|  |  |  |  |
| --- | --- | --- | --- |
| 1down vote[favorite](https://drupal.stackexchange.com/questions/217407/properly-adding-an-action-to-a-contententityform) | I've created an add/edit form for a custom entity type, which extends Drupal\Core\Entity\ContentEntityForm. By default this form always comes with a "Save" button, and a "Delete" button if editing. I've been asked to add a "Cancel" button as well, but can't figure out how this is supposed to be done in Drupal.  I've seen plenty of answers about using hook\_form\_alter(), but nothing that is really working. My current solution is to add  $form['cancel'] = array(  '#type' => 'submit',  '#weight' => 10001,  '#submit' => array('cancel\_submit'),  '#value' => t('Cancel'),  );  to the form's form() function, where cancel\_submit() is a custom function, but that renders like [this](https://i.stack.imgur.com/J9NTI.png)which looks awful.  How can I get the two buttons to be in line with each other? Is there a way to properly add a button to the form actions?  [forms](https://drupal.stackexchange.com/questions/tagged/forms) [8](https://drupal.stackexchange.com/questions/tagged/8) [entities](https://drupal.stackexchange.com/questions/tagged/entities)   |  |  | | --- | --- | | [share](https://drupal.stackexchange.com/q/217407)[improve this question](https://drupal.stackexchange.com/posts/217407/edit) | asked Oct 11 '16 at 13:52  [saramm1](https://drupal.stackexchange.com/users/63407/saramm1)  **389**114 | |
|  | add a comment |

1 Answer

[active](https://drupal.stackexchange.com/questions/217407/properly-adding-an-action-to-a-contententityform?answertab=active#tab-top)[oldest](https://drupal.stackexchange.com/questions/217407/properly-adding-an-action-to-a-contententityform?answertab=oldest#tab-top)[**votes**](https://drupal.stackexchange.com/questions/217407/properly-adding-an-action-to-a-contententityform?answertab=votes#tab-top)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote3down voteaccepted | This is relatively simple when you define the form class yourself and one of the reasons Drupal 8 has gone all out on OOP. What you need to do is to extend the form class and implement the ::actionsmethod. This would look like this:  class CustomEntityForm extends ContentEntityForm {  ...  protected function actions(array $form, FormStateInterface $form\_state) {  $actions = parent::actions($form, $form\_state);  $actions['cancel'] = [  '#type' => 'submit',  '#value' => $this->t('Cancel'),  '#submit' => ['::cancelSubmit'],  ];  return $actions;  }  public function cancelSubmit($form, FormStateInterface $form\_state) {  // Du custom submit validation  }  }  You could also achieve this with normal form alter, but would need to place the submit button inside the actions element, which is usually called actions, fx  $form['actions']['cancel'] = array(  '#type' => 'submit',  '#weight' => 10001,  '#submit' => array('cancel\_submit'),  '#value' => t('Cancel'),  );   |  |  |  | | --- | --- | --- | | [share](https://drupal.stackexchange.com/a/217412)[improve this answer](https://drupal.stackexchange.com/posts/217412/edit) | [edited Oct 11 '16 at 14:45](https://drupal.stackexchange.com/posts/217412/revisions) | answered Oct 11 '16 at 14:12  [[https://www.gravatar.com/avatar/5953314abb5194a939a7a79309a397c0?s=32&d=identicon&r=PG](https://drupal.stackexchange.com/users/190/googletorp)](https://drupal.stackexchange.com/users/190/googletorp)  [googletorp](https://drupal.stackexchange.com/users/190/googletorp)**♦**  **31.6k**74795 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Thanks for the answer. I'm attempting your first method, but it's crashing my site with this message: "The website encountered an unexpected error. Please try again later. Recoverable fatal error: Argument 1 passed to Drupal\Core\Render\Element::children() must be of the type array, null given, called in {directory}\core\lib\Drupal\Core\Entity\EntityForm.php on line 206 and defined in Drupal\Core\Render\Element::children() (line 71 of core\lib\Drupal\Core\Render\Element.php)." – [saramm1](https://drupal.stackexchange.com/users/63407/saramm1)[Oct 11 '16 at 14:34](https://drupal.stackexchange.com/questions/217407/properly-adding-an-action-to-a-contententityform#comment265513_217412) | | |  |  | | --- | --- | |  |  | | @saramm1 I updated the code in the first method, you should note that code is mostly example code, not meant for copy/paste without edits – [googletorp](https://drupal.stackexchange.com/users/190/googletorp" \o "31,645 reputation)**[♦](https://drupal.stackexchange.com/users/190/googletorp" \o "31,645 reputation)** [Oct 11 '16 at 14:47](https://drupal.stackexchange.com/questions/217407/properly-adding-an-action-to-a-contententityform#comment265517_217412) | | |  |  | | --- | --- | |  |  | | Thanks, your second method also worked nicely. – [saramm1](https://drupal.stackexchange.com/users/63407/saramm1) [Oct 11 '16 at 14:50](https://drupal.stackexchange.com/questions/217407/properly-adding-an-action-to-a-contententityform#comment265519_217412) | |

**How to select fields from entities for forms**

<https://www.foreach.be/blog/how-select-fields-entities-forms>

http://www.microsoft.com/windows/internet-explorer/default.aspx

*Rendering a form is simple when the form and the entity relate one to one. But what if you need fields from multiple entities? Or what if you want to show the fields of an entity in a multistep format? This post is about a function that gets a render array for a field from an entity. This function enables you to select fields from one or more entities.*

When making custom forms in Drupal 8 I was limited by the restriction to use all the fields from a single entity. To break open the one-to-one relation between entity and form, I created a function that gets the render array from the field settings. These field settings are defined in the baseFieldDefinitions method of the entity.

modules/utils/Plugin/Field/FieldUtils.php

use Drupal\Core\Entity\Entity\EntityFormDisplay;

class FieldUtils

{

   public static function getFieldRenderarray($field\_name,

                   $entity, &$form, $form\_state, $options) {

     // the #parents key is needed for the form method to add

     // the fields to

     if(!isset($form['#parents'])) {

      $form['#parents'] = [];

     }

     // store the render display collection to reduce the

     // queries if multiple fields use the function

     $entity\_form\_display = $entity->getEntityTypeId().

                               '\_form\_display';

     $display = $form\_state->get($entity\_form\_display);

     if(empty($display)) {

       // get the display for all the fields of the entity

       $display = EntityFormDisplay::collectRenderDisplay(

                                   $entity, 'default');

       $form\_state->set($entity\_form\_display, $display);

     }

     $render\_array = $display

                       // get the fields render object

                       ->getRenderer($field\_name)

                       // get the render array

                       ->form($entity->get($field\_name), $form,

                                   $form\_state);

     // manipulate the render array

     if(!empty($options)) {

       if(isset($render\_array['widget'][0])) {

         $render\_array['widget'][0]['value'] =

               array\_merge($render\_array['widget'][0]['value'],

                   $options);

       }else{

         $render\_array['widget'] =

           array\_merge($render\_array['widget'], $options);

       }

     }

     return $render\_array;

   }

}

I added comments to the function to give you a better idea how the render array is fetched. There are a few things I want to provide more details for:

1. The EntityFormDisplay::collectRenderDisplay method uses the following code to fetch the BasefieldDefinitions settings:

\Drupal::entityManager()->getStorage('entity\_form\_display')->load($entity\_type . '.' . $bundle . '.' . $form\_mode)

This is the database query that I'm caching in the getFieldRenderarray method.  Drupal uses the collectRenderDisplay method in the ContentEntityForm and the QuickEditFieldForm classes.

1. The output of the getRenderer method contains the base render array. To obtain the render array of a specific widget for a form, you have to call theWidgetBase::form method.
2. The inspiration for the getFieldRenderarray method was the EntityFormDisplay::buildForm. This method contains the #parents key addition and both the getRenderer and form methods. If you want to change the field order, you have to add the #weight key to the fields. If you want to control access to the fields, you have to add the #access key. Both keys are set in EntityFormDisplay::buildForm.

*In this article I've shown you how to get render arrays for fields from the BaseFieldDefinition settings. Building a form based on this render array should be straightforward. I also gave a peek into the way how Drupal builds its forms. If you dig a little you can make Drupal work for you instead of against.*

* 3 Comments
* [**foreach**](https://disqus.com/home/forums/foreach/)
* [**Login**](https://disqus.com/embed/comments/?base=default&f=foreach&t_i=node%2F152&t_u=https%3A%2F%2Fwww.foreach.be%2Fblog%2Fhow-select-fields-entities-forms&t_e=How%20to%20select%20fields%20from%20entities%20for%20forms&t_d=How%20to%20select%20fields%20from%20entities%20for%20forms&t_t=How%20to%20select%20fields%20from%20entities%20for%20forms&s_o=default)
* [**1**](https://disqus.com/home/inbox/)
* [**Recommend**](https://disqus.com/embed/comments/?base=default&f=foreach&t_i=node%2F152&t_u=https%3A%2F%2Fwww.foreach.be%2Fblog%2Fhow-select-fields-entities-forms&t_e=How%20to%20select%20fields%20from%20entities%20for%20forms&t_d=How%20to%20select%20fields%20from%20entities%20for%20forms&t_t=How%20to%20select%20fields%20from%20entities%20for%20forms&s_o=default)
* [**Share**](https://disqus.com/embed/comments/?base=default&f=foreach&t_i=node%2F152&t_u=https%3A%2F%2Fwww.foreach.be%2Fblog%2Fhow-select-fields-entities-forms&t_e=How%20to%20select%20fields%20from%20entities%20for%20forms&t_d=How%20to%20select%20fields%20from%20entities%20for%20forms&t_t=How%20to%20select%20fields%20from%20entities%20for%20forms&s_o=default)
* [**Sort by Newest**](https://disqus.com/embed/comments/?base=default&f=foreach&t_i=node%2F152&t_u=https%3A%2F%2Fwww.foreach.be%2Fblog%2Fhow-select-fields-entities-forms&t_e=How%20to%20select%20fields%20from%20entities%20for%20forms&t_d=How%20to%20select%20fields%20from%20entities%20for%20forms&t_t=How%20to%20select%20fields%20from%20entities%20for%20forms&s_o=default)

Top of Form



Join the discussion…

**LOG IN WITH**

**OR SIGN UP WITH DISQUS**



Bottom of Form

[](https://disqus.com/by/cforcloud/)

[**cforcloud**](https://disqus.com/by/cforcloud/) • [2 years ago](https://www.foreach.be/blog/how-select-fields-entities-forms#comment-2499828735)

Could you provide us the complete source and a demo

* + •
  + [Reply](https://disqus.com/embed/comments/?base=default&f=foreach&t_i=node%2F152&t_u=https%3A%2F%2Fwww.foreach.be%2Fblog%2Fhow-select-fields-entities-forms&t_e=How%20to%20select%20fields%20from%20entities%20for%20forms&t_d=How%20to%20select%20fields%20from%20entities%20for%20forms&t_t=How%20to%20select%20fields%20from%20entities%20for%20forms&s_o=default)
  + •
  + Share ›
    - 

[](https://disqus.com/by/davidduymelinck/)

[**david duymelinck**](https://disqus.com/by/davidduymelinck/) [cforcloud](https://www.foreach.be/blog/how-select-fields-entities-forms#comment-2499828735) • [2 years ago](https://www.foreach.be/blog/how-select-fields-entities-forms#comment-2504227321)

If you have a CustomUser ContentEntity with a birth\_place field, you can use the method as follows.

$form[‘birth\_place’] = FieldUtils::getFieldRenderArray(‘birth\_place’, new CustomUser(), $form, $form\_state);

* + - •
    - [Reply](https://disqus.com/embed/comments/?base=default&f=foreach&t_i=node%2F152&t_u=https%3A%2F%2Fwww.foreach.be%2Fblog%2Fhow-select-fields-entities-forms&t_e=How%20to%20select%20fields%20from%20entities%20for%20forms&t_d=How%20to%20select%20fields%20from%20entities%20for%20forms&t_t=How%20to%20select%20fields%20from%20entities%20for%20forms&s_o=default)
    - •
    - Share ›
      * 
    - [−](https://disqus.com/embed/comments/?base=default&f=foreach&t_i=node%2F152&t_u=https%3A%2F%2Fwww.foreach.be%2Fblog%2Fhow-select-fields-entities-forms&t_e=How%20to%20select%20fields%20from%20entities%20for%20forms&t_d=How%20to%20select%20fields%20from%20entities%20for%20forms&t_t=How%20to%20select%20fields%20from%20entities%20for%20forms&s_o=default)

[](https://disqus.com/by/disqus_foreach/)

[**foreach**](https://disqus.com/by/disqus_foreach/) **Mod** [cforcloud](https://www.foreach.be/blog/how-select-fields-entities-forms#comment-2499828735) • [2 years ago](https://www.foreach.be/blog/how-select-fields-entities-forms#comment-2504095654)

Hello,

All the code is on this website. Feel free to copy-paste it if you want or need it. It doesn't seem enough to make a module for this

Drupal 8 - Entity API - Part 1

[Olya Kosovskaya](https://www.bigbluedoor.net/team-member/olya-kosovskaya) | 20 March 2017

<https://www.bigbluedoor.net/blog/drupal-8-entity-api-part-1>

Entities are awesome!

Leveraging the Entity API lets us create more lightweight and flexible solutions; they provide a unified way to work with different data units in Drupal - that’s why this post is about Entity API in D8. Entity API is already in core - we have all the necessary tools available inside the Admin UI to create and manage website data. However, the Entity API can also be easily extended to create custom entities.

How to create your own entity in D8?

Here is a very basic sample of creating a custom bbd\_team\_member entity. Enjoy! Create a new module called  bbd\_team\_member and bbd\_team\_member.info.yml within it.

name: BBD Team

description: Provides custom bbd\_team\_member entity

type: module

core: 8.x

Visit /admin/modules - you should already be able to install the BBD Team module. Create a src directory - this directory contains all the object-oriented code (classes, interfaces etc) in Drupal 8. Next, create an Entity directory within it and a file with the class for our entity src/Entity/BBDTeamMember.php with the following code:

<?php

/\*\*

\* Description of BBDTeamMember

\*

\* @author Olga

\*/

namespace Drupal\bbd\_team\_member\Entity;

use Drupal\Core\Entity\ContentEntityBase;

use Drupal\Core\Entity\EntityTypeInterface;

use Drupal\Core\Field\BaseFieldDefinition;

use Drupal\Core\Field\FieldStorageDefinitionInterface;

/\*\*

\* @ContentEntityType(

\* id = "bbd\_team\_member",

\* label = @Translation("Team Member"),

\* label\_singular = @Translation("team member"),

\* label\_plural = @Translation("team members"),

\* label\_count = @PluralTranslation(

\* singular = "@count team member",

\* plural = "@count team members"

\* ),

\* base\_table = "bbd\_team\_member",

\* entity\_keys = {

\* "id" = "id",

\* "uuid" = "uuid",

\* "label" = "title",

\* }

\* )

\*/

class BBDTeamMember extends *ContentEntityBase* {

}

Let's have fun and add some fields.

public static function baseFieldDefinitions(EntityTypeInterface $entity\_type) {

// Get field definitions for 'id' and 'uuid' from the parent.

$fields = parent::baseFieldDefinitions($entity\_type);

$fields['title'] = BaseFieldDefinition::create('string')

->setLabel(t('Title'))

->setRequired(TRUE)

->setDisplayOptions('view', [

'label' => 'hidden',

'weight' => 1,

])

->setDisplayOptions('form', ['weight' => 1]);

$fields['image'] = BaseFieldDefinition::create('image')

->setLabel(t('Image'))

->setRequired(TRUE)

->setDisplayOptions('view', [

'label' => 'hidden',

'weight' => 2,

])

->setDisplayOptions('form', ['weight' => 2]);

$fields['position'] = BaseFieldDefinition::create('string')

->setLabel(t('Position'))

->setRequired(TRUE)

->setDisplayOptions('view', [

'label' => 'inline',

'weight' => 3,

])

->setDisplayOptions('form', ['weight' => 3]);

return $fields;

}

Run the drush entity-updates. Right, now we can see the ‘bbd\_team\_member’ table in the database with the appropriate fields. Create an interface src/Entity/BBDTeamMemberInterface.php with the following code:

<?php

/\*\*

\* Description of BBDTeamMember Interface

\*

\* @author Olga

\*/

namespace Drupal\bbd\_team\_member\Entity;

use Drupal\Core\Entity\ContentEntityInterface;

interface BBDTeamMemberInterface extends *ContentEntityInterface* {

/\*\*

\* @return string

\*/

public function getTitle();

/\*\*

\* @param string $title

\*

\* @return $this

\*/

public function setTitle($title);

/\*\*

\* @return string

\*/

public function getPosition();

/\*\*

\* @param string $position

\*

\* @return $this

\*/

public function setPosition($position);

And update the class:

public function getTitle() {

return $this->get('title')->value;

}

public function setTitle($title) {

return $this->set('title', $title);

}

public function getPosition() {

return $this->get('position')->value;

}

public function setPosition($position) {

return $this->set('position', $position);

}

You’ll want to see some results of our coding in the admin panel, won’t you? To add a route we need to add the following lines to the annotation.

handlers = {

"view\_builder" = "Drupal\Core\Entity\EntityViewBuilder",

"route\_provider" = {

"html" = "Drupal\Core\Entity\Routing\DefaultHtmlRouteProvider",

},

"form" = {

"add" = "Drupal\Core\Entity\ContentEntityForm",

"delete" = "Drupal\Core\Entity\ContentEntityDeleteForm",

},

"views\_data" = "Drupal\views\EntityViewsData",

},

links = {

"canonical" = "/bbd-team-member/{bbd\_team\_member}",

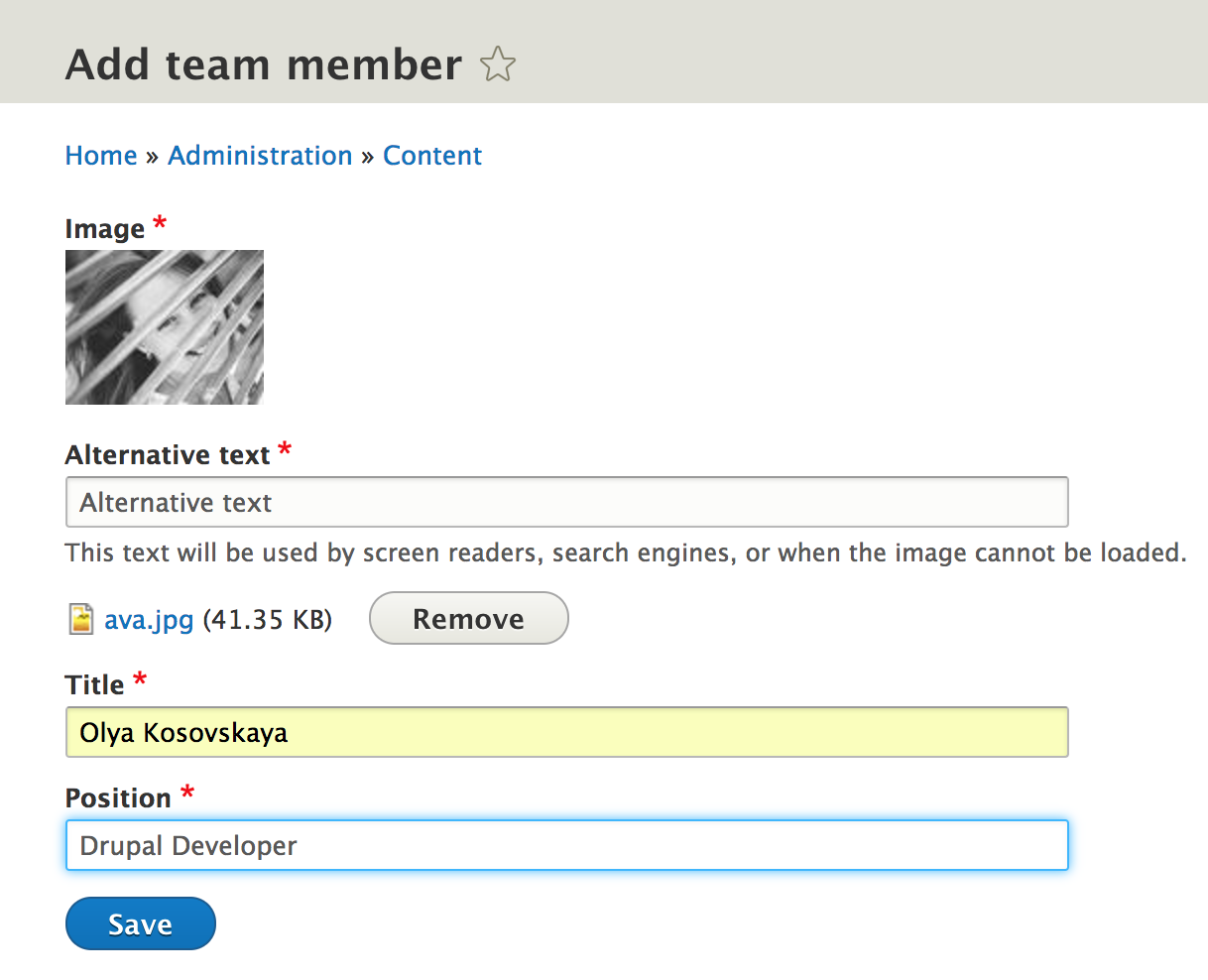
"add-form" = "/admin/content/bbd-team-member/add",

"delete-form" = "/admin/content/bbd-team-members/manage/{bbd\_team\_member}/delete",

},

admin\_permission = "administer bbd\_team\_members",

We can easily create a “bbd\_team\_member” from the admin panel, using admin/content/bbd-team-member/add. Don’t forget to run the drush entity-updates and drush cr all.



Quite nice, isn’t it? However, after the member has been created we get ‘Add team member’ page again. It’s a bit confusing - I want to see a notification that the member was created successfully and be able to see the created entity, just like nodes behave. Let’s do it right away.

Create a Form directory within crs, and add BBDTeamMemberForm.php with this code:

<?php

namespace Drupal\bbd\_team\_member\Form;

use Drupal\Core\Entity\ContentEntityForm;

use Drupal\Core\Form\FormStateInterface;

class BBDTeamMemberForm extends *ContentEntityForm* {

public function save(array $form, FormStateInterface $form\_state) {

parent::save($form, $form\_state);

$entity = $this->getEntity();

$entity\_type = $entity->getEntityType();

$arguments = [

'@entity\_type' => $entity\_type->getLowercaseLabel(),

'%entity' => $entity->label(),

'link' => $entity->toLink($this->t('View'), 'canonical')->toString(),

];

$this->logger($entity->getEntityTypeId())->notice('The @entity\_type %entity has been saved.', $arguments);

drupal\_set\_message($this->t('The @entity\_type %entity has been saved.', $arguments));

$form\_state->setRedirectUrl($entity->toUrl('canonical'));

}

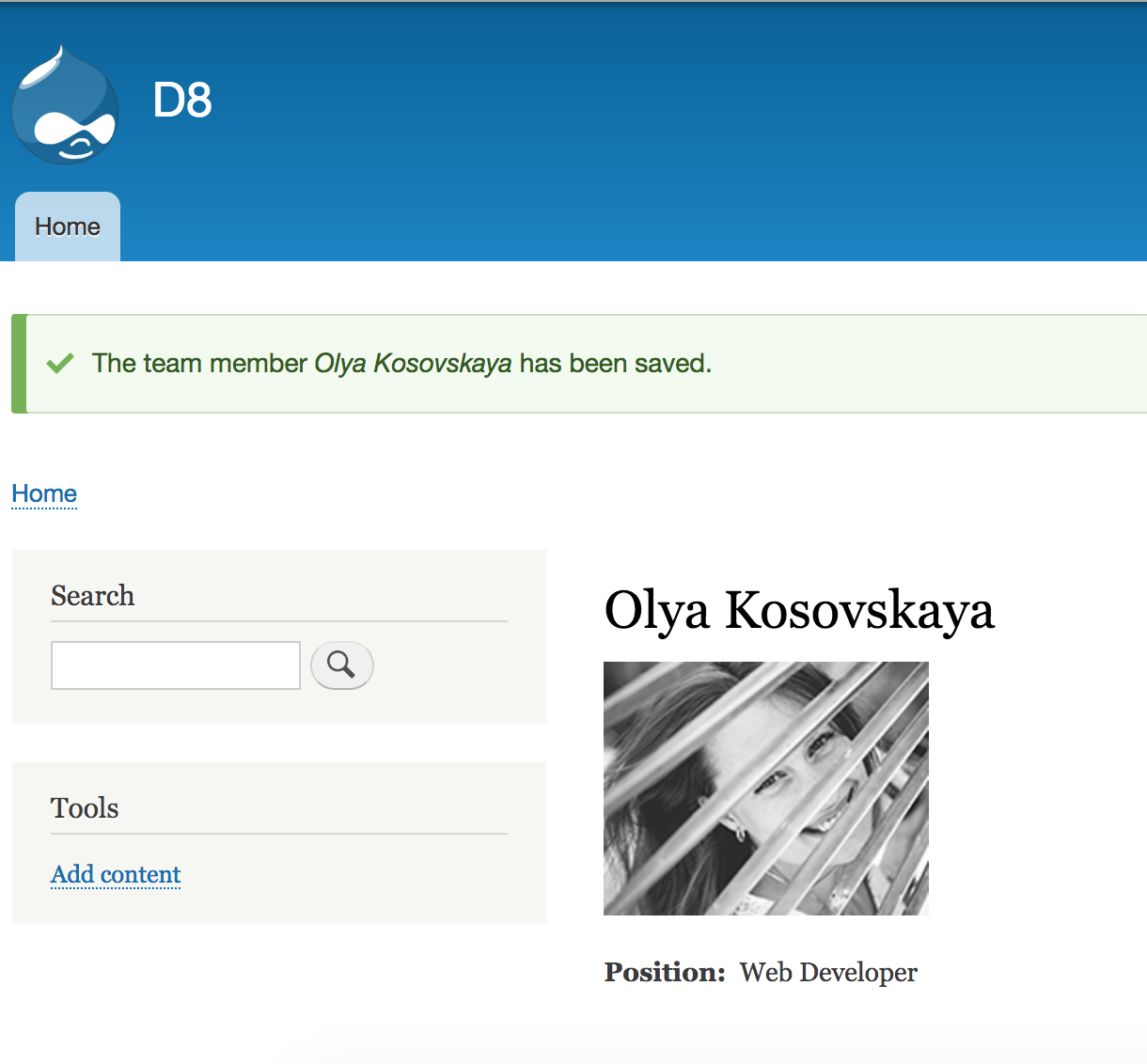
}

And change the announcement - we can already use our custom class:

"add" = "Drupal\bbd\_team\_member\Form\BBDTeamMemberForm",

"edit" = “Drupal\bbd\_team\_member\Form\BBDTeamMemberForm",

Cheers! That’s what I want to see!

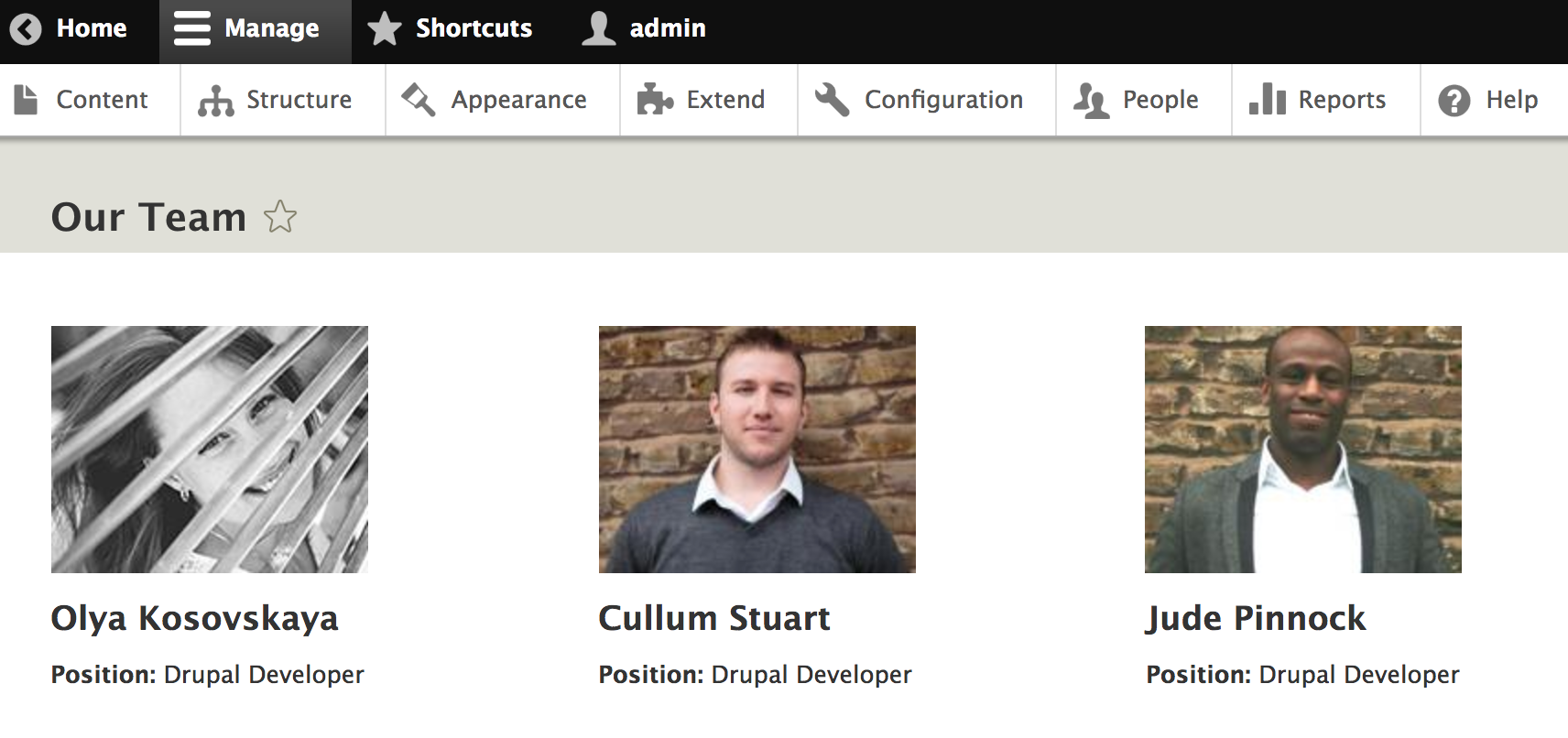


Let’s get a list of team members; fortunately Views are already part of Drupal 8 core. If you want to add views support to our entity just add this code to announcement:

"views\_data" = "Drupal\views\EntityViewsData",

Keep in mind, don’t forget about cache ;)

Now we can just create new view showing ‘Team Member’ entities and use the limitless possibilities of views :)



This is only a basic example. Entity API is more powerful than we may expect and lots of exciting features are still ahead. Stay tuned