Topincs

A RESTful Web Service Interface for Topic Maps

Robert Cerny

In der Klauer 27, D-55128 Mainz, Germany
robert@cerny-online.com
http://www.cerny-online.com

Abstract. Topincs is a RESTful web service interface for retrieval and manipulation of topic maps. It allows the creation and publication of information in a way that is understandable by humans as well as computers. To accomplish this, it uses the Topic Maps Data Model and overcomes the weakness of HTML documents where most of the meaning is buried in natural language and invisible for computers to act upon. The items of a topic map are exposed and identified through URLs and manipulated with the HTTP methods. The Topincs Interface is implemented in a software product for authoring, storing and publishing topic maps, called the Topincs Server which comes with a web based editor.

1 Motivation

The World Wide Web was built around the Uniform Resource Locator (URL), the HyperText Transfer Protocol (HTTP), and the HyperText Markup Language (HTML). It started as a system for information management within the European Organisation for Nuclear Research, CERN [2]. HTML was easy to learn, documents were quickly published and whoever was interested could access them. As a matter of fact, it was so easy that it quickly spread outside CERN into the realms of science, business, and government administration, making it one of most successful computer systems to date. With the majority of the information in a document being in natural language, it requires a person to extract the meaning. This is the reason why a search engine cannot distinguish between *Topincs*, the software product and *topincs*, the misspelling of *topics*.

With computers and the Internet changing many aspects of our lives, the density of information has increased dramatically leaving however the individual with the same information processing capabilities they were equipped with some thousand years ago. A scientist has to keep track of publications, new ideas, and their source, not to mention the content of the field he is working in. A software developer has to keep up with new languages, standards and technologies, products and companies, web sites, and many more topics that he encounters during his work. A project manager has to gather information from various sources, encode, transform, and distribute it. Last but not least she has to ensure that the information she is spreading is understood.

Many positions that were created in the last 30 years deal with gathering, transforming, and distributing knowledge. For most people it becomes evident that the software products that are installed on their computer today do not enable them to manage their personal and organizational knowledge. It rather keeps them within the boundaries of natural language and additionally confronts them with artificial constructs like programs and file systems, leaving them with frustrating, endless searches through mailboxes and hard drives.

With the fore mentioned situation in mind a software system was sketched to allow a more effective approach to personal knowledge management. The first intention was to build a sophisticated notebook, which allows programmatic manipulation and retrieval of its content. Access to this notebook should not be restricted to one location or one device, but should be possible at home, work, and eventually on the road. The administration of the software should be simple and inexpensive.

These requirements led to a Client-Server Architecture consisting of a user interface running in a web browser and a web server for processing requests for Creation, Retrieval, Update, and Deletion (CRUD) of entities in a RESTful manner. This eliminates the need for installing and updating clients and allows fine grained access control to knowledge based on the URL. The blessings of modern web technology should also allow more comfort than an ordinary web page, creating an almost application like feeling within the browser. A first version of the software was presented by the author at the Open Space Session of TMRA 2005 [12].

2 Status

Topincs is mature enough to allow for the creation of topic maps in collaboration over the Internet or an intranet, allowing individuals or groups to share, publish and exchange their knowledge. One Topincs Server can be used to host many *topic map stores*, which are identified by an URL path prefix and have independent content and users. The *Topincs Server* is set on top of Apache, PHP and MySQL. The *Topincs Editor* requires the Firefox or Opera web browser. Content is served in JTM (see below), XTM 1.0, XTM 2.0, and XHTML, for viewing in any browser.

Topincs can be facilitated for Content Management, as a semantic Wiki or a Personal Information Manager. It brings humans and computers in the same starting position in the Web, simplifying the production of structured web content for humans and enabling processing for computers. If you put focus on software as a consumer of web content, Topincs can be used for Web services, since the semantics of the resources is interpretable by a machine.

Topincs was used during the course of TMRA 2006 as a knowledge logging tool. Every participant was equipped with an account to join the collaborative effort to create a conference topic map. The result of this endeavour, which was driven by the author and the conference organiser Lutz Maicher, is available at `http://www.topincs.com/tmra/2006`. Currently, access to the store is lim-

ited to the participants of the conference. The interested reader can request an account from the author.

The remainder of this paper introduces a new notation for topic maps (JSON Topic Maps), describes the web service interface and discusses the Topincs Server and Editor. An outlook for future extensions closes this paper.

3 JSON Topic Maps (JTM)

The JavaScript Object Notation (JSON) is a built-in JavaScript feature that allows literal notation of objects in programs. It fulfills a similar need as XML, the expression of structured data and the storage in human readable form. Douglas Crockford calls JSON *the fat-free alternative to XML* [6]. Parsers and generators are available for many popular programming languages [5]. The ratio between content and markup is better in JSON than in XML which the following example illustrates.

This XML document lists a few movies:

```
<collection>
  <movie year="1993"><title>Short Cuts</title></movie>
  <movie year="1999"><title>Magnolia</title></movie>
  <movie year="2004"><title>Crash</title></movie>
</collection>
```

Writing this data structure in JSON below not only reduces its size, but also yields a better content to markup ratio:

```
{"movies" : [
    {"year": 1993, "title":"Short Cuts"},
    {"year": 1999, "title":"Magnolia"},
    {"year": 2004, "title":"Crash"}]}
```

Since Topincs exposes a RESTful web service interface, it can support many representations of topic map items. While being a more compact format than XML, the parsing of a JSON string on a web browser is done by simple evaluation. The created object can then be augmented with JavaScript functions. This very small gap between data and program creates many straight forward solutions and is very comfortable to work with.

The Topics Map Data Model (TMDM) is the container a topic map can be filled into. [11] defines the structure of topic maps independent of any syntax. The notation of the TMDM in JSON is quite simple. An item is enclosed by braces. The properties of the item are written down as a comma separated list of the form *"name" : "value"*. Arrays are enclosed by square brackets.

One big advantage of Topic Maps is that the only thing that can be referred to is a topic. A topic has three means of being referred to: subject identifiers, subject locators and item identifiers. Wherever the TMDM does define a reference to a

topic, e.g. the type of occurrences, an URI identifying the topic can be mentioned directly.

The following topic map shall illustrate the JSON Topic Maps (JTM) format. It states the existence of a movie entitled *Dear Wendy* released in 2005.

```
{"topics":[
  {"subject_identifiers":[
     "http://psi.topincs.com/movies/dear-wendy"],
   "names":[
    {"value":"Dear Wendy",
     "type":"http://psi.topincs.com/title",
     "scope":[
       "http://www.topicmaps.org/xtm/1.0/country.xtm#US",
       "http://www.topicmaps.org/xtm/1.0/country.xtm#DE"]}],
   "occurrences":[
    {"value":"2005",
     "type":"http://psi.topincs.com/publication-year",
     "datatype":"http://www.w3.org/2001/XMLSchema#gYear",
     "item_identifiers":[
       "http://psi.topincs.com/movies/dear-wendy?pubyear"]}]}],
 "associations":[
  {"type":"http://psi.topicmaps.org/iso13250/model/type-instance",
   "roles":[
     {"player":"http://psi.topincs.com/movies/dear-wendy",
      "type":"http://psi.topicmaps.org/iso13250/model/instance"},
     {"player":"http://psi.topincs.com/movie",
      "type":"http://psi.topicmaps.org/iso13250/model/type"}]}]}
```

4 The Topincs Web Service Interface

This section might also be titled "To Rest!", in response to Lars Marius Garshols question "To REST Or Not to REST" in [10]. Given the requirement that the application has to run in an ordinary web browser installation without any additional modifications, the best proven possibility to communicate with a server is the HTTP protocol. Since this allows a very different way of working as the countless non RESTful web applications out in the World Wide Web demonstrate, a decision towards a RESTful API for the sake of clarity and simplicity was made. Representational State Transfer (REST) is the architectural style that guides the development of the World Wide Web [8].

The application of REST creates an interface which exposes a small number of verbs to manipulate an unlimited number of nouns. In the case of Topincs the nouns are the items of a topic map. Robert Barta already applied REST to Topic Maps [1], creating the Topic Map Interaction Protocol (TMIP). This approach is fundamentally different from TMIP, but both should be able to coexist peacefully on one server.

4.1 Addressing items

The Topincs Server segments the URL space into stores, which are identified by a unique prefix to the path. The store is a container for topic maps. A store of the imaginary movie reviewer Thomas V. could be located at `http://www.topincs.com/thomasv`. The items of a store are located in separate spaces, e.g. the topic representing the subject *Director*, might be located at `http://www.topincs.com/thomasv/topics/621`. This URL has two functions: it exposes the resource to be retrieved and manipulated with HTTP in different representations and it functions as an item identifier within the TMDM. The movie topic map of Thomas V. would be addressable under `http://www.topincs.com/thomasv/topicmaps/movies`. The item identifiers are kept independent of their location within the topic map. This way topics and associations can be moved from one map to another within the store without rendering URLs that were distributed in topic map exports broken. Consumers of such topic maps can still use the item identifiers to contact the server for updated information.

4.2 Creating items

To create a resource, which in our case is an item of the TMDM, a client must know the parent of the item. The TMDM defines a parent for every construct except for topic map. For topics and associations the parent is a topic map, names and occurrences belong to topics, variants to names and roles to associations. In Topincs the parent of a topic map is the store it resides in.

An item is created by sending a POST request with a representation of the item to its parent-to-be. The representation of the topic map item may hold children. This way topics, associations and complete topic maps can be created in one request.

This procedure for creating new resources is in accordance with [7], which specifies the usage of the POST method for creating resources as subordinate of the resource identified by the Request-URI in the Request-Line. It further enumerates some examples, one of them being the extension of a database through an append operation.

The representation format used in the examples of this paper is JTM. Yet, the Topincs interface is not limited to any specific format. To illustrate the creation of an item, we shall add the name “Regisseur” in the scope German to the topic representing the subject *Director*.

```
POST /thomasv/topics/621 HTTP/1.1
Host: http://www.topincs.com
Content-Type: application/json

{"value":"Regisseur",
 "type":"http://psi.topicmaps.org/iso13250/model/name-type",
 "scope":["http://www.topicmaps.org/xtm/1.0/language.xtm#de"]}
```

The Topincs server will process this request and in case of success return the following response:

```
HTTP/1.1 201 Created
Location: http://www.topincs.com/thomasv/names/1978
```

The URL in the *Location* header of the response is functioning as the item identifier of the created name, but allows its retrieval and manipulation as will be shown shortly.

4.3 Reading items

To retrieve an item, the client must send a GET request to its identifier. HTTP requires a GET request to be *safe* and *idempotent*. A request is *safe* if it does not have side effects and *idempotent* if numerous identical requests have the same side effects as one. The safe methods are GET and HEAD, the idempotent methods are GET, HEAD, PUT and DELETE. Furthermore, HTTP supports the retrieval of different representations of a resource [7]. To specify which representation a client can deal with, the *Accept* header is used. To communicate a certain scope the client is set in, an *Accept-Scope* header is used [1].

In case of the Topincs Editor, items are requested as *application/json*. If the client retrieves the topic *Director*, it sends the following request to the Topincs server:

```
GET /thomasv/topics/621 HTTP/1.1
Host: http://www.topincs.com
Accept: application/json
```

The server will respond:

```
HTTP/1.1 200 OK

{"subject_identifiers":["http://psi.topincs.com/director"],
 "names":[
   {"value":"Regisseur",
    "type":"http://psi.topicmaps.org/iso13250/model/name-type",
    "scope":[
      "http://www.topicmaps.org/xtm/1.0/language.xtm#de"]}]}
```

4.4 Updating items

An update request is sent directly to the item to be updated. The HTTP method for sending a new version of a resource to the server is PUT. If the orthographical rules for the German language change once more, the following request might become necessary:

```
PUT /thomasv/names/1978 HTTP/1.1
Host: http://www.topincs.com
Content-Type: application/json

{"value":"Reschisoer",
 "type":"http://psi.topicmaps.org/iso13250/model/name-type",
 "scope":["http://www.topicmaps.org/xtm/1.0/language.xtm#de]}
```

On successful completion of the update the server will respond:

```
HTTP/1.1 200 OK
```

4.5 Deleting items

If a resource becomes obsolete, the DELETE method allows us to remove it from the server. The identifier of the item to be deleted must be known to the client in order to accomplish the removal. It becomes the Request-URI of the DELETE request. To delete the German name for director the following request must be sent:

```
DELETE /thomasv/names/1978 HTTP/1.1
Host: http://www.topincs.com
```

On successful deletion the server will respond:

```
HTTP/1.1 200 OK
```

With a name all its variants are deleted. In general, the deletion of an item results in the deletion of all children. Furthermore, after deletion, the item can no longer be referred to within the store, e.g. if a topic that represents the role type *Director* is deleted, all roles of that type will be deleted. Programs that encounter the item identifier within a topic map and subsequently contact the server, will receive a response stating `404 Not Found`.

5 The Topincs Server

Currently there exists one implementation of the Topincs Interface. This Topincs Server is written in PHP and requires Apache. For persistance it uses MySQL. For every store that is maintained on the server there is one MySQL database instance. This implementation supports XTM 1.0 and 2.0, JTM, and a generic XHTML representation. The last one can be retrieved by pointing a web browser to an item identifier.

6 The Topincs Editor

The Topincs Editor is a browser based application, which is at the time of writing the only human interface with editing capabilities to the content of a store on a Topincs Server. It is written in JavaScript and makes use of the `XMLHttpRequest` object to send HTTP requests to the Server. This technique is also referred to as Ajax, a term coined by Jesse James Garrett [9]. It makes use of the Topincs Interface to manipulate and retrieve resources on the Server.

Any number of users with valid credentials can point their web browser to the URL of a Topincs Store and collaboratively edit topic maps. The Editor has a simple ontology browser, a search function, and a journal to see the development over time. It supports editing of all features of the TMDM except for Reification and Variants and allows exporting to XTM 1.0, XTM 2.0 and JTM.

When the user creates individuals, the Topincs Editor offers support by making suggestions regarding occurrence types and association types. This is done by examining the type of occurrences and associations of topics of the same type as the individual. For example if a user creates a new occurrence for a movie, a list of all occurrence types that movies have in the store is displayed. On selecting one particular type the datatype of the first occurrence of this type is looked up and suggested to the user.

This *ontological prototyping* combined with suggestions provides some guidance to avoid deviation in the usage of typing topics. An example for a deviation is, if one user uses *Director* as a role type and another uses it as a topic type. This might lead to unrecoverable situations for programmatic consumers and to confusion for human consumers. Another form of deviation, which probably affects only machines, is occurrences of the same type having different data types.

The Topincs Server and Editor are currently distributed together under a proprietary license. For both products the source code is available. The distribution can be downloaded at [4].

7 Outlook

Topincs can provide infrastructure for a semantic domain, where users and groups work in separate stores on the same domain, but have the option to publish topics with a PSI, a name and description, so that users in fellow stores in this domain can start talking in the same terms. There should be an easy way for users of the Topincs Editor to publish and to search already published PSIs to see if a topic already exists that represents the subject one wants to talk about.

Topincs needs an import hub so that legacy data following a rigid structure (like an Internet message according to RFC 822) can be easily imported into a map.

Topincs can be used as a tool to communicate across language boundaries. Typing topics need to be translated once and then people can easily start communicating.

The ontological burden and freedom that the Topincs Editor offers is too overwhelming for everyday use. Topincs will receive another client, which will work very similar to a Wiki. It will limit the user in its expressibility and at the same time offer a stronger ontological guidance, so that a deviation in the usage of typing topics can be omitted and a larger user base can targeted. The current Editor will stay in place to provide unlimited expressibility for creating the ontology by prototyping. In another step, the possibility of forms, which will fill a topic map template, will be provided. This way even stronger restrictions on the ontology can be realized.

References

1. Barta, R.: TMIP, A RESTful Topic Maps Interaction Protocol. Available at: `http://www.mulberrytech.com/Extreme/Proceedings/html/2005/Barta01/EML2005Barta01.html`
2. Berners-Lee, T.: Information Management: A Proposal. Available at: `http://www.w3.org/History/1989/proposal.html`
3. Berners-Lee, T., Hendler J., Lassila, O.: The Semantic Web. In: Scientific American, 284(5):34–43, May 2001
4. Cerny, R.: Topincs. Available at: `http://www.cerny-online.com/topincs`
5. Crockford, D.: Introducing JSON. Available at: `http://www.json.org`
6. Crockford, D.: JSON: The Fat-Free Alternative to XML. Available at: `http://www.json.org/xml.html`
7. Fielding, R., Mogul J.C., Frystyk, H., Masinter, L., Leach, P., Gettys, J., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. Available at: `http://www.ietf.org/rfc/rfc2616.txt`, Internet Engineering Task Force, 1999
8. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. Doctoral Dissertation, University of California, Irvine, 2000
9. Garrett, J.J.: Ajax: a new approach to web applications. Available at: `http://www.adaptivepath.com/publications/essays/archives/000385.php`
10. Garshol, L.M.: TMRAP — Topic Maps Remote Access Protocol. In: Charting the Topic Maps Research and Applications Landscape, Heidelberg, 2006
11. ISO/IEC FDIS 13250-2: Topic Maps — Data Model, 2005-12-16, International Organization for Standardization, Geneva, Switzerland. `http://www.isotopicmaps.org/sam/sam-model/2005-12-16/`
12. Sigel, A.: Report on the open space sessions. In: Proceedings of the First International Workshop on Topic Maps Research and Applications (TMRA'06) Leipzig; Springer LNCS, 2006