Taking Maximum Advantage of Drupal Core's Composer Template

...

Core skills for a professional Drupal 8/9 developer

Michael Anello

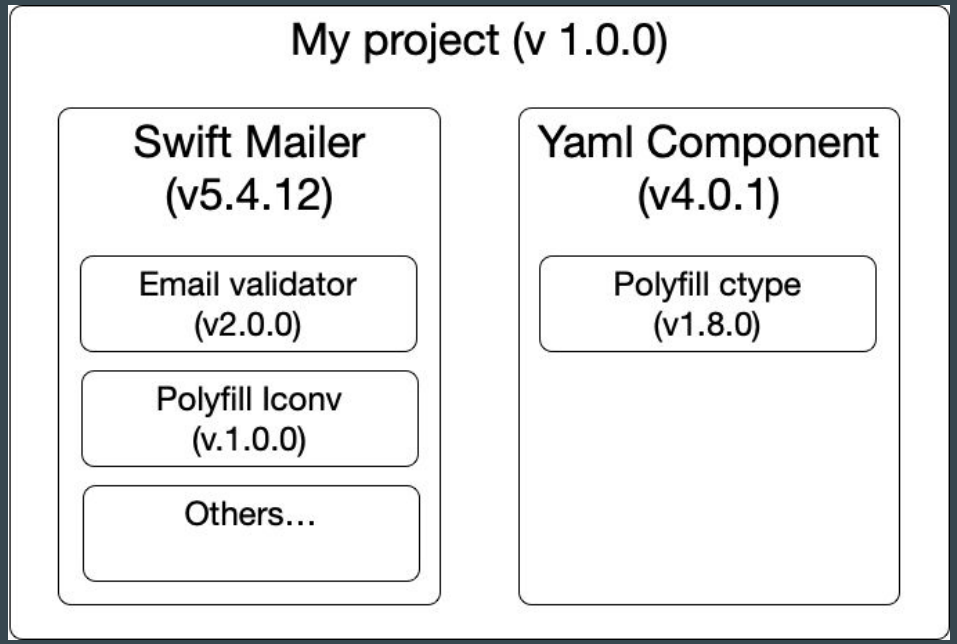
- DrupalEasy/Anello Consulting, Inc. (drupaleasy.com)
- ultimike on drupal.org (drupal.org/u/ultimike)
- @ultimike
- Web developer since 199-something
- Drupal developer since 2006

These slides: bit.ly/max-composer

Composer - what is a dependency manager?

Composer - what is a dependency manager?

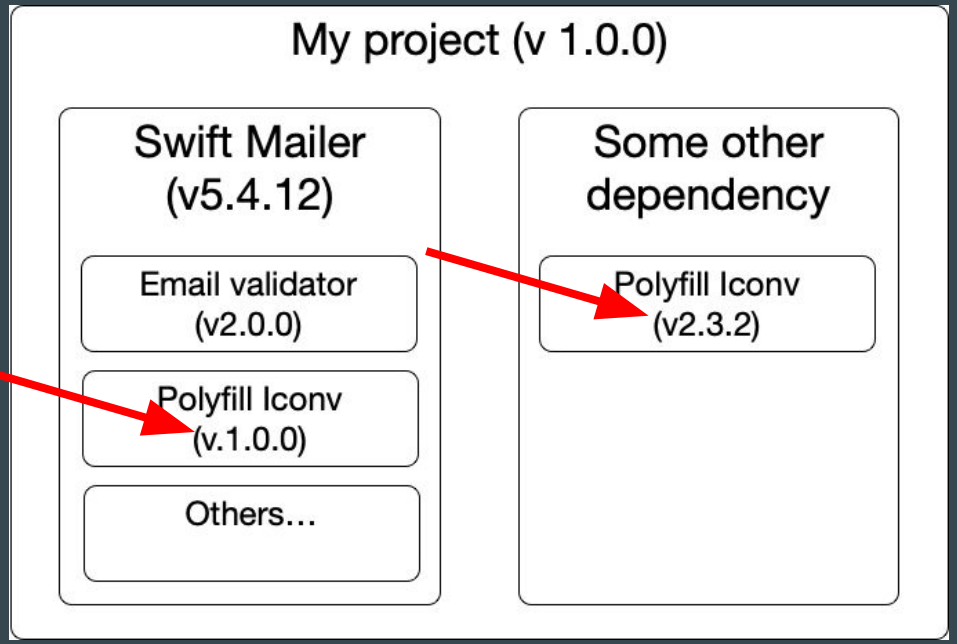
- Each PHP class can have its own dependencies.



Composer - what is a dependency manager?

- Consider the following:
  - When you build your project, it works with particular versions of its dependencies.
  - There are no guarantees that the next version of a dependency will still work with your project (APIs change!)
  - There needs to be a way to "lock" a particular version of your project with a particular version of each of your dependencies.

Composer - what is a dependency manager?



Composer - what is a dependency manager?

- Keeping each dependency's version locked as well as checking for version conflicts is challenging.
- Assembling a code base that includes dependencies and dependencies of dependencies, making sure that the version numbers are correct, etc.. is extremely tedious!
- "Humans are not dependable" - Ryan Szrama, Commerce Guys.
- Composer does it for us: we tell Composer we need a particular version of a dependency (or a range of versions) and it takes care of the rest. Checking for version conflicts, downloading it, getting its dependencies, etc...

Basic Composer commands - review

- `composer list`
- `composer help <command>`
- `composer require`
- `composer update`
- `composer remove`
- `composer install`

Creating a new project with Composer using drupal/recommended-project

But first, a quick history lesson...

- Since Drupal 8.0 was released (November, 2015!), the Drupal Composer/Drupal Project template has been the "best practice".
  - https://github.com/drupal-composer/drupal-project/
- ...until Drupal 8.8 was released (December, 2019) with the drupal/recommended-project template
  - https://github.com/drupal/recommended-project
- Differences?

Overview of the drupal/recommended-project template

- Considered a best practice for managing a Drupal 8.8+ project codebase.
- Provides a robust starting point.
- Supported by the community as part of Drupal core.

Overview of the drupal/recommended-project template

- Makes Drupal core (core-recommended) a dependency of the project.
- Provides a nested document root.
- Automatically keeps Drupal's scaffolding files up-to-date.
- Installs Drupal-specific project dependencies (modules, themes, profiles, etc..) in the proper directories.
- Provides access to Drupal's own public project repository (Drupal modules and themes are not available via packagist.org).

Live demo/exercise

Live demo

- `composer create-project drupal/recommended-project` *`sitename`*
- **`cd`** ***`sitename`***
- `composer require drupal/pathauto`
- `composer update drupal/pathauto`
- `composer remove drupal/pathauto`

Let's maximize! PHP versions.

- PHP versions and Composer
  - Add `"php": ">=7.4.0"` to composer.json, try updating something.
- Be careful when using virtual machines or Docker containers.

Let's maximize! Custom dependencies.

- Add a custom dependency from a private GitHub project
  - Try composer require ultimike/privaterepo
  - Add new repository:

```
{
  "type": "vcs",
  "url":  "git@github.com:ultimike/privaterepo.git"
}
```

  - Get token from GitHub.

Let's maximize! Composer outdated command.

- See which dependencies have updates available.
- `composer outdated`
- Better yet, use…
- `composer outdated --direct`

Let's maximize! Dry run, why-not, validate

- When updating or requiring, running into issues? Troubleshoot with:
  - `composer update --dry-run`
  - `composer why-not twig/twig`
  - `composer depends twig/twig`
  - `composer validate`

Let's maximize! Require alpha, beta, RC, and dev

- Need to install a pre-release version of a dependency?
  - `composer require vendor/name:^1.0@beta`
- `@alpha, @beta, @RC, -dev` all valid
  - When using `-dev`, the dependency is *cloned*.
    - `composer require drupal/pathauto:1.x-dev`

Let's maximize! Applying patches

- Use the Composer patches dependency
  - `composer require cweagans/composer-patches`
- Add the patch URL to your project's composer.json

```
"patches": {
        "drupal/draggableviews": {
"Not working with Group By feature":
"https://www.drupal.org/files/issues/2020-06-22/not_working_with_group_by-draggableviews-2867348-40.patch"
        }
    },
```

- Run `composer install`

Let's maximize! Customize Drupal scaffolding files

- The drupal/core-composer-scaffold allows you to customize (remove, append, edit, add) Drupal scaffolding files.
- https://www.drupal.org/docs/develop/using-composer/starting-a-site-using-drupal-composer-project-templates#s-drupalcore-composer-scaffold
- Example: remove the README.txt via configuration in the project's composer.json's "drupal-scaffold" section (delete the original README.txt first).

Let's maximize! Requiring Drush or Drupal Console

- Have Drush and/or Drupal Console as part of your codebase
  - `composer require drush/drush`
- Don't forget to update them!

Composer 2.0?

Questions?