GF Eclipse Plugin

MOLTO 4th Project Meeting, Zurich

John J. Camilleri

University of Gothenburg

john.j.camilleri@chalmers.se

7 March 2012

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. FP7-ICT-247914.

Motivation

How do we write GF grammars?

Text editor + console

- Editor modes for Emacs, Gedit, Geany
- Syntax highlighting, rudimentary auto-completion

but...

- Not tailored for GF

Motivation

Web IDE

- Great for jumping right in
- No installation/compilation, always up-to-date
- Storage in the cloud
- Controlled environment

but...

- Limited module inheritance
- Impractical for large projects

The Plugin

The GF Eclipse Plugin (GFEP)

- Work Package 2:
  Grammar Developer's Tools
- Version 1.4.0 (yesterday!)
- Uses Xtext framework
- September 2011 – present

Motivation

Why Eclipse?

- Leverage existing IDE features
- Syntax errors & semantic warnings
- Inline documentation
- Context-sensitive suggestions
- Wizards, code snippets
- Code completion, formatting
- Navigate external libraries
- Run code and test suites directly from IDE

The Plugin

Approach

- Layered *on top* of GF, not replacing it
- Requires GF already installed on system
- Eclipse **project** concept
    - Regular GF files + some metadata files
- Eclipse **builder** concept
    - Compiles your code as you write it

Target audience

- All levels of GF development
- People who like IDEs

Getting started

Requirements

- GF $\geq$ 3.3.3
- Eclipse $\geq$ 3.6

Installation

- Use Eclipse update site URL:
  `http://www.grammaticalframework.org/eclipse/release/`
- Open GF perspective

Example

Foods grammar

1. Open/edit some existing files
2. Syntax errors, other warnings
3. Auto-complete
4. Outline view

Example

- Automatic building
- Unresolved names
- Jump to definition
- External libraries

Treebank testing

1. Create a treebank
2. Compile grammar and linearize trees
3. Manually correct and save as *Gold Standard*
4. For each change, repeat (2) and compare against (3)

We can now do this directly in the plugin!
Naming convention:

- `abc.trees`
- `abc.trees.out`
- `abc.trees.gold`

Example

Let's add a new language

1. Clone from English to Dutch
2. Change some strings
3. Create gold standard
4. Test against it, iterate

Status

What GFEP can and can't do

GFEP **does**:

- leverage useful IDE features
- give you errors and warnings as you type
- help you navigate local and external cross-references
- ease the development-test cycle

Status

What GFEP can and can't do

GFEP **does**:

- leverage useful IDE features
- give you errors and warnings as you type
- help you navigate local and external cross-references
- ease the development-test cycle

GFEP **doesn't**:

- type-check its suggestions
- write your grammars for you
- expose any models, bindings or APIs

Back matter

We need feedback!
Please report bugs and request features.

Links

Web	`www.grammaticalframework.org/eclipse`
Source	`github.com/GrammaticalFramework/gf-eclipse-plugin`
Bug tracker	`github.com/GrammaticalFramework/gf-eclipse-plugin/issues`
Email	`john.j.camilleri@chalmers.se`