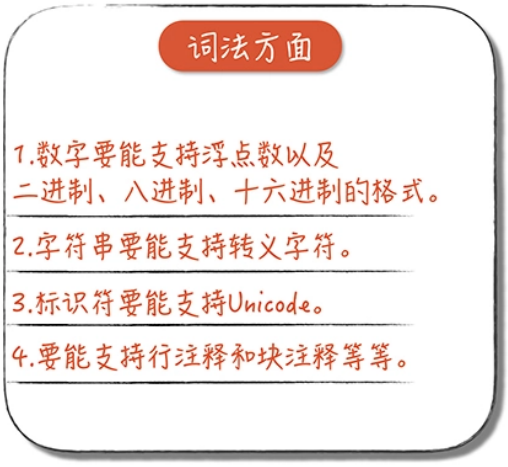
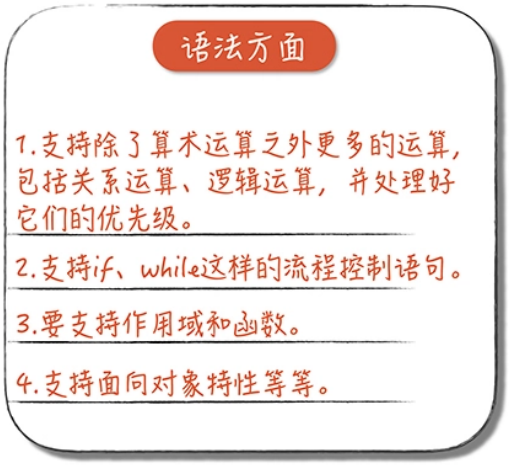
06 | 编译器前端工具（一）：用Antlr生成词法、语法分析器

2019-08-26 宫文学 来自北京

《编译原理之美》

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。





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词法规则，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

```
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

//关键字
If :                'if';
Int :               'int';

//字面量
IntLiteral:         [0-9]+;
StringLiteral:       '"' .*? '"' ;  //字符串字面量

//操作符
AssignmentOP:        '=' ;
RelationalOP:        '>'|'>='|'<' |'<=' ;
Star:               '*';
Plus:               '+';
Sharp:              '#';
SemiColon:          ';';
Dot:                '.';
Comm:               ',';
LeftBracket :       '[';
RightBracket:       ']';
```

```
LeftBrace:          '{';
RightBrace:         '}';
LeftParen:          '(';
RightParen:         ')';

//标识符
Id :                [a-zA-Z_] ([a-zA-Z_] | [0-9])*;

//空白字符，抛弃
Whitespace:         [ \t]+ -> skip;
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

```
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

```
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

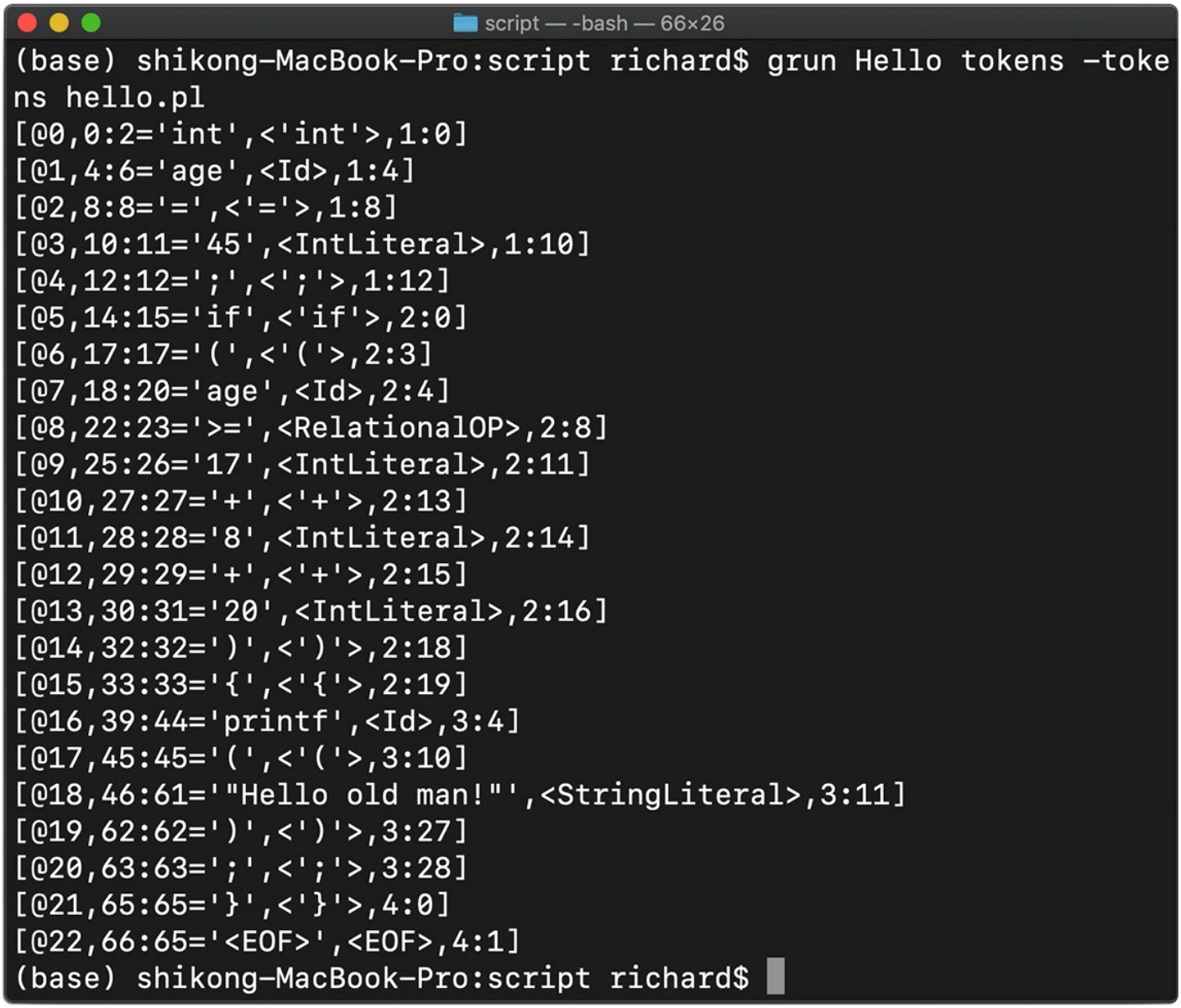
```
int age = 45;
if (age >= 17+8+20){
  printf("Hello old man!");
}
```

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：

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

复制代码

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：



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

```
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

```
STRING_LITERAL:     '"' (~["\\\r\n] | EscapeSequence)* '"';

fragment EscapeSequence
    : '\\' [btnfr"'\\]
    | '\\' ([0-3]? [0-7])? [0-7]
    | '\\' 'u'+ HexDigit HexDigit HexDigit HexDigit
    ;

fragment HexDigit
    : [0-9a-fA-F]
    ;
```

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 >=、>、< 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”：

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

复制代码

再把测试用的脚本 hello.play 中的“if”也改成“如果”，写成：

```
1 如果 (age >= 17+8+20){
```

复制代码

重新生成词法分析器并运行，你会发现输出中有这么一行：

```
1 [@5,14:15='如果',<If>,2:0]
```

复制代码

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

```
grammar PlayScript;
import CommonLexer;   //导入词法定义

/*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。*/
@header {
package antlrtest;
}
```

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

```
expression
    :   assignmentExpression
    |   expression ',' assignmentExpression
    ;

assignmentExpression
    :   additiveExpression
    |   Identifier assignmentOperator additiveExpression
    ;

assignmentOperator
    :   '='
    |   '*='
    |  '/='
    |   '%='
    |   '+='
    |   '-='
    ;

additiveExpression
    :   multiplicativeExpression
```

```
    |   additiveExpression '+' multiplicativeExpression
    |   additiveExpression '-' multiplicativeExpression
    ;

multiplicativeExpression
    :   primaryExpression
    |   multiplicativeExpression '*' primaryExpression
    |   multiplicativeExpression '/' primaryExpression
    |   multiplicativeExpression '%' primaryExpression
    ;
```

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

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

然后测试一下生成的语法分析器：

复制代码

```
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。

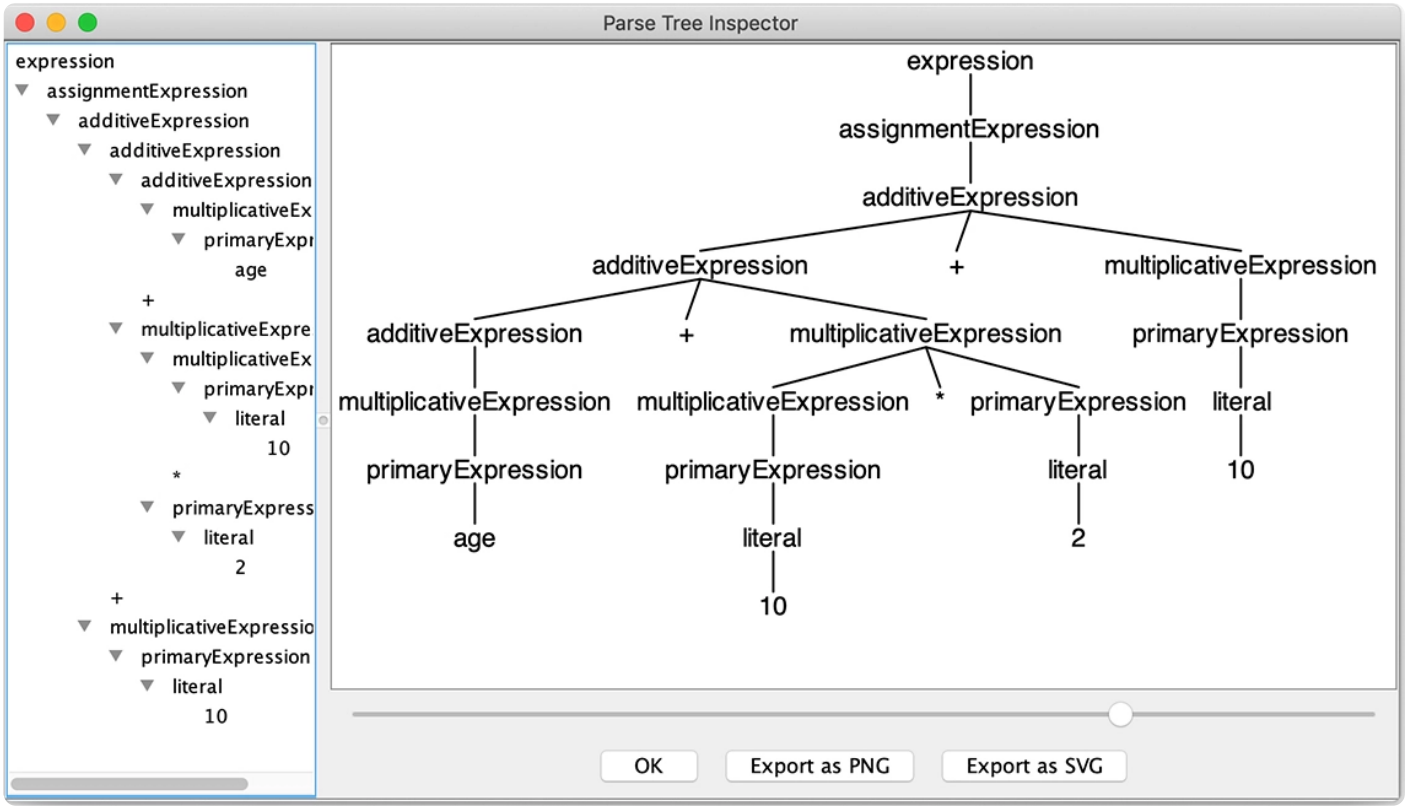
我们在控制台界面中输入下面的内容：

复制代码

```
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

精选留言 (59)

京京beaver
2019-09-10

1. windows环境下配置
可执行文件，放在D:\tools\antlr\antlr-4.7.2-complete.jar下面

2.配置环境文件
CLASSPATH=%CLASSPATH%;D:\tools\antlr\antlr-4.7.2-complete.jar
PATH=%PATH%n;D:\tools\antlr

3.手写文件antlr4.bat和grun.bat
antlr4.bat
java org.antlr.v4.Tool %*

grun.bat
@ECHO OFF
SET TEST_CURRENT_DIR=%CLASSPATH:.;=%
if "%TEST_CURRENT_DIR%" == "%CLASSPATH%" (SET CLASSPATH=.;%CLASSPATH%)
@ECHO ON
java org.antlr.v4.gui.TestRig %*

4.然后就可以执行antlr4和grun命令了
比如antlr4 Hello.g4，javac Hello*.java,
$ grun Hello r -gui
hello parrt
^Z (windows)

5.java执行路径要注意事项
在命令行执行java命令，记得把目录设置到src/main/java这里，然后输入包名.类名，才能找到。
这是java执行的基本规则，一般IDE里面替你做了路径转换，在命令行要自己敲入。切记。

例如
F:\study_repo\mygeek_time\myclang-03-grammar-analysis\src\main\java>
grun com.babayetu.myclang03gramm
aranalysis.antlrtest.PlayScript expression -gui

作者回复: 非常感谢你的整理!
其他同学可以参考你的配置方法!

Smallfly
2019-09-05

这一讲走的有点艰难，记录一下：

1、
开始执行下面的命令，报找不到 CommonLexer。

antlr PlayScript.g4

看了下 Github 才发现有这个文件。

import CommonLexer, 在语法规则文件 PlayScript.g4 中导入词法规则。

2、

antlr PlayScript.g4

上面的命令后应该先 cd .. 回退一级目录，再执行：

javac antlrtest/*.java

或者，直接:

javac ./*java

3、

grun antlrtest.PlayScript expression -gui

文中说执行这条命令结果会以图形化界面显示，我执行之后什么都没有输出，以为前面步骤有什么问题，重新来了一次，还是这样，往下看才意识到没输出才是正常的。。。

——

看到最后弹出的 AST 树还是蛮有意思的。

我写了一版简单的 Swfit 规则文件：

https://github.com/iostalks/PlayWithCompiler/tree/lecture-6/PlayWithCompiler/Antlr

作者回复: 哇，练习自己写规则，太帮了！
自己动手所获得的感觉是难以替代的。
中间过程遇到的每个坑，都是自己的积累！

另外，这里也有几个swift的规则文件可以参考：
https://github.com/antlr/grammars-v4

6

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.

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

4

七月有风
2019-12-23

macOS下，需要将把 Antlr 的 JAR 文件设置到 CLASSPATH 环境变量中：
如果是用Homebrew 安装的 Antlr，安装路径是：/usr/local/Cellar/antlr/4.7.2/antlr-4.7.2-complete.jar；
可以使用vi ~/.bash_profile命令打开bash_profile文件，将export CLASSPATH=".:/usr/local/Cellar/antlr/4.7.2/antlr-4.7.2-complete.jar:$CLASSPATH"这段代码复制到里面。
然后就可以运行javac *.java了

作者回复: 感谢分享！
我这两天整理了Antlr使用的要点，可以参见这里：
https://github.com/RichardGong/PlayWithCompiler/blob/master/antlr_install.md

3

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。

共 3 条评论 >

3

江世民
2020-06-28

遇到了一个大坑。
Windows环境下，添加jar包到CLASSPATH中时，最好写在前面。如果是追加在后面，系统很可能不识别。

作者回复: 感谢分享你的经验!
你遇到的坑，会让其他同学少走弯路!

2

七月有风
2019-12-25

不知道是什么问题?
$ grun Hello tokens -tokens Hello.play
Exception in thread "main" java.lang.NoClassDefFoundError: antlrtest/Hello (wrong name: Hello)
at java.base/java.lang.ClassLoader.defineClass1(Native Method)
at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1016)
at java.base/java.security.SecureClassLoader.defineClass(SecureClassLoader.java:174)
at java.base/jdk.internal.loader.BuiltinClassLoader.defineClass(BuiltinClassLoader.java:802)
at java.base/jdk.internal.loader.BuiltinClassLoader.findClassOnClassPathOrNull(BuiltinClassLoader.java:700)
at java.base/jdk.internal.loader.BuiltinClassLoader.loadClassOrNull(BuiltinClassLoader.java:623)
at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:58

1)

at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:178)

at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:521)

at org.antlr.v4.gui.TestRig.process(TestRig.java:135)

at org.antlr.v4.gui.TestRig.main(TestRig.java:119)

作者回复: 可能是CLASSPATH设置的问题，或者是运行grun命令的目录不对。参见我整理的一篇文字，梳理了antlr使用的要点：
https://github.com/RichardGong/PlayWithCompiler/blob/master/antlr_install.md

共 2 条评论 >

2

mudfrog
2019-09-01

老师，我的程序能编译通过，也能正常运行，能正常的解析和运算出来，但就是想看看语法树直观一些。我使用grun的时候总是提示Can't load CalExpr as lexer or parser，这里CalExpr到底是G4文件还是tokens文件呢，我把这两个文件都拷贝到src目录下了。我用的是win7底下的eclipse，

作者回复: 1.grun是用的Class文件，是java类。
2.但运行grun的时候，最好就在CLASSPATH的目录下。
假设，class和g4文件位于：
playscript-java/src/main/play

其中play是包名，而CLASSPATH设置的是：
playscript-java/src/main
那么你就在main这个目录下运行grun。

如果都不带包，就更简单一些，让grun就在带有class和.g4的目录中运行就好。

如果还有问题的话，请继续再问！

共 2 条评论 >

2

式溪Chon W. Lam
2022-05-06

我开发了一个vscode插件, 把antlr的功能封装进去, 安装java就可以使用: https://marketplace.

visualstudio.com/items?itemName=ChonLam.justAntlr

(暂时只有windows可以使用antlr的测试功能)

1

Geek_6304e3
2022-02-10

'grun antlrtest.PlayScript expression -gui' 执行的时候提示 'Can't load antlrtest.PlayScript as lexer or parser'，前面步骤都正常

1

minghu6
2021-03-18

ANTLR的使用一定要有 “The Definitive ANTLR 4 Reference” 推荐电子版 https://github.com/antlr/antlr4/blob/master/doc/index.md

可以做关键字搜索，查点儿语法概念性的东西比较方便，用一位网友说的话：
If you do not already have "The Definitive ANTLR 4 Reference" book I recommend getting hold of it. Will save you a lot of time.
话说极客时间的内容还不错，没有那么多花里胡哨的噱头，但是这个书签笔记和评论的体验太差！
书签笔记甚至没有结构，留言也不支持markdown

一定要ommonLexer.g4

> 作者回复: 谢谢推荐Antlr4电子书！
> 以及提出的建议！

1

englefly
2020-06-16

宫老师，Antlr的性能怎样？我们用antlr做了一个sql解析，当遇到比较长的sql语句是，antlr解析花了100ms，而mysql 只用了9ms。不知道是我们用法的问题，还是antlr本身为了使用简单牺牲了一些性能。

作者回复: 我要了解更多一点信息：你用Antlr生成的是C++代码还是Java代码？这两个不太一样。因为JVM只对热点代码做优化编译。如果只是解释执行或者用C1编译器编译，就会比较慢。详细你可以参考《编译原理实战课》中Java JIT编译器的部分。

另外，在实战课中，我也介绍了MySQL的编译器。MySQL的语法分析器也是用工具生成的，用的是Bison，生成的是C++代码。我估计，同样都使用C++代码的情况下，性能可能不会差很多。但具体也要测试一下。
或者，你就像MySQL一样，用Bison生成就好了，并且可以参考MySQL的源代码。

另外补充一下，MySQL Workbench这个客户端工具采用了Antlr来生成语法解析器，也是基于C++的，你也可以参考。

1

漏网之渔
2020-03-24

弄了三个小时，终于好了。老师文档写的很清楚。
https://github.com/RichardGong/PlayWithCompiler/blob/master/antlr_install.md
我遇到的问题有两点：
1.执行 grun 的时候 没带package名如grun Hello tokens -tokens hello.play，会提示java.lang.NoClassDefFoundError: Hello (wrong name: antlrtest/Hello)，这是因为老师源码Hello带了pack保命antlrtest

我的解决方法是在src\antlrtest\目录下执行命令grun antlrtest.Hello tokens -tokens hello.play

2.执行grun命令报错误java.nio.file.NoSuchFileException: hello.play，这是因为我在antlrtest\src\目录下运行的grun，而hello.play在antlrtest\src\antlrtest\下，解决方法是在antlrtest\src\antlrtest\下执行grun命令
，或者把hello.play文件代码拷贝到src\目录下再执行grun命令。

另外如果遇到javac命令遇见GBK错误，是因为编码格式问题，解决方法：javac -encoding utf-8 PlayScript.java

如果遇到找不到parse lexer问题，Class Path需要添加src目录

作者回复: 感谢详细的分享！
你遇到的某些问题，如GBK问题，我都没遇到过。其他人如果遇到类似的问题，可以从你的分享中受

益!

共 2 条评论 > 1

火火
2019-11-28

Hello.java:2: 错误: 程序包org.antlr.v4.runtime不存在
import org.antlr.v4.runtime.Lexer;
^
Hello.java:3: 错误: 程序包org.antlr.v4.runtime不存在
import org.antlr.v4.runtime.CharStream;
^
Hello.java:4: 错误: 程序包org.antlr.v4.runtime不存在
import org.antlr.v4.runtime.Token;
^
Hello.java:5: 错误: 程序包org.antlr.v4.runtime不存在
import org.antlr.v4.runtime.TokenStream;
^
Hello.java:8: 错误: 程序包org.antlr.v4.runtime.dfa不存在
import org.antlr.v4.runtime.dfa.DFA;
^
Hello.java:12: 错误: 找不到符号

作者回复: 这个问题的原因是antlr的jar包没有放到CLASSPATH中。
我这两天整理了antlr安装、配置、使用的一些信息，写了一个说明文件，你可以参考一下：
https://github.com/RichardGong/PlayWithCompiler/blob/master/antlr_install.md

共 3 条评论 > 1

沉淀的梦想
2019-08-29

老师，为什么我用antlr生成的AdditiveExpressionContext就没有示例程序中的ADD() SUB()这些直接以token名字命名的方法呢?

作者回复: 再检查一下规则文件。
你看看你的词法规则里是否给加号和减号起了ADD和SUB这样的名字，就像下面这样。
ADD : '+';

SUB : '-';

只要有这样的定义，在语法里无论用ADD还是用'+'，都会生成ADD()这样的方法。

共 2 条评论 >

1

雲至

2019-08-26

老师可以讲一下规则文件里主要单词的意义吗？ 看的有点懵

1

温雅小公子

2022-11-09 来自河北

为什么我没有生成 Hello.class 文件，只有 Hello Lexer.class 文件呢

Geek_216bc5

2022-05-25

我为什么没有生成 PlayScriptBaseVisitor.java文件。

A君

2022-05-07

如果源文件有多个且不在同一目录，是要对每个文件都单独做词法语法分析吗，那多个AST树要如何合并?

ggggggb

2022-04-11

D:\antlr 的目录

```
2022/04/11  22:45    <DIR>          .
2022/04/11  22:45    <DIR>          ..
2021/08/07  12:16         2,100,564 antlr-4.9.2-complete.jar
2022/04/11  21:45                63 antlr4.bat
2022/04/11  21:45                71 grun.bat
2022/04/11  22:45             5,663 Hello.class
2022/04/11  21:34               872 Hello.g4
```

```
2022/04/11  22:45             4,525 Hello.interp
2022/04/11  22:45             6,322 Hello.java
2022/04/11  22:38                 0 hello.play
2022/04/11  22:45               374 Hello.tokens
               9 个文件      2,118,454 字节
               2 个目录 262,626,590,720 可用字节

D:\antlr>grun Hello tokens -tokens hello.play

D:\antlr>java  -cp D:\antlr\antlr-4.9.2-complete.jar org.antlr.v4.gui.TestRig Hello tokens -to
kens hello.play
Can't load Hello as lexer or parser
```

这个是什么情况?