Web Services in Language Technology and Terminology Management

Uwe Quasthoff, Christian Wolff

Leipzig University
Computer Science Institute, NLP Dept.
Augustusplatz 10/11
04109 Leipzig, Germany
{quasthoff, wolff}@informatik.uni-leipzig.de

Abstract

In this paper we describe the application of web service towards language technology and terminology management. Starting from a short review of web development the notion of web services is introduced and relevant standards in this area are briefly described. Following a motivation for converting language technology applications into web services we give examples for such services based on a large language and terminology service developed over the last years. The question of modeling language technology services is discussed as well. Finally, some technical details illustrate or web service prototype.

1. Introduction

If we compare a terminological database with written resources, we find the following well known advantages:

- Size: Usually a database contains much more data than the books in typical book shelf.
- Quick look-up: Looking up a database is quicker than finding an entry even in one dictionary. This becomes even worse if we have to look in several dictionaries.

During the last years several useful databases were created. While many areas of language technology and terminology management have been covered so far, some shortcomings of this approach have become obvious as well:

- Similar steps have to be repeated to look up information on the same words or concepts in several databases.
- Databases from different vendors have different user interfaces.
- Different databases may have different data structures or different query capabilities.

In this paper, we describe a web service-based approach for overcoming these disadvantages. Web services can be used for a standardized and unified access to terminology information using a single user interface. The terminology vendor only provides the data via the web service. Using an automatic mapping of the database structure the presentation in the user interface can be customized to the user's needs. Moreover, several databases with different internal structures can be presented in a unique way.

We illustrate the methods using the web service of the various databases at the *Leipzig Wortschatz* at *http://wortschatz.uni-leipzig.de* (see [Quasthoff & Wolff 00] and [Heyer, Quasthoff, Wolff 02] for more information on this project).

2. The Web: From Static Hypertext to Modular Web Services

2.1. Web Development

In its short, 10+ year history, the web has seen dramatic technological change. It may roughly be categorized into three major phases of development (see also [Preece & Decker 02:15 who propose a simpler, two-phase development model):

1. Initially the web was designed as a means for electronically publishing distributed hypertext based on a simple protocol (HTTP) and introducing the Web Browser as client software.
2. The second phase starting in the later part of the nineties brought information systems to the web: Web information systems allow for the presentation of arbitrary information system functionality via common standards and common client software (the ubiquitous web browser).
3. The third phase which has begun only very recently and introduces two additional innovations: On the one hand, languages of enriching information on the web by providing meta information standards like the Resource Description Frame work (RDF) or the Topic Map ISO-standard. The aim is the vision of the "semantic web" which makes more complex web based application possible by better resource description. The second development, which is described in more detail in this paper, is the generalization of web-based functionality as web services. While web information systems employ a broad variety of not necessarily compatible technologies, web services make functionality and information available via standardized descriptions and access protocols.

All three stages of web development have been picked up by the language technology and terminology management community:

1. Terminology lists and dictionaries have been published as static hypertext on the web for many years (phase 1)
2. More recently, language technology applications like stemming, machine translation or information extraction have become available as web information systems (phase 2).
3. Finally, the language technology and terminology management community is beginning to transform web information systems into more flexible web services (phase 3).

A good overview on available resources may be found at "Language Technology World" (http://www.lt-world.org/ns_index.html), a comprehensive repository for all kinds of language technology-related resources and applications.

2.2. Standards for Web Services

A key issue in web service development is standardization, as only by providing common grounds for web service definition essential features of the third phase of web development can be achieved:

- Modular composition of different web services
- Integration of web services into complex applications
- Universal access to web services from different types of client software and client applications.

Currently, three standards have been proposed which address key aspects of web services and which are also employed in the language technology examples given below:

- SOAP, a simple messaging protocol for actual web service deployment (cf. [Gudgin et al. 02]),
- UDDI (Universal Description, Discovery and Integration), a standard for web service directories and web service lookup (cf. Bellwood et al. 02]), and
- the Web Service Description Language (WSDL, cf. Chinnici et al. 02]) which provides a common framework for the abstract description of web service functionality.

In a technical perspective, web services reinvent the traditional idea of remote procedure calls for the web, as arbitrary functionality can be accessed via the web. From a ("naïve") user's perspective, a web service simply returns information to given a question.

For the client side, a so-called *generic soap client* is available which allows instant access to an existing web service via a browser. Especially in the case of language services the service provider can maintain his large database locally and the user has always access to the current data.
In a user driven approach, we can specify the data wanted in a typical dictionary lookup. This will give us a set of useful methods one wants to have.

From the providers pint of view, these methods have to be implemented for their database. This hides the actual database structure and all the related technical details from the user. Additional practical examples of existing language related web services may be found at http://www.remotemethods.com/home/valueman/converte/humanlan.

3. Web Services for Language Technology and Terminology Management

In this section, we want to describe methods which are useful for terminology lookup and terminology generation. Examples are taken from the context of the *Leipzig Wortschatz* project mentioned above which comprises

- a large text corpus
- a comprehensive dictionary of inflected forms with a rich data structure for each entry (statistical information, semantic attributes, morphological and syntactical information)
- additional features extracted from text via text mining like collocations for each entry
- a rich set of tools for corpus and dictionary setup, analysis, and maintenance.

3.1. Query Types

Different web service methods may be categorized either *structurally* with respect to the underlying database model developed, or concerning the *information need* modeled by a web service method. As the structural aspect is a genuinely technical one, we will concentrate on different typical information needs in the following.

3.2. Full dictionary lookup: `give_entry`

The `give_entry` method returns the "classic" dictionary entry for a given term: The whole entry is returned as a block of text in XML format with XML tags delimiting (and describing) the logical parts of the dictionary entry. While this is useful for typical dictionary (or terminology database) usage by terminologists, the flexibility of a composable web service is not fully exploited: There are several reasons to get only *parts* of a dictionary entry using more specific or atomic rather than composite web service methods as will be shown in the next section.

3.3. Partial Dictionary Lookup

Linguistic databases can contain huge information about a single word: The monolingual part may contain statistical, grammatical and semantic information. There may be additional multilingual parts.

In the translation process one might be interested in a special language pair and, moreover, subject area information contained in the monolingual part. Only these fields are relevant. Hence, we are able to define special web service methods which gives just the desired fields, not all information which might be available. A very simple example for such an atomic web service methods is given in the appendix where a SOAP example of a `getBaseForms` web service method for the *Leipzig Wortschatz* database is given (see ch. 6 below).

3.4. Terminology Extraction

In addition to dictionary lookup, text analysis is an interesting application for Web Services. Results can be mono-

lingual terminology lists derived from the text given. Combined with bilingual resources also a bilingual terminology list can be produced. An example for this kind of web service is the *Concept Extractor*, a web service-based software tool developed on top of the *Leipzig Wortschatz* infrastructure which extracts relevant terminology from given texts via an application of differential corpus analysis (see [Faulstich et al. 02] for further details).

4. Conclusion

In the past months we have started developing and offering web services for terminological information which may be used for information presentation as well as integration into language technology applications. While on a technological level, standards for offering such services have become available, further standardization is needed for services naming and easier service discovery. We are working on a complete set of web service functions, atomic as well as composite for the most pressing needs of our users in the language technology and terminology management area.

5. References

[Bellwood et al. 02] Bellwod, T. et al.; "UDDI Version 3.0"; Universal Description, Discovery and Integration (UDDI) Project, Published Specification, July 2002, http://uddi.org/pubs/uddi_v3.htm.

[Chinnici et al. 02] Chinnici, R. et al.; Web Services Description Language (WSDL) Version 1.2. World Wide web Consortium Working Draft, July 2002, http://www.w3.org/TR/wsdl12.

[Dorai & Yacoob 02] Dorai, G. T.; Yacoob, Y.; "Embedded Grammar Tags: Advancing Natural Language Interaction on the Web"; IEEE Intelligent Systems 17(1) (2002), 53.

[Faulstich et al. 02] Faulstich, L. C.; Quasthoff, U.; Schmidt, F.; Wolff, Ch. "Concept Extractor - Ein flexibler und domänenspezifischer Web Service zur Beschlagwortung von Texten." In: Hammwöhner, R.; Wolff, Ch.; Womser-Haccker, Ch. (2002). Information und Mobilität, Proc. 8. International Symposion in Information Science, Regensburg, October 2002 [to appear].

[Gudgin et al. 02] Gudgin, M. et al.; "SOAP Version 1.2 Part 1: Messaging Framework"; World Wide web Consortium Working Draft, June 2002, http://www.w3.org/TR/soap12-part1.

[Heyer, Quasthoff, Wolff 00] Heyer, G.; Quasthoff, U.; Wolff, Ch.; "Aiding Web Searches by Statistical Classification Tools. " Proc. Proc. 7. Intern. Symposium f. Informationswissenschaft ISI 2000, UVK, Konstanz (2000), 163-177.

[Heyer, Quasthoff, Wolff 02] Heyer, G.; Quasthoff, U.; Wolff, Ch.; "Knowledge Extraction from Text: Using Filters on Collocation Sets." Proc. LREC-2002. Third International Conference on Language Resources and Evaluation. Las Palmas, May 2002, Vol. III, 241-246.

[Quasthoff & Wolff 00] Quasthoff, U.; Wolff, Ch.; "An Infrastructure for Corpus-Based Monolingual Dictionaries." Proc. LREC-2000. Second International Conference on Language Resources and Evaluation. Athens, May/June 2000, Vol. I, 241-246.

[Preece & Decker 02] Preece, A.; Decker, M. "Intelligent Web Services". IEEE Intelligent Systems 17(1) (2002), 15-17.

[Schatz 02] Schatz, B.; "The Interspace: Concept Navigation across Distributed Communities"; IEEE Computer 35, 1 (2002), 54-62.

[Vinoski 02a] Vinoski, St.; "Web Services Inetraction Models, Part 1: Current Practice"; IEEE Internet Computing 6(3) (2002), 89-91.

[Vinoski 02b] Vinoski, St.; "Putting the 'Web' into Web Services. Web Services Interaction Models, Part 2"; IEEE Internet Computing 6(4) (2002), 90-92.

6. Appendices

6.1. Soap Request – Reponse Example

```
Request:
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <soapenv:Body>
  <ns1:getBaseForms xmlns:ns1="urn:LdbApi">
   <word xsi:type="xsd:string">Sachsen</word>
  </ns1:getBaseForms>
 </soapenv:Body>
</soapenv:Envelope>


Response:
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
```

```
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getBaseFormsResponse
     soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:ns1="urn:LdbApi">
   <getBaseFormsReturn xsi:type="soapenc:Array"
      soapenc:arrayType="xsd:any[3]"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item xsi:type="xsd:string">Sachsen</item>
    <item xsi:type="xsd:string">Sachs</item>
    <item xsi:type="xsd:string">Sachse</item>
   </getBaseFormsReturn>
  </ns1:getBaseFormsResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Figure 1: SOAP Source Code for the getBaseForm *example*

6.2. Generic Soap Client Screenshots for the getBaseForm Web Service

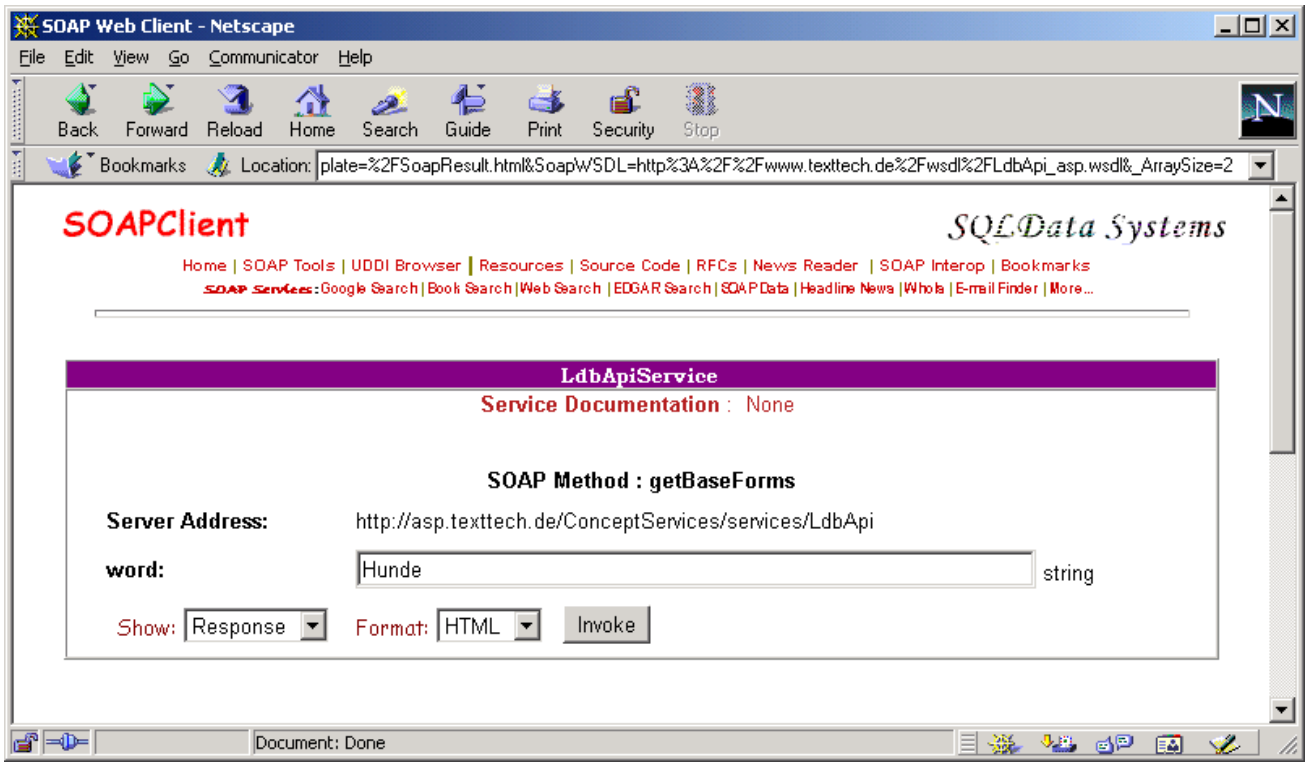

Figure 2: Generic SOAP Client Interface or the getBaseForm Service

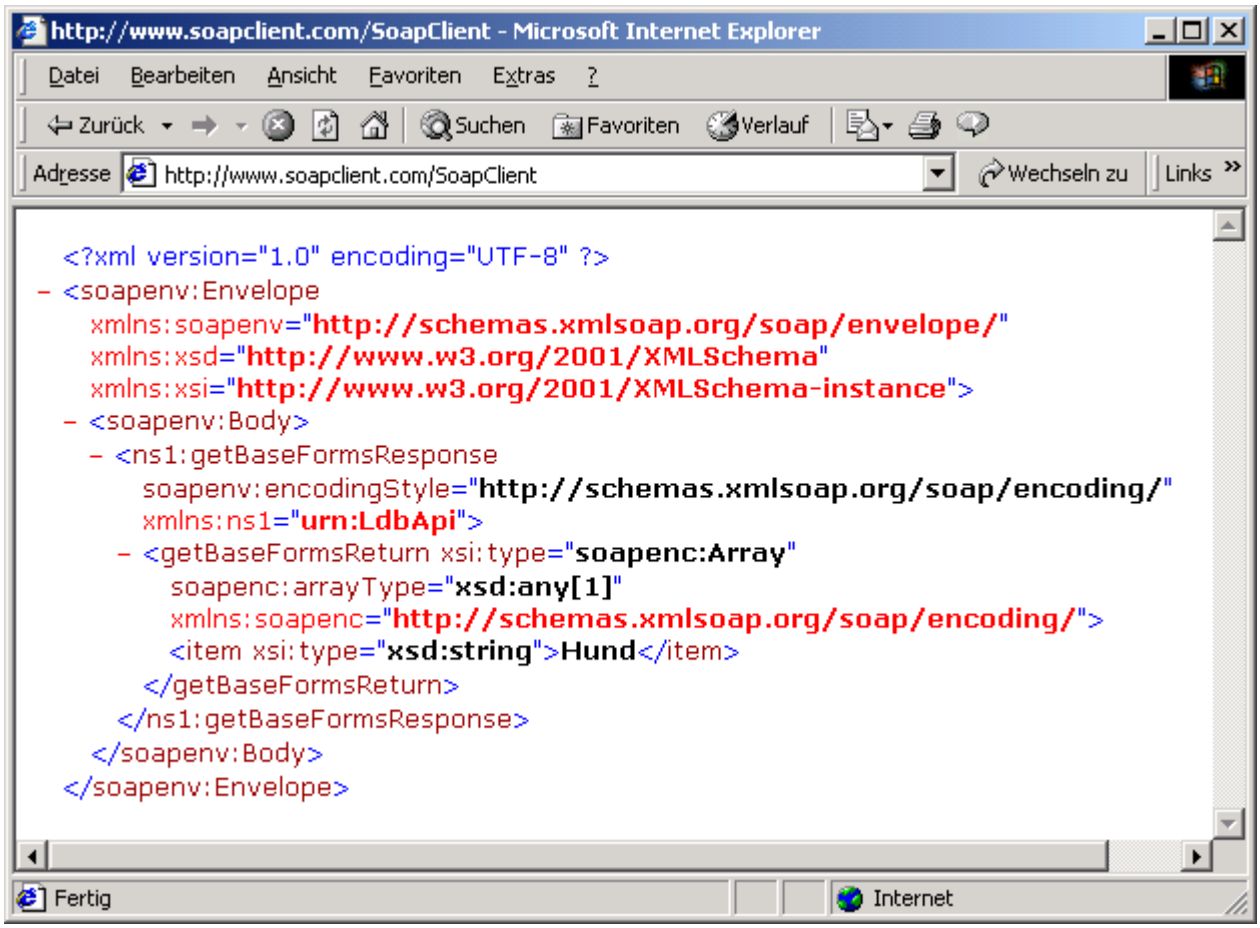

Figure 3: XML Output for the getBaseForm Service