dropsolid
DIGITAL EXPERIENCE COMPANY

June 2025

Drupal Training

Six Group

Jasper Lammens

Drupal BE Developer

- 12 years working with Drupal
- Worked on projects as Acerta, Wit-Gele Kruis and Limburg
- Speaker at events

Six Group

Tell me!

- Yourselves, company, environment
- Your day to day jobs
- (Sitebuilding) experience?

Approach

- ~~Start by setting up a local development environment for the training~~
- Learn Drupal site building concepts
- Tailored course: room for questions and possibility to take detours from the course if wanted/needed
- Focus on Drupal 11

Agenda

Day 1	Day 2	Day 3	Day 4
Local env	Custom modules	Themes	Local debugging
Architecture and CMS	Services	Twig	Drush
Content and Content Types	Controllers	Templates	
Other entities	Plugins	User Management	
	Tests	Security	

00

Local environment

docker & ddev

ddev is a docker-based development environment

- Install Docker Desktop:
  - https://docs.docker.com/get-started/get-docker/
- Install ddev:
  - https://ddev.com/get-started/

Composer

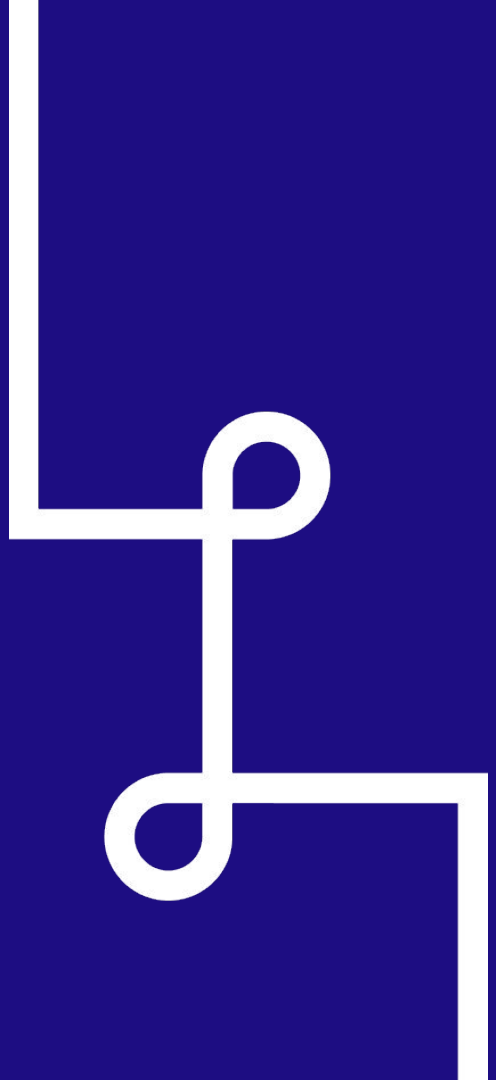
- PHP package manager
- Dependency management
- Install composer (if no ddev, otherwise pre-installed):
- https://getcomposer.org/doc/00-intro.md

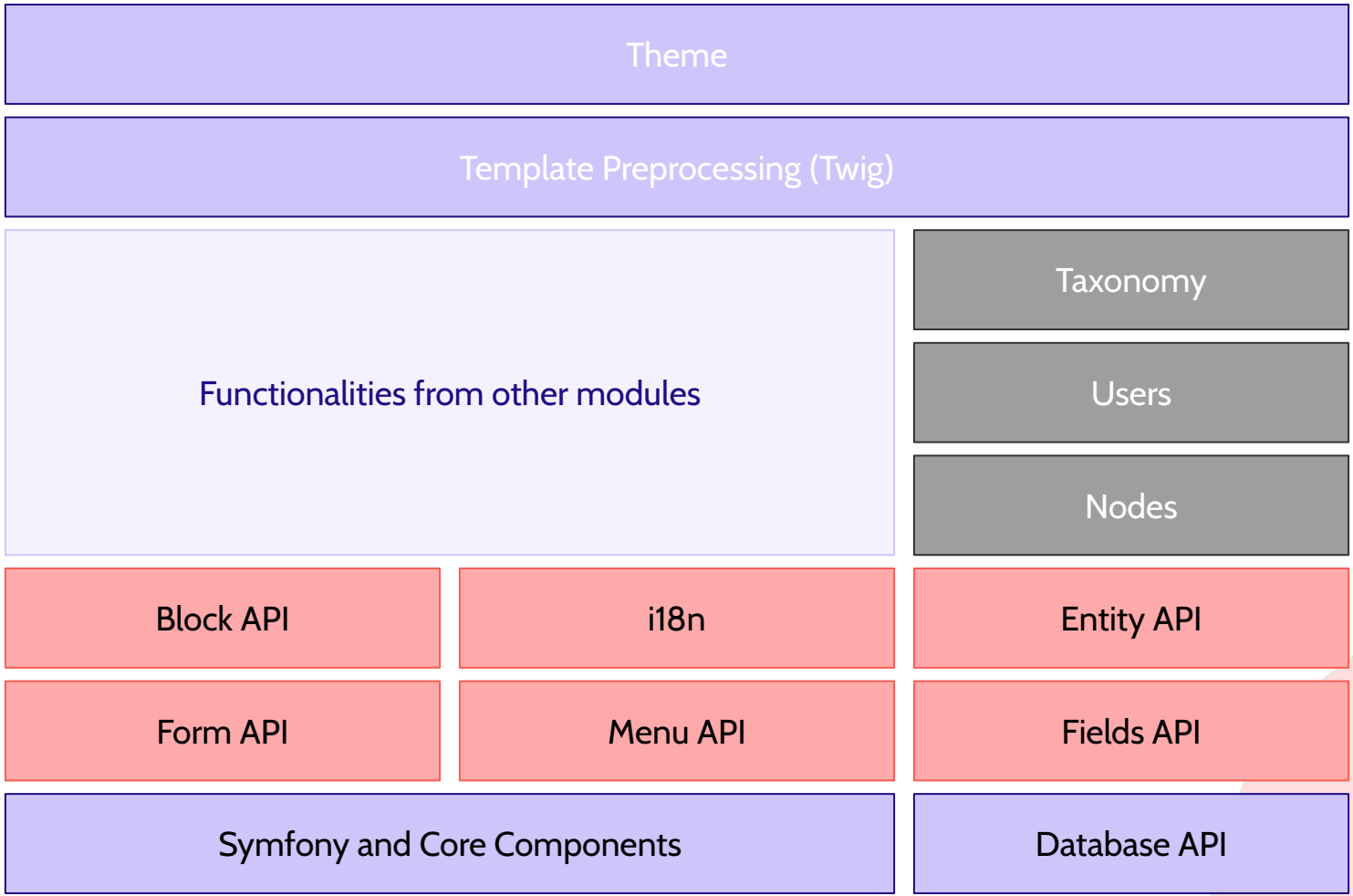
Setup project in ddev

- mkdir six_group && cd six_group
- ddev config --project-type=drupal11 --docroot=web
- ddev start
- ddev composer create drupal/recommended-project:^11
- ddev composer require drush/drush
- ddev drush site:install --account-name=admin --account-pass=admin -y
- ddev launch
- # or automatically log in with
- ddev launch $(ddev drush uli)

01

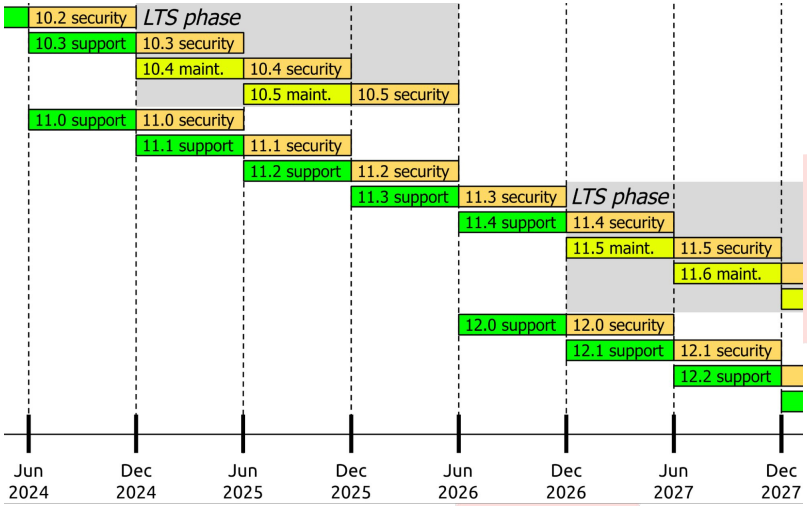
Architecture and CMS

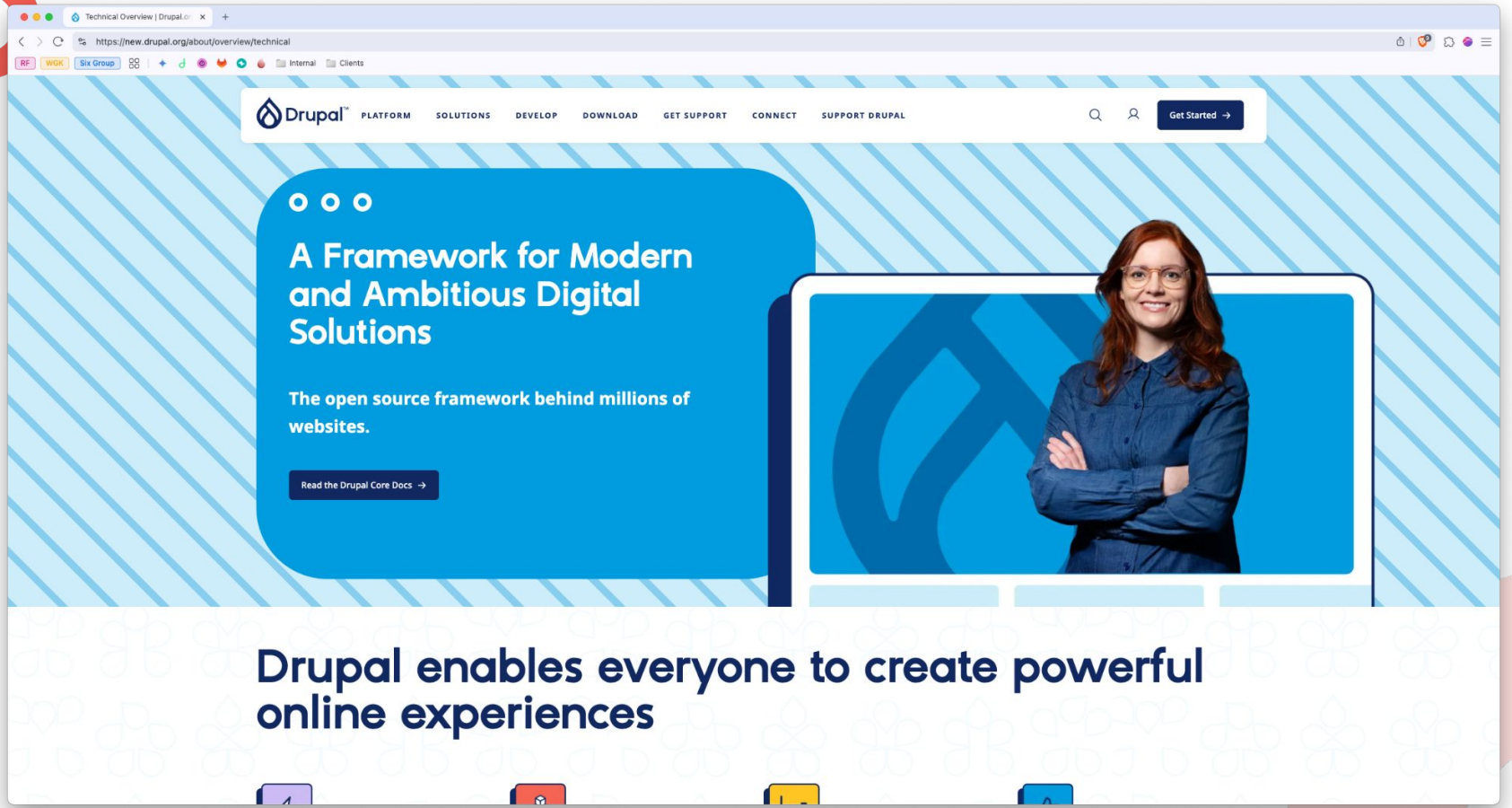



Theme
Template Preprocessing (Twig)
Functionalities from other modules
Taxonomy
Users
Nodes
Block API
i18n
Entity API
Form API
Menu API
Fields API
Symfony and Core Components
Database API

Drupal Core

- Framework for **building** solutions
- Modular architecture and API approach
- Cornerstone of a lot of projects worldwide
- The latest stable version is 11 and will be supported until 2027




Technical Overview | Drupal.org
https://new.drupal.org/about/overview/technical
RF
WGK
Six Group
Internal
Clients
Drupal
PLATFORM
SOLUTIONS
DEVELOP
DOWNLOAD
GET SUPPORT
CONNECT
SUPPORT DRUPAL
Get Started
A Framework for Modern and Ambitious Digital Solutions
The open source framework behind millions of websites.
Read the Drupal Core Docs
Drupal enables everyone to create powerful online experiences

Drupal CMS

- Built on top of Core
- Collection of best practices from the community
- Aimed at ambitious site builders
- No additional coding required
- No additional code was written to create that derivative of Core
    - Only specific configuration

Drupal CMS | Drupal.org

https://new.drupal.org/drupal-cms

Six Group RF WGK Internal Clients

Drupal PLATFORM SOLUTIONS DEVELOP DOWNLOAD GET SUPPORT CONNECT SUPPORT DRUPAL Get Started →

Drupal CMS for Ambitious Marketing

Drupal CMS puts the power of Drupal into the hands of marketers, designers and content creators.

Start building with Drupal CMS →

Read Drupal CMS documentation →

Drupal CMS is your launchpad for marketing success

02

Content and content types

Content basics

- Content type:
  - Describes the structure of “content” on the website
  - Blueprint
- Example: Basic Page (title and body)
- Content
  - title, body, ...
- ... and meta-data
  - url alias, publication status, metatags...
- **DEMO**: Create a basic page

Field types

Choose a type of field *

- **Plain text** — Text field that does not support markup.
- **Formatted text** — Text field with markup support and optional editor.
- **Number** — Field to store number. I.e. id, price, or quantity.
- **Reference** — Field to reference other content.
- **Media** — Field to reference media. Allows uploading and selecting from uploaded media.
- **File upload** — Field to upload any type of files.
- **Selection list** — Field to select from predefined options.
- **Date and time** — Field to store date and time values.
- **Boolean** — Field to store a true or false value.
- **Email** — Field to store an email address.
- **Link** — Stores a URL string, optional varchar link text, and optional blob of attributes to assemble a link.

Exercise

Create a new content type for Events

- Go to /admin/structure/types and click on the main button
- Name: Event
- Description: expose events and organize gatherings
- Go over the various options
- Ensure it’s not possible to an Event to a menu

Exercise

Create a new content type for Events

- Add a long text field for the description
- Add a date field
- Add a boolean field for indicating whether it's a remote-only event
- Add a link field to for the registration platform
- Enable the Media module
  - Go to /admin/modules and look for "Media"
  - Enable both modules
- Add a media field for the banner image

Content basics

- Additional field types available in core
- … and can be provided by modules
- Some are only available when they're downloaded from the external repository

Example of contrib field types: Address

Exercise

Create a new content type for Events

- Download and install the Address module
  - `ddev composer req drupal/address`
  - `ddev drush en address`
- Add an address field

Form View modes vs Display View modes

- You can control how content is displayed
- Different view modes (teaser, full, search index, ...)
- Fields or Layout Builder

Manage Form display

- You can control how the “edit”-page looks.
- Order fields
- “Disable” fields
- Field Group module

Manage View display

- You can control how the “view”-page looks.
- Order fields
- “Disable” fields
- Field Group module

Exercise

Configure how Events are **managed** and viewed

- Download and install the Field Group module
  - `ddev composer req drupal/field_group`
  - `ddev drush en field_group`
- Go to /admin/structure/types/manage/event/form-display
- Add a new field group for the **tabs** container
- Add a new field group for the **tab** “Content”
  - Add title, description and banner
- Add a new field group for the **tab** “Details”
  - Add date, remote-only and location

Exercise

Configure how Events are managed and **viewed**

- Go to /admin/structure/types/manage/event/display
- Enable the “Full Content” and “Teaser” view mode
- Full Content VM
  - Banner
  - Date
  - Location
  - Description
  - Registration link

Exercise

Configure how Events are managed and **viewed**

- Go to /admin/structure/types/manage/event/display
- Teaser VM
  - Banner
  - Date
  - Location

Exercise

Create your first event

- Go to /admin/content
- Click on “Add content”
- Follow the flow and provide some details about your event

03

Other entities

Media

- Handles all kinds of media items
  - Images
  - Video
  - Documents
- Great for re-use instead of re-uploading the same file

Side - Image styles

- Allows u to handle how images should be displayed
- Responsive images

Taxonomies

- Creating vocabularies and terms.
- Applying taxonomy to content.

Exercise

Add categories to Event

- Go to /admin/structure/taxonomy
- Create a new vocabulary “Event Categories”
- Add a few terms
- Go to /admin/structure/types/manage/event/fields
- Add a new reference field for using those categories

Menus

- Building primary, secondary, and custom menus.
- Menu item configuration

Config

- All settings and their structure are entities as well
- Defined as php classes
- Can be exported as yml-files
- Can be imported by another Drupal site
  - Part of the CI/CD pipeline
- Diff between staged config and what's in the config directory

Exercise

Explore and export configuration

- Go to /admin/config/development/configuration
- Notice that it's empty because we didn't export our config yet
- Open the file ./web/sites/default/settings.php
- Look config_sync_directory
- Uncomment it and assign it the value "../config/sync"
- Run `ddev drush cex -y`
- Verify the contents of new directory

Exercise

Explore and export configuration

- Go to /admin/config/system/site-information
- Change the name of the website
- Go to /admin/config/development/configuration
- Notice that a file has changed
- Run `ddev drush cex -y`
- Verify the contents of system.site.yml

Views

- What is the Views module?
- Creating lists of content (node views).
- Filtering and sorting data.
- Displaying views as blocks or pages.

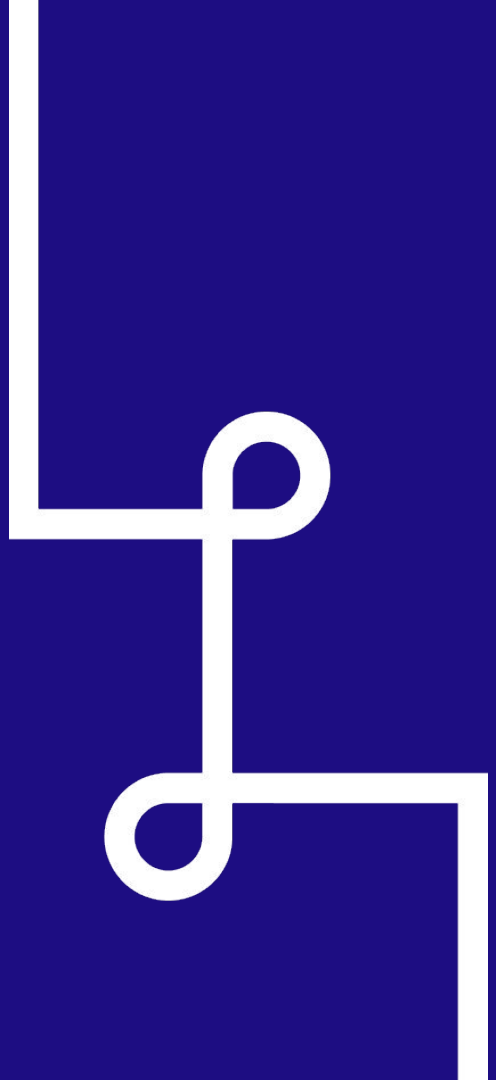
Exercise

Create an overview of Events

- Step 1: Create a page listing all Events (teasers), and add a menu item.
- Step 2: Improve the teasers
- Step 3: Only show Events that have a physical location

04

Custom Modules



Exercise

Our first custom module

- They live in web/modules/custom
- Create that directory if it doesn’t exist yet

Exercise

Our first custom module

- Use the command line to generate the boilerplate code
  - ddev drush gen module
  - Name: Pirate Weather
  - Machine name: pirate_weather
  - Description: Provides weather information based on location
  - Package: Six Group
  - No dependencies on other modules
  - No module- or install-file
  - README.md is not required but it's in the interest of others to explain what the module does

YAML

- Drupal makes extensive use of yml-files (info.yml, libraries.yml, services.yml, permissions.yml, ...)
- Yet Another Markup Language / YAML Ain't a Markup Language.
- key: value pairs
- Hierarchy (via indentation)
- Comments start with '#'
- Sequences & mappings
- https://yaml.org/

```yaml
langcode: en
status: true
dependencies:
  theme:
    - claro
id: claro_secondary_local_tasks
theme: claro
region: pre_content
weight: 0
provider: null
plugin: local_tasks_block
settings:
  id: local_tasks_block
  label: 'Secondary tabs'
  label_display: '0'
  provider: core
  primary: false
  secondary: true
visibility: {  }
```

Dependency Injection

- A service is an object managed by Services Container
- Decouple functionality
- Build upon the container system from Symfony

Dependency Injection

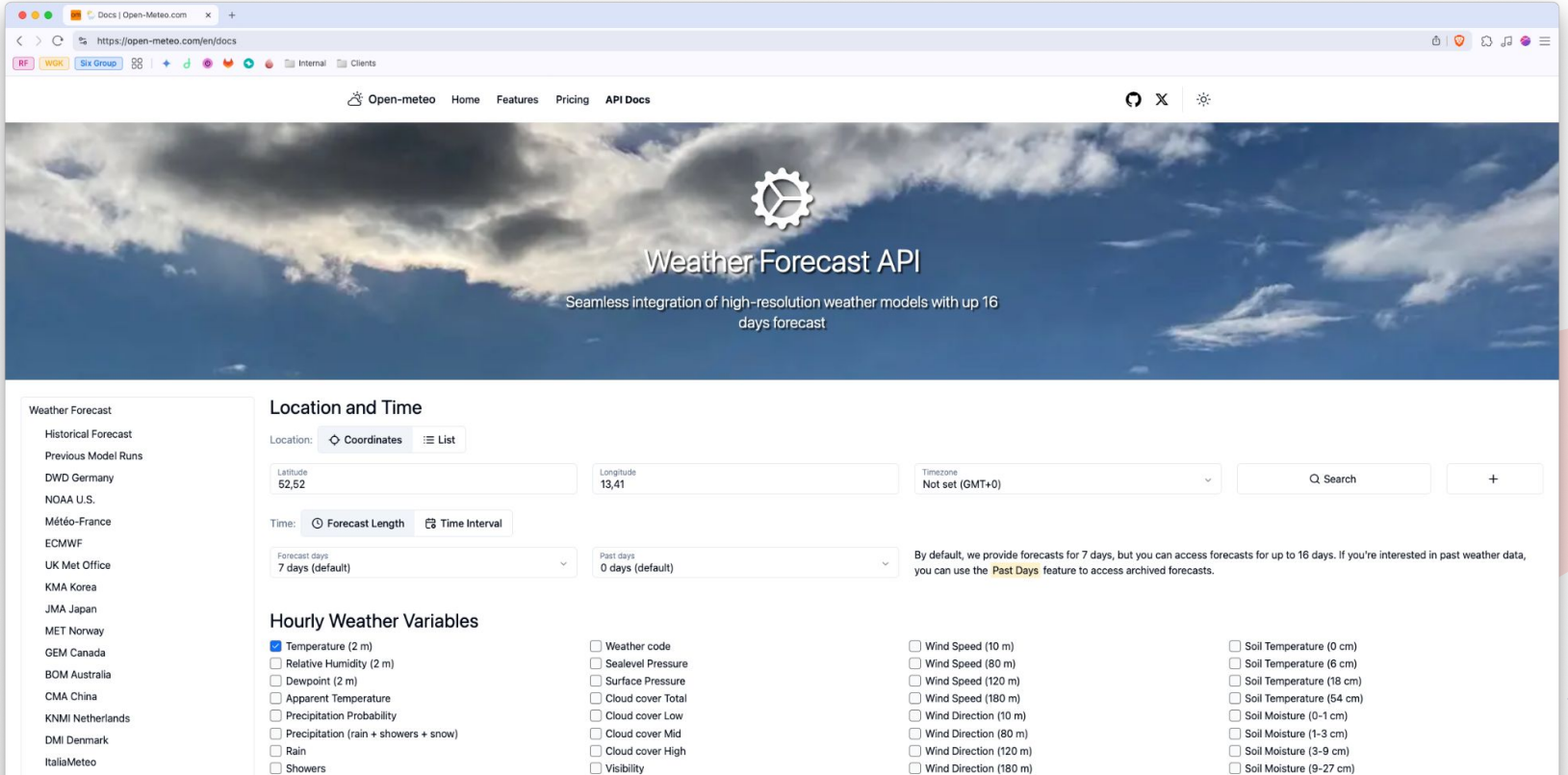
- \Drupal::service(<name>);
- \Drupal::logger() ...
- “Bad practice”, “code smell”, ...
- Beter: inject, service “is injected from the outside”
- Block plugin has access to the service container...
- ContainerFactoryPluginInterface

05

Services

API

https://open-meteo.com/en/docs



Exercise

Create Forecast service

- Use the command line to generate the boilerplate code
  - `ddev drush en pirate_weather`
  - `ddev drush gen custom-service`
  - Module machine name: pirate_weather
  - Service name: pirate_weather.services.forecast
  - Class: ForecastService
  - Let it create an interface
  - Dependencies: http_client, logger.factory

Exercise

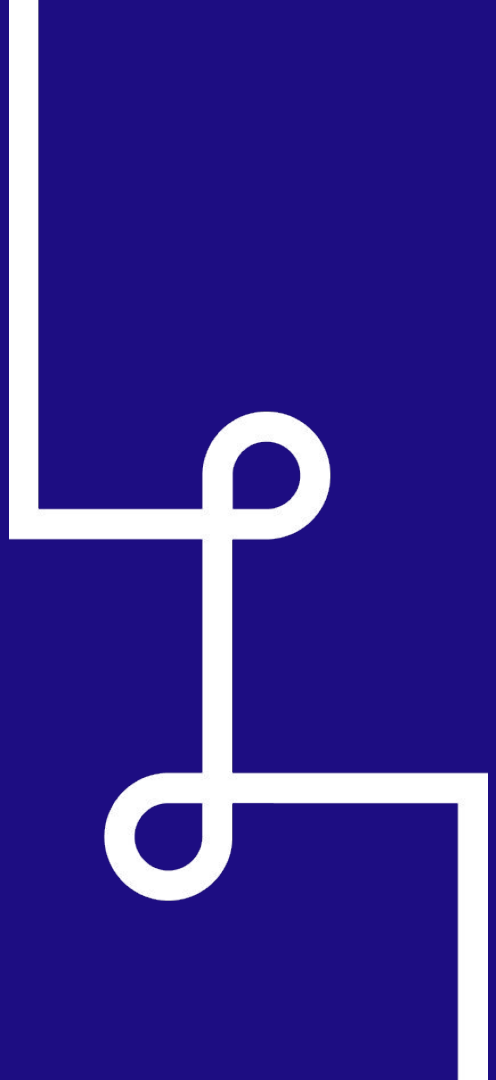
Create Forecast service

- The interface should expose one method "getForecast" and it receives the `$latitude` and `$longitude` as string parameters
- The actual class has
  - A protected constant `BASE_URL` with value "https://api.open-meteo.com/v1/forecast"
  - An implementation of the getForecast-method

```php
/**
 * Implements the Pirate Weather API for a weather forecast.
 */
no usages
final class ForecastService implements ForecastServiceInterface {

  1 usage
  protected const string BASE_URL = 'https://api.open-meteo.com/v1/forecast';

  /**
   * Constructs a ForecastService object.
   */
  no usages
  public function __construct(
    protected readonly ClientInterface $httpClient,
    protected readonly LoggerChannelFactoryInterface $logger,
  ) {}

  /**
   * {@inheritdoc}
   */
  2 usages
  public function getForecast(string $latitude, string $longitude): array {
    $url = Url::fromUri( uri: self::BASE_URL);
    $url->setOption( name: 'query', [
      'latitude' => $latitude,
      'longitude' => $longitude,
      'current' => implode( separator: ',', ['weather_code', 'temperature_2m', 'is_day']),
      'daily' => implode( separator: ',', ['weather_code', 'sunrise', 'sunset', 'temperature_2m_max', 'temperature_2m_min']),
      'models' => 'best_match',
      'timezone' => 'Europe/Berlin',
    ]);

    try {
      $response = $this->httpClient->get($url->toString());
    } catch (ClientException $e) {
      $this->logger->get('pirate_weather')->error($e->getMessage());

      return [];
    }

    return Json::decode((string) $response->getBody());
  }

}
```

06

Controllers



Controllers

- Easiest entry into Drupal
- Can be used to render html or JSON
- Define URL parameters and access

Exercise

Create Forecast controller

- Run `ddev drush gen controller`
  - Machine name: pirate_weather
  - Label: Weather Forecast
  - The suggestion for the class is OK
  - Dependencies: pirate_weather.services.forecast
  - Route name: pirate_weather.forecast
  - Route path: same as the suggestion
- Open the generated files

Render Array

- Drupal uses Render Arrays to pass renderable content.

```
7    return ['#markup' => 'Hello World!'];
```

- A render array has “properties” (start with ‘#’) and “elements” (without ‘#’).
- An element contains another render array (nesting)
- https://www.drupalatyourfingertips.com/render

```
return [
  'title' => [
    '#type' => 'html_tag',
    '#tag' => 'h2',
    '#value' => $this->t( string: 'Hello World!'),
  ],
  'body' => [
    '#type' => 'html_tag',
    '#tag' => 'p',
    '#value' => 'Our very first module.',
  ],
];
```

Exercise

Return forecast data

- Execute the getForecast method of the injected service
- Return some data as markup

```php
/**
 * Builds the response.
 */
public function __invoke(): array {
  $forecast = $this->pirateWeatherForecast->getForecast( latitude: '37.859444',  longitude: '20.624444');
  if (empty($forecast)) {
    throw new NotFoundHttpException( message: 'Could not find weather forecast.');
  }

  $build['content'] = [
    '#markup' => $this->t( string: 'Currently, the temperature is @temp @units', [
      '@temp' => $forecast['current']['temperature_2m'],
      '@units' => $forecast['current_units']['temperature_2m'],
    ]),
  ];

  return $build;
}
```

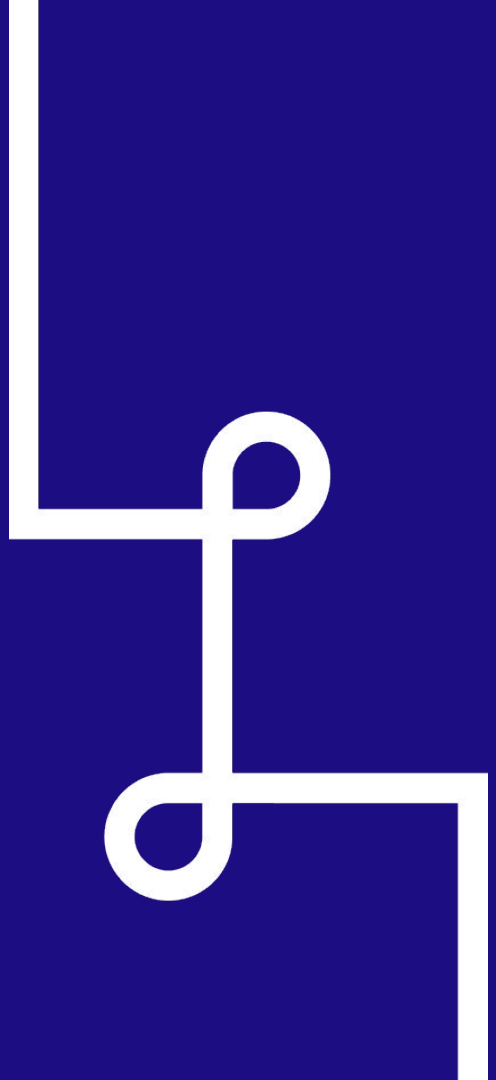
Exercise

Return forecast data

- Clear the cache
  - `ddev drush cr`
- Navigate to the route you specified

07

Plugins



Plugins

- A plugin is a piece of “swappable” functionality
- Plugin Discovery Service: “find” plugins, for example via annotations or attributes
- Plugin Factory Service: instantiating a plugin
- Example: image transformation (scale, crop, desaturate)
  - Each type acts in the same way
  - It accepts an image, performs a transformation and returns the image
- Plugin vs tagged service
  - Should be controllable by the user? Yes -> plugin, no -> service

Exercise

Visualize plugins

- Download and install Plugin
  - `ddev composer req drupal/plugin`
  - `ddev drush en plugin`
- Visit /admin/structure/plugin

Exercise

Create a Forecast block

- Run `ddev drush gen plugin:block`
  - Machine name: pirate_weather
  - Label: Weather Forecast
  - The suggestion for the Plugin ID and Class are OK
  - Category: Pirate Weather
  - Yes, the block is configurable
  - Dependencies: pirate_weather.services.forecast
  - No access callback necessary
- Open the generated php class

Exercise

Create a Forecast block

- Execute the getForecast method of the injected service
- Return the summary as markup

Exercise

Administer the coordinates

- Open the pirate_weather.schema.yml file
- Adjust the mapping of the block config
  - Add 2 new keys for latitude and longitude

Exercise

Administer the coordinates

- Open the block class and adjust the configuration
  - Specify a default set of coordinates
  - Create 2 input fields
  - Save the submitted values to the block's configuration array
- Use the saved values as input parameters for the injected service

```php
/**
 * {@inheritdoc}
 */
public function defaultConfiguration(): array {
  return [
    'latitude' => '37.859444',
    'longitude' => '20.624444',
  ];
}

/**
 * {@inheritdoc}
 */
public function blockForm($form, FormStateInterface $form_state): array {
  $form['latitude'] = [
    '#type' => 'textfield',
    '#title' => $this->t('Latitude'),
    '#default_value' => $this->configuration['latitude'],
  ];
  $form['longitude'] = [
    '#type' => 'textfield',
    '#title' => $this->t('Longitude'),
    '#default_value' => $this->configuration['longitude'],
  ];
  return $form;
}

/**
 * {@inheritdoc}
 */
public function blockSubmit($form, FormStateInterface $form_state): void {
  $this->configuration['latitude'] = $form_state->getValue('latitude');
  $this->configuration['longitude'] = $form_state->getValue('longitude');
}
```

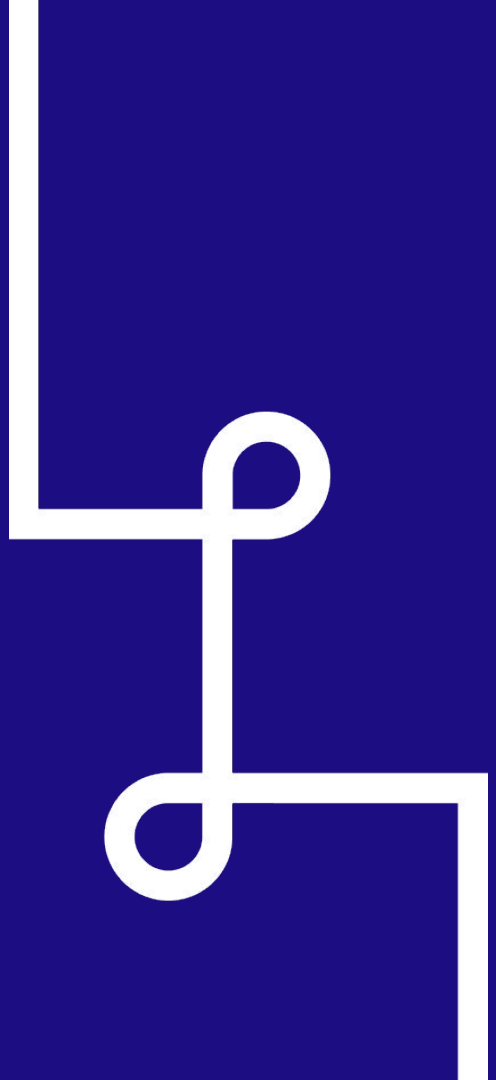
Exercise

Create a Forecast block

- Go to Block Layout (/admin/structure/blocks)
- Place the new block in the sidebar
  - Look up some coordinates of your favorite spot
- Navigate to the home page

08

Tests



Tests

- Drupal has support for various types, in order of complexity
  a. PHPUnit: clean plain php
  b. Kernel: PHPUnit test, but with bootstrapped kernel and minimal number of modules enables
  c. Functional: PHPUnit test, but with fully booted Drupal
- Also support for the Nightwatch framework
  a. Javascript testing

Exercise

Prepare Drupal for testing

- Require the dev-dependencies:
  - `ddev composer require drupal/core-dev --dev --update-with-all-dependencies`
- Copy web/core/phpunit.xml.dist to web/core/phpunit.xml
- Change configuration in web/core/phpunit.xml:
  - SIMPLETEST_BASE_URL: http://localhost
  - SIMPLETEST_DB: mysql://db:db@db/db

Exercise

Prepare Drupal for testing

- Run an existing test:
- `ddev exec ./vendor/bin/phpunit -c web/core web/core/modules/ban`

09

Themes

Templates and structure

- Drupal utilizes a robust theming layer for templating
- Themes control appearance using template files
- Templates define layout and content presentation
- Twig is the default template engine in Drupal

Core, contrib and custom themes

- **Core Themes** like Olivero and Claro are located in `core/themes`.
- **Contributed Themes** should be placed in `themes/contrib`, which Composer uses as the default path, facilitating updates and management.
- **Custom Themes** should be placed in `themes/custom`, segregating site-specific customizations from community contributions.
- **Separation** between admin interface and frontend.

Exercise

Download themes from the public repository and use them accordingly: admin theme “Gin”

- `ddev composer req drupal/gin`
- Go to /admin/appearance
- Install Gin
- Select as administration theme
- Explore the different settings
- Go to /admin/modules
- Install **Gin Toolbar**

Exercise

Download themes from the public repository and use them accordingly: FE theme “Bootstrap”

- `ddev composer req drupal/bootstrap`
- Go to /admin/appearance
- Install Bootstrap **and set as default**
- Explore the different settings

Exercise

Download themes from the public repository and use them accordingly: enable debug mode

- Go to /admin/config/development/performance
- Disable all aggregation options
- Go to /admin/config/development/settings
- Enable all options

```
<div class="views-view-responsive-grid__item">
  <div class="views-view-responsive-grid__item-inner">
    <!-- THEME DEBUG -->
    <!-- THEME HOOK: 'node' -->
    <!-- FILE NAME SUGGESTIONS:
       ▪ node--view--frontpage--page-1.html.twig
       ▪ node--view--frontpage.html.twig
       ▪ node--1--card-common.html.twig
       ▪ node--1.html.twig
       ▪ node--recipe--card-common.html.twig
       ▪ node--recipe.html.twig
       ▪ node--card-common.html.twig
      ✅ node.html.twig
    -->
    <!-- 💡 BEGIN CUSTOM TEMPLATE OUTPUT from
```

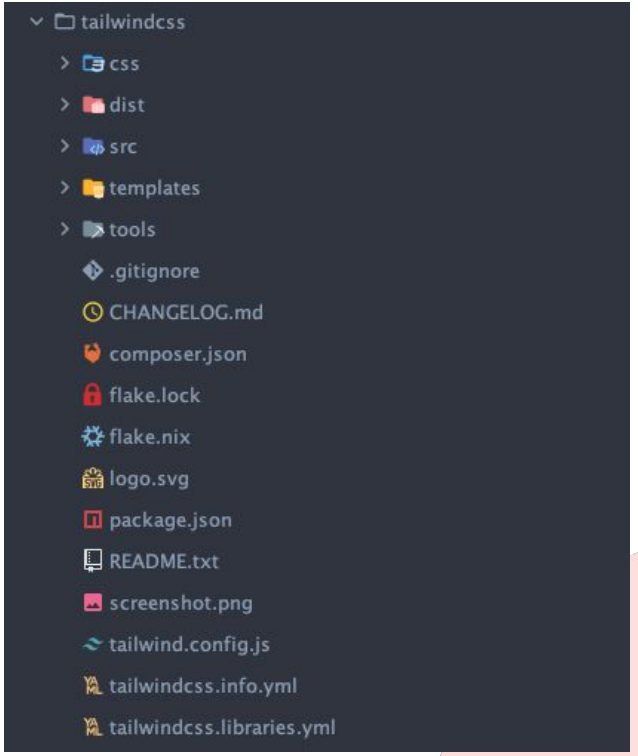
Exercise

Generate your own theme

- Run `ddev drush gen theme`
  - Name: Six Group
  - Base theme: stable9 *
  - The suggestion for the description is OK
  - Package: Six Group
  - “Yes” to breakpoints and “Yes” theme settings
- Open the generated files
- Go to /admin/appearance and set the theme as default

Anatomy of a theme

```
|-my_new_theme.breakpoints.yml
|-my_new_theme.info.yml
|-my_new_theme.libraries.yml
|-my_new_theme.theme
|-config
|   |-install
|   |   |-my_new_theme.settings.yml
|   |-schema
|   |   |-my_new_theme.schema.yml
|-css
|   |-my_new_theme.css
|-js
|   |-my_new_theme.js
|-images
|   |-buttons.png
|-logo.svg
|-screenshot.png
|-templates
|   |-maintenance-page.html.twig
|   |-node.html.twig
```



Exercise

Add Tailwind CSS

- Enter the web container (`ddev ssh`) and navigate to the theme directory (`./web/themes/custom/six_group`)
- Run `npm init`
  - Name: six_group
  - Description: see six_group.info.yml
  - All the other suggestions are OK
- Remove livereload: `npm remove livereload`
- Install Tailwind CSS: `npm install -D tailwindcss @tailwindcss/cli @tailwindcss/postcss postcss autoprefixer`

Exercise

Add Tailwind CSS

- Configure PostCSS (see postcss.config.js on GH)
- Add the build-shortcut to your package.json file
  - See package.json in sol/day/3 on GH
- Create a new .css file in the src-directory (does not exists yet)
  - See ./src/index.css for the contents (GH)
- Run `npm run build`
- Take a look at the generated css file (./dist/index.css)

Exercise

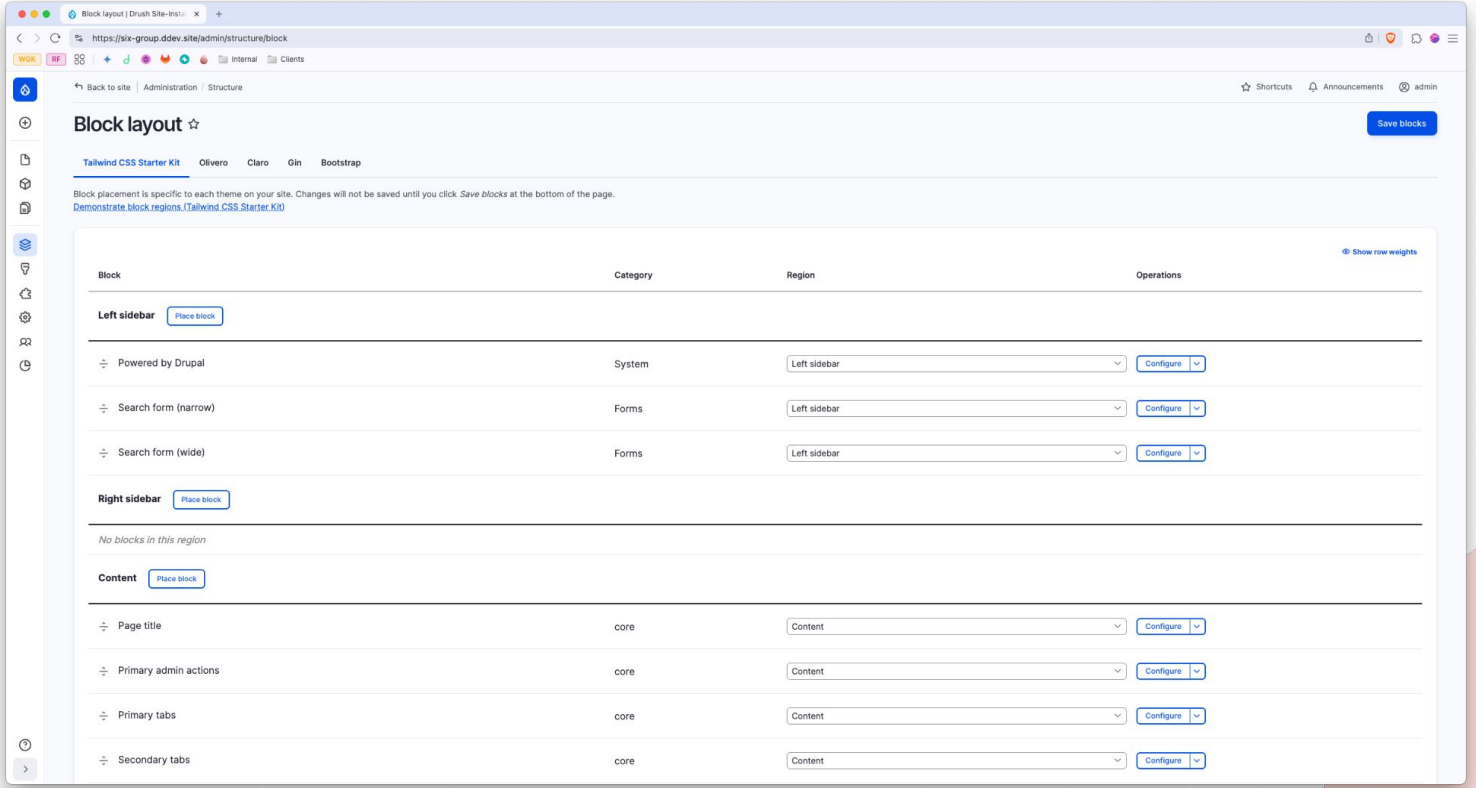
Add Tailwind CSS

- Register the generated css file as a library
  - See six_group.libraries.yml
- Use the library when the theme is activated
  - See six_group.info.yml

Blocks

- Small separate pieces of content **OR** functionality
- Can be placed in a region
- Content blocks vs Code blocks
- Regions are defined by the theme

Regions


Block layout | Drush Site-Insta
https://six-group.ddev.site/admin/structure/block
Internal
Clients
Back to site
Administration / Structure
Shortcuts
Announcements
admin
Block layout
Save blocks
Tailwind CSS Starter Kit
Olivero
Claro
Gin
Bootstrap
Block placement is specific to each theme on your site. Changes will not be saved until you click Save blocks at the bottom of the page.
Demonstrate block regions (Tailwind CSS Starter Kit)
Show row weights
Block
Category
Region
Operations
Left sidebar
Place block
Powered by Drupal
System
Left sidebar
Configure
Search form (narrow)
Forms
Left sidebar
Configure
Search form (wide)
Forms
Left sidebar
Configure
Right sidebar
Place block
No blocks in this region
Content
Place block
Page title
core
Content
Configure
Primary admin actions
core
Content
Configure
Primary tabs
core
Content
Configure
Secondary tabs
core
Content
Configure

Exercise

Show some love for Drupal

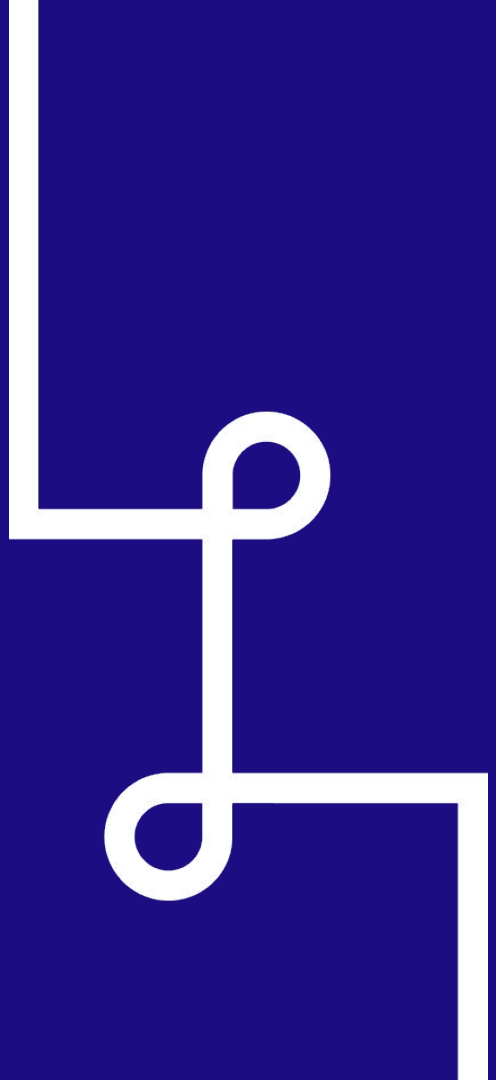
- Go to /admin/structure/block
- Add the “Powered by Drupal”-block to the footer region
- Do not show the title
- Only render it for logged in users

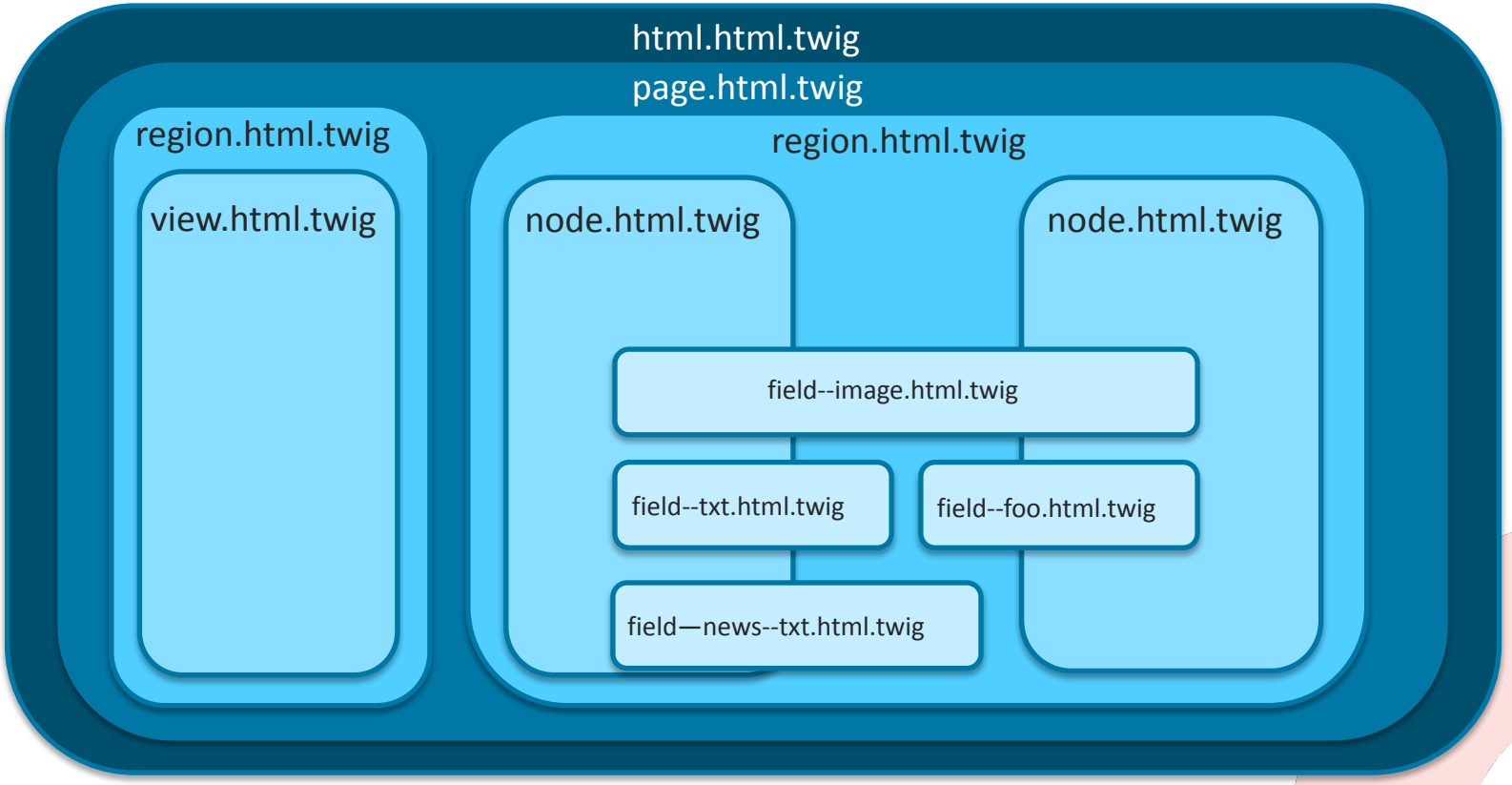
Exercise

Describe who you are

- Go to /admin/content/block
- Add a new block
- Fill in the title and the body
- Go to /admin/structure/block
- Add the new block to the Right Sidebar region

10
Templates




html.html.twig
page.html.twig
region.html.twig
view.html.twig
region.html.twig
node.html.twig
node.html.twig
field--image.html.twig
field--txt.html.twig
field--foo.html.twig
field—news--txt.html.twig

Overriding templates

Can be in a module or a theme

Call the same template name or use suggestions

Template suggestions

Template suggestions have their own dedicated theme_hook

hook_theme_suggestions_alter()

hook_theme_suggestions_HOOK_alter()

```php
/**
 * Implementation of hook_theme_suggestions_entitytype_alter().
 */
function dropsolid_theme_suggestions_node_alter(array &$suggestions, array $variables, $hook) {
  $suggestions[] = 'hello--world';
}

/**
 * Implementation of hook_theme_suggestions_alter().
 */
function dropsolid_theme_suggestions_alter(&$suggestions, $variables, $base_theme_hook) {
  if ($base_theme_hook == 'ds_1col') {
    $suggestions[] = 'hello-world';
  }

  if (isset($variables['content'])) {
    // It is a node.
    // Set a reference to the content.
    $content = &$variables['content'];
    if (isset($content['#view_mode']) && $content['#view_mode'] == 'full'
      && isset($content['#bundle']) && $content['#bundle'] == 'news'
    ) {
      $suggestions[] = 'foo-bar';
    }
  }
}
```

```
<!-- THEME DEBUG -->
<!-- THEME HOOK: 'node' -->
<!-- FILE NAME SUGGESTIONS:
   * node--view--frontpage--page-1.html.twig
   * node--view--frontpage.html.twig
   * node--1--teaser.html.twig
   * node--1.html.twig
   * node--article--teaser.html.twig
   * node--article.html.twig
   * node--teaser.html.twig
   x node.html.twig
-->
<!-- BEGIN OUTPUT from
'core/themes/classy/templates/content/node.html.twig' -->

<article data-history-node-id="1" data-quickedit-entity-
id="node/1" role="article" class="contextual-region node node--
type-article node--promoted node--view-mode-teaser"
about="/node/1" typeof="schema:Article" data-quickedit-entity-
instance-id="0">
....
</article>

<!-- END OUTPUT from
'core/themes/classy/templates/content/node.html.twig' -->
```

Extend/Include Twig Files

Extending of twig files can be used to make small changes to a «parent» template file.

Child templates use this method to add attributes / classes, printed in the parent template.

Child templates can optionally override the parent's block content.

Include literally includes one template in the other

```
{% extends "field.html.twig" %}

{% set attributes = attributes.addClass('clearfix', 'text-formatted') %}

```

```
{% extends "block.html.twig" %}
{#
/**
 * @file
 * Theme override for tabs.
 */
#}
{% block content %}
  {% if content %}
    <nav class="tabs" role="navigation" aria-label="{{ 'Tabs'|t }}">
      {{ content }}
    </nav>
  {% endif %}
{% endblock %}

```

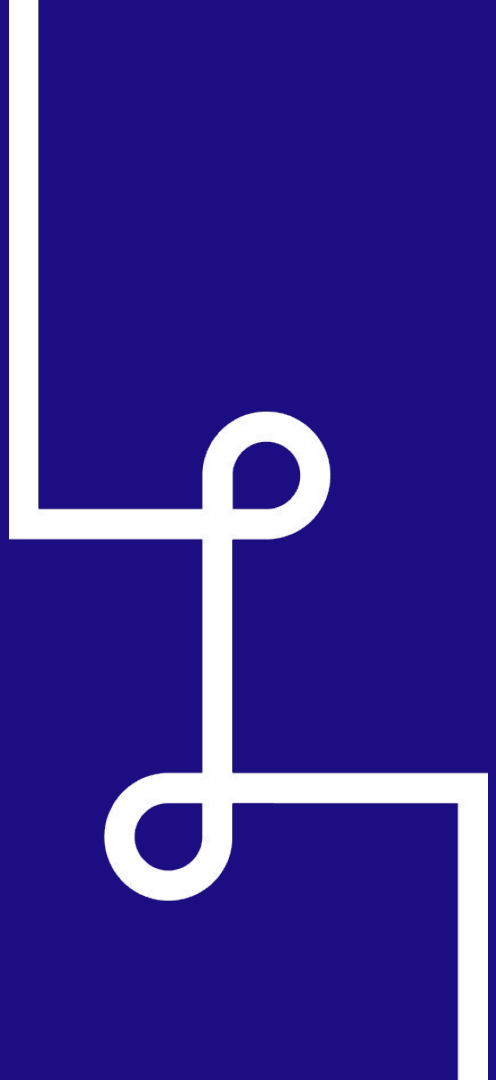
Exercise

Add a border and some styling for the page title

- Look for the theme suggestions of the page title block
- Create a new template file, based on the suggestions
  - block--page-title-block.html.twig
- Add the necessary Tailwind class
  - `attributes.addClass('page-title')`

11

Twig



Comments

Twig comments are written between curly braces with hashes

{# comment #}

```
{#
/**
 * @file
 * Theme override to display a node.
 *
 * Available variables:
 * - node: The node entity with limited access to object properties and methods.
 *   Only method names starting with "get", "has", or "is" and a few common
 *   methods such as "id", "label", and "bundle" are available. For example:
 *   - node.getCreatedTime() will return the node creation timestamp.
 *   - node.hasField('field_example') returns TRUE if the node bundle includes
 *     field_example. (This does not indicate the presence of a value in this
 *     field.)
 *   - node.isPublished() will return whether the node is published or not.
 *   Calling other methods, such as node.delete(), will result in an exception.
 *   See \Drupal\node\Entity\Node for a full list of public properties and
 *   methods for the node object.
 */
 #}
```

Print Markup

The twig template files use the dot markup.

```
51    {{ page.primary_menu }}
52    {{ page.secondary_menu }}
53
54    {{ page.breadcrumb }}
55
56    {{ page.highlighted }}
```

This is also useful to grab the value directly from certain elements

```
57
58    {{ content.field_intro.0['#markup'] }}
59
```

Statements

You can add if-statements and even use the not keyword to inverse the result.

You can also loop over items and use the indexes.

{% for key, item in items %}

{% for item in items %}

```
{% if not page %}
  <h2{{ title_attributes }}>
    <a href="{{ url }}" rel="bookmark">{{ label }}</a>
  </h2>
{% endif %}
```

```
{# Now generate the actual pager piece. #}
{% for key, item in items.pages %}
  <li class="pager__item{{ current == key ? ' is-active' : '' }}">
    {% if current == key %}
      {% set title = 'Current page'|t %}
    {% else %}
      {% set title = 'Go to page @key'|t({'@key': key}) %}
    {% endif %}
    <a href="{{ item.href }}" title="{{ title }}"
            {{ item.attributes|without('href', 'title') }}>
      <span class="visually-hidden">
        {{ current == key ? 'Current page'|t : 'Page'|t }}
      </span>
      {{- key -}}
    </a>
  </li>
{% endfor %}
```

Merge

The merge filter can append an array to an existing array variable.

Useful when adding classes to view rows.

```
27
28    {% if loop.first %}
29      {% set row_classes = row_classes|merge(['first-run']) %}
30    {% endif %}
31    {% if loop.last %}
32      {% set row_classes = row_classes|merge(['last-run']) %}
33    {% endif %}
34
35    {% set row_classes = row_classes|merge([cycle(['odd','even'], loop.index0)]) %}
36
```

http://twig.sensiolabs.org/doc/filters/merge.html

Split & Join

The split filter splits a string by the given delimiter and returns a list of strings.
http://twig.sensiolabs.org/doc/filters/split.html

```
"one,two,three"|split(',') %}
{# foo contains ['one', 'two', 'three'] #}
```

The join filter returns a string whichs is a concatenation of the items of a sequence.
http://twig.sensiolabs.org/doc/filters/join.html

```
{{ [1, 2, 3]|join }}
{# returns 123 #}
```

```
{{ [1, 2, 3]|join('|') }}
{# outputs 1|2|3 #}
```

Default Filter

The default function returns the passed default value if the value is undefined or empty, otherwise the value of the variable

http://twig.sensiolabs.org/doc/filters/default.html

```
{{ var|default('var is not defined') }}

{{ var.foo|default('foo item on var is not defined') }}

{{ var['foo']|default('foo item on var is not defined') }}

{{ ''|default('passed var is empty')  }}
```

Translation Filters

The translation filter is a Drupal defined filter.

It is possible to combine filters and functions to translate a string and replace an argument with a url.

```
<a href="{{ url('<front>') }}" title="{{ 'Home'|t }}" rel="home"
class="site-logo"></a>
```

```
101
102 {{ 'Create a <a href="@url">news page</a>.'|t({'@url': url('node/add/news')}) }}
103
```

Passthrough Filters

The passthrough filter is a Drupal defined filter.

By default all printed content is sanitized by Drupal, the passthrough filter bypasses that.

{{ variable|passthrough }}

This will print the variable as-is.

Without Filters

The without filter is mainly applied on the content in a twig file, but can be applied on all kind of variables with sub-items.

{{ content|without('links') }}

This will result in printing the content without the links element

Clean Class & Id Filters

The filters will format the class and ID as a valid css item.

```
40  {%
41      set classes = [
42          'field',
43          'field--name-' ~ field_name|clean_class,
44          'field--type-' ~ field_type|clean_class,
45          'field--label-' ~ label_display,
46      ]
47  %}
```

```
36
37  {% set heading_id = attributes.id ~ '-menu'|clean_id %}
38
```

Format Date Filters

This filter will format a timestamp as a proper date.

You can specify the format (defined in Drupal) as an argument.

```
34        {% if updates %}
35          <ul>
36            {% for update in updates %}
37              <li>{{ update.name }} ({{ update.timestamp|format_date('html_date') }})</li>
38            {% endfor %}
39          </ul>
40        {% endif %}
```

Render Filter

This filter is a wrapper for the render function. If a variable contains a render array, this filter can be applied to generate it's output.

```
17
18  {% set left = left|render %}
19  {% set right = right|render %}
20
```

Variables

Print attributes in the article tag

Define a classes variable inside the template with a setter.

Define a variable in the preprocess hook in *.theme file

```
<article{{ attributes }}>Lorem ipsum</article>

{%
  set classes = [
    'node',
    node.isPromoted() ? 'node--promoted',
    node.isSticky() ? 'node--sticky',
    not node.isPublished() ? 'node--unpublished',
  ]
%}
```

```
/**
 * Implements hook_preprocess_node().
 */
function mytheme_preprocess_node(&$variables) {
  $variables['my_variable'] = 'My custom variable';
}
```

Attribute Functions

We can use twig function to add or remove classes to / from attributes.

.addClass
.removeClass

It is also possible to check for a certain class in the attributes.

.hasClass

If you want to add / remove an ID to / from element attributes.

.setAttribute
.removeAttribute

```
{# Single class #}
<div{{attributes.addClass('my-class')}}></div>

{# Multiple classes #}
{%
  set classes = [
    'red',
    'green',
  ]
%}
<div{{ attributes.addClass(classes) }}></div>
```

```
<div{{ attributes.addClass(classes).removeClass('green') }}></div>
```

```
{% if attributes.hasClass('myClass') %}
  {# do stuff #}
{% endif %}
```

```
<div{{ attributes.setAttribute('id', 'myID') }}></div>
```

```
<div{{ attributes.removeAttribute('id') }}></div>
```

Loop Functions

There are special functions for loops.

cycle([array], index)
devisibleby(index)

Useful to add classes on an overview

```
35  {% for row in rows %}
36    {%
37      set row_classes = [
38        default_row_class ? 'views-row',
39        default_row_class ? 'views-row-' ~ loop.index,
40        loop.first ? 'first-loop',
41        loop.last ? 'last-loop',
42        cycle(['first', 'second', 'third'], loop.index0),
43        cycle(['odd', 'even'], loop.index0),
44        loop.index is divisibleby(5) ? 'views-row--clear',
45        'reverse-' ~ loop.revindex,
46        'reverse-machine-' ~ loop.revindex0,
47      ]
48    %}
49    {% if loop.index is divisibleby(5) %}
50      {# do something #}
51    {% endif %}
52    <div{{ row.attributes.addClass(row_classes,loop_classes) }}>
53      {{ row.content }}
54    </div>
55  {% endfor %}
```

Twig xdebug

Install https://www.drupal.org/project/twig_xdebug

Enable a breakpoint in a template that is rendered on your page and view the context.

{{ breakpoint() }}

Twig documentation

Official https://twig.symfony.com/doc/3.x/

Drupal: https://www.drupal.org/docs/develop/theming-drupal/twig-in-drupal

Extra filters and Functions

Exercise

Show copyright information in the footer

- In page.html.twig of our custom theme
- Make sure that the footer is always printed
- Render the copyright icon and the site name
- Render the current date via functions

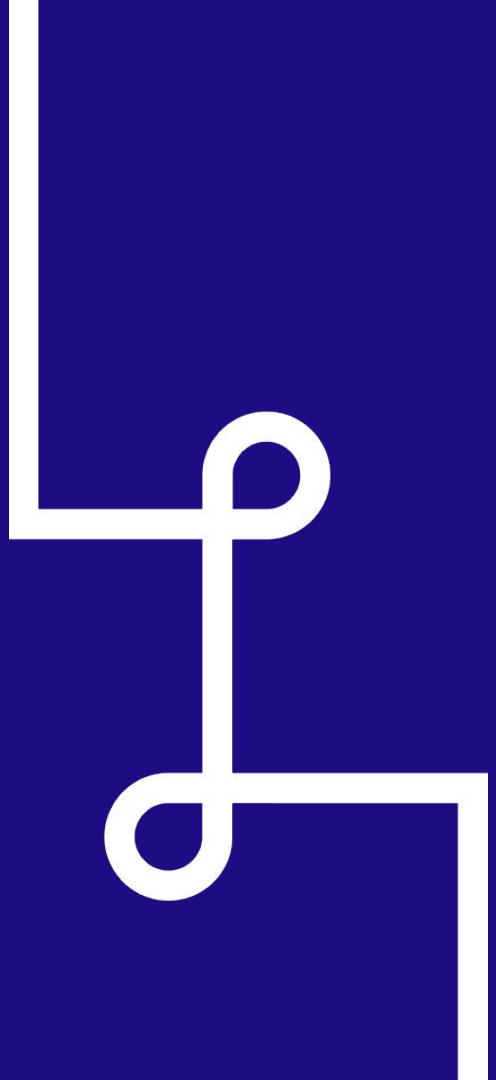
Exercise

Render the forecast of the Pirate Weather module via twig

- Create a new hook class `PirateWeatherHooks` and implement the `theme`-hook
  - Register theme `forecast_long`-theme
  - Create a template file with the same name, but replace underscores with hyphens
- Use the new theme in the controller and inject the necessary parts of the response

12

User Management



Roles and permissions

- Permissions are assigned to roles (not users)
- Users are assigned zero, one or more roles
- ‘Automatic’ roles:
  - Authenticated
  - Anonymous
- We can use the Masquerade module to debug permission issues by switching to another user (no password required)
  - https://www.drupal.org/project/masquerade

User 1

- Root user
- No access / permission checks (!)
- Best practice: use webmaster / admin accounts instead

Exercise

Create an Article Editor

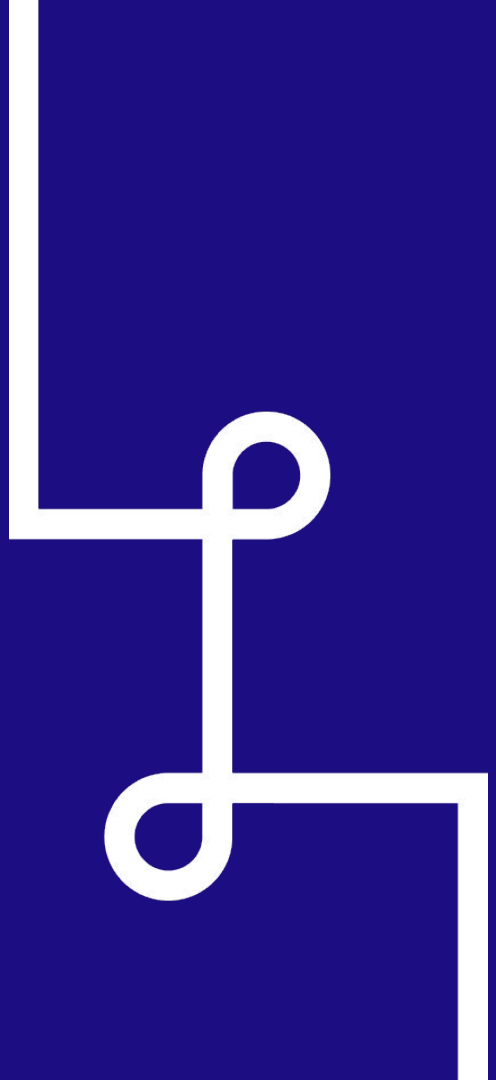
- Go to /admin/people/roles
- Create a new role called “Article Editor”
- In the hierarchy, he’s just below the general Content editor
- Go to /admin/people/permissions
- Give him the necessary permissions to alter the Article content type
    - Hint: use the search bar
- Make sure he can use the admin interface
- Go to /admin/people
- Create a new user and assign him that role

Exercise

Create an Article Editor

- Download and install drupal/masquerade
  - `ddev composer req drupal/masquerade`
  - `ddev drush en masquerade`
- Go to /admin/people and use the new option for the new user

13
Security



In code

- Prevent Cross Site Scripting (XSS) attacks by using Twig templates
  - `{{ }}` sanitizes content
  - Use placeholders when translating content
    - `t('I am @variable', ['@variable' => 'Six'])`
- Use API functions from Drupal
  - t(), Url-helpers, cleanCssIdentifier
- Use the database abstraction layer to avoid SQL Injection
  - Bad: `\Database::getConnection()->query('SELECT foo FROM {table} t WHERE t.name = '. $_GET['user']);`
  - Good: `\Database::getConnection()->query('SELECT foo FROM {table} t WHERE t.name = :name', [':name' => $_GET['user']]);`

Via contrib modules

- Password management to impose a policy: length, numbers, special characters, expiry...
- Authentication improvements via 2FA, SSO providers
- SPAM control via reCAPTCHA or Honeypot
- IP blocklists
- Various other modules:
  - Security Kit (seckit) for applying best practices on http-level (Content-Security-Policy, Strict-Transport-Security etc.)
  - Security Review (security_review) tests various settings and flags insecure ones
  - Paranoia (paranoia) can evaluate PHP via Drupal's web interface and then block those

Exercise

Create and apply a password policy

- Download and install drupal/password_policy
  - `ddev composer req drupal/password_policy`
  - `ddev drush en password_policy`
- Go to /admin/config/security/password-policy and create a new password policy
  - Name: Default
- `No password constraint available!`
  - Plugin type “PasswordConstraint”
  - Submodules contains implementations

Exercise

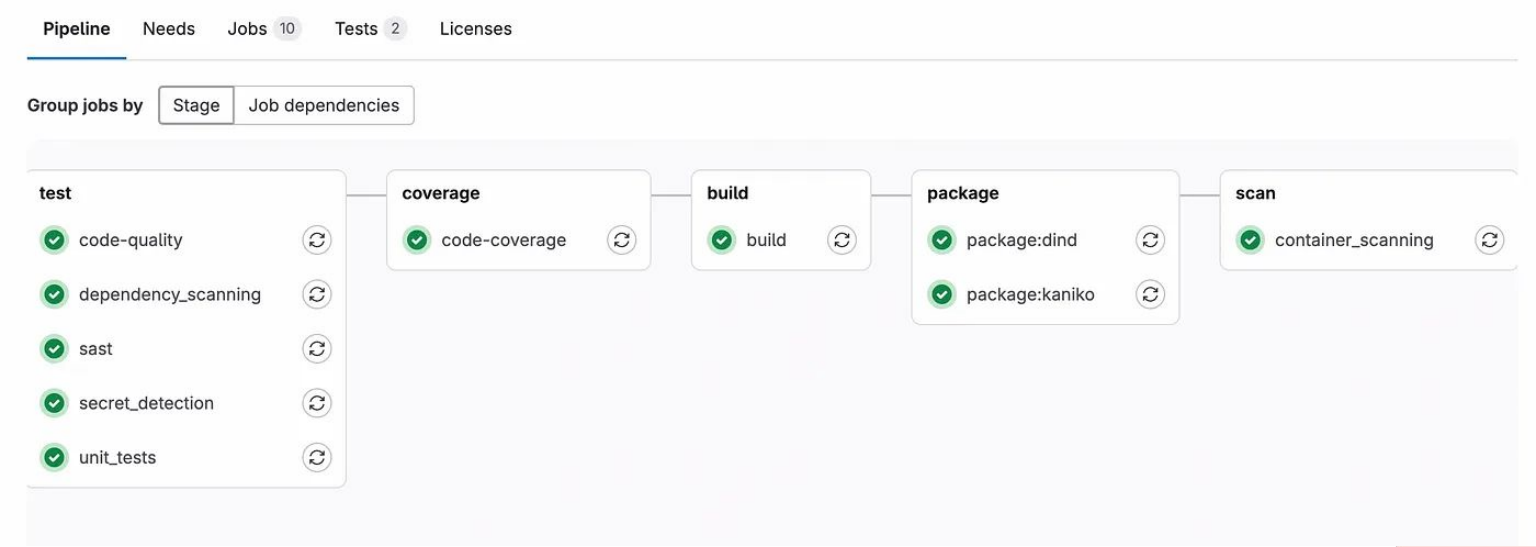
Create and apply a password policy

- Enable submodules “character_types”, “characters”, “length”, “history” and “username”
  - The machine name should be prefixed with “password_policy_”
- Go to /admin/config/security/password-policy and adjust the existing policy
  - For every enabled submodule, add a new constraint

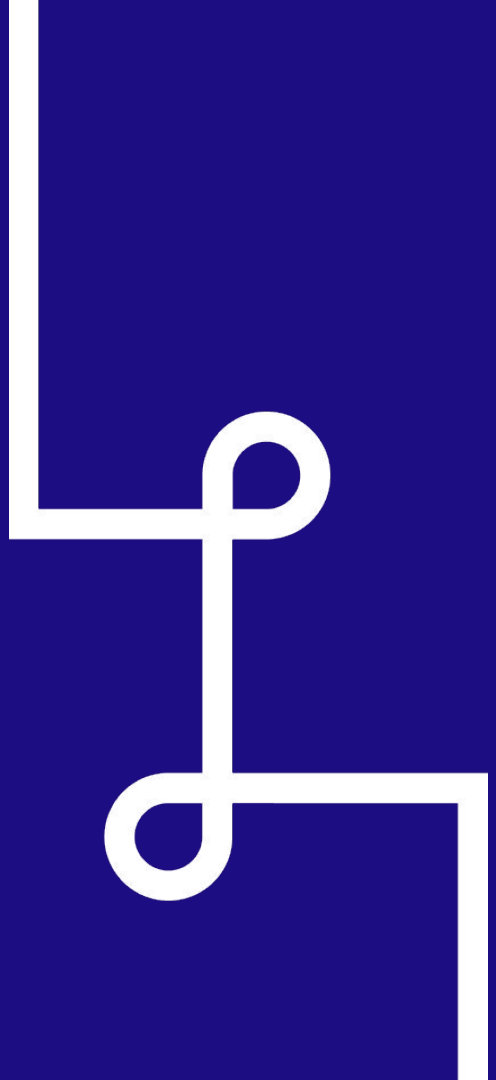
Dependencies

- Static Application Security Testing
  - SQL injection, XSS, insecure API usage
  - Bearer CLI or Semgrep
- Dynamic Application Security Testing
  - Scans a running web application for vulnerabilities
  - OWASP ZAP
- Software Composition Analysis (Dependency Scanning)
  - `composer audit`
- Container image scanning
  - Trivy
- Infrastructure as Code Scanning
  - Checkov or Terrascan
- https://www.youtube.com/watch?v=fZ1dLb0DmTc

Dependencies



14
Debugging



3 levels of code debugging

Native

Using some of the builtin methods of PHP, a lot can be discovered already.

Devel

A contrib module which contains a bunch of debug dependencies and helper methods.

Web Profiler

Look at stats of the current page and solve different kind of issues

Exercise

Use var_dump and backtrace

- var_dump the response from our ForecastService
  - `var_dump($forecast);`
  - In the controller
- Look at the backtrace when the service is being executed
  - `var_dump(debug_backtrace(DEBUG_BACKTRACE_IGNORE_ARGS));`
  - In ForecastService.php

Exercise

Use Devel

- Download and install Devel
  - `ddev composer req drupal/devel [dash][dash]dev`
  - `ddev drush en devel`
- Go through the different options in the backend
- Use dpm and dd for debugging the response of the service
  - dpm($response);
  - dd($response);
  - In the block

Exercise

Use Webprofiler

- Download and install Webprofiler
  - `ddev composer req drupal/webprofiler [dash][dash]dev`
  - `ddev drush en webprofiler`
- Open a page in the frontend
- Look at the different debug options

Profiling

Get a feeling of what is going on with your Drupal website.

Xdebug

Add breakpoints in your code and go through its execution, step by step.

XHProf

Analyses the used classes and runtimes. It's great for discovering bottlenecks and points to improve the code.

Blackfire

Same as XHProf, but exposed as a SaaS product by Platform.sh.

Exercise

Use xdebug to step through our service

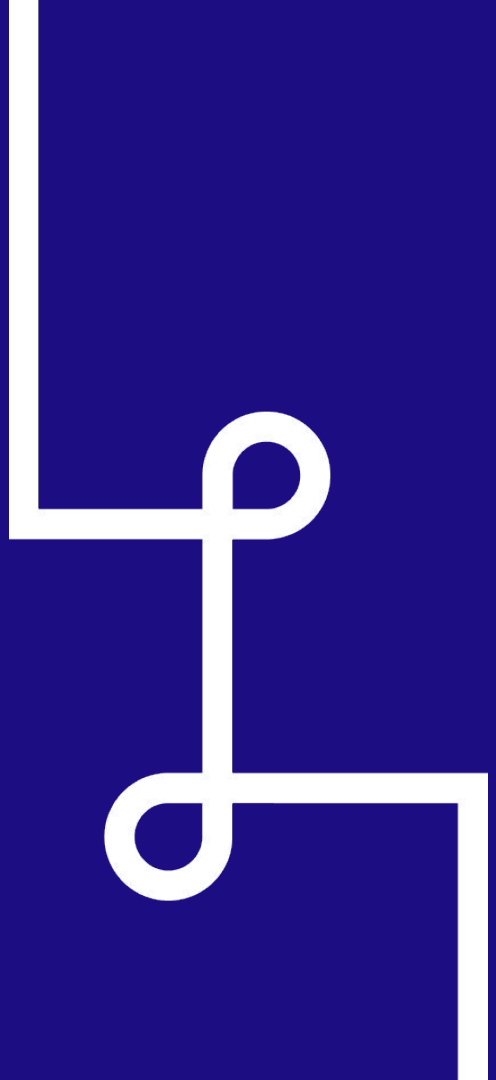
- Enable xdebug
  - `ddev xdebug`
- Place a breakpoint on line 32 in ForecastService, where the URL is being created
- Toggle the listening mode of PhpStorm
- Open the homepage

Exercise

Use XHProf to discover bottlenecks

- Enable XHProf
  - `ddev config global --xhprof-mode=xhgui`
  - `ddev restart`
- Open the UI
  - `ddev xhgui`
- Navigate around the website
- Reload the UI and look at the different metrics

15
Logging



Logs

What has happened in the past?

Watchdog

Drupal's internal log system. It can use the database (dblog) or the server itself (syslog).

Monolog

A PHP framework for handling logs in a more architectural way, deciding when and where to store logs.

Sentry.io

Send your logs and metrics to a remote system and analyse them there.

Exercise

Discover Watchdog

- Make sure that dblog is installed
- Go to /admin/reports/dblog
- Play around with the different filter options

Monolog

Handlers

A component to determine what to do with a log message. Eg. save it in the database, send it to Slack or write it in a file.

Formatters

It alters the message format so it can be handled correctly. Eg. line formatter or json formatter.

Processors

Processors can add metadata to a log message, such as the current user, the http request etc.

Exercise

Discover Monolog

- Download and install Monolog
  - `ddev composer req drupal/monolog`
  - `ddev drush en monolog`

Exercise

Discover Monolog

- Enable the private directory
  - Alter ./web/sites/default/settings.php
  - Line 615:
    `$settings['file_private_path'] = '../private';`
- Configure Monolog to store logs in a separate file
  - `mkdir -p ./private/logs`
  - Copy monolog.services.yml from GH to ./web/sites/default
  - Enable the use of the monolog.services.yml file by adding
    `$settings['container_yamls'][] = 'sites/default/monolog.services.yml';` to settings.php (after line 718)
- Clear the cache

Exercise

Discover Monolog

- Test the setup by uninstall the Announcements module
  - `ddev drush pmu announcements_feed`
- Verify that a new log file is created in the private directory

Sentry.io

Spans

A basic unit of a trace. Multiple spans with the same parent ID, make up a trace. Think a single database transaction.

Traces

View records of operations being performed by the website. Eg. a page load.

Insights

Discover slow pages by examining queries, queues, caches etc.

Others

Monitoring

A contrib module that provides a 360 degree perspective on Drupal's system health.

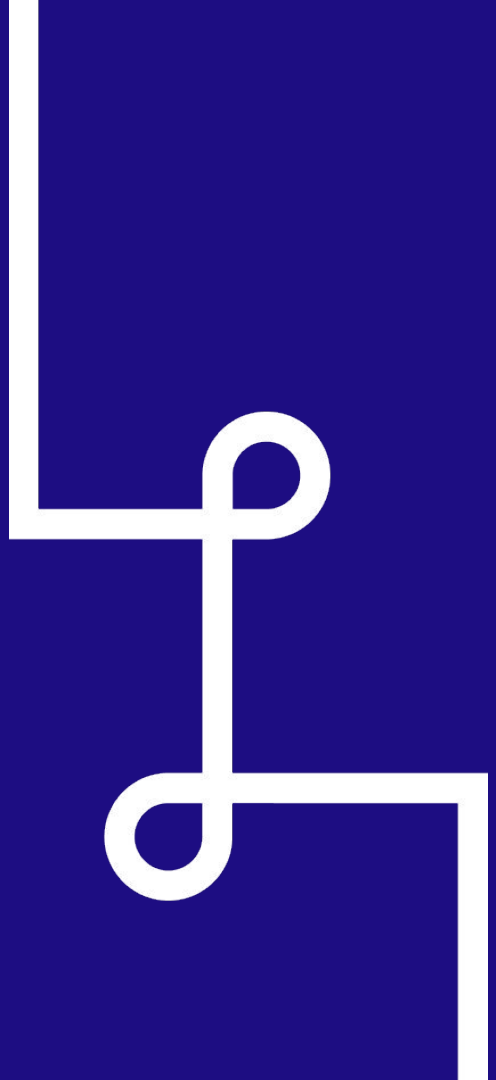
OpenTelemetry

Provides observability via easy-to-install server components like Grafana, Prometheus, Loki and Tempo

Gander

An open-source automated performance testing framework.

15
Command Line Interface



Drush

Or how I ignore the admin interface

- It automates administrative tasks
- Simplifies module management
  - Think `ddev drush en [module]`
- Enables quick database operations (`updb`) and config updates (`cex`/`cim`)

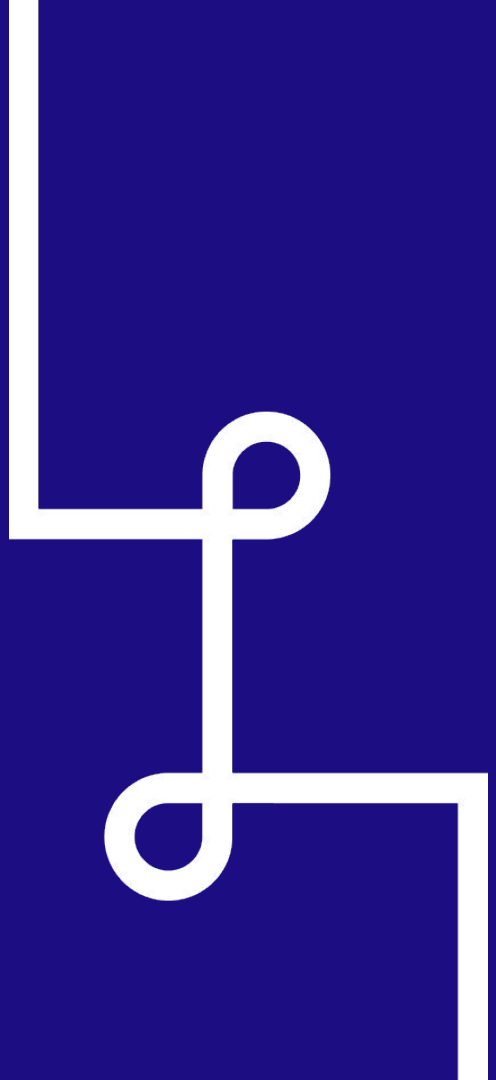
Exercise

Fetch weather data via CLI

- Run the generate-command to create the boilerplate code
  - `ddev drush gen drush:command-file`
  - Module machine name: pirate_weather
  - Inject the service that we created last time
- Open the generated code

16

DevOps



Steps of a CI/CD pipeline

- Configure your local setup
- Check code quality
- Active config splits
- Validate configuration
- Run automated tests
- Install dependencies
- Install Drupal
- Import configuration
- Compile FE code
- Build artifact

Steps of CI/CD pipeline

1. Local setup

Make sure that your local setup, whether it's ddev, lando or something else, matches your PRD environment as closely as possible.

2. Install dependencies

Keep using composer as package manager for PHP.
The built-in security features make it a valuable tool.

3. Check code quality

Linters for php, Javascript, CSS and YAML can, in an early stage, detect problems and flag code smells.

Steps of CI/CD pipeline

4. Install Drupal

Start from an empty Drupal installation for automated tests.

5. Activate config splits

Enable the correct dependencies (or keep some of them disabled) and provide config to overwrite.

6. Import configuration

You might want to run this twice as it is a crucial part of the pipeline.

Steps of CI/CD pipeline

7. Validate config

It can't hurt to run this additional step, just to be sure.

8. Compile FE code

Use npm to build the necessary front-end files, whether or not in a minimised version.

9. Run automated tests

Whatever framework you used to create your tests, now is the time to validate them.

Steps of CI/CD pipeline

10. Build artifact

Strip out everything that is not required on the production server (like node_modules or css source files) and package it.

See you next time!