Content entities

A content entity is a generic fieldable model that is defined in code. It inherits all functionality from the base Entity model defined in drupal core.

Take a look at **core/modules/user/src/Entity/User.php** and click further on the classes the current class you are in extends:

class User extends **ContentEntityBase** implements UserInterface
-> abstract class ContentEntityBase extends **EntityBase**
-> abstract class EntityBase implements **EntityInterface**

This Object-oriented way of data-modelling makes sense. All the classes that inherit from EntityBase get a ton of functionality for free:

- Makes the entity **fieldable** if desired (start adding fields via the ui on-the-go)
- Makes the entity **translatable,** if desired
- I**nherits various generic methods** for obtaining data
- Can make use of the core **workflow** (for moderation status of the entities: draft, published, expired, ...)
- Can make use of the powerful core **revisioning** system (keeps a history of the entity on every change)
- Create, read, update, and delete (**CRUD**) functionality with the desired security.
- **Views** integration by default. Views is the core module for creating lists with filtering, search etc.

The following generic methods for content entities are provided in core:

```
Entity::create()
Entity::load()
Entity::save()
Entity::id()
Entity::bundle()
Entity::isNew()
Entity::label()
```

An entity is purely defined in code, unless:

- We add bundles to the entity (f.e. Node entity has bundle ‘article’, ‘page’ in the standard profile). This is not what we will do in this course.
- We add fieldable functionality to the entity (this is an option when creating content entities) to add fields via the interface. We will do this in this course.

When one of the previous conditions is met, the extra configuration gets stored in the database and comes with yaml files when exporting configuration.