



Drupal 8  
Développeurs

# Introduction à Drupal 8



# Introduction à Drupal 8

## Refonte Orienté Objet

Ce qu'il faut savoir :

- POO
- Composer
- Symfony
- Routage et contrôleurs
- Plugins
- Injection de dépendances et le conteneur de services



# Introduction à Drupal 8

- Drupal 8 est une solution "jeune"
- Tout a changé depuis Drupal 7
- Peu de retours d'expérience
- En phase de stabilisation sur les implémentations ou process



# Introduction à Drupal 8

## Les normes PHP

Viennent du PHP Framework Interoperability Group (FIG)

Différentes normes

- PSR-0 : autoloader standard
- PSR-1 : normes de codage de base (Drupal les suit presque)
- PSR-2 : normes de codage plus poussées (Drupal les suit presque)
- PSR-3 : interface du logger (pas implémentée dans Drupal)
- PSR-4 : autoloader amélioré (choisi par Drupal) : <https://www.drupal.org/node/2156625>



# Introduction à Drupal 8

## Composants Symfony

Le plus populaire des frameworks PHP aujourd'hui  
(<http://symfony.com/doc/current/index.html>)

Ensemble de composants réutilisables  
Certains sont réutilisés par Drupal

### Symfony2 Framework

#### Vital components

##### Libraries & Assets

3rd party libraries for Symfony.

WebProfilerBundle

FrameworkBundle

SwiftMailerBundle

Services configuration

##### Kernel

Provides Symfony's bundle functionality.

##### HttpKernel

Building blocks for creating any web application.

##### Routing

Maps URLs to code (previously done in hook\_menu)

##### EventDispatcher

Provides the functionality for event dispatching; this is used a lot by other components.

##### DependencyInjection

Manages the configuration of services and their dependencies.

##### Http foundation

HTTP Request & Response

#### Helper components

##### Browser Kit

##### Config

##### Form

.. other helper components ..

##### Debug

Nice debug messages.

##### Process

API for shell executions (like shell\_exec).

##### Serializer

XML/JSON serialization. Replaces drupal\_json\_decode.

##### Validator

Data validation framework and classes.

##### Translation

Contains translation services and translator interface, which is used by the validation component.

##### Yaml

YAML config file parsing.

#### Legend

Not in Drupal 8

In Drupal 8 (partially)

In Drupal 8

# Installation de l'environnement



# Les standards de codage

- Indentation, espaces : lisibilité du code
- Nommage, toujours commencer par le nom système : éviter les conflits
  - fonctions
  - constantes
  - variables persistantes
  - classes
  - fichiers
- Tags <?php non fermés : éviter l'envoi du buffer

[À connaître](#)





# Les modules Drupal utiles au développement

- *Devel* : debug et informations sur les données
- *Drupal Console* : générateur de code
- *Drush* : administration (DRUpal SHell) & téléchargement de modules et librairies
- *Coder* : revue de code
- *Masquerade* : changer d'utilisateur sans se déconnecter
- *Examples for developers* : démonstrations de l'utilisation de l'API
- Ne pas utiliser les caches durant le développement  
(<https://www.drupal.org/node/2598914> / <http://www.tothenew.com/blog/is-your-drupal-8-ready-for-writing-codes/>)

# Composer



- Gestionnaire de dépendances utilisé par la communauté PHP
- Installation uniquement locale au projet
- composer.json
- "composer install"
- composer.lock
- Contient un autoloader
- Pour les performances : <https://github.com/hirak/prestissimo>
- `curl -sS https://getcomposer.org/installer | php mv composer.phar /usr/local/bin/composer`
- [D8 Composer definitive introduction](#)
- [Composer install vs. composer update](#)



# Le serveur Web

xAMP (Apache, MySQL, PHP) conseillé

D'autres possibilités : Nginx / IIS, PostgreSQL

Liste des langages utilisés :

- SQL
- PHP
- Javascript
- HTML
- CSS



# L'éditeur de code

Le meilleur est celui que vous maîtrisez

Différence IDE/Editeur simple :

- Autocomplétion
- XDebug
- Refactoring
- Erreurs de syntaxe

Possibilité de télécharger des configurations

Exemples : PhpStorm / Eclipse / Netbeans / Atom / Vim



# TP : Drush

- Lancer Drush
- Regarder la liste des commandes
- Installer un module (features)
- Regarder à nouveau la liste des commandes
- Sauvegarder la base de données
- Installer les modules utiles au développement : devel, masquerade, examples
- Désactiver le module help



# TP : Console

- Installer la console Drupal([Documentation](#))
- Vérifier que c'est correctement installé
- Regarder la liste des commandes (drupal list)

```
devel
  devel:dumper (dd)                commands.devel.dumper.messages.change-devel-dumper-pi
develop
  develop:contribute               Download Drupal + Drupal Console to contribute.
dotenv
  dotenv:init                     Dotenv initializer.
entity
  entity:delete (ed)              Delete an specific entity
field
  field:info (fii)                View information about fields.
generate
  generate:ajax:command (gac)      Generate & Register a custom ajax command
  generate:authentication:provider (gap) Generate an Authentication Provider
  generate:breakpoint (gb)         Generate breakpoint
  generate:cache:context (gcc)     Generate a cache context
  generate:command (gco)           Generate commands for the console.
  generate:controller (gcon)       Generate & Register a controller
  generate:entity:bundle (geb)     Generate a new content type (node / entity bundle)
  generate:entity:config (gec)     Generate a new config entity
  generate:entity:content (geco)   Generate a new content entity
```



# Exemples de commandes souvent utilisées

- drupal site:mode dev
- drupal generate:module
- drupal generate:plugin:block
- drupal generate:routesubscriber
- drupal generate:form:config



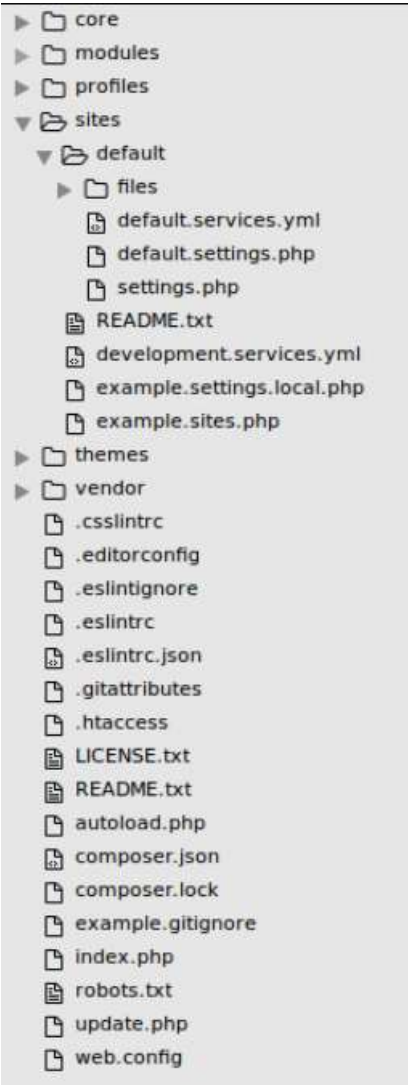
# Le meilleur moyen d'installer Drupal 8

- `composer create-project drupal-composer/drupal-project:8.x-dev some-dir --stability dev --no-interaction`
- `composer create-project drupal/drupal my_site_name` installe un nouveau site
- `composer require drupal/core ~8.5 --update-with-dependencies`` met à jour le cœur





# Organisations des répertoires



Cœur de Drupal  
Modules de tous les sites  
Profils d'installations  
Répertoire spécifique au site  
Répertoire d'upload par défaut  
Fichiers de configuration

Thèmes de tous les sites

Multisites

# Créer son premier module Drupal 8



- Fichiers .yaml
- Format de représentation de données par sérialisation
- Facilement modifiable et lisible
- Pas de tabulation (2 espaces)

```
key: 'value'  
tableau:  
  - valeur 1  
  - valeur 2
```



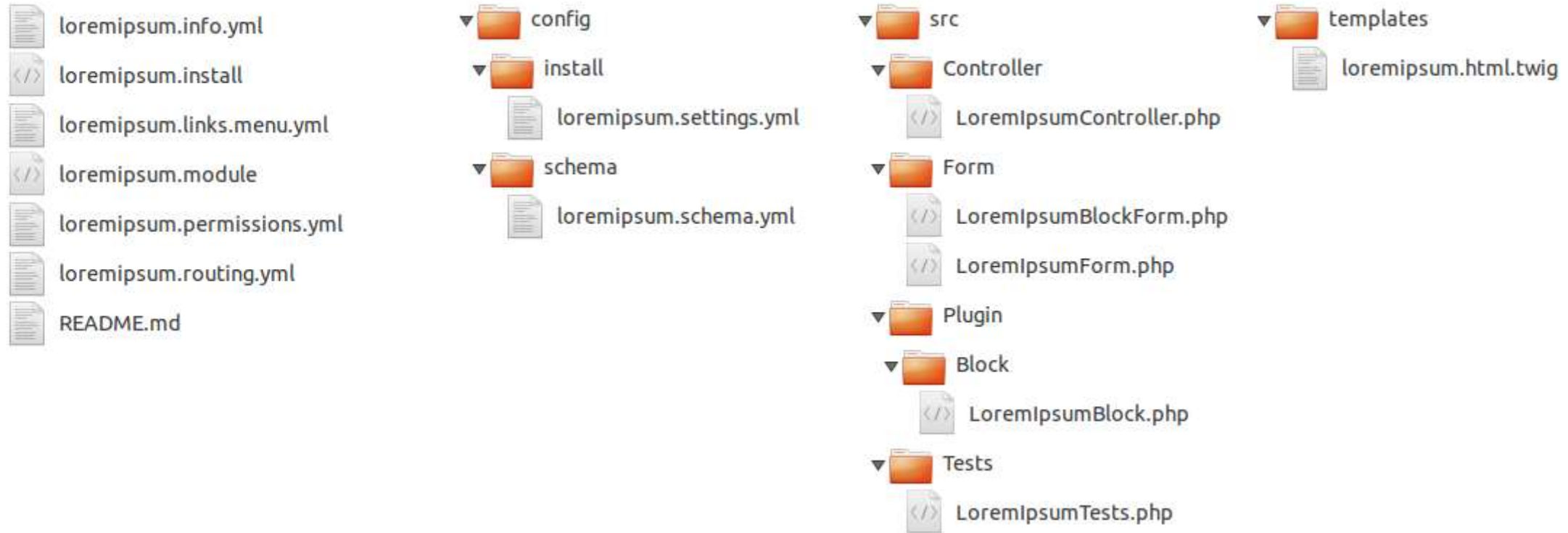
# Un module Drupal 8

- .info.yml (<https://www.drupal.org/node/2000204>)
  - name
  - core
  - type: module /!\
- .module (souvent vide en D8)
- .install facultatif (configuration désormais indépendante)
- répertoire "config/install" pour la configuration
- répertoire "src" pour les Plugins, Controller, Form, ...

```
name: Hello World
description: A Hello World example module
package: Hello World
type: module
core: 8.x
dependencies:
  - node
```



# Structure d'un module



<https://www.drupal.org/node/2560405>



# Les hooks

- Concept historique Drupal
- Implémenter `hook_form_alter()` donnera `mon_module_form_alter()`
- Poids des modules et altération
- Répondent à des déclencheurs
- Des hooks peuvent être déclarés par des modules contrib
- Rappel: on ne « hack » JAMAIS le core (sauf en cas de module bogué)
- Tend à disparaître avec Drupal 8 (Plugins, yml, events), mais existe encore...
- Les implémentations sont mise en cache
- Liste des hooks
  - <https://api.drupal.org/api/drupal/core%21core.api.php/group/hooks/8.5.x>



# Les hooks

## Exemple d'implémentation :

```
/**
 * Implements hook_form_alter().
 */
function mymodule_form_alter(&$form, \Drupal\Core\Form\FormStateInterface $form_state, $form_id) {
  // Custom code and comments go here ...
}
```

```
/**
 * Implements hook_form_alter().
 */
function hooks_example_form_alter(&$form, FormStateInterface $form_state, $form_id) {
  // This is an example of what is known as an alter hook. The $form parameter
  // in this case represents an already complete Form API array and our hook
  // implementation is being given the opportunity to make changes to the
  // existing data structure before it's used. Invoking and alter hooks is a
  // common pattern anytime lists or complex data structures are assembled.
  // hook_form_alter(), which allows you to manipulate any form, is one of the
  // most commonly implemented hooks.
  //
  // @see hook_form_alter()
  // @see hook_form_FORM_ID_alter()
  //
  // If this is the user login form, change the description text of the username
  // field.
  if ($form_id === 'user_login_form') {
    $form['name']['#description'] = t('This text has been altered by hooks_example_form_alter().');
  }
}
```



# Les permissions

## Page de permissions (/admin/people/permissions)

Permissions

List

Permissions

Roles

Home » Administration » People

Permissions let you control what users can do and see on your site. You can define a specific set of permissions for each role. (See the [Roles](#) page to create a role.) Any permissions granted to the Authenticated user role will be given to any user who is logged in to your site. From the [Account settings](#) page, you can make any role into an Administrator role for the site, meaning that role will be granted all new permissions automatically. You should be careful to ensure that only trusted users are given this access and level of control of your site.

Hide descriptions

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR
Block			
Administer blocks	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Comment			
Administer comment types and settings <small>Warning: Give to trusted roles only; this permission has security implications.</small>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Administer comments and comment settings	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Edit own comments	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Post comments	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Skip comment approval	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
View comments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Configuration Manager			
Export configuration <small>Warning: Give to trusted roles only; this permission has security implications.</small>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Import configuration	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>





# Les permissions

Fichier .permissions.yml

Statiques :

```
'delete contact entity':  
  title: Delete entity content  
  description: Allows users to delete contacts.  
'add contact entity':  
  title: Add entity content  
'view contact entity':  
  title: View entity content  
'edit contact entity':  
  title: Edit entity content  
  description: Allows users to edit contacts.  
'administer contact entity':  
  title: Administer settings  
  description: Allows users to alter the settings related to contacts.  
  restrict access: TRUE
```

Dynamique :

```
permission_callbacks:  
  - \Drupal\node\NodePermissions::nodeTypePermissions
```



# Système de routage / menus

Quelques définitions :

- routage : faire pointer une route (node/{node}) à une action (afficher un noeud)
- chemin (ou path) : route dont les arguments sont définis (ex: node/123 est un chemin, pointant vers la route node/{node} où l'argument est 123)
- lien de menu : Texte (ou titre) pointant vers un chemin
- alias : associe un chemin système (node/123) vers un chemin arbitraire renseigné par le contributeur (mon-noeud)

Les propriétés d'une route

- `_Permissions`
- Les arguments sont nommés ({node}) et peuvent être chargés dans le Controller (en les typant avec une classe)
- Vous pouvez passer des paramètres fixes au controller en les indiquant dans la route



# Les controllers

## Example (src/Controller/HelloController.php)

```
<?php

namespace Drupal\hello_world\Controller;

use Drupal\Core\Controller\ControllerBase;

class HelloController extends ControllerBase {

    /**
     * Display the markup.
     *
     * @return array
     */
    public function content() {
        return [
            '#type' => 'markup',
            '#markup' => $this->t('Hello, World!'),
        ];
    }
}
```



# Les menus

## Type d'élément de menu

- \*.routing.yml -> définit une URL
- \*.links.menu.yml -> lien de menu dans l'arborescence
- \*.links.task.yml -> onglet
- \*.links.action.yml -> "action" (back-office)

Paramètres de routage : <https://www.drupal.org/node/2092643>



# Exemples de routage

```
#.routing.yml
my_module.content:
  path: '/hello'
  defaults:
    _title: "HellWorld"
    _controller: "\Drupal\module\Controller\HelloController::content"
  requirements:
    _permission: 'access my module'

# , = ET
requirements:
  _permission: 'access my module,access content'

# + = OU
requirements:
  _permission: 'access my module+access content'

# Personnalisé
requirements:
  _custom_access: '\Drupal\my_module\MyClass::my_function'
```

La fonction renvoie alors `AccessResult::allowed()` ou `AccessResult::forbidden()`



# TP : le module Premium

- Créer le module Premium
- Créer deux permissions pour les rôles, une pouvant affecter le statut premium aux contenus et l'autre le voir
- Créer un rôle premium avec les permissions créées précédemment.
- Créer une page Suis-je Premium ? Qui affiche « Vous pouvez voir les contenus premium » ou « Vous ne pouvez pas voir les contenus premium »
  - url: suis-je-gold
- Créer une page *Est-il Premium ?* affichant la même chose, mais avec en argument l'uid de l'utilisateur
  - url d'exemple : est-il-gold/2
- Créer une page *Page Premium* avec du contenu "Lorem Ipsum" et ne s'affichant que si l'utilisateur courant a la permission de voir le contenu premium
  - url : page-gold

`\Drupal::currentUser()->hasPermission();` pour vérifier les permissions



# Gestion des nodes et des users

Quelques fonctions de l'API à connaître :

- `\Drupal::currentUser()` : utilisateur actuellement connecté
- `Node::load()` et `Node::loadMultiple()` pour charger des nœuds
- `User::load()` et `User::loadMultiple()` pour charger des utilisateurs
- `\Drupal::entityTypeManager()->getStorage('node')->loadMultiple($nids);`
- `$entity->save()` pour enregistrer un nœud, un utilisateur, ...
- `$user->getDisplayName()` pour afficher un nom d'utilisateur
- `Node::create(['type' => 'article']->save();`
- `$node->set('body' => ['value' => 'My body']); $node->save();`



# Quelques routes spéciales

- <front>
- <nolink>

Pour voir les routes, utilisez la console : `drupal router:debug`



# Menu





# Les liens

- Menu links
- Action links
- Local tasks
- Contextual links





# Menu links

## Fichier .links.menu.yml

```
entity.node_type.collection:  
  title: 'Content types'  
  parent: system.admin_structure  
  description: 'Create and manage fields, forms, and display settings for your  
content.'  
  route_name: entity.node_type.collection  
node.add_page:  
  title: 'Add content'  
  route_name: node.add_page
```



# Action links

## Fichier .links.action.yml

```
node.type_add:  
  route_name: node.type_add  
  title: 'Add content type'  
  appears_on:  
    - entity.node_type.collection  
node.add_page:  
  route_name: node.add_page  
  title: 'Add content'  
  appears_on:  
    - system.admin_content
```



# Local task links

Fichier .links.task.yml

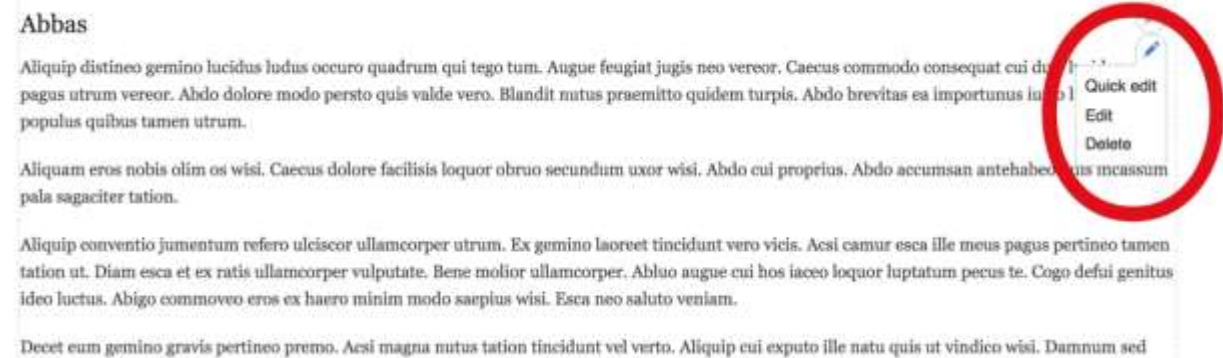
Rendu par défaut en onglets

```
entity.node.canonical:
  route_name: entity.node.canonical
  base_route: entity.node.canonical
  title: 'View'
entity.node.edit_form:
  route_name: entity.node.edit_form
  base_route: entity.node.canonical
  title: Edit
entity.node.delete_form:
  route_name: entity.node.delete_form
  base_route: entity.node.canonical
  title: Delete
  weight: 10
entity.node.version_history:
  route_name: entity.node.version_history
  base_route: entity.node.canonical
  title: 'Revisions'
  weight: 20
entity.node_type.edit_form:
  title: 'Edit'
  route_name: entity.node_type.edit_form
  base_route: entity.node_type.edit_form
entity.node_type.collection:
  title: List
  route_name: entity.node_type.collection
  base_route: entity.node_type.collection
```



```
entity.node.edit_form:
  route_name: entity.node.edit_form
  group: node
  title: Edit

entity.node.delete_form:
  route_name: entity.node.delete_form
  group: node
  title: Delete
  weight: 10
```



# Les Plugins





# Les plugins

Le principe de plugins dans Drupal est de permettre au système de fournir une fonctionnalité extensible et remplaçable de manière simple.

Deux concepts clés sont liés aux Plugins :

- Les Plugins
- Les types de Plugins (Plugins Type)

Les blocs sont par exemple des Plugins, chaque bloc en est un. Ils sont du Plugin Type Block.

Les Plugins sont utiles quand il est nécessaire de pouvoir facilement étendre une fonctionnalité générique mais que les implémentations possibles ne partagent que peu de code commun.





# Les types de plugins

Liste des types de plugin :  
drupal debug:plugin

Plugin type	Plugin manager class
action	Drupal\Core\Action\ActionManager
archiver	Drupal\Core\Archiver\ArchiverManager
block	Drupal\Core\Block\BlockManager
ckeditor.plugin	Drupal\ckeditor\CKEditorPluginManager
condition	Drupal\Core\Condition\ConditionManager
display_variant	Drupal\Core\Display\VariantManager
editor	Drupal\editor\Plugin\EditorManager
element_info	Drupal\Core\Render\ElementInfoManager
entity_reference_selection	Drupal\Core\Entity\EntityReferenceSelection\SelectionPluginManager
field.field_type	Drupal\Core\Field\FieldTypePluginManager
field.formatter	Drupal\Core\Field\FormatterPluginManager
field.widget	Drupal\Core\Field\WidgetPluginManager
filter	Drupal\filter\FilterPluginManager
help_section	Drupal\help\HelpSectionManager
image.effect	Drupal\image\ImageEffectManager
language_negotiation_method	Drupal\language\LanguageNegotiationMethodManager
link_relation_type	\Drupal\Core\Http\LinkRelationTypeManager
mail	Drupal\Core\Mail\MailManager



# Les annotations

Très utilisé dans le cœur : pour tous les plugins

<https://www.drupal.org/node/1882526>

<https://api.drupal.org/api/drupal/core%21core.api.php/group/annotation/8.1.x>

# Les blocs





# Les blocs

Fichier src/Pluglin/Block/HelloBlock.php  
Le nom de la classe et du fichier doivent être indentiques.

```
namespace Drupal\hello_world\Plugin\Block;

use Drupal\Core\Block\BlockBase;

/**
 * Provides a 'Hello' Block.
 *
 * @Block(
 *   id = "hello_block",
 *   admin_label = @Translation("Hello block"),
 *   category = @Translation("Hello World"),
 * )
 */
class HelloBlock extends BlockBase {

  /**
   * {@inheritdoc}
   */
  public function build() {
    return array(
      '#markup' => $this->t('Hello, World!'),
    );
  }

}
```



# Ajouter un formulaire au bloc

```
use Drupal\Core\Block\BlockPluginInterface;  
use Drupal\Core\Form\FormStateInterface;
```

```
extends BlockBase implements BlockPluginInterface {
```

```
/**  
 * {@inheritdoc}  
 */  
public function blockForm($form, FormStateInterface $form_state) {  
  $form = parent::blockForm($form, $form_state);  
  
  $config = $this->getConfiguration();  
  
  $form['hello_block_settings'] = array (  
    '#type' => 'textfield',  
    '#title' => $this->t('Who'),  
    '#description' => $this->t('Who do you want to say hello to?'),  
    '#default_value' => isset($config['hello_block_settings']) ? $config['hello_block_settings'] :  
    ..  
  );  
  
  return $form;  
}  
  
public function blockSubmit($form, FormStateInterface $form_state) {  
  parent::blockSubmit($form, $form_state);  
  $values = $form_state->getValues();  
  $this->configuration['hello_block_name'] = $values['hello_block_name'];  
}
```



# Ajouter un formulaire au bloc

```
/**
 * {@inheritdoc}
 */
public function build() {
    $config = $this->getConfiguration();

    if (!empty($config['hello_block_name'])) {
        $name = $config['hello_block_name'];
    }
    else {
        $name = $this->t('to no one');
    }
    return array(
        '#markup' => $this->t('Hello @name!', array(
            '@name' => $name,
        )),
    );
}
```



# Configuration par défaut

Configuration : fichier .yml

config/install/hello\_word.settings.yml

```
hello:  
  name: 'Hank Williams'
```

```
/**  
 * {@inheritdoc}  
 */  
public function defaultConfiguration() {  
  $default_config = \Drupal::config('hello_world.settings');  
  return [  
    'hello_block_name' => $default_config->get('hello.name'),  
  ];  
}
```

Valeur utilisée à l'installation du module (nécessite donc de désinstaller/installer le module après ajout si le module est déjà activé)



# TP : Créer un bloc

Créer un bloc :

- dont le titre côté administration est "Statut premium de l'utilisateur"
- dont l'id (nom machine) est premium\_status
- qui affiche "Vous pouvez voir les contenus premium" ou "Vous ne pouvez pas voir les contenus premium«
- qui affiche un second message configuration (exemple : « Bienvenue »)

Rappel: `\Drupal::currentUser()->hasPermission();` pour vérifier les permissions



# Theming





# Les Render Arrays

Les render arrays sont les blocs constituant une page Drupal. Ce sont des arrays PHP qui définissent des données (c-a-d la structure) ; On est obligés de produire des render arrays. Ceci afin qu'ils puissent être modifiés via les hooks d'altérations ou par la couche de theming.

Les propriétés sont toujours préfixées par un # et la propriété par défaut est #markup, elle permet d'indiquer du balisage simple. Un render array est converti en HTML avec la fonction render();

```
// Un render array simple
'ma_cle1' => array(
  '#markup' => "<h2>Du texte basique</h2>",
),

// Des propriétés utiles
'ma_cle2' => array(
  '#markup' => "Du texte basique",
  '#prefix' => '<h2>',
  '#suffix' => '</h2>',
),
```



# Paramètres du render array et propriétés

Une fonction de #theme peut être renseignée ainsi que ses paramètres (<https://www.drupal.org/developing/api/8/render/arrays>)

```
// Un render array qui produit un tableau HTML
'ma_cle1' => array(
  '#theme' => 'table',
  '#header' => $header,
  '#rows' => $rows,
  '#empty' => "Aucune donnée pour ce tableau",
),
```

Des propriétés utiles :

- #type: Le type d'élément
- #cache: contexts, tags, ... /!\
- #markup: Pour fournir directement de l'HTML
- #pre\_render / #post\_render: agit sur le tableau
- #prefix / #suffix, #weight, #attached, #access, ...



# TP : Manipuler les render arrays

Ajouter un `<h3>` autour d'un block précédent



# TP: Altérer les render arrays

Altérer le bloc dans un `hook_block_view_BASE_ID_alter()` afin de changer le `<h3>` en `<h4>`

-> Impossible à faire directement... Il faut ajouter un `#pre_render` dans le `hook_block_view_alter()`, et la fonction appelée dans le `pre_render` aura accès au `$build` du contenu du bloc.

Exemple : <http://www.drupal8.ovh/en/tutoriels/150/change-drupal-8-powered-by-block>

# Hook\_theme()



Le hook\_theme() définit des hooks/clés qui pourront ensuite être utilisés via la propriété #theme des render arrays. Ces theme hooks généreront ensuite le markup HTML via un template.

On peut fournir des variables à ces theme hooks. Des fonctions preprocess peuvent ajouter ou modifier des variables, et également ajouter des suggestions de templates.



# Déclaration et appel

```
function forum_theme() {  
    return array(  
        'forums' => array(  
            'template' => 'forums',  
            'variables' => array(  
                'forums' => NULL,  
                'topics' => NULL,  
            ),  
        ),  
    );  
}  
  
// Mais toujours privilégier les render arrays, car altérables  
$build['forums'] = array(  
    '#theme' => 'forums',  
    '#forums' => $forums,  
    '#topics' => $topics,  
);
```



Exemples d'utilisations de theme() :

- table
- item\_list
- pager
- links
- Image

[Liste complete des implementation de theme du cœur](#)

```
$image = array(  
  '#theme' => 'image',  
  '#path' => drupal_get_path('module', 'monmodule') . '/monimage.png',  
);
```

<https://www.drupal.org/developing/api/8/render/pipeline#html-main-content-renderer-pipeline>



Un template engine pour PHP  
Créé par Fabien Potencier (Symfony)

[Documentation](#)

Très sécurisé

Extensible



# Exemple de syntaxe

```
<!DOCTYPE html>
<html lang="{{ language.language }}" dir="{{ language.dir }}">
<head>
  <meta charset="utf-8">
  {{ head }}
  <title>{{ head_title }}</title>
  {{ styles }}
</head>
<body class="{{ classes }}" {{ attributes }}>
{{ page_top }}
{{ page }}
{{ scripts }}
{{ page_bottom }}
</body>
</html>
```



# 3 syntaxes à connaître

- `{{ afficher }}`
- `{# commenter #}`
- `{% "programmer" %}`

## TWIG VARIABLE RESOLUTION

---

```
<p>{{ user.name }}</p>
```

---

```
$user['name']
```

```
$user->name
```

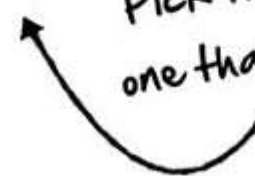
```
$user->name()
```

```
$user->getName()
```

```
$user->isName()
```

```
NULL
```

*Pick the first  
one that exists*



# Programmer



- `{% if expr %} {% else %} {% endif %}`
- `{% for item in items %} ( {% else %} ) {% endfor %}`



# Les filtres

- `{{ messages | join(', ') }}`
- `{{ "now" | date('d/m/Y H:i') }}`
- `{{ 'Home' | t }}` (ajouté par Drupal)
- `{{ content.field_date | format_date('short') }}` (ajouté par Drupal)
- Voir `core/lib/Drupal/Core/Template/TwigExtension.php` pour la liste des rajouts Drupal
- Sur toute une section : `{% filter upper %} Texte {% endfilter %}`
- `abs, batch, capitalize, convert_encoding, date, date_modify, default, escape, first, format, join, json_encode, keys, last, length, lower, merge, nl2br, number_format, raw, replace, reverse, round, slice, sort, split, striptags, title, trim, upper, url_encode`



# Les fonctions

`dump(node)`

`{{ max(1, 3, 2) }}`

`{{ random(['pomme', 'orange', 'citron']) }}`

+ celles [fournies par Drupal](#) :

- `link()`
- `path()`
- `url()`
- `attach_library()`



# Inclusion de template

- `{% include 'page.html.twig' with {'foo': 'bar', 'baz': 'bat'} %}`





# Héritage de template

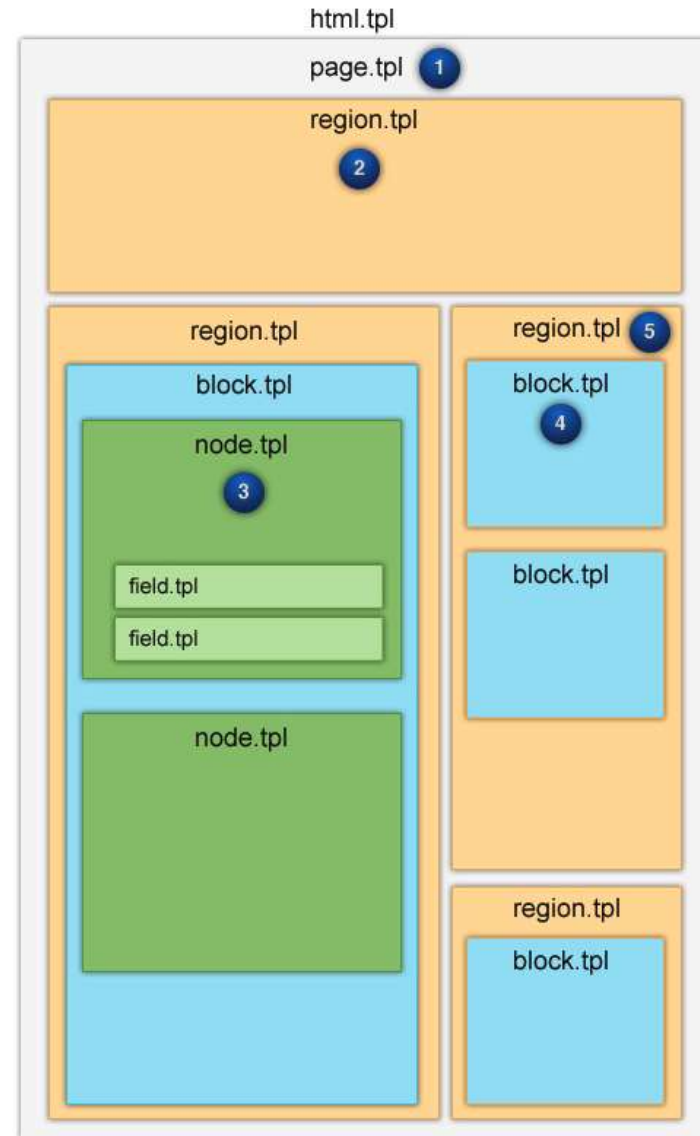
```
{# Template A #}
{% block foo %}
    <p>Mon premier paragraphe</p>
{% endblock %}

{# Template B #}
{% extends 'TemplateA' %}
{% block foo %}
    {{ parent() }}
    <p>Mon deuxième paragraphe</p>
{% endblock %}

{# Rendu Template B #}
<p>Mon premier paragraphe</p>
<p>Mon deuxième paragraphe</p>
```



# Hiérarchie des templates Drupal





# Le nommage des templates (surchage)

- <https://www.drupal.org/node/2354645>
- <https://www.drupal.org/docs/8/theming/twig/working-with-twig-templates>
- Pour Views : <http://redcrackle.com/blog/drupal-8/theme-views-templates>



# TP : Créer un template

Créer un template twig pour la page Suis-je Premium ? avec une couleur de texte différente en fonction du résultat