Build a Drupal 8 Module: Routing, Controllers and Menu Links

Daniel Sipos

Published June 13, 2014

This entry is part 1 of 5 in the series How to Build a Drupal 8 Module

Please be aware that due to the development process Drupal 8 has been undergoing at the time of writing, some parts of the code might be outdated. Take a look at this repository in which I try to update the example code and make it work with the latest Drupal 8 release.

Drupal 8 brings about a lot of changes that seek to enroll it in the same club other modern PHP frameworks belong to. This means the old PHP 4 style procedural programming is heavily replaced with an object oriented architecture. To achieve this, under the initiative of *Proudly Found Elsewhere*, Drupal 8 includes code not developed specifically for Drupal.

How to Build a Drupal 8 Module

Build a Drupal 8 Module: Routing, Controllers and Menu Links

Building a Drupal 8 Module: Blocks and Forms

Building a Drupal 8 Module – Config and the Service Container

The Drupal 8 version of EntityFieldQuery

Drupal 8 Hooks and the Symfony Event Dispatcher

One of the most important additions to Drupal are Symfony components, with 2 major implications for Drupal developers. First, it has the potential to greatly increase the number of devs that will now want to develop for Drupal. And second, it gives quite a scare to some of the

current Drupal 7 developers who do not have much experience with modern PHP practices. But that's ok, we all learn, and lessons taken from frameworks like Symfony (and hopefully Drupal 8), will be easily extensible and applicable to other PHP frameworks out there.

In the meantime, Drupal 8 is in a late stage of its release cycle, the current version at the time of writing being `alpha11`. We will use this version to show some of the basic changes to module development Drupal 7 devs will first encounter and should get familiar with. I set up a Git repo where you can find the code I write in this series so you can follow along like that if you want.

How do I create a module?

The first thing we are going to look at is defining the necessary files and folder structure to tell Drupal 8 about our new module. In Drupal 7 we had to create at least 2 files (`.info` and `.module`), but in Drupal 8, the YAML version of the former is enough. And yes `.info` files are now replaced with `.info.yml` files and contain similar data but structured differently.

Another major change is that custom and contrib module folders now go straight into the root `modules/` folder. This is because all of the core code has been moved into a separate `core/` folder of its own. Of course, within the `modules/` directory, you are encouraged to separate modules between *custom* and *contrib* like in Drupal 7.

Let's go ahead and create a module called `demo` (very original) and place it in the `modules/custom/` directory. And as I mentioned, inside of this newly created `demo/` folder, all we need to begin with is a `demo.info.yml` file with the following required content:

```
name: Drupal 8 Demo module
description: 'Demo module for Drupal 8 alpha11'
type: module
core: 8.x
```

Three out of four you should be familiar with (name, description and core). The `type` is now also a requirement as you can have yml files for themes as well. Another important thing to keep in mind is that white spaces in yml files mean something and proper indentation is used to organize data in array-like structures.

You can check out this documentation page for other key|value pairs that can go into a module `.info.yml` file and the change notice that announced the switch to this format.

And that's it, one file. You can now navigate to the *Extend* page, find the Demo module and enable it.

As I mentioned, we are no longer required to create a `.module` file before we can enable the

module. And architecturally speaking, the `.module` files will be significantly reduced in size due to most of the business logic moving to service classes, controllers and plugins, but we'll see some of that later.

What is 'routing' and what happened to hook_menu() and its callbacks?

In Drupal 7, `hook_menu()` was probably the most implemented hook because it was used to define paths to Drupal and connect these paths with callback functions. It was also responsible for creating menu links and a bunch of other stuff.

In Drupal 8 we won't need `hook_menu()` anymore as we make heavy use of theSymfony2 components to handle the routing. This involves defining the routes as configuration and handling the callback in a controller (the method of a `Controller` class). Let's see how that works by creating a simple page that outputs the classic `Hello world!` .

First, we need to create a routing file for our module called `demo.routing.yml` . This file goes in the module root folder (next to `demo.info.yml`). Inside this file, we can have the following (simple) route definition:

```
demo.demo:
  path: '/demo'
  defaults:
    _content: '\Drupal\demo\Controller\DemoController::demo'
    _title: 'Demo'
  requirements:
    _permission: 'access content'
```

The first line marks the beginning of a new route called `demo` for the module `demo` (the first is the module name and the second the route name). Under `path`, we specify the path we want this route to register. Under `defaults`, we have two things: the default page title `_title`) and the `_content` which references a method on the `DemoController` class. Under `requirements`, we specify the permission the accessing user needs to have to be able to view the page. You should consult this documentation page for more options you can have for this routing file.

Now, let's create our first controller called `DemoController` that will have a method named `demo()` getting called when a user requests this page.

Inside the module directory, create a folder called `src/` and one called `Controller/` inside of it. This will be the place to store the controller classes. Go ahead and create the first one:

`DemoController.php`.

The placement of the Controllers and, as we will see, other classes, into the `src/` folder is part of the adoption of the PSR-4 standard. Initially, there was a bigger folder structure we had to create (PSR-0 standard) but now there is a transition phase in which both will work. So if you still see code placed in a folder called `lib/`, that's PSR-0.

Inside of our `DemoController.php` file, we can now declare our class:

```
<?php
/**
 * @file
 * Contains \Drupal\demo\Controller\DemoController.
 */

namespace Drupal\demo\Controller;

/**
 * DemoController.
 */
class DemoController {
  /**
   * Generates an example page.
```

```
     */
    public function demo() {
      return array(
        '#markup' => t('Hello World!'),
      );
    }
  }
```

This is the simplest and minimum we need to do in order to get something to display on the page. At the top, we specify the class namespace and below we declare the class.

Inside the `DemoController` class, we only have the `demo()` method that returns a Drupal 7-like renderable array. Nothing big. All we have to do now is clear the caches and navigate to `http://example.com/demo` and we should see a Drupal page with*Hello World* printed on it.

What about menu links?

In Drupal 7, when we implement `hook_menu()`, we can also add the registered paths to menus in order to have menu links showing up on the site. This is again no longer handled with this hook but we use a yml file to declare the menu links as configuration.

Let's see how we can create a menu link that shows up under the `Structure` menu of the administration. First, we need to create a file called `demo.menu_links.yml` in the root of our module. Inside this yml file we will define menu links and their position in existing menus on the site. To achieve what we set out to do, we need the following:

```
demo.demo:
  title: Demo Link
  description: 'This is a demo link'
  parent: system.admin_structure
  route_name: demo.demo
```

Again we have a yml structure based on indentation in which we first define the machine name of the menu link (`demo`) for the module `demo` (like we did with the routing). Next, we have the link title and description followed by the parent of this link (where it should be placed) and what route it should use.

The value of `parent` is the parent menu link (appended by its module) and to find it you need to do a bit of digging in `*.menu_links.yml` files. I know that the `Structure` link is defined in the core System module so by looking into the `system.menu_links.yml` file I could determine the name of this link.

The `route_name` is the machine name of the route we want to use for this link. We defined ours earlier. And with this in place, you can clear the cache and navigate to `http://example.com/admin/structure` where you should now see a brand new menu link with the right title and description and that links to the `demo/` path. Not bad.

Conclusion

In this article we began exploring module development in Drupal 8. At this stage (alpha11 release), it is time to start learning how to work with the new APIs and port contrib modules. To this end, I am putting in writing my exploration of this new and exiting *framework* that will be Drupal 8 so that we can all learn the changes and hit the ground running when release day comes.

For starters, we looked at some basics: how you start a Drupal 8 module (files, folder structure etc), all compared with Drupal 7. We've also seen how to define routes and a Controller class with a method to be called by this route. And finally, we've seen how to create a menu link that uses the route we defined.

In the next tutorial, we will continue building this module and look at some other cool new things Drupal 8 works with. We will see how we can create blocks and how to work with forms and the

configuration system. See you then.

How to Build a Drupal 8 Module

Building a Drupal 8 Module: Blocks and Forms >>

Daniel Sipos

Daniel Sipos is a Drupal developer who lives in Brussels, Belgium. He works professionally with Drupal but likes to use other PHP frameworks and technologies as well. He runs webomelette.com, a Drupal blog where he writes articles and tutorials about Drupal development, theming and site building.

You might also like:

Web

DrupalCon Amsterdam 2014 Report

by Chris Ward Oct 11, 2014

Learnable

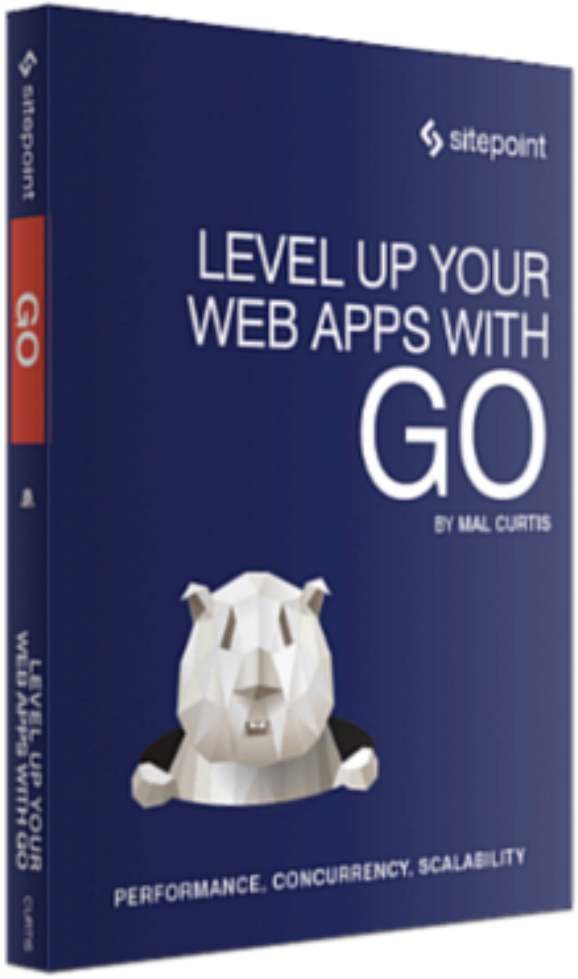
Course: PHP & MySQL Web Development for Beginners

by Kevin Yank PREMIUM

PHP

Building a Drupal 8 Module: Blocks and Forms

by Daniel Sipos Jun 16, 2014



Get your free chapter of *Level Up Your Web Apps with Go*

Get a **free chapter** of *Level Up Your Web Apps with Go*, plus updates and exclusive

offers from SitePoint.

email address

Yes Please!

27 Comments SitePoint

Recommend 2 Share

Join the discussion…

Mike • 10 months ago

Hi! First of all: thanks for the post, it really helped me out.

Second, in the current alpha (14), <name>.menu_links.yml doesn't work anymore. It has been char
<name>.links.menu.yml

I realise Drupal 8 is still in alpha (it is hard to keep up with all of the changes), but I figured this coulc

people who are trying to write their first module and are using the latest version to do so.

Cheers!

3 ^ | v • Reply • Share ›

Guest ➚ Mike • 7 months ago

Thanks Mike,
Thats helps and seems to be the way on the beta as well.

^ | v • Reply • Share ›

Tatsh • a year ago

Symfony 2 but locked down (and hidden away basically) for no good reason. Forget about Compos
good that comes from that. It is also clear that PSR is not being used at all since I see 2 spaces ins
Drupal code (minor complaint). Also, forget about the Symfony Routing component which solves al
Drupal was ever trying to solve in its routing system (hook_menu() and the like) other than form har

How come the Drupal 'community' cannot stop being such recluses? Maybe it is licensing (which is
seem to stay in their own cocoon (drupal.org and the like) and pretend the rest of the PHP commur
or is not even necessary since 'they can do it all'. Symfony 2 is basically the first time they actually
there is better code than their own that is well-written and tested. Maybe this even hurt Dries a little.

As usual, code written for Drupal 8 will be stuck to Drupal 8, just as code written for 7 is stuck to tha
versions). There will probably zero migration path to 9 other than 'porting', which is the situation with
Those code-bases are also a nightmare to deploy using just the CLI. It is also clear that in 8, the da
still the same, which means your code depends on your database having the correct data in it. Whe
reinvented several times, guaranteed. Solved problems will be 're-solved' in the 'Drupal way'.

This is why I do not get the point of using Drupal when Symfony provides almost everything in its co

can put together all your libraries with a very nice and optimised auto-loader (one less thing for a de

see more

4 ^ | v • Reply • Share ›

Larry Garfield ➚ Tatsh • a year ago

Greetings. It seems like the first part of your post may have been cut off, as it starts in the m I think there may be some confusion, though, so permit me to clarify.

Symfony2 is a pure framework with a component base. Out of the box it doesn't do anything write code to make it do anything useful. That is by design. Drupal, in contrast, is a full CMS of the box it can do all sorts of things, most of it from the GUI. So "I do not get the point of us Symfony provides..." Symfony provides a lot of good things, certainly, and Drupal 8 makes u them. But by design, Drupal does far more than Symfony does. For instance, Symfony has menu link hierarchy tool; Drupal has one that scales to tens of thousands of items. Symfony definition system; Drupal IS a content definition system. :-) Etc. So there are many many be over "plain" Symfony, if what you want is a CMS. (If what you want is "just" a framework ther please go use Symfony2 fullstack, not Drupal.)

Drupal 8 still has a lot of its own code in it that's not from a 3rd party, but the last time I looke code from 9 different sources, Symfony being just one of them. It's actually the least insular ever, and actually the least "reclusive" PHP project around of those born before the Age of C Routing system you mention is the Symfony CMF Routing component, which was actually c Drupal and Symfony developers together (myself and David Buchmann, primarily) for both t

see more

6 ^ | v • Reply • Share ›

chx ↗ Larry Garfield • a year ago

Minor nit: about menu links you say "Drupal has one that scales to tens of thousands was designed for and tested with hundreds of thousands of items (I tested it with a s Open Directory Project so, so long ago). I have reached out to Vadim Tropashko (Gc knows his SQL trees) over it and I am reasonably sure within the confines of MySQL can't be functions) our tree implementation (storing the materialized path without any of the best compromises between read performance, write performance and code c ten thousands of items were never a challenge or a serious concern of mine :)

Another minor nit: "As chx already noted, there WILL be a Drupal 6->8 migration path future tense in it, it's definitely present tense: there IS a Drupal 6->8 migration path. I and has bugs, well, Drupal 8 itself is in alpha too :)

2 ▲ | ▼ • Reply • Share ›

Larry Garfield ↗ chx • a year ago

I stand corrected. :-) Thanks.

▲ | ▼ • Reply • Share ›

Tatsh ↗ Larry Garfield • a year ago

You are correct. I do not like the idea of configuration via the GUI when it comes to ac code or manipulating the database schema: modules, themes, the PHP filter module that could easily break a site). In Drupal 6 and 7 the database is heavily used for que code: what modules and themes are enabled? what views are enabled and displaya types are there besides the ones defined in code and same for fields? As I am sure security-wise the PHP filter presents a huge vulnerability similar to WordPress' back can just edit any plugin or theme code (but not as bad). Developers are not reluctant hoping the site never gets hacked.

I would like to know if there is an improvement in Drupal 8, because when Drupal is ' almost have no problems with it. Code-based content types and views are so easily drush. But here is what I think is a common scenario that is there to 'save time and r contractors I have seen this pattern or similar way too many times:

- On the development site running locally (usually just the user's own development d create a new content type using the GUI (rather than in pure code in a module)
- Check the database for changes made (yes, 'SHOW TABLES')
- Write a module that depends on table field_somename_somefield existing
- Test it (manually by hand 90% of the time), commit, etc

see more

1 ▲ ▼ • Reply • Share ›

chx ➚ Tatsh • a year ago

Enabled modules, enabled themes, views, content types, fields, field instanc everything configuration) are all in a consistent config system. If you do it fror create it from code, doesn't matter, it uses the same config API, stores them uses the same workflow. There is no need for ctools exports any more nor F sort of work (which was not the point of Features in the first place, it kinda gr

Code lives with the code but PHP code no longer defines configurable things a configuration object and then save it from PHP but that's no different from c GUI. The configuration lives on the configuration storage and content lives in storage. Storages are pluggable, of course. By default the configuration and t storage happens to be in SQL tables both but they can be in MongoDB for ex That is not relevant for what you are asking. Configuration deployment happe them into YAML files and then importing them into say a staging or productior

Of course WordPress has a wider adoption -- there are more simple sites th
that's a given.

3 ^ | v • Reply • Share ›

Tatsh ➚ chx • a year ago

I hope you can understand where my scepticism comes from having worked
sites since version 5, then working on Symfony 2 sites and seeing how vastl
Drupal 7 is by comparison when you look at mostly 2 things: configuration an
deployment. If these things are resolved in Drupal 8 like the way you describe
pluggable), then a lot of ops teams are going to be happy to deploy Drupal 8 s
compared to 6 and 7. Developers will be happier in that getting started will be
(`drush rs` was a start (and then PHP's own built-in server, which it now use
default)). And outside developers will also be happy that they can easily put th
party code into Drupal without (or with little) compromise.

^ | v • Reply • Share ›

Larry Garfield ➚ Tatsh • a year ago

The expected workflow for the changes you describe in Drupal 8 would go sc
like this:

* Create new content type through the GUI. Also create a View to go with it, n
some display modes, an image style or two, etc.
* Push "export configuration" (GUI button or CLI command)
* Check your config/staging directory into Git.
* Checkout from Git on your staging/production site.
* Push "import configuration" (GUI button or CLI command)
* Profit!!1!

It's similar to the recommend features-based workflow in Drupal 7 if you're g on features (which any Drupal shop that knows what it's doing is already doi much more streamlined and a native "first-class citizen"; there's far far far le going on, so it should be vastly more stable than features-based deployment

The exported files are all YAML, so you *could* craft them by hand and them them if you wanted to, but I don't expect most people will want to do so most time. Maybe for small edits and tweaks, but primary site building will still be a

see more

2 • Reply • Share ›

Danny Tatsh • a year ago

You keep insisting on is this "developer-centric" view. Comparing Drupal (an with the Symfony framework is like comparing apples and oranges. Might as Zend, Yii, and CakePHP into the mix...

Drupal is a CMS, Symfony is a framework. Drupal can be installed and used builders as well, while frameworks are tools developers need time, skill and r to use and turn into some application with an interface site builders can actu So please stop making this comparison.

As for the developer experience, your grievances about Drupal 6 and 7 have addressed already in the previous comments. Deployment and configuration been their strong suit indeed. But it's also unfair to talk about Drupal 8 withou used it and based on a bias you have from 6 and 7. There are many improve made, probably not all that will satisfy everyone fully, but that's something left 9 etc.

So please stop comparing a CMS with a PHP framework. This comparison c
have it's place.

1 · Reply · Share ›

chx ➚ Tatsh • a year ago

I will leave it to others who know the Symfony - related code better to answer those.

> Another thing: no migration path. Most of the Drupal sites I maintained in 6 ...

Funny you saying that with those exact words: Drupal 8 contains a migration path from D6 t

While judging the usability is not mine to do, have you seen quickedit in D8? :)

2 · Reply · Share ›

Tatsh ➚ chx • a year ago

Yes I have seen that in Drupal 8. I like it. It is a big improvement. Even clients have th
Drupal?' after knowing what it is like to have a barely optimal WYSIWYG'd Drupal 7 e

· Reply · Share ›

Bruno Skvorc SitePoint Staff ➚ Tatsh • a year ago

Interesting perspective, thank you for taking the time to write it

· Reply · Share ›

Luis Eduardo Telaya Escobedo • 6 months ago

Oh! change from _content to _controller so that it works perfect! https://www.drupal.org/node/20...

1 · Reply · Share ›

Arpit Rastogi • an hour ago

Thanks for sharing such a nice post.
Just a small fix instead of _content in routing.yml it should be _controller

Reply • Share ›

Dominique • 2 months ago

I am wondering if there is a simple way to have a dynamic title in the demo.menu_links.yml config fi

something like:

demo.demo:
_title_callback: \callback\route::function

I have been looking for a solution without success.

Reply • Share ›

fabien • 5 months ago

Hi !
Very good article but i've a question : How can we use our custom route in twig template ? I've tryin
path('demo.demo')}} but there is an error "Route does not exist". Any idea ?

Reply • Share ›

cjwest • 6 months ago

Great demo! Simple and clear instructions.
I could use some guidance. I'm able to enable the module and clear the cache, but I'm getting a "pa
/demo. I think I'm missing the secret handshake. Any suggestions for troubleshooting?

Reply • Share ›

AbbyG cjwest • 5 months ago

Daniel - Thanks for this tutorial!

I'm having the same problem as cjwest (am getting a 'page not found') error. Has anyone er issue, then resolved it? If yes, please share solution.

Reply • Share ›

Kathrin ➔ AbbyG • 5 months ago

As Luis Eduardo Telaya Escobedo wrote in his post: in the demo.routing.yml-File ch "_content" to "_controller". I just had the same problem...

1 Reply • Share ›

please_just_stop • 6 months ago

I saw "'#markup' => t('Hello World!')," and thought, yeah, never again.

How can this still be happening in 2014? The Drupal value proposition is to learn OO concepts and fudge them into the Drupal 'way' whilst dancing a circular jig to the sound of the Acquia money flute

If you're going to this much trouble to route a module then why in the hell not just use a mature MVC used Drupal since 6 but this direction makes no sense to me. Somehow we're expected to find dev understand higher level abstractions but at the same time are happy to put up with render arrays of

It's almost as if the whole point of the project is to create expensive bi-annual upgrades to support a group of cheerleading middlemen #sadface :(

Reply • Share ›

DarkPres • 10 months ago

Very smooth walk through till the 'menu_links' section when you go "digging around in the *.menu_li could not replicate this. How precisely do I determine what my parent menu link is?

Reply • Share ›

Danny ↗ DarkPres • 10 months ago

Hey there,

You need to look in the core files and see the names of the menu links defined by core.

D

• Reply • Share ›

zy • a year ago

Thanks Daniel, for this good and concrete intro. Looking forward to read more from you on the D8 t

• Reply • Share ›

Danny ↗ zy • a year ago

Thanks and I am glad you found it useful. A couple of more articles are coming soon! Stay t

• Reply • Share ›

Антон • 8 months ago

Hi! Please change "demo.menu_links.yml" file name to "demo.links.menu.yml" to worked sample.

• Reply • Share ›