Atitit 利用hb hqlparser解析jpql范例

Atitit jqpl的解析与相关类

Atitit jqpl的解析与相关类

目录

[1.1. 相关历程 1](#_Toc32634)

[1.2. 打印还可以 3](#_Toc25833)

[1.3. 使用node转换json则可以。。。 4](#_Toc29737)

[1.4. 查询所欲字段的解决 6](#_Toc14961)

[1.5. 小结与sql比较 6](#_Toc13167)

[2. Wehre ast 7](#_Toc11560)

[2.1. 单条件 7](#_Toc31059)

[2.2. And 逻辑条件 7](#_Toc29310)

[2.3. 函数调用ast 7](#_Toc4330)

# JPQL（Java Presistence Query Language ）和hibernate HQL很相似。

2、JPQL（Java Presistence Query Language ）是EJB3.0中的JPA造出来的对象查询语言。JPQL是完全面向对象的，具备继承、多态和关联等特性，

和hibernate HQL很相似。

## 相关历程

返回form之前的token位数

getStartingPositionFor

自己调试，但是不好调试太复杂

//new HqlParser(hql);  
String[] concreteQueryStrings = QuerySplitter.*concreteQueries*(hql, SessionFactoryImplAtiimp.*newSessionFactoryImplementor*());  
  
  
HQLQueryPlan qplan=new HQLQueryPlan(hql,false,enabledFilters, SessionFactoryImplAtiimp.*newSessionFactoryImplAti*());  
System.*out*.println(qplan);

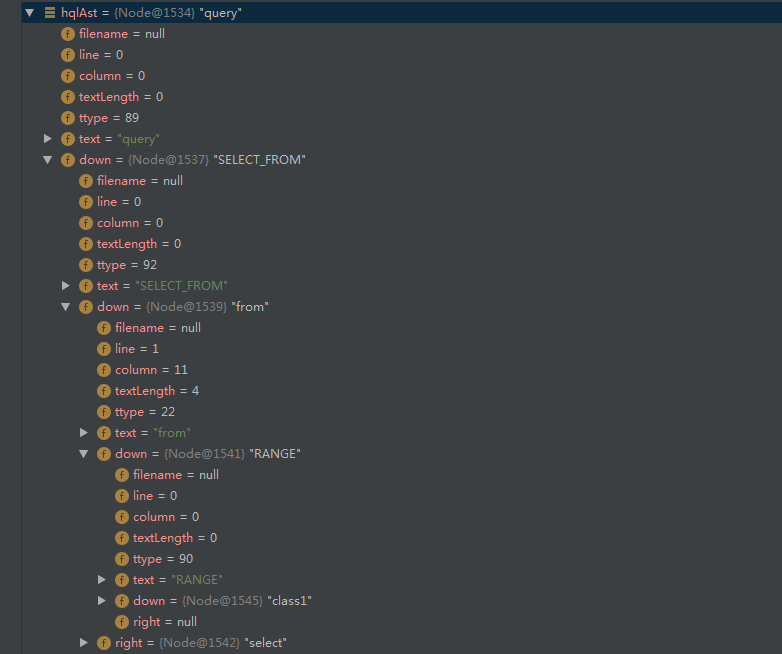
，找源码下载没有找到。。

自己翻看类 有个 HqlParser，，单调用没有得到ast

上网查询，终于搞到语句

public class JqplHqlParser {  
@SneakyThrows  
 public static void main(String[] args) {  
  
 // try{}catch{}  
  
 Map<String, Filter> enabledFilters= Maps.*newLinkedHashMap*();  
  
 String hql = "select \* from class1";  
  
 HqlParser hqlParser1= HqlParser.*getInstance*(hql);  
 hqlParser1.statement();  
  
 final AST hqlAst = hqlParser1.getAST();  
 System.*out*.println(JSON.*toJSONString*(hqlAst,true));  
  
 NodeTraverser walker = new NodeTraverser( new QueryTranslatorImpl.JavaConstantConverter( null ) );  
 walker.traverseDepthFirst( hqlAst );  
 hqlParser1.showAst(hqlAst,System.*out*);  
  
  
 Map m=hqlParser1.getTreatMap();  
  
   
 }  
}

但是ast json序列号不行啊，只能调试器查看



## 打印还可以

\-[QUERY] Node: 'query'

\-[SELECT\_FROM] Node: 'SELECT\_FROM'

\-[FROM] Node: 'from'

\-[RANGE] Node: 'RANGE'

\-[IDENT] Node: 'class1'

String hql = "select c1 from class1";

\-[QUERY] Node: 'query'

\-[SELECT\_FROM] Node: 'SELECT\_FROM'

+-[FROM] Node: 'from'

| \-[RANGE] Node: 'RANGE'

| \-[IDENT] Node: 'class1'

\-[SELECT] Node: 'select'

\-[IDENT] Node: 'c1'

## 使用node转换json则可以。。。

public class JqplHqlParser {  
@SneakyThrows  
 public static void main(String[] args) {  
  
 // try{}catch{}  
  
 Map<String, Filter> enabledFilters= Maps.*newLinkedHashMap*();  
  
 String hql = "select c1 from class1";  
  
 HqlParser hqlParser1= HqlParser.*getInstance*(hql);  
 hqlParser1.statement();  
  
 final AST hqlAst = hqlParser1.getAST();  
 Node node1= (org.hibernate.hql.internal.ast.tree.Node) hqlAst;  
 System.*out*.println( JSON.*toJSONString*(node1,true) );  
  
 // System.out.println(JSON.toJSONString(hqlAst,true));  
  
 NodeTraverser walker = new NodeTraverser( new QueryTranslatorImpl.JavaConstantConverter( null ) );  
 walker.traverseDepthFirst( hqlAst );  
 hqlParser1.showAst(hqlAst,System.*out*);  
  
  
  
  
 System.*out*.println("--f");  
 }  
}

{

"column":0,

"firstChild":{

"column":0,

"firstChild":{

"column":11,

"firstChild":{

"column":0,

"firstChild":{

"column":16,

"line":1,

"numberOfChildren":0,

"text":"class1",

"textLength":6,

"type":108

},

"line":0,

"numberOfChildren":1,

"text":"RANGE",

"textLength":0,

"type":90

},

"line":1,

"nextSibling":{

"column":1,

"firstChild":{

"column":8,

"line":1,

"numberOfChildren":0,

"text":"c1",

"textLength":2,

"type":108

},

"line":1,

"numberOfChildren":1,

"text":"select",

"textLength":6,

"type":45

},

"numberOfChildren":1,

"text":"from",

"textLength":4,

"type":22

},

"line":0,

"numberOfChildren":2,

"text":"SELECT\_FROM",

"textLength":0,

"type":92

},

"line":0,

"numberOfChildren":1,

"text":"query",

"textLength":0,

"type":89

}

## 查询所欲字段的解决

\*信号不行，doller不行，汉子可以 ，百分号不行

内置下划线开头字段可以 '\_all'

## **小结与sql比较**

**貌似不支持insert？？**



# Wehre ast

## 单条件

\-[WHERE] Node: 'where'

\-[EQ] Node: '='

+-[IDENT] Node: 'c1'

\-[NUM\_INT] Node: '123'

--f

## And 逻辑条件

String hql = "select \_all from class1 where c1=123 and c2=456";

\-[QUERY] Node: 'query'

+-[SELECT\_FROM] Node: 'SELECT\_FROM'

| +-[FROM] Node: 'from'

| | \-[RANGE] Node: 'RANGE'

| | \-[IDENT] Node: 'class1'

| \-[SELECT] Node: 'select'

| \-[IDENT] Node: '\_all'

\-[WHERE] Node: 'where'

\-[AND] Node: 'and'

+-[EQ] Node: '='

| +-[IDENT] Node: 'c1'

| \-[NUM\_INT] Node: '123'

\-[EQ] Node: '='

+-[IDENT] Node: 'c2'

\-[NUM\_INT] Node: '456'

## 函数调用ast

String hql = "select \_all from info\_class1 where c1=123 and fun1(c2) >456";

\-[WHERE] Node: 'where'

\-[AND] Node: 'and'

+-[EQ] Node: '='

| +-[IDENT] Node: 'c1'

| \-[NUM\_INT] Node: '123'

\-[GT] Node: '>'

+-[METHOD\_CALL] Node: '('

| +-[IDENT] Node: 'fun1'

| \-[EXPR\_LIST] Node: 'exprList'

| \-[IDENT] Node: 'c2'

\-[NUM\_INT] Node: '456'

# Jpql到sql的翻译

import org.hibernate.hql.internal.ast.util.ASTPrinter;

public void showAst(AST ast, PrintWriter pw) {  
 ArrayList<AST> parents = new ArrayList();  
 this.showAst(parents, pw, ast);  
 pw.flush();  
}

private void showAst(ArrayList<AST> parents, PrintWriter pw, AST ast) {  
 if (ast == null) {  
 pw.println("AST is null!");  
 } else {  
 Iterator var4 = parents.iterator();  
  
 AST child;  
 while(var4.hasNext()) {  
 child = (AST)var4.next();  
 if (child.getNextSibling() == null) {  
 pw.print(" ");  
 } else {  
 pw.print(" | ");  
 }  
 }  
  
 if (ast.getNextSibling() == null) {  
 pw.print(" \\-");  
 } else {  
 pw.print(" +-");  
 }  
  
 this.showNode(pw, ast);  
 ArrayList<AST> newParents = new ArrayList(parents);  
 newParents.add(ast);  
  
 for(child = ast.getFirstChild(); child != null; child = child.getNextSibling()) {  
 this.showAst(newParents, pw, child);  
 }  
  
 newParents.clear();  
 }  
}

hibernate中antlr对于hql的词法分析源码解析 - 普 通 上 班 族 - CSDN博客.html

HQL & JPQL - Part Ⅰ - 简书.html

JPQL\_百度百科.html