WikiNext, a JavaScript Semantic Wiki

Pavel Arapov, Michel Buffa
KEWI/Wimmics Group, I3S Laboratory, University of Nice, France.
{arapov, buffa}@i3s.unice.fr

ABSTRACT

In this position paper we present WikiNext[1], a semantic wiki we have been developing for eight months, written in JavaScript, from database to client code. It uses the HTTP/WebSocket NodeJS server, several frameworks such as NowJS for distributing JavaScript objects between server and client code or MongoDB for persistence. WikiNext proposes a new approach to deal with classical problems like data storage and representation (both for semantic data and CMS data), working with semantics, including and developing small applications within the wiki, sharing objects between client code running in the browser and server code, mixing HTTP asynchronous communication means with synchronous ones like web sockets, exploit original HTML5 features and finally use an event based programmatic style on the server side with an dedicated micro HTTP server.

Categories and Subject Descriptors

K.4.3 [**Organizational Impacts**]: Computer-supported collaborative work.

General Terms

Management, Measurement,

Keywords

Wiki, Semantic Web, Social Tagging, Ontology, Web 2.0.

1. Introduction

In the past many semantic wikis have been initiated (see [3] for an overview), most popular ones are Semantic Media Wiki [8] or IkeWiki/Kiwi [6], [7]. Our research group developed SweetWiki [2] that was from the first generation of semantic wikis (2005-2008), written from scratch in Java, and since 2008 SweetDeki [4], based on an open source industrial wiki engine named Mindtouch Core. Most semantic wiki engines had to deal with storage and handling of both semantic data and classic wiki data together. With WikiNext, we started again from scratch, trying to take into account emergent technologies and tools that appeared recently in the web development landscape, including HTML5 and micro web servers (lightweight web servers dedicated to a single application). (1) On the client side, HTML5 proposes new tags for creating web pages, but it also comes with many new JavaScript APIs that increases the momentum already existing around this language. For example, APIs like WebSockets[2] for synchronous communication between web browsers and servers, are very appealing for implementing some features in a collaborative edition platform, such as notifications or collaborative synchronous edition. Around HTML5 and JavaScript, interesting applications appeared like IDEs written in JavaScript, enabling development and testing of JavaScript code directly in the browser, like jsbin.com, jsfiddle.net or Cloud9IDE.com, this latter enabling the development of JavaScript code both for the client side but also for the server side. (2) Another trend is server side JavaScript; indeed, the CommonJS[3] specification allows developers to create different applications that run in JavaScript interpreters like the V8 engine (from Google) or SpiderMonkey/TraceMonkey (from Mozilla). For example, the NodeJS HTTP server is written in Python but embeds the V8 interpreter for developing server side code in JavaScript. (3) The JavaScript Object Notation (JSON[4]) also became very popular and slowly outperforms XML as the format of choice used with RESTful web services. Another example is the document-oriented database MongoDB[5] that uses the JSON format for storing data and uses the SpiderMonkey JavaScript interpreter for handling requests.

In this position paper we present WikiNext, a semantic wiki currently in early stages of development, written in JavaScript, which runs on the NodeJS Server and uses the MongoDB database for storing its data, using JavaScript and JSON in all the different layers of its architecture.

In the next section we will present the main design principles of WikiNext and we will show that our approach reduces the need to rewrite data in different formats (i.e. from SQL to objects on the server side, from objects to HTML, XML or JSON for the server-client exchange), a recurrent problem in classic design of web applications. We will also give some details on how we integrated an IDE into the wiki, turning it into an application wiki: users will be able to write documents but also small applets in JavaScript, directly from the wiki itself. This is quite interesting for automatic annotation or for integrating dynamic data from different data sources into a document. In section 3 we will present WikiNext's architecture.

2. WikiNext design principles

WikiNext is both (1) *a semantic wiki* (i.e. a wiki that uses technologies from the semantic Web[6] to embed formalized knowledge, content, structures and links, that can reason on this knowledge, etc.) and (2) *an application wiki* as it can be used both for writing documents but also for writing small

[1] Demo of prototype is available at http://wikinext.herokuapp.com/

[2] Even if not officially part of the HTML5 standard, see http://en.wikipedia.org/wiki/WebSocket.

[3] http://www.commonjs.org/

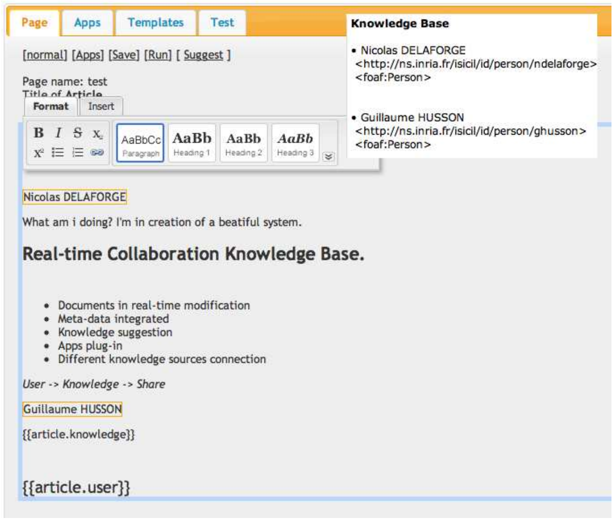
[4] http://www.json.org

[5] http://www.mongodb.org

[6] http://www.w3.org/2001/sw/

applications within the documents. The wiki exposes a JavaScript API for performing the main tasks from these applications, like manipulating the wiki data themselves: i.e. perform automatic annotation of certain keywords in the current wiki page, auto-tag a document, insert dynamically details about one user, etc.

For WYSIWYG document editing we use the Aloha in-place editor[7], a recent HTML5 based editor that produces clean xHTML5 code, as shown in Figure 1, and for writing embedded applets we rely on the Ace Cloud9 Editor[8] (see Figure 2).



Knowledge Base

- Nicolas DELAFORGE <http://ns.inria.fr/isicil/id/person/ndelaforge> <foaf:Person>
- Guillaume HUSSON <http://ns.inria.fr/isicil/id/person/ghusson> <foaf:Person>

Page name: test
Title of Article

Nicolas DELAFORGE

What am i doing? I'm in creation of a beatiful system.

Real-time Collaboration Knowledge Base.

- Documents in real-time modification
- Meta-data integrated
- Knowledge suggestion
- Apps plug-in
- Different knowledge sources connection

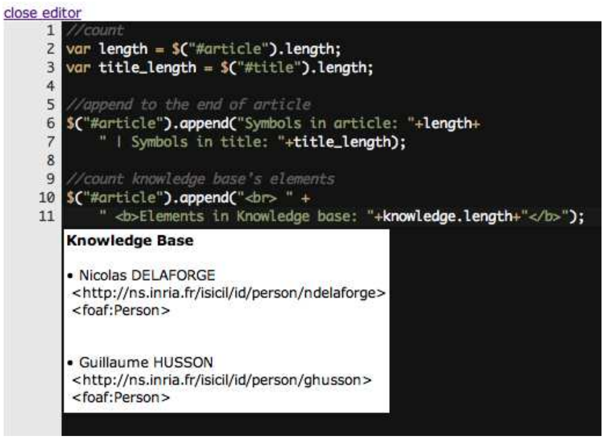
User -> Knowledge -> Share

Guillaume HUSSON

{{article.knowledge}}

{{article.user}}

Figure 1 : A view of the in place editor.



```
//count
var length = $("#article").length;
var title_length = $("#title").length;

//append to the end of article
$("#article").append("Symbols in article: "+length+
     " | Symbols in title: "+title_length);

//count knowledge base's elements
$("#article").append("<br> " +
     " <b>Elements in Knowledge base: "+knowledge.length+"</b>");
```

Knowledge Base

- Nicolas DELAFORGE <http://ns.inria.fr/isicil/id/person/ndelaforge> <foaf:Person>
- Guillaume HUSSON <http://ns.inria.fr/isicil/id/person/ghusson> <foaf:Person>

Figure 2: The JavaScript editor integrated in the wiki.

We use also the Mustache template engine for a clean separation of the content of a page and its decoration (menus, header, and footer). When a page is requested, the server computes the XHTML page and sends it to the browser. Each page contains a default script that parses the page and builds a JSON representation of the RDFa metadata present in the page. For this task we rely on the VIE framework[9]. Having important data available in JSON format makes them easier to handle by JavaScript code.

Furthermore, the default script opens a web socket connection with the server for synchronizing the page content. Having such a synchronous communication stream open makes it easier to synchronize the page content if other pages share the same data/metadata, or better, if an application embedded in one page manipulates global data in the wiki, other pages that share this data will be updated without the need to be reloaded in the browser or without the need to use Ajax-based pulling techniques. One may think that keeping a large number of open sockets simultaneously (one for each page opened) may prevent the system from scaling; this subject has been discussed a lot[10] but it makes sense to use web sockets for most new interactive web applications that need to communicate frequently with the server, as this protocol dramatically reduces the HTTP verbose protocol (no headers to parse), as well as the establishment and closure of connections as we would do with Ajax interaction.

This makes also easier to track and notify the activity of one user and opens the way to synchronous edition by multiple users (we are currently investigating the ShareJS Operational Transform library that allows Google-Wave like synchronization).

For modeling wiki resources we reuse the ontology we developed for SweetDeki (Figure 3), the semantic wiki of the ISICIL ANR project[11]. In pink we have the SweetDeki ontology that models the pages (class `sd:WikiPage`), the documents that can be attached (class `sd:attachment`, relation `sd:attachedTo`) or included in a page, revisions (class `sd:WikiPageRevision`), and the actions of users (top left `sd:WikiAction`), such as collaborative actions. Note that the classes of the SweetDeki ontology extend or reuse classes from the popular SIOC[12] ontology that models user activity.

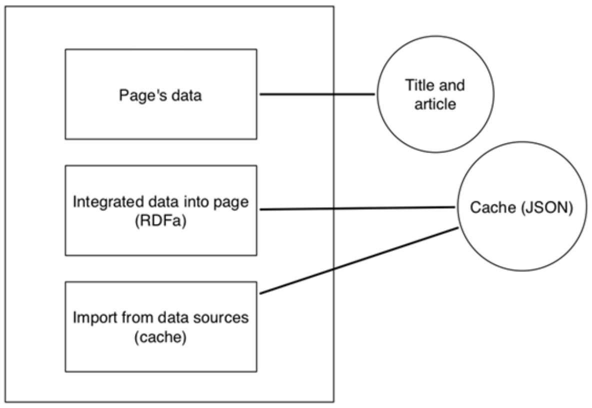

Page's data
Title and article
Integrated data into page (RDFa)
Cache (JSON)
Import from data sources (cache)

Figure 3: data structure of a wiki page.

[7] http://www.aloha-editor.org

[8] ACE : Ace Cloud9 Editor, http://ace.ajax.org/

[9] Vienna IKS Editable, http://www.slideshare.net/bergie/vie-using-rdfa-to-make-content-editable

[10] See : http://peterlubbers.sys-con.com/node/1315473

[11] http://isicil.inria.fr

[12] http://sioc-project.org/ontology

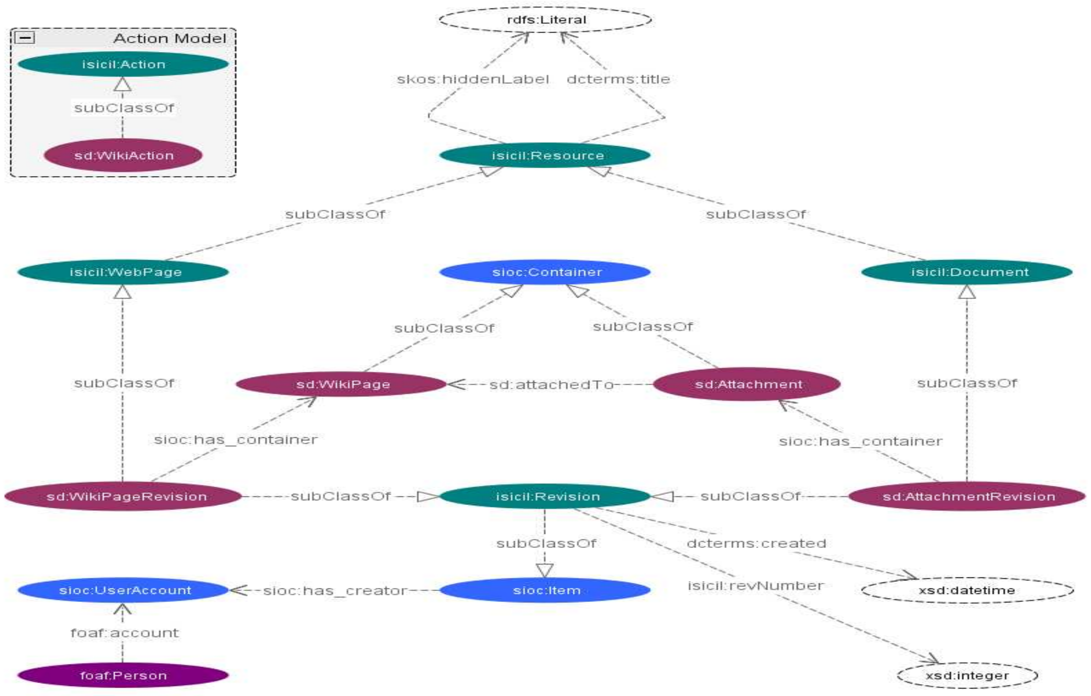


Figure 4: an extract of the ontology used by WikiNext for modeling documents.

Figure 3 shows data that compose a typical wiki page and that are persisted into the MongoDB database as JSON objects. From top to bottom we have (a) the standard XHTML of the page i.e. the page content, (b) RDFa metadata like details about the author, geo-localization data, tags, etc., and (c) data generated by applications embedded in the page, i.e. a list of countries resulting from a request or a web services call to DBPedia.org. In that case we will persist as a cache the data so that they can be searched. This cache is re-written each time we execute the application.

3. WikiNext's architecture

WikiNext runs on the NodeJS web server with several extension modules like the Express framework for implementing RESTful Web Services or socket.io[13] for the web sockets server implementation and NowJS for distributing JavaScript objects between server and client code over web sockets. We implemented a classical MVC framework for handling HTTP requests (GET and POST) but most operations can be done as well through the web socket connector. The persistence layer uses MongoDB, a schema-free document-oriented storage, for its flexibility and its native support of JSON objects[14]. Choosing MongoDB allowed us to store wiki data in a natural way, without need to rewrite formats, as we are handling JSON objects in all the process chain.

Figure 5 shows a diagram of WikiNext's server side architecture. Notice that we store both page content (HTML) and metadata (RDF/RDFa triplets) in JSON.

Currently, we are developing a "knowledge controller" in order to plug into our system the KGRAM reasoning engine[5] that will handle in sync with MongoDB semantic metadata and provide SPARQL capabilities. In that case, MongoDB will act as a cache for these metadata: basic metadata access will be handled directly by MongoDB while complex requests will be delegated to the reasoning engine.

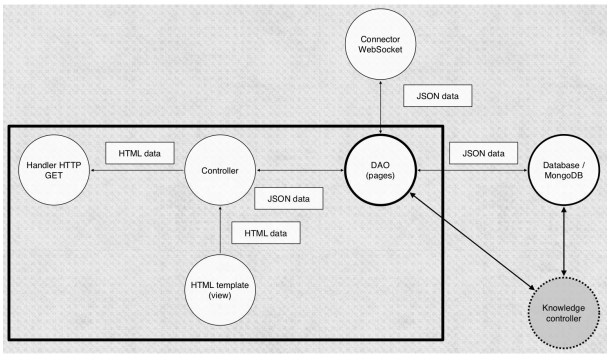


Figure 5: Back end architecture

4. An eight months developer's experience

WikiNext is an eight months developer's experience with emergent tools and technologies, and still a work in progress. Developing a semantic wiki purely in JavaScript, from database

[13] http://www.socket.io, note that while socket.io uses WebSockets when available, it will choose the best alternative transport system (without affecting the API) in case the browser is incompatible (like Flash sockets, Ajax polling, etc.), enabling the applications to run even on old browsers.

[14] MongoDB manages collections of BSON (binary JSON) objects.

to client code, can be seen as a challenge, even in a research group that has already developed two semantic wikis with more conventional technologies. NodeJS, MongoDB, HTML5 and semantics with JavaScript draw a lot of interests now and it is time to give a good return of experience on a mid-size project development.

5. REFERENCES

[1] AceWiki : http://gopubmed.biotec.tu-dresden.de/AceWiki/

[2] Michel Buffa, Fabien Gandon, Guillaume Ereteo, Peter Sander and Catherine Faron, SweetWiki: A semantic wiki, Special Issue of the Journal of Web Semantics on Semantic Web and Web 2.0, Volume 6, Issue 1, February 2008 , Edited by Mark Greaves and Peter Mika, Elsevier, Pages 84-97

[3] Orlandi, F. (2008). Using and extending the sioc ontology for a fine-grained wiki modeling. Master’s thesis, Università degli Studi di Modena e Reggio Emilia.

[4] M.Buffa, N.Delaforge and G.Husson, ”SweetDeki : le wiki sémantique couteau suisse du réseau social ISICIL”, conférence EGC 2012 (Extraction et Gestion des Connaissances), 31 Janvier – 4 février 2012, Bordeaux, France.

[5] Olivier Corby and Catherine Faron-Zucker,
The KGRAM Abstract Machine for Knowledge Graph Querying, IEEE/WIC/ACM International Conference, September 2010, Toronto, Canada.

[6] Schaffert, S. et Al. (2009). KiWi - A Platform for Semantic Social Software. In 4th Semantic Wiki Workshop (SemWiki 2009) at the 6th European Semantic Web Conference (ESWC 2009), Hersonissos, Greece, June 1st, 2009.

[7] Schaffert, S., R. Westenthaler, et A. Gruber (2006). IkeWiki : A user-friendly semantic wiki. In 3rd European Semantic Web Conference (ESWC06).

[8] Völkel, M., M. Krötzsch, D. Vrandecic, H. Haller, et R. Studer (2006). Semantic wikipedia. In WWW ’06 : Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, pp. 585–594. ACM.