OntoWiki Mobile – Knowledge Management in your Pocket

Timofey Ermilov
AKSW, IfI, Universität Leipzig

Norman Heino
AKSW, IfI, Universität Leipzig

Sören Auer
AKSW, IfI, Universität Leipzig

PF 100920, D-04009 Leipzig, Germany
{lastname}@informatik.uni-leipzig.de

Categories and Subject Descriptors: D.0 Software: General, E.2 Data: Data Storage and Representations [*Linked representations*]

General Terms: Algorithms, Design

Keywords: Semantic Web, RDF, Replication, Mobile Web Application, Knowledge Engineering

1. OVERVIEW

As comparatively powerful mobile computing devices become more common, mobile web applications have started to gain popularity. Such mobile web applications as Google Mail or Calendar are already in everyday use of millions of people. Some first examples of these applications use Semantic Web technologies and information in the form of RDF (e.g. TripIt). An important feature of these applications is their ability to provide offline functionality with local updates for later synchronization with a web server. The key problem to this functionality is the reconciliation, i.e. the problem of potentially *conflicting updates* from *disconnected clients*. In this paper we present an approach for a mobile semantic collaboration platform based on the OntoWiki framework [1]. It allows users to collect instance data and refine the structure knowledge bases on the go. A crucial part of OntoWiki Mobile is the advanced conflict resolution for RDF stores. The approach is based on the EvoPat [2] method for data evolution and ontology refactoring.

2. USE CASE

The development of OntoWiki Mobile was triggered by users aiming to gather data in field conditions. To simplify the data collection, we created a mobile interface that allows users to enter data instances on mobile devices such as mobile phones and tablets. For instance, there is a community of scientists who collect data about spiders in the Caucasus region[1]. They spend much time in the Caucasian mountains, searching and cataloging spider species including pictures, geo-coordinates of the find spot, classifications etc. Up to now, they have two options: either to carry a laptop computer with them in the mountains or to use a camera and a paper notebook for field recording, entering and structuring information later at home. The aim with OntoWiki Mobile was to provide them with an additional option: using OntoWiki on a mobile device even without any data connection available and limited electric power supply.

[1] `http://db.caucasus-spiders.info/`

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.

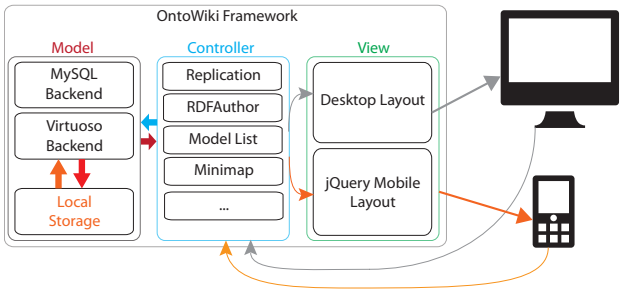

Figure 1: OntoWiki Mobile architecture.

3. ARCHITECTURE

OntoWiki Mobile is (also as OntoWiki itself) based on the Zend Framework and follows an MVC architecture (cf. Fig. 1). Both, MySQL and Virtuoso can be used as storage backends for knowledge bases. The controller layer contains all required logic and has an extensible architecture. Based on data from controllers, views create a layout which is sent to the browser (on the user's mobile device). Such architecture allows for easily extending not only the functionality of OntoWiki but also to change the layout based on context parameters of the request. Resource editing in OntoWiki is done by using RDFauthor [3] that has been adapted to be used in a mobile version.

The mobile user interface is built by using HTML5 and the jQuery Mobile[2] framework, which is a cross-platform and cross-device Javascript library that can be used on any modern mobile device. The extended HTML5 API[3] allows to get access to the device hardware (e. g. camera, gps sensor). Data replication and conflict resolution is the most complex part of OntoWiki Mobile. The process consists of three parts: (1) a client-side replication component utilizing HTML5 local storage, (2) a server-side replication component, and (3) a server-side conflict resolver (explained in more detail in the next section). Local storage (part of the HTML5 application caches) is used as a client-side storage backend for replicated parts of knowledge bases.

[2] `http://jquerymobile.com/`
[3] `http://w3.org/TR/html5/offline.html#offline`

Figure 2: OntoWiki Mobile interface screenshots. A demo version is available at: `http://m.ontowiki.net`.

4. USER INTERFACE

Figure 2 shows the OntoWiki Mobile user interface in five different browsing states. The first screenshot shows the list of classes contained in the selected knowledge base. Once a particular class was chosen, a list of instances of this class is presented (Screenshot 2) and after selecting a particular instance all properties are rendered (Screenshot 3). The rendering of properties and their values is based on the datatype and ontology structure (cf. the display of a map for attached geo-coordinates). Screenshot 4 shows how an instance can be created or edited by using forms which are automatically created by the RDFauthor [3] based on the underlying ontology structure in the knowledge base. Finally, Screenshot 5 shows an example of the interaction of OntoWiki mobile with the sensors of the mobile device in terms of accessing the integrated camera for adding a picture to an ontology instance. In accordance with popular touch-oriented mobile software platforms, the user interface was based on lists so as to simplify navigating through interlinked resources.

5. REPLICATION

While the user browses through data in OntoWiki Mobile, the data (i.e. Concise Bounded Descriptions[4] of all explored resources) is automatically replicated to the local store of the mobile device for offline use. The replication of parts of knowledge bases is necessary due to HTML5 offline web application limitations, i.e. only 5mb of local application caches can be used. When the user goes offline (i. e. mobile device looses data connection), OntoWiki Mobile starts using the local storage as data provider. All the changes made by the user, while being in offline mode are saved to the local storage and marked as new data. When data connection is re-established, the client-side replication components generate diffs for all user-generated changes. A diff represents all changes between two sets of triples. Let S be a set of triples. By making changes to S, the user yields a modified set S' of statements. The diff C is defined as the triple $C := (T, A, D) = (T, S' \setminus S, S \setminus S')$, where T is a timestamp, A contains the additions and D the deletions. Computation of a diff minimizes the amount of data to be transferred. The server-side replication component checks for conflicts and if necessary uses a conflict resolver. There are nine possible scenarios (shown in the following table) with four levels of conflict complexity: (1) no conflicts, (2) easily resolvable, (3) regular resolution, (4) complex resolution:

	$\mathsf{S} \cap A = 0$	$0 \subset \mathsf{S} \cap A \subset A$	$A \subseteq \mathsf{S}$
$D \subseteq \mathsf{S}$	1	2	2
$0 \subset D \cap \mathsf{S} \subset D$	2	4	3
$D \cap \mathsf{S} = 0$	2	3	4

Diffs from clients are received by the server-side replication component which determines the complexity of required actions and executes them accordingly. In the simplest case (i.e. without any conflicts) the received diff is applied. Easily resolvable cases are solved by partially applying the diff, i.e. fully applying only additions or deletions and solving conflicts with other parts of the diff by using standard resolution logics (i.e. omitting already added additions and already deleted deletions and logging a warning). Cases that require a regular approach are solved by using the standard resolution logics for the complete diff. To solve complex resolution scenarios, OntoWiki Mobile uses *Evolution Patterns* [2] with a specific set of rules. Moving the complex conflict resolution computations to the server side allows to save mobile device resources increasing battery life and working speed of the application.

6. CONCLUSIONS

OntoWiki Mobile is a comprehensive knowledge management tool for mobile use. It employs the new HTML5 application cache functionality to support offline work and has advanced conflict resolution features built-in. Future work will focus on the representation of provenance and use of the mobile device's sensors for context-aware knowledge base exploration.

7. REFERENCES

[1] N. Heino, S. Dietzold, M. Martin, and S. Auer. Developing Semantic Web Applications with the Ontowiki Framework. SCI. Springer, 2009.

[2] C. Rieß, N. Heino, S. Tramp, and S. Auer. EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In *ISWC 2010*, LNCS, 2010.

[3] S. Tramp, N. Heino, S. Auer, and P. Frischmuth. RDFauthor: Employing RDFa for Collaborative Knowledge Engineering. In *EKAW 2010*, LNAI, 2010.

[4] `http://www.w3.org/Submission/CBD/`