Atitit 发帖机系列（8） 词法分析器v5 版本新特性说明）

v5 增加对sql单引号的内部支持。可以作为string

结构调整，使用递归法重构循环发。。放弃循环发。

V4 java dsl词法分析 使用循环

V3 sql的词法分析

### [atitit..sql update语法的词法分析,与语法ast构建 -](http://www.baidu.com/link?url=2KjauhPFVtWQGskBUuXYBE5HPOtG_8d_g2sqp9Bd_JjKRU-_rIht0JIEDFrVAwwCVlp5dvtE7do5qX8R4TAyaq" \t "https://www.baidu.com/_blank)

V1版 anno 解析器 基于fsm设计模式

V2

### [Atitit.antlr实现词法分析 - attilax](http://www.baidu.com/link?url=Ihgo55lEzADYvZzaCk5lGcw1DykxEF0RCf6M7TUlSgc4U8fS-fjn2xnKKB4EdiYWhqWtPArm2sSsVCVwP6eul_" \t "https://www.baidu.com/_blank)

/AtiPlatf\_ee/src/com/attilax/fsm/JavaTokenScannerV2.java

**package** com.attilax.fsm;

**import** java.util.Collection;

**import** java.util.List;

**import** com.attilax.io.filex;

**import** com.attilax.io.pathx;

**import** com.attilax.json.AtiJson;

**import** com.google.common.collect.Lists;

**public** **class** JavaTokenScannerV2 {

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

String **f** = pathx.*prjPath\_webrootMode*() + "/tokenT/a.txt";

String **s** = "meth(\\\"select from table where a='bb' \",'str2',\'s3\') ";

s = "meth(\"select from tab where a='abc'\",'str2','s3')";

// s = filex.read(f);

List<Token> **li** = **new** JavaTokenScannerV2().getTokens(s);

System.***out***.println(AtiJson.*toJson*(li));

}

List<Token> tokens = Lists.*newArrayList*();

String curTokenTxt = "";

String splitors = "(),";

String curStat = "ini";

String curDbquoStat = "ini";

**private** String code;

**public** **char**[] code\_char\_arr;

**public** **char** cur\_char;

**int** gColumn = -1;

@SuppressWarnings("unchecked")

**public** List<Token> **getTokens**(String codeStr) {

List<Token> **li** = Lists.*newArrayList*();

code\_char\_arr = codeStr.toCharArray();

**while** (**true**) {

Object **tk**;

**try** {

tk = nextTokens();

} **catch** (TokenEndEx **e**) {

**break**;

}

**if** (tk **instanceof** Token)

li.add((Token) tk);

**else** **if** (tk **instanceof** List)

li.addAll((Collection<? **extends** Token>) tk);

**else**

**throw** **new** RuntimeException("token type err,curchar:" + **this**.cur\_char + ",colidx:" + **this**.gColumn);

}

**return** li;

}

**public** Object **nextTokens**() **throws** TokenEndEx {

// code\_char\_arr = code.toCharArray();

gColumn++;

**if** (gColumn > code\_char\_arr.length - 1)

**throw** **new** TokenEndEx(**new** String(code\_char\_arr));

cur\_char = code\_char\_arr[gColumn];

// cur\_char=cur\_char;

**if** (**this**.curTokenTxt.equals("1598"))

System.***out***.println("dbg");

**if** (**this**.gColumn == 30)

System.***out***.println("dbg");

// get next char,,then change stat

// jude cur char and cur stat...then if or not chage stat

**switch** (cur\_char) {

**case** '(':

**return** BrkStartEvt();

// break;

**case** ')':

**return** brkEndEvt();

**case** '\'':

**return** sQuoEvt();

**case** '\"':

**return** dbQuoEvt();

**case** ':':

**return** colonEvt();

**case** ',':

**return** commaEvt();

**default**:

**return** normalCharEvt();

// break;

}

}

**private** Object **normalCharEvt**() **throws** TokenEndEx {

// normal char

curTokenTxt = curTokenTxt + String.*valueOf*(cur\_char);

// gColumn += 1;

**return** nextTokens();

}

**private** Object **commaEvt**() **throws** TokenEndEx {

List<Token> **tokens** = Lists.*newArrayList*();

**if** (cur\_char == ',' && !**this**.curStat.equals("squoStart") && !**this**.curStat.equals("dbquoStart")) {

**if** (**this**.curTokenTxt.trim().length() > 0) {

String **curTokenTye** = "con";

**if** (**this**.curTokenTxt.startsWith(":"))

curTokenTye = "op";

**else** **if** (**this**.curStat.equals("colon"))

curTokenTye = "op";

Token **tk4** = **new** Token(**this**.curTokenTxt).setType(curTokenTye);

tokens.add(tk4);

**return** tokens;

}

Token **tk3** = **new** Token(String.*valueOf*(cur\_char)).setType("spltr");

tokens.add(tk3);

**this**.curTokenTxt = "";

**this**.curStat = "commaStat";

**return** tk3;

}

**return** normalCharEvt();

}

**private** Object **colonEvt**() {

List<Token> **tokens** = Lists.*newArrayList*();

**if** (cur\_char == ':' && !**this**.curStat.equals("strStart")) {

**if** (**this**.curTokenTxt.trim().length() > 0) {

String **curTokenTye** = "con";

Token **tk4** = **new** Token(**this**.curTokenTxt).setType(curTokenTye);

tokens.add(tk4);

}

Token **tk3** = **new** Token(String.*valueOf*(cur\_char)).setType("op");

tokens.add(tk3);

**this**.curTokenTxt = "";

**this**.curStat = "colon";

**return** tokens;

}

**return** tokens;

}

**private** Object **dbQuoEvt**() **throws** TokenEndEx {

// ---------------str type

// first dbquo

**if** ((cur\_char == '\"') && (**this**.curDbquoStat != "dbquoStart")) //

{

**this**.curStat = "dbquoStart";

**this**.curDbquoStat = "dbquoStart";

**this**.curTokenTxt = "";

**return** nextTokens();

}

// for close dbquo

**if** ((cur\_char == '\"') && **this**.curDbquoStat.equals("dbquoStart")) {

Token **tk3** = **new** Token(**this**.curTokenTxt).setType("str");

tokens.add(tk3);

**this**.curTokenTxt = "";

**this**.curStat = "dbquoEnd";

**this**.curDbquoStat = "dbquoEnd";

**return** tk3;

}

**return** normalCharEvt();

}

**private** Object **sQuoEvt**() **throws** TokenEndEx {

**char** **c** = **this**.cur\_char;

// for in dbquo ..single quo ..none process as normal char

// first single quoe

**if** (c == '\'' && (**this**.curStat != "dbquoStart") && **this**.curStat != "squoStart") //

{

**this**.curStat = "squoStart";

**this**.curTokenTxt = "";

**return** nextTokens();

}

// for close single quoue

**if** ((c == '\'') && **this**.curStat.equals("squoStart")) {

Token **tk3** = **new** Token(**this**.curTokenTxt).setType("str");

tokens.add(tk3);

**this**.curTokenTxt = "";

**this**.curStat = "squoEnd";

**return** tk3;

}

**return** normalCharEvt();

}

**private** Object **brkEndEvt**() {

**char** **c** = **this**.cur\_char;

**if** (c == ')' && !**this**.curStat.equals("strStart")) { // && cur stat

// =brk start

**if** (**this**.curTokenTxt.length() > 0) // jeig cant smp... last end brk

// is impt..if smp continue

// ,then cant add end brk to

// token

{

String **type** = gettype\_4curCharIsBrkend(**this**.curTokenTxt, **this**.curStat);

Token **tk3** = **new** Token(**this**.curTokenTxt).setType(type);

tokens.add(tk3);

}

Token **tk2** = **new** Token(")").setType("op");

tokens.add(tk2);

**this**.curTokenTxt = "";

**this**.curStat = "brkEnd";

**return** tk2;

}

**return** c;

}

**private** Object **BrkStartEvt**() {

**char** **c** = **this**.cur\_char;

**if** (c == '(' && !**this**.curStat.equals("strStart")) { // && cur stta=ini

List<Token> **li** = Lists.*newArrayList*();

Token **tk** = **new** Token(**this**.curTokenTxt).setType("var");

li.add(tk);

Token **tk2** = **new** Token("(").setType("op");

li.add(tk2);

**this**.curTokenTxt = "";

**this**.curStat = "brkStart";

**return** li;

}

**throw** **new** RuntimeException("BrkStartEvt");

}

// if (c == '[' && !this.curStat.equals("strStart"))

// {

// this.curStat="sqBrkStart";

// Token tk2=new Token("[").setType("spltr");

// tokens.add(tk2);

// this.curTokenTxt = "";

// continue;

//

// }

//

// if (c == ']' && !this.curStat.equals("strStart"))

// {

// this.curStat="sqBrkEnd";

// Token tk2=new Token("]").setType("spltr");

// tokens.add(tk2);

// this.curTokenTxt = "";

// continue;

// }

/\*

\* if (c == '.' && this.curStat.equals("brkEnd")) {

\*

\* Token tk2=new Token(".").setType("op"); tokens.add(tk2); curTokenTxt =

\* ""; continue; }

\*

\* if(c==' ' && this.curStat.equals("ini")) {

\* if(this.curTokenTxt.trim().toLowerCase().equals("new")) { this.curStat =

\* "brkStart"; Token tk=new Token(this.curTokenTxt).setType("keyword");

\* tokens.add(tk); this.curTokenTxt = ""; continue; }

\*

\* }

\*/

/\*\*

\* attilax 2016年10月24日 下午3:40:27

\*

\* **@param** curTokenTxt2

\* **@param** curStat2

\* **@return**

\*/

**private** String **gettype\_4curCharIsBrkend**(String curTokenTxt2, String lastStat2) {

**if** (curTokenTxt2.contains("."))

**return** "var";

**else** **if** (curTokenTxt2.startsWith(":") && !lastStat2.equals("strStart"))

**return** "op";

**else** **if** (lastStat2.equals("colon"))

**return** "op";

**else** **if** (lastStat2.equals("strEnd"))

**return** "str";

**else** **if** (lastStat2.equals("comma"))

**return** "con";

**return** "con";

}

}

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ）

汉字名：艾提拉（艾龙），   EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

--Atiend