

ANTLR3

C++ Codegeneration & Support Library

Ric Klaren
ric.klaren@gmail.com



Outline

- Aims & Problems
- Runtime
- Configuration
- Codegeneration
- Future



Aims

- Greater flexibility
 - Custom source position tracking
 - Read various encodings (UTFx)
- Tighter code
 - Templates?

Disadvantages?



Disadvantages

- Templates introduce fuzzier errors
- Can be trickier to debug
- Need more expertise to use



Prototype Lexer

Why the lexer?

- Contains most of the tricky bits

Approach:

- Handtranslate a small lexer in Java
- Port runtime bits until it works



Runtime Concepts

- Position → Tracks source location
- Stream → Stream of characters
- Lexer → Brings the above together + ...



Position with everything..

```
class FullPosition {
    unsigned line;
    unsigned column;
    size_t    offset;
public:
    FullPosition() : line(0), column(0), offset(0) {}
    FullPosition( unsigned l, unsigned c, size_t off )
        : line(l), column(c), offset(off) {}
    static void updatePosition( FullPosition* pos, unsigned char c )
    {
        pos->offset++;
        pos->column++;
        if ( c == '\n' )
        {
            pos->line++;
            pos->column = 0;
        }
    }
    unsigned int getLine() const throw()      { return line; }
    unsigned int getColumn() const throw()    { return column; }
    size_t getOffset() const throw()         { return offset; }
};
```

Stream

```
template<typename Position>
class CharStream : public Position
{
public:
    typedef Position position_type;
    typedef unsigned char char_type;
    typedef char_type item_type;
    typedef unsigned long offset_t;

    virtual char_type LA(offset_t i) const;
    virtual char_type LT(offset_t i) const;
    virtual bool eof(offset_t i);
    virtual void consume();

    const string& getFilename() const throw();
    const Position& getPosition() const;
    Position& getPosition();
    void setPosition( const position_type& pos );
    virtual string substring( offset_t start, offset_t stop ) const;
};
```

Lexer

```
template<typename StreamType, typename TokenType>
class Lexer : public TokenSource {
    TokenType* token;

    StreamType& getInput();
    const StreamType& getInput() const;

    void emit(TokenType* token);
    void match( const item_type* s ) throw(LexerMismatchException);
    void match( const str& s ) throw(LexerMismatchException);
    void matchAny();
    void match( item_type c ) throw(LexerMismatchException);
    void matchRange( item_type a, item_type b);

    void reportError(const BaseRecognitionException& e);
    void recover(const BaseRecognitionException& re);

};
```

Bring it together

```
typedef CharStream<FullPosition> MyStream;

class MyTokenBuilder {
public:
    static TokenFullPos* build( token_type tp,
                               const FullPosition& start,
                               const MyStream& stream,
                               channel_type chan = DEFAULT_CHANNEL )
    {
        return new TokenFullPos( tp,
                                 stream.substring( start.getOffset(), stream.getOffset() ),
                                 start.getLine(), start.getColumn(),
                                 stream.getFilename() );
    }
};

...

MyStream input( file, filename );
JavaParserLexer<MyStream,TokenFullPos,MyTokenBuilder> lexer( &input );

...

TokenFullPos* t = lexer.nextToken();
```

Position minimalist style...

```
class MinPosition {
    size_t    offset;
public:
    MinPosition() : offset(0) {}
    MinPosition( size_t off ) : offset(off) {}
    static void updatePosition(
        MinPosition* pos,
        unsigned char )
    {
        pos->offset++;
    }
    size_t getOffset() const throw()
    {
        return offset;
    }
};
```

Position minimalist style...

```
class MyTokenNoPosBuilder {
public:
    static TokenNoPos* build( token_type tp,
        const MinPosition&,
        const MyNoPosStream& )
    {
        return new TokenNoPos( tp );
    }
    static TokenNoPos* build( token_type tp,
        const MinPosition&,
        const MyNoPosStream&,
        channel_type )
    {
        return new TokenNoPos( tp );
    }
};
```

Gain?

- About 11% codesize on the java lexer
- Speed? Dunno...
-



Implementing codegenerator template

- One CPPTarget.java
- One CPP.stg

Time investment

- About a week
- 1 day for the antlr3 side
- Rest for support library tinkering...



TODO

- Figure out how to efficiently add UTFx decoding
 - Only decode character once
 - On the template or dynamic?
 - Figure out what can be template and what needs to be a class
 - Parser/Treeparser/Rewrite
-
-

Thanks to

- Terence
- The ANTLR Community
- PTS

