Atitit 语法分析的实现过程attilax总结

[1.1. 前几天正好做rest接口，传入一个类似于java语法 的dsl，后端使用反射调用api。。 1](#_Toc13370)

[1.2. 第一步，你要使用ast view先查看java代码的ast结构。 1](#_Toc240)

[1.3. 第二步，如果需要详细了解java的ast结构体系，需要查阅jdt api资料 2](#_Toc12414)

[1.4. 第三部，构建ast的方法主流使用递归下降法。。至于其他方法较少采用 2](#_Toc5743)

[1.5. 代码如下 2](#_Toc9660)

## 前几天正好做rest接口，传入一个类似于java语法 的dsl，后端使用反射调用api。。

这里我就dsl语法仿照java了，语法仿照dsl代码

" new aaaPKg .DslParser().m2(123) ";

" aaaPKg. DslParser.m3(123) "

一个动态调用new的操作。。一个调用静态方法。。至于方法链嘛。比如

" new aaaPKg.DslParser().m3(11).m4()." 。。我就先不实现了。过两天弄，这两天没时间了。。

## 大概原理。即是在括号，点号那里判断，根据前后元素，构建ast语法树的部件。即可。。

如果多语句，还要在分号那里判断

## 第一步，你要使用ast view先查看java代码的ast结构。

方法调用其实就是构建一个 MethodInvocation 对象。。这个对象扩展来自Expression 。。。 MethodInvocation 对象有个属性express.指明了是动态new调用还是静态static调用。。

如何构建这个属性express.呢？？？

语法解析new方式调用的dsl，，需要构建一个ClassInstanceCreation 对象。。

**public** **class** ClassInstanceCreation **extends** Expression

如果静态方法dsl，则需要构建一个SimpleName对象就可以了。。

## 第二步，如果需要详细了解java的ast结构体系，需要查阅jdt api资料

## 第三部，构建ast的方法主流使用递归下降法。。至于其他方法较少采用

## 代码如下

/AtiPlatf\_cms/src/com/attilax/fsm/JavaTokener.java

List<Token> tokens;

**int** m\_index=0;

**public** Expression buildAstV2(List<Token> tokens) {

**this**.tokens = tokens;

// System.out.println(AtiJson.toJson(tokens));

List<String> tokens\_slice\_li = Lists.*newLinkedList*();

String stat = "ini";

MethodInvocation mi = **new** MethodInvocation();

Expression exp;

**boolean** isConstrutsced = **false**; // qaa is first

**boolean** isNewDysnInvoke = **false**;

**boolean** isStaticInvoke = **false**;

Token cur = tokens.get(m\_index);

**if** (cur.getText().equals("(") && cur.getType().equals("op"))

{

stat = "brkStart";

String text = tokens.get(m\_index - 1).getText();

//newxx first

**if** (isNewOpV2(tokens) && isConstrutsced == **false**)// newxxx

{

String className = javaUtil.*clr*(text);

mi.Exp = getClassInstance(className);

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

m\_index++;

cur = tokens.get(m\_index);

**if** (cur.getText().equals("(") && cur.Type.equals("op")) {

mi.Name=tokens.get(m\_index-1).getText();

//return mi;

**break**;

}

}

// param push

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

m\_index++;

cur = tokens.get(m\_index);

**if** (cur.getText().equals(")") && cur.Type.equals("op")) {

**break**;

}

mi.arguments.add(cur.Text);

}

**return** mi;

}

**if** (isConstrutsced == **false**) { // static //first

// new method invoke exp

String className = refx.*getClassName*(text);

className = javaUtil.*clr*(className);

mi.Exp = **new** SimpleName(className); // cls name

mi.Name = refx.*getMethodName*(text);

isConstrutsced = **true**;

// param push

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

m\_index++;

cur = tokens.get(m\_index);

**if** (cur.getText().equals(")") && cur.Type.equals("op")) {

**break**;

}

mi.arguments.add(cur.Text);

}

**return** mi;

}

// new xxx() meth.chain

/\*

\* if(mi.Name.equals("\_\_con")) mi.Name=text; else //static ,meth

\* chain { MethodInvocation tmp=(MethodInvocation) mi.clone();

\* MethodInvocation mi\_Outer\_new = new MethodInvocation();

\* mi\_Outer\_new.Exp=tmp; mi\_Outer\_new.Name=text; mi=mi\_Outer\_new; }

\* tokens\_slice\_li = Lists.newLinkedList(); continue;

\*/

// tokens\_slice\_li = Lists.newLinkedList();

// continue;

}

m\_index++;

**return** buildAstV2(tokens);

}

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