DB2API配置化开发

第四组

杜星亮　任双寅　韩畅　黄毅婷

2023.06.04

目录

CONTENTS

01 需求理解

需求理解

程序员小人平时在进行应用开发时，发现需要开发大量 API，实现对数据库的增删改查和拼装。并且常常由于前后端缺乏统一的接口描述、文档未及时更新和重复性工作较多等原因，整个开发过程效率不高。为了提升开发效率，小A想开发一个工具，实现只需通过简单的配置就能实现对数据库的某个表中的记录的操作，并对外暴露 API。

具体有如下功能:

(1) 实现一行或多行记录的插入
(2) 实现一行或多行记录的删除，并允许添加 where 过滤
(3) 实现一行或多行记录的更新
(4) 实现条件查询
(5) 实现分页查询和字段排序
(6) 实现分组查询
(7) 实现聚合查询
(8) 实现连表查询
(9) 具有接口安全访问机制，具备权限管控、防 SQL 注入等
(10)其他合理的功能完善或优化亦可加分，包括但不限于接口文档生成、校验参数、支持正则匹配、支持嵌套子查询、查询性能优化、错误提示优化等。

为关系型数据库提供
一个通用的，
基于配置文件进行访问控制的
Web API 查询修改接口

02 架构设计

方案设计

需求理解

为关系型数据库提供一个通用的，基于配置文件进行访问控制的 Web API 查询修改接口。

设计思路

I. 设计 API 调用规范，描述对数据表的增删改查操作

- 使用 JSON 格式，对不同种类操作提供相应字段，指定过滤条件，分页等查询选项

II. 设计配置文件规范，描述数据表名称映射、用户信息和访问权限控制

- 使用 JSON 格式，提供全局和用户两个层面在表粒度上的权限控制

III. 系统处理结构

- 解析配置文件，生成配置数据模型，得到用户信息和授权信息
- 收到请求后，解析请求得到 API 请求数据模型，得到调用用户信息和具体查询操作
- 鉴权步骤先根据用户信息进行认证，再根据请求信息中访问到的表，配置文件中用户的权限设置进行授权
- 授权通过后，通过实现的SQL翻译器将API数据模型中的查询操作翻译成SQL语句
- 调用数据库接口执行SQL，获得查询结果

方案设计——系统架构与分层

构建请求处理管道，使用一系列中间件表示各个层次处理过程。
使各部分解耦、可组合，并实现通用的异常处理和响应构建。

HTTP 请求（FaaS）

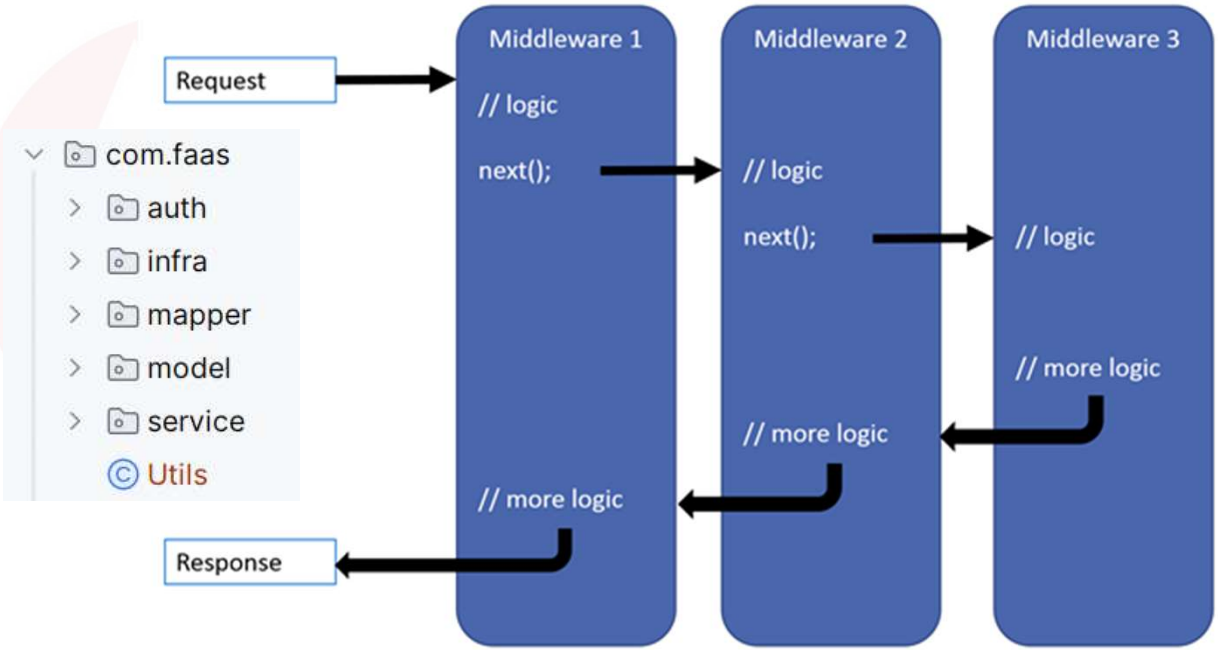
转换层（HTTP请求解析和校验）

鉴权层（用户身份认证和授权）

服务层（SQL翻译）

数据库访问层（SQL执行）

数据存储层



方案设计


HTTP 请求
转换层
mapper
api
BaseJsonApiParser
config
BaseJsonConfigParser
ModelJsonParser
ModelParser
Config Model
API Model
model
ApiModel
ConfigModel
HttpContext
SqlStatements
User
auth
AccessChecker
BaseAccessChecker
鉴权层
service
BaseSqlTranslator
SqlTranslator
服务层（SQL翻译）
数据库访问

方案设计

系统处理结构如下（1-2）

- **解析配置文件**，生成配置数据模型，得到用户信息和授权信息
- 收到请求后，**解析请求**得到 API 请求数据模型，得到调用用户信息和具体查询操作

```
public ApiModel parseJSON(JSONObject json) {

    ApiModel res = new ApiModel();
    String method = json.getStr( key: "method");
    if (method.equals("query")) {
        res = query(json);
    } else if (method.equals("insert")) {
        res = insert(json);
    } else if (method.equals("delete")) {
        res = delete(json);
    } else if (method.equals("update")) {
        res = update(json);
    }
    res.name = json.getStr( key: "name");
    res.user = user(json.getJSONObject( key: "user"));
    return res;
}
```

```
public ConfigModel parseJSON(JSONObject jsonObject) {
    // new一个configmodel
    ConfigModel config_model = new ConfigModel();

    // 提取UserConfig
    JSONArray arrayUser = jsonObject.getJSONArray( key: "UserConfig");
    Map<String, UserConfig> usermap = new HashMap<>(arrayUser.size());
    // 根据user数量
    for (int i = 0; i < arrayUser.size(); i++) {
        ConfigModel.UserConfig tmpuserconfig = config_model.new UserConfig();
        JSONObject obj = arrayUser.getJSONObject(i);
        String name = (String) obj.get("name");
        String password = (String) obj.get("password");
        tmpuserconfig.setName(name);
        tmpuserconfig.setPassword(password);
        // 提取每个user的OperationConfig
        JSONArray opconfig = obj.getJSONArray( key: "OperationConfig");
        Map<String, OperationConfig> map = new HashMap<>(opconfig.size());
```

方案设计

系统处理结构如下（3）

- **鉴权**步骤先根据用户信息进行认证，再根据请求信息中访问到的表，配置文件中用户的权限设置进行授权

```
public ArrayList<String> check(ApiModel request, ConfigModel config) {

    ArrayList<String> err = new ArrayList<>();
    var all_user_config = config.USER_CONFIG_MAP;

    var username = request.user.userName;
    var password = request.user.userPassword;

    System.out.printf("username: %s, password: %s", username, password);

    if (!all_user_config.containsKey(username)) {
        err.add(String.format("User %s not found!", username));
        return err;
    }

    if (!all_user_config.get(username).getPassword().equals(password)) {
        err.add("Wrong Password!");
        return err;
    }
```

方案设计

系统处理结构如下（4-5）

- 授权通过后，通过实现的**SQL翻译器**将API数据模型中的查询操作翻译成SQL语句
- 调用数据库接口执行SQL，获得查询结果

```
public SqlStatements translate(ApiModel request, ConfigModel config)
{
    ArrayList<String> sqls = new ArrayList<>();

    if (request instanceof InsertRequest) {
        var insertRequest = (InsertRequest) request;
        sqls.add(insertOperation(insertRequest.name, insertRequest.value));
    } else if(request instanceof DeleteRequest) {
        var deleteRequest = (DeleteRequest) request;
        sqls.add(deleteOperation(deleteRequest.name, deleteRequest.condition));
    }

    //sqls.add("select * from test_table_txx where `name`='张三'");

    SqlStatements result = new SqlStatements();
    result.setStatements(sqls);
    return result;
}
```

方案设计

分层表示为中间件，构建处理流水线。

```
pipeline.middlewares.add((context, data) -> {
    ArrayList<String> result = new BaseAccessChecker().check(data.<ApiModel>get(DATA_API),
            data.<ConfigModel>get(DATA_CONFIG));
    logger.info(format: "Auth Result: {}", Utils.toJSON(result));
    if (result.size() > 0) {
        context.response.status = 401;
        context.response.message = String.join("\n", result);
        return true;
    }
    return false;
});
pipeline.middlewares.add((context, data) -> {
    SqlStatements result;
    result = new BaseSqlTranslator().translate(data.<ApiModel>get(DATA_API),
            data.<ConfigModel>get(DATA_CONFIG));
    data.push(DATA_SQL, result);
    logger.info(format: "Translate Result: {}", Utils.toJSON(result));
    return false;
});
```

方案设计

流水线提供统一异常处理和响应构建机制。

```
public HttpContext.Response resolve(HttpContext.Request request) {
    HttpContext context = new HttpContext();
    context.request = request;
    for(Middleware middleware : middlewares) {
        try {
            if (middleware.resolve(context, data)) {
                return context.response;
            }
        } catch (Exception ex) {
            LOGGER.error(msg: "Error when processing middleware", ex);
            context.response.status = 500;
            context.response.message = "Error when processing middleware: " + ex.getMessage();
            return context.response;
        }
    }
    try {
        handler.resolve(context, data);
    }
    catch (Exception ex) {
        LOGGER.error(msg: "Error when processing handler", ex);
        context.response.status = 500;
        context.response.message = "Error when processing handler: " + ex.getMessage();
    }
    return context.response;
}
```

03 规范描述

规范描述——API调用规范

描述对数据表的增删改查操作，约束合法的API调用路径和请求体。

- 使用 JSON 格式，对不同种类操作提供相应字段，指定过滤条件，分页等查询选项
- HTTP POST 方法，根据URL路径分发到不同解析器对请求体进行解析

/insert

```
{
    "user": {                               // 鉴权用户信息
        "name": "userName",                 // 用户名
        "password": "userPassword",         // 用户密码
    },
    "name": "table_name",                   // 表的名字为"table_name"

    "value": {                              // 插入数据
        "col1": "value1",                   // 列名：数据值
        "col2": "value2",
    }
}
```

/delete

```
{
    "user": {
        "name": "userName",
        "password": "userPassword",
    },
    "name": "table_name",                   // 表的名字为"table_name"

    "cond": "where clause expression",      // 筛选条件，SQL where 子句表达式
}
```

/update

```
{
    "user": {
        "name": "userName",
        "password": "userPassword",
    },
    "name": "table_name",                   // 表的名字为"table_name"

    "cond": "where clause expression",      // 筛选条件，SQL where 子句表达式

    "value": {                              // 更新数据
        "col1": "value1",                   // 列名：数据值
        "col2": "value2",
    }
}
```

规范描述——API调用规范

描述对数据表的增删改查操作，约束合法的API调用路径和请求体。

- 使用 JSON 格式，对不同种类操作提供相应字段，指定过滤条件，分页等查询选项
- HTTP POST 方法，根据URL路径分发到不同解析器对请求体进行解析

/query

```
{
    "user": {
        "name": "userName",
        "password": "userPassword",
    },
    "name": "table_name",                    // 表的名字为"table_name"

    "cond": "where clause expression",       // 筛选条件，SQL where 子句表达式
    "sort": "col1+,col2-,col3+",             // 排序，+/-表示对应列采用升序和降序，若无此属性，表示不排序
    "page": "pageNumber@pageSize",           // 分页，如 2@5 表示第二页，每页5行，若无此属性，表示不分页
    "group": "colName",                      // 分组列名，若无此属性，表示不分组
    "cols": [                                // 查询目标列名
        "col1",                              // 列名
        "func:col2",                         // 聚合查询，如 SUM:col2 表示对 col2 列求和
    ],
    "joinTable": "join_table_name",          // 联表查询，第二张表名
    "joinCond": "join on clause expression",// 连接条件，JOIN ON 子句表达式
}
```

规范描述——配置文件规范

描述数据表名称映射、用户信息和访问权限控制

- 使用 JSON 格式，提供全局和用户两个层面在表粒度上的权限控制

`GlobalConfig`的字段规范如下：

- "table_name"：`String`类型，设置表的名称
- "name"：`String`类型，设置表的别名
- "add"：`Boolean`类型，设置此表最高的添加操作权限
- "update"：`Boolean`类型，设置此表最高的更新操作权限
- "delete"：`Boolean`类型，设置此表最高的删除操作权限
- "get"：`Boolean`类型，设置此表最高的查询操作权限

```
"OperationConfig": [
    {
        "table_name": "test_table",
        "add": true,
        "delete": true,
        "update": true,
        "get": true
    },
    {
        "table_name": "join_table",
        "add": true,
        "delete": true,
        "update": true,
        "get": true
    }
]
```

`UserConfig`的字段规范如下：

- "name"：`String`类型，设置用户的账户
- "password"：`String`类型，设置用户的密码
- "OperationConfig"：`OperationConfig`类型（此类型和GlobalConfig类型一致），设置此用户对表的操作权限

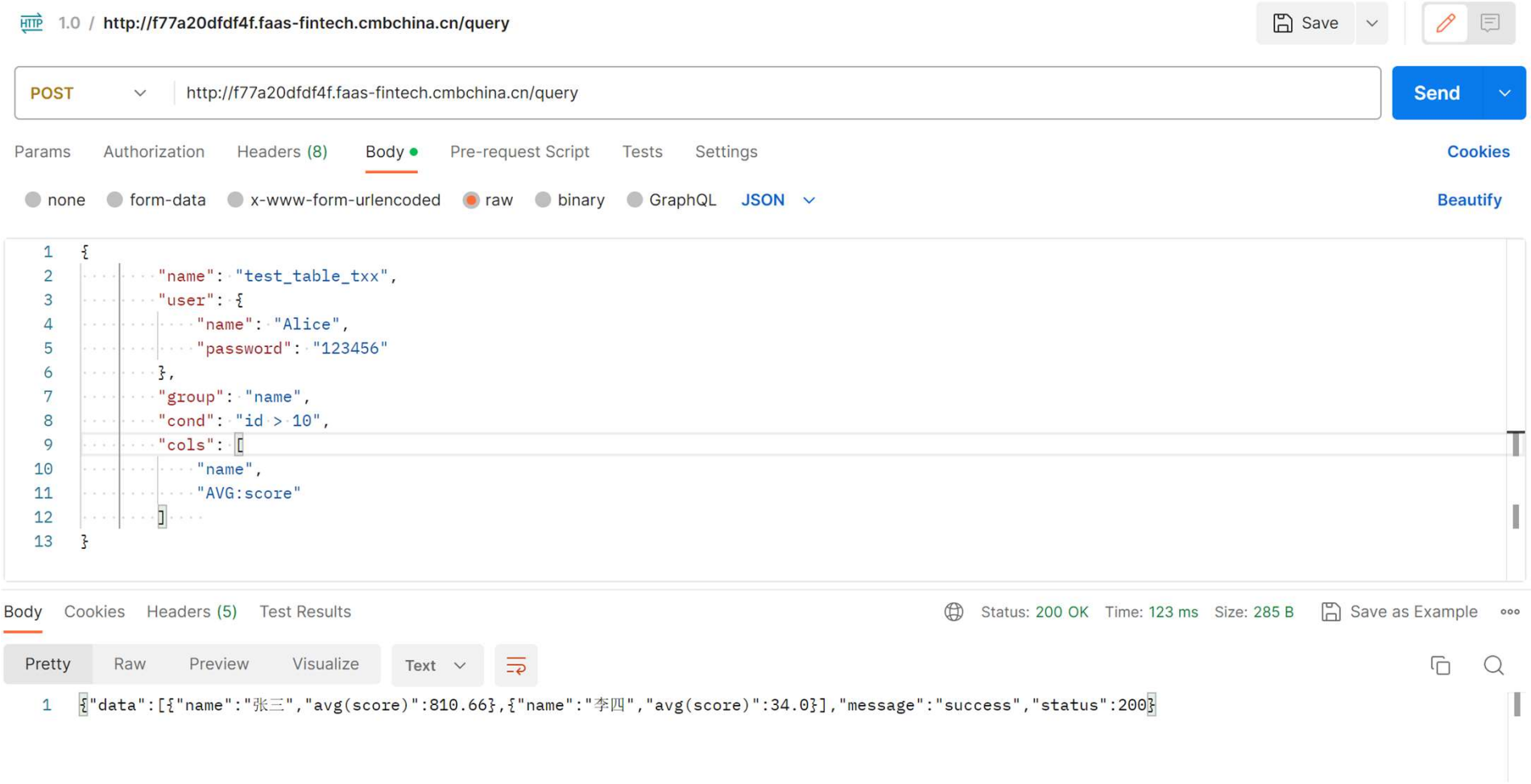
```
{
    "name": "Bob",
    "password": "654321",
    "OperationConfig": [
        {
            "table_name": "test_table",
            "add": false,
            "update": false,
            "delete": true,
            "get": true
        }
    ]
}
```

04 功能实现

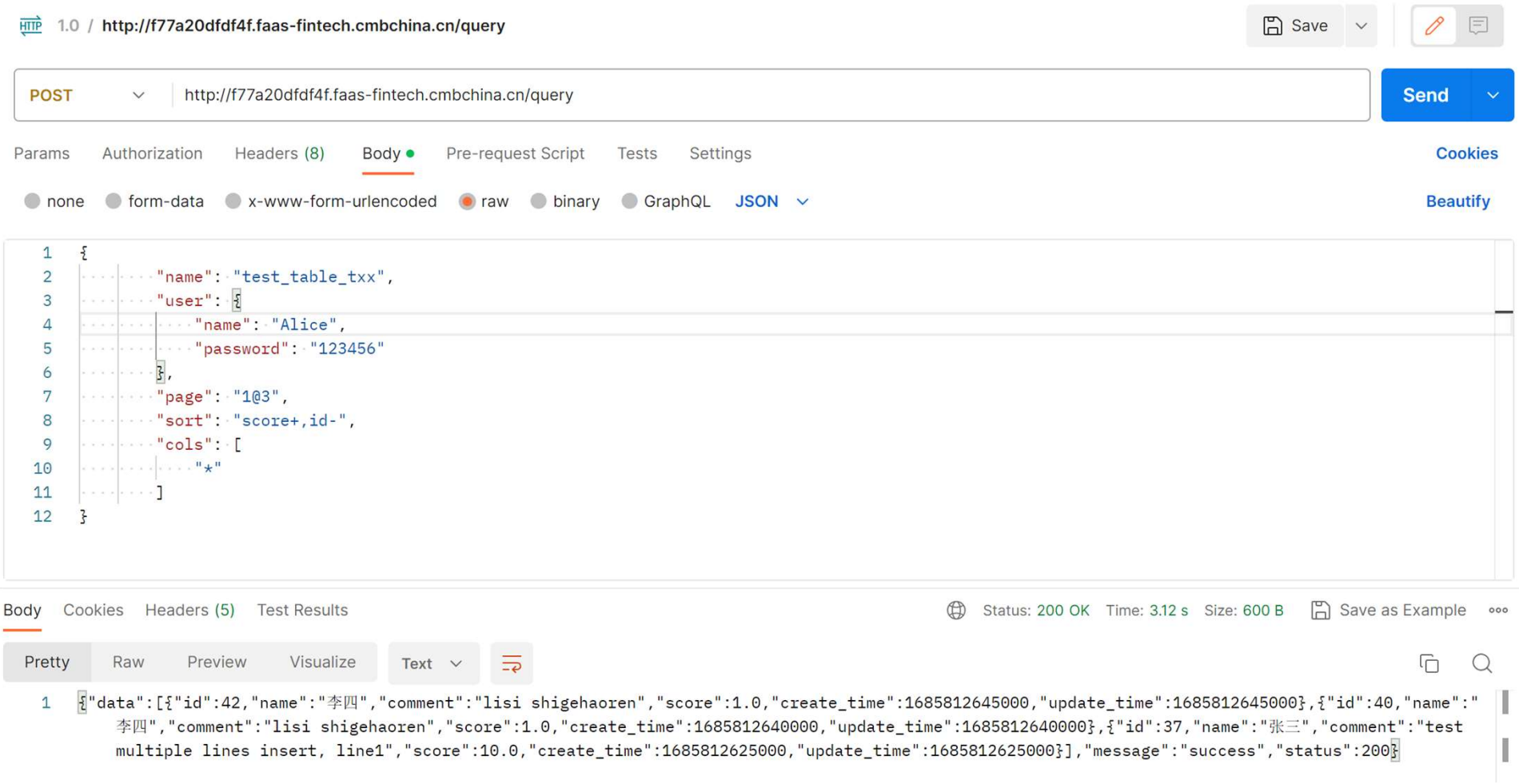
功能实现

1. ☑ 实现一行或多行记录的插入
2. ☑ 实现一行或多行记录的删除和过滤
3. ☑ 实现一行或多行记录的更新
4. ☑ 实现条件查询
5. ☑ 实现分页查询和字段排序
6. ☑ 实现分组查询
7. ☑ 实现聚合查询
8. ☑ 实现连表查询
9. ☑ 具有接口权限管控机制
10. ☑ 请求参数与格式校验
11. ☑ 错误提示优化

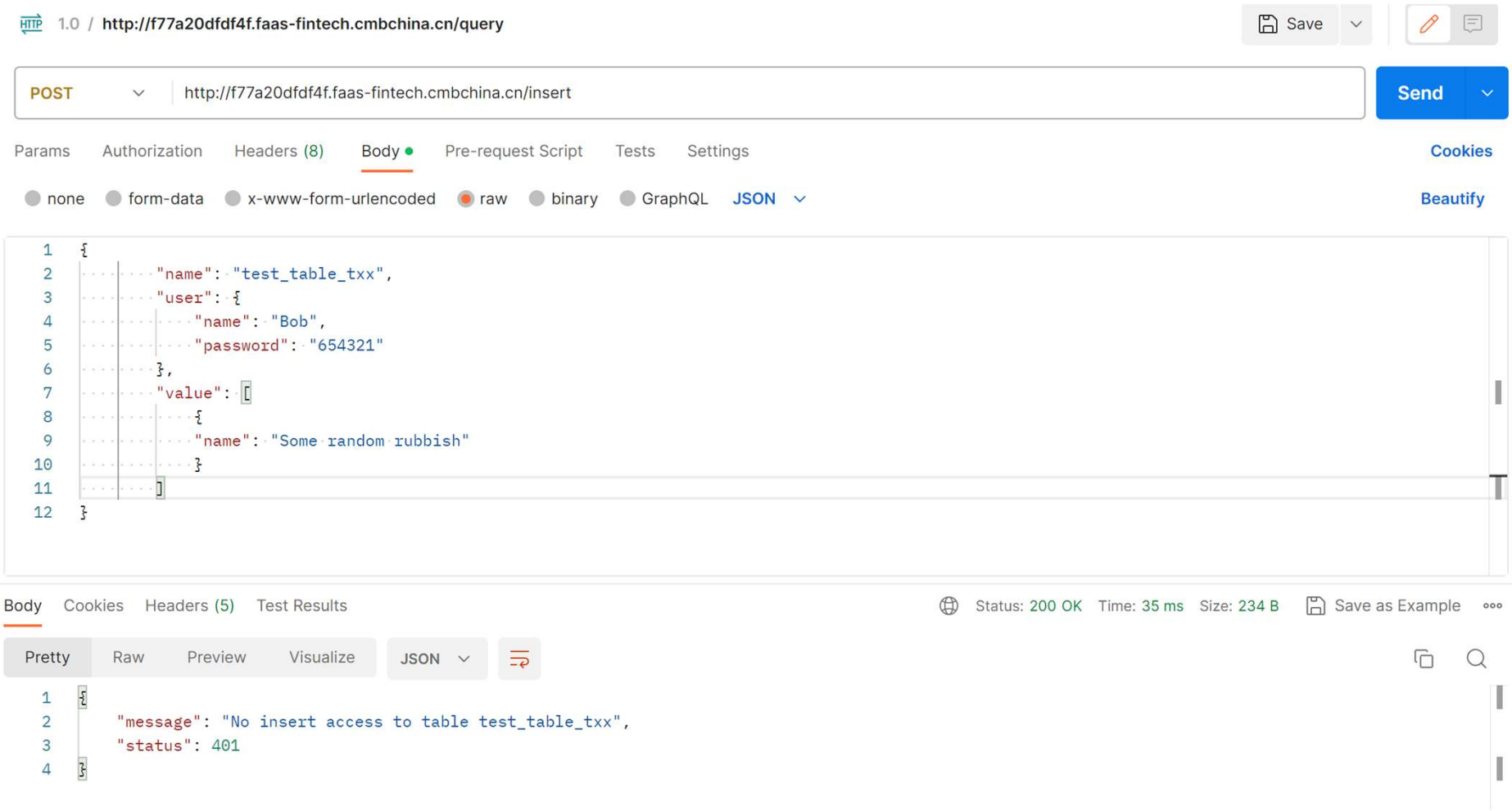
功能实现：过滤与聚合查询



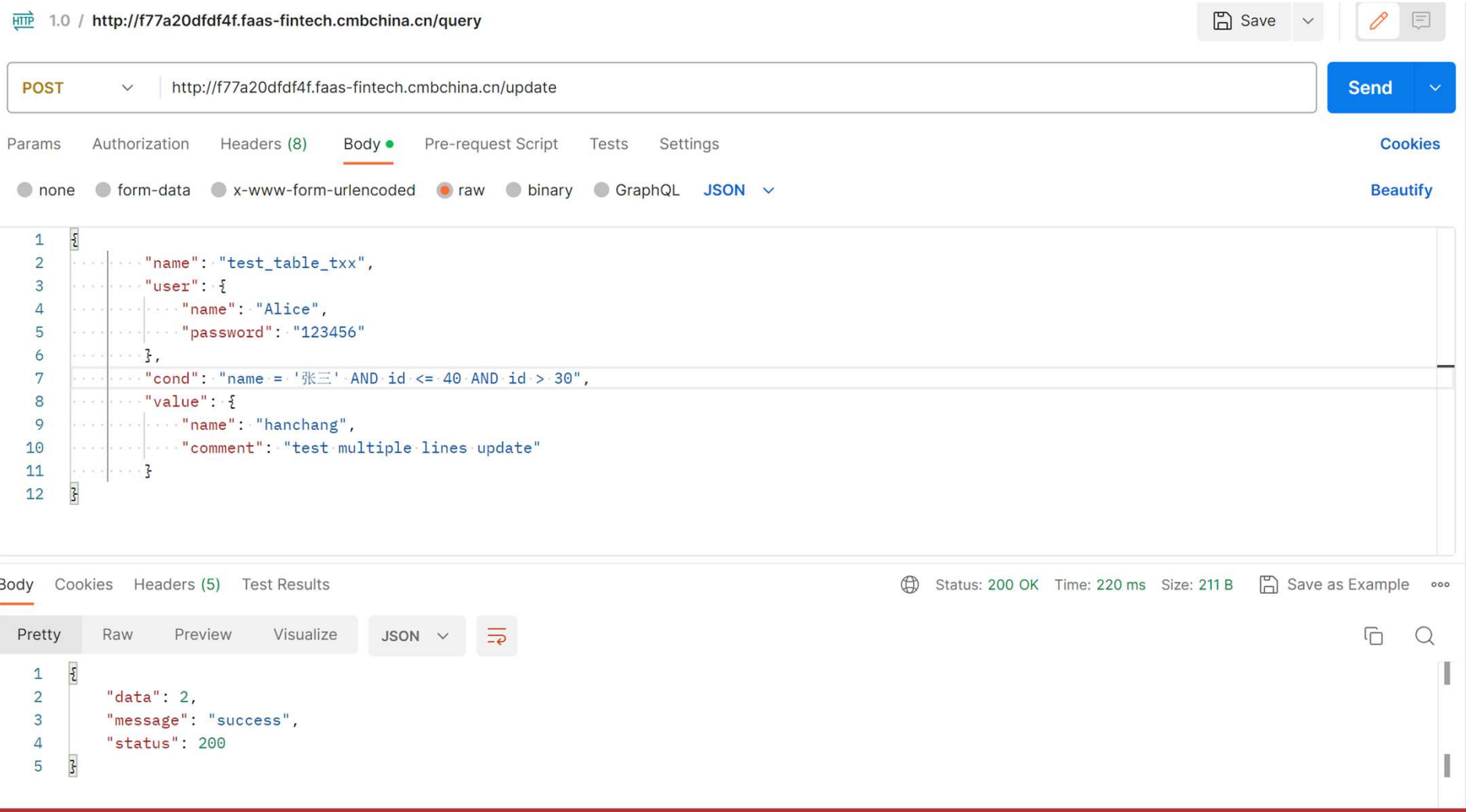
功能实现：排序与分页


1.0 / http://f77a20dfdf4f.faas-fintech.cmbchina.cn/query
Save
POST
http://f77a20dfdf4f.faas-fintech.cmbchina.cn/query
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "test_table_txx",
"user": {
"name": "Alice",
"password": "123456"
},
"page": "1@3",
"sort": "score+,id-",
"cols": [
"*"
]
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 3.12 s
Size: 600 B
Save as Example
Pretty
Raw
Preview
Visualize
Text
{"data":[{"id":42,"name":"李四","comment":"lisi shigehaoren","score":1.0,"create_time":1685812645000,"update_time":1685812645000},{"id":40,"name":"
李四","comment":"lisi shigehaoren","score":1.0,"create_time":1685812640000,"update_time":1685812640000},{"id":37,"name":"张三","comment":"test
multiple lines insert, line1","score":10.0,"create_time":1685812625000,"update_time":1685812625000}],"message":"success","status":200}

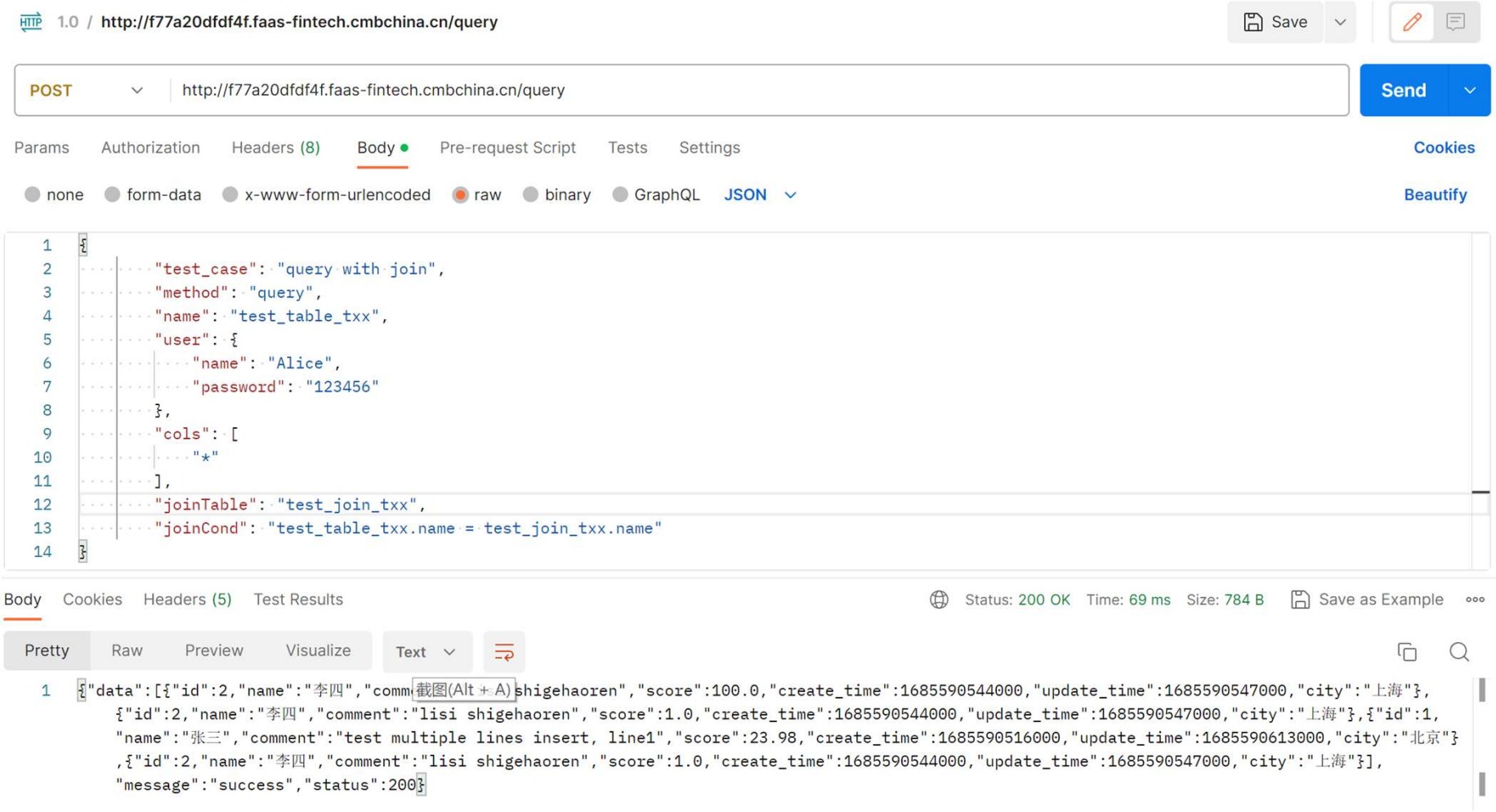
功能实现：用户认证授权


1.0 / http://f77a20dfdf4f.faas-fintech.cmbchina.cn/query
Save
POST
http://f77a20dfdf4f.faas-fintech.cmbchina.cn/insert
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "test_table_txx",
"user": {
"name": "Bob",
"password": "654321"
},
"value": [
{
"name": "Some random rubbish"
}
]
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 35 ms
Size: 234 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"message": "No insert access to table test_table_txx",
"status": 401
}

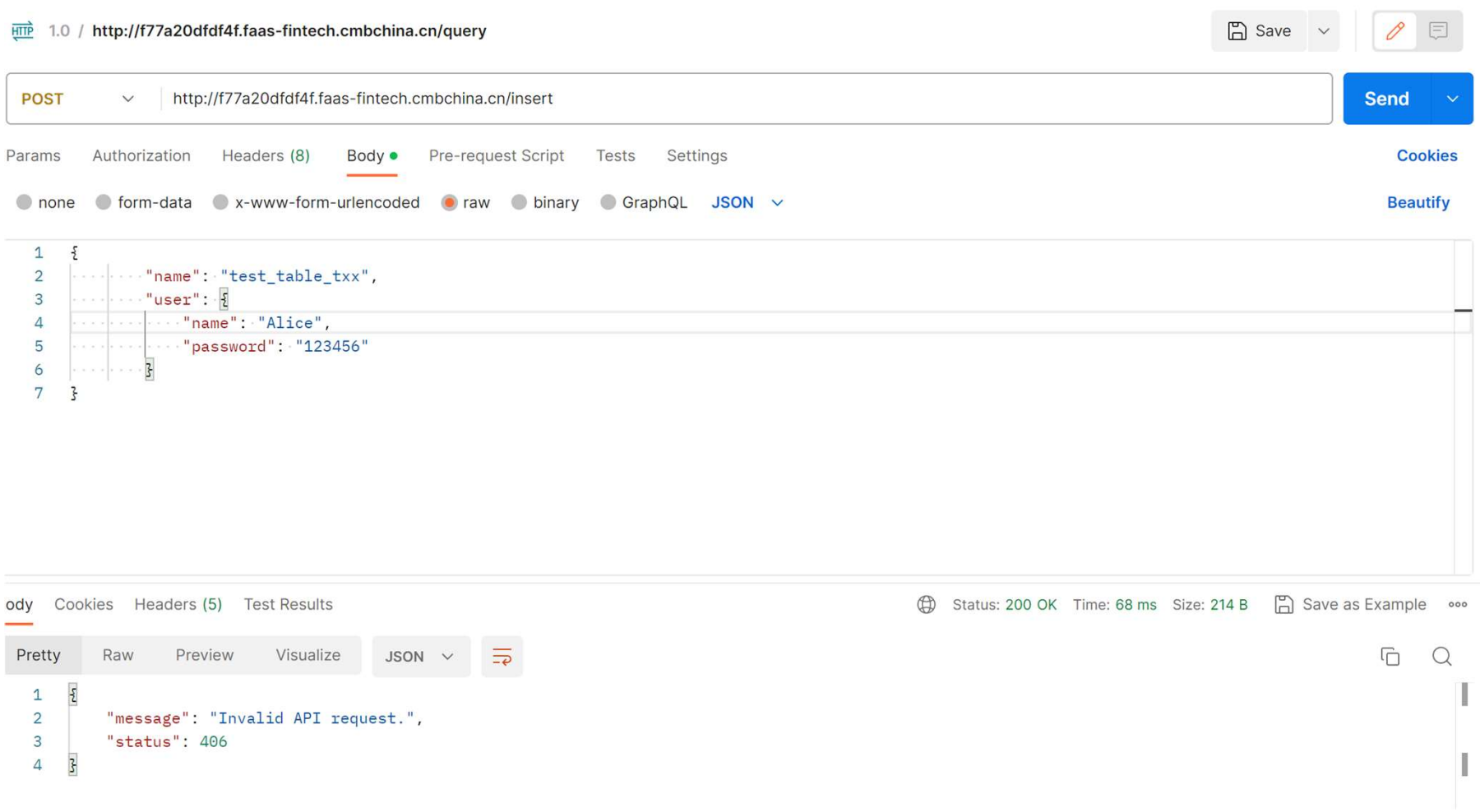
功能实现：批量数据更新


1.0 / http://f77a20dfdf4f.faas-fintech.cmbchina.cn/query
Save
POST
http://f77a20dfdf4f.faas-fintech.cmbchina.cn/update
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "test_table_txx",
"user": {
"name": "Alice",
"password": "123456"
},
"cond": "name = '张三' AND id <= 40 AND id > 30",
"value": {
"name": "hanchang",
"comment": "test multiple lines update"
}
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 220 ms
Size: 211 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"data": 2,
"message": "success",
"status": 200
}

功能实现：连表数据查询


1.0 / http://f77a20dfdf4f.faas-fintech.cmbchina.cn/query
Save
POST
http://f77a20dfdf4f.faas-fintech.cmbchina.cn/query
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"test_case": "query with join",
"method": "query",
"name": "test_table_txx",
"user": {
"name": "Alice",
"password": "123456"
},
"cols": [
"*"
],
"joinTable": "test_join_txx",
"joinCond": "test_table_txx.name = test_join_txx.name"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 69 ms
Size: 784 B
Save as Example
Pretty
Raw
Preview
Visualize
Text
{"data":[{"id":2,"name":"李四","comm截图(Alt + A)shigehaoren","score":100.0,"create_time":1685590544000,"update_time":1685590547000,"city":"上海"},
{"id":2,"name":"李四","comment":"lisi shigehaoren","score":1.0,"create_time":1685590544000,"update_time":1685590547000,"city":"上海"},{"id":1,
"name":"张三","comment":"test multiple lines insert, line1","score":23.98,"create_time":1685590516000,"update_time":1685590613000,"city":"北京"}
,{"id":2,"name":"李四","comment":"lisi shigehaoren","score":1.0,"create_time":1685590544000,"update_time":1685590547000,"city":"上海"}],
"message":"success","status":200}

功能实现：请求校验与错误提示



感谢聆听，请老师同学指教

第四组

杜星亮 任双寅 韩畅 黄毅婷

2023.06.04