Drupal 8 Kickstart

An Overview for Developers

Stanford Drupal Camp 2016

Stanford, CA -- April 1 - 2, 2016

Peter Sawczynec
Customer Success Engineer

D8 Ecosystem

PANTHEON® Website Management Platform

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 Gulp Twig Compass SASS SMACSS

Guzzle Memcache Varnish CDN Service

Jenkins Chef Splunk Apache Nginx

phpStorm Sublime NetBeans

D8: Mission

- **D8**: 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

D8: Mission

PANTHEON®
Website Management Platform

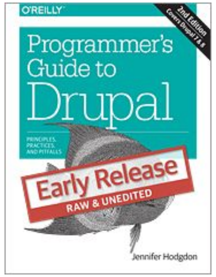
- Leverage existing industry-standard technologies so that **D8** can interface with and be programmed like other globally-recognized PHP frameworks using PHP OOP concepts
- To achieve the missions **D8** is built on top of the Symfony framework components

D8 Essential Reading

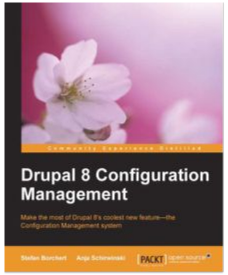
PANTHEON®
Website Management Platform

Drupal 8 API Reference [http://api.drupal.org/api/drupal/8]

Programmer's Guide to Drupal (2nd Edition)


O'REILLY
2nd Edition
Programmer's Guide to Drupal
PRINCIPLES, PRACTICES, AND PITFALLS
Early Release
RAW & UNEDITED
Jennifer Hodgdon

Drupal 8 Configuration Management



D8 Online Documentation

- Check the documentation creation date to judge the timeliness and accuracy of online **D8** documentation
- Other than docs found on Drupal.org, **D8** online documentation older than October or November 2015 is very unlikely to be fully accurate

D8 / Symfony: Special Note

PANTHEON®
Website Management Platform

D8 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...

Drush

Composer

YAML

PHP OOP

Comments / Annotations

Testing

D8 and Drush

Drush

A command line tool for managing Drupal that provides uncountable shortcut commands

- Drush executes Drupal admin tasks 3 - 10x faster than using the admin pages
- Install drush with Composer

D8 and Drush

PANTHEON®
Website Management Platform

Drush can run update.php, clear cache, log you in, change user passwords, disable/enable modules, execute sql queries, manage features.

Example drush commands:

```
drush status      drush uli      drush cc all      drush updb

drush en devel -y     drush pmi devel

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

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

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

D8 and Composer

PANTHEON®
Website Management Platform

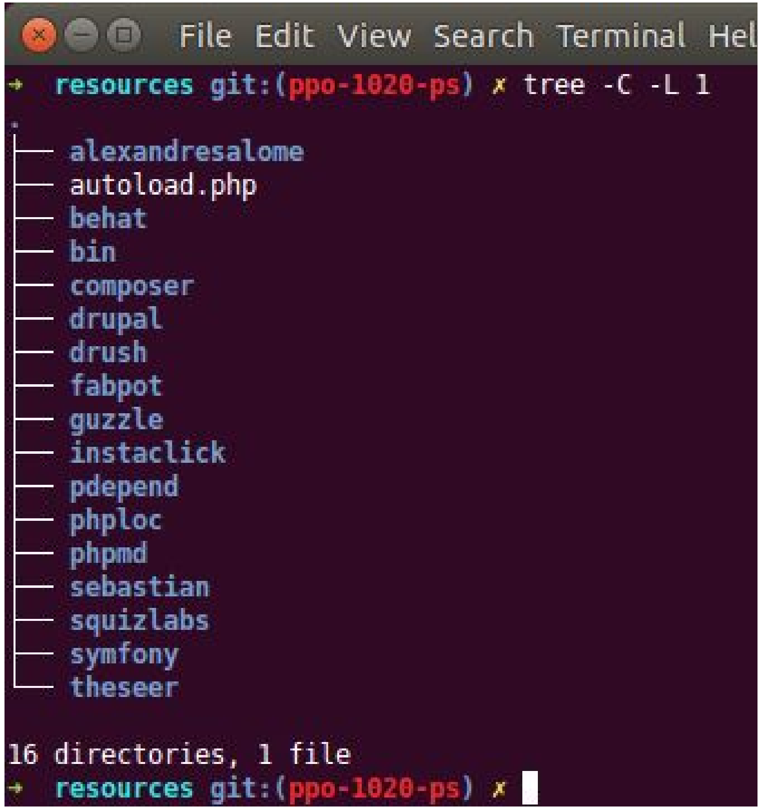
- Composer helps you declare, manage and install dependencies of PHP projects, ensuring you have the right stack everywhere
- Composer uses .json files to keep track of the versions of php libraries and other software that you might employ in your website. Then when you need with a single composer command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

D8 and Composer

PANTHEON®
Website Management Platform

An enterprise **D8** website with a large resources dir 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) ✗ 
```

D8 and Composer Manager

PANTHEON®
Website Management Platform

Composer Manager Module

Allows contributed modules and your own custom modules to manage the inclusion of PHP and other supporting libraries via Composer.

Info Documentation

D8 and YAML Files

YAML (*.yml files)

- A simple, clean format (similar to JSON) for storing structured data that is easier to read/write than XML
- YAML is a recursive acronym for: "YAML Ain't Markup Language"
- All Drupal 8 configuration is created using YAML and during installation pulled from *.yml files

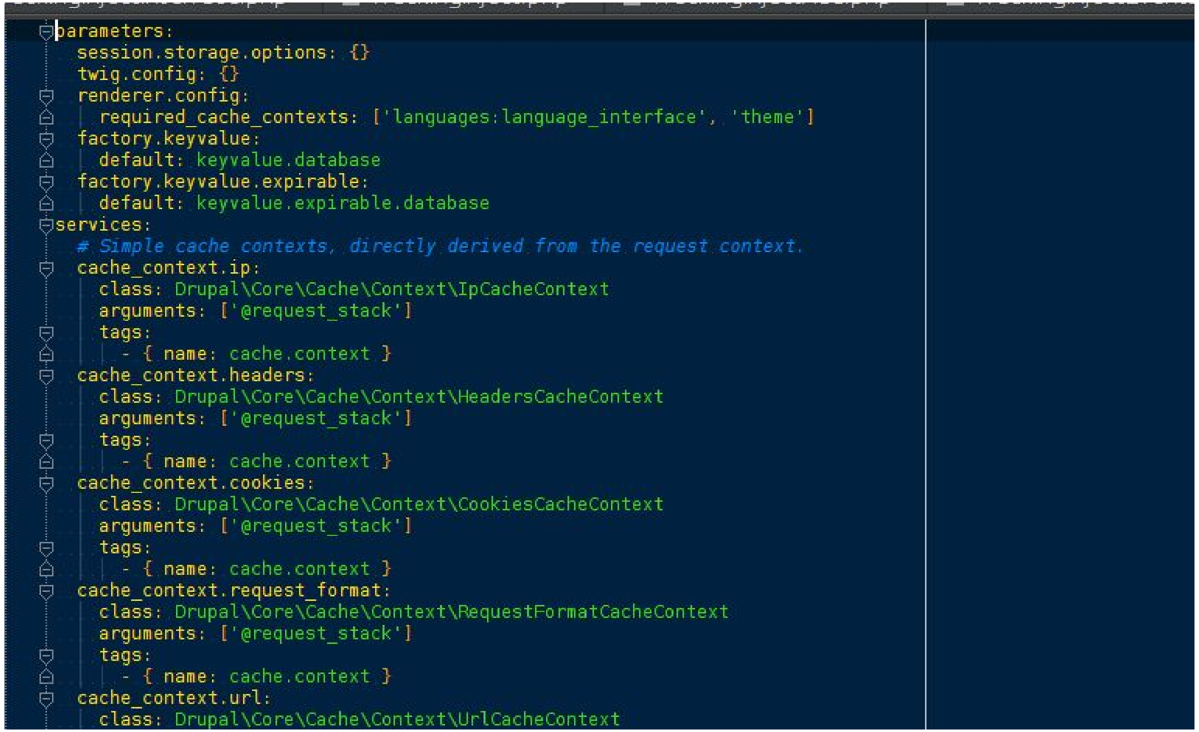
D8 and YAML Files

- YAML is case sensitive
- YAML structure is created by using indenting with spaces. YAML does not allow the use of tabs
- Use 2 spaces for YAML indenting in Drupal

Schema Files (*.schema.yml files)

- Schema files define the expected structure and allowed elements of YAML files (like DTD for XML)

D8 and YAML Files



```
parameters:
  session.storage.options: {}
  twig.config: {}
  renderer.config:
    required_cache_contexts: ['languages:language_interface', 'theme']
  factory.keyvalue:
    default: keyvalue.database
  factory.keyvalue.expirable:
    default: keyvalue.expirable.database
services:
  # Simple cache contexts, directly derived from the request context.
  cache_context.ip:
    class: Drupal\Core\Cache\Context\IpCacheContext
    arguments: ['@request_stack']
    tags:
      - { name: cache.context }
  cache_context.headers:
    class: Drupal\Core\Cache\Context\HeadersCacheContext
    arguments: ['@request_stack']
    tags:
      - { name: cache.context }
  cache_context.cookies:
    class: Drupal\Core\Cache\Context\CookiesCacheContext
    arguments: ['@request_stack']
    tags:
      - { name: cache.context }
  cache_context.request_format:
    class: Drupal\Core\Cache\Context\RequestFormatCacheContext
    arguments: ['@request_stack']
    tags:
      - { name: cache.context }
  cache_context.url:
    class: Drupal\Core\Cache\Context\UrlCacheContext
```

Sample YAML file

core-services.yml

D8 and PHP OOP

PANTHEON®
Website Management Platform

Class

- A set of functions and properties ***organized in a file*** that offer a service
- **Controllers**, **Routers**, **Forms,** and **Plugins** are all major types of classes in **D8.**
- In general all functionality created for **D8**, including your custom modules, is expected to be created in class files

D8 and PHP OOP

PANTHEON®
Website Management Platform

Interface

- A class with empty default methods that all other classes based on it must offer
- Every single method declared in an Interface will have to be implemented in the subclass. A class can implement multiple Interfaces

```
class MyClass implements ParentInterface

class MyClass implements SomeInterface, OtherInterface
```

Abstract Class

- A class with default abstract methods that classes based on it must offer
- Only Abstract methods have to be implemented by the subclass. A class can only implement one abstract class at a time.

```
class MyClass extends ParentClass
class MyClass extends ParentClass implements SomeInterface, OtherInterface
```

Trait

- A set of php functions in one file that supply a useful set of related functions

D8: Genric PHP Class Overview

PANTHEON® Website Management Platform

```
file: Mammal.php (an abstract class file)
abstract class Mammal {
        protected $age_;
        //below are functions I think all mammals will have,including people
        abstract public function setAge($age);
        abstract public function getAge();
        abstract public function eat($food);
}
```

```
file: Plane.php (an interface file)
interface Plane {
        public function Fly();
}
```

```
file: Gun.php (an interface file)
interface Gun{
        public function shoot();
}
```

D8: Genric PHP Class Overview

PANTHEON® Website Management Platform

```
file: Person.php (a class file)
use /Mammal.php
use /Person.php
use /Gun.php
class Person extends Mammal implements Plane,Gun {
        protected $job_;
        // Person features from Mammal
        public function setAge($age){ $this->age_ = $age; }
        public function getAge(){ return $this->age_; }
        public function eat($food){ echo '<br/>I eat ' ,$food ,' today<br/>'; }
        // Only a person has these feature.
        public function setJob($job){ $this->job_ = $job; }
        public function getJob(){ echo 'My job is ' , $this->job_; }
        //-----------------------------------------
        // Below methods are implementations from interfaces
        public function fly(){ echo '<br/>I use plane,so I can fly<br/>'; }
        public function shoot(){ echo 'I use gun,so I can shoot<br/>'; }
}
```

D8: Genric PHP Class Overview

PANTHEON® Website Management Platform

```
file: people_in_action.php (a regular php file)

use /Person.php

$People = new Person();

print '<pre>';
print_r(get_class_methods('People'));
print '</pre>';

$People->setAge(24);
print $People->getAge();
$People->eat('egg');
$People->setJob('PHP devepop');
print $People->getJob();
$People->fly();
$People->shoot();
```

D8 and PHP OOP

Dependency Injection

- Initiating a class, but telling the class what you want it to use to work.
- See: [What is Dependency Injection?]
  by Fabien Potencier

D8 and PHP OOP

Services

- Something a class offers, e.g. “map this node’s location by address, returns 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

D8 and Code Comments

Comments and special comments called Annotations are very important in **D8**

Properly formatted comments are used by D8 to create documentation, identify tests, and in some cases for D8 to make discovery of services and other plugin functionality

Links: Drupal Comments Annotations in Drupal

Method Chaining

Method Chaining (used by jQuery, PHP, Drupal)

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 applied to it

jQuery Method Chaining example:

```
$("#edit-button").css("color","red").slideUp(2000).slideDown(2000);
```

Method Chaining (multiline)

PANTHEON® Website Management Platform

jQuery method chaining (multiline):

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

D8 Example (multiline):

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

Above using the Database Abstraction Layer where db_update returns an UpdateQuery object

Short Array Syntax

As of PHP 5.4. In use throughout D8. The short array syntax replaces **array()** with **[]**.

```php
// Array Language Construct, Constructor function

$build = array();
$build = array(0 => 'value0');

// Short Array Syntax (requires PHP 5.4)

$build = [];
$build = [0 => 'value0'];
$build = ['value0', 'value1', 'key2' => ['value2', 'value3']];
```

Type Hinting

Use type hinting to specify the expected data type of an argument in a function declaration. When you call the function, PHP checks whether or not the arguments are of the specified type. If not, the run-time will raise an error and execution will be halted.

```
function tracking_inject_page_attachments(array &$page) {
              $tracking = new TrackingInject();
              $tracking_header_items = $tracking->getTrackingInjectCollections();
              $tracking_header = $tracking_header_items['html_head'];
              // Add each header tracking element into page HEAD area.
              if (!empty($tracking_header)) {
                foreach ($tracking_header as $tracking_header_item) {
                    ….
                }
 }

public function onResponse(FilterResponseEvent $event) {
              $response = $event->getResponse();
              if ($response instanceOf RedirectResponse && !devel_silent()) {
                ….
              }
}
```

Drupal 8

Install Drupal 8

[Create your database]

```
mkdir <new site name>
cd <new site name>
git clone http://git.drupal.org/project/drupal.git
git tag
git checkout -b tags/<tagname> <tagname>
```

[Perform Apache config updates, e.g. virtual directory]

[Start the Drupal 8 install. Pauses for settings updates]

```
cd sites/default
mkdir files
chmod 777 files
mv default.services.yml services.yml
cp default.settings.php settings.php
chmod 777 services.ymnl
chmod 777 settings.php
```

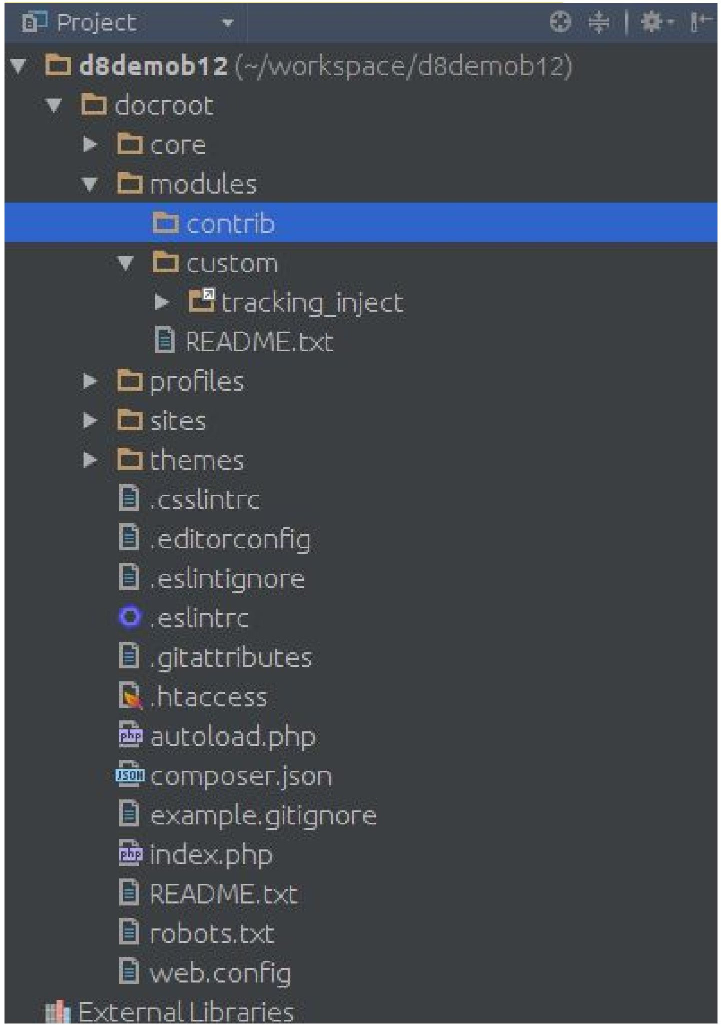
[Continue and complete Drupal 8 install]

```
chmod 655 services.ymnl
chmod 655 settings.php
cd ../../
composer install
```

D8 Top-level Directory Structure

PANTHEON®
Website Management Platform

● **/core**	Core modules and files provided by **D8**
● **/libraries**	Create yourself when needed for common 3rd party libraries, e.g. a wysiwyg editor
● **/modules**	Contrib and custom modules using sub-dirs **contrib** and **custom** (used to be sites/all/modules)
● **/profiles**	Contrib and custom profiles
● **/sites**	Site specific modules, themes and files. Including files uploaded by users, such as images. The site’s YAML configuration files, **active** and **staged**
● **/themes**	Contrib themes, custom themes and subthemes
● **/vendor**	External 3rd party libraries and projects, e.g. phpUnit, Behat


```
Project
▼ d8demob12 (~/workspace/d8demob12)
  ▼ docroot
    ▶ core
    ▼ modules
        contrib
      ▼ custom
        ▶ tracking_inject
          README.txt
    ▶ profiles
    ▶ sites
    ▶ themes
      .csslintrc
      .editorconfig
      .eslintignore
      .eslintrc
      .gitattributes
      .htaccess
      autoload.php
      composer.json
      example.gitignore
      index.php
      README.txt
      robots.txt
      web.config
  External Libraries
```

PANTHEON®
Website Management Platform

D8 top-level directory structure.

Your global custom modules reside in:

/modules/custom

Your global contrib modules reside in:

/modules/contrib

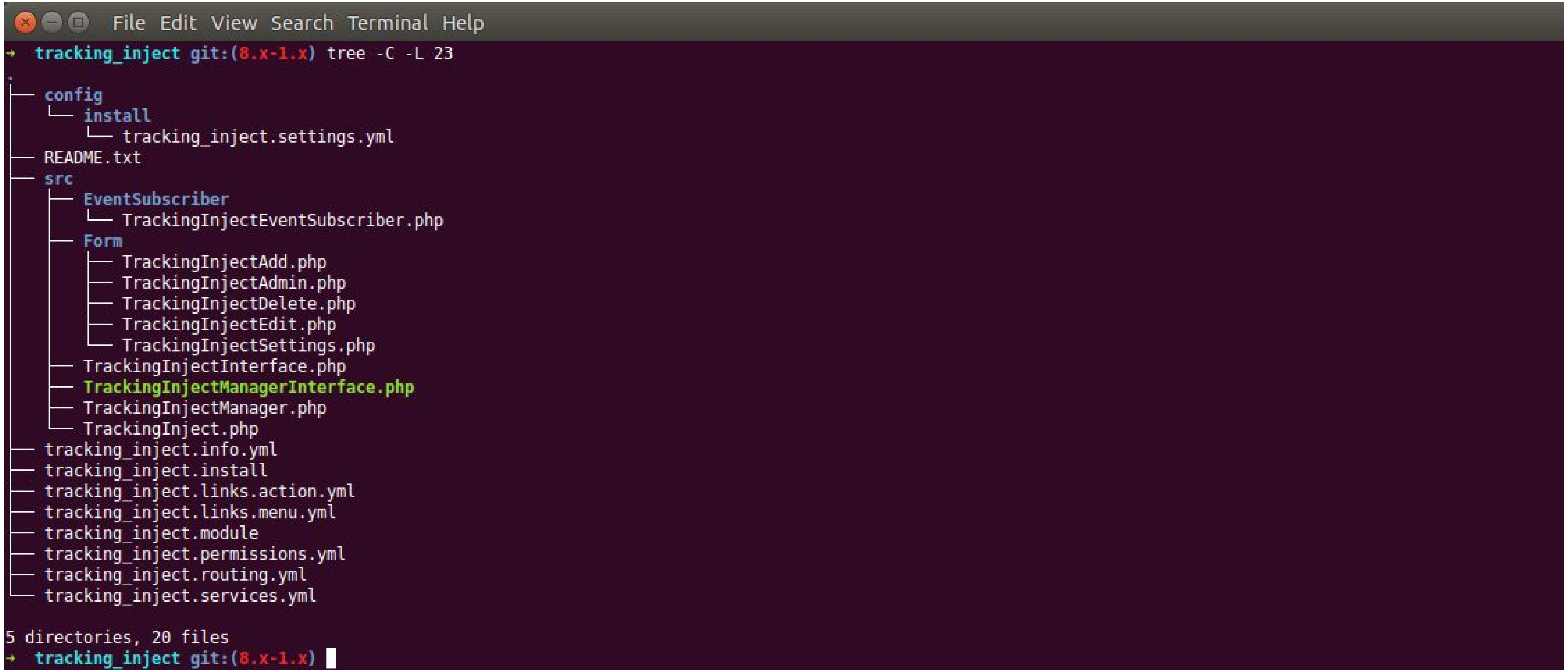
D8 Core Directory Structure

PANTHEON® Website Management Platform

Inside /core directory:

/core/assets	Various external libraries used by Core. jQuery, underscore, modernizer etc
/core/config	Configuration YAML 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

D8 Module Structure


File Edit View Search Terminal Help

```
➜  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)
```

D8 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

D8 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
7. Load other necessary include files

D8 Bootstrap

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

D8 YAML (*.yml) Files

PANTHEON® Website Management Platform

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

Pronounced: “YA-MUL” is short for: “YAML Ain’t Markup Language”

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

Sample Module Services .yml File

PANTHEON®
Website Management Platform

```
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

D8 Hooks

Request Event Example

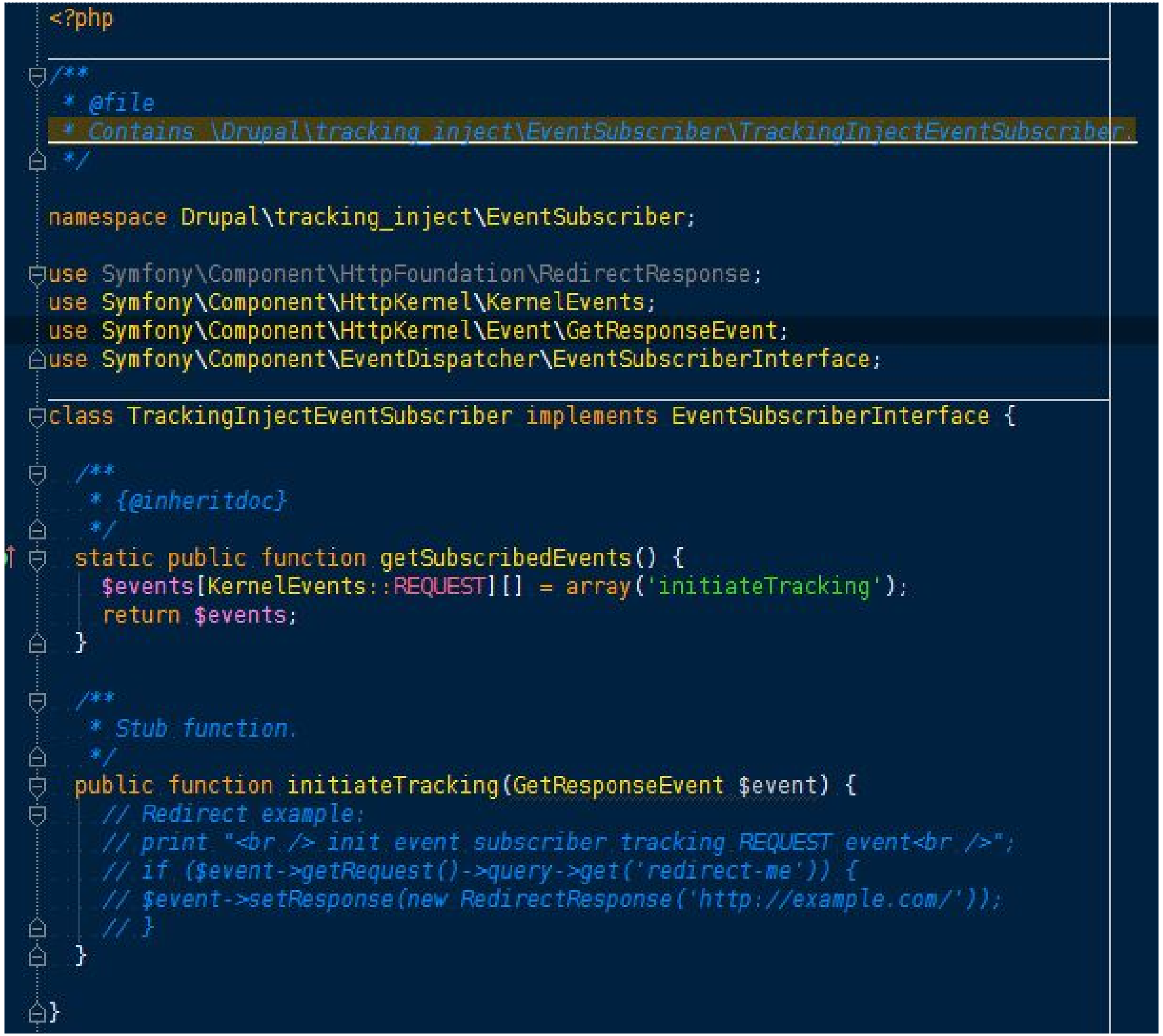
D8 Events

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

D8 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



```php
<?php

/**
 * @file
 * Contains \Drupal\tracking_inject\EventSubscriber\TrackingInjectEventSubscriber.
 */

namespace Drupal\tracking_inject\EventSubscriber;

use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpKernel\KernelEvents;
use Symfony\Component\HttpKernel\Event\GetResponseEvent;
use Symfony\Component\EventDispatcher\EventSubscriberInterface;

class TrackingInjectEventSubscriber implements EventSubscriberInterface {

  /**
   * {@inheritdoc}
   */
  static public function getSubscribedEvents() {
    $events[KernelEvents::REQUEST][] = array('initiateTracking');
    return $events;
  }

  /**
   * Stub function.
   */
  public function initiateTracking(GetResponseEvent $event) {
    // Redirect example:
    // print "<br /> init event subscriber tracking REQUEST event<br />";
    // if ($event->getRequest()->query->get('redirect-me')) {
    // $event->setResponse(new RedirectResponse('http://example.com/'));
    // }
  }

}
```

PANTHEON®
Website Management Platform

Sample **D8** EventSubscriber class file

Services in D8

Core functionality in **D8** such as current user info, current path, node info, is logged in, module exists… these are all called services

- Core services in **D8** are declared in: /core/core.services.yml
- Services can be accessed throughout **D8** via the global Drupal namespace \Drupal

Services in D8

PANTHEON®
Website Management Platform

Examples of using **D8** core services:

```
\Drupal::moduleHandler()->moduleExists('content_translation');

$account = \Drupal::currentUser();

$config = \Drupal::config('some_module.settings');
```

Services in D8

Examples of using **D8** core services:

```
$id = $config->get('domain_id');

$request = \Drupal::request();

$exception = $request->attributes->get('exception');

$status = $exception->getStatusCode();
```

D8: variable_get, variable_set

- Replaced by using a **D8** core service…
  (also understand states, settings and overrides)
- **Config** is the global **D8** configuration object and holds the changeable site or module configurations, e.g.:
  \Drupal::config('system.site') ->get('page.front');

D8: variable_get, variable_set

PANTHEON®
Website Management Platform

- Getting a variable:

```
\Drupal::config(‘module_name.settings’)->get(‘var_name’);
\Drupal::config('system.site') ->get('page.front');
```

- Setting a variable:

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

- Unsetting a variable value:

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

D8: Config vs Settings

Settings is the global D8 settings object and holds site settings like the database settings that are in settings.php.

- A get settings example:

  ```
  use\Drupal\Core\Site\Settings
  $theme = Settings::get()->('maintenance_theme', ‘bartik’);
  ```
- A set settings in settings.php example:

  ```
  $settings['maintenance_theme'] = ‘my_custom_theme’;
  ```

D8: Caching

D8 Caching has two main concepts:

Caching and Cache Invalidation

"***Caching*** is easy: it's merely storing the result of an expensive computation, to save time the next time you need it. ***Cache invalidation*** is hard: if you fail to invalidate all the things that should be invalidated, you end up with incorrect results. If you invalidate *too many* things, your cache hit ratio is going to suffer, and you'd be inefficiently using your caches. Invalidating *only the affected things* is very hard."

-- ***Wim Leers***

D8: Caching

PANTHEON®
Website Management Platform

Key concepts:

BigPipe

SmartCache

Dynamic Page Caching

Cache Keys

Cache Contexts

Cache Tags

Max-Age

References:

D8 Block Cache

Exploring the Cache API in D8

Caching in D8

Cachability of Render Arrays

Cache Contexts

D8: Caching Thought Process

1. Render Array Caching. I'm rendering something. That means think of cacheability.
   My Render Array Caching
2. Is this something that's expensive to render, and therefore is worth caching? If the answer is "yes", then what identifies this particular representation of the thing I'm rendering? Those are the **cache keys**.
   My Render Array Caching Invalidation (Cacheablility Metadata)
3. Does the representation of the thing I'm rendering vary per combination of permissions, per URL, per interface language, per … something? Those are the **cache contexts**. *Note: cache contexts are completely analogous to HTTP's Vary header.*
4. What causes the representation of the thing I'm rendering to become outdated? I.e. which things does it depend upon, so that when those things change, so should my representation? Those are the **cache tags**.
5. When does the representation of the thing I'm rendering become outdated? I.e. is the data valid for a limited period of time only? That is the **max-age** (maximum age). It defaults to "permanently (forever) cacheable" (Cache::PERMANENT). When the representation is only valid for a limited time, set a max-age, expressed in seconds. Zero means that it's not cacheable at all.

Cache contexts, tags and max-age **must always be set**, because they affect the cacheability of the entire response.
Therefore they "bubble": parents automatically receive them.
Cache keys must only be set if the render array should be cached.

D8: Caching

Cache Contexts

```
cookies
  :name
headers
  :name
ip
languages
  :type
request_format
route
  .book_navigation
  .menu_active_trails
         :menu_name
  .name
session
theme
timezone
url
  .host
  .query_args
         :key
         .pagers
         :pager_id
  .site
user
  .is_super_user
  .node_grants
         :operation
  .permissions
  .roles
         :role
```

PANTHEON® Website Management Platform

Cache max-age

What max-age allows you to do:

When **$build['#cache']['max-age']** is not set:

permanent cacheability (Cache::PERMANENT) is assumed.

To indicate that a render array is not cacheable at all, set: **$build['#cache']['max-age'] = 0** (i.e. zero seconds).

And to indicate that a render array is cacheable only for a limited amount of time, e.g. 5 minutes, set:

$build['#cache']['max-age'] = 300; // set in seconds, i.e. 300 / 60 = 5 min.

D8: Render Array w/ Caching

PANTHEON®
Website Management Platform

```
function my_module_build_array() {
$build = [
            '#prefix' => '<aside>',
            '#markup' => t('Hi, %name, welcome back to @site!', [
               ' % name' => $current_user->getUsername(),
               '@site' => $config->get('name'),
            ]),
            '#suffix' => '</aside>',
            '#theme' => 'my_module_build_arry_theme',
            '#cache' => [
                'contexts' => ['user', 'url . query_args:quantity'],
                'keys' => ['my_module_build_render', 'cache', 'demo'],
                'tags' => ['node:42:en, config . system . performance],
                'max-age' => 300,
            ],
            '#pre-render' => 'my_module_build_pre_render',
            '#attached' => [
               'library' => 'core / jquery',
               'drupalSettings' => ['foo' => 'bar'],
            ],
          ];
}
```

Render Array Reference

D8: Basic Twig

PANTHEON® Website Management Platform

Basic Variables

```
{{ title }}
```

Conditional Logic

```
{% if title %}
  <h3>{{ title }}</h3>
{% endif %}
```

Filters

```
{{ ponies|safe_join(", ")|lower }}
```

Attribibutes

Attributes is an object available to every twig template. Its job is to store all the relevent attributes of the parent container and give the themer helpful methods to interact with that data.

There should not be any space between the tag name and the twig syntax. See red text in below:

```
<div{{ attributes }}></div>

{%
  set classes = [
          'red',
          'green',
  ]
%}
<div{{ attributes.addClass(classes) }}>
  {% if options.alignment == 'horizontal' %}
    {% for row in items %}
      <div{{ row.attributes.addClass(row_classes, options.
row_class_default ? 'row-' ~ loop.index) }}>
     {% endfor %}
      </div>
  {% endif %}
```

Routing System in D8

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 Blocks Plugin

Block plugin creation overveiw.

D8 is looking for block content to be returned as render arrays.

Configuration Management Initiative

- Saving **D8** global and module settings into and reading settings from *.yml files and also special **D8** CMI tables in the database. Links:

  Configuration Mangement Initiative

  Principles of Configuration Management - Pt 1

  Principles of Configuration Management - Pt 2

  D8 CMI critical analysis

Drupal Configuration Inspector

- A module that exposes the configuration settings in use throughout your site using nice visual organization. Links:

  [Configuration Inspector Module]

Drupal Console

PANTHEON®
Website Management Platform

Drupal Console

- An app that you can use to quickly make the scaffold of a **D8** module and a **D8** service within that module. Links:

  Drupal Console on Github

  Install Drupal Console

  Drupal Console Docs

  Available Commands

D8 Console

PANTHEON® Website Management Platform

Partial list of **D8** Console commands:

generate:controller	Generate and 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:service	Generate service

See all Console commands

D7 to D8 Module Upgrader

PANTHEON®
Website Management Platform

Module Upgrader

- A **D8** module that can analyze your Drupal 7 module for needed changes and/or attempt the actual upgrade. Links:
  About Module Upgrader
  Download D8 ‘Module Upgrader’ Module

Testing

PHPUnit, Mink, Goutte headless or a browser

Above paradigm to replace SimpleTest in the long run with Behat

D8 Training/Resources

PANTHEON® Website Management Platform

Drupalize.me

Buildamodule.com

Safaribooksonline.com

http://youtube.com/user/DrupalAssociation