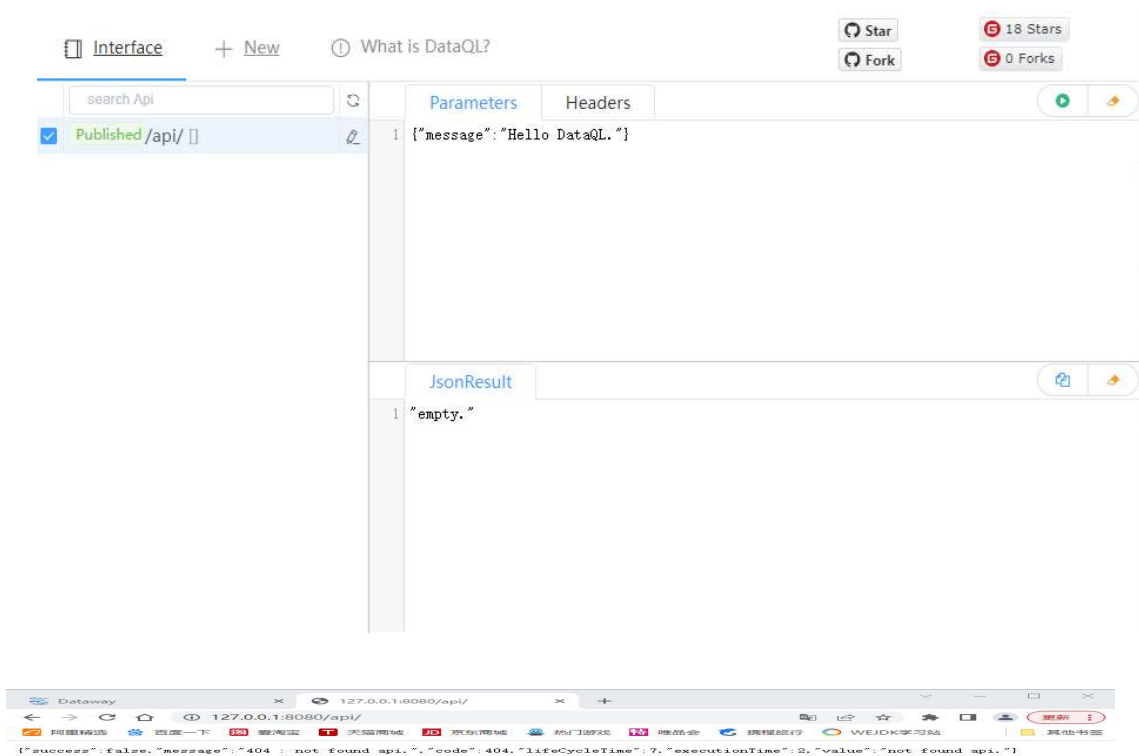


上级需求，期望实现的目标是：

- 1、可使用 SQL 或 DataQL 配置出一个查询 API， 供前端调用；
- 2、支持多数据源， 可动态配置
- 3、通过 DataQL 可做一些简单的数据处理， 也需要研究看它能实现到哪一步。
- 4、跨库多表关联查询，（理论上 SQL 带上库名应该是可以的， 但确认一下）
- 5、如何实现分页查询

问题总结：

## 1、可使用 SQL 或 DataQL 配置出一个查询 API， 供前端调用；



问题：

- 1、前端开发者，只能 JSON 方式来传参，x-www-form-urlencoded 方式传参无法接收  
<https://github.com/ClouGence/hasor/issues/53>

方案：在 dataway 微服务的过滤器里面进行处理，方案类似与 spring-redis-session，对于 x-www-form-urlencoded 的 request 使用拦截器，进行特殊处理为 json 数据

## 2.多数据源动态配置

- (1) 动态配置：从 nacos 动态拉取配置见附件工程：springboot-dataway-nacos
- (2) 多数据库支持 sqlserver、mysql 等，（JDBC 层的兼容性）
- (3) 支持 Redis （待验证）<https://github.com/ClouGence/hasor/issues/60>

## 3.通过 DataQL 可做一些简单的数据处理

-1、DataQL 具有基本的图灵可备性，符合基本编程语言规范

```DataQL

```
var convertSex = (sex) -> {  
    return (sex == 'F') ? '男' : '女'  
};
```

```
var data = {  
    "userID" : 1234567890,  
    "age"      : 31,  
    "name"     : "this is name.",  
    "nick"     : "my name is nick.",  
    "sex"      : "F",  
    "status" : true  
};
```

```
return data => {  
    "name",  
    "age" : age + "岁",  
    "sex" : convertSex(sex)  
}  
```
```

<https://www.dataql.net/docs/dataql/syntax/lexical>

-2、DataQL 支持内部插入 sql

// 声明一个 SQL

```
var dataSetFun = @@sql(itemCode) <%  
    select * from category where co_code = #{itemCode} limit 10;  
%>
```

// 执行这个 SQL，并返回结果

```
return dataSetFun("#{itemCode});
```

<https://www.dataql.net/docs/dataql/sql/about>

-3、DataQL 支持开发 UDF 插件具有灵活的扩展性

```

|
| @FunctionalInterface
| public interface Udf {
|     /** UDF 的返回值必须是一个 对象或者数组 */
|     public default Object call(Object... params) throws Throwable {
|         return call(new HintsSet(), params);
|     }
|
|     /** UDF 的返回值必须是一个 对象或者数组 */
|     public Object call(Hints readOnly, Object... params) throws Throwable
| }

```

案例（待验证）使用 UDF 插件实现对 Redis 的访问

<https://github.com/ClouGence/hasor/issues/60>

问题：

1、目前项目只是纯 select 业务,如果是 update、insert 要考虑事务问题

方案：利用 hasor 生态下面的一个事务库

```DataQL

import 'net.hasor.dataql.fx.db.TransactionUdfSource' as tran;

```

## 4、跨库多表关联查询 (SQL)

使用 import 其他已经写好的 DataQL 查询，在主 DataQL 和引入的 DataQL hint 不同的数据源，可以实现分别查询

<https://github.com/ClouGence/hasor/issues/87#issuecomment-783012488>

### 5、如何实现分页查询

Hasor 有分页插件

步骤 1: 定义分页 SQL

```
hint FRAGMENT_SQL_QUERY_BY_PAGE = true
```

```
var dimSQL = @@sql(userName)<%
```

```
    select * from user_info where `name` like concat('%',{userName},'%')
```

```
%>;
```

步骤 2: 获取分页对象

```
var queryPage = dimSQL(${userName});
```

步骤 3: 设置分页信息

```
run queryPage.setPageInfo({
```

```
    "pageSize"    : 5, // 页大小
```

```
    "currentPage": 3  // 第 3 页
```

```
});
```

```
// 步骤 1: 定义分页 SQL
hint FRAGMENT_SQL_QUERY_BY_PAGE = true
var dimSQL = @@sql(userName)<%
    select * from user_info where `name` like concat('%',{userName},'%') order by id asc
%>;

// 步骤 2: 获取分页对象
var queryPage = dimSQL(${userName});

// 步骤 3: 设置分页信息
run queryPage.setPageInfo({
    "pageSize"    : 5, // 页大小
    "currentPage": 3  // 第 3 页
});

return queryPage.data()
// {
//   "data" :
// }
```

分页查询

Parameters Headers

1 {"userName": ""}

所有数据

JsonResult

```
1 {
2   "success": true,
3   "message": "OK",
4   "code": 0,
5   "lifeCycleTime": 10,
6   "executionTime": 7,
7   "value": [
8     {
9       "id": 16,
10      "name": "user-16",
11      "sex": 0
12    },
13    {
14      "id": 17,
15      "name": "user-17",
16      "sex": 1
17    },
18    {
19      "id": 18,
20      "name": "user-18",
21      "sex": 0
22    },
23    {
24      "id": 19,
25      "name": "user-19",
26      "sex": 1
27    }
28  ]
29 }
```

问题:

### 1、分页查询需要按照规范进行编写,否则会导致全表查询

使用第二种写法,当查询的数据库数据量大时,会导致服务器CPU使用过高

我查询的库数据量在 90W,执行,页面一直 loading

cpu 持续 300%

服务打印错误日志:

```
net.hasor.dataql.runtime.InstructRuntimeException: [line 14:22~14:28 ,QIL 0:28] GC overhead limit exceeded
at net.hasor.dataql.runtime.mem.RefCall.lambda$invokeMethod$0(RefCall.java:63) ~[hasor-dataql-4.2.0.jar!/:na]
at net.hasor.utils.ExceptionUtils.toRuntimeException(ExceptionUtils.java:40) ~[hasor-commons-4.2.0.jar!/:na]
at net.hasor.dataql.runtime.mem.RefCall.invokeMethod(RefCall.java:62) ~[hasor-dataql-4.2.0.jar!/:na]
at net.hasor.dataql.runtime.inset.CALL.doWork(CALL.java:57) ~[hasor-dataql-4.2.0.jar!/:na]
at net.hasor.dataql.runtime.inset.OpcodesPool.doWork(OpcodesPool.java:45) ~[hasor-dataql-4.2.0.jar!/:na]
at net.hasor.dataql.runtime.QueryImpl.execute(QueryImpl.java:83) ~[hasor-dataql-4.2.0.jar!/:na]
at net.hasor.dataql.runtime.QueryImpl.execute(QueryImpl.java:37) ~[hasor-dataql-4.2.0.jar!/:na]
at net.hasor.dataql.Query.execute(Query.java:50) ~[hasor-dataql-4.2.0.jar!/:na]
at net.hasor.dataway.service.ApiCallService._doCall(ApiCallService.java:143) [hasor-dataway-4.2.0.jar!/:na]
at net.hasor.dataway.service.ApiCallService.doCallWithoutError(ApiCallService.java:54) [hasor-dataway-4.2.0.jar!/:na]
at net.hasor.dataway.web.PerformController.doPerform(PerformController.java:78) [hasor-dataway-4.2.0.jar!/:na]
at net.hasor.dataway.web.PerformController$Auto$6.aop$doPerform(Unknown Source) [na:na]
```

方案:

正确的写法

// 设置分页信息

```
run pageQuery.setPageInfo({
    "pageSize" : ${pageSize},
    "currentPage" : ${currentPage}
});
```

```
var result =pageQuery.data() =>[
{
    "zl":ZL,
    "qlr":QLR,
    "xzld":XZ_ID
}
];
return {
    "total": pageQuery.pageInfo(),
    "result": result
}
```

错误的写法

```
var result =pageQuery.data() =>[
{
    "zl":ZL,
    "qlr":QLR,
    "xzld":XZ_ID
}
];
```

```
// 设置分页信息
run pageQuery.setPageInfo({
  "pageSize" : ${pageSize},
  "currentPage" : ${currentPage}
});

return {
  "total": pageQuery.pageInfo(),
  "result": result
}
```

## 前端部分

根据查询语句 响应数据 json 实现动态页面展示，调研中