Save time upgrading from Drupal 7 to Drupal 10 using Retrofit for Drupal

Matt Glaman

Maintainer of phpstan-drupal, composer-drupal-lenient, retrofit-drupal

u/mglaman   mglaman.dev

Drupal 7 end of life
January 5, 2025

Q4 '23

Q1 '24 **Q2 '24** **Q3 '24** **Q4 '24**

Q1 '25

If you haven't started, you only have *four quarters*.

Challenges migrating from Drupal 7 → Drupal 10

Migrate **existing** configuration and content to its **new** schema.

Rewriting custom code from **legacy** APIs to their **modern** API equivalents.

Rewriting custom themes from the **PHPTemplate** **template engine to Twig**

Maintain your existing platform's bug tickets & change requests

Retrofit
for Drupal

What if you didn't have to rewrite all of the **code** for your custom modules?

What if you didn't have to rebuild your **theme** from scratch?

What if you only need to refactor **40%** of your code versus **98%**?

Retrofit provides compatibility layers to run legacy Drupal code.

Retrofit enables you to migrate off of Drupal 7 faster and easier

How **Retrofit** works

How **Retrofit** integrates

- Retrofit *is not* a module

- Retrofit integrates with Drupal's service container to register its backward compatibility layers

- Leverages event subscribers and service decoration to enable supporting legacy API implementations

- Provides plugins and derivers for APIs that became plugin-based.

```php
$namespaces = $container→getParameter('container.namespaces');
$namespaces['Retrofit\Drupal'] = __DIR__;
$container→setParameter('container.namespaces', $namespaces);

$container
  →register(HookMenuRoutes::class)
  →setAutowired(true)
  →addTag('event_subscriber');

$container
  →register(GlobalUserSetter::class)
  →addTag('event_subscriber');

$container→setDefinition(
    Registry::class,
    (new ChildDefinition('theme.registry'))
    →setDecoratedService('theme.registry')
)
```

Overview of supported compatibility layers

- Replacements for procedural functions

- The global `$user` object for the current user.

- `hook_menu` to the new routing, menus, and links APIs.

- `hook_block_*` to the new plugin format.

- `hook_form_*` to objects format, including access form state as an array.

Overview of supported compatibility layers

- Register hook_process_* hooks as hook_preprocess

- hook_theme_* Theme functions and PHPTemplate templates

- Theme support (template overrides)

- drupal_add_js for JavaScript settings

How to use
Retrofit

Retrofit runs in your **new** Drupal 10 code base

How to use **Retrofit**

1. Create a Drupal 10 code base
2. Add `retrofit-drupal/retrofit`
3. Copy your Drupal 7 modules and themes
4. Migrate your data
5. Refactor your code
6. Repeat 4 & 5!

```
composer create-project \
 drupal/recommended-project \
 drupal

cd drupal

composer require drupal-retrofit/retrofit
```
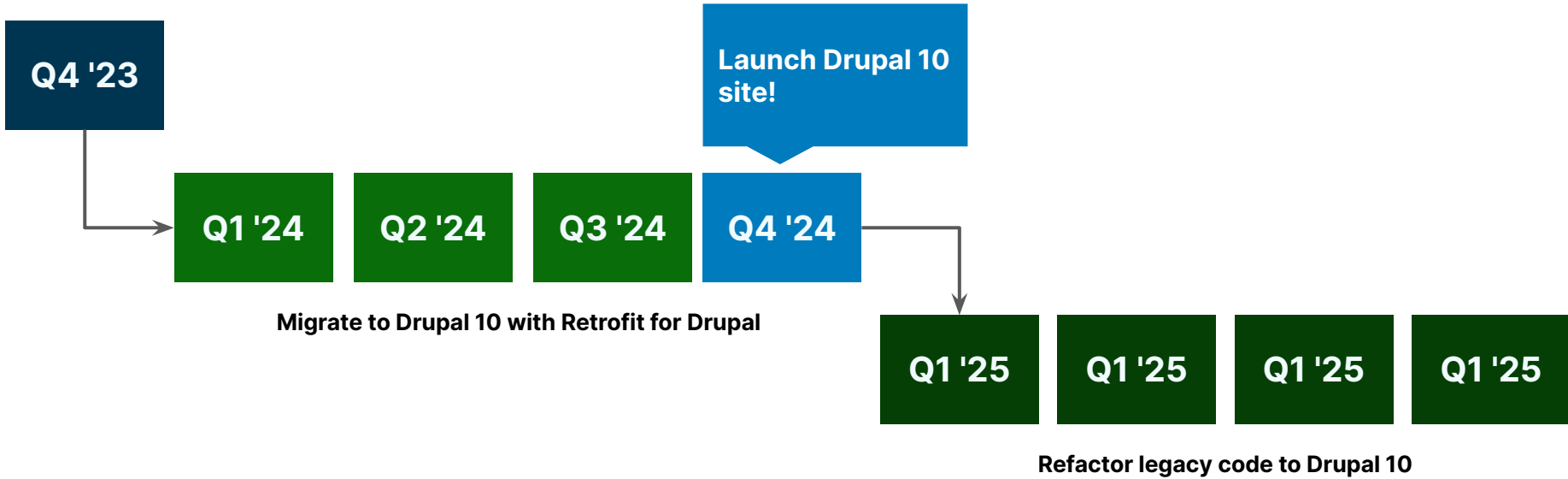
Example: https://github.com/retrofit-drupal/demo-examples

Benefits of
Retrofit

Allows for incremental updates instead of upfront major changes

Increase **velocity** by reducing refactoring and cognitive load

Upgrade becomes safer and more reliable by refactoring less code.

Provides a longer timeline for refactoring legacy code.

Migrate in 2024, refactor through 2025 and beyond

Retrofit

for Drupal...
10, 11, 12, 13+

Support
Retrofit

Funding through GitHub Sponsors supports development

Support is available to help organizations with using Retrofit

Support options

Community: public support through GitHub discussions and issues

Premium: Upcoming private support plans using tickets for handling private code or more in depth solutions

Enterprise: A more high touch offering, if needed

Resources

Resources

- **Website** [Retrofit for Drupal](#)
- **Blog** [Retrofit: Running legacy Drupal 7 code on your Drupal 10 site](#)
- **Blog** [Protect your investment in Drupal 7 with Retrofit for Drupal](#)
- **Podcast** [Drupal 7 End-of-Life Ep 08: Building a Bridge to Drupal 7 with Matt Glaman | Chromatic](#)
- **Video** [Retrofit: Running legacy Drupal 7 code on your Drupal 10 site](#)

Questions about **Retrofit**?