Web Page Interface Localisation in Devanagari for Commercial Interactive Applications by Enhancing basic Functionality of Apache Server

M. L. Dhore
Vishwakarma Institute of Technology, Pune, India

S. K. Dixit
Walchand Institute of Technology, Solapur, India.

J. B. Karande
Vishwakarma Institute of Technology, Pune, India.

ABSTRACT

The Internet is a global medium of communication and information sharing with the evolution of e-Commerce and e-Governance applications along with enterprise architecture evolution. It provides vast opportunities for existing marketing professionals to establish business dimensions and capture benefits with software resources. The serviceability factors of the Internet have proven to be beneficial to society, by and large, with the only limitation being the linguistic barrier, since English has been the predominant language for the World Wide Web since its inception and hence it's usage is confined to a specific community of people have a good grasp of the English language.. The solution to this problem is web page interface localisation. Web Page interface localisation of commercial web based interactive applications is the path-breaking solution which provides efficient and effective user-friendly interface by translating the Web page interface stored on the Web Server to the target native language on the fly. Authors focused on how to provide local language access for Indian National Language Hindi and Marathi Language of Maharashtra state for the interface of commercial web based application (domain specific application) by using controlled language.

General Terms

Localisation and Internationalisation.

Keywords

Localisation, Bi-lingual dictionary, Web Interface. Interactive Application.

1. INTRODUCTION

As the Internet continues to grow globally, Websites are providing businesses around the globe by means of viewing products & services. Over 100 million people access the Internet in a language other than English. Over 50% of Web Users speak a native language other than English. Research have shown time and time again that Web users are up to four times more likely to purchase from site that communicates in customer's language. In this paper, we have developed secure, cost-effective method for language localisation of the Web pages, which provides effective user-friendly interface by translating the Web pages stored on the Web Server to the target language on the fly. It is an attempt to provide local language access for Indian National Language Hindi and Marathi, the language of Maharashtra state for the interface of commercial web based application (domain specific application) by using controlled language [1][2][3]. It is evident that computer usage in India can only reach the masses if the human – computer interface is in Indian languages [1][3].

For translating the Web page interface in target language in Hindi or Marathi in the Web Page Translation, users type the URL of Web page. After this user selects target language and gets translated version of Web page in target language. In traditional Web Page Translation technique, Web page interface is translated into target language and then translated target language Web page is stored on the Web Server. So whenever the Client requests the Server for Web page in target language like Hindi or Marathi, corresponding translated Web page is selected & sent back in response to Client. The original English Web pages are translated into languages in which Website is providing access to Clients. But if some change has made in any original Web page then change should be performed in all translated Web pages. If change performed in original Web page is not updated properly in all corresponding translated Web pages then incorrect information gets displayed to Client.

The proposed solution is based on translating the interfaces of commercial Web pages stored on the Web Server in the target language on the fly. Proposed method solves the problem of Web Page Translation since only original English Web pages are stored on the Web Server. If Client requests Web page in target language Hindi or Marathi, then original Web page is dynamically translated into corresponding target language.

The developed Web Page Localisation software has several characteristics. First, it provides safe, quick and cost effective method for converting the existing Web pages. Second, The Web pages can be localised into any language –Indian languages Hindi and Marathi. The "translation on the fly" methodology used by Web Page Localisation results in significant reduction of the space required for storing the translated Web pages in different languages on the Web Server and continuing the stability of the original Web pages and reducing complexity for updating.

2. RELATED WORK

The major contributors in the area of localisation in India are C-DAC (Center for Development of Advanced Computing), NCST (National Center for Software Technology) and Indictrans Team. The applications they have localised are Indian Railways Reservation Charts, Mahanagar Telephone Nigams and Bilingual Telephone Directories. The C-DAC has developed the localised Unicode version of the existing online 'Web based Railway Enquiry System' hosted at www.trainenquiry.com and its related application modules. Indictrans has done the conversion of live data and file-journey-management database from nonunicode to Unicode standard as well as localised the voter list search engine for

Chief Electoral Officer of Maharashtra. FACT Financial accounting package is available in Hindi from Vedika Software. Yudit is the localised editor for Hindi, Marathi and Gujarati. Inputbhaaratii has developed several applications for inputting indic scripts for web based applications[4]. Multilingual User Interface software is developed by IIT, Chennai. Starvos Kokkotos has proposed architecture for development of internationalized software called ISDA-i [5], which is highly modular design and costly for implementation as it requires to built up different libraries and configuration files. Terence Parr [6] had proposed XML based string template for localisation of strings and other data types like currency, date, time etc. using locale. N. Anbarasan [7] detailed on localisation process for web in Indian languages and several issues related with localisation process. Cardenosa J [8] proposed approach for localisation of software. Our method describes how to modify Apache http core module to support on the fly localisation for better response time of the system.

3. SYSTEM ARCHITECTURE AND IMPLEMENTATION

Following figure (Figure 1.) shows the overall architecture of system.

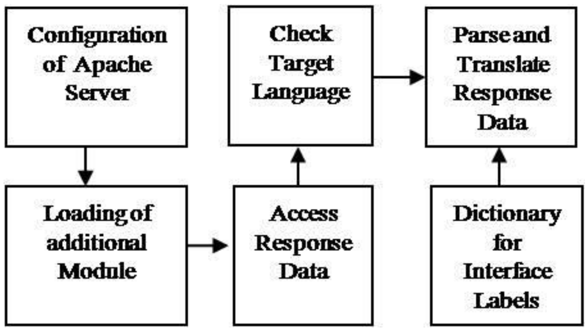

Configuration of Apache Server
Check Target Language
Parse and Translate Response Data
Loading of additional Module
Access Response Data
Dictionary for Interface Labels

Figure 1. System Architecture

3.1 Configuration of Apache Server

To develop Web Page Localisation it is necessary to first configure Apache 2.0 HTTP Server. The Apache Web Server is currently the most popular Web Server, according to a Net Craft Survey. Since Apache is an "open source" project, so any one can have access to Apache code source, and anyone can write its own modules to suits one needs. For translating the Web pages into target language on the fly, writing & loading own module is required in Web page interface Localisation[11][12]. Since Apache is open source project so writing own module & loading it is possible. For that purpose, first Apache needs to be configured [13][14].

The configuration for Apache Server is located in a file called httpd.conf, which is usually located in conf directory. The httpd.conf file contains the main Server configuration information. The basic behavior of the Server is contained in this file, such as how it runs, what port it listens to, and information on how to find other configuration files. This file consists of configuration directives, which allows developers to configure Apache Web Server according to needs. The httpd core is responsible for reading and parsing the httpd.conf configuration file.

We have configured DocumentRoot, ServerRoot and DirectoryIndex configuration directives in configuration file. DocumentRoot sets the directory from which httpd will serve files to the application. It provides the path where the Web pages of Website are stored on the Apache Web Server. ServerRoot sets the absolute path to Server directory. This directive tells the Server where to find all the resources and configuration files. Directory Index provides Web pages to which users can view[13][4].

3.2 Loading of Additional Module to Extend the Functionality of Apache Server

At the high level, the Apache Server architecture is composed of a core set of services that implement the most basic functionality of a Web Server and a set of standard modules that actually service the phases of handling an HTTP request. The Server core accepts a HTTP request and implicitly invokes the appropriate handlers, sequentially, in the appropriate order, to service the request. Apache modules actually implement the different phases of handling the request. The role of the modules is to extend the functionality of the Apache Web Server.

Apache permits loading of modules when they are needed. A handler is for Apache the action that must be performed in some phase of servicing a request. Modules define handlers and a module might specify handlers for one, many or none of the phases of a request. Handlers are the part of the module that is called when the processing of the request enters the phase for which the handler is defined. Handler generates the response sent back to Client. Apache's modular structure has simplified the way that functionality is added to the Server. Its generalized Application Programmer's Interface greatly simplifies the process of adding new or enhancing existing functionality. Many of these modules are so useful that they are included on a default Server configuration.

In Web page interface Localisation, we have developed our own module for accessing response data posted by Server as well as checking for target language. The module that is coded by developers is dynamically loaded into the Apache Web Server by using configuration directive “LoadModule”. The LoadModule directive links in the object file or library filename and adds the module structure named module to the list of active modules. We have loaded our own replace_module as “LoadModule replace_module modules/mod_replace.so” [14].

3.3 Access Response Data Posted by Server

In Web page interface Localisation for translating Web pages into target language on the fly, it is required to access response data, which Server sent as response to Client for request. We have accessed response data with the help of output filters. A filter is process that is applied to data that is sent or received by the Server. Data sent by Clients to the Server is processed by input filter and data sent by Server to Clients is processed by output filter.

When the Apache developers first began talking about Apache 2.0, one of the major goals was for one module to be able to modify the output of another. This is possible with the help of filters. Filters operate using a "chaining" mechanism. The filters are chained together into a sequence. When output is generated, it is passed through each of the filters on this chain, until it reaches the end (or "bottom") and is placed onto the network.

The top of the chain, the code generating the output, is typically called a "content generator." The content generator's output is fed into the filter chain using the standard Apache output mechanisms: ap_rputs(), ap_rprintf(), ap_rwrite(), etc. Each filter is defined by a callback. This callback takes the output from the previous filter (or the content generator if there is no previous filter), operates on it, and passes the result to the next filter in the chain. This pass-off is performed using the ap_fc_* functions, such as ap_fc_puts(), ap_fc_printf(), ap_fc_write(), etc . When content generation is complete, the system will pass an "end of stream" marker into the filter chain. The filters will use this to flush out any internal state.

In Apache filter terminology; each chunk is stored in a bucket, and lists of buckets form brigades. Lists of brigades can then create a Web document. Filters operate on one brigade at a time, and are called upon repeatedly until the entire document has been processed. This allows the Server to stream information to the Client. With the output filters, the content generator started with nothing, generated the base content, and passed that content down the filter stack to be modified; at the bottom of the filter stack, the data is sent to the network and an empty brigade is returned back up the stack. Apache filters are called as many times as necessary to process all of the data produced by the handler. An output filter is given a bucket brigade, does whatever it does, and hands a new brigade (or brigades) down to the next filter in the output filter stack. To be used at all, a filter must first be registered. Before a filter can be enabled for a given request, it must be registered with the Server. This is done using the ap_register_output_filter function. This function is invoked with four arguments: the filter name, the filter function pointer and the filter type, such as ap_register_output_filter (const char* name, ap_out_filter_func filter_func, ap_init_filter_func filter_init, ap_filter_type ftype) where name is the name to attach to filter function filter_func is the filter function to name filter_init is the function to call before the filter handlers are invoked ftype is the type of filter function either

AP_FTYPE_RESOURCE, AP_FTYPE_CONTET_SET()

This function type is used for filter callbacks. It will be passed a pointer to "this" filter, and a "bucket" containing the content to be filtered. In filter→ ctx, the callback will find its context. This context is provided, so that it installed multiple times, each receiving its own per-install context pointer [11].Callbacks are associated with a filter definition, which is specified by name. If the initialization function argument passed to the registration functions is non-NULL, it will be called if and only if the filter is in the input or output filter chains and before any data is generated to allow the filter to prepare for processing.

The *bucket structure (and all those referenced by →next and →prev) should be considered "const". The filter is allowed to modify the next/prev to insert/remove/replace elements in the bucket list, but the types and values of the individual buckets should not be altered [5].
After this registration is performed, then a filter may be added into the filter chain by using ap_add_output_filter(). The filter may beap_add_output_filter (const char* name, void *ctx, request_rec *r, conn_rec *c) where name is the name of filter to add, ctx is the context data to add in filter , r is the request data to add filter for configuration file, c is the connection to add this filter. Filters are added in a FIFO manner. The first filter added would be the first filter called. Associating a request with a filter is done when adding the filter to the filter stack. It is also possible to insert the filter automatically by using the AddOutputFilter or SetOutputFilter directives. AddOutputFilter Directive maps filename extensions to the filters that will process responses from the Server as

AddOutputFilter filter [; filter...] extension [extension]

If more than one filter is specified, semicolons in the order in which they should process the content must separate them. Both the filter and extension arguments are case-insensitive, and the extension may be specified with or without a leading dot. The AddOutputFilter directive maps the filename extension to the filters, which will process responses from the Server before they are sent to the Client. We have added AddOutputFilter REPLACE .html. After registration & addition of output filter, we have accessed data by calling apr_bucket_read() function. The brigade serves to enable flexible and efficient manipulation of data, and is the unit that gets passed to and from your filter. In Apache filter terminology; each chunk is stored in a bucket, and lists of buckets form brigades. So for each brigade, read bucket data by using apr_bucket_read(). Implementation has shown in the following flowchart figure.2.

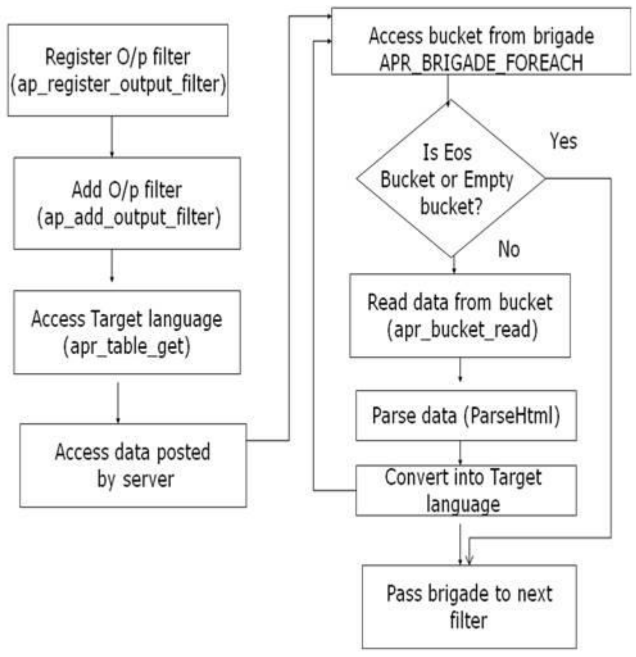

Register O/p filter (ap_register_output_filter)
Add O/p filter (ap_add_output_filter)
Access Target language (apr_table_get)
Access data posted by server
Access bucket from brigade APR_BRIGADE_FOREACH
Is Eos Bucket or Empty bucket?
Yes
No
Read data from bucket (apr_bucket_read)
Parse data (ParseHtml)
Convert into Target language
Pass brigade to next filter

Figure 2. Implementation Flowchart

3.4 Checking Target Language

In Web page interface Localisation for translating Web pages into target language on the fly, it is required to access “Accept- Language” header, which Client sent as target language for request. The request_rec structure is used to hold information about a current request between the Web Server and a Client. The request_rec structure is key to Apache's processing of Client requests. A pointer to a request_req structure is passed to all phase handlers and many API routines[14].

The request_rec structure describes a particular output filter by using “AddOutputFilter” directive in behalf of a Client. In most cases, each connection to the Client generates only one request_rec structure. The request_rec contains pointers to a resource pool, which will be cleared when the Server is finished handling the request; to structures containing per-Server and per-connection information, and most importantly, information on the request itself.

The most important such information is a small set of character strings describing attributes of the object being requested, including its URI, filename, content-type and content-encoding (these being filled in by the translation and type-check handlers which handle the request, respectively). Other commonly used data items are tables giving the MIME headers on the Client's original request, MIME headers to be sent back with the response. These tables are manipulated using the table_get and table_set routines.

The request_rec structure contains information about incoming HTTP headers in the form of table as table *headers_in. This table contains a key/value pair for every field in the request header. Some of the fields are also represented in other ways, such as the Range field, but all of the original request header fields are stored in this table in their raw form [13]. We have accessed “Accept-Language” header with the help of apr_table_get() function from request_rec structure for this table.

3.5 Parsing and Translating Response Data

The response from the Web Server to an HTTP request contains a header and usually the actual response. The header contains status information and information on the resource. We have first accessed actual response data and then we parsed text strings that need replacing in the response data [16]. The text strings are then replaced with corresponding target language strings if available in dictionary otherwise only original strings are sent in response data. We have created a dictionary of 300 multilingual (English-Hindi-Marathi) labels of commercial web based interactive applications.

4. EXPERIMENTATION

We have developed Web page interface Localisation for translating existing Web pages into Devanagari language on the fly. This software have been tested and verified for some Web pages for target language Devanagari.

Following are the details of modules incorporated for translation into target language.

apr_bucket_read(): To obtain response data in string from bucket in order to parse and translate data into corresponding target language.

APR_BRIGADE_FOREACH(): To obtain each bucket from brigade in order to perform operations on each bucket.

apr_get_table(): To obtain the value of “Accept-Language” header i.e. to obtain target language.

StoreDict(): To store corresponding target language strings to English strings in dictionary.

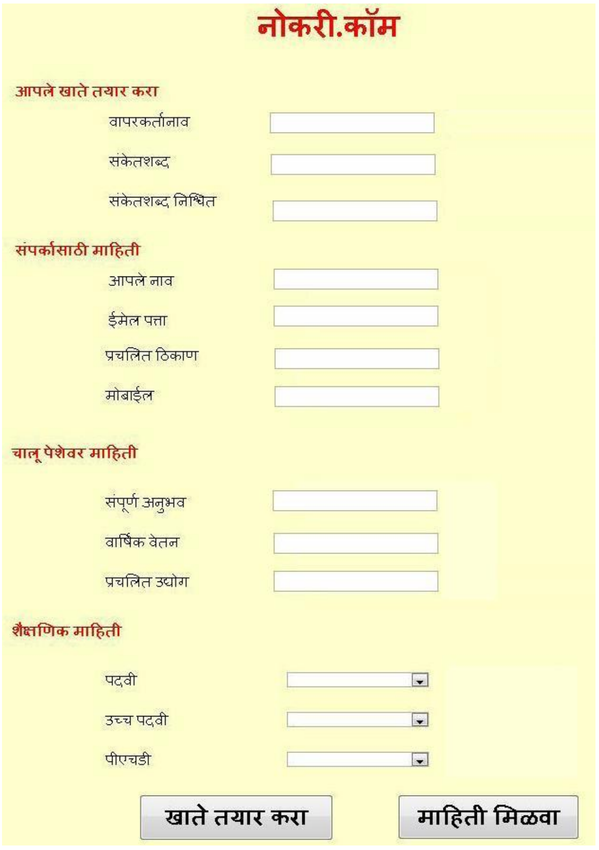
ParseHtml(): To parse response HTML data in order to retrieve strings that are required to be converted. These strings are then passed to StringConvert (). Thus response data is translated into corresponding target language.

StringConvert(): To search target language string for English string in order to return target language string to ParseHtml().

During experimentation, we simultaneously observed translation of Web pages for target languages Hindi and Marathi.

5. RESULTS AND FINDINGS

Our implementation is for the domain specific applications. It is found that there are about 300 hundred labels which are commonly used. We have preferred memory based translation using translation memory. The database is created by using the XML. Firstly, the dictionary is prepared for these labels using hash function with chaining where the labels hashed to the same location are chained with the link list. Then it is converted into XML database. As the dictionary size is too small and hash functions provide direct mapping the recall is 100 percent [15]. Web page interface localisation is the path-breaking solution for language localisation of the web pages, which provides effective user-friendly interface by translating the web pages stored on the web server to the target language on the fly. Here we have shown the output of web page interface into Hindi and Marathi. We have carried out our experimentation for seven interactive websites related with the banking and job search portals. One example from experimentation is shown in figure 3.


नोकरी.कॉम

आपले खाते तयार करा

वापरकर्तानाव

संकेतशब्द

संकेतशब्द निश्चित

संपर्कासाठी माहिती

आपले नाव

ईमेल पत्ता

प्रचलित ठिकाण

मोबाईल

चालू पेशेवर माहिती

संपूर्ण अनुभव

वार्षिक वेतन

प्रचलित उद्योग

शैक्षणिक माहिती

पदवी

उच्च पदवी

पीएचडी

खाते तयार करा

माहिती मिळवा

Figure 3. Web Page Interface in Marathi

6. CONCLUSION

We proposed efficient Web page interface localisation method for translating existing Web pages into any target language on the fly. Web page interface localisation is the proposed solution for the growing demand of non-English language interface for the Web pages. If someone has existing Web pages, or is developing a separate site for a target market, Web page interface Localisation is essential to preserve and convey the right message to target audience e.g. online banking, online shopping organizations. The results of experimentation show that the software works well in the sense that the existing Web pages are dynamically translated into target languages like Hindi and Marathi.

7. ACKNOWLEDGMENTS

We express our gratitude to Prof. Abhyankar H K Former Director of Vishwakarma Institute of Technology, for his continuous encouragement to carry out the research in the area of localisation. We also thank to Jayshree, Sanjesh, Vaishali and Viren for contributing the research work in the area of localisation at our institute. We also thank Disha, Ruchi, and Jyoti Dhore for their help in creating the multilingual dictionary. Special thanks to M R Dube, Assistant Professor who has strengthened the linguistic of this paper.

8. REFERENCES

[1] Frost & Sullivan, Local Language Information Technology Market in India, TDIL, Department of IT, Ministry of Communications and Information Technology, India, 2003.

[2] John Hutchins, University of East Anglia, Norwich , “Current commercial machine translation systems and computer-based translation tools: system types and their uses” http://ourworld.compuserve/homepages

[3] R. Sharma, “Development of a bilingual electronic glossary for automated assistance in user interface localisation”, TIDL,2002.

[4] J. Shah, S. Hajare, “Localisation for e-governance”, Journal of Language Technology, 2004, pp. 154-160.

[5] Stavros kokkotos and constantine spyropoulos, An architecture for designing internationalized software," Software Technology and Engineering Practice, pp. 13-1,July,1997

[6] T. Parr, Web application internationalization and localisation in action, in 6th International conference on Web engineering, vol. 263, pp. 64 70, ACM New York, NY,USA,2006

[7] N. Anbarasan, Software localisation process and issues," Tamil Internet, 2003. .J. Cardenosa, C. Gallardo, and A. Martin, Internationalization and localisation after system development: A practical case, International Journal of Information Technologies and Knowledge, vol. Vol.1, pp. 121-127, 2007.

[8] Cardenosa, J., Gallardo, C., and Martin, A. Internationalization and localisation after system development : A practical case. International Journal of Information Technologies and knowledge. Vol 1, 121-127, 2007

[9] D. Chakrabarti, D. Narayan, P. Pandey, P. Bhattacharyya, “Experiences in building the Indo Wordnet- A wordnet for Hindi”, In, Proceedings of the First Global Word Net Conference. Central Institute of Indian Languages, Mysore, 2002, pp. 57-64.

[10] S. Dave, J. Parikh, P. Bhattacharyya, “Interlingua-based English–Hindi Machine Translation and Language Divergence”, Journal of Machine Translation, Volume 17, September, 2002.

[11] The Apache HTTP Server Project Fielding, R.T; Kaiser, G: Volume 1, Issue 4, July-Aug 1997 pp: 88-90.

[12] Apache HTTP Server Documentation httpd.apache.org/docs, httpd.apache.org/docs-2.0/modules

[13] www.onlamp.com (Writing Filters in Apache, Writing Output Filters in Apache, Writing Input Filters in Apache)

[14] Design considerations for the Apache Server API, Robert Thau, Fifth International World Wide Web Conference, 1996, Paris.

[15] M L Dhore, Jayshri Khachane ,M.R. Dube, A. M. Kulkarni , “Web Page Localisation for Sites Hosted on Linux”, IAENG International Conference on Internet Computing & Web Services, Hong Kong, March 2007

[16] M. L. Dhore,A. K. Dhote, “Automating the HTML Localisation Process: An Implementation Using a Java Internalisation Approach,”, 11th Annual Localisation and Internationalisation Conference organized by LRC, 2006 http://www.localisation.ie.