Atitit 数据源连接查询数据拼接解决方案

目录

[1.1. 优先使用json字段存储解决减少数据拼接 1](#_Toc32476)

[1.2. 通用查询，子查询对象拼接 1](#_Toc18634)

[2. 通用查询，子查询对象拼接 2](#_Toc30381)

[2.1. 多个子查询分号分隔，子查询sql模式 2](#_Toc11181)

[2.2. 外接key使用${}模式 2](#_Toc27551)

[3. other 5](#_Toc32292)

## 优先使用json字段存储解决减少数据拼接

## 通用查询，子查询对象拼接

@Test

**public** **void** queryPageSubquery() **throws** Exception {

MockHttpServletRequest reqt = **new** MockHttpServletRequest();

reqt.addParameter("from", "ball\_match\_t");

reqt.addParameter("page", "1");

reqt.addParameter("pagesize", "3");

reqt.addParameter("subquery", " ( select \* from ball\_score\_t where match\_id={id} limit 1 ) as zhudui\_scoreObj");

ApiController ApiControllerForAdv = **new** ApiController();

ApiControllerForAdv.MybatisMapper1 = MybatisUtil.*getMybatisMapper*();

// assertNotNull(object);

ApiControllerForAdv.req = reqt;

Object bizSql = ApiControllerForAdv.queryPage();

LinkedHashMap li = (LinkedHashMap) bizSql;

System.***out***.println(li.get("count").toString());

System.***out***.println( JSON.*toJSONString*(li, **true**));

}

**public** **static** **void** main(String[] args) **throws** Exception {

MockHttpServletRequest reqt = **new** MockHttpServletRequest();

reqt.addParameter("from", "ball\_match\_t");

reqt.addParameter("page", "1");

reqt.addParameter("pagesize", "3");

String sbq = " ( from ball\_score\_t where match\_id={id} limit 1 ) as zhudui\_scoreObj;( from ball\_score\_t where match\_id={id} and team\_type=2 limit 1 ) as kedui\_scoreObj";

reqt.addParameter("subquery", sbq);

System.out.println( URLEncoder.encode(sbq,"utf8"));

ApiController ApiControllerForAdv = **new** ApiController();

ApiControllerForAdv.MybatisMapper1 = MybatisUtil.getMybatisMapper();

// assertNotNull(object);

ApiControllerForAdv.req = reqt;

Object bizSql = ApiControllerForAdv.queryPage();

LinkedHashMap li = (LinkedHashMap) bizSql;

System.out.println(li.get("count").toString());

System.out.println( JSON.toJSONString(li, **true**));

# 通用查询，子查询对象拼接

## 多个子查询分号分隔，子查询sql模式

## 外接key使用${}模式

@Test

**public** **void** queryPageSubquery() **throws** Exception {

MockHttpServletRequest reqt = **new** MockHttpServletRequest();

reqt.addParameter("from", "ball\_match\_t");

reqt.addParameter("page", "1");

reqt.addParameter("pagesize", "3");

reqt.addParameter("subquery", " ( select \* from ball\_score\_t where match\_id={id} limit 1 ) as zhudui\_scoreObj");

ApiController ApiControllerForAdv = **new** ApiController();

ApiControllerForAdv.MybatisMapper1 = MybatisUtil.*getMybatisMapper*();

// assertNotNull(object);

ApiControllerForAdv.req = reqt;

Object bizSql = ApiControllerForAdv.queryPage();

LinkedHashMap li = (LinkedHashMap) bizSql;

System.***out***.println(li.get("count").toString());

System.***out***.println( JSON.*toJSONString*(li, **true**));

}

**public** **static** **void** main(String[] args) **throws** Exception {

MockHttpServletRequest reqt = **new** MockHttpServletRequest();

reqt.addParameter("from", "ball\_match\_t");

reqt.addParameter("page", "1");

reqt.addParameter("pagesize", "3");

String sbq = " ( from ball\_score\_t where match\_id={id} limit 1 ) as zhudui\_scoreObj;( from ball\_score\_t where match\_id={id} and team\_type=2 limit 1 ) as kedui\_scoreObj";

reqt.addParameter("subquery", sbq);

System.out.println( URLEncoder.encode(sbq,"utf8"));

ApiController ApiControllerForAdv = **new** ApiController();

ApiControllerForAdv.MybatisMapper1 = MybatisUtil.getMybatisMapper();

// assertNotNull(object);

ApiControllerForAdv.req = reqt;

Object bizSql = ApiControllerForAdv.queryPage();

LinkedHashMap li = (LinkedHashMap) bizSql;

System.out.println(li.get("count").toString());

System.out.println( JSON.toJSONString(li, **true**));

/\*\*处理子查询

\* problm leave::: 1. fldval maybe int 2.mult fld on...

\* **@param** m

\* **@param** rzt

\* **@param** mybatisMapper1

\*/

**private** **static** **void** processSubquery(Map m5, List<Map> rzt, MybatisMapper mybatisMapper1) {

**if** (m5.get("subquery") == **null**)

**return**;

String sqls=m5.get("subquery") .toString().trim();

String[] a=sqls.split(";");

List<Map> li\_subquery=Lists.*newArrayList*();

**for** (String sql : a) {

LinkedHashMap m=Maps.*newLinkedHashMap*();

**int** lastAs=sql.lastIndexOf("as");

String asObj=sql.substring(lastAs+2).trim();

**int** lastBrack=sql.lastIndexOf(")");

sql=sql.substring(1,lastBrack);

// sql="select \* from "+sql;

List<String> outCols=regFind.*SqlParam*(sql);

m.put("asObj", asObj.toString().trim());

m.put("sql", sql);

m.put("outCols", outCols);

// // 新建 MySQL Parser

// SQLStatementParser parser = new MySqlStatementParser(sql);

//

// // 使用Parser解析生成AST，这里SQLStatement就是AST

// // 安全设置2，只允许执行一条sql only one sql,,

// List<SQLStatement> statements = parser.parseStatementList();

// SQLStatement statement = statements.get(0);

li\_subquery.add(m);

}

**for** (Map rec : rzt) {

**for** (Map subqM : li\_subquery) {

List<String> outCols=(List<String>) subqM.get("outCols");

String sql=subqM.get("sql").toString().trim();

**for** (String outCol : outCols ) {

String onfldVal="'"+rec.get(outCol).toString()+"'";

sql=sql.replaceAll("\\{"+outCol+"\\}", onfldVal);

// 简化使用性ux，增加select和limit参数的默认模式，翻页转换page等

}

sql=sql.trim();

**if**(!sql.startsWith("select"))

sql="select \* "+sql;

HashMap m=Maps.*newHashMap*();

// 安全性检查过滤

m.put("$blacklistTbs", "sys\_user\_t");

sqlutil.*checkSafe*(m,sql);

//process where

// convert mybatis fmt fragment

//paddParam(subqM);

List rs = mybatisMapper1.querySql(sql);

Object asObj=subqM.get("asObj");

rec.put(asObj.toString().trim(), rs);

}

}

}

# other

后端视图 json——object拼接 （麻烦mid

String contgack 拼接 （麻烦，用json obj fun代替

Mybatis xml 数据集拼接 （麻烦

代码里面拼接（麻烦