



Best Practices for Building Multilingual Sites in Drupal 8

BADCamp 2020

Mohit Aghera (mohit_aghera)

About Me

Mohit Aghera

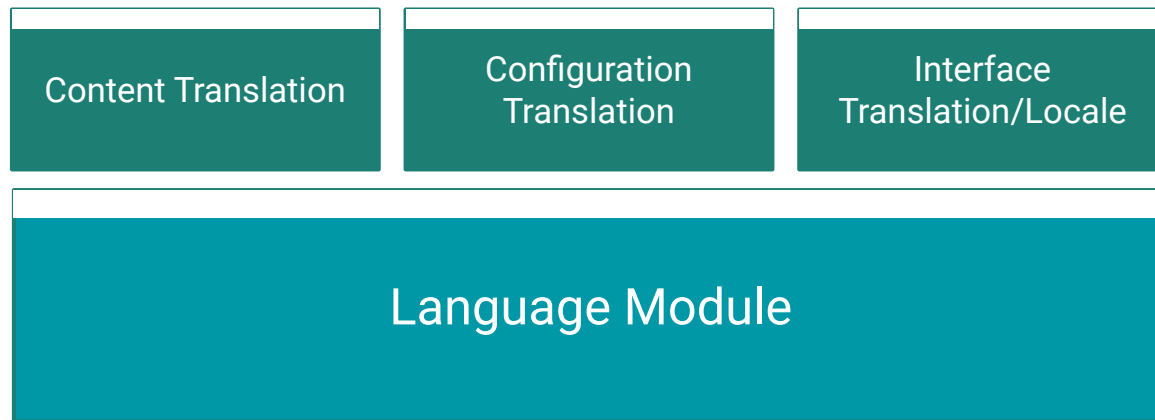
Drupal Technical Architect @Axelerant

Drupal.org: mohit_aghera

Twitter: @mohit_rocks



Modules Stack



Two Personas

Site Builders

Developers



Site Builders

Translating Content Entities

- Content Types
- Detailed options are visible on Regional and language > Content Language page.

Edit *Advanced Recipe* content type ☆

[Edit](#)[Manage fields](#)[Manage form display](#)[Manage display](#)[Translate content type](#)

[Home](#) » [Administration](#) » [Structure](#) » [Content types](#)

Name *

Machine name: advanced_recipe

The human-readable name of this content type. This text will be displayed as part of the list on the *Add content* page.

Description

Advanced Recipe content type for demonstration purposes.
It contains various paragraphs like an accordion, WYSIWYG, and other necessary fields.

This text will be displayed on the *Add new content* page.

Submission form settings

Title

Publishing options

Published, Create new revision

Language settings

Site's default language (English),
Show language selector on create
and edit pages, Enable
translation

Display settings

Don't display post information

Menu settings

Default language

Explanation of the language options is found on the [languages list page](#).

☒ Show language selector on create and edit pages

☒ Enable translation

[Save content type](#)[Delete](#)

Translating Content Entities

- Blocks translation
- Blocks visibility per language
- Translation of various attributes for block instance like title etc.

Edit *Banner block* custom block type ☆

[Edit](#)[Manage fields](#)[Manage form display](#)[Manage display](#)[Tra](#)

[Home](#) » [Administration](#) » [Structure](#) » [Block layout](#) » [Custom block library](#)

Label *

Machine name:

Provide a label for this block type to help identify it in the administration pages.

Description

A banner block contains a title, summary, link to content and a background image. The back

Enter a description for this block type.

☐ Create new revision

Create a new revision by default for this block type.

▼ LANGUAGE SETTINGS

Default language

Explanation of the language options is found on the [languages list page](#).

☒ Show language selector on create and edit pages

☒ Enable translation

[Save](#)[Delete](#)

Making Fields Translatable

- Enable from the field settings
- Detailed settings related to content types can be found on “admin/config/regional/content-language”
-

Body settings for *Basic page* ☆

[Edit](#)[Field settings](#)[Translate content fields](#)

[Home](#) » [Administration](#) » [Structure](#) » [Content types](#) » [Basic page](#) » [Manage fields](#)

Label *

Help text

Instructions to present to the user below this field on the editing form.

Allowed HTML tags: `<a>` `<b>` `<big>` `<code>` `<del>` `<em>` `<i>` `<ins>` `<pre>` `<q>` `<small>`

This field supports tokens.

☐ Required field

☒ Users may translate this field

☒ Summary input

This allows authors to input an explicit summary, to be displayed instead of the automa

☐ Require summary

The summary will also be visible when marked as required.

Label *

Additional Information

Help text

Instructions to present to the user below this field on the editing form.

Allowed HTML tags: <a> <big> <code> <i> <ins> <pre> <q> <sn>

This field supports tokens.

☐ Required field

☐ Users may translate this field

⚠ Paragraphs fields do not support translation. See the [online documentation](#).

▼ REFERENCE TYPE

Reference method *

Default

Which Paragraph types should be allowed?

☐ Exclude the selected below

☒ Include the selected below

Paragraph types

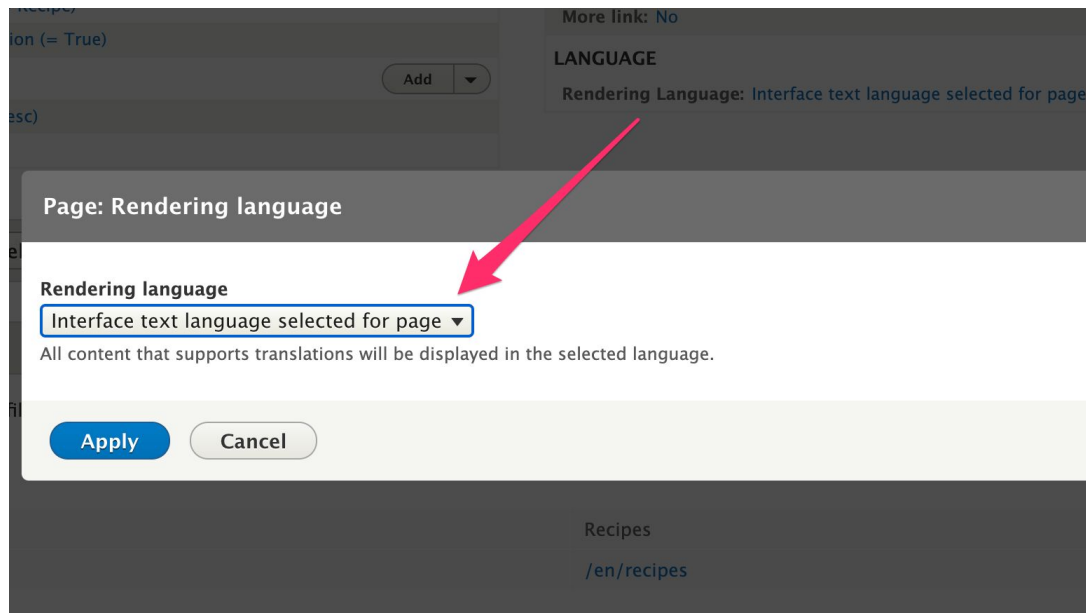
Paragraph Translation

Region & Language > Content Language

☐	Cooking time
☐	Difficulty
☒	Ingredients
☒	Media Image
☐	Number of servings
☐	Preparation time
☐	Additional Information (* unsupported)

Views - Content Translation

- Display plugin provides various options to render the content.
- Also takes care of simple translation fallback



Views - Content Translation

- We can use “select_translation” for advance translation fallback.
- Similar modules
 - active_translation
 - language_fallback
 - entity_language_fallback

FILTER CRITERIA Add

- Content: Published (= Yes)
- Content: Content type (= Recipe)
- Content: Select translation (Default site language fallback)

SORT CRITERIA Add

- Content: Authored on (desc)
- Content: ID (asc)

NO RESULTS BEHAVIOR

PAGER

Use pager: Mini | Mini pager, 12 it

More link: No

LANGUAGE

Rendering Language: Interface text

Configure filter criterion: Content: Select translation

Select which translation of a node should be displayed

☐ Expose this filter to visitors, to allow them to change it

Save **Cancel**

Preview

Preview with contextual fil

Title	Path
Recipes	Laddu - En

SELECT TRANSLATION SELECTION MODE

- ☐ Use the current interface language; if not available use the original node language
- ☐ Use the current interface language; if not available use the default site language; if not
- ☒ Custom language priorities

Language priorities

fr,en,current,default,original

If the selection mode is set to "Custom language priorities", a comma separated list of language. The filter will then return the node in the first available language in that list; falling back to the

Some special values are recognized:

- "current" will be replaced with the current interface language;
- "default" will be replaced with the default site language;
- "original" will be replaced with the original node language.

Example:
en,fr,current,default,original
This will return:

Views - Interface Translation

- Translate all the titles, help text, default text, no result behaviour text, button labels for the views.
- We can export it via config export
- Drupal adds it to separate directory based on language code.

Administrative description	Administrative description
Recipes listing	Listado de recetas

▼ DISPLAYS

▼ MASTER DISPLAY SETTINGS

Title

Master

Title

Máster

▼ RECIPES DEFAULT DISPLAY OPTIONS

▼ EXPOSED FORM

▼ RESET OPTIONS

Submit button text

Apply

Submit button text

Aplicar

Reset button label

Reset

Reset button label

Restablecer

Exposed sorts label

Sort by

Exposed sorts label

Ordenar por

Ascending

Asc

Ascending

Asc

Translated Views

- Gets exported to
“<your-config-directory>/language/<langcode>/”
- In demo setup: config/sync/language/es/

```
label: Recetas
description: 'Listado de recetas'
display:
  default:
    display_title: Máster
    display_options:
      exposed_form:
        options:
          submit_button: Aplicar
          reset_button_label: Restablecer
          exposed_sorts_label: 'Ordenar por'
          sort_asc_label: Asc
          sort_desc_label: Desc
    pager:
      options:
        tags:
          previous: <<
          next: >>
        expose:
          items_per_page_label: 'Elementos por página'
          items_per_page_options_all_label: '- Todo -'
          offset_label: Desplazamiento
    fields:
      title:
        separator: ','
      title: Recetas
page_1:
  display_title: Página
  display_options:
    menu:
      title: Recetas
```

Multilingual Solr Search

- Multilingual solr search is part of “search_api” module now.
- URL on demo setup: “admin/config/search/search-api/index/umami_search/edit”

Translations Contribution Tools

- TMGMT module and ecosystem
- It supports automated or user contributed translations

Developers

Entity Translation API

- Fields inherit language from parent language aware entity
- `$entity->getTranslation()`
- `$entityRepository->getTranslationFromContext()`
- Several other methods to `EntityInterface`
- `hook_entity_translation_create()`
- `hook_entity_translation_insert()`
- `hook_entity_translation_delete()`

Entity Translation API

```
* @ContentEntityType(  
*   id = "recipe_review",  
*   label = @Translation("Recipe review"),  
*   base_table = "recipe_review",  
*   data_table = "recipe_review_field_data",  
*   revision_table = "recipe_review_revision",  
*   revision_data_table = "recipe_review_field_revision",  
*   translatable = TRUE, ←  
*   entity_keys = {  
*     "id" = "id",  
*     "revision" = "vid",  
*     "label" = "name",  
*     "uuid" = "uuid",  
*     "uid" = "user_id",  
*     "langcode" = "langcode", ←  
*     "published" = "status",  
*   },
```

Making Entity Fields Translatable

```
$fields['name'] = BaseFieldDefinition::create( type: 'string')
    →setLabel(t( string: 'Name'))
    →setDescription(t( string: 'The name of the Recipe review entity.'))
    →setRevisionable( revisionable: TRUE)
    →setTranslatable( translatable: TRUE)
    →setSettings([
        'max_length' ⇒ 50,
        'text_processing' ⇒ 0,
    ])
    →setDefaultValue( value: '')
    →setDisplayOptions( display_context: 'view', [
        'label' ⇒ 'above',
        'type' ⇒ 'string',
        'weight' ⇒ -4,
    ])
    →setDisplayOptions( display_context: 'form', [
        'type' ⇒ 'string_textfield',
        'weight' ⇒ -4,
    ])
    →setDisplayConfigurable( display_context: 'form', configurable: TRUE)
    →setDisplayConfigurable( display_context: 'view', configurable: TRUE)
    →setRequired( required: TRUE);
```

Configuration Entity Translation

- Configuration translations are provided by “config_translation” module.
- It works out-of-box for all the Drupal Core’s config entities

Translating Custom Configuration

- Create “my_module.schema.yml”
- Add the fields that we want to make translatable.
- Create “my_module.config_translation.yml” to link it to schema.yml

Translating Custom Configuration

```
umami_multilingual.adminconfig:  
  type: config_object  
  label: 'Umami Multilingual basic config'  
  mapping:  
    help_text:  
      type: text  
      label: 'Help Text'  
    default_description:  
      type: text  
      label: 'Default Description'
```

umami_multilingual_schema.yml

```
3 umami_multilingual.admin_config_form:  
4   title: 'Umami Configuration Form'  
5   # base_route_name is the route that provides configuration  
   form.  
6   base_route_name: umami_multilingual.admin_config_form  
7   # names should contain all the configuration keys in  
   "umami_multilingual.schema.yml"  
8   names:  
9     - umami_multilingual.adminconfig
```

umami_multilingual.config_translation.yml

Configuration Translation APIs

- Override the configuration
- Fetch translated configuration
- hook_config_translation_info_alter()

```
$language = $language_manager->getLanguage($params['account']->getPreferredLangcode());  
$original_language = $language_manager->getConfigOverrideLanguage();  
$language_manager->setConfigOverrideLanguage($language);  
$mail_config = \Drupal::config( name: 'user.mail');  
  
$token_options = [ 'langcode' => $langcode, 'callback' => 'user_mail_tokens', 'clear' => TRUE];  
$message['subject'] = PlainTextOutput::renderFromHtml($token_service->replace($mail_config->get  
($key . '.subject'), $variables, $token_options));  
$message['body'][] = $token_service->replace($mail_config->get($key . '.body'), $variables,  
$token_options);  
  
$language_manager->setConfigOverrideLanguage($original_language);
```

Language Negotiation Plugins

- Determines which language to pick
- We can also write custom language negotiation plugins

Interface text language detection

Order of language detection methods for interface text. If a translation of interface text is available in the detected language, it will be displayed.

DETECTION METHOD	DESCRIPTION
✚ Custom Language Negotiation	Language from the custom query parameters.
✚ Account administration pages	Account administration pages language setting.
✚ URL	Language from the URL (Path prefix or domain).
✚ Session	Language from a request/session parameter.
✚ User	Follow the user's language preference.
✚ Browser	Language from the browser's language settings.
✚ Selected language	Language based on a selected language.

Content language detection

Order of language detection methods for content. If a version of content is available in the detected language, it will be displayed.

☐ Customize *Content* language detection to differ from Interface text language detection settings

Save settings

Language Negotiation Plugins

```
/**
 * Custom Language Negotiation Plugins.
 *
 * @LanguageNegotiation(
 *   id = \Drupal\umami_multilingual\Plugin\LanguageNegotiation\LanguageNegotiationCustom::METHOD_ID
 *   weight = -1,
 *   name = @Translation("Custom Language Negotiation"),
 *   description = @Translation("Language from the custom query parameters."),
 * )
 */
class LanguageNegotiationCustom extends LanguageNegotiationMethodBase {

  /**
   * The language negotiation method id.
   */
  const METHOD_ID = 'language-custom';

  /**
   * @inheritdoc
   */
  public function getLangcode(Request $request = NULL) {
    $langcode = NULL;

    if ($this->languageManager && $request && $request->query->get( key: 'test-language') !== NULL) {
      $query_langcode = $request->query->get( key: 'test-language');
      $langcodes = array_keys($this->languageManager->getLanguages());
      if (in_array($query_langcode, $langcodes)) {
        $langcode = $query_langcode;
      }
    }

    return $langcode;
  }
}
```

Interface Text Translation

- Use “StringTranslationTrait”
- Translate by `$this->t(“Hello World”)`
- Context parameters to `t()` helps to provide contextual and disambiguous translations

Interface Text Translation

```
public function build() {  
    $build = [];  
    $text_1 = $this->t( string: 'Hello World', [], ['context' => 'String-1']);  
    $text_2 = $this->t( string: 'Hello World', [], ['context' => 'String-2']);  
  
    return [  
        '#theme' => 'item_list',  
        '#items' => [$text_1, $text_2],  
    ];  
}
```



SOURCE STRING	TRANSLATION FOR SPANISH
Hello World In Context: String-1	Hello World - ES1
Hello World In Context: String-2	Hello World - ES2

Save translations

Translation in Twig

- t() twig function for the translation
- trans filter

```
{% if site_logo %}
<a href="{{ path('<front>') }}" rel="home">
  
</a>
{% endif %}
```

Translation in JS

- Drupal.t()
- Drupal.formatPlural()

```
if ($toolbarEscape.length && pathInfo.currentPathIsAdmin) {
  if (escapeAdminPath !== null) {
    $toolbarEscape.attr('name', 'href', escapeAdminPath);
  } else {
    $toolbarEscape.text(Drupal.t(str: 'Home'));
  }
}
```

Adding Translation in Custom Modules

- “interface translation project” and “interface translation server pattern” properties in the info.yml files helps to provide custom .po files.

umami_multilingual/umami_multilingual.info.yml

```
interface translation project: umami_multilingual
interface translation server pattern:
  modules/custom/umami_multilingual/translations/%language.po
```

```
# Umami Multilingual translations for Spanish language.
msgid "Hello World"
msgstr "Hola Mundo"
```



umami_multilingual/translations/es.po

Translations Deployment

- Drush Language Commands: https://www.drupal.org/project/drush_language
- We can leverage Drush commands with CI pipeline.

Sample Repository

- Github: <https://github.com/mohit-rocks/drupal8-multilingual>

Coming up next Friday 11 am

- **Accessible SVGs: Inclusiveness Beyond Patterns** with Carie Fisher
- **Decoupling Drupal: Gatsby Live Preview from the same project** with Chad Carlson
- **Making a better community, better software, and a better world** with Tara King, Ruby Sinreich, and Elli Lugwigson



Coming up next
Friday 10:45 am



amaze.io

- **Coffee Break** with amaze.io in the Expo Hall



Thank you!!