

Backdrop Live October 2023 Using hooks in Backdrop CMS

With a few examples

Alejandro Cremaschi (@argiepiano)
TenutoWeb LLC



What are hooks?

- Functions added by the site developer
- Executed by Backdrop at specific points of the page request
- Documented as `hook_xxx()` ;
- Follow the pattern: `MODULE_xxx()` ; (replace `hook` with module name)
- Typically placed in your custom `.module` files
- Documented in files that end with `api.php` - e.g., `system.api.php`
- Full documentation also found on [Backdrop site](#)



How can hooks help?

Extend Backdrop

For example:


- `hook_menu()`
- `hook_block_info()`
- `hook_token_info()`
- `hook_entity_info()`
- `hook_permission()`
- `hook_field_widget_info()`
- `hook_field_formatter_info()`
- `hook_action_info()`
- `hook_theme()`



How can hooks help?

React to an event

For example:

- `hook_node_presave()` / `hook_node_insert()` / `hook_node_delete()` / `hook_node_update()`
 - `hook_field_attach_form()`
 - `hook_preprocess_HOOK()`
 - `hook_node_view()`
- 

How can hooks help?

Alter “stuff” such as forms and output arrays

For example:

- `hook_form_alter()`
- `hook_form_FORM_ID_alter()`
- `hook_block_view_alter()`
- `hook_node_view_alter()`



Alter hooks

Alter hooks often receive parameters by reference (see & before parameters)

Alter hooks typically don't return values

```
function MODULE_form_alter(&$form, &$form_state, $form_id) {  
  $form['some_text'] = array(  
    '#markup' => 'Hello, world!',  
  );  
}
```



Restaurant Review Sample Site



Sample site

- Simple site - just an example
- Node type `restaurant_reviews`
- Roles: `reviewer`, `junior_reviewer`, `editor`
- Reviewers create nodes with restaurant reviews
- Editor or admin approve reviews



hook_form_FORM_ID_alter()

Syntax used for nodes:

```
function bdlive_demo_form_restaurant_reviews_node_form_alter(&$form, &$form_state, &$form_id) { }
```

node_form = default form for node editing

restaurant_reviews_node_form = form for content type restaurant_review



Pre-select location to match user's field

User account has `field_location` (region of the US) (`list (text)` field type)

The `restaurant_reviews` content type also contains this field

Hook is used to pre-select the location to match the user's in the node form



Pre-select location to match user's field

```
function bdlive_demo_form_restaurant_reviews_node_form_alter(&$form, &$form_state, &$form_id) {  
  global $user;  
  $current_user = user_load($user->uid);  
  
  $location_value = field_get_value($current_user, 'field_location');  
  
  $form['field_location'][LANGUAGE_NONE]['#default_value'][] = $location_value;  
}
```

Q: How do we know where to "put" the default value within the structure of the \$form array?

A: `dpm($form); !`



Remove option for specific role

Remove the "Sushi" select option from the form, for field taxonomy term `field_restaurant_type` if the user has the role "junior reviewer" (they are not allowed to review sushi places!)



Remove option from select list of term ref field for specific role

```
function bdlive_demo_form_restaurant_reviews_node_form_alter(&$form, &$form_state, &$form_id) {  
  $term_array = taxonomy_term_load_multiple_by_name('Sushi', 'restaurant_type');  
  $first_term_in_array = reset($term_array);  
  
  if (user_has_role('junior_reviewer')) {  
    unset($form['field_restaurant_type'][LANGUAGE_NONE]['#options'][$first_term_in_array->tid]);  
  }  
}
```

Add custom validation callback

Validation callback will disallow certain selections:

- Prevent junior reviewers to give low ratings to American restaurants



Add custom validation callback

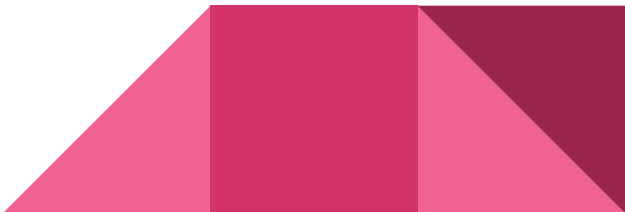
```
function bdlive_demo_form_restaurant_reviews_node_form_alter(&$form, &$form_state, &$form_id) {  
  $form['#validate'][] = 'bdlive_demo_custom_validation';  
}  
  
function bdlive_demo_custom_validation($form, &$form_state) {  
  $values = $form_state['values'];  
  
  // Get the tid for American restaurants.  
  $term_array = taxonomy_term_load_multiple_by_name('American', 'restaurant_type');  
  $first_term_in_array = reset($term_array);  
  
  if (user_has_role('junior_reviewer')  
    && $values['field_restaurant_type'][LANGUAGE_NONE][0]['tid'] == $first_term_in_array->tid  
    && $values['field_score'][LANGUAGE_NONE][0]['value'] == 'Not for me') {  
    form_set_error('field_score[und]', t('What??? As a Junior Reviewer you are not allowed to  
give bad reviews to American restaurants!'));  
  }  
}
```

About form_set_error()

```
form_set_error('field_score'[und', t('What??? As a Junior Reviewer you are not allowed to  
give bad reviews to American restaurants!'));
```

How did we get the first parameter for `form_set_error()`?

- `dpm($form);`
- Inspect for field array up to the last child (non-child elements start with #)
- Build a string:
 - `FIELD_NAME[CHILD_1][CHILD_2`



field_score (Array, 7 elements)

#type (String, 9 characters) **container**

#attributes (Array, 1 element)

#weight (String, 1 characters) **6**

#tree (Boolean) **TRUE**

#language (String, 3 characters) **und**

und (Array, 19 elements)

#entity (Object) **Node**

#entity_type (String, 4 characters) **node**

#bundle (String, 18 characters) **restaurant_reviews**

#field_name (String, 11 characters) **field_score**

#language (String, 3 characters) **und**

#field_parents (Array, 0 elements)

#columns (Array, 1 element)

#title (String, 5 characters) **Score**

#description (String, 0 characters)

#required (Boolean) **TRUE**

#delta (Integer) **0**

#type (String, 6 characters) **select**

#default_value (Array, 0 elements)

#multiple (Boolean) **FALSE**

#options (Array, 5 elements)

#value_key (String, 5 characters) **value**

#element_validate (Array, 1 element)

#properties (Array, 5 elements)

#after_build (Array, 1 element)

#access (Boolean) **TRUE**

Hide the `field_approved` boolean field for users without a custom permission

Only users with ``approve reviews`` custom permission should be able to see (and change) this field

This requires creating a custom permission with ``hook_permission()``

To hide a field, set its `#access` property to `FALSE`

Alternatively, you can disable it by setting `#disabled` to `TRUE`



Hide the field_approved boolean field for users without a custom permission

```
function bdlive_demo_form_restaurant_reviews_node_form_alter(&$form, &$form_state, &$form_id) {  
  $form['field_approved'][LANGUAGE_NONE]['#access'] = user_access('approve reviews');  
}  
  
/**  
 * Implements hook_permission().  
 */  
function bdlive_demo_permission() {  
  return array(  
    'approve reviews' => array(  
      'title' => t('Approve reviews'),  
      'description' => t('Approve submitted reviews.'),  
    ),  
  );  
}
```



Hooks to alter the display of nodes

Hooks to react upon saving/editing/deleting of nodes

Goal

When viewing a `restaurant_reviews` node (only "full" view mode):

- Attach custom text indicating:
 - Total number of reviews by this reviewer (both approved and unapproved)
 - Total number of approved reviews
 - These data are kept in two number fields in the user account entity for each reviewer
- Attach a view table summarizing all reviews by this reviewer



How to modify output of nodes

We use `hook_node_view_alter(&$build)`, which is invoked before rendering the output

This hook receives a "build array" or "renderable array" - an array of elements that will be eventually be turned into markup by functions like `render()`

Typically, the render array contains a key `'#type'` or `'#theme'` that tells the Backdrop which theming functions to use to render the element



```

function bdlive_demo_node_view_alter(&$build) {
  // Only add the view for full view mode for this type of nodes..
  if ($build['#view_mode'] == 'full' && $build['#node']->type == 'restaurant_reviews') {
    $node = $build['#node'];
    $author = user_load($node->uid);

    $build['view_title'] = array(
      '#type' => 'markup',
      '#markup' => '<h3>Reviews by ' . field_get_value($author, 'field_full_name') . '</h3>',
      '#weight' => 199,
    );

    // We also add the number of approved reviews.
    $build['approved_number'] = array(
      '#type' => 'help',
      '#markup' => 'Number of approved reviews: ' . field_get_value($author, 'field_number_of_approved_reviews'),
      '#weight' => 200,
    );

    // We also add the total number of reviews.
    $build['total_number'] = array(
      '#type' => 'help',
      '#markup' => 'Total number of reviews: ' . field_get_value($author, 'field_number_of_reviews'),
      '#weight' => 201,
    );

    $build['review_view'] = array(
      '#type' => 'help',
      '#markup' => views_embed_view('reviews', 'default', $node->uid),
      '#weight' => 202,
    );
  }
}

```

How do we keep track of number of reviews?

Use `hook_node_insert()`, `hook_node_update()`, `hook_node_delete()`




```

function bdlive_demo_node_update($node) {
  if ($node->type == 'restaurant_reviews') {
    // In order to determine if the approved field was edited, we must load the
    // original node.
    $original_node = $node->original;
    $original_value = (int) field_get_value($original_node, 'field_approved');
    $new_value = (int) field_get_value($node, 'field_approved');
    if ($original_value != $new_value) {
      $user = user_load($node->uid);
      $number_of_approved = field_get_value($user, 'field_number_of_approved_reviews');
      // If approval was changed from "no" or "empty" to "yes", increment.
      if (empty($original_value)) {
        $number_of_approved++;
      }
      else {
        // The approval was "yes" and now is "no".
        $number_of_approved--;
      }
      $user->field_number_of_approved_reviews[LANGUAGE_NONE][0]['value'] = $number_of_approved;
      $user->save();
    }
  }
}


```

Another way to hide ANY field based on permission

Above, we hid `field_approved` based on a permission

Now, we'll add a custom setting to any field instance to hide it.

We'll use:

- `bdlive_demo_form_field_ui_field_edit_form_alter()` to alter the "Manage fields" form for nodes of bundle `restaurant_reviews`. This allows us to add a new setting
 - `bdlive_demo_field_widget_form_alter()` to alter the widget and hide the field that has this setting
- 

Adding custom settings to the field settings form in "Manage fields"

We add the settings as a subarray inside `$instance[' settings']`. These boolean setting to hide/show the field in the form will be stored in `$instance[' settings'][' restaurant_settings'][' hide_in_form']`



```

function bdlive_demo_form_field_ui_field_edit_form_alter(&$form, &$form_state) {
  $instance = $form['#instance'];
  // If the current field instance is not locked and is attached to a node of type
  // restaurant_reviews, show this setting element in the UI.
  if (empty($form['locked']) && $instance['entity_type'] == 'node' && $instance['bundle'] ==
'restaurant_reviews') {
    $form['instance']['settings']['restaurant_settings'] = array(
      '#type' => 'fieldset',
      '#title' => t('Restaurant Review Settings'),
      '#weight' => 5,
      '#collapsible' => FALSE,
    );
    $form['instance']['settings']['restaurant_settings']['hide_in_form'] = array(
      '#type' => 'checkbox',
      '#title' => t('Hide this field'),
      '#description' => t('Hide this field in the node add/edit form for users without the
"Approve reviews" permission.'),
      '#default_value' => !empty($instance['settings']['restaurant_settings']['hide_in_form']) ?
$instance['settings']['restaurant_settings']['hide_in_form'] : FALSE,
    );
  }
}

```

Actually hiding the field in the form

We use `bdlive_demo_field_widget_form_alter()` and check if the `hide_in_form` instance setting is true for this field

If true, then we assign `['#access'] = FALSE` to hide the field, but only for users without permission `approve reviews`



```
function bdlive_demo_field_widget_form_alter(&$element, &$form_state, $context)
{
  if ($element['#entity_type'] == 'node' && $element['#bundle'] ==
'restaurant_reviews') {
    if (isset($context['instance']['settings']['restaurant_settings'])
        && $context['instance']['settings']['restaurant_settings']['hide_in_form']
        && !user_access('approve reviews')) {

      $element['#access'] = FALSE;
    }
  }
}
```

Bonus hooks



Sample module includes

- `hook_menu()` and page callback to display a node using a specific view mode with path `node/NID/view-mode/VIEW-MODE`
- `hook_block_view_alter()` to alter the display of the "Powered by Backdrop" block
- `hook_views_pre_render()` to add the AJAX and Dialog libraries to a View, to allow to open a listed node within a Dialog box
- `hook_action_info` to define new node bulk actions "Approve review" and "Unapprove review" for admins and Editors

