Using Xtext and Xtend to Create Bmod
Reading Report

Randy Paredis, s0151613

Master in Computer Science, University of Antwerp

Abstract

This document is meant to lead as an introduction to the actual project. It is a culmination of all the sources I've read and explains the basic concepts of Xtext and Xtend. All this information can also be found in the final report.

Keywords: Bmod, DSL, Java, Modelling, Model Driven Engineering, Xtend, Xtext

1. Introduction

In this introductionary paper, I will try and list the strengths and drawbacks for using Xtext and Xtend. In Section 3, it will also discuss the general workflow needed to obtain a custom-made DSL (Domain Specific Language).

In the final section (Section 4), a series of possible next steps is also listed as example, but the project is not contained to these possibilities.

2. Xtext

Xtext [1] is a Java library/framework that allows software engineers to easily create custom DSLs.

It is an **open-source** project, which allows for full flexibility, but unfortunately the code is (as far as I've been able to go through it) quite complex and an unfortunate overhead for trying to understand certain features. Coupled with this, Xtext also undergoes some **continuous integration**, which implies it is not a dead project, in fact it is very much still in development.

Using the strong `Antlr 4` [2] grammar behind the scenes, `Xtext` is fully **grammar-driven** and implements some custom and complex features that usually don't appear in a grammar file (cross references, easy range operator...).

The project is based at the Eclipse Org [3], but was not designed to be solemnly an Eclipse plug-in. Originally, the project was meant to also be available for JetBrains' IntelliJ and web applications, but unfortunately these other implementations have some difficulties.

> *For IntelliJ IDEA the situation is different. Neither the Xtext integration has been updated with the last release, nor has Jetbrains yet started to work on LSP support. The code for the IDEA integration is quite extensive and deep. So deep that we get regularly broken because we use non-public API. Since the demand for IDEA integration is not high, maintaining it doesnt make sense to us.* [4]

This quote made me choose to use `Xtext` in a normal fashion, by using Eclipse Photon. Although, it was possible to get the ACE-webeditor have the valid syntax-highlighting[1].

3. Workflow

When you create a new project in Eclipse, you will go through a wizard that helps you by defining the base grammar name, the file extension to be associated with it and some optional testing frameworks. At this point in time, I've experienced some difficulties with the JUnit 5-framework, leaving me to use the JUnit 4-framework, where these issues do not occur.

When you have your project, a default grammar file that allows you to create certain greetings has been created. This file can be turned into any grammar you desire for your `DSL`. When this file is compiled with the `MWE2`-workflow file, you will get a generator, validator and a scopehandler class. These

[1]This general idea was discarded when it became clear that the McGill site did not allow for uploading `JavaScript` files.

classes respectively allow for generating code from the DSL-files, (semantically) validating the syntax of these files and make sure the scoping of these files is done as required (by default, this is the normal Java-scoping).

These classes can be implemented in both Java and Xtend, which is a custom DSL, based upon Java.

4. Next Steps

In this project, it seems that the following steps can introduce a good project[2]:

1. Create a grammar for Bmod, a building modelling language, as was defined in the assignments.
2. Allow for exporting it to HTML, CSS and JavaScript, so it would be possible to simulate any floor in a browser.
3. Get the web-editor to work as Eclipse does (make it so the ACE-engine also calls the generator).
4. Make the project available on the McGill site for life testing of Bmod.

References

[1] Xtext home page.
URL https://xtext.org

[2] Antlr home page.
URL https://www.antlr.org/

[3] Eclipse home page.
URL https://eclipse.org

[4] TypeFox, Xtext LSP vs. Xtext Web.
URL https://typefox.io/xtext-lsp-vs-xtext-web

[2]Note: these steps are not necessarily the steps that will be taken in this project.