Language Translator widget - PoC

Technical info & How to (AS3 ONLY)

1 Technical info - how the language translator widget works

1.1 The database

The database is currently hosted on numbersandpictures.com.

There is one table called "trans" which has the following fields:

#	Column	Type
1	id	int(11)
2	lessonid	int(11)
3	lang	varchar(8)
4	search	varchar(1024)
5	replacer	varchar(1024)

'lang' is the string representing the language - for example "fr", "es", "hi"

'search' is the contents of the text field that is going to be searched for, and 'replacer' is the new translated text to replace 'search' with.

At the moment, two methods access the database, 'getTrans' and 'addTrans'

'getTrans' is passed 'lessonid' and just retrieves all data with that lessonid, for example lessonid = 165 gives:

{

"success":true,

"data":
[
{"id":"51","lessonid":"165","lang":"es","search":"test","replacer":"abcdef"},
{"id":"52","lessonid":"165","lang":"es","search":"test","replacer":"abcdefghijklmn"}
,
{"id":"53","lessonid":"165","lang":"de","search":"Patoja no quiere perder a sus paticos.","replacer":"Mrs. duck avec les ducklings"}
,
{"id":"54","lessonid":"165","lang":"de","search":"Patoja no quiere perder a sus paticos. Algunas veces a los paticos les gusta esconderse.","replacer":"Look! A translation"}
,
{"id":"55","lessonid":"165","lang":"he","search":"El rescate de los paticos","replacer":"Story of ducks"}

] // end data

}// end

As you can see, all data is stuck into one great big table and the entire thing is returned to the lesson at run time, nothing clever.

At run time, an array of "Dictionary"s are created, each of which has a language (eg. "es") and an array of "search" and "replacer" Strings.

IMPORTANT: Notice that the word "test" appears twice, once with id 51 and once with id 52. At runtime the translations will be searched *backwards*, meaning that when the "es" dictionary is searched for the word "test", id=52 will be the first matching one and the search will stop, returning "abcdefghijklmn".

The 'id' field is just an auto-incremented integer which uniquely represents that entry in the database, but it is not used (apart from to get the order right).

1.2 The AS code

My new code resides in two flex projects (actually pure AS3, no mxml or anything).

They are:

- langHM

This is the code for the widget and the database communication, so 99% of the code I have written is in here.
I found it best to compile it to a swc (a flash compiled component which is basically just like a Java jar file) - more about that below. It uses the langHMLib library.

- langHMLib

This is a very small project, just 2 files. It consists of the files that are needed for both:

  - the fla of a lesson when it is converted to "language-enabled"
  - and the fla for content_player.fla

This one is also compiled to a swc and langHM lists that swc on its buildpath.

Both of these are in git.

Question: Why are there two projects???

Answer: Each lesson flash file (dropdown) is processed by a jsfl file which adds a few lines of code to each StaticText element. The code basically just dispatches an event which says "please start managing me".

So, this code has to be imported into each and every drop down. Imagine what happens if all the code was in one project and someone changed a button label or some GUI change. You would re-compile the content_player to get the new changes, but you wouldn't want to re-publish all the lessons. But actually, there would be a new version of the "please start managing me" class (even if you hadn't actually changed anything in that part of the code), and now the two swfs would have imported two different version of the same class, so you'll get run time errors.
So, I separate the "please start managing me" code into a separate library which hopefully doesn't change hardly at all (although if a change is necessary then ALL fla's will need to be re-published)

It would be possible to make it so that you never need to re-publish the fla files by dispatching an event like:

dispatchEvent(new Event("PleaseProcessMe"...))

rather than how it is now:

dispatchEvent(new FlashLangEvent(FlashLangEvent.PROCESS...).

As you can see, that would mean that the FlashLangEvent class never needs to be imported at all, but it is not really the recommended way of handling events - I'm welcome to suggestions about that though.

1.3 content_player

Notes, audio, pagination, loading lesson swfs etc is all handled in content_player (fla compiled to swf). I haven't changed much of this file, and it is a bit messy to be honest.

I just added a one line to the end to initiate the widget:

new LangManager(this, 165, 18172, {debug:true});

165 is the lesson id (currently hardcoded as 165 but in reality content_player has access to the lessonid anyway).

18172 is the userid (which we might need at some point to decide whether to show the Edit and Save buttons to add new translations, or just the language selector for teachers who just view the lesson). it is not used at the moment at all, and 18172 is just a random hardcoded number.

As I said above, content_player.fla must have both langHM.swc and langHMLib.swc on its build path.

The debug option shows a textfield used for my debugging.

It is passed a reference to "this" because "this" is the content_player and it listens to the "process me" events as they bubble up from a lesson.

1.4 Each dropdown fla

1.4.1 The jsfl file - preprocessing

Each dropdown fla must be pre-processed by a jsfl file. The jsfl file is in git.

It does the following:

a)

Scan the main stage timeline for any StaticText fields on the stage. Go through every layer one by one, and if it's not a guide layer, then inside that layer go through every frame. Look at each frame and see if it is a keyframe. If it isn't then ignore it. If it is then go through every element in that frame and see if there are any textfields. If there are then convert them to movieclips (and do nothing else). Starting with x=0, each of these movieclips is given an id such as "HMTranslate_x". If you run the jsfl again I can't guarantee that the names will be the same - you may well have interchanged two layers for example. Since it goes through the layers in order it will process them in a different order. So, these names are not used at all outside of the jsfl process.

b)

Go through every item in the library [notice that this now includes all the movieclips from a)]. Repeat the process in part a) but this time, when a static text field is converted to a movieclip, attach some code on frame 0. You might have noticed that there is some redundancy in this process - textfields on the stage get processed twice, but it is the only way I have been able to get it working!

The code attached to each frame 0 is this:

new FlashLang(this, 'HMTtranslate_33', new Rectangle(232,2,607,74));

More about this below.

c)

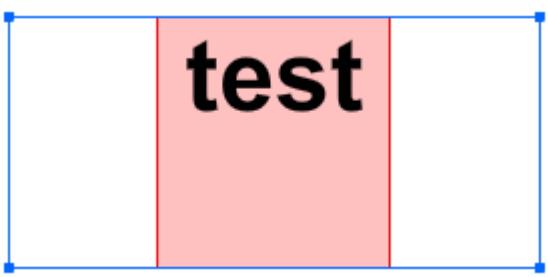
Update the build path to include the langHMLib.swc and then save and publish with a new name "..._LANG". Currently this is all a bit manual and the swc has an absolute path for my laptop (C:/Users/John...)

1.4.2 langHMLib.swc - FlashLang and FlashLangEvent

FlashLang is one of the classes in the shared library (langHMLib.swc).

HMTranslate_33 is the name of the movieclip (as I said, not used yet but I thought I'd pass it anyway) and the Rectangle contains the x/y/width/height parameters.

The width and height parameters are important because I have noticed that sometimes (when text is centred for example), using actionscript's MovieClip.getBounds() method does not return the correct width. Sometimes it returns the red rectangle, not the blue one.



My code uses the blue rectangle to add the new textfield for the translation. There are advantages and disadvantages to that - we lose the align="center" property so text will get pushed to the left. But we do have more room to play with. I can experiment some more if necessary. I had a few issues with this, and it returning a different bounding box depending on whether text was left, right or centre aligned.

x and y are not actually used at the moment because MC.getBounds() is great for getting the x/y position.

FlashLang simply does the following:

- If movieclip.stage exists then dispatch the PROCESS event
- else, add a "added to stage" listener and on added, dispatch the event.

I think using Events is the best way to decouple the lesson swfs from the content_player swf.

1.5 The LangManager class

As described above, LangManager listens to the PROCESS events, makes the GUI and does the database connection.

In more detail it does the following:

- It is added to content_player
- Begins listening to "PROCESS" events immediately, and load the JSON from the database.
- When it hears an event it starts managing that movieclip. That movieclip is guaranteed to have exactly one child (a StaticText element)
- By "managing" it means:
    - add rollover, rollout, click handlers.
    - draw a white, alpha = 0 box in the movieclip so that mouse events trigger on the whole of the box, not just the black pixels of text.
    - Add one extra text box inside the movieclip (which I call a TextComponent) to show the translated text. The TextComponent is made in pure AS, and is a completely new DisplayObject - this means that we can do what we want with it independently of the original static text field, so we can change the font size, we can embed fonts, we can change the text direction to Right to Left - anything possible in actionscript is possible. It is set with the same width and height as the original StaticText field.
    - add a "removed from stage" handler, which cleans up all the listeners and the TextComponent when the textfield leaves the stage. So, if you scrub the timeline back and forth a static textfield might be "managed" and the cleaned up several times.
- Create and empty movieclip for the widget GUI
- When the JSON loads, parse it into an Array of Dictionary's and create the GUI itself.

The code is not beautiful at the moment, it is full of Singletons and should probably be re-written. It uses minimalcomps (http://www.minimalcomps.com/) for the buttons etc.

2 How to "langify" a lesson in 6 easy steps

The repository is: https://github.com/jgrindall/langHM

Check this out first. I am waiting for Ajith to make me a HeyMath! git repository.

2.1 Import the langHM and langHMLib projects into Flex.

I made these in Flex 3, but you should be able to compile them in any version of Flex, or even on the command line. You don't need mxml, and langHMLib has no dependencies at all.

Compile langHMLib to a swc. I do it on the command line using compc, in a one-line windows batch file (bat/buildlib.bat)

```
compc  -source-path langHMLib\src
       -include-sources langHMLib\src
       -output langHMLib\langHMLib.swc
```

You will then have a swc called langHMLib.swc in the langHMLib folder.

2.2 Compile langHM

langHM requires langHMLib.swc to be in its build path. I link the swc in Flex's build path to get syntax highlighting, but ultimately I compile on the command line. I have a windows batch file (bat/build.bat)

```
SETLOCAL ENABLEDELAYEDEXPANSION

set lib=langHMLib\langHMLib.swc

compc -source-path langHM\src
      -include-sources langHM\src
      -external-library-path+="%lib%" -output langHM\langHM.swc


REM  notice that that += is important or it won't compile
```

You will need to change the lib parameter.

2.3 Download a lesson from CMS:

You'll get the assets folder and the lesson folder. As you can see, I am storing my test lessons in folders called lesson1, lesson2 etc. At the time of writing this doc I have only done 2. I suggest you do the same for now.

2.4 Open AS3_Template from git. Open content_player.fla. I haven't changed any other files - it is the same as the one Sankar sent me, with the extra line I mentioned in section 1.3 and the swcs included in the build path. Notice that at the time I wrote this doc all lessons are hardcoded with random lesson id 165 and userid 18172! Compile it in flash, and copy it to the assets/players folder of your chosen lesson(s)

2.5 Process each lesson fla with jsfl

Download langHM.jsfl from git. Copy it to your Adobe Flash Command folder (mine is C:\Users\John\AppData\Local\Adobe\Flash CS5\en_US\Configuration\Commands), and open it in flash (File > Open, don't drag it in) Then open each fla one by one and run the jsfl. You'll get a new file called filename_LANG.swf. Let me know if there are any javascript errors or anything. These are the swfs that should be loaded in the lesson.

2.6 Deploy!

So that you don't have to rename the files and delete fla's before moving to a server I have a script (bat/copy.bat) which copies all the files to my localhost and renames _LANG to the original filenames so you don't have to edit the lesson dropdown code. You could host the files on your local host if you have a web server running. It will still be able to connect to numbersandpictures.com because I have enabled it.

3 Disadvantages

3.1 The text replacement does a simple look-up. That is, if you have two lines of text "This is a square" and you replace one of them with "Esta es un quadrado" and one of them with "Este es el quadrado" then whichever you did last will replace both of them, and this happens throughout the lesson. Suggested solution: see 4. Adding an 'id'

3.2 Objects underneath other objects are immune to mouse effects. Not sure if this is a big issue or not - as long as you add the translation to the database at some point before it gets hidden it will be ok. Suggested solution: We could add another "page" of the widget that simply allows you to type in two strings, one to search for and one to replace, and you click ok. If you turn on "display" then the 2nd will appear

3.3 Font information is lost. All styling (font colour, size, bold/italic, formatting like left align, centre align etc) is lost. Suggested solution: none - StaticText does not inherit from TextField, these properties exist only in the Flash IDE and then the text is stored as a graphic with just a "text" property. It may be possible to get the size of the textfield, and count the number of characters and then apply some estimate saying that if the text field is 100x100 and there are 5 characters in it, then it must be font size 48px or something??

3.4 Dynamic textfields always screw things up - there is no way I can think of to get any kind of translation on any dynamic text fields. There values are set at run time by code, we cannot touch them with this method, and they will not move to align themselves next to any adjacent text (which may have got longer and now overlaps the dynamic textfield)

3.5 Custom Actionscript. If a flash person has added code such as "this.mouseEnabled=false" or stopped mouse events in some other way (for example stopping an event propagating) then my language manager will never hear that event. I think there is an example of this in the glide reflection lesson (try changing the Activity1, Activity2 tabs at the top left)

4 Adding an 'id'

You cannot add any kind of id to an object in a swf - a swf is not an fla, it has been compiled.

As I said above, my jsfl does give a name (unique to that fla, not globally) to each textfield that it processes. The name is not used yet, I simply added it so that it was possible if needed in future to get the language manager to check if name.substring(0,11)=”HMTranslate_”. Might be needed in future for some hacks?

It is certainly possible at compile time to add GUID (globally unique ids) to each object, and then they could be stored in the database. But I'm not sure how it would work in practice if someone changes the lesson.

You can't run it through the jsfl again because it will change the id's of all textfields (I don't see how we can assign the same id as before to each object). Am welcome to suggestions, maybe there is some way of doing it??

My general approach has been that

- The fla creation process by the flash developer should be as it is now, they shouldn't have to worry about language translations, or object ids
- The language translation and fla should be totally disconnected - I don't like the idea that they are connected in any way, using id's or otherwise. It has some disadvantages but I liked the idea of keeping the two things totally separate.

5 Other

5.1 I need to work on RTL (right to left) languages. Definitely possible, have started on it.

5.2 I need to work on adding different fonts for different languages - a different font for Hindi - definitely possible.