Uduvudu Editor: Creating Mashups on top of the Semantic Web

Michael Luggen[1,2]

[1] University of Fribourg, eXascale Infolab, 1700 Fribourg—Switzerland,
[2] Bern University of Applied Sciences, Division of Computer Science,
2500 Biel—Switzerland

Abstract. Highly relevant information in diverse fields of our life become available online, mostly originating from different organizations and thus also different locations. The need to combine, split and generally reorganize this information for secondary and personal use cases naturally evolve with the emerging data. Information publishers aware of this need, enhance their human optimized representation (HTML) with a standardized data model like RDF. As the fundament of the semantic web, RDF provides machine readable formats such as JSON-LD, Turtle or alike. We propose the *Uduvudu Editor* which is build on top of the visualization engine Uduvudu. This powerfull tool allows users with basic HTML skills to create, in a fast and intuitive way, complex visualization on top of the semantic web. The *Uduvudu Editor* enables through simple *combining and ordering* to layout any kind of information regardless of their inherent structure. With the *Uduvudu Editor* we present a novel interface metaphor which allows the creation of templates for the Uduvudu Framework. We show that the creation of unlimited, robust mashups based on top of the semantic web is possible for everybody, even without knowledge of the underlying mechanics.

1 Introduction

To browse the Semantic Web we need to render and visualize the provided Linked Data in a human friendly form. There is a plethora of frameworks and tools (see Table 1) focusing with different ease of use on this task. Often, however, Linked Data is transformed into less expressive data formats that are then rendered in a simplistic fashion in the interface. The exported Linked Data typically looses some of its structure, its semantics and/or some of its links. Handling arbitrary Linked Data in order to render it properly in a UI is a singularly tedious task.

Our framework, called Uduvudu, considerably speeds up the development of Linked Data UIs by streamlining the data extraction, transformation, and rendering process through an explicit workflow. It allows several types of experts including data specialists, user interface experts, and graphical designers, to efficiently collaborate to produce the UI. The framework is furthermore flexible and robust, in the sense that it can render any valid RDF data irrespective of its schema or missing values, and produces reusable UIs and patterns that can

	Uduvudu[5]	Callimachus[2]	Balloon Syn.[7]	Fresnel[6]	Exhibit[4]	LESS[1]
Level of Template	Subgraph	Application	JS Selector	Subgraph	Subgraph	Projection
Description of T.	underscore.js	RDFa Templates	Handlebars	Fresnel Formats	Exhibit Lenses	Smarty
Recursive Use of T.	Y	N	N	Y	N	N
Context Awareness	Y	N	N	N	N	N
Separation of Concerns	Y	N	Y	Y	N	Y
Editor	Y	Y	N	N	N	Y

Table 1. Comparison of frameworks on their template capabilities.

be applied to different contexts and data. To support this process, we introduce Uduvudu, an architecture dedicated to Linked Data.

In contrast most of the related systems focus on the exploration and visualization of Linked Data. The focus of many systems is hence either on how to effectively explore large quantities of Linked Data, or on how to compactly aggregate large data to visualize it. Less attention has been given to the case where a data publisher already knows exactly which parts of the data he/she wants to visualize. Table 1 summarizes how Uduvudu's template capabilities compare to existing systems. As the table indicates, Uduvudu is, to the best of our knowledge, the only framework allowing the flexible description and recursive use of context-aware templates to visualize Linked Data.

2 Uduvudu

In this section we give an summary of Uduvudu[3][5]. This allows us in the last section to sketch the *Uduvudu Editor* itself.

There are three main components in the Uduvudu architecture. The *Data Selector*, the *Structure Matcher* and finally the *Adaptive Renderer*. Figure 1 gives an overview of our architecture.

Data Selector First, we start with the Data Selector, which decides which data that will be transformed and rendered.

Any selector which supports classical Linked Data input, e.g., through a SPARQL query, an RDF/XML dump or triples serialized in one or several text files can be used. Uduvudu does not expect any inherent structure from the data (though it must be well-formed), which does not need to comply to any specific ontology or structure at this stage. Hence, the selector takes as input a superset of all informations that need to be shown to the end-user, and trim them to an input graph G containing exactly the information that needs to be rendered. This step can be facilitated by specialized interfaces targeting again non-expert users which are not part of this work. [3]

Structure Matcher The second component in our architecture, the Structure Matcher, holds a catalogue M of known structures (matchers) and tries to match parts of this catalogue onto the input graph G. The matcher takes as input an input graph G and one or several corresponding known structures from

[3] find more detailed information and the source code of Uduvudu at *http://uduvudu.org*

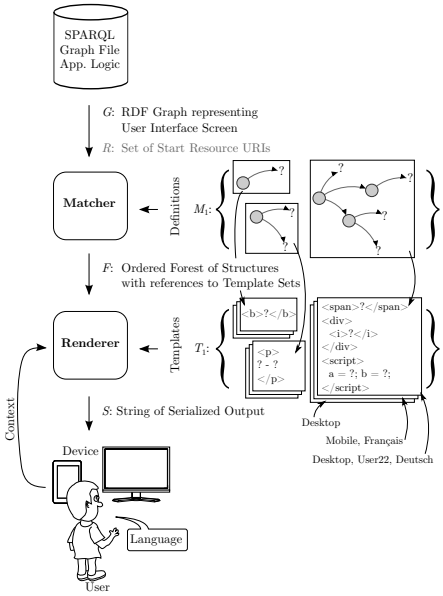

Fig. 1. Overview of the Uduvudu architecture with the main components *Matcher* and *Renderer*.

its catalogue and returns as output a tree structure with at least one pointer to a rendering structure from the Renderer. All input data is eventually consumed through this process.

Adaptive Renderer The adaptive renderer takes as input the tree structure given by the matcher (F), the available context variables (language, user, device) and the provided templates (T) to finally render the output. The templates are written in HTML and access the tree structure through escaped variable definitions (see Table 2 for details).

Usage	Description	Example
Delimiters		
`<%- %>`	Output variable HTML-escaped.	`<%- label.u %>`
`<%= %>`	Output variable non-escaped.	`<%= html_desc.u %>`
`<%  %>`	Execute JavaScript: Use print() for output.	`<%  print(label.u.toUpperCase()) %>`
Variables		
`label.u`	Literal in context language.	`<%- label.u %>`
`label.l.en`	Literal in specific language (lang tag).	`<%- label.u.ja %>`
`city.label.u`	Literal deeper in matched structure.	`<%- city.label.u %>`
`template(city.label)`	Rendered template for sub element.	`<%- template(city.label) %>`

Table 2. Overview of the most important template commands available.

3 Uduvudu Editor

The Uduvudu Editor is a simple yet intuitive and powerful editor which helps to create the necessary Matchers and Templates described in the preceding section in an interactive fashion. As it is possible to render an output at any time, the editor is simply based on the basic result of Uduvudu incorporating the available fallback templates.

3.1 Combine and Order

Matchers and their corresponding templates can be added iteratively to create templates for bigger structures through a *combine and order* metaphor.

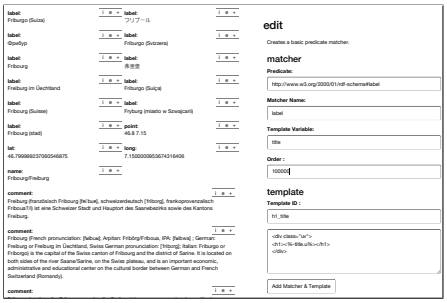

Fig. 2.
Rendering *http://dbpedia.org/page/Fribourg* with no templates defined (left), in edit mode with *rdf:label* staged (right).

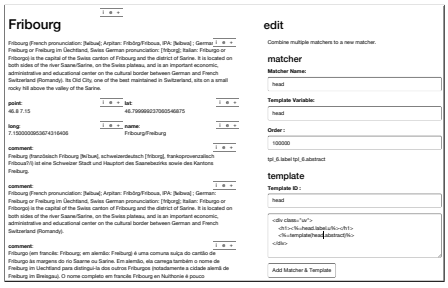

Fig. 3. The same interface as in Figure 2 after the definition of a matcher and a template respectively for *rdf:label* and *rdf:abstract*. Further a new combine matcher merging the mentioned is staged.

Figure 2 shows on the left-hand side one resource rendered using the simple fallback template rendering. On the right-hand side, a first predicate matcher (*rdf:label*) is already staged; by simply clicking on the + icon (shown in edit mode only), the template was changed to render the literal as an HTML title.

Through the same mechanism (using the + icon), templates can be *combined* together to create more expressive templates. This is shown in Figure 3, where the *rdf:label* gets combined with a pre-existing *rdf:abstract* template to form a new *head* template. In the staged example, the template of the abstract is reused (`<%-template(head.abstract)%>`), while the variable of the *rdf:label* is accessed directly (`<%-head.label.u%>`). This further allows to *order* the appearance of the combined facts to the users liking.

3.2 Possible extensions

A more advanced editor allows users to define multiple templates for different contexts like different devices or languages. The matchers and templates created

in the editor can be persisted in a public or private triple store. It is also possible to mix-in multiple matchers and template sources, which allows distinct users to create or adapt the templates to their own needs.

4 Conclusion

We sketched the *Uduvudu Editor* which provides the power of Uduvudu to the general public. Through the simple *combine and order* interface metaphor the user is able to create complex reusable visualizations without the need to understand the underlying mechanisms. All this enables the Uduvudu Editor to create fast and also robust mashups directly on top of the semantic web.

References

1. Sören Auer, Raphael Doehring, and Sebastian Dietzold. LESS - template-based syndication and presentation of linked data. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications*, number 6089 in Lecture Notes in Computer Science, pages 211–224. Springer Berlin Heidelberg, January 2010.
2. Steve Battle, David Wood, James Leigh, and Luke Ruth. The Callimachus Project: RDFa as a Web Template Language. In *COLD*, 2012.
3. Paolo Bottoni and Miguel Ceriani. Sparql playground: a block programming tool to experiment with sparql.
4. David F. Huynh, David R. Karger, and Robert C. Miller. Exhibit: Lightweight structured data publishing. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 737–746, New York, NY, USA, 2007. ACM.
5. Michael Luggen, Adrian Gschwend, Bernhard Anrig, and Philippe Cudré-Mauroux. Uduvudu: a graph-aware and adaptive UI engine for linked data. In *Proceedings of the Workshop on Linked Data on the Web, LDOW 2015, co-located with the 24th International World Wide Web Conference (WWW 2015), Florence, Italy, May 19th, 2015.*, 2015. http://www.slideshare.net/eXascaleInfolab/ldow2015-uduvudu-a-graphaware-and-adaptive-ui-engine-for-linked-data.
6. Emmanuel Pietriga, Christian Bizer, David Karger, and Ryan Lee. Fresnel: A browser-independent presentation vocabulary for RDF. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora M. Aroyo, editors, *The Semantic Web - ISWC 2006*, number 4273 in Lecture Notes in Computer Science, pages 158–171. Springer Berlin Heidelberg, January 2006.
7. Kai Schlegel, Thomas Weißgerber, Florian Stegmaier, Michael Granitzer, and Harald Kosch. Balloon Synopsis: A jQuery plugin to easily integrate the Semantic Web in a website. *CEUR Workshop Proceedings*, 1268, October 2014.