Internationalization

Introduction

eZ Platform offers the ability of creating different language versions of all content in the repository.

In this topic:

Using multiple languages

The system allows for multiple language versions (translations) of a Content item to be created. Translations are created per version of the item, so each version of the content can have a different set of translations.

At minimum, a version always has one translation which by default is the *initial/main* translation. Further versions can be added, but only for languages that have previously been added to the global translation list, that is a list of all languages available in the system. The maximum number of languages in the system is 64.

Different translations of the same Content item can be edited separately. This means that different users can work on translations into different languages at the same time.

Translatable and untranslatable Fields

Language versions actually concern translating values of the Fields of a Content item. Every Field in its definition in a Content Type can be set to be Translatable or not.

Translating the Field values is natural in some cases, for example for the body of an article. However, there are Fields where translating is impractical, for instance images without text, numbers, e-mail addresses and so on. Platform still gives the possibility to mark all Fields as translatable regardless of their Field Type. It is only the user's (content manager's) decision to exclude the translation possibility for those Fields where it makes no sense.

When a Field is not flagged as Translatable, its value will be copied from the initial/main translation when a new language version is created. This copied value cannot be modified. When a Field is Translatable, its value in a new language version will have to be entered by the user.

For example, let's say that you need to store information about marathon contestants and their results. You build a "contestant" Content Type using the following Fields: name, photo, age, nationality, time reached. Allowing the translation of anything else than nationality would be pointless, since the values stored by the other Fields are the same regardless of the language used to describe the runner. In other words, the name, photo, age and time reached would be the same in, for example, both English and Norwegian.

Access control

It is possible to control whether a User or User group is able to translate content or not. This can be done by adding a Language limitation to Policies that allow creating or editing content. This limitation lets you define which Role can work with which languages in the system. (For more information of the permissions system, see Permissions.)

In addition, it is also possible to control access to the global translation list by using the Content/Translations Policy. This makes it possible to allow users other than the site administrator to add and remove translations that can be used.

How to create different language versions?

The multilanguage system operates based on a global translation list that contains all languages available in the installation. Languages can be added to this list from the Admin Panel in the user interface. **The new language must then be added to the SiteAccess configuration**. Once this is done, any user with proper permissions can create Content item versions in these languages in the user interface.

How to make translations available to the visitor?

Once more than one language is defined in the global translation list and there is content in different languages, the question is how can this be exposed to use by the visitor. There are two ways to do this:

1. Implement a mechanism called **Language Switcher**. It lets you create links that allow switching between different translations of a Content item.
2. If you want to have completely separate versions of the website, each with content in its own language, you can use siteaccesses. In this case, depending on the uri used to access the website, a different site will open, with a language set in configuration settings. All Content items will then be displayed in this primary language.

Multilanguage and permissions

LanguageLimitation allows you to limit users' editing rights depending on the language of the Content item. See Permissions and List of Limitations for more information.

Configuration

One of the basic ways of using multiple languages is setting up a separate siteaccess for each language.

Explicit *translation siteaccesses*

Configuration is not mandatory, but can help to distinguish which siteaccesses can be considered *tr anslation siteaccesses*.

ezplatform.yml

```
ezpublish:
    siteaccess:
        default_siteaccess: eng
        list:
            - ezdemo_site
            - eng
            - fre
            - ezdemo_site_admin

        groups:
            ezdemo_frontend_group:
                - ezdemo_site
                - eng
                - fre

    # ...

    system:
        # Specifying which SiteAccesses are used for
translation
        ezdemo_frontend_group:
            translation_siteaccesses: [fre, eng]
        eng:
            languages: [eng-GB]
        fre:
            languages: [fre-FR, eng-GB]
        ezdemo_site:
            languages: [eng-GB]
```

Note: Top prioritized language is always used for as the siteaccess language reference (e.g. `fre-FR` for `fre` siteaccess in the example above).

If several translation siteaccesses share the same language reference, **the first declared siteaccess always wins**.

More complex translation setup

There are some cases where your siteaccesses share settings (repository, content settings, etc.), but you don't want all of them to share the same `translation_siteaccesses` setting. This can be for example the case when you use separate siteaccesses for mobile versions of a website.

Solution is as easy as defining new groups:

ezplatform.yml

```
ezpublish:
    siteaccess:
        default_siteaccess: eng
        list:
            - ezdemo_site
            - eng
            - fre
            - mobile_eng
            - mobile_fre
            - ezdemo_site_admin

        groups:
            # This group can be used for common front
settings
            ezdemo_common_group:
                - ezdemo_site
                - eng
                - fre
                - mobile_eng
                - mobile_fre

            ezdemo_frontend_group:
                - ezdemo_site
                - eng
                - fre

            ezdemo_mobile_group
                - mobile_eng
                - mobile_fre

    # ...

    system:
        # Translation SiteAccesses for regular frontend
        ezdemo_frontend_group:
            translation_siteaccesses: [fre, eng]

        # Translation SiteAccesses for mobile frontend
        ezdemo_mobile_group:
            translation_siteaccesses: [mobile_fre,
mobile_eng]

        eng:
            languages: [eng-GB]
        fre:
            languages: [fre-FR, eng-GB]
        ezdemo_site:
            languages: [eng-GB]

        mobile_eng:
            languages: [eng-GB]
        mobile_fre:
            languages: [fre-FR, eng-GB]
```

Using implicit *related siteaccesses*

If `translation_siteaccesses` setting is not provided, implicit *related siteaccesses* will be used instead. Siteaccesses are considered *related* if they share:

- The same repository
- The same root location Id (see Multisite feature)

Fallback languages and missing translations

When setting up siteaccesses with different language versions, you can specify a list of preset languages for each siteaccess. When this siteaccess is used, the system will go through this list. If a Content item is unavailable in the first (prioritized) language, it will attempt to use the next language in the list, and so on. Thanks to this you can have a fallback in case of a lacking translation.

You can also assign a Default content availability flag to Content Types (available in the Admin Panel). When this flag is assigned, Content items of this type will be available even when they do not have a language version in any of the languages configured for the current siteaccess.

> Note that if a language is not provided in the list of prioritized languages and it is not the Content item's first language, the URL alias for this content in this language will not be generated.
>
> This is different than in legacy, where this behavior was covered by a global control switch `ShowUntranslatedObjects`.

Usage

Language Switcher

Description

A Content item can be translated into several languages. Those languages are configured in the system and exposed in siteaccesses via a prioritized list of languages:

ezplatform.yml

```
ezpublish
    system:
        eng:
            languages: [eng-GB]
        # In fre siteaccess, fre-FR is always preferred,
and fallback to eng-GB if needed.
        fre:
            languages: [fre-FR, eng-GB]
```

When visiting a Content item, it may be useful to let the user switch from one translation to another, more appropriate to them. This is precisely the goal of the language switcher.

The language switcher relies on the Cross-SiteAccess linking feature to generate links to the Content item's translation, and on RouteReference feature.

> **Tip**
> If you install the DemoBundle with at least 2 different languages, you will be able to see the Language Switcher and to test it.

In a template

To generate a language switch link, you need to generate the `RouteReference`, with the `langua ge` parameter. This can easily be done with `ez_route()` Twig function:

```
{# Given that "location" variable is a valid Location
object #}
<a href="{{ url( ez_route( location, {"language":
"fre-FR"} ) ) }}">{{ ez_content_name( content ) }}</a>

{# Generating a link to a declared route instead of
Location #}
<a href="{{ url( ez_route( 'my_route', {"language":
"fre-FR"} ) ) }}">My link</a>
```

You can also omit the route, in this case, the current route will be used (i.e. switch the current page):

```
{# Using Twig named parameters #}
<a href="{{ url( ez_route( params={"language": "fre-FR"}
) ) }}">My link</a>

{# Identical to the following, using ordered parameters
#}
<a href="{{ url( ez_route( null, {"language": "fre-FR"}
) ) }}">My link</a>
```

Using sub-requests

When using sub-requests, you lose the context of the master request (e.g. current route, current location, etc.). This is because sub-requests can be displayed separately, with ESI or Hinclude.

If you want to render language switch links in a sub-request with a correct `RouteReference`, you must pass it as an argument to your sub-controller from the master request.

```
{# Render the language switch links in a sub-controller
#}
{{ render( controller(
'AcmeTestBundle:Default:languages', {'routeRef':
ez_route()} ) ) }}
```

```
namespace Acme\TestBundle\Controller;

use eZ\Bundle\EzPublishCoreBundle\Controller;
use eZ\Publish\Core\MVC\Symfony\Routing\RouteReference;

class DefaultController extends Controller
{
    public function languagesAction( RouteReference
$routeRef )
    {
        return $this->render(
'AcmeTestBundle:Default:languages.html.twig', array(
'routeRef' => $routeRef ) );
    }
}
```

```
{# languages.html.twig #}

{# Looping over all available languages to display the
links #}
{% for lang in ezpublish.availableLanguages %}
    {# This time, we alter the "siteaccess" parameter
directly. #}
    {# We get the right siteaccess with the help of
ezpublish.translationSiteAccess() helper #}
    {% do routeRef.set( "siteaccess",
ezpublish.translationSiteAccess( lang ) ) %}
    <a href="{{ url( routeRef ) }}">{{ lang }}</a><br />
{% endfor %}
```

- `ezpublish.translationSiteAccess( language )` returns the siteaccess name for provided language (or `null` if it cannot be found)
- `ezpublish.availableLanguages()` returns the list of available languages.

Using PHP

You can easily generate language switch links from PHP too, with the `RouteReferenceGenerator` service:

```
// Assuming we're in a controller
/** @var
\eZ\Publish\Core\MVC\Symfony\Routing\Generator\RouteRefe
renceGeneratorInterface $routeRefGenerator */
$routeRefGenerator = $this->get(
'ezpublish.route_reference.generator' );
$routeRef = $routeRefGenerator->generate( $location,
array( 'language' => 'fre-FR' ) );
$link = $this->generateUrl( $routeRef );
```

You can also retrieve all available languages with the `TranslationHelper`:

```
/** @var \eZ\Publish\Core\Helper\TranslationHelper
$translationHelper */
$translationHelper = $this->get(
'ezpublish.translation_helper' );
$availableLanguages =
$translationHelper->getAvailableLanguages();
```

`ez_trans_prop` Twig helper

`ez_trans_prop()` is a generic, low level Twig helper which corrects the translated value of a multi valued(translations) property.

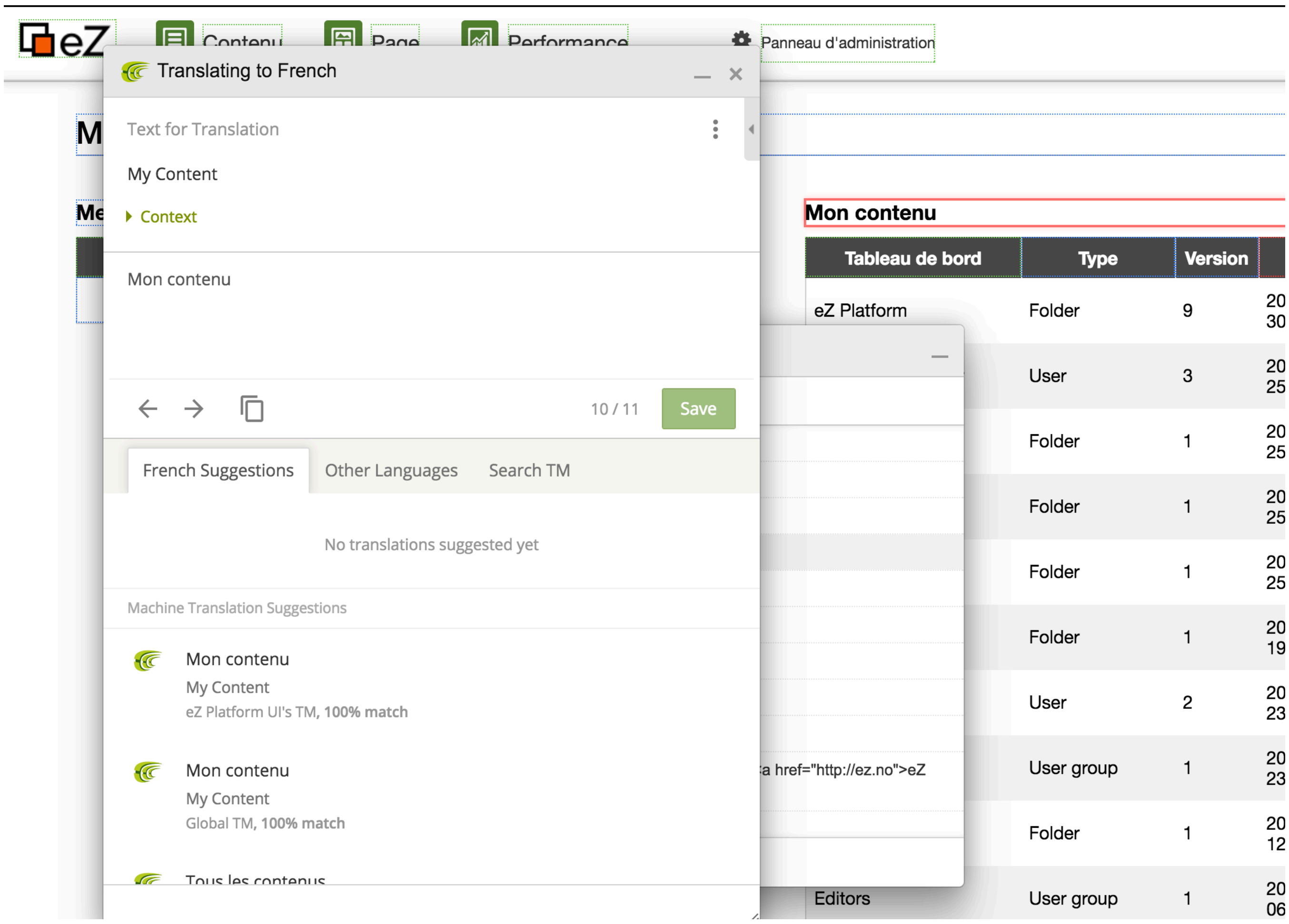
See ez_trans_prop reference for more information.

Translating UI of eZ Platform

In-context UI translation

V1.8

Since eZ Platform 1.7.0, the interface has been fully translatable. Version 1.8.0 introduces official support for Crowdin as a translation management system. In addition, it integrates support for in-context translation, a feature that allows you to translate strings from the interface, *in context*.



Translating to French

Text for Translation

My Content

Context

Mon contenu

10 / 11 Save

French Suggestions | Other Languages | Search TM

No translations suggested yet

Machine Translation Suggestions

Mon contenu
My Content
eZ Platform UI's TM, **100% match**

Mon contenu
My Content
Global TM, **100% match**

Tous les contenus

Mon contenu

Tableau de bord	Type	Version	
eZ Platform	Folder	9	20 30
	User	3	20 25
	Folder	1	20 25
	Folder	1	20 25
	Folder	1	20 25
	Folder	1	20 19
	User	2	20 23
	User group	1	20 23
	Folder	1	20 12
Editors	User group	1	20 06

a href="http://ez.no">eZ

Toggling in-context translation

To start translating, you need to set a browser cookie. There are several ways to do this, but we will highlight a couple here.

Using the debugging console

One way is to open the development console and run these lines:

To enable it:

Enable the In-Context Translation

```
document.cookie='ez_in_context_translation=1;path=/;';
location.reload();
```

To disable it:

Disable the In-Context Translation

```
document.cookie='ez_in_context_translation=;expires=Mon,
05 Jul 2000 00:00:00 GMT;path=/;'; location.reload();
```

Using bookmarks

You can easily create two bookmarks to toggle in-context on/off.
Right-click your browser's bookmark bar, and create a new one, with the following label and link:

Adding a bookmark to enable in-context translation

Enable Bookmark

```
javascript:(function()
{document.cookie='ez_in_context_translation=1;path=/;';
location.reload();})()
```

Adding a bookmark to disable in-context translation

Disable bookmark

```
javascript:(function()
{document.cookie='ez_in_context_translation=;expires=Mon
, 05 Jul 2000
00:00:00 GMT;path=/;'; location.reload();})()
```

Then click on the bookmarks from platform UI to enable/disable in-context.

Using in-context translation

The first time you enable in-context, if you're not logged into Crowdin, it will ask you to log in or register an account. Once done, it will ask you which language you want to translate to, from the list of languages configured in Crowdin.

Choose your language, and you can start translating right away. Translatable strings in the interface will be outlined in red (untranslated), blue (translated) or green (approved). When moving over them, an edit button will show up on the top left corner of the outline. Click on it, and edit the string in the window that shows up.

Troubleshooting

Make sure you clear your browser's cache in addition to eZ Platform's. Some of the translation resources use aggressive HTTP cache.

More info...
Crowdin: https://crowdin.com
In-context: https://support.crowdin.com/in-context-localization/