

Content Entity Entity Query

ENTITY, NODE, TERM, USER

```

\Drupal\Core\Access\AccessibleInterface
\D...Core\Cache\CacheableDependencyInterface
\Drupal\Core\Entity\ContentEntityBase
\Drupal\Core\Entity\ContentEntityInterface
\Drupal\Core\Entity\ContentEntityInterface

```

Content entities: 'aggregator_feed',
'aggregator_item', 'block_content', 'comment',
'contact_message', 'file', 'menu_link_content',
'menu_link_content', 'node', 'shortout',
'taxonomy_term', 'user'.

```

$entity = Entity::create($values); P
$entity = Entity::load($id); P
$entities = Entity::loadMultiple($ids); P
$entity->save();
$entity->delete();

```

```

$entity_id = $entity->id();
$bundle = $entity->bundle();
$not_saved = $entity->isNew();
$can_edit = $entity->access('edit',
    $account);

```

```

\Drupal\node\NodeInterface
\Drupal\node\Entity\Node
\Drupal\user\Entity\OwnerInterface

```

```

$node = $this->entityTypeManager
->getStorage('node')
->load($nid); P
$node = Node::load($nid); P
$node->save();
$node->delete();

```

```

$node_type = $node->getType();
$title = $node->getTitle();
$visible = $node->isPublished();
$node->setPromoted(TRUE);
$uid = $node->getOwnerId();
$author = $node->getOwner();

```

```

Drupal\taxonomy\TermInterface
Drupal\taxonomy\Entity\Term

```

```

$term = $this->entityTypeManager
->getStorage('taxonomy_term')
->load($tid);
$term = Term::load($tid); P

```

```

$default = $entity->isDefaultRevision($new);
$entity->setNewRevision($value);
$revision_id = $entity->getRevisionId();

```

ENTITY FIELDS

```

\Drupal\Core\Entity\FieldableEntityInterface
\Drupal\Core\Field\FieldItemInterface
\Drupal\Core\Field\FieldItemListInterface
\Drupal\Core\TypedData\TypedDataInterface

```

```

$body text = $node->body->value;
$body text = $node->get('body')->value;
$body_array = $node->body->getValue();
$body_text example: <p>Hello</p>
$body_array example: array(
    0 => array(
        'value' => '<p>Hello</p>',
        'summary' => '',
        'format' => 'basic_html',
    ))

```

```

$field_item_list = $entity
->get('field_name');
$fields_array = $entity->toArray();
$field_item_lists = $entity->getFields();
$fields_array Array of raw values per field.
$field_item_lists Array of FieldItemList per field.

```

```

$text_format = $node->body->format;
$third_tag_id = $node->field_tags[2]
->target_id;

```

```

$has_body = $node->hasField('body');
$empty_body = $node->body->isEmpty();
$fields = $node->getTranslatableFields();
$entity->set('title', 'New title');

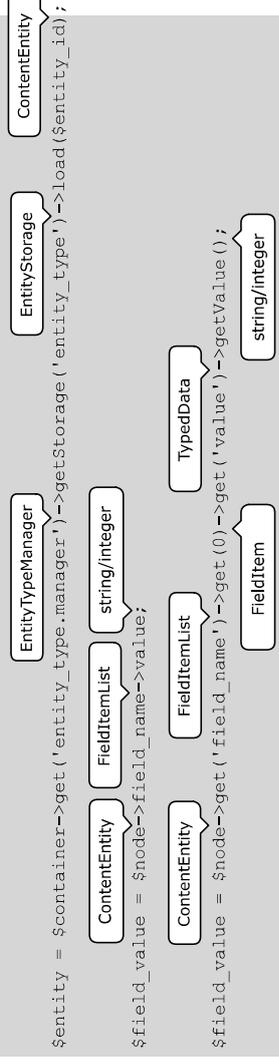
```

```

$field_names = array_keys($node
->getFields());
$field_names = array_keys($this
->entityFieldManager
->getFieldDefinitions('node', 'article'));
$definition = $entity
->getFieldDefinition('field_name');

```

Node field names: 'nid', 'uid', 'vid', 'type',
'langcode', 'title', 'uid', 'status', 'created', 'changed',



'promote', 'sticky', 'revision_timestamp', 'revision_uid',
'revision_log', 'path', 'body', 'comment', 'field_image',
'field_tags'.

```

$field_properties = array_keys($node
->getFieldDefinition('body')
->getFieldStorageDefinitions()
->getPropertyDefinitions());

```

Text plain field properties: 'value'.
Text formatted field properties: 'value', 'format',
'processed'.

Text with summary field properties: 'value', 'format',
'processed', 'summary', 'summary_processed'.

Taxonomy entity reference field properties:

```

'target_id', 'entity'.
File entity reference field properties: 'target_id',
'entity', 'display', 'description'.
Image entity reference field properties: 'target_id',
'entity', 'display', 'description', 'alt', 'title', 'width',
'height'.
Date field properties: 'value', 'date'.

```

ENTITY REFERENCE FIELDS

```

$vid = $node->field_tags->entity
->getVocabularyId();
$uri = $node->field_file->entity
->getFileUri();

```

```

$name = $node->getOwner()->getDisplayName();
$roles = $node->getOwner()->getRoles();

```



ENTITY STORAGE

```
\Drupal\Core\Entity\EntityStorageInterface
\D...Core\Entity\Sql\SqlContentEntityStorage
hook_entity_*(*)
* = create, load, insert, update, view,
view_alter, view_mode_alter,
view_display_alter, prepare_view,
presave, predelete, delete.
hook_ENTITY_TYPE_*(*)
* = create, load, insert, update, view,
view_alter, presave, predelete, delete.
```

```
$this->entityStorage = $container
->get('entity_type.manager')
->getStorage('entity_type');
$entityStorage = \Drupal::
entityTypeManager()
->getStorage('entity_type'); P
$entity = $this->entityStorage
->load($entity_id);
$entities = $this->entityStorage
->loadMultiple($entity_ids);
$revision = $this->entityStorage
->loadRevision($revision_id);
$entity = $this->entityStorage
->create($entity_values);
$this->entityStorage->delete($entities);
$this->entityStorage->save($entity);
```

ENTITY VIEW

```
\Drupal\Core\Entity\EntityViewBuilderInterface
\Drupal\Core\Entity\EntityViewBuilder
$this->viewBuilder = $this
->entityTypeManager
->getViewBuilder('node');
$view_builder = \Drupal::entityTypeManager()
->getViewBuilder('node'); P
$build = $view_builder
->view($node, 'teaser');
```

ENTITY ACCESS

```
\Drupal\Core\Access\AccessibleInterface
hook_entity_*(*)
hook_ENTITY_TYPE_*(*)
* = access, create_access, field_access.
$entity->access($operation);
$entity->field_name->access($operation);
Entity access operations: view, create, update, delete.
Entities can define additional operations.
Entity field_access operations: view, edit.
```

ENTITY QUERY

```
\Drupal\Core\Config\Entity\Query\Query
\Drupal\Core\Entity\Query\QueryFactory
\Drupal\Core\Entity\Query\QueryInterface
$this->entityQuery = $container
->get('entity.query');
$query = $this->entityQuery
->getAggregate('node');
$query = \Drupal::
entityQueryAggregate('node'); P
$node_ids = $this->entityQuery->get('node')
->condition('type', 'article')
->condition('status', 1)
->execute();
```

```
$query = $this->entityQuery
->get('aggregator_feed');
$query->condition = $query->orConditionGroup()
->condition('title', $feed->label())
->condition('url', $feed->getUrl());
$duplicates = $query
->condition($or_condition)
->execute();
```

Condition operators (3rd param):

```
'=', '<', '>', '>=', '<=', '<=',
'STARTS_WITH', 'CONTAINS',
'ENDS_WITH', 'IN', 'NOT IN',
'BETWEEN'.
```

Query array

EntityViewBuilder

(render) array

QueryFactory

EntityTypeManager

```
$query = $container->get('entity.query')->get('content_type')->execute();
```

```
$build = $this->entityTypeManager->getViewBuilder('node')->view($node, 'teaser');
```

ENTITY QUERY: RANGE, SORT, PAGER

```
$name_taken = (bool) $this->entityQuery
->get('user')
->condition('name', $name)
->range(0, 1)
->count()
->execute();
$sorted_terms = $this->entityQuery
->get('taxonomy_term')
->sort('weight')
->sort('name')
->addrag('term_access')
->execute();
$entity_query = $this->queryFactory
->get('user')
->condition('uid', 0, '<')
->pager(50);
$header = $this->buildHeader();
$entity_query->tableSort($header);
$uids = $entity_query->execute();
```

SERVICES

```
entity_query
entity_repository
entity_display_repository
entity_field_manager
entity_type.bundle.info
entity_type.manager
entity_type.repository
```



wizzlern
de Drupal trainers

DOCUMENTATION

Entity API docs: api.drupal.org
Community documentation: Entity.API-Drupal8
Hooks: file entity.api.php

P = Procedural code. Do not use this in OO code because it can not be unit tested or replaced.

