

A demonstration of technology metadata for ANTLR

Ralf Lämmel
on behalf of IOIcompanies.org

Set up rules to
assign **metadata**
describing the
roles of files as far
as **ANTLR** is
concerned.



METAdata

What's ANTLR?

<http://www.antlr.org>

```
company :  
    'company' STRING '{' department* '}' EOF;  
  
department :  
    'department' STRING '{'  
        ('manager' employee)  
        ('employee' employee)*  
        department*  
    '}' ;  
  
employee :  
    STRING '{'  
        'address' STRING  
        'salary' FLOAT  
    '}' ;
```

ANTLR style
grammar for
companies



Grammars were hot in the 70ies?
You are so wrong!

The ANTLR theme

[–] [ANTLR theme](#) **[ANTLR](#)-centric [grammarware](#) theme**

yapg	A ANTLR -based generator for text-to-object mappings
antlrAcceptor	An ANTLR -based acceptor for textual syntax
antlrLexer	Lexer-based processing with ANTLR
antlrObjects	Object/Text mapping for Java with ANTLR for parsing
antlrParser	Processing textual syntax with semantic actions of ANTLR
antlrTrees	Parsing text to trees and walk them with ANTLR
gra2mol	Grammar to model transformation with Grammar2Model
xtext	An XText- and Eclipse-based DSL editor

<http://101companies.org/index.php/>
Category:**[ANTLR theme](#)**

Wanted!

101 meta rules for ANTLR

When is a file ...

- ... the ANTLR library?
- ... an ANTLR grammar?
- ... an ANTLR-generated parser?
- ... a source that imports ANTLR?

(See more rules online.)

“101meta” is the
metadata
language at hand.

“Condition” of the rule

```
{  
  "basename" : "#^antlr-(.*)\\.jar$#",
```

Expressed relationship

```
  "metadata" : {  
    "partOf" : "ANTLR",  
    "comment" : "The ANTLR library, Version $1"  
  }  
}
```

Assigned metadata

*I0I*meta

```
{  
  "suffix" : ".g",  
  "metadata" : {  
    "inputOf" : "ANTLR",  
    "comment" : "An ANTLR grammar"  
  }  
}
```

Expressed relationship

```
{
  "basename" : "#^.*Parser\\.java$",
  "content" : "// \\$ANTLR.*\\.g",
  "metadata" : [
    {
      "outputOf" : "ANTLR",
      "comment" : "An ANTLR-generated parser"
    },
    { "concept" : Parser }
  ]
}
```

Condition on text of file

Metadata for software concept

```
{  
  "suffix" : ".java",  
  "predicate" :  
    "technologies/Java_platform/javaImport.sh",  
  "args" : [ "org.antlr.runtime" ],  
  "metadata" : {  
    "dependsOn" : "ANTLR",  
    "comment" : "A source that imports ANTLR"  
  }  
}
```

Programmatic condition

Java package to be tested for

Metadata management

- Metadata **declaration**
with rules, as demonstrated
- Metadata **assignment**
with a rule engine @ IOIworker
- Metadata **exploration**
in the browser at **data**.ioicompanies.org
or with **explorer**.ioicompanies.org

Disclaimer: This is not even beta.

Metadata declaration

All the previous rules are stored here:

[https://github.com/I0Icompanies/I0Irepo/blob/master/
technologies/ANTLR/.I0Imeta](https://github.com/I0Icompanies/I0Irepo/blob/master/technologies/ANTLR/.I0Imeta)

I0Imeta metadata may be declared at all levels:

- specific ***technologies***
- specific ***languages***
- specific ***contributions***

Have a look at the “**.I0Imeta**” files in *I0Irepo*.

Metadata assignment

```
{
  - units: [
    - {
      id: 33,
      - metadata: {
        outputOf: "ANTLR",
        comment: "An ANTLR-generated parser"
      }
    },
    - {
      id: 33,
      - metadata: {
        concept: "Parser"
      }
    },
    + { ... },
    + { ... },
    + { ... },
    + { ... },
    + { ... },
    + { ... },
    - {
      id: 36,
      - metadata: {
        comment: "A source that imports ANTLR",
        dependsOn: "ANTLR"
      }
    }
  ],
  + metrics: { ... }
}
```

IOI worker continuously walks over *IOI* repo to construct data for files matches *IOI* meta rules. In fact, each file is associated with such matches.

<http://data.ioicompanies.org/resources/contributions/antlrObjects/org/softlang/parser/CompanyParser.java.summary.json>

Metadata exploration

<http://explorer.10lcompanies.org/>

The screenshot shows a web browser window with the URL `explorer.10lcompanies.org/show.html?contributions/antlrObjects/`. The page displays a tree view of the `antlrObjects` contribution, organized into four panels: **Files**, **Languages**, **Technologies**, and **Concepts**. The **Files** panel shows a directory structure with `CompanyParser.java` selected. The **Languages** panel shows a tree where `CompanyParser.java` is highlighted under the `Java` language. The **Technologies** panel shows a tree where `CompanyParser.java` is highlighted under the `ANTLR` technology. The **Concepts** panel shows a tree where `CompanyParser.java` is highlighted under the `Parser` concept. Below these panels is a **Source view** showing the source code of `CompanyParser.java`. The code includes package declarations, imports, and the start of the `CompanyParser` class.

Source view:

```
1. // $ANTLR 3.2 Sep 23, 2009 12:02:23 Company.g 2011-05-04 01:08:48
2.
3. package org.softlang.parser;
4. import org.softlang.company.*;
5. import java.io.File;
6. import java.io.FileInputStream;
7. import java.io.IOException;
8.
9.
10. import org.antlr.runtime.*;
11. import java.util.Stack;
12. import java.util.List;
13. import java.util.ArrayList;
14.
15. public class CompanyParser extends Parser {
```

Disclaimer: This is not even beta.

A related paper

“Linking Documentation and Source Code in a Software Chrestomathy” by *Jean-Marie Favre, Ralf Lämmel, Martin Leinberger, Thomas Schmorleiz, and Andrei Varanovich*. Under submission.

<http://softlang.uni-koblenz.de/I0I/meta/>

Thanks!

- *Contact I0I companies*
- *Email: I0I companies@gmail.com*
- *Twitter: @I0I companies*
- Material for this demonstration:
 - <https://github.com/I0I companies/I0I media/tree/master/antlr I0I demo>