Roberto Koeneke

Middlebury Institute of international studies at monterey  Spring 2015

Website localization: Final project

**Website Localization Workflow**

**Introduction**

What is website localization? According to a Language Scientific article, website localization is “the process of modifying an existing website to make it accessible, usable and culturally suitable to a target audience.”[[1]](#footnote-1) Why is website localization important? The same article states that “more than 1/3 of all internet users are non-native English speakers, and according to Forrester Research, visitors stay for twice as long (site stickiness) if the website is in their own language.”[[2]](#footnote-2) Also, the cloud-based translation management platform company Smartling, outlines other benefits of website localization[[3]](#footnote-3):

* increase in sales,
* tailoring the services to local clients,
* reducing risks associated with improper translations, and
* improve the marketing campaign effectiveness.

Given the huge importance of localizing websites, what is the proper workflow for localizing a website or web app? Language service providers (LSPs), and other companies that provide translation and localization services, all have their own workflows and ways of localizing websites or web apps. However, Julia Rozwens discusses a few considerations that are important when implementing any website localization process[[4]](#footnote-4):

* special attention needs to be paid to cultural context,
* use experienced and trusted translators, and
* consider your target markets and technical requirements:
  + default display of the language
  + layout display
  + proper encoding (i.e. UTF-8)
  + time and date displays (i.e. mm/dd/yyyy vs. dd/mm/yyyy)

This paper will discuss the website localization used in-class during the Spring 2015 Website Localization course offered at the Middlebury Institute of International Studies at Monterey.

**Website Localization Workflow**

*Assumptions*

The website localization workflow discussed in this paper assumes that the localization is done within a PC, and the following programs are installed:

* Virtual Box, running Ubuntu (Operating System, 64-bit)
* Rails web application development framework (within Ubuntu)
* Sublime (within Ubuntu)
* Web browser (i.e. Google Chrome or Mozilla Firefox)
* Internet access

Also, basic knowledge of the Ruby programming language, HTML, and CSS are also assumed.

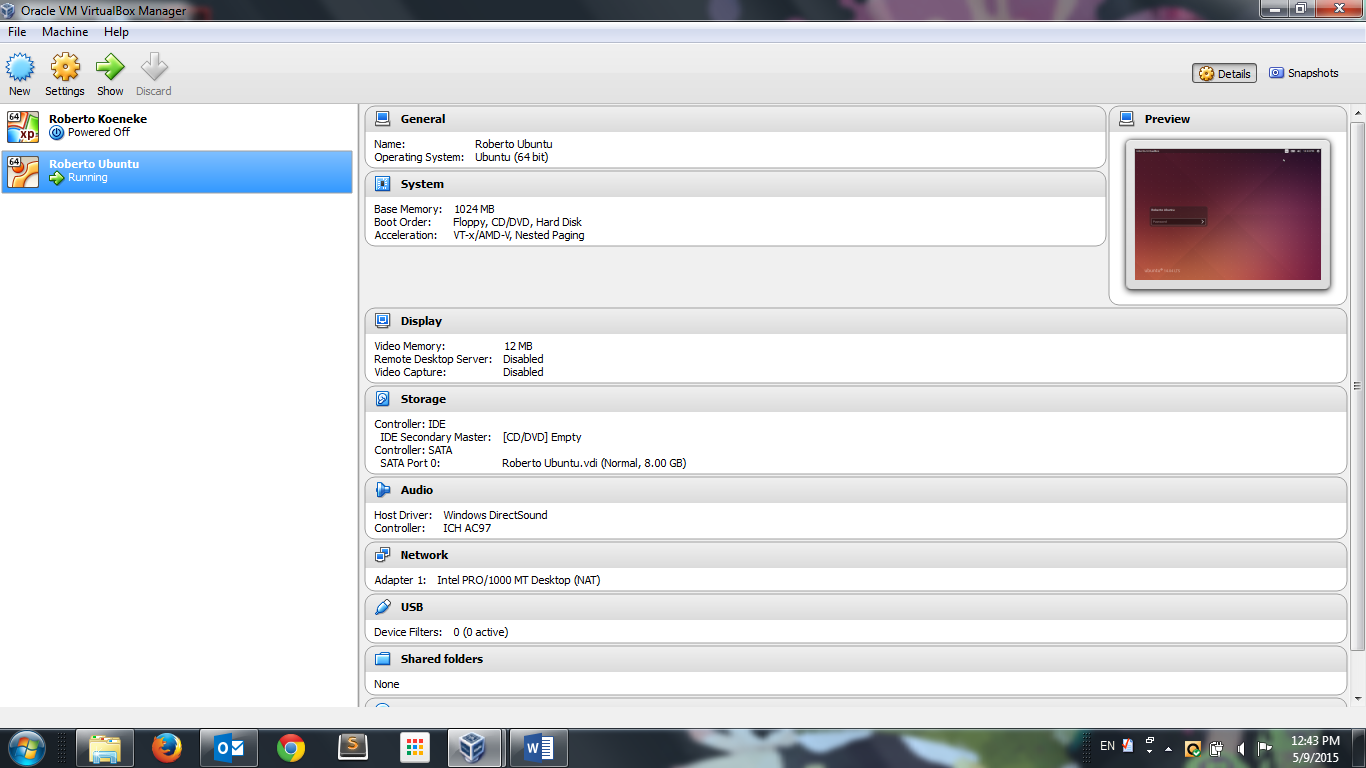


Figure : Oracle Virtual Box running on Windows 7

The last assumption made in this paper is that the localization of the web application or website is done locally, within the user’s PC, and there is no need for outside collaboration. In class, the students collaborated in groups and with the professor via GitHub, which is an excellent online collaboration platform. However, for the purposes of this paper, the localization professional is assumed to already have access to the necessary files (i.e. source and target files), and the translations used in this exercise are assumed to be correct, except during the illustration of the QA process before finalization.

*Workflow*

1) Open Virtual Box, and launch Ubuntu. Make sure that Virtual Box has access to the internet.

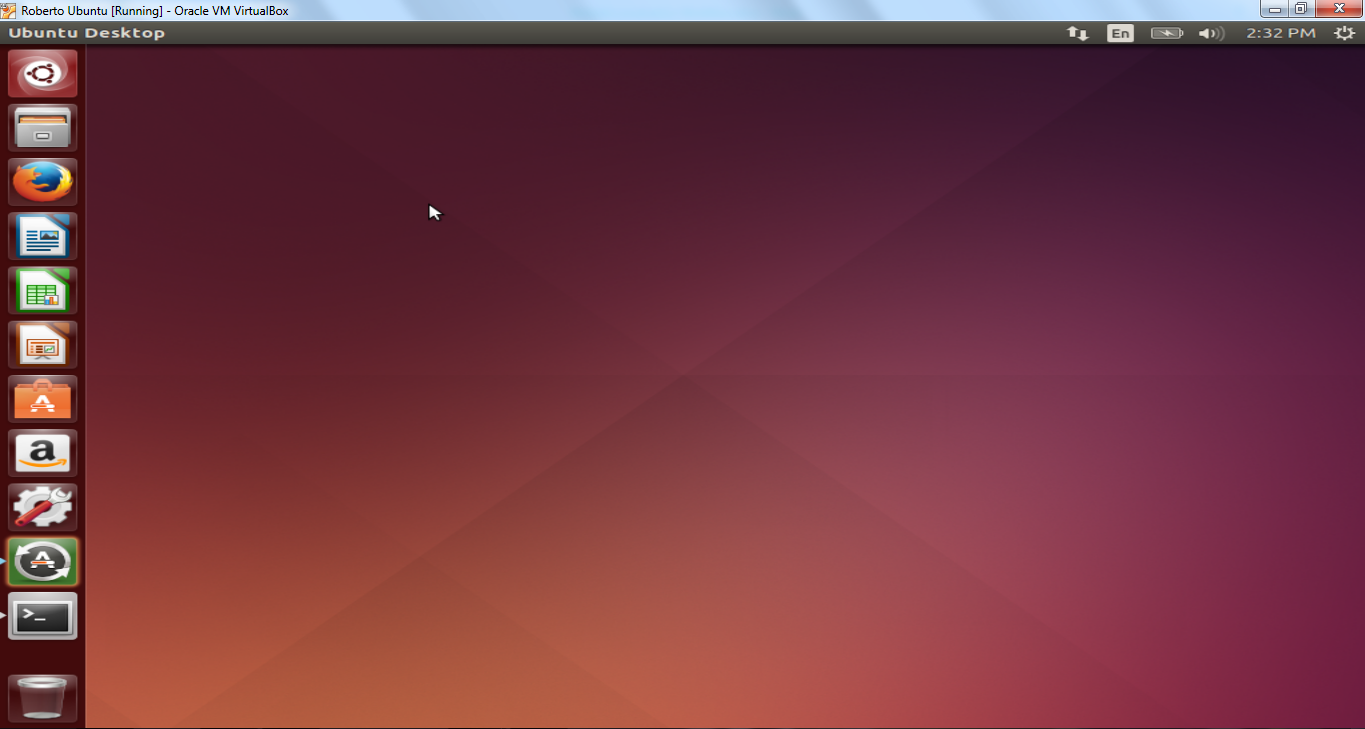


Figure : Ubuntu Desktop View

2) Once Ubuntu is fully operational, create a folder to put the web app source files in, and place the app files in the designated folder. At this point, the localization professional should already have the source files.

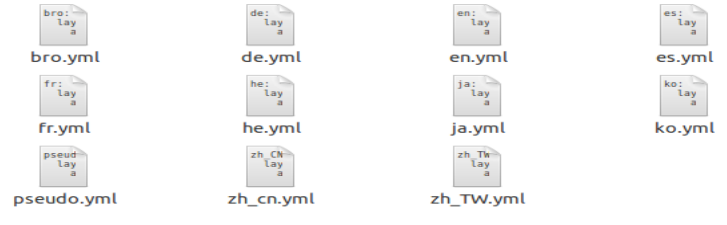


Figure : Locale YML Files

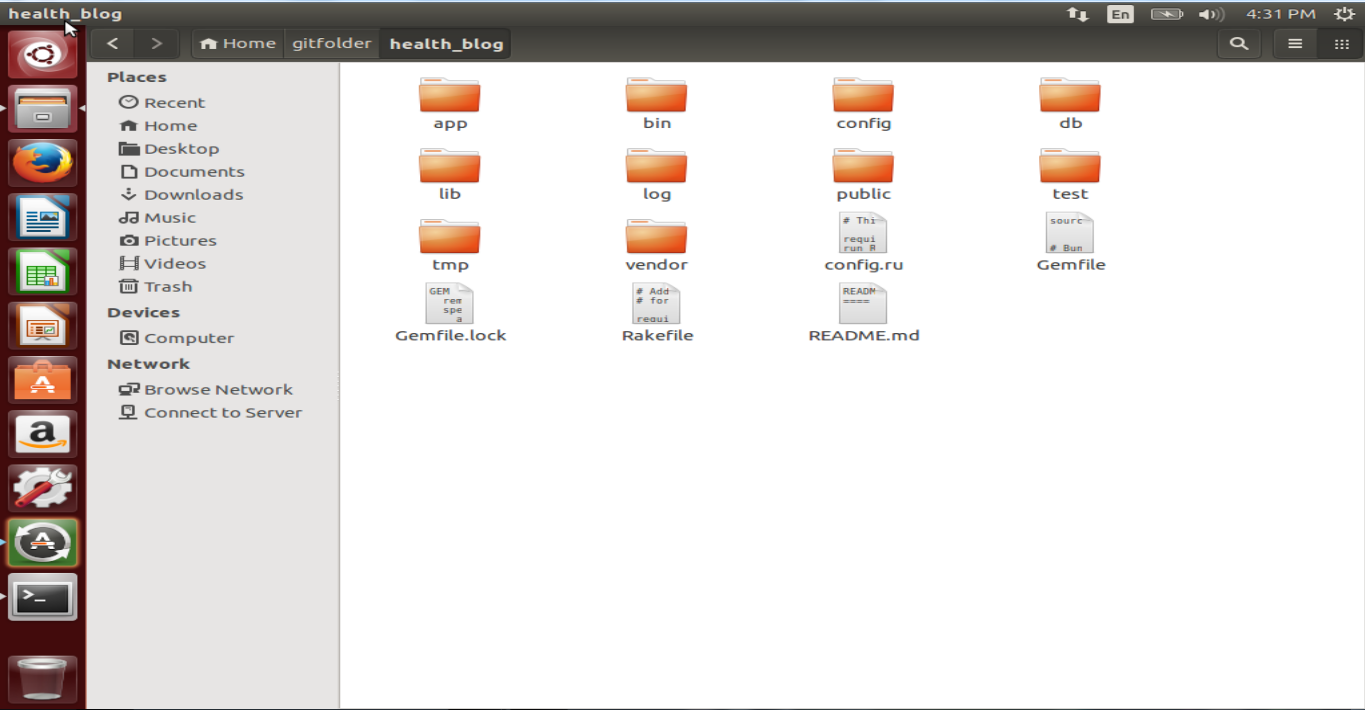


Figure : Folder with Website Files

3) Open the Terminal (CTRL + ALT + T), and move into the folder where the files associated with the web app are stored. In this example, the files for the health blog website are located in the *health\_blog* folder within the *gitfolder* folder.

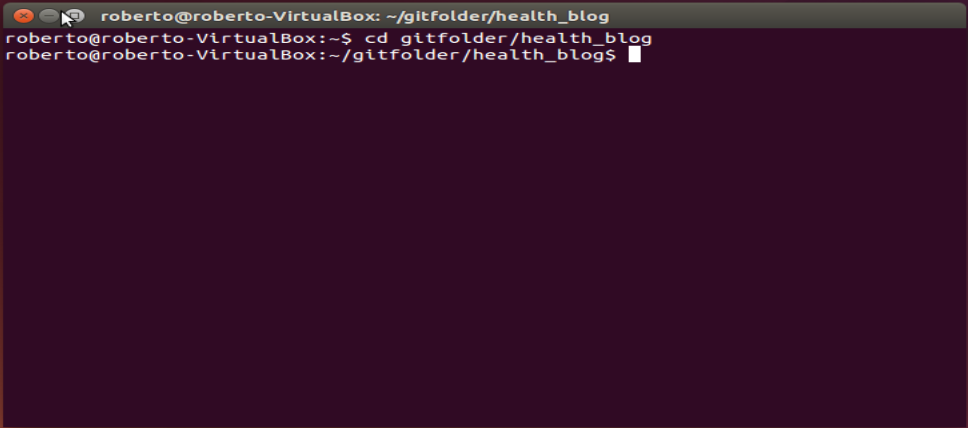


Figure : Terminal Window

4) In the Terminal, run the program Rails by typing *rails server*.

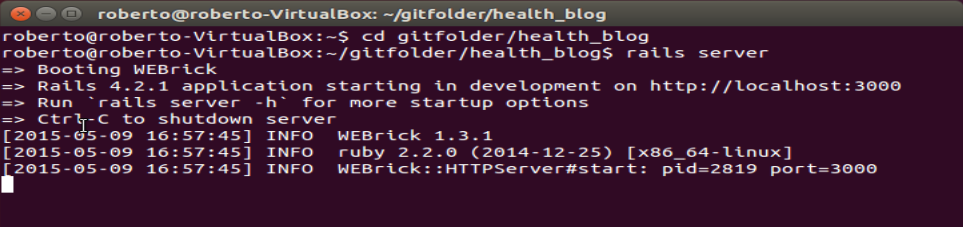


Figure : Running Rails in Terminal

5) Open the web browser, and access the URL where the app is being developed. In this case, the URL is http://localhost:3000. As can be seen from Figure 7 below, the website has five different tabs: Home, About, Static Blog, Dynamic Blog, and Statistics. The Home page is the landing page for the website, and contains a brief information section regarding the website. The About page contains more information about the website, and has three links: *Contact Us*, *Gallery*, and *Sign Up*. The Static Blog is dedicated to a permanent blog post on Hipster eating habits. The Dynamic Blog allows the website visitors to post and view older blog posts. Finally, the Statistics page contains health information on five different countries: USA, United Kingdom, China, Japan, and Brazil.

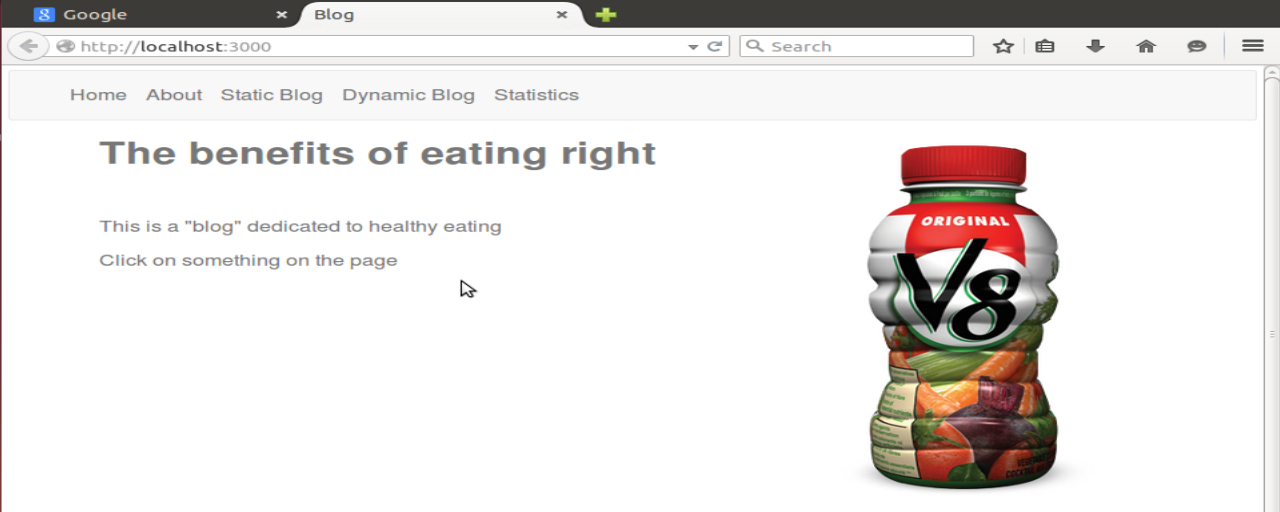


Figure : Website Home Page



Figure : Locale Changer

6) After having explored the website, open Sublime in order to edit the YML files. Drag and drop the entire folder (i.e. health\_blog folder) into Sublime. There should be a folder dedicated to locales, in which the YML files for each desired locale (i.e. English, Spanish, Hebrew, Japanese) is housed. If only the English YML file exists, create YML files for each language desired in the localization. In this exercise, the English and target language YML files are housed in the *locales* folder. YML files were created for each locale: German, Spanish, French, Hebrew, Japanese, Korean, Chinese (Traditional), and Chinese (Simplified).

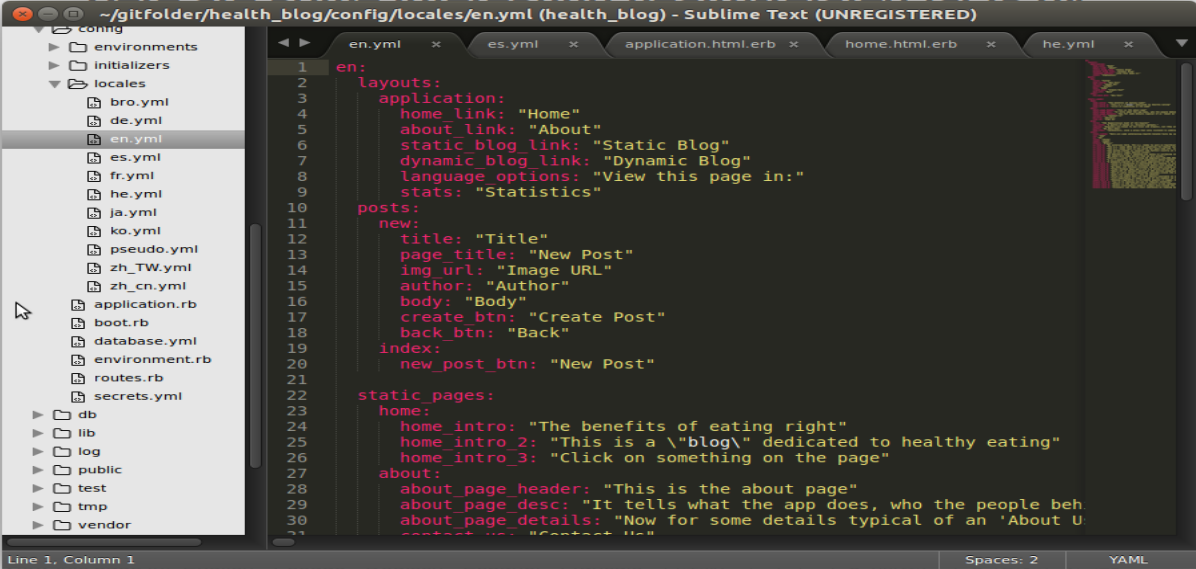


Figure : Files Open in Sublime

7) The next step is to externalize all of the text that needs to be translated in the relevant files (by t-wrapping the text in Ruby code), and include the externalized text in the English file (en.yml). This will help concentrate all of the translatable strings into one file, which makes the translation process easier.

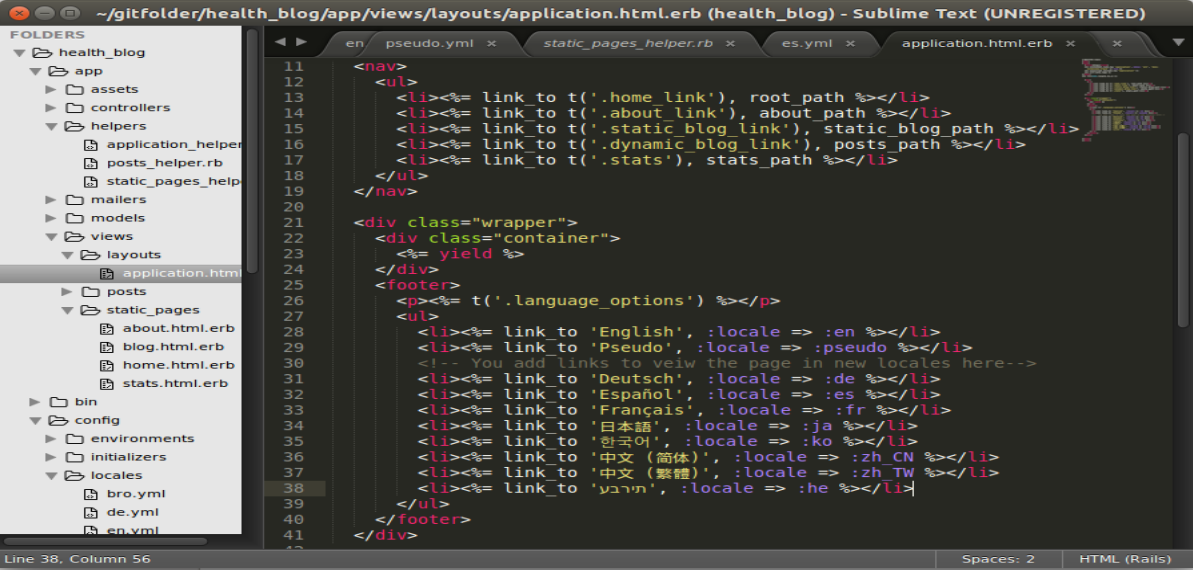


Figure : Externalized Text: Re-written in Ruby

8) A Pseudo-localization YML file should be created in the *locales* folder. Once the necessary strings are externalized, the file should be exported to a translation tool (i.e. Transifex, SDL Trados), and a pseudo-localization should be performed. Next, the pseudo-localized text should be imported back into the YML.

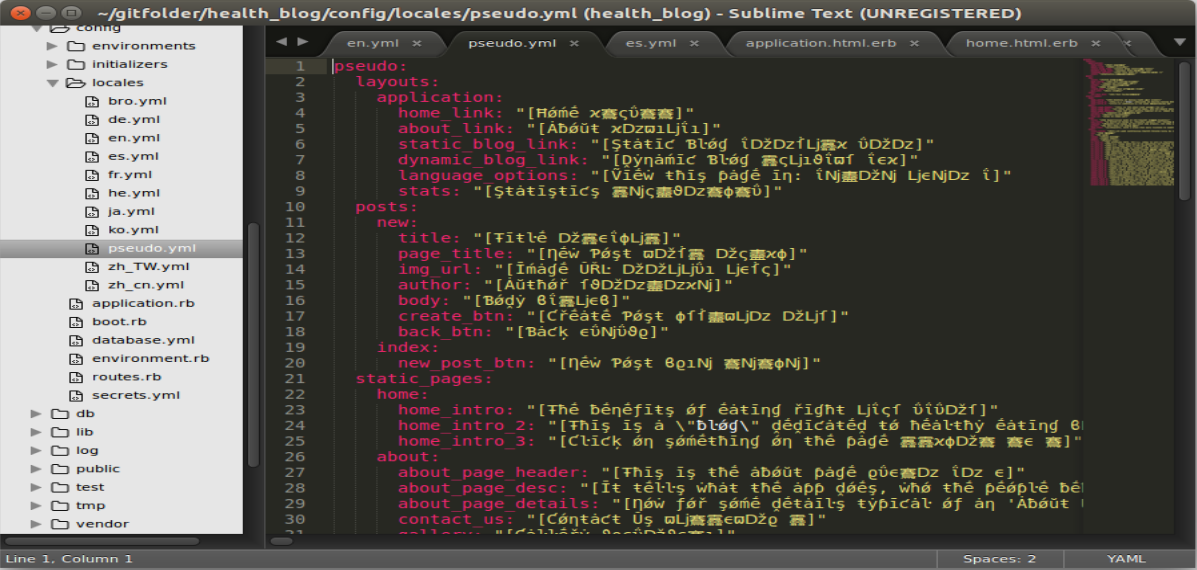


Figure : Pseudo-Localized YML File

NOTE: Each time a new locale file is added to the *locales* folder, in order to load the added locale on the website, Rails needs to be restarted in the Terminal. In order to restart Rails, press CTRL + C to stop the current process. Wait until the process has stopped. Re-type *rails server* in the Terminal in order to restart Rails.

9) Once Rails is restarted, re-open the blog and click-on the *Pseudo* locate at the bottom of the page.



Figure : Pseudo-Localized Website

10) Explore the pseudo-localized website to see how the text is displayed in a different locale, and identify potential issues that may arise when localizing the website into other languages. The pseudo-localization is also handy to see whether special characters will display properly in the page.

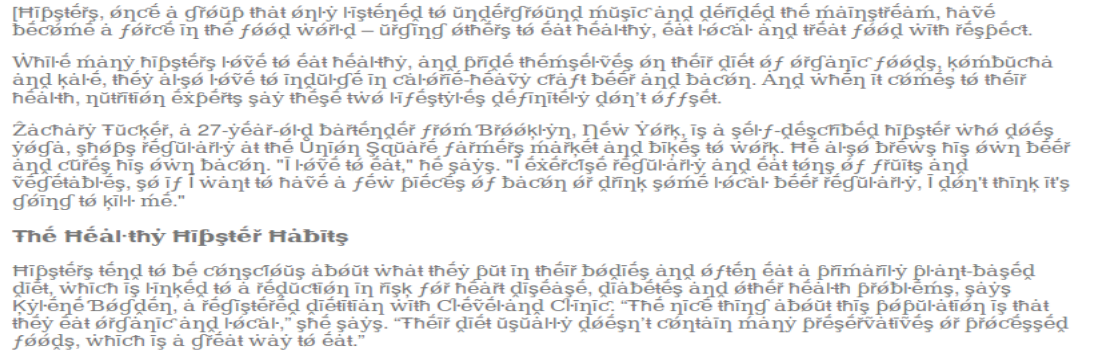


Figure : Pseudo-Localized Text in the Static Blog Page

11) Translate the strings into the different locales, and add the translated strings to the appropriate YML files. For instance, the strings translated into Spanish should be in the es.yml file, the Hebrew strings should be in the he.yml file, etc.



Figure : Translated Text: Hebrew Strings

12) Once all of the desired locales are added, and the relevant translated strings are incorporated into the appropriate YML files, restart Rails. After Rails restarts, re-open the blog, and explore the blog in the different languages.



Figure : About Page: Japanese

13) The website is now ready for quality assurance (QA). Each locale should be examined by a linguistic expert in the given language, to check for linguistic errors, mistakes, and language display. However, other issues, such as layout problems could be identified other members of the team. For instance, even if you do not speak French, it is evident that there is an issue with the table layout in the Statistics page (see Figure 16 below). The tabs in the table are currently below their appropriate location. The issues can be fixed via Sublime, by making the changes to the necessary files (YML and CSS), and saving the changes.



Figure : Display Problems: Statistics Page: French

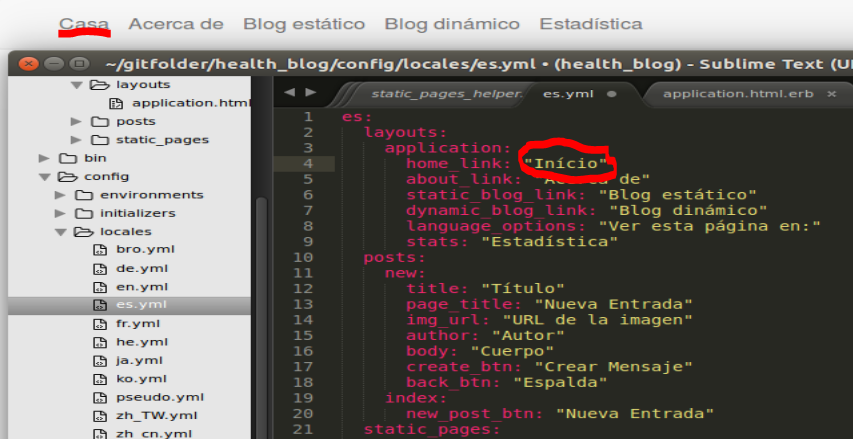


Figure : Fixing Linguistic Issues: Spanish

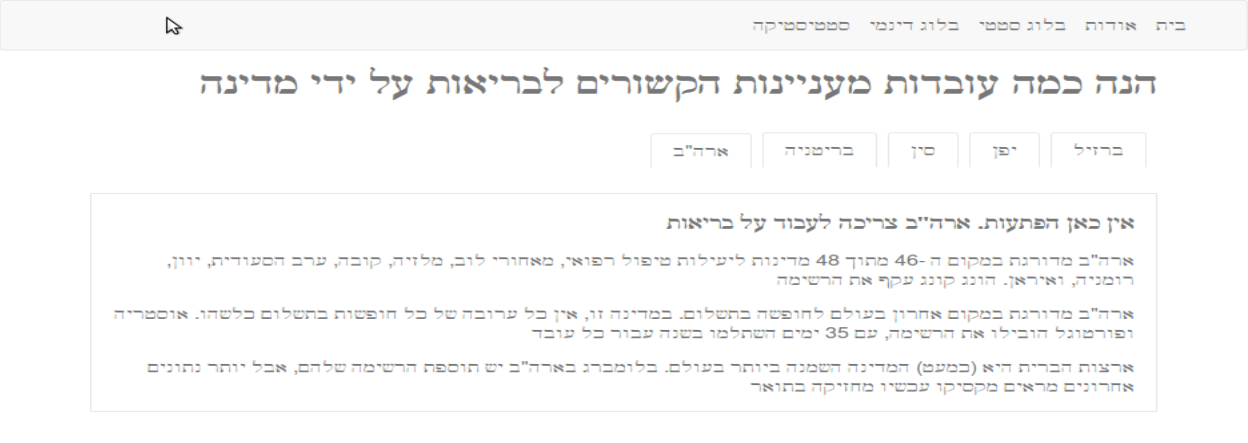


Figure : Right-to-Left Display: Hebrew

14) Each translated page needs to be systematically checked for linguistic and other technical issues, and the bugs need to be addressed.

15) Once the bugs are addressed, there needs to be a final check, to make sure that there are no additional errors or issues.

16) After final approval, the localized files can be returned to the client for publishing the website.

**Conclusion**

The website localization workflow can be a lengthy process, and a wide variety of tools are needed. However, the tools needed for each localization workflow will vary according to the project, and the resources available to the localization professional or team. The example used in the paper to describe the workflow utilized free tools that are readily available on the web: Sublime, Virtual Box, Ubuntu, and Rails.

While the needs may vary according to project and client, a basic workflow can applied to a wide variety of projects:

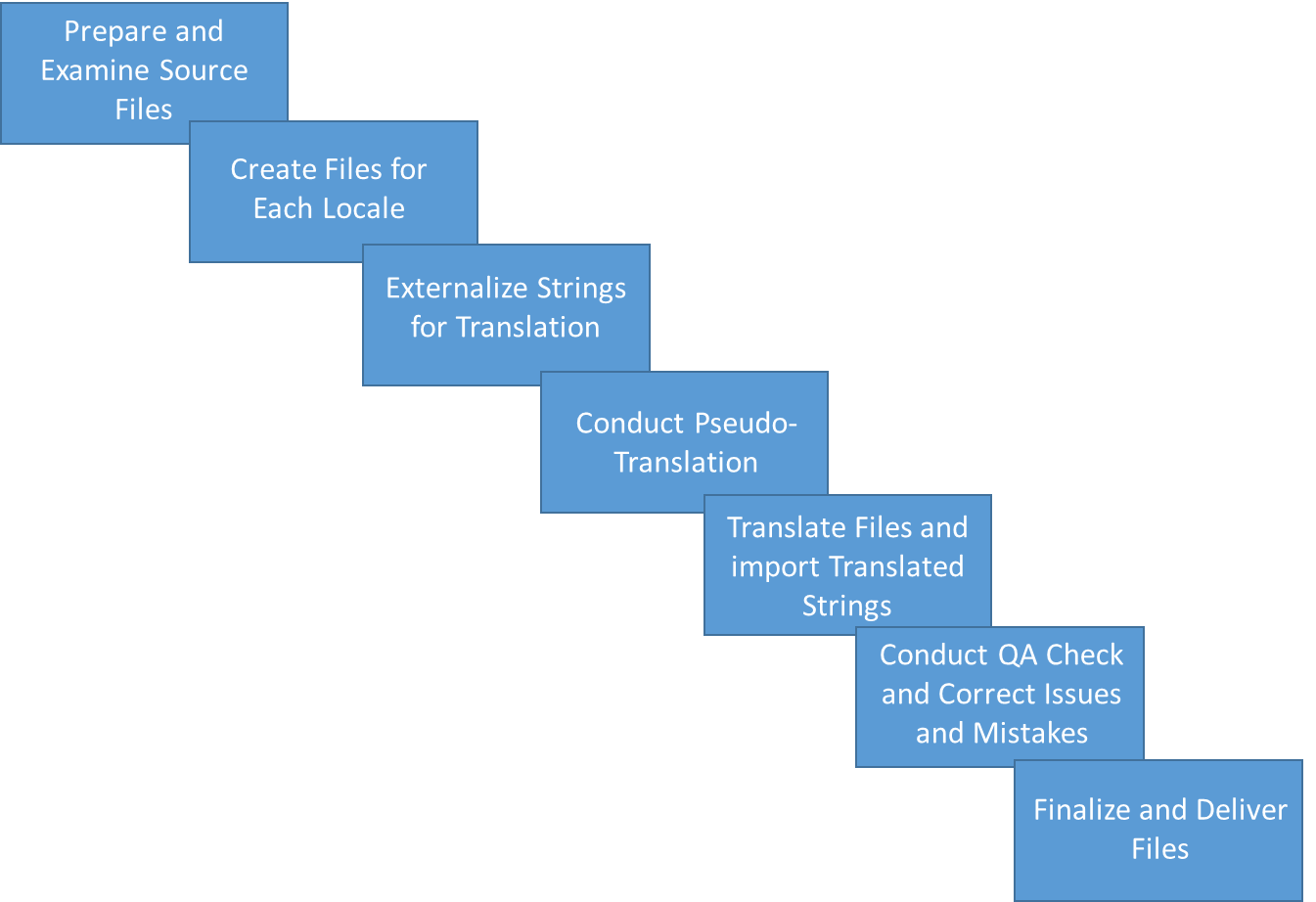


Figure : Website Localization Workflow

The website localization workflow described in Figure 19 (above) is quite simplistic; however, it provides the general and basic steps that are important to any website or web app localization workflow. While this workflow focuses on the technical aspects of localization, it is vital, as Julia Rozwens[[5]](#footnote-5) mentions in her article, to take into account the cultural aspects of the market that you are targeting. The technical details are important, but the qualitative aspects of localization are also imperative to the success of any website localization, especially in today’s highly globalized world.

1. Language Scientific. 2014. Website Localization and Website Translation—What Is Involved? Accessed on 07 May 2015, available at: http://www.languagescientific.com/translation-services/website-localization-services.html. [↑](#footnote-ref-1)
2. Language Scientific. 2014. Website Localization and Website Translation—What Is Involved? Accessed on 07 May 2015, available at: http://www.languagescientific.com/translation-services/website-localization-services.html. [↑](#footnote-ref-2)
3. Smartling, Inc. 2015. 5 Reasons Why Localization is Important. Accessed on 07 May 2015, available at: http://www.smartling.com/2014/09/04/5-reasons-localization-important/ [↑](#footnote-ref-3)
4. Julia Rozwens. 2014. Don’t Get Lost in Translation: How To Conduct Website Localization. Smashing Magazine. Accessed on 07 May 2015, available at: http://www.smashingmagazine.com/2014/12/03/how-to-conduct-website-localization/ [↑](#footnote-ref-4)
5. Julia Rozwens. 2014. Don’t Get Lost in Translation: How To Conduct Website Localization. Smashing Magazine. Accessed on 07 May 2015, available at: http://www.smashingmagazine.com/2014/12/03/how-to-conduct-website-localization/ [↑](#footnote-ref-5)