华东师范大学数据学院数据库管理实验报告二

课程名称：数据库管理	**年级：2017**
指导教师：周煊	**姓名：熊双宇**

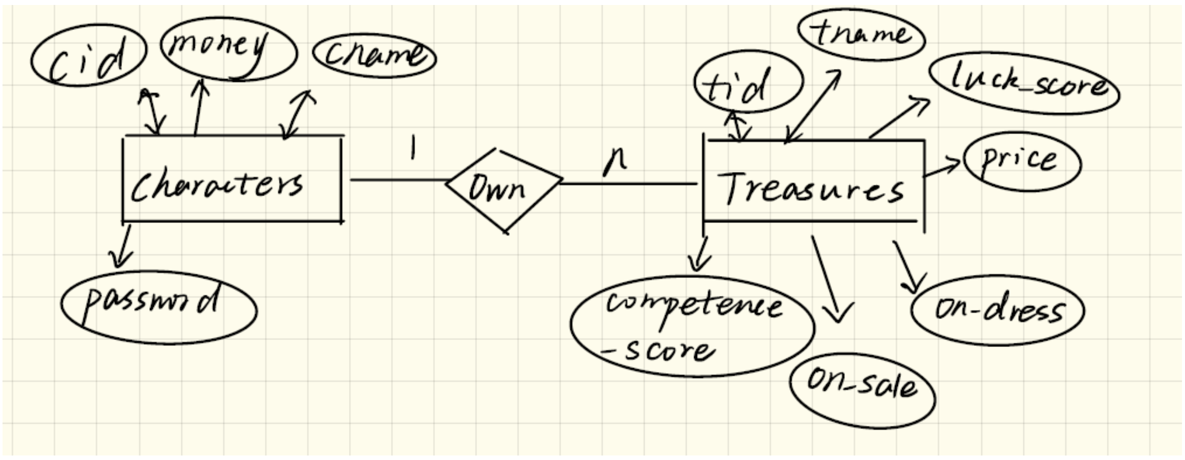
0. 环境

Packages version

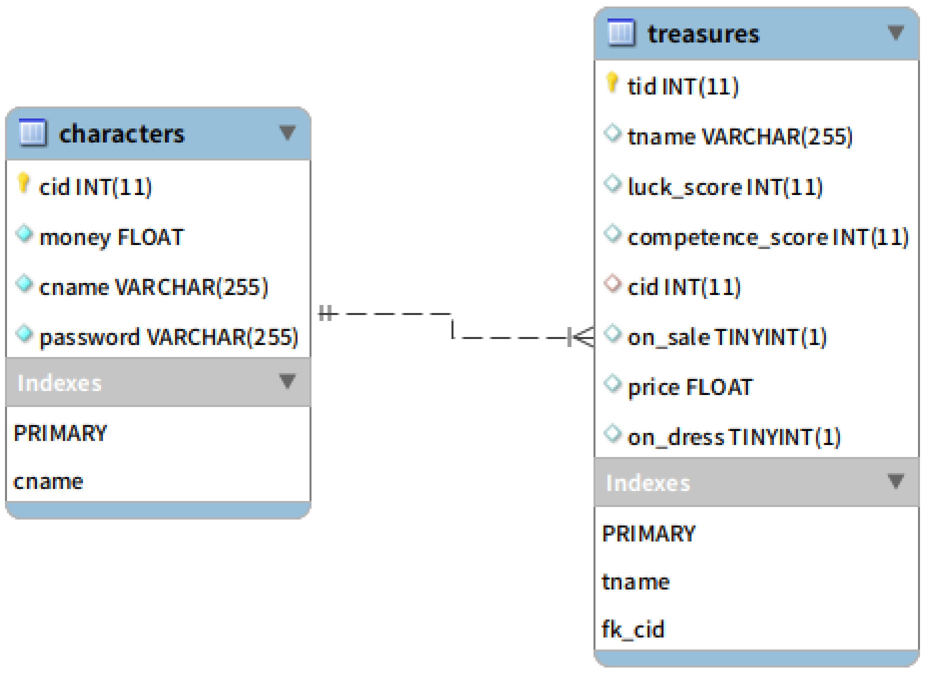
- APScheduler 3.6.3
- Flask 1.1.1
- Flask-APScheduler 1.11.0
- Flask-Bcrypt 0.7.1
- Flask-Login 0.4.1
- Flask-SQLAlchemy 2.4.1
- PyMySQL 0.9.3
- requests 2.22.0
- SQLAlchemy 1.3.11

1. 数据库设计

1.1 ER图



1.2 Model



1.2.1 Characters Schema

```
mysql> desc characters;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| cid      | int(11)      | NO   | PRI | NULL    | auto_increment |
| money    | float        | NO   |     | 0       |                |
| cname    | varchar(255) | NO   | UNI | NULL    |                |
| password | varchar(255) | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

- 说明:
  - cid: 角色ID [Primary Key, index];
  - money: 角色拥有财产额;
  - cname: 角色名称 [[Unique Key, index];
  - password: 角色密码 [用bcrypt得到用户密码的哈希值];

1.2.2 Treasures Schema

```
mysql> desc treasures;
+------------------+--------------+------+-----+---------+-------------
---+
| Field            | Type         | Null | Key | Default | Extra
   |
+------------------+--------------+------+-----+---------+-------------
---+
| tid              | int(11)      | NO   | PRI | NULL    |
auto_increment |
| tname            | varchar(255) | YES  |     | NULL    |
   |
| luck_score       | int(11)      | YES  |     | 0       |
   |
| competence_score | int(11)      | YES  |     | 0       |
   |
| cid                | int(11)       | YES  | MUL | NULL     |
   |
| on_sale            | tinyint(1)    | YES  |     | 0        |
   |
| price              | float         | YES  |     | 0        |
   |
| on_dress           | tinyint(1)    | YES  |     | 0        |
   |
+-------------------+---------------+-------+-----+---------+-------------
---+
8 rows in set (0.00 sec)
```

- 说明:
  - tid: 宝物ID [Primary Key, index];
  - tname: 宝物名称 [Unique Key, index];
  - luck_score: 用于计算角色幸运值
  - competence_score: 用于计算角色能力值
  - cid: 所属角色的ID [Foreign Key, index]
  - on_sale: 是否在售
    - True: 在售状态
    - False: 下架状态
  - price: 价格
    - 处于下架状态的宝物价格必须为0
  - on_dress: 是否被所属角色佩戴
    - True: 正佩戴
    - False: 未佩戴
- [注意]： 宝物的属性通过competence_score和luck_score来表示，即 competence_score 值为0的宝物为accessory(配饰)， luck_score值为0的宝物为tool(工具)， 这两个值不会同时为0，否则报错。

1.3 应用访问数据库的SQL指令

`from HuntGame import db`

1.3.1 Query

```
db.session.query(Class).filter(*criterion).all() # returns the results of a query as a list
db.session.query(Class).filter(*criterion).one_or_none() # return exactly one result or raise MultipleResultFound if multiple objects identities are returned.
db.session.query(Class).filter(*criterion).first() # return the first result of query or None if the result not contain any row
db.session.commit()
```

1.3.2 Delete

```
db.session.query(Treasures).filter(Treasures.tid==throw_t.tid).delete(synchronize_session=False)
db.commit()
```

```python
db.session.expunge(instance) # cascading will be applied according to the expunge rule
```

1.3.3 Alter

```python
# update
db.session.merge(instance)
db.commit()
```

```python
# insert
db.session.add(instance)
db.commit()
```

1.3.4 Rollback

```python
db.session.rollback()
```

2. JSON HTTP API

将介绍所有api对应的网络链接、输入格式和输出（包括成功执行的输出和检测到Error的输出）

[注意] 使用mysql实现的游戏API与mongodb实现的游戏API对比:

- 将部分函数拆分到Utils文件中
- 使用flask_login库, 改善登陆和登出功能, 详情见login API 'output' 部分
- 接口改动不大, 但由于Mysql+SQLAlchemy和Mongodb+pymongo的区别, 以及Schema的区别, 函数的具体语句改动大
- 将角色的**competence_score, luck_score, storage_box** 解除持久化, 每次需要使用时, 通过函数**count_competence, count_luck**计算得出, storage_box则转化为Treasures和Characters之间的relationshp

2.1 API format

```python
import requests
URL = 'http://127.0.0.1:5000'
```

2.1.1 API for register, log in and log out

Functions	url	input	methods	output
register	URL + '/register'	json: {'name': string, 'password': string}	GET/POST	1.successfully register and login, a character is inserted into ddatabase2.error because of duplicate 'name'
login	URL + '/login'	json: {'name': string, 'password': string}	GET/POST	1. successfully login: a cookie will be saved on the user's computer, and then Flask-Login will automatically restore the user ID from that cookie if it is not in the session, redirect to the '/home' . 2. error because of no existing

Functions	url	input	methods	output
				'name' in database or 'password' not match to name'
logout	URL + '/logout'	None	GET	1. successfully: session['username'] is popped, return '/login' page 2. if no users is logged in， redirect to the '/login' page
home	URL + '/home'	None	GET	1. if the user has logged in successfully, he/she will be redirected to this page, which present character's info

2.1.2 API for automatically hunting treasures and working

2.1.2.1 Config

```python
class Config(object):  # 创建配置，用类
    # 任务列表
    JOBS = [
        {  # 第1个任务，每隔15min执行一次
            'id': 'hunt_treasure',
            'func': '__main__:hunt_treasure',  # 方法名
            'args': (),  # 入参
            'trigger': 'interval',  # interval表示循环任务
            'seconds': 900,  # 15min
        },
        {  # 第2个任务，每隔15min执行一次
            'id': 'work_money',
            'func': '__main__:work_money',  # 方法名
            'args': (),  # 入参
            'trigger': 'interval',  # interval表示循环任务
            'seconds': 900,  # 15min
        }
    ]
app.config.from_object(Config())  # 为实例化的flask引入配置
```

2.1.2.2 hunt_treasure

Functions	config	algorithm
hunt _treasure	1. one user's storage box can have treasures less than **MAX_TREASURE(20)**; 2. one user hunt one treasure according to **luck_score** every **Time(15)** minutes.	None

```python
MAX_TREASURE = 20

def hunt_treasure():
    global MAX_TREASURE
    print(datetime.datetime.utcnow(), 'All characters auto-hunt for a treasure')

    for character in db.session.query(Characters).all():
        new_t_id = generate_treasure(character, character.competence_score
, character.luck_score )
        if count_treasure(character) >= MAX_TREASURE:
            throw_t = throw_treasure(character)
            if throw_t is None:
                print('Hunt failed: the character\'s storage box is full of
on-sale treasures')
                return 1
        # print("Successfully hunt a treasure")
    return 0
```

Output:

- Error:
  - if there are more than **MAX_TREASURE(20)** treasures with the character's **cid**, the treasure with lowest **luck_score** will be deleted from DataBase, and the new treasure's **cid** will be the the same as character's **cid**;
  - if character's **MAX_TREASURE(20)** treasures are all for sale , auto-hunt will fail and function will return 1 as error.
- Success: add a treasure with **cid** the same as character's **cid** , and print out "All characters successfully hunt a treasure"

2.1.2.3 work_money

Functions	config	algorithm
work_money	1. one user work for money according to **competence_score** every **Time(15)** minutes.	1. if user's **competence_score** <= 100, the user will get 101$; 2. else get (**competence_score**% 100)$ 3. **competence_score** is counted by the function **count_competence**

Output:

print 'All characters have worked one time!'

```python
def work_money():
    print(datetime.utcnow(), 'All characters auto-worked for money one time')

    for character in db.session.query(Characters).all():
        if  character.competence <= 100:
            money = 101
        else:
            money = count_competence(character)

        character.money += money % 100
    print('All characters have worked one time!')
    return 0
```

2.1.2.4 event listener

```python
def my_listener(event):
    if event.exception:
        print('The job crashed :(')
    else:
        print('The job worked :)')

scheduler.add_listener(my_listener, EVENT_JOB_EXECUTED | EVENT_JOB_ERROR)
```

2.1.3 API for dressing and taking off treasures

2.1.3.1 prerequisite

Treasure has one of the following properties:

- Tool: the treasure's **competence_score** > 0, and **luck_score** is 0;
- Accessory: the treasure's **luck_score** > 0, and **competence_score** is 0.

2.1.3.2 principle:

Every user can dress on limited number of treasures with different properties:

```python
MAX_TOOLS = 1 # a user can dress on at most one treasure with  'Tool'property
MAX_ACCESSORIES = 2 # a user can dress on at most two treasure with 'Accessory'property
```

2.1.3.3 Function

Function	url	input	methods
dress_treasure	URL + '/characters/dress/<task>'	1. json: {'name': string, 'treasure_name': list}; 2. task: on/off.	POST

Output:

- Error:
  - if the length of treasure list > 3, function will just return an error;
  - if treasures in list can't be found in database, an error will be returned;
- check property of treasure, if treasure isn't a tool or a accessory, return an error;

    - if treasure is not belong to user, it will not be dressed on.
- Success:
    - 'task' is 'on': change treasures' **on_dress** to **True**
    - If 'task' is 'off': change treasures' **on_dress** to **False**

2.1.4 API for looking through market

Function	url	input	methods	output
get_all_onsale_treasures	URL + '/market'	None	GET	information of all treasures with 'True' for the key **'on_sale'**

2.1.4 API for transaction, quotation and undo-quotation treasure

2.1.4.1 transaction

Function	url	input	methods
transaction_treasures('purchase')	URL + '/market/transaction/purchase'	json: {'name': string, 'treasure_name': string}	POST

Output:

- Error:
    - If user not exists in database, return {'Error': 'name is invalid!'}
    - If treasure not exits in database, return {'Error': 'treasure_name is invalid!'}
    - If treasure not for sale, return {'Error': 'treasure is not on_sale now!'}
    - If user can't afford treasure, return {'Error': 'you have money:{}, which is less than the price:{}'.format(character.money, treasure.price) }
- Success:
    - Update seller's and purchaser's profile:
        - **'money'**
    - Update treasure's information: change the **cid'**, set **'on_sale'** key as **False**, update the **'price'** as 0;
    - return the treasure information.

2.1.4.2 quotation

Function	url	input	methods
transaction_treasures('for_sale')	URL + '/market/transaction/for_sale'	json: {'name': string, 'treasure_name': string, 'price': float}	POST

Output:

- Error:
    - If treasure's **'on_sale'** is **True**, return {"Error":'treasure has been on_sale!'};
    - If treasure is not belong to user, return {'Error': 'treasure isn't in you storage_box!'}.
- Success:

    - treasure's information: set **'on_sale'** as **True**, change **'price'**
    - return the treasure information.

2.1.4.3 undo-quotation

Function	url	input	methods
transaction_treasures('off_sale')	URL + '/market/transaction/off_sale'	json: {'name': string, 'treasure_name': string}	POST

Output:

- Error:
    - If treasure is not for sale, return {'Error': 'the treasure has been off_sale!'};
    - If treasure is not belong to user, return {'Error': 'treasure does not belong to you!'}.
- Success:
    - treasure's information: set **'on_sale'** as **False**, change **'price'** back to 0;
    - return the treasure information.

3. 测试案例

3.1 测试注册角色

```
def test_register():
    '''
    1. username is universal time (to avoid duplicate name)
    2. password is 'test_register'
    3. use requests.post(url,json) to send request
    4. return the name and password
    '''
```

"注册角色"的测试被包含在"登陆角色"中

3.2 测试登陆角色

```
def test_login():
    '''
    1. First register as a user
    2. Then use requests.post(url,json) to login
    '''
```

Output:

```
First, register to create an existing account:
Response from the server:  Welcome! Please log in.
Then log in:
Response:  {
  "HomePage": [
    [
      {
        "competence_score": 0,
        "id": "482",
        "luck_score": 0,
        "money": 0.0,
        "name": "1574579677.0",
        "number of treasures": 0
      }
    ]
  ]
}

Finally log out.
Response:  Welcome! Please log in.
```

3.3 测试登出角色

```python
def test_logout():
    '''
    1. First register as a user
    2. Then log in the user
    3. Finally use requests.get(url) to log out
    '''
```

"登出角色"的测试被包含在"登陆角色"中

3.4测试穿脱宝物

```python
def test_dress_treasure(task, name = None, treasure_name_list  = None):
    '''
    1. If user is None, register and log in one
    2. Add a treasure to the user's storage_box(use the function: 
'add_treasure()' which will be described later)
    3. Test the dress function
    '''
```

Output:

- "on"

```
Response from the server:  Welcome! Please log in.
Response for adding a treasure:  {
  "Add treasure:": [
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure2019-11-24 15:15:41.508447",
      "on_sale": false,
      "owner_id": 483,
      "price": 0.0,
      "tid": 317
    }
  ]
}
```

```
Response for dressing a treasure:  {
  "Dress": [
    {
      "cid": 483,
      "cname": "1574579740.0",
      "competence_score": 3.0,
      "luck_score": 3,
      "off_dress": [],
      "on_dress": [
        "Test_treasure2019-11-24 15:15:41.508447"
      ]
    }
  ]
}

Test_treasure2019-11-24 15:15:41.508447
Treasure: 'tid:317','tname: Test_treasure2019-11-24 15:15:41.508447','price: 0.0',
 'luck_score:3','competence_score:0,
 'on_sale: False','on_dress: True'
 'cid: 483
```

We can see the treasure's **'cid'** is added to user's **'cid'**, and the **on_dress** is set as **True**!

- "off"

```
Response from the server:  Welcome! Please log in.
Response for adding a treasure:  {
  "Add treasure:": [
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure2019-11-24 15:16:47.976457",
      "on_sale": false,
      "owner_id": 484,
      "price": 0.0,
      "tid": 318
    }
  ]
}
```

```
Response for dressing a treasure:  {
  "Dress": [
    {
      "cid": 484,
      "cname": "1574579807.0",
      "competence_score": 0,
      "luck_score": 0,
      "off_dress": [
        "Test_treasure2019-11-24 15:16:47.976457"
      ],
      "on_dress": []
    }
  ]
}

Test_treasure2019-11-24 15:16:47.976457
Treasure: 'tid:318','tname: Test_treasure2019-11-24 15:16:47.976457','price: 0.0',
 'luck_score:3','competence_score:0,
 'on_sale: False','on_dress: False'
 'cid: 484
```

3.5 测试浏览市场

```python
def show_market():
    '''
    1. use requests.get(url)
    '''
```

Output:

```
Market:  {
  "All on-sale treasures": [
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure: 2019-11-21 14:41:32.605813",
      "on_sale": true,
      "price": 0.0
    },
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure: 2019-11-21 14:42:40.567244",
      "on_sale": true,
      "price": 0.0
    },
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure: 2019-11-21 14:45:07.554955",
      "on_sale": true,
      "price": 0.0
    },
```

All treasures' **'on_sale'** are True

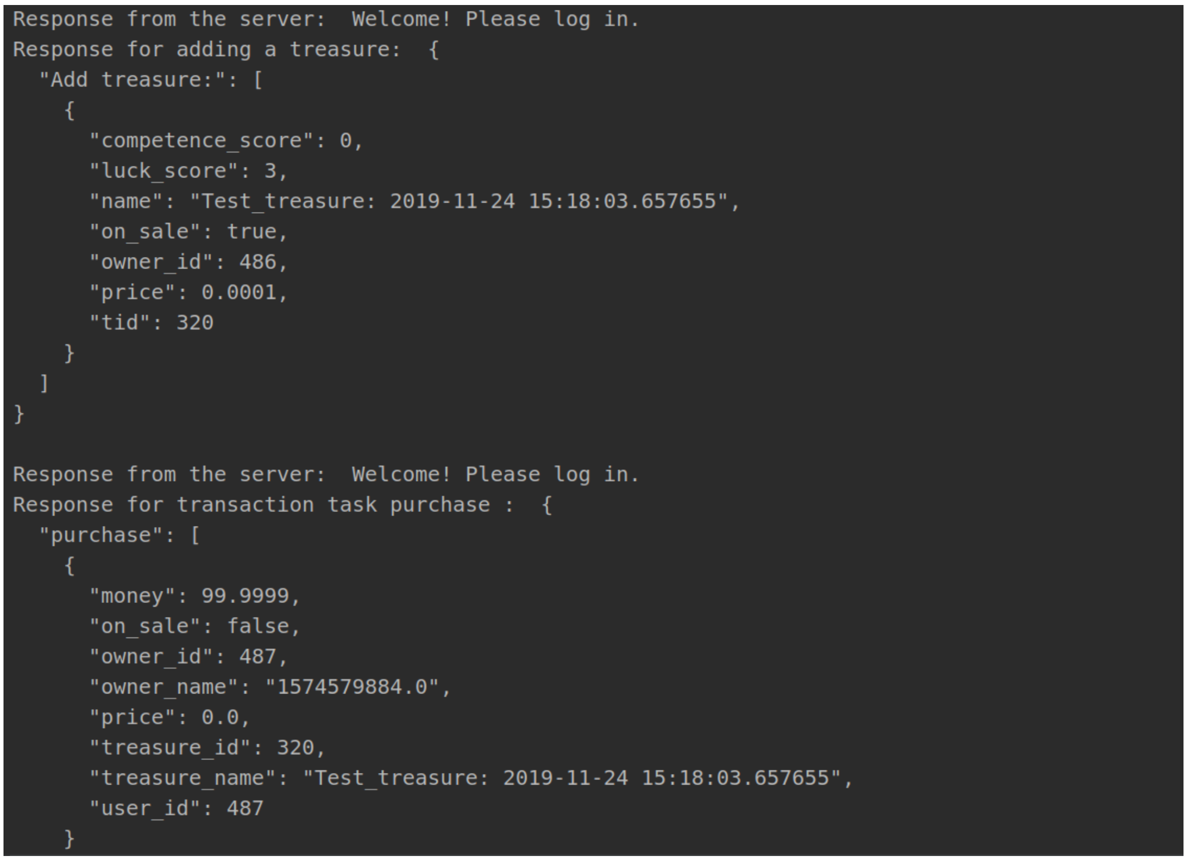
3.6 测试宝物交易、挂牌和收回

```python
def test_transaction(task, treasure_name=None, name=None, price=0):
    '''
    1. 交易：task='purchase'；挂牌：task='for_sale';收回：task='off_sale'
    2. If name is None, register and login a user
    3. If user has no treasure(treasure_name is None), add a treasure into the user's storage_box
    4. Use requests.post(url, json) to send a request
'''
```

3.6.1 宝物交易 "purchase"

- 创建新的角色, 其**money**值为100;
- 创建新的宝物, 其**price**值为0.0001

Output:



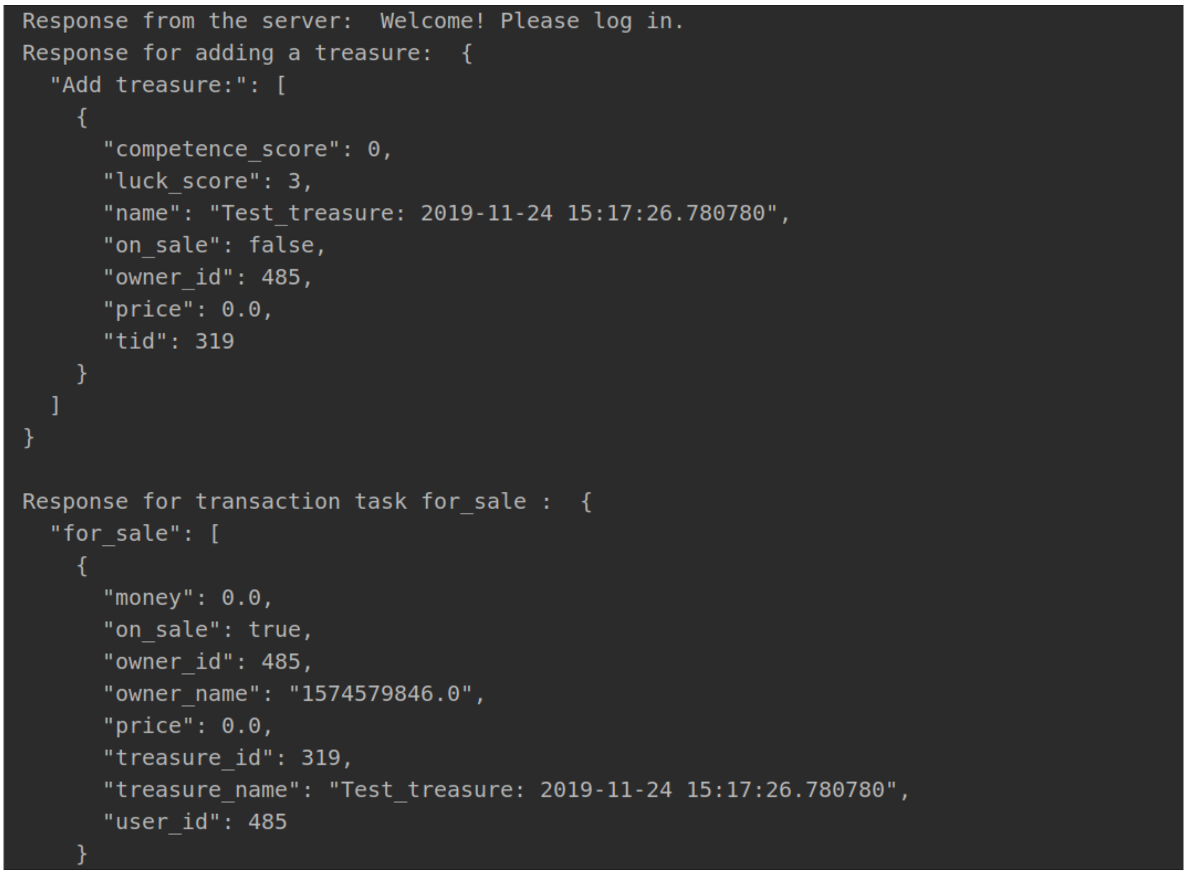
```
Response from the server:  Welcome! Please log in.
Response for adding a treasure:  {
  "Add treasure:": [
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure: 2019-11-24 15:18:03.657655",
      "on_sale": true,
      "owner_id": 486,
      "price": 0.0001,
      "tid": 320
    }
  ]
}

Response from the server:  Welcome! Please log in.
Response for transaction task purchase :  {
  "purchase": [
    {
      "money": 99.9999,
      "on_sale": false,
      "owner_id": 487,
      "owner_name": "1574579884.0",
      "price": 0.0,
      "treasure_id": 320,
      "treasure_name": "Test_treasure: 2019-11-24 15:18:03.657655",
      "user_id": 487
    }
```

After transaction:

- treasure's **'on_sale'** is changed to **False**
- treasure's **'price'** is changed from **0.0001** to **0**
- character's **'tid'** matches to treasure's **'tid'**
- character's **'money'** is decreased from **100** to **99.9999**

3.6.2 挂牌



```
Response from the server:  Welcome! Please log in.
Response for adding a treasure:  {
  "Add treasure:": [
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure: 2019-11-24 15:17:26.780780",
      "on_sale": false,
      "owner_id": 485,
      "price": 0.0,
      "tid": 319
    }
  ]
}

Response for transaction task for_sale :  {
  "for_sale": [
    {
      "money": 0.0,
      "on_sale": true,
      "owner_id": 485,
      "owner_name": "1574579846.0",
      "price": 0.0,
      "treasure_id": 319,
      "treasure_name": "Test_treasure: 2019-11-24 15:17:26.780780",
      "user_id": 485
    }
```

We can see:

- treasure's **'on_sale'** change from False to **True**;

3.6.3 收回

Output:

```
Response from the server:  Welcome! Please log in.
Response for adding a treasure:  {
  "Add treasure:": [
    {
      "competence_score": 0,
      "luck_score": 3,
      "name": "Test_treasure: 2019-11-24 15:18:56.642295",
      "on_sale": true,
      "owner_id": 488,
      "price": 0.0,
      "tid": 321
    }
  ]
}

Response for transaction task off_sale :  {
  "off_sale": [
    {
      "money": 0.0,
      "on_sale": false,
      "owner_id": 488,
      "owner_name": "1574579936.0",
      "price": 0.0,
      "treasure_id": 321,
      "treasure_name": "Test_treasure: 2019-11-24 15:18:56.642295",
      "user_id": 488
    }
```

After off_sale:

- The treasure's **'on_sale'** is changed to **False**
- **'price'** is changed to **0**

3.7 测试自动寻宝和工作

测试代码：

```python
def test_auto_jobs():
    with app.app_context():
        # scheduler.get_jobs()
        for job_id in scheduler.get_jobs():
            print(job_id)
            print(job_id.next_run_time)
```

输出结果：

```
hunt_treasure (trigger: interval[0:15:00], next run at: 2019-11-24 23:29:02 CST)
2019-11-24 23:29:02.383153+08:00
work_money (trigger: interval[0:15:00], next run at: 2019-11-24 23:29:02 CST)
2019-11-24 23:29:02.383385+08:00
```

4. 其他相关函数解释

input内:

- 没要求的**URL**的函数, 都放在文件**utils.py**下
- 要求了**URL**的函数, 都放在**routes.py**下

```python
URL = 'http://127.0.0.1/'
```

4.1 给数据库添加角色

Function	input	output
add_character(name, hashpass)	args: name(string), hashpass(string)	{'name': new_character['name'], 'date': new_character['date']}

- 该函数将在用户注册时被调用;
- 将用户注册和角色创建分开的初衷是想实现一个用户多个角色，后默认一个用户只有一个角色，即用户等价于角色；

4.2 给角色添加宝物

Function	url	input	method
add_treasure()	URL + '/treasure/add'	json: {'name': string, 'treasure_name':list, 'competence_score': int, 'luck_score':int}	POST

Output:

- Error:
    - if 'treasure_name' is not a list, return an error: {"Error: ": 'treasure_name {} is not a list'.format(treasure_name)}
    - if the number of treasures belonging to user with 'name' is equals to **MAX_TREASURE**, return an error: {'Error': 'The owner's storage is full of {} treasures!'.format(MAX_TREASURE)}
- Success:
    - Treasure's **cid** will be changed to user's **cid**

PS:

- 该函数将只测试案例中被调用，如给宝物挂牌时为了防止角色没有宝物，将先为其添加一件宝物;
- 该函数的初衷是允许角色按照自己的能力，将多件宝物合成出新的宝物给自己【未实现】；

4.3 计算角色能力和运气

4.3.1 能力

Function	input
count_competence(task, t_score, num)	task: 'on'/'off', t_score: int, c_score: int, num: int

```python
def count_competence(task, t_score, c_score, num):
    '''
    function will be called in 'dress_treasure'
    task: on/off (on: dress on a treasure, off: take off a treasure)
    t_score: the competence_score of treasure
    c_score: the competence_score of character
    num: the number of treasures characer has dressed on
    '''
```

该函数在角色穿脱宝物时调用；

Output:

- Success: return the competence score after dressing on or taking off a treasure;
- Error:
    - if the competence score > MAX_COMPETENCE_SCORE, return None as error;
    - if user want to take off a treasure, but he/she has dressed on no treasures, return None as error;

4.3.2 运气

与计算能力值函数同理

4.4 丢弃存储柜里的宝物

Condition: 在**'storage_box'**中挑选出运气值最低的的宝物

Function	input	output
throw_treasure(character)	None	the **tid** of treasure with lowest **'luck_score'**

- 该函数将在角色宝物数量达到**MAX_TREASURE**后调用
- 被丢弃的宝物在数据库中不会被删除，只是**'cid'** 被置为None，该设计的初衷是回收所有被丢弃的宝物给系统的宝物库，合成具有新的属性的宝物【'合成宝物'未实现】

Output:

- Error: Not found a treasure satisfied with the **Condition**， return {'Error':'Found throw_treasure with lowest luck!'}
- Success: return the id of treasure satisfying the **Condition**

4.5 给角色生成宝物

Function	input
generate_treasure(character, competence_score, luck_score)	args: 1. character: object; 2. competence_score: float; 3. luck_score: int

Output:

- Error:
    - if the character is None, return ({'Error': 'The character is None!'};
    - if the treasure generated has duplicate name with one treasure in database, return {'Error':'There exists two duplicated name in DB!' }
- Success: return the **cid** of the treasure generated;

Algorithm:

- 宝物的luck_score分为50个等级,competence_score有11个档次：

```python
treasures_luck_level = [num for num in range(50)]
treasures_competence_score = [0, 28, 70, 96, 120, 250, 345, 427, 513,
624, 718]
```

宝物名字的第一部分从以下列表中选出：

```python
treasure_names_list = ['烛龙', '帝江', '英招', '饕餮', '白泽', '应龙', '九尾狐', '青鸟', '狻猊', '穷奇', '凤凰']
```

- 给新的宝物设置属性
  - 名字：为了防止重名，采用这样的格式：'part1_part2_part3'
    - part1: 随机在treasure_names_list中选出;
    - part2: 角色名称;
    - part3: 世界时间

      ```python
      import datetime
      datetime.datetime.utcnow()
      ```

  - competence_score 和 luck_score:
    - 用分位数将’**treasures_luck_level**’划分为6部分，计算角色的'**luck_score**' 属于哪部分;
    - 然后分别在’**treasures_luck_level**‘和‘**treasures_competence_score**’中从最低级到比该部分更高一级中随机选择一个值，作为新的宝物的luck_score和competence_score
    - 由于宝物被划分为两种属性，即作为tool的宝物的luck_score 为0， 作为accessory的宝物的competence_score为0，所以对新的宝物的luck_score和competence_score值随机挑选出一个置为0，并且保证另一个不为0。

4.6 测试schedule函数

```python
def test_auto_jobs():
    from flask_apscheduler import APScheduler

    with app.app_context():
        for job_id in scheduler.get_jobs():
            print(job_id)
            print(job_id.next_run_time)
```

思路：打印出schedule.get_jobs中的所有job的下一次运行时间

初衷是查看下一次运行时间和本次运行时间的间隔是否是设置的时间间隔。