GN

Blog Guides Snippets Contact About

Display a modal on page load in Drupal

FEB 23, 2022 DRUPAL

Displaying a modal dialog containing some text on page load sounds like a simple task, but I think it's quite messy to get it to work. Let's explore one possible way to do it using Javascript and AJAX to open a popup window.

To display a modal dialog on page load we first have to define and attach a library to every user-facing page. Attaching the library is easy, we can use the *hook_page_attachments_alter()* hook for that:

```
/**
 * Implements hook_page_attachments_alter().
 */
function MY_MODULE_page_attachments_alter(array &$attachments) {
  $attachments['#attached']['library'][] = 'MY_MODULE/my_modal';
}
```

Now let's define the library in the *MY_MODULE.libraries.yml* file. Nothing fancy here, just a Javascript file and some dependencies:

```
my_modal:
  version: VERSION
  js:
    js/my-modal.js: {}
  dependencies:
    - core/jquery
    - core/drupal
    - core/drupal.dialog.ajax
```

The *core/drupal.dialog.ajax* library is necessary to render the modal dialogs so don't forget to include it.

The next step is to write Javascript logic that will open our popup on page load. You can add some conditional logic so that popup is not displayed on every page load, but in our case, we'll keep things simple and show the modal on every page load.

So, let's create a new file *js/my-modal.js* and add the following code:

```
(function ($, Drupal) {
  'use strict';

  Drupal.behaviors.myModal = {
    attach: function(context) {
      $(context).find('body')
        .once('my-modal')
        .each(function () {
          var ajaxSettings = {
            url: '/modals/my-modal'
          };
          var myAjaxObject = Drupal.ajax(ajaxSettings);
          myAjaxObject.execute();
        });
    }
  };

})(jQuery, Drupal);
```

As you can see our Javascript code is opening */modals/my-modal* URL, so let's define that URL in our *MY_MODULE.routing.yml* file. It's a simple route that calls the appropriate callback method to open the modal form:

```
MY_MODULE.my_modal:
  path: 'modals/my-modal'
  defaults:
    _controller: '\Drupal\MY_MODULE\Controller\PageController::openMyModal'
    _title: 'My Modal'
  requirements:
    _permission: 'access content'
```

Inside the callback method, we are using the *AjaxResponse* to add a command for opening the modal dialog. The modal will display our custom form, so we are using the Form builder service to build the form.

The *$modal_form* variable is then passed as the second argument to the *OpenModalDialogCommand* class constructor. The first argument is the title of the

modal window, and the third argument is dialog options (any jQuery UI dialog option).

```
<?php

namespace Drupal\MY_MODULE\Controller;

use Drupal\Core\Ajax\AjaxResponse;
use Drupal\Core\Ajax\OpenModalDialogCommand;
use Drupal\Core\Controller\ControllerBase;

class PageController extends ControllerBase {

  public function openMyModal() {
    $response = new AjaxResponse();
    $modal_form = $this->formBuilder()->getForm('Drupal\MY_MODULE\Form\MyModalForm');
    $options = [
      'width' => '75%',
    ];
    $response->addCommand(new OpenModalDialogCommand('My Modal', $modal_form, $options));
    return $response;
  }

}
```

And the last step is to define our custom form. Again nothing fancy here, just a simple form with a text and a button:

```
<?php

namespace Drupal\MY_MODULE\Form;

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

class MyModalForm extends FormBase {

  public function getFormId() {
    return 'my_modal_form';
  }

  public function buildForm(array $form, FormStateInterface $form_state) {
    $form['markup'] = [
      '#type' => 'markup',
      '#markup' => $this->t('This is my modal.'),
```

```
      ];

      $form['submit'] = [
        '#type' => 'submit',
        '#value' => $this->t('OK'),
      ];

      return $form;
    }

    public function submitForm(array &$form, FormStateInterface $form_state) {
      $this->messenger()->addMessage('Hello World');
      $form_state->setRedirect('<front>');
    }

  }
```

That's pretty much it. If you followed along then flush the caches and you should see your modal window in all its glory. Now go on and add some modals to annoy the heck out of your users. And if you need help with anything let me know.

#drupal #modal

Further reading

- Integrate Facebook Pixel and Drupal Webform via Google Tag Manager
- Programmatically create a menu link for a node
- Set date field programmatically