Drupal 8 Kickstart

An overview of D8 for developers

FLORIDA DRUPALCAMP 2015

April 11, 2015

Peter Sawczynec

Engineer

CivicActions

Drupal 8 Ecosystem

Drupal 8 Symfony PHP OOP Drush Git GitHub

Markdown Composer **Linux shell** zshell SSH

Behat Gherkin PHPUnit jMeter

MySQL Workbench Regex JSON jQuery

AngularJS Node.js Twig Compass SASS SMACSS

Guzzle Memcache Varnish CDN Service

Jenkins Chef Splunk Apache Nginx

phpStorm Sublime NetBeans

Drupal and Drush

- Drush helps execute Drupal admin tasks 3 - 10x faster than using the admin pages
- Drush is a command line shell for Drupal providing uncountable shortcut commands
- Drush runs update.php and clears cache, login, change passwords, disable/enable modules, execute sql queries, manage features…
- Install drush with Composer

Drupal and Drush

```
drush cc all

drush updb

drush sqlq 'SELECT schema_version FROM system WHERE name="views"'

drush sqlq "UPDATE system SET schema_version = 12 WHERE name='views'"

drush en devel -y

drush uli

drush upwd --password="newsecurepasswoed" "admin"

drush pmi views

drush status
```

Drupal 8: Missions

CivicActions

- **Drupal 8**: A service returning a response of format-agnostic data structures
- Whether the request comes from a desktop browser, mobile phone, or another website the response ***data*** will be returned consistently
- How the response data gets formatted is, as much as possible, a distinct and separate set of actions

Drupal 8: Missions

- Leverage existing industry-standard technologies so that **D8** can interface with and be programmed like other globally-recognized PHP frameworks using PHP OOP concepts
- To achieve the missions **D8** is built on top of the Symfony framework components

Drupal 8: Headless Drupal

- Drupal is a service first, not an HTML output provider
- Drupal provides format-agnostic data for a request
- A website's browsing visitor may not even interact with Drupal's themed output
- For example, pages can be created by JS frameworks such as Angular or Backbone and Drupal is the data/content store

Drupal 8 Essential Reading

CivicActions

D8 API reference and concepts page:
http://api.drupal.org/api/drupal/8

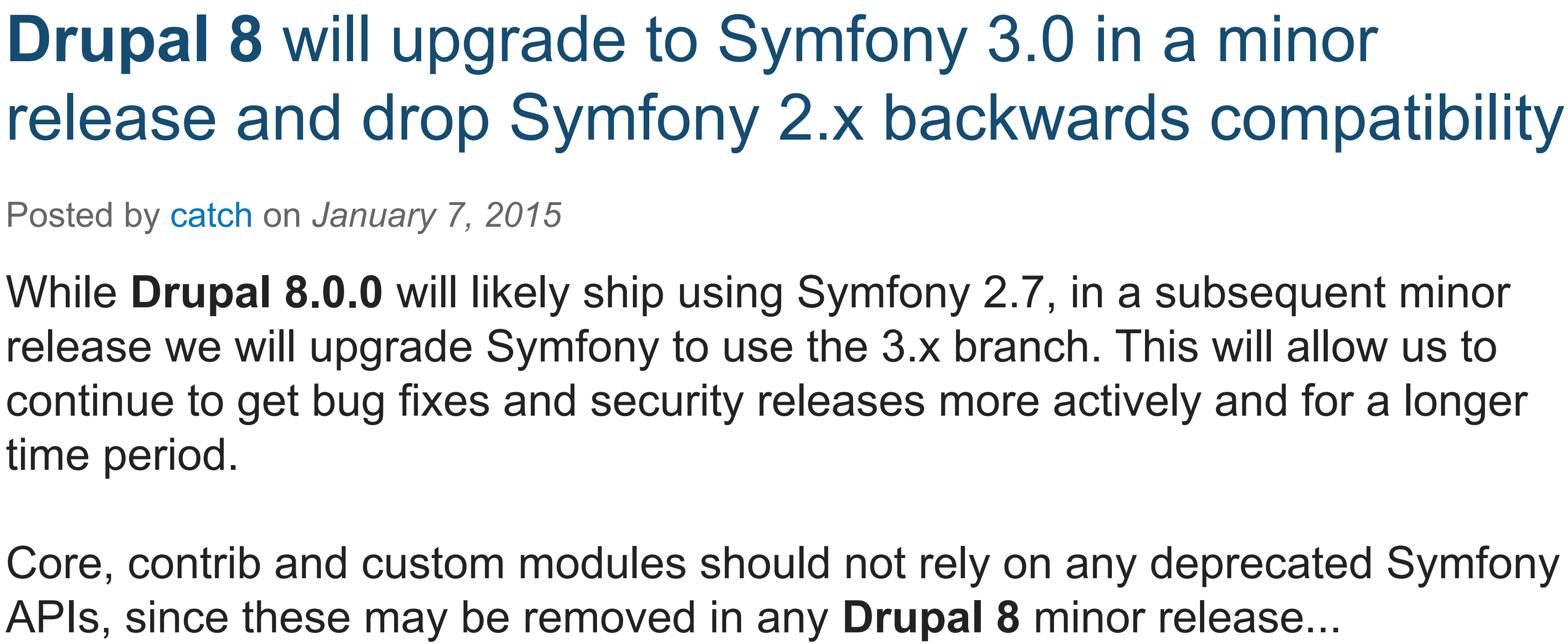
O’Reilly **Drupal 7 / 8** book:

Programmer’s Guide to Drupal *2nd Edition*

Drupal 8 Online Documentation

- Check the documentation creation date to judge the timeliness and accuracy of online **D8** documentation
- Other than on Drupal.org, **D8** online documentation older than October 2014 is unlikely to be fully accurate

D8 / Symfony: Special Note



Drupal 8 will upgrade to Symfony 3.0 in a minor release and drop Symfony 2.x backwards compatibility

Posted by catch on *January 7, 2015*

While **Drupal 8.0.0** will likely ship using Symfony 2.7, in a subsequent minor release we will upgrade Symfony to use the 3.x branch. This will allow us to continue to get bug fixes and security releases more actively and for a longer time period.

Core, contrib and custom modules should not rely on any deprecated Symfony APIs, since these may be removed in any **Drupal 8** minor release...

D8 and Code Comments

Comments and special comments called annotations are more important that ever in **D8**

Drupal 8 / OOP

- **Trait** - A collection of php functions in one file that supply a useful set of related functions
- **Interface** - A class with empty default methods that all other classes based on it must offer [class TrackingInject implements TrackingInjectInterface]
- **Abstract Class** - A class with default abstract methods that classes based on it must offer [class TrackingInject extends WebTracking]

Drupal 8 / OOP

- **Class** - A collection of functions and properties that offer a service
  [class TrackinInject extends WebTracking implements TrackingInjectInterface, CacheInterface]
- **Controllers**, **Routers**, **Forms** - Major types of classes in **D8**

Drupal 8 / OOP

- **Dependency Injection** - Initiating a class, but telling the class what you want it to use to work

  What is Dependency Injection?
  by Fabien Potencier

Drupal 8 / OOP

- **Services** - Something your class offers, such as “find this node’s location on a map by address, returns a latitude and longitude”
- **Plugins** - In **D8** plugins are used, for example, to make Blocks, in that your Block and what describes it, builds it, and controls access to the Block is found in a special kind of class called a plugin

Drupal 8 / OOP

Interface:

Every single Method declared in an Interface will have to be implemented in the subclass. A class can implement multiple Interfaces

[class TrackingInject implements TrackingInjectInterface]

Drupal 8 / OOP

Abstract Class:

Only Abstract methods have to be implemented by the subclass. An Abstract class can have normal methods with implementations. Abstract class can also have class variables. A class can only implement one abstract class at a time.

[class TrackingInject extends WebTracking]

[class TrackinInject extends WebTracking implements TrackingInjectInterface, CacheInterface]

More details interface and abstract class…   Differences interface and abstract class…

Chaining (or Method Chaining)

jQuery, PHP, and D8 feature method chaining

Allows us to run a series of methods, one after the other (or in a chain), because each method in the chain after it executes returns a full object with the the changes on it that have just been performed

Chaining (or Method Chaining)

jQuery examples:

```
$(#edit-button).changeColor(‘blue’).move(‘left’, 21px).save();

$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

Chaining written multiline if that aids clarity:

```
$("#p1").css("color","red")
   .slideUp(2000)
   .slideDown(2000);
```

Chaining (or Method Chaining)

D8 Example (multiline) using the Database Abstraction Layer. db_update returns an UpdateQuery object

```
db_update('example')
      ->condition('id', $id)
      ->fields(array('field2' => 10))
      ->execute();
```

D8 Top-level Directory Structure

- **/core** Core modules and files provided by **D8**
- **/libraries** Common 3rd party libraries, eg. a wysiwyg editor
- **/modules** Contrib and custom modules using sub-dirs **contrib** and **custom**
- **/profile** Contrib and custom profiles
- **/themes** Contrib and custom (sub)themes
- **/sites/[domain** OR **default]/{modules, themes, files}**

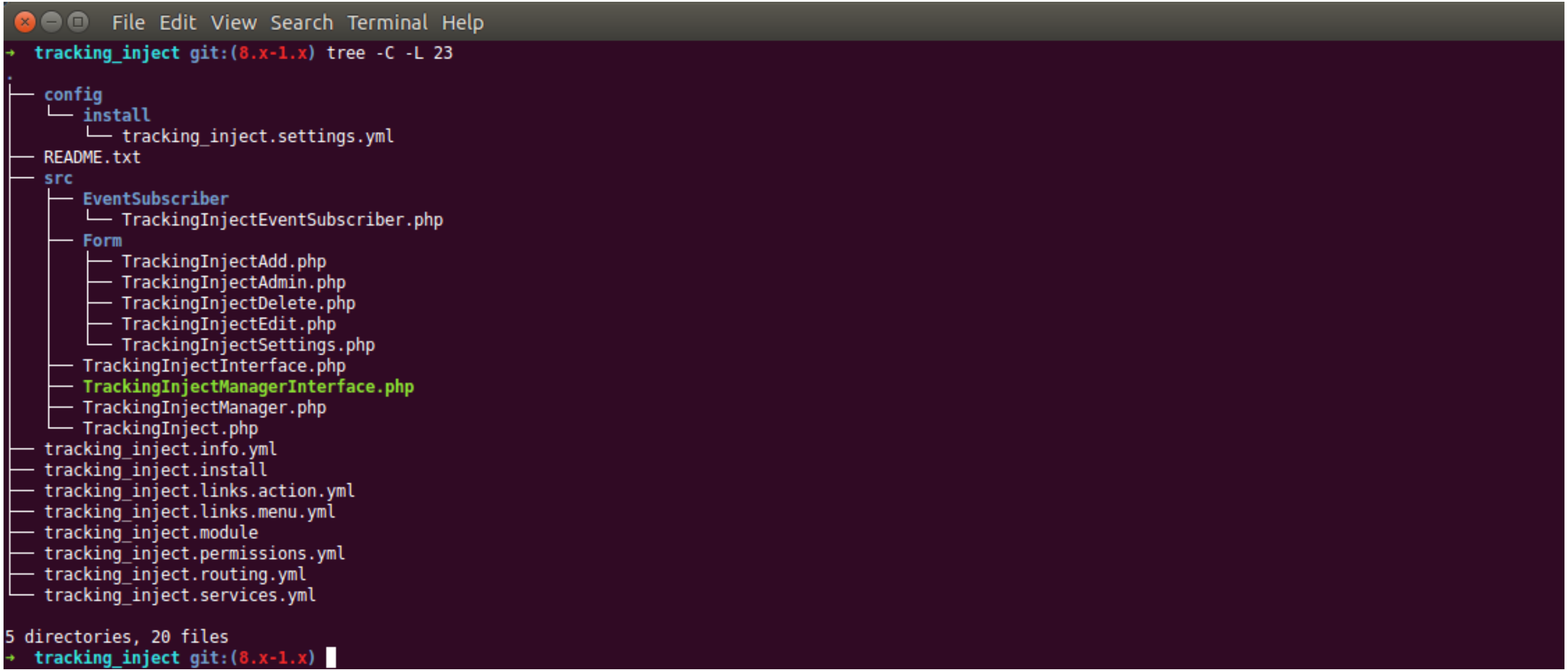
  Site specific modules, themes and files. Including files uploaded by users, such as images. Also includes the Configuration files, **active** as well as **staged** config files. Configuration files are read and written by **D8**, and should have the minimal amount of privileges required for the webserver, and the only the webserver, to read and modify them

D8 Core Directory Structure

Inside /core directory:

- **/core/assets** Various external libraries used by Core. jQuery, underscore, modernizer etc
- **/core/config** Configuration files
- **/core/includes** Functionality that is to low level to be modular. Such as the module system itself
- **/core/lib** Drupal Core classes
- **/core/misc** Frontend libraries that Drupal Core depends on. (jQuery, modernizer, etc)
- **/core/modules** Drupal Core modules
- **/core/profiles** Drupal Core profiles. Empty at the time of writing
- **/core/scripts** Various CLI scripts, mostly used by developers
- **/core/tests** Drupal Core tests
- **/core/themes** Drupal Core themes
- **/core/vendor** Backend libraries that Drupal Core depends on. (Symfony, Twig, etc) http://drupal.stackexchange.com/questions/84811/what-are-all-the-directories-for-in-the-new-drupal-8-structure

Drupal 8 Module Structure



```
➜ tracking_inject git:(8.x-1.x) tree -C -L 23
.
├── config
│   └── install
│       └── tracking_inject.settings.yml
├── README.txt
├── src
│   ├── EventSubscriber
│   │   └── TrackingInjectEventSubscriber.php
│   ├── Form
│   │   ├── TrackingInjectAdd.php
│   │   ├── TrackingInjectAdmin.php
│   │   ├── TrackingInjectDelete.php
│   │   ├── TrackingInjectEdit.php
│   │   └── TrackingInjectSettings.php
│   ├── TrackingInjectInterface.php
│   ├── TrackingInjectManagerInterface.php
│   ├── TrackingInjectManager.php
│   └── TrackingInject.php
├── tracking_inject.info.yml
├── tracking_inject.install
├── tracking_inject.links.action.yml
├── tracking_inject.links.menu.yml
├── tracking_inject.module
├── tracking_inject.permissions.yml
├── tracking_inject.routing.yml
└── tracking_inject.services.yml

5 directories, 20 files
➜ tracking_inject git:(8.x-1.x)
```

Drupal 8 Bootstrap

1. Bootstrap configuration:
    - Read the settings.php file, generate some other settings dynamically, and store them both in global variables and the Drupal\Component\Utility\Settings singleton object
    - Start the **class loader**, takes care of loading classes
    - Set the Drupal error handle.
    - Detect if Drupal is actually installed. If it is not, redirect to the installer script

Drupal 8 Bootstrap

2. Create the Drupal kernel
3. Initialize the service container (either from cache or from rebuild)
4. Add the container to the Drupal static class
5. Attempt to serve page from static page cache
6. Load all variables (variable_get)
7. Load other necessary (procedural) include files

Drupal 8 Bootstrap

7. Register stream wrappers
   (public://, private://, temp:// and custom wrappers)
8. Create the HTTP Request object
   (using the Symfony HttpFoundation component)
9. Let the DrupalKernel handle it and return response
10. Send response
11. Terminate request
    (modules can act upon this event)

D8 Yaml (*.yml) Files

Replaces .info files and used for Configuration, Routes, Menu Links, and Services

D8		D7
<module_name>.info.yml	<-->	<module_name>.info file
<module_name>.routing.yml	<-->	hook_menu for page paths
<module_name>.links.menu.yml	<-->	hook_menu for entries on admin menu
<module_name>.permissions.yml	<-->	hook_permissions
<module_name>.services.yml	<-->	Describes a class: machine name, class path, mandatory arguments

Services .yml File

```yaml
services:
  tracking_inject.manager:
    class: Drupal\tracking_inject\TrackingInjectManager
    arguments: ['@database']
    tags:
      - { name: backend_overridable }
  tracking_inject.response_event:
    class: Drupal\tracking_inject\EventSubscriber\TrackingInjectEventSubscriber
    tags:
      - { name: event_subscriber }
  tracking_inject.injections:
    class: Drupal\tracking_inject\TrackingInject
    arguments: ['@config.factory']
```

D8 Hooks to Events

Drupal 8 Hooks

Request Event Example

Drupal 8 Events

D8 uses Symfony kernel and events. Kernel events available in **D8** are as follows:

- **KernelEvents::CONTROLLER**
  CONTROLLER event occurs once a controller was found for handling a request
- **KernelEvents::EXCEPTION**
  EXCEPTION event occurs when an uncaught exception appears
- **KernelEvents::FINISH_REQUEST**
  FINISH_REQUEST event occurs when a response was generated for a request

Drupal 8 Events

- **KernelEvents::REQUEST**
  REQUEST event occurs at the very beginning of request dispatching
- **KernelEvents::RESPONSE**
  RESPONSE event occurs once a response was created for replying to a request
- **KernelEvents::TERMINATE**
  TERMINATE event occurs once a response was sent
- **KernelEvents::VIEW**
  VIEW event occurs when the return value of a controller is not a Response instance

Services in Drupal 8

All core services in **D8** are declared in: /core/core.services.yml

Services can be used throughout **D8** via the global Drupal namespace, e.g.:

```
\Drupal::moduleHandler()->moduleExists('content_translation');
\Drupal::entityManager()->getTranslationFromContext($node) !== $node->getUntranslated()
$account = \Drupal::currentUser();

$config = \Drupal::config('tracking_inject.settings');
$id = $config->get('domain');

$request = \Drupal::request();
$exception = $request->attributes->get('exception');
$status = $exception->getStatusCode();
```

D8 variable_get, variable_set

Replaced by using a **Drupal 8** core service…

Getting a variable:

```
\Drupal::config(‘module_name.settings’)->get(‘var_name’);
```

Setting a variable:

```
\Drupal::config(‘module_name.settings’)
->set(‘var_name’, ‘some_value’)
->save;
```

D8 variable_get, variable_set

For example, getting the Home Page:

```
\Drupal::config('system.site') ->get('page.front');
```

Unsetting a variable value:

```
$config = \Drupal::config('system.performance');
$config->clear('cache.page.max_age')->save();
```

D8 Routes and Controllers

Routing System in Drupal 8

A route is a path which is defined for Drupal to return some sort of content on.

For example, the default front page, '/node' is a route. When Drupal receives a request, it tries to match the requested path to a route it knows about. If the route is found, then the route's definition is used to return content. Otherwise, Drupal returns a 404.

Drupal's routing system works with the Symfony HTTP Kernel.

The routing system is responsible for matching paths to controllers, and you define those relations in routes. You can pass on additional information to your controllers in the route. Access checking is integrated as well.

D8 Routes and Controllers

example.routing.yml

```
example.content:
  path: '/example'
  defaults:
        _controller: '\Drupal\example\Controller\ExampleController::content'
        _title: 'Example Route Response '
  requirements:
        _permission: 'access content'
```

D8 CMI

- **Configuration Management Initiative**
  Saving Drupal global and module settings into and reading settings from .yml files and also special **D8** cmi tables in the database

  [Configuration Mangement Initiative]

  [Principles of Configuration Management - Pt 1]

  [Principles of Configuration Management - Pt 2]

  [D8 CMI critical analysis]

Drupal 8 Blocks Plugin

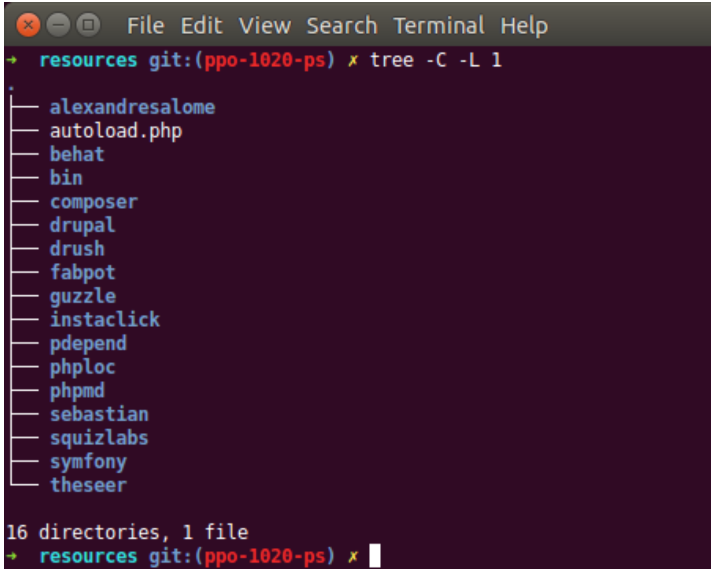
Block plugin creation overveiw.

D8 and Composer

- Composer helps you declare, manage and install dependencies of PHP projects, ensuring you have the right stack everywhere
- Composer uses .json files to keep track of the versions of php libraries and other software that you might employ in your website. Then when you need with a single composer command one can download new or update all the software

D8 and Composer

- An enterprise **D8** website with a large resources directory like shown right can download and keep current all that software with two commands:
composer -install
composer -update



```
➜  resources git:(ppo-1020-ps) ✗ tree -C -L 1
.
├── alexandresalome
├── autoload.php
├── behat
├── bin
├── composer
├── drupal
├── drush
├── fabpot
├── guzzle
├── instaclick
├── pdepend
├── phploc
├── phpmd
├── sebastian
├── squizlabs
├── symfony
└── theseer

16 directories, 1 file
➜  resources git:(ppo-1020-ps) ✗
```

Drupal Console

- **Drupal Console**
  An app that you can use to quickly make the shell structure of a **Drupal 8** module and a **Drupal 8** service within that module.

  Drupal Console on Github

  Install Drupal Console

  Drupal Console Docs

  Available Commands

Drupal 8 Console

Partial list of **D8** Console commands:

```
generate
  generate:command                    Generate commands for the console.
  generate:controller                 Generate & Register a controller
  generate:entity:config              Generate a new "EntityConfig"
  generate:entity:content             Generate a new "EntityContent"
  generate:form:config                Generate a new "ConfigFormBase"
  generate:module                     Generate a module.
  generate:plugin:block               Generate plugin block.
  generate:plugin:imageeffect         Generate image effect plugin.
  generate:plugin:rest:resource       Generate plugin rest resource
  generate:service                    Generate service
```

See all commands

D7 to D8 Module Upgrader

- **Module Upgrader**
  A **D8** module that can analyze your Drupal 7 module for needed changes and/or attempt the actual upgrade

  About Module Upgrader

  Download D8 ‘Module Upgrader’ Module

Drupal 8 Training/Resources

- Drupal knowledge training resources: Drupalize.me, Buildamodule.com Safaribooksonline.com youtube.com/user/DrupalAssociation