[Drupal 8: Creating custom contact us block with form field](http://valuebound.com/resources/blog/drupal-8-creating-custom-contact-us-block-with-form-field)

1. Drupal 8: Creating custom contact us block with form field

Before we jump into the custom block writing. I would like to go through the blocker, which came to my learning. Hope it would be clear and simplest way to explain.

**Annotations based plugin discovery**  
Annotation is a meta data for our block. similar to hook\_block\_info did in drupal 7.

which carries machine readable name, admin label, category etc. drupal has Annotations-based plugin discovery mechanism which read the meta data and let the module knows that particular plugin is an implementation of block. Because of that it implement block plugin interface. it provides various method like build and block form, which drupal knows it exist & called up whenever  we require displaying content in our block.

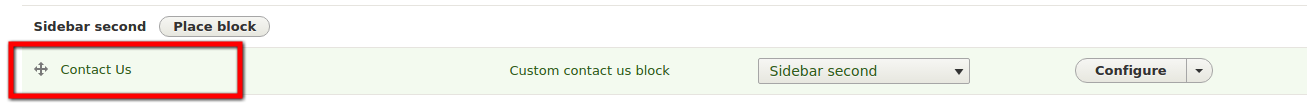
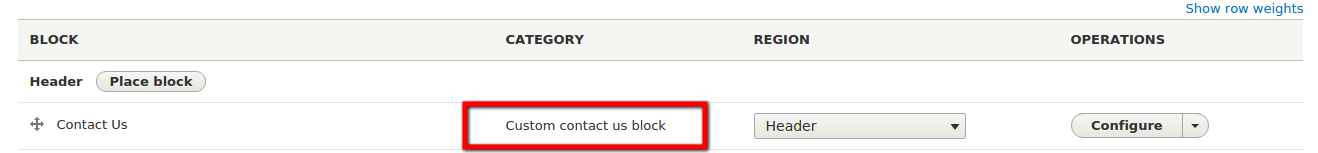
**Here's what the Drupal handbook has to say about plugins:**

Most of the plugins in Drupal 8 will use annotations to register themselves and describe their metadata. Some of the plugins types provided by core are:

1. Blocks (see \*/src/Plugin/Block/\* for many examples)
2. Field formatters, Field widgets (see \*/src/Plugin/Field/\* for many examples)
3. All Views plugins (see \*/src/Plugin/views/\* for many examples)

Conditions( used for Block visibility in core)



* The 'id' property in annotation defines unique & machine readable ID for block, and the name of the block to be seen by other code.
* The 'admin\_label' annotation defines the human readable name of the block that will be used when displaying your block in the administration UI.  
    
  
* The ‘category’ defines the package for your custom block.  
    
  

**Plugin API**

In Drupal 8 Plugin allow our module to provide additional functionality by reusing the code & implement the interface with different functionality. In Drupal every block consists of the same parts, label, content, and various settings related to visibility and cachability. How that label and content is generated is different varies from one module to another.

The plugin system has three base elements:

**Plugin Types:** Is the controlling class that defines how the plugins of this type will be discovered and instantiated. The type will describe the central purpose of all plugins of that type; e.g. blocks, etc.

**Plugin Discovery:** is the process of finding plugins within the available code base that qualify for use within this particular plugin type's use case.

**Plugin Factory:** The Factory is responsible for instantiating the specific plugin(s) chosen for a given use case.

In this Custom block i’m going to add some fields(org name/ org location/ email id/ phone/ address). i’m saving theses values to my block configuration page. and also validating one of the field (org name) should not be numeric.

|  |  |
| --- | --- |
|  | <?php |
|  | /\*\* |
|  | \* @file |
|  | \* Contains \Drupal\article\Plugin\Block\ContactBlock. |
|  | \*/ |
|  |  |
|  | namespace Drupal\article\Plugin\Block; |
|  |  |
|  | use Drupal\Core\Block\BlockBase; |
|  | use Drupal\Core\Form\FormStateInterface; |
|  |  |
|  |  |
|  | /\*\* |
|  | \* Provides a 'contact' block. |
|  | \* |
|  | \* @Block( |
|  | \* id = "contact\_block", |
|  | \* admin\_label = @Translation("Contact Us"), |
|  | \* category = @Translation("Custom contact us block") |
|  | \* ) |
|  | \*/ |
|  | class ContactBlock extends BlockBase { |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function blockForm($form, FormStateInterface $form\_state) { |
|  | $form = parent::blockForm($form, $form\_state); |
|  |  |
|  | // Retrieve existing configuration for this block. |
|  | $config = $this->getConfiguration(); |
|  |  |
|  | // Add a form field to the existing block configuration form. |
|  |  |
|  | $form['org\_name'] = array( |
|  | '#type' => 'textfield', |
|  | '#title' => t('Organization:'), |
|  | '#default\_value' => isset($config['org\_name'])? $config['org\_name'] : '', |
|  | ); |
|  | $form['org\_loca'] = array( |
|  | '#type' => 'textfield', |
|  | '#title' => t('Location:'), |
|  | '#default\_value' => isset($config['org\_loca'])? $config['org\_loca'] : '', |
|  | ); |
|  | $form['org\_mail'] = array( |
|  | '#type' => 'email', |
|  | '#title'=> t('Email ID:'), |
|  | '#default\_value' => isset($config['org\_mail'])? $config['org\_mail'] : '', |
|  | ); |
|  | $form['org\_phn'] = array( |
|  | '#type' => 'number', |
|  | '#title'=> t('Contact:'), |
|  | '#default\_value' => isset($config['org\_phn'])? $config['org\_phn'] : '', |
|  | ); |
|  | $form['org\_add'] = array( |
|  | '#type' => 'textfield', |
|  | '#title'=> t('Address:'), |
|  | '#default\_value' => isset($config['org\_add'])? $config['org\_add'] : '', |
|  | ); |
|  | return $form; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function blockSubmit($form, FormStateInterface $form\_state) { |
|  | // Save our custom settings when the form is submitted. |
|  | $this->setConfigurationValue('org\_name', $form\_state->getValue('org\_name')); |
|  | $this->setConfigurationValue('org\_loca', $form\_state->getValue('org\_loca')); |
|  | $this->setConfigurationValue('org\_mail', $form\_state->getValue('org\_mail')); |
|  | $this->setConfigurationValue('org\_phn', $form\_state->getValue('org\_phn')); |
|  | $this->setConfigurationValue('org\_add', $form\_state->getValue('org\_add')); |
|  | } |
|  |  |
|  |  |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function build() { |
|  | $config = $this->getConfiguration(); |
|  |  |
|  | $org\_name = isset($config['org\_name']) ? $config['org\_name'] : ''; |
|  | $org\_loca = isset($config['org\_loca']) ? $config['org\_loca'] : ''; |
|  | $org\_mail = isset($config['org\_mail']) ? $config['org\_mail'] : ''; |
|  | $org\_phn = isset($config['org\_phn']) ? $config['org\_phn'] : ''; |
|  | $org\_add = isset($config['org\_add']) ? $config['org\_add'] : ''; |
|  |  |
|  | return array( |
|  | '#markup' => $this->t('@org, @loc. Email id : @mail Phn: @phn. Address: @add', array('@add'=> $org\_add,'@phn'=> $org\_phn,'@mail'=> $org\_mail,'@org'=> $org\_name,'@loc' => $org\_loca)), |
|  | ); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function blockValidate($form, FormStateInterface $form\_state) { |
|  | $org\_name = $form\_state->getValue('org\_name'); |
|  |  |
|  | if (is\_numeric($org\_name)) { |
|  | drupal\_set\_message('needs to be an integer', 'error'); |
|  | $form\_state->setErrorByName('org\_name', t('Organization name should not be numeric')); |
|  | } |
|  | } |
|  | } |

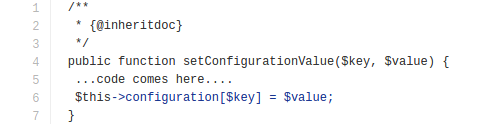
[**view raw**](https://gist.github.com/xaiwant/d799a6c5be37544d2e01/raw/611c9afd424ba56dc3ea37355df9d2ab1ae19e7e/ContactBlock.php)[**ContactBlock.php**](https://gist.github.com/xaiwant/d799a6c5be37544d2e01#file-contactblock-php) hosted with ❤ by **[GitHub](https://github.com/)**

Defines a base block implementation that most blocks plugins will extend.

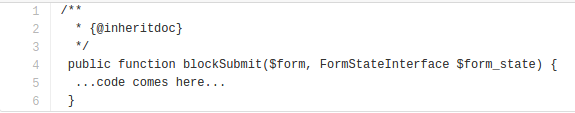
This class provides the generic block configuration form, default block settings, and handling for general user-defined block visibility settings.

**$form = parent::blockForm($form, $form\_state);**

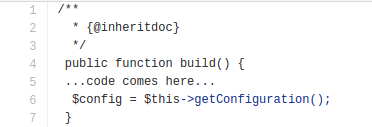
It creates a form array and get all the data from the parent method.



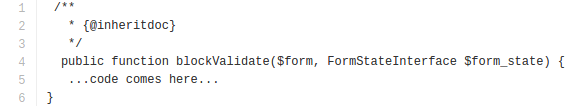
saves the block configuration data.



Process the block's submission handling if no errors occurred.



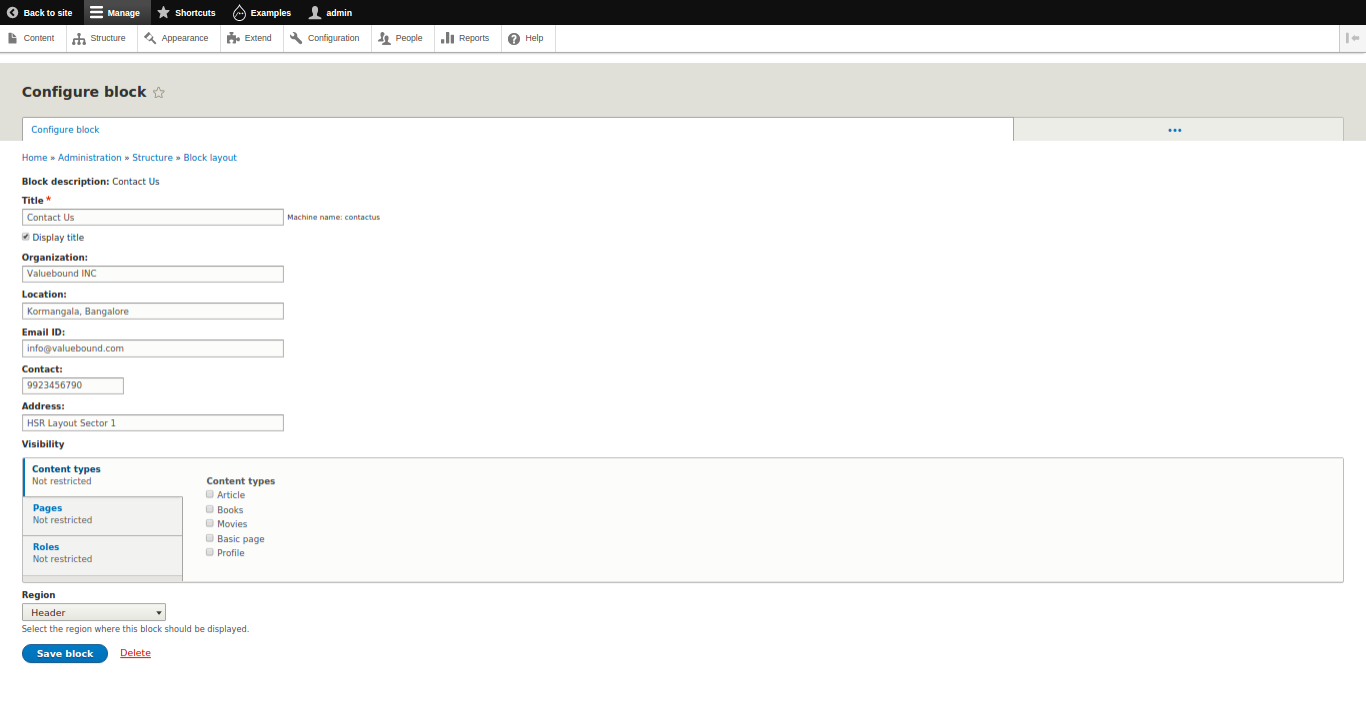
Get the block configuration.



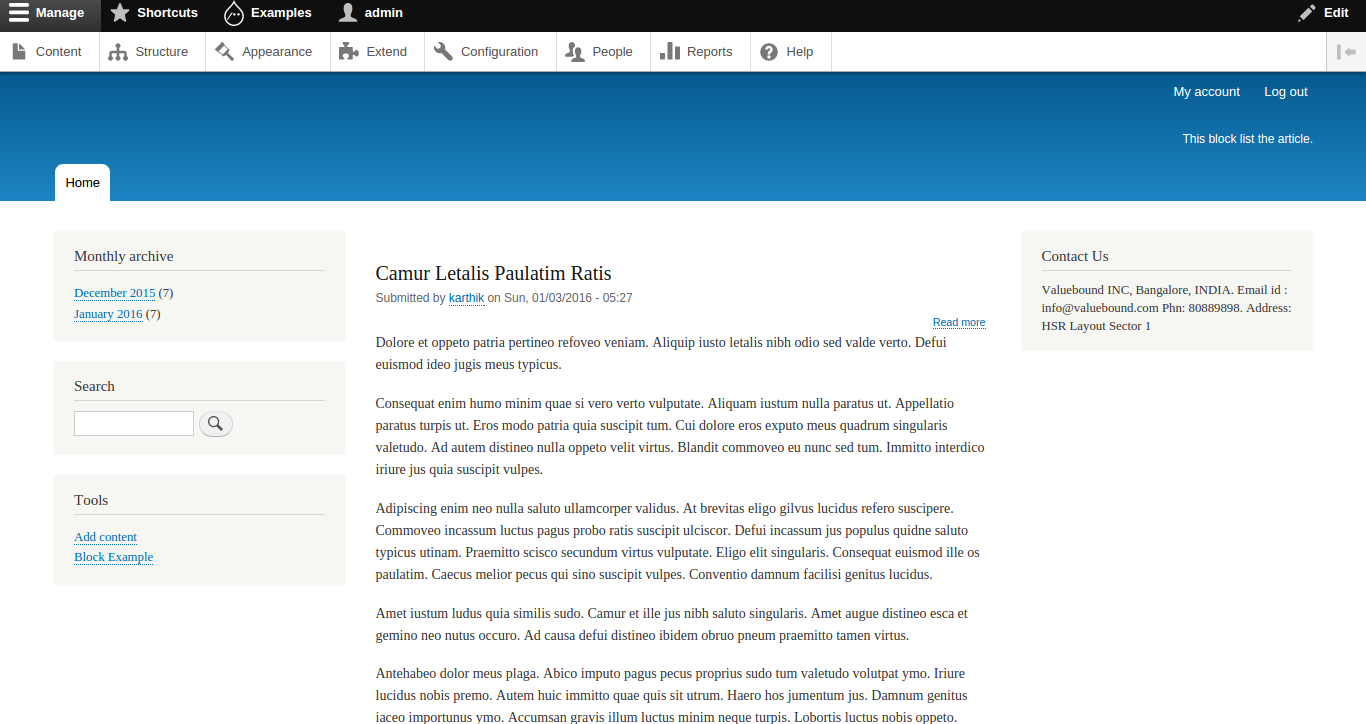
To add validation for a specific block type, override **BlockBase::blockValidate()**.

Once we have done with block build, block form, save configuration, get configuration, validate and submit. we will find our custom block to be listed in our block listing page.

Below is the Block form layout of our custom Block module



After filling up all the detail save the block. and our custom block start displaying in the specified region.



**Git Source code:** <https://github.com/xaiwant/D8custom-block-with-field-data>