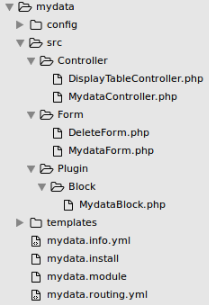
# [How to create custom Form with CRUD (Create, Delete, Update) operations in Drupal 8](https://www.valuebound.com/users/BSPavan)

Apr 19, 2017

Custom form with CRUD Operations is basically building the form with different fields like UID, Name, Mobile Number, Address, Email id etc. The CRUD operations for these fields is Deleting the value of the fields from the database or updating the existing value with new value, and printing the updated value.

**How To Create a Custom form with CRUD Operations?**

To create a [custom form](http://valuebound.com/resources/blog/step-by-step-method-to-create-a-custom-form-in-drupal-8) with CRUD Operations, we need to follow the following folder structure as shown in the image. To perform the same operation of crud in drupal we need to follow the following folder structure.



The data from the database will be displayed in the form of a table and the table will contain operations as a new column. The role of operations column is to perform edit(update) operation or the delete operation. The edit operation is used to update the present value into new values and delete operations is used to delete the entire row. The following files and procedures help us to create a custom form with CRUD operations. Create the files as shown in the image.

**mydata.info.yml:**  
The .info.yml files is most important part of Drupal Modules. The .info.yml file is a collection of static text file, used to define and configuring a module. First, we have to create info.yml file which contains metadata about the project like the name of the module, description, core, package.

|  |  |
| --- | --- |
|  | name: Form |
|  | description: A Form to enter user details. |
|  | core: 8.x |
|  | package: Custom |
|  | type: module |

**mydata.install:**

mydata.install helps to create a table in the database. Here we have a table called mydata in the database.hook\_schema helps to store the values into the database. In our custom form the crud database will be stored in mydata as the table name. If the value is stored in the database, it helps in performing CRUD operations,like edit or delete. Here we have the fields as name, id, mobile number, email, age gender. Id is autoincrement with the help of serial type.

|  |  |
| --- | --- |
|  | <?php |
|  | function mydata\_schema() { |
|  | $schema['mydata'] = array( |
|  | 'fields' => array( |
|  | 'id'=>array( |
|  | 'type'=>'serial', |
|  | 'not null' => TRUE, |
|  | ), |
|  | 'name'=>array( |
|  | 'type' => 'varchar', |
|  | 'length' => 40, |
|  | 'not null' => TRUE, |
|  | ), |
|  | 'mobilenumber'=>array( |
|  | 'type' => 'varchar', |
|  | 'length' => 40, |
|  | 'not null' => TRUE, |
|  | ), |
|  | 'email'=>array( |
|  | 'type' => 'varchar', |
|  | 'length' => 40, |
|  | 'not null' => TRUE, |
|  | ), |
|  | 'age'=>array( |
|  | 'type' => 'varchar', |
|  | 'length' => 25, |
|  | 'not null' => TRUE, |
|  | ), |
|  | 'gender'=>array( |
|  | 'type' => 'varchar', |
|  | 'length' => 40, |
|  | 'not null' => TRUE, |
|  | ), |
|  | 'website'=>array( |
|  | 'type' => 'varchar', |
|  | 'length' => 25, |
|  | 'not null' => TRUE, |
|  | ), |
|  | ), |
|  | 'primary key' => array('id'), |
|  | ); |
|  | return $schema; |
|  | } |

**mydata.routing.yml**  
mydata.routing.yml helps to transfer the function to the Controller to display of data in table format, to create form and placing that form in blocks. When '/mydata/hello/table' is passed to the url the Controller  DisplayTableController is called and the controller will display the data from database in table format. Similarly when the ‘/mydata/form/mydata’ is passed to the url the form containing credentials like name, id , age etc is shown to the user. The user has to enter the details.

|  |  |
| --- | --- |
|  | mydata.mydata\_controller\_display: |
|  | path: '/mydata' |
|  | defaults: |
|  | \_controller: '\Drupal\mydata\Controller\MydataController::display' |
|  | \_title: 'display' |
|  | requirements: |
|  | \_permission: 'access content' |
|  |  |
|  |  |
|  | mydata.mydata\_form: |
|  | path: '/mydata/form/mydata' |
|  | defaults: |
|  | \_form: '\Drupal\mydata\Form\MydataForm' |
|  | \_title: 'MydataForm' |
|  | requirements: |
|  | \_access: 'TRUE' |
|  |  |
|  | mydata.display\_table\_controller\_display: |
|  | path: '/mydata/hello/table' |
|  | defaults: |
|  | \_controller: '\Drupal\mydata\Controller\DisplayTableController::display' |
|  | \_title: 'display' |
|  | requirements: |
|  | \_permission: 'access content' |
|  |  |
|  | mydata.delete\_form: |
|  | path: '/mydata/form/delete/{cid}' |
|  | defaults: |
|  | \_form: '\Drupal\mydata\Form\DeleteForm' |
|  | \_title: 'DeleteForm' |
|  | requirements: |
|  | \_access: 'TRUE |

When the path  '/mydata/form/delete/{cid}' is passed to url DeleteForm.php is called. By this file you can delete the fields present in the database row by row.

**Controller**  
In the src/Controller create a file called   
**DisplayTableController :**

In this file, we will display the output in the table format. We have used the standard way of displaying the values of the row in table format.

[**Creating a custom form in a block in two steps in Drupal 8**](https://www.valuebound.com/resources/blog/creating-a-custom-form-in-a-block-in-two-steps-in-Drupal-8)

The code for displaying the value in table format. The $header\_table must be defined, like what are the column values that need to be displayed in the table format.

|  |  |
| --- | --- |
|  |  |
|  | <?php |
|  | namespace Drupal\mydata\Controller; |
|  | use Drupal\Core\Controller\ControllerBase; |
|  | use Drupal\Core\Database\Database; |
|  | use Drupal\Core\Url; |
|  |  |
|  | /\*\* |
|  | \* Class DisplayTableController. |
|  | \* |
|  | \* @package Drupal\mydata\Controller |
|  | \*/ |
|  | class DisplayTableController extends ControllerBase { |
|  |  |
|  | public function getContent() { |
|  | // First we'll tell the user what's going on. This content can be found |
|  | // in the twig template file: templates/description.html.twig. |
|  | // @todo: Set up links to create nodes and point to devel module. |
|  | $build = [ |
|  | 'description' => [ |
|  | '#theme' => 'mydata\_description', |
|  | '#description' => 'foo', |
|  | '#attributes' => [], |
|  | ], |
|  | ]; |
|  | return $build; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* Display. |
|  | \* |
|  | \* @return string |
|  | \* Return Hello string. |
|  | \*/ |
|  | public function display() { |
|  | /\*\*return [ |
|  | '#type' => 'markup', |
|  | '#markup' => $this->t('Implement method: display with parameter(s): $name'), |
|  | ];\*/ |
|  | //create table header |
|  | $header\_table = array( |
|  | 'id'=> t('SrNo'), |
|  | 'name' => t('Name'), |
|  | 'mobilenumber' => t('MobileNumber'), |
|  | //'email'=>t('Email'), |
|  | 'age' => t('Age'), |
|  | 'gender' => t('Gender'), |
|  | //'website' => t('Web site'), |
|  | 'opt' => t('operations'), |
|  | 'opt1' => t('operations'), |
|  | ); |
|  |  |
|  | //select records from table |
|  | $query = \Drupal::database()->select('mydata', 'm'); |
|  | $query->fields('m', ['id','name','mobilenumber','email','age','gender','website']); |
|  | $results = $query->execute()->fetchAll(); |
|  | $rows=array(); |
|  | foreach($results as $data){ |
|  | $delete = Url::fromUserInput('/mydata/form/delete/'.$data->id); |
|  | $edit = Url::fromUserInput('/mydata/form/mydata?num='.$data->id); |
|  |  |
|  | //print the data from table |
|  | $rows[] = array( |
|  | 'id' =>$data->id, |
|  | 'name' => $data->name, |
|  | 'mobilenumber' => $data->mobilenumber, |
|  | //'email' => $data->email, |
|  | 'age' => $data->age, |
|  | 'gender' => $data->gender, |
|  | //'website' => $data->website, |
|  |  |
|  | \Drupal::l('Delete', $delete), |
|  | \Drupal::l('Edit', $edit), |
|  | ); |
|  | } |
|  | //display data in site |
|  | $form['table'] = [ |
|  | '#type' => 'table', |
|  | '#header' => $header\_table, |
|  | '#rows' => $rows, |
|  | '#empty' => t('No users found'), |
|  | ]; |
|  | return $form; |
|  | } |

**Form**  
In the src create a folder called Form, In the form folder, we will create a file which will help to create form fields like Textfield, checkbox, select list, number etc..

**MydataForm:**

In this file, we will create the form fields for  UID, Name, Mobile Number, Email, age etc. For more info on form API, visit <https://www.drupal.org/docs/8/api/form-api/introduction-to-form-api> . The complete code for the Mydata form is shown below

|  |  |
| --- | --- |
|  | <?php |
|  | namespace Drupal\mydata\Form; |
|  | use Drupal\Core\Form\FormBase; |
|  | use Drupal\Core\Form\FormStateInterface; |
|  | use Drupal\Core\Database\Database; |
|  | use Symfony\Component\HttpFoundation\RedirectResponse; |
|  |  |
|  | /\*\* |
|  | \* Class MydataForm. |
|  | \* |
|  | \* @package Drupal\mydata\Form |
|  | \*/ |
|  | class MydataForm extends FormBase { |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function getFormId() { |
|  | return 'mydata\_form'; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function buildForm(array $form, FormStateInterface $form\_state) { |
|  |  |
|  | $conn = Database::getConnection(); |
|  | $record = array(); |
|  | if (isset($\_GET['num'])) { |
|  | $query = $conn->select('mydata', 'm') |
|  | ->condition('id', $\_GET['num']) |
|  | ->fields('m'); |
|  | $record = $query->execute()->fetchAssoc(); |
|  | } |
|  | $form['candidate\_name'] = array( |
|  | '#type' => 'textfield', |
|  | '#title' => t('Candidate Name:'), |
|  | '#required' => TRUE, |
|  | //'#default\_values' => array(array('id')), |
|  | '#default\_value' => (isset($record['name']) && $\_GET['num']) ? $record['name']:'', |
|  | ); |
|  | $form['mobile\_number'] = array( |
|  | '#type' => 'textfield', |
|  | '#title' => t('Mobile Number:'), |
|  | '#default\_value' => (isset($record['mobilenumber']) && $\_GET['num']) ? $record['mobilenumber']:'', |
|  | ); |
|  | $form['candidate\_mail'] = array( |
|  | '#type' => 'email', |
|  | '#title' => t('Email ID:'), |
|  | '#required' => TRUE, |
|  | '#default\_value' => (isset($record['email']) && $\_GET['num']) ? $record['email']:'', |
|  | ); |
|  |  |
|  | $form['candidate\_age'] = array ( |
|  | '#type' => 'textfield', |
|  | '#title' => t('AGE'), |
|  | '#required' => TRUE, |
|  | '#default\_value' => (isset($record['age']) && $\_GET['num']) ? $record['age']:'', |
|  | ); |
|  |  |
|  | $form['candidate\_gender'] = array ( |
|  | '#type' => 'select', |
|  | '#title' => ('Gender'), |
|  | '#options' => array( |
|  | 'Female' => t('Female'), |
|  | 'male' => t('Male'), |
|  | '#default\_value' => (isset($record['gender']) && $\_GET['num']) ? $record['gender']:'', |
|  | ), |
|  | ); |
|  | $form['web\_site'] = array ( |
|  | '#type' => 'textfield', |
|  | '#title' => t('web site'), |
|  | '#default\_value' => (isset($record['website']) && $\_GET['num']) ? $record['website']:'', |
|  | ); |
|  | $form['submit'] = [ |
|  | '#type' => 'submit', |
|  | '#value' => 'save', |
|  | //'#value' => t('Submit'), |
|  | ]; |
|  | return $form; |
|  | } |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function validateForm(array &$form, FormStateInterface $form\_state) { |
|  | $name = $form\_state->getValue('candidate\_name'); |
|  | if(preg\_match('/[^A-Za-z]/', $name)) { |
|  | $form\_state->setErrorByName('candidate\_name', $this->t('your name must in characters without space')); |
|  | } |
|  | if (!intval($form\_state->getValue('candidate\_age'))) { |
|  | $form\_state->setErrorByName('candidate\_age', $this->t('Age needs to be a number')); |
|  | } |
|  | /\* $number = $form\_state->getValue('candidate\_age'); |
|  | if(!preg\_match('/[^A-Za-z]/', $number)) { |
|  | $form\_state->setErrorByName('candidate\_age', $this->t('your age must in numbers')); |
|  | }\*/ |
|  | if (strlen($form\_state->getValue('mobile\_number')) < 10 ) { |
|  | $form\_state->setErrorByName('mobile\_number', $this->t('your mobile number must in 10 digits')); |
|  | } |
|  | parent::validateForm($form, $form\_state); |
|  | } |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function submitForm(array &$form, FormStateInterface $form\_state) { |
|  |  |
|  | $field=$form\_state->getValues(); |
|  | $name=$field['candidate\_name']; |
|  | //echo "$name"; |
|  | $number=$field['mobile\_number']; |
|  | $email=$field['candidate\_mail']; |
|  | $age=$field['candidate\_age']; |
|  | $gender=$field['candidate\_gender']; |
|  | $website=$field['web\_site']; |
|  | if (isset($\_GET['num'])) { |
|  | $field = array( |
|  | 'name' => $name, |
|  | 'mobilenumber' => $number, |
|  | 'email' => $email, |
|  | 'age' => $age, |
|  | 'gender' => $gender, |
|  | 'website' => $website, |
|  | ); |
|  | $query = \Drupal::database(); |
|  | $query->update('mydata') |
|  | ->fields($field) |
|  | ->condition('id', $\_GET['num']) |
|  | ->execute(); |
|  | drupal\_set\_message("succesfully updated"); |
|  | $form\_state->setRedirect('mydata.display\_table\_controller\_display'); |
|  | } |
|  | else |
|  | { |
|  | $field = array( |
|  | 'name' => $name, |
|  | 'mobilenumber' => $number, |
|  | 'email' => $email, |
|  | 'age' => $age, |
|  | 'gender' => $gender, |
|  | 'website' => $website, |
|  | ); |
|  | $query = \Drupal::database(); |
|  | $query ->insert('mydata') |
|  | ->fields($field) |
|  | ->execute(); |
|  | drupal\_set\_message("succesfully saved"); |
|  | $response = new RedirectResponse("/mydata/hello/table"); |
|  | $response->send(); |
|  | } |
|  | } |
|  | } |

In the above file we have performed both Insert and update operations. If the user clicks the edit operations then it will call the update functions or it will be insert operation.We have used three methods to perform the operation

* Buildform: It is the method which is used to construct the form with different types like text field email, select checkbox etc
* ValidateForm : In this method we provide validation conditions for the form, that has to be validated before the form values get submitted
* Submit form : In this method the values will be submitted to the database so, it can be retrieved.

**2.  DeleteForm.php :**

In this file we will perform delete operation. I have followed certain standards to delete operation, we can also user db\_delete operation. The complete code for the delete operation is shown below

|  |  |
| --- | --- |
|  | <?php |
|  | namespace Drupal\mydata\Form; |
|  | use Drupal\Core\Form\FormBase; |
|  | use Drupal\Core\Form\FormStateInterface; |
|  | use Drupal\Core\Form\ConfirmFormBase; |
|  | use Drupal\Core\Url; |
|  | use Drupal\Core\Render\Element; |
|  | /\*\* |
|  | \* Class DeleteForm. |
|  | \* |
|  | \* @package Drupal\mydata\Form |
|  | \*/ |
|  | class DeleteForm extends ConfirmFormBase { |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function getFormId() { |
|  | return 'delete\_form'; |
|  | } |
|  | public $cid; |
|  |  |
|  | <?php |
|  | namespace Drupal\mydata\Form; |
|  | use Drupal\Core\Form\FormBase; |
|  | use Drupal\Core\Form\FormStateInterface; |
|  | use Drupal\Core\Form\ConfirmFormBase; |
|  | use Drupal\Core\Url; |
|  | use Drupal\Core\Render\Element; |
|  | /\*\* |
|  | \* Class DeleteForm. |
|  | \* |
|  | \* @package Drupal\mydata\Form |
|  | \*/ |
|  | class DeleteForm extends ConfirmFormBase { |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function getFormId() { |
|  | return 'delete\_form'; |
|  | } |
|  | public $cid; |
|  | public function getQuestion() { |
|  | return t('Do you want to delete %cid?', array('%cid' => $this->cid)); |
|  | } |
|  | public function getCancelUrl() { |
|  | return new Url('mydata.display\_table\_controller\_display'); |
|  | } |
|  | public function getDescription() { |
|  | return t('Only do this if you are sure!'); |
|  | } |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function getConfirmText() { |
|  | return t('Delete it!'); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function getCancelText() { |
|  | return t('Cancel'); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function buildForm(array $form, FormStateInterface $form\_state, $cid = NULL) { |
|  |  |
|  | $this->id = $cid; |
|  | return parent::buildForm($form, $form\_state); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function validateForm(array &$form, FormStateInterface $form\_state) { |
|  | parent::validateForm($form, $form\_state); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function submitForm(array &$form, FormStateInterface $form\_state) { |
|  | $query = \Drupal::database(); |
|  | $query->delete('mydata') |
|  | ->condition('id',$this->id) |
|  | ->execute(); |
|  | drupal\_set\_message("succesfully deleted"); |
|  | $form\_state->setRedirect('mydata.display\_table\_controller\_display'); |
|  | } |
|  | } |

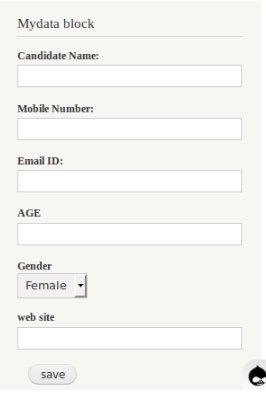
Note : cid is the current value form the url and cid is stored in this->id(current id).

**BLOCK**

**1 MydataBlock:**

This block helps to place the form fields anywhere in the UI. Placing the block can be done by going to admin/Structure/block and placing the block in any of the regions. The code of MydataBlock is shown below

|  |  |
| --- | --- |
|  |  |
|  | <?php |
|  | namespace Drupal\mydata\Plugin\Block; |
|  | use Drupal\Core\Block\BlockBase; |
|  | /\*\* |
|  | \* Provides a 'MydataBlock' block. |
|  | \* |
|  | \* @Block( |
|  | \* id = "mydata\_block", |
|  | \* admin\_label = @Translation("Mydata block"), |
|  | \* ) |
|  | \*/ |
|  | class MydataBlock extends BlockBase { |
|  | /\*\* |
|  | \* {@inheritdoc} |
|  | \*/ |
|  | public function build() { |
|  | ////$build = []; |
|  | //$build['mydata\_block']['#markup'] = 'Implement MydataBlock.'; |
|  |  |
|  | $form = \Drupal::formBuilder()->getForm('Drupal\mydata\Form\MydataForm'); |
|  |  |
|  | return $form; |
|  | } |
|  | } |

The output of the above module is shown below  


The above screenshot displays all the form fields that are to be entered by the user .It is a block that is placed

**CrudOperationPage(output page):**

It display the option for delete or edit

