A reporting plugin for OWL ontologies

Massimo Coletti
Banca Finnat Euramerica S.p.A.

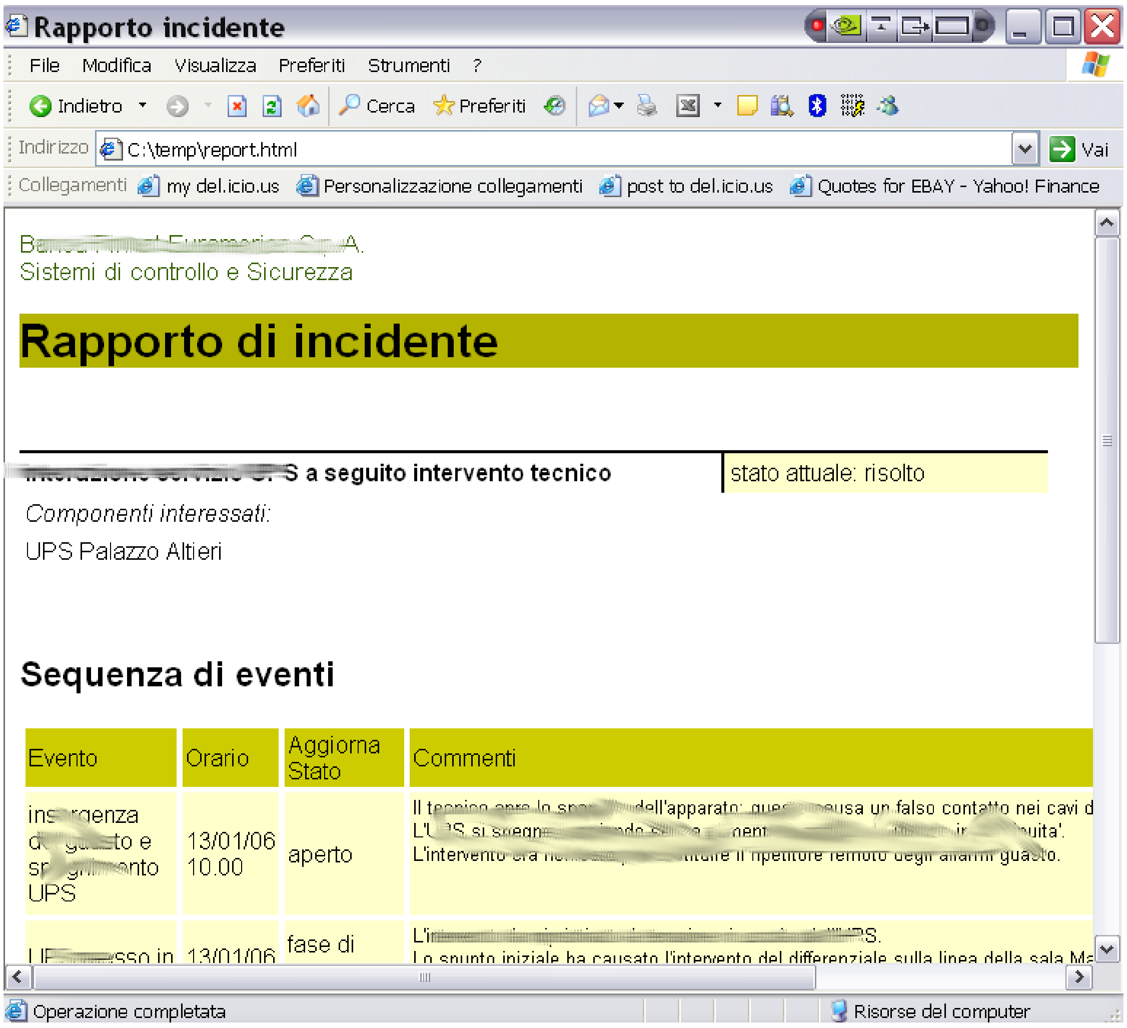
September 26, 2006

Introduction

This document is intended as a user documentation for **velociowl** (this is still a temporary name), a plugin that integrates the power of the Velocity templating tool inside the Protégé platform.

Despite the power of Protégé, and the documentation and graphical plugins available, if you want to create a customized report (or generate other kind of files with the ontology content), you will find this tool useful.

Velocity is a component that parses a *template* (a file with some custom content interspersed with *velocity macros*) substituting the actual values of some variables in the text. The variables can be any Java object. Velociowl simply makes the OWL model, implemented by the OWL plugin API, available in the template; the user can access the model content using the methods of the API.



Ba[redacted]A.
Sistemi di controllo e Sicurezza

Rapporto di incidente

[redacted]S a seguito intervento tecnico | stato attuale: risolto

Componenti interessati:

UPS Palazzo Altieri

Sequenza di eventi

Evento	Orario	Aggiorna Stato	Commenti
ins[redacted]genza d[redacted]to e sp[redacted]nto UPS	13/01/06 10.00	aperto	Il te[redacted]dell'apparato: que[redacted]usa un falso contatto nei cavi d L'U[redacted]S si spegne[redacted]uita'. L'intervento[redacted]tuire il ripetitore remoto degli allarmi guasto.
UP[redacted]sso in	13/01/06	fase di	L'i[redacted]PS. Lo spunto iniziale ha causato l'intervento del differenziale sulla linea della sala Ma

The above screen shot shows a sample of an html report generated from the ontology.

Although Velocity is really a java-oriented tool, it is quite easy to use it also by non-programmers; you may want to jump to the “Cookbook” chapter for some examples.

The plugin is actually made by two components:

- a standalone processor: it can be invoked from the command line, for batch execution, or called by other applications;
- a Protégé OWL plugin, which allow you to manage the tool from within the normal user interface.

The following chapters are intended as a quick-start guide. Some further considerations can be found in the “Cookbook” or in the “Notes” chapters.

A last note: I am not an English professional writer: I am afraid that you will be aware of this.

Setting up

As for any Protégé plugin, simply unzip the distribution content under the *plugins* directory, and restart Protégé.

You should end up with a new directory: net.mcoletti.protege.velocity, with the velociowl jar and a separate jar with the velocity distribution. In this way, you will be free to update to newer versions of Velocity.

If everything is ok, you will find a new entry “VelocityTab”, among the configurable tabs.

WARNING: The Velocity Tab is designed to work with OWL ontologies, I don't think that it will work with frames projects. I hope to fix this in the future.

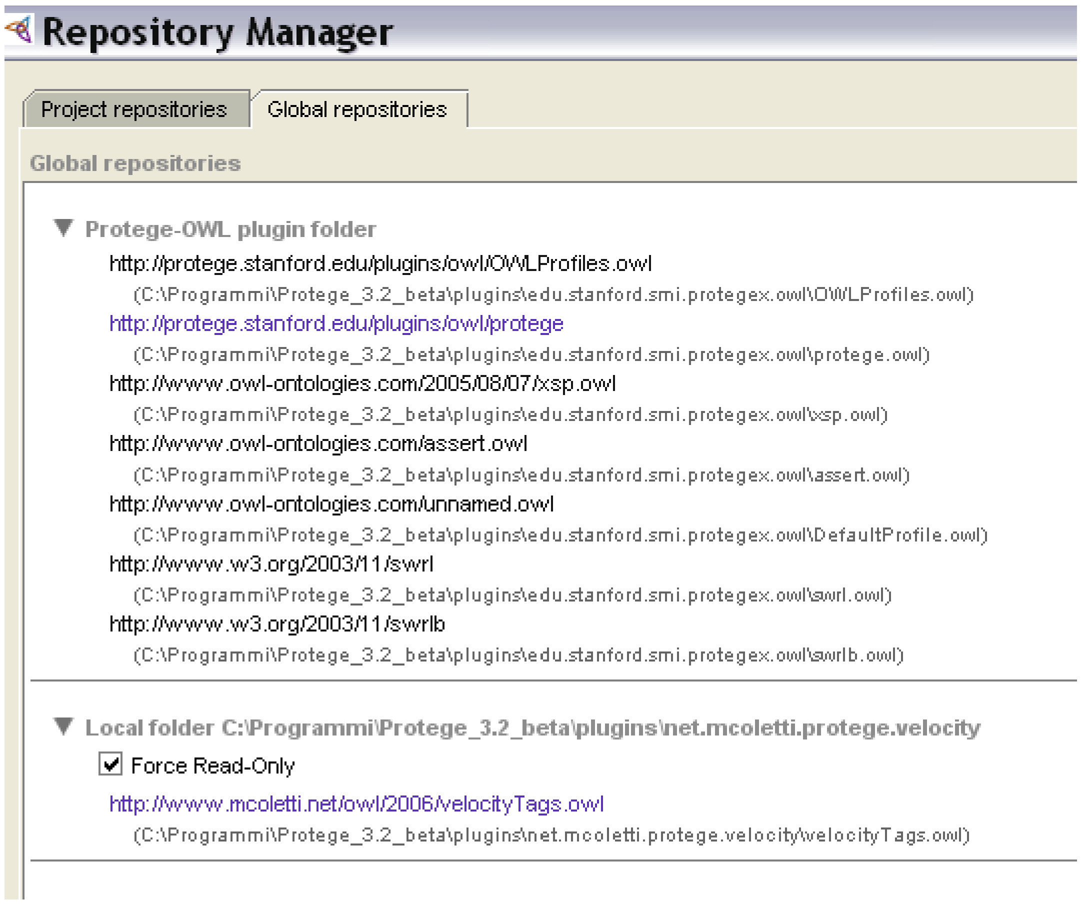
The annotations ontology

In order to use easily the plugin, you may import an ontology with few annotation properties. These properties allow you to store "inside" your ontology, some references to the report templates and the templates libraries.

The user guide will provide more details about this feature.

If you want to import this ontology, my suggestion is to create a *global* repository, pointing at the plugin directory.

Then you will import the ontology: and you **must** specify "velocity" as the ontology prefix. Unfortunately, this prefix is hardcoded in the application.



Protege-OWL plugin folder

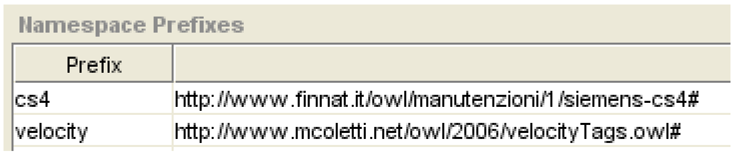
```
http://protege.stanford.edu/plugins/owl/OWLProfiles.owl
   (C:\Programmi\Protege_3.2_beta\plugins\edu.stanford.smi.protegex.owl\OWLProfiles.owl)
http://protege.stanford.edu/plugins/owl/protege
   (C:\Programmi\Protege_3.2_beta\plugins\edu.stanford.smi.protegex.owl\protege.owl)
http://www.owl-ontologies.com/2005/08/07/xsp.owl
   (C:\Programmi\Protege_3.2_beta\plugins\edu.stanford.smi.protegex.owl\xsp.owl)
http://www.owl-ontologies.com/assert.owl
   (C:\Programmi\Protege_3.2_beta\plugins\edu.stanford.smi.protegex.owl\assert.owl)
http://www.owl-ontologies.com/unnamed.owl
   (C:\Programmi\Protege_3.2_beta\plugins\edu.stanford.smi.protegex.owl\DefaultProfile.owl)
http://www.w3.org/2003/11/swrl
   (C:\Programmi\Protege_3.2_beta\plugins\edu.stanford.smi.protegex.owl\swrl.owl)
http://www.w3.org/2003/11/swrlb
   (C:\Programmi\Protege_3.2_beta\plugins\edu.stanford.smi.protegex.owl\swrlb.owl)
```

Local folder C:\Programmi\Protege_3.2_beta\plugins\net.mcoletti.protege.velocity

Force Read-Only [✓]

```
http://www.mcoletti.net/owl/2006/velocityTags.owl
   (C:\Programmi\Protege_3.2_beta\plugins\net.mcoletti.protege.velocity\velocityTags.owl)
```

The above screen shot shows the repository manager.



Namespace Prefixes

Prefix	
cs4	http://www.finnat.it/owl/manutenzioni/1/siemens-cs4#
velocity	http://www.mcoletti.net/owl/2006/velocityTags.owl#

This picture shows how the imported ontology will appear in your Metadata Tab.

Velocity Templates

Now, you should spend a little time learning the Velocity tool.

The documentation is available on: http://jakarta.apache.org/velocity/docs/user-guide.html

Using the plugin

The first paragraphs will instruct you on using the velocity: annotations in your ontology as a facilitator in report management. This is not mandatory, however.

The annotation elements can be "named", I may want to coll the library /share/dept/somethingawful with the nickname "mary". As annotation properties in OWL have a simple content, you will prefix your content with the name, using a triple semicolon as separator, for example: "mary:::/share/dept/somethingawful".

All the annotation properties will be created under the "Ontology" element of your OWL file, i.e. in the Metadata Tab.

The templates library

Likely, you will keep your templates in a folder, or in many different folders (personal, testing, production, etc.).

You can store in the ontology the location for your libraries, in order to easily remember them.

Each library will be coded using a `velocity:templateLibrary` annotation,.

The report templates

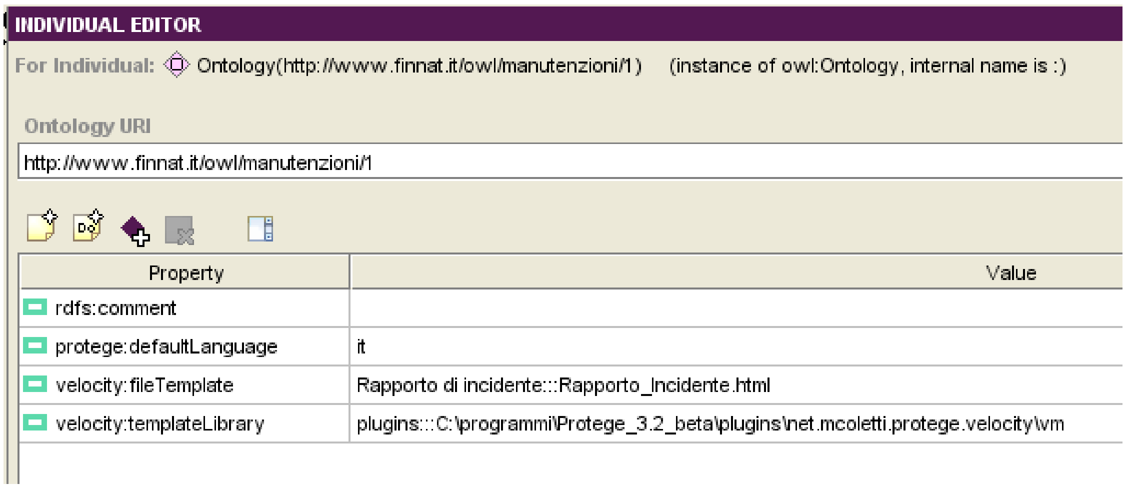
You may choose to store the templates in files. In order to remember the names, you will create a `velocity:fileTemplate` annotation.

The file template require you to code an (optional) nickname, a triple semicolon separator, and the template file name.

You may also want to store the template content inside your ontology, allowing you to distribute easily the templates with the ontology.

In this case you will use a `velocity:stringTemplate` annotation. A string template requires the usual optional nickname (but in this case is strongly recommended), followed by the template content.

The following example shows a Metadata Tab with two annotations, defining a library and a file template.


INDIVIDUAL EDITOR

For Individual: Ontology(http://www.finnat.it/owl/manutenzioni/1) (instance of owl:Ontology, internal name is :)

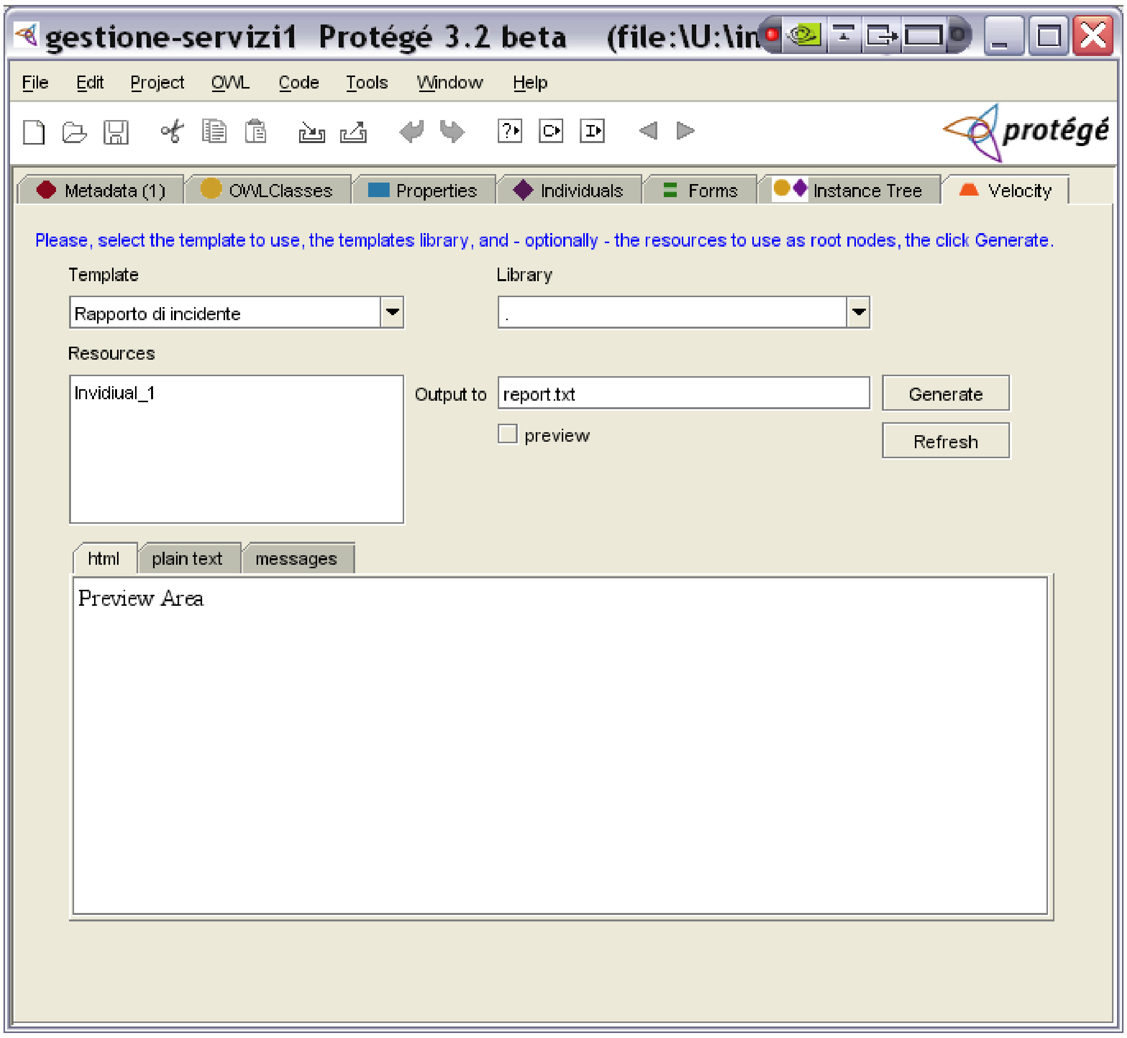
Ontology URI

http://www.finnat.it/owl/manutenzioni/1

Property	Value
rdfs:comment	
protege:defaultLanguage	it
velocity:fileTemplate	Rapporto di incidente:::Rapporto_Incidente.html
velocity:templateLibrary	plugins:::C:\programmi\Protege_3.2_beta\plugins\net.mcoletti.protege.velocity\vm

The plugin Tab

Once you have enabled the plugin, it will show up in a dedicated Tab.


gestione-servizi1 Protégé 3.2 beta (file:\U:\in

File Edit Project OWL Code Tools Window Help

Metadata (1) | OWLClasses | Properties | Individuals | Forms | Instance Tree | Velocity

Please, select the template to use, the templates library, and - optionally - the resources to use as root nodes, the click Generate.

Template: Rapporto di incidente

Library: .

Resources: Invidiual_1

Output to: report.txt

preview

Generate

Refresh

html | plain text | messages

Preview Area

The Tab shows the following controls:

Template selector

You can select (using the nickname) all the file and string templates that you have defined.

It is also possible to enter directly the file name.

Library selector

Again you will see the available libraries. As a default, there is always the '.' (dot) library. This library will read template form the installation directory of Protégé (not the plugin directory).

It is also possible to enter a path here.

Resources

In this box, you can enter, one per line, the internal names of some resources that you want to make available in your report.

As an example, you may select some individuals, or a class.

The resources that you will list, will be available in the Velocity Context as a Collection, with the name `$resources`. It is up on you to math the resources that you have selected with the type expected by your report.

Yes, this control requires many improvements!

Output to

Finally, you can enter here the full name of your desired output file. If the directory is missing, the file will be written in the Protégé directory.

Generate

This button starts the Template processing. Some status messages are written in the text area in the lower part of the screen. The file specified in the "output" box will be written (or overwritten).

You can also have a preview of your output, if you check the **preview** box. In this case, the file is not written.

Refresh

Reloads the lists of libraries and templates; useful if you have added something.

Finally, there is a display area, where you can preview the report output in html or plain text format without writing the file, as well as you can see the messages (and errors from the Velocity parser) generated during the transformation.

The Template Renderer Component

Behind the scenes, operates a little class, responsible for the integration of Protégé and Velocity.

This class puts in the Velocity Context some objects, that are required for a proper processing of the OWL Model.

$name

A simple string.

$kb

The OWL Model. This is perhaps the most important.

If you want a reference about this, you may look at the OWL Plugin documentation: http://protege.stanford.edu/plugins/owl/api/index.html

$owl

This is a sort of Swiss Army knife for the report designer. It is a java Class, equipped with some utility methods that provide a simplified access to the OWL Plugin API of Protégé. I think that many non-programmers will find it useful.

Please, be aware that the $owl object is related to $kb: if you want to add other ontologies to the Context, you can still use $owl on them, but some methods rely on $kb.

Most of time, methods of $owl accept a generic RDFResource object as an argument.

hasValuesForProperty(OWLIndividual, String)

Test if the inidividual has some values for the given property name.

getResourceByName(String)

Return the resource in the default model ($kb), given its name.

getPropertyLiteral(OWLIndividual, String)

Returns, as a String, the first value of the given Datatype Property for the individual specified.

navigateProperty(OWLIndividual, String)

Returns a collection of resources which are the values for the Object Property, given by name, of the specified individual. Typically, you will use a #foreach loop to process them individually.

navigatePropertySorted(OWLIndividual,String,String)

Returns a collection like navigateProperty, but the values are sorted according to the literal value of the second property, specified by name.

navigateFunctional(OWLIndividual, String)

Returns the single value of a functional property.

getLabel(RDFResource)

Returns a string label for the resource that you provide as an argument. It works on individuals, classes, properties.

The label selection criteria is similar to the one used by Protégé:

- first try the default language defined in $owl;
- then try the default language of the Ontology;
- as a fallback, a label with no language defined.

formatDate(String)

Expects a date formatted according to XML (yyyy-mm-ddThh:mm:ss) and returns a string with the same date formatted according to the default locale.

Cookbook

This chapter provides some concrete examples ("pattern" is too ambitious) of using **velociowl** for simple tasks.

foreach loop in HTML documents

This is not Protégé-specific. The best way to include Velocity macros in an HTML document, is to embed them in a comment. In this way, you will still be able to edit the template with a standard WYSIWYG editor.

As an example, if you want to fill a table with the items in a collection:

```
<TABLE>
     <TR> --- headings --- </TR>
     <!-- #foreach($item in $collection) -->
     <TR>
          <TD>$item.field1</TD>
          <TD>$item.field2</TD>
          ...
     </TR>
     <!-- #end -->
</TABLE>
```

Notes

In this chapter, I will provide some further notes on this project.

The Open Source Power

I think that this plugin represents a clear example of the power of the open source approach.

Protégé is a sophisticated suite for knowledge engineering, the result of a distributed effort among several groups. Unfortunately, there is not a clear commercial market for this kind of tools (or, at least, the software industry is not really convinced about this, with few exceptions); a big part of the community around the project comes from the academic world.

Without an open source approach, and the fundamental economic support of the project financiers, we won't have seen Protégé. At the same time, this product is a sort of catalyst: the opportunity to "play" with ontologies, and ontology engineering, offered to a wide community of practitioners, is a fundamental instrument for the diffusion of the technology. The diffusion of the technology – if it is proven to be a good technology on the field – is a stimulus for further research and development, and this is a concrete return for the institution that financed the project.

Velocity is a simple (on the surface) tool, that gives the user the tremendous power of generating "anything from java", with a "develop by example" approach. Do you want to create an HTML page, an XMI file, a PDF, a graph? Create an example file, fill the variable contents with fantasy names, and turn the result to a programmer: "Please, create some objects to fill those variables”. Velocity is part of the bigger Jakarta project from Apache Foundation, and is widely used embedded in several applications.

The development language was the same, Protégé has a sophisticated and well-documented API, same for Velocity, it was really easy to put the two things together and obtain a reporting tool with a lot of flexibility. I don't know if you have tried to process an OWL file (or an OWL Ontology modularized across several files), in order to create something different; i did, mainly using XSLT, and I found it really hard!

Finally, producing reports directly from the Protégé platforms, allows you to operate on the *inferred* model (effectively using a third component, that is the *reasoner*); this is impossible – or challenging – if you work directly on the OWL file.

License