Symfony & Drupal

Symfony & Drupal

Jonathan Daggerhart

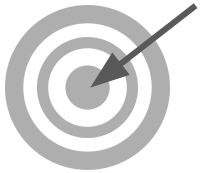
- **Developer at Hook 42**
- **Organizer for Drupal Camp Asheville**

Drupal.org: daggerhart
Twitter: @daggerhart
Blog: https://www.daggerhart.com

Drupal Camp Asheville
Site: https://drupalasheville.com
Twitter: @drupalasheville

Symfony & Drupal

What we will cover



1. What is Symfony?
2. Symfony Components overview
3. Configuration Formats
    a. YAML
    b. Annotations
4. Explore Drupal 8 File Structures

What is Symfony?

"Symfony is a set of reusable PHP components, and together can be used as a PHP framework for web projects."

A philosophy of Symfony is to be a "3-in-1 framework". Meaning it can serve our application in up-to three ways.

1. **Full stack**: Symfony is the foundation of our application

2. **Brick by brick**: Use as much of Symfony as we need, when we need it. Components work well together, build our own framework atop these components.

3. **Microframework**: Use a small set of components to develop services for our application.

Drupal is a Symfony application

Considering the discussed Symfony's philosophy, which of the three does Drupal best fit into?

Answer: Brick by brick- Use as much of Symfony as we need.

What does this imply about Drupal 8?

Practically, it means is that beneath many (but not all) Drupal 8 functionality lies a Symfony component.

Obvious next question: Considering the Drupal 8 API, what is Symfony and what is Drupal?

Symfony Components List

Symfony Components:

Asset
BrowserKit
Cache
ClassLoader
Config
Console
CssSelector
Debug
DependencyInjection
DomCrawler
EventDispatcher
ExpressionLanguage
Filesystem
Finder
Form
HttpFoundation
HttpKernel
Intl
Ldap
OptionsResolver
PHPUnit Bridge
Process
PropertyAccess
PropertyInfo
Routing
Security
Serializer
Stopwatch
Templating
Translation
Validator
VarDumper
Yaml

Symfony Components in Drupal

Symfony Components used by Drupal:

Asset	ExpressionLanguage	PropertyAccess
BrowserKit	Filesystem	PropertyInfo
Cache	Finder	**Routing**
ClassLoader	Form	Security
Config	**HttpFoundation**	**Serializer**
Console	**HttpKernel**	Stopwatch
CssSelector	Intl	Templating
Debug	Ldap	**Translation**
DependencyInjection	OptionsResolver	**Validator**
DomCrawler	PHPUnit Bridge	VarDumper
EventDispatcher	**Process**	**Yaml**

New Configuration Formats

With Symfony comes new formats for configuration. The most common format we need to understand is by far YAML. If we need to extend Drupal in certain ways, we'll need to understand Annotations.

1. **YAML** - stands for "YAML Ain't Markup Language" often abbreviated as ".yml"

2. **Annotations** - PHP comments that are parsed by the framework into structures and values

YAML - YAML Ain’t Markup Language

“YAML is a human friendly data serialization standard for all programming languages.” - http://yaml.org/

Component: The Symfony Yaml component parses YAML strings to convert them to PHP arrays. It is also able to convert PHP arrays to YAML strings.

In Drupal 8, we use YAML in place of some of the more configuration-y hooks we used in Drupal 7. Specifically, `hook_menu` has been broken into multiple YAML and PHP files.

Symfony & Drupal

YAML syntax

YAML syntax allows for strings, integers, booleans, lists, and structure.

Data type	Syntax example
String	`my_string: “hello world”`
Integer	`my_int: 15`
Boolean	`is_boolean: false`

List
(two variations)
	`my_list:`
	`  - “string one”`
	`  - false`
	`another_list: [‘string two’]`

structure	`address:`
	`  street1: “123 Fake St”`
	`  city: “Nowhere”`

Note: no end of line delimiters

Symfony & Drupal

YAML in Drupal 8

The YAML format is used for the following things in Drupal 8

- Module and Theme info files
- List of routes provided by a module
- List of libraries a module or theme uses
- Permissions a module provides
- Menus and links provided by a module
- Services a module provides
- Schema definitions
- Configuration Management

Annotations

The Drupal plugin system has a set of reusable components that developers can use, override, and extend in their modules. Plugins use annotations to register themselves with the system and describe their metadata.

Some of the plugins types provided by core are:

- Blocks
- Field formatters, Field widgets
- All Views plugins
- Conditions
- Migrate source, process & destination plugins

Symfony & Drupal

Annotations - Example

An example of a plugin using this annotation class is the `UserNameUnique` used for user validation.

This annotation contains just the ID and label:

```
/**
 * Checks if a user name is unique on the site.
 *
 * @Constraint(
 *   id = "UserNameUnique",
 *   label = @Translation("User name unique", context = "Validation"),
 * )
 */
class UserNameUnique extends UniqueFieldConstraint {
 //...
}
```

Symfony & Drupal

Drupal Core File Structure

Drupal 8 has moved its own files to within the “core” directory. All the contents of the other top-level folders are site dependent.

Our site’s modules go here.

The file system location hasn’t changed, making it easier to upgrade.

Symfony (and other library) components.

Folder	Description
`/core`	Core modules and files provided by Drupal 8.
`/modules`	Contrib and custom modules (used to be sites/all/modules)
`/profiles`	Contrib and custom profiles.
`/sites`	Filesystem. Files generated by Drupal and files uploaded by users.
`/themes`	Contrib and custom themes and subthemes (used to be sites/all/themes)
`/vendor`	External 3rd party libraries and projects, e.g. phpUnit, Behat

Symfony & Drupal

Drupal Module File Structure

More on module file structure in later sections.

Discussed more along with “Hooks”.

Folder	Description
`/<module>.info.yml`	Module’s metadata provided to the system. Includes module name, description, etc.
`/<module>.module`	*(Optional)* Module’s custom code entry point. Necessary for implementing hooks.
`/<module>.*.yml`	Module’s YAML related metadata. Necessary for implementing certain features.
`/config`	*(Optional)* Configuration YAML files, includes data type definitions and initial values.
`/src`	Classes the module provides or overrides.
`/templates`	*(Optional)* Site specific modules, themes,& files. Including files uploaded by users.
`/tests`	Contrib and custom themes and subthemes
`/(js|css)`	Libraries our module needs. Place JavaScript files within a folder named “js”, CSS files within a folder named “css”, etc.