

06 | 编译器前端工具（一）：用Antlr生成词法、语法分析器

2019-08-26 宫文学

编译原理之美

[进入课程 >](#)



讲述：宫文学

时长 16:12 大小 14.85M



前面的课程中，我重点讲解了词法分析和语法分析，在例子中提到的词法和语法规则也是高度简化的。虽然这些内容便于理解原理，也能实现一个简单的原型，在实际应用中却远远不够。实际应用中，一个完善的编译程序还要在词法方面以及语法方面实现很多工作，我这里特意画了一张图，你可以直观地看一下。

词法方面

1. 数字要能支持浮点数以及二进制、八进制、十六进制的格式。
2. 字符串要能支持转义字符。
3. 标识符要能支持Unicode。
4. 要能支持行注释和块注释等等。

语法方面

1. 支持除了算术运算之外更多的运算，包括关系运算、逻辑运算，并处理好它们的优先级。
2. 支持if、while这样的流程控制语句。
3. 要支持作用域和函数。
4. 支持面向对象特性等等。

如果让编译程序实现上面这么多工作，完全手写效率会有点儿低，那么我们有什么方法可以提升效率呢？答案是借助工具。

编译器前端工具有很多，比如 Lex（以及 GNU 的版本 Flex）、Yacc（以及 GNU 的版本 Bison）、JavaCC 等等。你可能会问了：“那为什么我们这节课只讲 Antlr，不选别的工具呢？”主要有两个原因。

第一个原因是 Antlr 能支持更广泛的目标语言，包括 Java、C#、JavaScript、Python、Go、C++、Swift。无论你用上面哪种语言，都可以用它生成词法和语法分析的功能。而我们就使用它生成了 Java 语言和 C++ 语言两个版本的代码。

第二个原因是 Antlr 的语法更加简单。它能把类似左递归的一些常见难点在工具中解决，对提升工作效率有很大的帮助。这一点，你会在后面的课程中直观地感受到。

而我们今天的目标就是了解 Antlr，然后能够使用 Antlr 生成词法分析器与语法分析器。在这个过程中，我还会带你借鉴成熟的词法和语法规则，让你快速成长。

接下来，我们先来了解一下 Antlr 这个工具。

初识 Antlr

Antlr 是一个开源的工具，支持根据规则文件生成词法分析器和语法分析器，它自身是用 Java 实现的。

你可以[下载 Antlr 工具](#)，并根据说明做好配置。同时，你还需要配置好机器上的 Java 环境（可以在[Oracle 官网](#)找到最新版本的 JDK）。

因为我用的是 Mac，所以我用 macOS 平台下的软件包管理工具 Homebrew 安装了 Antlr，它可以自动设置好 antlr 和 grun 两个命令（antlr 和 grun 分别是 java org.antlr.v4.Tool 和 java org.antlr.v4.gui.TestRig 这两个命令的别名）。这里需要注意的是，你要把 Antlr 的 JAR 文件设置到 CLASSPATH 环境变量中，以便顺利编译所生成的 Java 源代码。

[GitHub](#)上还有很多供参考的语法规则，你可以下载到本地硬盘随时查阅。


现在你已经对 Antlr 有了初步的了解，也知道如何安装它了。接下来，我带你实际用一用 Antlr，让你用更轻松的方式生成词法分析器和语法分析器。

用 Antlr 生成词法分析器

你可能对 Antlr 还不怎么熟悉，所以我会先带你使用前面课程中，你已经比较熟悉的那些语法规则，让 Antlr 生成一个新的词法分析器，然后再借鉴一些成熟的规则文件，把词法分析器提升到更加专业、实用的级别。

Antlr 通过解析规则文件来生成编译器。规则文件以.g4 结尾，词法规则和语法规则可以放在同一个文件里。不过为了清晰起见，我们还是把它们分成两个文件，先用一个文件编写词法规则。

为了让你快速进入状态，我们先做一个简单的练习预热一下。我们创建一个 Hello.g4 文件，用于保存词法规则，然后把之前用过的一些词法规则写进去。

 复制代码

```
1 lexer grammar Hello; //lexer 关键字意味着这是一个词法规则文件，名称是 Hello，要与文件名相同
2
3 // 关键字
4 If :                'if';
5 Int :               'int';
6
7 // 字面量
8 IntLiteral:         [0-9]+;
9 StringLiteral:      '"' .*? '"'; // 字符串字面量
10
11 // 操作符
12 AssignmentOP:       '=' ;
13 RelationalOP:        '>' | '>=' | '<' | '<=' ;
14 Star:                '*';
15 Plus:                '+';
16 Sharp:               '#';
```

```

17 SemiColon:      ';' ;
18 Dot:            '.' ;
19 Comm:           ',' ;
20 LeftBracket :   '[' ;
21 RightBracket:   ']' ;
22 LeftBrace:       '{' ;
23 RightBrace:      '}' ;
24 LeftParen:       '(' ;
25 RightParen:      ')' ;
26
27 // 标识符
28 Id :             [a-zA-Z_] ([a-zA-Z_] | [0-9])*;
29
30 // 空白字符，抛弃
31 Whitespace:      [ \t]+ -> skip;
32 Newline:         ( '\r' '\n'? | '\n') -> skip;

```


你能很直观地看到，每个词法规则都是大写字母开头，这是 Antlr 对词法规则的约定。而语法规则是以小写字母开头的。其中，每个规则都是用我们已经了解的正则表达式编写的。

接下来，我们来编译词法规则，在终端中输入命令：

 复制代码

```
1 antlr Hello.g4
```

这个命令是让 Antlr 编译规则文件，并生成 Hello.java 文件和其他两个辅助文件。你可以打开看一看文件里面的内容。接着，我用下面的命令编译 Hello.java：

 复制代码

```
1 javac *.java
```


结果会生成 Hello.class 文件，这就是我们生成的词法分析器。接下来，我们来写个脚本文件，让生成的词法分析器解析一下：

 复制代码

```
1 int age = 45;
```

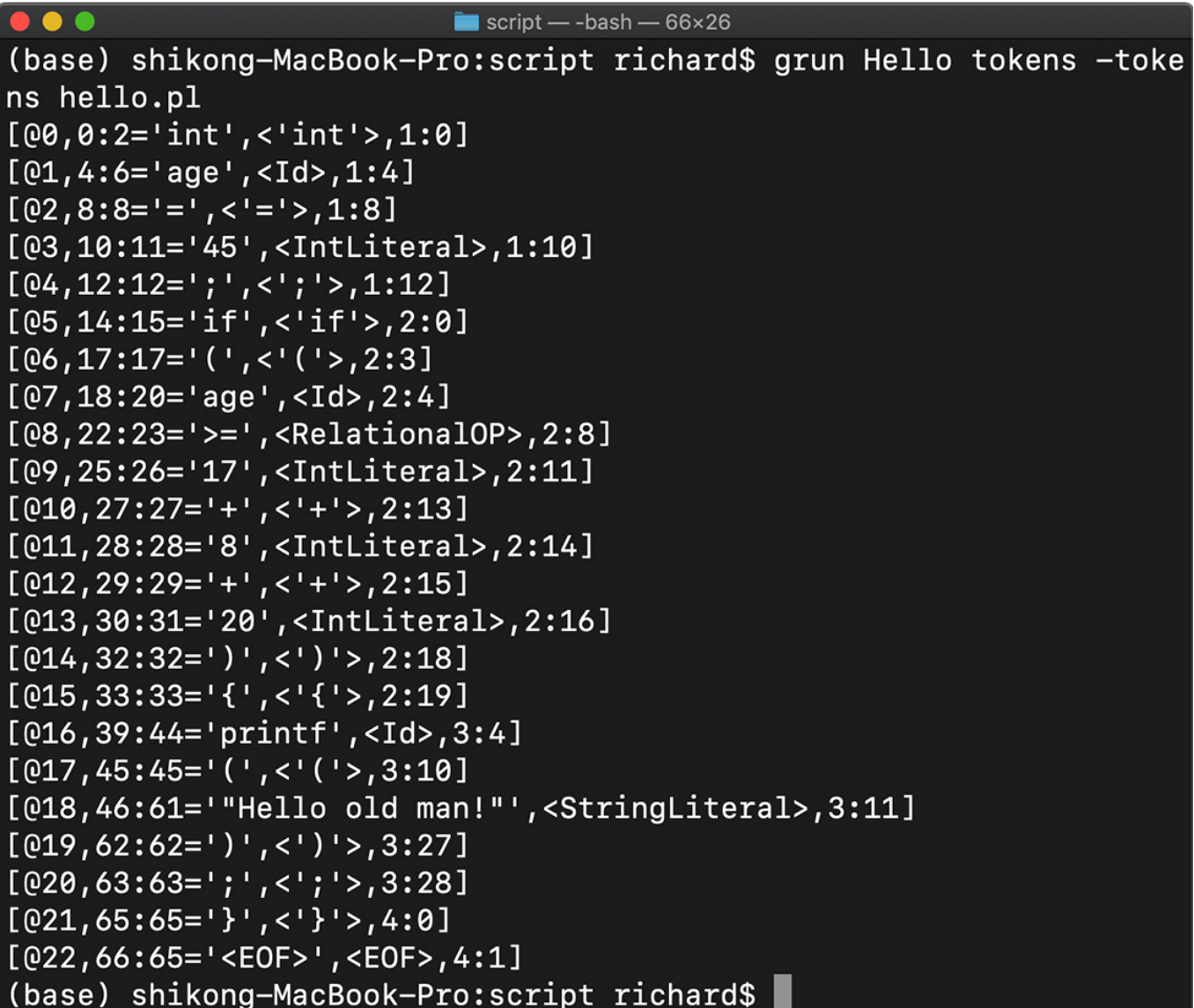
```
2 if (age >= 17+8+20){
3     printf("Hello old man!");
4 }
```

我们将上面的脚本存成 hello.play 文件，然后在终端输入下面的命令：

 复制代码

```
1 grun Hello tokens -tokens hello.play
```

grun 命令实际上是调用了我们刚才生成的词法分析器，即 Hello 类，打印出对 hello.play 词法分析的结果：




```
(base) shikong-MacBook-Pro:script richard$ grun Hello tokens -tokens hello.pl
[00,0:2='int',<'int'>,1:0]
[01,4:6='age',<Id>,1:4]
[02,8:8='=',<'='>,1:8]
[03,10:11='45',<IntLiteral>,1:10]
[04,12:12=';',<'>',1:12]
[05,14:15='if',<'if'>,2:0]
[06,17:17='(',<'('>,2:3]
[07,18:20='age',<Id>,2:4]
[08,22:23='>=',<RelationalOP>,2:8]
[09,25:26='17',<IntLiteral>,2:11]
[010,27:27='+',<'+'>,2:13]
[011,28:28='8',<IntLiteral>,2:14]
[012,29:29='+',<'+'>,2:15]
[013,30:31='20',<IntLiteral>,2:16]
[014,32:32=')',<'>',2:18]
[015,33:33='{',<'{'>,2:19]
[016,39:44='printf',<Id>,3:4]
[017,45:45='(',<'('>,3:10]
[018,46:61='"Hello old man!"',<StringLiteral>,3:11]
[019,62:62=')',<'>',3:27]
[020,63:63=';',<'>,3:28]
[021,65:65='}',<'}'>,4:0]
[022,66:65='<EOF>',<EOF>,4:1]
(base) shikong-MacBook-Pro:script richard$
```


从结果中看到，我们的词法分析器把每个 Token 都识别了，还记录了它们在代码中的位置、文本值、类别。上面这些都是 Token 的属性。

以第二行 [`@1, 4:6= 'age' ,< Id >,1:4`] 为例，其中 `@1` 是 Token 的流水编号，表明这是 1 号 Token；`4:6` 是 Token 在字符流中的开始和结束位置；`age` 是文本值，`Id` 是其 Token 类别；最后的 `1:4` 表示这个 Token 在源代码中位于第 1 行、第 4 列。

非常好，现在我们已经让 Antlr 顺利跑起来了！接下来，让词法规则更完善、更严密一些吧！**怎么做呢？当然是参考成熟的规则文件。**


从 Antlr 的一些示范性的规则文件中，我选了 Java 的作为参考。先看看我们之前写的字符串字面量的规则：

 复制代码

```
1 StringLiteral:      ''' .*? ''' ; // 字符串字面量
```

我们的版本相当简化，就是在双引号可以包含任何字符。可这在实际中不大好用，因为连转义功能都没有提供。我们对于一些不可见的字符，比如回车，要提供转义功能，如 `"\n"`。同时，如果字符串里本身有双引号的话，也要将它转义，如 `"\""`。Unicode 也要转义。最后，转义字符本身也需要转义，如 `"\\"`。

下面这一段内容是 Java 语言中的字符串字面量的完整规则。你可以看一下文稿，这个规则就很细致了，把各种转义的情况都考虑进去了：

 复制代码

```
1 STRING_LITERAL:      ''' (~["\\r\n"] | EscapeSequence)* ''';
2
3 fragment EscapeSequence
4     : '\\' [btnfr"'\n]
5     | '\\' ([0-3]? [0-7])? [0-7]
6     | '\\' 'u'+ HexDigit HexDigit HexDigit HexDigit
7     ;
8
9 fragment HexDigit
10    : [0-9a-fA-F]
11    ;
```

在这个规则文件中，fragment 指的是一个语法片段，是为了让规则定义更清晰。它本身并不生成 Token，只有 StringLiteral 规则才会生成 Token。

当然了，除了字符串字面量，数字字面量、标识符的规则也可以定义得更严密。不过，因为这些规则文件都很严密，写出来都很长，在这里我就不一一展开了。如果感兴趣，我推荐你在下载的规则文件中找到这些部分看一看。你还可以参考不同作者写的词法规则，体会一下他们的设计思路。和高手过招，会更快地提高你的水平。

我也拷贝了一些成熟的词法规则，编写了一个 CommonLexer.g4 的规则文件，这个词法规则是我们后面工作的基础，它基本上已经达到了专业、实用的程度。

在带你借鉴了成熟的规则文件之后，我想穿插性地讲解一下在词法规则中对 Token 归类的问题。在设计词法规则时，你经常会遇到这个问题，解决这个问题，词法规则会更加完善。

在前面练习的规则文件中，我们把 \geq 、 $>$ 、 $<$ 都归类为关系运算符，算作同一类 Token，而 $+$ 、 $*$ 等都单独作为另一类 Token。那么，哪些可以归并成一类，哪些又是需要单独列出的呢？


其实，这主要取决于语法的需要。也就是在语法规则文件里，是否可以出现在同一条规则里。它们在语法层面上没有区别，只是在语义层面上有区别。比如，加法和减法虽然是不同的运算，但它们可以同时出现在同一条语法规则中，它们在运算时的特性完全一致，包括优先级和结合性，乘法和除法可以同时出现在乘法规则中。你把加号和减号合并成一类，把乘号和除号合并成一类是可以的。把这 4 个运算符每个都单独作为一类，也是可以的。但是，不能把加号和乘号作为同一类，因为它们在算术运算中的优先级不同，肯定出现在不同的语法规则中。

我们再来回顾一下在 [“02 | 正则文法和有限自动机：纯手工打造词法分析器”](#) 里做词法分析时遇到的一个问题。当时，我们分析了词法冲突的问题，即标识符和关键字的规则是有重叠的。Antlr 是怎么解决这个问题的呢？很简单，它引入了优先级的概念。在 Antlr 的规则文件中，越是前面声明的规则，优先级越高。所以，我们把关键字的规则放在 ID 的规则前面。算法在执行的时候，会首先检查是否为关键字，然后才会检查是否为 ID，也就是标识符。

这跟我们当时构造有限自动机做词法分析是一样的。那时，我们先判断是不是关键字，如果不是关键字，才识别为标识符。而在 Antlr 里，仅仅通过声明的顺序就解决了这个问题，省了很多事儿啊！


再说个有趣的题外话。之前国内有人提“中文编程语言”的概念，也就是语法中的关键字采用中文，比如“如果”“那么”等。他们似乎觉得这样更容易理解和掌握。我不太提倡这种想法，别的不说，用中文写关键字和变量名，需要输入更多的字符，有点儿麻烦。中国的英语教育很普及，用英语来写代码，其实就够了。

不过，你大可以试一下，让自己的词法规则支持中文关键字。比如，把“if”的规则改成同时支持英文的“if”，以及中文的“如果”：

 复制代码


```
1 If:                'if' | '如果';
```

再把测试用的脚本 hello.play 中的“if”也改成“如果”，写成：

 复制代码

```
1 如果 (age >= 17+8+20){
```

重新生成词法分析器并运行，你会发现输出中有这么一行：

 复制代码


```
1 [@5,14:15='如果',<If>,2:0]
```

这个 Token 的文本值是“如果”，但类别仍然是“if”。所以，要想实现所谓的“中文编程语言”，把 C、Java 等语言的词法规则改一改，再把编译器重新编译一下就行了！

用 Antlr 生成语法分析器


说回我们的话题。现在，你已经知道如何用 Antlr 做一个词法分析器，还知道可以借鉴成熟的规则文件，让自己的词法规则文件变得更完善、更专业。接下来，试着用 Antlr 生成一个语法分析器，替代之前手写的语法分析器吧！

这一次的文件名叫做 PlayScript.g4。PlayScript 是为我们的脚本语言起的名称，文件开头是这样的：

 复制代码

```
1 grammar PlayScript;
2 import CommonLexer;    // 导入词法定义
3
4 /* 下面的内容加到所生成的 Java 源文件的头部，如包名称，import 语句等。*/
5 @header {
6 package antlrtest;
7 }
```

然后把之前做过的语法定义放进去。Antlr 内部有自动处理左递归的机制，你可以放心大胆地把语法规则写成下面的样子：


 复制代码

```
1 expression
2     :   assignmentExpression
3     |   expression ',' assignmentExpression
4     ;
5
6 assignmentExpression
7     :   additiveExpression
8     |   Identifier assignmentOperator additiveExpression
9     ;
10
11 assignmentOperator
12     :   '='
13     |   '*='
14     |   '/='
15     |   '%='
16     |   '+='
17     |   '-='
18     ;
19
20 additiveExpression
21     :   multiplicativeExpression
22     |   additiveExpression '+' multiplicativeExpression
23     |   additiveExpression '-' multiplicativeExpression
```

```
24      ;
25
26 multiplicativeExpression
27     :   primaryExpression
28     |   multiplicativeExpression '*' primaryExpression
29     |   multiplicativeExpression '/' primaryExpression
30     |   multiplicativeExpression '%' primaryExpression
31     ;
```

你可能会问：“既然用 Antlr 可以不管左递归问题，那之前为什么要费力气解决它呢？”那是因为当你遇到某些问题却没有现成工具时，还是要用纯手工的方法去解决问题。而且，有的工具可能没有这么智能，你需要写出符合这个工具的规则文件，比如说不能有左递归的语法规则。**还是那句话：懂得基础原理，会让你站得更高。**

我们继续运行下面的命令，生成语法分析器：

 复制代码

```
1 antlr PlayScript.g4
2 javac antlrtest/*.java
```

然后测试一下生成的语法分析器：

 复制代码

```
1 grun antlrtest.PlayScript expression -gui
```

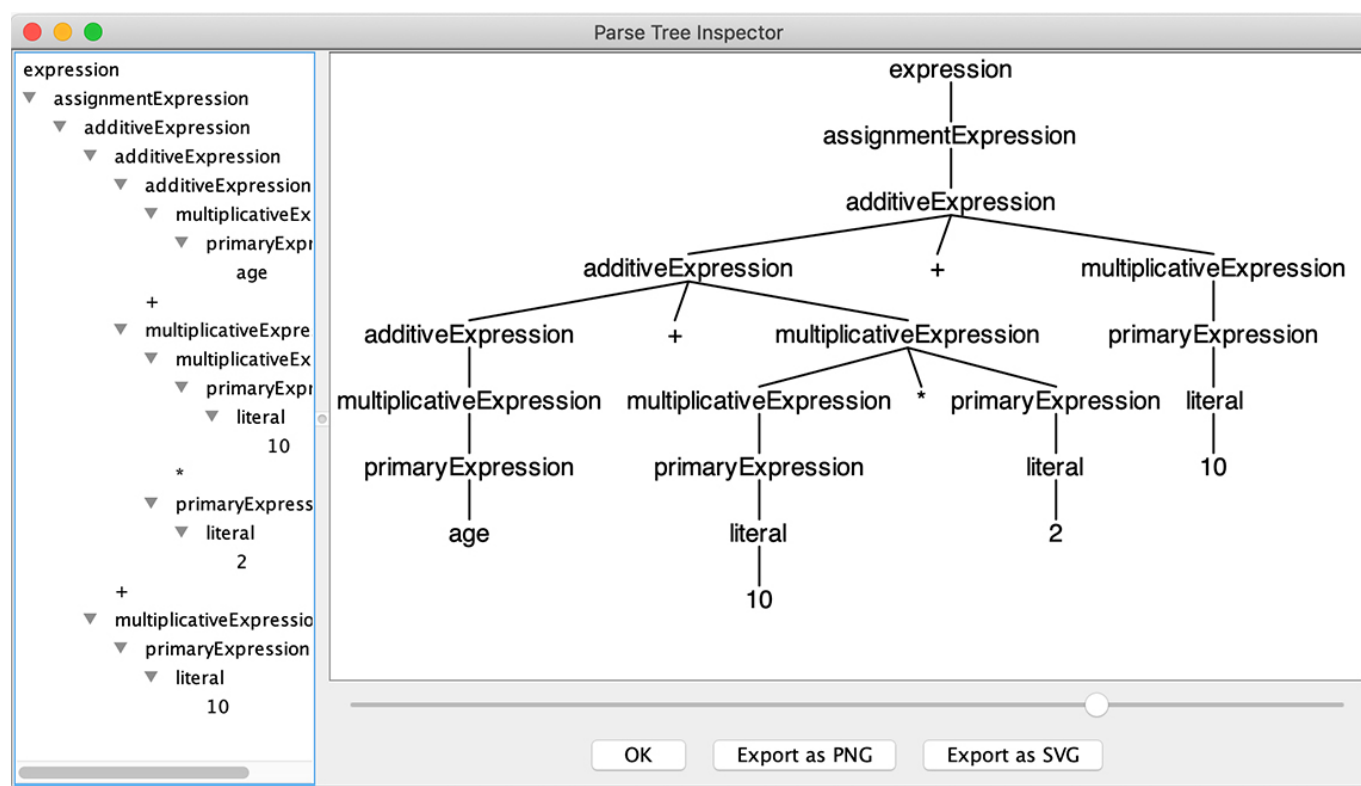
这个命令的意思是：测试 PlayScript 这个类的 expression 方法，也就是解析表达式的方法，结果用图形化界面显示。

我们在控制台界面中输入下面的内容：

 复制代码

```
1 age + 10 * 2 + 10
2 ^D
```

其中 ^D 是按下 Ctl 键的同时按下 D，相当于在终端输入一个 EOF 字符，即文件结束符号（Windows 操作系统要使用 ^Z）。当然，你也可以提前把这些语句放到文件中，把文件名作为命令参数。之后，语法分析器会分析这些语法，并弹出一个窗口来显示 AST：



看得出来，AST 完全正确，优先级和结合性也都没错。所以，Antlr 生成的语法分析器还是很靠谱的。以后，你专注写语法规则就行了，可以把精力放在语言的设计和应用上。

课程小结

今天，我带你了解了 Antlr，并用 Antlr 生成了词法分析器和语法分析器。有了工具的支持，你可以把主要的精力放在编写词法和语法规则上，提升了工作效率。

除此之外，我带你借鉴了成熟的词法规则和语法规则。你可以将这些规则用到自己的语言设计中。采用工具和借鉴成熟规则十分重要，站在别人的肩膀上能让自己更快成长。

在后面的课程中，我会带你快速实现报表工具、SQL 解析器这种需要编译功能的应用。那时，你就更能体会到，用编译技术实现一个功能的过程，是非常高效的！与此同时，我也会带你扩展更多的语法规则，并生成一个更强大的脚本语言解释器。这样，你就会实现流程控制语句，接着探索函数、闭包、面向对象功能的实现机制。几节课之后，你的手里就真的有一门不错的脚本语言了！

一课一思

今天我们介绍了 Antlr 这个工具，你有没有使用类似工具的经验？在使用过程中又有什么心得或问题呢？欢迎在留言区分享你的心得或问题。

最后，感谢你的阅读，如果这篇文章让你有所收获，也欢迎你将它分享给更多的朋友。

本讲的示例代码位于 lab/antlrtest，代码链接我放在了文末，供你参考。

Hello.g4（用 Antlr 重写了前几讲的词法规则）：[码云](#) [GitHub](#)

CommonLexer.g4（比较成熟的词法文件）：[码云](#) [GitHub](#)

PlayScript.g4（用 Antlr 重写了前几讲的语法规则）：[码云](#) [GitHub](#)

ASTEvaluator.java（对 AST 遍历，实现整数的算术运算）：[码云](#) [GitHub](#)

PlayScript.java（一个测试程序，实现词法分析、语法分析、公式计算）：[码云](#) [GitHub](#)



编译原理之美

手把手教你实现一个编译器

宫文学

北京物演科技CEO



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 05 | 语法分析（三）：实现一门简单的脚本语言

精选留言 (5)

写留言



PythonAI

2019-08-26

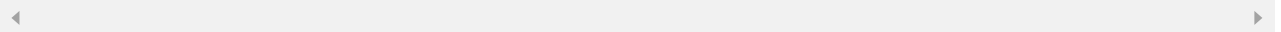
→ antlr grun antlrtest.PlayScriptexpression -gui

Can't load antlrtest.PlayScriptexpression as lexer or parser

作者回复: expression前要有空格, 前面没有antlr。

grun antlrtest.PlayScript expression -gui

我请编辑把那个空格加上。



2

1



kaixiao7

2019-08-26

在Windows下需要用 ^z 即Ctrl+z 来弹出AST窗口

That ^D means EOF on unix; it's ^Z in Windows.

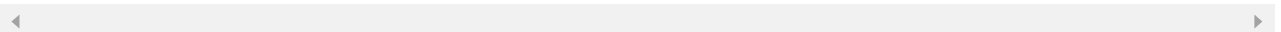
展开 ∨

作者回复: windows下用^Z? 我都没注意到这点。

好的, 一个有用的知识, 应该加到文稿中去。

谢谢你的提醒!

20年前开始学unix命令的时候, 就一直用^D, 完全没注意到在windows下的用法:-D



Giacomo

2019-08-26

还有就是当我输入

\$ javac Hello.java

\$ grun Hello tokens -tokens hello.play

后, java会报错为

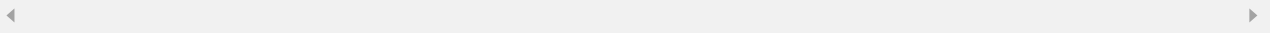
Exception in thread "main" java.lang.NoClassDefFoundError: Hello (wrong name: ...

展开 ∨

作者回复: 是CLASSPATH的问题。

设置CLASSPATH包含当前目录, 也就是.

到Hello.java的目录下去运行。



2



Giacomo

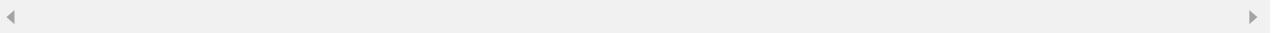
2019-08-26

老师，我的电脑是win10系统，但我安装了Ubuntu子系统。我想知道直接用
`sudo apt remove pccts` 和 `sudo apt install grun` 安装的是我所需要的命令吗？

作者回复: antlr的安装，请参考官方文档：<https://github.com/antlr/antlr4/blob/master/doc/getting-started.md>

基本上就是：

- 1.安装java环境
- 2.下载jar包；
- 3.设置CLASSPATH
- 4.(可选)设置两个shell的别名，这样打命令少敲点字。



2



安排

2019-08-26



展开 ∨

2

