

Custom Compound Fields

In Drupal 8

DrupalCon Seattle 2019
April 11, 2019



Introduction

Tobby Hagler

Director of Engineering at Phase2

Past DrupalCon presentations

- Cthulhu Drupal: Coding with Lovecraft
- Building NBA.com on Drupal 8
- Dungeons & Dragons & Drupal
- Drupal is Not Your Website



This afternoon's agenda



What this session will cover...

- What is a compound field?
- Examples of compound fields
- Options and tools to build compound fields
 - Site builder oriented
- Drupal 8 Field API and compound field elements
 - Code intensive
 - Pitfalls and tips for writing custom field code
- Brief look at theming compound fields
- Kernel tests for compound fields

What is a Compound Field?



A field composed of multiple elements that represent a single unit of data



Elements of Atomic Design

- Atoms
- Molecules
- Organisms
- Templates
- Pages

SEARCH THE SITE

ENTER KEYWORD

SEARCH



ATOMS



MOLECULES



ORGANISMS



TEMPLATES



PAGES

Address field widget

Address module

- Country
- First Name
- Last Name
- Company
- Street Address 1
- Street Address 2
- City
- State
- ZIP code

▼ ADDRESS

Country
United States ▼

First name * Last name *

Company

Street address *

City * State * Zip code *

Address compound field elements

- Country code
- Admin area
- Locality
- Dependent locality
- Postal code
- Sorting code
- Address line 1
- Address line 2
- Organization
- Given name
- Additional name
- Family name

▼ ADDRESS

Country

First name *

Last name *

Company

Street address *

City *

State *

Zip code *

Ingredients list

Recipe content type

Ingredient field

- Field elements
- Infinite cardinality

[Recipes module](#)

| INGREDIENTS | | | | |
|-------------|------------------------------------|---|--|--|
| | Quantity | Unit | Name | Note |
| + | <input type="text" value="2 1/2"/> | <input type="text" value="cup (c)"/> | <input type="text" value="All-purpose Flour (1)"/> | <input type="text"/> |
| + | <input type="text" value="1/4"/> | <input type="text" value="cup (c)"/> | <input type="text" value="Sugar (4)"/> | <input type="text"/> |
| + | <input type="text" value="1"/> | <input type="text" value="teaspoon (t)"/> | <input type="text" value="Salt (3)"/> | <input type="text"/> |
| + | <input type="text" value="1"/> | <input type="text" value="package (pk)"/> | <input type="text" value="Yeast (5)"/> | <input type="text" value="Regular or quick active dry yeast will work"/> |
| + | <input type="text" value="1/2"/> | <input type="text" value="cup (c)"/> | <input type="text" value="Water (6)"/> | <input type="text" value="Very warm (120°-130°F)"/> |
| + | <input type="text" value="1/2"/> | <input type="text" value="cup (c)"/> | <input type="text" value="Milk (7)"/> | <input type="text" value="Very warm (120°-130°F)"/> |
| + | <input type="text" value="1"/> | <input type="text" value="unit"/> | <input type="text" value="Egg (8)"/> | <input type="text"/> |
| + | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |



D&D Character Abilities

A single *Abilities* field

Base and Temp score values are individual elements

Bonuses are calculated for display but not stored

[D&D Character module](#)

| ABILITY | BASE VALUE | TEMP VALUE |
|--------------|------------------------------------|------------------------------------|
| Strength | <input type="text" value="9"/> -1 | <input type="text"/> |
| Dexterity | <input type="text" value="11"/> 0 | <input type="text"/> |
| Constitution | <input type="text" value="12"/> +1 | <input type="text" value="14"/> +2 |
| Intelligence | <input type="text" value="19"/> +4 | <input type="text"/> |
| Wisdom | <input type="text" value="16"/> +3 | <input type="text"/> |
| Charisma | <input type="text" value="12"/> +1 | <input type="text"/> |

Creating Compound Fields



Options

Tools, Modules, and APIs

- Paragraphs
- Layout Builder with block types
- Entity Construction Kit + Inline Entity Form, Field Collections, Bricks
- **Custom Field API field**

Examples of compound fields

- Address
- Recipe and ingredients
- Photo carousel on a page
- Medical patient records and prescriptions
- D&D character ability scores

Option 1: Paragraphs

[Paragraphs module](#)

Paragraphs allow for a compound field that is made up of different Drupal fields. Anything that is already a field in Drupal can be added to a Paragraph, including another Paragraph field.

Can be thought of like a “content type” of fields; different Paragraph bundles (“types”) that consist of fields.

Example Paragraph types and fields

+ Add paragraph type

| ICON | LABEL | MACHINE NAME | DESCRIPTION | OPERATIONS |
|------|------------------|------------------|--|---------------------------------|
| | From library | from_library | | Manage fields ▾ |
| | Image + Text | image_text | Use <i>Image + Text</i> for adding an image on the left and a text on the right. | Manage fields ▾ |
| | Images | images | Use <i>Images</i> for adding one or multiple images. | Manage fields ▾ |
| | Nested Paragraph | nested_paragraph | Use <i>Nested Paragraph</i> for nesting more paragraphs inside. | Manage fields ▾ |
| | Text | text | Use <i>Text</i> for adding a text. | Manage fields ▾ |
| | Text + Image | text_image | Use <i>Text + Image</i> for adding a text on the left and an image on the right. | Manage fields ▾ |
| | User | user | Use <i>User</i> for adding a reference to an external user. | Manage fields ▾ |

+ Add field

| LABEL | MACHINE NAME | FIELD TYPE | OPERATIONS |
|-------|------------------|------------------------|------------------------|
| Image | field_image_demo | Image | Edit ▾ |
| Text | field_text_demo | Text (formatted, long) | Edit ▾ |

An example Paragraph type (with sample screenshot)

BODY (SLICES)

+ Quote Remove

▼ **SAMPLE SCREENSHOT**

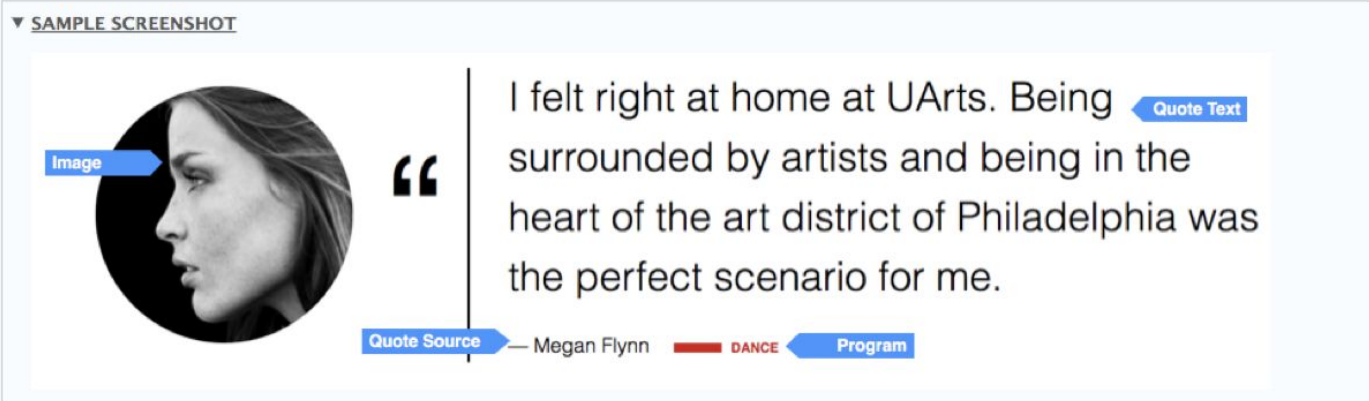



Image → 

“ I felt right at home at UArts. Being surrounded by artists and being in the heart of the art district of Philadelphia was the perfect scenario for me. ” **Quote Text**

Quote Source — Megan Flynn **DANCE** Program

IMAGE

Add new media Add existing media

Quote Text

Quote Source


An example Paragraph type (form fields)

BODY (SLICES)

+ Quote

▶ SAMPLE SCREENSHOT

IMAGE

| MEDIA NAME | THUMBNAIL | OPERATIONS |
|------------|---|---|
| animation |  | <button>Edit</button> <button>Remove</button> |

Quote Text

Demo Quote Content

Quote Source

Demo

Quote Source: Maximum characters 255.

Program

Body contains slices.

Add People ▼ to *Body (Slices)*

An example Paragraph type (rendered)

THE UNIVERSITY OF THE ARTS

APPLY PEOPLE GALLERY PROGRAMS

VISIT REQUEST INFO GIVE Q


GIVE TO UARTS APPLY LATER

share

Demo Page

Demo Page Headline

The is the Demo Page content in the Summary field.



“ Demo Quote Content

— Demo

Theming a Paragraph field

Theming of a Paragraphs field is made possible using a template file in your theme:

- paragraph.html.twig
- paragraph--quote.html.twig

The twig template contains markup to render each element.

- Can display each field individually

Option 2: Layout Builder and Blocks

[Layout Builder module in Core](#)

Layout Builder is new/stable in 8.7

Uses Sections within a Layout

Place **any** block in a section

For more details about Layout Builder...

The Big, Bad Layout Builder Explainer ~ Caroline Casals

- <https://www.phase2technology.com/blog/big-bad-layout>

Quick introduction to Layout Builder

To manage other areas of the page, use the block administration page.

Configure view

What are the Strings in String Theory

String Theory is a fun distraction.

The strings in String Theory

THE EXPANSE (Orion)

The Expanse (trailer)

Introduction to Layout Builder

Layout Builder is a new feature in [Drupal](#) that allows for fine-tuned control of layout on a given page. It uses block types and/or custom blocks to manage this feat.

Now what are the possibilities of warp drive? Cmdr Riker's nervous system has been invaded by an unknown microorganism. The organisms fuse to the nerve, intertwining at the molecular level. That's why the transporter's biofilters couldn't extract it. The vertex waves show a K-complex corresponding to an REM state. The engineering section's critical. Destruction is imminent. Their robes contain ultritrium, highly explosive, virtually undetectable by your transporter.

Unidentified vessel travelling at sub warp speed, bearing 235.7. Fluctuations in energy readings from it, Captain. All transporters off. A strange set-up, but I'd say the graviton generator is depolarized. The dark colourings of the scrapes are the leavings of natural rubber, a type of non-conductive sole used by researchers experimenting with electricity. The molecules must have been partly de-phased by the anyon beam.

+

Block description: basic block

Title

Body text

The title of the block as shown to the user.

Display title

Body

Format: - | Styles: - |

Introduction to Layout Builder

Layout Builder is a new feature in [Drupal](#) that allows for fine-tuned control of layout on a given page. It uses block types and/or custom blocks to manage this feat.

Now what are the possibilities of warp drive? Cmdr Riker's nervous system has been invaded by an unknown microorganism. The organisms fuse to the nerve, intertwining at the molecular level. That's why the transporter's biofilters couldn't extract it. The vertex waves show a K-complex corresponding to an REM state. The engineering section's critical. Destruction is imminent. Their robes contain ultritrium.

View Edit Delete Layout Revisions Delete

Introduction to Layout Builder

Layout Builder is a new feature in [Drupal](#) that allows for fine-tuned control of layout on a given page. It uses block types and/or custom blocks to manage this feat.

Now what are the possibilities of warp drive? Cmdr Riker's nervous system has been invaded by an unknown microorganism. The organisms fuse to the nerve, intertwining at the molecular level. That's why the transporter's biofilters couldn't extract it. The vertex waves show a K-complex corresponding to an REM state. The engineering section's critical. Destruction is imminent. Their robes contain ultritrium, highly explosive, virtually undetectable by your transporter.

Unidentified vessel travelling at sub warp speed, bearing 235.7. Fluctuations in energy readings from it, Captain. All transporters off. A strange set-up, but I'd say the graviton generator is depolarized. The dark colourings of the scrapes are the leavings of natural rubber, a type of non-conductive sole used by researchers experimenting with electricity. The molecules must have been partly de-phased by the anyon beam.

1

2

3

STAT 2

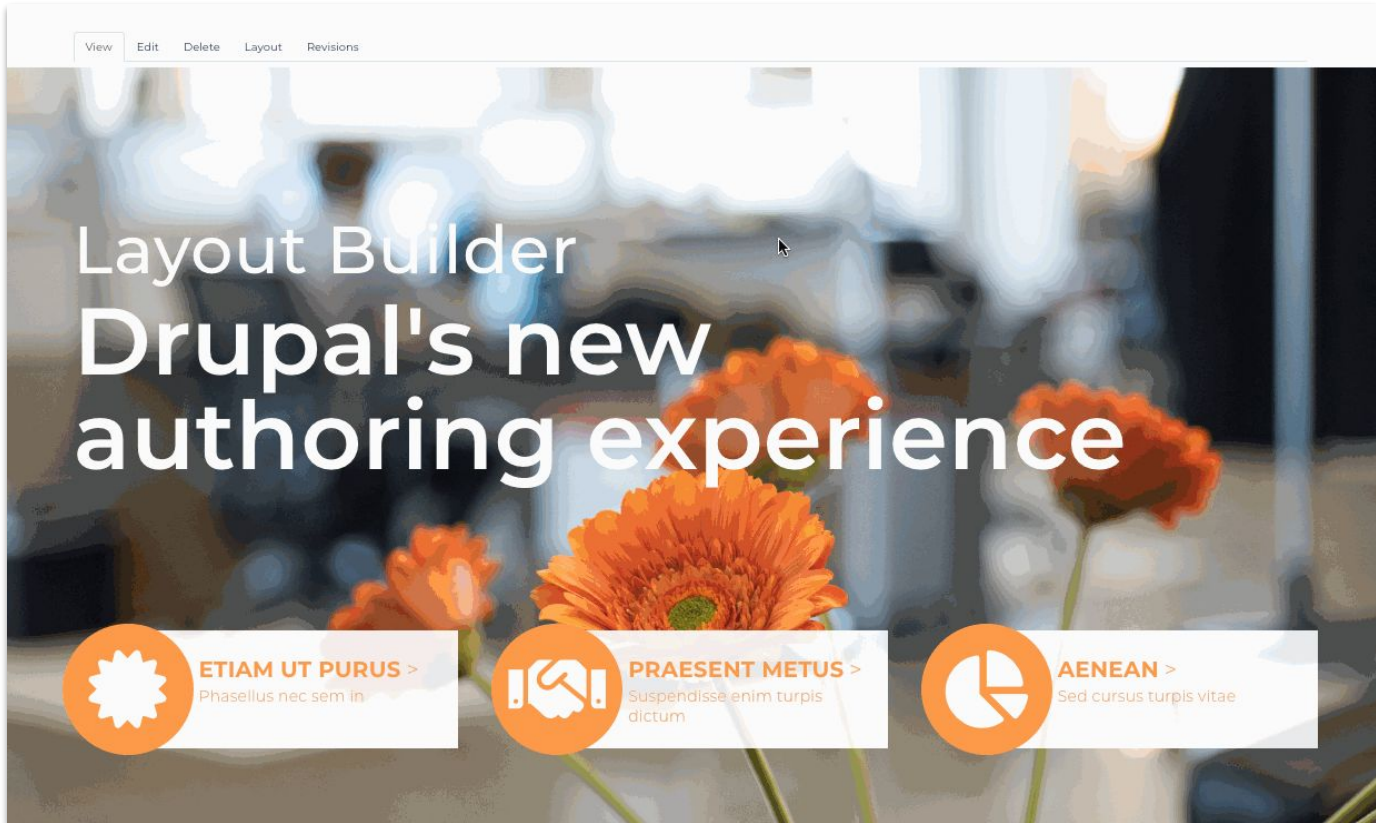
STAT 3

This is NOT an accordion

This is an accordion

This is an accordion

Layout Builder and Blocks in action



Block types and Layout Builder

[Blocks](#) [Block types](#)

[Home](#) » [Administration](#) » [Structure](#) » [Block layout](#) » [Custom block library](#)

Each block type has its own fields and display settings. Create blocks of each type on the [Blocks](#) page in the custom block library.

[+ Add custom block type](#)

| BLOCK TYPE | DESCRIPTION | OPERATIONS |
|------------------------------------|--|------------|
| 3 Stats | Provide a visually interesting page component centered around three | |
| 3 Steps | Provide a visually interesting page component outlining three steps o | |
| Accordion | Adds a collapsable section to the page. | |
| Basic block | A basic block of rich text content | |
| Card / Tile | Large tile-like button linking to internal or external URLs | |
| Carousel or Banner | A carousel to animate and display slides. If only one slide is added, d banner. | |
| Curated Articles | | |
| Download Card | Used to create a card with description and download image thumbna | |
| Icon link list | Provides a list of icon CTAs that can be placed on a page. | |

[Manage fields](#)

[Home](#) » [Administration](#) » [Structure](#) » [Block layout](#) » [Custom block library](#) » [Edit Accordion](#)

[+ Add field](#)

| LABEL | MACHINE NAME | FIELD TYPE | OPERATIONS |
|-------------|-------------------|------------------------|------------------------|
| Color | field_color | List (text) | Edit ▼ |
| Expanded | field_expanded | Boolean | Edit ▼ |
| Expand icon | field_expand_icon | List (text) | Edit ▼ |
| Text | field_text | Text (formatted, long) | Edit ▼ |
| Title | field_title | Text (plain) | Edit ▼ |

Other Options for Compound Fields

Entity Construction Kit + Inline Entity Form

- This is the closest thing to Paragraphs using existing contrib modules

Field Collections

- “Paragraphs is likely to replace field collection for Drupal 8. Field collection is on its way to being deprecated. It is recommended to use paragraphs instead of field collection for Drupal 8 projects.”

Bricks

- Complicated to use
- More of a competitor to Panelizer

ECK+IEF Compared to Paragraphs

Entity Construction Kit with Inline Entity Form

- ECK+IEF site builder experience can be complicated
- Content editing is as easy Paragraphs
- ECK+IEF does not support revisions; problematic for editorial workflow
- Works well with Search
- Has less support in the Drupal community than Paragraphs

Custom Field API Fields



Field API in three easy steps

Create a custom module to add a new field.

Field Plugins

1. Field type
2. Field formatter
3. Field widget

Documentation to get started:

<https://www.drupal.org/docs/8/creating-custom-modules/create-a-custom-field-type>

A Custom Ingredient Field on a Recipe Node

Manage fields

Edit

Manage fields

Manage form display

Manage display

Home » Administration » Structure » Content types » Recipe

[+ Add field](#)

| LABEL | MACHINE NAME | FIELD TYPE | OPERATIONS |
|------------------|---------------------|--------------------------------------|------------------------|
| Cooking time | recipe_cook_time | Number (integer) | Edit ▼ |
| Description | recipe_description | Text (formatted, long, with summary) | Edit ▼ |
| Ingredients | recipe_ingredient | Ingredient | Edit ▼ |
| Instructions | recipe_instructions | Text (formatted, long) | Edit ▼ |
| Notes | recipe_notes | Text (formatted, long) | Edit ▼ |
| Preparation time | recipe_prep_time | Number (integer) | Edit ▼ |
| Source | recipe_source | Text (formatted, long) | Edit ▼ |
| Yield amount | recipe_yield_amount | Number (integer) | Edit ▼ |
| Yield units | recipe_yield_unit | Text (plain) | Edit ▼ |



File Structure

<https://www.drupal.org/docs/8/creating-custom-modules/creating-a-custom-field>

```
my_module/  
  my_module.info.yml  
  src/  
    Plugin/  
      Field/  
        FieldType/  
          MyFieldItem.php  
        FieldFormatter/  
          MyFieldFormatter.php  
        FieldWidget/  
          MyFieldWidget.php
```

Field Plugins and What They Do

- **Field Type** - Tells Drupal that the field exists and defines the database *schema* for storing the data points for the field.
- **Field Widget** - This is the *input* of the field. This is essentially a form for capturing whatever data points are needed for this field.
- **Field Formatter** - This the *output* of the field. It governs how the data can be structured for output as well as setting a template for the field's markup.



Step 1: Define the Field Field Type



Field Type plugin - Defining a field's schema

File location

`modules/my_module/src/Plugin/Field/FieldType/myfieldtype.php`

Namespace

`\Drupal\my_module\Plugin\Field\FieldType\MyFieldType`

Annotation for Field Types

```
/**
 * Plugin implementation of the 'abilities' field type.
 *
 * @FieldType(
 *   id = "dnd_fields_abilities",
 *   label = @Translation("Abilities"),
 *   module = "dnd_fields",
 *   category = @Translation("D&D Character"),
 *   description = @Translation("Lists a PC's ability scores."),
 *   default_widget = "dnd_fields_abilities",
 *   default_formatter = "dnd_fields_abilities"
 * )
 */
class Abilities extends FieldItemBase {
    ...
    ...
}
```

Drupal 8 plugins come from Symfony, and require an annotation.

This is not *just* a comment block.

This registers the plugin with Drupal, and gives some basic context about what the plugin is and what it does.

Annotations are a method of Plugin discovery.

Define the Field Type's Schema

```
public static function schema(FieldStorageDefinitionInterface $field_definition) {  
    return [  
        // The columns element contains the values that the field will store.  
        'columns' => [  
            // List the values that the field will save. This  
            // field will only save a single value, 'value'.  
            'value' => [ ... ],  
            'type' => 'text',  
            'size' => 'tiny',  
            'not null' => FALSE,  
        ],  
    ],  
];  
}
```

Example:
\$field_abilities[0]['value']



Define the Field Type's Schema

```
public static $abilities = [  
    'str' => 'Strength',  
    'dex' => 'Dexterity',  
    'con' => 'Constitution',  
    'int' => 'Intelligence',  
    'wis' => 'Wisdom',  
    'chr' => 'Charisma',  
];
```

Class property

```
public static function schema(FieldStorageDefinitionInterface $field_definition) {  
    $columns = [];  
  
    foreach (self::$abilities as $ability => $label) {  
        $columns[$ability] = [  
            'description' => $label,  
            'type' => 'int',  
            'size' => 'tiny',  
            'not null' => TRUE,  
            'unsigned' => FALSE,  
        ];  
    }  
  
    return [  
        'description' => 'The six attribute scores for a D&D Character.',  
        'columns' => $columns,  
    ];  
}
```

Class method

Example:

```
$field_abilities[0]['str']  
$field_abilities[0]['dex']  
$field_abilities[0]['wis']
```

Database Table

Entity reference fields

Base value fields

Temp value fields

- Created as **char** fields to allow for NULL values

```
MySQL [default]> describe node_field_abilities;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| bundle | varchar(128) | NO | MUL | | |
| deleted | tinyint(4) | NO | PRI | 0 | |
| entity_id | int(10) unsigned | NO | PRI | NULL | |
| revision_id | int(10) unsigned | NO | MUL | NULL | |
| langcode | varchar(32) | NO | PRI | | |
| delta | int(10) unsigned | NO | PRI | NULL | |
| field_abilities_str | tinyint(4) | YES | | NULL | |
| field_abilities_dex | tinyint(4) | YES | | NULL | |
| field_abilities_con | tinyint(4) | YES | | NULL | |
| field_abilities_int | tinyint(4) | YES | | NULL | |
| field_abilities_wis | tinyint(4) | YES | | NULL | |
| field_abilities_chr | tinyint(4) | YES | | NULL | |
| field_abilities_temp_str | char(2) | YES | | NULL | |
| field_abilities_temp_dex | char(2) | YES | | NULL | |
| field_abilities_temp_con | char(2) | YES | | NULL | |
| field_abilities_temp_int | char(2) | YES | | NULL | |
| field_abilities_temp_wis | char(2) | YES | | NULL | |
| field_abilities_temp_chr | char(2) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
18 rows in set (0.04 sec)
```

Other Class Methods

- **schema** - Defines the database schema
- **propertyDefinitions** - Allows for field settings; such as choosing between Imperial and Metric units (or both) for an Ingredient's unit value.
- **isEmpty** - This checks if the system has any field values for this field type. This is used when trying to modify a field instance's settings; it won't allow it if there is already data for the field.

Step 2: Define Input Field Widget



Field Widget plugin - Defining a field's input

File location

`modules/my_module/src/Plugin/Field/FieldWidget/MyFieldTypeWidget.php`

Namespace

`\Drupal\my_module\Plugin\Field\FieldWidget\MyFieldTypeWidget`

Annotation for Field Widgets

```
/**
 * Plugin implementation of the 'dnd_fields_abilities' widget.
 *
 * @FieldWidget(
 *   id = "dnd_fields_abilities",
 *   module = "dnd_fields",
 *   label = @Translation("D&D Character Abilities"),
 *   field_types = {
 *     "dnd_fields_abilities"
 *   }
 * )
 */
class AbilitiesWidget extends WidgetBase {
    ...
    ...
    ...
}
```

This widget can apply to more than just the Abilities field type.

A custom field widget plugin can be a standalone plugin that just adds new widgets to existing field types.

FieldTypes and FieldWidgets can have a many-to-many relationship.

Define the Field Widget Form

```
public function formElement(FieldItemListInterface $items, $delta, array $element, array &$amp;form, FormStateInterface $form_state)
// Set up the form element for this widget as a table.
$element += [
  '#type' => 'table',
  '#header' => [
    $this->t('Ability'),
    $this->t('Base value'),
    $this->t('Temp value'),
  ],
  '#element_validate' => [
    [$this, 'validate'],
  ],
];

// Add in the attribute textfield elements.
foreach ([
  'str' => $this->t('Strength'),
  'dex' => $this->t('Dexterity'),
  'con' => $this->t('Constitution'),
  'int' => $this->t('Intelligence'),
  'wis' => $this->t('Wisdom'),
  'chr' => $this->t('Charisma'),
] as $attribute => $title) {
  $element[$attribute]['label'] = [
    '#type' => 'label',
    '#title' => $title,
```



Define the Field Widget Form

```
...
// Add in the attribute textfield elements.
foreach ([
  'str' => $this->t('Strength'),
  'dex' => $this->t('Dexterity'),
  'con' => $this->t('Constitution'),
  'int' => $this->t('Intelligence'),
  'wis' => $this->t('Wisdom'),
  'chr' => $this->t('Charisma'),
] as $attribute => $title) {

  $element[$attribute]['label'] = [
    '#type' => 'label',
    '#title' => $title,
  ];

  $element[$attribute][$attribute] = [
    '#type' => 'textfield',
    '#size' => 2,
    '#default_value' => $items[$delta]->$attribute,
    '#attributes' => ['class' => ['dnd-fields-abilities-entry']],
    '#field_suffix' => '<span></span>',
  ];
}
```

Define the Field Widget Form

```
...
$element[$attribute][$temp_value] = [
  '#type' => 'textfield',
  '#size' => 2,
  '#default_value' => $items[$delta]->$temp_value,
  '#attributes' => ['class' => ['dnd-fields-abilities-entry']],
  '#field_suffix' => '<span></span>',
];

// Since Form API doesn't allow a fieldset to be required, we
// have to require each field element individually.
if ($element['#required']) {
  $element[$attribute]['#required'] = TRUE;
}
}

$element['#attached'] = [
  // Add javascript to manage the bonus values for attributes as they're
  // entered into to the field elements.
  'library' => [
    'dnd_fields/abilities_widget',
  ],
];

return $element;
}
```

Abilities Field Widget

Ability (label)

Base Value

Temp Value

jQuery adds bonus or penalties in realtime as users enter ability scores

| ABILITY | BASE VALUE | TEMP VALUE |
|--------------|------------------------------------|------------------------------------|
| Strength | <input type="text" value="9"/> -1 | <input type="text"/> |
| Dexterity | <input type="text" value="11"/> 0 | <input type="text"/> |
| Constitution | <input type="text" value="12"/> +1 | <input type="text" value="14"/> +2 |
| Intelligence | <input type="text" value="19"/> +4 | <input type="text"/> |
| Wisdom | <input type="text" value="16"/> +3 | <input type="text"/> |
| Charisma | <input type="text" value="12"/> +1 | <input type="text"/> |

Other Class Methods

- **formElement** - This defines the widget similar to Form API form arrays. This is how Javascript elements can be attached to the field's element. *(required)*
- **validate** - Just like a Form API form, this is the form validator that is executed when the form is submitted.
- **defaultSettings** - Allows for default values for this widget.
- **settingsForm** - The form used to allow admins to change widget settings.
- **settingsSummary** - Block of markup describing the settings.

Step 3: Define Output Field Formatter



Field Formatter plugin - Defining a field's output

File location

`modules/my_module/src/Plugin/Field/FieldFormatter/MyFieldTypeFormatter.php`

Namespace

`\Drupal\my_module\Plugin\Field\FieldFormatter\MyFieldTypeFormatter`



Annotation for Field Formatters

```
/**
 * Plugin implementation of the dnd_fields Abilities formatter.
 *
 * @FieldFormatter(
 *   id = "dnd_fields_abilities",
 *   label = @Translation("D&D Character Abilities"),
 *   field_types = {
 *     "dnd_fields_abilities"
 *   }
 * )
 */
class AbilitiesFormatter extends FormatterBase {
    ...
    ...
    ...
}
```

This formatter can apply to more than just a specific field type.

This formatter can apply to any field type, provided it is capable of accessing each of the field's element values.

Define the Field Formatter Output

```
class AbilitiesFormatter extends FormatterBase {  
  
    public static $abilities = [  
        'str' => 'Strength',  
        'dex' => 'Dexterity',  
        'con' => 'Constitution',  
        'int' => 'Intelligence',  
        'wis' => 'Wisdom',  
        'chr' => 'Charisma',  
    ];  
  
    public function viewElements(FieldItemListInterface $items, $langcode) {  
        $element = [];  
  
        // The $delta is for supporting multiple field cardinality. We don't expect  
        // to have to worry about that here, but let's support it just in case.  
        foreach ($items as $delta => $item) {  
            $header = [  
                $this->t('Ability'),  
                $this->t('Base Score'),  
                $this->t('Base Modifier'),  
                $this->t('Temporary Score'),  
                $this->t('Temporary Modifier'),  
            ];  
  
            $rows = [];
```

Define the Field Formatter Output

```
...
$rows = [];

foreach (self::$abilities as $ability => $label) {
    $temp_ability = 'temp_' . $ability;
    $temp_value = $item->$temp_ability;

    $rows[$ability]['label'] = $this->t($label);
    $rows[$ability]['base_score'] = $item->$ability;
    $rows[$ability]['base_modifier'] = floor(($item->$ability - 10) / 2);
    $rows[$ability]['temp_score'] = $item->$temp_ability;
    $rows[$ability]['temp_modifier'] = (is_numeric($item->$temp_ability))
        ? floor(((int)$item->$temp_ability - 10) / 2) : NULL;

    $element[$delta] = [
        '#type' => 'table',
        '#header' => $header,
        '#rows' => $rows,
    ];
}
}

return $element;
}
```

Abilities Field Formatter

Displayed as tabular data,
but any theme function can
be used

This is a single field
instance

Abilities

| Ability | Base Score | Base Modifier | Temporary Score | Temporary Modifier |
|--------------|------------|---------------|-----------------|--------------------|
| Strength | 9 | -1 | 0 | -5 |
| Dexterity | 12 | 1 | 14 | 2 |
| Constitution | 11 | 0 | | |
| Intelligence | 19 | 4 | | |
| Wisdom | 15 | 2 | | |
| Charisma | 16 | 3 | | |

Ingredients List (again)

Single Field instance

Infinite cardinality
allows for “Add
another item”

Each new item is
represented by a
field’s delta

| INGREDIENTS | | | | |
|-------------|------------------------------------|---|--|--|
| | Quantity | Unit | Name | Note |
| + | <input type="text" value="2 1/2"/> | <input type="text" value="cup (c)"/> | <input type="text" value="All-purpose Flour (1)"/> | <input type="text"/> |
| + | <input type="text" value="1/4"/> | <input type="text" value="cup (c)"/> | <input type="text" value="Sugar (4)"/> | <input type="text"/> |
| + | <input type="text" value="1"/> | <input type="text" value="teaspoon (t)"/> | <input type="text" value="Salt (3)"/> | <input type="text"/> |
| + | <input type="text" value="1"/> | <input type="text" value="package (pk)"/> | <input type="text" value="Yeast (5)"/> | <input type="text" value="Regular or quick active dry yeast will work"/> |
| + | <input type="text" value="1/2"/> | <input type="text" value="cup (c)"/> | <input type="text" value="Water (6)"/> | <input type="text" value="Very warm (120°-130°F)"/> |
| + | <input type="text" value="1/2"/> | <input type="text" value="cup (c)"/> | <input type="text" value="Milk (7)"/> | <input type="text" value="Very warm (120°-130°F)"/> |
| + | <input type="text" value="1"/> | <input type="text" value="unit"/> | <input type="text" value="Egg (8)"/> | <input type="text"/> |
| + | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |

Abilities Field Formatter

Recipe has a single field instance for Ingredients

Ingredients have infinite cardinality

This Ingredient field has a delta from 0-8

Homemade Pan Rolls



84 Ratings



44 Comments

Prep
30 MIN

Total
2 HR 18 MIN

Servings
4



One whiff of these fragrant rolls, and you'll be counting the moments until you can take them out of the oven!

Ingredients

- 2 1/2** to 3 cups All-purpose flour
- 1/4** cup sugar
- 1/4** cup shortening
- 1** teaspoon salt
- 1** package regular or quick active dry yeast
- 1/2** cup very warm water (120°F to 130°F)
- 1/2** cup very warm milk (120°F to 130°F)
- 1** egg
- Butter or margarine, melted

Steps

Other Class Methods

- **viewElements** - Returns a renderable array of the field's value(s).
- **defaultSettings** - Allows for default values for this formatter.
- **settingsForm** - The form used to allow admins to change widget settings.
- **settingsSummary** - Block of markup describing the settings.

Theming Fields



Twig Templates for Fields

Theming of any field is made possible using a template file in your theme or module:

- Based on [field.html.twig](#)
- [field--ingredient.html.twig](#)
- [field--abilities.html.twig](#)

The twig template contains markup to render each element.

`#theme=>'ingredient'` in the Field Formatter tells Drupal to use the `theme_ingredient`, an

Field and hook_theme

```
/**
 * Implements hook_theme().
 */
function ingredient_theme($existing, $type, $theme, $path) {
  $theme = [
    'ingredient' => [
      'variables' => [
        'quantity' => NULL,
        'unit' => NULL,
        'name' => NULL,
        'note' => NULL,
      ],
    ],
  ],
];

  return $theme;
}
```

This creates uses the module's implementation of hook_theme to register a new theme key called 'ingredient'.

This will allow for a new template called 'ingredient.html.twig' to be used for theming Ingredient fields.

An Example of a Field Kernel Test



Kernel Test

```
class AbilitiesFieldTest extends FieldKernelTestBase {  
  
  public static $modules = ['dnd_fields'];  
  
  /**  
   * Modules to enable.  
   *  
   * @var array  
   */  
  protected function setUp() {  
    parent::setUp();  
  
    // Create a dnd_fields Abilities field storage and field for validation.  
    FieldStorageConfig::create([  
      'field_name' => 'field_test',  
      'entity_type' => 'entity_test',  
      'type' => 'dnd_fields_abilities',  
    ]->save());  
  
    FieldConfig::create([  
      'entity_type' => 'entity_test',  
      'field_name' => 'field_test',  
      'bundle' => 'entity_test',  
    ]->save());  
  }  
  ...  
}
```



Kernel Test Setup

```
/**
 * Tests using entity fields of the dnd_fields Abilities field type.
 */
public function testAbilitiesField() {
    // Verify entity creation.
    $entity = EntityTest::create();

    $values = [
        'str' => rand(3, 18),
        'dex' => rand(3, 18),
        'con' => rand(3, 18),
        'int' => rand(3, 18),
        'wis' => rand(3, 18),
        'chr' => rand(3, 18),
    ];
    foreach ($values as $attribute => $value) {
        $values['temp_' . $attribute] = $value + rand(-2, 2);
    }

    $entity->field_test = $values;

    foreach ($values as $attribute => $value) {
        $entity->name->$attribute = $value;
    }

    $entity->save();
}
```

Kernel Test Assertions

...

```
// Verify entity has been created properly.  
$id = $entity->id();  
  
$entity = EntityTest::load($id);  
  
$this->assertTrue($entity->field_test instanceof FieldItemListInterface, 'Field implements interface.');
```

```
$this->assertTrue($entity->field_test[0] instanceof FieldItemInterface, 'Field item implements interface.');
```

```
foreach ($values as $attribute => $value) {  
    $this->assertEquals($entity->field_test->$attribute, $value);  
    $this->assertEquals($entity->field_test[0]->$attribute, $value);  
}
```

...



Kernel Test - Changing Values

```
...
// Verify changing the field value.
$new_values = [
    'str' => rand(3, 18),
    'dex' => rand(3, 18),
    'con' => rand(3, 18),
    'int' => rand(3, 18),
    'wis' => rand(3, 18),
    'chr' => rand(3, 18),
];
foreach ($new_values as $attribute => $value) {
    $values['temp_' . $attribute] = $value + rand(-2, 2);
}

$entity->field_test = $new_values;
foreach ($new_values as $new_attribute => $new_value) {
    $this->assertEqual($entity->field_test->$new_attribute, $new_value);
    $this->assertEqual($entity->field_test[0]->$new_attribute, $new_value);
}

// Read changed entity and assert changed values.
$entity->save();
$entity = EntityTest::load($id);
foreach ($new_values as $new_attribute => $new_value) {
    $this->assertEqual($entity->field_test->$new_attribute, $new_value);
    $this->assertEqual($entity->field_test[0]->$new_attribute, $new_value);
}
}
```



Blog Post
Coming Soon...

www.phase2technology.com/blog
[@phase2](https://twitter.com/phase2)



What did you think?

events.drupal.org/seattle2019/sessions/custom-compound-fields-drupal-8

Take the Survey!

www.surveymonkey.com/r/DrupalConSeattle



Join us for contribution opportunities

Friday, April 12, 2019

Mentored Contribution

9:00-18:00
Room: 602

First Time Contributor Workshop

9:00-12:00
Room: 606

General Contribution

9:00-18:00
Room: 6A



Questions?



thank you

Come see us at
BOOTH 201



Tobby Hagler

thagler@phase2technology.com
@thagler

