Views integration of our custom entity to add a listing

> 👉 In this section we go over the possibility of adding **views support** for **custom entities**. At the end you will be able to create powerful listings with views in the same way you can do with nodes.

bash-5.0$ drush en views views_ui
[success] Successfully enabled: views, views_ui

> 💻 Enable the **views** and **views_ui** module.
>
> ```
> bash$ drush en views views_ui
> [success] Successfully enabled: views, views_ui
> ```

First, to ensure that our entities have integration with views we have to add an annotation with a link to the class.

In our **custom/offer/src/Entity/Offer.php** we add the views integration in the annotations like this:

```
*   handlers = {
*     "access" = "Drupal\offer\OfferAccessControlHandler",
*     "views_data" = "Drupal\offer\OfferViewsData",
*     "form" = {
*       "add" = "Drupal\offer\Form\OfferForm",
*       "edit" = "Drupal\offer\Form\OfferForm",
*       "delete" = "Drupal\offer\Form\OfferDeleteForm",
*     },
*   },
```

To the **modules/custom/offer/src/OfferViewsData.php**, add this code:

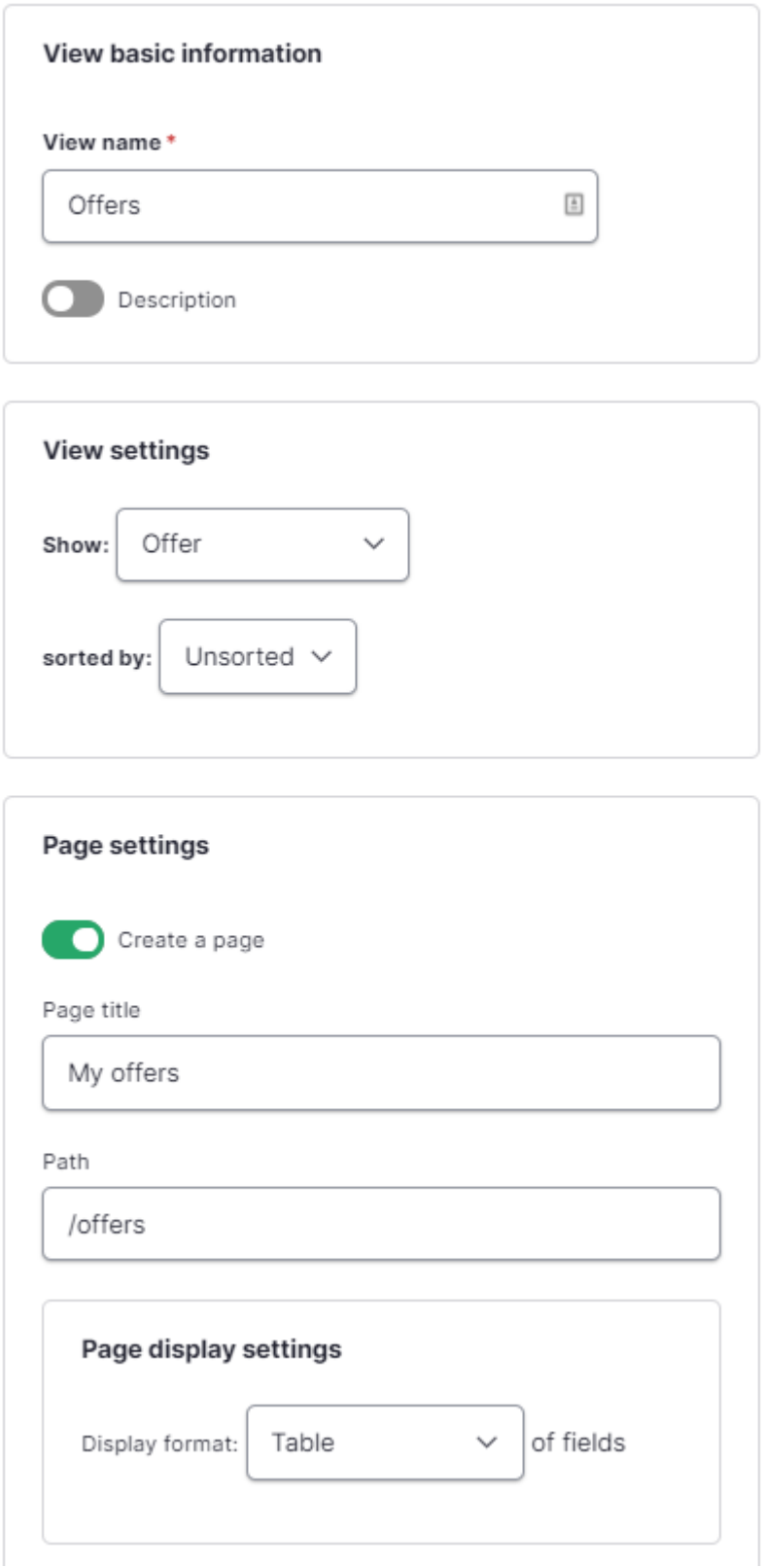
```
<?php

namespace Drupal\offer;
```

```php
use Drupal\views\EntityViewsData;

/**
* Provides views data for Offer entities.
*
*/
class OfferViewsData extends EntityViewsData {

 /**
  * Returns the Views data for the entity.
  */
 public function getViewsData() {
   $data = parent::getViewsData();
   return $data;
 }
}
```

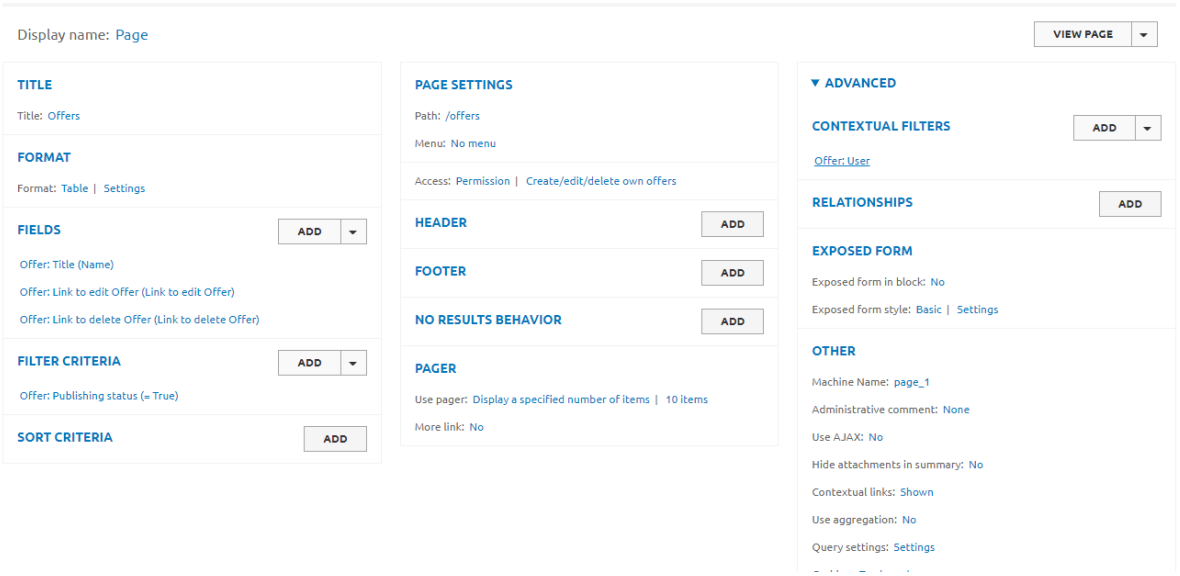
Clear caches and head to *admin/structure/views/add*. When adding a new view, we can now select “offer” as an entity to make a listing. This is pretty powerful, we get full access to all of views finest functionality:

- Quickly make (advanced) listings
- Search
- Filtering
- Pagination
- Bulk operations
- …

But for now, we make an easy listing of our entities.



We add title, edit and a delete button as fields.

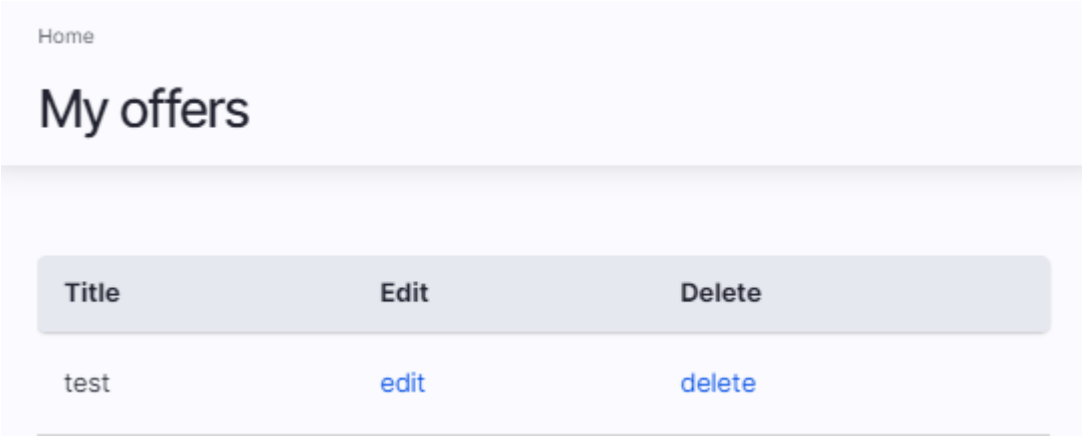


Important

Views is an extremely powerful tool. But think about it as a querying interface and a tool for building easy and advanced entity listings. If your entities are secured like we did with the offers, this does not mean they can not appear in views listings for other users.

For that, always make sure you only show entities that have the logged in user as author (*Advanced > Contextual filter > User > Provide default value > User id from logged in user*). Of course, if you make the users author of their own entities like we are doing.

Also, check for permissions on the listing. In this case, we gave permission to users that have “administer own offers“ (Create/edit/delete own offers) permission.



We’ve got our listing! We will tweak it further in a later chapter, but this is the functionality we want at this moment.

💻 Time to get our CRUD operations complete by adding our entity listing to as the collection of this entity. To **offer/offer.routing.yml**, add:

```
entity.offer.collection:
  path: '/offers'
  requirements:
    _permission: 'administer own offers'
```

Test by adding, editing and deleting an offer. But rebuild caches first!

Export your configuration now

```
drush cex -y
```

and look for **views.view.offers.yml**. Copy the file to **modules/custom/offer/config/install**.
This will keep our view inside our offers module and on next installation, the view will be installed.

To our **offer.info.yml** file, we add:

```
dependencies:
 - drupal:views
```

We'll disable the module and reïnstall to see if it all works. Delete our entities first like this:

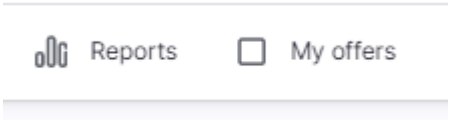
```
bash$ drush entity:delete offer
[success] Deleted offer entity Ids: 1, 2
bash$ drush pmu offer
[success] Successfully uninstalled: offer
bash$ drush en offer
[success] Successfully enabled: offer
```

We have now full CRUD operations up and working in a generic and reusable module.

To make our lives a bit more comfortable, we add a menu link to the toolbar for direct access. To **custom/modules/offer/offer.links.menu.yml** add:

```
offer.toolbar.my_offers:
 title: 'My offers'
 route_name: entity.offer.collection
 parent: system.admin
 weight: 99
```

This shows a link to the “My offers” overview page in the toolbar.



> 💻 Give authenticated users access to the toolbar first (via permissions), but note that this is only for the development phase of the platform.

Next, this listing needs a big “Add an offer” button. Drupal has a system for this. Add **offer.links.action.yml** and add:

```
offer.add_offer:
  route_name: offer.add
  title: 'Add an offer'
  appears_on:
    - entity.offer.collection
```

Clear caches and we now have our action button: