WikiNEXT: a wiki for exploiting the web of data

Pavel Arapov, Michel Buffa, Amel Ben Othmane
Wimmics research group
INRIA and I3S Laboratories
Sophia Antipolis, France
arapov@i3s.unice.fr, buffa@i3s.unice.fr, amel.ben-othmane@inria.fr

ABSTRACT

This paper presents WikiNEXT, *a semantic application wiki*. WikiNEXT lies on the border between application wikis and modern web based IDEs (Integrated Development Environments) like jsbin.com, jsfiddle.net, cloud9ide.com, etc. It has been initially created for writing documents that integrate data from external data sources of the web of data, such as DBPedia.org or FreeBase.com, or for writing interactive tutorials (e.g. an HTML5 tutorial, a semantic web programming tutorial) that mix text and interactive examples in the same page. The system combines some powerful aspects from (i) wikis, such as ease of use, collaboration and openness, (ii) semantic web/wikis such as making information processable by machines and (iii) web-based IDEs such as instant development and code testing in a web browser. WikiNEXT is for writing documents/pages as well as for writing web applications that manipulate semantic data, either locally or coming from the web of data. These applications can be created, edited or cloned in the browser and can be used for integrating data visualizations in wiki pages, for annotating content with metadata, or for any kind of processing. WikiNEXT is particularly suited for teaching web technologies or for writing documents that integrate data from the web of data.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Design, Experimentation, Languages.

Keywords

Semantic Web, Web2.0, Wikis, Semantic Wikis, Knowledge Management, Web Applications

1. Introduction / motivation

Since Ward Cunningham's wiki in 1995, a lot of wiki engines have appeared, and propose common functionalities that we can see as the "DNA of wikis": they allow users to easily create, link together, update, and delete wiki pages in a collaborative way. Rapidly some wiki engines provided "industrial" features adapted to private organizations [5] like backend integration, security management, versioning, possibility to embed HTML forms, templates, write small applications within the wiki pages, plugins and extensions systems, attachments to wiki pages, etc. We call this trend "application wikis" [1] where we find TWiki, FOSWiki, TikiWiki, Confluence, XWiki, Mindtouch Core (previously DekiWiki), etc. Another interesting trend appeared in 2005 with "semantic wikis": wikis that use the wiki way for creating/maintaining ontologies (data models) collaboratively and/or for annotating the content of the wiki using ontologies, and enable reasoning on the data. See [4][10] for an overview. Our group developed two semantic wikis: SweetWiki [4] in 2006-2010 and SweetDeki[6] recently, giving us a strong experience in that field. Furthermore, we authored the online HTML5 certification course for the W3C[2] and had to develop many HTML5 interactive examples in HTML/JavaScript/CSS, using the jsbin.com web-based IDE. The integration of these examples in the moodle[3] e-learning platform (that the W3C imposes to the authors) was painful and very limited (using just links or iframes): we imagined rapidly how it could have been done using a wiki. Since 2012, we are also in charge of DBPedia.fr, the French version of DBPedia.org, a semantically structured version of Wikipedia that accepts SPARQL[4] requests via REST web services. For this project, we started to develop WikiNEXT, a wiki engine that could act as a front-end for DBPedia: one could write applications from within the wiki, which can talk to DBPedia.fr, could write tutorials made both of text and application code that can be modified collaboratively. More generally we designed WikiNEXT for integrating HTML5/JavaScript applications that can take advantage of the web of data (in other words, process data from different data sources, eventually cache them, display and manage them in applications developed in the wiki). Writing semantic web applications can be cumbersome as there are lots of technologies to handle: RDF, RDF/S, SPARQL, datasets use different models/vocabularies/ontologies that need to be learnt, etc. WikiNEXT seamlessly integrates the editing of "documents" and the editing of "applications within the documents", hopefully with some templates, examples, tutorials, possibility to add and use external JavaScript libraries, etc. Finally, we will see that we considered each wiki page as "an application" by itself, not as a document with some restricted rectangular areas with interactive applications inside. Indeed, some applications can be "not visible": for example, applications that collect data from DBpedia.org for enriching the text of the document (highlight words that are Wikipedia entries, add semantic annotations, etc.), whereas others can be complex data visualizations with interactive parameters.

[1] http://twiki.org, http://foswiki.org, http://tiki.org, http://xwiki.org, https://atlassian.com/fr/software/confluence, http://mindtouch.com

[2] See http://www.w3devcampus.com

[3] https://moodle.org/

[4] http://www.w3.org/TR/rdf-sparql-query/, the W3C standard for querying the web of data.

This paper is organized as follows: in Section 2, we give an overview of wikis, focusing on application and semantic wikis and we explain how they are used as tools to address specific knowledge needs. We also present the concept of web-based IDEs that became popular these last years and influenced WikiNEXT a lot. In section 3 we will present WikiNEXT design principles, its architecture and its implementation. In section 4 we describe how users could use our wiki to develop semantic web applications and detail one example. Section 5 explains how we tested and validated WikiNEXT on a test group of web developers, and how we will conduct more tests in a near future. Finally, Section 6 concludes and presents some perspectives.

2. Convergence between wikis, application wikis, semantic wikis and web-based IDEs

2.1 Application wikis

Among successful Web 2.0 platforms, wikis, exemplified by Wikipedia, are known for promoting the convergence of collaborative writing. Besides enabling people to contribute content to the web, wiki usages have also been extended to other domains. Some wikis, called "applications wikis", rapidly integrated features for "programming in the pages". TWiki for example, in 1998, pioneered this approach by proposing sorts of macros for generating tables, forms, and integrated a system of plugins for extending the standard set of macros. Rapidly other wikis such as Jot (that became Google Sites), XWiki or Mindtouch[7], integrated one or several script languages for integrating "dynamic parts" in the pages. This let developers collectively build web applications within wikis including data modeling, data management, data processing and data presentation. Editing, copying and pasting code from other pages: Ward Cunningham, the author of the very first wiki, calls it *"the wiki way"*[1] and turns the wiki into an easy tool for "learning from the others' work". The source scripts and data of such *wiki applications* are invisible to web users who read pages, but editing a page reveals the code behind.

Here are some examples of macros/applications written using such scripts: instantiate a whole application (a spreadsheet, a mini blog, a photo gallery…), insert a table of contents, a form, etc. These macros are either pre-defined or written by users using a scripting language (Velocity[8] or Groovy[9] for XWiki, DekiScript, -a derivative of JavaScript- for Mindtouch Core, etc.). The set of macros may use server-side plugins/extensions. In this case the developer of plugins/extensions needs to have access to the source code of the wiki, or at least to an SDK (in Java for XWiki, perl for TWiki/FOSWiki, PHP or C#/Mono for Mindtouch), and sometimes needs some external tools (like the full Mono development environment in the case of Mindtouch). Most of these application wikis generate business and have been in development for several years, a large set of plugins/extensions is available. However, they are heavyweight tools and only a few of them integrate emergent technologies such as WebSockets[10] or HTML5 advanced APIs, as they target private organizations where browsers are generally several generations late.

[7] http://www.mindtouch.com/, uses DekisScript, a language inspired by JavaScript, for writing code within the pages.

[8] http://velocity.apache.org/

[9] http://groovy.codehaus.org/

[10] http://tools.ietf.org/html/rfc6455

2.2 Semantic wikis

The semantic web is about adding formal semantics (metadata, knowledge) to the web content for the purpose of more efficient access and management. It enables structural and semantic definitions of documents providing completely new possibilities: intelligent search instead of keyword matching, query answering instead of information retrieval, suggestions, collaborative filtering, etc. Since its vitality depends on the presence of a critical mass of metadata, the acquisition of this metadata is a major challenge for the semantic web community. However, just as the success of the World Wide Web can be attributed to the ease of use and ubiquity of web browsers, we believe that the unfolding of the semantic web vision depends on users getting powerful but easy-to-use tools for managing their information. Unlike HTML, which can be easily edited in any text editor, the W3C RDF language, the most popular standard for representing knowledge on the web of data, is more complicated to author and does not have an obvious presentation mechanism[19].

As part of this process of simplification, researchers investigated different approaches for producing and authoring knowledge. Semantic wikis is one of them. Semantic wikis such as Semantic Media Wiki[2], IkeWiki[3], SweetWiki[4], started as research projects. They allow users to add semantic context to wiki pages content while preserving the simplicity and the collaborative essence of wikis. Several semantic wiki engines also integrate features of application wikis. A semantic wiki enables users to additionally describe resources in a formal language: RDF (Resource Description Framework), OWL (Web Ontology Language), or Conceptual Graphs[12]. Annotations added to wiki pages are stored in a knowledge base. These annotations are then used to augment a wiki with functionalities like enhanced navigation, search and retrieval or reasoning. SPARQL is the standard W3C language[11] for querying RDF data. The RDF data-model is a directed, labeled, multi-graph and, thus, SPARQL is essentially a graph-matching query language. Annotations in semantic wikis are formal and possibly semantic, e.g. they are formally defined, and possibly use ontological terms. Systems such as Semantic MediaWiki[2], based on the Media Wiki engine, or SemperWiki[17] require to use special wiki syntax or to directly embed RDF in order to add semantic annotations to wiki pages. While this is an open approach in the spirit of wiki principles, this can lead to semantic heterogeneity problems since any user can use its own vocabulary to add annotations in a document, making them difficult to re-use. A system like IkeWiki[3] combines plain-text feature of wikis and a dedicated triples-based form interface to help users annotating content by re-using existing ontologies, while OntoWiki[7] can be used as a complete ontology editor, with a user-friendly interface that offers different views and browsing and editing interfaces over existing data.

Eventually, it seems important to reference DBPedia[18], a project that aims to represent Wikipedia content in RDF, as well as other semantic wikis, like SweetWiki[4] which does not focus on ontology population but on using semantic web technologies to let users tag their pages and organize those tags, focusing on pages meta-data rather than modeling content of those pages.

Today, as detailed in [20], semantic wikis which are still active have chosen different approaches: (1) extend their wiki with different functionalities or (2) externalize some parts . In the

[11] http://www.w3.org/TR/sparql11-query/

first category we find, for example, the Halo[12] extension of Semantic Media Wiki which proposes forms, auto-completion, a WYSIWYG editor, the integration of audios and videos files, and the integration of a SPARQL endpoint, or more complex extensions such as MoKi[25] (focuses on enterprise modeling) and OWiki[24] (ontology-driven generation of wiki content), also based on Semantic Media Wiki. In the second category we find the KiWi project (Knowledge in Wiki[21]), which involved authors of IkeWiki, or SweetDeki [22] the wiki that succeeded to SweetWiki [4], and has been integrated in the ISICIL ANR project [22]. These projects tried to put semantic wikis in an industrial context, including the issue of project management.

There are other issues, as illustrated by the introduction text to the first workshop about Semantic Web programming[13] (2012): *"The Web of Data is growing at an enormous pace. However, the development of dedicated software applications, capable to deal efficiently in information-rich spaces, of which the Semantic Web is one dimension, is not yet mainstream. Reasons for that include one (or more) of the following research issues: lack of integrated development environments (IDEs, such as Visual Studio and Eclipse), poor programming language support, lack of standard test beds and/or benchmarks, inadequate training, and perhaps the need for curricula revision. Properly addressing these issues requires interdisciplinary skills and the collaboration between academia and industry".* This comment is true for many semantic wiki engines too. For example, while a lot of plugins/extensions are available for Semantic Media Wiki, adding a new functionality or plugging a new external data sources often requires server side programming and modifying several files. Modern web-based IDEs are presented in the next section. They influenced WikiNEXT design principles and are a possible solution to this inconvenience.

2.3 Web based IDEs

Writing applications using a web based IDE is not new, we saw early experiments in the 90's, such as WebWriter[8], advanced tools based on interpreted languages and Ajax appeared as soon as this technology was born[9]. With Ajax and the development of JavaScript frameworks, online source code editors kept improving. Editors like Code Mirror[14], ACE Cloud9[15] or ternjs[16] propose features such as syntax highlighting, auto-completion, and have been used by many online IDEs[17]. Well-known multi-language IDEs such as compile-online.com, ideone.com or compilr.com are useful for showing to students small examples of code written in a variety of programming languages (Java, C, C++, Lips, JavaScript, etc.), the code is sent to a server, compiled, executed, and output results are displayed in another iframe in the web interface. Now, some lightweight online IDEs are well known by *web developers*: jsbin.com, jsfiddle.net or tinkerbin.com integrate popular libraries like jQuery, Sencha, Dojo, etc. and propose really interesting features such as real time editing of HTML/CSS/JavaScript code, instant preview when files are updated, versioning using Git[18], making all examples developed online sharable as each version has a unique URL. Most of these tools however, are sandboxes for trying code online or for writing examples that can be shared, like the ones we developed using jsbin.com and integrated in the HTML5 course we wrote for the W3C. Finally, there are more ambitious online tools[19] for web development in the cloud, like cloud9ide.com, shiftedit.net, coderun.com, codeanywhere.net, that makes possible creating real multi files web projects with the possibility to distinguish client-side and server-side code, with a complete team of programmers working on them. WikiNEXT can be considered as a powerful web-based IDE as it proposes most of the features found in the last generation of these tools (instant preview, run code in the browser, etc.).

3. WikiNEXT

Application wikis, semantic wikis, modern online IDEs: we tried to mix them all with WikiNEXT; *we considered each page as a web application*, composed of HTML/CSS/JavaScript code. We provided several editors in the browser, for editing the text and the JavaScript source code, and provided an API for "talking with the wiki engine internals". Consider the text content of a wiki page as data: the application part of the page (JavaScript code) can collect semantic data from an external source, use them for annotating the document, extract information from the text for suggesting tags, add graphic visualizations, save data in a cache, etc. WikiNEXT has been designed for making the process of writing such application easier, and without the need to use any extra tool. ScraperWiki[20] is a wiki that pioneered this kind of external data integration. It has been designed for monitoring government data over time (to see how public money is used). Its users upload or create/update Python/Ruby/PHP code for data scrapers directly in the wiki editor that will collect, visualize and save government data from external sources (mainly URL of web pages) [11]. WikiNEXT proposes to go further with a real client side application layer that enables richer data manipulation using semantic web standards, an API for caching and updating data, and more.

WikiNEXT [13] is both (1) a semantic wiki (e.g. a wiki that uses technologies from the semantic Web to embed formalized content, structures and links, that can reason on this knowledge, etc.) and (2) an application wiki as it provides an API and a platform of services to use from applications written in the wiki and (3) a modern web-based IDE. Unlike a traditional wiki, pages in WikiNEXT are considered as applications that can be managed in the wiki way: look at an application, clone it, look at the code, and modify it. The application layer of a wiki page is made of JavaScript code that can be edited the same way as the text content of the page, but using a dedicated, embedded source code editor. It is then possible from the JavaScript code to manipulate the Document Object Model of the page and to perform interactive actions. Any HTML based GUI element can be added in a page with JavaScript code for handing events. Talking with the wiki internals or requesting external data sources is done through the WikiNEXT API. WikiNEXT

[12] http://www.projecthalo.com/
[13] http://www.inf.puc-rio.br/~psw12/, numerous papers details the problems cited.
[14] http://codemirror.net/
[15] http://ace.c9.io/
[16] http://ternjs.net/
[17] See http://en.wikipedia.org/wiki/Comparison_of_JavaScript-based_source_code_editors
[18] http://fr.wikipedia.org/wiki/Git
[19] See for example http://designbeep.com/2011/06/28/18-useful-web-based-code-editors-for-developers/ for a list of such online services. More appear every month.
[20] https://scraperwiki.com/

provides tools for plugging external JavaScript libraries such as D3.js[21].

The prototype is open source[22] and available online[23]. It includes many examples and tutorials, e.g. for writing applications that exploit the semantic database DBPedia.org, and use the WikiNEXT API, for processing the semantic data collected and enhancing wiki documents. This makes WikiNEXT a good tool for teaching semantic web technologies, without the need to install any software. It is also a good tool for collecting data and integrating them in documents.

The next section gives an overview of the API architecture and how WikiNEXT has taken advantages of recent approaches and trends in web development.

3.1 Architecture overview

In order to realize the self-organized reuse in software engineering – in particular when performed under the agile software development paradigm[14] – wikis can be seen as a lightweight platform for exchanging reusable artifacts between and within software projects. On the other hand, based on the latest development in web technologies, improved JavaScript engines, and the introduction of HTML5/JavaScript, there are now several efficient browser-based source code editors. Combined with the existing functions of wiki, e.g. collaboration and documentation, WikiNEXT is expected to support software communities to develop JavaScript applications in a fully on-line manner which makes it a mixture between a wiki and a web based IDE. An article/page in WikiNEXT is considered as a web application: users can use different editors (1) a HTML or (2) a JavaScript editor (see Figure 1).

The WikiNEXT JavaScript API provides numerous objects, events and methods that can be used by applications developed within the wiki. In an MVC paradigm, the JavaScript part of the page will be the business layer and the Controller, the View part will be done in the HTML/CSS of the page, and the Model part is composed of RDF metadata (see Figure 2).

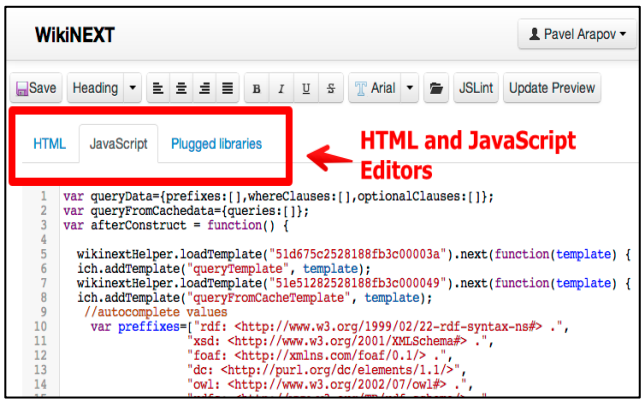


Figure 1: WikiNEXT page editors, the different tabs show the HTML editor, the JavaScript editor (current view) and the library management tab.

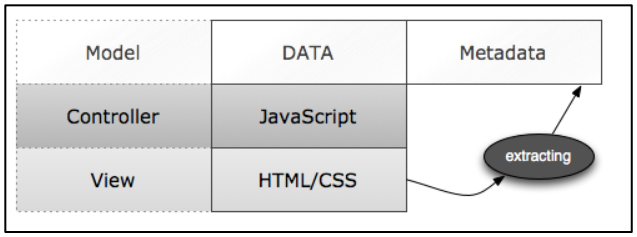


Figure 2: WikiNEXT page model.

WikiNEXT articles/pages are associated with a set of metadata that describe their basic characteristics: Title, Article, Author, Contributor, Date, etc. But articles are also containers: they can hold metadata created manually or added automatically by an application (see Figure 3). WikiNEXT uses schema.org[24] vocabularies to describe the default structure of articles, users' details, etc. Each article is represented semantically as a named graph made of RDFa/RDFalite annotations, whose name is based on its URI (Uniform Resource Identifiers). We call this graph the *Local Knowledge Base*. The full stack of wiki articles give us the *Global Knowledge Base*, the union of all the named graphs.

The metadata generated by applications embedded in the page/article, e.g. a list of countries resulting from a request or a web services call to DBPedia.org, are also represented in the page as RDFa/RDFaLite[25] annotations. WikiNEXT parses these annotations and saves them in the integrated triple store [16]. We use a modified version of RDFStore-js [16], which supports SPARQL 1.1. The traditional page content is stored as objects in a MongoDB[26] database, and indexes for the textual page content are generated this way.

3.2 Implementation

WikiNEXT has been written from scratch in JavaScript and relies on the Node.js HTTP server and on the MongoDB database. Node.js[27] (Node) is an I/O environment built on top of Google Chrome's JavaScript runtime — essentially, a server-side implementation of a JavaScript interpreter. MongoDB is a NoSQL document-oriented database. Instead of storing serialized objects in rows and columns as one would with a relational database, it stores objects using a binary form of JSON objects, called BSON[28]. RDFstore-js triplets are also persisted in MongoDB, then, RDF triplets are both accessible through RDFstore-js using SPARQL, but also directly through MongoDB, Notice that some researchers have also tried to use MongoDB directly as a triple store[15].

WikiNEXT also takes advantage of templates for creating pages with semantic annotations. Our wiki uses the Mustache[29] template engine that has a "logic-less" syntax: it doesn't rely on procedural statements (if, else, for, etc.).

With these technological choices, we tried to minimize the number of languages and the number of data formats used in

[21] http://d3js.org
[22] https://github.com/pavel-arapov/wikinext
[23] http://wikinext.gexsoft.com

[24] http://schema.org/ and http://schema-rdfs.org have been created by the main search engine vendors and propose several RDF/S schemas for representing persons, articles, etc.
[25] http://www.w3.org/TR/xhtml-rdfa-primer/ and http://www.w3.org/TR/rdfa-lite/
[26] http://mongodb.org
[27] http://nodejs.org
[28] http://www.json.org, a text based notation for JavaScript objects.
[29] http://mustache.github.io

WikiNEXT: JavaScript can be used in the whole process of development both for programming and for persistence, both on the client side and on the server side.

In the next section, we detail a typical example of an application that can be developed in this environment.

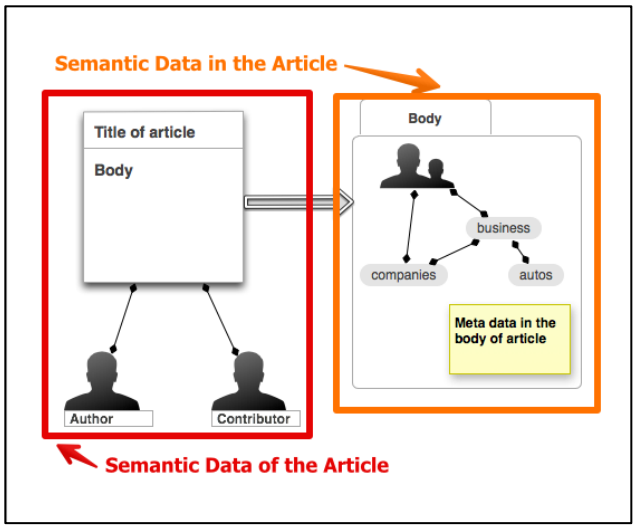


Figure 3: WikiNEXT articles contain metadata that describe the articles themselves (title, date, author, etc.) and metadata that describes the articles' content.

4. Scenario

In this section, we describe a concrete scenario of usage of WikiNEXT: we write an application in a wiki page that uses the DBPEdia.org SPARQL endpoint to retrieve linked data about cities (description, pictures, population, etc.). This application requests DBPedia.org, gets the results and uses them to populate wiki pages with semantic annotations using RDFaLite. These annotations are persisted in the wiki RDF store and we will show how this data can be reused by other applications, e.g. for a semantic mashup that displays all collected metadata about cities on a single map page.

4.1 City application

This application is available on WikiNEXT[30] and can be run, edited or cloned by any user. It is composed of:

1. A View form: as shown in Figure 4, it allows users to look for information about a city and creates a new page with the found information or makes an update if the page for this city already exists in the wiki.
2. An application code that finds information about a city by querying DBPedia.org[31]'s SPARQL endpoint.
3. A template for displaying the data collected, we use a template that gives a nice layout.
4. Some code for creating a new page: in order to reuse information, WikiNEXT offers the possibility to create pages that contain both information about cities and semantic annotations.

[30] http://wikinext.gexsoft.com/wiki/519e04c580194c4178000001
[31] http://dbpedia.org/

4.2 Retrieving data

The WikiNEXT API proposes some "default" rendering for displaying RDF data, we call this mode "sparqlOnFly", see the example of Figure 6. On the other hand, it is possible to create some "template pages" in order to have a more meaningful and structured view of the information, as shown in Figure 7. The HTML of the corresponding template page is shown in Figure 10.

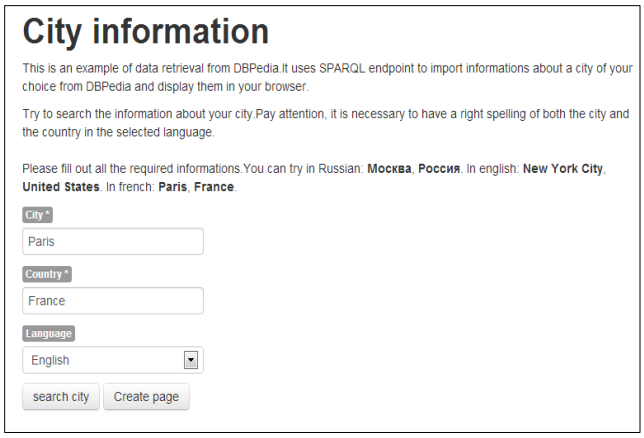


Figure 4: City application, here we see the form that will request DBPedia.org and display the results in the local page or in a new page.

The principle is simple to use and permits the decoupling of the UI (User Interface) definition from the data. The user needs first to define the UI in the HTML part as the example below:

```
Welcome to {{city}}.
```

In the JavaScript part data will take this form:

```
var data = {city: 'Nice'};
```

The result will be:

```
Welcome to Nice.
```

WikiNEXT proposes an API with various features to cache the data collected from external data sources, refresh existing ones, etc. Figure 5 shows the complete functional diagram of the WikiNEXT architecture.

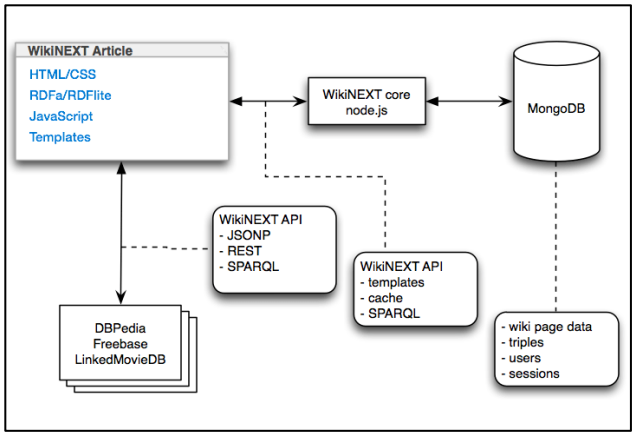


Figure 5: WikiNEXT Architecture.

4.3 Processing data

The template shown in Figure 10 both handles the display of the data retrieved from DBPedia, and the RDFa Lite annotations.. For example, a city is defined by:

```
<span about={{_id}} typeof="City">
<h1 property="name">{{city}}</h1>
…
</span>
```

In the code above we defined an object with identification "{{_id}}" and that has an RDF type "City", from the vocabulary http://schema.org/City. This vocabulary states that a City has a property "name", and we will give it the value "{{city}}". This syntax comes from the templating engine, see above.

The JavaScript code below fills the template with real values:

```
var pagedata = {};
pagedata.city = city; // dbpedia result
article = wikinextHelper.city(pagedata);
```

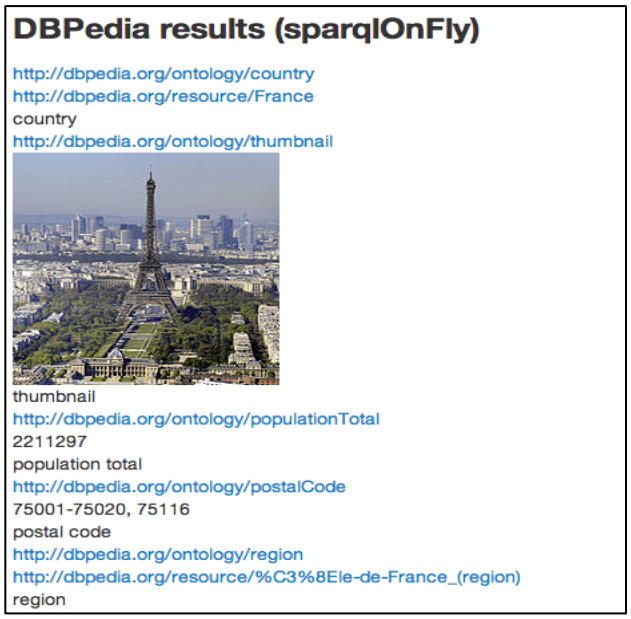


Figure 6: Results from DBPedia, default rendering without using a template. We call this rendering mode "sparqlOnFly".

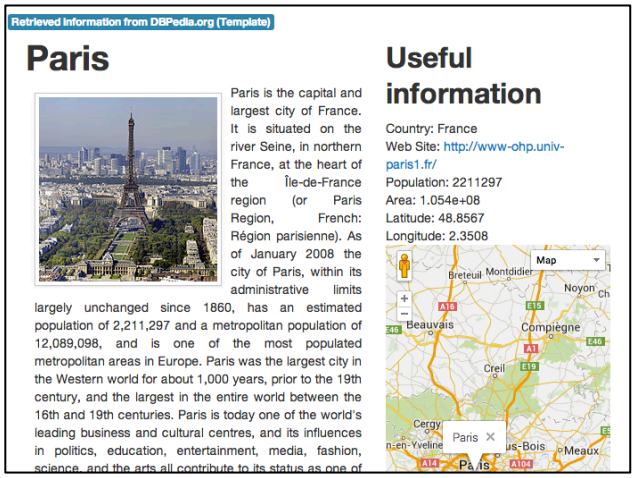
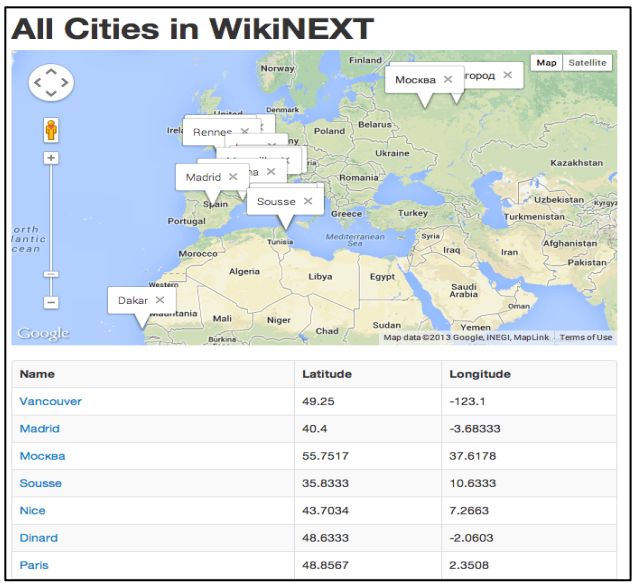


Figure 7: Results using a template.



Name	Latitude	Longitude
Vancouver	49.25	-123.1
Madrid	40.4	-3.68333
Москва	55.7517	37.6178
Sousse	35.8333	10.6333
Nice	43.7034	7.2663
Dinard	48.6333	-2.0603
Paris	48.8567	2.3508

Figure 8: Map view of all cities in WikiNEXT

4.4 Reusing data

To demonstrate data reuse, we try to recombine data sources on the wiki, creating what is typically known as mash-up by developing a web application that combines all the created cities on WikiNEXT. The Figure 8 displays the output map of the wiki application that retrieves the location of all cities available in our knowledge base and shows all cities on a map and in a table. WikiNEXT uses SPARQL to query its knowledge base (see Figure 9). Hence, queries can be written without having to deal with semantic heterogeneity problems since we use predefined ontologies.

Besides the individual display of data, WikiNEXT enables semantic search. Figure 11 shows the search result of a keyword "Paris" for example. Since we have created a web page that contains an article named Paris, the semantic search algorithm gives us as result that Paris is both the name of a web page (article) and a city in WikiNEXT.

```
21  // constructing the query
22  var query = "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
23                  "PREFIX schema: <http://schema.org/>" +
24                  "SELECT ?g ?o ?name ?v ?v1 WHERE { GRAPH ?g " +
25                  "{ " +
26                  "?s rdf:type schema:City. " +
27                  "?s schema:name ?o. " +
28      "?o ?p2 ?name ." +
29      "?s schema:geo ?coord ." +
30      "?coord schema:latitude ?lat ." +
31      "?lat ?p ?v ." +
32      "?coord schema:longitude ?lon ." +
33      "?lon ?p1 ?v1 ." +
34                  "} }";
35  // call SPARQL endpoint via the WikiNEXT API
36  wikinextHelper.endpoint(query).next(function(results) {
37    // format data for a template
38    var data_for_template = {cities:[]};
39    // pass the result data
40    _.each(results,function(value, index) {
41
42      data_for_template.cities.push({
43        name:value.name.value,
44        url:value.g.value,
45        lat:value.v.value,
46        long:value.v1.value});
47      // google maps infowindow with the city name and position
48      var infowindow = new google.maps.InfoWindow();
49      infowindow.setContent(value.name.value);
50      infowindow.setPosition(new google.maps.LatLng(value.v.value,value.v1.value));
51      infowindow.open(window.map);
52
53    });
54    // append generated template to the page
55    $("#result").append(ich.city(data_for_template));
56  });
```

Figure 9: Source code of SPARQL query to find all cities. We see the use of the WikiNEXT API line 36, and the use of templates for displaying the results (lines 42-55), see also Figure 10 for the HTML part of the template. The final rendered page is shown in Figure 7.

```
<div class="container-fluid">
  <div class="row-fluid">
    <div class="span6" style="text-align: justify" >
      <span about="{{_id}}" typeof="City">
        <h1 property="name">{{city}}</h1>
        <span rel="photo"><span typeof="ImageObject"><img src='{{thumbnail}}' align='left' class='img-
polaroid' style='margin: 10px' property="embedUrl"></span></span>
        <p style='margin: 10px'>{{abstract}}</p></span>
    </div>
    <div class="span4">
      <h2>Useful information</h2>
      Country: {{country}}<br/>
      {{#website}}Web Site: <a href="{{website}}" target="_blank" property="url">{{website}}</a><br/>
{{/website}}
      Population: {{population}}<br/>
      {{#area}}Area: {{area}} <br/> {{/area}}
      <span rel="geo">
      <span typeof="GeoCoordinates">
      Latitude:<span property="latitude"> {{latitude}}</span><br/>
      Longitude: <span property="longitude">{{longitude}}</span></span><br/>
      </span>
      <div id="map_canvas" style="width: 100%; height:400px"></div>
    </div>
  </div>
</div>
```

Figure 10: Template used for generating the results shown in Figure 7.

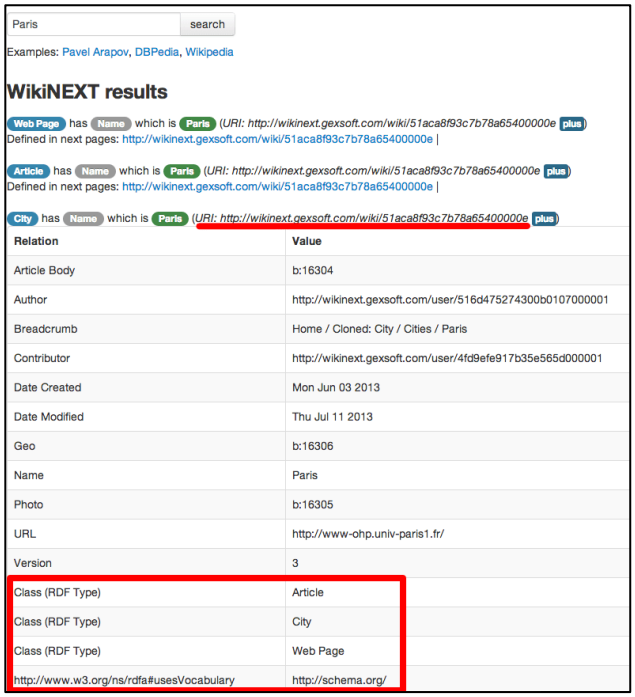


Figure 11: Search result example

5. Testing and validating WikiNEXT

We wrote some online tutorials, using WikiNEXT itself, and set up a testing protocol[32]. We choose a group of people with a diverse level of web programming knowledge, and asked them to complete various tutorials available in WikiNEXT, that go from writing very simple examples to developing the application described in section 5. Our testers were all members of the Wimmics[33] research group at the INRIA/I3S Research Laboratory (master students, PhD students, engineers, researchers, visitors). We measured the time spent for completing each tutorial, observed behaviors of our testers and took note of all the problems encountered. Finally we conducted an interview, and asked them to compare this experience with classical web development techniques for performing the same tasks (e.g. using Java/Eclipse). Results[34] showed that the system is easy to use, saves time and enables easy data reuse between applications, in particular writing HTML/CSS/JavaScript code, using the WikiNEXT API for creating pages, talking to external or internal data sources was very simple. The major issue raised by users was in the writing and debugging of SPARQL queries (the syntax itself, and the need to escape some characters when used in JavaScript code was problematic to some testers). We rapidly wrote some tools to help for this particular task (a query composer with auto completion, a toolbox with ready to go requests, history of requests used by the user, etc.) These helper applications have all been written as WikiNEXT applications[35]. Larger-scale experiments in classrooms with master students from the University of Nice who follow the semantic web course, and with students who follow the HTML5 W3C online course will be conducted next year[36].

6. Discussion and Conclusions

WikiNEXT has been written from scratch using emerging technologies, as a reaction to the difficulties to learn and teach semantic web/linked data programming, also for writing documents that integrate data from the web of data. In particular it started as a front end for DBPedia.fr.

We developed WikiNEXT after writing two semantic wikis since 2005 (SweetWiki[4] and SweetDeki[22]); from these previous experiences we wanted to develop a semantic wiki that structures its content and maintains a global knowledge base from information added by different users and also keeps an up-to-date value of changing data in a RDF store in a "more synchronous way". But we also designed WikiNEXT as a "programmable wiki", a wiki in which we can write documents with an application layer. The small applications written in this layer can be written using standard HTML/CSS/JavaScript, and can be shared, cloned, improved collaboratively. The

[32] http://wikinext.gexsoft.com/wiki/52304bb57a61be9c29000010
[33] http://wimmics.inria.fr

[34] http://wikinext.gexsoft.com/wiki/52304d287a61be9c29000011
[35] See for example the SPARQL query helper http://wikinext.gexsoft.com/wiki/51d540a7528188fb3c000038
[36] The W3C HTML5 course is available on the http://w3devcampus.com platform. It has been written by one of the authors of this paper. WikiNEXT is planned to become the platform for hosting and sharing the online examples, in replacement of jsbin.com.

applications can also be self-documented as they are parts of a wiki document. We also designed an API and protocols for keeping a tight, synchronous link between the data embedded in the pages and the databases on the server side. We showed how WikiNEXT can automatically create and/or update pages with metadata by using applications into the pages and templates, when other wikis such as Semantic Media Wiki uses extensions or plugins for this purpose. We have proposed also a new approach of combining web 2.0 technologies with semantic web technologies to support software development. The motivations of this effort, as well as related work and the research context were introduced, and the WikiNEXT API was described, in terms of provided functionalities and architecture. A case study was also presented. This paper introduces a real experience of the combination of collaborative technology (Wiki), knowledge technology (Semantic web) and web2.0 technologies to support software development. WikiNEXT will be used at the University of Nice and by the w3devcampus, the online platform that hosts the HTML5 W3C course.

7. REFERENCES

[1] Leuf, B., & Cunningham, W. (2001). The Wiki way: quick collaboration on the Web.

[2] Krötzsch, M., Vrandečić, D., & Völkel, M. (2006). Semantic mediawiki. In The Semantic Web-ISWC 2006 (pp. 935-942). Springer Berlin Heidelberg.

[3] Schaffert, S. (2006, June). IkeWiki: A semantic wiki for collaborative knowledge management. In Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on (pp. 388-396). IEEE.

[4] Buffa, Gandon, Ereteo, Sander, & 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

[5] Michel Buffa "*Intranet Wikis*", workshop "intrawebs" of WWW 2006, Edinburgh, Scotland

[6] Gandon, Abdessalem, Buffa, & Al., ISICIL: Information Semantic Integration through Communities of Intelligence online, 10th IFIP Working Conference on Virtual Enterprises, Thessaloniki, Greece, 7-9 October 2009

[7] Auer, S., Dietzold, S., & Riechert, T. (2006). OntoWiki–A tool for social, semantic collaboration. In The Semantic Web-ISWC 2006 (pp. 736-749). Springer Berlin Heidelberg.

[8] Crespo, Arturo & Bier, Eric A. WebWriter: A browser-based editor for constructing web applications. Computer Networks and ISDN Systems, 1996, vol. 28, no 7, p. 1291-1306.

[9] Serrano, M., Gallesio, E., & Loitsch, F. (2006, October). Hop: a language for programming the web 2. 0. In Conference on Object Oriented Programming Systems Languages and Applications: Companion to the 21 st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (Vol. 22, No. 26, pp. 975-985).

[10] Krotzsch, Markus, SCHAFFERT, Sebastian, & Vrandecic, Denny. Reasoning in semantic wikis. In: Reasoning Web. Springer Berlin Heidelberg, 2007. p. 310-329.

[11] Kienle, H. M. (2012). Open Data: Reverse Engineering and Maintenance Perspective. arXiv preprint arXiv:1202.1656.

[12] Oren, E., Breslin, J. G., & Decker, S. (2006, May). How semantics make better wikis. In Proceedings of the 15th international conference on World Wide Web (pp. 1071-1072). ACM

[13] Arapov, P., & Buffa, M. (2012, April). WikiNext, a JavaScript semantic wiki. Indeveloper track, WWW2012 Conference, Lyon

[14] Rech, J., Bogner, C., & Haas, V. (2007). Using wikis to tackle reuse in software projects. Software, IEEE, 24(6), 99-104.

[15] Tomaszuk, D. (2010) Document-oriented triple store based on RDF/JSON. Studies in Logic, Grammar and Rhetoric, (22 (35)).

[16] Hernández, A. G., & García, M. N. M. A JavaScript RDF store and application library for linked data client applications.

[17] E. Oren. SemperWiki: a semantic personal Wiki. In SemDesk. 2005.

[18] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In The semantic web (pp. 722-735). Springer Berlin Heidelberg.

[19] Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M. ... & Lee, R. (2009). Media meets semantic web–how the bbc uses dbpedia and linked data to make connections. In The semantic web: research and applications (pp. 723-737). Springer Berlin Heidelberg.

[20] Meilendera, T., Jayb, N., Lieberb, J., & Palomaresa, F. Semantic wiki engines: a state of the art. Semantic-web-journal.net. IOS Press, 2010.

[21] Sebastian Schaffert, Julia Eder, Matthias Samwald, & Andreas Blumauer. Kiwi - knowledge in a wiki. In European Semantic Web Conference 2008, 2008

[22] Buffa, M., Delaforge, N., Erétéo, G., Gandon, F., Giboin, A., & Limpens, F. (2013). ISICIL: Semantics and Social Networks for Business Intelligence. In SOFSEM 2013: Theory and Practice of Computer Science (pp. 67-85). Springer Berlin Heidelberg.

[23] Pérez, J., Arenas, M., & Gutierrez, C. (2006). Semantics and Complexity of SPARQL. In The Semantic Web-ISWC 2006 (pp. 30-43). Springer Berlin Heidelberg.

[24] Angelo Di Iorio, Alberto Musetti, Silvio Peroni, Fabio Vitali: Ontology-driven generation of wiki content and interfaces. The New Review of Hypermedia and Multimedia 16(1&2): 9-31 (2010)

[25] Marco Rospocher, Chiara Ghidini, Viktoria Pammer, Luciano Serafini, Stefanie N. Lindstaedt: MoKi: the Modelling wiKi. SemWiki 2009