Drupal Book   ☰

Routes and Controllers

views 114

Overview

Route

A route connects a URL path to a controller. In hello_world.routing.yml (e.g. in modules/custom/hello_world/hello_world.routing.yml) The path /hello maps to the controller HelloWorldController and the member function: helloWorld(). When a user visits /hello, Drupal checks to see that the user has access content permission and the helloWorld() function is executed.

```
hello_world.hello:
  path: '/hello'
  defaults:
    _controller: '\Drupal\hello_world\Controller\HelloWorldController::helloWorld'
    _title: 'Our first route'
  requirements:
    _permission: 'access content'
```

Controller

This is the PHP function that takes info from the HTTP request and constructs and returns an HTTP response (as a Symfony ResponseObject). The controller contains your logic to render the content of the page.

The controller will usually return a render array but they can return an HTML page, an XML document, a serialized JSON array, an image, a redirect, a 404 error or almost anything else.

A simple render array looks like this:

```
return [
  '#markup' => 'blah',
]
```

Connecting to a twig template

Most often, you will have a twig template connected to your controller. You do this by a combination of a `#theme` element in the render array and a `hook_theme` function in a `.module` file.

In the example below, the controller returns a large render array and the theme is identified as `abc_teks_srp__correlation_voting`.

```
return [
  '#theme' => 'abc_teks_srp__correlation_voting',
  '#content' => $content,
  '#breadcrumbs' => $breadcrumbs,
  '#management_links' => $management_links,
  '#correlation' => $correlation_info,
  '#citations' => $citations,
];
```

In a module file, there is a hook_theme function which corresponds to the `abc_teks_srp_theme` and identifies the template name as `abc-teks-srp-correlation-voting`. Here is the significant part of the `hook_theme()` function

```
/**
 * Implements hook_theme().
 */
function abc_teks_srp_theme() {
  $variables = [
    'abc_teks_srp' => [
      'render element' => 'children',
    ],
    'abc_teks_srp__correlation_voting' => [
      'variables' => [
        'content' => NULL,
        'breadcrumbs' => NULL,
        'management_links' => NULL,
        'correlation' => NULL,
        'citations' => NULL,
      ],
      'template' => 'abc-teks-srp--correlation-voting',
    ],
```

The template will therefore be `abc-teks-srp--correlation.voting.yml`

Simple page without arguments

This route is for a page with no arguments/parameters.

In the file page_example.routing.yml (e.g. `web/modules/contrib/examples/page_example/page_example.routing.yml` and the controller is at web/modules/contrib/examples/page_example/src/Controller/PageExampleController.php

```
# If the user accesses https://example.com/?q=examples/page-example/simple,
# or https://example.com/examples/page-example/simple,
# the routing system will look for a route with that path.
# In this case it will find a match, and execute the _controller callback.
# Access to this path requires "access simple page" permission.
page_example_simple:
  path: 'examples/page-example/simple'
  defaults:
    _controller: '\Drupal\page_example\Controller\PageExampleController::simple'
    _title: 'Simple - no arguments'
  requirements:
    _permission: 'access simple page'
```

Page with arguments

From web/modules/contrib/examples/page_example/page_example.routing.yml {first}/{second} are the arguments.

```
# Since the parameters are passed to the function after the match, the
# function can do additional checking or make use of them before executing
# the callback function. The placeholder names "first" and "second" are
# arbitrary but must match the variable names in the callback method, e.g.
# "$first" and "$second".
page_example_arguments:
  path: 'examples/page-example/arguments/{first}/{second}'
  defaults:
    _controller: '\Drupal\page_example\Controller\PageExampleController::arguments'
  requirements:
    _permission: 'access arguments page'
```

Simple form

From web/modules/custom/rsvp/rsvp.routing.yml. This route will cause Drupal to load the form: RSVPForm.php so the user can fill it out.

```
rsvp.form:
  path: '/rsvplist'
  defaults:
    _form: 'Drupal\rsvp\Form\RSVPForm'
    _title: 'RSVP to this Event'
  requirements:
    _permission: 'view rsvplist'
```

Admin form (or settings form)

From web/modules/custom/rsvp/rsvp.routing.yml this route loads the admin or settings form RSVPConfigurationForm.

```
rsvp.admin_settings:
  path: '/admin/config/content/rsvp'
  defaults:
    _form: 'Drupal\rsvp\Form\RSVPConfigurationForm'
    _title: 'RSVP Configuration Settings'
  requirements:
    _permission: 'administer rsvplist'
  options:
    _admin_route: TRUE
```

Routing permissions

These are defined in your module.permissions.yml e.g. rsvp.permissions.yml. If you add this file to a module, a cache clear will cause the new permissions to appear on the permissions page.

This requires the user to be logged in to access this route:

```yaml
requirements:
  _user_is_logged_in: 'TRUE'
```

To skip permissions, set _access to TRUE like this:

```yaml
requirements:
  _access: 'TRUE'
```

A specific permission

To specify a particular permission, use the following. Note. Case is critical!

```yaml
requirements:
  _permission: 'administer rsvplist'
```

Multiple permissions

Drupal allows stacking permissions with the plus(+) sign. Note the + sign means OR. e.g.

```yaml
requirements:
  _permission: 'vote on own squishy item+manage squishy process'
```

Set the page title dynamically

This code shows how to set the title statically in the `module.routing.yml` file, as well as how to call a function like `getTitle()` to return it so it can be dynamically generated:

```yaml
org_onions_summary:
  path: 'onions/{term_id}'
  defaults:
    _controller: '\Drupal\org_onions\Controller\OnionsController::buildOnionsSummary'
#    Static Title
#    _title: 'Opinions Summary'
#    Dynamic Title
    _title_callback: '\Drupal\org_onions\Controller\OnionsController::getTitle'
  requirements:
    _permission: 'access content'
```

In your controller, add the function `getTitle()`. This function can actually be called whatever you like.

```php
/**
 * Returns a page title.
 */
public function getTitle() {
  $current_path = \Drupal::service('path.current')->getPath();
  $path_args = explode('/', $current_path);
  $boss_name = $path_args[2];
  $boss_name = ucwords(str_replace("-", " ", $boss_name));

  $config = \Drupal::config('system.site');
  $site_name = $config->get('name');
  return  $boss_name . ' Onions | ' . $site_name;

  //or
  return  $boss_name . ' onions | ' . \Drupal::config('system.site')->get('name');
}
```

Disable caching on a route

This will cause Drupal to rebuild the page internally on each page load but won't stop browsers or CDN's from caching. The line: no_cache: TRUE is all you need to disable caching for this route.

```yaml
requirements:
  _permission: 'access content'
options:
  no_cache: TRUE
```

Generate route and controller with Drush

Drush has the ability to generate code to start you off. Use drush generate module and or drush generate controller to get a nice starting point for you to write your own controllers.

For more, on generating controllers see https://www.drush.org/latest/generators/controller/

This is what it looks like to generate a controller:

```
$ drush generate controller

 Welcome to controller generator!
––––––––––––––––––––––––––––––––––

 Module machine name [web]:
 ➤ general

 Class [GeneralController]:
 ➤ ExampleController

 Would you like to inject dependencies? [No]:
 ➤

 Would you like to create a route for this controller? [Yes]:
 ➤

 Route name [general.example]:
 ➤ general.book_example

 Route path [/general/example]:
 ➤ /general/book_example

 Route title [Example]:
 ➤ Book Example

 Route permission [access content]:
 ➤

 The following directories and files have been created or updated:
 –––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
 • /Users/selwyn/Sites/d9book2/web/modules/custom/general/general.routing.yml
 • /Users/selwyn/Sites/d9book2/web/modules/custom/general/src/Controller/ExampleController.php
```

The file general.routing.yml will then contain:

```yaml
general.book_example:
  path: '/general/book_example'
  defaults:
    _title: 'Book Example'
    _controller: '\Drupal\general\Controller\ExampleController::build'
  requirements:
    _permission: 'access content'
```

The `ExampleController.php` file has these contents:

```php
<?php

namespace Drupal\general\Controller;

use Drupal\Core\Controller\ControllerBase;

/**
 * Returns responses for General routes.
 */
class ExampleController extends ControllerBase {

  /**
   * Builds the response.
   */
  public function build() {

    $build['content'] = [
      '#type' => 'item',
      '#markup' => $this->t('It works!'),
    ];

    return $build;
  }

}
```

This is a huge timesaver!

Finding routes with Drush

Drush lets you figure out the controller associated with a route since version 10.5. Here are some of the options:

```
$ drush route
$ drush route --path=/user/1
$ drush route --name=update.status
$ sh route --url=https://example.com/node/1
```

more at https://www.drush.org/latest/commands/core_route/

All routes

Output from drush route. It lists the routes by name and the path they apply to.

```
$ drush route

'<button>': /
'<current>': /<current>
'<front>': /
'<nolink>': /
'<none>': /
admin_toolbar.run.cron: /run-cron
admin_toolbar.settings: /admin/config/user-interface/admin-toolbar
admin_toolbar_tools.cssjs: /admin/flush/cssjs
admin_toolbar_tools.flush: /admin/flush
admin_toolbar_tools.flush_menu: /admin/flush/menu
admin_toolbar_tools.flush_rendercache: /admin/flush/rendercache
admin_toolbar_tools.flush_static: /admin/flush/static-caches
admin_toolbar_tools.flush_twig: /admin/flush/twig
admin_toolbar_tools.flush_views: /admin/flush/views
admin_toolbar_tools.plugin: /admin/flush/plugin
admin_toolbar_tools.settings: /admin/config/user-interface/admin-toolbar-tools
admin_toolbar_tools.theme_rebuild: /admin/flush/theme_rebuild
batch_examples.batch: /batch-examples/batchform
...
```

Specific path

Output when checking a specific path

```
$ drush route --path /admin/flush

name: admin_toolbar_tools.flush
path: /admin/flush
defaults:
  _controller: '\Drupal\admin_toolbar_tools\Controller\ToolbarController::flushAll'
  _title: 'Flush all caches'
requirements:
  _permission: 'administer site configuration'
  _csrf_token: 'TRUE'
options:
  _admin_route: true
  _access_checks:
    - access_check.permission
    - access_check.csrf
```

Specific route name

```
$ drush route --name=admin_toolbar_tools.flush

name: admin_toolbar_tools.flush
path: /admin/flush
defaults:
  _controller: '\Drupal\admin_toolbar_tools\Controller\ToolbarController::flushAll'
  _title: 'Flush all caches'
requirements:
  _permission: 'administer site configuration'
  _csrf_token: 'TRUE'
options:
  _admin_route: true
  _access_checks:
    - access_check.permission
    - access_check.csrf
```

Getting some help from Chat GPT

At the time of this writing, [OpenAI](#) has a research release of its ChatGPT available. You could always ask it to generate a controller for you as I did. I was amazed by the result. I asked it to:

Generate a Drupal 10 controller which displays a table of temperatures from 0 to 100 degrees in both celsius and fahrenheit

To which it replied:

Here is a Drupal 10 controller that generates a table of temperatures from 0 to 100 degrees in both Celsius and Fahrenheit:

```php
<?php

namespace Drupal\mymodule\Controller;

use Drupal\Core\Controller\ControllerBase;

class TemperatureController extends ControllerBase {

  /**
   * Display a table of temperatures.
   */
  public function displayTable() {
    $rows = [];
    for ($i = 0; $i <= 100; $i++) {
      $rows[] = [
        'celsius' => $i,
        'fahrenheit' => $this->celsiusToFahrenheit($i),
      ];
    }
    $header = [
      ['data' => $this->t('Celsius'), 'field' => 'celsius'],
      ['data' => $this->t('Fahrenheit'), 'field' => 'fahrenheit'],
    ];
    $build['table'] = [
      '#type' => 'table',
      '#header' => $header,
      '#rows' => $rows,
      '#empty' => $this->t('No temperatures to display.'),
    ];
    return $build;
  }

  /**
   * Convert a temperature from Celsius to Fahrenheit.
   *
   * @param float $celsius
   *   The temperature in Celsius.
   *
   * @return float
   *   The temperature in Fahrenheit.
   */
  protected function celsiusToFahrenheit($celsius) {
    return ($celsius * 9 / 5) + 32;
  }

}
```

followed by this thoughtful addition:

To use this controller, you would need to define a route in your module's routing file that points to the displayTable method. For example:

```
mymodule.temperature_table:
  path: '/temperature-table'
  defaults:
    _controller: '\Drupal\mymodule\Controller\TemperatureController::displayTable'
    _title: 'Temperature Table'
  requirements:
    _permission: 'access content'
```

It concluded with:

This would define a route at the path /temperature-table that displays the table of temperatures when accessed.

Resources

- [Introductory Drupal routes and controllers example from Drupal documentation updated October 2022](#)
- [Structure of Routes from Drupal documentation updated September 2022](#)
- [Symfonycasts.com excellent course in Drupal 8](#)
- [Controllers article from Drupalize.me updated December 2021](#)

[Back to top](#)

Page last modified: Jul 23 2023.

[Edit this page on GitHub](#)

This site uses [Just the Docs](#), a documentation theme for Jekyll.