Usage

- Default Values
- Required Fields
- Help Text
- Width
  - NB: This feature does nothing in term meta container due to the table layout used by WordPress term
- Classes
- Attributes

- REST API

Fields are the building block of every container.

New field are created using the `make` method `Field::make( $type, $name, $label = null )`, where:

`$type`

The type of the field. This parameter should be valid class name of a field. For example `text` will create field of class `Field_Text`.

`$name`

Name of the field. Used as a key when stored in the database and for you to retrieve its value. Please note, that all fields have their names automatically prefixed with an underscore when stored (e.g. `bgcolor` becomes `_bgcolor`).

`$label` *(optional)*

The label of the field is displayed in the back-end only, where the container is visible. When the parameter is omitted it is automatically derived from the `$name` If this parameter is omitted, and the `$name` starts with `crb_`, the `Crb` part will not be displayed in the generated label.

The factory greatly simplifies the field creation process, since it returns the field object itself and you don't need to assign it to a variable. The fields API supports method chaining (as seen in the example below).

```
// Create image field with name "customer_photo" and label "Photo"
Field::make( 'image', 'crb_customer_photo', 'Photo' )

// Here the title is automatically set to "Custom Sidebar"
Field::make( 'sidebar', 'crb_custom_sidebar' )

// Method chaining
Field::make( 'select', 'crb_color' )
    ->add_options( array('red', 'blue', 'green') )
    ->set_help_text( 'Pick a color' )
```

All field types originate from a single class named `Field` and inherit the following basic features:

Default Values

You can assign a default value for each field in every container. The default value is used when there is currently no value for the particular field in the database. This is the case for example, when you add a new post, or you add a new theme options field to existing container.

To assign a default value, you use:

`Field::make(...)->set_default_value( $default_value )`

Required Fields

You can mark any field as required, in which case the user will need to fill it out before submitting. To set a field as required, you use:

`Field::make(...)->set_required( true )`

Help Text

Help text is used as a hint to the user, who will use the field. It is usually rendered under the field and contains more information about what it should contain – requirements, examples, links, etc. HTML tags are allowed.

You add help text using:

`Field::make(...)->set_help_text( $text )`

Width

You can set the width to fields that are next to each other and they will align on one row. To set the field width %, use:

```
Field::make(...)->set_width( 50 )
```

NB: This feature does nothing in term meta container due to the table layout used by WordPress term editing screen.

Classes

Custom field classes can be added using the `set_classes()` method, for example:

```
Field::make(...)->set_classes( 'my-custom-class' )
```

Attributes

You can override certain field attributes using the `set_attribute()` method of fields based on <input> and <textarea>:

```
Field::make( 'text', 'crb_phone', 'Phone' )
    ->set_attribute( 'placeholder', '(***) ***-****' );
```

For a full list of what attributes are allowed visit the Text or Textarea field documentation pages.

REST API

Fields can be exposed in WordPress' REST API by using their `set_visible_in_rest_api()` method:

```
Field::make(...)->set_visible_in_rest_api( $visible = true )
```

□□□ #2
□ □□□ □□□ 2026-08-29 16:13:57 UTC
□ □□□ □□□ 2026-08-29 16:13:57 UTC