Introductory Drupal 8 routes and controllers example

This is a quick introduction to routes and controllers in Drupal 8. If you're only modifying or extending existing functionality, you may not need to know about routes. However, if you want to expose content or functionality on your own URIs on a site, routing is an important part of writing your module. This can help provide functionality at specific URIs of a website or just modify or augment existing functionality.

Routing in routing.yml files

When you named your module's .info.yml file, it was of the form module\_name.info.yml. That module\_name is registered with core Drupal as the "Machine name" of your module. You can open up the details of your module in the module list to see it. Drupal then loads the routing file that is of the form module\_name.routing.yml that it will use to define how Drupal will behave when a specific path is encountered. For example, if we had a file example.info.yml that had defined our module, we would then have a file example.routing.yml:

example.content:

path: '/example'

defaults:

\_controller: '\Drupal\example\Controller\ExampleController::content'

\_title: 'Hello World'

requirements:

\_permission: 'access content'

This tells Drupal 8 that a route named 'example.content' (named with the module name as prefix) exists and is bound to the URI '/example' on the site. When that path is accessed, the 'access content' permission is checked on the accessing user and, if access is granted, the ExampleController::content method is invoked and a page is displayed with the title 'Hello World'.

Note that Drupal's autoloading mechanism loaded the file at modules/example/src/Controller/ExampleController.php because when it encountered the machine name "example" in the string "\_controller", it looks in the src folder of that module. If your files are not autoloading, you may want to recheck the registered machine name of your module.

If you built Drupal 7 modules before, this is pretty similar to how Drupal 7 associated page callbacks to paths in hook\_menu(). (However, the routing system is not responsible for managing tabs, action links and contextual links).

Building a page controller

Drupal 8 uses the Symfony HTTP Kernel, a system which gets the request and asks other systems to produce the requested output (a response object) and sends the response back to the client. The output is generated by a piece of code called the controller. In theory the controller can be either a pure PHP 4 function, a method on an object or even an anonymous function. [See more about the Routing system](https://www.drupal.org/node/2046371)

The second part of creating a page that outputs our content is to put the **page controller** in place. This could be a PHP 4 style function, but best practice in Drupal 8 is to use a controller class. This class is in a file named exactly after the controller provided in the routing file, ExampleController.php

For the above route the class needs to be placed in example/src/Controller directory and with name ExampleController.php. So the full path will look like, example/src/Controller/ExampleController.php.

namespace Drupal\example\Controller;

use Drupal\Core\Controller\ControllerBase;

/\*\*

\* An example controller.

\*/

class ExampleController extends ControllerBase {

/\*\*

\* {@inheritdoc}

\*/

public function content() {

$build = array(

'#type' => 'markup',

'#markup' => t('Hello World!'),

);

return $build;

}

}

With only the routing file and the page controller, we made a page available on our site at /example that outputs 'Hello World!' on a page titled 'Hello World'.