A Significant Analysis of Multi-regional and Multi-Lingual site on Multiple Servers

Satinder Singh[1], Dr. Amit Goel[2]

[1]M. Tech 4th Sem Student (Computer Science & Engineering), SGT University, Gurgaon
[2]Head of Department (CSE), SGT University, Gurgaon

ABSTRACT

In this paper, the author has studied about multi-regional and multi-lingual site on multiple servers. CMS, or content management systems, are platforms for managing and administering website content. There is no denying that CMSes are important in today's web ecosystem. These content management systems not only provide an easy way to build and maintain websites, but they also lend a helping hand in updating and editing website content without the need to spend hours or days writing and altering codes and scripts.

Keywords: multiple, CMS, lingual, regional, servers.

INTRODUCTION

The Web contains vast amounts of linguistic data for many languages. One key issue for linguists and language technologists is how to access it. The drawbacks of using commercial search engines are presented in Kilgarriff (2003) paper. An alternative is to crawl the Web ourselves. The author have studied this for multiple languages and here we report on the pipeline of processes which give us reasonably well-behaved, ‘clean’ corpora for each language.

The World Wide Web was an international space from its start with visitors who speak different languages and reside in various countries. Even with only taking multilingual countries into account our web presence needs to cater to visitors speaking different languages. If we add international requirements to our task list, we should also consider cultural differences, local customs, time zones, shipping costs, and other issues. As the user base of web sites and services expands, it becomes natural to provide interface and even content in more languages. Because existing monolingual web site implementations are often complicated to migrate to a Multilanguage model, it is important to keep this issue in mind when planning a new project that might involve support for multiple languages.

Fortunately building web sites has become easier in recent years with many content management systems now available that allow “click and type” web site creation. These systems help users create and manage content, and often even a community, online. Open source content management solutions first widely became popular among small businesses and hobbyists, and eventually big companies and institutions like Yahoo, NASA, Lufthansa and Nokia realized the benefits offered by these systems and deployed them.

MULTILINGUAL WEB SITE

Terminology

Because the terminology is not clearly defined and is used differently in other papers, it is important to define the basic terms. For my thesis I followed the definitions set forth by the World Wide Web Consortium (W3C) Internationalization (I18n) Activity.

Multilingual Web Site

A web site available in multiple languages. Several countries (for example Canada and Belgium) have more than one official language, so a multilingual web site is not necessarily an international one.

International web site

A web site intended to be used internationally. This type of web site is not necessarily a multilingual one because residents of multiple countries can speak the same language. A web site can both be multilingual and international, thus serving people in multiple countries with different languages available. Unfortunately language alone is not always enough to consider when presenting information to a web site visitor.

Web Standards

When building multilingual web sites, we need to first consider technical requirements and possibilities. Web standards (recommendations and specifications) define our communication means between web servers and clients, so these must be examined first. It is important to note that these standards are applicable to single language web sites too, although they are not widely known in English speaking areas of the world because the defaults provided are adequate there, so there is no immediate reason to think of these building blocks.

Internationalized Resource Identifiers

First we need an address to access a web resource. These days' users demand web sites in their own languages, including both the interface and the address. Web addresses are typically expressed using Uniform Resource Identifiers or URIs. The URI syntax, as defined in RFC 3986, restricts addresses to a small number of characters: upper and lower case letters of the English alphabet, European numerals and a small number of symbols. Unfortunately a URI does not allow for non-English characters, which limits its usability internationally. Internationalized Resource Identifiers (IRIs), as specified in RFC 3987, allow for domain names and paths to contain any Unicode character, thus allowing for fully localized web addresses. (Unicode is explained in the next subsection.) For IRIs to work, the underlying protocol (HTTP, SMTP and so on) should be able to carry the information, the format used (HTML, XML and others) should support Unicode characters, the applications handling these formats should be capable of dealing with them, and the servers hosting the resources addressed should be able to match IRIs to files and other types of resources. Unfortunately IRI supportive web browsers are not yet widely used as of this writing. While the latest Microsoft Internet Explorer Version 7 supports IRIs, this browser is not yet adopted by mainstream users. Microsoft Internet Explorer 6 only supports IRIs with an add-on installed separately. Other browsers have good support for IRIs as W3C test results show.

LANGUAGE INFORMATION AND TEXT DIRECTION

Some Sections of the HTML 4.01 recommendation specifies three key attributes of HTML that allow for language identification and directionality setting. The lang attribute, when applied to an element, specifies a language code that defines the base language of the element's attributes and content. This is useful for a number of reasons, as explained by the recommendation referenced above:

- Assisting search engines
- Assisting speech synthesizers
- Helping a user agent select glyph variants for high quality typography
- Helping a user agent choose a set of quotation marks
- Helping a user agent make decisions about hyphenation, ligatures and spacing
- Assisting spell checkers and grammar checkers

The HTTP Content-language header can also be used to specify language, but HTML attributes should override the language where appropriate in a mixed language context. The hreflang attribute has a similar role. It informs the user agent of the language of a resource being linked to in an HTML link tag. Language codes in HTML were originally constructed according to RFC 1766, which was most recently replaced by RFC 4646 and RFC 4647 and are jointly referred to as BCP 47. The structure of a language code is as follows:

The only mandatory part is a language code, which can be followed by an optional script name (Latin, Cyrillic and so on), a regional variant identifier (for example US and UK in English), any number of variant and extension identifiers and an optional private suffix (maintained for backwards compatibility). More information about these tags can be found in the W3C I18N article database. Because different scripts are used for specific languages, it is possible that text be written left-to-right (LTR) or right-to-left (RTL) independently of the language being used. Latin scripts from several languages appeared through the years, replacing or adding to RTL written ones. Although Unicode defines a few control characters to specify direction, it is generally suggested that HTML documents should not use them and build on the related HTML features instead.

Multilanguage Interface

Single language web site builders are in a convenient position to build a system in their native language and post content in the same language. However when building on an existing system, most of the time an English-based engine is working behind the scenes. This is because English is the most common language used by developers around the world, and thus has become a de facto default language for CMS products. Creating a single language web site in a different language with such a system could immediately become difficult, if internationalization is not taken into account in that product. Going further into the requirement of having a multilingual interface and content opens up new layers of required features. The interface can consist of built-in text provided by the system, as well as input provided by the administrators (site name, menu items, disclaimers, etc.).

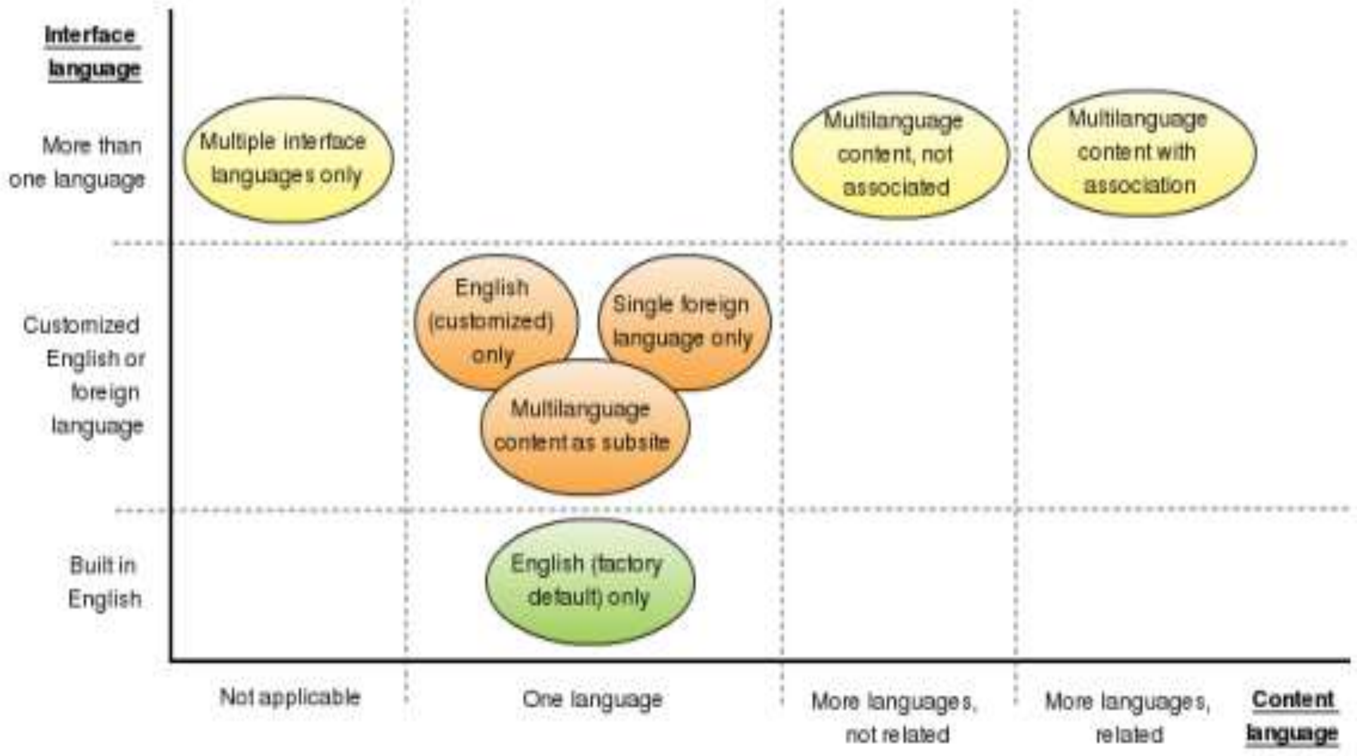
Interface language
More than one language
Customized English or foreign language
Built in English
Multiple interface languages only
Multilanguage content, not associated
Multilanguage content with association
English (customized) only
Single foreign language only
Multilanguage content as subsite
English (factory default) only
Not applicable
One language
More languages, not related
More languages, related
Content language

Figure 1: Types of foreign language based web sites

As the figure shows, sub-sites actually have no Multilanguage requirements and as a result, workflows and permissions are not related to available languages. This is often a practical way to side-step issues with multilingual content handling but results in a weak user experience and uncontrolled editorial flow. It should be noted, however, that any of the site types involving one or more content languages or at least one non-default interface language require internationalization. Although it is possible to build sites with requirements in most other parts of the type matrix shown above, only the types shown are relevant in this thesis. Naturally actual web projects often move between these models, so an ideal system should support seamless adaptation to the chosen type of site.

LANGUAGE MANAGEMENT FUNCTIONALITY

While examining the existing extensions for Drupal and analyzing the requirements posted on my request, it became apparent that low level, built-in language management support in is needed Drupal. The limited locale functionality only allowed for a set of languages to be specified. Directionality or native names of these languages were not known and language selection and detection implementation was not included in Drupal. The goal of the first layer of language support improvements for Drupal was to elevate language awareness among Drupal developers by including better built-in language support. Requirements of the language management improvements are as follows:

1. Decouple language management from the locale functionality, given that language related configuration is not limited to the interface language anymore.

2. Maintain the writing direction, native names and weights of languages to be used when displaying a language or a list of languages.

3. Add web address (domain and path) based language selection, with freely configurable domain names and path prefixes. Keep in mind that IRIs should be usable here.

4. Implement a browser setting (HTTP Accept-language) based language detection to select a default language for the user when first visiting a page without explicitly asking for a language.

5. Provide a configuration mechanism for the language selection algorithm used and how it takes domains and paths into account.

6. Implement an upgrade path for earlier Drupal versions so previously set language properties are kept in the new version.

MULTILINGUAL WEB DOCUMENTS

In order to realize true international information sharing on the Internet, it is important to be able to read and retrieve documents from every part of the world, in the same manner. Recently, WWW browsers that support Unicode are becoming popular, and operating systems are becoming to support Unicode as the internal character code. Consequently, it is much easier today to handle multilingual documents than before. However, for languages such as Japanese, Chinese, Korean, and some minor languages, character fonts and input methods that are necessary for displaying and inputting texts, are not always installed on the client side.

In ordinary personal computer environments, usually only fonts for the native language in addition to English are installed by default. Therefore, in order to display documents in other languages, the user has to install additional fonts for that language. This process is rather a hard task for casual users of personal computers and the Internet. Moreover, even if a multilingual coding system such as Unicode and the browsers that support it become popular, it is not realistic to prepare fonts for all languages supported by Unicode in every client, especially for small clients with a limited storage, such as PDAs (Personal Digital Assistants) and mobile phones.

The simplest and the easiest solution to this problem is to realize a browser that does not require fonts on the client side. That is, display of a text will be possible regardless of the installed fonts on the client, by transmitting character glyphs instead of character codes. In this approach, it is inevitable that the number of bytes to be transmitted will increase, but on the other hand, it can avoid the problem of installing and storage of fonts. Incidentally, it can be used to display characters that are not included in existing character sets. As practical methods to implement such an approach utilizing an existing Web browser, we can think of following methods.

Coding System Identification

On the other hand, for automatic identification of coding systems, the algorithms are well established for some languages such as Japanese, and are already incorporated into applications such as Web browsers and code conversion filters. These algorithms check differences of code ranges that do not overlap for each coding system (e.g. Shift JIS, EUC-JP, and ISO-

2022-JP for Japanese), and are accurate enough even for short texts. However, such algorithms are not applicable for coding systems that have the same or similar code ranges, such as language versions of EUC (Extended Unix Code).

As a method to support multiple languages, Kikui proposes a method which uses statistical language model. His method first extracts East-Asian language parts in the target document using decoder for each East-Asian language. Then, it calculates the likelihood score of each decoded character, which is obtained from the training data, and the coding system which has the highest average score is selected as the result for the East-Asian language part. The same method is applied for each 4-gram at the tail of a word in the remaining one-byte part. In his experiment, over 95% correct rate was achieved for 15 languages and 11 coding systems. However, this method needs to decode the target document assuming every supported languages and coding systems. Therefore, the addition of the language and coding system to support might lead a significant drop in performance.

CONCLUSION

Multilanguage web projects provide a very wide range of possible requirements. While the author studied several key areas, found that actual implementations might involve more specific needs, for which he tried to provide some starting points. Although the open source content management system market is very crowded, he also needed to restrict his scope by selecting a few of the available solutions to examine deeply, therefore representing different approaches to multilanguage support.

REFERENCES

[1] Script direction and languages, 20 November 2006, http://www.w3.org/ International/questions/qa-scripts.en.php.
[2] Text Expansion (or Contraction), http://www.omnilingua.com/resourcecenter/ textexpansion.aspx.
[3] Designing Web Usability: The Practice of Simplicity, Jakob Nielsen, New Riders Publishing, Indianapolis, December 20, 1999, ISBN 1-56205-810-X, pp. 312-344.
[4] Looking for internationalization use cases, 4 October 2006, http://groups.drupal. org/node/1545.
[5] Pro Drupal Development, John K. VanDyk and Matt Westgate, Apress, Berkeley, California, April 16, 2007, ISBN 1-59059-755-9.
[6] Hungarian Drupal install profile, http://drupal.hu/files/hu-5.1.tar.gz.
[7] Hungarian locale functionality (mini modules), http://drupal.org/project/ hungarian.
[8] Auto locale import module for Drupal, http://drupal.org/project/autolocale.
[9] Content Construction Kit, http://drupal.org/project/cck.
[10] Introduce dynamic object translation API, http://drupal.org/node/141461.