TYPO3 EXT:campus_lingo

© 2026 Brain Appeal GmbH, Mannheim
This documentation is for EXT:campus_lingo 4.0.0
Last Modified: 2026-08-20

What is EXT:campus_lingo for?

- Auto-translates all records for language fallbacks to default language via Campus Lingo API.
- Respects backend translations.
- Auto-detect updates of default language contents and/or field configuration changes.
- Only re-translates on detection of changes, minimizing requests to the API (and therefore costs) via intelligent caching.

Requirements

- ☐ PHP 8.1-8.5
- ☐ TYPO3 CMS (Versions 12 LTS, 13 LTS and 14 LTS are supported)

How do I get set up?

1. Install Extension TYPO3 via Extension Manager or by composer update
2. Configuration via Extension Configuration
   At least: **Enter your Campus Lingo API key.**
3. Configuration via Sites backend module (or the site's config.yaml)
   At least: **Languages must be configured to fall back to the default language in order to translate them automatically.**
   EXT:campus_lingo enriches the languages in the Sites backend module with help depicting configuration problems.
4. Optionally, configure pages that must not be translated via page properties in page's "Language" register.
5. Something isn't working? Have a look at the section Troubleshooting, the FAQs and Release Notes or contact us.

Advanced integration features

For the realization of feature requests that aren't covered by backend interfaces, this document contains a section that is aimed at TYPO3 integrators or developers. For example, you will find here how to:

- Show your visitors, whether a page is automatically translated.
- Customize the automatic translation of content stored in certain tables and fields.
- Automatically translate content saved in FlexForms.
- Prevent automatic translations using a PSR-14 event.
- Use EXT:campus_lingo commands.
- Automatically translate an extension's locallang.xlf.
- Translate "arbitrary" content that isn't stored in the TYPO3 database.
- Custom route enhancers with enabled slug translation.

Support

If you have any questions, please contact us via our ticket system or your contact person at Brain Appeal GmbH, Mannheim.
In case you don't have a login or contact person you can reach us via email.

Page properties to toggle automatic translation

EXT:campus_lingo adds the form element “Automatic translation” to the page properties on register “Language”.
This form element provides two checkboxes depending on the current activation status:

- **For the current page**
  If active: Don’t show automatic translated content on this page
  If inactive: Enable automatic translated content on this page
- **For subpages of the current page**
  If active: Don’t show automatic translated content on subpages
  If inactive: Enable automatic translated content on subpages

The form element on pages “deeper” in the TYPO3 Pagetree respects settings of their parent pages.
Therefore, (de-)activation for subpages may be activated (or deactivated resp.) for sub-subpages again.

The form element contains an information panel to show, if the automatic translation is activated for the current page and for its subpages.
To avoid confusion, this panel also provides the information of the parent page, where the (de-)activation for the current page is set. A link to edit this parent page directly is also provided.

Disable automatic translation per language

When a disable flag (Don’t show automatic translated content on this page or Don’t show automatic translated content on subpages) is active, an additional field **“Languages that are not automatically translated”** (`tx_campus_lingo_features_languages`) appears on the same page properties register.

This field is a multi-select checkbox list that is populated dynamically with all languages configured for the current site. Editors can select individual languages for which automatic translation should be suppressed, while keeping it active for all other languages.

- **No languages selected** — automatic translation is disabled for all languages (equivalent to the previous, global disable behaviour).
- **One or more languages selected** — automatic translation is disabled only for the selected languages; the remaining languages continue to be translated automatically.

The backend status indicator now distinguishes three states:

State	Meaning
Enabled	Automatic translation is active for all languages on this page.
Partially enabled	Automatic translation is disabled for at least one language but active for others.
Disabled	Automatic translation is suppressed for all languages on this page.

Configuration

Site Configuration

All languages

- Campus Lingo Language Code `site_language.[].campus_lingo_lang_code`
  If set, this code is used for Campus Lingo. Else, the "Two Letter ISO Code" is used.
  Examples when using DeepL: EN-GB, EN-US, PT-BR, PT-PT (see DeepL documentation).

Default language

- Campus Lingo needs "Opt-In" `site_language.0.campus_lingo_opt_in`
  If enabled, no automatic translation takes place on this site without configuring the page properties accordingly (see section above).
  Automatic translation can be switched on in the page properties for the respective page or subpages.
- Campus Lingo translation service can be deactivated `site_language.0.campus_lingo_translation_service_toggle_enabled`
  If enabled, usage of the translation service can be switched off in the page properties' form element "Automatic translation". Automatically translated content in the EXT:campus_lingo cache will still be used. However, no new translations are obtained from the Campus Lingo API.
- Hide untranslated pages `site_language.0.campus_lingo_hidePageIfNotTranslated`
  Hides pages that are neither translated manually nor automatically.
  If activated, the checkbox "Hide page if no translation for current language exists" is removed from page properties.

Additional languages (i.e. translations)

- Fallback Type `site_language.[].fallbackType`
  Must be set to `Strict` or `Fallback`.
  Attention The Fallback Type `Free mode` isn't supported. It isn't planned to support it.

Extension Manager Configuration

Campus Lingo API

- `campusLingoApiKey`
  Campus Lingo API key: If you don't have a Campus Lingo API key, visit academic-hero.de/produkte/campus-lingo
- `campusLingoApiEndpoint = https://api.campus-lingo.com/api.php`
  API endpoint
- `feature_disableSubscription = 0`
  Disable automatic subscription: EXT:campus_lingo by default automatically subscribes its endpoints at Campus Lingo API. If these endpoints aren't accessible (e.g. due to firewall or password protection), you may want turn off this automatic subscription. For the features provided by these endpoints (e.g. automatic updates of glossary entries), you must manually configure those endpoints in Campus Lingo API.

Translation

- `formality = (default|more|less)`
  Formality: Sets whether the translated text should lean towards formal or informal language. This feature currently only works for target languages "DE", "FR", "IT", "ES", "NL", "PL", "PT-PT", "PT-BR", "JA" and "RU".
- `translateFields_allowLanguageSynchronization`
  Translate optional fields: If set, fields that are marked as optionally translatable are automatically translated, too. Technical background: Activating this option will translate fields with TCA "allowLanguageSynchronization". If not activated, those fields aren't translated.
- `dontTranslateWrap = ~~~`
  Don't translate wrap: If set, this string can be used to "wrap" text or words (i.e., prepend and append the string) that won't be translated automatically. The wrap is removed before a page is sent to cache or outputted. E.g.: The default setting `~~~` will prevent the translation of `~~~This text isn't translated~~~` and will eventually be outputted as `This text isn't translated`.
- `ignoreTags = notranslate`
  Ignore tags: Content within these tags is to be ignored by automatic translation.
- `feature_enableSlugTranslation = 0`
  Enable translation of slugs: If enabled, slugs (e.g. URL segments) are

also translated automatically. The slugs are then used to generate urls for pages and Extbase records.

- `cacheLifetimeInDays = 0`
  Cache lifetime (days): Controls how long cached translations are stored. Use 0 to disable expiration. Any value greater than 0 defines the number of days before cached entries expire.

Frontend

- `translationInformationPlaceholder = <!-- CAMPUS_LINGO_AUTOTRANSLATION_INFORMATION -->`
  Translation information placeholder: Placeholder for templates, replaced with information if a page is (partially) translated. If some auto translated content is on the current page, this placeholder will be replaced with information about auto translated content (for modifying the HTML, see TypoScript `lib.campusLingo.translationInformation` and Translation State).
- `excludeTablesFromAutotranslationRegisterCount = pages, sys_file_metadata`
  Excluded tables from auto-translation count: Comma separated list of database tables that aren't counted in `TranslationAspect::autoTranslatedContentRecordCount`. This value determines whether the translation information placeholder renders output (see above).
- `strict_excludeTablesFromFallback`
  Tables that mustn't fall back: Comma separated list of database tables that aren't allowed to fallback for auto translation when language has Fallback Type set to "Strict".

Backend

- `feature_showAutoTranslationStateInfo = 0`
  Show status of automatic translation (TYPO3 v12+):
  If enabled, the current auto translations setting for pages and configured records is displayed in the backend. The feature can be disabled per user or usergroup by adding `options.campusLingo.showAutoTranslationStateInfo = 0` in the corresponding TSConfig.
  Backend changes triggered by this option:

  1. The flag icon for the default language of each record is overlaid with a symbol and given an additional title text that describes the status of the automatic translation for the record (Web>Page, Web>List, File>Filelist, IRRE elements).
  2. In the Web>Page module, a status icon with help text about the page's automatic translation status is displayed next to the language selector. Clicking the status icon opens a drop-down menu with links to all translated pages in the frontend, allowing for a preview to the translated pages even without backend access to these languages.
  3. The Localization Overview in the Web>Info module shows the status of automatic translation for each page.

- `feature_enableBackendTranslationQueue = 0`
  Enable backend translation queue: If enabled, records saved in the backend (default language only) are automatically added to a translation queue. The queue is then processed in the background, translating queued records into all configured target languages.
  Important: The queue is **only** processed if a scheduler task for the console command `campuslingo:process-translation-queue` is created in the TYPO3 Scheduler backend module. Without this scheduler task, records will be added to the queue but never translated.
  The command accepts the following options:

  - `--limit=20` Number of records to process per run (default: 20)
  - `--dry-run` Preview which records would be translated without actually translating them

  To initially enqueue all existing default-language records (e.g. after enabling the feature on an existing site), use the command `campuslingo:enqueue-all-records`:

  - `--tables=pages,tt_content` Restrict to specific tables (default: all translatable tables)
  - `--exclude-tables=sys_file_metadata` Exclude specific tables
  - `--dry-run` Preview which records would be enqueued

The main use case for this command is to translate all records before a site goes live. The command only adds the records to the queue, the translation process is handled by the command `campuslingo:process-translation-queue`.

- `feature_disableChoicesNoTransS = 0`
  Deactivate “disable translation service”: Deactivates the possibility to configure the usage of the “Campus Lingo translation service can be deactivated” in sites configuration. If checked, disabling the usage of translation service while still using EXT:campus_lingo translation cache isn’t possible at all. If not checked, disabling the usage of translation service while still using EXT:campus_lingo translation cache via page properties may be enabled via sites configuration.
- `feature_reuseCacheItemsOnLocalization = 0`
  Use existing cache items for backend localization: When this option is enabled, the translation cache is consulted during backend localization. If a cached translation for the target language exists, all translatable fields are pre-populated with the available values. This function only reuses existing cache items. It does not translate the content of the record.

Debug

- `debugOutputTimes`
  Output times: Outputs summed time in seconds needed for API calls and getting translations from cache. Times are outputted as HTML comment before the closing `</body>`.
- `debugAdditionalTranslationInformation`
  Debug Output of additional translation information: Outputs additional information about translated content as HTML comment below the Translation Information Placeholder.
- `debugLogRepeatingMessages`
  Log repeating messages to System > Log: If enabled, the same log message of campus_lingo is written to the table sys_log (System > Log) on each occurrence. If disabled, the same log message is only written to sys_log, if its last occurrence is older than an hour.

Development

Please see section Translate locallang.xlf files.

Troubleshooting / incompatible configurations

If (some) pages aren't translated automatically after installing EXT:campus_lingo, ensure to **disable** the following settings: Under certain circumstances, they may hide untranslated pages before campus_lingo is able to translate them.

Disable inverse control of hidden translations

Ensure that the following setting is not set, since campus_lingo isn't fully tested with this setting turned on:

```
$GLOBALS['TYPO3_CONF_VARS']['FE']['hidePagesIfNotTranslatedByDefault'] = 1;
```

See Inverse control of hidden translations at "Frontend Localization Guide" on docs.typo3.org

Field "Localization" of pages

Ensure that the field "Localization" of table pages `pages.l18n_cfg` isn't set to 2, i.e. "Hide page if no translation for current language exists" shouldn't be checked. Furthermore, ensure that `pages.l18n_cfg = 2` isn't set by default (e.g. by `TCAdefaults.pages.l18n_cfg = 2`).
campus_lingo works best without this setting, since it isn't fully tested for compatibility with l18n_cfg=2.

For Developers & Integrators

Output information about auto-translated content in frontend

Insert the following placeholder to your template

```
<!-- CAMPUS_LINGO_AUTOTRANSLATION_INFORMATION -->
```

If there is auto translated content on the current page, the placeholder will show the information, that (some) content on the current page is auto translated.
If no auto translated content is used, the placeholder is removed.

To modify HTML and contents shown by the placeholder, modify the TypoScript object `lib.campusLingo.translationInformation`.

Translation State (TranslationAspect / Context API)

EXT:campus_lingo stores the current page's translation state in `BrainAppeal\CampusLingo\Context\TranslationAspect`, which is registered as a custom aspect in TYPO3's Context API under the name `campus_lingo_translation`. This replaces the former TSFE register approach (removed in TYPO3 14) and works identically in TYPO3 12, 13, and 14.

The following properties are available on the aspect:

Property	Type	Getter	Description
`pageIsAutoTranslated`	`bool`	`isPageIsAutoTranslated()`	Page record itself is auto-translated
`pageUsesAutoTranslatedPages`	`bool`	`isPageUsesAutoTranslatedPages()`	Any page record used on this page (e.g. in menus) is auto-translated
`autoTranslatedContentRecordCount`	`int`	`getAutoTranslatedContentRecordCount()`	Number of auto-translated records on the current page (excluding page records)
`manualTranslatedContentRecordCount`	`int`	`getManualTranslatedContentRecordCount()`	Number of manually translated records on the current page (excluding page records)

Property	Type	Getter	Description
`autoTranslatedContentRecords`	`array`	`getAutoTranslatedContentRecords()`	`table.uid` strings of auto-translated records
`manualTranslatedContentRecords`	`array`	`getManualTranslatedContentRecords()`	`table.uid` strings of manually translated records

Accessing the aspect in PHP

```
$aspect = \BrainAppeal\CampusLingo\Context\ContextManager::getInstance()-
        >getTranslationAspect();
$isAutoTranslated = $aspect->isPageIsAutoTranslated();
$count = $aspect->getAutoTranslatedContentRecordCount();
```

Accessing the aspect in Fluid templates

The ViewHelper `BrainAppeal\CampusLingo\ViewHelpers\GetTranslationAspectViewHelper` returns the current `TranslationAspect` object directly, so all its getters are accessible as Fluid properties:

```
<html xmlns:cl="http://typo3.org/ns/BrainAppeal/CampusLingo/ViewHelpers"
      data-namespace-typo3-fluid="true">

<f:variable name="translationAspect" value="{cl:getTranslationAspect()}"/>
<f:if condition="{translationAspect.pageIsAutoTranslated} ||
        {translationAspect.contentAutoTranslatedRecordCount} > 0">
    This page is (partially) auto-translated.
</f:if>
```

> **Note:** The TypoScript object `lib.campusLingo.translationInformation` no longer passes register values as TypoScript variables. The built-in template `Resources/Private/Templates/TranslationInfo.html` uses the `{cl:getTranslationAspect()}` ViewHelper directly. Custom templates that previously relied on TypoScript variables such as `{pageIsAutoTranslated}` must be adapted accordingly.

Customize tables and fields

1) Add option to disable auto-translation for records of a certain table `your_table`

In *ext_localconf.php* or *Configuration/TCA/Overrides/your_table.php* add

```
if (\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::isLoaded('campus_lingo')) {
    \BrainAppeal\CampusLingo\Utility\TCA\ApiRecordTranslations::addAutoTranslationSettin
        gs(
        'your_table'
    );
}
```

2) Enforce auto-translation of a certain field `your_table.your_field`

In *Configuration/TCA/Overrides/your_table.php* add

```
$GLOBALS['TCA']['your_table']['columns']['your_field']['autotranslate_enforce'] = true;
```

3) Prohibit auto-translation of a certain field `your_table.your_field`

Beware: `autotranslate_enforce` has priority.
In *Configuration/TCA/Overrides/your_table.php* add

```
$GLOBALS['TCA']['your_table']['columns']['your_field']['autotranslate_forbidden'] = true;
```

4) Preprocess/Postprocess records - PSR-14 TranslationServiceProcessRowEvent

As an example, have a look at the EventListener `BrainAppeal\CampusLingo\EventListener\CampusLingo\TranslationServiceProcessRowFlexFormEventListener` that implements auto-translation of FlexForm properties.
Following this example, this Event should enable you to implement the auto-translation of data stored sort of “unstructured” to database as being the case for the FlexForms’ XML.

5) Set/modify TCA on the fly: PSR-14 TranslationServiceModifyTCAEvent

As an example, have a look at the EventListener `BrainAppeal\CampusLingo\EventListener\CampusLingo\TranslationServiceModifyTCAEventListener`

The following snippets ready for copy&paste ensure auto-translation of fields without the need of modifying (or defining) `$GLOBALS['TCA']` for a table.

your_extension/Configuration/Services.yaml

```
services:

        Vendor\YourExtension\EventListener\CampusLingo\TranslationServiceModifyTCAEventL
        istener:
    tags:
      - name: event.listener
        identifier: 'yourExtension/campusLingo/translationServiceModifyTCA'
        event: BrainAppeal\CampusLingo\Event\TranslationServiceModifyTCAEvent
```

your_extension/Classes/EventListener/CampusLingo/
TranslationServiceModifyTCAEventListener.php

```
<?php

namespace Vendor\YourExtension\EventListener\CampusLingo;
use BrainAppeal\CampusLingo\Event\TranslationServiceModifyTCAEvent;

class TranslationServiceModifyTCAEventListener
{
    /**
     * sets or modifies TCA for the auto-translation process on table 'your_table'
     * to ensure that fields 'field_a' and 'field_b' are auto-translated
     * @param TranslationServiceModifyTCAEvent $event
     */
    public function __invoke(TranslationServiceModifyTCAEvent $event): void
    {
        if ('your_table' === $event->getTable()) {
            $fieldsToTranslate = ['field_a', 'field_b'];
            $tcaOverride = [];
            foreach ($fieldsToTranslate as $fieldToTranslate) {
                $tcaOverride['columns'][$fieldToTranslate]['autotranslate_enforce'] = tr
        ue;
            }
            $event->setTCA(array_replace_recursive(
                ($event->getTCA() ?? []),
                $tcaOverride
            ));
        }
    }
}
```

6) Configure auto-translation for FlexForm fields

To enable auto-translation of content stored in FlexForms, additional configuration is required. You must configure, which properties of a FlexForm shall be auto-translated.
Configuration is registered in TCA. For ease of use, the following API method exists which will ensure the needed TCA:

In `ext_localconf.php` or `Configuration/TCA/Overrides/your_table.php` add

```
if (\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::isLoaded('campus_lingo')) {
    \BrainAppeal\CampusLingo\Utility\TCA\ApiRecordTranslations::addAutoTranslationForFle
        xFormField(
        $table = 'your_table',

        // field containing FlexForm data (e.g. pi_flexform for table tt_content)
        $fieldName = 'pi_flexform',

        // array of FlexForm properties to auto-translate
        // or
        // "compact notation": An associative array with
        // - keys representing the dataStructureKey
        // - values of keys are arrays of FlexForm properties to auto-translate
        $flexFormProperties = [
            'settings.yourFlexformKey'
        ],

         // $dataStructureKey isn't needed for the compact notation.
         // It is optional recommended for specific $flexFormProperties
         // auto-translation behaviour depending on the given data structure.
        $dataStructureKey = '*'
    );
}
```

FlexForm sections can be configured either as a whole or by using `*` for their dynamic item keys. Configuring a section property directly recursively translates all scalar values below it. To translate only selected fields of every section item, use wildcard paths:

```
\BrainAppeal\CampusLingo\Utility\TCA\ApiRecordTranslations::addAutoTranslationForFlexFor
        mField(
    'tt_content',
    'pi_flexform',
    [
        'settings.items.*.container.title',
        'settings.items.*.container.description',
    ],
    '*,your_plugin_signature'
);
```

Examples for auto translation of FlexForms

The following examples are based on the infamous TYPO3 extension “powermail”.

To auto-translate the subject and body of emails to sender (“Mail to User”) and the thank you text (“Text on submit page”), insert the following into `ext_localconf.php` or `Configuration/TCA/Overrides/tt_content.php`

```
if (\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::isLoaded('campus_lingo')) {
    \BrainAppeal\CampusLingo\Utility\TCA\ApiRecordTranslations::addAutoTranslationForFle
        xFormField(
        'tt_content',
        'pi_flexform',
        [
            'settings.flexform.sender.subject',
            'settings.flexform.sender.body',
            'settings.flexform.thx.body'
        ],
        '*,powermail_pi1'
    );
}
```

or using the compact notation with an equivalent result

```
if (\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::isLoaded('campus_lingo')) {
    \BrainAppeal\CampusLingo\Utility\TCA\ApiRecordTranslations::addAutoTranslationForFle
        xFormField(
        'tt_content',
        'pi_flexform',
        [
            '*,powermail_pi1' => [
                'settings.flexform.sender.subject',
                'settings.flexform.sender.body',
                'settings.flexform.thx.body'
            ]
        ]
    );
}
```

Prevent auto-translations dynamically: PSR-14 PreventAutoTranslationsEvent

The event `BrainAppeal\CampusLingo\Event\PreventAutoTranslationsEvent` can be used to prevent automatic translations dynamically based on custom logic, such as the current table or the requested target language.

Available methods on the event:

- `getTable()`: Returns the table name (or an empty string for global checks, see `isGlobalAutoTranslationCheck()` below).
- `getRecord()`: Returns the untranslated record data as array (might be an empty array for global checks).
- `getCurrentSite()`: Returns the current site (might be empty, e.g. if called via LanguageOverlayService::autoTranslateRow).
- `getTargetLanguageAspect()`: Returns the `TYPO3\CMS\Core\Context\LanguageAspect` of the target language.
- `getTargetLanguageId()`: Returns the target language id, if available (wrapper for `getTargetLanguageAspect()`).
- `isGlobalAutoTranslationCheck()`: Returns true if the translation request is not related to a specific table.
- `preventAutoTranslations()`: Disables automatic translation for the current translation request.
  This method implements a **final veto**: Once called, automatic translations are disabled for the current translation request and the event is stopped. This means, that event listeners with a lower priority won't be processed and can't unset the veto.

Example for the PreventAutoTranslationsEvent

your_extension/Classes/EventListener/CampusLingo/PreventAutoTranslationsEventListener.php

```php
<?php

namespace Vendor\YourExtension\EventListener\CampusLingo;
use BrainAppeal\CampusLingo\Event\PreventAutoTranslationsEvent;

class PreventAutoTranslationsEventListener
{
    public function __invoke(PreventAutoTranslationsEvent $event): void
    {
        // Example 1: Disable automatic translation for news if the requested language has the language id 2
        if ($event->getTable() === 'tx_news_domain_model_news' && $event->getTargetLanguageId() === 2) {
            $event->preventAutoTranslations();
        }
        // Example 2: Disable automatic translations if the requested language has the language id 3
        if ($event->getTargetLanguageId() === 3) {
            $event->preventAutoTranslations();
        }
        // Example 3: Disable automatic translations for a content element if it's layout is 66
        // beware: availability of record data is not guaranteed within this event
        $recordLayout = $event->getRecord()['layout'] ?? null;
        if ($event->getTable() === 'tt_content' && $recordLayout === 66) {
            $event->preventAutoTranslations();
        }
    }
}
```

your_extension/Configuration/Services.yaml

```yaml
services:
  Vendor\YourExtension\EventListener\CampusLingo\PreventAutoTranslationsEventListener:
    tags:
      - name: event.listener
        identifier: 'yourExtension/campusLingo/preventAutoTranslations'
        event: BrainAppeal\CampusLingo\Event\PreventAutoTranslationsEvent
```

Commands

campuslingo:delete-manual-translations

The console command `campuslingo:delete-manual-translations` can be used to delete existing editorial translations for pages or subtrees. This command may be handy in order to delete certain editorial translations that are henceforth ought to be translated automatically (e.g. due to “stuck” or outdated editorial translations).

Use `./vendor/bin/typo3 campuslingo:delete-manual-translations --help` for more information.

campuslingo:process-translation-queue

Normally the automatic translation process is only triggered by calling a page in the frontend. This may cause longer loading times if many records are to be translated. If the feature `feature_enableBackendTranslationQueue = 1` is enabled and console command `campuslingo:process-translation-queue` is executed regularly (e.g. via a scheduler task) to process the translation queue and translate records that are added to the queue when saved in the backend.

Use `./vendor/bin/typo3 campuslingo:process-translation-queue --help` for more information.

campuslingo:enqueue-all-records

The console command `campuslingo:enqueue-all-records` can be used to enqueue all records for automatic translation. The translation queue is then processed by the console command `campuslingo:process-translation-queue` as described above. This command is especially useful to enqueue all existing records after enabling automatic translations on an existing site.

Use `./vendor/bin/typo3 campuslingo:enqueue-all-records --help` for more information.

campuslingo:cleanup-translation-cache

The console command `campuslingo:cleanup-translation-cache` removes obsolete entries from the translation cache table `tx_campus_lingo_translation_cache`. A cache entry is considered obsolete if any of the following applies to the record it references via `translated_table` + `translated_id`:

- `referenced_table_unknown`: the referenced table is no longer registered in TCA.
- `referenced_record_missing`: the referenced record uid no longer exists in its table.

- `referenced_record_deleted`: the referenced record is marked as deleted via the TCA `ctrl/delete` flag.
- `expired`: the cache row's `tstamp` is older than the extension setting `cacheLifetimeInDays` (only evaluated when `cacheLifetimeInDays > 0`).

Optional reasons (off by default, opt-in via flag):

- `--respect-hidden-restriction` also deletes cache items whose referenced record is currently hidden (TCA `enablecolumns/disabled`).
- `--respect-time-restrictions` also deletes cache items whose referenced record is outside its visibility window (TCA `enablecolumns/starttime` / `enablecolumns/endtime`).

Use `--dry-run` to preview what would be deleted without modifying the database. The command always prints a summary table grouped by referenced table and reason:

```
./vendor/bin/typo3 campuslingo:cleanup-translation-cache --dry-run
./vendor/bin/typo3 campuslingo:cleanup-translation-cache --dry-run --respect-hidden-restriction --respect-time-restrictions
./vendor/bin/typo3 campuslingo:cleanup-translation-cache
```

Running this command regularly (e.g. as a scheduler task) keeps the cache table free of stale entries that would otherwise accumulate as referenced records are deleted or expire.

Use `./vendor/bin/typo3 campuslingo:cleanup-translation-cache --help` for more information.

Translate locallang.xlf files

Prerequisite: The applicationContext of the TYPO3 system must be set to development.

1. Open the Extension Configuration of EXT:campus_lingo and switch to tab "Development".
2. Enter the extension keys [extKey], for which you want automatically translate the locallang.xlf file.
3. If the target locallang.xlf file exists (e.g. de.locallang.xlf for the German translation file) and you want to overwrite it, check the corresponding checkbox.
4. Switch to the backend module "Maintenance" and click "Manage languages…".
5. You may need to add the target language if it isn't already there.

6. Download the language of the extension(s) configured in step 2. Or simply click “Update all”.
7. The translations for the extension(s) configured in step 2 are automatically translated from on `EXT:[extKey]/Resources/Private/Language/locallang.xlf`,
   and are saved to `EXT:[extKey]/Resources/Private/Language/[target_language].locallang.xlf`

Your extension is now ready for use in the target language.

Translation of arbitrary content

The API Method `\BrainAppeal\CampusLingo\Service\LanguageOverlayService::autoTranslateRow` allows automatic translation of literally “arbitrary” content given as an associative array (e.g. some database row from an external database).

Features

- Respects the requested language on frontend request.
- Uses site’s default language as source language by default. The source language may be given as parameter, allowing for automatic translation of content given in languages different from the site’s default language.
  Note: This allows for automatic translation to the site’s default language: If the passed source language is different from the site’s default language, the API method will even trigger automatic translation, if the default language is requested.
- Respects (de)activation of the auto-translation via page properties.
- The placeholder `<!-- CAMPUS_LINGO_AUTOTRANSLATION_INFORMATION -->` reflects automatic translations via this API method.
- Multiple automatic translation of the same content is avoided by using the translation cache of EXT:campus_lingo. Changes of the same content source trigger automatic retranslation, whereas the content source is identified by two parameters (id and identifier).
- On errors, the configured language fallbacks of the site’s language are respected.

Configuration

Apart from the method’s parameters, you may need (or want) to configure the automatic translation by establishing an event listener for the event `\BrainAppeal\CampusLingo\Event\TranslationServiceModifyTCAEvent` (see above)

Example for using LanguageOverlayService::autoTranslateRow

```
/**
 * @param array $row
 * @param string $identifier
 * @param string $idField
 * @return array
 */
function autoTranslateRow(array $row, string $identifier, string $idField = 'id'): array
{
    /**
     * for all parameters:
     * @see \BrainAppeal\CampusLingo\Service\LanguageOverlayService::autoTranslateRow
     *
     * you may need (or want) to configure, which keys of $row are automatically
        translated
     * by establishing an event listener for
     * @see \BrainAppeal\CampusLingo\Event\TranslationServiceModifyTCAEvent
     */
     return \BrainAppeal\CampusLingo\Service\LanguageOverlayService::getInstance()-
        >autoTranslateRow(
        $row,
        $identifier,
        $idField
    );
}

$translatedRow = autoTranslateRow(
    [
        'id' => 12345,
        'textProperty' => 'Please translate this text automatically.'
    ],
    'someIdentifierToEnsureThatIdIsUnique',
    'id'
);
```

Custom route enhancers with enabled slug translation

If automatic slug translation is enabled (`feature_enableSlugTranslation = 1`), the PersistedAliasMapper for Extbase routes is extended to support custom route enhancers.

If you use a custom route enhancer this also must be extended.

To do this, you need to implement a class in your sitepackage that extends the custom route enhancer:

```
<?php

namespace Vendor\YourExtension\Routing\Aspect;

use BrainAppeal\CampusLingo\Routing\Aspect\AutoTranslatedRouteTrait;
use GeorgRinger\News\Routing\NewsTitleMapper;

class AutoTranslatedNewsTitleMapper extends NewsTitleMapper
{
    use AutoTranslatedRouteTrait;
}
```

Then add the following to your ext_localconf.php to override the original route enhancer:

```
if (\BrainAppeal\CampusLingo\Service\ConfigurationService::enableSlugTranslations()) {
    $GLOBALS['TYPO3_CONF_VARS']['SYS']['routing']['aspects']['MyCustomAliasMapper'] =
        \Vendor\YourExtension\Routing\Aspect\AutoTranslatedNewsTitleMapper::class;
}
```

The news extension provides alias mapper classes for news, categories and tags. These are already extended by campus_lingo in `Classes/Routing/Aspect/News`.