ANTLR: Building DSL

Domain-specific language

DSL 是一种基于特定领域的语言，它使工作更贴近于客户的理解，而不是实现本身，这样有利于开发过程中，所有参与人员使用同一种语言进行交流。

DSL 是一种为了特定任务而设计的开发语言。

Domain-specific language

SQL

Domain-specific language

SQL

```
SELECT *
    FROM Book
    WHERE price > 100.00
    ORDER BY title;
```

Domain-specific language

Makefile

Domain-specific language

Makefile

```
hello: main.o hello.o
   g++ main.o hello.o -o hello

main.o: main.cpp
   g++ -c main.cpp

hello.o: hello.cpp
   g++ -c hello.cpp
```

Domain-specific language

CSS

Domain-specific language

CSS

```
body {
   background-color:#d0e4fe;
}

h1 {
   color:orange;
   text-align:center;
}
```

Domain-specific language

External

or

Internal

Domain-specific language

Internal DSLs

Domain-specific language

```
new Mailer()
   .from("build@example.com")
   .to("example@example.com")
   .subject("build notification")
   .message("some details ")
   .send();
```

Domain-specific language

```
User.find_each(:start => 2000, :batch_size => 5000) do |user|
    NewsLetter.weekly_deliver(user)
end
```

Domain-specific language

Internal DSL

简单、容易实现，但是功能有限

Domain-specific language

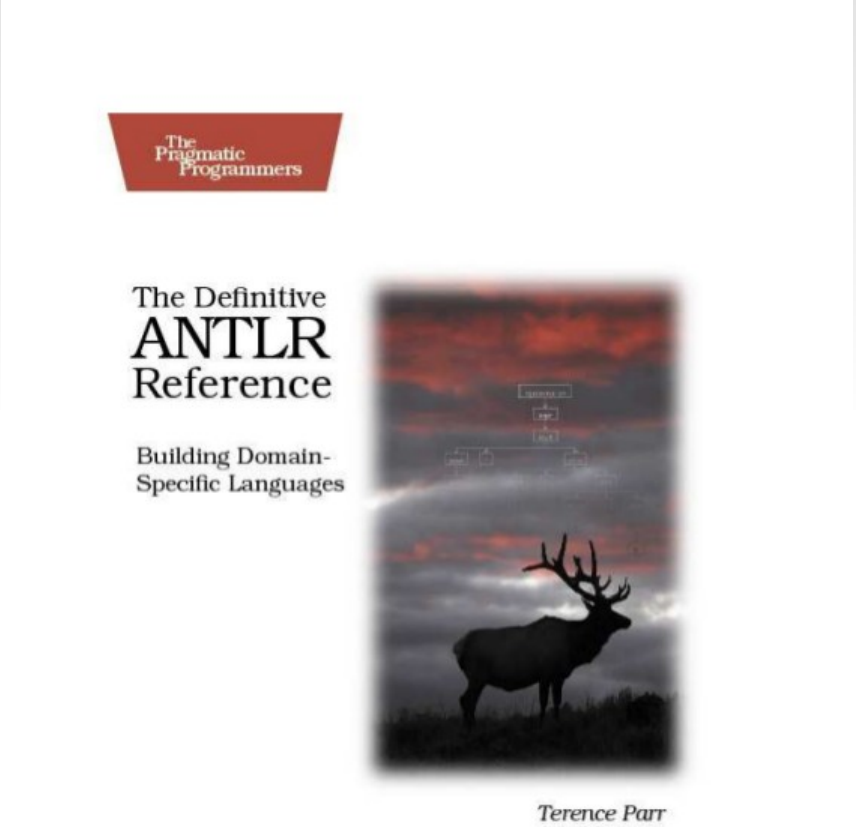
External DSL

灵活、复杂，同时功能强大

Domain-specific language

- flex/bison (PostgreSQL)
- JavaCC
- Yacc (MySQL)
- Lemon (SQLite)

Domain-specific language



Domain-specific language

ANTLR: Another Tool for Language Recognition

Domain-specific language

ANTLR 由旧金山大学 (University of San Francisco) 的教授 Terence Parr 开发并维护的，其始于 1989 年，到了现在过了 20 多年，一直都是一个很活跃的项目。

ANTLR 一般用于构建 Domain-Specific Languages (DSL)。用户编写好特定语言的语法文件后， ANTLR 会根据该语法文件生成相应的源代码来识别该语言。 ANTLR 3.4（截至 2011-10-15 最新的版本）支持的编程语言 (runtime) 包括： ActionScript ， Csharp2 ， Delphi ， JavaScript ， Perl5 ， Ruby ， C ， CSharp3 ， Java ， ObjC ， Python.

Domain-specific language

ANTLR 中有主要类有两种

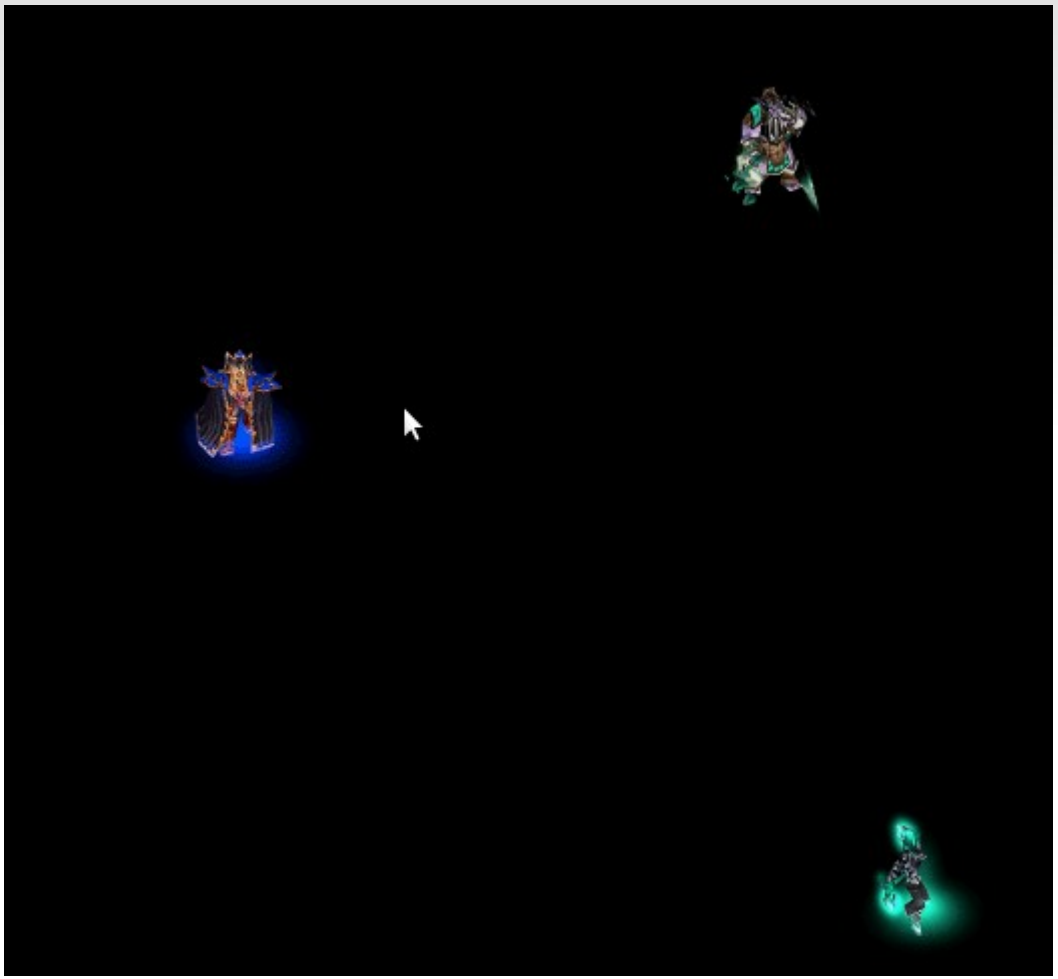
Lexer ： 文法分析器类。主要用于把读入的字节流根据规则分段。
既把长面条根据你要的尺寸切成一段一段，并不对其作任何修改。

Parser ： 解析器类。主要用于处理经过 Lexer 处理后的各段。一些具体的操作都在这里。

Domain-specific language

RPG Game Engine: Wisp

Domain-specific language



Domain-specific language

RPG Game Engine: Wisp

```
move up 10 steps

move left 1 step

attack
```

```
move down 2 steps and move left 1 step and attack
```

Domain-specific language

RPG Game Engine: Wisp

```
statement : command EOF
    ;

command : attack
    | move
    | ( move | attack ) ( KW_AND ( move | attack ) )+
    ;

move : KW_MOVE ( KW_DOWN | KW_UP | KW_LEFT | KW_RIGHT )
        stepnum=Integer ( KW_STEP | KW_STEPS )
        -> ^(TOK_MOVE KW_DOWN? KW_UP? KW_LEFT? KW_RIGHT? $stepnum)
    ;

attack: KW_ATTACK
        -> ^( TOK_ATTACK )
    ;
```

Domain-specific language

RPG Game Engine: Wisp

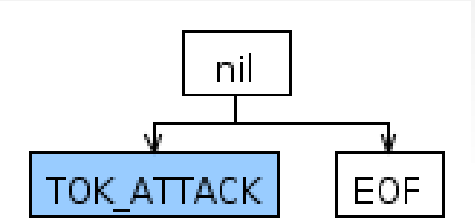
Engine:

```
var lexer = new WispLexer(
    new org.antlr.runtime.ANTLRStringStream(command));

var parser = new WispParser(
    new org.antlr.runtime.CommonTokenStream(lexer));

var root = parser.statement().getTree();
```

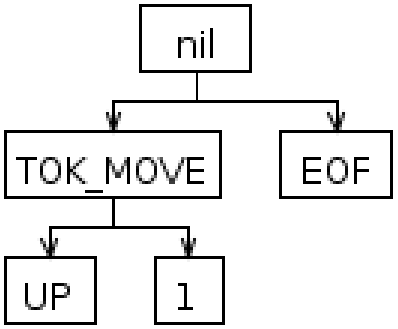
Domain-specific language

RPG Game Engine: Wisp

ATTACK



MOVE UP 1 STEP



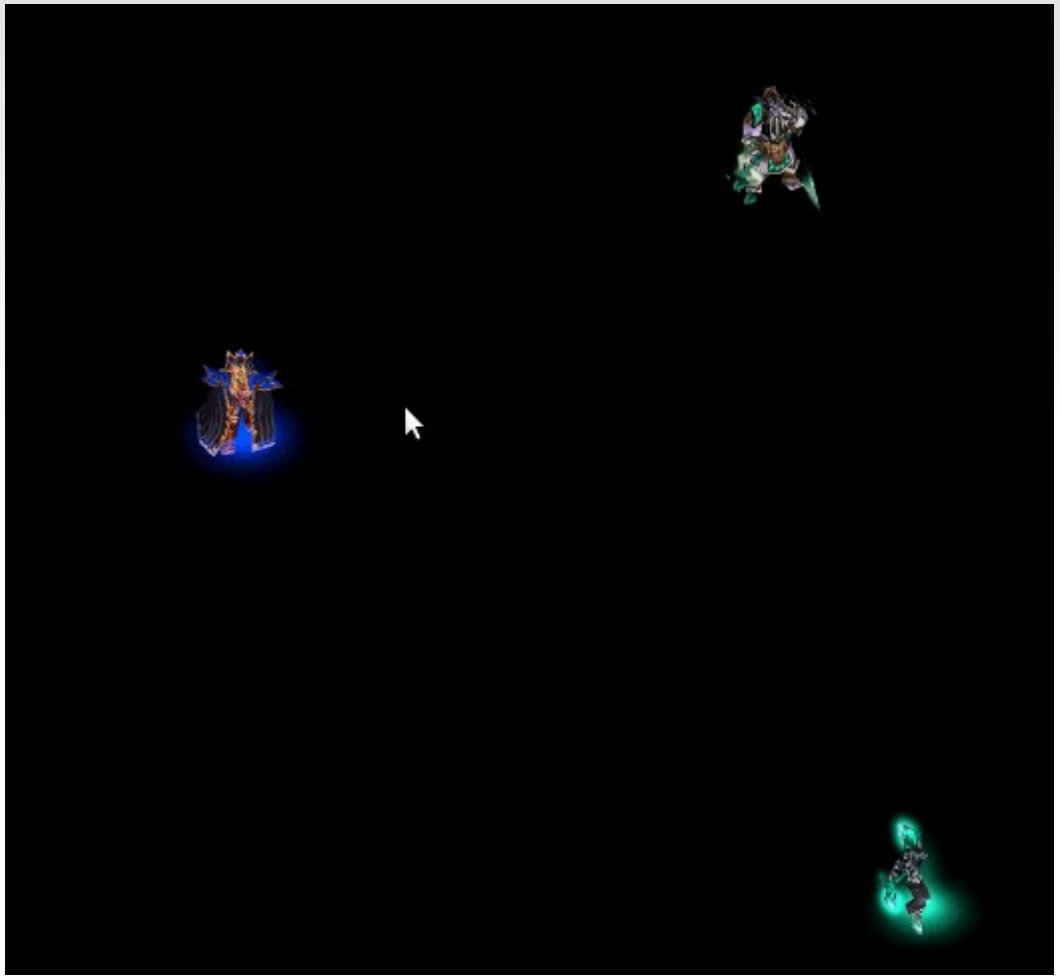
Domain-specific language

RPG Game Engine: Wisp

Client:

```
engine = new wisp.engine();

engine.execute('MOVE DOWN 1 STEP');

engine.execute('ATTACK');
```

Domain-specific language



移动： A S D W

攻击： J

END