Macros vs. scripting in VPOET *

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 present our experience on the provision and use of macros for the management of semantic data in semantically-enabled web applications. Macros can be considered as a lightweight version of scripting languages, mostly oriented to end users instead of to developers. We have enabled the use of macros in a wiki-based application named VPOET, oriented to web designers, and have confirmed through evaluations that macros satisfy a wide audience.

1 Introduction

VPOET [3] [1] is a semantic web application aimed at web designers (i.e. client-side web specialists) without any knowledge in Semantic Web. It allows them to create web templates to display semantic data (output templates) or to request information from users that is then converted into semantic data (input templates).

These templates are shared and reused by the web designers community, and can be used by third parties easily. Any developer can request VPOET templates by means of simple HTTP messages (GET and POST), created in any programming language, sent to a specific VPOET servlet. The response to these messages is a piece of client code (HTML, CSS, Javascript). A typical request is "render the semantic data at URL Z by using the output template X created by designer Y", which can be 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 could specify a specific individual in the data source. In this case, only the individual is rendered. This is a real example using semantic data provided by W3C:

*We would like to thank Roberto García for his collaboration concerning Rhizomic.

†This work has been partially funded by the Spanish Ministry of Science and Innovation under the project HADA (TIN2007-64718), METEORIC (TIN2008-02081), and DEDICON (TIC-4425)

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

http://ishtar.ii.uam.es/fortunata/servlet/VPoetRequestServlet?
action=renderOutput&
designID=FOAFOutputConditionalGraphics&
provider=mra68&
source=http:/www.w3.org/People/Berners-Lee/card&
indvID=http://www.w3.org/People/Berners-Lee/card#i

Unlike other templates systems based on scripting languages (shown in next section), VPOET provides web designers with simple macros to embed in the template source code. A 20 min. tutorial [4] is enough to start creating templates. The experimental evaluation [2] shows that web designers, in a wide skills range, from amateur to professionals, are satisfied with these macros. We argue that macros is a well known concept for web designers, while the current systems are oriented to developers.

2 Scripting Templates. A Comparative Use Case

This section describes briefly some representative semantic web applications that handle templates to present semantic data, such as semantic wikis, semantic browsers, and semantic portals. We have selected one representative application or infrastructure from each group: Semantic Media Wiki, Fresnel (used by the Longwell browser), and Rhizomer (used by the Rhizomik portal) respectively.

Semantic Media Wiki allows users create templates employing an *ad hoc* syntax with parsing functions [5]. The left part of table 2 shows the source code of a template [6], and the right part of the figure shows the renderization of semantic data by using this template. The creator of a template must know the wiki syntax, basics of programming, and basics of ontology components.

Fresnel [3] is a template infrastructure for semantic data, used by a faceted semantic web browser named Longwell. However, as table 3 (top) shows, the Fresnel syntax [7] requires skills in semantic web technologies that severely limit the number of designers available.

Rhizomer [4] is an infrastructure to browse and edit semantic data. The presentation of RDF data is achieved by means of Extensible Stylesheet Language Transformations (XSLT). As one can see in table 3 (middle), web designers require additional skills in XSLT programming.

Table 1 summarizes the features of these templates and the competencies required to the creators of templates for the semantic applications considered. The current state of the art is oriented to developers with competencies in both Semantic Web Technologies and Web Client technologies. Therefore, it 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, or properties with no value) and complexity, in order to reduce the required level of competencies.

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

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

[6] Edit page `http://semanticweb.org/wiki/Template:Person` to see the template source code between `<includeonly>` and `</includeonly>`. Note: the code previous to this block shows an usage example.

[7] See `http://www.w3.org/2005/04/fresnel-info/manual/`

	SMW	Fresnel	Rhizomik	VPOET
Programming Language	Wiki syntax, Programming, HTML/CSS	OWL, Fresnel ont., CSS	XSLT, OWL/XML	HTML, CSS, Javascript, macros
Allows				
Template reusing	No	Yes	Yes	Yes
Conditional rendering	Yes	Yes	Yes	Yes
Images	No	No	Yes	Yes
Dynamics (Web 2.0)	No	No	Yes	Yes
Oriented to	Average programmers	Semantic Web developers + CSS	Semantic Web developers + XSLT	Web designers + Macros
Competencies				
Sem. Web Techs.	Low	High	Very high	Low
Web Client Techs.	Low	High	Very high	Low-Very high
Other	Wiki syntax	High	XSLT	
Pros	Medium requirements	Pure OWL	Generic solution	Low requirements
Cons	Error prone syntax	Too complex for a web designer	Too complex for a web designer	

Table 1. Features of some frameworks to create a template

```
{| cellspacing="0" cellpadding="5" style="position:relative; margin: 0 0 0.5em 1em;
   border-collapse: collapse; border: 1px solid #aaa; background: #fff; float: right;
   clear: right; width: 20em"
! colspan="2" style="background: #86ba0c; color: white" |<span style="font-size: 80%;
                    float: right; ">{{#ask: [[{{{FULLPAGENAME}}}]]
| format=vcard
| ?Name
| ?Affiliation=organization
| ?Email
| ?Foaf:phone=workphone
| ?Homepage
| searchlabel=vCard
}}</span> [[Name::{{{Name|{{PAGENAME}}}}}]]
|-
{{#ifeq:{{{Picture|}}}|||{{Tablelongrow|Value=[[Image:{{{Picture
       }}}|150px|{{{Name|{{PAGENAME}}}}}]]|Color=white}}}}
|-
{{#ifeq:{{{Email|}}}|||{{Tablelongrow|Value={{Mailbox|{{{Email
       }}}}}|Color=#e4f8b6}}}}
|-
{{#ifeq:{{{Affiliation|}}}|||{{Tablerow|Label=Affiliation:|
   Value=[[member of::affiliation::{{{Affiliation}}}]]}}}}
|-
{{#ifeq:{{{Homepage|}}}|||{{Tablerow|Label=Homepage:|
   Value=[[homepage::http://{{{Homepage}}}|{{{Homepage label|
   {{{Homepage}}}}}}]]}}}}
|-
{{#ifeq:{{{Phone|}}}|||{{Tablerow|Label=Phone:
   |Value=[[foaf:phone::{{{Phone}}}]]}}}}
|-
<!--  *** Events ***  -->
{{Tablelongrow|Align=Left|Font size=80%|
  Value={{#ask:[[has PC member::{{PAGENAME}}]]
| format=list
| sort=start date | order=desc
| sep=,_
| intro=PC member of:_ }}|Color=#e4f8b6}}
|-
{{Tablelongrow|Align=Left|Font size=80%|
  Value={{#ask:[[has OC member::{{PAGENAME}}]]
| format=list
| sort=start date | order=desc
| sep=,_
| intro=OC member of:_ }}|Color=#e4f8b6}}
|-
{{#ifeq:{{{FOAF|}}}|||{{Tablerow|Label=See also:|
Value=[[see also::{{{FOAF}}}|FOAF]]}}}}|Color=white}}
|}[[Category:Person]]
```

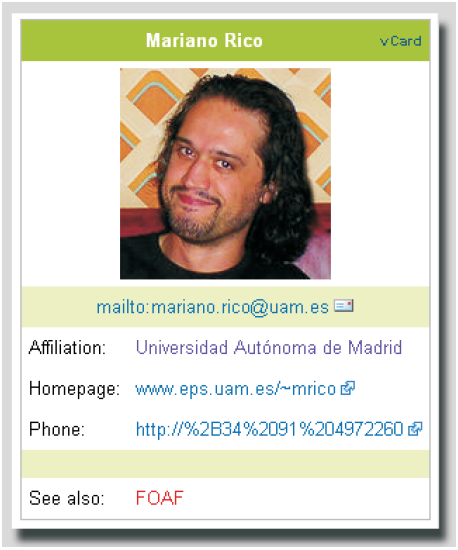
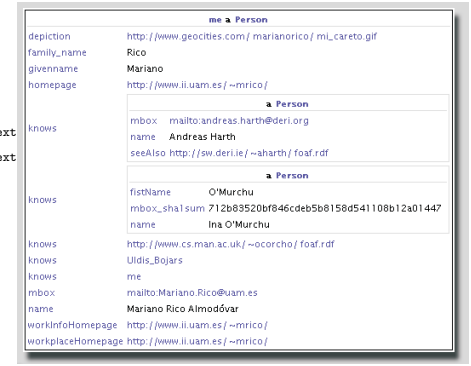


Table 2. Template in Semantic Media Wiki. Left: template code. Right: rendering an example.

```
:foafGroup rdf:type fresnel:Group ;
     fresnel:stylesheetLink <http://www.example.org/example.css> ;
     fresnel:containerStyle "background-color: white;"
                            ^^fresnel:stylingInstructions ;

:foafPersonFormat rdf:type fresnel:Format ;
     fresnel:classFormatDomain foaf:Person ;
     fresnel:resourceStyle "background-color: gray;"
                           ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .

:nameFormat rdf:type fresnel:Format ;
     fresnel:propertyFormatDomain foaf:name ;
     fresnel:propertyStyle "border-top: solid black;"
                           ^^fresnel:stylingInstructions ;
     fresnel:labelStyle "font-weight: bold;"
                        ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .

:urlFormat rdf:type fresnel:Format ;
     fresnel:propertyFormatDomain foaf:homepage ;
     fresnel:propertyFormatDomain foaf:mbox ;
     fresnel:value fresnel:externalLink ;
     fresnel:propertyStyle "border-top: solid black;"
                           ^^fresnel:stylingInstructions ;
     fresnel:labelStyle "font-weight: bold;"
                        ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .

:depictFormat rdf:type fresnel:Format ;
     fresnel:propertyFormatDomain foaf:depiction ;
     fresnel:label fresnel:none ;
     fresnel:value fresnel:image ;
     fresnel:propertyStyle "border-top: solid black;"
                           ^^fresnel:stylingInstructions ;
     fresnel:group :foafGroup .
```


```
<?xml version="1.0"?>
<?xml-stylesheet href="./xsltdoc.xsl" type="text/xsl" media="screen"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <xsl:import href="rdf2html-functions.xsl"/>
  <xsl:param name="language">en</xsl:param>
  <xsl:output media-type="text/xhtml" version="1.0" encoding="UTF-8"
              indent="yes"/>
  <xsl:strip-space elements="*"/>
  <xsl:template match="/">
     <xsl:apply-templates select="rdf:RDF"/>
  </xsl:template>
  <xsl:template match="rdf:RDF">
    <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8"/>
      <title>Rhizomik - ReDeFer - RDF2HTML</title>
      <link href="http://rhizomik.net/style/rhizomik.css" type="text
            rel="stylesheet" />
      <link href="http://rhizomik.net/style/rhizomer.css" type="text
            rel="stylesheet" />
    </head>
    <body>
      <!-- div id="browser">
        <a href="javascript:history.back()">back</a> - go to... -
        <a href="javascript:history.forward()">forward</a>
      </div -->
      <xsl:if test="count(child::*)=0">
        <p>No data retrieved.</p>
      </xsl:if>
      <xsl:for-each select="child::*">
        <xsl:sort select="@rdf:about" order="ascending"/>
        <xsl:call-template name="rdfDescription"/>
      </xsl:for-each>
... 224 lines more ...
</xsl:stylesheet>
```




```
<table border="0" cellpadding="0" cellspacing="0">
<tr>
 <td><img src="OmemoBaseURL/attach/Mra68Graphics/xample_body_esi.gif"></td>
 <td background="OmemoBaseURL/attach/Mra68Graphics/xample_body_upper_pat.gif"></td>
 <td><img src="OmemoBaseURL/attach/Mra68Graphics/xample_body_esd.gif">
 </td>
</tr>
<tr>
 <td background="OmemoBaseURL/attach/Mra68Graphics/
                 xample_body_left_patt.gif"></td>
<td>
<table border="0" cellpadding="0" cellspacing="0">
<tr><td colspan=2>OmemoConditionalVizFor(title, mra68,
   SimpleFOAFOutput.title)OmemoConditionalVizFor(name, mra68,
   SimpleFOAFOutput.name)</td></tr>
<tr><td colspan=2>OmemoConditionalVizFor(givenname, mra68,
   SimpleFOAFOutput.givenname)</td></tr>
<tr><td colspan=2>OmemoConditionalVizFor(family_name, mra68,
   SimpleFOAFOutput.family_name)</td></tr>
<tr><td colspan=2>OmemoConditionalVizFor(homepage, mra68,
   SimpleFOAFOutput.homepage)</td></tr>
<tr><td colspan=2>OmemoConditionalVizFor(depiction, mra68,
   SimpleFOAFOutput.depiction)</td></tr>
<tr><td colspan=2>OmemoConditionalVizFor(knows, mra68,
   SimpleFOAFOutput.knows)</td></tr>
</table>
</td>
    <td background="OmemoBaseURL/attach/Mra68Graphics/
                    xample_body_right_patt.gif"></td>
  </tr>
  <tr>
    <td width="17" style="font-size: 2px"><img src="OmemoBaseURL/
          attach/Mra68Graphics/xample_body_eii.gif"> </td>
    <td background="OmemoBaseURL/attach/Mra68Graphics/
          xample_body_bottom_pat.gif"></td>
    <td width="17" style="font-size: 2px"><img src="OmemoBaseURL/
          attach/Mra68Graphics/xample_body_eid.gif"> </td>
  </tr>
</table>
```

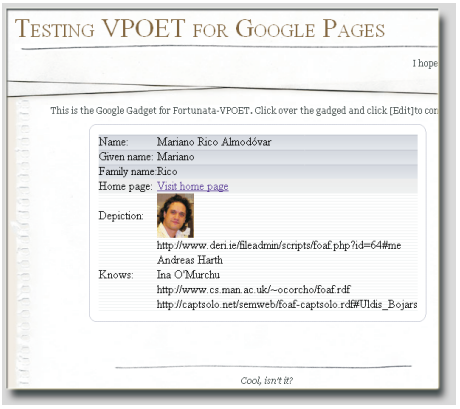


Table 3. Template examples: Fresnel (top), Rhizomic (middle) and VPOET (bottom).

Macro	Arguments	Explanation
`OmemoGetP`	`propName`	It is replaced by the property value `propName`
`OmemoBaseURL`	No arguments	It is replaced by the URL of the server where VPOET is running
`OmemoConditionalVizFor`	`propName`, `designerID`, `designID`	It renders the property `propName` only if it has a value, using the template indicated by (`designerID`, `designID`)
`OmemoGetLink`	`relationName`, `designerID`, `designID`	It is replaced by a link capable of displaying components of the type pointed by the relation `relationName` using the template indicated by (`designerID`, `designID`)

Table 4. Main macros available for web designers in VPOET.

VPOET reduces the requirements needed to create templates, lowering the adoption barrier for web designers. VPOET "speaks"the web designers language, i.e. client side languages such as HTML, CSS and javascript, not requiring competencies in semantic web technologies or additional programming languages. Web designers use simple macros (see table 4) embedded in the source code, with the additional benefit of being capable of detecting errors in the macros in "development time"(checks in every modification of the template) and in "runtime"(checks in every template usage). The information about the structural components of a given ontology component are provided to web designers by OMEMO[8], 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.

Table 3 (bottom) shows an example of VPOET template source code (left part), and the rendering of that code inside a Google Page by using VPOET templates in a Google Gadget named GG-VPOET [9]. In this example one can see the macro `OmemoConditionalVizFor`, which renders a given property only if the property has a value. The rendering is transfered to a specified template, which uses the macro `OmemoGetP` to render the property value. As properties use to be multi-valuated, a mechanism based in PPS (pre-condition, post-condition, and sep-arator) has been added to the macro `OmemoGetP`. For example, let us assume a semantic data source with individuals having the property `FOAF:depiction` with values $\{url_1, url_2, ..., url_N\}$. If a web designer wants to create a VPOET template, like the one shown in bottom of table 3, which renders multiple values of this property with HTMLcode like this:

```
<img src="url_1"><BR>
....
<img src="url_N">
```

she can use the conditional rendering provided by `OmemoConditionalVizFor` typing:

```
OmemoConditionalVizFor(urlprop, mra68, SimpleFOAFOutput.depiction)
```

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

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

which renders the property by means of the template `SimpleFOAFOutput.depiction` (created by designer `mra68`). The code of the template uses `OmemoGetP` in its 3 arguments flavor `OmemoGetP(pre, post, sep)` to code the template like this:
`OmemoGetP(<img src=",">,<BR>)`

Another common requirement are "nested properties", e.g. displaying only `FOAF:name` for `FOAF:knows` objects. This can be achieved by means of the macro `OmemoGetP` in its 5 arguments flavor `OmemoGetP(relation, propPref, pre, post, sep`. A simple (multi-values separated by `<BR>`) codification could be:
`OmemoGetP(knows, name„,<BR>)`

But, if additionally you want to associate a link to the value, you have to use the macro `OmemoGetLink`. The final template code would be like this:
`<A href ="OmemoGetLink(knows)">OmemoGetP(knows, name„,<BR>) </A>`
This macro is replaced in runtime by a link capable of displaying ontology components of the type pointed by the relation `knows`, i.e. a `FOAF:Person`. In the current implementation, clicking the link would "change to a new page", but it can be modified to place the client code in a new DOM object in the same page by using AJAX.

3 Conclusions and further work

Our experience in VPOET shows that, in the context of templates creation, macros provides web designers with a friendly environment, requiring lower competencies than other equivalent frameworks such as Semantic Media Wiki, Rhizomer, or Fresnel. An additional advantage of VPOET templates is that these templates can be used easily by other developers skilled in any programming language due to the simplicity of the HTTP messages involved.

An initial small set of macros were obtained from the analysis of the semantic templates in Semantic Media Wiki. The experiments [2] carried out with fifteen real users confirmed that a that small number of macros satisfied the needs of most web designers. These users suggested, by filling detailed questionnaires, new macros or additional parameters for the existing ones.

Our current work is focused on providing web designers with new features such as creation of containers to handle individuals sets in a more attractive way. The current implementation lists individuals in a simple sequence. Probably, this will require new macros or modifications to the existing ones.

References

1. Rico, M., Camacho, D., Corcho, Ó.: VPOET: Using a Distributed Collaborative Platform for Semantic Web Applications. In proc. IDC2008. SCI 162, pp. 167–176. Springer (2008)
2. Rico, M., Macías, J.A., Camacho, D., Corcho, Ó.: Enabling Web Designers to Handle Semantic Data in Web Applications. Submitted to Journal of Web Semantics (2009)
3. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In proc. ISWC. LNCS 4273, pp. 158Ű-171. Springer (2006)
4. García, R., Gil, R.: Improving Human–Semantic Web Interaction: The Rhizomer Experience. In proc. SWAP'06. CEUR-WS 201, pp. 57–64. (2006)