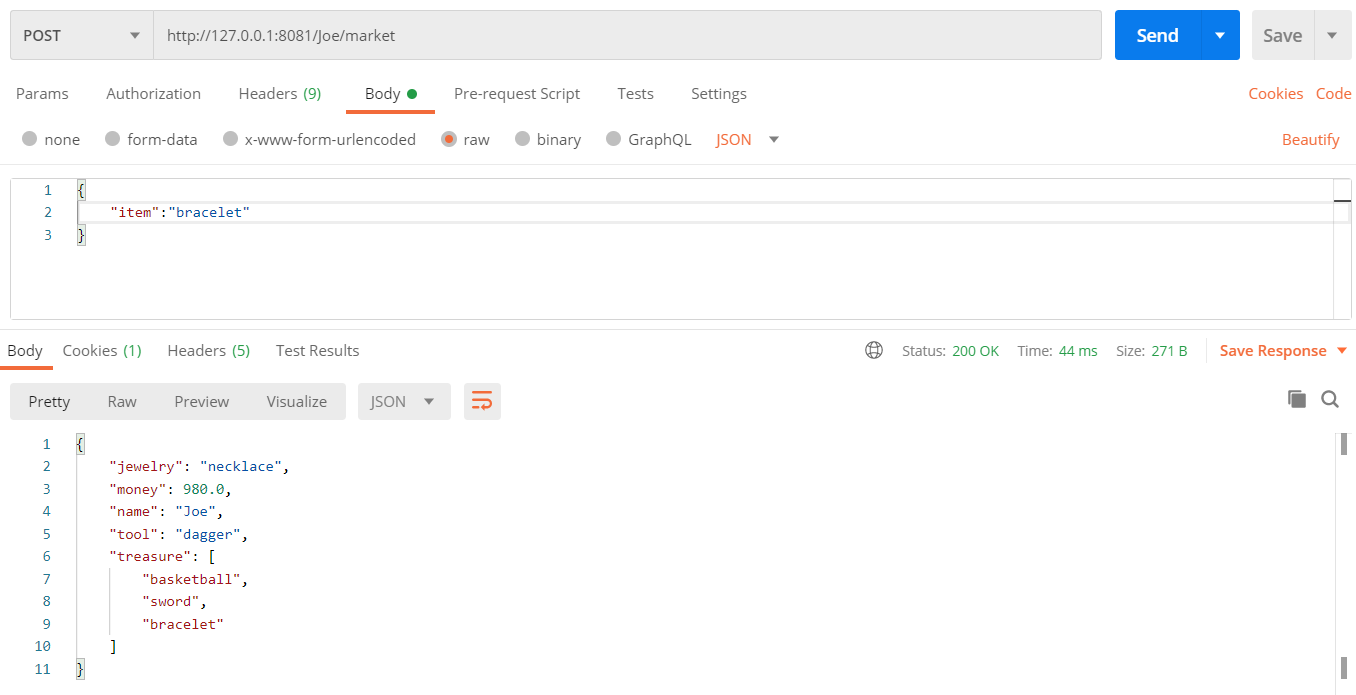
不能佩戴玩家没有的工具：

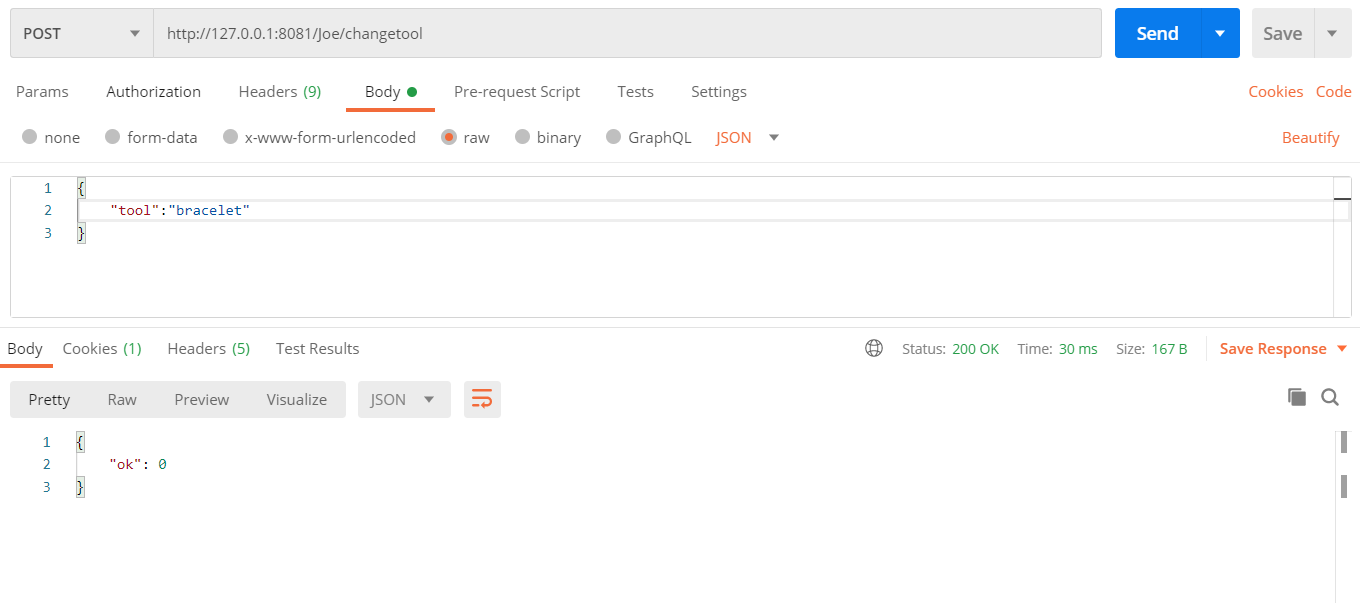


不能佩戴不在宝物箱的工具：



不能在更换工具时换成配饰：





(8) 更换佩戴配饰

由于代码逻辑与第七个功能一模一样，不重复测试。

# 八、postman测试过程中发现的bug

一开始认为如果find\_one()函数没有找到文档，会返回空字典，代码判断都写成了if result == {}，经测试发现应该是if result == None。

在(4)玩家每天自动寻宝中，当宝物个数超过10，需要丢弃宝物时，我一开始的代码是mytreasure.update\_one({"name": temp1["name"]}, {"$set": {"player": "root"}})。在测试中发现数据库里有两个bag，一个是Joe的，一个是Kate的。这样操作无意中将Kate的”bag”的拥有者修改为”root”，而应该修改的是Joe。代码改写为mytreasure.update\_one({"player": username, "name": temp1["name"]}, {"$set": {"player": "root"}})后正常工作。

在(5) 浏览市场并买宝物中，一开始没有考虑到post购买自己的宝物的情况，后补上，不允许该操作

pull会把所有符合的元素从数组中删除，而我们只要删除一个宝物

# 九、一些在设计过程中改变的部分(前端)

最开始我不知道是要用postman进行json数据类型的交互，因此写了该游戏的部分前端html文件，设计了跳转结构，具体的html文件在templates文件夹下，相关代码与现有部分改动较大。

最开始设计的所有的postman测试都需要手动输入用户名，这显然与现实中的逻辑不符。在使用一个web服务时，用户登陆后即可在自己的权限范围内进行操作，为了模拟这一实现，引入flask框架的session模块。

用户在login页面登陆时需要输入用户名和密码，这两个用户的状态信息被存储在session中，可以随时取用。

登陆后再进行测试，比如想要更换佩戴的工具类宝物，测试http://127.0.0.1:8081/changetool之前先post http://127.0.0.1:8081/，在x-www-form-urlencoded 中输入用户名和密码的key-value对，之后测试不同的功能都在这个已登陆的用户下操作，那么post的JSON数据可以去掉”player”:”Joe”，使用{

    "tool":"dagger"

}

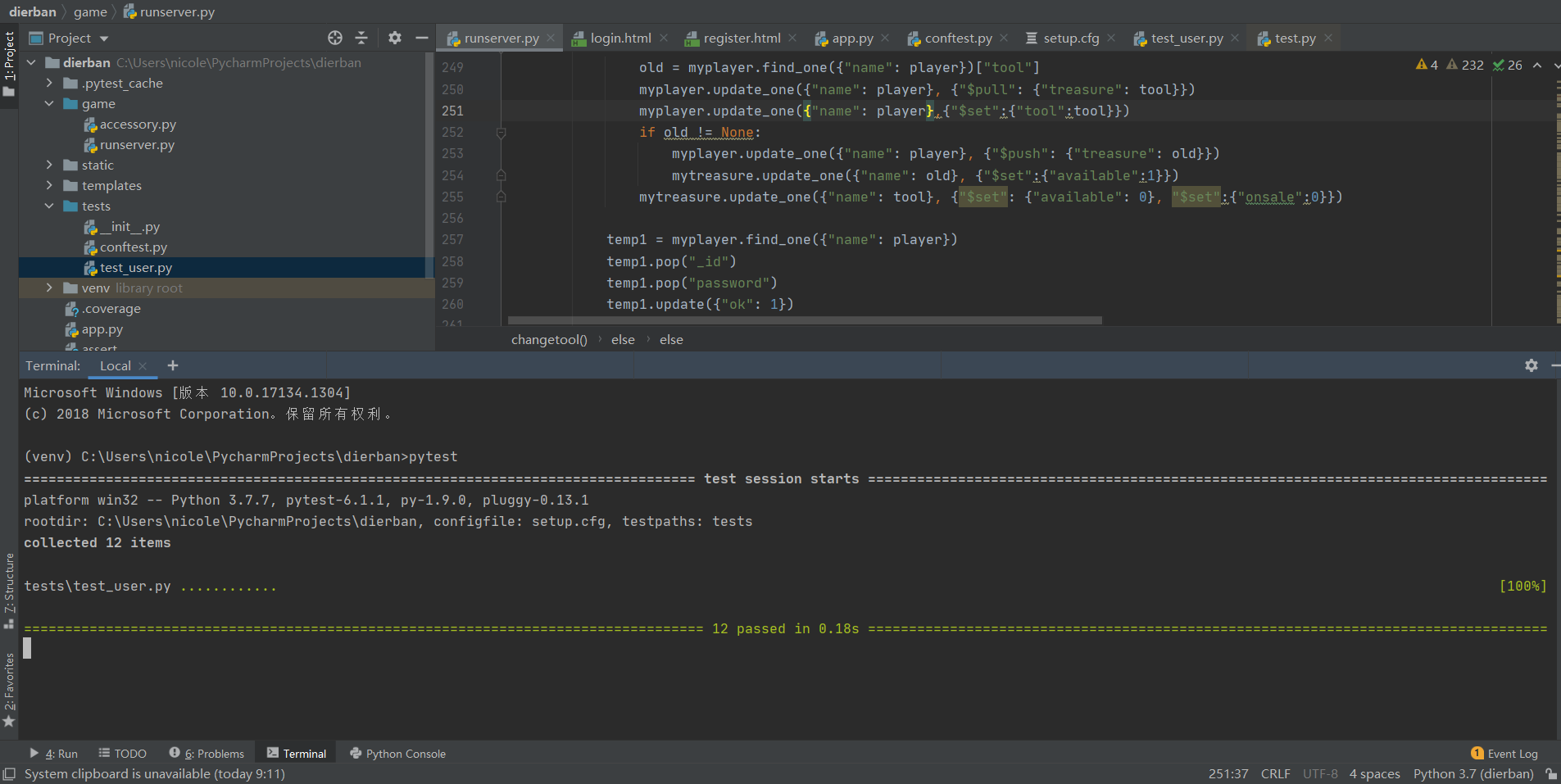
即可。原来需要的用户信息从session[“username”]中取得。

# 十、pytest测试

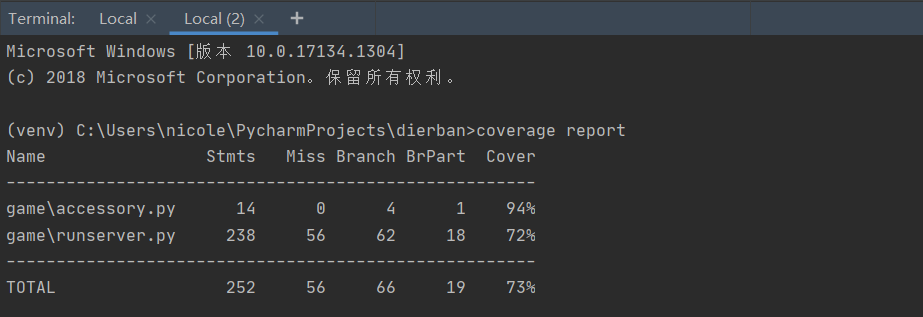
测试时需要post的部分放在文件开头的全局变量中，需要post不同的内容时修改这些变量：



上述参数下pytest测试结果如下图，12个测试结果全部通过：



覆盖率测试结果如下：

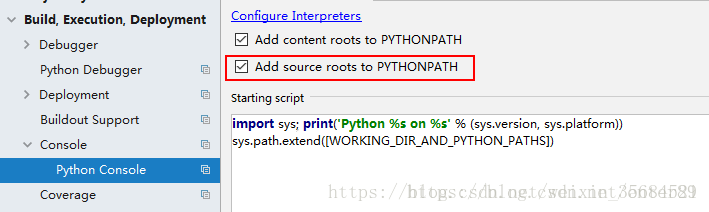


pytest测试过程中遇到的问题：在import时无法识别自己写的模块，即tests/test\_user无法导入我的游戏的模块进行pytest测试

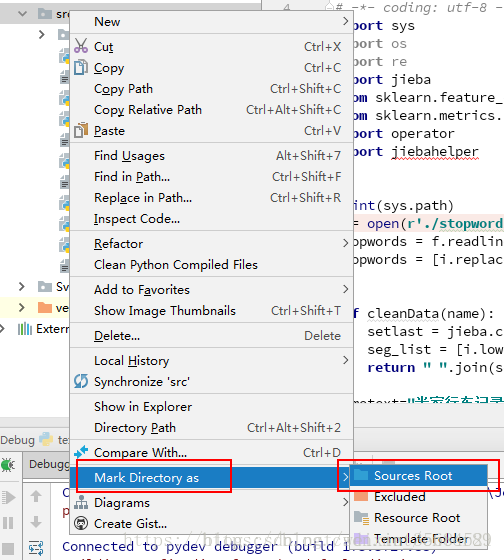
关于pycharm的这个问题网上有多种解决方案：

1.

(1). 打开File–> Setting—> 打开 Console下的Python Console，把选项（Add source roots to PYTHONPAT）点击勾选上

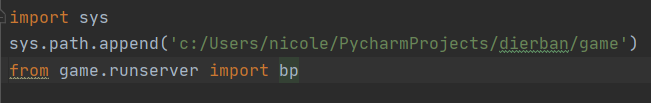


(2). 右键点击自己的工作空间文件夹，找到Mark Directory as 选择Source Root



按此方案执行后我的pycharm中的红线消失了，甚至可以有导入该模块的提示，但是仍然报错ModuleNotFoundError: No module named 'game'

2.尝试了sys.path.apppend



成功解决问题