# Multi-lingual Textpattern

*Just Write*… in any language.

[Contributors: Stef Dawson, Uli Neitzel, Aleth]

With the demise of the Multi-lingual Publishing system (MLP) in recent versions, Textpattern needs an improved multi-lingual experience. This could either take the form of providing a framework to enable a replacement plugin to be built, or perhaps out-of-the-box functionality in core. This document explores the latter, with a nod towards allowing plugin authors to be able to augment the experience.

Originally, unlimited custom fields felt like a good fit, because plugin authors could define a Body/Title/Excerpt field for each language and authors could put content in there, but after some deliberation, it became clear it was going to become unwieldy in terms of UI, not to mention a chore to manage.

At the article level, multi-linguality boils down to being able to write a new article in a different language and indicate it is "connected" to a group of other articles that contain the same information in various other languages. The number of fields are the same. One way to achieve this is to use a single table with an immutable ID column that also contains content that is not required for translation, and connect it to multiple other rows in a translation table that has one row per language.

Converting the current textpattern table to do this requires only:

- Moving the ID column to its own table (along with any other fields that don't need translation, like the bloguid and maybe section and category etc).
- Adding a column for the language identifier of the article (en, fr, zn-ch,etc).
- Joining the two (perhaps via a View).

Under a mono-language install, only one row is returned and the 'lang' field holds the only installed language identifier.

After installing at least one other admin-side language, the Write panel adds a dropdown somewhere near the top; with one option per language. Selecting between them switches to an almost-identical-looking Write panel in which to enter text in the chosen language. A nearby 'clone from' facility would permit text to be imported from any other article in the set so it can be translated. When the article is saved, it's stored as a brand new article row linked to the ID of the original. If you have five languages installed, of which only three have translations for the currently-viewed article, you can clone one of those three to make a new article in one of the remaining untranslated languages.

Thus all renditions of an article are tied together and can be easily selected as a group via SQL, to offer "also in this language" lists to be displayed to both visitors (via tags) and site admins (via the UI, for cloning / editing purposes). MLP has a separate tab for the clone process but a clearer in-page solution is more desirable, perhaps tied to the 'notify an author about this rendition' if it requires attention for subscribers to that language.

From the Article list panel, every rendition will be listed separately. Search options in the search selector can be used for filtering by language or collection. Clicking on an article will jump to the Write panel with the particular language option selected from the dropdown, ready for direct editing.

Multi-lingual category names, image alt text, caption, etc. require more thought but could be achieved in a similar manner by offloading some or all of the core content and an ID to a separate table and having an extended table for the multilingual content.

For example, with images, selecting a new rendition language from the dropdown would allow entry of information in fields against the selected language. When saving, a new record is written to the database linked to the ID of the original image, but *no new image is created*. The act of being linked to the first image is enough: they share the asset.

This might not be practical if, for example, a different image is required for one of the renditions, so this might need some thought. But a convention makes more sense: one image, multiple pieces of language-aware metadata. A callback could be introduced so plugin authors can override the behaviour.

Across all content types, multi-edit facilities would permit:

- "Connect" to link the selected renditions into a single set with thr ID from one of the selected renditions. It shouldn't be necessary for someone to "know" the ID to connect articles together. As long as the ID of one of the articles is present, they are connected.
- "Delete" the selected renditions.

That should pretty much be it for admin-side functionality. On the back of this there are a couple of things to experiment with for the public site. One is for Sections. Thinking about the permlinks for typical multi-lingual sites, they're normally of the form:

```
example.com/lang/section/title
```

or some derivative thereof. We could extend Txp's URL parser to understand such language-based URLs, though perhaps supporting the full language code, which would better cater for language variations in regions. If the first part of the URL doesn't match 'ab-cd' or 'ab' (and it must match an installed language), then it's assumed to be a regular Section, which retains backwards compatibility and a fallback for visitors to the base language.

But the problem with Txp's URL handler at present is that a Section is an immutable entity. Thus if you have a glossary Section, the URLs would be:

```
example.com/en/glossary/hospital
example.com/fr/glossary/l-hopital
example.com/de/glossary/krankenhaus
```

Ideally (perhaps?), these would be:

```
example.com/glossary/hospital
example.com/glossaire/l-hopital
example.com/glossar/krankenhaus
```

In other words, the section name (not just its title) is localised. Is this better SEO? Probably not. We've already put wheels in motion to move away from localised URLs for category/kategorie and author, so it makes sense to continue that tradition to sections. The only sticking point is that sections are *content-based* whereas the trigger words for category, author, tag, and so on are simply that: hooks that Textpattern can use to trigger a different context.

The language identifier in the URL is arguably redundant in this day of semantic crawlers, as they can figure out the language of a page from its content, and various hints supplied in the markup. The URL itself is just a hint for humans and/or a way for the CMS to cater to their information desires, but it's not really needed.

It could be a permlink option, but multi-lingual content should be able to be delivered regardless of whether the identifier is there or not, in the same way that articles from the correct Section are still delivered even if the '/title' permlink scheme is used. Articles with the same title are handled gracefully now, so they should be able to do the same under multi-lingual environments. Building in support for the lang identifier into the pretext URL parser isn't too hard.

In current Txp, you would need to make a Section per language if you wanted to do this, which is an administrative overhead, and tying the renditions together requires a custom field and almost infinite patience (or adi_matrix).

But what if on the Sections panel you could create a Section as normal, but could also specify section aliases that are linked to a language. Without any aliases you get the URL structure as given in the first set of three above. But if you specify that "glossaire" is the name of a section, "Glossaire" is its title, and it's linked to the French language, when Txp parses (or builds) the URL it can look up the aliases, recognise it as such, internally substitute the root section name ("glossary" in this case) and thus find the article.

Such aliases are no more than fields associated with a section and language, the same as the Section's internationalised title. Whatever mechanism is chosen for editing such details (e.g. a Section panel per language, different fields for each language, etc) would apply to section aliases.

At the moment in MLP, only the Section title is internationalised. This proposal merely extends that to the Section name too. You still have only one real Section against which to store articles. The rest just make the URL prettier and redirect silently behind the scenes.

This does pose a slight issue insofar as using the original section name would also work for internationalised content. In other words:

example.com/glossary/hospital
example.com/glossary/l-hopital
example.com/glossaire/l-hopital
example.com/glossary/krankenhaus
example.com/glossar/krankenhaus

would all be valid URLs. So we'd need to tread carefully to avoid duplicate URLs when dealing with SEO. Canonical links might solve that. In HTML, rel="alternate" lang="fr" is one way to define alternate content in different languages and Textpattern should make the correct codes available at all times in the markup, via a tag.

In reality, all renditions of any given article are stored under the same Section (in fact, this part of the UI could perhaps be 'persistent', outside the dropdown portion in the Write panel's interface), and there's really only one actual Section that exists, there's just a bit of decoupling going on between what the URL sees and what is in the DB. Like a presentation layer that maps the pretty URL to the physical article. That's exactly what the permlink schemes do now. This proposal achieves the same one-to-one mapping of URL-to-article (at the article level), but done via a virtual Section.

In terms of UI, the Section applies to all renditions so when you're changing languages to edit, the Section dropdown selector is entirely separate on the screen.

Internally we'd keep track of the visitor's browser language via the session / cookies. Since it's an essential part of delivering content, it's a grey area in GDPR. We're not tracking a user's browser habits, we're delivering content in their chosen language and if they choose not to participate in cookies/sessions then they'll be forced to receive default language content.

Freeing ourselves of the need for a language marker in the URL would be a pretty sweet step. So even if your article or Section alias is identical to a word in another language ("Taxi", for example), Txp can still serve the correct rendition by reading the session lang parameter or using the section or title to disambiguate. Haven't thought about other permlink schemes. If titles

clash, the schemes without a Section identifier will use the lang identifier of the currently selected visitor's language or the default master lang if none have been selected.

As a concrete example, if the Section name is localised in the URL, we can imply the correct rendition to display by looking up the section name in the alias list, fetching the language to which it's assigned and then using that as the basis for determining which article ID to display.

If the localized Section name is the same for several languages, we fall back on the language used when viewing the previously-displayed article (if such a rendition exists) or ultimately the default language. If all the above fail due to a rendition not being available, it's 404 time.

MLP does this now. If no specific language identifier exists (as a lang marker in the URL or, in the proposed core case, implied via the section alias) then it makes an intelligent stab based on your browser language and sets the session value appropriately.

From that point on, we could use that value as a fallback if at some point it becomes difficult to decide which rendition to display. A failsafe default, if you will.

A couple of new tags for generating a list of site languages / available languages for the current article, a language conditional tag and a 'language' attribute for the tag suite should sort out people's need for displaying multi-lingual content. Some of that code can be ripped from MLP, just done a bit better. Making them container tags would be a major improvement. The lang variable would be set inside the container so any child tags default to it (thus saving you having to add language="ab-cd" to each inner tag unless you need to override it).

Originally, the txp_lang table was considered for all of this, but it's messy at best. That should probably be reserved for doing what MLP terms 'snippets': single strings that exist in multiple languages and can be inserted into templates via the built-in <txp:text> tag. A lang string manager panel (or a plugin like smd_babel) for performing CRUD operations on language strings is a boon here. That would allow strings to be tidied up or to tweak translation strings and make Textpacks to share, etc.

Feeds can be beefed up to target per-language feeds. A preference could govern whether an update to a "collected" article triggers a change to the LastMod date stamp of all renditions, or just the single article.

We should also think about repurposing (or extending) the article status fields to make more of a publishing workflow, so you can subscribe to be notified if something needs translating (e.g. the Draft state in your preferred language(s)?) and possibly put it in the 'done' (pending?) status ready for final editing / publication to Live. Not sure how to do this exactly. Probably extend the core feature set (see the status-mods branch) and let plugins take over.

A 'subscribe to changes in language ab-cd' feature might be handy so authors could choose to be notified if a new article is created and does not have a rendition in their chosen lang, as well as if an article is purposely put into 'draft' mode by another user in your subscribed language.

Alternatively, when an article is first created it could be cloned automatically into all available languages with their status bits set to Draft. That action would automatically trigger an email to subscribed users of each language.

This type of functionality seems as if it should be deferred to plugins as after-market workflow enhancements tied to after_clone/after_save callback hooks. We should simply make it possible in core through callbacks.

The article list panel could highlight the "not yet cloned article", perhaps in a different background colour and/or symbology. So a given author browsing the article list could immediately pick out which ones need a translation. Searching for untranslated content might also be an option.

The overall vision is that the only way to serve front-side multi-lingual content is by installing a back-end language pack. Not an unreasonable constraint. That convention means we have the admin-side UI covered for people who want to edit in their native language and adds some nice guarantees and fallbacks.

Multilingual websites can also combine several languages on one page. The implications of that are not yet known. For example, with the upcoming CSS hyphens property there ought to be a mechanism to label HTML sections or strings. (Textile seems to be prepared: http://txstyle.org/doc/26/language, also for four letter codes on inline tags. If, however, it needed numeric ones like es-419 for Latin American Spanish, Textile would have to be adapted to accept those). But language codes would have to be output also by Txp tags.

It sounds big, but it's not a massive amount of coding effort (compared to other things at least). There's a possibility that the 'collection' idea could actually be abused for other purposes. Maybe someone who wants to make multi-step articles, or to draft 'versions' of content prior to release. But maybe that's too ambitious and it should just be used for the single job of offering a multi-lingual Txp experience.

Questions, comments, constructive criticism, improvements, and of course, thoughts on how to achieve any or all of this in practical terms welcome.