Getting up-to-speed: making the entity fieldable

👉 This section is about making your custom entity fieldable. At the end you will be able to connect your **custom entity** with the power of the **drupal user-interface for adding fields, form modes and view modes**. You will be able to add fields to your entity via the UI and manage the display. The created fields will be exportable as configuration.

As told earlier, the entity data structure in drupal made the software famous. We can extend our custom entity with this dynamic and extremely feature-rich UI.

Advantages of working this way are:

- **Definition and naming can be done in the UI**. You get access to powerful fields with features as autocomplete entity reference, multifield, private files, ...
- Create **view modes** of the entity via the UI (a teaser and full entity view for example) and set the display of each field
- Manage **form modes** (configure the appearance of the add and edit form via the UI)

With that UI, we can add the desired fields to our entity by just clicking:

- Images
- A PDF upload
- Description
- Short text
- Tags
- … There are hundreds of modules available that define all sorts of fields!

Note that we already have a description defined in our **Offer.php** entity structure. We're about to delete it. We'll delete our entities first and uninstall again:

```
bash$ drush entity:delete offer
[success] Deleted offer entity Ids: 1, 2
bash$ drush pmu offer
[success] Successfully uninstalled: offer
```

Now **remove** the *$fields['message']* from the *BaseFieldDefinitions* in your **Offer.php** entity file.

💻 make sure the core **field_ui** module is enabled.

```
bash$ drush en field_ui
[success] Successfully enabled: field_ui
```

To the annotations in **modules/custom/offer/src/Entity/Offer.php** add a fiel_ui_base_route key:

```
*      "create" = "/offer/c...
*    },
*    field_ui_base_route = "entity.offer.settings",
*    revision_me....
```

The route **entity.offer.settings** will be the landing page of the ui settings of our entity. If we configure this route to /admin/structure/offer, the following routes will become available automatically:

- admin/structure/offer/fields (fields ui)
- admin/structure/offer/form-display (add form display)
- admin/structure/offer/display (view modes display management)

The settings landing page is usually a settings form. While we will not use additional settings in this form, let's keep the good practice and add this route as a form.

To **modules/custom/offer/routing.yml** we add:

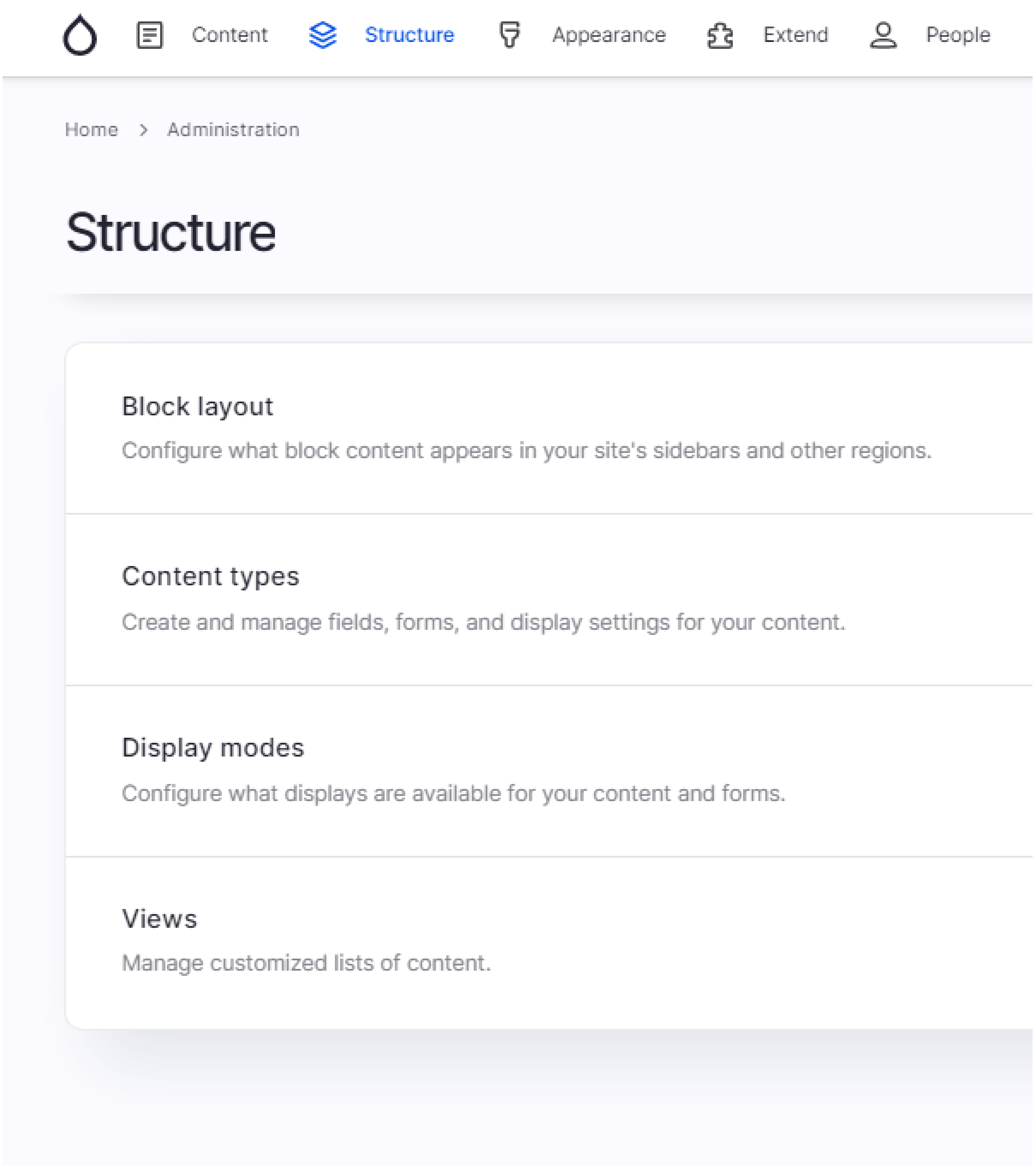
```
entity.offer.settings:
  path: 'admin/structure/offer'
  defaults:
    _form: '\Drupal\offer\Form\OfferSettingsForm'
    _title: 'Offer settings'
  requirements:
    _permission: 'administer own offers'
```

To **modules/custom/offer/src/Form/OfferSettingsForm**:

```
<?php
/**
* @file
```

```php
* Contains \Drupal\offer\Form\OfferSettingsForm.
*/

namespace Drupal\offer\Form;

use Drupal\Core\Form\FormBase;
use Drupal\Core\Form\FormStateInterface;

/**
* Class OfferSettingsForm.
*
* @package Drupal\offer\Form
*
* @ingroup offer
*/
class OfferSettingsForm extends FormBase {
 /**
   * Returns a unique string identifying the form.
   *
   * @return string
   *   The unique string identifying the form.
   */
 public function getFormId() {
   return 'offer_settings';
 }

 /**
   * {@inheritdoc}
   */
 public function submitForm(array &$form, FormStateInterface
$form_state) {
   // Empty implementation of the abstract submit class.
 }

 /**
   * {@inheritdoc}
   */
 public function buildForm(array $form, FormStateInterface $form_state)
{
   $form['offer_settings']['#markup'] = 'Settings form for offer. We
don\'t need additional entity settings. Manage field settings with the
tabs above.';
   return $form;
 }

}
```

What we want is to the settings on the admin/structure page to our offer settings form.



Structure

Block layout
Configure what block content appears in your site's sidebars and other regions.

Content types
Create and manage fields, forms, and display settings for your content.

Display modes
Configure what displays are available for your content and forms.

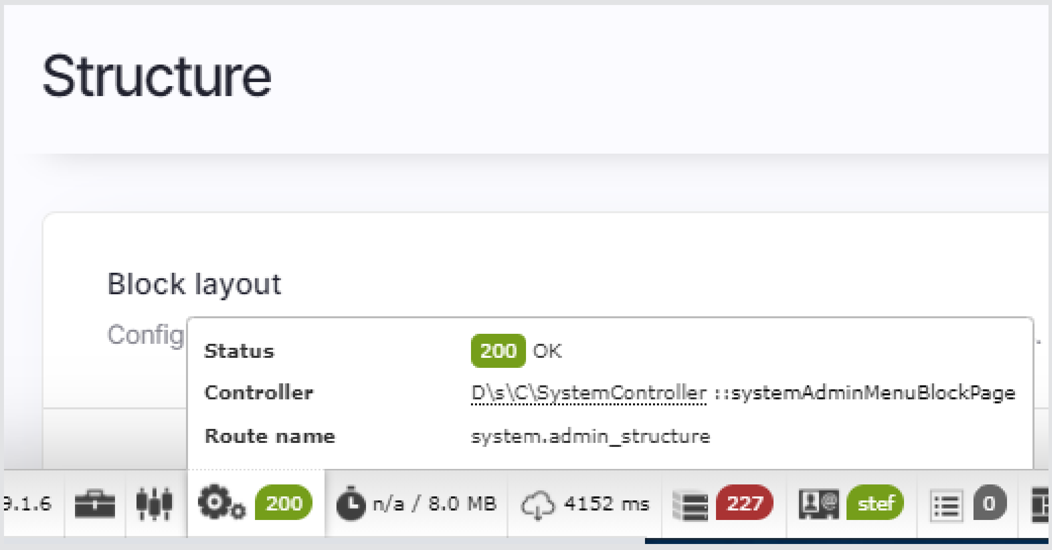
Views
Manage customized lists of content.

With the webprofiler toolbar (see [developer tools]) we check the route name for admin/structure.

💻 Enable the **weprofiler** module, which is part of the **devel** module and re-nable the

offer module.

```
bash$ drush en webprofiler offer
[success] Successfully enabled: webprofiler, offer
```



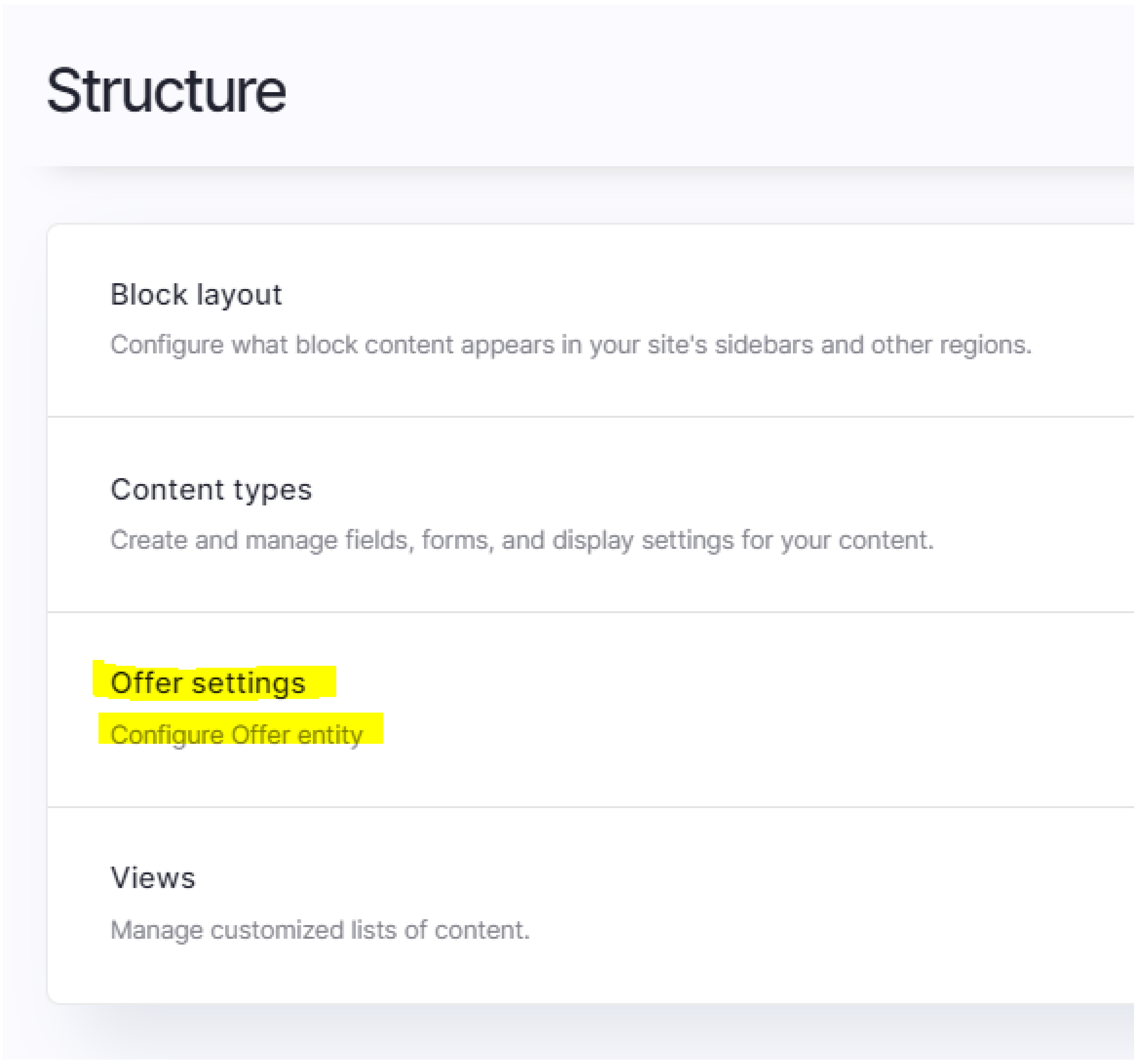
Status 200 OK
Controller D\s\C\SystemController ::systemAdminMenuBlockPage
Route name system.admin_structure

We discovered the route name is *system.admin.structure*. We'll use it next as a parent for our settings link.

Next, we add a file named **custom/offer/offer.links.menu.yml**:

```
offer.admin.structure.settings:
 title: Offer settings
 description: 'Configure Offer entity'
 route_name:  entity.offer.settings
 parent: system.admin_structure
```

💻 Always clear the caches after adding/editing new routes, menu links or other YAML config!



Structure

Block layout
Configure what block content appears in your site's sidebars and other regions.

Content types
Create and manage fields, forms, and display settings for your content.

Offer settings
Configure Offer entity

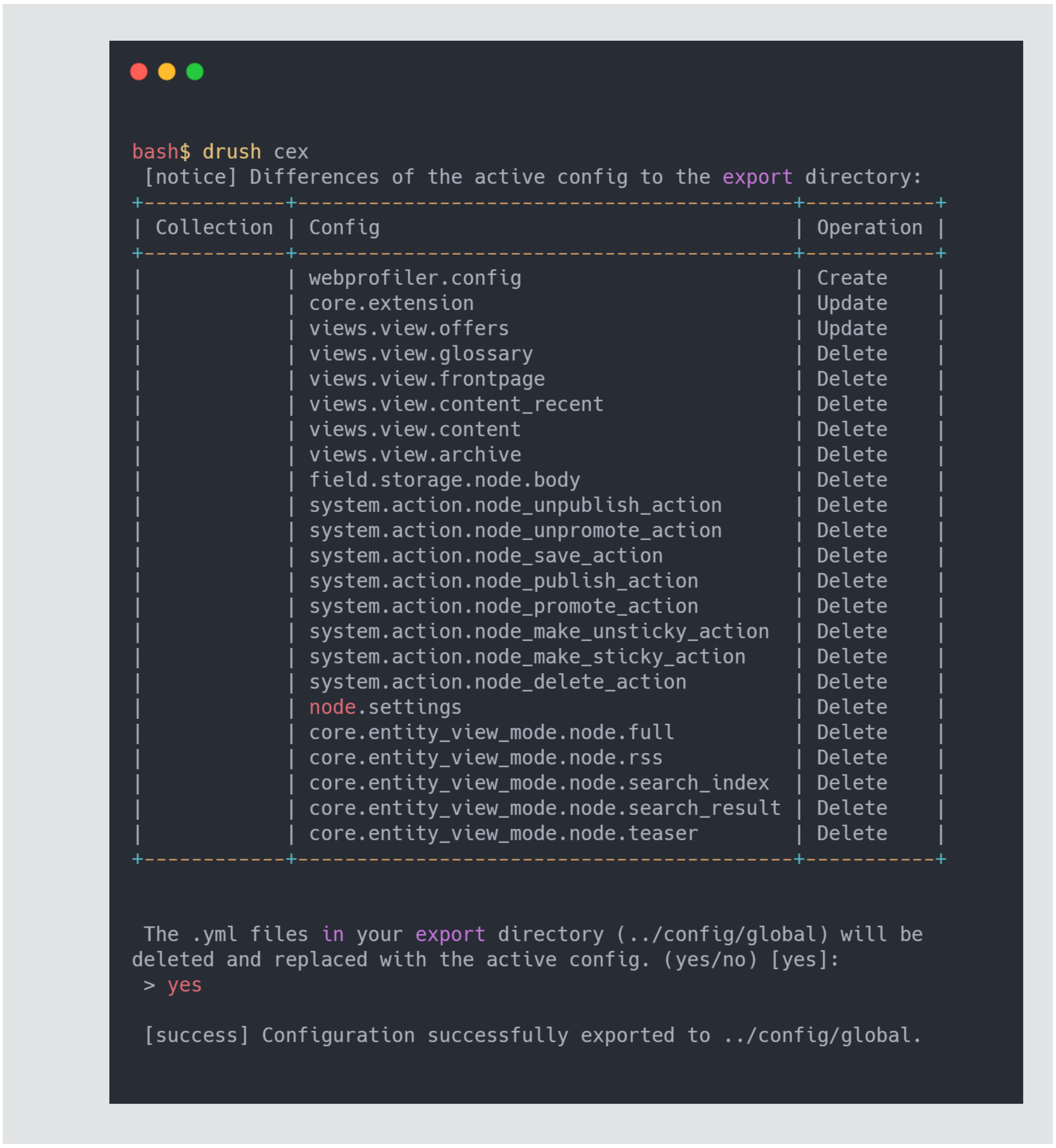
Views
Manage customized lists of content.

The link appeared in our structure tree, and the build-up is the same way as “content types”, which we know from the Node entity.

💻 If you still see “Content types”, the time is here to uninstall the Node entity type:

```
bash$ drush pmu node
[success] Successfully uninstalled: node
```

If you export config, this will delete some default settings from the node entity type as well.

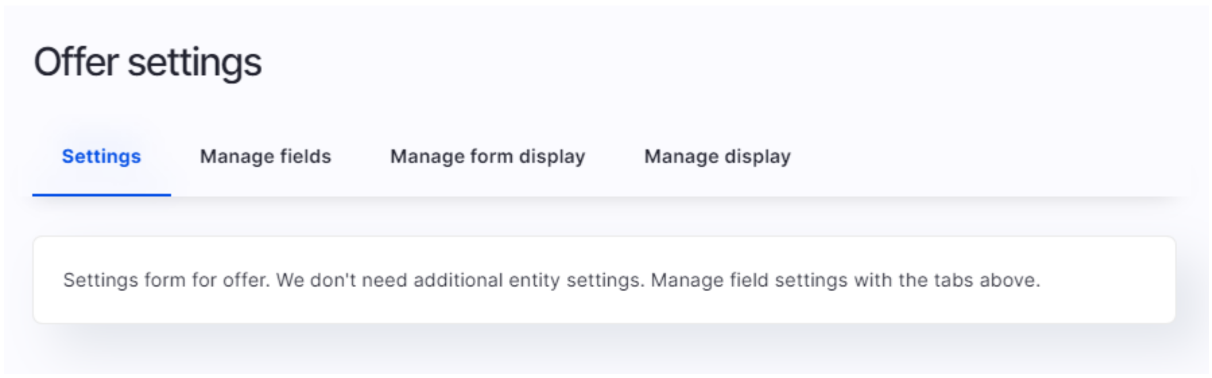


```
bash$ drush cex
 [notice] Differences of the active config to the export directory:
+------------+------------------------------------------+-----------+
| Collection | Config                                   | Operation |
+------------+------------------------------------------+-----------+
|            | webprofiler.config                       | Create    |
|            | core.extension                           | Update    |
|            | views.view.offers                        | Update    |
|            | views.view.glossary                      | Delete    |
|            | views.view.frontpage                     | Delete    |
|            | views.view.content_recent                | Delete    |
|            | views.view.content                       | Delete    |
|            | views.view.archive                       | Delete    |
|            | field.storage.node.body                  | Delete    |
|            | system.action.node_unpublish_action      | Delete    |
|            | system.action.node_unpromote_action      | Delete    |
|            | system.action.node_save_action           | Delete    |
|            | system.action.node_publish_action        | Delete    |
|            | system.action.node_promote_action        | Delete    |
|            | system.action.node_make_unsticky_action  | Delete    |
|            | system.action.node_make_sticky_action    | Delete    |
|            | system.action.node_delete_action         | Delete    |
|            | node.settings                            | Delete    |
|            | core.entity_view_mode.node.full          | Delete    |
|            | core.entity_view_mode.node.rss           | Delete    |
|            | core.entity_view_mode.node.search_index  | Delete    |
|            | core.entity_view_mode.node.search_result | Delete    |
|            | core.entity_view_mode.node.teaser        | Delete    |
+------------+------------------------------------------+-----------+


 The .yml files in your export directory (../config/global) will be
deleted and replaced with the active config. (yes/no) [yes]:
 > yes

 [success] Configuration successfully exported to ../config/global.
```

Finally, we add a “local tasks”-link to the settings page. This will activate the tasks tabs above to get the structure that is known when administering nodes. By this I mean the “Edit”, “Manage fields”, “Manage form display” and “Manage display” tabs. To **modules/custom/offer/offer.links.task.yml**, add:

```
# Activates the tabs on the entity admin pages
(/admin/structure/offer)
offer.settings_tab:
 route_name: entity.offer.settings
 title: 'Settings'
 base_route: entity.offer.settings
```

We re-enable the module and see the desired entity ui extensions on admin/structure/offer.



Settings form for offer. We don't need additional entity settings. Manage field settings with the tabs above.

Cool! Time to get up to speed by adding some fields.