

ANTLR4入门【打造你自己的语法规则】

文章目录

- 什么是ANTLR4
 - ANTLR4的特点
 - ANTLR4的语法规则
 - 语法规则的声明
 - .g4文件的代码规则
 - 词法规则
 - 语法规则
 - 语法规则中的操作
- 如何使用ANTLR4
- 实战
 - IDEA安装ANTLR插件
 - 新建项目
 - 创建一个.g4文件
 - 通过插件生成java代码
 - 创建一个EvalVisitor遍历AST
 - 测试

什么是ANTLR4

ANTLR4（全称为"ANother Tool for Language Recognition"）是一种用于构建语言识别器的强大工具。它是一个自动生成的解析器生成器，使用ANTLR4助开发人员快速创建自定义的语言或DSL。

ANTLR4的特点

- 支持多种目标语言，包括Java、C#、Python、JavaScript等。
- 支持LL(*)（LL star）语法分析器，可以处理包含任意数目的向前看标记（lookahead）的语法。
- 支持词法和语法错误处理，可以生成清晰的错误消息和恢复策略。
- 支持生成AST（抽象语法树），方便语言处理器进行语法分析和代码生成。
- 提供了丰富的API和工具集，可以自定义和优化ANTLR4生成的解析器。

ANTLR4的语法规则

语法规则的声明

ANTLR4的语法规则必须以“grammar”关键字开始，后面跟着语法规则的名称。例如：

```
1 | grammar MyGrammar;
```

这里声明了一个名为“MyGrammar”的语法规则。

.g4文件的代码规则

词法规则

ANTLR4语法规则以词法规则（lexer rules）开始。词法规则定义了输入文本中的各个单词（或标记），这些单词将被语法规则使用。每个词法规则定义的名称和匹配该单词的正则表达式。

以下是一个简单的例子：

```
1 | // 定义变量名为一个字母后跟零个或多个字母、数字或下划线
2 | ID : [a-zA-Z][a-zA-Z0-9_]*;
```

在这个例子中，我们定义了一个名为ID的词法规则，该规则匹配一个以字母开头，后跟零个或多个字母、数字或下划线的字符串。在语法规则中，我们用ID来匹配变量名。

语法规则

ANTLR4语法规则定义了输入文本的结构。语法规则由词法规则和其他语法规则组成，定义了输入文本的结构。语法规则通常以一个称为start的语法规则，这个语法规则定义了整个输入文本的结构。

以下是一个简单的例子：

```
1 // 定义一个简单的算术表达式语法规则
2 expr : INT
3       | expr op=('*' | '/') expr
4       | expr op=('+' | '-') expr
5       | '(' expr ')'
6       ;
7
8 INT : [0-9]+;
```

在这个例子中，我们定义了一个名为 `expr` 的语法规则，该规则定义了一个简单的算术表达式。该规则由四个子规则组成，每个子规则定义了一种表示子规则匹配整数，第二个和第三个子规则匹配乘法、除法、加法和减法，最后一个子规则匹配括号中的表达式。我们还定义了一个名为INT的词法规则匹配一个或多个数字。在这个例子中，我们使用INT来匹配整数。

语法规则中的操作

```
1 // 定义一个带有语义谓词的语法规则
2 expr : INT { $INT.text.equals("0") }
```

在这个例子中，我们定义了一个名为expr的语法规则，该规则匹配整数，并使用语义谓词来指定该规则是否适用于输入文本。在这个例子中，语义谓词是否为零。

如何使用ANTLR4

1. 下载并安装ANTLR4：只需要下载官方网站上提供的ANTLR4 JAR文件，然后将其添加到您的Java类路径中即可。您还可以选择安装ANTLR4插件到您的IDE中，以便更方便地使用ANTLR4。下面我们演示如何在IDEA中使用ANTLR4插件。
2. 编写语法文件：在ANTLR4中，您需要编写一个.g4文件来定义词法和语法规则。您可以使用文本编辑器编写.g4文件。
3. 生成代码：一旦您编写了.g4文件，您可以使用ANTLR4生成Java代码。使用以下命令可以生成Java代码：

```
1 java -jar antlr-4.9.2-complete.jar YourGrammar.g4
```

在上面的命令中，您需要将YourGrammar.g4替换为您的.g4文件的名称。

4. 编写解析器：一旦您生成了Java代码，您可以编写一个Java解析器来解析输入文本。您可以使用ANTLR4提供的Java API来编写解析器。

```
1 CharStream input = CharStreams.fromStream(System.in);
2 YourGrammarLexer lexer = new YourGrammarLexer(input);
3 CommonTokenStream tokens = new CommonTokenStream(lexer);
4 YourGrammarParser parser = new YourGrammarParser(tokens);
5 YourGrammarParser.StartContext tree = parser.start();
```

在上面的代码中，您需要将YourGrammarLexer和YourGrammarParser替换为您生成的Lexer和Parser类的名称。您还需要替换start()方法，该方法应返回语法规则中的start规则的ParseTree对象。

实战

我们来做一个具体的例子，经典的计算器，主要是实现 **四则运算**，带括号等等。

IDEA安装ANTLR插件

Settings

Search: plu

Appearance & Behavior

- System Settings
 - Data Sharing
 - Notifications
- Keymap
- Editor
 - Color Scheme
 - Angular Template
 - Inspections
 - File and Code Templates
 - Live Templates
- Plugins** (2)
- Build, Execution, Deployment
 - Build Tools
 - Maven
 - Importing
 - Compiler
 - Scala Compiler
 - Bytecode Indices
 - Required Plugins
 - Languages & Frameworks
 - Lombok plugin
 - Scala
 - Tools

Plugins

Search: antlr

Search Results (3) Sort By: Relevance

ANTLR v4 (Installed) 471.9K 4.6 ANTLR Project

ANTLRWorks (Install) 14.7K 3.4 ANTLR Project

Verilog support (Install) 6.5K 3.8 MrTsepa

ANTLR v4

471.9K 4.6 ANTLR Project

Tools Integration 1.19.3 11月2

Plugin homepage

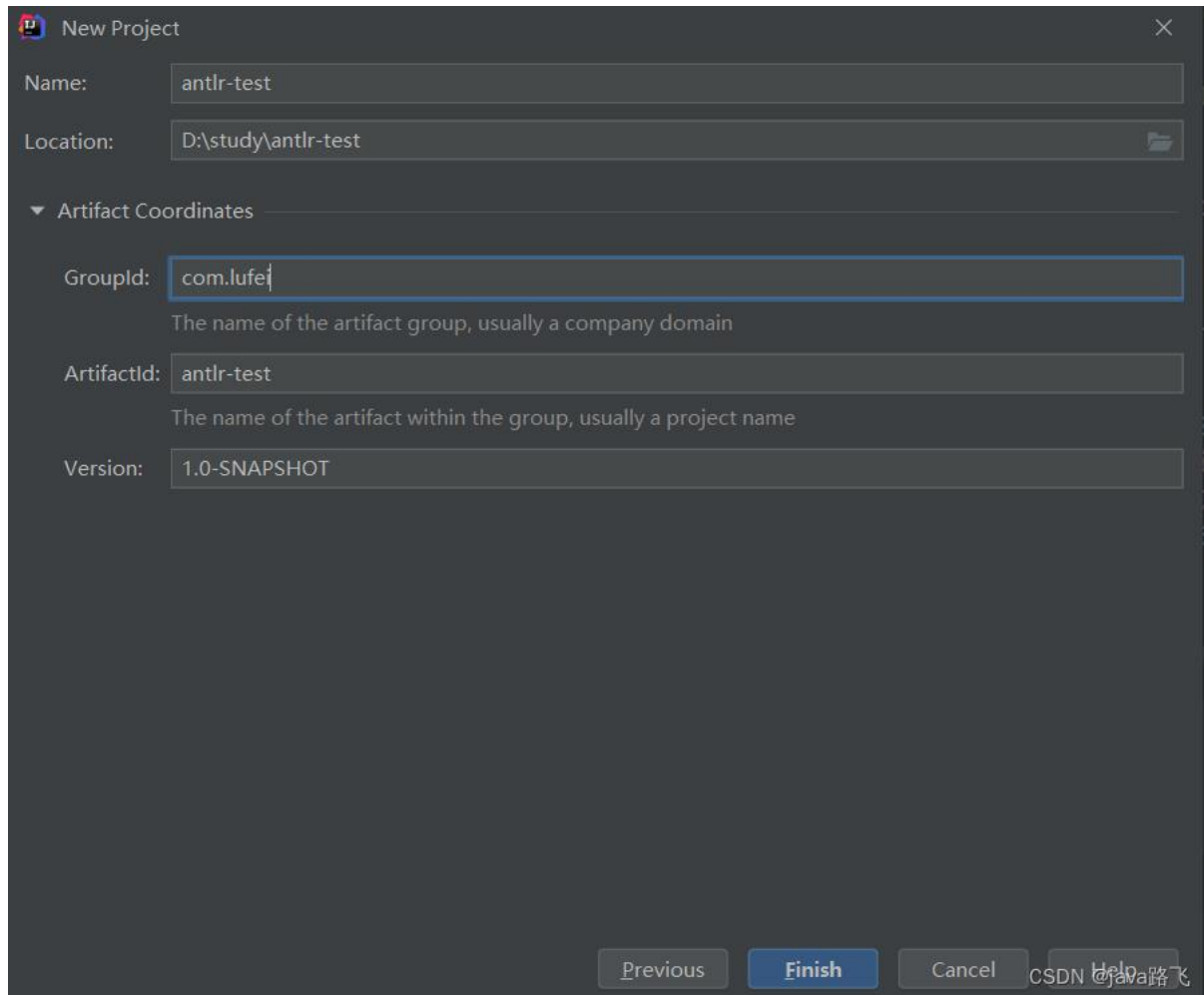
This plugin is for ANTLR v4 grammars and includes ANTLR 4.11.1. It works with 2020.2.1 and above. It should work with other JetBrains IDEs.

- syntax highlighting
- syntax error checking
- semantic error checking
- navigation window
- goto-declaration
- find usages
- rename tokens
- rename rules
- comment grammar rule lines with meta-/ (1.7)
- save parse trees as svg/jpg/png; right click in parse tree view (1.9)
- grammar/comment folding (1.7)
- generates parser code; shortcut (ctrl-shift-G / meta-shift-G) but it's in Tools menu and popups.
- code completion for tokens, rule names;
- finds tokenVocab option for code gen if there is tokenVocab option, don't warn about implicit tokenVocab
- handles separate parsers and lexers like TParser.g and TLexer.g4 (1.7)
- parse tree nodes show the alternative number that the parser chose to match that node. (1.7)
- has live grammar interpreter for grammar preview

OK Cancel

新建项目

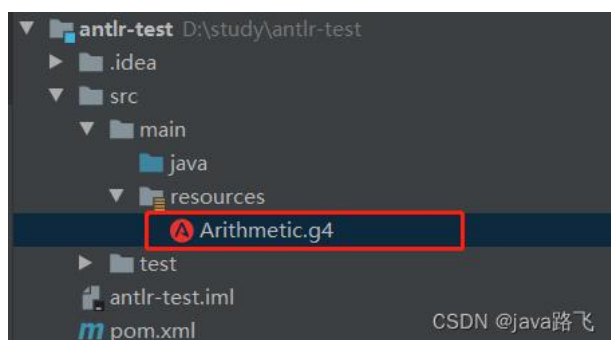
新建一个maven项目



pom.xml引入antlr的依赖

```
1      <dependency>
2          <groupId>org.antlr</groupId>
3          <artifactId>antlr4-runtime</artifactId>
4          <version>4.10.1</version>
5      </dependency>
```

创建一个.g4文件



```
1 grammar Arithmetic;
2
3 /*
4  * Parser rules
5  */
6
7 parse
8     : expr EOF
9     ;
10
11 ...
```

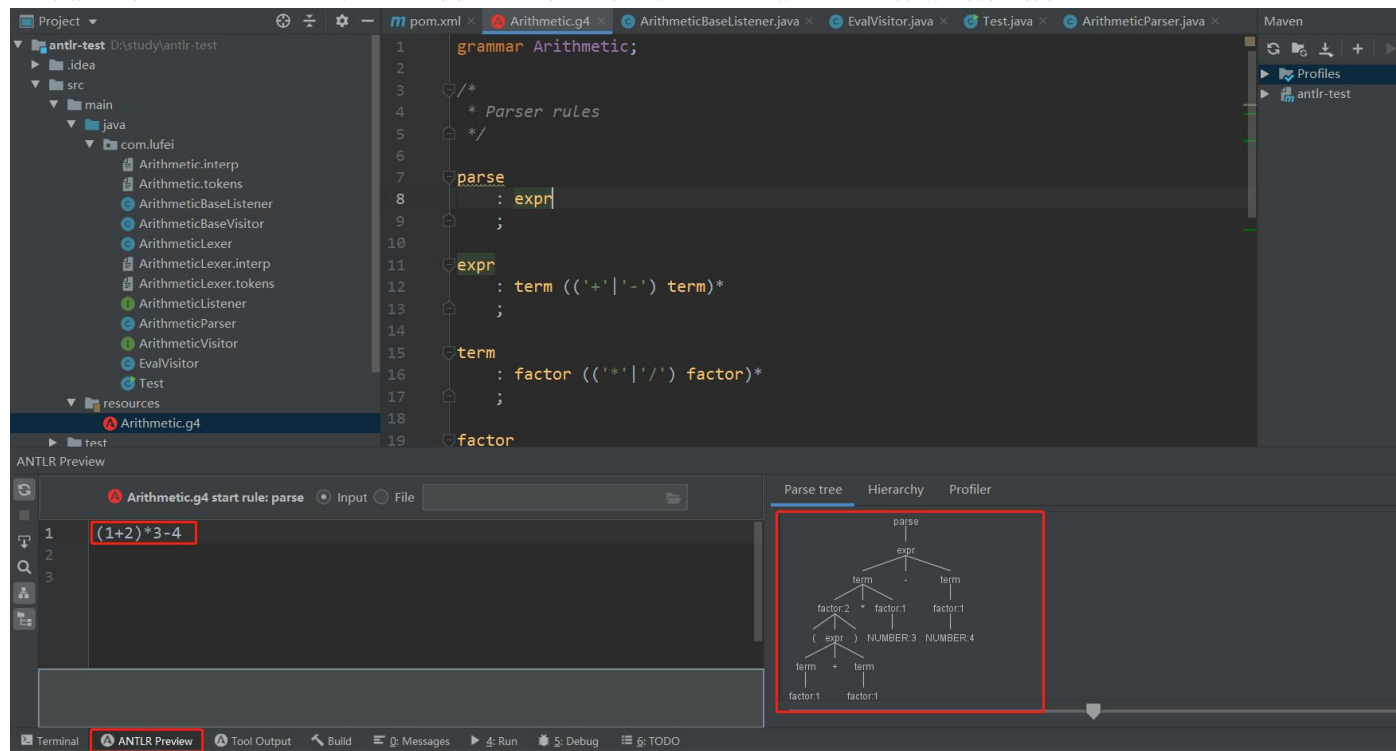
在这个语法规则中，我们定义了四个规则：

- parse：最高级别的规则，表示一个完整的表达式。它由一个expr规则后面跟着一个EOF标记组成。
- expr：表示一个加法或减法表达式。它由一个term规则后面跟着零个或多个加法或减法符号，后面再跟着一个term规则组成。
- term：表示一个乘法或除法表达式。它由一个factor规则后面跟着零个或多个乘法或除法符号，后面再跟着一个factor规则组成。
- factor：表示一个数字或括号内的表达式。它可以是一个数字，或者是一个由左括号、一个表达式和右括号组成的组合。

注意，在这个示例中，我们还定义了一些Lexer规则，用于识别数字和忽略空格和换行符。

我们可以通过idea的antlr插件来检查我们的语法规则：

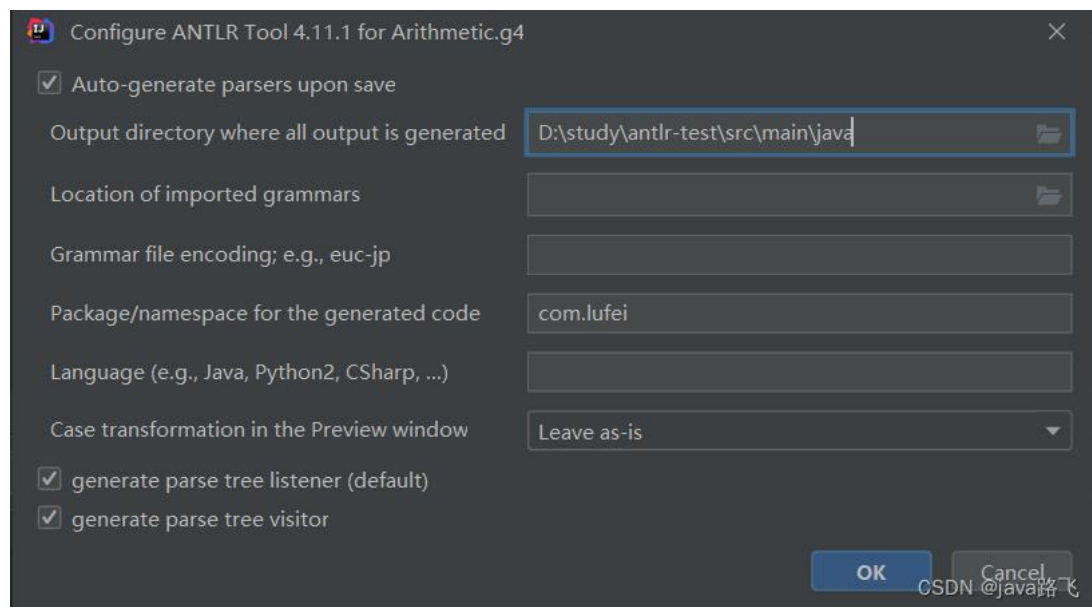
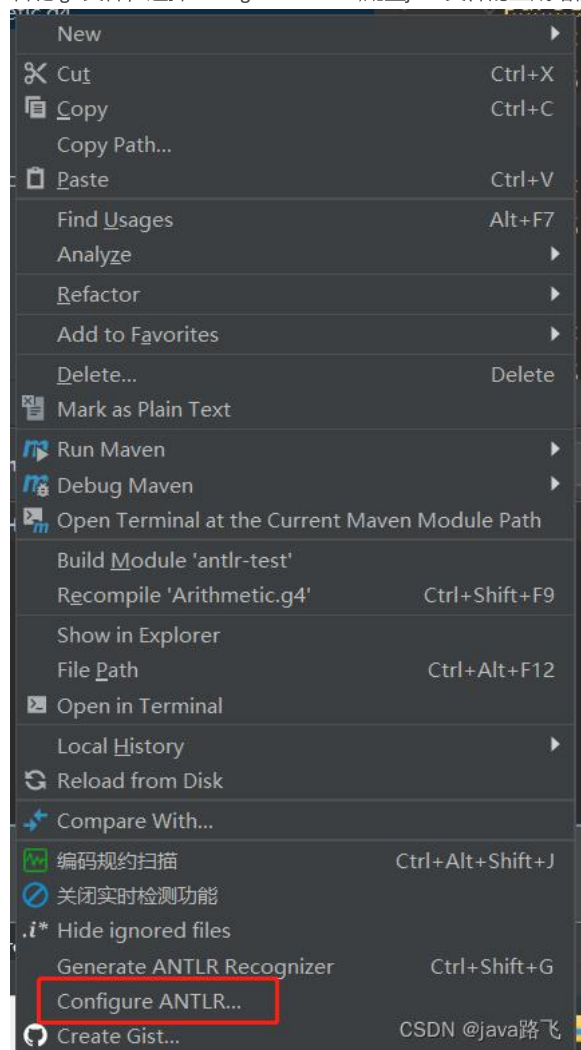
如下图，点击ANTLR Preview可以来到如下界面。左边的框填写需要校验的表达式，右边就是语法树解析的结果。



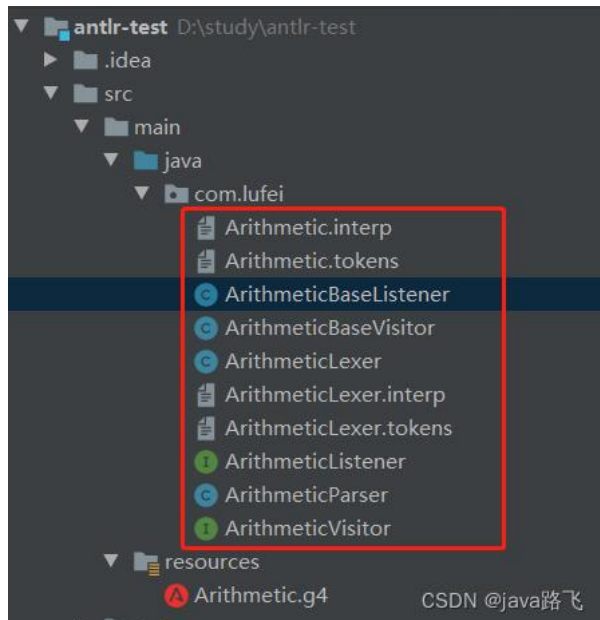
接下来，我们需要使用ANTLR来生成Java代码。

通过插件生成java代码

右键.g4文件, 选择Configure ANTLR,配置java文件的生成路径。



再次右键.g4文件，选择Generate ANTLR Reconizer 即可生成java代码



创建一个EvalVisitor遍历AST

我们创建一个EvalVisitor遍历AST，并计算表达式的值。最终，我们打印计算结果。

EvalVisitor是一个我们需要自己实现的类，它继承了ArithmeticBaseVisitor，并重写了其中的方法。下面是一个简单的实现示例：

新建类EvalVisitor.java

```
1 public class EvalVisitor extends ArithmeticBaseVisitor<Double> {
2     // 使用一个Map来存储变量名和值的映射关系
3     Map<String, Double> memory = new HashMap<String, Double>();
4
5
6     // 重写visitExpr方法，用于计算加法和减法
7     @Override
8     public Double visitExpr(ArithmeticParser.ExprContext ctx) {
9         Double result = visit(ctx.term(0));
10         for (int i = 1; i < ctx.term().size(); i++) {
11             String op = ctx.getChild(2*i - 1).getText();
```

测试

新建测试类Test.java

```
1 public class Test {
2
3     public static void main(String[] args) throws Exception {
4         String input = "(1+2)*3-4";
5         // 创建一个词法分析器，用于将输入转换为标记
6         ArithmeticLexer lexer = new ArithmeticLexer(CharStreams.fromString(input));
7
8         // 创建一个标记流，用于将标记传递给解析
9         CommonTokenStream tokens = new CommonTokenStream(lexer);
10
11         // 创建一个解析器 用于将标记转换为AST
```