



DrupalCon

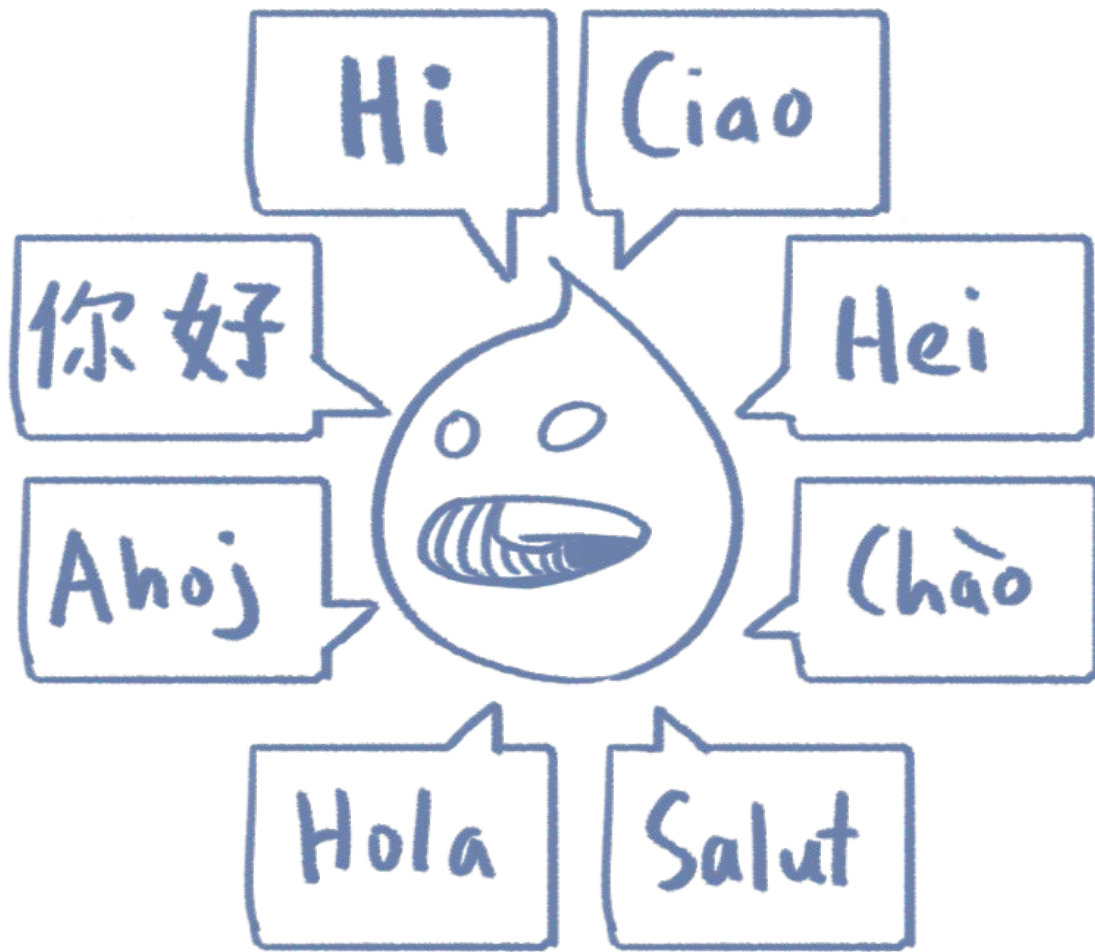
GLOBAL 2020

JULY 14-17

Building Healthy Multilingual Relationships: Correctly Nest Entities

Aimee Hannaford
aimeerae

Christian López Espínola
penyaskito





Aimee Hannaford

CPWA, PMP, SCPM, CSM, CSPO

aimeerae

aimee@aimeerae.com



Christian López Espínola

Lingotek, D8/D9 integration development

penyaskito

clopez@lingotek.com

Thanks for all the fish!

Thank you to the Drupal community and its evolving initiatives to support a more inclusive web.

Thank you to Hook 42 & Lingotek for eight years of multilingual collaboration.

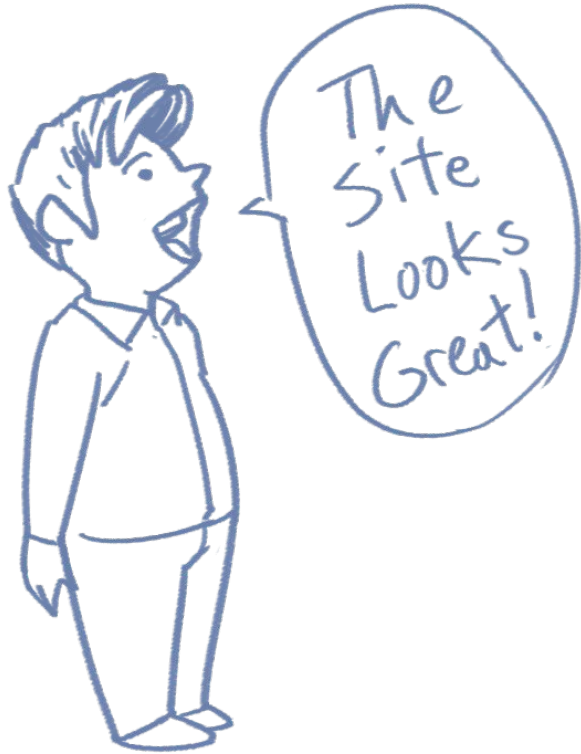


About this session

- Primarily focused on Drupal 8+.
- We can't cover everything.
- Multilingual nesting is complex.
- Focus on the most common 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.

This session is recorded the presentation deck will be be shared.

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
- 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 other requirements of a specific target market (a locale).

Translation vs. Localization

Translation



Localization



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.

Expected display

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. 

Content creation & editing 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 discovery

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, 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.


- 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 entities.

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

Content entities

- **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 if the content bundle is configured for translation.*
- A field can be one of two types:
 - It contains content.
 - It points to another content entity (entity relationship) or even a concrete revision (entity reference revision)!

* *we will cover paragraphs later*

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.

Field-level 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.

Field-level multilingual configurations

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.

Field-level multilingual configurations

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.

Nesting

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



First rule of Entity Nesting: Don't nest.

Avoid nesting in your content strategy as much as possible.

Second rule of Entity Nesting: Don't use a data model for layout.

“Avoid leveraging systems meant for data structure for layout definition.”

- *Kris Vanderwater, Acquia*

Third rule of Entity Nesting: Keep it simple and consistent.

There are numerous permutations that affect predictable display.

Use consistent site-building patterns and display logic across content entities.

Paragraphs.
Node References.
Custom Content Entities.

**There wasn't anything
else available for
editors to control
layouts!**

**Layout management
for end users is
improving!**

**Some sites may still use
deep nesting
approaches.**

Common issues

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?

Common issues

Core + configuration + 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 - 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

Why is the site slower with multilingual?

- 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.



**Common
Nesting
Examples**

Single Entity Reference

Node with Taxonomy Term Entity Reference

- **Taxonomy 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
- **Taxonomy 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.
- **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

Single Entity Reference

Node with Media Reference

- **The Media asset file is the same for all languages:**
 - A picture without embedded text.
 - Do not translate the Media Entity Reference field.
- **Are the Media assets 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. 
- **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. 

Single Entity Reference

Node with a reusable Custom Block

- **Is the block used across multiple entities?**
 - Translate the block through its own separate workflow.
- **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
- **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.

Entities inside entities

Working with the many other entities.

- Most follow the same multilingual configuration pattern.
- 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 their own? (e.g. can be navigated to?)
- This also applies to *headless* sites using Drupal as a backend.

Single Entity Reference

Node with Paragraphs (Entity Reference Revisions)

- Paragraphs goal is 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.

Single Entity Reference

Node with Paragraphs (Entity Reference Revisions)



[Home](#)

[View](#) [Edit](#) [Delete](#) [Revisions](#) [Translate](#) [Lingotek Metadata](#) [Manage Translations](#)

Submitted by [admin](#) on Wed, 07/15/2020 - 12:41

This is the body of my landing page. Some components below.

Logo



Name

DrupalCon Global 2020

Location

Everywhere

Logo



Name

DrupalCon Barcelona 2020

Location

Barcelona, Catalonia



Complex Nesting


Component-based
site-building approaches

Asymmetrical vs Symmetrical

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.**

Paragraphs

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.

Nest level 2 - Node with a single-level paragraph

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?



[Home](#)

[View](#) [Edit](#) [Delete](#) [Revisions](#) [Translate](#) [Lingotek Metadata](#) [Manage Translations](#)

Submitted by [admin](#) on Wed, 07/15/2020 - 12:41

This is the body of my landing page. Some components below.

Logo



Name

DrupalCon Global 2020

Location

Everywhere

Logo



Name

DrupalCon Barcelona 2020

Location

Barcelona, Catalonia

Nest level 2 - Node with a single-level paragraph

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?




Catalan
English
German
Italian
Español

Inicio

Ver Editar Eliminar Revisiones Traducir Lingotek Metadata Manage Translations

Enviado por [admin](#) el Mié, 15/07/2020 - 12:41

Este es el cuerpo de mi página de aterrizaje. Algunos componentes a continuación.

Logo	Name	Location
	DrupalCon Global 2020	Donde quiera
	DrupalCon Barcelona 2020	Barcelona, Cataluña
	DrupalCamp España Málaga 2020	Cancelado (Málaga, España)

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 layout configs.
- **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.

- 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.

- 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 from 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.

Common deep-level nesting cases

Node > Para 1 (layout) > Para 2 (content fields) > Media Reference

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

Asymmetrical Considerations

Paragraphs

If you do NOT have the extra module, then don't translate the ER field for paragraphs (it 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.

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 after content exists may lead to content loss.

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 select list is used to define layout, don't translate it.
- Consider other field constraints, like character limits.
- Use of content moderation and locale-specific publication status.
- Consider timing source content completion and initial translation, especially in asymmetrical translation models.

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.

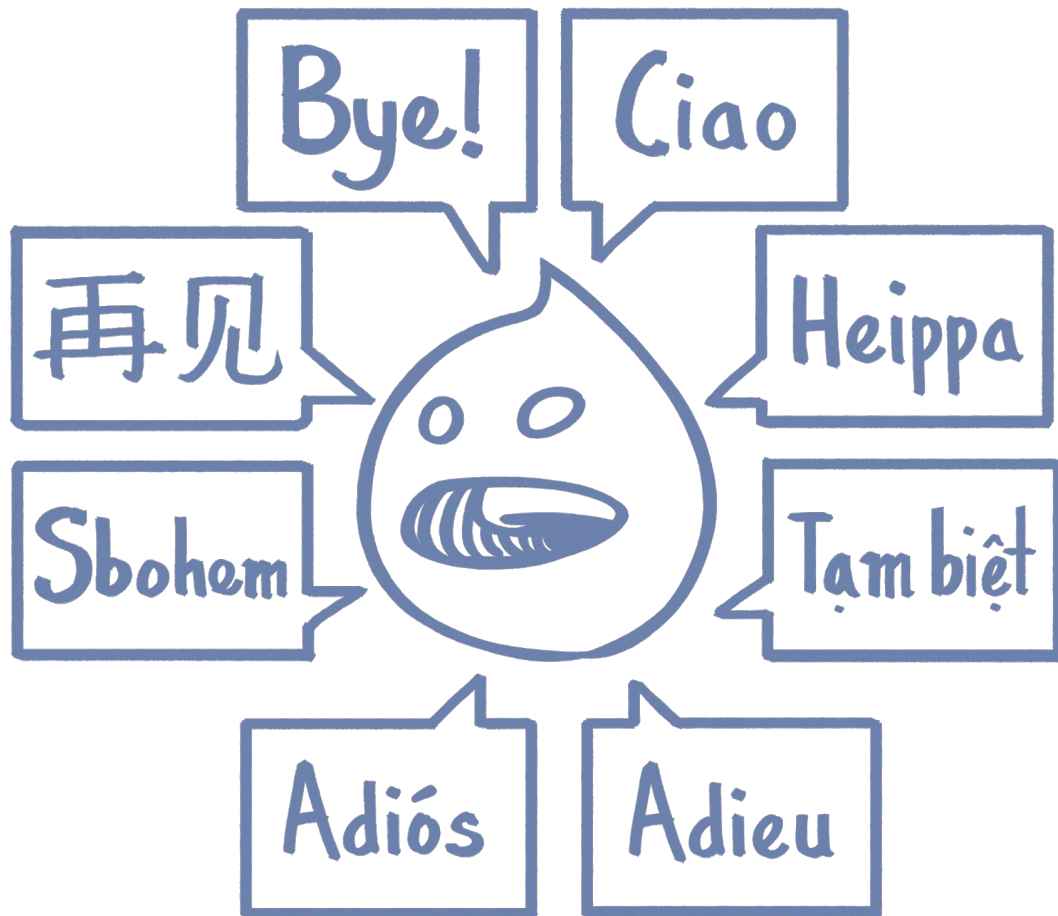
Plan.

Test.

Don't Assume.

Revisit.

Translate.



Resources

Presentations:

- 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@aimeerae.com
 - clopez@lingotek.com



DrupalCon GLOBAL

The Open Source Digital Experience Conference

ONLINE EVERYWHERE

2020 JULY 14-17