EBook Gratis

APRENDIZAJE drupal

Free unaffiliated eBook created from
Stack Overflow contributors.

#drupal

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con drupal

Observaciones

Drupal es un sistema de gestión de contenido de código abierto integrado en PHP. Drupal está diseñado para ser flexible y potente, lo que permite a los desarrolladores crear una amplia variedad de sitios, desde blogs y sitios tipo folleto hasta complejas plataformas de comercio electrónico. A través de su arquitectura modular impulsada por la comunidad, Drupal puede proporcionar herramientas para ampliar las funciones básicas para ayudar a acelerar el desarrollo de proyectos grandes y complejos.

Actualmente hay dos versiones compatibles de Drupal: 7 y 8. Drupal 8 se basa en componentes del framework Symfony y muchas otras bibliotecas de terceros para proporcionar estructuras de desarrollo modernas.

Examples

Instalando Drupal con 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
```

Instalación de Drupal 8 con Drupal Console

Consola Drupal

El nuevo CLI para Drupal. Una herramienta para generar código repetitivo, interactuar y depurar Drupal.

Primero, necesitamos instalar la Consola Drupal.

Drupal Console es necesaria no solo para este momento, sino también para futuras instalaciones.

```
# 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
```

Puede llamar a la `drupal list` para ver todos los comandos disponibles.

En el siguiente paso descargaremos el código fuente de Drupal.

```
drupal site:new
```

La consola le pedirá que elija una carpeta para descargar Drupal. Y en el siguiente paso se te pedirá que elijas la versión de Drupal para descargar. Recomiendo seleccionar el último.

Entonces, cuando se descarga Drupal necesitas instalarlo.

```
drupal site:install
```

Después de unos simples pasos, su sitio de Drupal estará listo.

Con esta metodología, una nueva instalación de Drupal nos lleva entre 5 y 7 minutos, todo desde la línea de comandos.

Conceptos de drupal

Versiones

```
Release Date
```

Versión	Fecha de lanzamiento
8.2.4	07 de diciembre de 2016
7.53	07 de diciembre de 2016
6.38 (no soportado)	24 de febrero de 2016
5.23 (no soportado)	11 de agosto de 2010

Tipos de entidad

En versiones anteriores de Drupal, el sistema de campo solo se usaba en tipos de contenido. Ahora, gracias a la API de la entidad, podemos agregar campos a otras cosas, como comentarios. Las entidades de campo hacen que Drupal sea sumamente flexible. Un tipo de entidad es una abstracción útil para agrupar campos. A continuación se muestran los tipos de entidad en el núcleo de Drupal:

- Nodos (contenido)
- Comentarios

- Archivos
- Términos de taxonomía
- Vocabularios de taxonomía
- Usuarios

También puede crear nuevos tipos de tipos de entidades donde las opciones anteriores no se ajusten a sus necesidades.

manojos

Los paquetes son una implementación de un tipo de entidad al que se pueden adjuntar campos. Puede considerar los paquetes como subtipos de un tipo de entidad. Con los nodos de contenido (un tipo de entidad), por ejemplo, puede generar paquetes (subtipos) como artículos, publicaciones de blog o productos. Sin embargo, no todos los tipos de entidades tienen paquetes. Por ejemplo, los usuarios no tienen paquetes separados (subtipos). Para los tipos de entidades que permiten paquetes, puede crear tantos paquetes (subtipos) como desee. Luego, utilizando el sistema Field, puede agregar diferentes campos a cada paquete. Los ejemplos incluyen un campo de descarga de archivos en Páginas Básicas y un campo de subtítulos en Artículos.

Campos

Un campo es una pieza reutilizable de contenido. En términos técnicos, cada campo es un tipo de datos primitivo, con validadores personalizados y widgets para editar y formateadores para mostrar. Puede leer más para obtener una guía para desarrolladores sobre el uso de la API de Drupal 7 Fields .

Lo que es importante saber en relación con las entidades es que los campos se pueden agregar a cualquiera de los paquetes (o tipos de entidades) para ayudar a organizar sus datos.

Digamos, por ejemplo, que creas un tipo de contenido con un campo de texto no estructurado y usas HTML para estructurar partes de él, como una sección de resumen o precios. Eso haría más difícil, entonces, controlar cómo se mostraban, o hacer conexiones entre diferentes tipos de contenido relacionado.

Aquí es donde el uso de campos es esencial. Puede crear un campo de resumen de tipo Texto largo, así como campos de precios de tipo Decimal.

Entidad

Una entidad sería una instancia de un tipo de entidad en particular, como un comentario, un término de taxonomía o un perfil de usuario o un paquete como una publicación de blog, un artículo o un producto.

Puedes usar entity_load para cargar cualquier entidad. Sin embargo, tenga en cuenta que el núcleo no proporciona una función de guardar o eliminar, pero gracias al módulo Entity API se agregan las piezas faltantes (entity_create (), entity_save (), entity_delete (), entity_view () y entity_access ()).

Poniendo esto en términos de diseño / programación orientados a objetos ...

Si proviene de un fondo de OOD / P y está tratando de entender mejor cuáles son estos conceptos clave, la siguiente asignación sugerida podría ayudar (aunque no sea estrictamente cierto desde la perspectiva de un purista):

- Un ***tipo de entidad*** es una ***clase base***
- Un ***paquete*** es una ***clase extendida***
- Un ***campo*** es un ***miembro de clase*** , ***propiedad*** , ***variable*** o ***instancia de campo*** (dependiendo de su preferencia de nombre)
- Una ***entidad*** es un ***objeto*** o ***instancia*** de una ***clase base*** o ***extendida***

Todos estos cuatro conceptos de OOD / P son especiales porque se pueden serializar (almacenados, por ejemplo, en una base de datos o archivo). La serialización se realiza a través de la API de la entidad.

Lea Empezando con drupal en línea: https://riptutorial.com/es/drupal/topic/1135/empezando-con-drupal

Capítulo 2: Desarrollo de módulos - Drupal 7

Observaciones

Los ejemplos para el módulo de desarrolladores deberían usarse como una referencia para el desarrollo de módulos idealmente. Tiene una explicación de todas las API principales, uso bien documentado. Es todo para que los principiantes entiendan el desarrollo del módulo.

Examples

Módulo básico que proporciona una página sencilla.

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!"
}
```

Módulo básico que proporciona un bloque personalizado.

custom_module.info

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

módulo_modulo personalizado

```
/**
 * 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;
}
```

Formulario personalizado básico para incluir en la página o en ejemplos de bloque.

Funciones simples de formulario, validación y envío para crear una función de "lista de correo". Esto puede aplicarse a la página básica o a los ejemplos de bloque básicos.

Asume que ha creado una tabla en la base de datos de drupal llamada 'mailing_list' con los campos nombre, apellido y dirección de correo electrónico.

Información adicional sobre la API del formulario y opciones de campo adicionales: 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.
```

```
}
```

Módulo básico que proporciona un bloque personalizado.

custom_module.info

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

módulo_modulo personalizado

```
/**
 * 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;
```

```
}
```

Formulario personalizado básico para incluir en la página o en ejemplos de bloque.

Funciones simples de formulario, validación y envío para crear una función de "lista de correo". Esto puede aplicarse a la página básica o a los ejemplos de bloque básicos.

Asume que ha creado una tabla en la base de datos de drupal llamada 'mailing_list' con los campos nombre, apellido y dirección de correo electrónico.

Información adicional sobre la API del formulario y opciones de campo adicionales: 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.
}
```

Ejemplo de archivo custom_module.install para crear una tabla de base de datos

*Se puede utilizar junto con el **ejemplo de formulario personalizado** para crear una tabla en la base de datos de Drupal para una función de Lista de correo.*

Este ejemplo se creó al crear la tabla directamente en mi base de datos de desarrollo, luego se crearon los datos para hook_schema () usando el módulo de esquema .

Esto permite la creación automática de tablas durante la instalación del módulo en los sitios de preparación y producción.

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,
      ),
    ),
  );
}
```

Lea Desarrollo de módulos - Drupal 7 en línea:
https://riptutorial.com/es/drupal/topic/2456/desarrollo-de-modulos---drupal-7

Capítulo 3: Desarrollo del tema - Drupal 7

Examples

Escribiendo archivos temáticos .info

El archivo .info es un archivo de texto estático para definir y configurar un tema. Cada línea en el archivo .info es un par clave-valor con la clave a la izquierda y el valor a la derecha, con un "signo igual" entre ellas (por ejemplo, *nombre = mi_tema*).

Los puntos y coma se utilizan para comentar una línea. Algunas teclas utilizan una sintaxis especial con corchetes para crear una lista de valores asociados, denominada "matriz". Si no está familiarizado con los arreglos, eche un vistazo a los archivos .info predeterminados que vienen con Drupal y lea las explicaciones de los ejemplos que siguen. Aunque la extensión del archivo .info no está abierta de forma nativa por una aplicación, puede usar TextEdit en una Mac o Bloc de notas en una computadora con Windows para ver, editar y guardar sus cambios.

Requisitos del nombre del tema

El nombre debe comenzar con un carácter alfabético, puede contener números y guiones bajos, pero no guiones, espacios o puntuación. Drupal usará el nombre para formar varias funciones en PHP y, por lo tanto, tiene las mismas limitaciones.

No elija nombres que ya estén siendo utilizados por los módulos instalados , ya que todos los componentes instalados deben tener nombres únicos.

Una de las mejores prácticas es usar prefijos al nombrar el tema personalizado de un sitio, para garantizar nombres únicos para los temas. Un sitio llamado example.com puede usar nombres de temas como ex_themename.

Debido a que el archivo .info está en caché, debe borrar la caché antes de que se muestren los cambios en su sitio.

El archivo .info también puede especificar a qué configuración de tema se debe acceder desde la interfaz de administración de Drupal, como verá pronto.

Codificación

El archivo debe guardarse como UTF-8 sin una marca de orden de bytes (BOM).

Contenido

Drupal entiende las claves enumeradas a continuación. Drupal usará valores predeterminados para las claves opcionales que no están presentes en el archivo .info. Vea los ejemplos establecidos para los temas centrales.

- nombre ***requerido***

- descripción ***recomendada***
- captura de pantalla
- versión ***desalentada***
- núcleo ***requerido***
- motor
- tema base
- regiones
- caracteristicas
- configuración de temas
- hojas de estilo
- guiones
- php

Archivo .info del tema

```
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
```

Lea Desarrollo del tema - Drupal 7 en línea: https://riptutorial.com/es/drupal/topic/2715/desarrollo-del-tema---drupal-7

Capítulo 4: Drupal 8 Entity API

Introducción

El Entity System en Drupal 8 permite a los desarrolladores crear fácilmente tipos de contenido personalizados y modelar relaciones de datos en torno a estos tipos. La API extensa proporciona soporte para la generación de formularios, validación de datos, configuración, enrutamiento y muchas otras características.

Examples

Crea una entidad de contenido usando Drupal Console.

La consola de Drupal lleva los andamios al ecosistema de Drupal y facilita la generación de una entidad de contenido.

En la mayoría de los casos, le resultará más fácil trabajar con una entidad personalizada dentro de un módulo personalizado.

Paso 1: Generar un módulo

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

Sigue las indicaciones y crea tu módulo personalizado.

Paso 2: Generar una entidad de contenido

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

Siga las indicaciones en su línea de comando asegurándose de seleccionar el módulo personalizado creado en el paso anterior.

Lea Drupal 8 Entity API en línea: https://riptutorial.com/es/drupal/topic/10681/drupal-8-entity-api

Capítulo 5: Drupal caché y performace

Introducción

Se ha utilizado la memoria caché para el sitio o el sistema para mejorar la entrega de contenido rápidamente para los usuarios finales. Este tema se crea para explorar sobre el mecanismo de almacenamiento en caché incorporado de Drupal y proporcionar información sobre cómo usarlo. Necesitamos explorar la función de almacenamiento en caché incorporada de Drupal con los módulos contribuidos externos como Varnish, Memcache, Authcache, File Cache, etc. Están disponibles para mejorar el rendimiento del sitio. Puede encontrar el ejemplo y las opciones de almacenamiento en caché más adecuados para usar con su sitio en este tema.

Examples

Habilitar el sitio Drupal y bloquear el caché

El propio Drupal proporciona buenas opciones de almacenamiento en caché para aumentar la velocidad de la página y servir las páginas rápidamente a los usuarios finales. Los cachés se utilizan para mejorar el rendimiento de su sitio Drupal. Pero también tiene el inconveniente de que a veces podría llevar los datos "obsoletos". Esto significa que, a veces, el sistema puede comenzar a servir las páginas antiguas del caché.

¿Cómo habilitar el sitio de Drupal y bloquear el caché en varias versiones de Drupal? Ver abajo para respuestas: Drupal 6:

1. Vaya a Administrar -> Configuración del sitio -> Rendimiento.
2. Activar opciones de caché
3. Habilite la agregación de archivos JS / CSS y guárdelo.

Drupal 7:

1. Vaya a Administrar -> Configuración -> Desarrollo -> Rendimiento.
2. Activar opciones de caché
3. Habilite la agregación de archivos JS / CSS y guárdelo.

Lea Drupal caché y performace en línea: https://riptutorial.com/es/drupal/topic/10082/drupal-cache-y-performace

Capítulo 6: Drush

Observaciones

¿Qué es Drush?

Drush es una interfaz de secuencias de comandos de línea de comandos para sitios Drupal. Permite la gestión de línea de comandos de los sitios Drupal.

Examples

Comandos de drush

Estado de drush

```
drush status
```

Esto le dará una visión general de su sitio Drupal. Versión, URI, ubicación de la base de datos, rutas de archivo, tema predeterminado, etc. Si usa este comando y no ve esta información, significa que está en una carpeta incorrecta y Drush no sabe a qué sitio de Drupal se refiere.

Restablecimiento de contraseña para cualquier usuario

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

Donde "admin" es un nombre de usuario existente y "newpassword" es la contraseña deseada.

Generar una sola vez la URL de inicio de sesión del administrador

```
drush uli
```

Genera una URL que se puede utilizar para iniciar sesión en la sección de administración. La URL tiene un token de uso único. El resultado debería verse así:

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

A veces, el nombre de host o la IP no se pueden resolver, y el resultado es una advertencia como esta:

```
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 solución está usando el parámetro "--url" como el siguiente ejemplo:

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

Borrando los cachés

```
drush cache-rebuild
```

Reconstruye el caché para Drupal 8. Para drupal 7, puedes usar

```
drush cache-clear
```

Estos comandos también están disponibles más cortos con

```
drush cr
```

o

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

Habilitar módulos

```
drush pm-enable mymodule
```

Habilitar 'mymodule' como activar un módulo en la interfaz de administración

Estos comandos también están disponibles más cortos con

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

Mantenimiento

Para habilitar el modo de mantenimiento usando Drush puedes usar este comando:

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

Recuerde borrar las cachés después de habilitar / deshabilitar el modo de mantenimiento.

Configuración de exportación

En Drupal 7 y versiones inferiores, su configuración probablemente se almacena utilizando el módulo de Características . Para actualizar una característica con cambios de las bases de datos use este comando:

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

También puedes usar esta taquigrafía:

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

Drupal 8 utilizando "Gestión de configuración". Para exportar su configuración con drush use este comando.

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

También puedes usar el comando abreviado

```
drush cex -y
```

Instalar drush

Instalación global manual

Para OS X y Linux:

1. Muestra Terminal, Bash o tu shell normal.
2. Escriba lo siguiente en la 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
```

Para ventanas:

1. Descarga Drush desde GitHub.
2. Extraiga el archivo comprimido en la unidad deseada, por ejemplo.

```
C:\
```

3. Instale Drush, defina la ruta de la carpeta extraída en la variable de ruta del entorno que también debe incluir `Apache, PHP and MySQL` .

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

4. Verifique que Drush funcione:

```
drush status
```

Instalación global del compositor

1. Instala Composer globalmente .
2. Agregue el directorio `bin` de Composer a la ruta del sistema colocando `export PATH="$HOME/.composer/vendor/bin:$PATH"` en su ~ / .bash_profile (OS X) o ~ / .bashrc (Linux).
3. Instalar Drush:

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

4. Verifique que Drush funcione:

```
drush status
```

Más detalles de Drush Docs

Instalar drush

Instalación manual

Escriba los siguientes comandos en la 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.
```

Compositor

Suponiendo que el compositor está instalado.

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

Lea Drush en línea: https://riptutorial.com/es/drupal/topic/2062/drush

Capítulo 7: Ejemplo para Drupal 8 Queue API y Batch API

Examples

Un módulo de ejemplo para ayudar a comprender la API de cola y la API de lote en 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 {}
```

Así que este es el módulo de trabajo, puede probarlo en su sitio.

Si visita la cola **/ get-contents-from-xml** URL 20, los elementos de la cola se hacen desde una matriz de contenido.

Src / Plugin / QueueWorker / ImportContentFromXMLQueue.php contiene esta anotación: **cron = {"time" = 60}**

Entonces, si ejecuta cron, los elementos de la cola se procesan durante 60 segundos como máximo. Puede aumentar o disminuir este tiempo, con esa anotación.

Si elimina la línea **cron = {"time" = 60}** , cron no hace nada con los elementos de la cola.

Si desea procesar todos los elementos de la cola en su navegador, debe visitar la siguiente url: **/ process-all-queue-items**

Recolectará todos los elementos de la cola, creará operaciones por lotes a partir de ellos y, luego, procesará uno tras otro.

Lea Ejemplo para Drupal 8 Queue API y Batch API en línea:
https://riptutorial.com/es/drupal/topic/3786/ejemplo-para-drupal-8-queue-api-y-batch-api

Capítulo 8: El modulo de reglas

Introducción

El módulo de Reglas es un motor que permite a los administradores del sitio automatizar acciones para ser ejecutadas condicionalmente, ya sea programáticamente o en respuesta a eventos predeterminados.

Las reglas pueden reaccionar a los **eventos de reglas que se** producen en un sitio de Drupal, como el inicio de sesión de un usuario. Y puede realizar **acciones de reglas de** seguimiento personalizadas, como redirigir a una página determinada, que se ejecutarán condicionalmente si se cumplen algunas **condiciones de reglas** .

Observaciones

Recursos

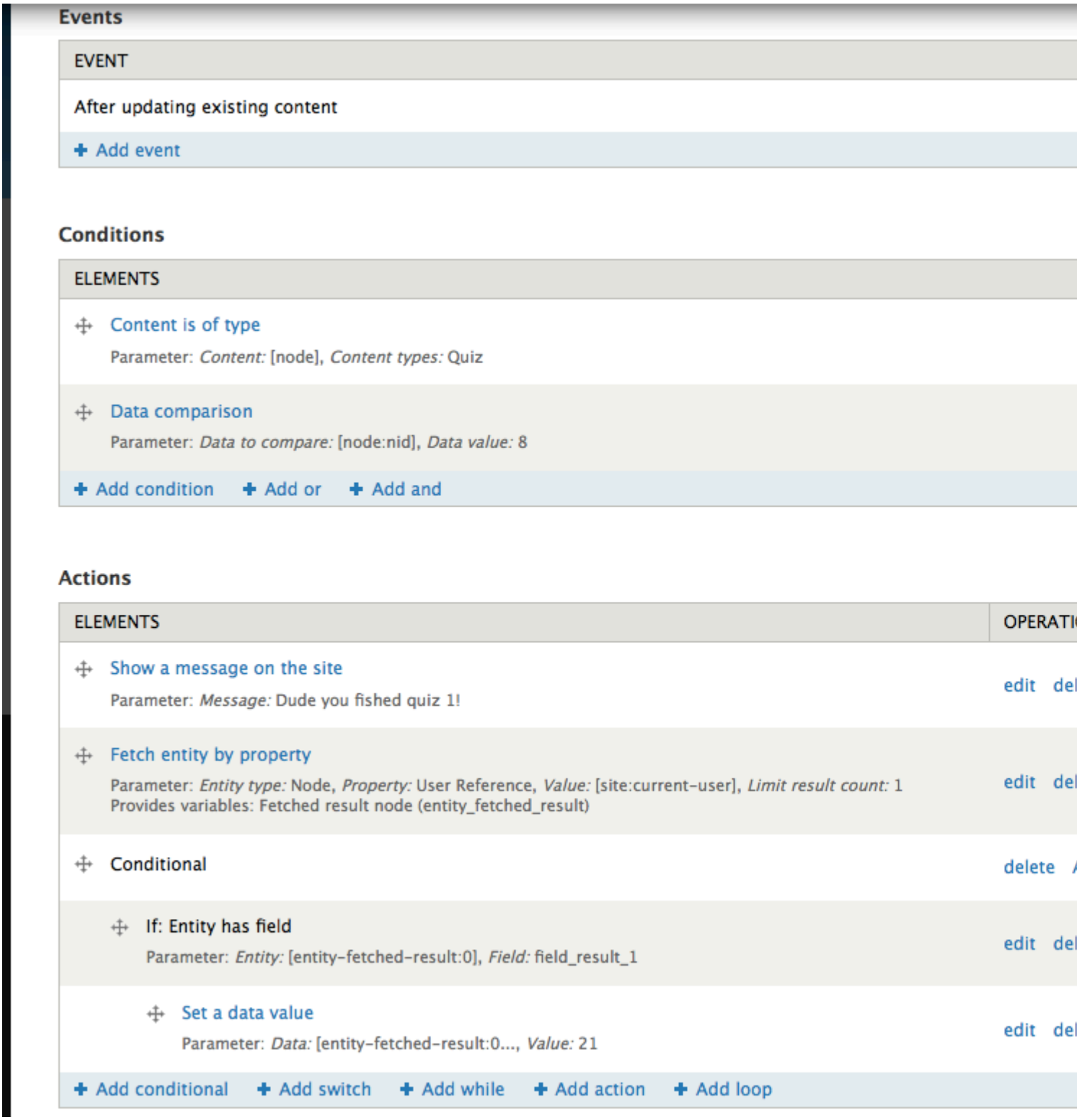
- **Video tutoriales** : Johan Falk hizo un trabajo increíble en los primeros 7 días de Drupal al crear un impresionante conjunto de tutoriales para " *Aprender el marco de reglas* " con un conjunto de más de 30 videos y `nodeone.se` de `nodeone.se` relacionados alojados en `nodeone.se` (license = Attribution-Noncommercial -Compartir igual 3.0).

  Sin embargo, el dominio nodeone.se ya no los aloja. Learn Rules es un intento de recuperar estas valiosas entradas de blog (con enlaces relacionados a los videos correspondientes).

- **El Tiny Book of Rules** es un inicio rápido (15 páginas) sobre el módulo de Reglas .

Examples

Una regla personalizada que se muestra mediante la interfaz de usuario de reglas.



Una regla personalizada que se muestra en el formato de exportación de reglas.

Aquí hay un ejemplo de ***reglas*** en ***formato de exportación de reglas*** :

```
{ "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] ...)." } }
            ]
          }
        ]
      }
    ]
  }
}
```

Recupera, como la primera Acción de Reglas (¡no la Condición de las Reglas!) La cantidad actual de puntos de usuario de un usuario. Si la cantidad es de al menos 20, se mostrará un mensaje que comienza con "Tienes suficientes puntos ...", de lo contrario, el mensaje comienza con "NO TIENES suficientes puntos ...".

Procesando elementos de colección de campo con reglas

¡Procesar artículos de colección de campo con reglas es divertido, en serio! Eche un vistazo a esta regla (en formato de exportación de reglas):

```
{ "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 ..." } }
    ]
  }
}
```

Algunos detalles más sobre esta regla están abajo ...

Reglas del evento:

Se visualiza el contenido (de tipo Artículo), adapte el nombre de la máquina del `article` tipo de contenido a cualquier ajuste (o use cualquier otro Evento de Reglas que se ajuste).

Condición de las reglas:

La entidad tiene campo, mientras que la entidad es "nodo", y el nombre de la máquina de mi campo de colección de campo es `field_article_details` .

Reglas de Acciones:

Despierte, aquí es donde ocurrirá la magia (¿diversión?) ... Estas son las Acciones de Reglas involucradas:

1. `Mostrar un mensaje en el sitio` , con un mensaje como el siguiente:

   Drupal calculadora comenzó ...

2. `Agregue una variable` , mientras que es una variable llamada `total_price` , decimal (2 dígitos), valor inicial 0.

3. `Agregue un bucle`

, para iterar sobre cada elemento de mi campo de colección de campo (con el nombre de la máquina `field_article_details`), y realice estas Acciones de reglas para cada iteración:

- `Calcule un valor` , que calcula la suma de `total_price` (definido en las Reglas Acción 2 arriba) y `article-details-item:field-price` (este es el nombre de la máquina del campo en la colección de campos que contiene los precios, decimal con 2 dígitos)), y almacena el resultado (suma) en la variable `calculation_result` .

- `Establecer un valor de datos,` que simplemente copia el valor almacenado en la variable `calculation_result` en mi `total_price` (definidos en las Reglas Acción 2 anterior). Nota: no estoy seguro (no probado), pero quizás esta variable de `calculation_result` se puede reemplazar directamente por `total_price` (en la acción anterior), por lo que no necesitaría esta acción.

- `Mostrar un mensaje en el sitio` , con un mensaje como el siguiente:

    Después de agregar un precio de 3.40 para el artículo de recolección de campo con id 3, el subtotal es 15.00.

4. `Mostrar un mensaje en el sitio` , con un mensaje como el siguiente:

    El precio total para todos los precios incluidos como artículos de recolección de campo es 26.23.

5. `Mostrar un mensaje en el sitio` , con un mensaje como el siguiente:

    Calculadora Drupal terminó ...

Obviamente, esta regla es más bien un prototipo. Una vez que esté convencido de que funciona como debería, simplemente elimine todas las Acciones de reglas con `Mostrar un mensaje en el sitio` . De modo que solo los elementos 2 y 3 (sin su última sub-viñeta) quedan como Acciones de Reglas.

Tiempo de la funcion ...

Aquí hay una muestra de los resultados de mis pruebas, es decir, los mensajes de Drupal que se muestran:

```
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 ...
```

Más información

Si no está familiarizado con las colecciones de campo, intente primero digerir la respuesta a " esta pregunta ".

Lea El modulo de reglas en línea: https://riptutorial.com/es/drupal/topic/8921/el-modulo-de-reglas

Capítulo 9: El módulo de Vistas

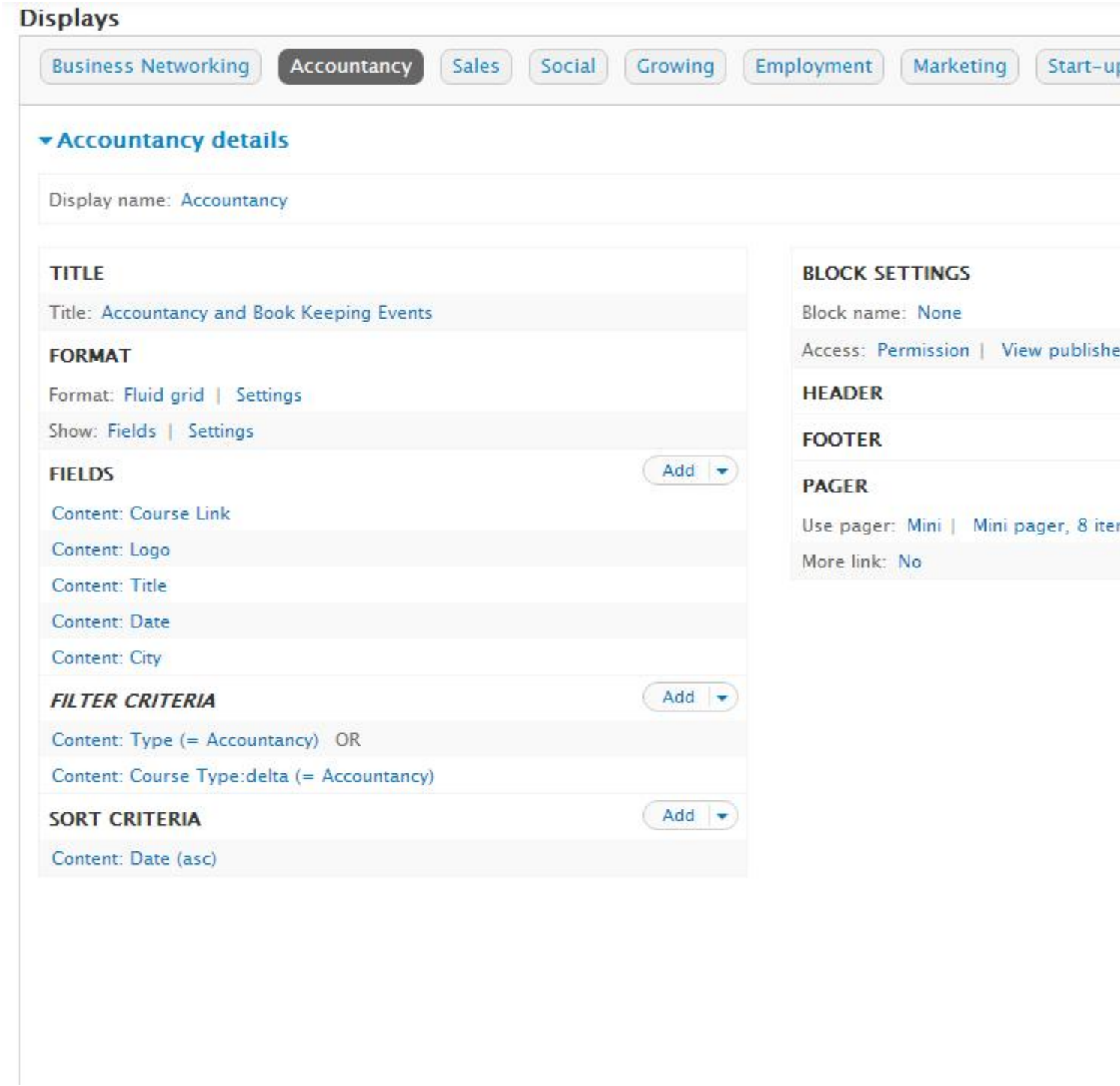
Introducción

El módulo Vistas es un motor de informes que permite a los creadores de sitios crear todo tipo de listas. Estas listas se pueden formatear como un bloque o una página.

Para crear (diseñar) una vista, se requiere ingresar las especificaciones deseadas, como campos, relaciones, filtros, criterios de clasificación, pantallas, etc.

Examples

Una vista que se muestra mediante la interfaz de usuario de Vistas.



Lea El módulo de Vistas en línea: https://riptutorial.com/es/drupal/topic/9553/el-modulo-de-vistas

Capítulo 10: Formateador de campo

Introducción

Un formateador de campos especifica la forma en que se representa un campo en las plantillas de Drupal. El formateador utilizado por cada campo se puede configurar en la pestaña de *visualización Administrar* asociada con el tipo de entidad que está configurando.

Los campos pueden tener diferentes formateadores dependiendo del modo de visualización que se muestra, lo que le permite controlar cómo se representa un campo en diferentes partes de su sitio web.

Observaciones

Algunas cosas son importantes a considerar cuando se implementa un formateador de campos.

La implementación de su formateador debe estar dentro de su módulo en la carpeta `src/Plugin/Field/FieldFormatter` . Las anotaciones también son críticas, ya que identifican su módulo y los tipos de campo a los que se aplica.

En este ejemplo, este formateador solo es aplicable a los campos del tipo de `email` . Puede aplicar su formateador a varios campos si es necesario. Si su formateador, por cualquier motivo, sería aplicable a los campos de correo electrónico y fecha:

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

Un error que enfrenté cuando implementé los formateadores de campo por primera vez es que los ajustes no se guardaron cuando se modificaron. No existe un método de guardado explícito y la solución es implementar el método `defaultSettings()` y especificar los nombres de los campos que conforman su formulario de configuración. Tampoco olvide establecer el valor `#default_value` en el método `settingsForm` .

Si usted quiere tener una plantilla RAMITA específica para su formateador es tan simple como la configuración de una `#theme` tecla mientras la construcción de la matriz en el render `viewElements` método a continuación, en su `.module` archivo aplicar `hook_theme`

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

Luego cree la carpeta de `templates` en la raíz de su módulo y tenga un archivo llamado `obfuscator-field-formatter.twig.html` en el que imprima el marcado que necesita. En este ejemplo, las variables del render `#title` y `#url` estarán disponibles.

Examples

Ofuscador formateador de correo electrónico

En nuestro ejemplo, crearemos un formateador personalizado para las direcciones de correo electrónico que nos permitirá mostrar direcciones de correo electrónico confusas para engañar a esos desagradables spammers.

El formateador tendrá algunas opciones de configuración que nos permitirán controlar cómo se confunde la dirección de correo electrónico:

- Eliminar @ y. y reemplazarlos con un espacio,
- Reemplace @ por en y. por punto,

Tenga en cuenta que este es solo un ejemplo académico para mostrar cómo se muestran y configuran los formateadores de campo y, obviamente, no es muy útil para el antispam real.

Este ejemplo asume que usted tiene un módulo llamado `obfuscator_field_formatter` configurado y activado correctamente.

```
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;
  }
}
```

Lea Formateador de campo en línea: https://riptutorial.com/es/drupal/topic/8792/formateador-de-campo

Capítulo 11: Ramita

Introducción

Twig es el motor de plantillas que forma parte de Drupal 8. En Drupal 8, los archivos de Twig tienen la extensión `.html.twig` y se usan en todos los aspectos de la temática de Drupal. Las entidades, los campos y las vistas se pueden representar utilizando archivos `.html.twig` .

En este tema, el objetivo es tener un libro de cocina sobre cómo trabajar con Twig en el contexto de Drupal. Si desea obtener más información sobre la sintaxis o las funciones disponibles, consulte la documentación .

Examples

Filtro de ramita

Contrariamente a Drupal 7, no puede llamar a las funciones regulares de PHP en sus plantillas. En Drupal 8, el camino a seguir es crear filtros y funciones.

Debe usar un **filtro** cuando: desea transformar los datos que desea mostrar. Imagina que tienes un título que siempre quieres que esté en mayúsculas. Por ejemplo, twig tiene el filtro de `capitalize` por defecto que le permite transformar cualquier texto en su equivalente en mayúsculas.

Para este ejemplo, crearemos un filtro que nos permitirá barajar una cadena. La forma de crear filtros y funciones es exactamente la misma que la Twig normal .

La principal diferencia entre Twig regular y Drupal 8 Twig es que en Drupal 8 debe crear una definición de servicio de la clase que está creando y la clase también debe pertenecer a un espacio de nombres, de lo contrario no se registrará como un filtro Twig dentro del entorno Drupal. .

Este ejemplo asume que tienes un módulo llamado `twig_shuffle_extension` .

Esta será la definición básica del servicio inn `twig_shuffle_extension.services.yml`

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

La clave de `tags` también es absolutamente necesaria y es lo que le dice a Drupal qué se supone que debe hacer esta clase (es decir, registrarla como una extensión de Twig).

Y ahora el código fuente que se debe colocar en la ruta definida en la clave de `class` de la definición del servicio.

```
// 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;
  }
}
```

Borre sus cachés y ahora, si todo va según lo planeado, puede usar el filtro en sus plantillas.

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

Inyección De Dependencia En Extensiones De Ramita

Este ejemplo le mostrará cómo usar Dependency Inject para usar otros servicios registrados en el entorno Drupal.

Imagina que tienes un archivo de imagen SVG que cambia de color dependiendo de algo aleatorio de CSS / Javascript en tu proyecto. Para poder apuntar al SVG con CSS, debe tener el archivo SVG en el DOM. Así que creas un archivo SVG base sin ningún color y lo colocas en tu carpeta de temas.

Por supuesto, puedes pegar el contenido del archivo en la plantilla de Twig, pero eso no estaría bien. Puede crear una extensión Twig, pero tampoco desea codificar la ruta de su tema en el código fuente de la extensión.

Esto significa que tenemos que obtener el camino dinámicamente. Tienes dos opciones:

1. Use el equivalente a una variable global llamando a `\Drupal::theme()->getActiveTheme()->getPath();`
2. Inyecte el `ThemeManager` (dado por `\Drupal::theme()`) en su clase de extensión

En este ejemplo, tomaremos el segundo ejemplo porque puede ser ampliamente aplicable a cualquier servicio (usted importa la Solicitud o la Conexión de la Base de Datos si lo desea).

Esto supone que tiene un módulo llamado `twig_svg_extension` y un archivo `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 }
```

Por favor, no la clave de `arguments` que le indica a Drupal el servicio a inyectar.

```
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';
  }
}
```

Tenga en cuenta el constructor que contiene la dependencia que inyectamos en la configuración del servicio, así como el `svgFilter` que obtiene la ruta del tema activo actual.

`$filepath` debe ser una ruta relativa a su carpeta de temas. La extensión convertirá la ruta del archivo en el contenido del archivo al que apunta.

Lea Ramita en línea: https://riptutorial.com/es/drupal/topic/8804/ramita

Creditos

S. No	Capítulos	Contributors
1	Empezando con drupal	acrosman, Adrian Cid Almaguer, chx, Community, dobeerman, Jimmy Ko, Karl Buys, kiamlaluno, Mika A., Morsok, pal4life, Pierre Buyle, Sidney Gijzen
2	Desarrollo de módulos - Drupal 7	Ajit S, doublejosh, Hermann Döppes, Jimmy Ko, Jimmyb_1991, Morsok, Pierre Buyle
3	Desarrollo del tema - Drupal 7	Adrian Cid Almaguer, Sam Thompson
4	Drupal 8 Entity API	Bernard Nandwa
5	Drupal caché y performace	Anurag
6	Drush	Adrian Cid Almaguer, darc1n, Jimmy Ko, lastYorsh, L-four, Mark Conroy, Morsok, pbonnefoi, Rishi Kulshreshtha, Sjoerd van der Vis, Wade Burelbach
7	Ejemplo para Drupal 8 Queue API y Batch API	David Czinege
8	El modulo de reglas	Pierre.Vriens
9	El módulo de Vistas	Pierre.Vriens
10	Formateador de campo	Ricardo Velhote
11	Ramita	Ricardo Velhote