Accepted for: The 8th IASTED International Conference on Software Engineering and Aapplications, SEA 2004, Nov. 9-11, 2004, MIT, Cambridge, USA

Generic Approach to Internationalization of Websites

Tomasz Müldner, Darcy Benoit and Fei Wang
Jodrey School of Computer Science, Acadia University, Wolfville, NS, Canada B4P 2R6
Email: {tomasz.muldner, darcy.benoit, 056330w}@acadiau.ca

Abstract

There is a growing interest in international collaboration, and therefore a need for internationalized software that can be localized to various languages. Traditionally, systems to develop internationalized software use a specific database to store translations, and either manual or automatic translations of data. This paper describes GIW, a generic system that can be used to create internationalized websites. GIW is able to use any kind of database management system and is also able to handle both manual and automatic translations. These changes can be made without any modification to the architecture of the GIW system.

Keywords:

internationalization, globalization, database, XML.

1. Introduction

Growing globalization results in software systems developed in one country and shipped to several other countries. To avoid multiple versions of the same software, software developers had to develop a more general approach, in which the system can be *internationalized*, that is adapted to various languages without making any significant changes to the architecture. *Localization* of the internationalized system refers to the adaptation of this system to a specific *locale*, which describes the language. Due to the length of the terms *internationalization* and *localization*, the short, mnemonic terms of I18N and L10N are used respectively.

For many reasons, software internationalization and localization are difficult. The most obvious reasons are the many facets of natural languages, such as alphabets and scripts, spacing rules, text direction, date and currency formats, sort orders, etc. Other essential obstacles in building systems to create internationalized software include the choice of persistent storage for storing and reusing translations and the decision as to whether translations should be *automatic* or *manual*. Automatic translations are translations performed that by specialized software while manual translations are performed by human beings. Many companies, such as EXCEL Translations [8] specialize in internationalizing existing applications.

The goal of our research is to use a generic approach which tackles the above mentioned obstacles. As a result, we have developed the **G**eneric **I**nternationalization of **W**ebsites (**GIW**) system. Our system can use any type of database for persistent storage and is also able to handle both manual and automatic translation. All of this can be done without modifying the architecture of the GIW system.

The design of GIW is based on our previous work on internationalization. In [20] and [21], we described the **I**nternationalized **F**aculty **W**ebsite (**IFW**) system which can be used to create an internationalized website showing the Curriculum Vitae (CV) for a faculty member. IFW was limited to a single format of a CV and used DB2, an XML-enabled database.

This paper is organized as follows. Section 2 covers related work. For the sake of completeness, we also briefly review our previous work. Section 3 provides details of GIW, and Section 4 concludes the document.

2. Related Work

This section briefly describes issues related to internationalization, the support for the internationalization given by XML, and an introduction to automatic translation; for more information on the internationalization process, see [26]. We also describe our previous research.

2.1 Internationalized Applications

More and more applications are being internationalized. One example of this is the Hotel Reservation System [13] where the user can choose one of 25 available languages. However, there are few internationalized personal web pages or educational applications, with some exceptions such as the Mozilla [19] and Opera [24] web browsers. Both browsers have a core binary that is able to function by loading a separate file that contains the appropriate information for a localized interface. They each provide language files for over 20 different languages, allowing for an easy switch in the interface language. Finally, Webmail [35] is a popular email client running from any browser, whose user can choose one of over 20 languages.

2.2 XML and Internationalization

There are many advantages of using XML [40] data for internationalization, including support for Unicode, and ease of converting to various formats, including HTML,

PDF, etc. For more guidelines for creating XML documents for internationalization see [40] and [27].

2.3 Automated Translation

At the time of writing this paper, Google [11] and other sites provide English translation of small text fragments or entire Web pages between several languages. A user may choose to set the Google homepage to one of more than 100 interface languages. Microsoft Word 2002 is able to perform automatic translation between Chinese, English, French, German, Italian, Japanese, Korean, Portuguese, Russian, and Spanish. Using a free upgrade from WordLingo [36] it is possible to translate between many other languages. While not perfect, these translations can help the translator to perform the required task. Repeated translations of the same string should be avoided in order to make the translation process more efficient. For example, translations of common phrases such as "Press any key to continue" should be stored for future reuse. It is expected that as translation systems become more accurate, we will be able to automate much of the translation process. *Computer-assisted translation* uses **T**ranslation **M**emory (**TM**) systems that typically consist of a translator module, an editor module, and a database of terms. TM software stores language segments translated by translator in a database for future reuse. Translators working on text segments can invoke fuzzy searches for these segments and use the results retrieved from the database. Some well known companies offering TM systems are Déjà Vu [5], the Translator's Workbench from Trados [32], and the STAR Transit [33]. TM software typically uses the **T**ranslation **M**emory e**X**change (**TMX**) format, a standardized XML document type for storing collections of segments in multiple languages. For more information on TMX, see [17].

Using TM software for translations has both advantages and drawbacks. Firstly, TM software views the source text as a collection of text units called *segments*. Segments may range in size from simple text strings to paragraphs. The technique used to break up the text into segments is called *segmentation*. Segmentation may remove the context in which the text segment appeared, resulting in an incorrect translation. An example in [29] shows that the English word "Help" translates to two different French words depending on the context in which the word appears. The common solution to the context problem is to use a verification phase in which the translator reads, verifies and possibly corrects the translation. The second problem with TM systems is that they are expensive, both in terms of the price of the software and the need to hire specialized personal able to use these systems. (More on automatic translation in [7]).

2.4 IFW, Internationalized Faculty Websites

The design of IFW uses **s**eparation **o**f **c**oncerns (**SOC**) to separate tasks that require different type of technical expertise.

2.4.1 Users of IFW

There are four kinds of IFW users:

- creators (specify the XML schema and enter XML data conforming to this schema)
- administrators (maintain accounts, XML schemas, forward documents, etc.)
- translators (translate documents submitted by administrators)
- verifiers (verify translations submitted by administrators)

IFW is a distributed system, consisting of a central server (IFW server) and users accessing the IFW server through the Internet. Any browser can be used to access the IFW system. Users must have accounts on the IFW server. Administrators of the IFW server create, modify and delete accounts for all users and maintain repositories of available translators and verifiers. Initially, the creator sends a request to the administrator to create an account.

2.4.2 Kinds of Languages used by IFW

There are several kinds of languages used in IFW:

- The *interface language* is the HCI language that appears in IFW's GUIs. The current version of our system uses English as an interface language.
- The *source language* is the language used by the creator to enter data
- The *primary language* is the default language selected for the final product. For example, if the primary language is Spanish, then initial access to the website will be in Spanish. The website will also be available in the other languages selected by the creator when using the IFW system.
- The *secondary language* is any language specified by the creator as one of the languages the final product can be displayed in (therefore the data have to be translated from the source language to the primary language, and all the secondary languages).

After the creator entered her or his data, they select various options to affect the creation of the final product. These options include the selection of:

- the primary display language for the website
- one or more secondary languages available for display
- data to be shown (all data, only journal publications, etc.)
- formats in which data can be rendered (HTML, PDF, PostScript, etc.).

2.4.3 Implementation of IFW

IFW is implemented with the help of several recently developed software tools using Java and XML; specifically, JAXB, relational databases that support XML, and versioning of XML documents. This section briefly describes the implementation; for more details, see [20] and [21]. Internally, all IFW data are stored in XML. The creator enters data by working as an IFW client (see below), and these data are stored in the database on the IDUX server. To implement the programs used by all kinds of users, such as creators or translators, we use JAXB [25]. JAXB is a Java technology that makes it easy to read, modify and write back XML data. JAXB hides the internal XML representation of data and shows appropriate GUIs to create data, translate them, etc. The text extracted from the database is handed over to one or more translators, who will be provided with the text

2.4.4 Databases

The implementation of IFW uses IBM's DB2/UDB relational database. IFW will access the database to perform several of its basic tasks, such as:

- building a localized CV for a specific language
- maintenance of the database, such as adding new faculty members, removing and updating data for existing faculty members
- performing translations and verification tasks.

2.4.5 Translation Process

In order to aid in the process of selecting translators, each translator has a *profil*e that lists pairs of languages; each language pair consists of the source language (a language to translate *from*) and the target language (the language to translate *to*). A translator's profile may include (English, French) and (English, German), while another translator's profile may include (French, Polish) and (Polish, French). We do not assume that the ability of translating from one language to another is reflexive. Similarly, each verifier has a profile listing all pairs of languages that this verifier can verify translations from and to. Administrators use repositories of translators and verifiers to determine the languages to include on the list of *IFW-available languages*. The verifier is able to accept a translation (possibly with minor corrections), or reject it. Note that all data, their translations and status (new, translated or accepted) are permanently stored on the IFW server. The translators and verifiers can re-use previously accepted translations. Translators and verifiers can request accepted translations of a standard CV e.g. from English to Polish translations of just expertise keywords from German to Chinese.

The translated text is permanently stored in the IFW server with no repetitions, allowing for reuse as needed. The creator is able to modify data and resubmit them for translation.

2.4.6 Final Product and Maintenance of CV Data

In the previous sections, we described the process of creating an internationalized website. This section provides a more detailed description of what the final product – an internationalized website – looks like, and what the specific requirements are on the system serving this website (besides the standard requirement of a web server).

The final product is a website, containing one default webpage (displayed using a primary language), which can be used to:

- choose a secondary language to display CV data
- choose which data will be displayed
- choose a format for display.

The final product may appear in one of two available kinds:

- transient. All translations are stored in the database, and webpages accessible from the default page are dynamic. The website has to provide the same functionality as the IFW server.
- persistent. All webpages are static, generated by retrieving data from the database, and applying all transformations.

From the perspective of the client, who is accessing the website, there is no difference in what kind of final product is used. However, choosing a specific kind influences requirements on the server side, and efficiency of the website.

The generation process is performed by the IFW administrator. This process is time consuming and therefore should be performed only if the CV data are not to be frequently changed. On the other hand, a persistent product, consisting entirely of static webpages, reduces the overhead caused by creating dynamic webpages, and does not impose any additional requirements on the website server. Therefore, the persistent product can be copied from the IFW server to any site with the standard web server. A transient product requires a website with the server that can handle servlets, access a database.

Maintenance of CV data, such as adding a new publication, requires a submission of the request to the administrator. Once translations of new data are preformed and verified, the transient product is available. However, if semi-persistent or persistent requested, then the administrator will generate this product.

2.5 Internationalization of Data Using Multiple XML Schemas

The next step in our previous research was a design and implementation of IDUX; a system for Internationalization of Data Using XML. IDUX generalized IFW by allowing clients to use multiple XML schemas. (The paper describing IDUX has been submitted to the IADIS WWW/Internet'04.)

3. Generic Internationalization of Websites

GIW generalizes our previous systems by allowing any kind of databases used for persistent storage, and switching between manual and automatic translations.

3.1 Manual and Automatic Translations

GIW uses the XML File Interchange Format (XLIFF) [38]. The advantage of the latter technique is that it is database-independent and there are various translations tools available that use XLIFF. Note that it is easy to convert XML to XLIFF using XSLT stylesheets [29]. The XLIFF format looks roughly like this:

```
<trans-unit>
  <target> … </target>
  <target> … </target>
</trans-unit>
```

Since the current state of automated translation systems requires human intervention, this version of IDUX uses no automated translation. However, its design, in particular the use of XLIFF, makes it possible to include future versions of automated translation systems with no major changes of the IDUX architecture.

Various administrative tasks may be automated, such as assigning submissions to translators. We are currently working on the implementation of algorithm, which automates the process of assigning translators and verifiers.

3.2 Persistent Storage

Recall that a database is *XML-enabled* if it allows the user to store and retrieve XML. XML-enabled relational databases are not specifically designed to store XML data and thus require some form of middleware to map the XML data to the relational tables. This is done by mapping the XML document schema (DTD, XML Schemas, RELAX NG, etc.) to the database schema. The data transfer software is then built on top of this mapping, (for details of this approach, see [2]). Mainstream database management systems (DBMSs) already support some type of mapping between XML and relational data. These methods usually involve creating a mapping between the XML schema and the relational schema. Tools are provided by the DBMS vendors to ease this particular task. Once the mapping has been determined, it is possible to insert, retrieve and update the XML data in the relational database. In keeping with their experiences, the mainstream DBMS vendors have requested that XML support be added to the next version of SQL. SQL/XML (or SQLX, as some refer to it) allows users to form SQL queries that create XML structures and to specify how relational data is to be converted to and from XML [3].

All translations used by GIW are permanently stored and can be reused in future translation tasks. To make GIW design database-independent, we have designed a common interface (see Appendix) that will be used by the implementation of GIW to talk to any database. With this design, we are able to plug-in different databases, including pure relational databases, XML-enabled databases (such as Microsoft SQL Server [18] that uses a schema language XML Reduced (XDR)), and native XML databases (such as Xindice [37]). Our current implementations use DB2 and Xindice.

3.3 Using GIW

This section provides several screenshots showing the use of GIW. Figure 1 shows the interface used by the creator to enter CV data. In this example, English is used as both the interface and the source language.

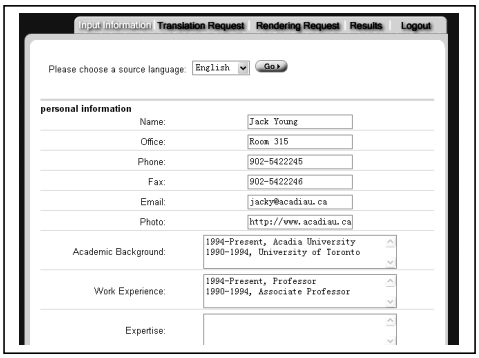

Figure 1. The GUI used to enter CV data

Figure 2 shows the interface used by the creator to select the source language, the primary language and one or more secondary languages (see Section 2.4.2). In this example, the creator selected Chinese as the primary language, and English and Polish as two secondary languages. The reason that the interface allows to select the source language is that the creator who knows several languages may wish to enter data in more than one language, and the respective translations will not be required. In our example, GIW will have to provide translations from English to Chinese and Polish.

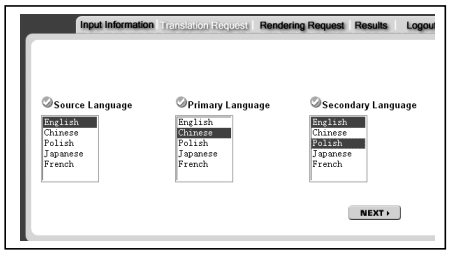

Figure 2. The GUI used to select languages

Figure 3 shows the interface used by the creator to determine the current state of translations.

Figure 3. The GUI showing the state of translations

The translation from English to Chinese has been completed, while the translation from English to Polish has not been performed. Note that the above figure also shows the type of rendering chosen by the creator; in this example it is HTML. (Recall that other types of rendering can be made available, such as PDF). Clicking the top "GO" button would show another window, with the Chinese version, in the HTML format.

4. Conclusions and Future Work

This paper described the design and implementation of a GIW system that can create internationalized websites. Currently, we are creating internationalized websites of various faculty members, which will include localizations to a number of languages, such as French, German, Japanese, Chinese, Polish, Spanish, Hungarian and Arabic.

Future work includes internationalizing the GIW system. Once the initial system is created in a single interface language, the system will be used on itself in order to generate new versions of the interface. GIW data will always be kept up-to-date with any translators that may be available to the system. As new translators are added and new languages are added, the interface will be translated to represent the new languages available for translation.

We will also experiment with XML versioning systems [6] to compare new translations and accepted translations to retrieve parts that have to be corrected. Finally, we are currently working on implementation of algorithms that automatically select translators and verifiers, releasing administrators from the task of assigning these duties.

References

[1] Aho, A. V., Hopcroft, J.E., and Ullman, J. D., Data Structures and Algorithms. Addison-Wesley, Mass, 1983.
[2] Bourret, R. (2003) XML and Databases, http://www.rpbourret.com/xml/XMLAndDatabases.htm
[3] DataDirect Technologies (2003). SQL/XML in JDBC Applications, http://www.datadirect-technologies.com/products/connectsqlxml/docs/sqlxml_whitep.pdf
[4] Deitsch, A., & Czarnecki, D. (2001). Java Internationalization. O'Reilly
[5] Déjà Vu (2003) http://www.atril.com/
[6] Delta XML (2003) http://www.deltaxml.com/
[7] Dennett, G., (1995). Translation Memory: Concept, products, impact and prospects. South Bank University http://www.star-uk.co.uk/About_us/People/Gerald_Dennett/msc.pdf
[8] EXCEL Translations (2003) http://www.xltrans.com/ser_tra.html
[9] Fitzgerald, M. (2003) Learning XSLT. O'Reilly
[10] Globalizing your e-business. (2004) http://www-306.ibm.com/software/globalization/topics/webservices/translation.jsp
[11] Google (2003) http://www.google.com
[12] Hall, M. (2001) Core Servlets and Java Server Pages. Sun Microsystems Press/Prentice Hall PTR.
[13] HRS (2003) http://www.hrs.de/
[14] Itagaki, M. (2000) Use XML as a Java Localization Solution. http://www.fawcette.com/Archives/premier/mgznarch/xml/2000/05win00/mi0005/mi0005.asp
[15] IBM, DB2 XML Extender webpage http://www-3.ibm.com/software/data/db2/extenders/xmlext/
[16] Java (2002) http://java.sun.com/j2se/1.3/docs/guide/intl/index.html
[17] LISA (2003). The Localization Industry Standards Association. http://www.lisa.org/tmx/
[18] Microsoft SQL Server (2003), http://www.microsoft.com/
[19] Mozilla project (1998) http://www.mozilla.org/docs/refList/i18n/
[20] Müldner, T., Wong, F. and Benoit, D. (2004). My webpage can speak many languages. Accepted for: EDMEDIA'04; Lugano, Switzerland
[21] Müldner, T., Wong, F. and Benoit, D. (2003). Internationalization of Websites. Technical Report 2003-04, Acadia University, 2003
[22] NetBeans (2003). http://www.netbeans.org/
[23] OmniFormat (2003) http://www.omniformat.com/
[24] Opera Browser http://www.opera.com/download/languagefiles/ , 2004.
[25] Ort, E. & Mehta, B. (2003) Java Architecture for XML Binding (JAXB) http://developer.java.sun.com/developer/technicalArticles/WebServices/jaxb/
[26] Raetzmann, M. & de Young, C. (2003) Galileo Computing Software Testing and Internationalization. TeriR@lemoine-international.com
[27] Rajgopalan, S. (2003) Software Internationalization: A Holistic View. Advisor. http://portalsadvisor.com/doc/12841
[28] RWS (2003) http://www.translate.com/locales/en-US/companyinfo.html
[29] Savourel, Y. (2001). XML Internationalization and Localization. SAMS.

[30] Seshadri, G. (2000) Internationalize JSP-based Websites. Java World, http://www.javaworld.com/javaworld/jw-03-2000/jw-03-ssj-jsp.html
[31] SUN (2002). Internationalization. http://java.sun.com/j2se/1.3/docs/guide/intl/index.html
[32] Trados (2003) http://www.trados.com/
[33] Transit (2003) http://www.star-ag.ch/eng/software/sprachtech/transit.html
[34] Unicode (2003) http://www.unicode.org/
[35] Webmail (2003) http://www.webmail.co.za
[36] WordLingo (2003) http://www.wordlingo.com/
[37] Xindice (2003) http://xml.apache.org/xindice/
[38] XLIFF (2003). XLIFF 1 Specification. http://www.oasis-open.org/committees/xliff/documents/xliff-specification.htm
[39] XML (2003) http://www.wordlingo.com/
[40] XML FAQ (2003) XML Internationalization and Localization FAQ. http://www.opentag.com/xmli18nfaq.htm
[41] XPath (2003) http://www.w3.org/TR/xpath
[42] Zydron, A. (2004) Translating XML Documents with xml:tm. http://www.xml.com/pub/a/2004/01/07/xmltm.html?page=1

Darcy Benoit 1/3/04 5:03 PM
Comment: Is this reference for 2004?

Appendix: Common Interface

```
package ca.acadiau.cs.datasource;
import java.util.Vector;
/**
  Interface for the data source layer.
  Provides the minimal required
    functionality to connect to any
    abstracted data source -- file system,
    relational database, XML database, etc.
*/
public interface DataSourceInf {
  /** Delimiter to break up tokens
        contained in Strings
    */
 public final static String DELIMITER =
      ":#:";

  /** Ask the datasource for some
    information. Queries to the datasource
    can be made against one entity only.
    The data requested for return is
    specified in returnedElements which
    are the names of the individual types
       (columns, nodes) seperated by
     the delimiter. Any conditions are also
     specified, seperated by the
     delimiter, in a key=value format.
   */

   public Vector query(String entity,
        String returnedElements,
        String condition);

  /** Removes anything matching the
     condition from the entity in the
     datasource. The entity is a single
     token string indicating the table,
     tree, file to delete from.  The
     condition is DELIMITER seperated
     key=value. String which determines
     which entries to remove from the
     datasource. Method returns true if
     something is deleted, based on the
     condition; and, returns false
     otherwise.
 */

 public boolean delete(String entity,
     String condition);

  /** Updates all elements in a given
    entity based on a condition.
    The element is a single key=value
     string of what will be updated
   based on the condition.  The condition
   is a String seperated by DELIMITER of
   key=value pairs that specify the
   conditions that an update will occur on.
 */

 public boolean update(String entity,
    String element, String condition);

 /** Inserts data into the data source.
   Adds data to the specified entity.  The
   elements to be added are specified in
   elements.  The elements parameter is a
    String of key=value pairs seperated by
    DELIMITER.
   */
   public boolean insert(String entity,
      String elements);

}
```