In-page Localization (Design Page)

Primary mentor	Burke Mamlin
Backup mentor	Rafal Korytkowski
Assigned to	Mykola Vorobey

Abstract

It is annoying for developers to make number of steps to provide default translation of single text message within OpenMRS. If developer needs to do it, he puts corresponding *spring:message* tag into jsp file. Then, he finds the messages.properties file and appends default English translation for this message. If developer should provide translation into another language he needs to edit one more message.properties file etc. From the other hand, another person (translator) needs to complete a lot of steps to provide further qualitative translation of this message into other language (in the most common case, he needs to open default message.properties file to obtain message key, then open message properties file to put new translation into and to runs web-app to see translating message in context of web page).

In more details, we use Spring's message taglib for putting strings onto our web pages currently. This gives us the ability to translate openmrs very easily. If you want to put a string into a jsp page into openmrs you put this *spring:message* in your jsp:

```
<spring:message code="MyPage.personSearchFieldPrompt" />: <input type="text" value="personId" />
```

And then you find the messages.properties file and add this:

```
MyPage.personSearchFieldPrompt=Enter the person id
```

And if you're bilingual and not too lazy you find the messages_es.properties file and append this:

```
MyPage.personSearchFieldPrompt=Entra el id de la persona
```

Now the text can be is displayed automatically in Spanish if the user has set their locale as Spanish. If the user is in the English locale or any other, they will see the english translation.

Problems:

- Its inefficient to have to edit two files to add a simple string.
- Many devs only know english and so will only ever edit the one file.
- When translating message bundles, the translator lacks the context of the message within the application.

Solution:

Allow for in-line adding/editing of the translated text.

Purpose

Implement and integrate pretty handy tool for in-page localization, which can greatly simplify work on translating OpenMRS.

Objectives:

- Create an OpenMRS JSTL tag for messages – i.e., `openmrs:message` – allowing us to control the output and enhance messaging within OpenMRS.
    - This can extend the existing spring:message tag; however, we will want to add the ability to output special HTML to support in-line localization as well as allow for locale & message to be specified in the web page.
    - At the same time, we'd like to enhance the utility for referencing messages within Java code to include a locale and message for that locale.
- Create a module that enables in-line localization.
    - When translation is enabled, render messages in a manner that support in-line changes.
    - Persist those changes back to the message bundles.
- Create a workflow that ensures that local changes to the message bundles are not lost when OpenMRS is upgraded.

Extra credit:

- Create a script to identify embedded messages in web pages and code and bring those into the appropriate message bundle(s).
- Create a script to take the default bundles (i.e., en) and import those messages into web pages and code as the default message.

Features list for future versions (2, 3 etc):

- Allow export of message bundles for core or another module that contain the sum of message from bundles + changes in database.
- Add a way to clean out changes in the database that are no longer different from message bundles.

Review of project requirements

Functional and non-functional requirements set to this project are currently collected into one document and placed on separate page.

Design overview

Basically, from the structural perspective, in-page localization tool for OpenMRS can consist of two parts. First is a dynamic web widgets, which work inside client browser. The second one might be a DWR-based backend for serving AJAX requests from web widgets. Both these parts can be built on the top of custommessages module 's codebase.

There are couple of child pages with detailed descriptions of proposed design:

- Cilent side design
- Server side design

Project implementation plan

See this wiki page to get details of project implementation plan.

See Also

- ~~TRUNK-1525~~ - Enhanced message handling for localization within web pages CLOSED
- JIRA project is https://tickets.openmrs.org/browse/CSTM
- Related module is Custom Messages Module

Interested Parties

-