

Xtext workshop – level: Advanced SIG Domain Language Engineering

Peter Vermeulen, Meinte Boersma

October 11th, 2010





Objective

“Advanced” objective

- Full-fledged grammar including expressions
- Enhance editor experience



Topics

“Advanced” topics

- Scoping
- Validation
- Chaining
- Expressions
- Outline
- Formatting
- And more ...

Scoping

Implement scoping:

- Look up `WebGuiScopeProvider` class
- Add methods with signature

`IScope scope_Type_crossReference (Type2, EReference)`

- Exercise: limit scope of referenced *Feature*-s to `context.features`



Validation

Implement validation:

- Look up `WebGuiJavaValidator` class
- Add methods with signature

```
@Check public void checkName(Type)
```

- Call `warning/error` to flag them
- Exercise: validate that name of *Entity* starts with a capital

Chaining

Implement chaining in our example DSL:

- Looks like:

feature1.feature2...

feature1 member of context entity, *feature2* member of *feature1*'s type, etc.

- Grammar:

```
DomainPath:      feature=[Feature] (tail=DomainPathTail)?;  
DomainPathTail: '.' feature=[Feature] (tail=DomainPathTail)?;
```

- Grammar forces instantiation of `DomainPathTail`:
scope is only called on instantiated objects.

Expressions 1

Often-recurring sub language, typical elements:

- operators; characteristics:
 - *pre-/in-/postfix*: $-a$, $c + d$, $d++$
 - *left/right associativity* of infix operators:
 $a * b * c$ is $(a * b) * c$ or $a * (b * c)$?
 - *precedence* (higher/lower):
 $a + b * c$ is $a + (b * c)$ or $(a + b) * c$?
- operands: literals, references, etc.
- parentheses for grouping and overriding precedence and associativity



Expressions 2

Typical problem:

- Xtext cannot handle *left-recursion*:

Expression:

```
( left=Expression '+' right=Expression ) | intLit=INT;
```

The rule call to Expression would cause infinite recursion in the parser, because the rule doesn't consume any input until a match.

- Solution: left-factor the grammar.
- New problems: precedence hard to get right or ASTs a lot larger.
- New solution: follow pattern outlined here.



Expressions 3

Simple arithmetical example language:

- Operators all are left-associative and infix
- Language elements, in ascending order of precedence:
 - $+$, $-$
 - $*$, $/$
 - integer literals, references to *Feature-s*
 - parentheses

Expressions 4

Xtext grammar concepts needed:

- Unassigned rule call: returns parse result as the **current** object.
- **returns**-clause: forces inheritance and defers type creation.
- Assigned action $\{ \textit{Type} . \textit{feature} = \textbf{current} \}$ instantiates *Type* and assigns the **current** object to *feature*.
- (not to be confused with) Simple action $\{ \textit{Type2} \}$: only instantiates object of *Type2*, no assignment.

Expressions 5

The resulting grammar:

Expression: Addition;

Addition **returns** Expression:

Multiplication

(({Add.left=**current**} '+' | {Subtract.left=**current**} '-')
right=Multiplication)*;

Multiplication **returns** Expression:

Value

(({Multiply.left=**current**} '*' | {Divide.left=**current**} '/')
right=Value)*;

Value: NumberLiteral | DomainPath;

NumberLiteral: value=INT;



Expressions 6

More info on implementing expressions:

- Check the excellent/definite [blog entry](#) by Sven Efftinge
- Movie from that blog to illustrate parsing behavior

Outline (contents)

To change the contents of the outline:

- Look up `WebGuiTransformer` class
- Add methods with signature:

```
public List<EObject> getChildren (Type)
```

- Return a list of children, or `NO_CHILDREN`
- Exercise:
 1. Remove the features from entities, and the page elements from pages.
 2. Show the used entities on a page



Outline (labels)

To change the label text, and icons in the outline:

- Look up `WebGuiLabelProvider` class
- Add methods with signature:
`String text(Type)`
`String image(Type)`
- Return a label text, and the file name of the icon
- Default icon location is the `icons` directory in the UI Eclipse project.
- Icons: use PNG format, and 16x16 is a good dimension
- Exercise: change some label texts and icons

Formatting

Implement formatting:

- Look up `WebGuiFormatter` class, adapt the method, add:

```
grammar = (WebGuiGrammarAccess) getGrammarAccess();
```

Provides a representation of our Xtext grammar:

- `MyDslGrammarAccess`, methods:
 - `TypeElements` `getTypeAccess()`
- `TypeElements`, methods:
 - `ParserRule` `getRule()`
 - `Keyword` `getKeywordKeyword_0()`
- Some utility functions in `AbstractGrammarElementFinder`



Formatting

Configure formatting:

- Maximum characters on a line:
 - `cfg.setAutoLinewrap(120)`
- Force a new line:
 - `cfg.setLinewrap().after(keyword or rule)`
- Indentation:
 - `cfg.setIndentationIncrement().after(keyword or rule)`
 - `cfg.setIndentationDecrement().after(keyword or rule)`
- Remove white space:
 - `cfg.setNoSpace().after(keyword or rule)`



Formatting

Exercise: make a nice formatter!

And more ...

- jvmType
- Run Xtext from the command line (easy!)
- Distribute your Xtext editor as RCP application
- Modularize your DSL
- Refactoring support (in next version)
- Use of an existing Ecore model



And still more ...

- Code templates
- Quick fixes
- UML reader
- Use of parse tree reconstructor
- Add instance data via resource provider
- Transformations with Xtend
- Add a graphical view with GMF

Xtext perfect?

No way!

- Defining grammars could be easier, see for example SDF
- You need to regenerate and restart Eclipse each time you change the grammar
- Formatting is way too difficult
- Where is my easy graphical representation?

Xtext in the Wild

Existing projects based on Xtext:

- Sculptor ([home](#))
 - Persistent domain model + CRUD web user interface.
 - Web applications based on JPA, Spring, Google App Engine, etc.
- ARText ([slide](#))
 - AUTOSAR (Automotive Open Systems Architecture) models
 - Huge models
 - Only available for members, for example BMW
- AXDT ([home](#))
 - Implementation of ActionScript of Adobe in Xtext
- Chess example ([link](#))



Together. Free your energies

www.capgemini.com