eBook Gratuit

APPRENEZ drupal

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#drupal

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: drupal

It is an unofficial and free drupal ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official drupal.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec drupal

Remarques

Drupal est un système de gestion de contenu open-source intégré à PHP. Drupal est conçu pour être flexible et puissant, permettant aux développeurs de créer une grande variété de sites, des blogs et des sites de type brochure aux plateformes de commerce électronique complexes. Grâce à son architecture modulaire axée sur la communauté, Drupal est en mesure de fournir des outils pour étendre les fonctions de base afin d'accélérer le développement de projets de grande envergure et complexes.

Actuellement, il existe deux versions de Drupal prises en charge: 7 et 8. Drupal 8 repose sur des composants du framework Symfony et de nombreuses autres bibliothèques tierces pour fournir des structures de développement modernes.

Examples

Installer Drupal avec Drush

```
drush dl drupal --drupal-project-rename=example
cd example
drush site-install standard --db-url='mysql://[db_user]:[db_pass]@localhost/[db_name]' --site-name=Example
```

Installation de Drupal 8 avec la console Drupal

Console Drupal

Le nouveau CLI pour Drupal. Un outil pour générer du code standard, interagir avec et déboguer Drupal.

Premièrement, nous devons installer la console Drupal.

La console Drupal est nécessaire non seulement pour cette période, mais aussi pour les futures installations.

```
# Run this in your terminal to get the latest project version:
curl https://drupalconsole.com/installer -L -o drupal.phar

# Or if you don't have curl:
php -r "readfile('https://drupalconsole.com/installer');" > drupal.phar

# Accessing from anywhere on your system:
mv drupal.phar /usr/local/bin/drupal

# Apply executable permissions on the downloaded file:
chmod +x /usr/local/bin/drupal
```

```
# Copy configuration files to user home directory:
drupal init --override

# Check and validate system requirements
drupal check
```

Vous pouvez appeler la `drupal list` pour voir toutes les commandes disponibles.

A l'étape suivante, nous téléchargerons le code source de Drupal

```
drupal site:new
```

La console vous invitera à choisir un dossier pour télécharger Drupal. Et à l'étape suivante, vous serez invité à choisir la version de Drupal à télécharger. Je recommande de sélectionner le dernier.

Ainsi, lorsque Drupal est téléchargé, vous devez l'installer.

```
drupal site:install
```

Après quelques étapes simples, votre site Drupal sera prêt.

Avec cette méthodologie, une nouvelle installation de Drupal nous prend entre 5 et 7 minutes à partir de la ligne de commande.

Concepts Drupal

Des versions

```
Release Date
```

Version	Date de sortie
8.2.4	07 décembre 2016
7,53	07 décembre 2016
6.38 (non pris en charge)	24 février 2016
5.23 (non pris en charge)	11 août 2010

Types d'entité

Dans les versions antérieures de Drupal, le système de champs était uniquement utilisé sur les types de contenu. Maintenant, grâce à l'API Entity, nous pouvons ajouter des champs à d'autres choses, comme des commentaires. Les entités sur le terrain rendent Drupal extrêmement flexible. Un type d'entité est une abstraction utile pour regrouper des champs. Vous trouverez ci-dessous les types d'entité dans le noyau Drupal:

- Nœuds (contenu)
- commentaires
- Des dossiers
- Termes taxonomiques
- Vocabulaires de taxonomie
- Utilisateurs

Vous pouvez également créer de nouveaux types d'entités dont les options ci-dessus ne répondent pas à vos besoins.

Liasses

Les bundles sont une implémentation d'un type d'entité auquel des champs peuvent être attachés. Vous pouvez considérer les ensembles comme des sous-types d'un type d'entité. Avec les noeuds de contenu (un type d'entité), par exemple, vous pouvez générer des ensembles (sous-types) tels que des articles, des articles de blog ou des produits. Cependant, tous les types d'entités n'ont pas de bundles. Par exemple, les utilisateurs ne disposent pas de lots séparés (sous-types). Pour les types d'entité qui autorisent les bundles, vous pouvez créer autant de bundles (sous-types) que vous le souhaitez. Ensuite, en utilisant le système de champ, vous pouvez ajouter différents champs à chaque ensemble. Les exemples incluent un champ de téléchargement de fichier sur Basic Pages et un champ de sous-titre sur les articles.

Des champs

Un champ est un élément de contenu réutilisable. En termes techniques, chaque champ est un type de données primitif, avec des validateurs personnalisés et des widgets pour l'édition et des formateurs pour l'affichage. Vous pouvez lire plus loin pour obtenir un guide du développeur sur l'utilisation de l' API Drupal 7 Fields .

Ce qui est important à savoir en ce qui concerne les entités, c'est que les champs peuvent être ajoutés à l'un des ensembles (ou types d'entités) pour aider à organiser leurs données.

Supposons, par exemple, que vous créez un type de contenu avec un champ de texte non structuré et que vous utilisiez le code HTML pour en structurer certaines parties, comme une section récapitulative ou des prix. Cela rendrait alors plus difficile le contrôle de la manière dont ces contenus étaient affichés ou des connexions entre différents types de contenus connexes.

C'est là que l'utilisation des champs est essentielle. Vous pouvez créer un champ récapitulatif de type Texte long ainsi que des champs de prix de type Décimal.

Entité

Une entité serait une instance d'un type d'entité particulier, tel qu'un commentaire, un terme de taxonomie ou un profil d'utilisateur, ou d'un ensemble tel qu'un article de blog, un article ou un produit.

Vous pouvez utiliser entity_load pour charger n'importe quelle entité. Notez, cependant, que le core ne fournit pas de fonction de sauvegarde ou de suppression, mais grâce au module d' API Entity, les pièces manquantes sont ajoutées (entity_create (), entity_save (), entity_delete (),

entity_view () et entity_access ()).

Mettre cela en termes de conception / programmation orientée objet ...

Si vous venez d'un contexte OOD / P et que vous essayez de mieux comprendre ce que sont ces concepts clés, le mappage suggéré suivant pourrait vous aider (bien que ce ne soit pas strictement puriste):

- Un ***type d'entité*** est une ***classe de base***
- Un ***bundle*** est une ***classe étendue***
- Un ***champ*** est un ***membre de classe*** , une ***propriété*** , une ***variable*** ou ***une instance de champ*** (selon votre préférence de dénomination)
- Une ***entité*** est un ***objet*** ou une ***instance*** d'une ***base*** ou d' ***une classe étendue***

Tous ces quatre concepts OOD / P sont spéciaux car ils sont sérialisables (stockés - par exemple dans une base de données ou un fichier). La sérialisation s'effectue via l'API Entity.

Lire Démarrer avec drupal en ligne: https://riptutorial.com/fr/drupal/topic/1135/demarrer-avec-drupal

Chapitre 2: API d'entité Drupal 8

Introduction

Le système d'entité de Drupal 8 permet aux développeurs de créer facilement des types de contenu personnalisés et des relations de données de modèles autour de ces types.

Examples

Créer une entité de contenu à l'aide de la console Drupal

La console Drupal apporte un échafaudage à l'écosystème Drupal et facilite la création d'une entité de contenu.

Dans la plupart des cas, vous trouverez plus facile de travailler avec une entité personnalisée dans un module personnalisé.

Etape 1: Générer un module

```
vendor/bin/drupal generate:module
```

Suivez les instructions et créez votre module personnalisé.

Étape 2: générer une entité de contenu

```
vendor/bin/drupal generate:entity:content
```

Suivez les invites sur votre ligne de commande en veillant à sélectionner le module personnalisé créé à l'étape précédente.

Lire API d'entité Drupal 8 en ligne: https://riptutorial.com/fr/drupal/topic/10681/api-d-entite-drupal-8

Chapitre 3: Brindille

Introduction

Twig est le moteur de template qui fait partie de Drupal 8. Dans Drupal 8, les fichiers Twig ont l'extension `.html.twig` et sont utilisés dans tous les aspects du thème Drupal. Les entités, champs, vues peuvent tous être rendus en utilisant les fichiers `.html.twig` .

Dans ce sujet, le but est d'avoir un livre de recettes sur la façon de travailler avec Twig dans le contexte de Drupal. Si vous souhaitez en savoir plus sur la syntaxe ou les fonctions disponibles, consultez la documentation .

Examples

Filtre brindille

Contrairement à Drupal 7, vous ne pouvez pas appeler des fonctions PHP régulières dans vos modèles. Dans Drupal 8, la solution consiste à créer des filtres et des fonctions.

Vous devez utiliser un **filtre** lorsque: vous souhaitez transformer les données que vous souhaitez afficher. Imaginez que vous ayez un titre que vous voulez toujours mettre en majuscule. Par exemple, twig a le filtre en `capitalize` par défaut qui vous permet de transformer n'importe quel texte en son équivalent en majuscule.

Pour cet exemple, nous allons créer un filtre qui nous permettra de mélanger une chaîne. La façon de créer des filtres et des fonctions est exactement la même que pour Twig classique .

La principale différence entre Twig standard et Drupal 8 Twig est que dans Drupal 8, vous devez créer une définition de service de la classe que vous créez et que la classe doit également appartenir à un espace de nommage, sinon elle ne sera pas enregistrée dans l'environnement Drupal .

Cet exemple suppose que vous avez un module appelé `twig_shuffle_extension` .

Ce sera la définition de service de base inn `twig_shuffle_extension.services.yml`

```
services:
  twig_shuffle_extension.twig_extension:
    class: Drupal\twig_shuffle_extension\TwigExtension\TwigShuffleExtension
    tags:
      - { name: twig.extension }
```

La clé de `tags` est également indispensable et indique à Drupal ce que cette classe est censée faire (c'est-à-dire l'enregistrer en tant qu'extension Twig).

Et maintenant, le code source qui doit être placé dans le chemin défini dans la clé de `class` de la définition du service.

```
// Don't forget the namespace!
namespace Drupal\twig_shuffle_extension\TwigExtension;

use Twig_Extension;
use Twig_SimpleFilter;

class TwigShuffleExtension extends Twig_Extension  {
  /**
   * This is the same name we used on the services.yml file
   */
  public function getName() {
    return 'twig_shuffle_extension.twig_extension';
  }

  // Basic definition of the filter. You can have multiple filters of course.
  // Just make sure to create a more generic class name ;)
  public function getFilters() {
    return [
      new Twig_SimpleFilter('shuffle', [$this, 'shuffleFilter']),
    ];
  }

  // The actual implementation of the filter.
  public function shuffleFilter($context) {
    if(is_string($context)) {
      $context = str_shuffle($context);
    }
    return $context;
  }
}
```

Videz vos caches et maintenant, si tout se passe comme prévu, vous pouvez utiliser le filtre dans vos modèles.

```
{{ "shuffle me!" | shuffle }}
```

Injection de dépendance dans les extensions de brindilles

Cet exemple vous montre comment utiliser Dependency Inject pour utiliser d'autres services enregistrés dans l'environnement Drupal.

Imaginez que vous ayez un fichier d'image SVG qui change de couleur en fonction d'un objet CSS / Javascript aléatoire dans votre projet. Pour pouvoir cibler le SVG avec CSS, vous devez avoir le fichier SVG dans le DOM. Donc, vous créez un fichier SVG de base sans aucune couleur et placez-le dans votre dossier de thème.

Bien sûr, vous pourriez simplement coller le contenu du fichier dans le modèle Twig, mais ce ne serait pas bien. Vous pouvez créer une extension Twig mais vous ne voulez pas non plus coder en dur votre chemin de thème dans le code source de l'extension.

Cela signifie que nous devons obtenir le chemin dynamiquement. Vous avez deux options:

1. Utilisez l'équivalent d'une variable globale en appelant `\Drupal::theme()->getActiveTheme()->getPath();`
2. Injecter le `ThemeManager` (donné par `\Drupal::theme()`) dans votre classe d'extension

Dans cet exemple, nous prendrons le deuxième exemple car il peut être largement applicable à tout service (vous importez la demande ou la connexion à la base de données si vous le souhaitez).

Cela suppose que vous avez un module appelé `twig_svg_extension` et un fichier `twig_svg_extension.services.yml` :

```
services:
  twig_svg_extension.twig_extension:
    class: Drupal\twig_svg_extension\TwigExtension\TwigSvgExtension
    arguments: ['@theme.manager']
    tags:
      - { name: twig.extension }
```

S'il vous plaît pas la clé des `arguments` qui dit à Drupal le service à injecter.

```
namespace Drupal\twig_svg_Extension\TwigExtension;

use Drupal\Core\Theme\ThemeManager;
use Twig_Extension;
use Twig_SimpleFilter;

class TwigSvgExtension extends Twig_Extension  {
  private $theme;

  // Dependency injection at work!
  public function __construct(ThemeManager $theme) {
    $this->theme = $theme;
  }

  public function getFilters() {
    return [
      'svg' =>new Twig_SimpleFilter('svg', [$this, 'svgFilter']),
    ];
  }

  public function getName() {
    return 'twig_svg_extension.twig_extension';
  }

  public function svgFilter(string $filepath) {
    $realpath = realpath($this->theme->getActiveTheme()-
>getPath().DIRECTORY_SEPARATOR.$filepath);
    $pathinfo = pathinfo($realpath);

    if($realpath !== false && strtolower($pathinfo['extension']) === 'svg') {
      return file_get_contents($realpath);
    }

    return '"'.$filepath.'" does not exist or is not an SVG';
  }
}
```

Notez le constructeur qui contient la dépendance que nous avons injectée dans la configuration du service, ainsi que le `svgFilter` qui obtient le chemin du thème actif actuel.

`$filepath` devrait être un chemin relatif vers votre dossier de thèmes. L'extension convertira le

chemin du fichier dans le contenu du fichier vers lequel il pointe.

Lire Brindille en ligne: https://riptutorial.com/fr/drupal/topic/8804/brindille

Chapitre 4: Cache Drupal et performace

Introduction

Le cache a été utilisé pour le site ou le système pour améliorer la diffusion du contenu rapidement pour les utilisateurs finaux. Cette rubrique est créée pour explorer le mécanisme de mise en cache intégré de Drupal et fournir des informations sur son utilisation. Nous devons explorer la fonctionnalité de mise en cache intégrée de Drupal avec les modules externes comme Varnish, Memcache, Authcache, File cache, etc., disponibles pour améliorer les performances du site. Vous pouvez trouver l'exemple et les options de mise en cache les mieux adaptés à votre site dans cette rubrique.

Examples

Activer le site Drupal et bloquer le cache

Drupal lui-même fournit de bonnes options de mise en cache pour augmenter la vitesse de la page et servir rapidement les pages aux utilisateurs finaux. Les caches sont utilisés pour améliorer les performances de votre site Drupal. Mais cela a aussi un inconvénient, à savoir que parfois, cela pouvait conduire à des données "obsolètes". Cela signifie que, parfois, le système peut commencer à servir les anciennes pages du cache.

Comment activer le site Drupal et bloquer le cache sur différentes versions de Drupal? Voir ci-dessous pour les réponses: Drupal 6:

1. Accédez à Administrer -> Configuration du site -> Performances.
2. Activer les options de cache
3. Activer l'agrégation des fichiers JS / CSS et l'enregistrer.

Drupal 7:

1. Allez dans Administer -> Config -> Development -> Performance.
2. Activer les options de cache
3. Activer l'agrégation des fichiers JS / CSS et l'enregistrer.

Lire Cache Drupal et performace en ligne: https://riptutorial.com/fr/drupal/topic/10082/cache-drupal-et-performace

Chapitre 5: Développement de modules - Drupal 7

Remarques

Des exemples de modules pour développeurs doivent être utilisés comme référence pour le développement de modules, idéalement. Il a une explication de toutes les principales API, une utilisation bien documentée. Les débutants doivent comprendre le développement de modules.

Examples

Module de base fournissant une page simple

really_neat.info

```
name = Really Neat Module
description = Provides a really neat page for your site
core = 7.x
```

really_neat.module

```
<?php

/**
 * @file
 * Hook implementation and shared functions for the Really Neat Module.
 */

/**
 * Implements hook_menu().
 */
function really_neat_menu() {
  $items = array();

  $items ['really/neat'] = array(
      'title' => 'A Really Neat Page',
      'page_callback' => 'really_neat_page',
      'access_callback' => TRUE, //Anyone can access.
      // Or replace with array([name-of-permission]),
    ),

  return $items;
}

/**
 * Page callback: Displays something really neat
 */
function really_neat_page() {
  return "Really Neat!"
}
```

Module de base fournissant un bloc personnalisé

custom_module.info

```
name = Custom Module
description = Creates a block containing a custom output.
core = 7.x
```

custom_module.module

```
/**
 * Initiates hook_block_info.
 *
 * Registers the block with Drupal.
 */
function custom_module_block_info() {
  $blocks = array();
    //Registers the machine name of the block.
  $blocks['custom_block'] = array(
      //Sets the human readable, administration name.
    'info' => t('My Custom Block'),
      //Tells Drupal not to cache this block.
      //Used if there is dynamic content.
    'cache' => DRUPAL_NO_CACHE,
  );
  return $blocks;
}

/**
 * Initiates hook_block_view().
 *
 * Sets the block title and content callback.
 */
function custom_module_block_view($delta = '') {
  $block = array();

  switch ($delta) {
      //Must be the machine name defined in the hook_block_info.
    case 'custom_block':
        //The blocks title.
      $block['subject'] = 'My custom block';
        //The string or function that will provide the content of the block.
      $block['content'] = custom_module_block_content();
      break;
  }

  return $block;
}

/**
 * Returns the content of the custom block.
 */
function custom_module_block_content() {
  $content = "This function only returns a string, but could do anything."

  return $content;
}
```

Formulaire personnalisé de base à inclure dans les exemples de page ou de bloc.

Fonctions simples de formulaire, de validation et de soumission pour créer une fonctionnalité de "liste de diffusion". Cela peut ensuite être appliqué à la page de base ou à des exemples de blocs de base.

Suppose que vous avez créé une table dans la base de données drupal appelée «mailing_list» avec les champs prénom, nom et adresse e-mail.

Informations supplémentaires sur l'API de formulaire et les options de champ supplémentaires: https://api.drupal.org/api/drupal/developer!topics!forms_api_reference.html/7.x/

```
function custom_module_form($form, &$form_state) {
  $form['first_name'] = array (
    '#type' => 'textfield',
    '#title' => 'First Name',
    '#required' => TRUE,
  );
  $form['last_name'] = array (
    '#type' => 'textfield',
    '#title' => 'Last Name',
    '#required' => TRUE,
  );
  $form['email'] = array (
    '#type' => 'textfield',
    '#title' => 'First Name',
    '#required' => TRUE,
  );

  return $form;
}

function custom_module_form_validate($form, &$form_state) {
  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    form_set_error('email', t('Please provide a valid email address.'));
  }
}

function custom_module_form_submit($form, &$form_state) {
  //Useful function for just getting the submitted form values
  form_state_values_clean($form_state);

  //Save time later by assigning the form values to variables.
  $first_name = $form_state['values']['first_name'];
  $last_name = $form_state['values']['last_name'];
  $email = $form_state['values']['email'];

  //Insert the submitted data to the mailing_list database table.
  db_insert('mailing_list')
    ->fields(array(
      'first name' => $first_name,
      'last name' => $last_name,
      'email' => $email,
    ))
    ->execute();
  //Set a thank you message.
```

```
  drupal_set_message('Thank you for subscribing to our mailing list!');

  //drupal_goto() could be used here to redirect to another page or omitted to reload the same
page.
  //If used, drupal_goto() must come AFTER drupal_set_message() for the message to be
displayed on the new page.
}
```

Module de base fournissant un bloc personnalisé

custom_module.info

```
name = Custom Module
description = Creates a block containing a custom output.
core = 7.x
```

custom_module.module

```
/**
 * Initiates hook_block_info.
 *
 * Registers the block with Drupal.
 */
function custom_module_block_info() {
  $blocks = array();
    //Registers the machine name of the block.
  $blocks['custom_block'] = array(
      //Sets the human readable, administration name.
    'info' => t('Titania Price Widget'),
      //Tells Drupal not to cache this block.
      //Used if there is dynamic content.
    'cache' => DRUPAL_NO_CACHE,
  );
  return $blocks;
}

/**
 * Initiates hook_block_view().
 *
 * Sets the block title and content callback.
 */
function custom_module_block_view($delta = '') {
  $block = array();

  switch ($delta) {
      //Must be the machine name defined in the hook_block_info.
    case 'custom_block':
        //The blocks title.
      $block['subject'] = 'My custom block';
        //The string or function that will provide the content of the block.
      $block['content'] = custom_module_block_content();
      break;
  }

  return $block;
}

/**
```

```
 * Returns the content of the custom block.
 */
function custom_module_block_content() {
  $content = "This function only returns a string, but could do anything."

  return $content;
}
```

Formulaire personnalisé de base à inclure dans les exemples de page ou de bloc.

Fonctions simples de formulaire, de validation et de soumission pour créer une fonctionnalité de "liste de diffusion". Cela peut ensuite être appliqué à la page de base ou à des exemples de blocs de base.

Suppose que vous avez créé une table dans la base de données drupal appelée «mailing_list» avec les champs prénom, nom et adresse e-mail.

Informations supplémentaires sur l'API de formulaire et les options de champ supplémentaires: https://api.drupal.org/api/drupal/developer!topics!forms_api_reference.html/7.x/

```
function custom_module_form($form, &$form_state) {
  $form['first_name'] = array (
    '#type' => 'textfield',
    '#title' => 'First Name',
    '#required' => TRUE,
  );
  $form['last_name'] = array (
    '#type' => 'textfield',
    '#title' => 'Last Name',
    '#required' => TRUE,
  );
  $form['email'] = array (
    '#type' => 'textfield',
    '#title' => 'First Name',
    '#required' => TRUE,
  );

  return $form;
}

function custom_module_form_validate($form, &$form_state) {
  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    form_set_error('email', t('Please provide a valid email address.'));
  }
}

function custom_module_form_submit($form, &$form_state) {
  //Useful function for just getting the submitted form values
  form_state_values_clean($form_state);

  //Save time later by assigning the form values to variables.
  $first_name = $form_state['values']['first_name'];
  $last_name = $form_state['values']['last_name'];
  $email = $form_state['values']['email'];

  //Insert the submitted data to the mailing_list database table.
```

```
  db_insert('mailing_list')
    ->fields(array(
      'first name' => $first_name,
      'last name' => $last_name,
      'email' => $email,
    ))
    ->execute();
  //Set a thank you message.
  drupal_set_message('Thank you for subscribing to our mailing list!');

  //drupal_goto() could be used here to redirect to another page or omitted to reload the same
page.
  //If used, drupal_goto() must come AFTER drupal_set_message() for the message to be
displayed on the new page.
}
```

Exemple de fichier custom_module.install pour créer une table de base de données

Peut être utilisé conjointement avec l' ***exemple de formulaire personnalisé*** *pour créer une table dans la base de données drupal pour une fonctionnalité de liste de diffusion.*

Cet exemple a été créé en créant la table directement dans ma base de données de développement, puis a créé les données pour hook_schema () à l'aide du module Schema .

Cela permet la création automatique de tableaux lors de l'installation de modules sur les sites de transfert et de production.

custom_module.install

```
/**
 * Installs the database schema.
 */
function custom_module_install() {
  drupal_install_schema('mailing_list');
}

/**
 * Uninstalls the database schema.
 */
function custom_module_uninstall() {
  drupal_uninstall_schema('mailing_list');
}

/**
* Creates the tables using the schema API.
*/
function custom_module_schema() {
  $schema['mailing_list'] = array(
    'description' => 'TODO: please describe this table!',
    'fields' => array(
      'first name' => array(
        'description' => 'TODO: please describe this field!',
        'type' => 'int',
        'not null' => TRUE,
      ),
      'last name' => array(
```

```
        'description' => 'TODO: please describe this field!',
        'type' => 'int',
        'not null' => TRUE,
      ),
      'email' => array(
        'description' => 'TODO: please describe this field!',
        'type' => 'int',
        'not null' => TRUE,
      ),
    ),
  );
}
```

Lire Développement de modules - Drupal 7 en ligne:
https://riptutorial.com/fr/drupal/topic/2456/developpement-de-modules---drupal-7

Chapitre 6: Développement du thème - Drupal 7

Examples

Écrire des fichiers de thème .info

Le fichier .info est un fichier texte statique permettant de définir et de configurer un thème. Chaque ligne du fichier .info est une paire clé-valeur avec la clé à gauche et la valeur à droite, avec un "signe égal" entre elles (par exemple, *name = my_theme*).

Les points-virgules sont utilisés pour commenter une ligne. Certaines clés utilisent une syntaxe spéciale avec des crochets pour créer une liste de valeurs associées, appelée "tableau". Si vous n'êtes pas familier avec les tableaux, consultez les fichiers .info par défaut fournis avec Drupal et lisez les explications des exemples qui suivent. Même si l'extension de fichier .info n'est pas ouverte en mode natif par une application, vous pouvez utiliser TextEdit sur un Mac ou un bloc-notes sur un ordinateur Windows pour afficher, modifier et enregistrer vos modifications.

Exigences relatives au nom du thème

Le nom doit commencer par un caractère alphabétique, peut contenir des chiffres et des traits de soulignement, mais pas de traits d'union, d'espaces ou de ponctuation. Le nom sera utilisé par Drupal pour former diverses fonctions en PHP et aura donc les mêmes limitations.

Ne choisissez pas les noms déjà utilisés par les modules installés , car tous les composants installés doivent avoir des noms uniques.

L'une des meilleures pratiques consiste à utiliser des préfixes pour nommer le thème personnalisé d'un site, afin de garantir des noms uniques pour les thèmes. Un site nommé example.com peut utiliser des noms de thèmes tels que ex_themename.

Étant donné que le fichier .info est mis en cache, vous devez effacer le cache avant que toute modification soit affichée sur votre site.

Le fichier .info peut également spécifier les paramètres de thème accessibles depuis l'interface d'administration de Drupal, comme vous le verrez bientôt.

Codage

Le fichier doit être enregistré au format UTF-8 sans marque BOM (Byte Order Mark).

Contenu

Drupal comprend les clés listées ci-dessous. Drupal utilisera les valeurs par défaut pour les clés facultatives non présentes dans le fichier .info. Voir les exemples pour les thèmes principaux.

- nom ***requis***
- description ***recommandée***
- capture d'écran
- version ***découragée***
- noyau ***nécessaire***
- moteur
- thème de base
- les régions
- fonctionnalités
- réglage des thèmes
- feuilles de style
- des scripts
- php

Thème Fichier .info

```
name = MyCompany Theme
description = A Bootstrap Sub-theme.
core = 7.x
base theme = bootstrap

;;;;;;;;;;;;;;;;;;;;
;; Regions
;;;;;;;;;;;;;;;;;;;;

regions[navigation]     = 'Navigation'
regions[header]         = 'Top Bar'
regions[highlighted]    = 'Highlighted'
regions[help]           = 'Help'
regions[content]        = 'Content'
regions[sidebar_first]  = 'Primary'
regions[sidebar_second] = 'Secondary'
regions[footer]         = 'Footer'
regions[page_top]       = 'Page top'
regions[page_bottom]    = 'Page bottom'

;;;;;;;;;;;;;;;;;;;;
;; MyCompany Custom Regions
;;;;;;;;;;;;;;;;;;;;
regions[footer_menu_left] = 'Footer menu left'
regions[footer_menu_right] = 'Footer menu right'

;;;;;;;;;;;;;;;;;;;;
;; CSS
;; these css files will be included on every page
;;;;;;;;;;;;;;;;;;;;

stylesheets[all][] = css/bootstrap.min.css
stylesheets[all][] = css/MyCompany.css

;;;;;;;;;;;;;;;;;;;;
;; JS
;; this JS file will be included on every page
;;;;;;;;;;;;;;;;;;;;

scripts[] = js/MyCompany.min.js
```

Lire Développement du thème - Drupal 7 en ligne:
https://riptutorial.com/fr/drupal/topic/2715/developpement-du-theme---drupal-7

Chapitre 7: Drush

Remarques

Qu'est-ce que Drush?

Drush est une interface de script de ligne de commande pour les sites Drupal. Il permet la gestion en ligne de commande des sites Drupal.

Examples

Commandes Drush

Statut Drush

```
drush status
```

Cela vous donnera un aperçu de votre site Drupal. Version, URI, emplacement de la base de données, chemins d'accès aux fichiers, thème par défaut, etc. Si vous utilisez cette commande et que vous ne voyez pas cette information, cela signifie que vous vous trouvez dans un mauvais dossier

---

Réinitialisation du mot de passe pour tout utilisateur

```
drush upwd admin --password="newpassword"
```

Où "admin" est un nom d'utilisateur existant et "newpassword" est le mot de passe souhaité.

Générer une URL de connexion admin à usage unique

```
drush uli
```

Génère une URL qui peut être utilisée pour se connecter à la section admin. L'URL a un jeton à usage unique. Le résultat devrait ressembler à ceci:

```
http://example.com/user/reset/1/1469178712/MEn1QOXo3YGKAUHCknFQF0rEPJ_itkS-a6I8LJwaNYs/login
```

Parfois, le nom d'hôte ou l'adresse IP ne peut pas être résolu, et le résultat est un avertissement comme celui-ci:

```
default does not appear to be a resolvable hostname or IP, not starting browser.   [warning]
You may need to use the --uri option in your command or site alias to indicate
the correct URL of this site.
http://default/user/reset/1/1469178629/-zFS_0u8is2N2uCKuLUdGBpJ3cZzV9am5_irsbtVAOs/login
```

La solution utilise le paramètre "--url" comme l'exemple suivant:

```
drush uli --uri="http://example.com/"
```

Effacer les caches

```
drush cache-rebuild
```

Reconstruit le cache pour Drupal 8. Pour drupal 7, vous pouvez utiliser

```
drush cache-clear
```

Ces commandes sont également disponibles plus courtes avec

```
drush cr
```

ou

```
drush cc // optionally pass all to clear all the caches
```

Activer les modules

```
drush pm-enable mymodule
```

Activer 'mymodule' comme activer un module dans l'interface d'administration

Ces commandes sont également disponibles plus courtes avec

```
drush en mymodule // optionally pass the -y option to avoid the interactive question
```

Mainteneace

Pour activer le mode maintenance à l'aide de Drush, vous pouvez utiliser cette commande:

```
drush vset maintenance_mode 1 // pass 0 to disable the maintenance
```

N'oubliez pas d'effacer les caches après avoir activé / désactivé le mode de maintenance.

Configuration de l'exportation

Dans Drupal 7 et versions ultérieures, votre configuration est probablement stockée à l'aide du module Fonctionnalités . Pour mettre à jour une fonctionnalité avec des modifications depuis les bases de données, utilisez cette commande:

```
drush features-update [feature-name] // e.g. drush features-update content_type_news
```

Vous pouvez également utiliser ce raccourci:

```
drush fu [feature-name]
```

Drupal 8 en utilisant "Configuration Management". Pour exporter votre configuration avec drush, utilisez cette commande

```
drush config-export // optionally add -y to not have to verify it
```

Vous pouvez également utiliser la commande abrégée

```
drush cex -y
```

Installer Drush

Installation globale manuelle

Pour OS X et Linux:

1. Apportez Terminal, Bash ou votre shell normal.
2. Tapez ce qui suit dans Terminal / Bash:

```
# Download latest stable release using the code below or browse to github.com/drush-
ops/drush/releases.
php -r "readfile('http://files.drush.org/drush.phar');" > drush
# Or use our upcoming release: php -r "readfile('http://files.drush.org/drush-
unstable.phar');" > drush

# Test your install.
php drush core-status

# Make `drush` executable as a command from anywhere. Destination can be anywhere on
$PATH.
chmod +x drush
sudo mv drush /usr/local/bin

# Optional. Enrich the bash startup file with completion and aliases.
drush init
```

Pour les fenêtres:

1. Téléchargez Drush depuis GitHub.
2. Extrayez le fichier compressé dans le lecteur de votre choix, par exemple.

```
C:\
```

3. Installez Drush, définissez le chemin du dossier extrait dans la variable du chemin d'environnement qui devrait également inclure `Apache, PHP and MySQL` .

```
C:\xampp\apache\bin;C:\xampp\mysql\bin;C:\xampp\php;C:\drush;
```

4. Vérifiez que Drush fonctionne:

```
drush status
```

Installation globale du compositeur

1. Installez Composer globalement .
2. Ajoutez le répertoire `bin` de Composer au chemin du système en plaçant `export PATH="$HOME/.composer/vendor/bin:$PATH"` dans votre ~ / .bash_profile (OS X) ou ~ / .bashrc (Linux).
3. Installez Drush:

```
composer global require drush/drush
```

4. Vérifiez que Drush fonctionne:

```
drush status
```

Plus de détails sur Drush Docs

Installer Drush

Installation manuelle

Tapez ci-dessous les commandes dans Terminal.

```
php -r "readfile('http://files.drush.org/drush.phar');" > drush
chmod +x drush
sudo mv drush /usr/local/bin
drush init # Add alias in bash startup file.
```

Compositeur

En supposant que compositeur est installé.

```
composer global require drush/drush:dev-master
```

Lire Drush en ligne: https://riptutorial.com/fr/drupal/topic/2062/drush

Chapitre 8: Exemple pour l'API Drupal 8 Queue et l'API Batch

Examples

Un exemple de module pour vous aider à comprendre l'API de file d'attente et l'API Batch dans Drupal 8

xml_import_example.info.yml

```
type: module
name: XML import example
package: Examples
description: "This module helps understanding the Batch API and Queue API with an XML import
example"
core: 8.x
```

xml_import_example.permissions.yml

```
import content from xml:
  title: 'Import content from xml'
  description: 'With this permission user can import contents from a XML source'
  restrict access: TRUE
```

xml_import_example.routing.yml

```
# Get contents from the xml source
xml_import_example.get_contents_from_xml:
  path: '/get-contents-from-xml'
  defaults: { _controller:
'\Drupal\xml_import_example\Controller\ImportContentFromXML::getContentsFromXMLPage' }
  requirements:
    _permission: 'import content from xml'
# Process all queue items with batch
xml_import_example.process_all_queue_items_with_batch:
  path: '/process-all-queue-items'
  defaults: { _controller:
'\Drupal\xml_import_example\Controller\ImportContentFromXML::processAllQueueItemsWithBatch' }
  requirements:
    _permission: 'import content from xml'
```

src / Controller / ImportContentFromXML.php

```
<?php
/**
 * @file
 * Contains \Drupal\xml_import_example\Controller\ImportContentFromXML.
 */

namespace Drupal\xml_import_example\Controller;
```

```
use Symfony\Component\DependencyInjection\ContainerInterface;
use Drupal\Core\Controller\ControllerBase;
use Drupal\Core\Queue\QueueWorkerManager;
use Drupal\Core\Queue\QueueFactory;

/**
 * You can use this constant to set how many queued items
 * you want to be processed in one batch operation
 */
define("IMPORT_XML_BATCH_SIZE", 1);

class ImportContentFromXML extends ControllerBase {

  /**
   * We add QueueFactory and QueueWorkerManager services with the Dependency Injection
solution
   */

  /**
   * @var QueueFactory
   */
  protected $queueFactory;

  /**
   * @var QueueWorkerManager
   */
  protected $queueManager;

  /**
   * {@inheritdoc}
   */
  public function __construct(QueueFactory $queue_factory, QueueWorkerManager $queue_manager)
{
    $this->queue_factory = $queue_factory;
    $this->queue_manager = $queue_manager;
  }

  /**
   * {@inheritdoc}
   */
  public static function create(ContainerInterface $container) {
    $queue_factory = $container->get('queue');
    $queue_manager = $container->get('plugin.manager.queue_worker');

    return new static($queue_factory, $queue_manager);
  }

  /**
   * Get XML from the API and convert it to
   */
  protected function getContentsFromXML() {
    // Here you should get the XML content and convert it to an array of content arrays for
example
    // I use now an example array of contents:
    $contents = array();

    for ($i = 1; $i <= 20; $i++) {
      $contents[] = array(
        'title' => 'Test title ' . $i,
        'body' => 'Test body ' . $i,
```

```
      );
    }

    // Return with the contents
    return $contents;
  }

  /**
   * Page where the xml source is preprocessed
   */
  public function getContentsFromXMLPage() {
    // Get contents array
    $contents = $this->getContentsFromXML();

    foreach ($contents as $content) {
      // Get the queue implementation for import_content_from_xml queue
      $queue = $this->queue_factory->get('import_content_from_xml');

      // Create new queue item
      $item = new \stdClass();
      $item->data = $content;
      $queue->createItem($item);
    }

    return array(
      '#type' => 'markup',
      '#markup' => $this->t('@count queue items are created.', array('@count' =>
count($contents))),
    );
  }

  /**
   * Process all queue items with batch
   */
  public function processAllQueueItemsWithBatch() {

    // Create batch which collects all the specified queue items and process them one after
another
    $batch = array(
      'title' => $this->t("Process all XML Import queues with batch"),
      'operations' => array(),
      'finished' =>
'Drupal\xml_import_example\Controller\ImportContentFromXML::batchFinished',
    );

    // Get the queue implementation for import_content_from_xml queue
    $queue_factory = \Drupal::service('queue');
    $queue = $queue_factory->get('import_content_from_xml');

    // Count number of the items in this queue, and create enough batch operations
    for($i = 0; $i < ceil($queue->numberOfItems() / IMPORT_XML_BATCH_SIZE); $i++) {
      // Create batch operations
      $batch['operations'][] =
array('Drupal\xml_import_example\Controller\ImportContentFromXML::batchProcess', array());
    }

    // Adds the batch sets
    batch_set($batch);
    // Process the batch and after redirect to the frontpage
    return batch_process('<front>');
  }
```

```
  /**
   * Common batch processing callback for all operations.
   */
  public static function batchProcess(&$context) {

    // We can't use here the Dependency Injection solution
    // so we load the necessary services in the other way
    $queue_factory = \Drupal::service('queue');
    $queue_manager = \Drupal::service('plugin.manager.queue_worker');

    // Get the queue implementation for import_content_from_xml queue
    $queue = $queue_factory->get('import_content_from_xml');
    // Get the queue worker
    $queue_worker = $queue_manager->createInstance('import_content_from_xml');

    // Get the number of items
    $number_of_queue = ($queue->numberOfItems() < IMPORT_XML_BATCH_SIZE) ? $queue-
>numberOfItems() : IMPORT_XML_BATCH_SIZE;

    // Repeat $number_of_queue times
    for ($i = 0; $i < $number_of_queue; $i++) {
      // Get a queued item
      if ($item = $queue->claimItem()) {
        try {
          // Process it
          $queue_worker->processItem($item->data);
          // If everything was correct, delete the processed item from the queue
          $queue->deleteItem($item);
        }
        catch (SuspendQueueException $e) {
          // If there was an Exception trown because of an error
          // Releases the item that the worker could not process.
          // Another worker can come and process it
          $queue->releaseItem($item);
          break;
        }
      }
    }
  }

  /**
   * Batch finished callback.
   */
  public static function batchFinished($success, $results, $operations) {
    if ($success) {
     drupal_set_message(t("The contents are successfully imported from the XML source."));
    }
    else {
      $error_operation = reset($operations);
      drupal_set_message(t('An error occurred while processing @operation with arguments :
@args', array('@operation' => $error_operation[0], '@args' => print_r($error_operation[0],
TRUE))));
    }
  }
}
```

src / Plugin / QueueWorker / ImportContentFromXMLQueueBase.php

```
<?php
```

```
/**
 * @file
 * Contains Drupal\xml_import_example\Plugin\QueueWorker\ImportContentFromXMLQueueBase
 */

namespace Drupal\xml_import_example\Plugin\QueueWorker;

use Drupal\Core\Plugin\ContainerFactoryPluginInterface;
use Drupal\Core\Queue\QueueWorkerBase;
use Drupal\Core\Queue\SuspendQueueException;
use Symfony\Component\DependencyInjection\ContainerInterface;
use Drupal\node\Entity\Node;

/**
 * Provides base functionality for the Import Content From XML Queue Workers.
 */
abstract class ImportContentFromXMLQueueBase extends QueueWorkerBase implements
ContainerFactoryPluginInterface {

  // Here we don't use the Dependency Injection,
  // but the create method and __construct method are necessary to implement

  /**
   * {@inheritdoc}
   */
  public function __construct() {}

  /**
   * {@inheritdoc}
   */
  public static function create(ContainerInterface $container, array $configuration,
$plugin_id, $plugin_definition) {
    return new static();
  }

  /**
   * {@inheritdoc}
   */
  public function processItem($item) {
    // Get the content array
    $content = $item->data;
    // Create node from the array
    $this->createContent($content);
  }

  /**
   * Create content
   *
   * @return int
   */
  protected function createContent($content) {
    // Create node object from the $content array
    $node = Node::create(array(
      'type'  => 'page',
      'title' => $content['title'],
      'body'  => array(
        'value'  => $content['body'],
        'format' => 'basic_html',
      ),
    ));
```

```
    $node->save();
  }
}
```

src / Plugin / QueueWorker / ImportContentFromXMLQueue.php

```
<?php

namespace Drupal\xml_import_example\Plugin\QueueWorker;

/**
 * Create node object from the imported XML content
 *
 * @QueueWorker(
 *   id = "import_content_from_xml",
 *   title = @Translation("Import Content From XML"),
 *   cron = {"time" = 60}
 * )
 */
class ImportContentFromXMLQueue extends ImportContentFromXMLQueueBase {}
```

Donc, ceci est le module de travail, vous pouvez le tester sur votre site.

Si vous visitez l'URL **/ get-contents-from-xml** 20, les éléments de la file d'attente sont créés à partir d'un tableau de contenu.

Le **src / Plugin / QueueWorker / ImportContentFromXMLQueue.php** contient cette annotation: **cron = {"time" = 60}**

Donc, si vous exécutez cron, les éléments de la file d'attente sont traités pendant un maximum de 60 secondes. Vous pouvez augmenter ou diminuer cette fois-ci avec cette annotation.

Si vous supprimez la ligne **cron = {"time" = 60}** , cron ne fait rien avec vos éléments de file d'attente.

Si vous souhaitez traiter tous les éléments de la file d'attente de votre navigateur, vous devez visiter l'URL suivante: **/ process-all-queue-items**

Il collectera tous vos éléments de file d'attente, créera des opérations par lots à partir de ces éléments et les traitera ensuite l'un après l'autre.

Lire Exemple pour l'API Drupal 8 Queue et l'API Batch en ligne:
https://riptutorial.com/fr/drupal/topic/3786/exemple-pour-l-api-drupal-8-queue-et-l-api-batch

Chapitre 9: Formateur de champs

Introduction

Un formateur de champ spécifie la manière dont un champ est rendu dans les modèles Drupal. Le formateur utilisé par chaque champ peut être configuré dans l'onglet *Gérer* associé au type d'entité que vous configurez.

Les champs peuvent avoir différents formateurs en fonction du mode d'affichage affiché, ce qui vous permet de contrôler le rendu d'un champ dans différentes parties de votre site Web.

Remarques

Certaines choses sont importantes à prendre en compte lors de l'implémentation d'un Field Formatter.

L'implémentation de votre formateur doit être à l'intérieur de votre module dans le dossier `src/Plugin/Field/FieldFormatter` . Les annotations sont également essentielles car elles identifient votre module et les types de champs auxquels il s'applique.

Dans cet exemple, ce formateur est applicable uniquement aux champs du type `email` . Vous pouvez appliquer votre formateur à un certain nombre de champs si nécessaire. Si votre formateur serait, pour quelque raison que ce soit, applicable aux champs de courrier électronique et de date:

```
field_type = {
  "email",
  "date",
}
```

Un écueil auquel j'ai dû faire face lors de la première mise en œuvre de formateurs de champs avec des paramètres est que les paramètres n'ont pas été enregistrés lorsqu'ils ont été modifiés. Il n'y a pas de méthode de sauvegarde explicite et la solution consiste à implémenter la méthode `defaultSettings()` et à spécifier les noms de champs qui constituent votre formulaire de configuration. N'oubliez pas non plus de définir la valeur `#default_value` dans la méthode `settingsForm` .

Si vous voulez avoir un modèle TWIG spécifique pour votre formateur, c'est aussi simple que de configurer une clé `#theme` lors de la construction du tableau de rendu dans la méthode `viewElements` puis, dans votre fichier `.module` , implémenter `hook_theme`

```
function obfuscator_field_formatter_theme() {
  return [
    'obfuscator_field_formatter' => [
      'variables' => array('title' => NULL, 'url' => NULL),
      'template' => 'obfuscator-field-formatter'
    ],
```

```
  ];
}
```

Ensuite, créez le dossier `templates` à la racine de votre module et ayez un fichier nommé `obfuscator-field-formatter.twig.html` où vous produirez le balisage dont vous avez besoin. Dans cet exemple, les variables from the render `#title` et `#url` seront disponibles.

Examples

Formateur de courrier électronique obscurci

Dans notre exemple, nous allons créer un formateur personnalisé pour les adresses e-mail, ce qui nous permettra d'afficher des adresses e-mail obscurcies pour tromper ces méchants spammeurs.

Le formateur aura quelques options de configuration qui nous permettront de contrôler la façon dont l'adresse e-mail est masquée:

- Supprimer @ et. et les remplacer par un espace,
- Remplacez @ par à et. par point,

S'il vous plaît noter que ceci est juste un exemple académique pour montrer comment les formateurs de terrain sont affichés et configurés et n'est évidemment pas très utile pour l'anti-spaming réel.

Cet exemple suppose que vous avez un module nommé `obfuscator_field_formatter` correctement configuré et activé.

```
namespace Drupal\obfuscator_field_formatter\Plugin\Field\FieldFormatter;

use Drupal\Core\Field\FieldDefinitionInterface;
use Drupal\Core\Field\FieldItemInterface;
use Drupal\Core\Field\FieldItemListInterface;
use Drupal\Core\Field\FormatterBase;
use Drupal\Core\Form\FormStateInterface;

/**
 * Plugin implementation of the 'example_field_formatter' formatter.
 *
 * @FieldFormatter(
 *   id = "email_obfuscator_field_formatter",
 *   label = @Translation("Obfuscated Email"),
 *   field_types = {
 *     "email"
 *   }
 * )
 */
class ObfuscatorFieldFormatter extends FormatterBase {
  private $options = [];

  public function __construct($plugin_id, $plugin_definition, FieldDefinitionInterface
$field_definition, array $settings, $label, $view_mode, array $third_party_settings) {
    parent::__construct($plugin_id, $plugin_definition, $field_definition, $settings, $label,
$view_mode, $third_party_settings);
```

```
  $this->options = [
    'remove_chars' => $this->t('Remove @ and . and replace them with a space'),
    'replace_chars' => $this->t('Replace @ by at and . by dot'),
  ];
}

public static function defaultSettings() {
  return [
    'obfuscator_formatter_type' => '',
  ] + parent::defaultSettings();
}

public function settingsForm(array $form, FormStateInterface $form_state) {
  return [
    'obfuscator_formatter_type' => [
      '#type' => 'select',
      '#title' => $this->t('Obfuscation Type'),
      '#options' => $this->options,
      '#default_value' => $this->getSetting('obfuscator_formatter_type'),
    ]
  ] + parent::settingsForm($form, $form_state);
}

public function settingsSummary() {
  $summary = parent::settingsSummary();
  $type = $this->getSetting('obfuscator_formatter_type');

  if(!is_null($type) && !empty($type)) {
    $summary[] = $this->t('Obfuscation: @value', ['@value' => $this->options[$type]]);
  }

  return $summary;
}

public function viewElements(FieldItemListInterface $items, $langcode) {
  $elements = [];

  foreach ($items as $delta => $item) {
    $elements[$delta] = [
      '#type' => 'inline_template',
      '#template' => '{{ value|nl2br }}',
      '#context' => ['value' => $this->viewValue($item)],
    ];
  }

  return $elements;
}

protected function viewValue(FieldItemInterface $item) {
  $obfuscated = $item->value;
  $type = $this->getSetting('obfuscator_formatter_type');

  switch($type) {
    case 'remove_chars': {
      $obfuscated = str_ireplace(['@', '.'], ' ', $item->value);
      break;
    }

    case 'replace_chars': {
      $obfuscated = str_ireplace(['@', '.'], [' AT ', ' DOT '], $item->value);
```

```
        break;
      }
    }

    return $obfuscated;
  }
}
```

Lire Formateur de champs en ligne: https://riptutorial.com/fr/drupal/topic/8792/formateur-de-champs

Chapitre 10: Le module Règles

Introduction

Le module Rules est un moteur qui permet aux administrateurs du site d'automatiser les actions à exécuter de manière conditionnelle, soit par programmation, soit en réponse à des événements prédéterminés.

Les règles peuvent réagir aux **règles** des **événements** qui se produisent sur un site Drupal, comme un utilisateur ouvrant une session. Et il peut effectuer des **règles** de suivi personnalisées **actions,** telles que la redirection vers une certaine page, qui doivent être conditionnellement exécutée si certaines **conditions règles** sont satisfaites .

Remarques

Ressources

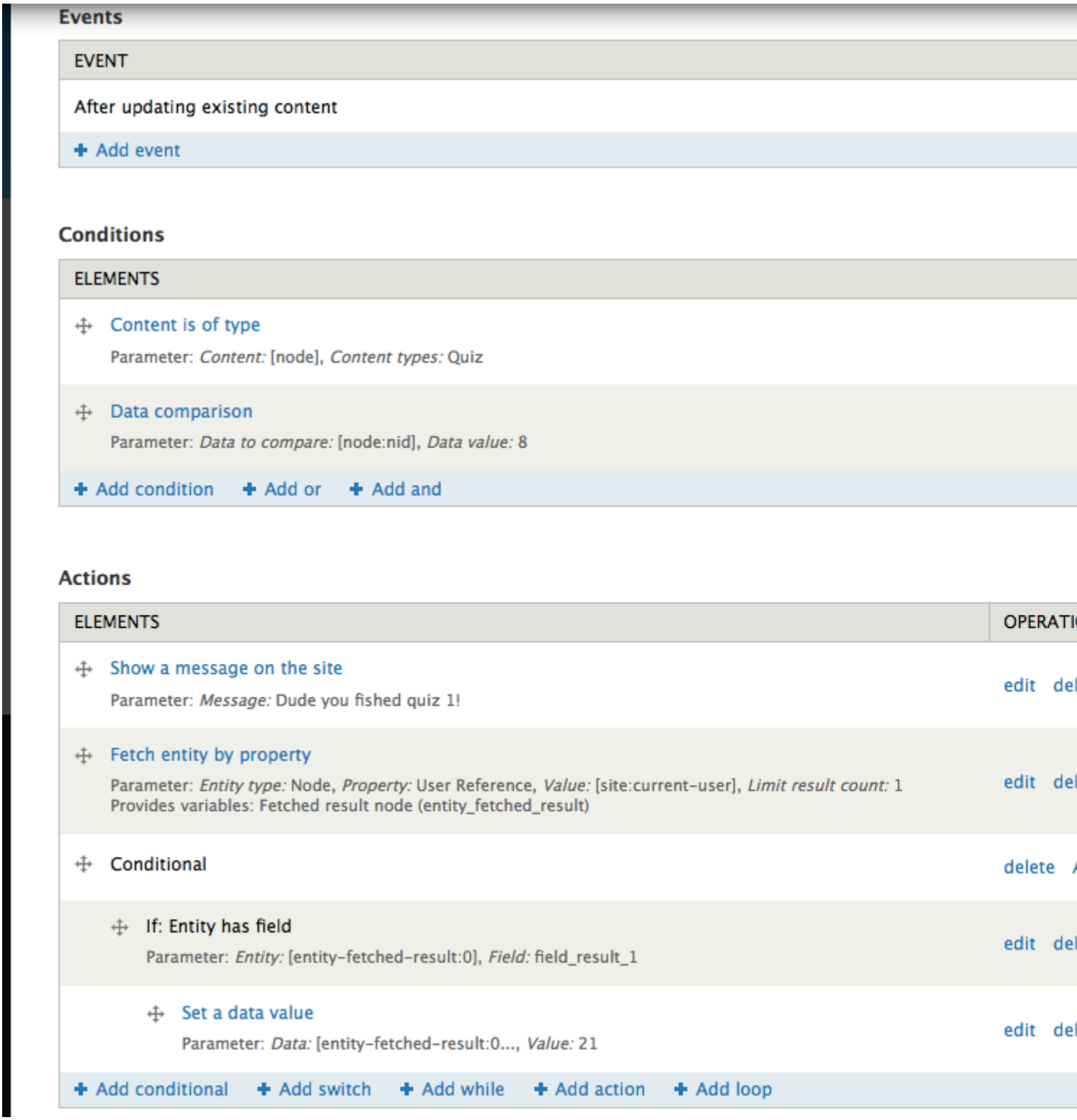
- **Didacticiels vidéo** : Johan Falk a fait un travail remarquable au début des 7 jours de Drupal en créant un ensemble impressionnant de didacticiels pour « *apprendre le cadre des règles* » avec un ensemble de plus de 30 vidéos et `nodeone.se` associés hébergés sur `nodeone.se` (license = Attribution-Noncommercial) -Share Alike 3.0).

  Cependant, le domaine nodeone.se ne les héberge plus. Learn Rules est une tentative de récupération de ces précieux articles de blog (avec des liens vers les vidéos correspondantes).

- **Le livre de règles minuscule** est un lien de 15 pages sur le module Règles .

Examples

Une règle personnalisée affichée à l'aide de l'interface utilisateur de règles



Une règle personnalisée affichée dans le format d'exportation de règles

Voici un exemple de ***règles au format d'exportation de règles*** :

```
{ "rules_display_userpoints_after_updating_content" : {
    "LABEL" : "Display userpoints after updating content",
    "PLUGIN" : "reaction rule",
    "OWNER" : "rules",
    "REQUIRES" : [ "userpoints_rules", "rules", "rules_conditional" ],
    "ON" : { "node_update" : [] },
    "DO" : [
```

```
      { "userpoints_rules_get_current_points" : {
          "USING" : { "user" : [ "site:current-user" ], "tid" : "all" },
          "PROVIDE" : { "loaded_points" : { "total_points" : "Number of points in all
categories together" } }
        }
      },
      { "drupal_message" : { "message" : "You now have [total-points:value] points" } },
      { "CONDITIONAL" : [
          {
            "IF" : { "NOT data_is" : { "data" : [ "total-points" ], "op" : "\u003C", "value" :
"20" } },
            "DO" : [
              { "drupal_message" : { "message" : "You have sufficient points (you still have
[total-points:value] ...)." } }
            ]
          },
          { "ELSE" : [
              { "drupal_message" : { "message" : "You DO NOT have sufficient points (you only
have [total-points:value] ...)." } }
            ]
          }
        ]
      }
    ]
  }
}
```

Il récupère, en tant que toute première action de règles (pas de condition de règles!), La quantité actuelle de points utilisateur d'un utilisateur. Si le montant est d'au moins 20, il affichera un message commençant par "Vous avez suffisamment de points ...", sinon le message commencera par "Vous n'avez pas suffisamment de points ...".

Traitement des éléments de la collection de champs avec des règles

Le traitement des éléments de la collection Field avec Rules est amusant, vraiment! Jetez un coup d'oeil à cette règle (dans le format d'exportation des règles):

```
{ "rules_calculate_sum_of_prices_in_all_field_collection_items" : {
    "LABEL" : "Calculate sum of prices in all field collection items",
    "PLUGIN" : "reaction rule",
    "OWNER" : "rules",
    "REQUIRES" : [ "rules" ],
    "ON" : { "node_view--article" : { "bundle" : "article" } },
    "IF" : [
      { "entity_has_field" : { "entity" : [ "node" ], "field" : "field_article_details" } }
    ],
    "DO" : [
      { "drupal_message" : { "message" : "\u003Cstrong\u003EDrupal
calculator\u003C\/strong\u003E started ..." } },
      { "variable_add" : {
          "USING" : { "type" : "decimal", "value" : "0" },
          "PROVIDE" : { "variable_added" : { "total_price" : "Price total" } }
        }
      },
      { "LOOP" : {
          "USING" : { "list" : [ "node:field-article-details" ] },
          "ITEM" : { "article_details_item" : "Article details item" },
```

```
          "DO" : [
            { "data_calc" : {
                "USING" : {
                  "input_1" : [ "total-price" ],
                  "op" : "+",
                  "input_2" : [ "article-details-item:field-price" ]
                },
                "PROVIDE" : { "result" : { "calculation_result" : "Calculation result" } }
              }
            },
            { "data_set" : { "data" : [ "total-price" ], "value" : [ "calculation-result" ] }
},
            { "drupal_message" : { "message" : "After adding a price of
\u003Cstrong\u003E[article-details-item:field-price]\u003C\/strong\u003E for field collection
item with id \u003Cstrong\u003E[article-details-item:item-id]\u003C\/strong\u003E, subtotal is
\u003Cstrong\u003E[calculation-result:value]\u003C\/strong\u003E." } }
          ]
        }
      },
      { "drupal_message" : { "message" : "The \u003Cstrong\u003ETotal
price\u003C\/strong\u003E for all prices included as field collection items is
\u003Cstrong\u003E[total-price:value]\u003C\/strong\u003E." } },
      { "drupal_message" : { "message" : "\u003Cstrong\u003EDrupal
calculator\u003C\/strong\u003E ended ..." } }
    ]
  }
}
```

Quelques détails supplémentaires sur cette règle sont ci-dessous ...

Événement de règles:

Le contenu est visualisé (de type Article), adaptez le nom de la machine de l' `article` type de contenu à tout ajustement (ou utilisez tout autre événement de règles adapté).

Règles Condition:

Entity a field, alors que l'entité est "node", et le nom de la machine de mon champ de collection de champs est `field_article_details` (adaptez ce nom de machine à tout ajustement, mais assurez-vous d'utiliser le champ de collection de champs lui-même).

Actions sur les règles:

Réveillez-vous, voici où la magie (amusante?) Va se produire ... Voici les règles à suivre:

1. `Afficher un message sur le site` , avec un message comme celui-ci:

   La calculatrice Drupal a commencé ...

2. `Ajoutez une variable` , alors que c'est une variable nommée `total_price` , decimal (2 chiffres), valeur initiale 0.

3. `Ajoutez une boucle` pour parcourir chaque élément de mon champ de collection de champs (avec le nom de la machine `field_article_details`) et effectuez ces actions de règles pour

chaque itération:

- `Calculer une valeur` qui calcule la somme de `total_price` (définie dans l'action de règles 2 ci-dessus) et `article-details-item:field-price` (il s'agit du nom de machine du champ dans la collection de champs contenant les prix, décimal avec 2 chiffres)), et stocke le résultat (somme) dans variable_result de `calculation_result` .

- `Définissez une valeur de données` , qui copie simplement la valeur stockée dans la variable `calculation_result` dans mon `total_price` (défini dans l'action de règles 2 ci-dessus). Remarque: pas sûr (non testé), mais peut - être ce `calculation_result` par la variable peut être remplacé directement `total_price` (dans l'action précédente), de sorte que vous ne auriez pas besoin de cette action.

- `Afficher un message sur le site` , avec un message comme celui-ci:

  > Après avoir ajouté un prix de 3,40 pour l'élément de collecte de champs avec l'identifiant 3, le sous-total est de 15,00.

4. `Afficher un message sur le site` , avec un message comme celui-ci:

   Le prix total pour tous les prix inclus dans les articles de collecte sur le terrain est de 26,23.

5. `Afficher un message sur le site` , avec un message comme celui-ci:

   Calculatrice Drupal terminée ...

Evidemment, cette règle est plutôt un prototype. Une fois que vous êtes convaincu que cela fonctionne comme il se doit, supprimez simplement toutes les actions de règles avec `Afficher un message sur le site` . Donc, seuls les éléments 2 et 3 (sans sa dernière sous-puce) sont laissés comme Actions de règles.

Afficher l'heure ...

Voici un exemple de mes résultats de test, à savoir les messages Drupal affichés:

```
Drupal calculator started ...
After adding a price of 2.45 for field collection item with id 1, subtotal is 2.45.
After adding a price of 9.15 for field collection item with id 2, subtotal is 11.60.
After adding a price of 3.40 for field collection item with id 3, subtotal is 15.00.
After adding a price of 1.23 for field collection item with id 4, subtotal is 16.23.
The Total price for all prices included as field collection items is 26.23.
Drupal calculator ended ...
```

Plus d'informations

Si vous n'êtes pas familier avec les collections de terrain, essayez d'abord de digérer la réponse à " cette question ".

Lire Le module Règles en ligne: https://riptutorial.com/fr/drupal/topic/8921/le-module-regles

Chapitre 11: Le module Vues

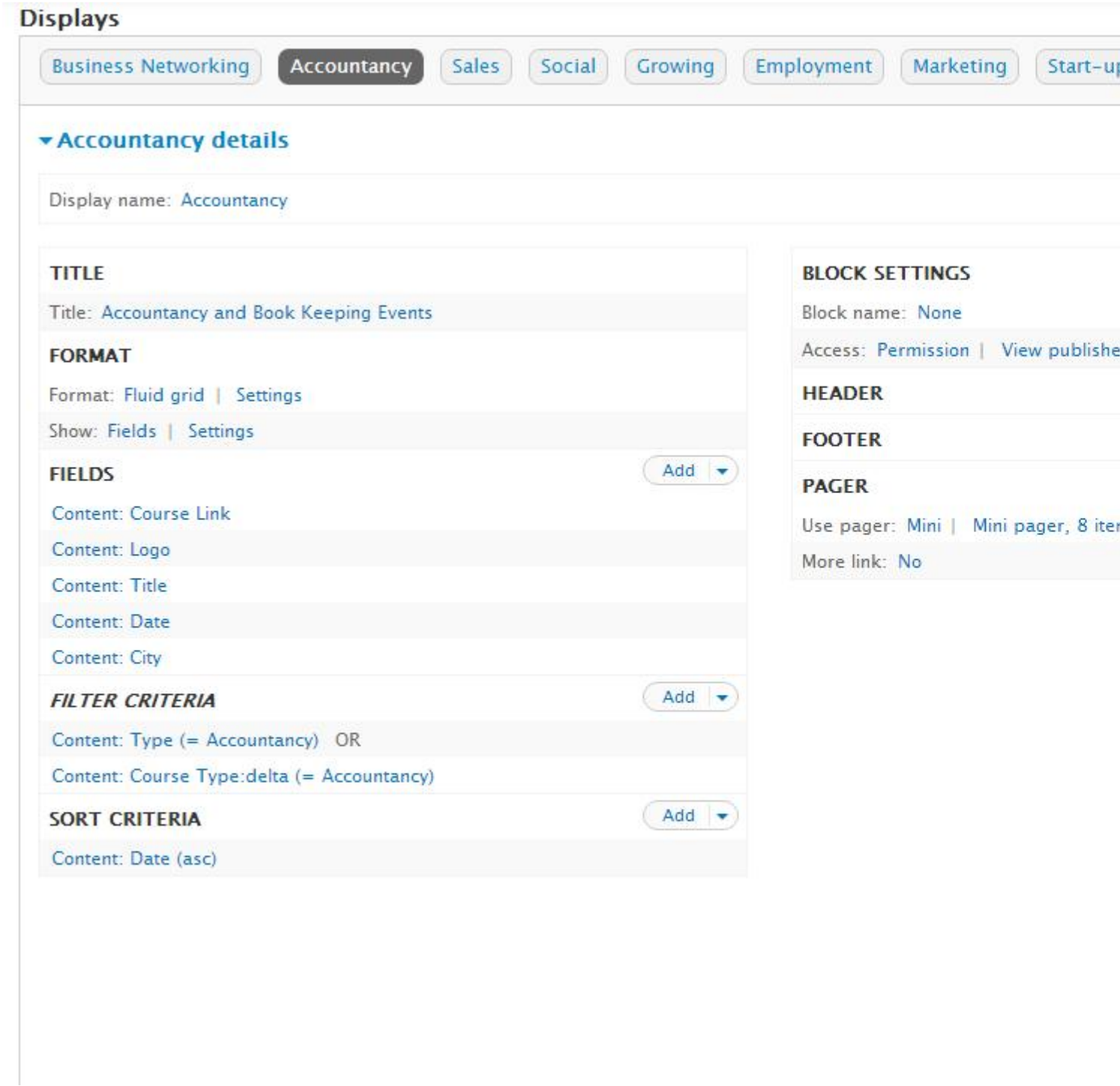
Introduction

Le module Views est un moteur de génération de rapports qui permet aux créateurs de sites de créer toutes sortes de listes. Ces listes peuvent être formatées sous forme de bloc ou de page.

Pour créer (concevoir) une vue, il est nécessaire d'entrer les spécifications souhaitées, telles que les champs, les relations, les filtres, les critères de tri, les affichages, etc.

Examples

Une vue affichée à l'aide de l'interface utilisateur Views



Lire Le module Vues en ligne: https://riptutorial.com/fr/drupal/topic/9553/le-module-vues

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec drupal	acrosman, Adrian Cid Almaguer, chx, Community, dobeerman, Jimmy Ko, Karl Buys, kiamlaluno, Mika A., Morsok, pal4life, Pierre Buyle, Sidney Gijzen
2	API d'entité Drupal 8	Bernard Nandwa
3	Brindille	Ricardo Velhote
4	Cache Drupal et performace	Anurag
5	Développement de modules - Drupal 7	Ajit S, doublejosh, Hermann Döppes, Jimmy Ko, Jimmyb_1991, Morsok, Pierre Buyle
6	Développement du thème - Drupal 7	Adrian Cid Almaguer, Sam Thompson
7	Drush	Adrian Cid Almaguer, darc1n, Jimmy Ko, lastYorsh, L-four, Mark Conroy, Morsok, pbonnefoi, Rishi Kulshreshtha, Sjoerd van der Vis, Wade Burelbach
8	Exemple pour l'API Drupal 8 Queue et l'API Batch	David Czinege
9	Formateur de champs	Ricardo Velhote
10	Le module Règles	Pierre.Vriens
11	Le module Vues	Pierre.Vriens