Building Healthy Multilingual Relationships

Correctly Nest Entities

Introductions

Spatial Path

Aimee Hannaford

- Owner & Architect of Spatial Path
- Previous CEO, Co-Founder, & Principal Architect of Hook 42
- Relevant Experience:
  - Enterprise site auditor: ML, A11y, Migration, Standards
  - ML Enterprise websites since 1997
  - Author of Drupal 8 Multilingual training assets
  - Enterprise-scale Drupal for 12 years!
- Certifications:
  - Management: PMP, SCPM, CSM, CSPO
  - Accessibility: CPWA
  - Drupal: D8 Acquia Certified Site Builder

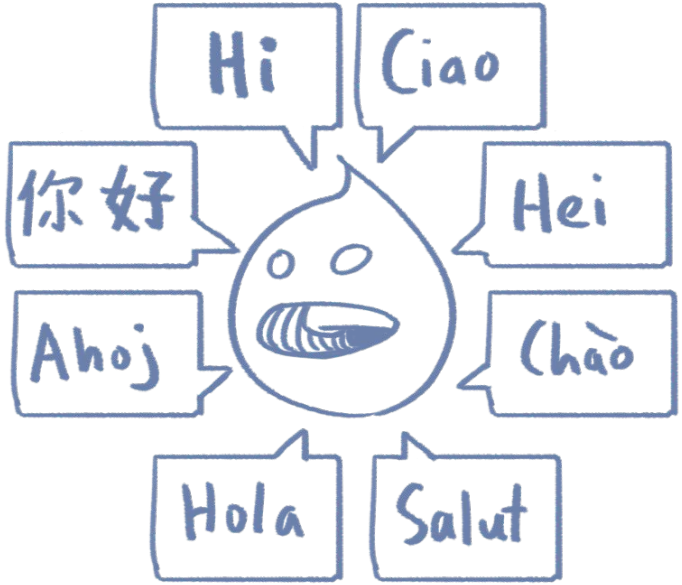
Introductions

Christian López Espínola

- penyaskito on Drupal.org and elsewhere
- Software developer at Lingotek
- Relevant Experience:
  - Drupal 8/9 integration Development for the Lingotek TMS, dealing with translating Drupal on a daily basis.
  - Drupal contributor with a focus on Drupal Multilingual support
  - Drupal Event organizer and speaker.
  - Contributor to the localize.drupal.org team

Thank yous

- DrupalCon Europe 2020 organizers
- Global Drupal Multilingual Team
- Pantheon Systems
- Lingotek
- Hook 42, Inc.



Some image assets and multilingual doodles are provided by Hook 42, Inc.

About the scope

Primarily focused on Drupal 8+.

- We can’t cover everything.
- Multilingual nesting is complex.
- Focus on common use cases and Content Entities.
- We will cover configurations and their expected outcomes.
- Every site is different, but use similar configuration patterns.
- Configuration patterns can be applied to your site.
- Acquia Site Studio and Cohesion is not covered.

This session is recorded and the presentation deck will be shared. :)

Information flow

We are covering a lot topics. Slides will be covered quickly to allow time for questions.

Why are we really here?

Drupal is a powerful platform

There are a lot of moving parts to make a functional website!

- Feature-rich
- Extensible
- Scalable
- Flexible
- Multilingual

Expectations vs. Reality

Prevent unhappiness with clear expectations and provide a predictable, functional website.

Reality

With great flexibility comes complexity.

So many permutations...

- Site-building approaches
- Content editing interfaces
- Display and theme layer
- Content publication
- Customization
- Integrations
- Evolving best-practices

Multilingual support exponentially increases test cases and complexity.

“Where there is great power there is great responsibility.”

- *Winston Churchill, 1906*
- *Uncle Ben Parker, Spiderman*

Getting Started

Understanding Multilingual Content & Site Strategy

Translation vs. Localization

Translation

The process of **translating** words or text from one language to another.

Localization

The process of **adapting** a product, application or document to meet the language, cultural and or other requirements of a specific target market (a locale).

Symmetrical vs. Asymmetrical

Symmetrical

Layout display is the same between the source and locales of the same content entity.

Example:
Node 1 in English is directly translated to other languages.

Each translated locale for Node 1 looks the same.

Asymmetrical

Layout display and translated versions of content differ between locales of the same content entity.

Content translations are not strictly tied to the source language are often decoupled from the source language, even though they are stored in the same content entity.

Translated locales can look different from the source locale.

Define content business rules

Use a realistic, language-first approach through discovery.

Key topics:

- **Resources:** Initial budget & time, ongoing support, people.
- **Markets:** Language, locale, and translation expectations.
- **Purpose:** Content types, products, use cases, frequency.
- **Interface:** Design and UI of the site, all locales.
- **Creation:** Content editing tools and workflows.
- **Storage:** Per-field content needs.

Define display expectations

Key questions:

- Is the whole page **structure** the same across languages?
- Does **any region** of the page's structure change per locale?
- Does the page structure change **drastically** across locales?
- Does **any part of** the **content** need to change per locale?
  - Page content is 90% the same as source language.
  - 10% of page has locale specific content

Define content creation system (people & tools)

Key questions:

- Who updates content? Who translates content?
- What control does an editor have? Copy? Layout?
- What responsibilities do the translators have?
- Are some people combination editors and translators?
- What is required for content approval?
- How and when does **content** change per locale?
- What is the proposed Drupal layout approach?

Content storage and field planning

Questions to ask for each **field** of a content entity:

- Does the field content stay the same for each language?
- Does the field content value(s) stay the same, but need to display in a translated language?
- Does the field content contain non-translated or language specific information?
- How will the field content be displayed?

Overcoming challenges in the discovery phase

Education & full team involvement

- Define what translation and localization mean to the project.
- Define lifecycle of **every type of content**, including translation.
- Teach everyone about multilingual basics (based on role).
- Define expected content differences across locales.
- Define content types, workflow, and translation needs.
- Define media content needs very early.
- **Document everything!**

“If you fail to plan, then you are planning to fail!”

- *Benjamin Franklin*

Multilingual & Nesting Fundamentals

Expectations

Drupal's multilingual system will work all the time because it is in core.

Drupal 8+ was built with a language first approach.

- Content entities are language aware.
- Content entities have fallback rules.
- Entity management APIs have multilingual support.
- Modules can build off the consistent ML system.

Reality

Functional multilingual configuration is not just content.

- Language detection
- Language fallback
- Display and theme layer logic
- Extensions from contrib
- Site customizations
- Interaction with content editing interfaces

Multilingual Pillars in Drupal 8+

Language

Base services for all modules dealing with data.

Not just multilingual.

Interface

Interface translation has a built-in update feature and improves usability.

Content

Field translation in a built-in API for all entities.

Content translation provides a user interface.

Config

Common configuration system handles blocks, views, field settings.

Unified translation.

Multilingual Pillars in use

Language

Detection
Available languages
Custom languages
URLs

Module:
Language

Interface

UI Text
Theme text
Layout

Module:
Interface
Translation

Content

Nodes
Users
Images
Comments
Taxonomy Terms
Blocks (content)

Module:
Content
Translation

Config

URLs
Contact form
Roles
Blocks (structure)
Vocabularies
Views

Module:
Configuration
Translation

Content entities (our focus)

- **Many:** Node, Taxonomy Term, Media, Custom Blocks, Paragraphs, Custom Entities, and more...
- Content entities can be **translated** or **not translated**.
- Content entities have **language display fallback** rules.
- Content entities **have fields** that can be translated, if desired.
- A content entity must be configured for translation if any fields within the content entity need to be translated.

Field-level multilingual configurations

Each field can be configured for translation **ONLY IF** the content bundle is configured for translation.

A field can be one of two types:

- Field contains content.
- Field points to another content entity (Entity Relationship) or even a concrete revision (Entity Reference Revision)!

Content fields vs. Reference fields

Content field types:

- Text:
  - Text
  - Textarea
  - List (select, checkbox, radio)
  - Check-boxes/radio buttons
- Number
- Boolean
- Link

Entity reference types:

- Image/File
- Taxonomy
- Media
- Node
- Blocks
  - System
  - Layout builder
- User
- Custom entities (yours)
- Custom entities (contrib)
  - Commerce products
  - Paragraphs

Field-level multilingual configurations

When a field is **translated**, the field's value **can change** across locales.

If a field is **not translated**, the value **stays the same** across locales.

Rules apply to **every** field and to **every** nesting depth.

Field-level considerations

- Don’t just mark every field translatable on your site.
  - It can cause confusion to the content team.
  - It can create unexpected site bugs during translation.
- If a text field or list is used to define layout, don’t translate it.
- Field constraints: character limits and required status.
- Field reuse across content bundles & global configurations.
- Use of content moderation and locale-specific publication status.

Content-field multilingual configurations

Translated Content (text field):

source en: green

translation spanish: verde

Expected behavior:

- Content for Spanish will be a **translation** of the English source.
- A translation of Spanish must exist to display in Spanish.
- If no Spanish translation exists, content is displayed using language fallback rules.

Entity reference field (translation overview)

Non-translated Entity Reference:
Source en value: entity1, entity2.
Parent entity es value: entity1, entity2

Expected behavior:

- Content for Spanish will be **a translation** of the English source.
- Parent entity displays **Spanish versions of entity1 and entity2**.
- Content for Spanish must exist in Spanish to display in Spanish.
- If no Spanish translation exists, the language fallback rules of the referenced entity apply.

Translated Entity Reference:
Source en: entity1, entity2 - Product Variant 1 and 2
Translation es: entity3, entity4 - Product Variant 3 and 4

Expected behavior:

- Content for Spanish will be **different** than the English source.
- Content for Spanish must exist in Spanish to display in Spanish.
- If no Spanish translation exists, the language fallback rules of the referenced entity apply.

Widgets

Field widgets are used to render the field inside forms. They are a plugin.

Why is this important? Some widgets alter the data model on form save and other form interactions.

BUG: Widgets that alter nested entities can lose track of language.

- Inline Entity Form (Simple, Complex)
- Client-side hierarchical select
- Entity browser
- Asymmetric paragraphs

Nesting

- To include one or more Content Entities within another content entity, either by an Entity Reference field or extended entity manager (Entity Browser).
- All multilingual content bundle and field level configuration expectations apply for each nested entity bundle.

Plan.

Test.

Don’t Assume.

Revisit.

Translate.

Multilingual Site Debugging

Common Issues with Multilingual Sites

Dealing with common issues - configs

Why is the wrong language showing?

- Are translations configured correctly?
- Check configs again! Settings must be correct on every nested entity bundle and field.
- Check for field reuse and misaligned configurations.
- Do translations exist? Are they published?
- Are the permissions correct?

Dealing with common issues - bugs

Why is the wrong language showing?

- Many modules and widgets manipulate content during editing and can create unexpected behaviors and/or bugs.
- Nested entities do not get the correct language passed from parents in content edit mode.
- Symptoms:
  - Nested entity language does not match intended translation language.
  - Nested entities display in source language.
  - Nested entities display in direct-parent language.
  - Nested entities are created in the default language.

Common issues of core + config + customization

People use content choices + display logic to drive multilingual display beyond what is provided in core. Each language-specific display customization can cause unexpected results and/or bugs.

- Display logic within views.
- Display logic at theme layer (Twig, preprocessors).
- Display logic driven by content fields (locale choosers).
- With many site-building permutations, edge cases exist.

Common issues: decreased site performance

- Complexity impacts page performance (in edit and display).
- High number of loaded entities per rendered page.
- Multiple revisions stored on each edit (large db size).
- Exponentially larger number of revisions and storage.
- Revisions multiply by the number of supported locales.
- Revisions multiply by revision frequency.

First rule of Entity Nesting:

Don’t Nest.

Second rule of Entity Nesting:

Don’t use a data model for layout.

Third rule of Entity Nesting:

Keep it simple and consistent.

Which entities are the worst nesting offenders?

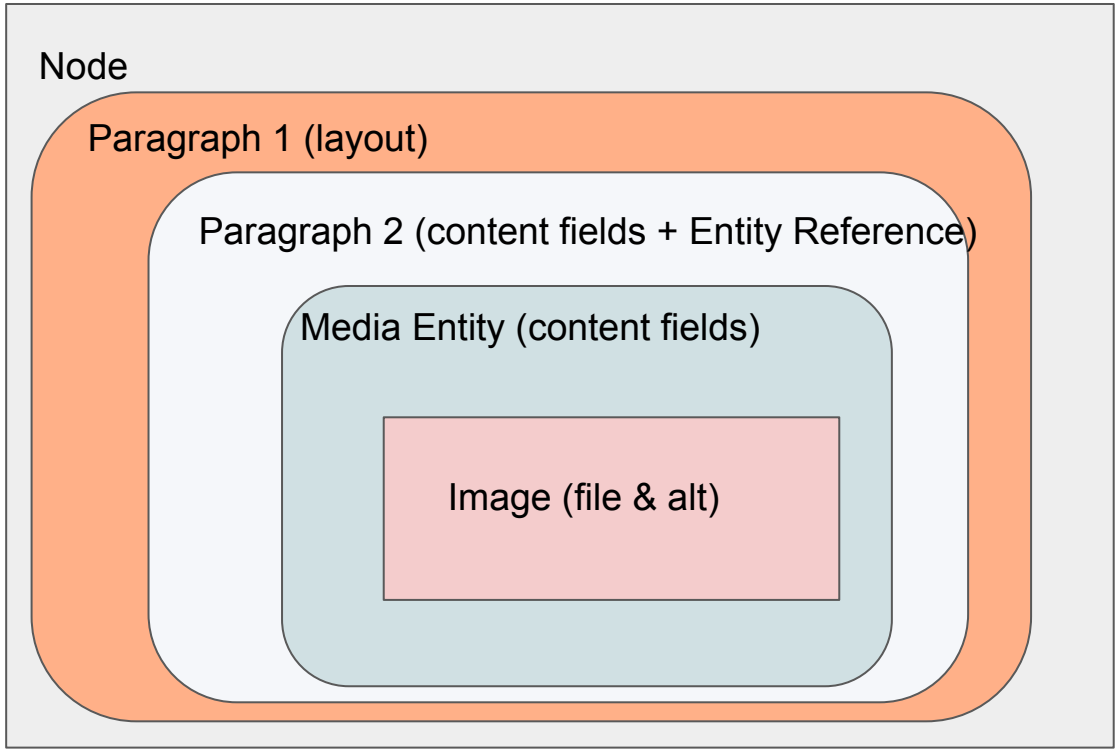
Who?: Paragraphs, Node references, Block references, Custom Content entities.

Why?: There wasn't anything else available for editors to control layouts!

Is there any hope?: Layout management for end users is improving!

But what about my site? Some sites may still use deep nesting approaches. Transitioning approaches may take migration efforts.

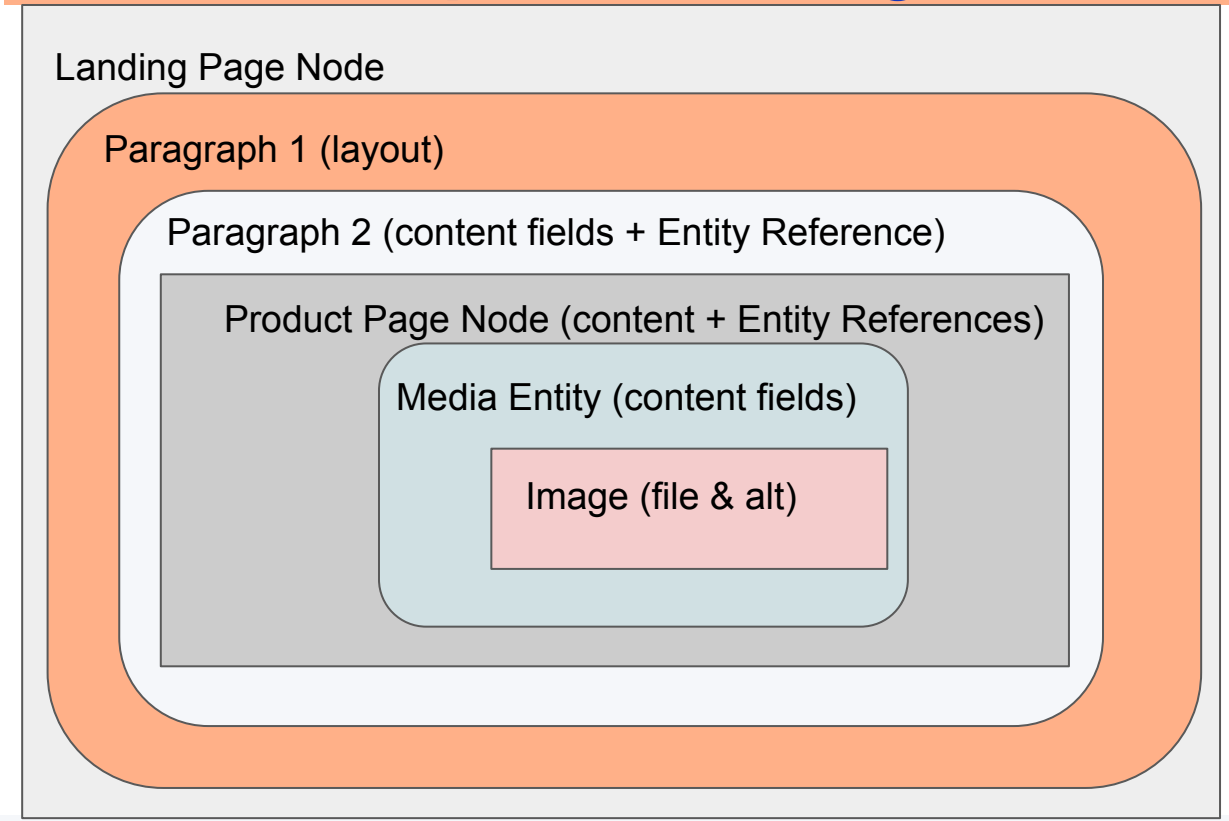
Common deep-level nesting cases



Yes, these nesting levels are really used!

Node > Paragraph 1 (layout) > Paragraph 2 (content fields) > Media Reference > Media Fields > Image

Common deep-level nesting cases



Landing Page > Paragraph 1 (layout) > Paragraph 2 (content fields) > Product page > Media Reference > Media Fields > Image

Using entities inside entities

Working with the many other entities.

- Fields should follow the same multilingual configuration basics.
- Questions to ask yourself for configuring translatability:
  - Is the nested entity reference different between locales?
  - Is the nested entity re-used somewhere else?
  - Is the nested entity used on its own? (e.g. can be navigated to?)
- This also applies to *headless* sites using Drupal as a backend.

Common Nesting Examples

Single Entity Reference

Node with Taxonomy Term Entity Reference

Is the taxonomy field used for Display Rules/CSS Styles?

- Don’t translate the taxonomy terms.
- Only translate the Entity Reference if you want the display to change per locale.
- Expected behavior: terms passing classes will not be translated/changed

Single Entity Reference

Node with Taxonomy Term Entity Reference

Is the taxonomy field used for Content Categorization?

- Create the taxonomy terms and translate them in a separate workflow.
- Entity Reference field is non-translated.
- Expected behavior: term applied to source language stays the same in all languages; translation is displayed.

Single Entity Reference

Node with Taxonomy Term Entity Reference

Is the taxonomy field used for free-form tagging and User Generated Content?

- Consider not translating the terms.
- Requires a richer context-based analysis to define the “best” configuration.

Single Entity Reference

Node with Taxonomy Term Entity Reference

Taxonomy Example - EN

Term ER is Not Translated

Blue
Green
Not Translated 1
Small

Term ER is translated

Yellow
Large
Not Translated 2

Taxonomy Example - ES

Term ER is Not Translated

Azul
Verde
Not Translated 1
Small

Term ER is translated

Amarillo
Small
Not Translated 1

Taxonomy Example - FR

Term ER is Not Translated

Bleu
Vert
Not Translated 1
Small

Term ER is translated

Juane
Large
Small

Repeat the process for each content element

- The taxonomy example quickly surfaces Entity Reference translation configuration in combination with parent content translation and referenced entity translation.
- Each content entity type does have unique questions and content specific considerations to address in your content strategy.
- See the appendix of this deck for Custom Blocks and Media examples.

Common Entity Reference Field Widgets

- Inline Entity Form (IEF)
- Entity browser
- Media browser
- Taxonomy term picker
- Multiple fields

Complex Nesting

Component-based site-building approaches

Symmetrical vs. Asymmetrical

Symmetrical

Layout display is the same between the source and locales of the same content entity.

Example:
Node 1 in English is directly translated to other languages.

Each translated locale for Node 1 looks the same.

Asymmetrical

Layout display and translated versions of content differ between locales of the same content entity.

Content translations are not strictly tied to the source language are often decoupled from the source language, even though they are stored in the same content entity.

Translated locales can look different from the source locale.

Multilingual Paragraphs

Overview

Paragraphs Overview

Req's from core:

- 4 Multilingual modules

Req's from contrib: ERR + Paragraphs

Extensions for ML support:

paragraphs_asymmetric_translation_widgets

Considerations: needs migration if changing translatability

(* unsupported) Paragraphs fields do not support translation. See the online documentation.

Paragraphs asymmetric translation widgets

View | Version control | View history | Automated testing

By efpapado on *21 March 2018*, updated *13 November 2019*

This module is offering asymmetric translations to paragraphs based in the paragraphs Classic widget. Support for the experimental widget is on our radar, but currently not being implemented.

To enable the functionality:

1. Install the module
2. Navigate to the paragraphs field on the entity type where you want to enable it ("Manage fields")
3. Enable "Users may translate this field". Note that the paragraphs module has added a big red warning here, that this won't work. But it will, that's what this module does
4. Navigate the the form mode settings for the paragraphs field on this entity type ("Mange form display") and choose the "Paragraphs Classic Asymmetric" widget.

Single Entity Reference

Node with Paragraphs (Entity Reference Revisions)

- Paragraphs goal is to provide flexible structure of content.
  - It was not conceived as a layout tool.
- Are different translations going to allow different paragraphs?
  - Requires good planning beforehand to define the “best” configuration.
  - Only translate the Entity Reference Revision field if you want the content to change per locale.

Asymmetrical Considerations

Paragraphs

If you do NOT have the extra contrib module, then don’t translate the ER field for paragraphs.

Default behavior assumes all paragraphs are the same and will only show translated field data.

If you use the asymmetric module, the ER field **must** be marked for translation.

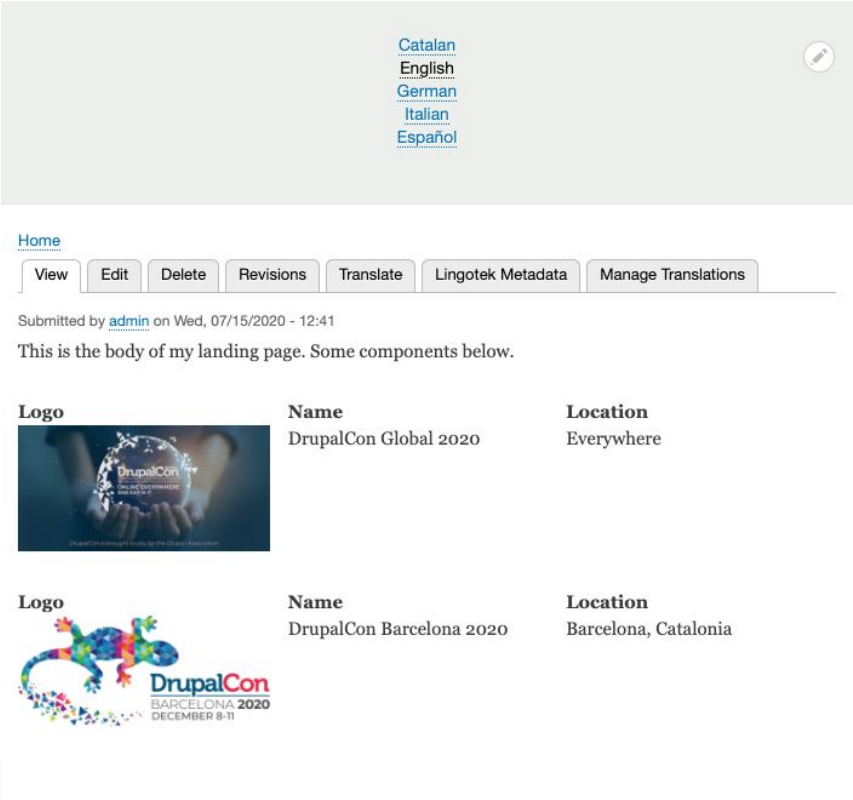
Single Entity Reference

Node with a single-level Paragraph (Entity Reference Revisions) - English source

The way paragraphs were intended to use.

Business expected behavior compared to configuration settings:

If we consider this a flexible component structure: can be customized per language?



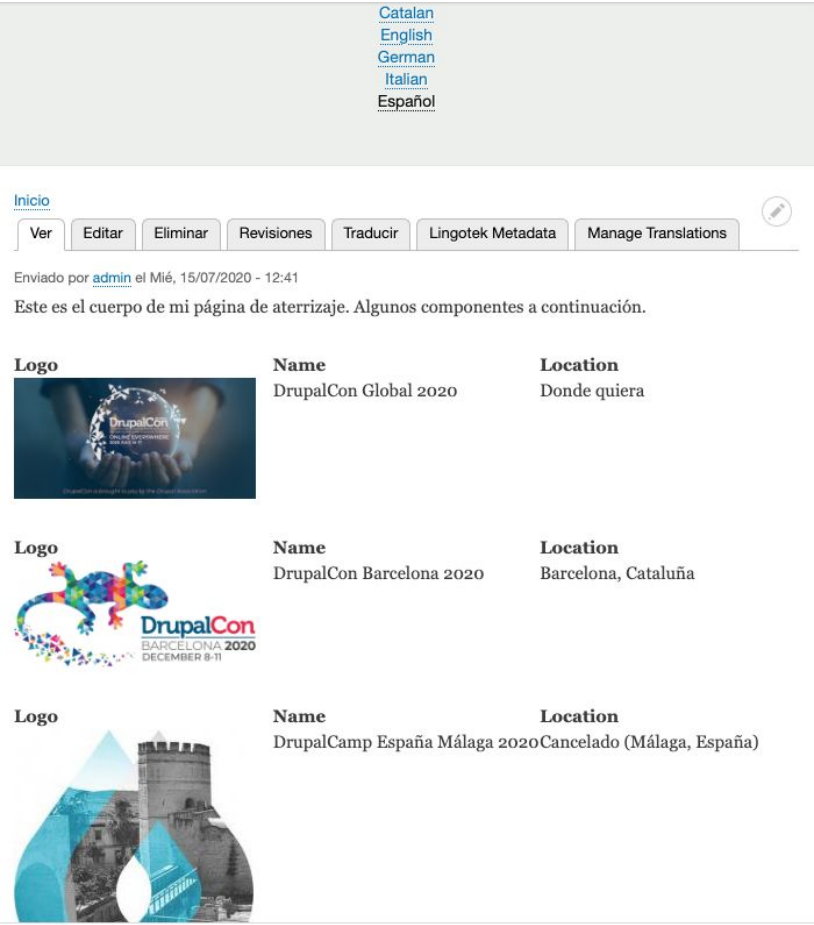
Single Entity Reference

Node with a single-level Paragraph (Entity Reference Revisions) - Spanish translation (asymmetric)

The way paragraphs were intended to use.

Business expected behavior compared to configuration settings:

If we consider this a flexible component structure: can be customized per language?



Multilingual Layout Builder

Overview

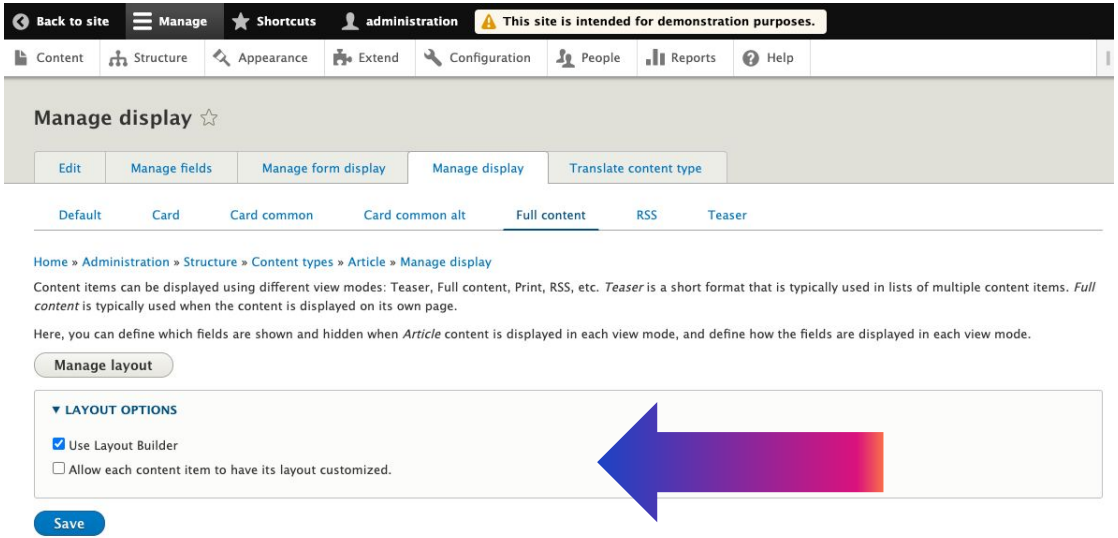
Layout builder basics

- Provided in core.
- Provides a user interface for layout configuration.
- Stores block and layout information in the content entity.
- Custom blocks created within Layout Builder are **non-reusable blocks** that only exist **within the scope of the content entity**.
- Enabling Layout Builder allows for **default display layouts**.
- **Layout Builder Overrides** provide layout edits per content entity.

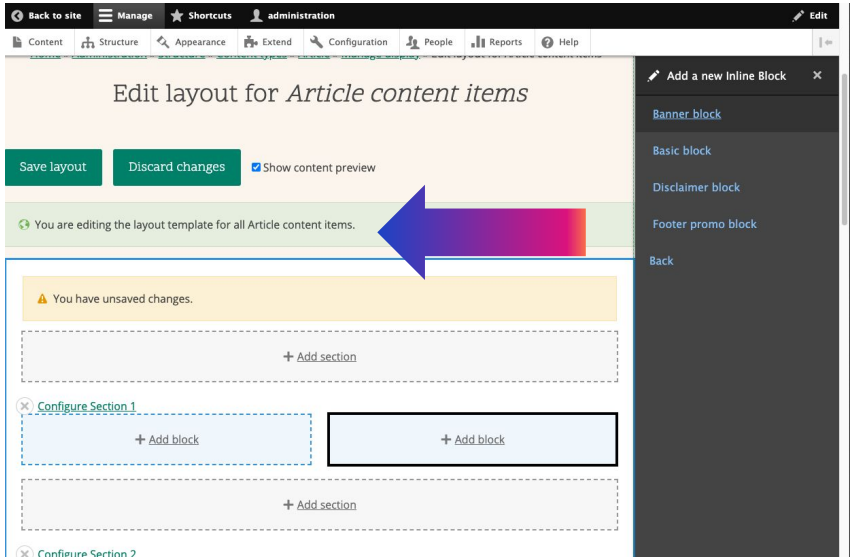
Layout builder for default display structure

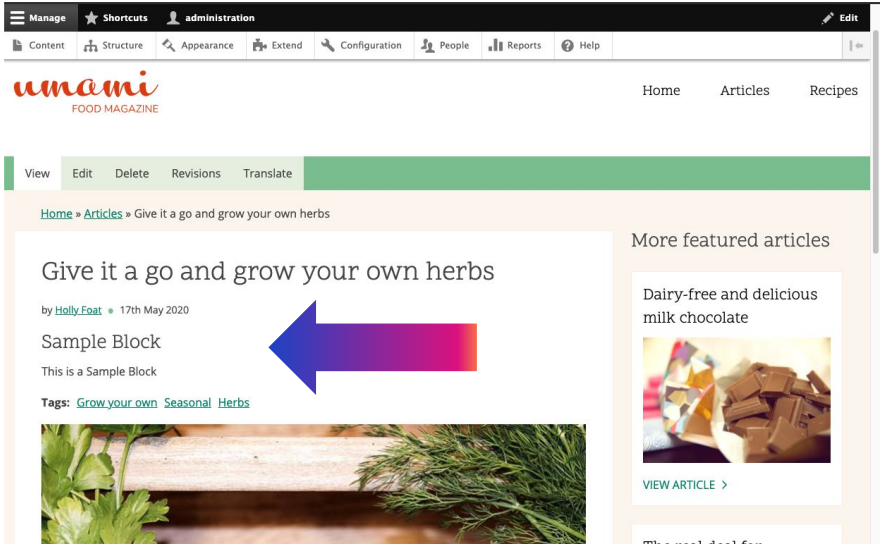
Layout builder is used define default display structures within a View Mode.

Layout builder for default display structure



Layout builder for default display structure





Layout builder for default display structure

- Content entities and fields follow all translation rules.
- If translation exists and field is displayed in Layout Builder, the field displays in the correct language.
- Use of Reusable Custom Blocks will follow the basic blocks pattern.

Layout builder for entity level overrides (core)

Layout Builder provides layout tools and a content creation interface configurable per content entity (e.g., node, block).

Layout builder for entity level overrides (core)

Back to site | Manage | Shortcuts | administration | This site is intended for demonstration purposes.

Content | Structure | Appearance | Extend | Configuration | People | Reports | Help

Manage display

Edit | Manage fields | Manage form display | Manage display | Translate content type

Default | Card | Card common | Card common alt | Full content | Teaser

Home » Administration » Structure » Content types » Recipe » Manage display

There is a security update available for your version of Drupal. To ensure the security of your server, you should update immediately! See the available updates page for more information and to install your missing updates.

Content items can be displayed using different view modes: Teaser, Full content, Print, RSS, etc. *Teaser* is a short format that is typically used in lists of multiple content items. *Full content* is typically used when the content is displayed on its own page.

Here, you can define which fields are shown and hidden when *Recipe* content is displayed in each view mode, and define how the fields are displayed in each view mode.

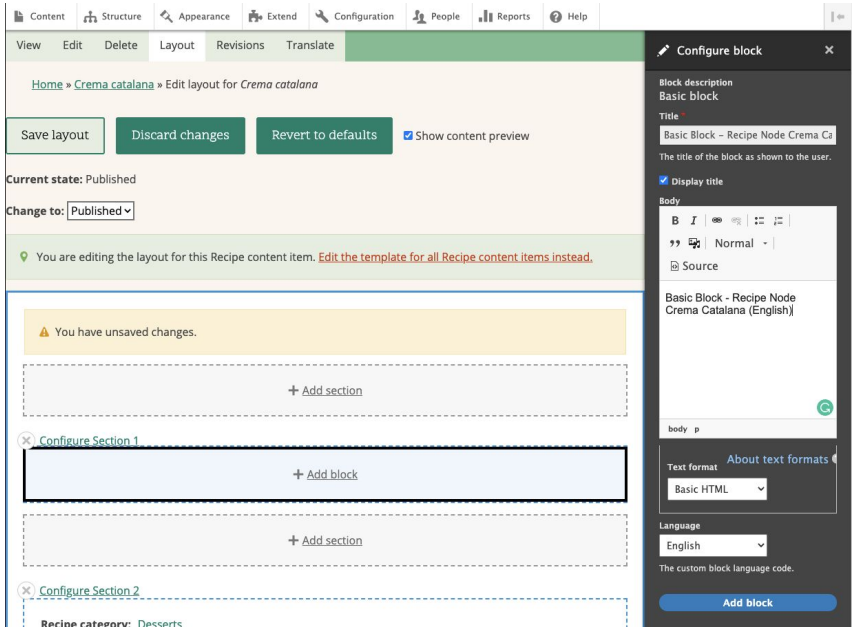
Manage layout

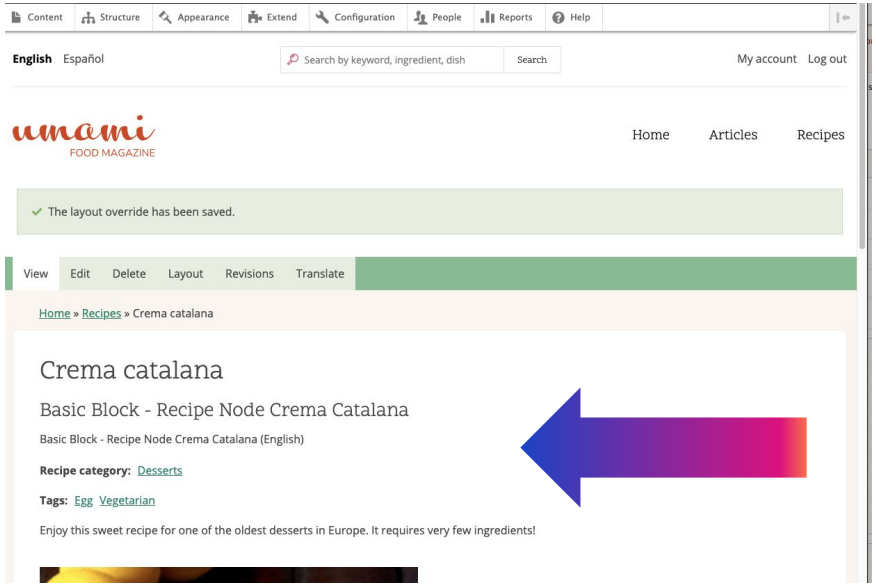
▼ LAYOUT OPTIONS

- [x] Use Layout Builder
- [x] Allow each content item to have its layout customized.

Save

Layout builder for entity level overrides (core)





Layout builder for entity level overrides (core)

- If only fields on the content entity are used, then translated language should display OK.
- All locales of the entity will have the same layout.
- Creation of custom blocks within Layout Builder maintain their source language.
- Using a reusable block will follow the blocks entity reference pattern. If translated, then block should show the correct language.

Layout builder multilingual extensions (contrib)

Layout Builder contrib provides two models for managing layout and content translation for **overrides**. You can only use **ONE** method per **site**. **Choose mindfully.**

- **Asymmetric Translations (layout_builder_at):**
  - **Pros:** Each locale can support localized content.
  - **Cons:** After the first translation from the source language, content and display are completely decoupled per locale.
- **Symmetric Translations (layout_builder_st):**
  - Pros: Source and translated locales are kept in sync
  - Pro/Con: Display is the same across all locales.

Asymmetrical Considerations

Layout Builder

Translated locales can look different from the source locale.

Each translation forks off of the source language and becomes its own layout.

You can choose only one method!

Switching methods after content exists may lead to content loss.

Ongoing multilingual tooling efforts

- Consistent Multilingual configuration patterns for each content entity, use case, and nesting approach.
- Enhanced editor tools, language switchers, workflow management.
- Switching asymmetrical to symmetrical models without a migration.
- Modules to clean up bloated revisions in the database.
- Testing! There many multilingual edge cases with contrib.
- Testing! Don’t only use the Drupal interface.

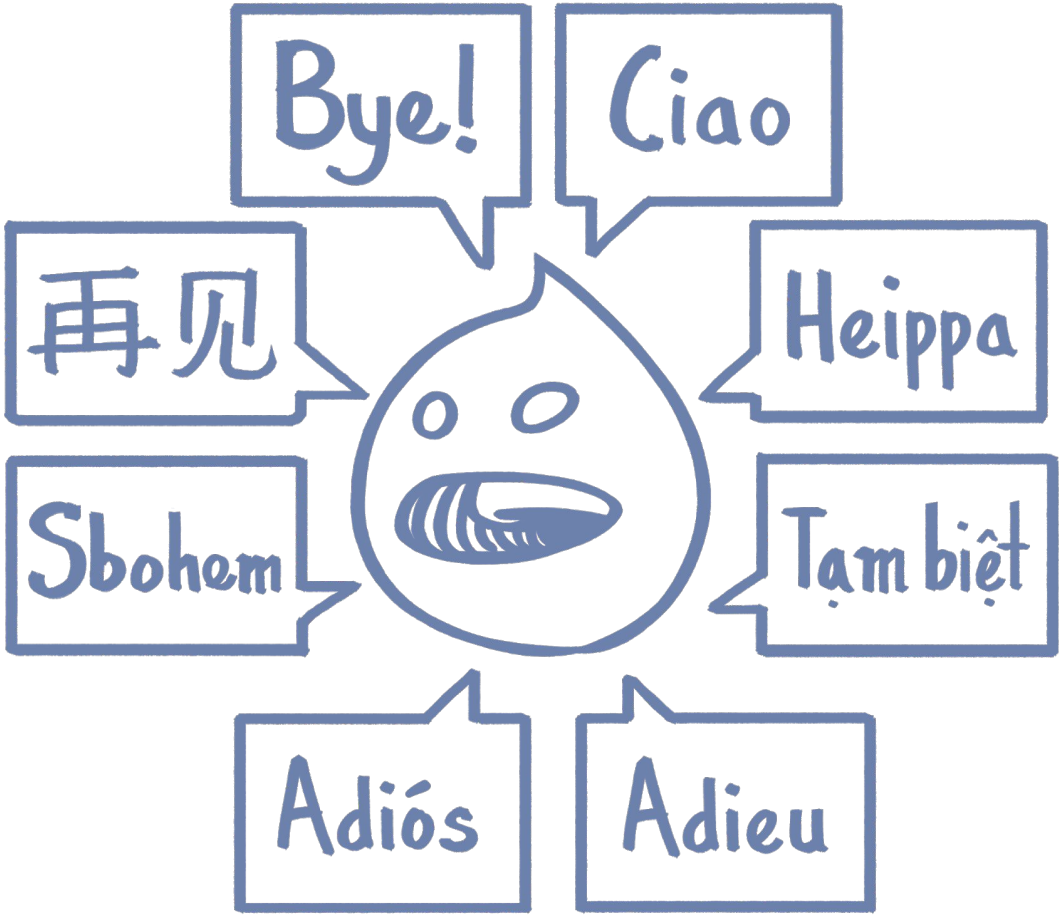
Plan.

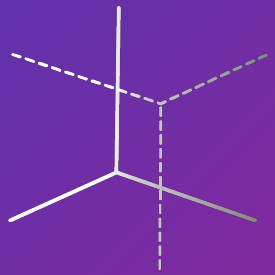
Test.

Don’t Assume.

Revisit.

Translate.





The End.

Thank you!

Resources

- **This presentation:** https://bit.ly/dcg-2020-ml-nesting
- **Entity references gone wild (Jakob Perry):** https://drupal.tv/external-video/2018-08-24/entity-references-gone-wild-how-relationships-can-sink-your-project
- **Your data model is terrible! (Kris Vanderwater):** https://www.midcamp.org/2020/topic-proposal/your-data-model-terrible-let-me-show-you-why

Connect with us!: aimee@spatialpath.com & clopez@lingotek.com

Appendix: additional field info

Overview

Text (list) multilingual configurations

- Available list values are actually configuration translation!
- If you change a list value with existing content, old values exist in the database if used!
- These are key-label pairs. Consider any logic using the key value. The key value will stay in the original source language.

Expected behavior:

If you translate a list field, the VALUE surfaced by the list can change, but the language will not be visible until the TRANSLATION is done via config translation.

Link multilingual configurations

- It is a compound field, like image.
- URL field and a Text field.
- Each can be configured for translation separately.

Expected behavior:

- Follows basic field expectations.
- Best-practice depends on usage of link within content strategy.
- Internal links, external links, etc.
- Compounded by use of modules and widgets like LinkIt.

Single Entity Reference

Node with a reusable Custom Block Entity Reference

Is the block used across multiple entities?

- **Examples:** sitewide info block, advertisement, footer block
- Translate the block through its own separate workflow.

Single Entity Reference

Node with a reusable Custom Block Entity Reference

Is the referenced block the same across translations?

- Do not translate the Custom Block Entity Reference field.
- All language/locale permutations are managed at the Block entity level.
- **Expected behavior:** shared/reusable custom blocks should have their own translation workflow

Single Entity Reference

Node with a reusable Custom Block Entity Reference

Are the Blocks language/locale specific?

- Consider translating the Custom Block Entity Reference field, it will allow different reusable Custom Block entities to be chosen per locale.
- **Sample:** Locale-specific blocks for promotion. Each promotion block is a single block entity with a specific language. The translated page can reference the correct locale block.

Single Entity Reference

Node with Media Entity Reference

Is the Media asset file is the same for all languages?

- A picture without embedded text.
- Do not translate the Media Entity Reference field.

Single Entity Reference

Node with Media Entity Reference

Are the Media assets translated? For example, the image and all associated language variants are translated.

- Do not translate the Media Entity Reference field.
- All language/locale permutations are managed at the Media entity level.
- Expected behavior: Media assets should have a separate translation workflow.

Single Entity Reference

Node with Media Entity Reference

Are the Media assets language/locale specific?

- Consider translating the Media Entity Reference field, it will allow different Media entities to be chosen per locale.
- **Sample:** Language-specific PDF manuals. Each manual is a single Media Entity with a specific language. The translated page can reference the correct manual.
- The **“right way”** would be to translate the actual file and bind all language manuals together.

Common issues with media

- Migration from old site, media assets are a mess.
- Working with a Digital Asset Management Systems
- Accessibility - context specific alt-text
- Multiple additional team members
- Mixture of translated and untranslated media states