# VPOET Templates to Handle the Presentation of Semantic Data Sources in Wikis

Mariano Rico[1], David Camacho[1], Óscar Corcho[2]

[1] Computer Science Department, Universidad Autónoma de Madrid, Spain*
{mariano.rico, david.camacho}@uam.es
[2] Ontology Engineering Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Spain
ocorcho@fi.upm.es

**Abstract.** We describe VPOET templates, web templates to present or request semantic data to or from end users. These templates can be used by web developers with no competencies in Semantic Web by means of simple HTTP messages. Although its application to wiki environments is only pointed, a real application oriented to end users is shown. This experience shows that VPOET templates can be easily integrated in web applications written in any programming language.

## 1 Introduction

Besides semantic annotation [1], some general-purpose wikis also provide support to create semantic web applications [2,3]. VPOET [3] [4] is one of these semantic web application, aimed at web designers without knowledge in Semantic Web. It allows them to create web templates designed to display semantic data (output templates) or to request information from the user to convert it into semantic data (input templates).

All the information is stored as semantic data that can be recovered through simple HTTP messages, in a much simpler way than SPARQL endpoints (which require SPARQL language skills to create the HTTP message), aimed at lowering the adoption barrier for common web developers. An additional benefit of using HTTP messages, compared to traditional programming libraries, is that it can be exploited by developers in any programming language, expanding further the developers scope.

As a usage example for testing how a common developer can exploit the semantic information stored in VPOET, we have created a Google Gadget that allows users with minimal knowledge of Web technologies (end users) to incorporate semantic information in their web sites and in applications such as Google Pages, Google Desktop, etc. This example shows that VPOET templates can be

*This work has been partially funded under the projects HADA (TIN2007-64718), METEORIC (TIN2008-02081), and DEDICON (TIC-4425)

[3] See http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=VPOET

easily integrated into any web application. In a wiki framework, a simple command (in "edition mode") with few arguments, can be used to render (in "view mode") semantic data, allowing users browse through the semantic relations.

## 2 Related work

The way in which templates can help users to homogenize semantic data presentation has been addressed before. Semantic Media Wiki allows users creating templates, but employs an intricate syntax and parsing functions [4]. Therefore, it is not easy to handle by a web designer, usually with lower competences in programming languages.

A template infrastructure for semantic data is Fresnel [5], used by Longwell, a faceted semantic web browser. However the Fresnel syntax [5] requires skills in semantic web technologies that cannot be accomplished by most web designers.

Rhizomer [6] is an infrastructure to browse and edit semantic data. The presentation of RDF data is achieved by means of XSLT. The competences required to create Rhizomer templates are not aimed at web designers either.

Therefore, the current state of the art does not provide web designers with authoring tools to create attractive an reusable templates for semantic data. There must be a balance between expressiveness, to address RDF features (e.g. multi-valued properties, properties with no value) and complexity, in order to reduce the required level of competencies. VPOET provides web designers with OMEMO[6], another application that generates simplified versions of a given ontology, specifically oriented to web designers. By reading the information provided by OMEMO, web designers can know the sub-components of a given ontology component, requiring only basics of semantic web technologies such as class, property, value or relation. Details such as restrictions, inverse functional properties, etc. are hidden on purpose.

## 3 VPOET as a templates source

VPOET has two faces, on the one hand it is a web application oriented to web designers ranging from amateur users to professional ones. Following a 20 min. online tutorial [7] is enough to start creating templates embedding simple macros in the client-side code (HTML, CSS, or Javascript) generated by the web designer favorite authoring tool (e.g. Dreamweaver). These macros are detailed in the aforementioned tutorial. Figure 1 shows an output template for `FOAF:Person`. Macros are framed within a thick rectangle (`OmemoGetP`, `OmemoConditionalVizFor`, among others), and reused templates are framed

[4] A template source code and usage example can be found at `http://en.wikipedia.org/wiki/Template:Infobox_Settlement`

[5] See an example at `http://www.w3.org/2005/04/fresnel-info/manual/`

[6] See `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=OMEMO`

[7] See `http://ishtar.ii.uam.es/fortunata/Wiki.jsp?page=VPOETTutorial`

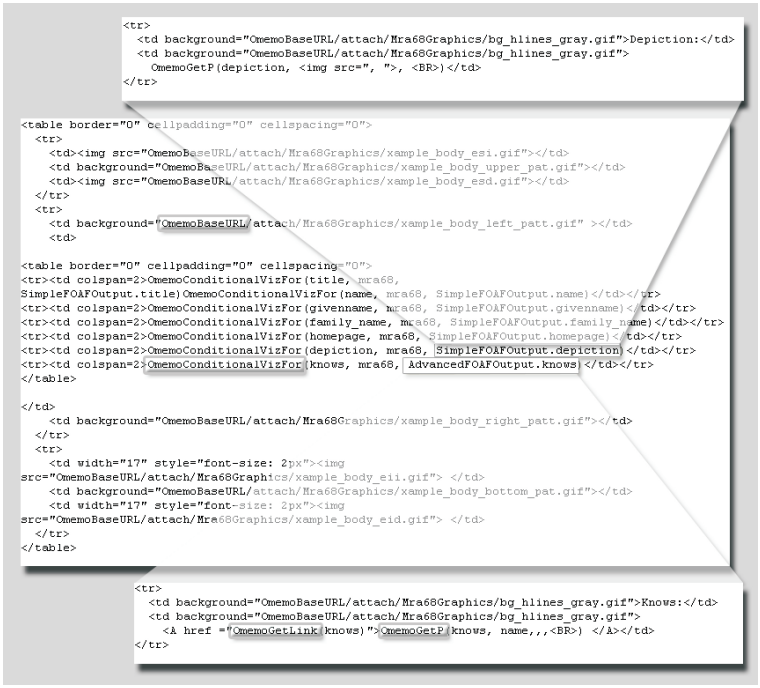


**Fig. 1.** Template example showing macros (thick rectangles) and templates reuse (thin rectangles).

within a thin stroke. It must be noticed that, unlike other templates systems, the code is essentially client-side code, substituting programming issues by simple macros, easier to understand by web designers. These macros allow a variable number of arguments and can be evaluated in design time, warning web designers when some of the arguments is wrong.

On the other hand, it is a semantic data source fed by the templates created by a community of web designers sharing and reusing templates. This source can be exploited easily by common web developers, in any programming language, by means of HTTP messages (GET and POST) like "render the semantic data at URL Z by using the output template X created by designer Y", codified as a HTTP GET message by means of the following URL:

```
http://URL-to-servlet/VPoetRequestServlet?action=renderOutput
&designID=X&provider=Y&source=Z.
```

An additional argument can specify a given individual in the source. In this case, only the individual is rendered. The full syntax of these HTTP messages can be found in the aforementioned tutorial.

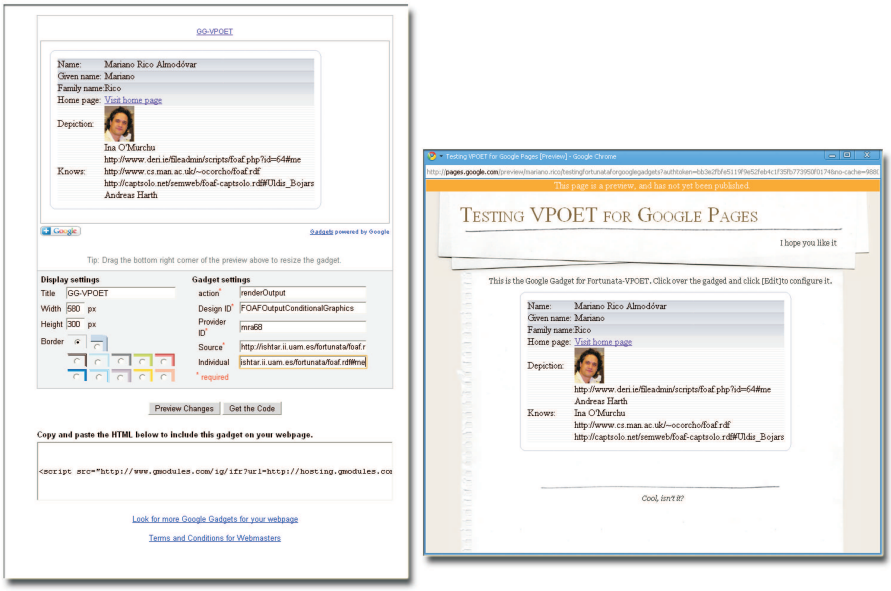


**Fig. 2.** Using GG-VPOET. Left: facilities to supply the HTML code that must be inserted by end users into a regular web page. Right: using this gadget in end user oriented application (Google Pages).

## 4 Using VPOET templates in wiki pages

Exploiting VPOET templates by means of the aforementioned HTTP messages is easy. As an usage example, a Google Gadget named GG-VPOET [8] has been created. By using this gadget, any end-user can render a semantic data source (by means of an output template) or provide a web interface to create semantic data (by means of an input template). GG-VPOET, as any other Google Gadget, can be inserted into a regular web page or in Google products such as iGoogle, Google Desktop or Google Pages. Figure 2 shows this gadget in action, rendering a given data source by means of the template shown in figure 1.

Following the same principles, the integration of VPOET templates in a given wiki framework (let us say JSPWiki) depends on the existence of a mechanism to add new tags in the wiki syntax and associate functionality to the tag. For example, JSPWIki implements these features by means of plugins, and getting the renderization shown in figure 2 (right side) is as easy as creating a JSPWiki plugin (let us name it VPOETPlugin) and use it like this in any JSPWiki page:

```
[{VPOETPlugin action='renderOutput' designID='X' provider='Y'
 source='Z' width='400' height='300'}]
```

[8] Available at the Google Gadgets Directory (look for GG-VPOET in `http://www.google.com/ig/directory`).

The implementation of `VPOETPlugin` reads the input parameters (`action`, `designID`, `provider` etc.), builds a HTTP message (a simple URL for HTTP GET messages like the one shown in section 3), and sends it to a running VPOET application (to a specific well-known VPOET servlet).

## 5 Conclusions and further work

The experience with GG-VPOET shows that the integration of VPOET templates into wikis could be easy in terms of wiki development effort and required competencies (basic skills in any programming language to create HTTP messages). This feature is specially remarkable in the wiki world in which wiki engines are written in most programming languages such as Java, Ruby, PHP or Perl. In any one of these languages it is easy to create and send a HTTP message. Additionally, wiki users benefit from professional web designs, providing attractive wiki pages which handle semantic data.

Our future work is to improve the features of VPOET in two directions: containers and user profiles. The former are intended to allow web designers create graphical containers (e.g. a tabs based agenda) to render multiple semantic individuals. The current implementation renders the individuals sequentially. The latter extends the HTTP messages to include a parameter pointing to a semantic description of the user profile. This profile can specify the user device (e.g. PC, handheld, TV) or the user interactive characteristics (e.g. color blindness, reduced visual sharpness). By providing this information, VPOET could return the "best"template for a given situation. An hypothetical wiki exploiting VPOET templates could request users to provide their user profile (URL) or provide an *ad hoc* user interface to create it and store it, providing wiki users with an interface adapted to their needs.

## References


1. Oren, E., Delbru, R., Moller, K., Volkel, M., Handschuh, S.: Annotation and Navigation in Semantic Wikis. In Proc. SemWiki. CEUR-WS vol. 206, (2006)
2. García, R., Gimeno, J.M., Perdrix, F., Gil, R., Oliva, M.: The Rhizomer Semantic Content Management System. In Proc. WSKS. LNAI Series, vol. 5288, pp. 387–394. Springer (2008)
3. Oren, E., Haller, A., Mesnage, C., Hauswirth, M., Heitmann, B., Decker, S.: A Flexible Integration Framework for Semantic Web 2.0 Applications. IEEE Software **24**(5), pp. 64–71, (Sept-Oct 2007)
4. Rico, M., Camacho, D., Óscar Corcho: VPOET: Using a Distributed Collaborative Platform for Semantic Web Applications. In Proc. IDC2008. SCI Series vol. 162, 167–176. Springer (2008)
5. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In Proc. ISWC. LNCS 4273, pp. 158–171. Springer (2006)
6. García, R., Gil, R.: Improving Human–Semantic Web Interaction: The Rhizomer Experience. In Proc. SWAP'06. CEUR-WS vol. 201, pp. 57–64, (2006)