Enhancing open-source localization

A Writing Project

Presented to

The Faculty of computer Science

San Jose State University

In Partial Fulfillment of the Requirement for the Degree

Master of Science

By

Farzana Forhad

May 2010

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Pollett

Dr. Robert Chun

Frank Butt

ABSTRACT

Pootle is a web portal which is designed to be a web translation tool. We can run Pootle like an internet server and run a local copy on an intranet. Pootle has a user friendly interface which ensures better quality and easier translation of projects. Users can log into the Pootle and create their own projects under any language, and they can also add or retrieve any language from any project or any project from any language. Although Pootle has many advantages, it also has its drawbacks, and one of the biggest drawbacks of Pootle is that whatever translation is made by the user remains in their local machine. Whenever Pootle is installed, users have to do everything from scratch. In this project this issue has been addressed, and a smart solution is provided for the users. An extension is made from the Pootle server to the svn repository which is an open source repository. With the implementation of this extension each user will be able to download all the works or translations that have been created so far, and they can also add their own work to the repository to make it useful globally.

ACKNOWLEDGEMENTS

I would like to extend my deepest appreciation to all the people who made the completion of this project possible. I would specially like to thank my advisor Professor Chris Pollett for his enthusiastic support throughout this work. Without his encouragement and guidance, completion of this project would have been quite difficult.

I would also like to acknowledge my committee members, which included Professor Robert Chun and Professor Frank Butt for their taking the time to read this report and giving me their comments. I would also like to thank my husband and my parents whose encouragement made my advanced study possible.

TABLE OF CONTENTS

INTRODUCTION	9
The Project	10
Report Overview	10
Comparing with Exiting tools	11
TECHNOLOGY USED	12
Gettext	12
Potable Object	12
Subversion and Beanstalk	13
CSS and Javascript	13
PRELIMINARY WORK	14
DESIGN	19
IMPLEMENTATION DETAILS	21
Setting up the directory structure	21
User interface	24
Backend	26
Testing the Software	31
Usability testing	32
Performance Testing	34
Robustness	38
Conclusion	40
Bibliography	39

INDEX OF FIGURES

FIGURE 1: .po input file in poEdit 15

FIGURE 2: Result of testing out localization with gettext 15

FIGURE 3: Database table setup specification 15

FIGURE 4: User interface for entering localized_string to the database 16

FIGURE 5: Screen shot of updated database information 17

FIGURE 6: Directory structure for the front end files 21

FIGURE 7: Directory structure of checkin.php 22

FIGURE 8: Directory structure of checkout.php 23

FIGURE 9: User interaction form 25

FIGURE 10: The result of the checkout.php 25

FIGURE 11: The change of interface after svn checkout 28

FIGURE 12: The result of the checkin.php 28

FIGURE 13: A sample code for svn connectivity 32

FIGURE 14: Manual process of checkout and generating ".po" file 35

FIGURE 15: checkout and generating ".po"file with extension 35

FIGURE 16: One of the steps to manually conn to svn repository 36

FIGURE17: Showing time difference with and without the form 37

FIGURE 18: Shows how new extension gets blended 38

INDEX OF LISTINGS

LISTING 1: PHP code for test out localization with gettext 14

LISTING 2: Sample code for backend support 17

LISTING 3: Part of HTML code for UI 24

INTRODUCTION

Overview

Many non-English speakers are more comfortable browsing the web in their mother tongue rather than English. Most of the time, they like to see web pages in their own language, such as Chinese, Japanese, Spanish, and so on. There are different kinds of software which have been used for translating the web pages. The particular kind of software which displays text in a user's local/native language is called localization (l10n) software. Related to localization, internationalization (i18n) is used to design the software application in such a way so that it could be used in any languages without doing any sort of engineering changes to it. An important part of localization and internationalization is “gettext” which is an open source tool for internationalization. Translating strings into different languages using “gettext” tools takes several steps. The goal of this project is to enhance “gettext” so that it takes fewer steps for the engineers to localize the text. The tool “gettext” requires several steps to create web pages that work in any language. To translate the English version of web pages to other languages requires a compilation for each translation, which is very time consuming and expensive. The enhancement to “gettext” eliminates the need for any compilation in producing the English version of web pages yet still supports unique identifiers for page items so that they can be easily localized to other languages. Apart from localization the purpose of this project is also to enhance the dependency of database on localization.

Pootle has a translation tool for translating into different languages. After logging in to a pootle server we can search as a project or as a language. If we enter inside a project we

can see the list of all the languages that are supported by that project. We can add or delete any language. We can also upload any file from the local directory. And then we can translate and save it in the local directory of the local machine. [15][16][18]

The Project

Pootle is a translation tool which is used for localization. Pootle does not have any easy steps to quickly add any locale directory. Pootle also can not manage file systems other than its default directory and also works only with the static portable object files. Pootle as well as other leading tools, works only with ".po" files to manage localization. One big bottleneck in Pootle is that for translation it has to import every file from outside, and after translation the file needs to be exported. Our enhancement to Pootle would be to get rid of these problems and actually make it work as a dynamic locale data rather than static portable object files. Our extension to Pootle will allow the user to their work and save everything to an open source repository. We will enhance Pootle in such a way so that the files could be directly managed inside the Pootle server so that it does not have to be exported to be used.

Report Overview

The report starts with an overview of the gettext tool and explanation of functionalities that are present in the Pootle server. Next is comparison with similar existing tools, and the section that describes the various technologies that are used all over the project. Then we talk about what kind of research work we have done before starting our implementation. The Design and Implementation involves the process of creating the

extension. Then there are details about various testing. The report ends with a conclusion and bibliography.

Comparing with Existing Tools

Mozilla has an add-on called Verbatim which has a language support extension. Some of the important features for language support in Verbatim are the Quick Locate Switcher, FoxLingo Translator, abc Tajpu, Quick Translation, Dictionary Switcher, and Answers. The Quick Translation of the FoxLingo helps to switch to a different language quickly in a Mozilla application. FoxLingo translator helps to learn language, translate text, and auto translating web pages. Some of the other important features of FoxLingo translators is it can contain directories, text-to-speech engines, language identifiers, and language search features. This supports 71 languages and 31 free online translators. Dictionary Switcher can be used to switch between the installed dictionaries which can be displayed in the currently selected dictionary. [10]

On the other hand Launchpad can translate free software projects and also can distribute packages into one's own language. Using Launchpad is simple, and only needs an account to register and a web browser. There are no requirements for special software. But if we compare these tools, none of them really has any collaboration with online repository. The new add-on of pootle will give advantage to the user to save their work to an online repository and download them as a portable object format. After translating the strings they can upload their translated files with the button click. [9]

Technology Used

As the project targets to make an extension of the Pootle server, we took help from several tools and technology.

GetText

GetText is the library for GNU internationalization and localization. Usually it is used for writing multilingual programs. To translate any string from one language to another we must first generate a portable object version of that file. To identify those strings inside a file like in a PHP file, we use the gettext function. When we run the "xgettext" command on a specific file (having a gettext function in it) it creates a portable object file. [5]

Portable Object

Portable object files are textual and editable. It holds the relation between a original and string and corresponding translated string. A portable object file identifies the string that needs to be translated as "msgid" and the language we are translating it to as "msgstr". Portable object file is created after executing an "xgettext" on the original language file which can be in any language like PHP, HTML, and C. [14]

Subversion and Beanstalk

Subversion is an open source online repository. It keeps an original version of a folder inside the "trunk" and creates branches where we usually can keep our updated versions. As it saves everything with dates, we can easily track down what is updated when and by

whom to avoid any kind of confusion. Beanstalk is a web application which can make things simple when working with subversion. It is used for hosting svn repository. [11]

CSS & Javascript

Cascading Style sheet (or CSS) works with the HTML which helps to improve the look and formatting of the user interface. Javascript is a client side scripting language that can be used to enhance the functionality of HTML forms. For this project, we used Javascript functionalities to work with our HTML form. [7]

PRELIMINARY WORK

Extensive research was conducted for this project. Most of the research was performed in the CS297 during the preparation for writing project. This research helped me to learn all the tools required for this project. We also went through the exiting work that was done earlier. The preliminary work had multiple deliverables. The first deliverable helped me understand how the gettext works. Second deliverable was all about taking help from PHP to translate the strings. And the third one helped to understand the interaction between a simple user interface and an underlying database layer.

Simple research on gettext

The very first deliverable for this project was to test out localization with gettext providing my own input. The testing was performed by choosing a set of inputs against which the localization was verified.

The part of the PHP file is shown below:

```
<?php
// I18N support information here
$language = 'bn';
putenv("LANG=$language");
putenv("LC_ALL=$language");
setlocale(LC_ALL,"bn");
$domain = 'bn';
bindtextdomain($domain, "./locale/");
textdomain($domain);
//the input strings goes like this
echo _("I stood in the wind ");
?>
```

Listing 1: PHP code for test out localization with gettext

The above listing explains how to setup the environment by “putenv”, set up the locale information by “setlocale.” The double underscore followed by the echo is the short hand of gettext function. Performing a “xgettext” operation on this file a “.po” file got generated. [5] These translated strings looked something like this in the poedit editor. The “.po” file looks like this:

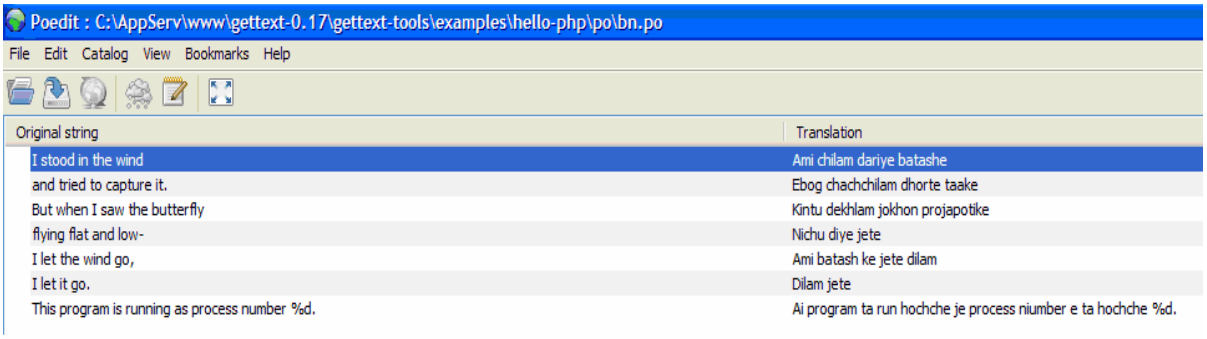

Figure 1: “.po” input file in poEdit editor

The “.po” files were stored in the “locale” directory and “msgfmt” operation was performed on those .po file to make a browser readable view.

The output looks like this:

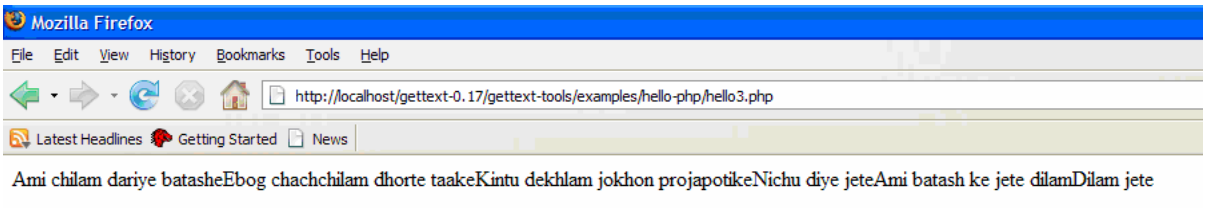

Figure 2: Result of testing out localization with gettext

Creating an example with PHP and gettext

The second deliverable was about doing some minor modification on gettext code. To carry on this research I did some modification in the gettext code. I took the original gettext php code and modified the strings. I used PHP "$_SESSION" variable to read the string from the gettext file. Then I tested the code to generate "pig-latinized (an English language game)" strings.

Front-end and Back-end interaction

The third deliverable was about performing some simple experiments to use the dynamic localization scheme based on database tables.

To do that, I set up a database table by following the some specification like:

```
+------------------+------------------+------+-----+---------+-------+
| Field            | Type             | Null | Key | Default | Extra |
+------------------+------------------+------+-----+---------+-------+
| id               | int(11) unsigned |      | PRI | 0       |       |
| locale           | varchar(10)      |      | PRI |         |       |
| localized_string | text             | yes  |     | NULL    |       |
+------------------+------------------+------+-----+---------+-------+
```

Figure 3: database table setup specification

The database table was setup for the backend support. For the front end I made a UI which takes strings to update the database.

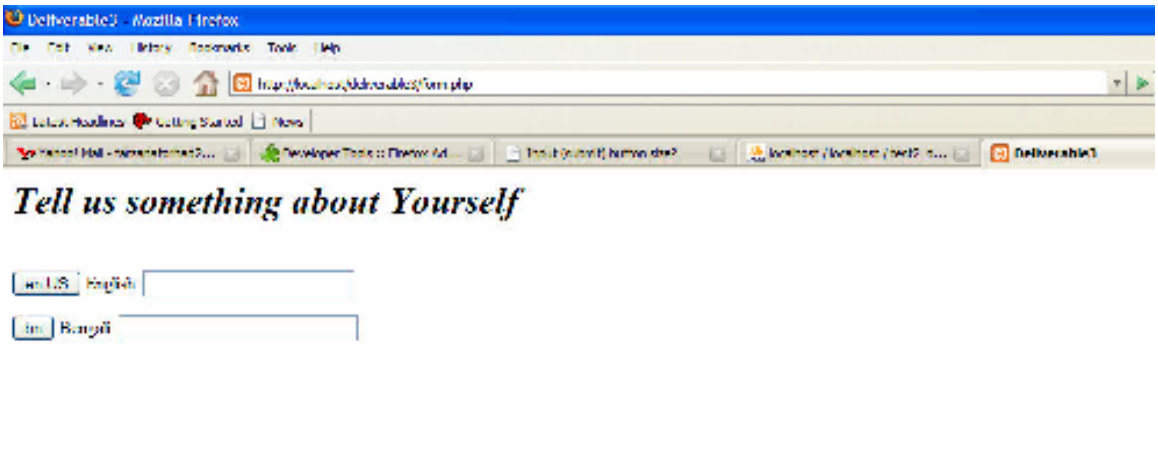

Figure 4: User Interface for entering localized_string to the database

After taking the string input it can store it in the database under different locale with a unique ID. And also shows update information in the browser window. The sample PHP code looks like this:

```
mysql_connect("localhost","root","tictac44") or die ('Error: ' .mysql_error());
mysql_select_db("test2_deli3");

$query="INSERT INTO TestTable (id, locale, localized_string)VALUES
('NULL','en-US','".$localized_string."')";
mysql_query($query) or die ('Error updating database');

echo "Database Updated With (en-US): " .$localized_string;
```

Listing 2: The sample PHP code for backend support

After performing the operation the output in the browser window shows the updated information in the screen which looks like this:

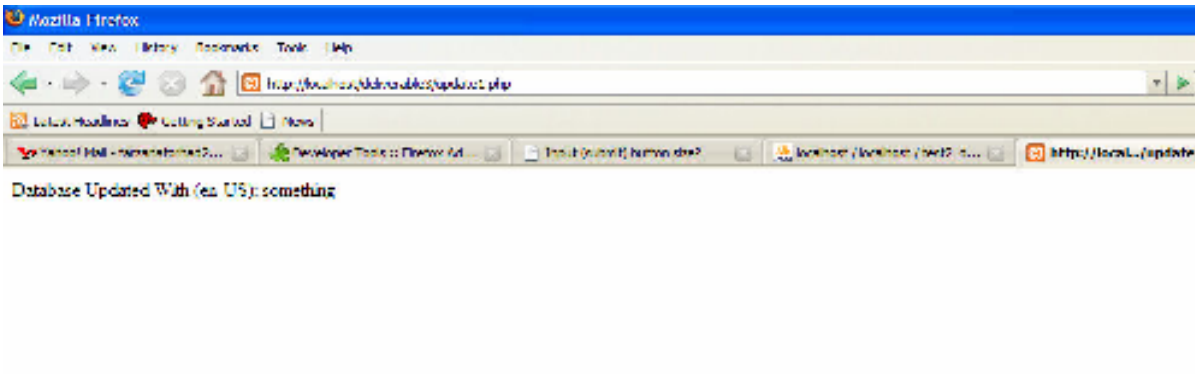

Figure 5: Screen shot for updated database information

By performing this operation the database updated with a unique id, locale value and strings which were taken from the users. The strings are stored in the database updating the id and locale in the database.

DESIGN

Pootle server is installed in a local machine. It has feature that is used for translating string into multiple languages. We worked both with back end and front end while making the extension to the Pootle server.

First of all we had to make a user friendly GUI that helped the user to make a connection with the subversion repository with a button click to checkout the current folders and files from the repository.

And then the front end html form calls some of the PHP functions to make the connection and finally user can see everything in the same page.

The Front End (User Interface)

Our extension to Pootle will allow the user to modify and save everything to an open source repository. For this, some specific information of the user is required, like the url, log in, and password. This information is required to connect to the svn repository. Some other information like what kind of file user wants to check out, which file they want to checkout is also required since we are providing them the portable object format of the file which the user have to name it. All these tasks can be achieved by simply clicking a button and this has been taken care with the help of one HTML form.

The form is responsible for taking the url, login and password information to connect to the svn repository and also which file they want to upload to the svn repository. [2]

The Back End

The two front end forms actually calls some set of functions in the back-end. When the user presses the “svn chekout” button the following things take place:

- it gets connected to the subversion repository.
- Downloads the specific file
- Make a portable object version of the file
- Uploads it to the pootle server

And whenever the user presses the “svn checkin” the following things happen:

A specific file gets uploaded to the sub version repository with a log message.

Implementation Details

To make the extension of pootle we had to learn the directory structure of pootle. Then according to that we started our implementation.

Setting Up the Directory Structure

It is required to setup the directory structure for the extension. For the front end we put our code inside an existing html. Below is the directory structure of the html file:

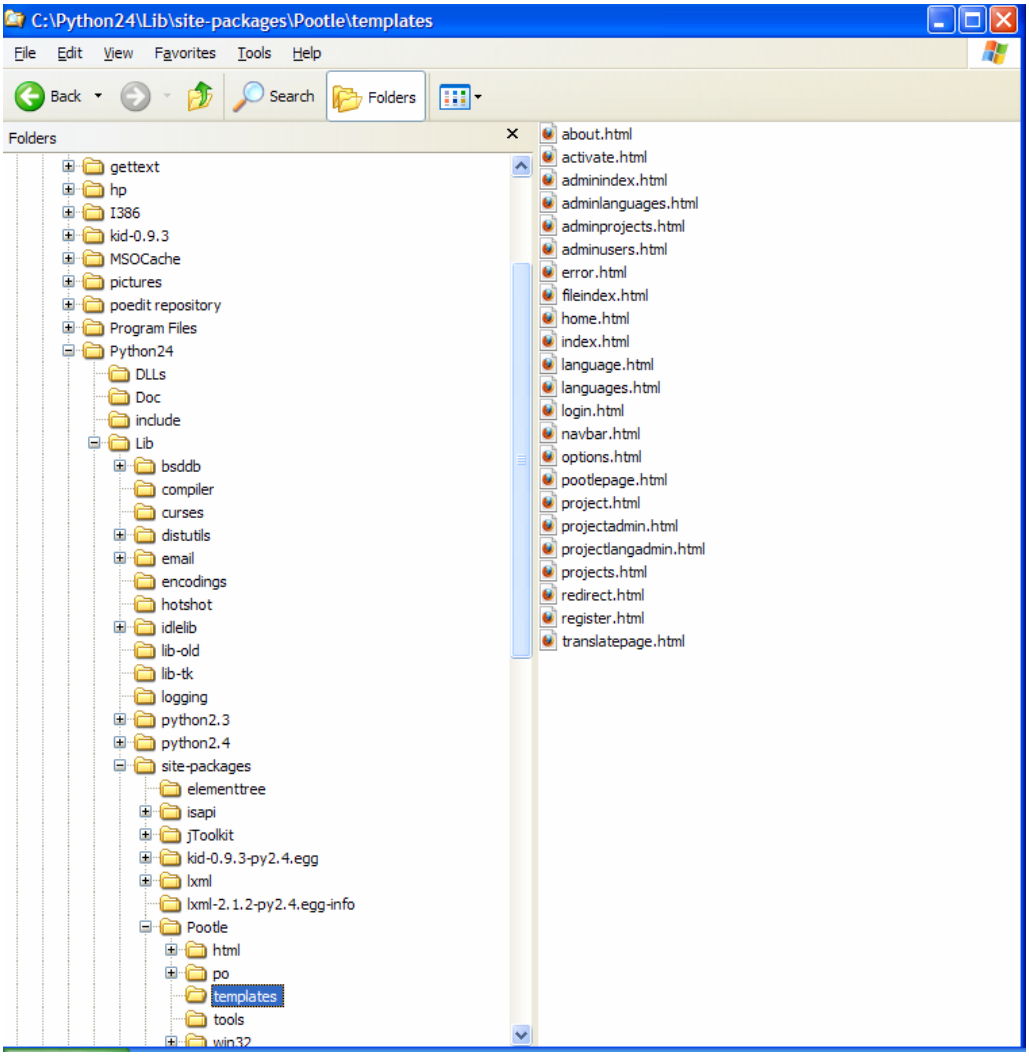

Figure 6: Directory structure for the front end file

We wanted our user interface to be blended with the current Pootle server. We also did not want the user to redirect to any special page to use our extension. We found an appropriate place to put our user interacting form. Pootle server allows the user to upload their translatable files to a specific location. Users have to first select a project and select a language and after that they can upload their file for translation. We took the same spot for our extension.

For the back-end support we had to have two separate PHP files one for svn check out and one for svn check. These two files are executed from the local host of the user.

The directory structure for the “checkin.php” is below:

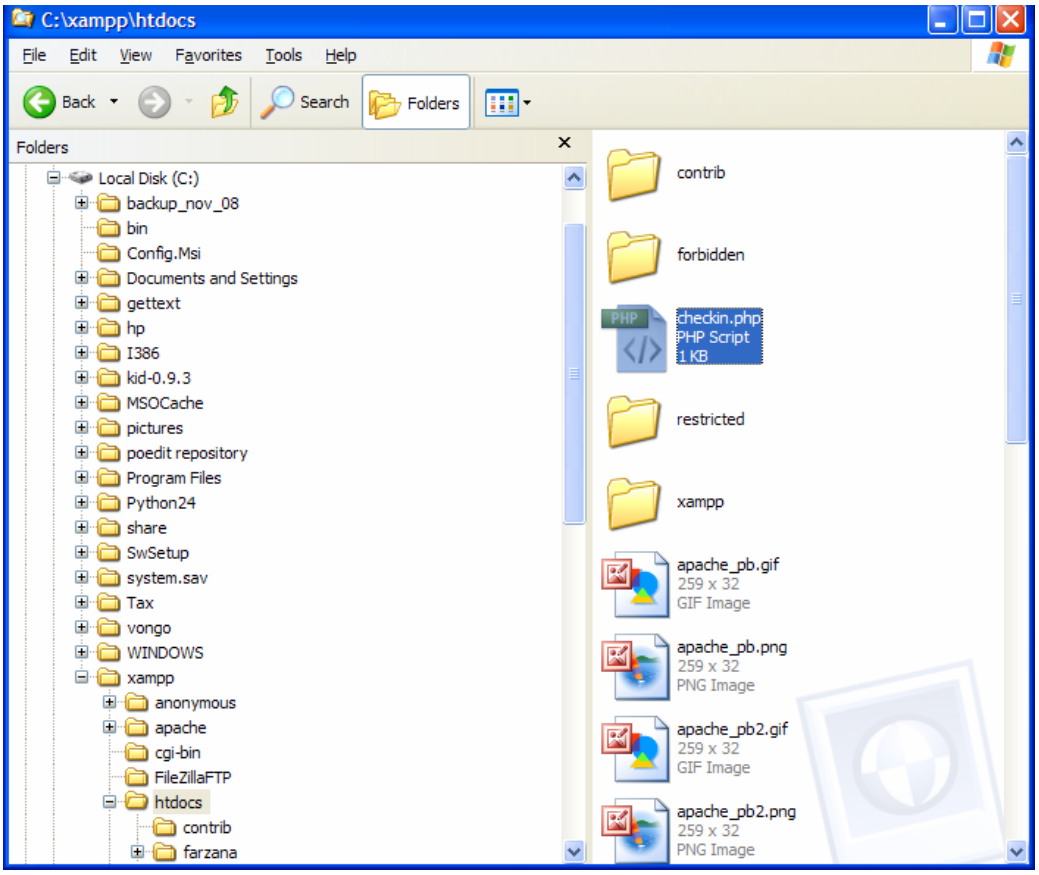

Figure 7: Directory structure of “checkin.php”

The directory structure for the “checkout.php” is below:

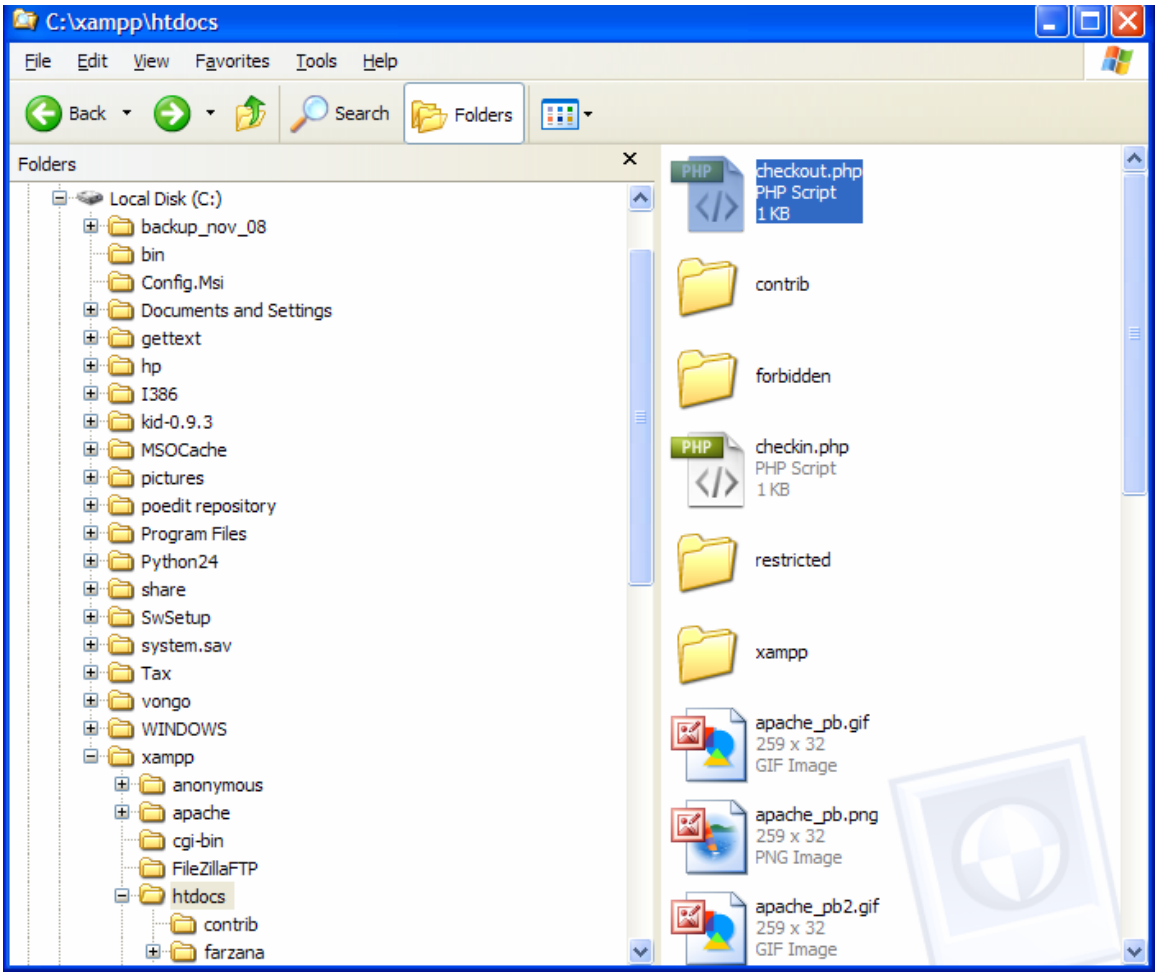

Figure 8: Directory structure of “checkout.php”

Creating the User Interface

To keep everything simpler we made the user interface very neat. The user will be asked very minimum information to make the extension work. This form takes exactly five different text fields type of user inputs and two submit buttons.

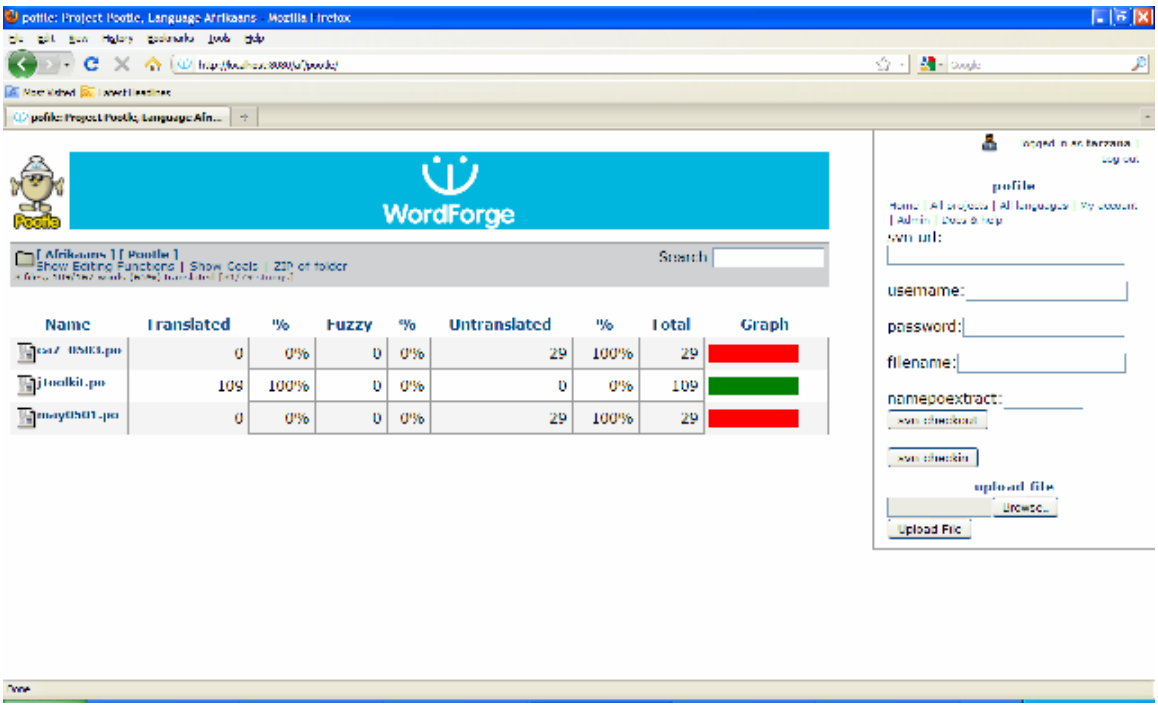

Figure 9: User interaction form

The first three text fields the information about subversion url, login and password information to connect to the svn repository. The next two text fields takes inputs of: which file they want to download and what they want to name the portable object file. Then there are two buttons: one is for svn checkout and the other one is for svn check in.

Here is a small part of the html code for this form:

```
…
svn url: <input type = "text" name="svnUrl" size="35"/><br /><br />
username:<input type="text" name="userName" size="23"
method="post"/><br /><br />
password:<input type="text" name="passWord" size="23"
method="post"/><br /><br />
 namepofile:<input type="text" name="namePoFile" size="5"
method="post"/><br /><br />
projectname:<input type="text" name="projectName" size="20"
method="post"/><br /><br />
languagename:<input type="text" name="languageName" size="17"
method="post"/><br /><br />
…
```

Listing 3: Part of html code for user interface

The form takes inputs from the user exactly once but makes it hidden for the check out and check in.

The Backend

The back-end is designed to make necessary connection to the svn repository, download files from there, and perform “xgettext” command to make a portable object file. It also supports uploading translated files to the svn repository. PHP has its own extension to bind it to subversion which allows PHP script to communicate with SVN repositories. When ever a user presses the “svn checkout” button the following happens at the back-end. [3]

1. A consolidated portable object file generates form the requested files such as, PHP, and HTML.
2. The portable object file gets saved in the Pootle project and current page on the Pootle server gets refreshed.

The following is a directory where the portable object file is stored:

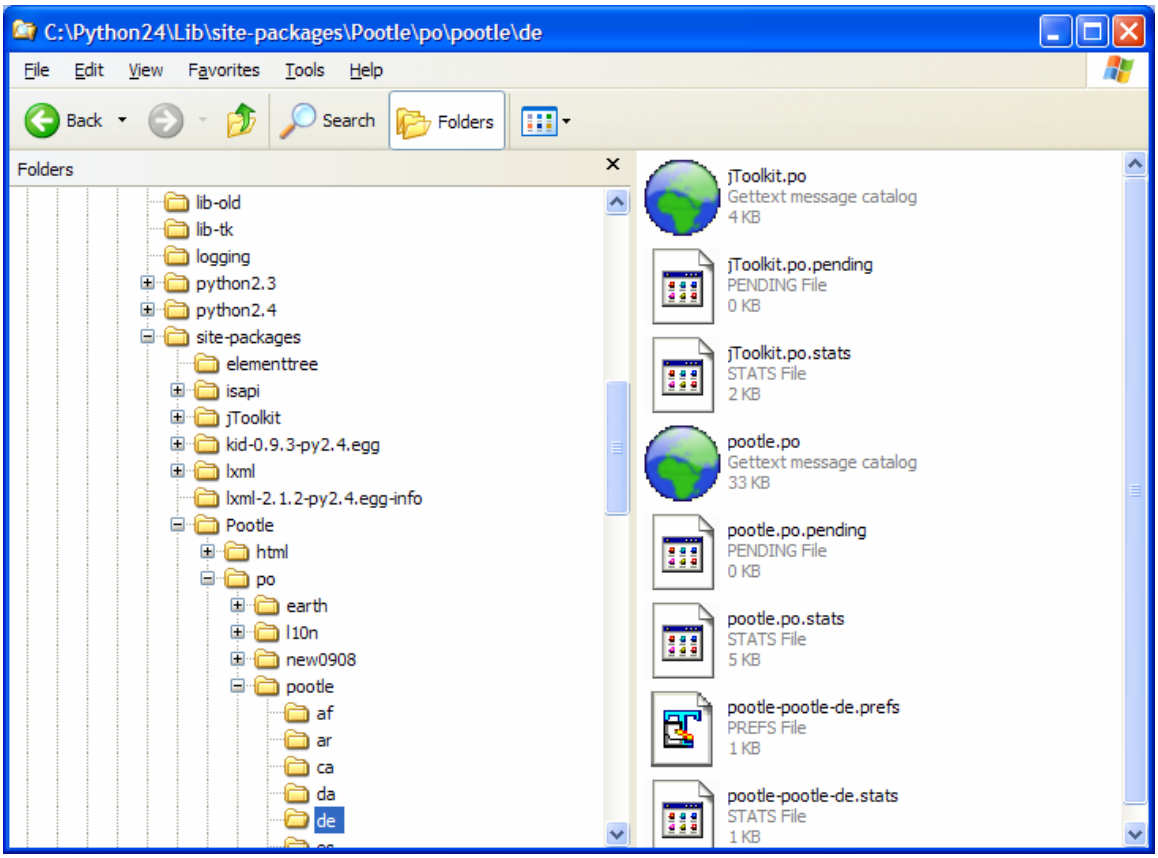

Figure 10: the result of the checkout.php

To implement these functionalities we took help form the PHP's SVN extension functions. Below is a brief description of those functions: The svn_auth_set_parameter() sets an authentication parameter. This function takes a string input and it ensures the property for username or password information to use when performing basic authentication. The “svn_checkout” is responsible for checking out the working copy of a repository. It takes two string parameters; one of them is the name of the URL of the destination and the other is the name of the name of the subdirectory. Next the “exec” function is called to execute an external program. It takes two parameters as input. The

first one is a string parameter which is nothing but the command that will be executed. To get the output of the executed command, there is an output parameter which needs to be set. The copy() makes a copy of the file from source to the destination. The first string parameter is the source path of the file whereas second one is the destination path of the file. The header() function is used to send the information of the HTTP header. Header() must be called before any actual function is sent. This function not only sends the header back to the browser but also returns a redirect (302) status code to the browser if the request fails or 201 status code if it has already sent. [4]

When the user presses the checkout button, he or she can see the output in the same page of the project or even in the directory.

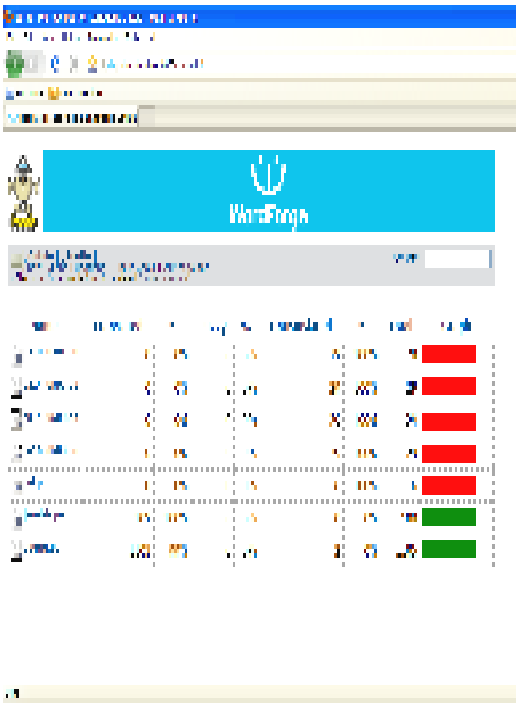

Figure 11: The change update in interface after svn checkout

Above was the directory structure, user also can see the currently downloaded file in the current page.

Whenever the user selects the “svn check-in” button, the updated and translated files get uploaded to the SVN repository.

For svn check in the copy() makes a copy of the file from source to the destination. The first string parameter is the source path of the file whereas second one is the destination path of the file. The “svn_add” function is responsible for scheduling the addition of an item in a working directory. It adds the file or the directory to the working directory. The item (directory or the file) will be added to the repository at the next time svn_commit() on this working copy. The parameter recursive is set as true by defaults, which recursively add all of its contents. If it set to “false” subversion will recurs into already versioned directories. The “svn_commit” function send changes from the working copy to the repository. It also generates a default log message whenever subversion check out is done. [4]

The following is a directory where the portable object file is stored:

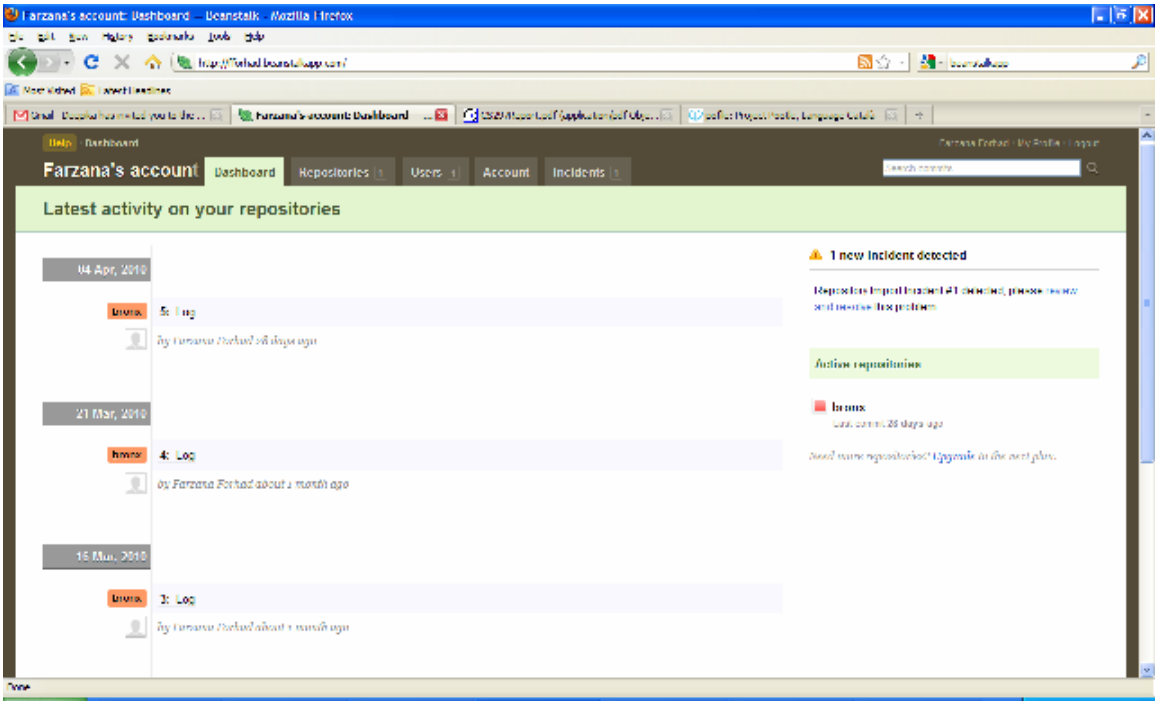

Figure 12: The result of the checkin.php

When the user presses the checkout button, he or she can see the output in the same page of the project or even in the directory.

Testing the Software

Software testing is a method, which verify and validate a software program and make sure it works as expected and satisfy the technical requirements.

There are risks implementing software and testing it will provide the user an objective and independent view of understanding the software and the risks associated with it. Therefore, it is very important to test software. We vigorously tested the pootle server extension that we made and it was tested in the following areas:

- making connection to the svn repository
- checking out files that are requested
- conversion of the files to a portable object file
- uploading the generated portable object to the pootle server
- putting the svn files back to the svn repository.

We also performed separate tests to make sure each of the features of the extension works properly.

In one instance, we found out that the files were not getting uploaded to the svn repository. We checked our system and found no error. Then we manually logged in to the svn repository and found out that they were going through a data transferring work which caused this instance. Once they were finished with data transferring there were no issue with uploading files anymore.

Usability Testing

Two users from Bangladesh, Partha Pratim Saha and Ali Akram from World Health Organization's uses pootle. They use it for translating WHO's documents from English to Bengali and to save the translated documents. It helped them a lot and saves them a lot of time since they can use one single source and destination to save or retrieve their translation. Now with this extension this all can be done by clicking a button and for this we did have to think about how the .po file is getting generated.

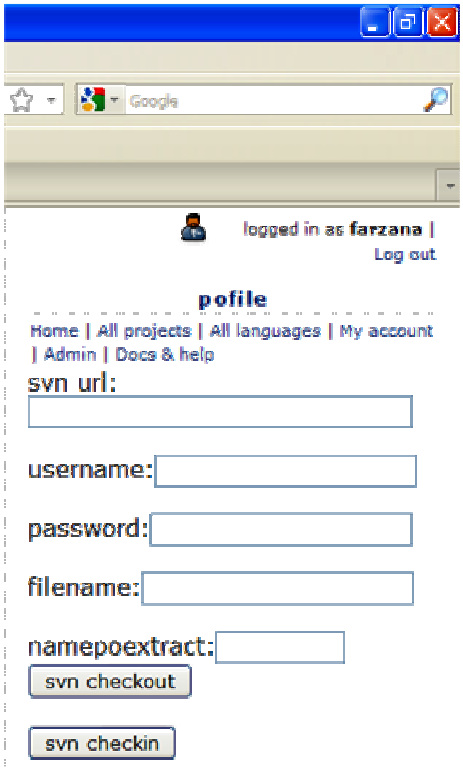

Figure 13: A simple UI for svn connection

The other two user of my project is form Australia. They are Lia Chakma and Shams Mawdud. They are looking for their masters projects and they are trying to work on localization.

After using the software they found out that same connectivity information is required for svn checkout and checkin which is not very efficient. Their response to this issue leads me to improve the extension which allows the two forms to blend together. Now the connectivity information is asked once. We made two separate svn checkout and checkin button and the user can press either of the button to get their job done.

Performance Testing

The purpose of this project was to make Pootle server more user friendly so that it can have stable performance. The performance of the pootle server was tested with and without the extension for both svn check-out and check-in.

First we tested the pootle server without the extension for checkout using the existing facility. For this, we created a project inside the Pootle server. Then we Logged in to my subversion form outside pootle, did svn check-out.

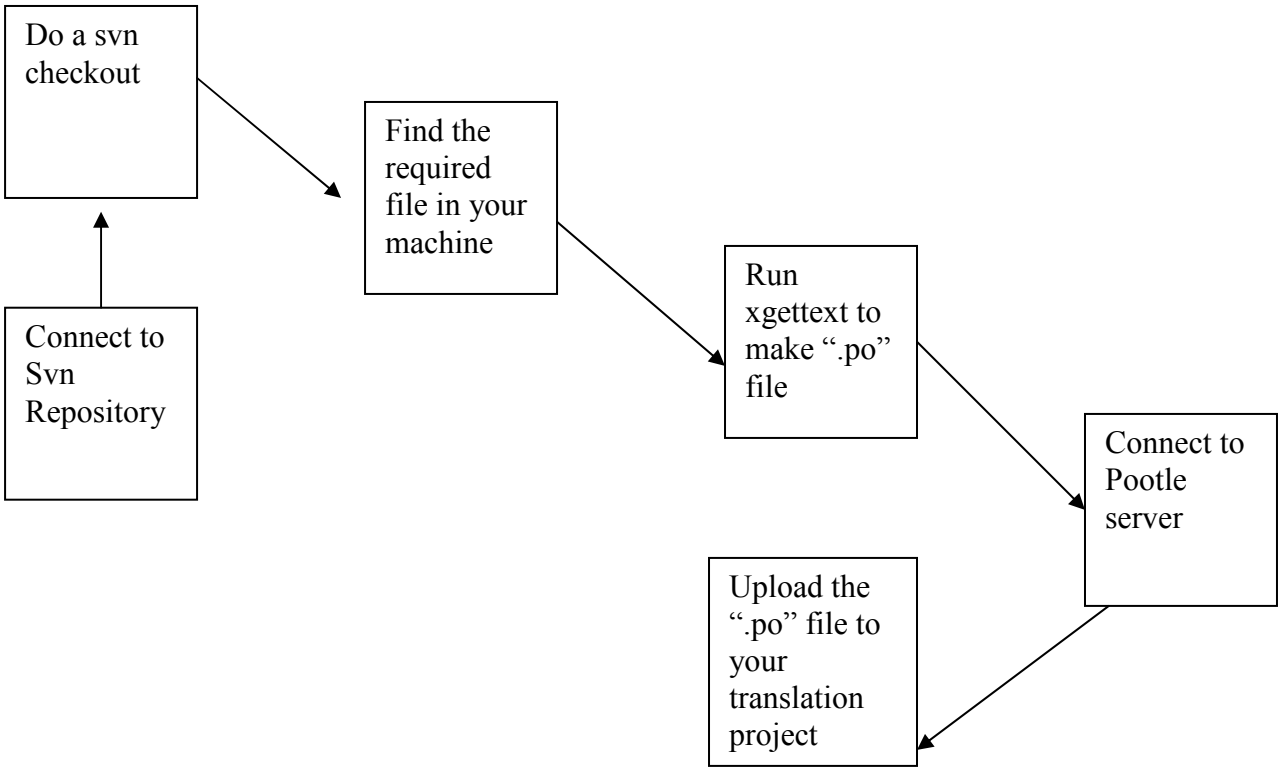

Figure 14: Manual process of checkout and generating .po file

Then we ran the xgettext command to generate the portable object file. Then came back to the Pootle project and uploaded my .po file. The whole process took approximately seven minutes from start to finish. After that we used the extension that we made and all the above process was performed by clicking a button and it took more or less three minutes.

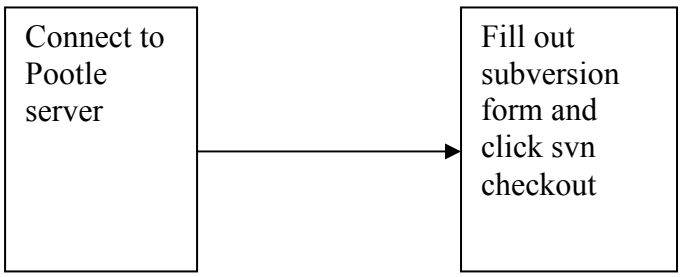

Figure 15: Checkout and generating .po file with the extension

For svn check-in we used the control panel and manually checked in or uploaded the translated file to the svn repository. From start to finish the whole process took us approximately one and a half minutes. But using the extension it took approximately one minute to upload the file to the svn repository. All we had to do is choose the portable object file and click the svn check-in button.

The figure below shows few of the steps of svn connectivity.

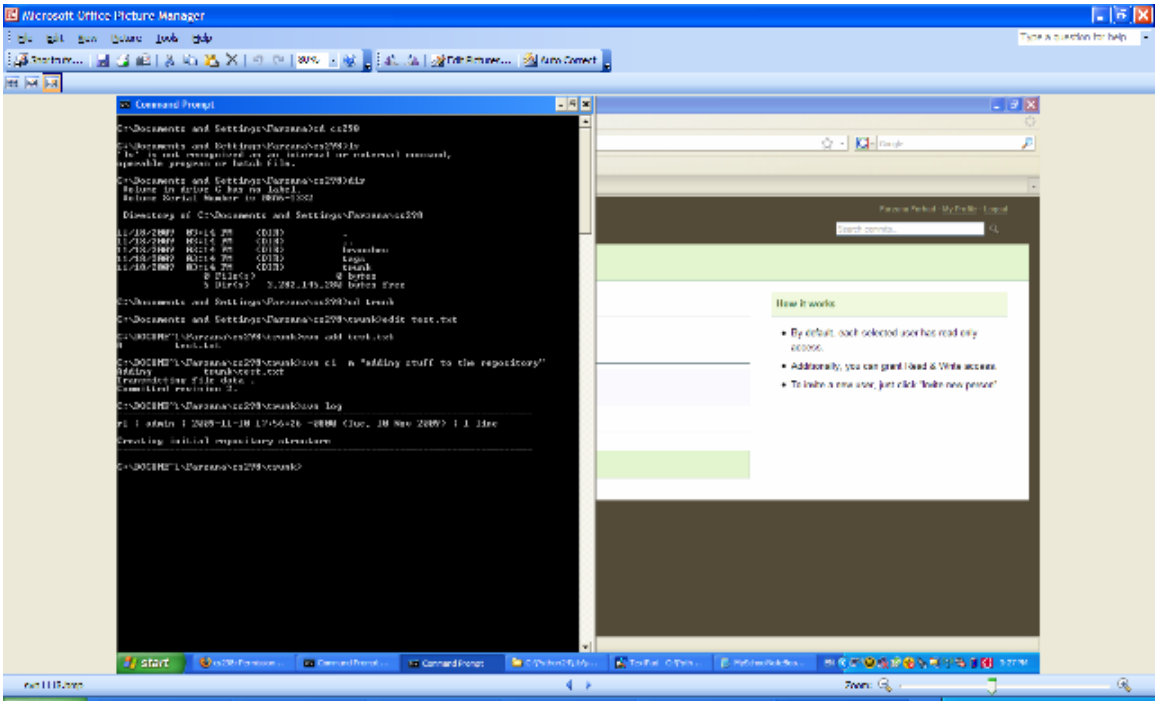

Figure 16: One of the steps to manually connect to svn repository

At the very beginning we created two separate forms to interact with users, one is for svn check out and one is for svn check in. The performance was tested based on the functionality of these svn check-out and svn check-in forms with and without the extension. The extension reduced the total time for svn check-in and check-out to four minutes from eight and half minutes. Although the extension helped to reduce the time significantly, we decided to further optimize this time. We found out that a user has to fill out two separate forms to complete the svn check-out and check-in functionalities, which is not a very efficient way. So we decided to use one single form to share the same

information for svn check-out and check-in. Now the total execution time for the entire process came down to approximately three minutes.

The response of this extension was also studied for high loads. This situation may occur when multiple users tries to use our extension. We tested the extension under different stress level: one user, five users, and twenty five users. We took the help of selenium's record and playback feature. In selenium we edited the umber of users each time to perform our testing. For all cases it took approximately three minutes to do a check out and check in to and from the svn repository. [11]

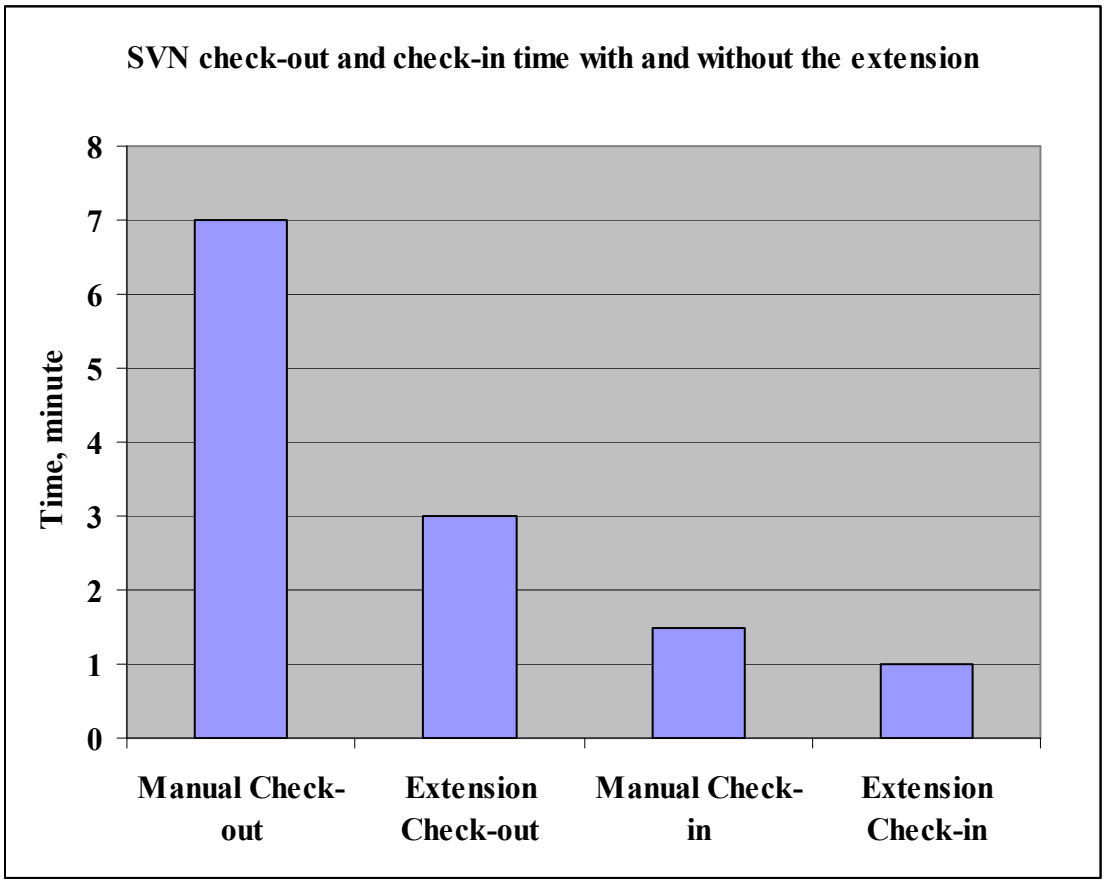

Figure17: Showing time difference with and without the form

Robustness

The pootle server software is written in Python. The software mainly provides some important features for localization; such as translation memory, alternative source language, version control, user management, translation interface and all these features were implemented in Python. For the user interface other than HTML, KID is used which is nothing but a XML based template language.

Our extension to Pootle is written in PHP and all the interface work is written in HTML. We did not have any issues on promoting the PHP code in pootle server whereas the main software is written in Python. Figure below is taken form pootle server page.

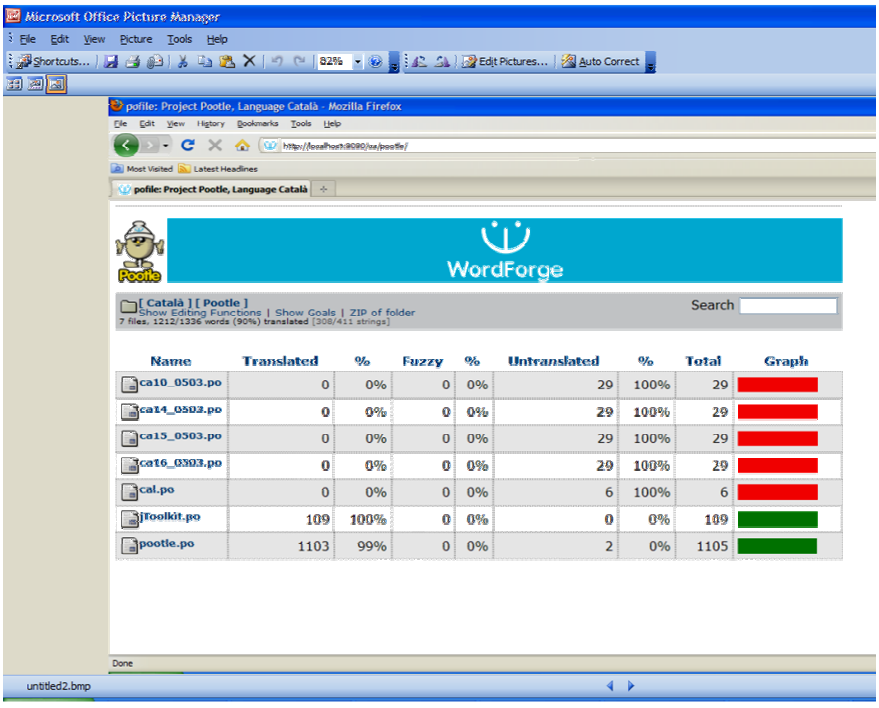

Figure 18: Shows how new extensions blended with the existing one.

After having the extension to the pootle a number of files were uploaded. The new files that were uploaded easily picked up all the remaining features.

Figure 16 shows the recently uploaded file. This file shows how much translation is made to those file and shows the amount (percentage) of work that is done. Any file that is just uploaded shows 100% translation left and if we translate the files and add strings to them the projection changes immediately. Any changes to these files are stored in Pootle server's main database. If we ever want to delete these files or want to add them to any other project of the Pootle server it works fine like any file which is uploaded by Pootle server's own file upload function. The files that are responsible for those projections are written in python whereas our extension to pootle is in PHP.

The files that are uploaded by our extension works exactly like the files that are uploaded by Pootle server's basic upload function. Other than adding translated strings the Pootle is designed very efficiently at the ground level. Therefore, it was possible to implement the extension in PHP. Any coder can easily add on additional extension to it. Our extension blended quite well with the Pootle server software.

Conclusion

The goal of the project is to give the user a Pootle server with the privilege to share the work globally. Our extension extended the functionality of the pootle server by allowing the user to work with the open source repository by making a bridge between the local server and open source repository. The users of the Pootle server used to save everything inside there local machine and never had any privilege to share other people's work. Now with this new feature they can have all their work available globally. By the click of a button they can make a connection to the subversion, checkout all the files into their local directory. And the user will see a .po version of their checked out file to be visually available to a certain page of their pootle project. After that they can translate or edit it and then save it to the svn repository directly.

During the course of this project we faced several challenges. The first challenge was to explore thousands of lines of code of Pootle server which is written in Python and to find a right logical location in pootle to make an extension. Next challenge was to make this extension user friendly so that the user has to enter minimum information to get the end result. A vigorous testing was performed by a number of users after implementation of the extension to the Pootle server. Also the extension was optimized to reduce the execution time.

Bibliography:

1. Official page for Pootle
http://translate.sourceforge.net/wiki/pootle/index

2. HTML tutorial
http://www.w3schools.com/html/default.asp

3. Official page for PHP
http://php.net/index.php

4. PHP svn manual
http://php.net/manual/en/book.svn.php

5. Official page for xgettext
http://www.gnu.org/software/hello/manual/gettext/xgettext-Invocation.html

6. Exec tutorial
http://php.net/manual/en/function.exec.php

7. CSS tutorial
http://www.w3schools.com/css/default.asp

8. Official page for Python
http://www.python.org/

9. Launchpad:
https://launchpad.net/

10. Mozilla Verbatim
http://localize.mozilla.org/

11. Subversion
http://subversion.tigris.org/

12. Selenium
http://seleniumhq.org/

13. Beanstalk
http://beanstalkapp.com/

14. Portable object
http://en.wikipedia.org/wiki/Portable_object_%28computing%29

15. [2000] Practical Guide to Localization (Language International World Directory). Bert Esselink. Iohn Benjamin's Publishing Co. 2000.

16. [2000] Localization A Global Manifesto. Hines, Colin. Stylus Pub Llc. March 2000.

17. [2004] Technical Reports &Notes
http://www.w3.org/International/publications

18. [2007] Internationalization Activity.
http://www.w3.org/blog/International?cat=33