



Xtext

Build your own DSL

Tomasz Kleszczyński,
2015-11-27

Agenda

- Introduction
- Xtext
- Demos
- What's new in Xtext 2.9

Domain specific language

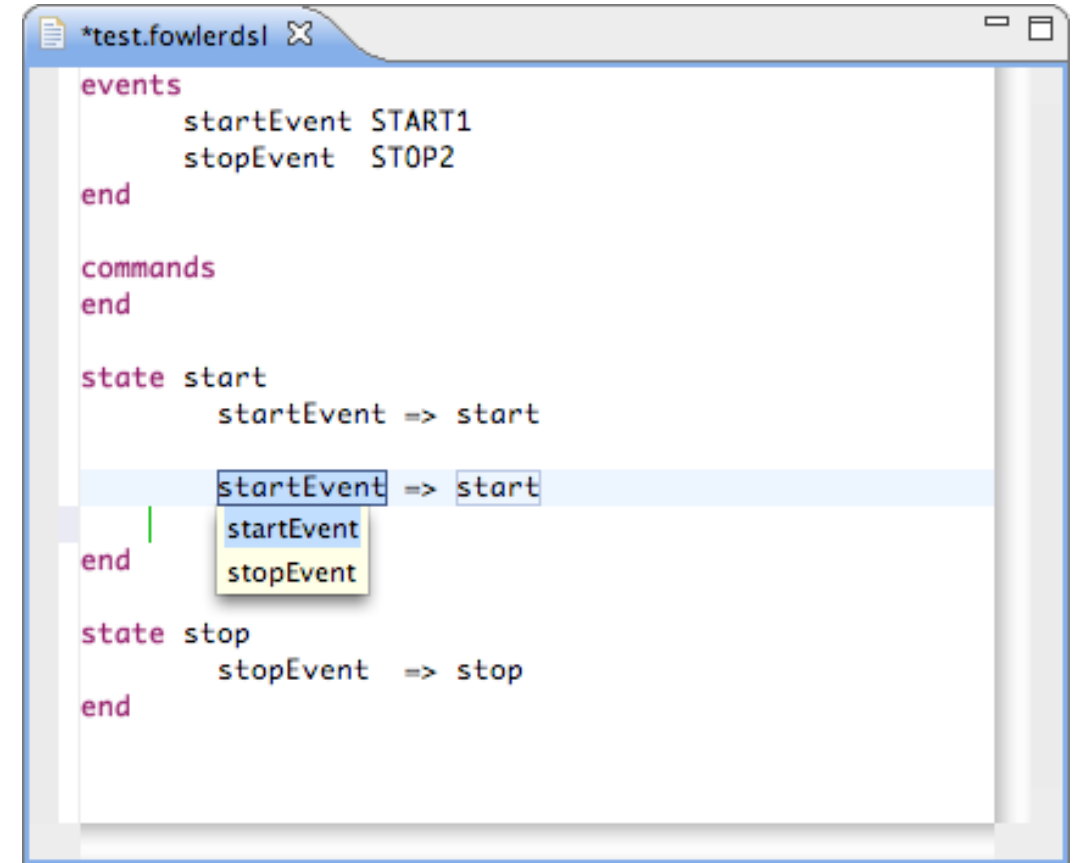
```
select * from Named n where n.name = "foo*"
      and n.used order by n.name desc
```

Examples:

- SQL
- Html (Haml)
- Css (LESS / SASS)
- MATLAB
- **YourOwnDSL**

Xtext

- Eclipse project – EPL license
 - Uses ANTLR3, EMF, Google Guice
- Language development framework
 - Set of DSLs and API to describe the different aspects of programming language
 - Compiler components independent of Eclipse or OSGi:
 - parser
 - type-safe abstract syntax tree (AST)
 - serializer
 - code formatter
 - scoping framework
 - linking
 - IDE components
 - Eclipse
 - IntelliJ IDEA
 - Translation to Java using XBase



```
*test.fowlerdsl
```

```
events
  startEvent START1
  stopEvent STOP2
end

commands
end

state start
  startEvent => start
end

state stop
  stopEvent => stop
end
```

The screenshot shows a code editor window titled '*test.fowlerdsl'. The code is written in a DSL style with sections for 'events', 'commands', and 'state'. The 'state start' section is highlighted, and a dropdown menu is open over the 'startEvent => start' line, showing options for 'startEvent' and 'stopEvent'.

Grammar

- Language Declaration
- EPackage Declarations
 - EPackage Generation
 - EPackage Import
- Rules
 - Terminal Rules
 - Parser Rules
 - Hidden Terminal Symbols
 - Data Type Rules
 - Enum Rules
- Ecore Model Inference
 - EPackage
 - EClass
 - EEnum
 - EStructuralFeature

```
1 grammar pl.zgora.jug.xtext.DemoDsl with org.eclipse.xtext.common.Terminals
2
3 generate demoDsl "http://jug.zgora.pl/xtext/DemoDsl"
4
5 LoggerConfig:
6     (debug?='debug'? & (scan?='scan' ('period' period=STRING)?))?
7     appenders+=Appender+
8     rootLogger=RootLogger;
9
10 Appender:
11     'appender' name=ID class=FQN pattern=STRING?;
12
13 RootLogger returns Logger:
14     'root' level=Level 'append-to' appenders+=[Appender|ID]+
15     (LBRACE loggers+=Logger* RBRACE)?;
16
17 Logger:
18     'logger' name=FQN level=Level? ('append-to' appenders+=[Appender]+)?
19     (LBRACE loggers+=Logger* RBRACE)?;
20
21 enum LevelEnum:
22     UNDEFINED | OFF="off" | ERROR="error" | WARNING="warn" | INFO="info" | DEBUG="debug" | TRACE="trace";
23
24 enum Level returns LevelEnum:
25     OFF="off" | ERROR="error" | WARNING="warn" | INFO="info" | DEBUG="debug" | TRACE="trace";
26
27 FQN:
28     ID ('.' ID)*;
29
30 terminal LBRACE: '{';
31
32 terminal RBRACE: '}';
```

Demo

All demos are based on Xtext 2.8

- Formatting
- Generation
- Scoping
- Validation
- Content assist
- Labels / outline
- Quick fix

Xtext 2.9 - What's new

- IntelliJ IDEA As An Alternative To Eclipse
 - Language Editors for IntelliJ IDEA
- Headless Builds For Xtext Projects
 - Gradle / Maven
 - Target selection (Eclipse, IntelliJ IDEA, web)
- New Project Wizard
- Web Support
- New Grammar Language Features
 - Explicit rule calls and super rule calls
 - **name=super::ID**
 - **feature=com::acme::MyLang::SomeRule**
 - Parser rule fragments
 - **fragment DeclarationOwner: declarations+=Declaration+;**
 - Parameterized rule calls
 - **IdOrKeyword<AllowKeyword>: <AllowKeyword> 'keyword' | ID**

References

- www.eclipse.org/Xtext
- zarnekow.blogspot.de
- blog.efftinge.de
- [Scoping and linking](#)

Examples:

- github.com/tkleszczynski/xtext-jug.git