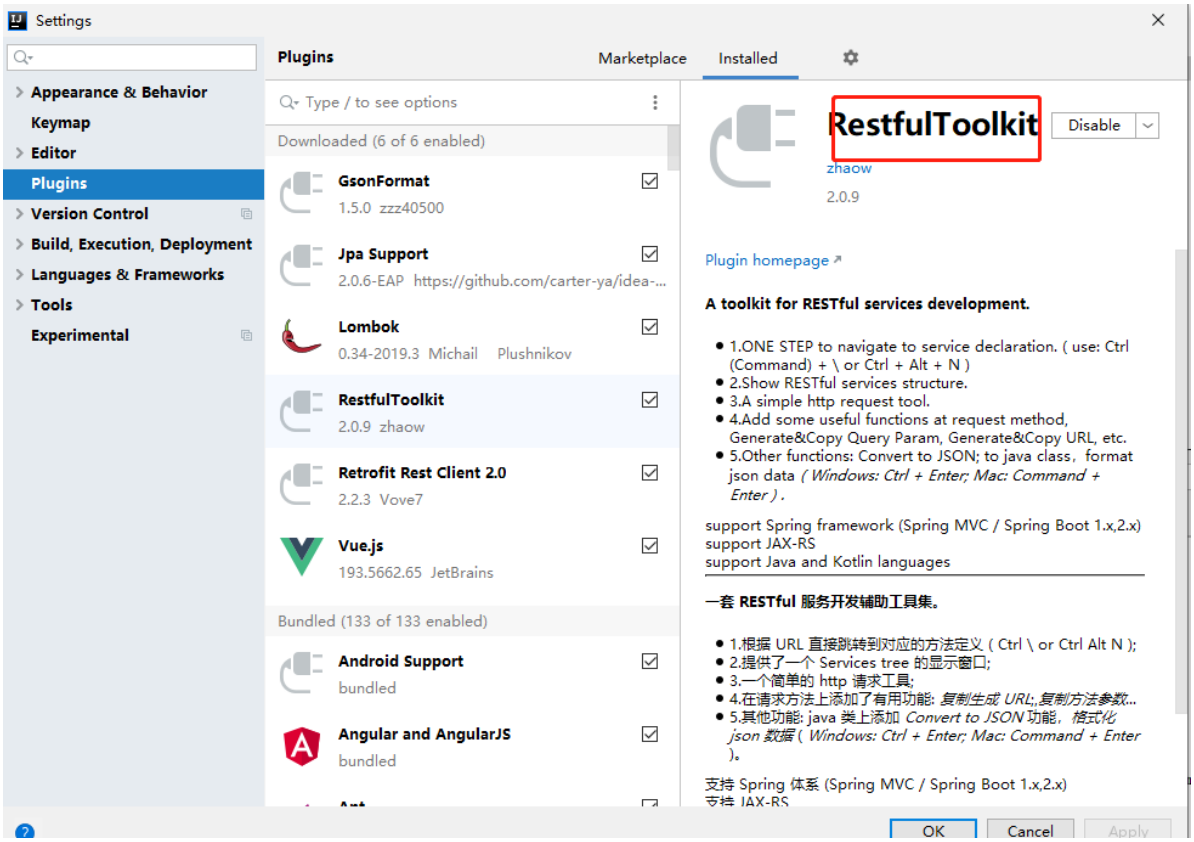
安装RestfulToolkit插件



使用Spring Boot打印SQL语句

如果你使用Spring Boot，可以在application.properties文件中添加以下配置，将JdbcTemplate打印的SQL语句输出到控制台或日志文件中：

```
logging.level.org.springframework.jdbc.core.JdbcTemplate=DEBUG
```

这将设置JdbcTemplate的日志级别为DEBUG，使其打印SQL语句。

查询—@GetMapping

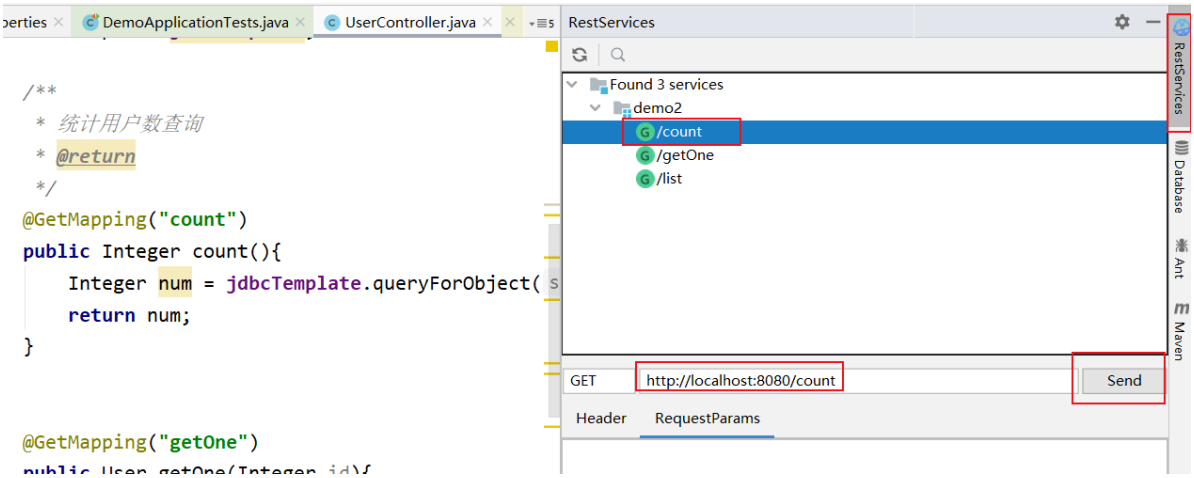
主要用于查询。 get请求特点： a. 请求参数会添加到请求资源路径的后面，只能添加少量参数（因为请求行只有一行，大约只能存放2K左右的数据） b. 请求参数会显示在浏览器地址栏，路由器会记录请求地址 （极为的不安全）

无参数的查询

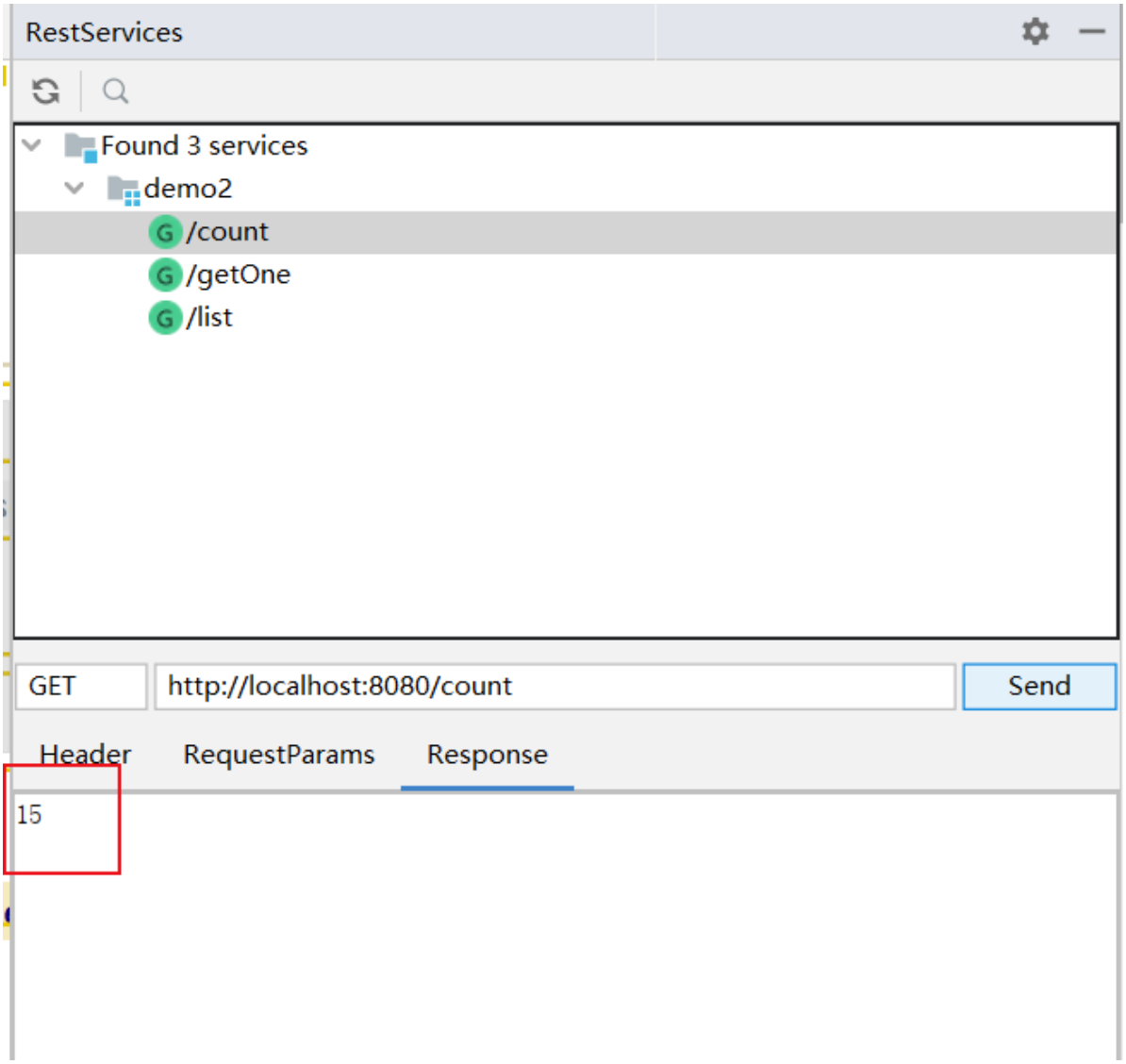
```
@RestController
public class UserController {
```

```
3
4
5       @Autowired
6       JdbcTemplate jdbcTemplate;
7
8       /**
9        * 统计用户数查询
10       * @return
11       */
12      @GetMapping("count")
13      public Integer count(){
14          Integer num = jdbcTemplate.queryForObject("select count(*) from
    user", Integer.class);
15          return num;
16      }
17
18
19
20  }
21
```

打开idea右侧的视图，点击count接口，执行请求



得到接口返回数据

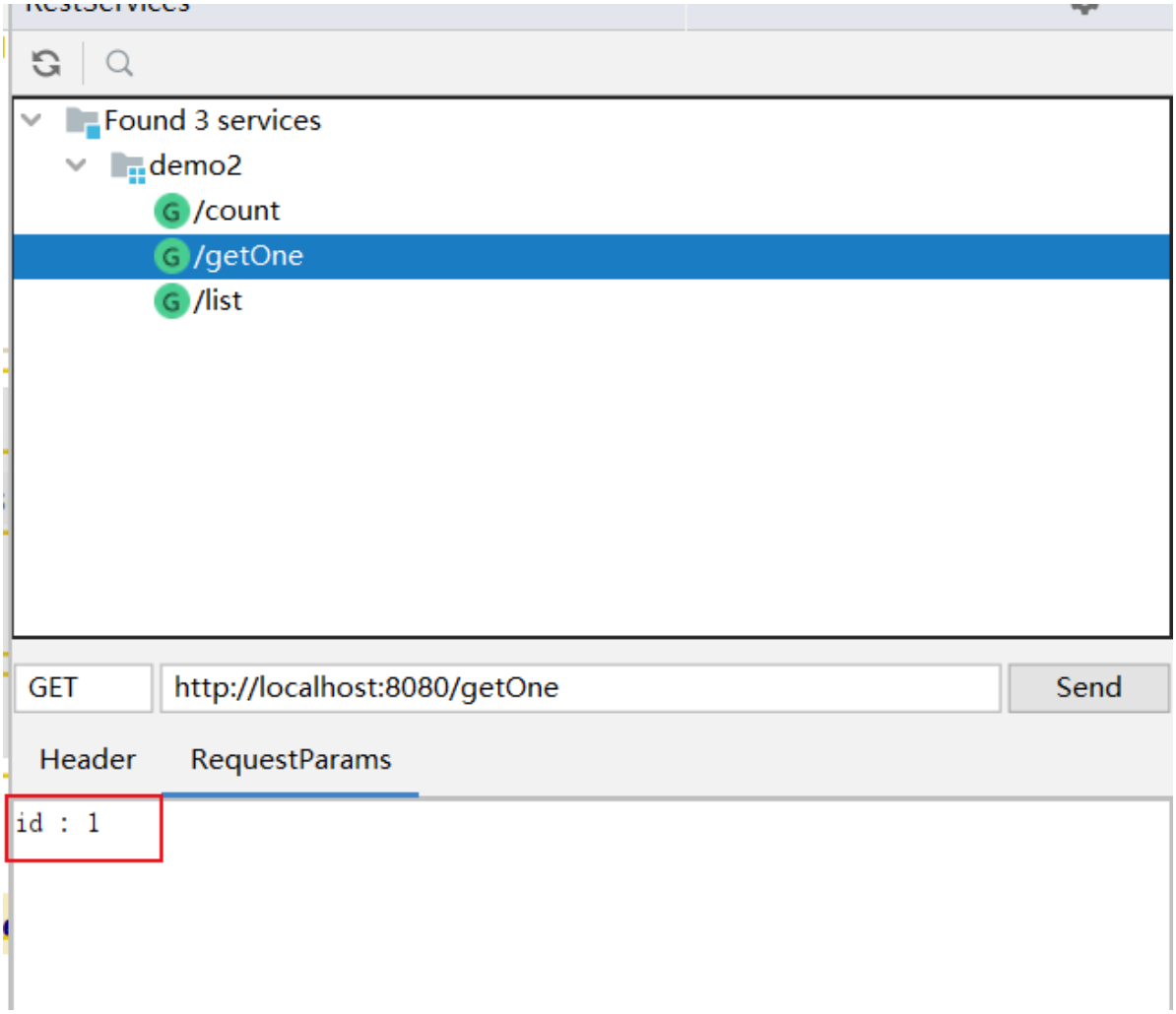


极少参数的查询，查询参数不包含实体类

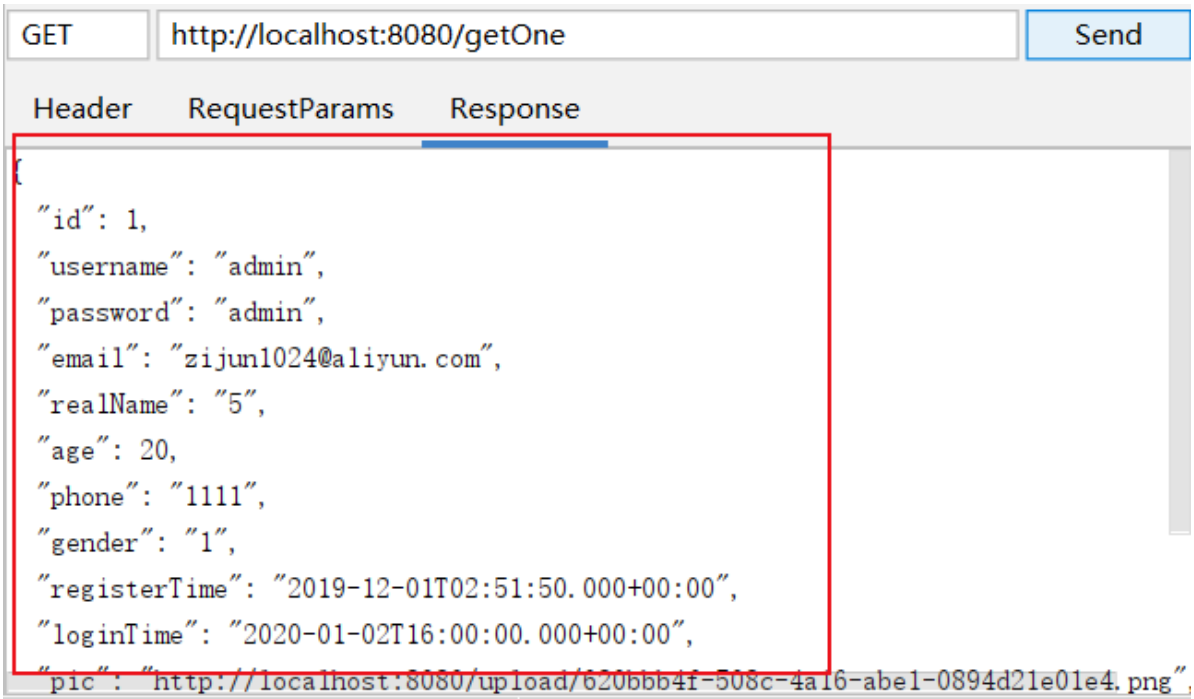
查询参数一般不超过3个，最多的情况是通过id获取详情。

```
@GetMapping("getOne")
    public User getOne(Integer id){
        return jdbcTemplate.queryForObject("select * from user where id
=?",new BeanPropertyRowMapper<>(User.class),"1");
    }
```

输入测试参数

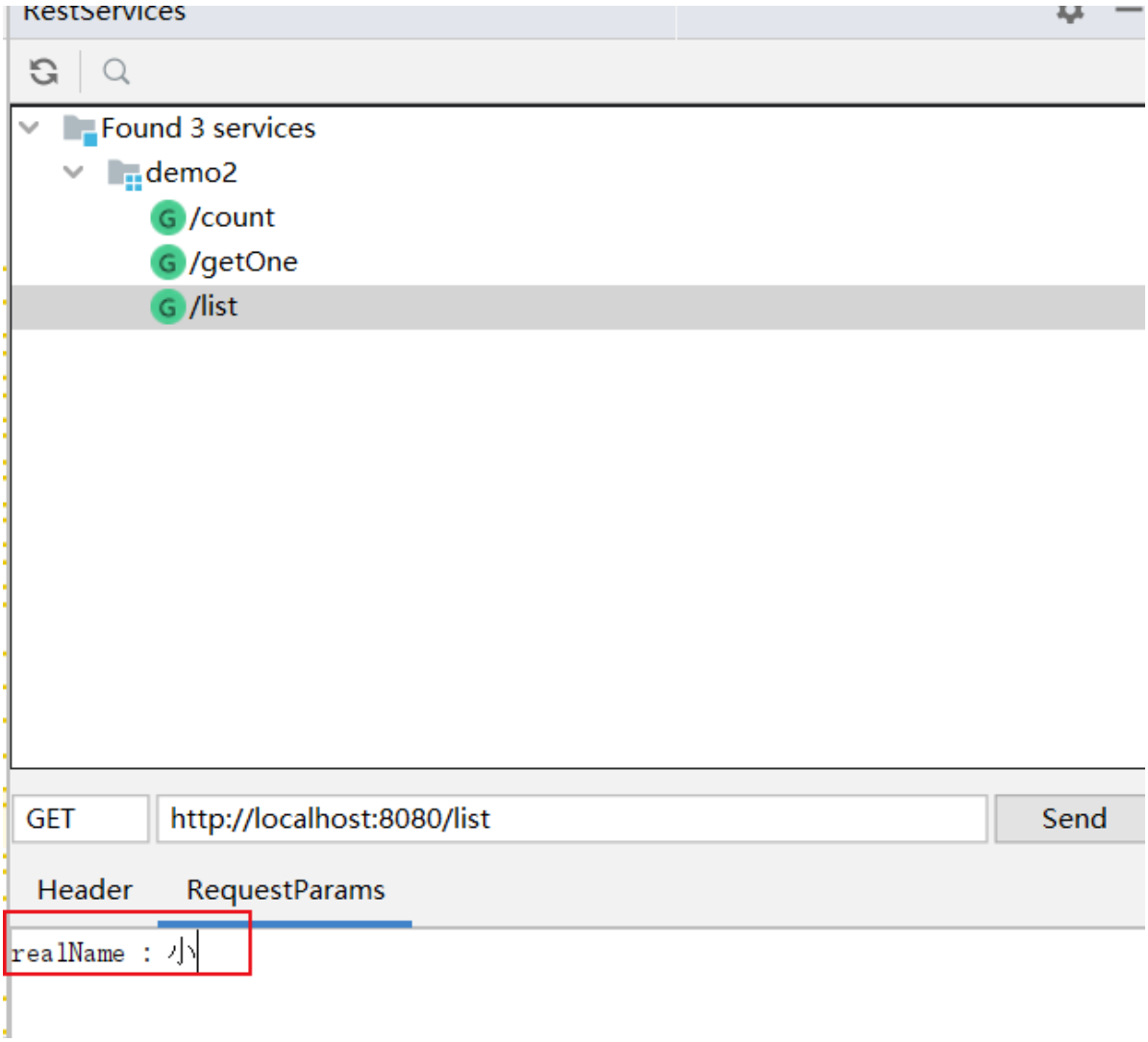

Found 3 services
demo2
/count
/getOne
/list
GET
http://localhost:8080/getOne
Send
Header
RequestParams
id : 1

执行请求,得到返回结果

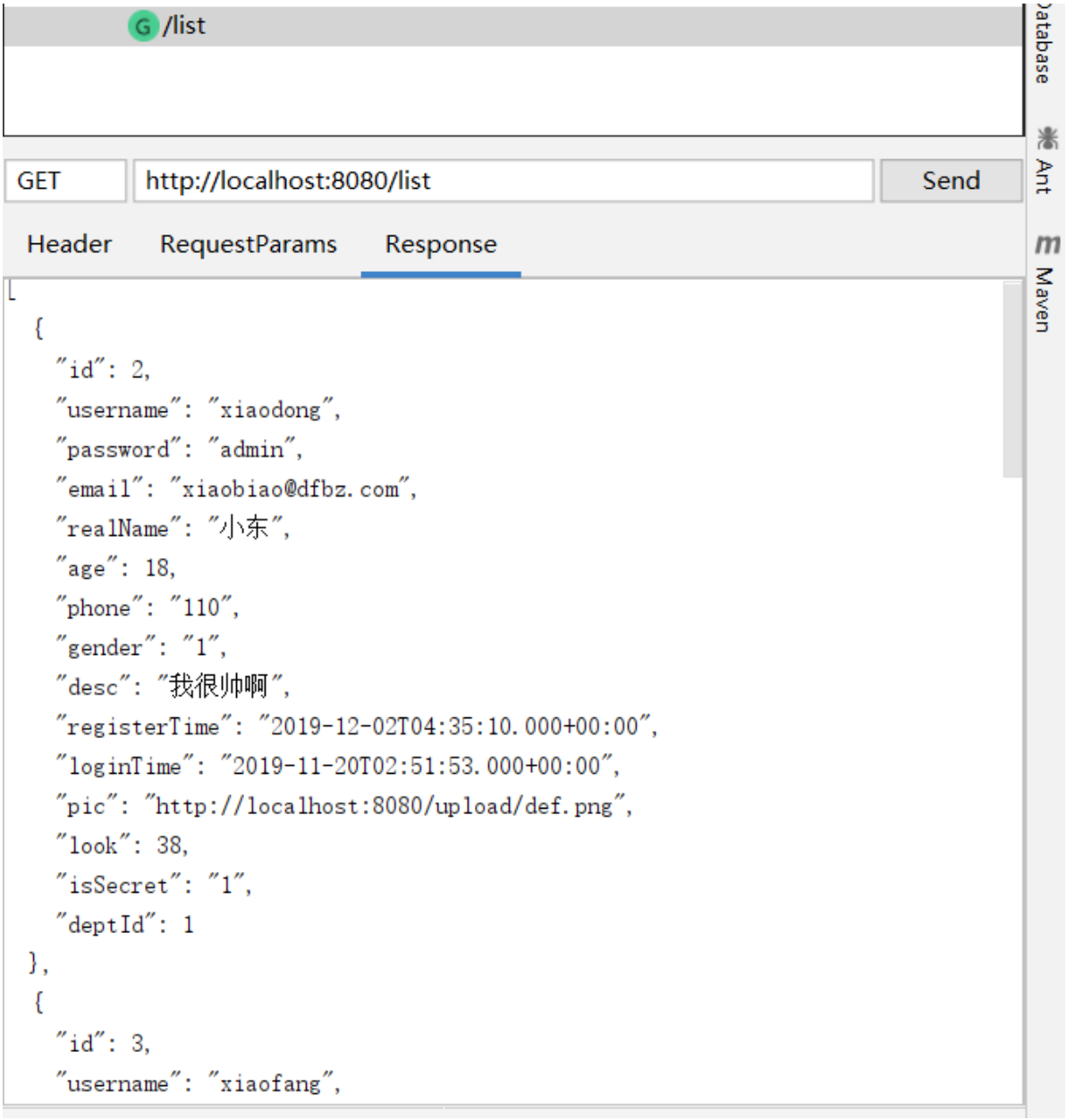

GET
http://localhost:8080/getOne
Send
Header
RequestParams
Response
{
"id": 1,
"username": "admin",
"password": "admin",
"email": "zijun1024@aliyun.com",
"realName": "5",
"age": 20,
"phone": "1111",
"gender": "1",
"registerTime": "2019-12-01T02:51:50.000+00:00",
"loginTime": "2020-01-02T16:00:00.000+00:00",
"pic": "http://localhost:8080/upload/620bbb4f-508c-4a16-abe1-0894d21e01e4.png",

```java
    /**
     * 查询名字带'小'的
     * @return
     */
    @GetMapping("list")
    public List<User> list(String realName){
        //如果没传值 realName = null 则赋值为空串，表示查全部
        if(realName==null){
            realName="";
        }
        return jdbcTemplate.query("select * from user where real_name like
concat('%',?,'%') ",new BeanPropertyRowMapper<>(User.class),realName);
    }
```

输入测试参数



执行并返回接口数据

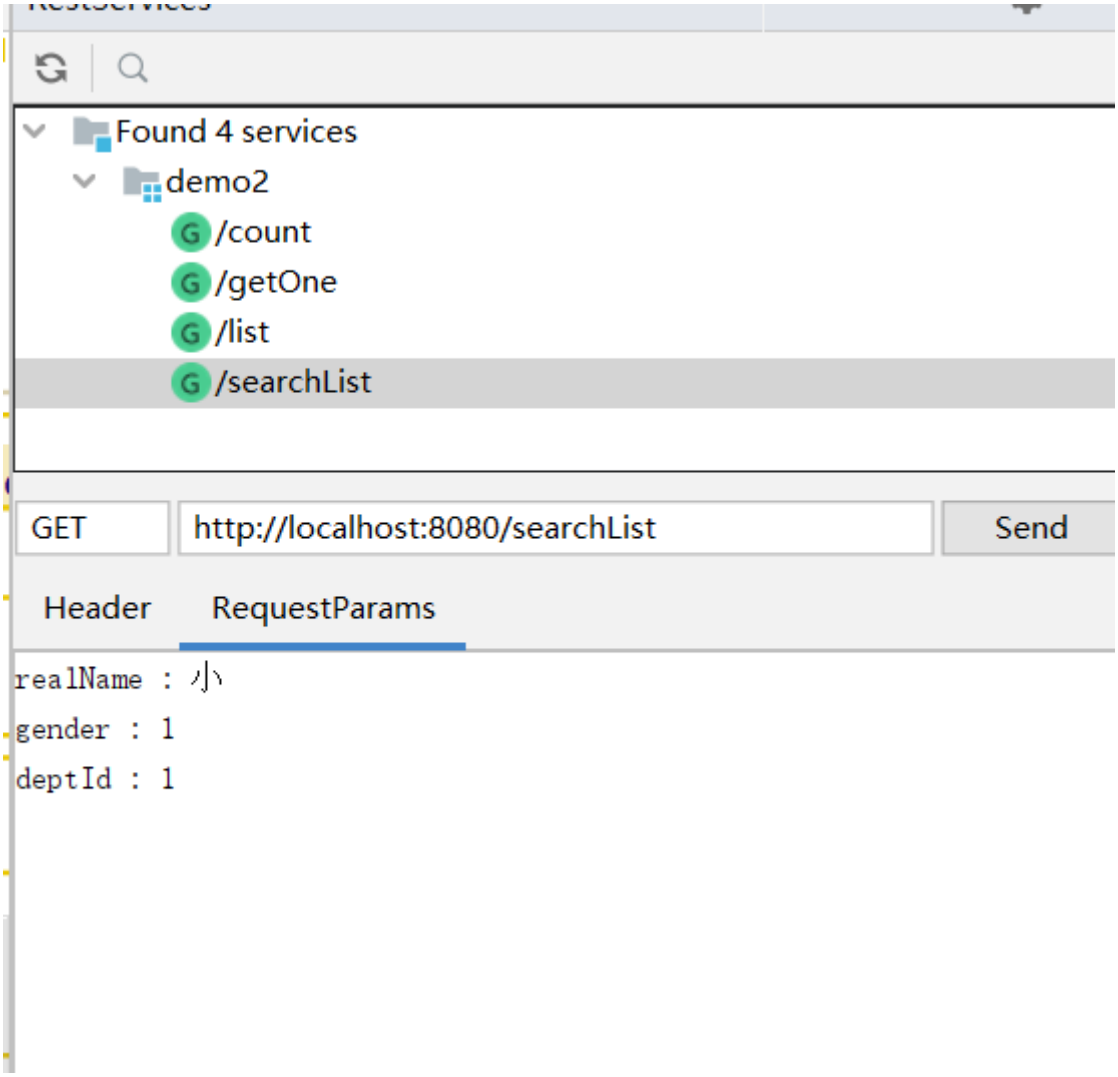


查询参数包含实体类

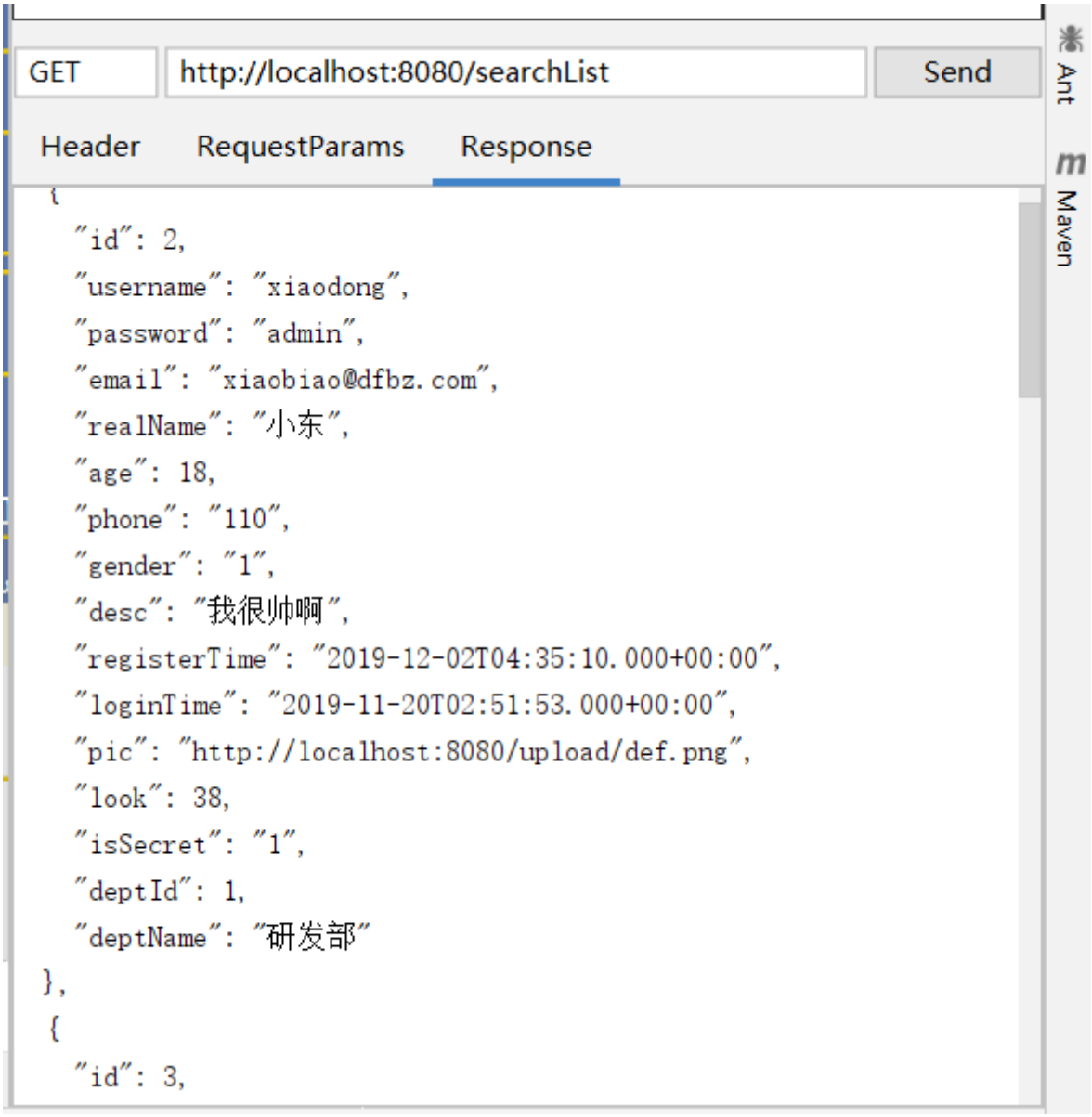
业务场景上，一个模块的查询一般都是某一个实体类的部分字段，如果字段比较多，就应该用实体类来传递。对于get请求，这时候这个其实只要保证传递的参数名与实体类的属性字段一致就可以自动封装到函数参数中

```java
/**
     * 查询deptId为1，gender为1，realName带小的
     */
    @GetMapping("searchList")
    public List<User> searchList(User user){
        //如果没传值 realName = null 则赋值为空串，表示查全部
        if(user.getRealName()==null){
            user.setRealName("");
        }
        String sql = "select * from user where dept_id=? and gender=? and real_name like concat('%',?,'%') ";
        return jdbcTemplate.query(sql,new BeanPropertyRowMapper<>(User.class),user.getDeptId(),user.getGender(),user.getRealName());
    }
```

输入测试参数,删除多余的参数信息



执行并返回接口数据



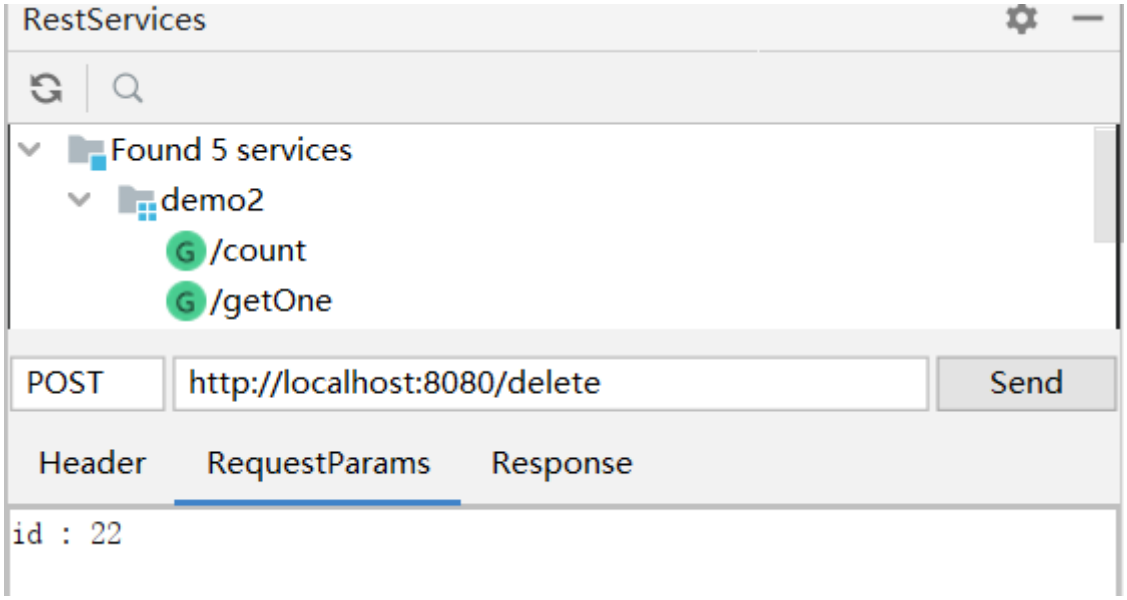
新增、修改、删除—@PostMapping

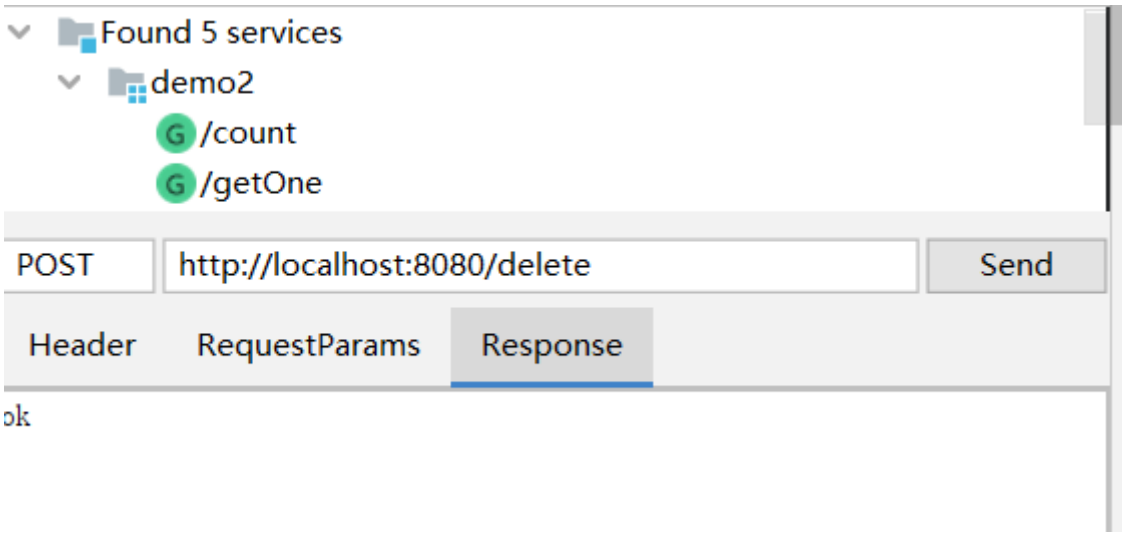
post请求

a. 请求参数添加到实体内容里面，可以添加大量的参数（浏览器地址栏不能发送post请求，在地址栏里我们只能填写URL，并不能进入到Http的实体当中） b. 相对get安全。

删除

```java
/**
     * 删除用户
     */
    @PostMapping("delete")
    public String delete(Integer id){
        jdbcTemplate.update("delete from user where id=?",id);
        return "ok";
    }
```

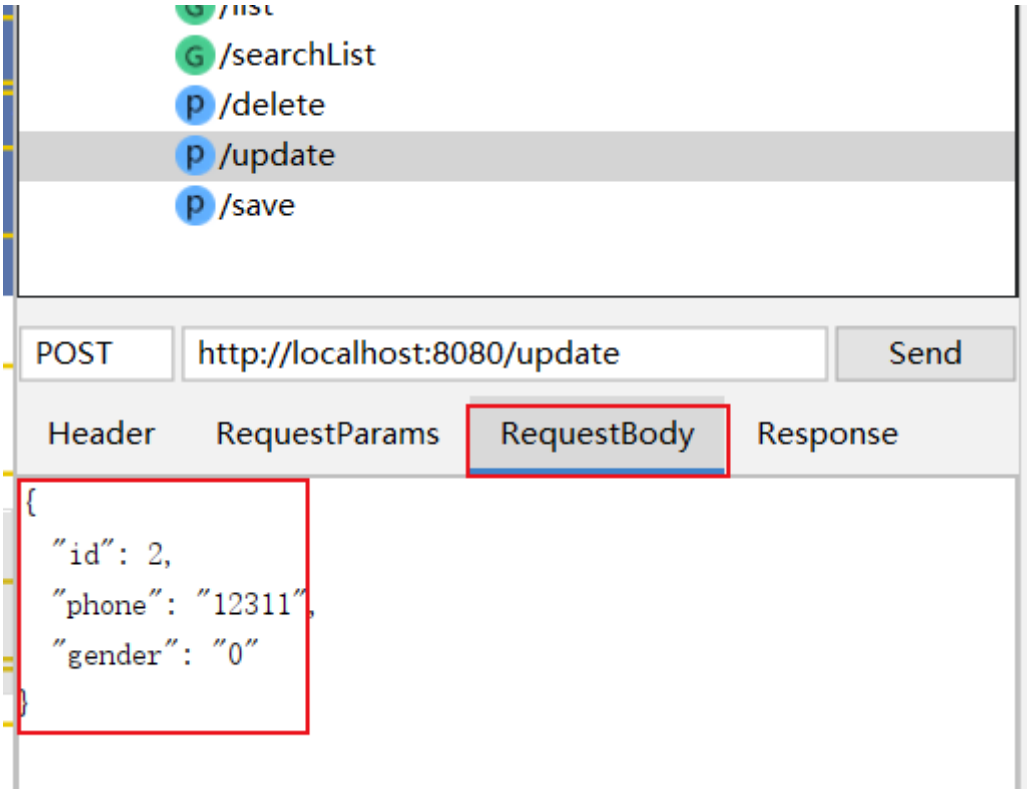




修改

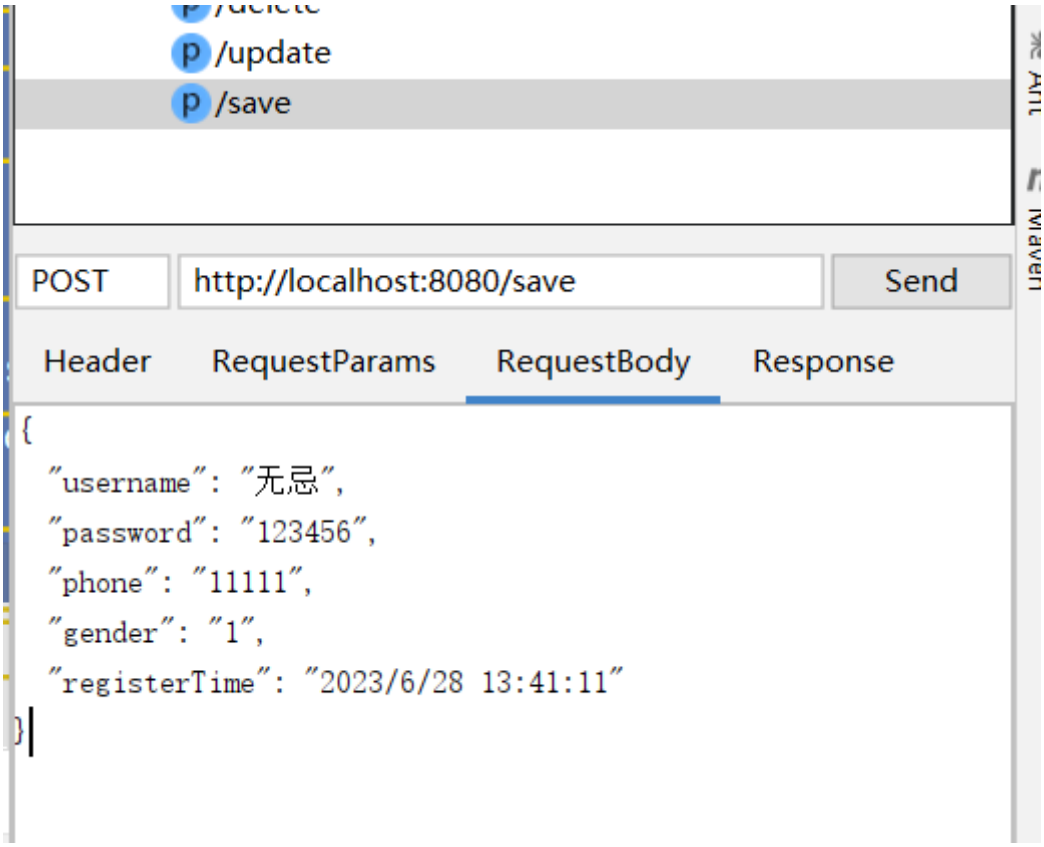
@*RequestBody*主要用来接收前端传递给后端的json字符串中的数据的(请求体中的数据)

```
 /**
     * 更新用户
     */
    @PostMapping("update")
    public String update(@RequestBody User user){
        Object[] params = new Object[]
{user.getGender(),user.getPhone(),user.getId()};
        int num = jdbcTemplate.update("update `user` set `gender` = ? and
`phone` = ? where `id` = ?",params);
        return "ok";
    }
```



新增

```
/**
     * 新增
     * @param user
     * @return
     */
    @PostMapping("save")
    public String save(@RequestBody User user){
        Object[] params =  new Object[]
{user.getUsername(),user.getPassword(),user.getPhone(),user.getGender(),user
.getRegisterTime()};
        String sql = "INSERT INTO `user`( `username`, `password`,  `phone`,
`gender`,`register_time`) VALUES (?,?,?,?,?)";
        jdbcTemplate.update(sql,params);
        return "ok";
    }
```



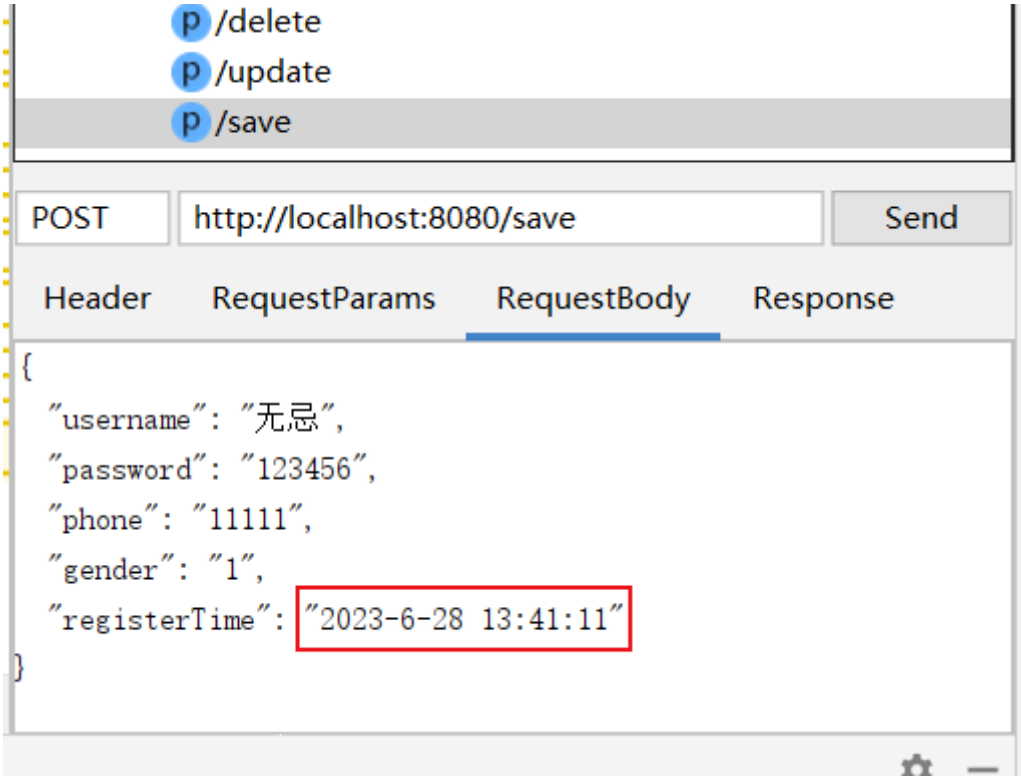
运行后发现报错

```
.HttpMessageNotReadableException: JSON parse error: Cannot deserialize value of type `java.util.Date` from String "2023/6/28 13:41:11": not a
valid representation (error: Failed to parse Date value '2023/6/28 13:41:11': Cannot parse date "2023/6/28 13:41:11": not compatible with any of
 standard forms ("yyyy-MM-dd'T'HH:mm:ss.SSSX", "yyyy-MM-dd'T'HH:mm:ss.SSS", "EEE, dd MMM yyyy HH:mm:ss zzz", "yyyy-MM-dd")); nested exception is
 com.fasterxml.jackson.databind.exc.InvalidFormatException: Cannot deserialize value of type `java.util.Date` from String "2023/6/28 13:41:11":
not a valid representation (error: Failed to parse Date value '2023/6/28 13:41:11': Cannot parse date "2023/6/28 13:41:11": not compatible with
any of standard forms ("yyyy-MM-dd'T'HH:mm:ss.SSSX", "yyyy-MM-dd'T'HH:mm:ss.SSS", "EEE, dd MMM yyyy HH:mm:ss zzz", "yyyy-MM-dd"))<EOL> at
[Source: (org.springframework.util.StreamUtils$NonClosingInputStream); line: 6, column: 19] (through reference chain: com.example.demo
.User["registerTime"])]
```

修改User类，找到registerTime和loginTime，添加json格式化成日期的字符串规则

```java
//设置json字符串转Date类型格式规则
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
private Date registerTime;
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
 private Date loginTime;
```

重新修改请求json参数的日期格式


/delete
/update
/save
POST
http://localhost:8080/save
Send
Header
RequestParams
RequestBody
Response
{
"username": "无忌",
"password": "123456",
"phone": "11111",
"gender": "1",
"registerTime": "2023-6-28 13:41:11"
}

再次访问成功