EBook Gratis

APRENDIZAJE magento

Free unaffiliated eBook created from
Stack Overflow contributors.

#magento

Tabla de contenido

Acerca de 1

Capítulo 1: Empezando con Magento 2

Observaciones 2

Versiones 2

Edición comunitaria 2

Edición de Empresa 3

Examples 3

Instalación y configuración 3

Requisitos previos y requisitos para Magento Community Edition 1.9 3

Instalación: 4

Solución de problemas comunes 5

Capítulo 2: Agregue un precio diferente para la tienda múltiple usando Magento SOAP API 6

Examples 6

Magento SOAP V1 6

Capítulo 3: Atributos personalizados 7

Introducción 7

Examples 7

Atributo de ventas 7

Capítulo 4: Ayudantes 8

Examples 8

Creando un ayudante 8

Capítulo 5: Colección completa de productos 9

Introducción 9

Examples 9

Filtrado de recogida de productos 9

Capítulo 6: Colecciones 11

Examples 11

Obtener colecciones de modelos 11

Obtener atributos de objeto adicionales 11

Filtración 11
Clasificación 11
Acceso 11
Obtener objeto de colección 12
Capítulo 7: Colecciones 13
Examples 13
Colección de productos 13
Categoría Colección de una tienda específica y nivel específico 13
Capítulo 8: Colecciones 15
Examples 15
Colección de productos 15
Capítulo 9: Cómo filtrar colecciones 16
Parámetros 16
Observaciones 16
Examples 17
Filtrando colecciones 17
Manejo de ANDs y ORs en Filtros 17
Capítulo 10: Crear tarjetas de regalo empresariales mediante programación 19
Examples 19
Crear una sola tarjeta de regalo 19
Capítulo 11: Datos de tienda y sitio web 20
Introducción 20
Examples 20
Obtener datos actuales de la tienda 20
Capítulo 12: EAV (valor de atributo de entidad) 21
Observaciones 21
Examples 26
Implementar la interfaz de atributos de frontend, fuente y backend 26
Capítulo 13: Entender los tipos de productos 28
Observaciones 28
Examples 31

Mage_Catalog_Model_Product_Type 31
Describir los tipos de productos estándar (simples, configurables, agrupados) 35
Capítulo 14: Estructura del módulo 38
Observaciones 38
Examples 38
Creando un módulo desde cero (Hello World) 38
Capítulo 15: Estructura MVC 42
Observaciones 42
Examples 42
Entender MVC en Magento 42
MVC Flow en Magento 43
Capítulo 16: Hoja de trucos para tareas rápidas 45
Observaciones 45
Examples 45
Obtener producto Cantidad en stock 45
Capítulo 17: Imagen del producto Urls 46
Introducción 46
Examples 46
Imágenes en caché de URL 46
Urls de imágenes no almacenadas en caché desde medios 46
Capítulo 18: Iniciar sesión para archivar 47
Sintaxis 47
Parámetros 47
Observaciones 47
El registro está desactivado de forma predeterminada a menos que el modo desarrollador est 47
Todas las excepciones se registran en exceptions.log sin importar si el registro está habi 47
Tipo de variable de mensaje 47
Nivel de registro 48
Examples 48
Archivo de registro personalizado 48
Registro predeterminado 48

Capítulo 19: Magento Caching 50
Examples 50
Cómo almacenar en caché datos personalizados en Magento 50
Limpiar caché por ID de caché 50
Usa Redis como un backend de caché 50
Capítulo 20: Manejo de errores magento, mensajes e informes. 52
Observaciones 52
Ubicaciones de registro de errores 52
/var/log/ 52
/var/report/ 53
Examples 53
Habilitar la visualización de informes de errores 53
Capítulo 21: Modelo 55
Examples 55
Modelo de carga 55
Crear un modelo vacío 55
Capítulo 22: Obtener el nombre de la categoría de la página del producto 56
Examples 56
Obtener la categoría principal 56
Obtener la categoría actual 56
Capítulo 23: Obtener productos de la base de datos 57
Examples 57
Obtener producto por sku 57
Obtener producto por ID 57
Colección de productos - consulta LIKE 57
Obtener colección de productos por atributo 57
Obtener datos de objeto de producto 57
Obtener datos del formulario de recogida de productos 57
Colección de productos - con atributos 58
Compruebe si el producto se cargó correctamente 58
Obtener identificación del producto por SKU 58
Obtener colección de productos de una lista de SKUs 58

Establecer límite en la colección de productos ... 58

Capítulo 24: Obtener usuario actual ... 60

Examples ... 60

Obtener usuario administrador actual ... 60

Obtener cliente actual ... 60

Compruebe si el usuario está conectado ... 60

Capítulo 25: Obtenga el nombre de la tienda y otros detalles de la configuración del siste ... 61

Examples ... 61

Obtenga el nombre del frontend para la vista de tienda actual ... 61

Obtener ID de tienda actual ... 61

Obtener código de tienda actual ... 61

Determine si la vista de tienda está habilitada ... 61

Obtener ID de sitio web para la tienda actual ... 61

Obtener el modelo de tienda actual ... 61

Obtener el nombre del grupo para la tienda ... 61

Obtener todas las tiendas de Magento ... 61

Capítulo 26: Obteniendo URLs de Magento ... 63

Sintaxis ... 63

Parámetros ... 63

Observaciones ... 63

Examples ... 63

En la interfaz actual / tema ... 63

Obtener url de piel ... 63

Obtener Base Url ... 63

Asegurar la url de la piel ... 63

Obtener url de medios ... 63

URL de piel insegura ... 63

Obtener url de tienda ... 64

Obtener js url ... 64

Obtener la URL actual ... 64

Capítulo 27: Optimizando Magento Para Velocidad ... 65

Examples ... 65

Optimizando Magento cambiando el archivo .htaccess 65
habilitar la compresión de archivos servidos de apache 65
http://developer.yahoo.com/performance/rules.html#gzip 65
Habilitación de encabezados caducidad 66
Configuración de administrador 66
Habilitar catálogos planos 66
Habilitar caché del sistema 67
Capítulo 28: Pedidos 68
Examples 68
Obtener orden por ID 68
Obtener pedido por ID de incremento 68
Añadir comentario al historial de pedidos 68
Capítulo 29: Representación 69
Observaciones 69
Examples 71
Diferentes mecanismos para desactivar la salida de bloque 71
diferentes tipos de bloques 71
Se puede acceder a las instancias de bloque desde el controlador 72
Capítulo 30: Shell, CLI 73
Observaciones 73
Lo esencial 73
Métodos de shell del núcleo por archivos 73
Shell PHP personalizado scripts 73
Examples 73
Usando shell sin extender Mage_Shell_Abstract 73
Bootstrapping Magento llamando: 73
Usando shell de la manera Magento - extienda Mage_Shell_Abstract 74
Camino magento 74
Realizando Reindex de CLI 75
Capítulo 31: Sql script para eliminar datos de prueba 76
Introducción 76

Examples 76
Eliminar datos de prueba del cliente 76
Eliminar datos de prueba del producto 77
Eliminar datos de prueba de ventas 78
Eliminar registros de datos de prueba 79
Capítulo 32: Url actual 81
Sintaxis 81
Examples 81
Página principal 81
Página del producto 81
Compruebe si la URL actual es segura 81
Capítulo 33: URLs específicas 82
Examples 82
Carro url 82
URL de salida 82
URL de inicio de sesión 82
URL de cierre de sesión 82
Olvidé la contraseña url 82
Cuenta del cliente url 82
Media, JS, URL de la piel 83
Creditos 85

Acerca de

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

It is an unofficial and free magento 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 magento.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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 Magento

Observaciones

Magento es una plataforma de comercio electrónico de código abierto escrita en PHP; una plataforma de comercio electrónico altamente personalizable y un sistema de gestión de contenido que se puede utilizar para crear tiendas en línea para vender mercancías.

Proporciona funciones comunes de comercio electrónico, como carritos de compras y gestión de inventario, y fomenta una amplia personalización para cumplir con los objetivos específicos de la organización.

Magento también es un marco PHP orientado a objetos que se puede usar para desarrollar aplicaciones web modernas y dinámicas que aprovechan las características de comercio electrónico de Magento.

Las principales características de la plataforma Magento son:

- capacidad de expansión
- escalabilidad
- flexibilidad
- personalización
- fuente abierta

Versiones

Edición comunitaria

Versión	Fecha de lanzamiento
1.9	2014-05-14
1.8	2013-12-11
1.7	2012-04-24
1.6	2011-08-08
1.5	2011-02-08
1.4	2010-02-12
1.3	2009-03-30
1.2	2008-12-29

Versión	Fecha de lanzamiento
1.0	2008-03-31

Edición de Empresa

Versión	Fecha de lanzamiento
1.14	2014-05-14
1.13	2013-10-11
1.12	2012-04-24
1.11	2011-08-08
1.10	2011-02-08
1.9	2010-07-19
1.8	2010-04-14
1.7	2010-01-19
1.6	2009-10-20
1.3	2009-04-15

Examples

Instalación y configuración

Requisitos previos y requisitos para Magento Community Edition 1.9

Alojamiento

- Apache 2.x (con mod_rewrite) o Nginx 1.7.x
- Debido a las demandas de procesamiento de las operaciones de Magento, se recomienda instalar Magento en un servidor con al menos 2 GB de RAM. Esto asegurará que todo el software involucrado en la administración de la tienda tenga suficiente memoria para funcionar.
- Capacidad para ejecutar trabajos programados (crontab) con PHP 5.

- Posibilidad de anular opciones en archivos .htaccess.

PHP

- PHP 5.4, PHP 5.5
- Extensiones requeridas: PDO_MySQL, simplexml, mcrypt, hash, GD, DOM, iconv, curl, SOAP (para API de servicios web)
- límite de memoria no menos de 256 MB (se recomiendan 512 MB)

Base de datos

- MySQL 5.6 (Oracle, Percona, MariaDB)

SSL

- Se requiere un certificado de seguridad válido para HTTPS.
- Los certificados SSL autofirmados no son compatibles

Instalación:

Descargar y configurar archivos Magento

Estamos usando OpenMage Mirror ya que la descarga directa para la sucursal 1.9.2.4 está deshabilitada y el sitio web de Magento requiere una cuenta. Pero le recomendamos que descargue una copia desde https://www.magentocommerce.com/download

```
cd /var/www/html
wget https://github.com/OpenMage/magento-mirror/archive/magento-1.9.zip
unzip magento-1.9.zip
rm magento-1.9.zip
rsync -avP magento-mirror-magento-1.9/. .
rm magento-mirror-magento-1.9 -r
sudo chown -R www-data:www-data /var/www/html/
chmod -R 0777 media var
```

Crear una base de datos MySQL y un usuario

acceder a la consola mysql

```
mysql -u root -p
```

en la consola mysql

```
CREATE DATABASE magento;
CREATE USER magento_db_user@localhost IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON magento.* TO magento_db_user@localhost IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
exit
```

Completa la instalación a través de la interfaz web.

Para acceder a la interfaz web con su navegador, navegue hasta el nombre de dominio o la dirección IP pública de su servidor:

```
http://domain_name/
```

Luego sigue las instrucciones en pantalla.

Solución de problemas comunes

Solo funciona la página de inicio, todas las demás páginas devuelven 404

Asegúrese de que el módulo `mod_rewrite` se haya instalado en Apache y se haya habilitado para cargar. Vea el **paso 2** para obtener información sobre cómo hacer esto aquí: https://www.digitalocean.com/community/tutorials/how-to-set-up-mod_rewrite-for-apache-on-ubuntu-14-04

Asegúrese de que permite cambios en el archivo .htaccess habilitándolo en su sitio conf. Consulte el **paso 3** : https://www.digitalocean.com/community/tutorials/how-to-set-up-mod_rewrite-for-apache-on-ubuntu-14-04

Su archivo `.htaccess` puede estar mal configurado o faltar: diríjase a la página de descarga de Magento: https://www.magentocommerce.com/download - descargue la versión correspondiente y extraiga el archivo .htaccess para colocarlo en su raíz de instalación de Magento.

El sitio funciona pero no se están cargando estilos o scripts

Asegúrese de haber configurado los permisos y la propiedad relevantes: **consulte aquí** para obtener más información: http://devdocs.magento.com/guides/m1x/install/installer-privileges_before.html

Solución común: intente reindexar y vaciar el caché manualmente (en caso de que el administrador sea demasiado difícil de navegar). Reindexar a través de la línea de comandos: https://www.atwix.com/magento/process-magento-indexes-from-command-line/ Flush cache (a través de admin o línea de comandos): https://www.properhost.com/ support / kb / 23 / How-to-Clear-The-Magento-Cache

Seguido

Lea Empezando con Magento en línea: https://riptutorial.com/es/magento/topic/812/empezando-con-magento

Capítulo 2: Agregue un precio diferente para la tienda múltiple usando Magento SOAP API

Examples

Magento SOAP V1

Debe cambiar el alcance del precio **'Global' al 'sitio web'** (Sysytem-> Configuration-> Catalog-> Catalog-> Price)

```
$client = new SoapClient('http://your-web-site/api/soap/?wsdl');
$API_USER = 'your-api-user';
$API_KEY = 'your-api-key';
$session = $client->login($API_USER, $API_KEY);
$result = $client->call($session, 'catalog_product.update', array('test-product',
array('price' => '100'),'your-store-code'));
print "<pre>";
print_r($result);
print "</pre>";
```

Lea Agregue un precio diferente para la tienda múltiple usando Magento SOAP API en línea: https://riptutorial.com/es/magento/topic/7410/agregue-un-precio-diferente-para-la-tienda-multiple-usando-magento-soap-api

Capítulo 3: Atributos personalizados

Introducción

Atributos personalizados para ventas, categoría, etc.

Examples

Atributo de ventas

Atributo personalizado en tablas relacionadas con las ventas como: sales_flat_quote, sales_flat_order_item, sales_flat_order, etc table

En su archivo de instalación sql / some_setup / mysql-install-0.1.0.php:

```
<?php
 $installer = $this;
 $installer->startSetup();
 $installer->addAttribute('quote', 'custom_field', array('type' => 'varchar'));
 $installer->addAttribute('order', 'custom_field', array('type' => 'varchar'));
 $installer->endSetup();
 ?>
```

Otra forma de hacerlo es:

```
<?php
$installer = $this;
$installer->startSetup();
$installer->run("ALTER TABLE sales_flat_order_item ADD COLUMN 'custom_field' DECIMAL(12,4)
NULL;");
$installer->endSetup();
?>
```

Asegúrese de borrar la caché después de esto.

Lea Atributos personalizados en línea: https://riptutorial.com/es/magento/topic/9267/atributos-personalizados

Capítulo 4: Ayudantes

Examples

Creando un ayudante

Los ayudantes deben extenderse desde `Mage_Core_Helper_Abstract` :

```
# File: app/code/local/Vendor/Package/Helper/Data.php
class Vendor_Package_Helper_Data extends Mage_Core_Helper_Abstract
{
    public function multiply($a, $b)
    {
        return $a * $b;
    }
}
```

Para poder acceder es a través de `Mage::helper` , debe definir un alias de ayuda en un archivo `config.xml` para permitir que el autocargador Magento encuentre su clase:

```
<!-- File: app/code/local/Vendor/Package/etc/config.xml -->
<global>
    <helpers>
        <alias_here>
            <class>Vendor_Package_Helper</class>
        </alias_here>
    </helpers>
</global>
```

Suponiendo que su módulo esté configurado correctamente y que haya borrado su caché, ahora debería poder usar su ayudante así:

```
$result = Mage::helper('alias_here')->multiply(2, 4); // int(8)
```

Nota: si está utilizando una clase de datos, su nombre de ayudante está implícito si no especifica uno. Por ejemplo, los siguientes dos ejemplos son idénticos:

```
Mage::helper('alias_here');
Mage::helper('alias_here/data');
```

Lea Ayudantes en línea: https://riptutorial.com/es/magento/topic/5904/ayudantes

Capítulo 5: Colección completa de productos

Introducción

Filtros completos para la recolección de productos en función de la visibilidad, la tienda y el OR, el estado del stock, el estado, etc.

Examples

Filtrado de recogida de productos.

```
$model = Mage::getModel('catalog/product')->getCollection()
```

Filtro basado en tienda:

```
$mode->addStoreFilter($storeId)
```

Filtro basado en tipo de producto:

```
$mode->addAttributeToFilter('type_id', 'configurable')
$mode->addAttributeToFilter('type_id', 'simple')
```

Filtro basado en estado:

```
$model->addAttributeToFilter('status',Mage_Catalog_Model_Product_Status::STATUS_DISABLED)
$model->addAttributeToFilter('status',Mage_Catalog_Model_Product_Status::STATUS_ENABLED)
```

Filtrar utilizando nulo y no nulo:

```
$model->addAttributeToFilter('short_description', array('null' => true))
$model->addAttributeToFilter('short_description', array('notnull' => true))
```

Filtro utilizando mayor que y menor que:

```
$model->addAttributeToFilter('entity_id', array('gt' => 64230))
$model->addAttributeToFilter('entity_id', array('lt' => 64230))
```

Filtro utilizando mayor que e igual a:

```
$model->addAttributeToFilter('entity_id', array('gteq' => 64230))
```

Filtrar utilizando menos que e igual a:

```
$model->addAttributeToFilter('entity_id', array('lteq' => 64230))
```

Filtrar utilizando en y no en:

```
$model->addAttributeToFilter('entity_id', array('in' => array(1,4,64231)))
$model->addAttributeToFilter('entity_id', array('nin' => array(1,4,64231)))
```

Filtrar productos por un rango de ID de entidad:

```
$model->addAttributeToFilter('entity_id', array(
             'from' => 64229,
             'to' => 64231
             ))
```

Filtro basado en la visibilidad del producto:

```
$model->addAttributeToFilter('visibility', 4) //catalog,search
$model->addAttributeToFilter('visibility', 3) //catalog
$model->addAttributeToFilter('visibility', 2) //search
$model->addAttributeToFilter('visibility', 1) //not visible individually
```

Filtrar utilizando como y no como

```
$model->addAttributeToFilter('sku', array('nlike' => '5713%'))
$model->addAttributeToFilter('sku', array('like' => '%shirt%'))
```

Filtro usando igual y no igual a:

```
$model->addAttributeToFilter('sku', array('neq' => 'shirt'))
$model->addAttributeToFilter('sku', array('eq' => 'shirt'))
```

Filtrar en stock productos:

```
$model->joinField('is_in_stock',
                'cataloginventory/stock_item',
                'is_in_stock',
                'product_id=entity_id',
                'is_in_stock=1', //make this 0 for out of stock products
                '{{table}}.stock_id=1',
                'left')
```

Establecer orden por:

```
$model->setOrder('entity_id','desc')
```

Establecer tamaño de página:

```
$model->setPageSize(100)
```

Lea Colección completa de productos en línea:
https://riptutorial.com/es/magento/topic/9261/coleccion-completa-de-productos

Capítulo 6: Colecciones

Examples

Obtener colecciones de modelos

```
// get existing collections
$orders = Mage::getModel('sales/order')->getCollection();
$products = Mage::getModel('catalog/product')->getCollection();
$customers = Mage::getModel('customer/customer')->getCollection();
```

Obtener atributos de objeto adicionales

```
// $orders is collection
$orders->addAttributeToSelect('status'); // get status attribute
$orders->addAttributeToSelect('*'); // get all attributes
```

Filtración

```
// filter by creation date
$date = new Zend_Date();
$toDate = $date->get(Zend_Date::W3C); // today
$fromDate = $date->sub('1', Zend_Date::MONTH)->get(Zend_Date::W3C); // one month ago

$orders->addAttributeToFilter('created_at', array('from' => $fromDate, 'to' => $toDate));

// more filtering, type AND
$orders->addAttributeToFilter('status', 'pending');
$orders->addAttributeToFilter('state', array('nlike' => 'new'));

// more filtering, type OR
$processing = array('eq' => 'processing');
$complete = array('like' => 'complete');
$orders->addAttributeToFilter('status', array($processing, $complete));
```

Clasificación

```
// sort by creation date
$orders->setOrder('created_at', 'asc');
$orders->setOrder('created_at', 'desc');
$orders->setOrder('created_at'); // default direction is 'desc'
```

Acceso

```
// iterating over items in collection
foreach ($orders as $singleOrder) {
    // do something with individual objects
    var_dump($singleOrder->getData());
}
```

```
// get first/last item in collection
$first = $orders->getFirstItem();
$last = $orders->getLastItem();
```

Obtener objeto de colección

```
    // get product collection object
    $productCollection = Mage::getResourceModel('catalog/product_collection');
    // get customer collection object
    $customerCollection = Mage::getResourceModel('customer/customer_collection');
    // get order collection object
    $orderCollection = Mage::getResourceModel('sales/order_collection');
```

Lea Colecciones en línea: https://riptutorial.com/es/magento/topic/5937/colecciones

Capítulo 7: Colecciones

Examples

Colección de productos

```
$ProductCollection=Mage::getModel('catalog/product')->getCollection();
```

Seleccionando el Atributo específico

```
$ProductCollection->addAttributeToSelect(array('name', 'product_url', 'small_image'));
```

Seleccionando el atributo All

```
$ProductCollection->addAttributeToSelect('*');
```

Añadir filtro en la colección

```
$ProductCollection->addFieldToFilter('is_active',1);
```

Añadir filtro de atributos del producto en la colección

```
$ProductCollection->addAttributeToFilter('weight', array('gt' => 100));
```

Establecer orden

```
$ProductCollection->setOrder('id','ASC');
```

Establecer limite

```
$ProductCollection->setPageSize(10);
```

Establecer página actual

```
$ProductCollection->setCurPage(1);
```

Categoría Colección de una tienda específica y nivel específico.

```
$rootId     = Mage::app()->getStore($storeId)->getRootCategoryId();
$categories = Mage::getModel('catalog/category')->getCollection()
    ->addAttributeToSelect('*')
    ->addFieldToFilter('path', array('like'=> "1/$rootId/%"))
    ->addAttributeToFilter('level', 2)
    ->addAttributeToFilter('is_active', 1);
```

Lea Colecciones en línea: https://riptutorial.com/es/magento/topic/6763/colecciones

Capítulo 8: Colecciones

Examples

Colección de productos

```
$productCollection = Mage::getModel('catalog/product')->getCollection();
```

Seleccionando el Atributo específico

```
$productCollection->addAttributeToSelect(array('name', 'product_url', 'small_image'));
```

Seleccionando todos los atributos

```
$productCollection->addAttributeToSelect('*');
```

Añadir filtro en la colección

```
$productCollection->addFieldToFilter('is_active', 1);
```

Establecer orden

```
$productCollection->setOrder('id', 'ASC');
```

Establecer limite

```
$productCollection->setPageSize(10);
```

Establecer página actual

```
$productCollection->setCurPage($page);
```

Lea Colecciones en línea: https://riptutorial.com/es/magento/topic/7206/colecciones

Capítulo 9: Cómo filtrar colecciones

Parámetros

Parámetro	Detalles
$ addFieldToFilter (**$ field** , $ condition = null)	{string} El campo que estamos agregando al filtro.
$ addFieldToFilter ($ field, **$ condition = null**)	{mixed} La definición del filtro que usaremos.
addAttributeToFilter (**$ attr** , $ condition = null, $ join = 'inner')	{string} El campo que estamos agregando al filtro.
addAttributeToFilter ($ attr, **$ condition = null** , $ join = 'inner')	{mixed} La definición del filtro que usaremos.
addAttributeToFilter ($ attr, $ condition = null, **$ join = 'inner'**)	{('inner', 'left')} El tipo de unión sql que se usará al unirse a la tabla EAV.

Observaciones

Argumentos de comparación de filtros

Magento también ofrece una forma flexible de filtrar utilizando operadores de comparación también. Aquí hay una lista de operadores válidos y su sintaxis:

Todos los argumentos de comparación se pueden pasar al segundo parámetro de los `addFieldToFielter()` o `addAttributeToFilter()` .

```
$collection_of_products->addAttributeToFilter('visible',array("eq"=>1));
```

Comparación	Matriz de argumentos	Fragmento de SQL resultante
Es igual a	array ("eq" => $ var)	DONDE (`my_field` = $ var)
No es igual	array ("neq" => $ var)	DONDE (`my_field`! = $ Var)
Me gusta	array ("like" => $ var)	DONDE (`my_field` ME GUSTA $ var)
Diferente a	array ("nlike" => $ var)	DONDE (`my_field` NO ME GUSTA $ var)
Es	array ("is" => $ var)	DONDE (`my_field` IS $ var)

Comparación	Matriz de argumentos	Fragmento de SQL resultante
En	array ("in" => $ var)	DONDE (`my_field` IN ($ var))
No en	array ("nin" => $ var)	DONDE (`my_field` NO EN ($ var))
Nulo	array ("null" => true)	DONDE (`my_field` ES NULL)
No nulo	array ("notnull" => true)	DONDE (`my_field` NO ES NULO)
Mas grande que	array ("gt" => $ var)	DONDE (`my_field`> $ var)
Menos que	array ("lt" => $ var)	DONDE (`my_field` <$ var)
Mayor que o igual	array ("gteq" => $ var)	DONDE (`my_field`> = $ var)
Menor o igual	array ("lteq" => $ var)	DONDE (`my_field` <= $ var)
Encontrar en set	array ("finset" => array ($ var))	DONDE (find_in_set ($ var, `my_field`)
Desde y hacia	array ("from" => $ var1, "to" => $ var2)	DONDE (`my_field`> = $ var1 Y` my_field` <= $ var2)

Examples

Filtrando colecciones

Magento tiene un poderoso conjunto de métodos para filtrar colecciones. Como hay dos tipos de Objetos que pueden estar contenidos en colecciones, primero debemos determinar con qué tipo de datos estamos trabajando antes de poder filtrarlos. Magento implementa un modelo de datos EAV para entidades como productos y categorías. Existe un conjunto diferente de métodos a utilizar si estamos filtrando una colección de Objetos EAV.

En Magento, los pedidos no se almacenan como objetos EAV. Esto hace que la colección de pedidos sea un buen ejemplo para filtrar una colección básica.

```
$collection_of_orders = Mage::getModel('sales/order')->getCollection();
$collection_of_orders->addFieldToFilter('status','processing');
```

Si observamos la colección de productos, podemos ver que los productos están almacenados en un modelo de datos EAV. También podemos filtrar fácilmente por atributos de EAV.

```
$collection_of_products = Mage::getModel('catalog/product')->getCollection();
$collection_of_products->addAttributeToFilter('visible',1);
```

Manejo de ANDs y ORs en Filtros

Cuando consultamos nuestros datos, a menudo necesitamos más de un filtro para obtener el conjunto de datos exacto que estamos buscando. En SQL, manejamos esto con cláusulas AND y OR. Podemos lograr lo mismo con las colecciones.

Para agregar una cláusula AND a su consulta, simplemente agregue otro método de llamada. Esto agregará el segundo filtro a la declaración WHERE original que lo une con un AND.

```
Mage::getModel('catalog/product')->getCollection()
        ->addFieldToFilter('sku',array('like'=>'a%'))
        ->addFieldToFilter('sku',array('like'=>'%b'));
```

La cláusula WHERE resultante se verá así:

```
WHERE (e.sku like 'a%') AND (e.sku like '%b')
```

Ahora digamos que queremos que todos los skus que comiencen con 'a' O terminen con 'b'. ¿Cómo agregamos una cláusula OR? Gracias a las colecciones de Magento, es bastante sencillo. Añadimos el filtro como un segundo elemento en la matriz de filtros.

```
Mage::getModel('catalog/product')->getCollection()
        ->addFieldToFilter('sku', array(
            array('like'=>'a%'),
            array('like'=>'%b')
        ));
```

Ahora, la cláusula WHERE resultante se verá así:

```
WHERE (((e.sku like 'a%') or (e.sku like '%b')))
```

Lea Cómo filtrar colecciones en línea: https://riptutorial.com/es/magento/topic/5849/como-filtrar-colecciones

Capítulo 10: Crear tarjetas de regalo empresariales mediante programación

Examples

Crear una sola tarjeta de regalo

```
// Instantiate Gift Card Model
$gift_card = Mage::getModel('enterprise_giftcardaccount/giftcardaccount');

// Populate Gift Card values
$gift_card
    // Your redeemable code, doesn't have to be in this format.
    ->setCode('2i2j2j-24k1ii1-67774k-2311')
    // Also has STATUS_DISABLED
    ->setStatus($gift_card::STATUS_ENABLED)
    // YYYY-MM-DD format date
    ->setDateExpires('2015-04-15')
    ->setWebsiteId(1)
    // Also has STATE_USED, STATE_REDEEMED, and STATE_EXPIRED
    ->setState($gift_card::STATE_AVAILABLE)
    // Also has NOT_REDEEMABLE value.
    ->setIsRedeemable($gift_card::REDEEMABLE)
    // Int or float (or String that can be parsed into either) currency amount.
    ->setBalance(25);

// Save the fleshed out gift card.
$gift_card->save();

// Can use the gift card for further use, return it, or return it's ID
return $gift_card // ->getId();
```

Lea Crear tarjetas de regalo empresariales mediante programación en línea: https://riptutorial.com/es/magento/topic/2387/crear-tarjetas-de-regalo-empresariales-mediante-programacion

Capítulo 11: Datos de tienda y sitio web

Introducción

Obtener magento store y datos relacionados con el sitio web

Examples

Obtener datos actuales de la tienda

```
$store = Mage::app()->getStore();
$storeId = Mage::app()->getStore()->getStoreId();
$storeCode = Mage::app()->getStore()->getCode();
$websiteId = Mage::app()->getStore()->getWebsiteId();
$storeGroupId = Mage::app()->getStore()->getGroupId();
$storeName = Mage::app()->getStore()->getName();
$storeSortOrder = Mage::app()->getStore()->getSortOrder();
$storeIsActive = Mage::app()->getStore()->getIsActive();
```

Lea Datos de tienda y sitio web en línea: https://riptutorial.com/es/magento/topic/9266/datos-de-tienda-y-sitio-web

Capítulo 12: EAV (valor de atributo de entidad)

Observaciones

Entidad

Almacena información sobre el tipo de datos que se almacenan. En el caso de Magento esto es cliente, producto, categoría, etc.

Atributo

Las propiedades individuales de cada una de las entidades, por ejemplo, nombre, peso, dirección de correo electrónico, etc.

Valor

El valor de una entidad y atributo dados. Por ejemplo, podemos especificar la entidad del cliente y el atributo de correo electrónico y luego asignarle el valor hello@example.com.

Esquema de base de datos

eav_entity

La tabla de entidades.

eav_entity_attribute

La tabla de atributos.

eav_entity_ {type}

Las tablas de valores. Los tipos son datetime, decimals, int, text y varchar.

Es importante tener en cuenta que la tabla eav_entity_varchar tiene el tipo varchar para los valores, incluso si la fecha o el entero se ajustan mejor al valor.

Modelos Versus Modelos de Recursos

Todos los modelos dentro de Mage / Eav / Model / Resource son Mysql4 y son modelos de recurso.

Además, Entity / Abstract.php y Entity / Setup.php.

Flat Versus EAV

Los modelos EAV son más complejos y proporcionan lógica para guardar y cargar desde múltiples tablas, mientras que los modelos planos o estándar son relativamente sencillos

(tradicionales).

Los modelos estándar gestionan principalmente sus propiedades con definidores de datos y captadores que trabajan con una sola tabla. Los modelos EAV gestionan principalmente sus modelos de atributos. Los modelos estándar solo guardan sus datos en una tabla y se cargan. Los modelos EAV cargan todos los atributos (o un conjunto específico) después de cargar los datos de base y guardan los atributos después de guardar los datos (incluyendo la inserción, actualización y eliminación de atributos).

Ejemplos de modelos de recursos de EAV

Para buscar entidades que utilizan el esquema de almacenamiento EAV, se puede utilizar la búsqueda de la cadena Mage_Eav_Model_Entity_Abstract. Esto debería revelar todos los modelos de recursos basados en la estructura de EAV. Sin embargo, la lista resultante incluirá muchas clases obsoletas del módulo Mage_Sales que ya no se están utilizando.

Los únicos módulos que contienen entidades que utilizan el esquema de almacenamiento EAV son Mage_Catalog (categorías y productos) y Mage_Customer (clientes y direcciones).

Los grupos de clientes utilizan el esquema de almacenamiento de tabla plana. Todas las entidades de ventas se convirtieron en entidades de tabla plana con el lanzamiento de Magento 1.4.

La razón por la que se utiliza EAV es para que las entidades puedan tener un número indeterminado de propiedades y, por lo tanto, seguir siendo flexibles. Por ejemplo, cuando agrega un nuevo atributo a una entidad de Cliente (que es una entidad de EAV), no es necesario modificar la tabla de la base de datos para que se agregue este nuevo atributo.

Ventajas

Flexibilidad El esquema de la base de datos no necesita cambiarse con el modelo Rápido de implementar

Desventajas

- Ineficiente

Una consulta que devuelve 20 columnas normalmente consistirá en 20 uniones individuales en EAV. Sin embargo, el módulo Mage_Eav generalmente no usa uniones para cargar los datos de valor de atributo. En su lugar, se utilizan las selecciones de unión. Las uniones solo se utilizan para filtrar colecciones de EAV.

- Ningún mecanismo para las relaciones entre subtipos.
- Sin agrupación de subtipos de entidad.

Sitio web y tienda de ámbitos

Para manejar los valores de atributo de ámbito de tienda y sitio web dentro de EAV, existe un valor de store_id en la entidad de catálogo para mostrar el alcance que enlaza de nuevo con core_store. Junto con las tiendas normales (vistas de tienda) también hay una tienda '0' que es el

valor global. Cuando se encuentra en una tienda en particular, el sistema primero verificará el valor de una entidad en la tienda actual y luego volverá a la entidad global. Las entidades de EAV de Mage_Customer no tienen una columna de ámbito store_id.

Insertar, actualizar y eliminar

Para determinar si una inserción, actualización o eliminación debe realizarse en un atributo, se realiza una comparación con el objeto original. El objeto original es básicamente un duplicado del objeto de datos cuando la entidad se recuperó de la base de datos.

- Si el atributo existe originalmente y su nuevo valor no está vacío; se actualiza.
- Si el atributo existe originalmente pero su nuevo valor se establece en vacío; se borra. - Si el atributo no existe originalmente y su nuevo valor no está vacío; se inserta.

Gestión de atributos

Modelos de atributo

Modelo de atributo *Representa el atributo en el formulario de la base de datos, su lógica es estándar en todos los atributos y es difícil de cambiar* .

Modelo frontend

La interfaz del atributo a la interfaz y proporciona cualquier lógica que el atributo requiera en la interfaz, por ejemplo, el método getUrl () en las imágenes.

Modelo de backend

Estos realizan la validación en el atributo antes de que se guarde en la base de datos. Por ejemplo, el modelo de back-end de contraseña convierte la contraseña en un hash antes de que se guarde. También verifica que la contraseña y la confirmación de la contraseña coincidan antes de guardar.

Modelos de fuente

Se utiliza para rellenar las opciones disponibles para un atributo, por ejemplo, catalog / product_status se ha habilitado y deshabilitado.

Métodos requeridos

Un modelo fuente requiere:

```
<?php
    public function getAllOptions();
    public function getOptionText($value);
?>
```

Por lo general, solo se necesita implementar getAllOptions (), ya que ya existe una implementación para getOptionText () en el modelo de fuente abstracta Mage_Eav_Model_Entity_Attribute_Source_Abstract.

Un modelo de frontend no requiere el método getValue ().

Un modelo de backend requiere:

```
<?php
    public function getTable();
    public function isStatic();
    public function getType();
    public function getEntityIdField();
    public function setValueId($valueId);
    public function getValueId();
    public function afterLoad($object);
    public function beforeSave($object);
    public function afterSave($object);
    public function beforeDelete($object);
    public function afterDelete($object);
    public function getEntityValueId($entity);
    public function setEntityValidId($entity, $valueId);
?>
```

Todos estos métodos se implementan en el modelo de backend abstracto Mage_Eav_Model_Entity_Attribute_Backend_Abstract. Para modelos de back-end personalizados, solo los métodos que requieren personalización deben ser anulados.

Modelos de fuente de configuración del sistema

No se puede utilizar para los atributos de EAV. Los modelos de origen de EAV implementan el método getAllOptions mientras que los modelos de origen adminhtml implementan el método toOptionArray ().

Los modelos de fuente de configuración del sistema predeterminados se pueden encontrar en Mage / Adminhtml / Model / System / Config / Source /.

Modelos de fuente de atributos

El propósito de los modelos de fuente de atributos es proporcionar la lista de opciones y valores para los atributos de selección y selección múltiple. También suministran la información de la columna al indexador de tabla plana de catálogo, si es necesario.

Para obtener una lista de todas las opciones para un atributo, realice lo siguiente:

```
<?php
    $options = $attribute->getSource()->getAllOptions(false);

    // or for admin
    $options = $_attribute->getSource()->getAllOptions(true, true);
?>
```

Modelos de atributos predeterminados

Si no se especifica ninguna clase como frontend, backend o - para los atributos de selección o selección múltiple - modelos de origen, se utiliza una clase predeterminada.

El modelo de frontend de atributo predeterminado es Mage_Eav_Model_Entity_Attribute_Frontend_Default.

El modelo de backend de atributo predeterminado depende del código de atributo y se determina en el método Mage_Eav_Model_Entity_Attribute :: _ getDefaultBackendModel ().

```
<?php
    protected function _getDefaultBackendModel()
    {
        switch ($this->getAttributeCode()) {
            case 'created_at':
                return 'eav/entity_attribute_backend_time_created';

            case 'updated_at':
                return 'eav/entity_attribute_backend_time_updated';

            case 'store_id':
                return 'eav/entity_attribute_backend_store';

            case 'increment_id':
                return 'eav/entity_attribute_backend_increment';
        }

        return parent::_getDefaultBackendModel();
    }
?>
```

Si el método cae hasta la última línea, se utiliza Mage_Eav_Model_Entity_Attribute_Backend_Default.

El modelo de origen predeterminado se establece en Mage_Eav_Model_Entity_Attribute_Source_Table. Esto se establece en el modelo de atributo de módulos de catálogo. El modelo de fuente de configuración predeterminado especificado en el módulo eav nunca se usa.

Añadir atributo

Para agregar atributos de EAV, use Mage_Eav_Model_Entity_Setup extendiendo en la clase de configuración.

addAttribute () Crea los atributos, los agrega a grupos y conjuntos (incluido el predeterminado), o actualiza si ya existe updateAttribute () Actualiza solo los datos del atributo. Se pueden usar clases de configuración personalizada para ampliar estos métodos, agregar datos adicionales o simplificar los argumentos necesarios.

Mesas planas

Los atributos del catálogo plano son gestionados por indexadores:

Mage_Catalog_Model_Resource_Product_Flat_Indexer :: updateAttribute () Mage_Catalog_Model_Resource_Category_Flat :: synchronize () Los atributos del producto se agregan a la tabla plana si lo son (ver Mage_Catalog_Model_Resource_Product_Flat_Inplatin_Plat_Inplanta).

Estático (tipo de backend) Filtrable Se usa en la lista de productos Se usa para reglas de promoción Se usa para ordenar por atributos del sistema Hay una tabla plana diferente para cada tienda, cada una contiene un valor de atributo de entidad de ámbito de tienda diferente. Los valores multilingües se administran teniendo diferentes tiendas para cada idioma.

Examples

Implementar la interfaz de atributos de frontend, fuente y backend.

Interfaz frontal

```
/**
* Entity attribute frontend interface
*
* Frontend is providing the user interface for the attribute
*
*/
interface Mage_Eav_Model_Entity_Attribute_Frontend_Interface
{

}
```

Interfaz de fuente

```
/**
* Entity attribute select source interface
*
* Source is providing the selection options for user interface
*
*/
interface Mage_Eav_Model_Entity_Attribute_Source_Interface
{
/**
* Retrieve All options
*
* @return array
*/
public function getAllOptions();

/**
* Retrieve Option value text
*
* @param string $value
* @return mixed
*/
public function getOptionText($value);
}
```

Interfaz de backend

```
/**
* Entity attribute backend interface
*
* Backend is responsible for saving the values of the attribute
```

```
* and performing pre and post actions
*
*/
interface Mage_Eav_Model_Entity_Attribute_Backend_Interface
{
public function getTable();
public function isStatic();
public function getType();
public function getEntityIdField();
public function setValueId($valueId);
public function getValueId();
public function afterLoad($object);
public function beforeSave($object);
public function afterSave($object);
public function beforeDelete($object);
public function afterDelete($object);

/**
* Get entity value id
*
* @param Varien_Object $entity
*/
public function getEntityValueId($entity);

/**
* Set entity value id
*
* @param Varien_Object $entity
* @param int $valueId
*/
public function setEntityValueId($entity, $valueId);
}
```

Lea EAV (valor de atributo de entidad) en línea: https://riptutorial.com/es/magento/topic/7121/eav--valor-de-atributo-de-entidad-

Capítulo 13: Entender los tipos de productos

Observaciones

Hay seis tipos diferentes de productos integrados en Magento.

- Sencillo

Una sola unidad de stock.

- Configurable

Primero de los productos compuestos. Permita que los clientes configuren su producto y agregue un solo producto simple a la cesta.

- Agrupados

El segundo producto compuesto, un producto agrupado, relaciona productos simples y brinda a los clientes la capacidad de elegir cantidades de cada artículo.

- Haz

El tercer tipo de producto compuesto, un paquete, relaciona productos simples para comprarlos como un solo artículo.

- Virtual

No se requiere ningún artículo físico para la entrega, por ejemplo, servicios

- Descargable

Un producto digital en lugar de físico. La mayoría de los tipos de productos se implementan como parte del módulo Mage_Catalog, además de Mage_Bundle y Mage_Downloadable.

Los productos agrupados, agrupados y configurables implementan una relación padre-hijo donde una serie de otros productos (de forma predeterminada, simple, virtual o descargable) se asignan a un producto principal. Esto luego maneja los datos del producto para toda la colección (por ejemplo, grupo, paquete o nombre de producto configurable, precio y estado).

Los productos descargables y de paquete tienen tablas adicionales en la base de datos, mientras que el resto se comparte con todos los demás tipos de productos. Los productos configurables tienen una tabla adicional para vincular a productos secundarios, catalog_product_super_link.

Tipo de producto personalizado

Para crear un tipo de producto que amplíe uno de los tipos de producto integrados, se debe ampliar el modelo de tipo de producto correspondiente. De lo contrario, el nuevo tipo de producto debería extender la clase Mage_Catalog_Model_Product_Type_Abstract.

También se requiere una entrada en el config.xml del módulo:

Los productos más complicados pueden requerir otras áreas personalizadas, como el modelo de precios y el recuperador de datos de índice.

Cálculo del precio

Cuando se trata de un solo producto, el precio siempre se calcula sobre la marcha. El atributo EAV de precio se carga con el producto y el precio final se calcula con el modelo de precio, Mage_Catalog_Model_Product_Type_Price.

Algunos tipos de productos lo tratan de manera diferente. En cuyo caso extienden esta clase e implementan su propia lógica. Por ejemplo, el producto configurable sobrescribe getFinalPrice () y agrega lógica adicional. Este modelo personalizado se puede especificar en config.xml con una etiqueta <price_model>.

Las colecciones de productos, sin embargo, utilizan el índice de precios para recuperar precios precalculados, eliminando la necesidad de calcularlo para cada producto.

El precio final puede ser ajustado por los observadores del evento catalog_product_get_final_price. Por defecto, solo el módulo Mage_CatalogRule observa este evento.

Otro método para anular el precio de producción es simplemente establecerlo en el producto. Si se establece el precio, el producto no lo recalculará.

El precio del nivel del producto es independiente del precio normal (aunque se tiene en cuenta al calcular el precio). Se implementa como una lista con un grupo de clientes y calificadores de cantidad mínima para cada nivel. Los precios de los niveles se muestran en una tabla, utilizando la plantilla catalog / product / view / tierprices.phtml.

Las opciones de productos personalizados se procesan al calcular el precio final. Cada opción tiene su propio precio definido, que se agrega al precio final.

Los precios de grupo, nivel y especiales se tienen en cuenta al mismo tiempo `($priceModel->getBasePrice())` y se elige el precio más bajo de los tres (o cuatro si incluye el precio regular) como precio base del producto.

Impuesto

El módulo Mage_Tax utiliza la clase de impuestos de un producto y si el precio del producto es inclusivo o no del impuesto para identificar la tasa correcta a aplicar.

Los siguientes factores se utilizan para calcular el impuesto sobre los productos:

- Clase de impuesto al producto
- El importe del impuesto ya incluido.
- Direcciones de facturación y envío
- Clase de impuesto al cliente
- Configuración de la tienda

Navegación en capas

Las clases responsables de renderizar la navegación en capas son:

- Mage_Catalog_Block_Layer_View

  -Manejar los filtros y opciones.

- Mage_Catalog_Block_Layer_State

-Control lo que actualmente se está filtrando.

Para implementar la navegación por capas en los atributos con modelos de origen personalizados, el método Mage_Catalog_Model_Layer_Filter_Abstract :: apply () se debería sobrescribir para determinar cómo se debe filtrar la colección de productos.

La navegación en capas es representada por los bloques Mage_Catalog_Block_Layer_View y Mage_Catalog_Block_Layer_State, que usan bloques de filtros para filtros individuales.

La navegación en capas utiliza la tabla de índice para la mayoría de los filtros, por ejemplo, precio, índice de atributo de producto, índice de producto decimal.

Categorías

Categorías en la base de datos

La jerarquía de categorías se administra almacenando el ID principal de una categoría. La jerarquía completa se muestra en la columna de ruta (ID separadas por barras). Hay una categoría especial con parent_id de 0. Esta es la verdadera categoría de raíz y cada una de las otras categorías de raíz, tal como se define en Magento, lo utiliza como padre compartido.

Para leer y administrar un árbol de categorías de la base de datos, se utilizan dos clases diferentes dependiendo de si el catálogo plano está habilitado, Mage_Catalog_Model_Resource_Category_Tree y Mage_Catalog_Model_Resource_Category_Flat.

La ventaja de las categorías planas es que es más rápido de consultar. Sin embargo, debe reconstruirse a partir de las tablas EAV cada vez que se produzca un cambio.

```
getChildren()
```

devuelve una cadena separada por comas de ID de hijos inmediatos

```
getAllChildren()
```

devuelve una cadena o matriz de todos los ID de hijos

```
getChildrenCategories()
```

devuelve una colección de categorías secundarias inmediatas Nota: si el catálogo plano está habilitado, las únicas categorías secundarias devueltas serán aquellas con include_in_menu = 1. En ambos casos, solo se devuelven las categorías activas.

Reglas de precios del catálogo

Las reglas de precios de catálogo aplican descuentos a los productos según la fecha, el producto, el sitio web y el grupo de clientes.

Cuando se llama a `getFinalPrice()` en un producto, se desencadena el evento catalog_product_get_final_price. Esto es observado por Mage_CatalogRule_Model_Observer que luego buscará cualquier regla de precio de catálogo que se aplique al producto. Si corresponde, luego mira la tabla de precios de la base de datos y vuelve a escribir el precio en el modelo del producto como un campo de datos Varien final_price.

Dentro de la base de datos, la tabla catalogrule describe las reglas, sus condiciones y sus acciones. catalogrule_product contiene los productos coincidentes y alguna información de la regla. Mientras tanto catalogrule_product_price contiene el precio después de que se haya aplicado la regla.

Tablas de indexación y planas

Las tablas de catálogo planas son gestionadas por indexadores de catálogo. Si la reconstrucción automática de los índices está habilitada, los indizadores de catálogo se reconstruyen cada vez que se actualiza un producto, una categoría o cualquier entidad relacionada. El método _afterSave () llama al proceso del indexador. De lo contrario, tienen que ser re-indexados manualmente a través de admin.

El tipo de producto afecta el índice de precios y el índice de acciones donde los productos pueden definir sus propios indexadores personalizados (en config.xml) para manejar sus datos para estos índices.

El módulo Mage_Index proporciona el marco con el que se pueden crear índices personalizados para ayudar a optimizar el rendimiento del sitio. La clase Mage_Index_Model_Indexer_Abstract debe ampliarse para crear un nuevo índice, implementando los métodos _registerEvent () y _processEvent (). Sin olvidar registrarlo en config.xml:

```
<global>
    <index>
        <indexer>
            <{name}>{model}</{name}>
        </indexer>
    </index>
</global>
```

Examples

Mage_Catalog_Model_Product_Type

```
/**
 * Magento
 *
 * NOTICE OF LICENSE
 *
 * This source file is subject to the Open Software License (OSL 3.0)
 * that is bundled with this package in the file LICENSE.txt.
 * It is also available through the world-wide-web at this URL:
 * http://opensource.org/licenses/osl-3.0.php
 * If you did not receive a copy of the license and are unable to
 * obtain it through the world-wide-web, please send an email
 * to license@magentocommerce.com so we can send you a copy immediately.
 *
 * DISCLAIMER
 *
 * Do not edit or add to this file if you wish to upgrade Magento to newer
 * versions in the future. If you wish to customize Magento for your
 * needs please refer to http://www.magentocommerce.com for more information.
 *
 * @category    Mage
 * @package     Mage_Catalog
 * @copyright   Copyright (c) 2012 Magento Inc. (http://www.magentocommerce.com)
 * @license     http://opensource.org/licenses/osl-3.0.php  Open Software License (OSL 3.0)
 */

/**
 * Product type model
 *
 * @category    Mage
 * @package     Mage_Catalog
 * @author      Magento Core Team
 */
class Mage_Catalog_Model_Product_Type
{
    /**
     * Available product types
     */
    const TYPE_SIMPLE       = 'simple';
    const TYPE_BUNDLE       = 'bundle';
    const TYPE_CONFIGURABLE = 'configurable';
    const TYPE_GROUPED      = 'grouped';
    const TYPE_VIRTUAL      = 'virtual';

    const DEFAULT_TYPE      = 'simple';
    const DEFAULT_TYPE_MODEL    = 'catalog/product_type_simple';
    const DEFAULT_PRICE_MODEL   = 'catalog/product_type_price';

    static protected $_types;
    static protected $_compositeTypes;
    static protected $_priceModels;
    static protected $_typesPriority;

    /**
     * Product type instance factory
     *
     * @param   Mage_Catalog_Model_Product $product
     * @param   bool $singleton
     * @return  Mage_Catalog_Model_Product_Type_Abstract
     */
    public static function factory($product, $singleton = false)
    {
```

```
        $types = self::getTypes();
        $typeId = $product->getTypeId();

        if (!empty($types[$typeId]['model'])) {
            $typeModelName = $types[$typeId]['model'];
        } else {
            $typeModelName = self::DEFAULT_TYPE_MODEL;
            $typeId = self::DEFAULT_TYPE;
        }

        if ($singleton === true) {
            $typeModel = Mage::getSingleton($typeModelName);
        }
        else {
            $typeModel = Mage::getModel($typeModelName);
            $typeModel->setProduct($product);
        }
        $typeModel->setConfig($types[$typeId]);
        return $typeModel;
    }

    /**
     * Product type price model factory
     *
     * @param   string $productType
     * @return  Mage_Catalog_Model_Product_Type_Price
     */
    public static function priceFactory($productType)
    {
        if (isset(self::$_priceModels[$productType])) {
            return self::$_priceModels[$productType];
        }

        $types = self::getTypes();

        if (!empty($types[$productType]['price_model'])) {
            $priceModelName = $types[$productType]['price_model'];
        } else {
            $priceModelName = self::DEFAULT_PRICE_MODEL;
        }

        self::$_priceModels[$productType] = Mage::getModel($priceModelName);
        return self::$_priceModels[$productType];
    }

    static public function getOptionArray()
    {
        $options = array();
        foreach(self::getTypes() as $typeId=>$type) {
            $options[$typeId] = Mage::helper('catalog')->__($type['label']);
        }

        return $options;
    }

    static public function getAllOption()
    {
        $options = self::getOptionArray();
        array_unshift($options, array('value'=>'', 'label'=>''));
        return $options;
    }
```

```
    static public function getAllOptions()
    {
        $res = array();
        $res[] = array('value'=>'', 'label'=>'');
        foreach (self::getOptionArray() as $index => $value) {
            $res[] = array(
               'value' => $index,
               'label' => $value
            );
        }
        return $res;
    }

    static public function getOptions()
    {
        $res = array();
        foreach (self::getOptionArray() as $index => $value) {
            $res[] = array(
               'value' => $index,
               'label' => $value
            );
        }
        return $res;
    }

    static public function getOptionText($optionId)
    {
        $options = self::getOptionArray();
        return isset($options[$optionId]) ? $options[$optionId] : null;
    }

    static public function getTypes()
    {
        if (is_null(self::$_types)) {
            $productTypes = Mage::getConfig()->getNode('global/catalog/product/type')-
>asArray();
            foreach ($productTypes as $productKey => $productConfig) {
                $moduleName = 'catalog';
                if (isset($productConfig['@']['module'])) {
                    $moduleName = $productConfig['@']['module'];
                }
                $translatedLabel = Mage::helper($moduleName)->__($productConfig['label']);
                $productTypes[$productKey]['label'] = $translatedLabel;
            }
            self::$_types = $productTypes;
        }

        return self::$_types;
    }

    /**
     * Return composite product type Ids
     *
     * @return array
     */
    static public function getCompositeTypes()
    {
        if (is_null(self::$_compositeTypes)) {
            self::$_compositeTypes = array();
            $types = self::getTypes();
```

```
            foreach ($types as $typeId=>$typeInfo) {
                if (array_key_exists('composite', $typeInfo) && $typeInfo['composite']) {
                    self::$_compositeTypes[] = $typeId;
                }
            }
        }
        return self::$_compositeTypes;
    }

    /**
     * Return product types by type indexing priority
     *
     * @return array
     */
    public static function getTypesByPriority()
    {
        if (is_null(self::$_typesPriority)) {
            self::$_typesPriority = array();
            $a = array();
            $b = array();

            $types = self::getTypes();
            foreach ($types as $typeId => $typeInfo) {
                $priority = isset($typeInfo['index_priority']) ?
abs(intval($typeInfo['index_priority'])) : 0;
                if (!empty($typeInfo['composite'])) {
                    $b[$typeId] = $priority;
                } else {
                    $a[$typeId] = $priority;
                }
            }

            asort($a, SORT_NUMERIC);
            asort($b, SORT_NUMERIC);

            foreach (array_keys($a) as $typeId) {
                self::$_typesPriority[$typeId] = $types[$typeId];
            }
            foreach (array_keys($b) as $typeId) {
                self::$_typesPriority[$typeId] = $types[$typeId];
            }
        }
        return self::$_typesPriority;
    }
}
```

Describir los tipos de productos estándar (simples, configurables, agrupados)

Sencillo

El tipo de productos simples debe usarse para que, por lo general, tengan una configuración única (talla única para todos). Esto podría incluir elementos tales como:

- Una caja de crayones, pequeña (24 colores)
- Una Caja de Crayones, Grande (64 Colores)
- Monitor SuperHighTech 26 "HD para computadora
- Figura de Acción de Barrack Obama (6 ")

Agrupados

Los productos agrupados le permiten crear un nuevo producto utilizando uno o más productos existentes en su tienda. Por ejemplo, supongamos que ya tiene una "Figura de Acción de Barrack Obama" y una "Figura de Acción de George W. Bush" en su tienda y desea venderlas como un paquete. Simplemente creará un nuevo producto agrupado (llamémoslo "Obama + Bush (¡obtenga ambos y gastemos el doble!)", Luego agregue ambas figuras de acción al grupo a través de la pestaña "Productos asociados".

Nota: Desafortunadamente, no puede establecer un precio especial de "grupo" directamente desde la página del producto. Para ofrecer un descuento por la compra de artículos en conjunto, deberá crear una nueva Regla de precios del carrito de compras.

Configurable

Producto configurable: este producto permite a sus clientes seleccionar la variante que desean al elegir las opciones. Por ejemplo, puedes vender camisetas en dos colores y tres tamaños. Debería crear seis productos simples como productos individuales (cada uno con sus propias SKU) y luego agregar estos seis a un producto configurable donde los clientes pueden elegir el tamaño y el color, y luego agregarlos a su carrito. Es posible una funcionalidad muy similar mediante el uso de Opciones personalizadas para productos simples. La diferencia entre un producto configurable y un producto que incluye opciones personalizadas es que el inventario no se verifica ni se actualiza para las opciones individuales durante la compra de las opciones personalizadas.

Virtual

Los productos virtuales son aquellos que no tienen una contraparte física o digital. No se envían, ni tienen un enlace de descarga. Este tipo de producto podría ser utilizado para servicios como:

- Limpieza de la casa
- Suscripción al boletín de 1 año

Nota: Si utiliza productos virtuales para "suscripciones", es importante tener en cuenta que no hay una forma integrada de administrar las suscripciones de renovación automática. Todas las compras realizadas en Magento, independientemente del tipo de producto, son compras únicas.

Haz

Este tipo de producto también se conoce como un "kit" en otro software de comercio electrónico. Este tipo de producto es ideal para circunstancias en las que el usuario debe seleccionar una cantidad de opciones configurables, pero al menos una opción. Esto podría incluir productos como:

- Sistemas informáticos personalizables
- Tuxedos / trajes personalizables
- Haga clic aquí para ver un video tutorial sobre el uso de paquetes

Descargable

Los productos descargables son similares a los productos virtuales, excepto que incluyen la capacidad de agregar uno o más archivos digitales para descargar. Los archivos pueden cargarse a través de la interfaz de administración, o bien cargarlos directamente en el servidor a través de FTP y luego agregarlos por URL. Cuando un cliente compra un producto descargable, Magento generará un enlace seguro y encriptado (para que los clientes no puedan ver la ubicación real del archivo) para que ese cliente descargue su archivo.

Esta categoría podría incluir productos tales como:

- Música / MP3s
- Software de ordenador

Nota : si tiene SSL habilitado para su sitio, las descargas pueden fallar en todas las versiones de IE, ya que IE contiene un error que impide la descarga a través de conexiones seguras si se establece el encabezado de no caché. Esto puede solucionarse fácilmente en un archivo htaccess eliminando los encabezados sin almacenamiento en caché y sin almacenamiento, o forzando que los enlaces de descarga no sean seguros.

Lea Entender los tipos de productos en línea:
https://riptutorial.com/es/magento/topic/7142/entender-los-tipos-de-productos

Capítulo 14: Estructura del módulo

Observaciones

Los módulos existen para ser extendidos. No puede cambiar `the app/code/` archivos sin prohibir futuras actualizaciones. En su lugar, agregamos un módulo a la `app/code/local` directorio `app/code/local` (puede que falte el directorio local, si es así, debe crearse manualmente. Esto es común en las versiones posteriores de Magento) para agregar funcionalidad local personalizada.

Todos los archivos de configuración del módulo comienzan con una etiqueta `<config>` . El nuevo módulo se declara dentro de la etiqueta `<modules>` . Llamaremos a un módulo llamado YOUR_COMPANY_HelloWorld, por lo tanto, se utilizan las etiquetas `<YOUR_COMPANY_HelloWorld>` . Aquí definimos la versión (primera = 0.0.1)

Puede encontrar una lista de **eventos XML** en: http://www.magentocommerce.com/wiki/5___modules_and_development/reference/module_config.xml

Si tiene algún problema, consulte: http://coding.smashingmagazine.com/2012/11/30/introducing-magento-layout/

Examples

Creando un módulo desde cero (Hello World)

El desarrollo de módulos personalizados de Magento es una parte fundamental de cualquier proyecto de Magento o de desarrollo de Magento, ya que en cualquier etapa es posible que desee integrar su propia funcionalidad / módulo en su proyecto Magento existente.

Lo primero que deben deshabilitar los desarrolladores es el caché del sistema. De lo contrario, el desarrollo se convertirá en un dolor, ya que cualquier cambio requerirá un lavado. Desde el panel de administración de magento: navegue a `System > Cache Management > Select All > Actions : Disable`. Utilice la siguiente guía para crear un nuevo módulo:

- Cree una carpeta en la convención de nomenclatura de `app/code/local/` - usualmente tome el nombre de su empresa, por ejemplo, `app/code/local/<YOUR_COMPANY>` .
- Ahora tenemos una ubicación personalizada para nuestros módulos. Cree otro directorio, llámelo como algo relacionado con el tipo de módulo que desea crear, por ejemplo, app / code / local / <YOUR_COMPANY> / HelloWorld / - "HelloWorld" es lo que llamaré este módulo.
- Este directorio necesita un `config.xml` para que Magento lo reconozca como un nuevo módulo. Crea otra carpeta llamada `etc` Seguido de un archivo xml llamado config.xml. El directorio debería verse como `app/code/local/<YOUR_COMPANY/HelloWorld/etc/config.xml` Esta es la estructura del archivo xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<config>
   <modules>
      <YOUR_COMPANY_HelloWorld>
         <version>0.0.1</version>
      </YOUR_COMPANY_HelloWorld>
   </modules>
</config>
```

- A continuación, los módulos deben ser declarados a Magento. Continuar con la `app/etc/modules` . Cree otro documento XML y `YOUR_COMPANY_HelloWorld` nombres elegidos de sus etiquetas: `YOUR_COMPANY_HelloWorld` en mi caso. En este documento escriba:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
   <modules>
      <YOUR_COMPANY_HelloWorld>
         <!-- Whether our module is active: true or false -->
         <active>true</active>
         <!-- Which code pool to use: core, community or local -->
         <codePool>local</codePool>
      </YOUR_COMPANY_HelloWorld>
   </modules>
</config>
```

- Una vez más, las etiquetas de configuración y módulo se utilizan para declarar un nuevo módulo a Magento. Activo es el valor predeterminado al que se puede acceder en el `Admin Panel under System > Configuration > Advanced` . `codePool` le dice a Magento en qué directorio buscar. `local` en nuestro caso

- Este módulo ya se ha configurado, por lo tanto, el modelo de nuestra estructura MVC. Debería poder ver su nuevo módulo en el Panel de administración en `System > Configuration > Advanced` . **Sin embargo, todavía no hace nada!** Deberá volver a nuestro archivo config.xml y definir los elementos XML.

- Siguiendo con el tutorial; Utilizaremos algunos de estos elementos XML para crear clases y manipular todas las páginas en la parte frontal de nuestro sitio. Para volver al archivo `config.xml` , escriba lo siguiente en la etiqueta `</modules>` :

```
<global>
   <!-- adding a new block definition -->
   <blocks>
       <!-- A unique short name for our block files -->
       <helloworld>
          <!-- the location of our modules block -->
          <class>YOUR_COMPANY_HelloWorld_Block</class>
       </helloworld>
   </blocks>
</global>
<!-- We are making changes to the frontend -->
<frontend>
   <!-- We are making changes to the layout of the front end -->
   <layout>
      <!-- we are adding a new update file -->
```

```
      <updates>
         <!-- Creating the name of our update and linking it the module -->
         <helloworld module="YOUR_COMPANY_HelloWorld">
             <!-- The name of the layout file we are adding -->
             <file>helloworld.xml</file>
         </helloworld>
     </updates>
  </layout>
</frontend>
```

- Como puede ver, estamos extendiendo constantemente en lugar de manipular archivos principales. La etiqueta `helloworld` está en minúscula porque apuntará a un Handle, y para la continuidad la nombraremos lo más cerca posible. Luego `YOUR_COMPANY_HelloWorld` esto con el módulo `YOUR_COMPANY_HelloWorld` .

- Estamos cambiando el diseño. Por lo tanto, necesitamos crear este Handle en el directorio de diseño. proceder a `app/design/frontend/base/default/layout` . Le dijimos al módulo que buscara el archivo `helloworld.xml` . Por eso debemos crearlo en este directorio. Que estas esperando. ¡¡Hazlo!! y poblarlo con:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Layout files begin with this code -->
<layout>
   <!-- this is the layout handle. Default is handled on all pages. We want this module
to execute on all pages -->
   <default>
      <!-- This is the block we want to bolt onto the existing before_body_end block -->
      <reference name="before_body_end">
          <!-- We define our new block and template to be added to before_body_end -->
          <block name="helloworld_footer" template="helloworld/footer.phtml"
type="helloworld/footer"/>
      </reference>
   </default>
</layout>
```

- Ahora, aquellos de ustedes que tengan una pequeña experiencia con Magento, o que hayan leído algunos tutoriales más notables de Magento, pueden estar quedándose boquiabiertos ante el hecho de que estamos realizando cambios en base / predeterminado ya que aquí es donde se encuentran los archivos principales de Magento. Sin embargo, no estamos modificando ningún archivo aquí, estamos creando nuevos y, además, estamos prefijando nuestro nombre de archivo con "helloworld", por lo que hay muy pocas posibilidades de que esto entre en conflicto con otros módulos o cause problemas con la actualización de Magento en el futuro. ¡Días felices!

- Como queremos afectar a todas las páginas, usamos la etiqueta predeterminada y la referenciamos al `before_body_end` estructural `before_body_end` . Esto actuará el papel de la Acción y activará la sección Ver de nuestra estructura MVC.

- Ahora entendemos que nos estamos atornillando al bloque `before_body_end` . y vinculándolo a nuestro bloque personalizado. Esto se llama una referencia y es un **gancho** . Actualmente no lo estamos conectando a nada existente, por lo tanto, debemos crear los archivos necesarios.

- En `helloworld.xml` indicamos en la plantilla un `footer.phtml` . Continúe con `app/design/frontend/base/default/template` y cree un directorio `helloworld` .
- Dentro de este directorio, cree el archivo `footer.phtml` y rellénelo con HTML, este tutorial simplemente lo escribe para mostrar algunas funciones de PHP vinculadas con nuestro archivo PHTML:

```
<p>Hello Everyone! Todays date is <?php echo $this->getDate() ?></p>
```

- Ahora necesitamos crear nuestro propio objeto de bloque para unir la plantilla con nuestra funcionalidad de bloque. Cree el directorio app / code / local / YOUR_COMPANY / HelloWorld / Block / y cree el archivo `Footer.php` dentro de este. Esto fue referenciado en nuestro zeta_layout.xml en el tipo "helloworld / footer". Rellene este archivo con:

```
<?php
class YOUR_COMPANY_HelloWorld_Block_Footer extends Mage_Core_Block_Template {
   public function getDate() {
      $date = date('Y-m-d');
      return urlencode($date);
   }
}
?>
```

- Esta es la funcionalidad que `getDate()` nuestra llamada `getDate()` que llamamos desde nuestro archivo `.phtml` . Extendemos el `Mage_Core_Block_Template` .
- Esta funcionalidad ahora está completa. ¡Prueba esto yendo a tu página de inicio donde deberías ver tu módulo en el pie de página de cada página!

Lea Estructura del módulo en línea: https://riptutorial.com/es/magento/topic/3962/estructura-del-modulo

Capítulo 15: Estructura MVC

Observaciones

MVC significa Model-View-Controller. Cualquier aplicación que separe su acceso a datos, lógica de negocios e interfaz de usuario se llama MVC. Puede haber dos tipos de MVC: basado en convenciones y basado en configuración. Por ejemplo, cakePHP está basado en convenciones, es decir, solo necesita seguir las instrucciones del sistema central para preparar su módulo en unas pocas líneas. Magento se basa en la configuración, es decir, debe especificar cada una de las cosas en el archivo de configuración de su módulo para que funcione. Magento tiene controlador (para enrutamiento), bloque, modelo y archivo de plantilla. Cómo funciona el MVC de Magento:

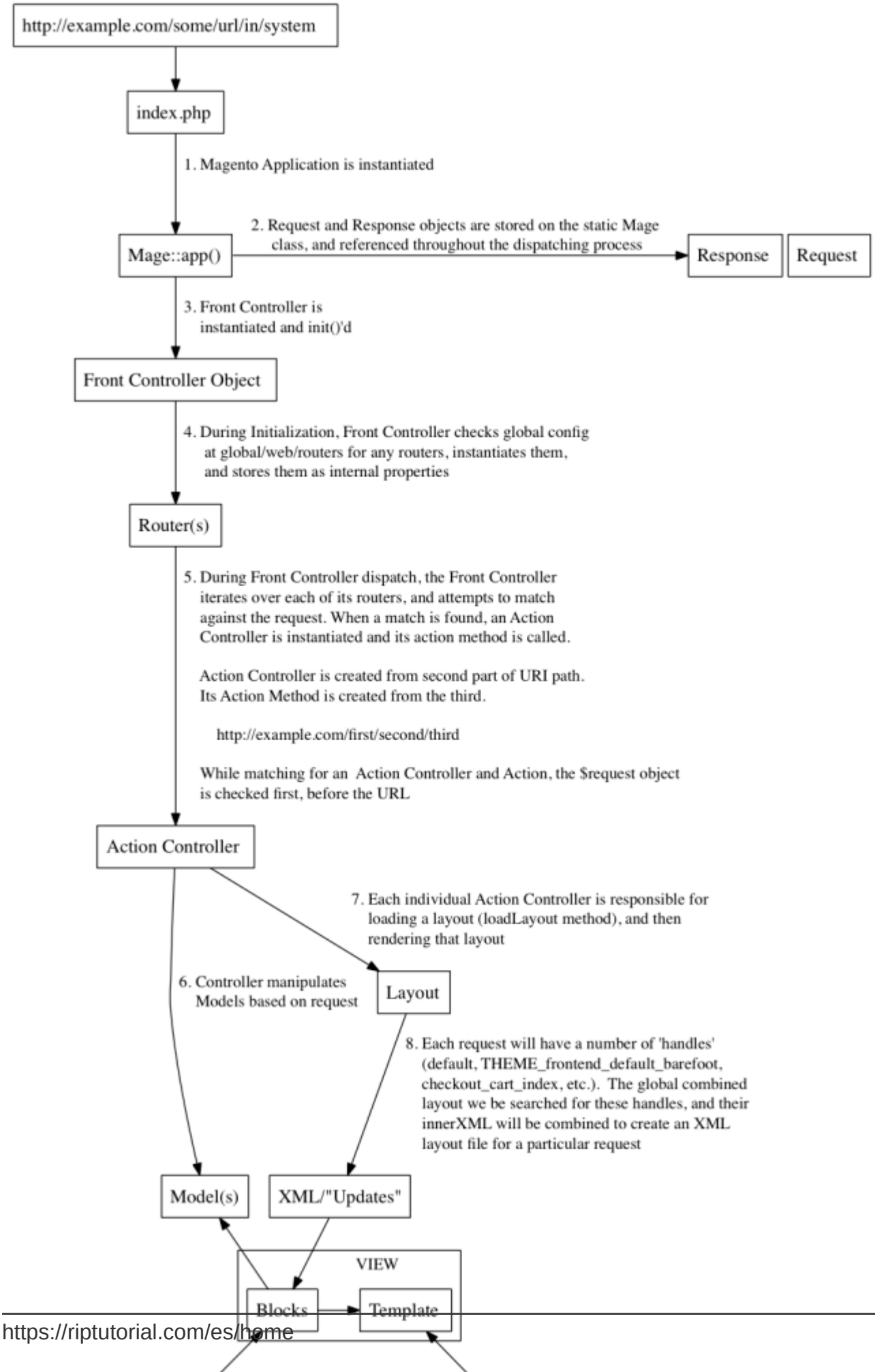
1. Cuando ingresas la URL (algo como http://mysite.com/frontname/controller/method/param1/value1/param2/value2) , esta URL es interceptada por un archivo PHP llamado index.php que crea una instancia de la aplicación Magento
2. La aplicación Magento crea una instancia del objeto Front Controller
3. Además, el controlador frontal crea una instancia de los objetos del enrutador (especificados en config.xml, etiqueta global del módulo)
4. Ahora, Router es responsable de "hacer coincidir" el nombre de usuario que se encuentra en nuestra URL
5. Si se encuentra "coincidencia", verá el nombre del controlador y el nombre del método en la URL, que finalmente se llama.
6. Ahora, dependiendo de lo que está escrito en el nombre de la acción (nombre del método), se ejecuta. Si se llama a alguno de los modelos, el método del controlador creará una instancia de ese modelo y llamará al método que se solicita.
7. Luego, la acción del controlador (método) crea una instancia del objeto Diseño, que llama al Bloque especificado para este nombre de acción (método) (Cada nombre de acción del controlador tiene un archivo de plantilla y bloque asociado, que se puede encontrar en app / design / frontend o adminhtml / namespace / module / layout / module.xml, nombre del archivo de diseño (module.xml) se puede encontrar en config.xml de ese módulo, en la etiqueta de actualizaciones de diseño).
8. El archivo de plantilla (.phtml) ahora llama al bloque correspondiente para cualquier solicitud de método. Por lo tanto, si escribe $ this-> methodName en el archivo .phtml, verificará "methodName" en el archivo de bloque que está asociado en el archivo module.xml.
9. Bloque contiene la lógica de PHP. Hace referencia a los modelos para cualquier dato de DB.
10. Si el bloque, el archivo de plantilla o el controlador necesitan obtener / establecer algunos datos de / a la base de datos, pueden llamar al modelo directamente como Mage :: getModel ('nombre de módulo / nombre de modelo').

Examples

Entender MVC en Magento

MVC Flow en Magento

Magento MVC Flow
Created By http://alanstorm.com



Capítulo 16: Hoja de trucos para tareas rápidas

Observaciones

https://gist.github.com/arosenhagen/2397824

Muchos muchos fragmentos muy útiles

Examples

Obtener producto Cantidad en stock

carga ($ id); // o cárguelo por SKU // $ sku = "microsoftnatural"; // $ _product = Mage :: getModel ('catalog / product') -> loadByAttribute ('sku', $ sku); $ stock = Mage :: getModel ('cataloginventory / stock_item') -> loadByProduct ($ _ product); print_r ($ stock-> getData ()); echo $ stock-> getQty (); echo $ stock-> getMinQty (); echo $ stock-> getMinSaleQty ();

Lea Hoja de trucos para tareas rápidas en línea:
https://riptutorial.com/es/magento/topic/7060/hoja-de-trucos-para-tareas-rapidas

Capítulo 17: Imagen del producto Urls

Introducción

Obtener urls de imagen de producto para miniatura, imagen pequeña e imagen base. Obtener en caché también y no cachés URLs de medios directos.

Examples

Imágenes en caché de URL

```
Mage::helper('catalog/image')->init($item->getProduct(), 'thumbnail');
Mage::helper('catalog/image')->init($item->getProduct(), 'small_image');
Mage::helper('catalog/image')->init($item->getProduct(), 'image');
```

Urls de imágenes no almacenadas en caché desde medios

```
Mage::getModel('catalog/product_media_config')->getMediaUrl($product->getThumbnail());
//Thumbnail
Mage::getModel('catalog/product_media_config')->getMediaUrl($product->getSmallImage());
//Small Image
Mage::getModel('catalog/product_media_config')->getMediaUrl($product->getImage()); //Base
```

Lea Imagen del producto Urls en línea: https://riptutorial.com/es/magento/topic/9262/imagen-del-producto-urls

Capítulo 18: Iniciar sesión para archivar

Sintaxis

- registro de función estática pública ($ mensaje, $ nivel = nulo, $ archivo = '', $ fuerzaLog = falso)

Parámetros

Parámetro	Detalles
cadena $ mensaje	El mensaje que será registrado.
entero $ nivel	Nivel de registro
cadena $ archivo	Ruta y nombre con extensión de archivo que se guardará en `var/log/` . Si NULL o no se especifica, se utilizará `system.log` .
bool $ forceLog	Si se establece en `TRUE` registro se escribirá aunque el modo de desarrollador esté desactivado y el registro esté inactivo.

Observaciones

El registro está desactivado de forma predeterminada a menos que el modo desarrollador esté activo.

Todas las excepciones se registran en `exceptions.log` sin importar si el registro está habilitado en la configuración.

El inicio de sesión se puede habilitar iniciando sesión en Magento Admin y procediendo a:

- Sistema> Configuración (barra superior)
- Desarrollador (menú izquierdo)
- Sección de configuración de registro
- Seleccione Sí en la lista desplegable `Enabled` .
- Guarda la configuración en la esquina superior derecha.

Tipo de variable de mensaje

Aunque la documentación define que el mensaje debe ser una cadena, si se pasa una matriz, hay un bloque de código en ese método para cuidarlo con `print_r` :

```
if (is_array($message) || is_object($message)) {
    $message = print_r($message, true);
}
```

Nivel de registro

Si el parámetro de nivel se establece en nulo, se toma el nivel DEBUG.

`$level = is_null($level) ? Zend_Log::DEBUG : $level;` Los niveles se declaran en el archivo: `lib\Zend\log.php`

```
const EMERG   = 0;  // Emergency: system is unusable
const ALERT   = 1;  // Alert: action must be taken immediately
const CRIT    = 2;  // Critical: critical conditions
const ERR     = 3;  // Error: error conditions
const WARN    = 4;  // Warning: warning conditions
const NOTICE  = 5;  // Notice: normal but significant condition
const INFO    = 6;  // Informational: informational messages
const DEBUG   = 7;  // Debug: debug messages
```

Las constantes en forma de `Zend_Log::INFO` o número entero en el rango especificado anteriormente se pueden pasar como parámetro de nivel de registro.

Examples

Archivo de registro personalizado

```
Mage::log('My log entry', null, 'mylogfile.log');
```

Esto se registrará en

```
/var/log/mylogfile.log
```

Registro predeterminado

```
Mage::log('My log entry');
Mage::log('My log message: '.$myVariable);
Mage::log($myArray);
Mage::log($myObject);
```

Esto se registrará en `/var/log/system.log`

Los objetos y las matrices se escriben automáticamente mediante una directiva `print_r()` . Tenga cuidado al usar objetos, ya que estos pueden tener un tamaño considerable.

```
Mage::logException($e);
```

Esto registrará la cadena de rastreo de excepciones en `/var/log/exception.log`

Lea Iniciar sesión para archivar en línea: https://riptutorial.com/es/magento/topic/2363/iniciar-sesion-para-archivar

Capítulo 19: Magento Caching

Examples

Cómo almacenar en caché datos personalizados en Magento

```
const CACHE_TAG_NAMESPACE_MODULE = "YOUR_MODULES_CACHE_TAGS";
$cacheGroup = 'namespace_module';
$useCache = Mage::app()->useCache($cacheGroup);
if (true === $useCache) {
    $cacheId = 'unique_name';
    if ($cacheContent = Mage::app()->loadCache($cacheId)) {
        $html = $cacheContent;
        return $html;
    } else {
        try {
            $cacheContent = $html;
            $tags = array(model::CACHE_TAG_NAMESPACE_MODULE);
            $lifetime = Mage::getStoreConfig('core/cache/lifetime');
            Mage::app()->saveCache($cacheContent, $cacheId, $tags, $lifetime);
        } catch (Exception $e) {
            // Exception = no caching
            Mage::logException($e);
        }
        return $html;
   }
}
// Default:
return $html;
```

Limpiar caché por ID de caché

```
Mage::app()->removeCache($cacheId);
```

Vaciar todas las entradas de caché de Magento

```
Mage::app()->cleanCache()
```

o:

```
Mage::app()->getCacheInstance()->flush();
```

Usa Redis como un backend de caché

Configuración de redis:

1. Instalar redis (se requiere 2.4+)
2. Instalar phpredis
3. Instale la extensión de Magento `Cm_Cache_Backend_Redis` (solo para Magento 1.7 e inferior)

4. Edita tu `app/etc/local.xml` :

```
<global>
  ...
  <cache>
    <backend>Cm_Cache_Backend_Redis</backend>
    <backend_options>
      <server>127.0.0.1</server> <!-- or absolute path to unix socket -->
      <port>6379</port>
      <persistent></persistent>
      <database>0</database>
      <password></password>
      <force_standalone>0</force_standalone>
      <connect_retries>1</connect_retries>
      <automatic_cleaning_factor>0</automatic_cleaning_factor>
      <compress_data>1</compress_data>
      <compress_tags>1</compress_tags>
      <compress_threshold>20480</compress_threshold>
      <compression_lib>gzip</compression_lib> <!-- Supports gzip, lzf and snappy -->
    </backend_options>
  </cache>
  ...
</global>
```

Lea Magento Caching en línea: https://riptutorial.com/es/magento/topic/4902/magento-caching

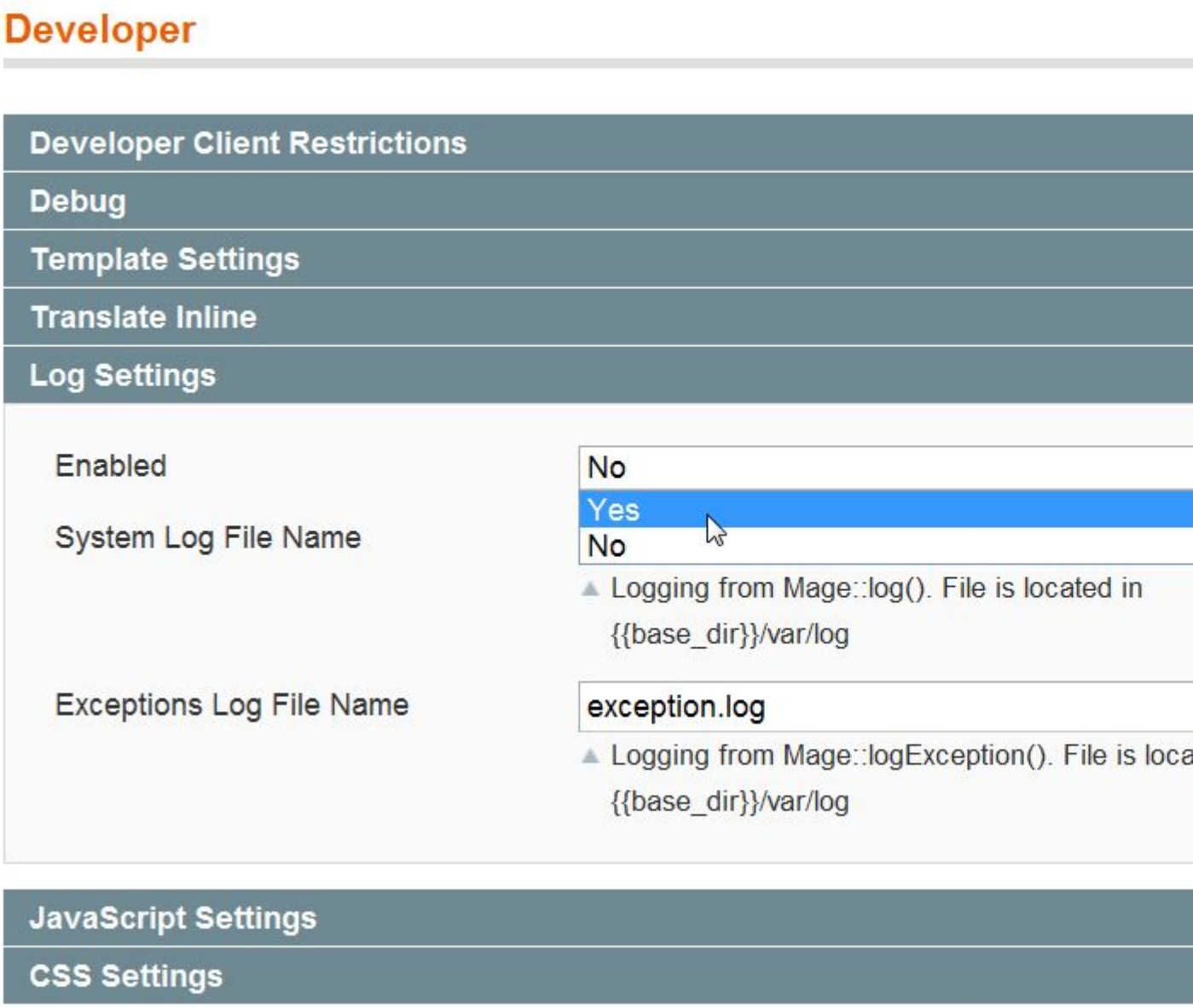
Capítulo 20: Manejo de errores magento, mensajes e informes.

Observaciones

Ubicaciones de registro de errores

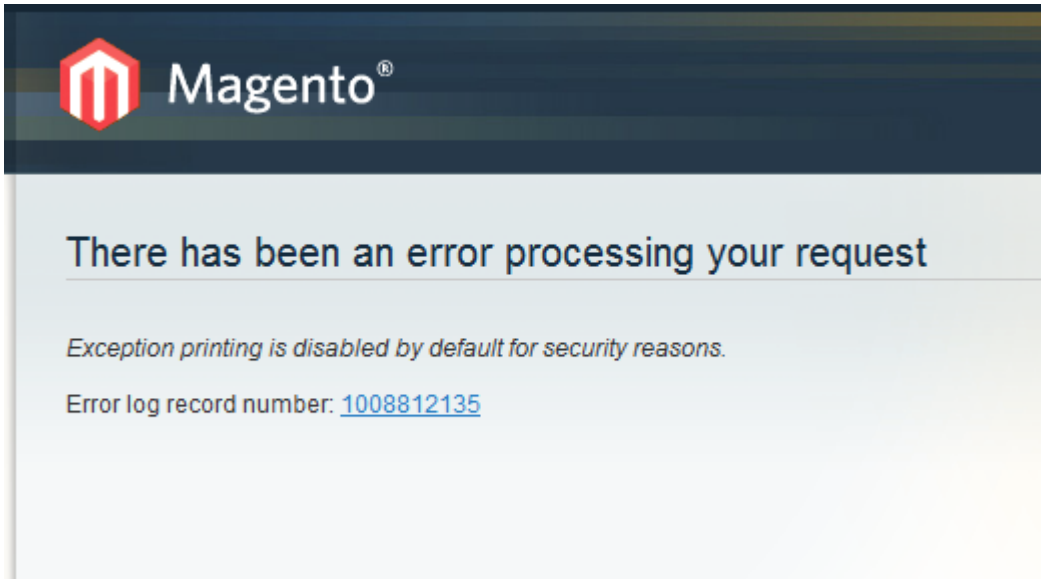
`/var/log/`

Normalmente, los archivos system.log y exception.log existirán en la carpeta `/var/log/` . Estos contienen la mayor parte de la información que necesitará. Puede verificar si están habilitados y cuáles son los nombres de la excepción y el registro del sistema yendo a `System > Configuration > System > Developer > Log Settings` .



/var/report/

Los archivos de informe se generan en esta carpeta después de que un usuario haya encontrado un error. Cada archivo solo incluye los detalles de un error. Estos se utilizan para ocultar los detalles del error del público. En la página de error habrá un número de informe que es una referencia al archivo correspondiente con el mismo nombre en la carpeta `/var/report/` .



Examples

Habilitar la visualización de informes de errores

En la página de índice cambia lo siguiente:

```
error_reporting(E_ALL | E_STRICT);
```

a

```
error_reporting(E_ALL);
```

Establecer `$_SERVER['MAGE_IS_DEVELOPER_MODE'] = true`

y descomenta esta línea (elimina el `#`)

```
#ini_set('display_errors', 1);
```

También puede establecer el modo Dev usando `SetEnv` en su archivo `.htaccess`

Para hacer que el error sea legible y más fácil de entender, haga lo siguiente:

1. Abra su directorio de instalación de Magento. Ve a la carpeta de errores.
2. Cambiar el nombre de `local.xml.sample` archivo a `local.xml` .

3. Actualizar la página de error en el navegador.

Lea Manejo de errores magento, mensajes e informes. en línea:
https://riptutorial.com/es/magento/topic/2369/manejo-de-errores-magento--mensajes-e-informes-

Capítulo 21: Modelo

Examples

Modelo de carga

Puede cargar el modelo de Magento usando el siguiente código: Mage :: getModel ('nombre de modelo / nombre de modelo')

Ejemplo: Mage :: getModel ('catalog / product') Esto cargará Mage_Catalog_Model_product

Crear un modelo vacío

Para crear un nuevo modelo en su módulo Agregue una carpeta `Model` en su carpeta raíz del módulo y cree un archivo Modelname.php en esta carpeta. por ejemplo Rick / Demo / Model / Modelname.php

El nombre de clase de tu modelo no es así, llámalo así:

```
<?php
class Rick_Demo_Model_Modelname {

}
```

asegúrese de que su modelo esté definido en su `config.xml` en la carpeta `etc` de su módulo

Aquí un ejemplo:

2.0.4 Rick_Demo_Model

Para cargar tu módulo usa el siguiente código:

```
Mage::getModel('customemodelname/modelname')
```

Lea Modelo en línea: https://riptutorial.com/es/magento/topic/7665/modelo

Capítulo 22: Obtener el nombre de la categoría de la página del producto

Examples

Obtener la categoría principal

```
$_cat = new Mage_Catalog_Block_Navigation();
$curent_cat = $_cat->getCurrentCategory();
$curent_cat_id = $curent_cat->getId();
$parentId=Mage::getModel('catalog/category')->load($curent_cat_id)->getParentId();
$parent = Mage::getModel('catalog/category')->load($parentId);
$categorydaddy = $parent->getName();
```

Obtener la categoría actual

```
    $categoryName = Mage::registry('current_category')->getName();
    foreach ($categoryName as $_category):
       $categoryName = $_category->getName();
    endforeach;
```

Lea Obtener el nombre de la categoría de la página del producto en línea: https://riptutorial.com/es/magento/topic/6078/obtener-el-nombre-de-la-categoria-de-la-pagina-del-producto

Capítulo 23: Obtener productos de la base de datos

Examples

Obtener producto por sku

```
$sku = 'sku-goes-here';
$product = Mage::getModel('catalog/product')->loadByAttribute('sku', $sku);
```

Obtener producto por ID

```
$id = 1;
$product = Mage::getModel('catalog/product')->load($id);
if($product->getId()){
    //product was found
}
```

Colección de productos - consulta LIKE

```
$collection = Mage::getModel('catalog/product')->getCollection();
$collection->addAttributeToFilter('sku', array('like' => 'UX%'));
```

Obtener colección de productos por atributo

```
$collection = Mage::getModel('catalog/product')->getCollection();
// Using operator
$collection->addAttributeToFilter('status', array('eq' => 1));
// Without operator (automatically uses 'equal' operator
$collection->addAttributeToFilter('status', 1);
```

Obtener datos de objeto de producto

```
// First load a product object

$product->getSku();
$product->getName();

// Alternative method
$product->getData('sku');
$product->getData('name');
```

Obtener datos del formulario de recogida de productos.

```
// First load a collection object
```

```
foreach($collection as $product) {

    $product->getSku();
    $product->getName();

    // Alternative method
    $product->getData('sku');
    $product->getData('name');
}
```

Colección de productos - con atributos.

```
//all attributes
$collection = Mage::getModel('catalog/product')
    ->getCollection()
    ->addAttributeToSelect('*');
//specific attributes
$collection = Mage::getModel('catalog/product')
    ->getCollection()
    ->addAttributeToSelect('name');
//certain attributes are special, such as price and images
//for images, then you can use 'getMediaGalleryImages'
$product->load('media_galley');
```

Compruebe si el producto se cargó correctamente

```
$productFound = ($product->getId() !== null)
```

Obtener identificación del producto por SKU

```
$sku = 'some-sku';
$productId = Mage::getModel('catalog/product')->getIdBySku($sku);
if($productId){
   //sku exists
}
```

Obtener colección de productos de una lista de SKUs

```
$skuList = array('SKU-1', 'SKU-2',...,'SKU-n);
```

```
$_productCollection = Mage::getModel('catalog/product')
->getCollection()
->addAttributeToFilter('sku', array('in' => $skuList));
```

O

```
$_productCollection = Mage::getResourceModel('catalog/product_collection')
->addAttributeToFilter('sku', array('in' => $skuList));
```

Establecer límite en la colección de productos

```
$collection = Mage::getModel('catalog/product')
            ->getCollection()
            ->setPageSize(20)
            ->setCurPage(1);
```

Lea Obtener productos de la base de datos en línea:
https://riptutorial.com/es/magento/topic/1102/obtener-productos-de-la-base-de-datos

Capítulo 24: Obtener usuario actual

Examples

Obtener usuario administrador actual

```
Mage::getSingleton('admin/session')->getUser();
```

Obtener cliente actual

```
Mage::helper('customer')->getCustomer();
```

o

```
Mage::getSingleton('customer/session')->getCustomer();
```

Compruebe si el usuario está conectado

```
Mage::getSingleton('customer/session')->isLoggedIn()
```

Lea Obtener usuario actual en línea: https://riptutorial.com/es/magento/topic/2157/obtener-usuario-actual

Capítulo 25: Obtenga el nombre de la tienda y otros detalles de la configuración del sistema

Examples

Obtenga el nombre del frontend para la vista de tienda actual

```
Mage::app()->getStore()->getFrontendName();
```

Obtener ID de tienda actual

```
Mage::app()->getStore()->getStoreId();
```

Obtener código de tienda actual

```
Mage::app()->getStore()->getCode();
```

Esto devuelve el código de la tienda, por ejemplo, 'en' para un escaparate que está configurado para el inglés y se llama 'en', y no la identificación numérica.

Determine si la vista de tienda está habilitada

```
Mage::app()->getStore()->getIsActive();
```

Obtener ID de sitio web para la tienda actual

```
Mage::app()->getStore()->getWebsiteId();
```

Obtener el modelo de tienda actual

```
Mage::app()->getStore();
```

Devuelve una instancia de `Mage_Core_Model_Store`

Obtener el nombre del grupo para la tienda

```
Mage::app()->getStore()->getGroup()->getName()
```

Obtener todas las tiendas de Magento

```
Mage::app()->getStores();
```

Devuelve una matriz de modelos `Mage_Core_Model_Store` .

Lea Obtenga el nombre de la tienda y otros detalles de la configuración del sistema en línea: https://riptutorial.com/es/magento/topic/1876/obtenga-el-nombre-de-la-tienda-y-otros-detalles-de-la-configuracion-del-sistema

Capítulo 26: Obteniendo URLs de Magento

Sintaxis

- $ this-> getSkinUrl ('images / my-image.jpg');

Parámetros

camino de imágenes	detalles
ejemplo: *'images / my-images.jpg'*	camino para la imagen

Observaciones

Obtenga url de imágenes formateadas y evite dependencias temáticas.

Examples

En la interfaz actual / tema

volver http://www.example.com/skin/frontend/ {interface} / {theme} /images/my-image.jpg

Obtener url de piel

```
Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_SKIN);
```

Obtener Base Url

```
Mage::getBaseUrl();
```

Asegurar la url de la piel

```
$this->getSkinUrl('images/imagename.gif', array('_secure'=>true));
```

Obtener url de medios

```
Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_MEDIA);
```

URL de piel insegura

```
$this->getSkinUrl('images/imagename.jpg');
```

Obtener url de tienda

```
Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_WEB);
```

Obtener js url

```
Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_JS);
```

Obtener la URL actual

```
Mage::helper('core/url')->getCurrentUrl();
```

Lea Obteniendo URLs de Magento en línea:
https://riptutorial.com/es/magento/topic/2148/obteniendo-urls-de-magento

Capítulo 27: Optimizando Magento Para Velocidad

Examples

Optimizando Magento cambiando el archivo .htaccess

Magento es una aplicación de comercio electrónico muy popular. Ofrece una gran cantidad de personalización y habilidades desde la instalación inicial. Aquí hay algunas sugerencias para optimizar una instalación de Magento.

Habilitar la compresión de salida

En su archivo .htaccess para Magento encontrará una sección de texto que comienza con la línea,

```
<IfModule mod_deflate.c> and ending at </IfModule>
```

Esta sección de código se puede usar para activar el módulo mod_deflate de Apache, que proporciona compresión para texto, css y javascript. Deseará descomentar (eliminar el símbolo #) varias líneas para que se vea así:

##

habilitar la compresión de archivos servidos de apache

http://developer.yahoo.com/performance/rules.html#gzip

```
# Insert filter on all content
SetOutputFilter DEFLATE
# Insert filter on selected content types only
AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css text/javascript

# Netscape 4.x has some problems...
BrowserMatch ^Mozilla/4 gzip-only-text/html

# Netscape 4.06-4.08 have some more problems
BrowserMatch ^Mozilla/4\.0[678] no-gzip

# MSIE masquerades as Netscape, but it is fine
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

# Don't compress images
SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary

# Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary
</IfModule>
```

Habilitación de encabezados caducidad

Los que visitan por primera vez cualquier página web deben realizar varias solicitudes HTTP. Al utilizar el encabezado "Expires", los componentes de las solicitudes se pueden almacenar en caché. Esto evita solicitudes HTTP innecesarias en vistas de página subsiguientes.

Desea buscar el área del archivo .htaccess que comienza con <IfModulemod_expires.c> y termina con la primera vez que lo ve, y que se vea así:

```
<IfModule mod_expires.c>

###########################################
## Add default Expires header
## http://developer.yahoo.com/performance/rules.html#expires
    ExpiresActive On
    ExpiresDefault "access plus 1 year"

</IfModule>
```

Configuración de administrador

Fusionar archivos JS y CSS

Este ajuste en particular reducirá la cantidad de solicitudes HTTP en su sitio de comercio electrónico. [box type = "alert" border = "full"] Nota: Esto puede romper algunas aplicaciones. Después de realizar los siguientes pasos, asegúrese de que el sitio siga funcionando como lo hizo antes de habilitar esta función. [/ Box]

1. Inicie sesión en su área de administración y vaya a - Sistema> Configuración> Desarrollador
2. En "Configuración de JavaScript", cambie "Combinar archivos JavaScript" a sí.
3. En "Configuración de CSS", cambia "Combinar archivos CSS" a sí.
4. Finalmente querrás borrar tu caché de Magento.

Habilitar catálogos planos

El modelo que usa Magento para almacenar resultados de datos de clientes y productos en consultas de SQL más largas que el promedio y más lecturas. Al habilitar la opción de catálogo plano para categorías y productos, los datos del producto se fusionarán en una tabla, lo que mejorará el rendimiento.

Inicie sesión en su área de administración y vaya a - Sistema> Configuración> Catálogo En “Frontend”, cambie “Usar categoría de catálogo plano” a sí. Bajo "Frontend", cambie "Usar producto de catálogo plano" a sí, esto es opcional. A continuación, tendrá que borrar su caché de Magento. Finalmente, necesitarás reindexar las tablas. Habilitar compilación

[box type = "alert" border = "full"] Nota: Esto puede romper algunas aplicaciones. Después de realizar los siguientes pasos, asegúrese de que el sitio siga funcionando como lo hizo antes de habilitar esta función. [/ Box]

1. Inicie sesión en su área de administración y vaya a - Sistema> Herramientas> Compilación
2. A continuación, simplemente haga clic en el botón Ejecutar proceso de compilación
3. Después de que la compilación se haya ejecutado, debería activarse automáticamente.

Habilitar caché del sistema

1. Inicie sesión en su área de administración y vaya a - Sistema> Caché administración
2. A continuación, haga clic en el enlace Seleccionar todo.
3. Finalmente, asegúrese de que las Acciones estén configuradas en Habilitar y haga clic en enviar

Deshabilitar el registro de errores

Inicie sesión en su área de administración y vaya a - Sistema> Configuración> Desarrollador. En la sección Configuración de registro, asegúrese de que Activado esté configurado en Sin sugerencias de mantenimiento de la base de datos

Hay varias tablas utilizadas por Magento para el registro. Si bien el registro es muy importante para saber qué tiene y qué está sucediendo en su tienda, los registros pueden crecer mucho muy rápidamente, por lo que el mantenimiento regular puede ser de gran ayuda.

Aquí están las tablas para el registro:

```
log_customer
log_visitor
log_visitor_info
log_url
log_url_info
log_quote
report_viewed_product_index
report_compared_product_index
report_event
catalog_compare_item
```

Lea Optimizando Magento Para Velocidad en línea:
https://riptutorial.com/es/magento/topic/8010/optimizando-magento-para-velocidad

Capítulo 28: Pedidos

Examples

Obtener orden por ID

```
$orderid = 12345;
$order = Mage::getModel('sales/order')->load($orderid);
```

El código anterior es aproximadamente análogo a la siguiente consulta SQL.

```
select * from sales_flat_order where entity_id=12345;
```

Obtener pedido por ID de incremento

```
$incrementid = 100000000;
$order = Mage::getModel('sales/order')->loadByIncrementId($incrementid);
```

El código anterior es aproximadamente análogo a la siguiente consulta SQL.

```
select * from sales_flat_order where increment_id=100000000;
```

El `increment_id` es el identificador de orden que enfrenta el cliente, mientras que `entity_id` es el identificador de nivel de base de datos para el pedido.

Añadir comentario al historial de pedidos

Puedes añadir comentario y estado a pedido. Obtener orden :

```
$orderid = 12345;
$order = Mage::getModel('sales/order')->load($orderid);
```

Y añadir comentario:

```
//$isNotify means you want to notify customer or not.

$order->addStatusToHistory($status, $message, $isNotify);
$order->save()
```

Lea Pedidos en línea: https://riptutorial.com/es/magento/topic/1556/pedidos

Capítulo 29: Representación

Observaciones

Personalizar la funcionalidad del núcleo con temas

Los temas tienen archivos de diseño que, entre otras cosas, se pueden usar para cambiar los bloques que aparecen en la página. La plantilla de bloque también se puede cambiar y se pueden llamar diferentes métodos.

Diseños a nivel de tienda

Los temas estructurados jerárquicos de Magento significa que un tema base puede extenderse y asignarse a nivel de tienda.

Registro de temas personalizados

Los temas se pueden configurar de tres maneras:

1. Por tienda en Sistema> Configuración> Diseño.
2. Cambio de diseño con límites de tiempo Sistema> Diseño.
3. Las excepciones de tema también se pueden establecer en una categoría y nivel de producto.

Paquete versus tema

Un paquete tiene múltiples temas. Cada uno de los temas en un paquete se hereda del tema predeterminado dentro de un paquete.

Diseño alternativo

El procedimiento de reserva de tema para localizar archivos de plantillas es:

1. {paquete} / {tema}
2. {paquete} / predeterminado
3. base / predeterminado

Para agregar más directorios al mecanismo de reserva del tema, es necesario volver a escribir el método Mage_Core_Model_Design_Package :: getFilename

Para el área de administración, el respaldo es predeterminado / predeterminado.

Plantilla y rutas de diseño

Bloques

Los bloques se utilizan para la salida. El bloque raíz es el padre de todos los bloques y es de tipo Mage_Page_Block_Html.

Los bloques Mage_Core_Block_Template usan archivos de plantilla para representar contenido. El nombre del archivo de la plantilla se establece dentro de setTemplate () o addData ('plantilla') con rutas relativas.

Las plantillas son solo piezas de PHP incluidas en Mage_Core_Block_Template. Por lo tanto $ esto en una plantilla se refiere al bloque.

Mage_Core_Block_Template utiliza un búfer antes de incluir una plantilla para evitar la salida prematura.

El método Mage_Core_Model_Layout :: createBlock crea instancias de bloques.

La clase Mage_Core_Model_Layout_Update considera qué bloques deben crearse para cada página mirando los controladores de diseño.

Todos los bloques de salida se representan, por ejemplo, llamando a HTML (), que a su vez puede elegir representar a sus hijos.

Los bloques Text y Text_List representan automáticamente su contenido.

Hay dos eventos que se activan alrededor de la representación de bloques que se pueden usar para modificar el bloque antes y después de representar el HTML:

core_block_abstract_to_html_before core_block_abstract_to_html_after Un bloque hijo solo se procesará automáticamente si es de la clase Mage_Core_Block_Textlist de lo contrario, se deberá llamar al método getChildHtml.

Se puede acceder a las instancias de bloques a través del diseño, por ejemplo, Mage :: app () -> getLayout () y $ controller-> getLayout (). La salida del bloque es controlada por la función _toHtml ().

Las plantillas se representan mediante los métodos renderView () / fetchView () dentro de un bloque de plantillas. El búfer de salida se puede desactivar con $ layout-> setDirectOutput.

Es posible agregar un bloque al diseño actual, pero se debe hacer antes de llamar al método renderLayout ().

Diseño XML

```
<reference>
  -edit a block
<block>
  - define a block
<action>
  - call method on a block
<update>
  - include nodes from another handle.
```

Los archivos de diseño se pueden registrar en config.xml:

```
<config>
```

```
    <{area}>
        <layout>
            <updates>
                <{name}>
                    <file>{filepath}</file>
                </{name}>
            </updates>
        </layout>
    </{area}>
</config>
```

La salida de la página se puede personalizar de las siguientes maneras:

- Cambios de plantilla
- Cambios de diseño
- Anulando bloques
- Observadores Las variables en bloques se pueden establecer de las siguientes maneras:
- Diseño -A través de acciones o atributos
- Controlador - $ this-getLayout () -> getBlock ()
- Bloques secundarios - $ this-> getChild ()
- Otro -Mage :: app () -> getLayout ()

Activos de bloque de cabeza

Los recursos de JavaScript y CSS se manejan en el bloque Mage_Page_Block_Html_head. Este bloque maneja la fusión de activos en un solo archivo para minimizar las solicitudes HTTP. El archivo combinado se basa en el tiempo de edición de los archivos de origen.

Al fusionar CSS, se llama a una función de devolución de llamada en Mage_Core_Model_Design_Package para actualizar las directivas @import o url () con las URL correctas

Examples

Diferentes mecanismos para desactivar la salida de bloque.

- Si la respuesta ya se ha creado y establecido en el objeto de respuesta fuera del proceso de representación normal (por ejemplo, en un observador), la marca 'no-renderLayout' se puede establecer en el controlador de acción usando

```
Mage::app()->getFrontController()->getAction()->setFlag('','no-renderLayout');
```

- Esto evita que `renderLayout()` procese los bloques de salida.
- Lo mismo se puede lograr llamando a setNoRender (true) en el controlador frontal: `Mage::app()->getFrontController()->setNoRender(true);`
- Establecer el `isDispatched()` en el objeto de respuesta podría ser más eficiente para lograr un efecto similar.

diferentes tipos de bloques

- Mage_Core_Block_Template
- Mage_Core_Block_Text_List
- Mage_Core_Block_Messages
- Mage_Core_Block_Text_Tag
- Mage_Core_Block_Text
- Mage_Page_Block_Template_Links Todos los bloques son bloques **estructurales** o Todos los bloques son bloques estructurales o bloques de contenido. Ejemplo:
- **core / text_list**

Ejemplo de un **bloque estructural** .

No utiliza plantillas; simplemente se utiliza para generar el contenido de todos sus bloques secundarios uno tras otro.

- **núcleo / plantilla**

Ejemplo de un **bloque de contenido** .

La salida de este tipo de bloque depende de la plantilla asignada. Sus bloques secundarios se envían dentro de su plantilla a través del método getChildHtml ('block_name'). Ejemplo: core / text_list - Ejemplo de un bloque estructural. No utiliza plantillas; simplemente se utiliza para generar el contenido de todos sus bloques secundarios uno tras otro. núcleo / plantilla - Ejemplo de un bloque de contenido. La salida de este tipo de bloque depende de la plantilla asignada. Sus bloques secundarios se envían dentro de su plantilla a través del método getChildHtml ('block_name').

Se puede acceder a las instancias de bloque desde el controlador

Desde un controlador de acción:

```
$this->getLayout()->getBlock('head')->getTemplate();

/**
* Get specified tab grid
*/
public function gridOnlyAction()
{
$this->_initProduct();
$this->getResponse()->setBody(
$this->getLayout()->createBlock('adminhtml/catalog_product_edit_tab_' .
$this->getRequest()->gerParam('gridOnlyBlock')
)
->toHtml()
);
}
```

Lea Representación en línea: https://riptutorial.com/es/magento/topic/7140/representacion

Capítulo 30: Shell, CLI

Observaciones

Lo esencial

- Necesitas tener una línea de comando de Linux o conectarte usando SSH a tu servidor para usar scripts de shell.
- Ve a tu `MAGENTO_ROOT/shell`
- Script se puede ejecutar escribiendo, es decir

```
php -f indexer.php help
```

Métodos de shell del núcleo por archivos

1. abstract.php
2. indexer.php
3. compiler.php
4. log.php

Shell PHP personalizado scripts

A veces necesitamos acceder a Magento fuera de un navegador web para permitir los tiempos de ejecución o establecer diferentes cosas que no afecten la interfaz.

Hay 2 formas de iniciar Magento, pero solo una es Magento. Lea más arriba en la sección de ejemplos.

Examples

Usando shell sin extender Mage_Shell_Abstract

Bootstrapping Magento llamando:

```
require_once 'app/Mage.php';
Mage::app();
// Your code
```

Esta es la forma más sencilla pero no realmente la de Magento porque no estamos usando la clase que amplía `Mage_Shell_Abstract` : la clase que, cuando se amplía, nos proporciona herramientas para analizar los argumentos de la línea de comandos, llama a `__applyPhpVariables()` en su constructor (función analiza .htaccess archivos y aplica la configuración de php a shell script).

Usando shell de la manera Magento - extienda Mage_Shell_Abstract

Camino magento

El archivo reside en `shell/custom.php`

```
<?php
require_once' abstract.php';

class Stackoverflow_Shell_Custom extends Mage_Shell_Abstract
{

protected $_argname = array();

    public function __construct() {
        parent::__construct();

        // Time limit to infinity
        set_time_limit(0);

        // Get command line argument named "argname"
        // Accepts multiple values (comma separated)
        if($this->getArg('argname')) {
            $this->_argname = array_merge(
                $this->_argname,
                array_map(
                    'trim',
                    explode(',', $this->getArg('argname'))
                )
            );
        }
    }

 // Shell script point of entry
    public function run() {

    }

    // Usage help
    public function usageHelp()
    {
        return <<<USAGE
Usage:  php -f scriptname.php -- [options]

  --argname <argvalue>       Argument description

  help                   This help

USAGE;
    }
}
```

```
// Instantiate
$shell = new Stackoverflow_Shell_Custom();

// Initiate script
$shell->run();

}
```

Realizando Reindex de CLI

Ver el estado de:

```
php indexer.php status
```

Reindexar todo

```
php indexer.php reindexall
```

Índice específico de reindexación

```
php indexer.php --reindex CODE (see list below)
```

Lista de códigos individuales

Índice	Código
Atributos del producto	catalog_product_attribute
Precios de productos	catalog_product_price
URL de catálogo reescritas	catalog_url
Datos planos del producto	catalog_product_flat
Datos planos de la categoría	catalog_category_flat
Categoría Productos	catalog_category_product
Índice de búsqueda del catálogo	catalogsearch_fulltext
Estado de Stock	cataloginventory_stock

Lea Shell, CLI en línea: https://riptutorial.com/es/magento/topic/5387/shell--cli

Capítulo 31: Sql script para eliminar datos de prueba

Introducción

Sql script para eliminar datos de prueba de productos, clientes, registros y ventas.

Examples

Eliminar datos de prueba del cliente

```
SET FOREIGN_KEY_CHECKS=0;

-- Customers
TRUNCATE `customer_address_entity`;
TRUNCATE `customer_address_entity_datetime`;
TRUNCATE `customer_address_entity_decimal`;
TRUNCATE `customer_address_entity_int`;
TRUNCATE `customer_address_entity_text`;
TRUNCATE `customer_address_entity_varchar`;
TRUNCATE `customer_entity`;
TRUNCATE `customer_entity_datetime`;
TRUNCATE `customer_entity_decimal`;
TRUNCATE `customer_entity_int`;
TRUNCATE `customer_entity_text`;
TRUNCATE `customer_entity_varchar`;
ALTER TABLE `customer_address_entity` AUTO_INCREMENT=1;
ALTER TABLE `customer_address_entity_datetime` AUTO_INCREMENT=1;
ALTER TABLE `customer_address_entity_decimal` AUTO_INCREMENT=1;
ALTER TABLE `customer_address_entity_int` AUTO_INCREMENT=1;
ALTER TABLE `customer_address_entity_text` AUTO_INCREMENT=1;
ALTER TABLE `customer_address_entity_varchar` AUTO_INCREMENT=1;
ALTER TABLE `customer_entity` AUTO_INCREMENT=1;
ALTER TABLE `customer_entity_datetime` AUTO_INCREMENT=1;
ALTER TABLE `customer_entity_decimal` AUTO_INCREMENT=1;
ALTER TABLE `customer_entity_int` AUTO_INCREMENT=1;
ALTER TABLE `customer_entity_text` AUTO_INCREMENT=1;
ALTER TABLE `customer_entity_varchar` AUTO_INCREMENT=1;

-- Search
TRUNCATE `catalogsearch_query`;
TRUNCATE `catalogsearch_fulltext`;
TRUNCATE `catalogsearch_result`;
ALTER TABLE `catalogsearch_query` AUTO_INCREMENT=1;
ALTER TABLE `catalogsearch_fulltext` AUTO_INCREMENT=1;
ALTER TABLE `catalogsearch_result` AUTO_INCREMENT=1;

-- Polls
TRUNCATE `poll`;
TRUNCATE `poll_answer`;
TRUNCATE `poll_store`;
TRUNCATE `poll_vote`;
ALTER TABLE `poll` AUTO_INCREMENT=1;
```

```
ALTER TABLE `poll_answer` AUTO_INCREMENT=1;
ALTER TABLE `poll_store` AUTO_INCREMENT=1;
ALTER TABLE `poll_vote` AUTO_INCREMENT=1;

-- Reports
TRUNCATE `report_viewed_product_index`;
ALTER TABLE `report_viewed_product_index` AUTO_INCREMENT=1;

-- Newsletter
TRUNCATE `newsletter_queue`;
TRUNCATE `newsletter_queue_link`;
TRUNCATE `newsletter_subscriber`;
TRUNCATE `newsletter_problem`;
TRUNCATE `newsletter_queue_store_link`;
ALTER TABLE `newsletter_queue` AUTO_INCREMENT=1;
ALTER TABLE `newsletter_subscriber` AUTO_INCREMENT=1;
ALTER TABLE `newsletter_problem` AUTO_INCREMENT=1;
ALTER TABLE `newsletter_queue_store_link` AUTO_INCREMENT=1;

-- Wishlist
TRUNCATE `wishlist`;
ALTER TABLE `wishlist` AUTO_INCREMENT=1;

SET FOREIGN_KEY_CHECKS=1;
```

Eliminar datos de prueba del producto

```
SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE TABLE `catalog_product_bundle_option`;
TRUNCATE TABLE `catalog_product_bundle_option_value`;
TRUNCATE TABLE `catalog_product_bundle_selection`;
TRUNCATE TABLE `catalog_product_entity_datetime`;
TRUNCATE TABLE `catalog_product_entity_decimal`;
TRUNCATE TABLE `catalog_product_entity_gallery`;
TRUNCATE TABLE `catalog_product_entity_int`;
TRUNCATE TABLE `catalog_product_entity_media_gallery`;
TRUNCATE TABLE `catalog_product_entity_media_gallery_value`;
TRUNCATE TABLE `catalog_product_entity_text`;
TRUNCATE TABLE `catalog_product_entity_tier_price`;
TRUNCATE TABLE `catalog_product_entity_varchar`;
TRUNCATE TABLE `catalog_product_link`;
TRUNCATE TABLE `catalog_product_link_attribute`;
TRUNCATE TABLE `catalog_product_link_attribute_decimal`;
TRUNCATE TABLE `catalog_product_link_attribute_int`;
TRUNCATE TABLE `catalog_product_link_attribute_varchar`;
TRUNCATE TABLE `catalog_product_link_type`;
TRUNCATE TABLE `catalog_product_option`;
TRUNCATE TABLE `catalog_product_option_price`;
TRUNCATE TABLE `catalog_product_option_title`;
TRUNCATE TABLE `catalog_product_option_type_price`;
TRUNCATE TABLE `catalog_product_option_type_title`;
TRUNCATE TABLE `catalog_product_option_type_value`;
TRUNCATE TABLE `catalog_product_super_attribute`;
TRUNCATE TABLE `catalog_product_super_attribute_label`;
TRUNCATE TABLE `catalog_product_super_attribute_pricing`;
TRUNCATE TABLE `catalog_product_super_link`;
TRUNCATE TABLE `catalog_product_enabled_index`;
TRUNCATE TABLE `catalog_product_website`;
```

```
TRUNCATE TABLE `catalog_product_entity`;
TRUNCATE TABLE `cataloginventory_stock_item`;
TRUNCATE TABLE `cataloginventory_stock_status`;
TRUNCATE TABLE `catalog_category_entity`;
TRUNCATE TABLE `catalog_category_entity_datetime`;
TRUNCATE TABLE `catalog_category_entity_decimal`;
TRUNCATE TABLE `catalog_category_entity_int`;
TRUNCATE TABLE `catalog_category_entity_text`;
TRUNCATE TABLE `catalog_category_entity_varchar`;
TRUNCATE TABLE `catalog_category_product`;
TRUNCATE TABLE `catalog_category_product_index`;
TRUNCATE TABLE `catalog_product_relation`;
TRUNCATE TABLE `catalog_product_flat_1`;
TRUNCATE TABLE `catalog_category_flat_store_1`;
TRUNCATE TABLE `catalog_category_flat_store_2`;
TRUNCATE TABLE `catalog_category_flat_store_3`;

-- Tags
TRUNCATE `tag`;
TRUNCATE `tag_relation`;
TRUNCATE `tag_summary`;
ALTER TABLE `tag` AUTO_INCREMENT=1;
ALTER TABLE `tag_relation` AUTO_INCREMENT=1;
ALTER TABLE `tag_summary` AUTO_INCREMENT=1;

SET FOREIGN_KEY_CHECKS = 1;
```

Eliminar datos de prueba de ventas

```
SET FOREIGN_KEY_CHECKS=0;

TRUNCATE `sales_payment_transaction`;
TRUNCATE `sales_flat_creditmemo`;
TRUNCATE `sales_flat_creditmemo_comment`;
TRUNCATE `sales_flat_creditmemo_grid`;
TRUNCATE `sales_flat_creditmemo_item`;
TRUNCATE `sales_flat_order`;
TRUNCATE `sales_flat_order_address`;
TRUNCATE `sales_flat_order_grid`;
TRUNCATE `sales_flat_order_item`;
TRUNCATE `sales_flat_order_status_history`;
TRUNCATE `sales_flat_quote`;
TRUNCATE `sales_flat_quote_address`;
TRUNCATE `sales_flat_quote_address_item`;
TRUNCATE `sales_flat_quote_item`;
TRUNCATE `sales_flat_quote_item_option`;
TRUNCATE `sales_flat_order_payment`;
TRUNCATE `sales_flat_quote_payment`;
TRUNCATE `sales_flat_quote_shipping_rate`;
TRUNCATE `sales_flat_shipment`;
TRUNCATE `sales_flat_shipment_item`;
TRUNCATE `sales_flat_shipment_grid`;
TRUNCATE `sales_flat_shipment_track`;
TRUNCATE `sales_flat_shipment_comment`;
TRUNCATE `sales_flat_invoice`;
TRUNCATE `sales_flat_invoice_grid`;
TRUNCATE `sales_flat_invoice_item`;
TRUNCATE `sales_flat_invoice_comment`;
TRUNCATE `sales_order_tax`;
```

```
TRUNCATE `sales_order_tax_item`;

-- Reports
TRUNCATE `sales_bestsellers_aggregated_daily`;
TRUNCATE `sales_bestsellers_aggregated_monthly`;
TRUNCATE `sales_bestsellers_aggregated_yearly`;
TRUNCATE `sales_invoiced_aggregated`;
TRUNCATE `sales_invoiced_aggregated_order`;
TRUNCATE `sales_order_aggregated_created`;
TRUNCATE `sales_order_aggregated_updated`;
TRUNCATE `sales_refunded_aggregated`;
TRUNCATE `sales_refunded_aggregated_order`;
TRUNCATE `sales_shipping_aggregated`;
TRUNCATE `sales_shipping_aggregated_order`;
TRUNCATE `coupon_aggregated`;
TRUNCATE `review`;
TRUNCATE `review_detail`;
TRUNCATE `review_entity_summary`;
TRUNCATE `rating_store`;


ALTER TABLE `sales_payment_transaction` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_creditmemo` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_creditmemo_comment` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_creditmemo_grid` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_creditmemo_item` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_order` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_order_address` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_order_grid` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_order_item` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_order_status_history` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_quote` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_quote_address` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_quote_address_item` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_quote_item` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_quote_item_option` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_order_payment` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_quote_payment` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_quote_shipping_rate` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_shipment` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_shipment_item` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_shipment_track` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_shipment_comment` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_invoice` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_invoice_grid` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_invoice_item` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_invoice_comment` AUTO_INCREMENT=1;
ALTER TABLE `sales_flat_shipment_grid` AUTO_INCREMENT=1;
ALTER TABLE `sales_order_tax` AUTO_INCREMENT=1;
ALTER TABLE `sales_order_tax_item` AUTO_INCREMENT=1;
ALTER TABLE `sales_invoiced_aggregated` AUTO_INCREMENT=1;
ALTER TABLE `sales_invoiced_aggregated_order` AUTO_INCREMENT=1;

TRUNCATE `eav_entity_store`;
ALTER TABLE `eav_entity_store` AUTO_INCREMENT=1;

SET FOREIGN_KEY_CHECKS=1;
```

Eliminar registros de datos de prueba

```
SET FOREIGN_KEY_CHECKS=0;

TRUNCATE `log_customer`;
TRUNCATE `log_visitor`;
TRUNCATE `log_visitor_info`;
TRUNCATE `log_visitor_online`;
TRUNCATE `log_quote`;
TRUNCATE `log_summary`;
TRUNCATE `log_summary_type`;
TRUNCATE `log_url`;
TRUNCATE `log_url_info`;
TRUNCATE `sendfriend_log`;
TRUNCATE `report_event`;
TRUNCATE `dataflow_batch_import`;
TRUNCATE `dataflow_batch_export`;
TRUNCATE `index_process_event`;
TRUNCATE `index_event`;
ALTER TABLE `log_customer` AUTO_INCREMENT=1;
ALTER TABLE `log_visitor` AUTO_INCREMENT=1;
ALTER TABLE `log_visitor_info` AUTO_INCREMENT=1;
ALTER TABLE `log_visitor_online` AUTO_INCREMENT=1;
ALTER TABLE `log_quote` AUTO_INCREMENT=1;
ALTER TABLE `log_summary` AUTO_INCREMENT=1;
ALTER TABLE `log_url_info` AUTO_INCREMENT=1;
ALTER TABLE `sendfriend_log` AUTO_INCREMENT=1;
ALTER TABLE `report_event` AUTO_INCREMENT=1;
ALTER TABLE `dataflow_batch_import` AUTO_INCREMENT=1;
ALTER TABLE `dataflow_batch_export` AUTO_INCREMENT=1;
ALTER TABLE `index_event` AUTO_INCREMENT=1;

SET FOREIGN_KEY_CHECKS=1;
```

Lea Sql script para eliminar datos de prueba en línea:
https://riptutorial.com/es/magento/topic/9263/sql-script-para-eliminar-datos-de-prueba

Capítulo 32: Url actual

Sintaxis

- `$this->helper('core/url')->getCurrentUrl();`

Examples

Página principal

volver: http://www.example.com/

Página del producto

volver: http://www.example.com/my-product.html

Compruebe si la URL actual es segura

```
$isSecure = Mage::app()->getStore()->isCurrentlySecure();
```

Esto devolverá verdadero si la URL actual es segura.

Lea Url actual en línea: https://riptutorial.com/es/magento/topic/2150/url-actual

Capítulo 33: URLs específicas

Examples

Carro url

```
$this->helper('checkout/url')->getCartUrl();
```

O

```
Mage::helper('checkout/url')->getCartUrl();
```

URL de salida

```
$this->helper('checkout/url')->getCheckoutUrl();
```

O

```
Mage::helper('checkout/url')->getCheckoutUrl();
```

URL de inicio de sesión

```
$this->helper('customer/data')->getLoginUrl();
```

O

Mage :: helper ('customer / data') -> getLoginUrl ();

URL de cierre de sesión

```
$this->helper('customer/data')->getLogoutUrl();
```

O

```
Mage::helper('customer/data')->getLogoutUrl();
```

Olvidé la contraseña url

```
$this->helper('customer/data')->getForgotPasswordUrl();
```

O

```
Mage::helper('customer/data')->getForgotPasswordUrl();
```

Cuenta del cliente url

```
$this->helper('customer/data')->getAccountUrl();
```

O

```
Mage::helper('customer/data')->getAccountUrl();
```

Media, JS, URL de la piel

Para recuperar la ruta de la URL en BLOQUE ESTÁTICO o en páginas CMS

Para obtener la URL de SKIN

```
{{skin url='images/sampleimage.jpg'}}
```

Para obtener la URL de los medios

```
{{media url='/sampleimage.jpg'}}
```

Para obtener la URL de la tienda

```
{{store url='mypage.html'}}
```

Para obtener la URL base

```
{{base url="}}
```

Para recuperar la ruta de la URL en PHTML

URL de piel no segura

```
<?php echo $this->getSkinUrl('images/sampleimage.jpg') ?>
```

URL de piel segura

```
<?php echo $this->getSkinUrl('images/ sampleimage.gif',array('_secure'=>true)) ?>
```

Obtener URL actual

```
<?php $current_url = Mage::helper('core/url')->getCurrentUrl();?>
```

Obtener URL de inicio

```
<?php $home_url = Mage::helper('core/url')->getHomeUrl();?>
```

Obtener Magento Media Url

```
<?php Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_LINK);?>
<?php Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_MEDIA);?>
```

Obtener Magento Skin Url

```
<?php Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_SKIN);?>
```

Obtener Magento Store Url

```
<?php Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_WEB);?>
```

Obtener Magento Js Url

```
<?php Mage::getBaseUrl(Mage_Core_Model_Store::URL_TYPE_JS);?>
```

Lea URLs específicas en línea: https://riptutorial.com/es/magento/topic/2144/urls-especificas

Creditos

S. No	Capítulos	Contributors
1	Empezando con Magento	7ochem, Chris Rogers, Community, Gabriel Somoza, goutam, Henry's Cat, Luke Rodgers, Marek Skiba, Pawel Dubiel, RamenChef, Robbie Averill, Stephen Leppik
2	Agregue un precio diferente para la tienda múltiple usando Magento SOAP API	Harsha Sampath
3	Atributos personalizados	Twinkal
4	Ayudantes	Robbie Averill
5	Colección completa de productos	Twinkal
6	Colecciones	gebrial, gulshan maurya
7	Cómo filtrar colecciones	wesleywmd
8	Crear tarjetas de regalo empresariales mediante programación	JPMC, RamenChef
9	Datos de tienda y sitio web	Twinkal
10	EAV (valor de atributo de entidad)	lalit mohan
11	Entender los tipos de productos	lalit mohan
12	Estructura del módulo	Chris Rogers, RamenChef
13	Estructura MVC	lalit mohan
14	Hoja de trucos para	Chris Richardson

	tareas rápidas	
15	Imagen del producto Urls	Twinkal
16	Iniciar sesión para archivar	Akif, gulshan maurya, versedi
17	Magento Caching	RamenChef, Robbie Averill, Yogendra - eCommerce Developer
18	Manejo de errores magento, mensajes e informes.	7ochem, Charles, Pankaj Pareek, RamenChef
19	Modelo	Rickert
20	Obtener el nombre de la categoría de la página del producto	user2925795
21	Obtener productos de la base de datos	Akif, Alex, baoutch, bpoiss, ctrimm, gulshan maurya, jignesh prajapati, mnoronha, Olavi Sau, pce, SH-
22	Obtener usuario actual	bpoiss, Rickert
23	Obtenga el nombre de la tienda y otros detalles de la configuración del sistema	Henry's Cat, Robbie Averill
24	Obteniendo URLs de Magento	Nolwennig, Robbie Averill, Steven Church
25	Optimizando Magento Para Velocidad	Lemon Kazi
26	Pedidos	bpoiss, Hemant Sankhla, Luke Rodgers
27	Representación	lalit mohan
28	Shell, CLI	djdy, versedi
29	Sql script para eliminar datos de prueba	Twinkal
30	Url actual	Nolwennig, Twinkal

31	URLs específicas	Mayank Pandeyz, Nolwennig, Qaisar Satti, Yogendra - eCommerce Developer