Intro to Backend

Drupal Training Week 5

Fatima | fkhalid@echidna.ca | @sugaroverflow

Slides: https://goo.gl/rCVSwS

Modules

Why do we use modules?

- Lots of core functionality out of the box

Types of Modules:

- Core
- Contrib
- Custom

How a Drupal site loads

Page Call Process

Request	Router	Controller	View	Response
HTTP request to the website.	Prepares the request to be processed.	Builds the response for the given request.	Creates the response.	What the website returns a the end of the call.

The Drupal 8 framework

Drupal 8 | PHP

- PHP provides a lot of useful `magic methods`

```
__call()                    _construct()
__clone()                   __destruct()
__toString()                __get()
__debuginfo()               __set()
__isset()                   __invoke()
```

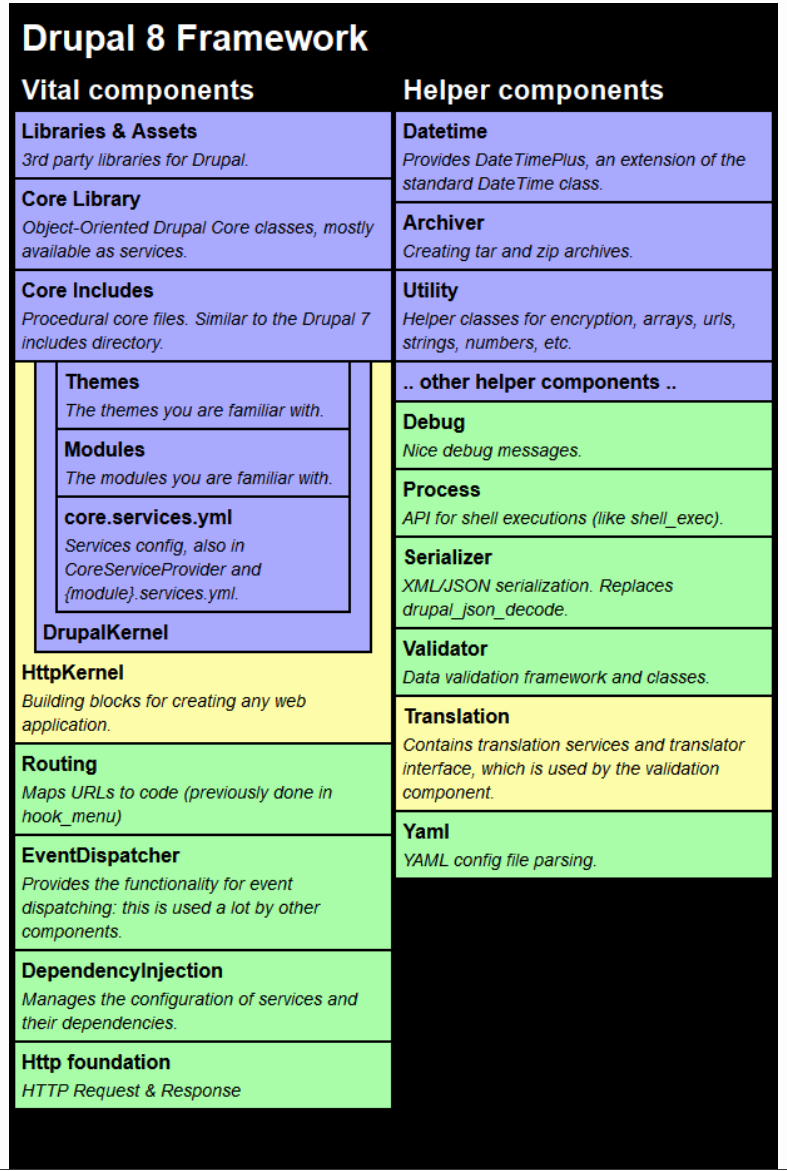
Drupal 8 | YAML

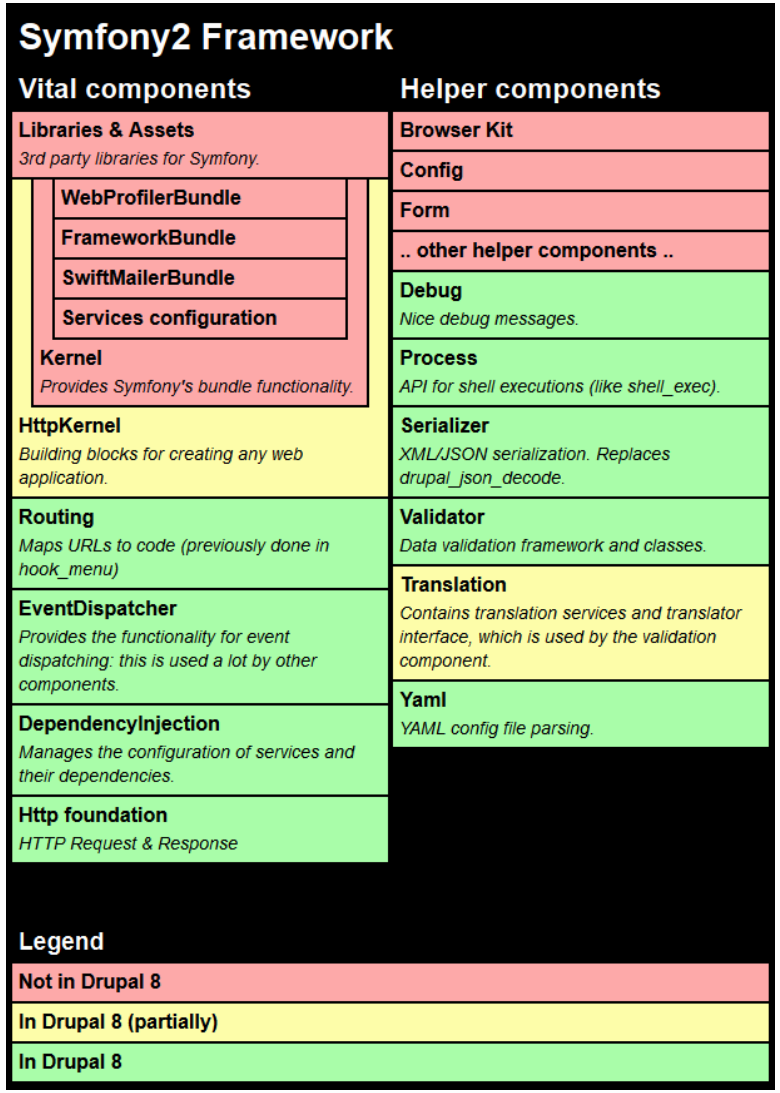
YAML Ain't Markup Language

- human-readable data serialization language
- commonly used for configuration files

```
name: 'YAML example'
type: module
description: 'Example for demonstration.'
package: Example
version: VERSION
core: 8.x
```

Drupal 8 | Symfony





Drupal 8 | Hooks

a way to alter Drupal core behavior or another module

- Understanding hooks by a real world example:
  - It's Movie Night
  - Your friend, Drupal, gets up to get a soda.
  - and asks if you want anything?

```
function drupal_soda {
  $drink_of_choice = 'soda';
  return $drink_of_choice;
}
```

```
function mymodule_soda_alter {
  $drink_of_choice = 'water';
  return $drink_of_choice;
}
```

Drupal 8 | Hooks

- `hook_form_alter`
  - Alter to output of forms before they're rendered on a page
- `hook_js_alter`
  - Performs alternations to JavaScript before it's presented
- `hook_install`
  - Perform setup tasks when the module is installed

To Find the Correct Hook:

- Look at list of hooks on api.drupal.org
- Read the documentation
- Look at other modules

Drupal 8 | OOP

Object Oriented Programming

OOP is a way to organize our code

- modular
- reusable
- flexible

```
class Shape {
  public function getArea();
}

class Circle extends Shape {
   public $radius;
}
class Rectangle extends Shape {
   public $length, $width;
}
```

OOP | Namespaces

Drupal requires you to use namespaces

- PS4 standard
- allows you to use simple class names

```php
namespace Drupal\my_module\Form;
class SignUpForm {
  //...do something
}
```

```php
namespace Drupal\drupal_training\Form;
class SignUpForm {
  //...do something
}
```

OOP | Classes

Classes are like blueprints

- Properties
  - data points
- Methods
  - class-specific functions

```
class BookOutlineForm {
  protected function actions(array $form, FormStateInterface $form_state
    $actions['delete']['#title'] = $this->t('Remove from book outline');
    return $actions;
  }
}
```

OOP | Interfaces

Interfaces are like contracts

- all methods are public
- the class must implement the methods

```
namespace Drupal\Core\Form;
interface FormInterface {
  public function getFormId();
}
```

```
namespace Drupal\my_module\Form;
class SignUpForm implements FormInterface {
  public function getFormId() {
    return 'sign_up_form';
  }
}
```

OOP | Services

services are like pluggable operations

- core services are defined in `core.services.yml`:

```
current_user:
class: Drupal\Core\Session\AccountProxy
```

- so if you want to load the current user:

```
$user = \Drupal\user\Entity\User::load(\Drupal::currentUser();
```

- you can also define custom services

OOP | Plugins

plugins are like functional lego blocks

- swappable
- different types
- can define custom ones

```
abstract class BlockBase {
    public function build() {
      // does basic stuff
    }
}
```

```
class CustomBlock extends BlockBase
  public function build() {
    // custom stuff
  }
}
```

OOP | Resources

there's so much more

Resources for learning OOP

- http://php.net/manual/en/language.oop5.php
- http://code.tutsplus.com

Resources for Drupal 8

- api.drupal.org
- www.drupal.org/documentation/develop
- http://www.drupalcontrib.org/api/drupal/8
- https://www.drupal.org/coding-standards

Workshop

My First Module

Workshop Class 5 | Overview

We are going to create our first module!

Our custom module will:

- Alter a Drupal core form
- Define a path to a custom page
- Define a menu item for that path

5.1: Create a Module

- Create your module directory
  - `/modules/custom/mymodule`
- Create a `info.yml` file
  - `/modules/custom/mymodule/mymodule.info.yml`

```
name: My First Custom Module
type: module
description: "This is the first module I've created."
core: '8.x'
package: 'Custom'
dependencies:
- book
```

- saved, FTP upload, and enable your module

5.2: Hook into a Form

Go to api.drupal.org Set to Drupal 8 Search for hook_form_alter

- Change the "Save" button label for Blog content type.
- Create a module file
  - `/mymodule/mymodule.module`
- Add a `hook_form_alter` and flush cache:

```php
<?php
use Drupal\Core\Form\FormStateInterface;

function mymodule_form_alter(&$form, FormStateInterface $form_state,
 $form_id) {
  if ($form_id === 'node_blog_form') {
    $form['actions']['submit']['#value'] = t('Save Blog');
  }
}
```

5.3 - Add Pages and Menu Items 1/4

Lets set up our route

- Create a routing file: `mymodule.routing.yml`

```
mymodule.content
  path: '/mymodule'
  defaults:
    _controller: '\Drupal\mymodule\Controller\FirstController::content'
  requirements:
    _permission: 'access content'
```

5.4 - Add Pages and Menu Items 2/4

Lets create a controller

- In `custom/mymodule`, create a `src` folder
  - in `src`, create a `Controller` folder
  - in `Controller`, create a `FirstController.php` file
- You should have:
  - `custom/mymodule/src/Controller/`
  - `FirstController/FirstController.php`
- namespace: -`\Drupal\mymodule\Controller\FirstController`

5.4 - Add Pages and Menu Items 3/4

Lets add code to our controller

- Add code to your controller
- FTP upload, clear cache, & go to `yoursite/mymodule`

```
namespace \Drupal\mymodule\Controller;
use \Drupal\Core\Controller\ControllerBase;

class FirstController extends ControllerBase {
  public function content( ) {
    return [
      '#type' => 'markup',
      '#markup' => t('This is my menu linked custom page'),
    ];
  }
}
```

5.4 - Add Pages and Menu Items 4/4

Lets add a custom menu link to our page

- Create a `mymodule/mymodule.links.menu.yml` file
- FTP upload, clear cache, & go to `yoursite`

```
mymodule.newpage:
  title: 'MyModule Stuff'
  description: 'Link to the page created by mymodule'
  route_name: mymodule.content
  menu_name: main
  weight: 100
```

Class 5 Recap

Overview of the Drupal backend

- Modules
- PHP
- YAML
- Symfomy
- the page call process
- OOP

Essential Module Building

- `info.yml` file
- `.module` file and adding a `hook_form_alter`
- creating a custom route on our site: `mymodule.routing.yml`
- creating a controller: `mymodule/src/Controller/FirstController`
- building a response in the controller for our page:`FirstController::content()`
- creating a custom menu link: `mymodule.links.menu.yml`