Lesson 07

Actions

STRU
MEN
TA

Why Not to Use Actions

The suggested way of interacting with the results of parsing is using a listener or a visitor:

- They provide a standard and portable way to work with the parse tree.
- They leave the grammar readable and usable with different target languages.
- They allow to make a clear distinction between the parsing phase and whatever comes next. Which means that you can reuse the same grammar for different applications.

Why to Use Actions

There are mainly two reasons to use actions:

- Efficiency and performance
- To parse context-sensitive languages

Efficiency and Performance

- There is no need to build a parse tree after parsing. So, you can get the exact result you need ready to use, after the parsing ends, saving memory and time.
- You can stop parsing as soon as there is an error. This saves time.

Context-sensitive Parsing

Sometimes you are parsing languages that are not context-free, but context-sensitive. ANTLR can parse a context-sensitive languages only if you help it.

Context-sensitive Parsing: Python

In Python whitespace can perform two syntactic functions:

- when inside a statement it’s irrelevant, such as between arguments of a function or between the operands of an operation
- when used outside statements, it indicates the boundaries of blocks of code

What Are Actions?

Actions are arbitrary pieces of code written inside two curly brackets

Where to Put Action Code

- in the header position, so that they will put before the parser class. And this is where you can do things like importing modules in Python.
- in the members position, which will output them at the beginning of the parser class
- next to each rule, or sub-rule, which will make ANTLR output them in the corresponding lexer or parser rule

STRU MEN TA

Actions in Lexer Rules

- It's rarer but actions can also be used in lexer rules.
- You cannot access attributes inside lexer actions.
- There is fundamentally one reason to use actions in lexer rule: alter the token.

TEXT Token Example

```
TEXT          : '[' ~[\]]+ ']' {self.text = self.text[1:-1]};
```

This rule can be erroneously be matched in some paratemerized type.

```
// the '>>' is matched as BITSHIFT token
List<Dictionary<string, int>> x;
```

In Summary

- Actions are quite easy to use, but also powerful.
- Sometimes you cannot avoid using them, but you should not use them if you can avoid it.
- Generic actions can be very useful, but most of the time are not strictly necessary.