WikiBhasha Beta

(Version 1.0)

DEVELOPMENT GUIDE

Table of Contents

1 INTRODUCTION

WikiBhasha beta is a browser-based application that helps a community of users to create multilingual content in their Wikipedia, leveraging existing content from other Wikipedias. This application manifests as a simple and intuitive user interface layer that stays on the target language Wikipedia article that is being created or enhanced through this application, for the duration of the user session. At the end of the session, all content created or modified are submitted to the target language Wikipedia.

The UI layer implements a simple 3-step process helping the user in (i) gathering content from source language Wikipedia, (ii) composing the target language Wikipedia article, and finally, (iii) submitting new content to the target Wikipedia. The UI layer integrates content discovery, linguistic and collaborative services, transparently. WikiBhasha beta's workflow may be customized as per the requirements of specific user communities. Also, customization is possible at many levels, including adding support for many 3rd party services or for handling Wikipedia specific issues.

Though WikiBhasha application is architected to be transparent with respect to languages of the source and target Wikipedias, the beta version focuses on leveraging the large English Wikipedia content primarily, for enhancing the content in non-English Wikipedias. Hence, WikiBhasha beta currently supports all translation pairs in Bing Translator (http://www.microsofttranslator.com), in which the source language is English. WikiBhasha beta currently tested on Internet Explorer (7 or above) on Windows XP, Windows Vista & Windows 7 and on Firefox (3.5 or above) on Windows and Linux (Fedora 11 or above).

1.1 Purpose

The purpose of this document is to provide a comprehensive overview of WikiBhasha beta's architecture and the code organization. It provides a detailed description of the Architecture, its components and classes. The document also talks about customization of WikiBhasha beta and its future enhancements which will help development community to customize WikiBhasha beta according the requirements of specific Wikipedias and user groups.

1.2 Goals

Listed below are the goals of the document.

- *To provide a high level view of WikiBhasha beta's Architecture.*
- *To get an overall understanding on WikiBhasha beta's architectural components and details of functionality supported in the components.*

1.3 Conventions used in this Document

Bold:	To emphasis
Italics:	Path of a file or a folder
Note:	To be noticed before proceeding further
URL:	Links to URLs
`Courier new:`	Code Snippets
/* */:	Comments or descriptions for the code snippets

2 FUNCTIONAL VIEW

The functionality of WikiBhasha beta on a given wiki-site enables an on-site, in-place multilingual content creation capabilities on Wikipedia refer figure 1.

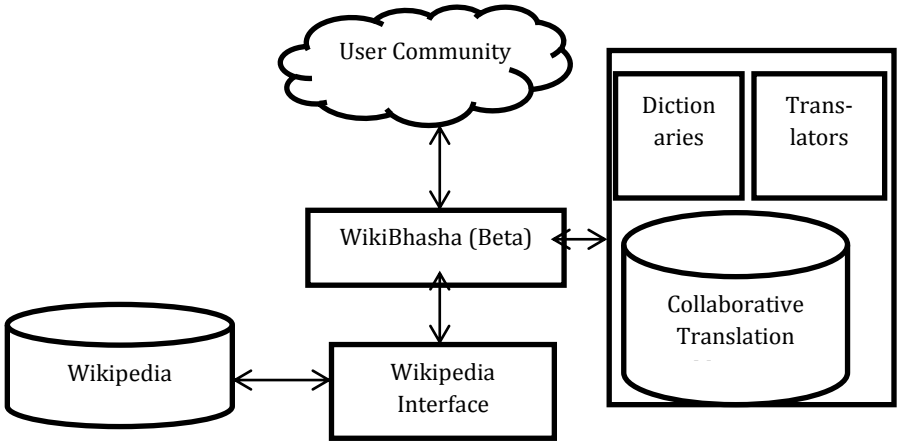

Figure 1

WikiBhasha beta helps wiki-sites to leverage the existing information skew among languages for creation of multilingual content, by community participation. The content is obtained from the abundantly available English Wikipedia and is enriched and translated to desired language. WikiBhasha beta uses translators, dictionaries and Collaborative Translation Memory to translate the content. It provides an intuitive interface to edit and improve the quality of the translated content. The edited and translated content can then be published to the respective language Wikipedia.

3 ARCHITECTURAL REPRESENTATION

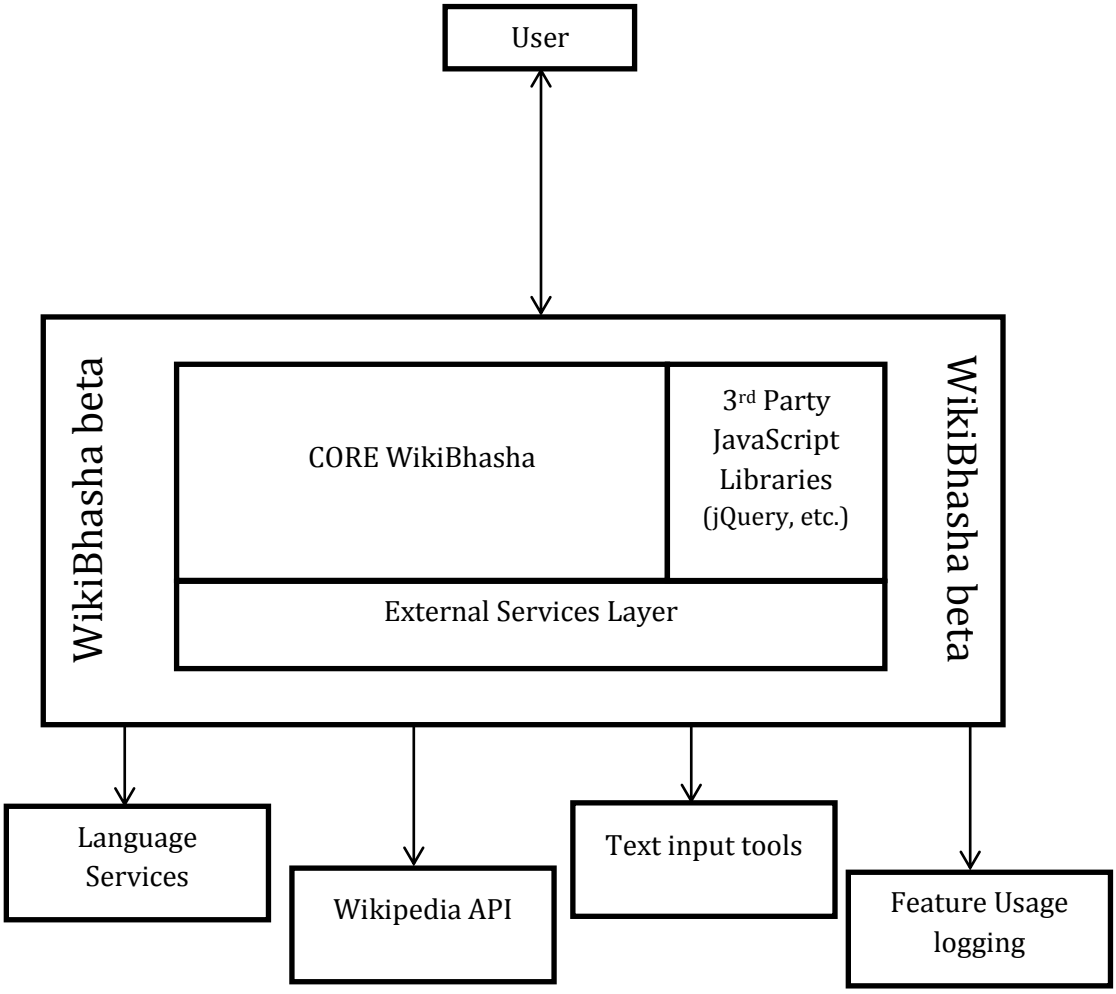

Figure 3

The Architecture consists of three main parts as referred figure 2

Core WikiBhasha

External Services Layer

3rd Party Libraries

4 CORE WIKIBHASHA

All the core functionalities of WikiBhasha beta are part of this component. Core WikiBhasha consists of following components:

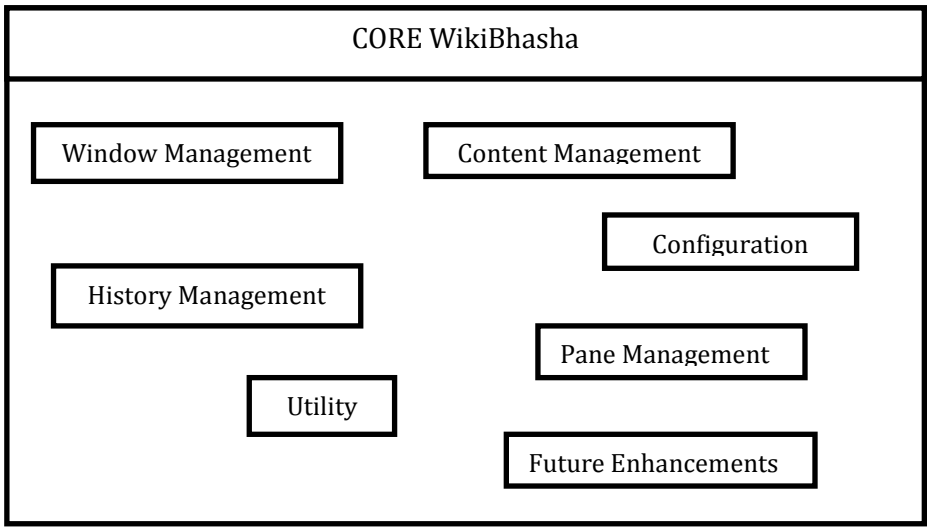

Figure 3

4.1 Windows Management

Classes that are responsible for window related functionalities are grouped under windows management namespace. Functionalities like maximizing and minimizing, theme management, context menu management, scratch pad management are examples that go under windows management.

All classes that represent a window in UI ends with suffix "Window" (for example, searchWindow, scratchPadWindow etc). Each of those window classes has windowId method that uniquely identifies the element id of the window container element, and show function that displays the window.

The following classes part of windows management:

4.1.1 contextMenuHandler

All the functionalities of the context menu like displaying the menu items and retrieving menu item information from configuration and add them to context menu are some of the key functionalities.

Class in File:\js\core\mainWindow.js

4.1.2 resizeHandler

This class is responsible for the maximizing, minimizing, resizing and restoring of the windows and the panes.

Class in File:\js\core\mainWindow.js

4.1.3 SplitterManager

The split between source pane and target pane, alignment between them and width of the panes are adjusted in accordance with the parent div using this class. Handles drag operations on splitter by user.

Class in File:\js\core\mainWindow.js

4.1.4 scratchPadWindow

This class manages all the functionalities of the scratch pad. The functionalities include interaction with language services for translation of given text, window drag, move and resize.

Class in File:\js\core\scratchPadWindow.js

4.1.5 searchWindow

This class manages the functionalities of the search window.

Class in File:\js\core\searchWindow.js

4.1.6 splashWindow

This class describes the splash window for the application, to notify the user that the application is in the process of loading.

Class in File*:\js\main.js*

4.1.7 tutorialWindow

This class describes the help window for the application, enables user to learn more about application

Class in File*:\js\core\mainWindow.js*

4.1.8 languageSelectionWindow

This class describes language selection window, which enables user to select target language to contribute. This appears in English Wikipedia for user to choose target language article for edit.

Class in File*:\js\core\languageSelectionWindow.js*

4.1.9 feedbackWindow

This class describes feedback window for the application, which enables user to give some feedback about the application.

Class in File*:\js\core\feedbackWindow.js*

4.1.10 themes

This deals with themes management in WikiBhasha beta. Handles switching of color themes.

Class in File: *\js\core\themes.js*

4.1.11 mainWindow

This class renders the UI in target language Wikipedia and coordinates with the other classes under windowsManagement namespace for handling various events.

Class in File*:\js\core\mainWindow.js*

4.2 Pane Management

WikiBhasha beta UI can have multiple steps, and each step has one or two panes. These panes are resizable and collapsible. All functionality that goes into these pane areas go under pane management namespace.

The following classes are part of Pane Management:

4.2.1 displayPaneManager

It manages the manipulation of panes according to the configuration defined in configuration.js file.

Class in File: \js\core\paneManagement.js

4.2.2 displayPaneInterface

This is the interface that has to be implemented by each pane class.

Required Methods to be implemented by each pane:

- contentDivId: An unique id of div element that contains all pane contents.
- contentElem: jQuery reference to the div element identified by above 'contentDivId' string. Typically, this element is created and initialized in this class's 'initialize()' call.
- initialize():This is called only once per WikiBhasha beta session before this pane is used or displayed.

Panes can also implement various optional members such as onShow, onHistoryChanged etc to get notification from pane manager on various events.

Class in File*:\js\core\paneManagement.js*

4.2.2.1 **ComposeDisplayPane**

This is the pane where the Wikipedia edit pane is shown in an iframe. This pane also takes care of functionalities such as removing unnecessary elements in edit page for cleaner UI, adding tracing information in submitted content etc.

Class in File*:\js\core\paneManagement.js*

4.2.2.2 **SourceOriginalPane**

This pane displays the original source article content.

Class in File*:\js\core\paneManagement.js*

4.2.2.3 **SourceTranslatedPane**

This pane is to show translated source article content. Also, invokes necessary classes to convert links & templates within the translated article. The source article and translated article are highlighted using parallel highlighting.

Class in File*:\js\core\paneManagement.js*

4.2.2.4 **targetContentPane**

This pane to show target article content.

Class in File*:\js\core\paneManagement.js*

4.2.3 displayPaneHelper

This class holds common utility methods that various panes could call. For example, functionalities such as setting pane titles, registering links for history etc go here.

4.3 History Management

Articles viewed in WikiBhasha beta and the edited histories of those articles are stores in history management. It is also responsible items displayed in the search dropdown box which shows the visited articles.

The following classes are part of history management:

4.3.1 historyEntity

This class creates an entity with information from user visited resource articles.

Class in File:\js\core\historyManager.js

4.3.2 historyManager

This class is responsible for all history related manipulations including drop down box which shows the history articles and all the properties and methods to handle user visited article in WikiBhasha beta.

Class in File:\js\core\historyManager.js

4.4 Content Management

Classes managing the content are grouped under content management. Its classes track the wiki-site content, its discovery and the history of content being created. It deals with parsing and interpreting the wiki formatted content into other formats (for wiki mode and hybrid mode). For example, the wiki-content may be in different languages, worked on in different sessions by multiple users of varying linguistic capabilities, and all these information put together may provide valuable parallel data of interest.

The following classes are part of content management:

4.4.1 WikiParser

wikiParser takes wiki markup text as input and outputs the parsed content as xml. The wiki content will not change, but it gets wrapped in XML based markup.

Class in File:\js\core\wikiParser.js

4.4.2 Trie

Trie stores a number of keys as a hash lookup tree for fast lookups .It contains the implementation of Trie which is used in wikiParser class.

Class in File:\js\core\wikiParser.js

4.4.3 baseWikiConverter

It is the base classes for converters that take xml version of wikimarkup and convert to displayable DOM trees.

Class in File: :\js\core\wikiModeConverters.js

4.4.4 wikiConverter

It converts the given html node, which contains a portion of wiki text into corresponding html node to show the wiki text along with the wiki tags. This mode is called wiki mode.

Class in File:\js\core\wikiModeConverters.js

4.4.5 hybridConverter

It converts the given html node which contains a portion of wiki text into corresponding html node to show the wiki text without the wiki tags but in a user friendly style and format. This mode is called the hybrid mode.

Class in File:\js\core\wikiModeConverters.js

4.4.6 wikiModeConverter

This class converts between wiki-markup and displayed Hybrid and Syntax Highlighted modes.

Class in File:\js\core\wikiModeConverters.js

4.4.7 templatesAndLinksConvertor

This class takes a Div element having wiki-markup as input and looks for all links & templates within that div. It looks for corresponding links & templates in the given target language parameter and replaces the source links/templates with the target language equivalent links/templates.

Class in File:\js\core\templatesAndLinksTranslator.js

4.4.8 templateMappers

This class contains the template mapping information. This allows the user to define template in target language for a given template in source language.

Class in File*:\js\core\configurations.templatemappers.js*

4.5 Configuration

Unlike the manager discussed above managers, configuration itself is a class which contains the configuration details of WikiBhasha beta's work flow, context menu, which are sub classes of configuration class. globalSettings is class that includes all the global settings for WikiBhasha beta any configuration changes in this class would reflect throughout the WikiBhasha beta application.

Class in file:\js\core\configuration.js

The following classes are the sub classes of configuration class:

4.5.1 WorkFlow

WorkFlow is a sub class of configuration class. Configuration of steps, buttons and their naming, the sequences in which the steps and buttons should be displayed will be customized in the WorkFlow sub class.

4.5.2 contextMenuItems

contextMenuItems is a subclass of configuration class. Menu items like scratch pad, search box are configured in contextMenuItems sub class.

4.5.3 globalSettings

globalSettings class includes all the global settings, and methods manipulating these settings. It also includes HTML for the UI components of application.

Class in file:\js\core\globalSettings.js

4.6 Future Enhancements

These are some of the future enhancements that are currently been suggested and can be implemented as per the community's requirements. The communication within the communities and external users out of those communities and management of the users based upon the manipulations and translations of articles are on prime focus.

4.6.1 User Communication

This is a future enhancement enables the users to communicate within a desired community to enrich the content and to get assistances on different languages. This also can display availability of the users and the communities.

4.6.2 User Management

User Management is a future enhancement which will track the user details, article contribution and the changed made to the existing article.

4.7 Utility classes

The following classes are helper and utility classes that contain common functions that all classes use.

4.7.1 util

This class includes all common functions that are not specific to WikiBhasha beta UI. For example, functions such as stringFormat, getQueryStringValue etc go here.

Class in File*:\js\core\utils.js*

4.7.2 UIHelper

This class includes all the helper methods that manipulate application UI. For example, functions such as createWindow, makeDraggable etc go here.

Class in File*:\js\core\utils.js*

5 EXTERNAL SERVICES LAYER

The components that interact with the external entities are grouped in this layer. External Services Layer consists of following components. This layer is responsible for the translation of articles and interaction with other APIs.

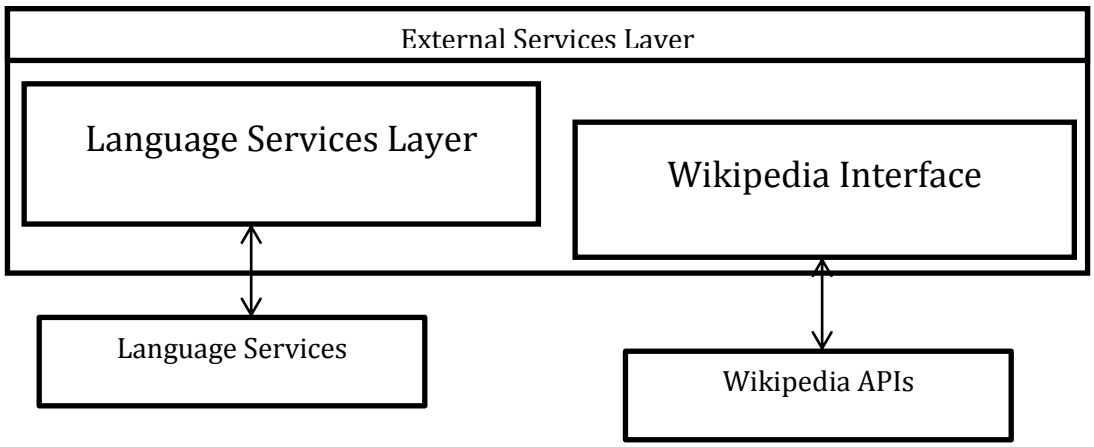

Figure 4

5.1 Language Services Layer

It interacts with External Language Services like machine translation APIs and Community Translation Framework APIs in order to get the translation for the given text.

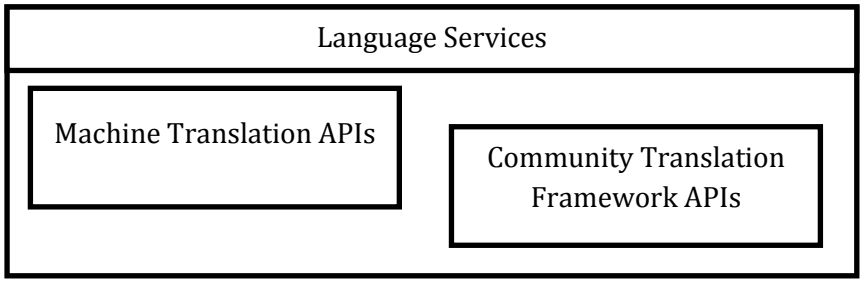

Figure 5

The following classes are part of External Services Layer:

5.1.1 microsoftTranslatorLanguageServicesInterface

This class is responsible for translating the content from source language to target language.

Class in File:\js\extern\languageServicesInterface.js

5.1.2 Machine Translation APIs

WikiBhasha beta uses Microsoft's translation engine. The APIs are detailed at: http://go.microsoft.com/?linkid=9722448 . WikiBhasha beta calls the AJAX version of translation APIs.

5.1.3 Community Translation Framework APIs

WikiBhasha beta uses the above mentioned Microsoft's translation engine, which itself provides the community translation framework.

5.2 Wiki Interface

The Wiki Interface component abstracts our all functionalities needed for interaction with the underlying wiki-system. Clearly, the functionalities depend both on what is supported by the wiki-site, and what is needed for the workflow.

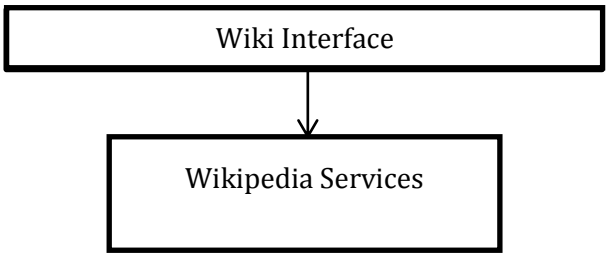

Figure 6

The following classes are being used:

5.2.1 wikiInterface

This class will have helper methods to attach various elements to the DOM.

Class in File*:\js\extern\wikipediaInterface.js*

5.2.2 Wikipedia APIs:

Wikipedia is an implementation of MediaWiki, and it exposes various APIs to access its content such as article html/wiki markup, cross link information etc. The APIs are described in detail at: http://www.mediawiki.org/wiki/API.

The APIs are accessed using two technologies. If the API is in the same URL domain as the current Wikipedia website, then AJAX is used. If the API is in different domain than current URL domain, then JSONP is used. JSONP technology is described in http://en.wikipedia.org/wiki/JSONP#JSONP. WikiBhasha beta uses JQuery library and that library abstracts which technology is used transparently.

5.3 Text Input Tools

Text input tools can be enabled in WikiBhasha beta for assisting users in typing text in their language. The integration of text input tools happens through external interface layer.

The following classes are part of External Services Layer:

5.3.1 TransliteratonServicesInterface

This class is responsible for integrating user input typing tools into WikiBhasha beta interface. This implementation includes Microsoft's tools such as Akshara for Hindi. But the code that integrates Akshara is commented for the initial release.

Class in File:\js\extern\transliterationServicesInterface.js

5.4 Feature Usage Logging

Feature usage logging is stubbed out for the initial release. The intention of the logging mechanism is to log what features the users use, how they use etc and analyze that data to improve usability of the tool. This involves building a logging service backend and making call to that service when user accesses specific features such as scratch pad or a specific step.

6 3RD PARTY JAVASCRIPT LIBRARIES:

The 3rd Party JavaScript Libraries consist are jQuery and jQuery plug-ins. It is used in DOM manipulations, displaying of context menu, keyboard shortcuts and other functionalities. The following are plug-ins and there functionalities:

Note: WikiBhasha beta remains with the jQuery that will be loaded with Wikipedia site on the browser

- *jquery.contextmenu*

  This class is responsible for context menus creation. For more information on this plug-in, refer http://www.trendskitchens.co.nz/jquery/contextmenu/.

  Class in file: \js\jsLib\ jquery.contextmenu.js

- *jquery.shortcut*

  This class handles creation of all keyboard shortcuts in the WikiBhasha beta application.

  Class in file: \js\jsLib\ jquery.shortcut.js

- *jquery-ui-1.7.2.min.js:*

  jquery-ui-1.7.2.min.js has been implemented in various UI components such as tabs

  Class in file: \js\jsLib\ jquery.jquery-ui-1.7.2.min.js

7 FILE ORGANIZATION OF SOURCES

The sources in source control have two main folders:

- *SRC: this contains source code of WikiBhasha beta*
- *DOC: This contains documents related to WikiBhasha beta.*

Under SRC folder, two main folders listed are:

- *WikiBhasha: This contains JavaScript based WikiBhasha beta UI sources*
- *HTML Engine: This contains sources of a console app that extracts HTML for UX Designer friendly*

HTML files and injects that html in JavaScript files. This is described further in section 7.2

7.1 WikiBhasha beta UI Sources:

This is contained under "WikiBhasha" folder as described above. The files are organized by type as "html", "images", "js", "lang" and "styles".

The folders that comprise each component are described in sections below.

7.1.1 'js' Folder:

This contains all the javascript files. The "External Services Layer" and "3rd party libraries" are just few JavaScript files and hence they are under "\js\extern\" and "\js\jsLib\" respective folders. WikiBhasha beta component files are under namespace "wikibhasha" and are the files are named with "wikibhasha" as prefix. The "miniJS" folder is basically compressed version of files under main folder. The compressed version is easy for transmission, but is difficult to read. It is yet is to be determined if compression of files is needed for Wikipedia integration. If it is needed, the files under miniJS can be generated using third party tools just once before deployment.

7.1.2 Image & Style Folder:

These folders contain images & style files. The files are divided by theme type. WikiBhasha beta presently supports three color themes. Each color theme has a separate folder. Theme specific files go under theme folder and general files go under main folder.

7.1.3 Lang Folder:

Localized strings are under Languages folder. Under this folder, there is one folder for each language. Under each language, there is a "strings.js" file that contains localized strings in JSON format. For localizing WikiBhasha beta to new language, create a new folder for that language and copy strings.js file with localized content.

7.1.4 HTML folder:

This contains HTML that users see in UI, and in a format that is friendly for UX designers to edit and improve. Check section 7.2on how this content is used.

7.2 HTMLEngine

WikiBhasha beta is built using pure JavaScript and all html content that is shown by WikiBhasha beta UI is injected by JavaScript. So, the html needs to be part of JavaScript as strings. However, keeping the user facing HTML as strings in JavaScript makes it very hard for a UX designer to tweak and change the UI.

So, the user facing html is kept as html files under "html" folder under sources. UX Designer can make changes to layout by editing HTML files directly there. HTMLEngine is a small tool that extracts the necessary HTML strings from the designer's html files and injects them into JavaScript files appropriately.

This step needs to be done as a pre-deployment or post-build type of action. This needs to be done just once before deployment.

8 API REFERENCES

APIs to extract data from Wikipedia: http://www.mediawiki.org/wiki/API

APIs for invoking Microsoft Translator: http://go.microsoft.com/?linkid=9655926

9 VERSION INFORMATION

Date	Who	Version Remarks
17 Oct 2010	WikiBhasha.MSR	Initial doc version when WikiBhasha was contributed as an Extension to MediaWiki