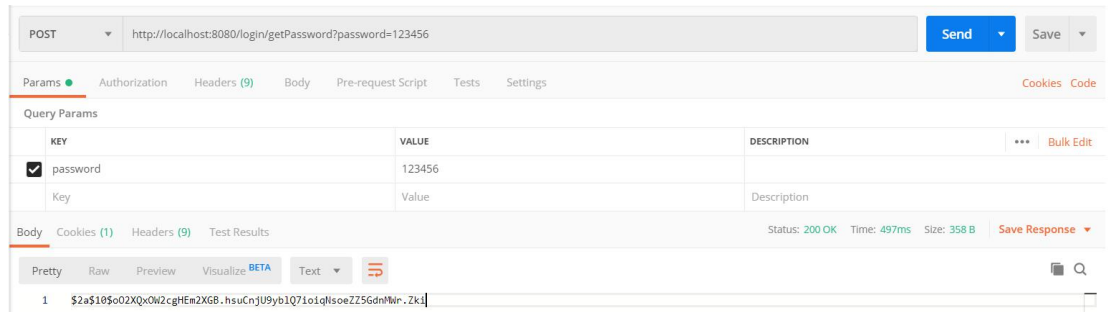


接口测试 — POSTMAN

准备工作：登陆 — 用于跳过安全验证

- 1、`http://localhost:8080/login/getPassword?password=123456`
获取加密代码



- 2、`http://localhost:8080/login/signIn?username=阎健&password=123456`
登录（进行安全验证：成功返回 0，失败返回 1）
成功



登录成功后记录 cookie 值：JSESSIONID=4FA6D256717F4082193DF29B22AA4F8B

之后的页面测试都需要在 Headers 中添加此 cookie 来跳过 security 安全验证

失败（数据库中没有此用户）：



一、Teacher Controller 接口测试

<http://localhost:8080/login/signIn?username=阎健&password=123456>

当前登录老师：阎健

1.老师查询自己已有课程

<http://localhost:8080/teacher/query/course>

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/teacher/query/course`. The response is a JSON array of two course objects. The first object has `courseId: 2`, `courseName: "篮球"`, `weekday: "周二"`, `startTime: "08:00"`, `endTime: "09:30"`, `gymId: 1`, `gymName: "中北大学生活动中心羽毛球馆"`, and `teacherName: "阎健"`. The second object has `courseId: 4`, `courseName: "排球"`, `weekday: "周五"`, `startTime: "10:00"`, `endTime: "11:30"`, `gymId: 2`, `gymName: "中北体育馆羽毛球馆"`, and `teacherName: "阎健"`.

```
1 {
2   {
3     "courseId": 2,
4     "courseName": "篮球",
5     "weekday": "周二",
6     "startTime": "08:00",
7     "endTime": "09:30",
8     "gymId": 1,
9     "gymName": "中北大学生活动中心羽毛球馆",
10    "teacherName": "阎健"
11  },
12  {
13    "courseId": 4,
14    "courseName": "排球",
15    "weekday": "周五",
16    "startTime": "10:00",
17    "endTime": "11:30",
18    "gymId": 2,
19    "gymName": "中北体育馆羽毛球馆",
20    "teacherName": "阎健"
21  }
22 }
```

2.老师添加课程

<http://localhost:8080/teacher/add/course?courseName=排球课&weekday=周四&startTime=10:00&endTime=11:30&gymid=1>

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/teacher/add/course?courseName=排球课&weekday=周四&startTime=10:00&endTime=11:30&gymid=1`. The response is a plain text message: "添加课程成功".

数据库中增加了相应课程

course_id	course_name	end_time	gym_id	start_time	teacher_name	weekday
1	乒乓球	11:30	1	10:00	钟晖	周一
2	篮球	09:30	1	08:00	阎健	周二
3	排球	09:30	2	08:00	潘国平	周二
4	排球	11:30	2	10:00	阎健	周五
5	排球课	11:30	1	10:00	阎健	周四

3. 老师删除自己的课程

<http://localhost:8080/teacher/delete/course?courseid=5>

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://localhost:8080/teacher/delete/course?courseid=5`
- Query Params:**

KEY	VALUE	DESCRIPTION
courseid	5	
- Status:** 200 OK, Time: 429ms, Size: 316 B
- Response Body:** 删除课程成功

数据库中对对应课程删除

course_id	course_name	end_time	gym_id	start_time	teacher_name	weekday
1	乒乓球	11:30	1	10:00	钟晖	周一
2	篮球	09:30	1	08:00	阎健	周二
3	排球	09:30	2	08:00	潘国平	周二
4	排球	11:30	2	10:00	阎健	周五

case:

当前老师删除不是自己教课的课程

<http://localhost:8080/teacher/delete/course?courseid=3>

(课程 id 为 3 的课程非当前老师教授)

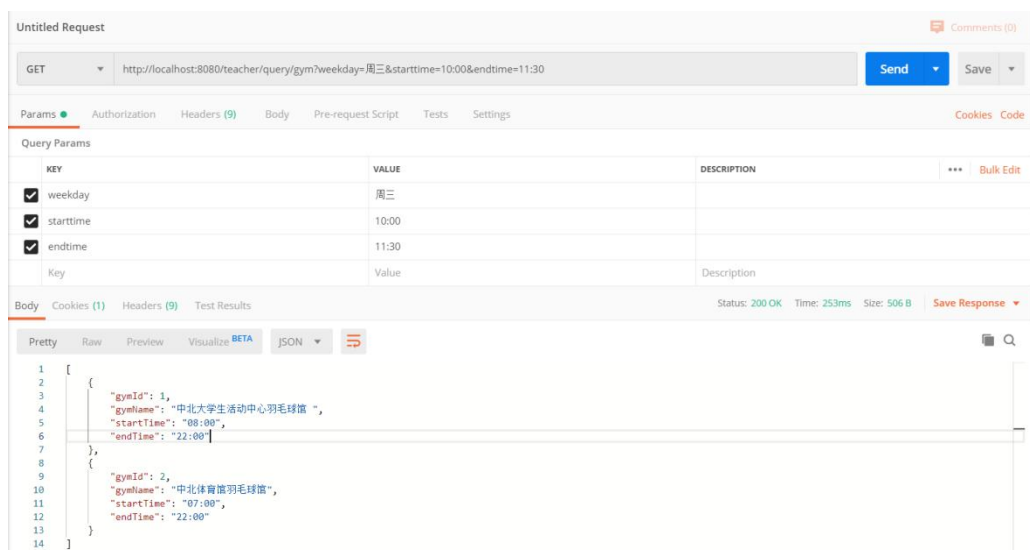
The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://localhost:8080/teacher/delete/course?courseid=3`
- Query Params:**

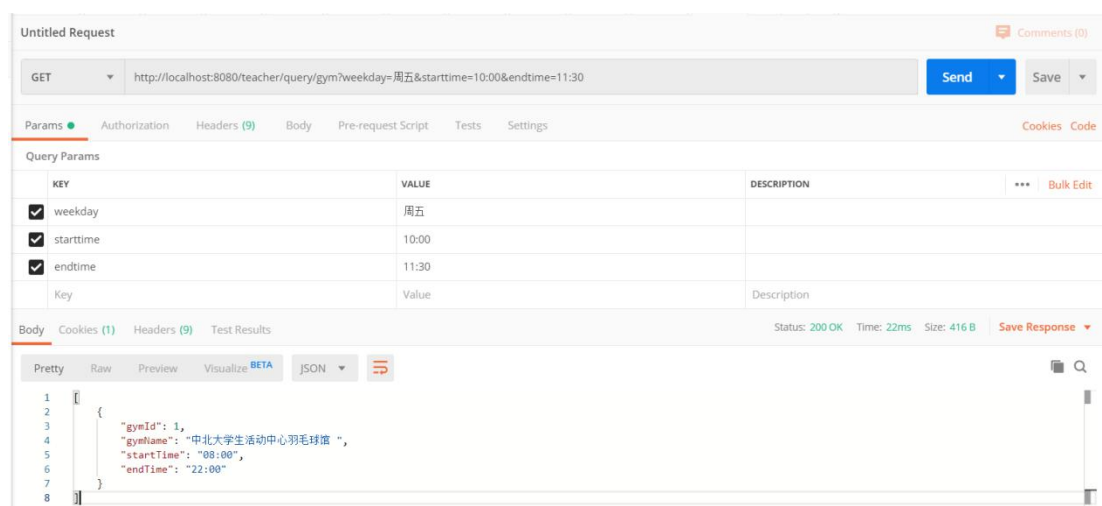
KEY	VALUE	DESCRIPTION
courseid	3	
- Status:** 200 OK, Time: 27ms, Size: 316 B
- Response Body:** 没有删除权限

4. 老师查询空余场馆

<http://localhost:8080/teacher/query/gym?weekday=周三&starttime=10:00&endtime=11:30>



<http://localhost:8080/teacher/query/gym?weekday=周五&starttime=10:00&endtime=11:30>



二、User Controller 接口测试

1. 用户查询个人信息

<http://localhost:8080/user/query/user>

用户 1: 阎健; 身份: ROLE_TEACHER

GET http://localhost:8080/user/query/user Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	*** Bulk Edit
Key	Value	Description	

Body Cookies (1) Headers (9) Test Results Status: 200 OK Time: 23ms Size: 627 B Save Response

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "userId": 2,
3   "userName": "闵健",
4   "password": "$2a$10$o2XQxOM2cgHEm2XGB.hsuCnjU9yb1Q7ioiqlNsoeZ75GdnMn.Zki",
5   "name": "Jack",
6   "roleId": 3,
7   "roleName": "ROLE_TEACHER",
8   "enabled": true,
9   "username": "闵健",
10  "accountNonLocked": true,
11  "authorities": [
12    {
13      "authority": "ROLE_TEACHER"
14    }
15  ],
16  "credentialsNonExpired": true,
17  "accountNonExpired": true
18 }
```

用户 2: 张诗晨; 身份: ROLE_STUDENT

GET http://localhost:8080/user/query/user Send Save

Headers (1)

KEY	VALUE	DESCRIPTION	*** Bulk Edit Presets
<input checked="" type="checkbox"/> Cookie	JSESSIONID=1A08B19FD30C04BA7D080CEC5F864255		
Key	Value	Description	

Temporary Headers (8)

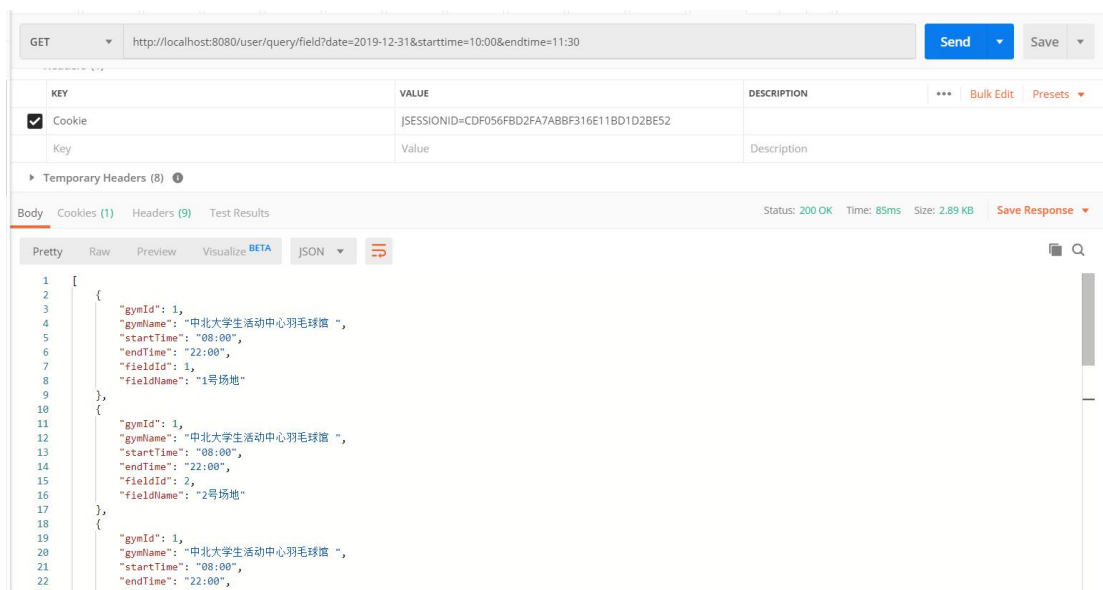
Body Cookies (1) Headers (9) Test Results Status: 200 OK Time: 16ms Size: 632 B Save Response

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "userId": 4,
3   "userName": "张诗晨",
4   "password": "$2a$10$o2XQxOM2cgHEm2XGB.hsuCnjU9yb1Q7ioiqlNsoeZ75GdnMn.Zki",
5   "name": "Amy",
6   "roleId": 2,
7   "roleName": "ROLE_STUDENT",
8   "enabled": true,
9   "username": "张诗晨",
10  "accountNonLocked": true,
11  "authorities": [
12    {
13      "authority": "ROLE_STUDENT"
14    }
15  ],
16  "credentialsNonExpired": true,
17  "accountNonExpired": true
18 }
```

2. 用户查询想要预定时间内空余的场馆

<http://localhost:8080/user/query/field?date=2019-12-31&starttime=10:00&endtime=11:30>



<http://localhost:8080/user/query/field?date=2019-12-24&starttime=13:00&endtime=15:00>

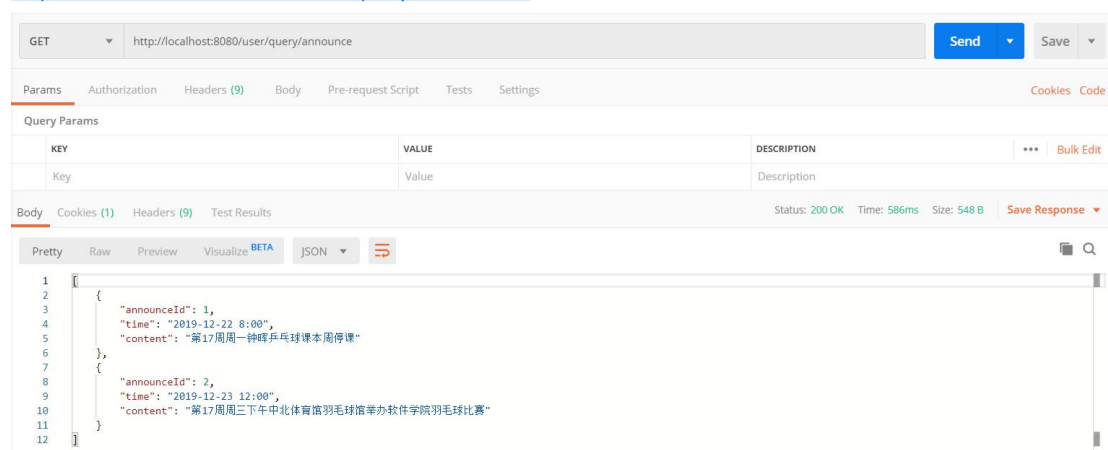
在数据库中，该时间段内 2 号场馆的 2 号场地被预定

reserve_id	date	end_time	field_id	gym_id	start_time	user_id
1	2019-12-23	12:00	1	2	08:00	3
2	2019-12-23	15:00	1	2	13:00	5
3	2019-12-24	11:00	1	1	09:30	7
4	2019-12-24	15:00	2	2	13:00	5

查询结果中没有该场馆

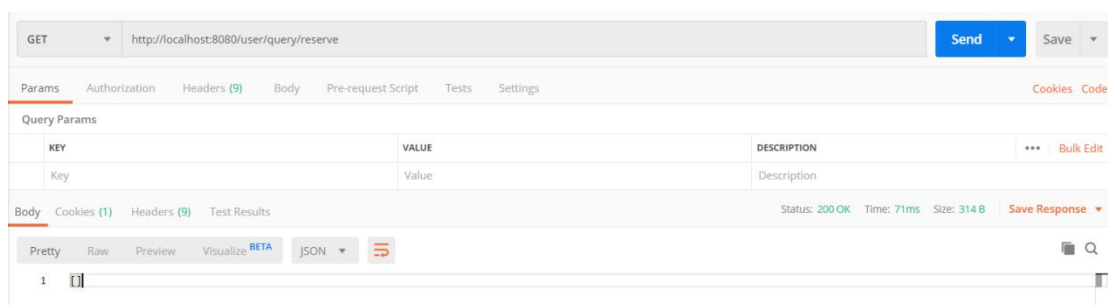
3. 用户查询管理员发布的通知

<http://localhost:8080/user/query/announce>

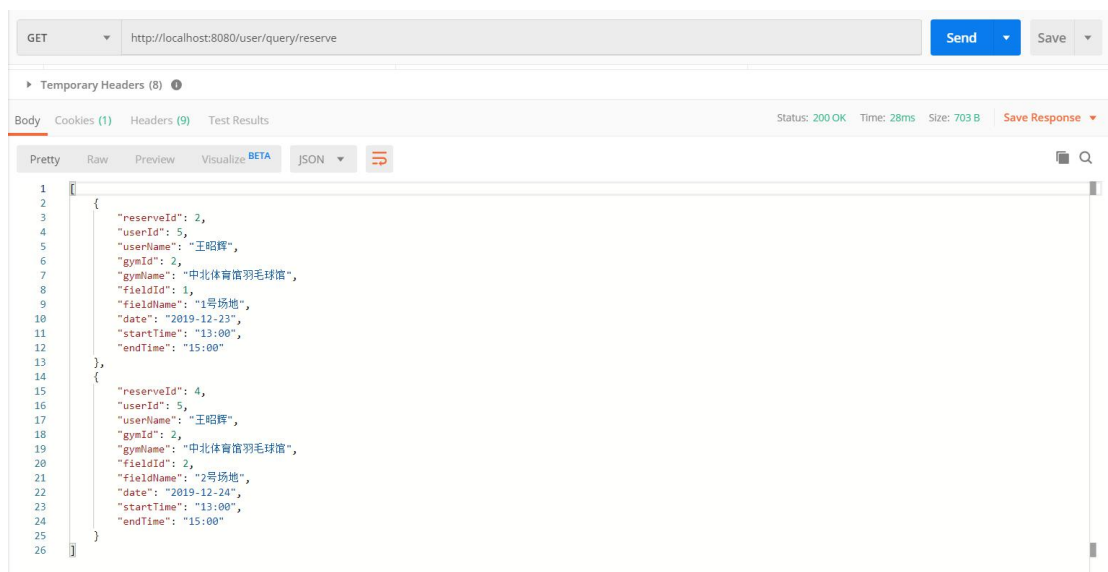


4. 用户查询自己的预定

<http://localhost:8080/user/query/reserve>



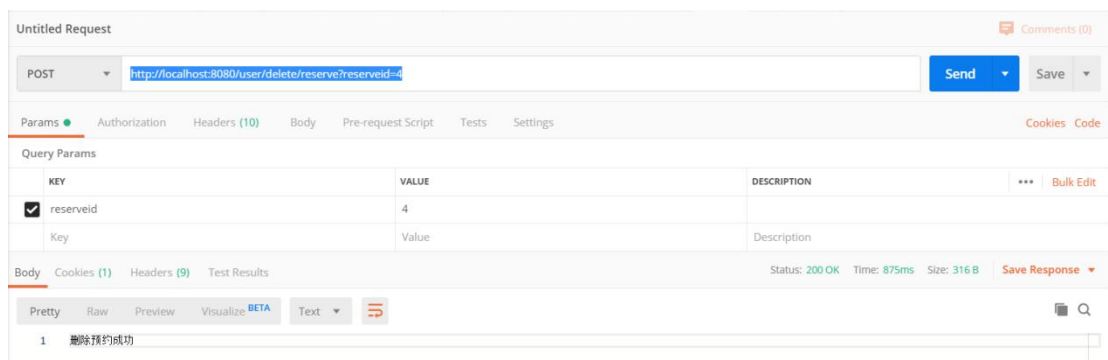
当前用户没有预定



该用户存在预定信息

5. 用户删除预定

<http://localhost:8080/user/delete/reserve?reserveid=4>



- 1、[http://localhost:8080/user/add/reserve?gymname=中北大学生活动中心羽毛球馆](http://localhost:8080/user/add/reserve?gymname=中北大学生活动中心羽毛球馆&gymid=1&fieldname=6号场地&fieldid=6&date=2019-12-31&starttime=16:00&endtime=18:00)
[&gymid=1&fieldname=6号场地&fieldid=6&date=2019-12-31&starttime=16:00&endtime=18:00](http://localhost:8080/user/add/reserve?gymname=中北大学生活动中心羽毛球馆&gymid=1&fieldname=6号场地&fieldid=6&date=2019-12-31&starttime=16:00&endtime=18:00)
 用户预约场地



三、Student Controller 接口测试

注：登陆全部以（王昭辉，123456）的学生身份登陆，student_id = 5。

1. 学生查询全部课程

测试 <http://localhost:8080/student/query/allcourse>

a) 未登陆

在未登陆的情况下，无论使用 GET 还是 POST，界面都会跳转到/login, 由于此时没有相应配置，所以返回 404。

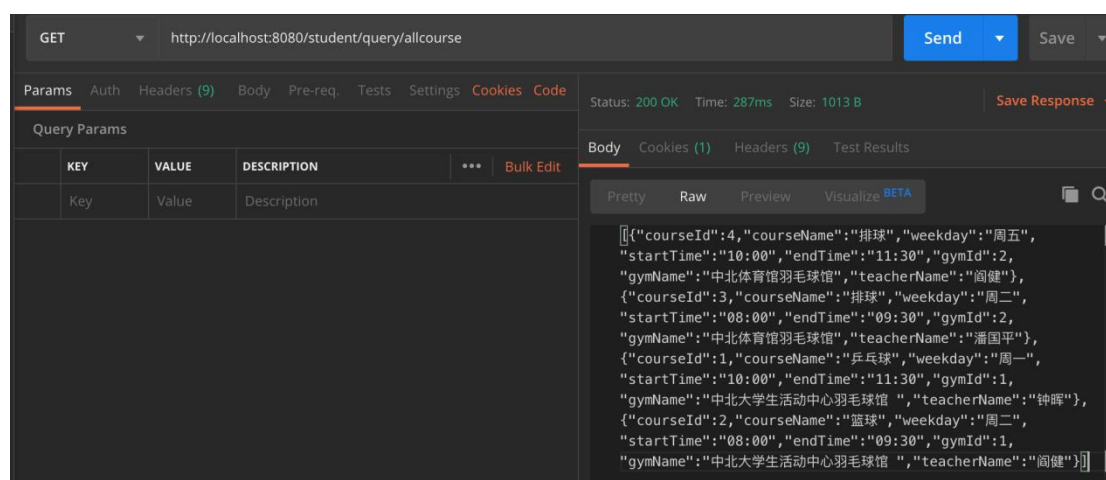
b) 已登陆

通过和数据库内容对比，可以看出来测试结果正确。

```
mysql> select * from course;
```

course_id	course_name	end_time	gym_id	start_time	teacher_name	weekday
4	排球	11:30	2	10:00	阎健	周五
3	排球	09:30	2	08:00	潘国平	周二
1	乒乓球	11:30	1	10:00	钟晖	周一
2	篮球	09:30	1	08:00	阎健	周二

4 rows in set (0.01 sec)



2. 学生查询本人课程

测试 <http://localhost:8080/student/query/mycourse>

c) 未登陆

同上

d) 已登陆

通过和数据库内容对比，可以看出来测试结果正确。

```
[mysql> select * from take;
+-----+-----+
| user_id | course_id |
+-----+-----+
|      4 |         1 |
|      4 |         2 |
|      5 |         2 |
|      6 |         3 |
+-----+-----+
4 rows in set (0.00 sec)
```

```
Pretty Raw Preview Visualize BETA JSON
[
  {
    "courseId": 2,
    "courseName": "篮球",
    "weekday": "周二",
    "startTime": "08:00",
    "endTime": "09:30",
    "gymId": 1,
    "gymName": "中北大学生活动中心羽毛球馆 ",
    "teacherName": "阎健"
  }
]
```

3. 学生添加课程（选课）

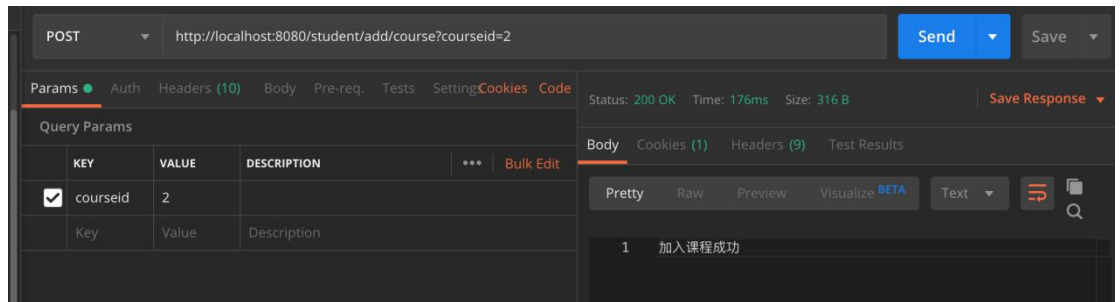
测试 <http://localhost:8080/student/add/course?courseid=1>

e) 未登陆

同上

f) 已登陆

f.1 尝试加入已经加入过的课程 2（成功 — 可能有缺陷？此处应提示不要加入重复课程）

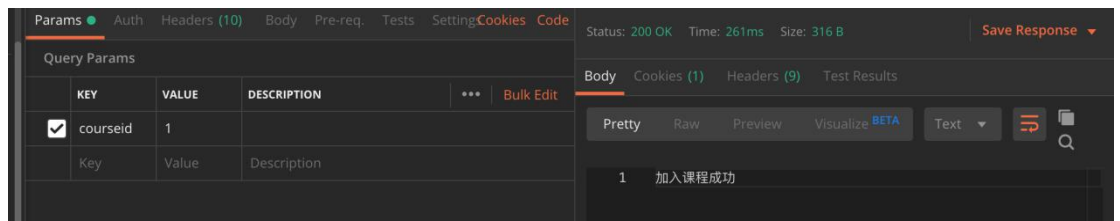


观察数据库, take 表并没有更新, 不影响程序正确性。

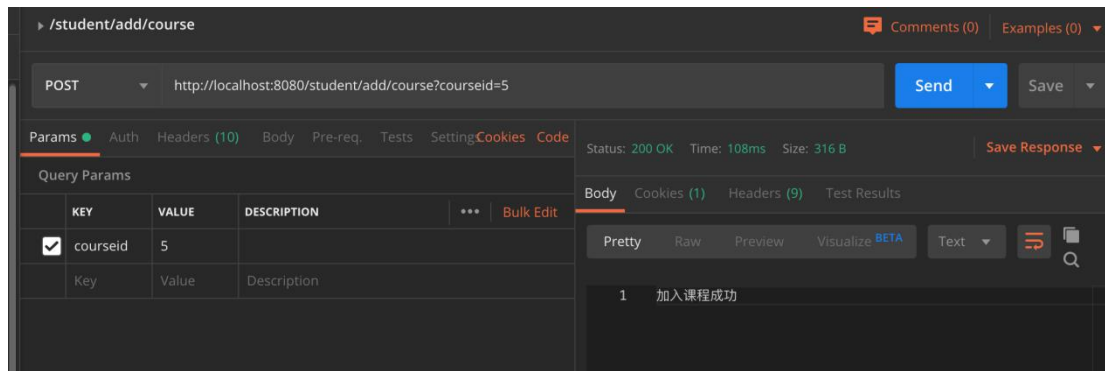
```
mysql> select * from take;
+-----+-----+
| user_id | course_id |
+-----+-----+
| 4       | 1       |
| 4       | 2       |
| 5       | 2       |
| 6       | 3       |
+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from take;
+-----+-----+
| user_id | course_id |
+-----+-----+
| 4       | 1       |
| 4       | 2       |
| 5       | 2       |
| 6       | 3       |
+-----+-----+
4 rows in set (0.00 sec)
```

f.2 尝试加入未加入过、且已经存在的课程 1（HAPPY PATH: 成功）



f.3 尝试加入不存在的课程 5（加入成功！可能造成缺陷！需要前端的限制。）



4. 学生删除课程（退课）

测试 <http://localhost:8080/student/delete/course?courseid=1>

g) 未登陆

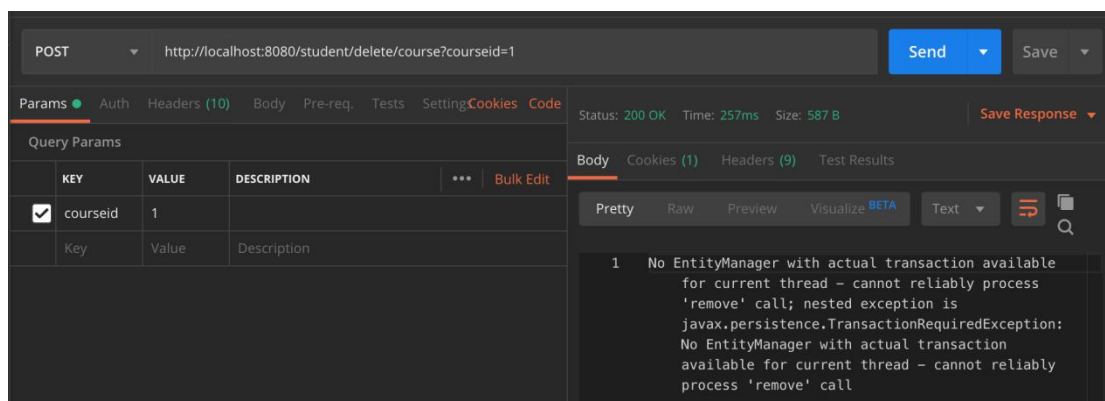
同上

h) 已登陆

h.1 删除已加入的课程 1（尝试失败，此处可能有 BUG 观察了打印出的 sql 语句，并没有执行 delete 语句 - 已解决）

原因：这个异常可能是由于在需要事务的方法上，没有开启事务，结果就操作需要事务的方法比如保存，修改数据库数据方法。

解决方法：对 service 层的删除方法加上@Transactional 的注解。



```
@RequestMapping(
    value = {"/delete/course"},
    method = {RequestMethod.POST}
)
public String quitCourse(@RequestParam(value = "courseid", required = true) int courseid) {
    UserDetails userDetails = (UserDetails)SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    String username = userDetails.getUsername();

    try {
        this.courseService.deleteCourseByStudent(username, courseid);
        return "删除课程成功";
    } catch (Exception var5) {
        return var5.getMessage();
    }
}
```

```

@Transactional
public void deleteCourseByStudent(String name, int course_id) throws Exception {
    UserAndRole user = this.userDao.findUserAndRoleByUserName(name);
    if (user == null) {
        throw new Exception("用户不存在");
    } else {
        int t = this.takeDao.deleteByUserIdAndCourseId(user.getUserId(), course_id);
        if (t <= 0) {
            throw new Exception("数据库异常");
        }
    }
}

```

再次执行，返回删除成功的提示，数据库中数据也相应更新成功。

POST http://localhost:8080/student/delete/course?courseid=1

Status: 200 OK Time: 1143ms Size: 316 B

Query Params:

KEY	VALUE	DESCRIPTION
courseid	1	

Body: 1 删除课程成功

```

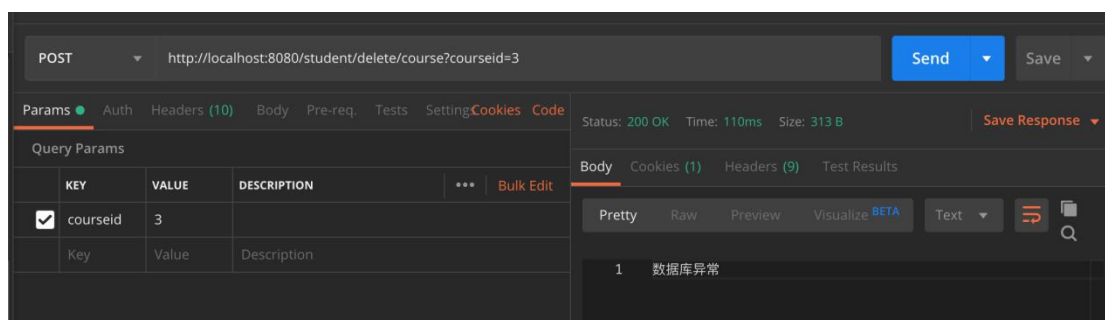
[mysql> select * from take;
+-----+-----+
| user_id | course_id |
+-----+-----+
| 4 | 1 |
| 4 | 2 |
| 5 | 1 |
| 5 | 2 |
| 6 | 3 |
+-----+-----+
5 rows in set (0.01 sec)

[mysql> select * from take;
+-----+-----+
| user_id | course_id |
+-----+-----+
| 4 | 1 |
| 4 | 2 |
| 5 | 2 |
| 6 | 3 |
+-----+-----+
4 rows in set (0.01 sec)

```

h.2 删除未加入的课程3（返回数据库异常，或者应该换个提示信息——此处应
该确保要删除的条目都是数据库中的有）

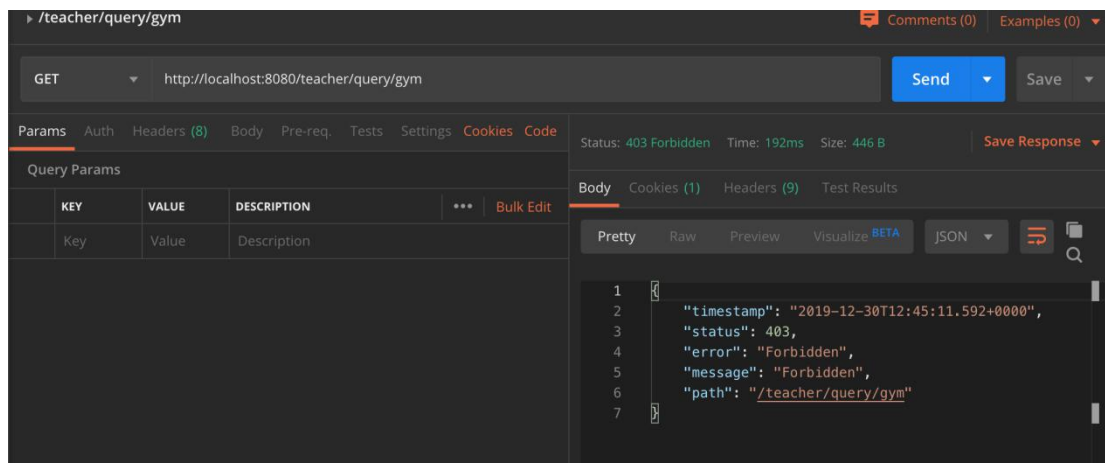
```
[mysql> select * from take;
+-----+-----+
| user_id | course_id |
+-----+-----+
|      4 |         1 |
|      4 |         2 |
|      5 |         2 |
|      6 |         3 |
+-----+-----+
4 rows in set (0.02 sec)
```



5. 其他

测试 <http://localhost:8080/teacher/query/gym>

学生没有权限查询场馆 gym，访问被禁止。



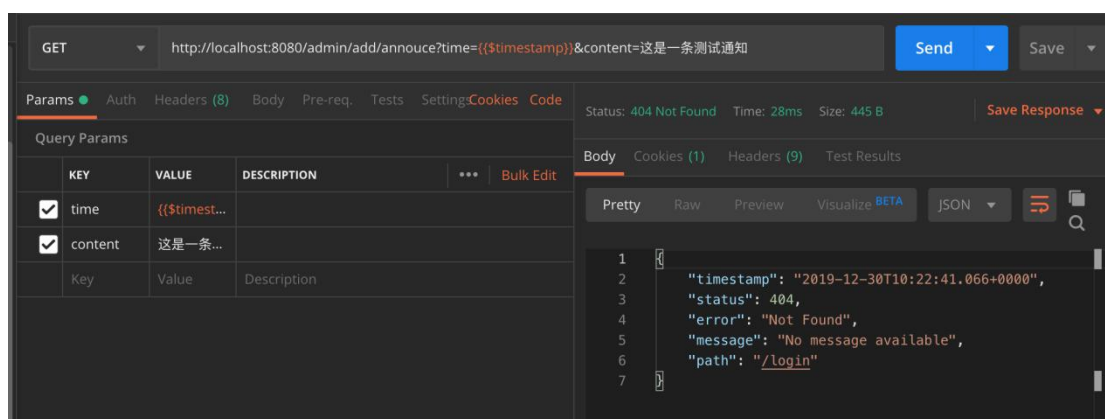
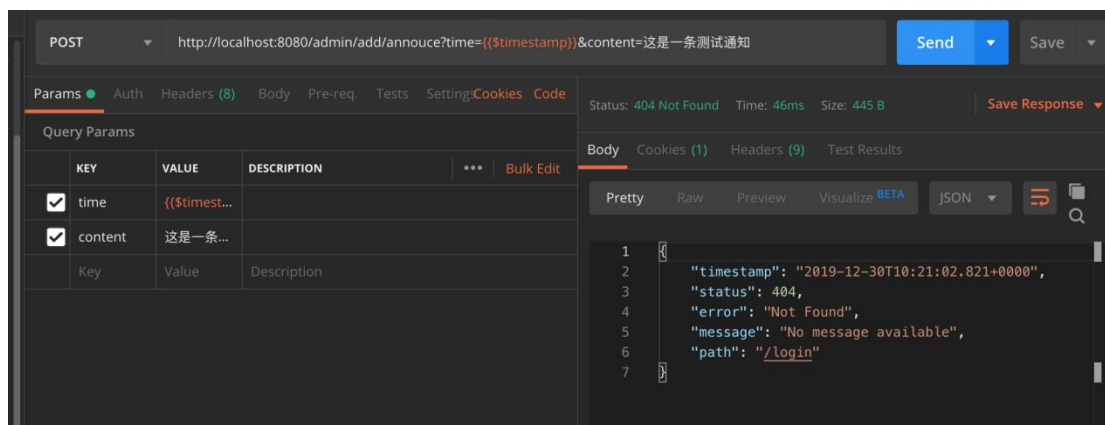
四、Admin Controller 接口测试

1. 在未登陆的情况下：

<http://localhost:8080/admin/add/announce?time={{timestamp}}&content=这是一条测试通>

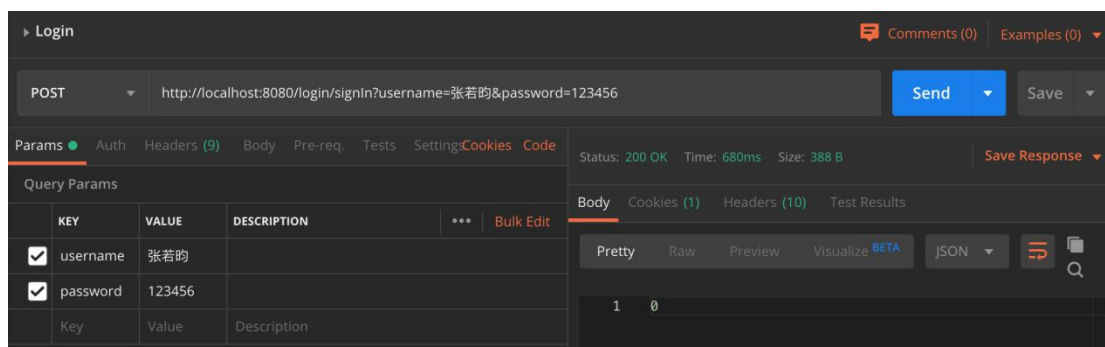
知

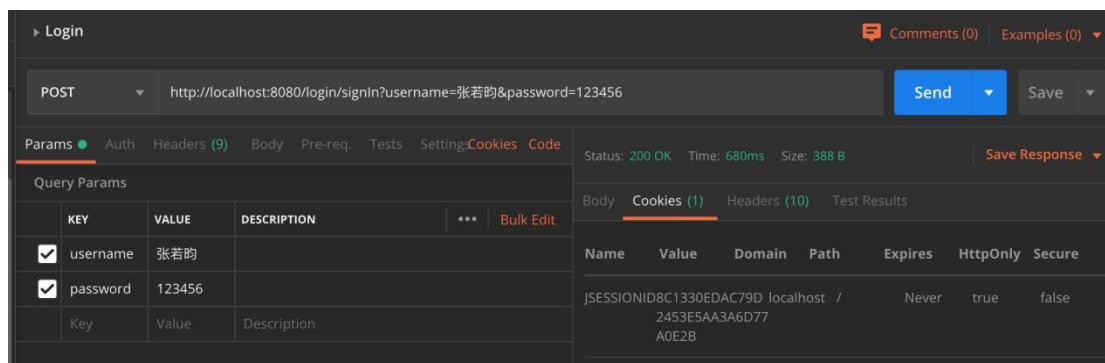
在未登陆的情况下，无论使用 GET 还是 POST，界面都会跳转到/login, 由于此时没有相应配置，所以返回 404。



2. 在已经登陆的情况下：测试发布通知功能

2.1 以管理员账号（张若昀，123456）登陆（返回 0 表示成功登陆，如下图 1），并获取到 JSESSION（如下图 2）。

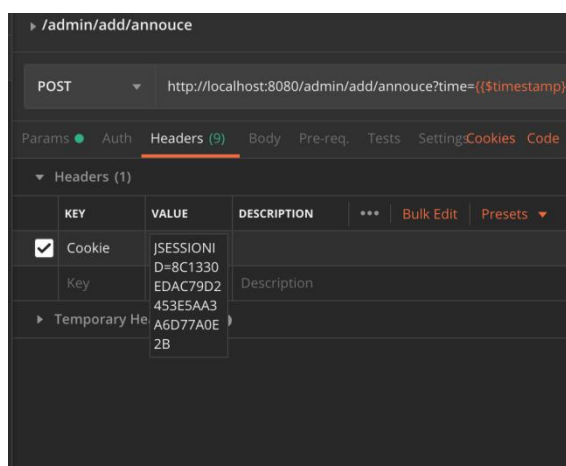




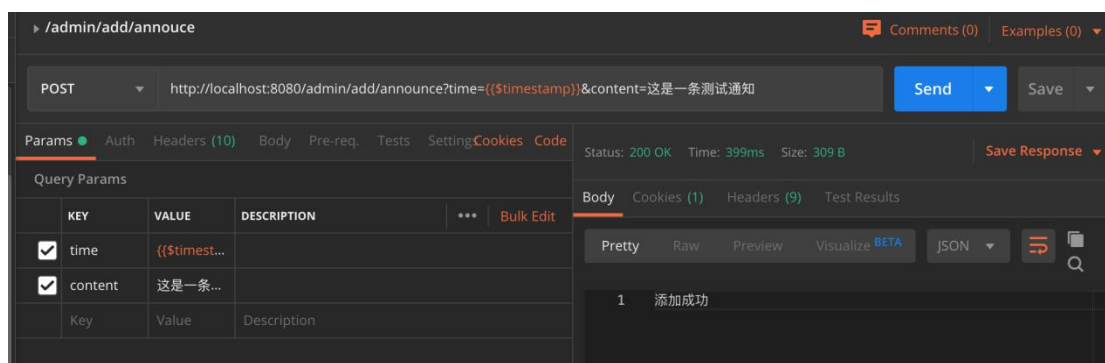
2.2

<http://localhost:8080/admin/add/announce?time={{timestamp}}&content=这是一条测试通知>

在 Headers 配置上刚刚获取的 JSESSIONID, 即可以管理员身份访 /admin//add/announce。



执行成功后, 获得返回数据-添加成功。



对比请求执行前后数据库的变化:

显然, 这里有两个问题: 1. 数据的 ID 不对 2. 数据的时间属性格式不对

问题 1: 数据的 ID 不对

原因: 数据库表 announce 没有自增 (constraint.sql 为所有表添加自增, 但是 announce 写错了, 所以这个表没有添加成功)

解决: 给 announce 加上自增即可。

问题 2: 数据的时间属性格式不对 (一个可能的缺陷)

原因: 系统没有对传入的时间参数做形式上的限制。

这个限制可以在前端/CONTROLLER/SERVICE 做。如果没有任何一层做, 那这里就是一

个缺陷。

```
mysql> select * from announce;
```

announce_id	content	time
1	第17周一钟晖乒乓球课本周停课	2019-12-22 8:00
2	第17周周三下午中北体育馆羽毛球馆举办软件学院羽毛球比赛	2019-12-23 12:00

2 rows in set (0.01 sec)

```
mysql> select * from announce;
```

announce_id	content	time
1	第17周一钟晖乒乓球课本周停课	2019-12-22 8:00
2	第17周周三下午中北体育馆羽毛球馆举办软件学院羽毛球比赛	2019-12-23 12:00
0	这是一条测试通知	1577701554

3 rows in set (0.01 sec)

2.3

<http://localhost:8080/admin/add/announce?time={{timestamp}}&content=这是一条测试通知>

由于问题 1 的出现，这次执行的时候，添加通知失败。

（原因是程序中实体类主键没有标注自增，jpa 不会为保存的实体生成主键，但是数据库中又没有添加主键自增，所以保存的实体 id 全为 0。这样就不可能插入第二条 announce_id=0 的数据。）

```
Hibernate: insert into announce (content, time, announce_id) values (?, ?, ?)
2019-12-30 18:26:35.270 WARN 16384 --- [nio-8080-exec-3] o.h.engine.jdbc.spi.SqlExceptionHelper : SQL Error: 1062, SQLState: 23000
2019-12-30 18:26:35.271 ERROR 16384 --- [nio-8080-exec-3] o.h.engine.jdbc.spi.SqlExceptionHelper : Duplicate entry '0' for key 'PRIMARY'
2019-12-30 18:26:35.294 ERROR 16384 --- [nio-8080-exec-3] o.h.i.ExceptionMapperStandardImpl : HHH000346: Error during managed flush [org.hibernate.exception.ConstraintViolationException: could not execute statement]
```

POST http://localhost:8080/admin/add/announce?time={{timestamp}}&content=这是一条测试通知

Status: 200 OK Time: 108ms Size: 466 B

Query Params

KEY	VALUE	DESCRIPTION
time	{{timestamp}}	
content	这是一条测试通知	

Body

```
1 could not execute statement; SQL [n/a
2 ]; constraint [PRIMARY
3 ]; nested exception is
   org.hibernate.exception.ConstraintViolationException: could not execute statement
```

2.4 为 announce 添加好主键自增后，插入的数据 id 正确了。

<http://localhost:8080/admin/add/announce?time={{now}}&content=这是一条测试通知>

POST http://localhost:8080/admin/add/announce?time={{timestamp}}&content=这是一条测试通知

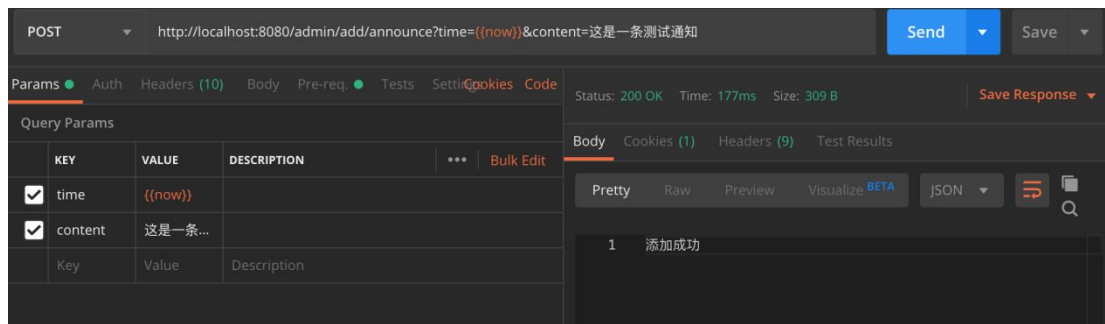
Status: 200 OK Time: 108ms Size: 466 B

Pre-request

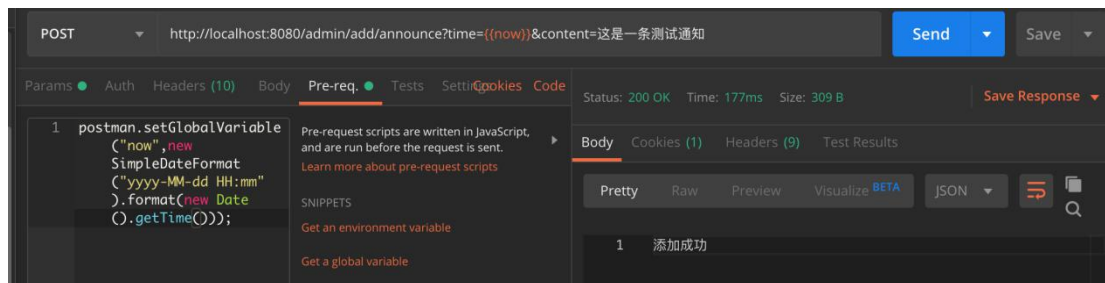
```
1 postman.setGlobalVariable("now", new Date());
```

Body

```
1 could not execute statement; SQL [n/a
2 ]; constraint [PRIMARY
3 ]; nested exception is
   org.hibernate.exception.ConstraintViolationException: could not execute statement
```

2.5 开始调整时间格式:用当前系统时间作为参数,并且将时间格式化为 YYYY-MM-DD HH:mm



(format 参数错误)



(正确的时间格式)



announce_id	content	time
1	第17周周一钟晖乒乓球课本周停课	2019-12-22 8:00
2	第17周周三下午中北体育馆羽毛球馆举办软件学院羽毛球比赛	2019-12-23 12:00
3	这是一条测试通知	1577701554
4	这是一条测试通知	Mon Dec 30 2019 18:37:14 GMT 0800 (CST)
5	这是一条测试通知	yy-12-06 18:51
6	我猜这是一个时间格式正确的通知	yyyy-12-Mo 18:52
7	这个时间格式到底对不对	yyyy-12-Mo 18:53
8	这个时间格式到底对不对	yyyy-12-Mo 18:55
9	这个时间格式到底对不对	2019-12-30 18:56
10	一条时间格式正确的通知	2019-12-30 18:57

2.6

<http://localhost:8080/admin/add/announce?time={{now}}&content=一条时间格式正确的通知>

最后一条，一个正确的通知添加成功。

announce_id	content	time
1	第17周周一钟晖乒乓球课本周停课	2019-12-22 8:00
2	第17周周三下午中北体育馆羽毛球馆举办软件学院羽毛球比赛	2019-12-23 12:00
3	这是一条测试通知	1577701554
4	这是一条测试通知	Mon Dec 30 2019 18:37:14 GMT 0800 (CST)
5	这是一条测试通知	yy-12-06 18:51
6	我猜这是一个时间格式正确的通知	yyyy-12-Mo 18:52
7	这个时间格式到底对不对	yyyy-12-Mo 18:53
8	这个时间格式到底对不对	yyyy-12-Mo 18:55
9	这个时间格式到底对不对	2019-12-30 18:56
10	一条时间格式正确的通知	2019-12-30 18:57

10 rows in set (0.00 sec)

总结

接口测试发现了一些问题，纪录并修改了一些程序错误。