Introduction to Entities in Drupal 8

Jennifer Hodgdon
(jhodgdon)

Poplar ProductivityWare
(poplarware.com)

Pacific Northwest Drupal Summit
February 3-4, 2018
Portland, OR

Types of Data in a Drupal Site

Entity data:

- **Content**: Data meant to be viewed by site visitors, or in a site-based app (text, images, files, ...)
- **Configuration**: Data defining how content is viewed or how the site functions (views, image styles, site email address, …)

Non-Entity data:

- **State**: Information about the site that changes frequently (when cron was last run, …)
- **Session**: Information about an individual's interaction with the site (current selection for a Views filter, when they logged in, …)
- **Cache**: Temporarily calculated and stored information that is used to speed up data retrieval

Why Use Entities?

- Entities are the main data storage mechanism in Drupal 8 for content and configuration
- Direct use of the database is deprecated
- Entities have APIs for queries and CRUD operations (create, read, update, delete) that maintain data integrity and allow modules to modify results with hooks
- Content entities are easily made translatable, fieldable, and usable in Views
- Configuration entities are automatically exportable/importable, are managed along with modules, and can have dependencies

Terminology of Entities

Entity: The main mechanism for content/configuration data storage in Drupal, or one item of entity data, or the PHP class representing an entity, or an instance of the class

Content entity: An entity meant for storing content data (node, taxonomy term)

Configuration entity: An entity meant for storing sets of configuration data that share a structure, where a site might have 0, 1, or more instances (examples: view, image style)

Simple configuration: A set of configuration data that has exactly one instance, and therefore doesn't use an entity (examples: settings for a particular module)

Entity type: A group of entity items sharing the same structure (examples: node, taxonomy term, view, image style)

Bundle: A content entity subtype (examples: content type for node, vocabulary for taxonomy term) – configuration entities do not have bundles

Property/Field: Within an entity, data is broken up into properties and/or fields (examples: title of a node, description of a taxonomy term, filter in a view). Properties are defined on the entity type. Fields (for content entities) can be added on later, in a per-bundle basis.

Entity Types in Core

Content entities: aggregator feed, aggregator item, custom block, comment, contact form message, file, menu link, node, taxonomy term, user profile

Configuration entities related to content entities: content entity bundles, entity display modes, entity form modes, field configuration, etc.

Other configuration entities: blocks, contact forms, date formats, WYSIWYG editors, text filters, image styles, languages, search pages, taxonomy vocabularies, tours, user roles, views

Defining a New Entity Type

- Create an interface extending ContentEntityInterface or ConfigEntityInterface – defines your get/set methods
- Create a class implementing your interface and usually extending ContentEntityBase, ConfigEntityBase, or a subclass
- Add ContentEntity or ConfigEntity annotation to the class documentation header
- Define classes for all the *handlers* that your annotation references – for access control, viewing, listing, providing data to Views, URLs/routes, edit forms, … – or use the default classes provided by Drupal Core
- If defining a content entity type, and it uses bundles, also define a configuration entity type for the bundle (use ConfigEntityBundleBase as the base class)
- If defining a configuration entity type, also define the configuration schema in a YAML file

====> I strongly suggest looking at existing examples! Try searching for classes/interfaces extending the base class/interface on api.drupal.org <===

Querying Entities

```
// Code without dependency injection:
  $query = \Drupal::entityQuery('myentity');

// Code with dependency injection:
  $query = $container
    ->get('entity_type_manager')
    ->getStorage('myentity')
    ->getQuery();


// Add conditions:
   $query->condition('label', '%foo%', 'LIKE');


// Execute to get list of entity IDs:
   $ids = $query->execute();
```

Loading Existing Entities

```
// Code without dependency injection:
  $storage = \Drupal::entityTypeManager()
    ->getStorage('myentity');

// Code with dependency injection:
  $storage = $container
    ->get('entity_type_manager')
    ->getStorage('myentity');


// Load one or multiple entities:
    $entity = $storage->load($id);
    $entities = $storage->loadMultiple($ids);


// Alternate method, using entity class:
    $entity = EntityClassName::load($id);
    $entities = EntityClassName::loadMultiple($ids);
```

Updating and Deleting Entities

```
// Update

   $entity
      ->set('label', 'Foo')
      // Call other methods...
      ->save();


// Delete


   $entity->delete();
   $storage->delete($entities);
```

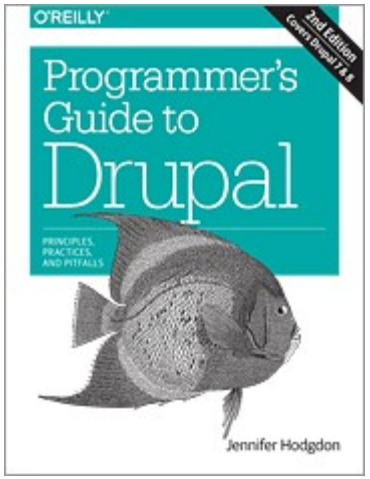
Creating New Entities

```
// Create from list of properties (can be empty
  array):
   $entity = $storage->create($properties);


// Alternate method, using entity class:
   $entity = EntityClassName::create($properties);


// Set additional properties and save:
  $entity
     ->set('label', 'Foo')
     // Call other methods...
     ->save();
```

Example: Help Topics

(cut to code editor...)

Shameless Plug



Programmer's Guide to Drupal, from O'Reilly Media
The second edition covers Drupal 7 and 8.

http://shop.oreilly.com/product/0636920034612.do