



# Leveraging the Drupal Core Entity API for your custom Entity Types

July 2023

Hristo Chonov @ 1xINTERNET GmbH



# Hristo Chonov

Studied Computer Science at TU Darmstadt, Germany

Co-Maintainer for Drupal 8/9/10 Core Entity API

Author / maintainer of multiple contrib extensions for Drupal

- Autosave
- Conflict 2.x
- Prefetch Cache

Lead Software Engineer at 1xINTERNET GmbH

# Credits

## Entity Type Walkthrough

<https://git.io/d8entity>

Tobias Zimmermann (tstoeckler)

Profile Posts Starred



DrupalCode profile

Facebook  
Twitter  
Google  
Gratipay  
GitHub

Essen, Germany

Current Role(s):

resc.

IRC: tstoeckler

### Professional Info

Companies Worked For  
[42robots](#), [Acquia](#), [bio.logis](#)  
 Genetic Information Management GmbH, [comm-press](#), [erdfisch](#),  
 Factorial GmbH, [MD Systems GmbH](#), [Reinblau](#), [Violka IT](#)

### Personal Info

Pronouns: he/him

Primary language: German

Other languages: English  
German

On Drupal.org for 16 years 6 months

Over 10 edits to documentation

#### Bio:

- Maintainer of the [Libraries API](#) module
- Winner of Trivia Night, DrupalCon 2013, Prague, Team "CREATE TABLE."
- Maintainer of the [Configuration Translation](#) module in Drupal 8 core
- Maintainer of the [Shortcut](#) module in Drupal 8 core
- 4th place at Trivia Night, DrupalCon 2015, Barcelona, Team "TABLE ALREADY EXIST"
- Maintainer of the [Entity API](#) subsystem in Drupal 8 core

#### Documentation guides

[Contributed module documentation – Libraries API 8.x](#)

[Contributed modules – Libraries API 7.x](#)

✓ Can opt projects into [security advisory coverage](#)

#### Drupal events:

DrupalCon Munich 2012  
 DrupalCon Prague 2013  
 DrupalCon Amsterdam 2014  
 DrupalCon Barcelona 2015  
 DrupalCon New Orleans 2016  
 DrupalCon Dublin 2016  
 Drupal Europe 2018

#### Credited on 3 security advisories

- [Drupal core](#), 2 issues
- [JSON:API](#), 1 issue

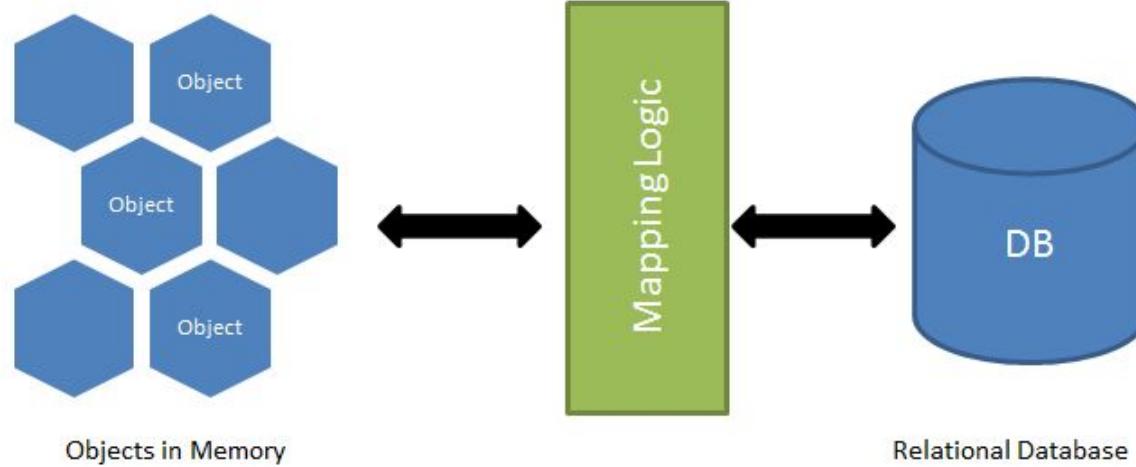
#### Credited on 496 fixed issues

- [Drupal core](#), 376 issues
- [Group](#), 1 issue
- [Paragraphs](#), 9 issues
- [Consumers](#), 1 issue
- [Linkit](#), 1 issue
- [Config Overlay](#), 11 issues
- [Field default token](#), 2 issues
- [Group Content with String IDs](#), 2 issues
- [Open Social](#), 1 issue
- [Community DACH](#), 1 issue
- [deGov](#), 1 issue
- [Image Effects](#), 1 issue
- [German translation](#), 4 issues
- [Voting API](#), 1 issue
- [Entity API](#), 12 issues

<https://1xinter.net/Zxv>

# What is ORM?

## O/R Mapping



# Drupal - Entity API

- ORM implementation
- supporting different variants of the same entity type
- supporting fields/properties defined in code
- supporting custom defined fields through configuration
- supporting translatability for each field
- supporting versions / revisions

# Basics

The screenshot shows a file browser interface with the following directory structure:

- umami-drupal (~workspace/umami-drupal)
  - .ddev
  - vendor
  - web
    - core
  - modules
    - contrib
    - custom
      - event
        - src
          - Entity
        - Event.php
- event.info.yml

The screenshot shows the content of the Event.php file:

```
<?php  
namespace Drupal\event\Entity;  
  
use Drupal\Core\Entity\ContentEntityBase;  
  
/**  
 * @ContentEntityType  
 *   id = "event",  
 *   label = @Translation("Event"),  
 *   base_table = "event",  
 *   entity_keys = {  
 *     "id" = "id",  
 *     "uuid" = "uuid",  
 *   },  
 * )  
 */  
class Event extends ContentEntityBase {}
```

The screenshot shows the structure of the event table:

event	/* The base table for event entities. */
uuid	varchar(128)
id	int(10) unsigned

## Basics...

```
1 <?php  
2  
3 use Drupal\event\Entity\Event;  
4  
5 $event = Event::create();  
6 $event->save();
```

The screenshot shows a database management interface with a table named 'event'. The table has two columns: 'id' and 'uuid'. There is one row with the value '1' in the 'id' column and '6fd5e14f-6ba8-464c-b10a-1274c8557958' in the 'uuid' column. The interface includes a toolbar with various icons for navigation and operations like insert, update, and delete.

WHERE

	id	uuid
1	1	6fd5e14f-6ba8-464c-b10a-1274c8557958

# Entity Keys

```
<?php

namespace Drupal\event\Entity;

use Drupal\Core\Entity\ContentEntityBase;

/**
 * @ContentEntityType(
 *   id = "event",
 *   label = @Translation("Event"),
 *   base_table = "event",
 *   entity_keys = {
 *     "id" = "id",
 *     "uuid" = "uuid",
 *     "owner" = "author",
 *     "published" = "published",
 *   },
 * )
 */

```



```
25   class Event extends ContentEntityBase implements EntityOwnerInterface, EntityPublishedInterface {

26
27     use EntityOwnerTrait, EntityPublishedTrait;
28
29     /**
30      * {@inheritDoc}
31     */
32     public static function baseFieldDefinitions(EntityTypeInterface $entity_type) {
33       // Get the field definitions for 'id' and 'uuid' from the parent.
34       $fields = parent::baseFieldDefinitions($entity_type);
35
36       // Get the field definitions for 'author' and 'published' from the traits.
37       $fields += static::ownerBaseFieldDefinitions($entity_type);
38       $fields += static::publishedBaseFieldDefinitions($entity_type);
39
40       return $fields;
41     }
42
43   }
```

	<b>event</b>	/* The base table for event entities. */
	<b>uuid</b>	varchar(128)
	<b>author</b> /* The ID of the target entity. */ int(10) unsigned	
	<b>published</b>	tinyint(4)
	<b>id</b>	int(10) unsigned

# Base fields

```
 /**
 * @ContentEntityType(
 *   id = "event",
 *   label = @Translation("Event"),
 *   base_table = "event",
 *   entity_keys = {
 *     "id" = "id",
 *     "uuid" = "uuid",
 *     "owner" = "author",
 *     "published" = "published",
 *     "label" => "title",
 *   },
 * )
 */
```

```
public static function baseFieldDefinitions(EntityType $entity_type) {
    // Get the field definitions for 'id' and 'uuid' from the parent class.
    $fields = parent::baseFieldDefinitions($entity_type);

    // Get the field definitions for 'author' and 'published'.
    $fields += static::ownerBaseFieldDefinitions($entity_type);
    $fields += static::publishedBaseFieldDefinitions($entity_type);

    // Create the 'title' field definition.
    $fields['title'] = BaseFieldDefinition::create(type: 'string')
        ->setLabel(t(string: 'Title'))
        ->setRequired(required: TRUE);

    // Create the 'date' field definition.
    $fields['date'] = BaseFieldDefinition::create(type: 'datetime')
        ->setLabel(t(string: 'Date'))
        ->setRequired(required: TRUE);

    // Create the 'description' field definition.
    $fields['description'] = BaseFieldDefinition::create(type: 'text_long')
        ->setLabel(t(string: 'Description')));

    return $fields;
}
```

 event	/* The base table for event entities. */
 <b>uuid</b>	varchar(128)
 <b>author</b> /* The ID of the target entity. */ int(10) unsigned	
 <b>published</b>	tinyint(4)
 <b>title</b>	varchar(255)
 <b>date</b> /* The date value. */	varchar(20)
 <b>description_value</b>	longtext
 <b>description_format</b>	varchar(255)
 <b>id</b>	int(10) unsigned

# Creating and saving an Entity

1x

```
$event = Event::create([
    'title' => 'Leveraging the Drupal Core Entity API for your custom entity types',
    'date' => '2023-07-21T10:45:00',
    'description' => [
        'value' => '<p>This session is a walk-through defining a custom entity type. Some of the major topics will be about:</p>
<p>
<ul>
<li>Bundles</li>
<li>Revision and translation support</li>
<li>Configuring the form display of the entity</li>
<li>Configuring the view display of the entity</li>
<li>Constraints and validation</li>
<li>Access control</li>
<li>Entity queries</li>
<li>Configuration vs content entities</li>
</ul>
</p>',
        'format' => 'restricted_html',
    ],
]);
$event->save();
```

## Base fields - Getters and Setters

```
/**                                     */
 * Returns the title of the event.      *
 * @return string                      *
 *   The event title.                 */
public function getTitle() {           *
|   return $this->get('title')->value;  *
}                                     *

> $event->get('title');              *
= Drupal\Core\Field\FieldItemList {#7951
    0: Drupal\Core\Field\Plugin\Field\FieldType\StringItem {#7955},
}

> $event->get('title')->value;       *
= "Leveraging the Drupal Core Entity API for your custom entity types"
```

# Entity view

- requires a route on which the entity's field values are output on a given path
- can be automated by amending the entity annotation
- contributed [Entity API](#) provides various enhancements to the core Entity API. One such enhancement is the ability to more easily provide permissions for entity types.

<https://www.drupal.org/project/entity>

```
* @ContentEntityType(  
*   id = "event",  
*   label = @Translation("Event"),  
*   base_table = "event",  
*   entity_keys = {  
*     "id" = "id",  
*     "uuid" = "uuid",  
*     "owner" = "author",  
*     "published" = "published",  
*     "label" = "title",  
*   },  
*   handlers = {  
*     "access" = "Drupal\entity\EntityAccessControlHandler",  
*     "permission_provider" = "Drupal\entity\EntityPermissionProvider",  
*     "route_provider" = {  
*       "default" = "Drupal\entity\Routing\DefaultHtmlRouteProvider",  
*     },  
*   },  
*   links = {  
*     "canonical" = "/event/{event}",  
*   },  
*   admin_permission = "administer event",  
* )  
) */
```

# Entity view...

1x

The screenshot shows a web browser window displaying the Umami Food Magazine website. The URL in the address bar is <https://umami-drupal.dddev.site/en/event/1>. The page title is "Leveraging the Drupal Core Entity API for your custom entity types". The Umami logo is visible at the top left, and the navigation menu includes "Home", "Articles", and "Recipes". A search bar is located at the top right. The main content area features a section titled "Recipe collections" with a grid of categories: Alcohol free, Cake, Dairy-free, Egg; Baked, Carrots, Dessert, Grow your own; Baking, Chocolate, Dinner party, Healthy; Breakfast, Cocktail party, Drinks, Herbs. Below this is a promotional image for the magazine, showing a cover with the headline "THE BAKING SPECIAL" and "3 ISSUE BUNDLE!". The magazine cover features a person holding a dessert. To the right of the magazine image, there is text about the magazine and a "Find out more" link. On the far right, there is a sidebar with the heading "Tell us what you think" and a "Contact" link.

Back to Administration

English Espanol

Search by keyword, ingredient, dish

Search

My account Log out

umami  
FOOD MAGAZINE

Home Articles Recipes

Home > Leveraging the Drupal Core Entity API for your custom entity types

## Leveraging the Drupal Core Entity API for your custom entity types

### Recipe collections

Alcohol free	Cake	Dairy-free	Egg
Baked	Carrots	Dessert	Grow your own
Baking	Chocolate	Dinner party	Healthy
Breakfast	Cocktail party	Drinks	Herbs

Umami Food Magazine

Skills and know-how. Magazine exclusive articles, recipes and plenty of reasons to get your copy today.

Find out more >

3 ISSUE BUNDLE!

Tell us what you think

Contact

# Entity view...

```
$fields['date'] = BaseFieldDefinition::create( type: 'datetime')
->setLabel(t( string: 'Date'))
->setRequired( required: TRUE)
->setDisplayOptions( display_context: 'view', [
  'label' => 'inline',
  'settings' => [
    'format_type' => 'short',
  ],
  'weight' => 0,
]);
});
```



```
$fields['description'] = BaseFieldDefinition
->setLabel(t( string: 'Description'))
->setDisplayOptions( display_context: 'view', [
  'label' => 'hidden',
  'weight' => 10,
]);
});
```

Home      Articles      Recipes

[Home](#) » Leveraging the Drupal Core Entity API for your custom entity types

## Leveraging the Drupal Core Entity API for your custom entity types

**Date:** 07/21/2023 - 13:45

This session is a walk-through defining a custom entity type. Some of the major topics will be about:

- Bundles
- Revision and translation support
- Configuring the form display of the entity type
- Configuring the view display of the entity type
- Constraints and validation
- Access control
- Entity queries
- Configuration vs content entity types

# Entity form

```
*   handlers = {
*     "access" = "Drupal\entity\EntityAccessControlHandler",
*     "permission_provider" = "Drupal\entity\EntityPermissionProvider",
*     "route_provider" = {
*       "default" = "Drupal\entity\Routing\DefaultHtmlRouteProvider",
*     },
*     "form" = {
*       "add" = "Drupal\Core\Entity\ContentEntityForm",
*       "edit" = "Drupal\Core\Entity\ContentEntityForm",
*       "delete" = "Drupal\Core\Entity\ContentEntityDeleteForm",
*     },
*   },
*
*   links = {
*     "canonical" = "/event/{event}",
*     "add-form" = "/admin/content/events/add",
*     "edit-form" = "/admin/content/events/manage/{event}",
*     "delete-form" = "/admin/content/events/manage/{event}/delete",
*   },
*
```

1x

```
$fields['title'] = BaseFieldDefinition::create( type: 'string')
->setLabel(t( string: 'Title'))
->setRequired( required: TRUE)
->setDisplayOptions( display_context: 'form', ['weight' => 0]);
```

```
$fields['date'] = BaseFieldDefinition::create( type: 'datetime')
->setLabel(t( string: 'Date'))
->setRequired( required: TRUE)
->setDisplayOptions( display_context: 'view', [
  'label' => 'inline',
  'settings' => [
    'format_type' => 'short',
  ],
  'weight' => 0,
])
->setDisplayOptions( display_context: 'form', ['weight' => 10]);
```

```
$fields['description'] = BaseFieldDefinition::create( type: 'text_long')
->setLabel(t( string: 'Description'))
->setDisplayOptions( display_context: 'view', [
  'label' => 'hidden',
  'weight' => 10,
])
->setDisplayOptions( display_context: 'form', ['weight' => 20]);
```

```
$fields['published']->setDisplayOptions( display_context: 'form', [
  'settings' => [
    'display_label' => TRUE,
  ],
  'weight' => 30,
]);
```

# Entity form...

The screenshot shows a web browser window displaying a Drupal administrative interface for adding a new event. The URL in the address bar is <https://umami-drupal.ddev.site/en/admin/content/events/add>. The page title is "Add event".

The form fields include:

- Title\***: An input field for the event title.
- Date\***: A date and time picker field.
- Description**: A rich text editor with a toolbar containing bold, italic, and other styling options. Below the toolbar is a large text area for the event description.

Below the form fields are the following controls:

- Text format**: A dropdown menu set to "Basic HTML".
- About text formats**: A link to learn more about text formats.
- Published**: A toggle switch that is currently turned on (green).
- Save**: A blue button to save the event.

A vertical sidebar on the left contains various icons for managing content, such as add, edit, delete, and move.

# Entity listing

```
<?php

namespace Drupal\event\Controller;

use Drupal\Core\Entity\EntityInterface;
use Drupal\Core\Entity\EntityListBuilder;

/** 
 * Defines an implementation to build a listing of event entities.
 */
class EventListBuilder extends EntityListBuilder {

  /** {@inheritDoc} ...*/
  public function buildHeader() {
    $header = [];
    $header['title'] = $this->t( string: 'Title' );
    $header['date'] = $this->t( string: 'Date' );
    $header['published'] = $this->t( string: 'Published' );
    return $header + parent::buildHeader();
  }

  /** {@inheritDoc} ...*/
  public function buildRow(EntityInterface $event) {
    /** @var \Drupal\event\Entity\Event $event */
    $row = [];
    $row['title'] = $event->toLink();
    $row['date'] = $event->date->date->format('m/d/y h:i:s a');
    $row['published'] = $event->isPublished() ? $this->t( string: 'Yes' ) : $this->t( string: 'No' );
    return $row + parent::buildRow($event);
  }
}
```

```
 /**
 * @ContentEntityType(
 *   id = "event",
 *   label = @Translation("Event"),
 *   label_collection = @Translation("Events"),
 *   label_singular = @Translation("event"),
 *   label_plural = @Translation("events"),
 *   base_table = "event",
 *   handlers = {
 *     "access" = "Drupal\entity\EntityAccessControlHandler",
 *     "permission_provider" = "Drupal\entity\EntityPermissionProvider",
 *     "route_provider" = {
 *       "default" = "Drupal\entity\Routing\DefaultHtmlRouteProvider",
 *     },
 *     "form" = {
 *       "add" = "Drupal\Core\Entity\ContentEntityForm",
 *       "edit" = "Drupal\Core\Entity\ContentEntityForm",
 *       "delete" = "Drupal\Core\Entity\ContentEntityDeleteForm",
 *     },
 *     "list_builder" = "Drupal\event\Controller\EventListBuilder",
 *     "local_action_provider" = {
 *       "collection" = "Drupal\entity\Menu\EntityCollectionLocalActionProvider",
 *     },
 *   },
 *   links = {
 *     "canonical" = "/event/{event}",
 *     "add-form" = "/admin/content/events/add",
 *     "edit-form" = "/admin/content/events/manage",
 *     "delete-form" = "/admin/content/events/manage/{event}",
 *     "collection" = "/admin/content/events",
 *   },
 * ),
```

# Entity listing

The screenshot shows a Drupal administrative interface for managing content. The page title is "Events" with a star icon. On the left, there is a vertical sidebar with various icons: a drop-down menu, a file icon, a folder icon, a trash bin, a square icon, a person icon, a clock icon, a gear icon, and a question mark icon. At the bottom of the sidebar is a circular arrow icon.

The main content area displays a table with the following columns: Title, Date, Published, and Operations. There is one row in the table:

Title	Date	Published	Operations
Leveraging the Drupal Core Entity API for your custom entity types	07/21/23 10:45:00 am	Yes	<a href="#">Edit</a> <a href="#">Delete</a>

At the top of the page, there is a browser header with the URL <https://umami-drupal.ddev.site/en/admin/content/events>. To the right of the URL are standard browser controls (Back, Forward, Stop, Refresh) and a zoom level of 100%. Below the browser header is the Admin Toolbar with a quick search bar, shortcuts, and a message: "This site is intended for demonstration purposes." A user icon for "admin" is also present.

# Entity constraints and validation

```
$fields['maximum'] = BaseFieldDefinition::create( type: 'integer')
->setLabel(t( string: 'Maximum number of attendees'))
->setSetting( setting_name: 'min', value: 1)
->setDisplayOptions( display_context: 'form', ['weight' => 23]);

$fields['attendees'] = BaseFieldDefinition::create( type: 'entity_reference')
->setLabel(t( string: 'Attendees'))
->setSetting( setting_name: 'target_type', value: 'user')
->setCardinality( cardinality: FieldStorageDefinitionInterface::CARDINALITY_UNLIMITED)
->setDisplayOptions( display_context: 'view', ['weight' => 20])
->setDisplayOptions( display_context: 'form', ['weight' => 27]);
```

<b>event</b> /* The base table for event entities. */	
• <b>uuid</b>	varchar(128)
• <b>author</b> /* The ID of the target entity. */	int(10) unsigned
• <b>published</b>	tinyint(4)
• <b>title</b>	varchar(255)
• <b>date</b> /* The date value. */	varchar(20)
• <b>description_value</b>	longtext
• <b>description_format</b>	varchar(255)
• <b>maximum</b>	int(11)
• <b>id</b>	int(10) unsigned

<b>event_attendees</b> /* Data storage for event field attendees. */	
• <b>bundle</b> /* The field instance bundle to which this row belongs, used whe...	
• <b>revision_id</b> /* The entity revision id this data is attached to, which for an...	
• <b>attendees_target_id</b> /* The ID of the target entity. */	int(10) unsigned
• <b>deleted</b> /* A boolean indicating whether this data item has been deleted */	
• <b>entity_id</b> /* The entity id this data is attached to */	int(10) unsigned
• <b>langcode</b> /* The language code for this data item. */	varchar(32)
• <b>delta</b> /* The sequence number for this data item, used for multi-value ... */	

# Entity constraints and validation

AttendeeCountConstraint.php ×

```
<?php

namespace Drupal\event\Plugin\Validation\Constraint;

use Symfony\Component\Validator\Constraint;

/**
 * @Constraint(
 *   id = "AttendeeCount",
 *   label = @Translation("Attendee count"),
 * )
 */
class AttendeeCountConstraint extends Constraint {

    /**
     * @var string
     */
    public $message = 'The event %title only allows %maximum attendees.';

}

$fields['attendees'] = BaseFieldDefinition::create( type: 'entity_reference')
->setLabel(t( string: 'Attendees'))
->setSetting( setting_name: 'target_type', value: 'user')
->setCardinality( cardinality: FieldStorageDefinitionInterface::CARDINALITY_UNI
->addConstraint( constraint_name: 'AttendeeCount')
->setDisplayOptions( display_context: 'view', ['weight' => 20])
->setDisplayOptions( display_context: 'form', ['weight' => 27]);
```

AttendeeCountConstraintValidator.php ×

```
<?php

namespace Drupal\event\Plugin\Validation\Constraint;

use Symfony\Component\Validator\Constraint;
use Symfony\Component\Validator\ConstraintValidator;

class AttendeeCountConstraintValidator extends ConstraintValidator {

    public function validate($value, Constraint $constraint) {
        /* @var \Drupal\Core\Field\FieldItemListInterface $value */
        /* @var \Drupal\event\Plugin\Validation\Constraint\AttendeeCountConstraint $constraint */
        /* @var \Drupal\Core\Entity\Plugin\DataType\EntityAdapter $adapter */
        $adapter = $value->getParent();
        /* @var \Drupal\event\Entity\Event $event */
        $event = $adapter->getEntity();
        $maximum = $event->maximum->value;
        if (count($value) > $maximum) {
            $this->context->buildViolation($constraint->message)
                ->setParameter('%title', $event->getTitle())
                ->setParameter('%maximum', $maximum)
                ->addViolation();
        }
    }
}
```

# Entity constraints and validation

```
PHP event.module ×

1 <?php
2
3 use Drupal\Core\Form\FormStateInterface;
4
5 /**
6  * Implements hook_form_BASE_FORM_ID_alter() event entity forms.
7 */
8 function event_form_event_edit_form_alter(&$form, FormStateInterface $form_state, $form_id) {
9   /** @var \Drupal\event\Entity\Event $entity */
10  $entity = $form_state->getFormObject()
11    ->getEntity();
12  if (\Drupal::currentUser()->id() != $entity->getOwnerId()) {
13    $form['maximum']['#disabled'] = TRUE;
14  }
15}
```

# Entity constraints and validation

The screenshot shows a Drupal administrative interface for managing events. On the left is a sidebar with various icons for content types like pages, users, and media. The main content area has a breadcrumb trail: Back to site | Administration / Content / Events / Edi... A yellow banner at the top right states, "This site is intended for demonstration purposes." Below it, the page title is "Edit Leveraging the Drupal Core Entity API for your custom entity types". A dark brown error message box contains the text: "Error message The event "Leveraging the Drupal Core Entity API for your custom entity types" only allows 5 attendees." The form fields shown are "Title\*" with the value "Leveraging the Drupal Core Entity API for your custom entity types" and "Date\*" with the value "21/07/2023" and "13:45:00".

Back to site | Administration / Content / Events / Edi...

Admin Toolbar quick search

This site is intended for demonstration purposes.

## Edit Leveraging the Drupal Core Entity API for your custom entity types

**Error message**

The event "Leveraging the Drupal Core Entity API for your custom entity types" only allows 5 attendees.

Title\*

Leveraging the Drupal Core Entity API for your custom entity types

Date\*

21/07/2023

13:45:00

# Entity constraints and validation

The screenshot shows a Drupal administrative interface for managing entities. The page title is "Edit Leveraging the Drupal Core Entity API for your custom entity types ☆". A sidebar on the left contains various icons for file operations like copy, move, delete, and settings.

The main content area displays a table of attendees:

Attendees
Margaret Hopper (2)
Grace Hamilton (3)
Megan Collins Quinlan (6)
Samuel Adamson (7)
Umami (4)
Agent Smith (8)

Below the table, there is a button labeled "Add another item" and a "Published" status indicator. At the bottom, there are "Save" and "Delete" buttons.

The URL in the browser bar is <https://umami-drupal.dddev.site/en/admin/content/events/manage/1>.

# Tracking changes to entities

```
class Event extends ContentEntityBase implements EntityOwnerInterface, EntityPublishedInterface, RevisionLogInterface {

    use EntityOwnerTrait, EntityPublishedTrait, RevisionLogEntityTrait;

    /**
     * {@inheritDoc}
     */
    public static function baseFieldDefinitions(EntityTypeInterface $entity_type) {
        // Get the field definitions for 'id' and 'uuid' from the parent.
        $fields = parent::baseFieldDefinitions($entity_type);
        $fields += static::revisionLogBaseFieldDefinitions($entity_type);
    }
}
```

OR

```
class Event extends RevisionableContentEntityBase implements EntityOwnerInterface, EntityPublishedInterface {

    use EntityOwnerTrait, EntityPublishedTrait;
```

# Tracking changes to entities

```
* entity_keys = {
*   "id" = "id",
*   "revision" = "vid", revision
*   "uuid" = "uuid",
*   "owner" = "author",
*   "published" = "published",
*   "label" = "title",
* },
* entity_keys = {
*   "id" = "id",
*   "revision" = "vid",
*   "uuid" = "uuid",
*   "owner" = "author",
*   "published" = "published",
*   "label" = "title",
* },
* revision_table = "event_revision",
* revision_metadata_keys = {
*   "revision_user" = "revision_author",
*   "revision_created" = "revision_created",
*   "revision_log_message" = "revision_log_message",
* },
* show_revision_ui = true,
```

```
* handlers = {
*   "access" = "Drupal\entity\EntityAccessControlHandler",
*   "permission_provider" = "Drupal\entity\EntityPermissionProvider",
*   "route_provider" = {
*     "default" = "Drupal\entity\Routing\DefaultHtmlRouteProvider",
*     "revision" = \Drupal\Core\Entity\Routing\RevisionHtmlRouteProvider::class,
*   },
*   links = {
*     "canonical" = "/event/{event}",
*     "add-form" = "/admin/content/events/add",
*     "edit-form" = "/admin/content/events/manage/{event}",
*     "delete-form" = "/admin/content/events/manage/{event}/delete",
*     "collection" = "/admin/content/events",
*     "version-history" = "/event/{event}/revisions",
*     "revision" = "/event/{event}/revisions/{event_revision}",
*     "revision-revert-form" = "/event/{event}/revisions/{event_revision}/revert",
*   },
* }
```

# Tracking changes to entities

<b>event</b>	/* The base table for event entities. */
<b>vid</b>	int(10) unsigned
<b>uuid</b>	varchar(128)
<b>author</b> /* The ID of the target entity. */	int(10) unsigned
<b>published</b>	tinyint(4)
<b>title</b>	varchar(255)
<b>date</b> /* The date value. */	varchar(20)
<b>description_value</b>	longtext
<b>description_format</b>	varchar(255)
<b>maximum</b>	int(11)
<b>id</b>	int(10) unsigned

<b>event_attendees</b>	/* Data storage for event field attendees. */
<b>bundle</b> /* The field instance bundle to which this row belongs, used whe...	varchar(128)
<b>revision_id</b> /* The entity revision id this data is attached to */)	unsigned
<b>attendees_target_id</b> /* The ID of the target entity. */	int(10) unsigned
<b>deleted</b> /* A boolean indicating whether this data item has been deleted */	tinyint(4)
<b>entity_id</b> /* The entity id this data is attached to */	int(10) unsigned
<b>langcode</b> /* The language code for this data item. */	varchar(32)
<b>delta</b> /* The sequence number for this data item, used for multi-value ... */	int(10) unsigned

<b>event_revision</b>	/* The revision table for event entities. */
<b>id</b>	int(10) unsigned
<b>revision_created</b>	int(11)
<b>revision_author</b> /* The ID of the target entity. */	int(10) unsigned
<b>revision_log_message</b>	longtext
<b>published</b>	tinyint(4)
<b>revision_default</b>	tinyint(4)
<b>vid</b>	int(10) unsigned

<b>event_revision_attendees</b>	/* Revision archive storage for event field attendees. */
<b>bundle</b> /* The field instance bundle to which this row belongs, used whe...	varchar(128)
<b>attendees_target_id</b> /* The ID of the target entity. */	int(10) unsigned
<b>deleted</b> /* A boolean indicating whether this data item has been deleted */	tinyint(4)
<b>entity_id</b> /* The entity id this data is attached to */	int(10) unsigned
<b>revision_id</b> /* The entity revision id this data is attached to */	int(10) unsigned
<b>langcode</b> /* The language code for this data item. */	varchar(32)
<b>delta</b> /* The sequence number for this data item, used for multi-value ... */	int(10) unsigned

# Tracking changes to entities

The screenshot displays two overlapping Drupal administrative pages. The top page is titled "Edit Leveraging the Drupal Core Entity API for your custom entity types" and shows a form for managing an event. The bottom page is titled "Revisions" and shows a history of changes made to the entity.

**Event Edit Form (Top):**

- Text area:** "Maximum number of attendees" set to 5.
- Attendees:** A list containing "Margaret Hopper (2)".
- Buttons:** "Add another item", "Published" (green switch), "Save" (blue button), and "Delete" (red button).
- Form Fields:** "Revision information" (New revision), "Create new revision" (green switch), "Revision log message" (Adding Attendees), and "Briefly describe the change".

**Revisions Page (Bottom):**

Revision	Operations
07/20/2023 - 10:41 by admin Adding attendees	Current revision <a href="#">Revert</a>
07/20/2023 - 10:40 by Anonymous (not verified)	<a href="#">Revert</a>

Both pages include standard Drupal navigation and toolbar elements.

# Config entity - Event Type

```

Event.php ×

<?php

namespace Drupal\event\Entity;

use Drupal\Core\Config\Entity\ConfigEntityBase;

/**
 * @ConfigEntityType(
 *   id = "event_type",
 *   label = @Translation("Event type"),
 *   label_collection = @Translation("Event types"),
 *   label_singular = @Translation("event type"),
 *   label_plural = @Translation("event types"),
 *   config_prefix = "type",
 *   config_export = {
 *     "id",
 *     "label",
 *   },
 *   entity_keys = {
 *     "id" = "id",
 *     "label" = "label",
 *   },
 * )
 */
class EventType extends ConfigEntityBase {

  /** The machine name of this event type. ...*/
  protected $id;

  /** The human-readable name of the node type. ...*/
  protected $label;

}

```

umami-drupal > web > modules > custom > event > config > schema > event.schema.yml

```

event.type.*:
  type: config_object
  mapping:
    id:
      type: string
      label: 'ID'
    label:
      type: label
      label: 'Label'

```

# Config entity - Event Type

```
=  
* handlers = {  
*   "access" = "Drupal\entity\EntityAccessControlHandler",  
*   "list_builder" = "Drupal\event\Controller\EventTypeListBuilder",  
*   "local_action_provider" = {  
*     "collection" = "Drupal\entity\Menu\EntityCollectionLocalActionProvider",  
*   },  
*   "permission_provider" = "Drupal\entity\EntityPermissionProvider",  
*   "route_provider" = {  
*     "default" = "Drupal\entity\Routing\DefaultHtmlRouteProvider",  
*   },  
* },  
* links = {  
*   "collection" = "/admin/structure/event-types",  
* },  
* admin_permission = "administer event_type",
```

The screenshot shows a code editor with a tab labeled 'EventTypeListBuilder.php'. The code is a PHP class definition:

```
<?php  
  
namespace Drupal\event\Controller;  
  
use Drupal\Core\Entity\EntityInterface;  
use Drupal\Core\Entity\EntityListBuilder;  
  
/**  
 * Defines an implementation to build a listing of event type entities.  
 */  
class EventTypeListBuilder extends EntityListBuilder {  
  
  /** {@inheritDoc} ... */  
  public function buildHeader() {  
    $header = [];  
    $header['label'] = $this->t( string: 'Label' );  
    return $header + parent::buildHeader();  
  }  
  
  /** {@inheritDoc} ... */  
  public function buildRow(EntityInterface $event) {  
    $row = [];  
    $row['label'] = $event->label();  
    return $row + parent::buildRow($event);  
  }  
}
```

# Config entity - Event Type

```
*     handlers = {
*         "access" = "Drupal\entity\EntityAccessControlHandler",
*         "list_builder" = "Drupal\event\Controller\EventTypeListBuilder",
*         "local_action_provider" = {
*             "collection" = "Drupal\entity\Menu\EntityCollectionLocalActionProvider",
*         },
*         "permission_provider" = "Drupal\entity\EntityPermissionProvider",
*         "route_provider" = {
*             "default" = "Drupal\entity\Routing\DefaultHtmlRouteProvider",
*         },
*         "form" = {
*             "add" = "Drupal\event\Form\EventTypeForm",
*             "edit" = "Drupal\event\Form\EventTypeForm",
*             "delete" = "Drupal\Core\Entity\EntityDeleteForm",
*         },
*     },
*     links = {
*         "collection" = "/admin/structure/event-types",
*         "add-form" = "/admin/structure/event-types/add",
*         "edit-form" = "/admin/structure/event-types/manage/{event_type}",
*         "delete-form" = "/admin/structure/event-types/manage/{event_type}/delete",
*     },
* }
```



The screenshot shows a code editor window with the title bar "Event Type Form.php". The main area contains PHP code for a form handler. The code includes imports for Drupal's Entity and Form modules, and annotations like `/** @InheritDoc */`. It defines a class `EventTypeForm` that extends `BundleEntityFormBase`. The `form` method sets up fields for the event type label and ID, including their types, titles, default values, and machine names. The code ends with a return statement for the \$form variable.

```
<?php

namespace Drupal\event\Form;

use Drupal\Core\Entity\BundleEntityFormBase;
use Drupal\Core\Entity\EntityTypeInterface;
use Drupal\Core\Form\FormStateInterface;

/**
 * Form handler for event type forms.
 */
class EventTypeForm extends BundleEntityFormBase {

  /** @InheritDoc */
  public function form(array $form, FormStateInterface $form_state) {
    $form = parent::form($form, $form_state);

    $event_type = $this->getEntity();

    $form['label'] = [
      '#type' => 'textfield',
      '#title' => $this->t( string: 'Label'),
      '#default_value' => $event_type->label(),
      '#required' => TRUE,
    ];

    $form['id'] = [
      '#type' => 'machine_name',
      '#title' => $this->t( string: 'ID'),
      '#maxLength' => EntityTypeInterface::BUNDLE_MAX_LENGTH,
      '#default_value' => $event_type->id(),
      '#machine_name' => [
        'exists' => [$event_type->getEntityType()->getClass(), 'load'],
      ],
      '#disabled' => !$event_type->isNew(),
    ];
  }

  return $form;
}
```

## Config entity - Event Type

```
/**  
 * {@inheritDoc}  
 */  
  
public function save(array $form, FormStateInterface $form_state) {  
    parent::save($form, $form_state);  
  
    $entity = $this->getEntity();  
    $entity_type = $entity->getEntityType();  
  
    $arguments = [  
        '@entity_type' => $entity_type->getSingularLabel(),  
        '%entity' => $entity->label(),  
        'link' => $entity->toLink($this->t('Edit'), rel: 'edit-form')->toString(),  
    ];  
  
    $this->logger($entity->getEntityType()->notice(message: 'The @entity_type %entity has been saved.', $arguments);  
    $this->messenger()->addMessage($this->t('The @entity_type %entity has been saved.', $arguments));  
  
    $form_state->setRedirectUrl($entity->toUrl(rel: 'collection'));  
}
```

# Config entity - Event Type

Back to site | Administration / Structure / Eve... Admin Toolbar quick search Shortcuts Go to admin

**Add event type ☆**

Label\*  
Webinar

Machine name: webinar [Edit]

Save

Back to site | Administration / Structure Admin Toolbar quick search

**Event types ☆**

Label	Operations
Onsite	Edit ▾
Webinar	Edit ▾

# Configuring bundles in the user interface

```
* Event.php x
\Drupal\event\Entity
59     * links = {
60     *   "canonical" = "/event/{event}",
61     *   "add-form" = "/admin/content/events/add/{event_type}",
62     *   "edit-form" = "/admin/content/events/manage/{event}",
63     *   "delete-form" = "/admin/content/events/manage/{event}/delete",
64     *   "collection" = "/admin/content/events",
65     *   "version-history" = "/event/{event}/revisions",
66     *   "revision" = "/event/{event}/revisions/{event_revision}",
67     *   "revision-revert-form" = "/event/{event}/revisions/{event_revision}/revert",
68     * },
69     * bundle_entity_type = "event_type",
70     * field_ui_base_route = "entity.event_type.edit_form",

```

```
c EventType.php x
\Drupal\event\Entity
38     * },
39     * links = {
40     *   "collection" = "/admin/structure/event-types",
41     *   "add-form" = "/admin/structure/event-types/add",
42     *   "edit-form" = "/admin/structure/event-types/manage/{event_type}",
43     *   "delete-form" = "/admin/structure/event-types/manage/{event_type}/delete",
44     * },
45     * bundle_of = "event",

```

# Configuring bundles in the user interface

The screenshot shows the Drupal administration interface for managing event types. The URL is <https://umami-drupal.ddev.site/admin/structure/event-types>. A banner at the top right states "This site is intended for demonstration purposes." A sidebar on the left contains various icons for different administrative tasks. The main content area is titled "Event types" and features a "Label" column and an "Operations" column. Two items are listed: "Onsite" and "Webinar". An "Edit" button is highlighted above the "Onsite" row, and a dropdown menu is open, listing options: "Manage fields", "Manage form display", "Manage display", "Translate", and "Delete". A blue button at the top right says "+ Add event type".

Label	Operations
Onsite	<a href="#">Edit</a>
Webinar	

+ Add event type

# Configuring bundles in the user interface

The screenshot shows a browser window with the URL <https://umami-drupal.ddev.site/en/admin/structure/event-types/manage/webinar/fields>. The page title is "Manage fields" with a star icon. On the left, there is a vertical sidebar with icons for Back to site, Administration, Structure, Event types, Edit W..., Admin Toolbar quick search, and a warning message: "This site is intended for demonstration purposes." A blue button at the top right says "+ Create a new field". Below the title, there are tabs: "Manage fields" (which is selected), "Manage form display", "Manage display", and "Translate event type". A table below lists fields with columns: Label, Machine name, Field type, and Operations. The message "No fields are present yet." is displayed.

Label	Machine name	Field type	Operations
No fields are present yet.			

# Configuring bundles in the user interface

The image consists of three side-by-side screenshots of the Drupal administrative interface, specifically the 'Event types' management screen.

**Screenshot 1: Adding a new field (General Type)**

The page title is 'Add field ☆'. A modal dialog is open, titled 'Add a new field'. The 'Select a field type -' dropdown is set to 'General'. The 'Link' option is selected and highlighted with a blue background. Other options visible include Boolean, Date, Email, Timestamp, Number (float), Number (integer), Number (decimal), Number (float), Number (integer), Reference (Content, File, Image, Media, Taxonomy term, User), and a 'File' icon.

**Screenshot 2: Adding a new field (Link Type)**

The page title is 'Add field ☆'. The 'Select a field type -' dropdown is now set to 'Link'. The 'Label \*' field contains 'URL'. Below it, the 'Machine name: field\_url [Edit]' is shown. A 'Save and continue' button is at the bottom.

**Screenshot 3: URL Field Settings**

The page title is 'URL ☆'. The 'Field settings' tab is active. A message states: 'These settings apply to the URL field once it has been created.' Under 'Allowed number of values', the 'Limited' radio button is selected, and the value '1' is entered in the input field. A 'Save field settings' button is at the bottom.

# Configuring bundles in the user interface

The screenshot shows the Drupal administrative interface for configuring a field within a content type. The URL path is `/en/admin/structure/event-types/manage/webinar/fields/event`. The page title is "URL settings for Webinar".

**Label:** URL

**Help text:** (Empty text area)

**Instructions:** Instructions to present to the user below this field on the editing form.  
Allowed HTML tags: <a> <b> <big> <code> <del> <em> <i> <ins> <pre> <q> <small> <span> <strong> <sub> <sup> <tt> <ol> <ul> <br> <p> <br> <img>  
This field supports tokens.

**Required field:** (checkbox checked)

**Users may translate this field:** (checkbox unchecked)

To configure translation for this field, [enable language support](#) for this type.

**Allowed link type:**

- Internal links only
- External links only
- Both internal and external links

**Allow link text:**

- Disabled
- Optional
- Required

**Set default value:** (checkbox unchecked)

Provide a pre-filled value for the editing form.

**Buttons:**

- Save settings** (blue button)
- Delete** (red button)

# Configuring bundles in the user interface

```
$fields['maximum'] = BaseFieldDefinition::create('type: integer')
->setLabel(t('Maximum number of attendees'))
->setSetting(setting_name: 'min', value: 1)
->setDisplayOptions(display_context: 'form', ['weight' => 23])
->setDisplayConfigurable(display_context: 'form', configurable: TRUE)
->setDisplayConfigurable(display_context: 'view', configurable: TRUE);
```

Manage form display ☆

Manage fields Manage form display **Manage display** Translate event type

Field	Widget	Format
Maximum number of attendees	Number field	Above
URL	Link	Default

**Disabled**

No field is hidden.

**Save**

Back to site | Administration / Structure / Event ... Admin Toolbar quick search This site is intended for demonstration purposes.

## Manage display ☆

Manage fields Manage form display **Manage display** Translate event type

Show row weights

Field	Label	Format
URL	Above	Link

**Disabled**

Maximum number of attendees	Above	Default
-----------------------------	-------	---------

**Layout options**

Use Layout Builder

**Save**

# Translatable entities

```

* entity_keys = {
*   "id" = "id",
*   "revision" = "vid",
*   bundle = "type",
*   langcode = "langcode",
*   uuid = "uuid",
*   owner = "author",
*   published = "published",
*   label = "title",
* },
-
* translatable = TRUE,
* data_table = "event_field_data",
* revision_data_table = "event_revision_field_data",

```

<b>event_field_data</b> /* The data table for event entities. */	
vid	int(10) unsigned
type /* The ID of the target entity. */	varchar(32)
author /* The ID of the target entity. */	int(10) unsigned
published	tinyint(4)
title	varchar(255)
date /* The date value. */	varchar(20)
description_value	longtext
description_format	varchar(255)
maximum	int(11)
default_langcode	tinyint(4)
revision_translation_affected	tinyint(4)
id	int(10) unsigned
langcode	varchar(12)

<b>event_revision_field_data</b> /* The revision data table for event entities. */	
id	int(10) unsigned
published	tinyint(4)
default_langcode	tinyint(4)
revision_translation_affected	tinyint(4)
vid	int(10) unsigned
langcode	varchar(12)

# Translatable and revisionable fields

```
$fields['title'] = BaseFieldDefinition::create( type: 'string')
    ->setLabel(t( string: 'Title'))
    ->setRequired( required: TRUE)
    ->setTranslatable( translatable: TRUE)
    ->setRevisionable( revisionable: TRUE)
    ->setDisplayOptions( display_context: 'form', [ 'weight' => 0]);
```

event_revision_field_data /* The revision data table for event entities. */	
• id	int(10) unsigned
• published	tinyint(4)
• default_langcode	tinyint(4)
• revision_translation_affected	tinyint(4)
• vid	int(10) unsigned
• langcode	varchar(12)

event_revision_field_data /* The revision data table for event entities. */	
• id	int(10) unsigned
• published	tinyint(4)
• title	varchar(255)
• default_langcode	tinyint(4)
• revision_translation_affected	tinyint(4)
• vid	int(10) unsigned
• langcode	varchar(12)

# Configuring translations per bundle

The screenshot shows a Drupal administrative interface. On the left is a sidebar with various icons representing different site configurations. The main content area is titled "Content language and translation" with a star icon. Below the title is a descriptive text: "Change language settings for *content types*, *taxonomy vocabularies*, *user profiles*, or any other supported element on your site. By default, language settings hide the language selector and the language is the site's default language." A section titled "Custom language settings" contains a list of checkboxes for different content bundles. The checked items are: Content, Content block, Event, Media, Taxonomy term, and URL alias.

Custom language settings

- Contact message
- Content
- Content block
- Content moderation state
- Custom menu link
- Event
- File
- Media
- Shortcut link
- Taxonomy term
- URL alias
- User

# Configuring translations per bundle

1x

Content language and translation 

Change language settings for *content types*, *taxonomy vocabularies*, *user profiles*, or any other supported element on your site. By default, language settings hide the language selector and the language is the site's default language.

Custom language settings	Translatable	Event type	Configuration
<input type="checkbox"/> Contact message	<input type="checkbox"/>	Onsite	<b>Default language</b> Site's default language (English) 
<input checked="" type="checkbox"/> Content	<input type="checkbox"/>	Webinar	<p>Explanation of the language options is found on the <a href="#">languages list page</a>.</p> <input checked="" type="checkbox"/> Show language selector on create and edit pages
<input checked="" type="checkbox"/> Content block	<input checked="" type="checkbox"/>	User ID	<b>Default language</b> Site's default language (English) 
<input type="checkbox"/> Content moderation state	<input type="checkbox"/>	Published	<p>Explanation of the language options is found on the <a href="#">languages list page</a>.</p> <input checked="" type="checkbox"/> Show language selector on create and edit pages <input type="checkbox"/> Hide non translatable fields on translation forms
<input type="checkbox"/> Custom menu link	<input type="checkbox"/>	Title	
<input checked="" type="checkbox"/> Event	<input checked="" type="checkbox"/>	Description	
<input type="checkbox"/> File	<input type="checkbox"/>		
<input checked="" type="checkbox"/> Media	<input checked="" type="checkbox"/>		
<input type="checkbox"/> Shortcut link	<input type="checkbox"/>		
<input checked="" type="checkbox"/> Taxonomy term	<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/> URL alias	<input checked="" type="checkbox"/>		
<input type="checkbox"/> User	<input type="checkbox"/>	URL	

# Translating entities

The screenshot shows the Drupal administrative interface for managing content. The URL in the browser is <https://umami-drupal.ddev.site/en/admin/content/events>. The page title is "Events". On the left, there is a vertical toolbar with various icons: a water drop (Content), a document (Pages), a grid (Blocks), a funnel (Search API), a grid with squares (Views), a person (Users), a gear (Configuration), and a question mark (Help).

The main content area displays a table with columns: Title, Date, Published, and Operations. One row is selected, showing the title "Leveraging the Drupal Core Entity API for your custom entity types", the date "07/21/23 10:45:00 am", and the status "Yes". In the "Operations" column, there is an "Edit" button with a dropdown arrow. A context menu is open from this button, containing the options "Translate" and "Delete".

# Translating entities

The screenshot shows a Drupal administrative interface for translating entities. On the left, there is a vertical sidebar with various icons: a water drop, a document, a gear, a magnifying glass, a square, a person, a clock, a gear, and a question mark. The main content area has a header bar with browser controls, a URL (https://umami-drupal.ddev.site/en/event/2/translations), and a zoom level (100%). A message in the top right corner reads: "This site is intended for demonstration purposes." Below the header, the page title is "Translations of Leveraging the Drupal Core Entity API for your custom entit...". There are two tabs: "Revisions" and "Translate", with "Translate" being active. A table lists translations for the English original language:

Language	Translation	Status	Operations
English (Original language)	Leveraging the Drupal Core Entity API for your custom entity types	Published	<button>Edit</button>
Spanish	n/a	Not translated	<button>Add</button>

# Translating entities

The screenshot shows a web browser displaying a Drupal administrative interface for translating entity types. The URL in the address bar is <https://umami-drupal.ddev.site/es/event/2/translations/add/en/es>. The page title is "Crear traducción *Español* de Leveraging the Drupal Core Entity API for you...". A note in the top right corner states "THIS SITE IS INTENDED FOR DEMONSTRATION PURPOSES".

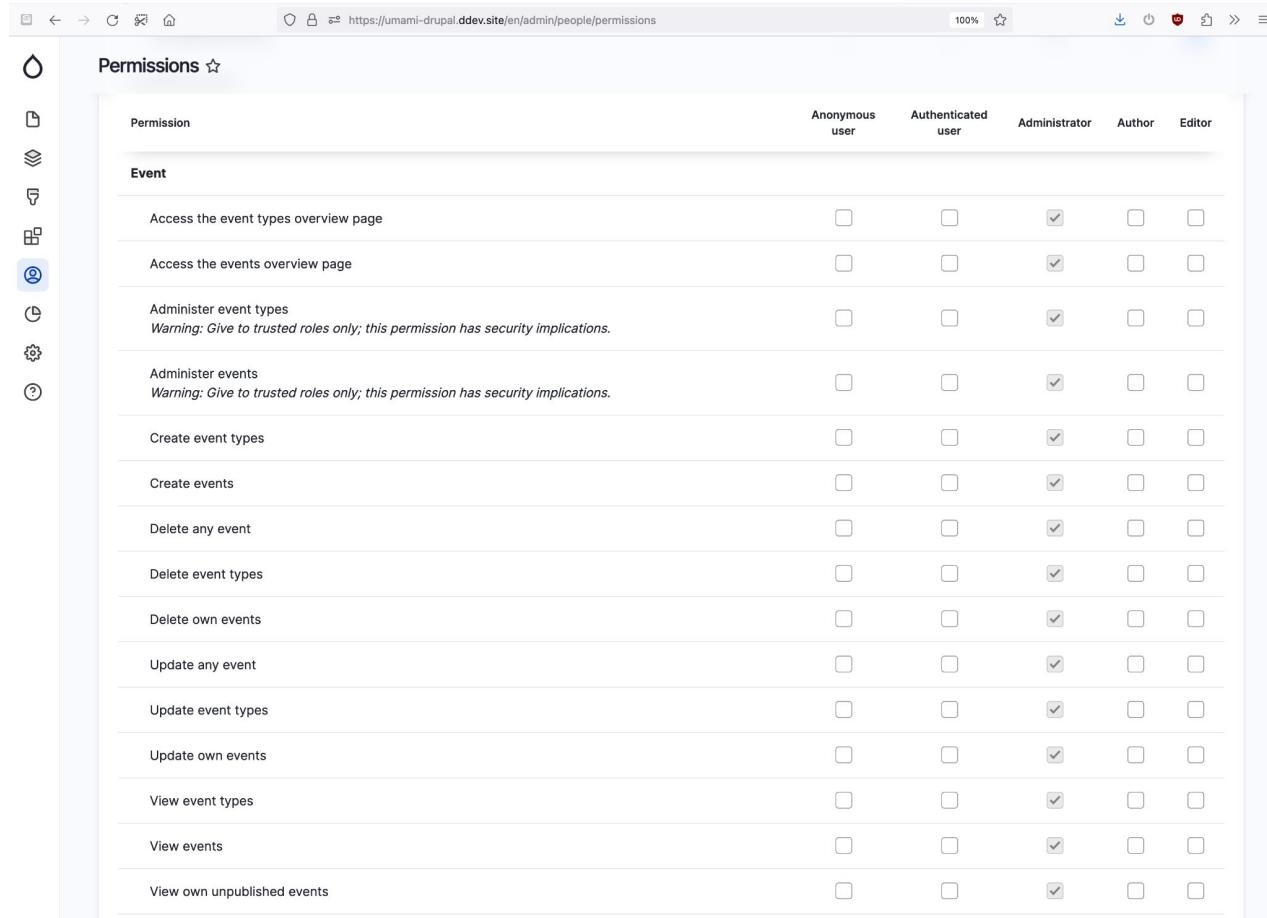
The form fields include:

- Título\***: "Leveraging the Drupal Core Entity API for your custom entity types"
- Fecha\***: "21/07/2023" and "13:45:00"
- Descripción**:  

```
<p>This session is a walk-through defining a custom entity type. Some of the major topics will be about:</p>
<p>
<ul>
<li>Bundles</li>
<li>Revision and translation support</li>
</ul>
...Continuing the form dialogue of the entity type...
```
- Formato de texto**: "HTML Restringido" (selected)
- Maximum number of attendees (todos los idiomas)**: "5"
- Attendees**: A search input field with a magnifying glass icon.
- Buttons**: "Anadir otro elemento" (Add another item) and a "Published" toggle switch.

# Access control and permissions

```
permission_granularity =  
  "entity_type"
```

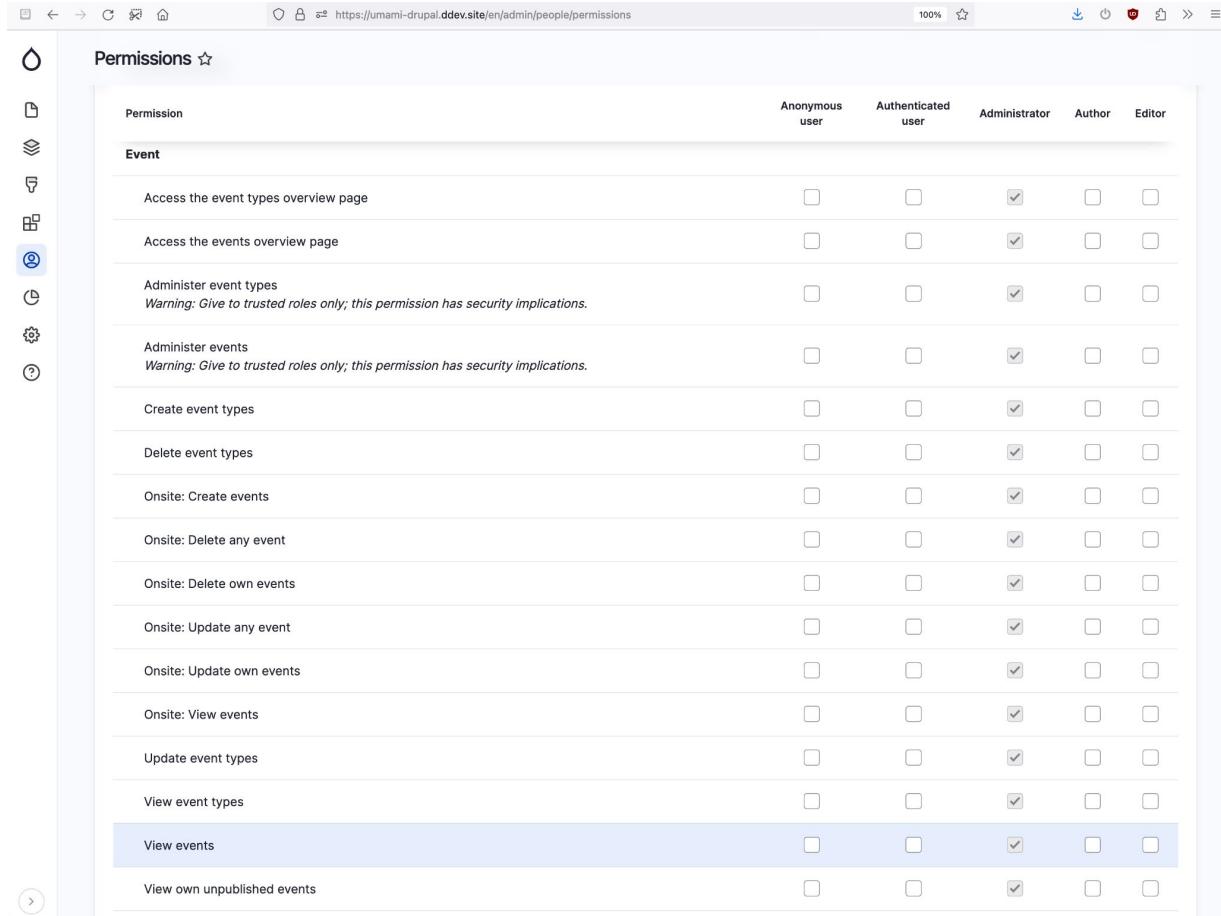


The screenshot shows the 'Permissions' page for the 'Event' entity type in a Drupal administrative interface. The URL is https://umami-drupal.dddev.site/en/admin/people/permissions. The page lists various permissions for four user roles: Anonymous user, Authenticated user, Administrator, Author, and Editor. Most permissions are checked for the Administrator role, while others like 'View own unpublished events' are checked for the Authenticated user role.

Permission	Anonymous user	Authenticated user	Administrator	Author	Editor
<strong>Event</strong>					
Access the event types overview page	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Access the events overview page	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Administer event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<i>Warning: Give to trusted roles only; this permission has security implications.</i>					
Administer events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<i>Warning: Give to trusted roles only; this permission has security implications.</i>					
Create event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Create events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delete any event	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delete event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delete own events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Update any event	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Update event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Update own events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
View event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
View events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
View own unpublished events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

# Access control and permissions

permission\_granularity =  
“bundle”



The screenshot shows the 'Permissions' page for the 'Event' bundle in the Drupal admin interface. The URL is https://umami-drupal.ddev.site/en/admin/people/permissions. The page lists various permissions under the 'Event' category, each with a warning message: 'Warning: Give to trusted roles only; this permission has security implications.' The permissions listed are:

Permission	Anonymous user	Authenticated user	Administrator	Author	Editor
Access the event types overview page	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Access the events overview page	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Administer event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Administer events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Create event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delete event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Onsite: Create events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Onsite: Delete any event	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Onsite: Delete own events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Onsite: Update any event	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Onsite: Update own events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Onsite: View events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Update event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
View event types	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
View events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
View own unpublished events	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## Entity queries

```
\Drupal::entityTypeManager()
  ->getStorage( entity_type_id: 'event' )
  ->getQuery()
  ->accessCheck( access_check: FALSE )
  ->condition( field: 'type', value: 'webinar' )
  ->condition( field: 'date', value: '2023-07-21%', operator: 'LIKE', langcode: 'es' )
  ->execute();
// Returns [2 => "2"]
```

```
\Drupal::entityTypeManager()
  ->getStorage( entity_type_id: 'event' )
  ->loadMultiple($entity_ids);
```

# Entity type annotation

```
/***
 * @ContentEntityType(
 *   id = "event",
 *   label = @Translation("Event"),
 *   label_collection = @Translation("Events"),
 *   label_singular = @Translation("event"),
 *   label_plural = @Translation("events"),
 *   base_table = "event",
 *   entity_keys = {
 *     "id" = "id",
 *     "revision" = "vid",
 *     "bundle" = "type",
 *     "langcode" = "langcode",
 *     "uuid" = "uuid",
 *     "owner" = "author",
 *     "published" = "published",
 *     "label" = "title",
 *   },
 *   revision_table = "event_revision",
 *   revision_metadata_keys = {
 *     "revision_user" = "revision_author",
 *     "revision_created" = "revision_created",
 *     "revision_log_message" = "revision_log_message",
 *   },
 *   show_revision_ui = true,
 *   translatable = TRUE,
 *   data_table = "event_field_data",
 *   revision_data_table = "event_revision_field_data",
 * )
 */

```

```
* handlers = {
*   "access" = "Drupal\entity\EntityAccessControlHandler",
*   "permission_provider" = "Drupal\entity\EntityPermissionProvider",
*   "route_provider" = {
*     "default" = "Drupal\entity\Routing\DefaultHtmlRouteProvider",
*     "revision" = \Drupal\Core\Entity\Routing\RevisionHtmlRouteProvider::class,
*   },
*   "form" = {
*     "add" = "Drupal\Core\Entity\ContentEntityForm",
*     "edit" = "Drupal\Core\Entity\ContentEntityForm",
*     "delete" = "Drupal\Core\Entity\ContentEntityDeleteForm",
*   },
*   "list_builder" = "Drupal\event\Controller\EventListBuilder",
*   "local_action_provider" = {
*     "collection" = "Drupal\entity\Menu\EntityCollectionLocalActionProvider",
*   },
*   "links" = {
*     "canonical" = "/event/{event}",
*     "add-form" = "/admin/content/events/add/{event_type}",
*     "edit-form" = "/admin/content/events/manage/{event}",
*     "delete-form" = "/admin/content/events/manage/{event}/delete",
*     "collection" = "/admin/content/events",
*     "version-history" = "/event/{event}/revisions",
*     "revision" = "/event/{event}/revisions/{event_revision}",
*     "revision-revert-form" = "/event/{event}/revisions/{event_revision}/revert",
*   },
*   bundle_entity_type = "event_type",
*   admin_permission = "administer event",
*   permission_granularity = "bundle",
*   field_ui_base_route = "entity.event_type.edit_form",
* )
*/

```

# Thank you for joining us!

Stay connected with Drupal Austria for more exciting events, news, and updates!

Also visit our website:

<https://ddd23.drupalcamp.at/>



And don't forget to visit our sponsors' stands.

Their support helps make events like this possible!



amazee.io  
part of Mirantis

