eBook Gratuit

APPRENEZ magento

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

#magento

Table des matières

À propos

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.

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

Chapitre 1: Commencer avec magento

Remarques

Magento est une plate-forme de commerce électronique open source écrite en PHP; une plate-forme de commerce électronique hautement personnalisable et un système de gestion de contenu pouvant être utilisé pour créer des magasins en ligne destinés à la vente de marchandises.

Il fournit des fonctionnalités de commerce électronique courantes, telles que les paniers d'achat et la gestion des stocks, et encourage une personnalisation poussée pour répondre aux objectifs spécifiques de l'organisation.

Magento est également un framework PHP orienté objet qui peut être utilisé pour développer des applications Web modernes et dynamiques qui exploitent les fonctionnalités de commerce électronique de Magento.

Les principales caractéristiques de la plate-forme Magento sont:

- extensibilité
- évolutivité
- la flexibilité
- possibilité de personnalisation
- Open source

Versions

Edition communautaire

Version	Date de sortie
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

Version	Date de sortie
1.0	2008-03-31

Edition pour entreprise

Version	Date de sortie
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

Installation et configuration

Conditions préalables et exigences pour Magento Community Edition 1.9

Hébergement

- Apache 2.x (avec mod_rewrite) ou Nginx 1.7.x
- En raison des exigences du traitement des opérations Magento, il est recommandé d'installer Magento sur un serveur avec au moins 2 Go de RAM. Cela garantira que tous les logiciels impliqués dans la gestion du magasin auront suffisamment de mémoire pour fonctionner.
- Possibilité d'exécuter des tâches planifiées (crontab) avec PHP 5.

- Possibilité de remplacer les options dans les fichiers .htaccess.

PHP

- PHP 5.4, PHP 5.5
- Extensions requises: PDO_MySQL, simplexml, mcrypt, hash, GD, DOM, iconv, curl, SOAP (pour API Webservices)
- memory_limit pas moins de 256 Mo (512 Mo recommandés)

Base de données

- MySQL 5.6 (Oracle, Percona, MariaDB)

SSL

- Un certificat de sécurité valide est requis pour HTTPS.
- Les certificats SSL auto-signés ne sont pas pris en charge

Installation:

Télécharger et configurer les fichiers Magento

Nous utilisons openMage mirror comme téléchargement direct pour la 1.9.2.4 La branche est désactivée et le site web de magento nécessite un compte. Mais vous êtes encouragé à télécharger une copie depuis 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
```

Créer une base de données MySQL et un utilisateur

accéder à la console mysql

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

dans la console 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
```

Terminez l'installation via l'interface Web

Pour accéder à l'interface Web avec votre navigateur, accédez au nom de domaine ou à l'adresse IP publique de votre serveur:

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

Suivez ensuite les instructions à l'écran

Dépannage des problèmes courants

Seule la page d'accueil fonctionne, toutes les autres pages renvoient 404

Assurez-vous que le module `mod_rewrite` a été installé sur Apache et a été activé pour le chargement. Voir l' **étape 2** pour savoir comment procéder ici: https://www.digitalocean.com/community/tutorials/how-to-set-up-mod_rewrite-for-apache-on-ubuntu-14-04

Assurez-vous que vous autorisez les modifications dans le fichier .htaccess en les activant dans votre site conf. Voir l' **étape 3** : https://www.digitalocean.com/community/tutorials/how-to-set-up-mod_rewrite-for-apache-on-ubuntu-14-04

Votre fichier `.htaccess` peut-être mal configuré ou manquant: rendez-vous sur la page de téléchargement de Magento: https://www.magentocommerce.com/download - téléchargez la version appropriée et extrayez le fichier .htaccess à placer dans votre racine d'installation Magento.

Le site fonctionne mais pas les styles ni les scripts ne se chargent

Assurez-vous d'avoir défini les autorisations et la propriété appropriées: **Voir ici** pour plus d'informations - http://devdocs.magento.com/guides/m1x/install/installer-privileges_before.html

Solution commune: essayez de réindexer et de vider le cache manuellement (au cas où l'administrateur serait trop difficile à naviguer). Réindexer via la ligne de commande: https://www.atwix.com/magento/process-magento-indexes-from-command-line/ Flush cache (via admin ou ligne de commande): https://www.properhost.com/ support / kb / 23 / How-To-Clear-Le-Magento-Cache

Suivi

Lire Commencer avec magento en ligne: https://riptutorial.com/fr/magento/topic/812/commencer-avec-magento

Chapitre 2: Ajouter un prix différent pour plusieurs magasins à l'aide de l'API SOAP Magento

Examples

Magento SOAP V1

Vous devez modifier le périmètre de prix **'Global' pour 'site'** (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>";
```

Lire Ajouter un prix différent pour plusieurs magasins à l'aide de l'API SOAP Magento en ligne: https://riptutorial.com/fr/magento/topic/7410/ajouter-un-prix-different-pour-plusieurs-magasins-a-l-aide-de-l-api-soap-magento

Chapitre 3: Attributs personnalisés

Introduction

Attributs personnalisés pour les ventes, la catégorie, etc.

Examples

Attribut de vente

Attribut personnalisé dans les tables liées aux ventes comme: sales_flat_quote, sales_flat_order_item, sales_flat_order, etc.

Dans votre fichier d'installation 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();
 ?>
```

Une autre façon de le faire est:

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

Veillez à effacer le cache après cela.

Lire Attributs personnalisés en ligne: https://riptutorial.com/fr/magento/topic/9267/attributs-personnalises

Chapitre 4: Collection complète de produits

Introduction

Filtres complets pour la collecte de produits en fonction de la visibilité, du magasin et de la salle d'opération, de l'état du stock, de l'état, etc.

Examples

Filtrage de la collection de produits

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

Filtre basé sur le magasin:

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

Filtre basé sur le type de produit:

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

Filtre basé sur le statut:

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

Filtrer en utilisant null et notnull:

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

Filtrer en utilisant plus que et moins que:

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

Filtre en utilisant plus grand que et égal à:

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

Filtrer en utilisant moins que et égal à:

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

Filtrer en utilisant et non en:

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

Filtrer les produits par une plage d'identifiant d'entité:

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

Filtrer en fonction de la visibilité du produit:

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

Filtrer en utilisant comme et pas comme:

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

Filtre en utilisant égal à et non égal à:

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

Filtrer dans les produits en stock:

```
$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')
```

Définir l'ordre par:

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

Définir la taille de la page:

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

Lire Collection complète de produits en ligne:
https://riptutorial.com/fr/magento/topic/9261/collection-complete-de-produits

Chapitre 5: Collections

Examples

Obtenir des collections de modèles

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

Obtenir des attributs d'objet supplémentaires

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

Filtration

```
// 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));
```

Tri

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

Accès

```
// 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();
```

Obtenir un objet de collection

```
    // 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');
```

Lire Collections en ligne: https://riptutorial.com/fr/magento/topic/5937/collections

Chapitre 6: Collections

Examples

Collection de produits

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

Sélection de l'attribut spécifique

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

Sélection de l'attribut All

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

Ajouter un filtre dans la collection

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

Ajouter un filtre d'attribut de produit dans la collection

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

Set Order

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

Définir la limite

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

Définir la page actuelle

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

Catégorie Collection d'un magasin spécifique et d'un niveau spécifique

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

Lire Collections en ligne: https://riptutorial.com/fr/magento/topic/6763/collections

Chapitre 7: Collections

Examples

Collection de produits

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

Sélection de l'attribut spécifique

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

Sélectionner tous les attributs

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

Ajouter un filtre dans la collection

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

Set Order

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

Définir la limite

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

Définir la page actuelle

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

Lire Collections en ligne: https://riptutorial.com/fr/magento/topic/7206/collections

Chapitre 8: Comment filtrer les collections

Paramètres

Paramètre	Détails
$ addFieldToFilter (**$ field** , $ condition = null)	{string} Le champ que nous ajoutons au filtre.
$ addFieldToFilter ($ field, **$ condition = null**)	{mixed} La définition du filtre que nous allons utiliser.
addAttributeToFilter (**$ attr** , $ condition = null, $ join = 'inner')	{string} Le champ que nous ajoutons au filtre.
addAttributeToFilter ($ attr, **$ condition = null** , $ join = 'inner')	{mixed} La définition du filtre que nous allons utiliser.
addAttributeToFilter ($ attr, $ condition = null, **$ join = 'inner'**)	{('inner', 'left')} Le type de jointure SQL à utiliser pour rejoindre la table EAV.

Remarques

Arguments de comparaison des filtres

Magento offre également un moyen flexible de filtrage en utilisant également des opérateurs de comparaison. Voici une liste des opérateurs valides et leur syntaxe:

Tous les arguments de comparaison peuvent être transmis au second paramètre des `addFieldToFielter()` ou `addAttributeToFilter()` .

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

Comparaison	Tableau d'argument	Extrait SQL résultant
Équivaut à	array ("eq" => $ var)	WHERE (`my_field` = $ var)
Inégal	array ("neq" => $ var)	WHERE (`my_field`! = $ Var)
Comme	array ("like" => $ var)	WHERE (`my_field` LIKE $ var)
Pas comme	array ("nlike" => $ var)	WHERE (`my_field` NOT LIKE $ var)
Est	array ("is" => $ var)	WHERE (`my_field` IS $ var)
Dans	array ("in" => $ var)	WHERE (`my_field` IN ($ var))

Comparaison	Tableau d'argument	Extrait SQL résultant
Pas dedans	array ("nin" => $ var)	WHERE (`my_field` NOT IN ($ var))
Nul	array ("null" => true)	WHERE (`my_field` IS NULL)
Pas nul	array ("notnull" => true)	WHERE (`my_field` N'EST PAS NULL)
Plus grand que	array ("gt" => $ var)	WHERE (`my_field`> $ var)
Moins que	array ("lt" => $ var)	WHERE (`my_field` <$ var)
Meilleur que ou égal	array ("gteq" => $ var)	WHERE (`my_field`> = $ var)
Moins que ou égal	array ("lteq" => $ var)	WHERE (`my_field` <= $ var)
Trouver dans l'ensemble	array ("finset" => array ($ var))	WHERE (find_in_set ($ var, `my_field`)
De et à	array ("from" => $ var1, "to" => $ var2)	WHERE (`my_field`> = $ var1 ET` my_field` <= $ var2)

Examples

Collections de filtrage

Magento dispose d'un puissant ensemble de méthodes pour filtrer les collections. Comme il existe deux types d'objets pouvant être contenus dans des collections, nous devons d'abord déterminer le type de données avec lequel nous travaillons avant de pouvoir le filtrer. Magento implémente un modèle de données EAV pour des entités telles que des produits et des catégories. Il existe un ensemble différent de méthodes à utiliser si nous filtrons une collection d'objets EAV.

Dans Magento, les commandes ne sont pas stockées en tant qu'objets EAV. Cela fait de la collection de commandes un bon exemple pour filtrer une collection de base.

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

Si nous examinons la collection de produits, nous pouvons voir que les produits sont stockés dans un modèle de données EAV. Nous pouvons facilement filtrer par attributs EAV.

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

Gestion des AND et des OR dans les filtres

Lorsque nous interrogeons nos données, nous avons souvent besoin de plusieurs filtres pour obtenir le jeu de données exact que nous recherchons. En SQL, nous traitons cela avec les

clauses AND et OR. Nous pouvons réaliser la même chose avec les collections.

Pour ajouter une clause AND à votre requête, ajoutez simplement un autre appel de méthode. Cela ajoutera le second filtre à l'instruction WHERE originale en la joignant à un ET.

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

La clause WHERE résultante ressemblera à ceci:

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

Maintenant, disons que nous voulons que tous les skus qui commencent par 'a' OU se terminent par 'b'. Comment ajoutons-nous une clause OR? Grâce aux collections de Magento, c'est assez simple. Nous ajoutons le filtre comme deuxième élément du tableau de filtres.

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

Maintenant, la clause WHERE qui en résulte ressemblera à ceci:

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

Lire Comment filtrer les collections en ligne: https://riptutorial.com/fr/magento/topic/5849/comment-filtrer-les-collections

Chapitre 9: Comprendre les types de produits

Remarques

Il existe six types de produits différents intégrés à Magento.

- Simple

Une unité de stock unique

- Configurable

Premier des produits composites. Autoriser les clients à configurer leur produit et ajouter un seul produit simple au panier.

- Groupé

Le deuxième produit composite, un produit groupé, associe des produits simples et permet aux clients de choisir des quantités de chaque article.

- Paquet

Troisième type de produit composite, un bundle relie des produits simples à un achat unique.

- Virtuel

Aucun article physique requis pour la livraison, p.ex. services

- Téléchargeable

Un produit numérique plutôt que physique. La plupart des types de produits sont implémentés dans le module Mage_Catalog, à l'exception de Mage_Bundle et Mage_Downloadable.

Les produits groupés, groupés et configurables implémentent une relation parent-enfant dans laquelle un certain nombre d'autres produits (par défaut, simples, virtuels ou téléchargeables) sont affectés à un produit principal. Cela traite ensuite les données de produit pour l'ensemble de la collection (par exemple, nom du groupe, du lot ou du produit configurable, prix et statut).

Les produits téléchargeables et regroupés ont des tables supplémentaires dans la base de données, tandis que le reste est partagé entre tous les autres types de produits. Les produits configurables ont une table supplémentaire à relier aux produits enfants, catalog_product_super_link.

Type de produit personnalisé

Pour créer un type de produit qui étend l'un des types de produits intégrés, le modèle de type de produit correspondant doit être étendu. Sinon, le nouveau type de produit devrait étendre la classe Mage_Catalog_Model_Product_Type_Abstract.

Une entrée dans le fichier config.xml du module est également requise:

Des produits plus compliqués peuvent nécessiter d'autres domaines personnalisés tels que le modèle de prix et la récupération des données d'index.

Calcul du prix

Lorsqu'il s'agit d'un seul produit, le prix est toujours calculé à la volée. L'attribut prix EAV est chargé avec le produit et le prix final est calculé par le modèle de prix, Mage_Catalog_Model_Product_Type_Price.

Certains types de produits le traitent différemment. Dans ce cas, ils étendent cette classe et implémentent leur propre logique. Par exemple, le produit configurable écrase getFinalPrice () et ajoute une logique supplémentaire. Ce modèle personnalisé peut ensuite être spécifié dans config.xml avec une balise <price_model>.

Les collections de produits, cependant, utilisent l'indice de prix pour récupérer les prix pré-calculés, éliminant ainsi le besoin de le calculer pour chaque produit.

Le prix final peut être ajusté par les observateurs de l'événement catalog_product_get_final_price. Par défaut, seul le module Mage_CatalogRule observe cet événement.

Une autre méthode pour remplacer le prix des produits consiste simplement à la définir sur le produit. Si le prix est défini, le produit ne le recalculera pas.

Le prix du produit est distinct du prix normal (bien que pris en compte lors du calcul du prix). Il est implémenté sous forme de liste avec un groupe de clients et des qualificateurs de quantité minimum pour chaque niveau. Les prix de niveau sont affichés dans un tableau à l'aide du modèle catalog / product / view / tierprices.phtml.

Les options de produits personnalisés sont traitées lors du calcul du prix final. Chaque option a son propre prix défini, qui s'ajoute au prix final.

Les prix de groupe, de niveau et spéciaux sont tous pris en compte en même temps `($priceModel->getBasePrice())` et le plus petit des trois (ou quatre si vous incluez le prix normal) est choisi comme prix de base du produit.

Impôt

Le module Mage_Tax utilise la classe de taxe d'un produit et indique si le prix du produit est inclus ou non, afin d'identifier le bon taux à appliquer.

Les facteurs suivants sont utilisés pour calculer la taxe sur les produits:

- Classe de taxe sur les produits
- Le montant de la taxe déjà inclus
- Adresses de facturation et d'expédition
- Classe de taxe client
- Paramètres de magasin

Navigation en couches

Les classes responsables du rendu de la navigation en couches sont les suivantes:

- Mage_Catalog_Block_Layer_View

  -Porte les filtres et les options

- Mage_Catalog_Block_Layer_State

-Contrôle ce qui est actuellement filtré par

Pour implémenter la navigation en couches sur des attributs avec des modèles sources personnalisés, la méthode Mage_Catalog_Model_Layer_Filter_Abstract :: apply () doit être remplacée pour déterminer la manière dont la collection de produits doit être filtrée.

La navigation en couches est rendue par les blocs Mage_Catalog_Block_Layer_View et Mage_Catalog_Block_Layer_State, qui utilisent des blocs de filtrage pour des filtres individuels.

La navigation par couches utilise la table d'index pour la plupart des filtres, par exemple le prix, l'indice d'attribut de produit, l'index de produit décimal.

Catégories

Catégories dans la base de données

La hiérarchie des catégories est gérée en stockant l'ID parent d'une catégorie. La hiérarchie complète est affichée dans la colonne du chemin (identifiants séparés par une barre oblique). Il y a une catégorie spéciale avec parent_id de 0. C'est la vraie catégorie racine et chacune des autres catégories racine telles que définies dans Magento l'utilisent comme parent partagé.

Pour lire et gérer une arborescence de catégories à partir de la base de données, deux classes différentes sont utilisées selon que le catalogue à plat est activé,
Mage_Catalog_Model_Resource_Category_Tree et
Mage_Catalog_Model_Resource_Category_Flat.

L'avantage des catégories plates est qu'il est plus rapide d'interroger. Cependant, il doit être reconstruit à partir des tables EAV chaque fois qu'il ya un changement.

```
getChildren()
```

renvoie une chaîne séparée par des virgules d'ID enfants immédiats

```
getAllChildren()
```

renvoie une chaîne ou un tableau de tous les ID enfants

```
getChildrenCategories()
```

renvoie une collection de catégories d'enfants immédiates. NB: Si le catalogue à plat est activé, les seules catégories enfants renvoyées seront celles avec include_in_menu = 1. Dans les deux cas, seules les catégories actives sont renvoyées.

Règles de prix catalogue

Les règles de prix du catalogue appliquent des remises aux produits en fonction de la date, du produit, du site Web et du groupe de clients.

Lorsque `getFinalPrice()` est appelé sur un produit, l'événement catalog_product_get_final_price est déclenché. Ceci est observé par Mage_CatalogRule_Model_Observer qui recherchera alors toute règle de prix catalogue applicable au produit. S'il y a lieu, il examine ensuite la table de prix de la base de données et inscrit le prix dans le modèle de produit en tant que champ de données Varien final_price.

Dans la base de données, la table catalogrule décrit les règles, leurs conditions et leurs actions. catalogrule_product contient les produits correspondants et des informations sur les règles. Pendant ce temps catalogrule_product_price contient le prix après application de la règle.

Tables d'indexation et plates

Les tables de catalogue plates sont gérées par des indexeurs de catalogue. Si la reconstruction automatique des index est activée, les indexeurs de catalogue sont reconstruits chaque fois qu'un produit, une catégorie ou toute entité associée est mise à jour. La méthode _afterSave () appelle le processus de l'indexeur. Sinon, ils doivent être réindexés manuellement via admin.

Le type de produit affecte l'index des prix et l'index boursier dans lesquels les produits peuvent définir leurs propres indexeurs personnalisés (dans config.xml) pour gérer leurs données pour ces index.

Le module Mage_Index fournit la structure avec laquelle des index personnalisés peuvent être créés pour optimiser les performances du site. La classe Mage_Index_Model_Indexer_Abstract doit être étendue pour créer un nouvel index, en implémentant les méthodes _registerEvent () et _processEvent (). Sans oublier de l'enregistrer dans 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;
    }
}
```

Décrire les types de produits standard (simple, configurable, groupé)

Simple

Le type Simple Products doit être utilisé pour ceux qui ont généralement une seule configuration (one-size-fits-all). Cela pourrait inclure des éléments tels que:

- Une boîte de crayons, petite (24 couleurs)
- Une boîte de crayons, grande (64 couleurs)
- Moniteur d'ordinateur HD 26 po SuperHighTech
- Figurine Barrack Obama (6 ”)

Groupé

Les produits groupés vous permettent de créer un nouveau produit en utilisant un ou plusieurs produits existants dans votre magasin. Par exemple, supposons que vous ayez déjà dans votre magasin une «Figurine d'action Barrack Obama» et une «Figurine d'action George W Bush» et que vous vouliez les vendre en groupe. Vous devez simplement créer un nouveau produit groupé (appelons-le «Obama + Bush (Obtenir les deux et dépenser deux fois plus!)», Puis ajoutez les deux chiffres d'action au groupe via l'onglet «Produits associés».

Remarque: Malheureusement, vous ne pouvez pas définir un prix spécial «groupe» directement à partir de la page du produit. Pour offrir un rabais sur les achats groupés, vous devez créer une nouvelle règle sur les prix du panier.

Configurable

Produit configurable: ce produit permet à vos clients de sélectionner la variante de leur choix en choisissant des options. Par exemple, vous pouvez vendre des T-shirts en deux couleurs et trois tailles. Vous créez six produits simples en tant que produits individuels (chacun avec ses propres références), puis vous les ajoutez à un produit configurable où les clients peuvent choisir la taille et la couleur, puis les ajouter à leur panier. Des fonctionnalités très similaires sont possibles en utilisant les options personnalisées pour les produits simples. La différence entre un produit configurable et un produit, y compris les options personnalisées, est que l’inventaire n’est ni vérifié ni mis à jour pour les options individuelles lors de l’achat des options personnalisées.

Virtuel

Les produits virtuels sont ceux qui n'ont pas de contrepartie physique ou numérique. Ils ne sont pas expédiés, et ils n’ont pas de lien de téléchargement. Ce type de produit peut être utilisé pour des services tels que:

- Nettoyage de la maison
- Abonnement de 1 an à la newsletter

Remarque: Si vous utilisez des produits virtuels pour les «abonnements», il est important de noter qu’il n’existe aucun moyen intégré de gérer le renouvellement automatique des abonnements. Tous les achats effectués dans Magento, quel que soit le type de produit, sont des achats ponctuels.

Paquet

Ce type de produit est également appelé «kit» dans d’autres logiciels de commerce électronique. Ce type de produit est idéal dans les cas où l'utilisateur doit sélectionner un certain nombre d'options configurables, mais au moins une option. Cela pourrait inclure des produits comme:

- Systèmes informatiques personnalisables
- Smokings / costumes personnalisables
- Cliquez ici pour un tutoriel vidéo sur l'utilisation de bundles

Téléchargeable

Les produits téléchargeables sont similaires aux produits virtuels, sauf qu'ils permettent d'ajouter

un ou plusieurs fichiers numériques au téléchargement. Les fichiers peuvent être téléchargés via l'interface d'administration ou en téléchargeant directement sur le serveur via FTP, puis ajoutés par URL. Lorsqu'un client achète un produit téléchargeable, Magento génère un lien sécurisé et crypté (afin que les clients ne puissent pas voir l'emplacement réel du fichier) pour que ce client télécharge son fichier.

Cette catégorie peut inclure des produits tels que:

- Musique / MP3
- Logiciel

Remarque : Si SSL est activé pour votre site, les téléchargements peuvent échouer sous toutes les versions d'IE, car IE contient un bogue qui empêche le téléchargement via des connexions sécurisées si l'en-tête no-cache est défini. Cela peut être facilement corrigé dans un fichier htaccess en supprimant les en-têtes no-cache et no-store, ou en forçant les liens de téléchargement à ne plus être sécurisés.

Lire Comprendre les types de produits en ligne:
https://riptutorial.com/fr/magento/topic/7142/comprendre-les-types-de-produits

Chapitre 10: Connexion au fichier

Syntaxe

- journal des fonctions statiques publiques ($ message, $ level = null, $ file = '', $ forceLog = false)

Paramètres

Paramètre	Détails
string $ message	Le message qui sera enregistré
niveau entier $	Niveau de journalisation
string $ file	Chemin et nom avec l'extension du fichier qui sera enregistré dans `var/log/` . Si NULL ou non spécifié, alors `system.log` sera utilisé.
bool $ forceLog	Si la valeur est définie sur `TRUE` journal sera écrit même si le mode développeur est désactivé et que la journalisation est inactive.

Remarques

La journalisation est désactivée par défaut, sauf si le mode développeur est actif.

Toutes les exceptions sont consignées dans `exceptions.log` peu importe si la journalisation est activée dans la configuration.

La journalisation peut être activée en vous connectant à Magento Admin et en procédant comme suit:

- Système> Configuration (barre supérieure)
- Développeur (menu de gauche)
- Section Paramètres de journal
- Sélectionnez Oui dans la liste déroulante `Enabled` .
- Enregistrer la configuration dans le coin supérieur droit.

Type de variable de message

Même si la documentation définit ce message comme étant une chaîne, si un tableau est passé, il y a un bloc de code dans cette méthode pour en prendre soin avec `print_r` :

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

Niveau de journalisation

Si le paramètre level est défini sur null, le niveau DEBUG est pris.

`$level = is_null($level) ? Zend_Log::DEBUG : $level;` Les niveaux sont déclarés dans le fichier: `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
```

Les constantes sous la forme de `Zend_Log::INFO` ou un nombre entier dans la plage spécifiée ci-dessus peuvent être transmises en tant que paramètre de niveau de journalisation.

Examples

Fichier journal personnalisé

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

Ce wil se connecter à

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

Journalisation par défaut

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

Cela se connectera à `/var/log/system.log`

Les objets et les tableaux sont automatiquement écrits via une directive `print_r()` . Faites attention lorsque vous utilisez des objets, car ils peuvent avoir une taille importante.

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

Cela enregistrera la chaîne de trace d'exception dans `/var/log/exception.log`

Lire Connexion au fichier en ligne: https://riptutorial.com/fr/magento/topic/2363/connexion-au-fichier

Chapitre 11: Créer des cartes-cadeaux d'entreprise par programme

Examples

Créer une carte cadeau unique

```
// 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();
```

Lire Créer des cartes-cadeaux d'entreprise par programme en ligne: https://riptutorial.com/fr/magento/topic/2387/creer-des-cartes-cadeaux-d-entreprise-par-programme

Chapitre 12: Des aides

Examples

Créer un assistant

Les helpers devraient s'étendre de `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;
    }
}
```

Pour pouvoir y accéder via `Mage::helper` vous devez définir un alias d'aide dans un fichier `config.xml` pour permettre à l'autoloader de trouver votre classe:

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

En supposant que votre module soit correctement configuré et que vous avez effacé votre cache, vous devriez maintenant pouvoir utiliser votre assistant comme ceci:

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

Remarque: si vous utilisez une classe Data, son nom d'assistance est implicite si vous n'en spécifiez pas. Par exemple, les deux exemples suivants sont identiques:

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

Lire Des aides en ligne: https://riptutorial.com/fr/magento/topic/5904/des-aides

Chapitre 13: Données du magasin et du site Web

Introduction

Obtenir les données relatives à magento store et au site Web

Examples

Obtenir les données du magasin en cours

```
$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();
```

Lire Données du magasin et du site Web en ligne:
https://riptutorial.com/fr/magento/topic/9266/donnees-du-magasin-et-du-site-web

Chapitre 14: EAV (Entity Attribute Value)

Remarques

Entité

Stocke des informations sur le type des données stockées. Dans le cas de Magento, il s'agit du client, du produit, de la catégorie, etc.

Attribut

Les propriétés individuelles de chacune des entités, par exemple le nom, le poids, l'adresse e-mail, etc.

Valeur

La valeur d'une entité et d'un attribut donnés. Par exemple, nous pouvons spécifier l'entité client et l'attribut email, puis lui donner la valeur hello@example.com.

Schéma de base de données

eav_entity

La table d'entités.

eav_entity_attribute

La table attributaire

eav_entity_ {type}

Les tables de valeurs. Les types sont datetime, decimals, int, text et varchar.

Important: notez que la table eav_entity_varchar a le type varchar pour les valeurs même si la date ou le nombre entier convient mieux à la valeur.

Modèles versus modèles de ressources

Tous les modèles de Mage / Eav / Model / Resource sont Mysql4 et sont des modèles de ressources.

En outre Entity / Abstract.php et Entity / Setup.php.

Plat Versus EAV

Les modèles EAV sont plus complexes, fournissant une logique pour enregistrer et charger à partir de plusieurs tables, tandis que les modèles plats ou standard sont relativement simples (traditionnels).

Les modèles standard gèrent principalement leurs propriétés avec les régleurs de données et les getters travaillant avec une seule table. Les modèles EAV gèrent principalement leurs modèles d'attributs. Les modèles standard n'enregistrent que leurs données dans un tableau et les chargent. Les modèles EAV chargent tous les attributs (ou un ensemble spécifique) après avoir chargé les données de base et sauvegardé les attributs après avoir enregistré les données (y compris l'insertion, la mise à jour et la suppression des attributs).

Exemples de modèles de ressources EAV

Pour rechercher des entités à l'aide du schéma de stockage EAV, la recherche de la chaîne extend permet d'utiliser Mage_Eav_Model_Entity_Abstract. Cela devrait révéler tous les modèles de ressources basés sur la structure EAV. Cependant, la liste résultante inclura de nombreuses classes obsolètes du module Mage_Sales qui ne sont plus utilisées.

Les seuls modules contenant des entités utilisant le schéma de stockage EAV sont Mage_Catalog (catégories et produits) et Mage_Customer (clients et adresses).

Les groupes de clients utilisent le schéma de stockage de table plate. Toutes les entités de vente ont été converties en entités de table plate avec la sortie de Magento 1.4.

La raison pour laquelle EAV est utilisé est que les entités peuvent avoir un nombre indéterminé de propriétés et donc rester flexibles. Par exemple, lorsque vous ajoutez un nouvel attribut à une entité client (qui est une entité EAV), il n'est pas nécessaire de modifier la table de base de données pour que ce nouvel attribut soit ajouté.

Avantages

Le schéma de flexibilité de la base de données n'a pas besoin de changer avec le modèle Mise en œuvre rapide

Désavantages

- Inefficace

Une requête renvoyant 20 colonnes serait normalement composée de 20 auto-jointures dans EAV. Cependant, le module Mage_Eav n'utilise généralement pas de jointure pour charger les données de valeur d'attribut. Au lieu de cela, les sélections d'union sont utilisées. Les jointures ne sont utilisées que pour filtrer les collections EAV.

- Aucun mécanisme pour les relations entre les sous-types
- Pas de regroupement des sous-types d'entité

Portée du site et du magasin

Pour gérer les valeurs d'attribut de site Web et d'étendue de magasin dans EAV, une valeur store_id existe sur l'entité de catalogue afin d'afficher l'étendue qui renvoie à core_store. En plus des magasins normaux (vues de magasin), il existe également un magasin «0» qui correspond à la valeur globale. Sur un magasin particulier, le système vérifie d'abord une valeur d'entité sur le magasin actuel, puis se rabat sur l'entité globale. Les entités EAV Mage_Customer n'ont pas de

colonne de portée store_id.

Insérer, mettre à jour et supprimer

Pour déterminer si une insertion, une mise à jour ou une suppression doit être effectuée sur un attribut, une comparaison est effectuée avec l'objet d'origine. L'objet d'origine est essentiellement un doublon de l'objet de données lorsque l'entité a été extraite de la base de données.

- Si l'attribut existe à l'origine et que sa nouvelle valeur n'est pas vide; il met à jour
- Si l'attribut existe à l'origine, sa nouvelle valeur est définie sur vide; ça supprime. - Si l'attribut n'existe pas à l'origine et que sa nouvelle valeur n'est pas vide; il insère

Gestion des attributs

Modèles d'attribut

Modèle d'attribut *Représente l'attribut dans le formulaire de base de données, sa logique est standard pour tous les attributs et est difficile à modifier* .

Modèle frontend

L'interface de l'attribut à l'interface et fournit toute logique requise par l'attribut sur l'interface, par exemple la méthode getUrl () sur les images.

Modèle de backend

Celles-ci effectuent la validation sur l'attribut avant qu'il soit enregistré dans la base de données. Par exemple, le modèle de mot de passe de mot de passe convertit le mot de passe en un hachage avant qu'il soit enregistré. Il vérifie également que le mot de passe et la confirmation du mot de passe correspondent avant l'enregistrement.

Modèles source

Utilisé pour remplir les options disponibles pour un attribut, par exemple catalog / product_status a activé et désactivé.

Méthodes requises

Un modèle source nécessite:

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

Normalement, seul getAllOptions () doit être implémenté car une implémentation de getOptionText () existe déjà dans le modèle source abstrait Mage_Eav_Model_Entity_Attribute_Source_Abstract.

Un modèle frontal ne nécessite pas la méthode getValue ().

Un modèle backend nécessite:

```
<?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);
?>
```

Toutes ces méthodes sont implémentées dans le modèle backend abstrait Mage_Eav_Model_Entity_Attribute_Backend_Abstract. Pour les modèles de gestion personnalisés, seules les méthodes nécessitant une personnalisation doivent être remplacées.

Modèles source de configuration système

Ne peut pas être utilisé pour les attributs EAV. Les modèles de source EAV implémentent la méthode getAllOptions, tandis que les modèles source adminhtml implémentent la méthode toOptionArray ().

Les modèles de source de configuration système par défaut peuvent être trouvés dans Mage / Adminhtml / Model / System / Config / Source /.

Modèles source d'attribut

Le but des modèles de source d'attribut est de fournir la liste des options et des valeurs pour les attributs select et multiselect. Ils fournissent également les informations de colonne à l'indexeur de table plate du catalogue si nécessaire.

Pour obtenir une liste de toutes les options pour un attribut, procédez comme suit:

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

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

Modèles d'attributs par défaut

Si aucune classe n'est spécifiée en tant que frontend, backend ou - pour les attributs select ou multiselect - pour les modèles source, une classe par défaut est utilisée.

Le modèle frontal d'attribut par défaut est Mage_Eav_Model_Entity_Attribute_Frontend_Default.

Le modèle principal d'attribut par défaut dépend du code d'attribut et est déterminé dans la méthode 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 la méthode passe à la dernière ligne, Mage_Eav_Model_Entity_Attribute_Backend_Default est utilisé.

Le modèle source par défaut est défini dans Mage_Eav_Model_Entity_Attribute_Source_Table. Ceci est défini dans le modèle d'attribut des modules de catalogue. Le modèle source de configuration par défaut spécifié dans le module eav n'est jamais utilisé.

Ajouter un attribut

Pour ajouter des attributs EAV, utilisez Mage_Eav_Model_Entity_Setup en étendant la classe setup.

addAttribute () Crée les attributs, les ajoute aux groupes et aux ensembles (y compris les paramètres par défaut), ou les met à jour s'il existe déjà. updateAttribute () Met à jour les données d'attribut uniquement. Des classes d'installation personnalisées peuvent être utilisées pour étendre ces méthodes, ajouter des données supplémentaires ou simplifier les arguments nécessaires.

Tables plates

Les attributs de catalogue plat sont gérés par des indexeurs:

Mage_Catalog_Model_Resource_Product_Flat_Indexer :: updateAttribute () Mage_Catalog_Model_Resource_Category_Flat :: synchronize () Les attributs de produit sont ajoutés à la table plate s'ils existent (voir Mage_Catalog_Model_Resource_Product_Flat_Indexer :: getAttributeCodes ()):

Statique (type dorsal) Filtrable Utilisé dans la liste de produits Utilisé pour les règles de promotion Utilisé pour le tri selon les attributs du système Il existe une table plate différente pour chaque magasin, chacun contenant une valeur d'attribut d'entité différente. Les valeurs multilingues sont gérées en ayant différents magasins pour chaque langue.

Examples

implémenter l'interface des modèles frontend, source et backend d'attribut

Interface Frontend

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

}
```

Interface source

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

Interface 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);
}
```

Lire EAV (Entity Attribute Value) en ligne: https://riptutorial.com/fr/magento/topic/7121/eav--entity-attribute-value-

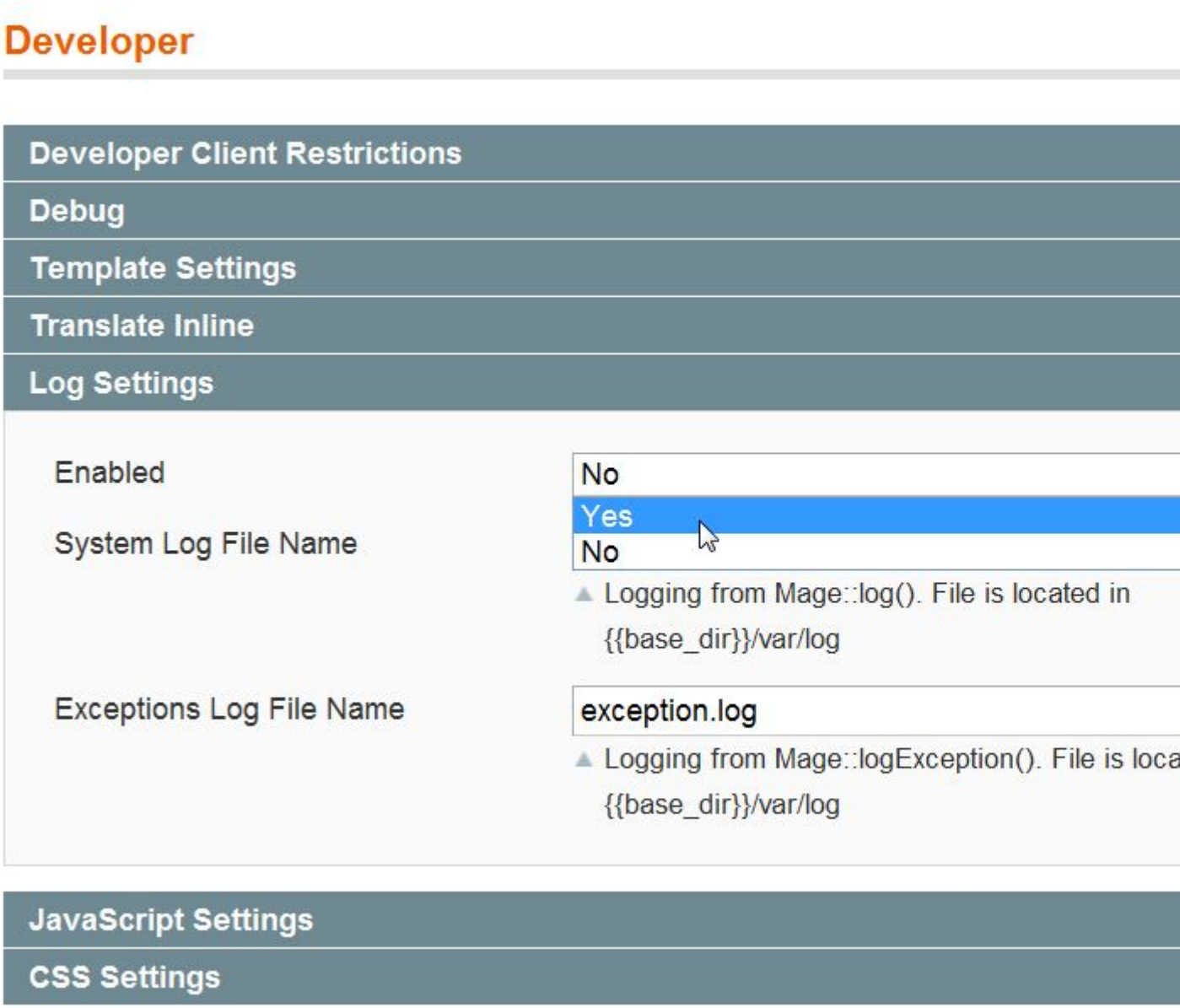
Chapitre 15: Gestion des erreurs Magento, messages et rapports

Remarques

Emplacement du journal des erreurs

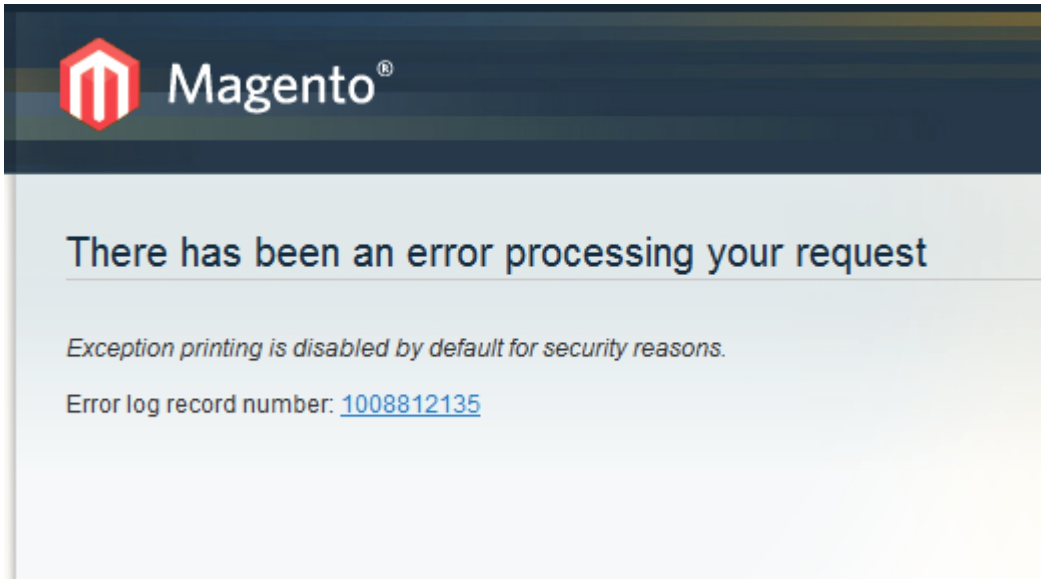
`/var/log/`

Généralement, le fichier system.log et le fichier exception.log existent dans le dossier `/var/log/` . Celles-ci contiennent la plupart des informations dont vous aurez besoin. Vous pouvez vérifier si ceux-ci sont activés et quels sont les noms des exceptions et du journal système en accédant à `System > Configuration > System > Developer > Log Settings` .



/var/report/

Les fichiers de rapport sont générés dans ce dossier après qu'un utilisateur a rencontré une erreur. Chaque fichier comprend uniquement les détails d'une erreur. Ceux-ci sont utilisés afin de cacher les détails de l'erreur au public. Sur la page d'erreur, il y aura un numéro de rapport qui est une référence au fichier correspondant avec le même nom dans le dossier `/var/report/` .



Examples

Activer l'affichage des rapports d'erreur

Dans la page d'index, modifiez les éléments suivants:

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

à

```
error_reporting(E_ALL);
```

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

et décommentez cette ligne (supprimez le `#`)

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

Vous pouvez également définir le mode Dev en utilisant `SetEnv` dans votre fichier `.htaccess`

Pour rendre l'erreur plus lisible et plus facile à comprendre, procédez comme suit:

1. Ouvrez votre répertoire d'installation de Magento. Allez dans le dossier des erreurs.

2. Renommer `local.xml.sample` fichier à `local.xml` .
3. Actualisez la page d'erreur dans le navigateur.

Lire Gestion des erreurs Magento, messages et rapports en ligne:
https://riptutorial.com/fr/magento/topic/2369/gestion-des-erreurs-magento--messages-et-rapports

Chapitre 16: Image du produit

Introduction

Obtenez les URL du produit pour les vignettes, les petites images et les images de base. Obtient également des caches et des caches de médias directs.

Examples

URL en cache

```
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');
```

URL des images non mises en cache à partir d'un média

```
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
```

Lire Image du produit en ligne: https://riptutorial.com/fr/magento/topic/9262/image-du-produit

Chapitre 17: Le rendu

Remarques

Personnalisation des fonctionnalités de base avec des thèmes

Les thèmes ont des fichiers de mise en page qui, entre autres, peuvent être utilisés pour modifier les blocs qui apparaissent sur la page. Le modèle de bloc peut également être modifié et différentes méthodes appelées.

Conceptions de niveau magasin

Les thèmes structurés hiérarchiques de Magento signifient qu'un thème de base peut être étendu et attribué au niveau d'un magasin.

Enregistrement de thèmes personnalisés

Les thèmes peuvent être configurés de trois manières:

1. Par magasin sous Système> Configuration> Conception.
2. Changement de conception avec des limites de temps Système> Conception.
3. Les exceptions de thèmes peuvent également être définies pour une catégorie et un niveau de produit.

Package versus Thème

Un package a plusieurs thèmes. Chacun des thèmes d'un package hérite du thème par défaut dans un package.

Design Fallback

La procédure de secours du thème pour localiser les fichiers de modèles est la suivante:

1. {package} / {theme}
2. {package} / default
3. base / défaut

Pour ajouter d'autres répertoires au mécanisme de secours du thème, la méthode Mage_Core_Model_Design_Package :: getFilename doit être réécrite

Pour la zone d'administration, la solution de secours est default / default.

Chemins de modèle et de mise en page

Blocs

Les blocs sont utilisés pour la sortie. Le bloc racine est le parent de tous les blocs et est de type Mage_Page_Block_Html.

Les blocs Mage_Core_Block_Template utilisent des fichiers modèles pour rendre le contenu. Le nom du fichier de modèle est défini dans setTemplate () ou addData ('template') avec des chemins relatifs.

Les templates ne sont que des morceaux de PHP inclus dans Mage_Core_Block_Template. Par conséquent, $ this dans un modèle fait référence au bloc.

Mage_Core_Block_Template utilise un tampon avant d'inclure un modèle pour empêcher une sortie prématurée.

La méthode Mage_Core_Model_Layout :: createBlock crée des instances de blocs.

La classe Mage_Core_Model_Layout_Update considère quels blocs doivent être créés pour chaque page en examinant les descripteurs de présentation.

Tous les blocs de sortie sont rendus, par exemple en appelant toHtml (), qui à son tour peut choisir de rendre leurs enfants.

Les blocs Text et Text_List rendent automatiquement leur contenu.

Deux événements sont déclenchés autour du rendu de bloc et peuvent être utilisés pour modifier le bloc avant et après le rendu du code HTML:

core_block_abstract_to_html_before core_block_abstract_to_html_after Un bloc enfant ne sera rendu automatiquement que s'il fait partie de la classe Mage_Core_Block_Textlist, sinon la méthode getChildHtml doit être appelée.

Les instances de bloc sont accessibles via la disposition, par exemple Mage :: app () -> getLayout () et $ controller-> getLayout (). La sortie de bloc est contrôlée par la fonction _toHtml ().

Les modèles sont rendus par les méthodes renderView () / fetchView () dans un bloc de modèle. La mise en mémoire tampon de sortie peut être désactivée avec $ layout-> setDirectOutput.

Il est possible d'ajouter un bloc à la mise en page actuelle, mais cela doit être fait avant d'appeler la méthode renderLayout ().

XML de mise en page

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

Les fichiers de mise en page peuvent être enregistrés dans config.xml:

```
<config>
    <{area}>
```

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

La sortie de la page peut être personnalisée des manières suivantes:

- Modifications du modèle
- Changements de disposition
- Blocs de substitution
- Observateurs Les variables sur les blocs peuvent être définies des manières suivantes:
- Mise en page - Actions ou attributs
- Controller - $ this-getLayout () -> getBlock ()
- Blocs enfants - $ this-> getChild ()
- Autre -Mage :: app () -> getLayout ()

Bloc de tête

Les ressources JavaScript et CSS sont gérées dans le bloc Mage_Page_Block_Html_head. Ce bloc gère la fusion des actifs dans un seul fichier afin de minimiser les requêtes HTTP. Le fichier fusionné est basé sur l'heure de modification des fichiers source.

Lors de la fusion de CSS, une fonction de rappel sur Mage_Core_Model_Design_Package est appelée pour mettre à jour les directives @import ou url () avec les URL correctes.

Examples

Différents mécanismes pour désactiver la sortie de bloc

- Si la réponse a déjà été créée et définie sur l'objet de réponse en dehors du processus de rendu normal (par exemple, dans un observateur), l'indicateur "no-renderLayout" peut être défini sur le contrôleur d'action en utilisant

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

- Cela empêche `renderLayout()` de traiter les blocs de sortie.
- La même chose peut être obtenue en appelant setNoRender (true) sur le contrôleur frontal: `Mage::app()->getFrontController()->setNoRender(true);`
- Définir l' `isDispatched()` sur l'objet de réponse peut être plus efficace pour obtenir un effet similaire.

différents types de blocs

- 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 Tous les blocs sont **des blocs structurels** ou Tous les blocs sont des blocs structurels ou des blocs de contenu. Exemple:
- **core / text_list**

Exemple d'un **bloc structurel** .

Il n'utilise pas de modèles; il est simplement utilisé pour afficher le contenu de tous ses blocs enfants l'un après l'autre.

- **core / template**

Exemple d'un **bloc de contenu** .

La sortie de ce type de bloc dépend du modèle assigné. Ses blocs enfants sont générés dans son modèle via la méthode getChildHtml ('nom_bloc'). Exemple: core / text_list - Exemple de bloc structurel. Il n'utilise pas de modèles; il est simplement utilisé pour afficher le contenu de tous ses blocs enfants l'un après l'autre. core / template - Exemple d'un bloc de contenu. La sortie de ce type de bloc dépend du modèle assigné. Ses blocs enfants sont générés dans son modèle via la méthode getChildHtml ('nom_bloc').

Les instances de bloc sont accessibles depuis le contrôleur

D'un contrôleur d'action:

```
$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()
);
}
```

Lire Le rendu en ligne: https://riptutorial.com/fr/magento/topic/7140/le-rendu

Chapitre 18: Magento Caching

Examples

Comment mettre en cache des données personnalisées dans 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;
```

Nettoyer le cache par ID de cache

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

Vider toutes les entrées du cache Magento

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

ou:

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

Utilisez Redis comme moteur de cache

Configuration Redis:

1. Installer redis (2.4+ requis)
2. Installer phpredis
3. Installez l'extension Magento `Cm_Cache_Backend_Redis` (uniquement pour Magento 1.7 et

inférieur)

4. Modifiez votre `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>
```

Lire Magento Caching en ligne: https://riptutorial.com/fr/magento/topic/4902/magento-caching

Chapitre 19: Modèle

Examples

Modèle de charge

Vous pouvez charger le modèle Magento en utilisant le code suivant: Mage :: getModel ('modulename / modelname')

Exemple: Mage :: getModel ('catalog / product') Cela va charger Mage_Catalog_Model_product

Créer un modèle vide

Pour créer un nouveau modèle dans votre module Ajoutez un `Model` dossier dans le dossier racine de votre module et créez un fichier Modelname.php dans ce dossier. par exemple Rick / Demo / Model / Modelname.php

Le nom de classe de votre modèle est important:

```
<?php
class Rick_Demo_Model_Modelname {

}
```

assurez-vous que votre modèle est défini dans votre `config.xml` dans le dossier `etc` de votre module

Voici un exemple:

2.0.4 Rick_Demo_Model

Pour charger votre module, utilisez le code suivant:

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

Lire Modèle en ligne: https://riptutorial.com/fr/magento/topic/7665/modele

Chapitre 20: Obtenir des produits de la base de données

Examples

Obtenir le produit par sku

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

Obtenir le produit par ID

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

Collection de produits - requête LIKE

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

Obtenir la collection de produits par attribut

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

Obtenir des données à partir de l'objet produit

```
// First load a product object

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

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

Obtenir une collection de produits sous forme de données

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

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

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

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

Collection de produits - avec attributs

```
//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');
```

Vérifiez si le produit a été correctement chargé

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

Obtenir l'identifiant du produit par SKU

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

Obtenir une collection de produits à partir d'une liste de références

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

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

OU

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

Définir la limite dans la collection de produits

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

Lire Obtenir des produits de la base de données en ligne:
https://riptutorial.com/fr/magento/topic/1102/obtenir-des-produits-de-la-base-de-donnees

Chapitre 21: Obtenir des URL Magento

Syntaxe

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

Paramètres

chemin des images	détails
exemple: *'images / my-images.jpg'*	chemin de l'image

Remarques

Obtenez des images formatées et évitez les dépendances de thème.

Examples

Dans l'interface / le thème actuel

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

Obtenir l'urine de la peau

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

Obtenir l'URL de base

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

URL de la peau sécurisée

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

Obtenir l'URL des médias

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

Urine de peau non sécurisée

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

Obtenir l'URL du magasin

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

Obtenez Jl Url

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

Obtenir l'url actuelle

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

Lire Obtenir des URL Magento en ligne: https://riptutorial.com/fr/magento/topic/2148/obtenir-des-url-magento

Chapitre 22: Obtenir l'utilisateur actuel

Examples

Obtenir l'utilisateur administrateur actuel

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

Obtenir le client actuel

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

ou

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

Vérifiez si l'utilisateur est connecté

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

Lire Obtenir l'utilisateur actuel en ligne: https://riptutorial.com/fr/magento/topic/2157/obtenir-l-utilisateur-actuel

Chapitre 23: Obtenir le nom de la catégorie à partir de la page du produit

Examples

Obtenir la catégorie parente

```
$_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();
```

Obtenez la catégorie actuelle

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

Lire Obtenir le nom de la catégorie à partir de la page du produit en ligne: https://riptutorial.com/fr/magento/topic/6078/obtenir-le-nom-de-la-categorie-a-partir-de-la-page-du-produit

Chapitre 24: Obtenir le nom du magasin et d'autres détails de la configuration du système

Examples

Obtenir le nom de l'interface pour la vue du magasin en cours

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

Obtenir l'ID du magasin actuel

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

Obtenir le code de magasin actuel

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

Cela renvoie le code magasin, par exemple "en" pour une vitrine configurée pour l'anglais et appelée "en", et non l'identifiant numérique.

Déterminer si la vue magasin est activée

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

Obtenir l'identifiant du site pour le magasin actuel

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

Obtenez le modèle de magasin actuel

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

Retourne une instance de `Mage_Core_Model_Store`

Obtenir le nom du groupe pour le magasin

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

Obtenez tous les magasins Magento

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

Retourne un tableau de modèles `Mage_Core_Model_Store` .

Lire Obtenir le nom du magasin et d'autres détails de la configuration du système en ligne: https://riptutorial.com/fr/magento/topic/1876/obtenir-le-nom-du-magasin-et-d-autres-details-de-la-configuration-du-systeme

Chapitre 25: Optimiser Magento pour la vitesse

Examples

Optimiser le changement de Magento Le fichier .htaccess

Magento est une application de commerce électronique très populaire. Il offre beaucoup de personnalisation et de capacités depuis l'installation initiale. Voici quelques suggestions pour optimiser une installation de Magento.

Activation de la compression de sortie

Dans votre fichier .htaccess pour Magento, vous trouverez une section de texte commençant par la ligne,

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

Cette section de code peut être utilisée pour activer le module mod_deflate d'Apache, qui fournit une compression pour le texte, le CSS et le javascript. Vous voudrez décommenter (supprimer le symbole #) plusieurs lignes pour qu'il ressemble à ceci:

##

activer la compression des fichiers servis 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>
```

Activation des en-têtes d'expiration

Les nouveaux visiteurs sur une page Web doivent faire plusieurs requêtes HTTP. En utilisant l'en-tête «Expires», vous pouvez mettre en cache les composants des requêtes. Cela évite les requêtes HTTP inutiles sur les pages suivantes.

Vous voulez trouver la zone du fichier .htaccess qui commence par <IfModulemod_expires.c> et se termine par le premier que vous voyez après, et le faire ressembler à ceci:

```
<IfModule mod_expires.c>

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

</IfModule>
```

Paramètres administrateur

Fusionner des fichiers JS et CSS

Ce réglage particulier réduira le nombre de requêtes HTTP sur votre site de commerce électronique. [box type = "alert" border = "full"] Remarque: cela peut parfois casser certaines applications. Après avoir effectué les étapes suivantes, assurez-vous que le site fonctionne toujours comme avant, avant d'activer cette fonctionnalité.

1. Connectez-vous à votre espace d'administration et accédez à - Système> Configuration> Développeur
2. Sous «Paramètres JavaScript», remplacez «Fusionner les fichiers JavaScript» par oui.
3. Sous «Paramètres CSS», remplacez «Fusionner les fichiers CSS» par oui.
4. Enfin, vous voudrez effacer votre cache Magento.

Activer les catalogues plats

Le modèle utilisé par Magento pour stocker les données client et produit se traduit par des requêtes SQL plus longues que la moyenne et plus de lectures. L'activation de l'option Catalogue plat pour les catégories et les produits fusionnera les données produit dans une seule table, améliorant ainsi les performances.

Connectez-vous à votre espace d'administration et accédez à - Système> Configuration> Catalogue Sous «Frontend», remplacez «Utiliser la catégorie du catalogue plat» par oui. Sous «Frontend», remplacez «Utiliser un produit de catalogue plat» par «oui» - cette option est facultative. Ensuite, vous voudrez effacer votre cache Magento. Enfin, vous devrez réindexer les tables. Activer la compilation

[box type = "alert" border = "full"] Remarque: cela peut parfois casser certaines applications. Après avoir effectué les étapes suivantes, assurez-vous que le site fonctionne toujours comme avant,

avant d'activer cette fonctionnalité.

1. Connectez-vous à votre espace d'administration et accédez à - Système> Outils> Compilation

2. Ensuite, cliquez simplement sur le bouton Exécuter le processus de compilation

3. Une fois la compilation lancée, elle devrait s'activer automatiquement

Activer le cache système

1. Connectez-vous à votre espace d'administration et accédez à - Système> Cache La gestion
2. Ensuite, cliquez sur le lien Sélectionner tout
3. Enfin, assurez-vous que les actions sont définies sur Activer et cliquez sur Soumettre.

Désactiver la journalisation des erreurs

Connectez-vous à votre domaine d'administration et accédez à - Système> Configuration> Développeur Dans la section Paramètres du journal, assurez-vous que l'option Activé est définie sur Aucun conseil de maintenance de base de données.

Il existe plusieurs tables utilisées par Magento pour la journalisation. Bien que la journalisation soit très importante pour savoir ce qui se passe et ce qui se passe dans votre magasin, les journaux peuvent devenir très volumineux, donc une maintenance régulière peut être très utile.

Voici les tableaux pour la journalisation:

```
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
```

Lire Optimiser Magento pour la vitesse en ligne:
https://riptutorial.com/fr/magento/topic/8010/optimiser-magento-pour-la-vitesse

Chapitre 26: Ordres

Examples

Obtenir la commande par ID

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

Le code ci-dessus est en gros analogue à la requête SQL suivante.

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

Obtenir la commande par Increment ID

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

Le code ci-dessus est en gros analogue à la requête SQL suivante.

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

L' `increment_id` est l'identificateur de la commande du client, alors que le `entity_id` est l'identificateur du niveau de la base de données pour la commande.

Ajouter un commentaire à l'historique des commandes

Vous pouvez ajouter un commentaire et un statut à l'ordre. Obtenir la commande:

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

Et ajouter un commentaire:

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

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

Lire Ordres en ligne: https://riptutorial.com/fr/magento/topic/1556/ordres

Chapitre 27: Quick Task Cheat Sheet

Remarques

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

Beaucoup de snippets très utiles

Examples

Obtenir la quantité de stock du produit

load ($ id); // ou le charger par 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 ();

Lire Quick Task Cheat Sheet en ligne: https://riptutorial.com/fr/magento/topic/7060/quick-task-cheat-sheet

Chapitre 28: Script SQL pour supprimer les données de test

Introduction

Script SQL pour supprimer les données de test des produits, des clients, des journaux et des ventes.

Examples

Supprimer les données de test du client

```
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;
```

Supprimer les données de test du produit

```
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;
```

Supprimer les données de test de vente

```
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;
```

Supprimer les données de test des journaux

```
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;
```

Lire Script SQL pour supprimer les données de test en ligne:
https://riptutorial.com/fr/magento/topic/9263/script-sql-pour-supprimer-les-donnees-de-test

Chapitre 29: Shell, CLI

Remarques

Les bases

- Vous devez disposer d'une ligne de commande Linux ou vous connecter via SSH à votre serveur pour pouvoir utiliser des scripts shell.
- Accédez à votre `MAGENTO_ROOT/shell`
- Le script peut être exécuté en tapant ie

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

Core shell méthodes par fichiers

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

Scripts PHP personnalisés

Parfois, nous devons accéder à Magento en dehors d'un navigateur Web pour omettre les temps d'exécution ou définir différentes choses qui n'affecteront pas le frontend.

Il y a 2 façons de démarrer Magento mais une seule est la méthode Magento. Lisez plus ci-dessus dans la section d'exemples.

Examples

Utiliser shell sans étendre Mage_Shell_Abstract

Amorcer Magento en appelant:

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

C'est le moyen le plus simple mais pas vraiment de Magento car nous n'utilisons pas la classe qui étend `Mage_Shell_Abstract` - la classe qui, lorsqu'elle est étendue, nous fournit des outils pour analyser les arguments en ligne de commande, appelle `__applyPhpVariables()` et applique les paramètres php au script shell).

Utiliser shell la manière Magento - étendre Mage_Shell_Abstract

Façon magento

Le fichier réside dans `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();

}
```

Réindexation à partir de CLI

Afficher le statut:

```
php indexer.php status
```

Reindex All

```
php indexer.php reindexall
```

Index spécifique Reindex

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

Liste des codes individuels

Indice	Code
Attributs de produit	catalog_product_attribute
Prix du produit	catalog_product_price
Réécriture d'URL du catalogue	catalog_url
Données plates du produit	catalog_product_flat
Catégorie données plates	catalog_category_flat
Catégorie Produits	catalogue_category_product
Index de recherche dans le catalogue	catalogsearch_fulltext
État des stocks	cataloginventory_stock

Lire Shell, CLI en ligne: https://riptutorial.com/fr/magento/topic/5387/shell--cli

Chapitre 30: Structure du module

Remarques

Les modules existent pour être étendus. Vous ne pouvez pas modifier `the app/code/` fichiers sans interdire les futures mises à jour. Au lieu de cela, nous ajoutons un module au répertoire `app/code/local` (le répertoire local est peut-être manquant, si c'est le cas, il doit être créé manuellement. Ceci est courant dans les versions ultérieures de Magento) pour ajouter des fonctionnalités locales personnalisées.

Tous les fichiers de configuration du module commencent par une `<config>` . Le nouveau module est déclaré à l'intérieur de la `<modules>` . Nous appellerons un module nommé YOUR_COMPANY_HelloWorld, par conséquent les balises `<YOUR_COMPANY_HelloWorld>` sont utilisées. Ici nous définissons la version (très premier = 0.0.1)

Une liste des **événements XML** peut être trouvée à l' adresse suivante : http://www.magentocommerce.com/wiki/5_-_modules_and_development/reference/module_config.xml

Si vous rencontrez des problèmes, consultez la page: http://coding.smashingmagazine.com/2012/11/30/introducing-magento-layout/

Examples

Créer un module à partir de rien (Hello World)

Le développement de modules personnalisés Magento fait partie intégrante de tout projet de développement Magento ou Magento, car à tout moment, vous souhaiterez peut-être intégrer vos propres fonctionnalités / modules dans votre projet Magento existant.

La première chose que les développeurs doivent désactiver est le cache du système. Sinon, le développement deviendra douloureux car tout changement nécessitera une couleur. Depuis le panneau d'administration de magento: accédez à `System > Cache Management > Select All > Actions : Disable`. Utilisez le guide suivant pour créer un nouveau module:

- Créez un dossier dans `app/code/local/` - les conventions de dénomination prennent généralement le nom de votre entreprise, par exemple `app/code/local/<YOUR_COMPANY>` .
- Nous avons maintenant un emplacement personnalisé pour nos modules. Créez un autre répertoire, appelez-le quelque chose en rapport avec le type de module que vous souhaitez créer, par exemple app / code / local / <VOTRE_COMPANY> / HelloWorld / - «HelloWorld» est ce que j'appellerai ce module.
- Ce répertoire nécessite un `config.xml` afin que Magento le reconnaisse comme un nouveau module. Créez un autre dossier nommé `etc` Suivi d'un fichier XML appelé config.xml. Le répertoire devrait ressembler à `app/code/local/<YOUR_COMPANY/HelloWorld/etc/config.xml` Voici

la structure du fichier xml:

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

- Ensuite, les modules doivent être déclarés à Magento. Passez à l' `app/etc/modules` . Créez un autre document XML et donnez-lui les noms choisis de vos tags: `YOUR_COMPANY_HelloWorld` dans mon cas. Dans ce document, écrivez:

```
<?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>
```

- Là encore, les balises de configuration et de module sont utilisées pour déclarer un nouveau module à Magento. Active est la valeur par défaut accessible dans le `Admin Panel under System > Configuration > Advanced` . `codePool` indique à Magento le répertoire à rechercher. `local` dans notre cas

- Ce module a été mis en place, donc le modèle de notre structure MVC. Vous devriez pouvoir voir votre nouveau module dans le Panneau d'administration sous `System > Configuration > Advanced` . **Cependant, il ne fait encore rien!** Vous devrez revenir à notre fichier config.xml et définir des éléments XML.

- Suite au tutoriel; Nous utiliserons certains de ces éléments XML pour créer des classes et manipuler toutes les pages du frontend de notre site. Retour au fichier `config.xml` écrivez ce qui suit sous la `</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>
```

- Comme vous pouvez le constater, nous étendons constamment plutôt que de manipuler les fichiers de base. La balise `helloworld` est en minuscule car cela `helloworld` une poignée, et pour plus de continuité, nous la nommerons aussi étroitement que possible. Nous `YOUR_COMPANY_HelloWorld` ensuite cela au module `YOUR_COMPANY_HelloWorld` .

- Nous changeons la mise en page. Par conséquent, nous devons créer cette poignée dans le répertoire de disposition. passez à `app/design/frontend/base/default/layout` . Nous avons demandé au module de rechercher le fichier `helloworld.xml` . Par conséquent, nous devons le créer dans ce répertoire. Qu'est-ce que tu attends. Fais le!! et le remplir avec:

```
<?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>
```

- Maintenant, ceux d'entre vous qui ont une petite expérience de Magento, ou qui ont lu d'autres didacticiels Magento, peuvent être découragés de constater que nous apportons des modifications à la base / par défaut, car c'est là que se trouvent les fichiers core Magento. Cependant, nous ne modifions aucun fichier ici, nous en créons de nouveaux, et nous préfixons également notre nom de fichier avec «helloworld». Il y a donc très peu de chance que cela entre en conflit avec d'autres modules ou que Magento soit mis à jour. avenir. Jours heureux!

- Comme nous voulons affecter toutes les pages, nous utilisons la balise par défaut et la `before_body_end` bloc structurel `before_body_end` . Cela va jouer le rôle de l'action et déclencher la section View de notre structure MVC.

- Maintenant, nous comprenons que nous nous `before_body_end` sur le bloc `before_body_end` . et le reliant à notre bloc personnalisé. Ceci est appelé une référence et est un **crochet** . Nous ne le connectons actuellement à rien d'existant, nous devons donc créer les fichiers

nécessaires.

- Dans `helloworld.xml` nous avons indiqué dans le modèle un `footer.phtml` . Passez à `app/design/frontend/base/default/template` et créez un répertoire `helloworld` .
- Dans ce répertoire, créez le fichier `footer.phtml` et remplissez-le avec HTML, ce tutoriel écrit simplement ceci pour afficher certaines fonctionnalités de PHP liées à notre fichier PHTML:

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

- Nous devons maintenant créer notre propre objet bloc pour coupler le modèle avec notre fonctionnalité de bloc. Créez le répertoire app / code / local / YOUR_COMPANY / HelloWorld / Block / et créez le fichier `Footer.php` intérieur de celui-ci. Cela a été référencé dans notre zeta_layout.xml dans le type "helloworld / footer". Remplissez ce fichier avec:

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

- C'est la fonctionnalité qui remplira notre appel `getDate()` appelé depuis notre fichier `.phtml` . Nous étendons le `Mage_Core_Block_Template` .
- Cette fonctionnalité est maintenant terminée. Testez-le en allant sur votre page d'accueil où vous devriez voir votre module dans le bas de chaque page!

Lire Structure du module en ligne: https://riptutorial.com/fr/magento/topic/3962/structure-du-module

Chapitre 31: Structure MVC

Remarques

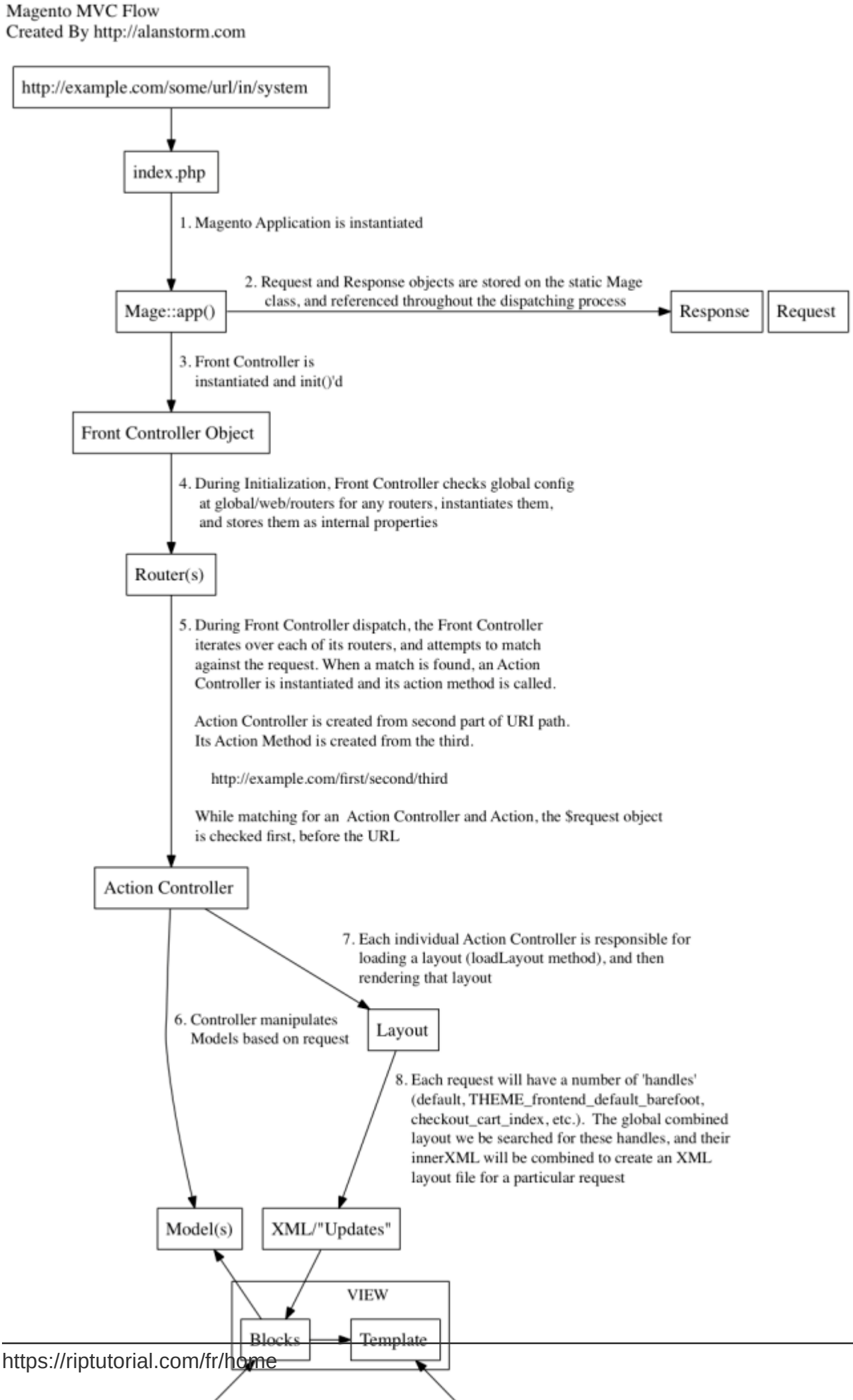
MVC signifie Model-View-Controller. Toute application qui sépare son accès aux données, sa logique applicative et son interface utilisateur s'appelle MVC. Il peut exister deux types de MVC: basés sur des conventions et sur des configurations. Exemple, CakePHP est basé sur des conventions, c.-à-d. Il suffit de suivre les instructions du système principal pour préparer votre module en quelques lignes seulement. Magento est basé sur la configuration, c'est-à-dire que vous devez spécifier chaque élément du fichier de configuration de votre module pour le faire fonctionner. Magento a un fichier Controller (pour le routage), Block, Model et Template. Comment fonctionne le MVC de Magento:

1. Lorsque vous entrez l'URL (quelque chose comme http://mysite.com/frontname/controller/method/param1/value1/param2/value2) , cette URL est interceptée par un fichier PHP appelé index.php qui instancie l'application Magento
2. L'application Magento instancie l'objet Front Controller
3. De plus, le contrôleur frontal instancie les objets du routeur (spécifiés dans le tag config.xml du module)
4. Maintenant, le routeur est responsable de "faire correspondre" le nom de front qui se trouve dans notre URL
5. Si «match» est trouvé, il voit le nom du contrôleur et le nom de la méthode dans l'URL, qui est finalement appelée.
6. Maintenant, selon ce qui est écrit dans le nom de l'action (nom de la méthode), il est exécuté. Si des modèles y sont appelés, la méthode du contrôleur instanciera ce modèle et appellera la méthode demandée.
7. Ensuite, l'action du contrôleur (méthode) instancie l'objet Layout, qui appelle Block spécifié pour ce nom d'action (méthode) (chaque nom d'action de contrôleur est associé à un fichier de bloc et de modèle, disponible dans app / design / frontend ou adminhtml / namespace / module / layout / module.xml, nom du fichier de mise en page (module.xml) se trouve dans config.xml de ce module, dans la balise de mise à jour de la mise en page).
8. Le fichier modèle (.phtml) appelle maintenant le bloc correspondant pour toute demande de méthode. Donc, si vous écrivez $ this-> methodName dans le fichier .phtml, il vérifiera «methodName» dans le fichier bloc qui est associé au fichier module.xml.
9. Block contient la logique PHP. Il référence les modèles pour toutes les données de la base de données.
10. Si Block, Template ou Controller doit obtenir / définir des données depuis / vers la base de données, ils peuvent appeler Model directement comme Mage :: getModel ('modulename / modelname').

Examples

Comprendre MVC dans Magento

MVC Flow dans Magento


Magento MVC Flow
Created By http://alanstorm.com
http://example.com/some/url/in/system
index.php
1. Magento Application is instantiated
Mage::app()
2. Request and Response objects are stored on the static Mage class, and referenced throughout the dispatching process
Response
Request
3. Front Controller is instantiated and init()'d
Front Controller Object
4. During Initialization, Front Controller checks global config at global/web/routers for any routers, instantiates them, and stores them as internal properties
Router(s)
5. During Front Controller dispatch, the Front Controller iterates over each of its routers, and attempts to match against the request. When a match is found, an Action Controller is instantiated and its action method is called.
Action Controller is created from second part of URI path. Its Action Method is created from the third.
http://example.com/first/second/third
While matching for an Action Controller and Action, the $request object is checked first, before the URL
Action Controller
7. Each individual Action Controller is responsible for loading a layout (loadLayout method), and then rendering that layout
6. Controller manipulates Models based on request
Layout
8. Each request will have a number of 'handles' (default, THEME_frontend_default_barefoot, checkout_cart_index, etc.). The global combined layout we be searched for these handles, and their innerXML will be combined to create an XML layout file for a particular request
Model(s)
XML/"Updates"
VIEW
Blocks
Template

Chapitre 32: URL actuelle

Syntaxe

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

Examples

Page d'accueil

retour: http://www.example.com/

Page produit

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

Vérifiez si l'URL actuelle est sécurisée

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

Cela retournera true si l'URL actuelle est sécurisée.

Lire URL actuelle en ligne: https://riptutorial.com/fr/magento/topic/2150/url-actuelle

Chapitre 33: URL spécifiques

Examples

URL du panier

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

OU

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

URL de paiement

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

OU

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

URL de connexion

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

OU

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

URL de déconnexion

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

OU

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

Mot de passe oublié? URL

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

OU

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

URL du client

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

OU

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

Media, JS, URL du skin

Récupérer le chemin de l'URL dans les pages STATIC BLOCK ou CMS

Pour obtenir l'URL de SKIN

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

Pour obtenir l'URL du média

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

Pour obtenir l'URL du magasin

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

Pour obtenir l'URL de base

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

POUR récupérer le chemin d'URL dans PHTML

URL de peau non sécurisée

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

URL de peau sécurisée

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

Obtenir l'URL actuelle

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

Obtenir l'URL de la maison

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

Obtenir l'URL des médias de Magento

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

Get Magento Skin Url

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

Obtenir l'URL du magasin Magento

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

Obtenez Magento Js Url

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

Lire URL spécifiques en ligne: https://riptutorial.com/fr/magento/topic/2144/url-specifiques

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec magento	7ochem, Chris Rogers, Community, Gabriel Somoza, goutam, Henry's Cat, Luke Rodgers, Marek Skiba, Pawel Dubiel, RamenChef, Robbie Averill, Stephen Leppik
2	Ajouter un prix différent pour plusieurs magasins à l'aide de l'API SOAP Magento	Harsha Sampath
3	Attributs personnalisés	Twinkal
4	Collection complète de produits	Twinkal
5	Collections	gebrial, gulshan maurya
6	Comment filtrer les collections	wesleywmd
7	Comprendre les types de produits	lalit mohan
8	Connexion au fichier	Akif, gulshan maurya, versedi
9	Créer des cartes-cadeaux d'entreprise par programme	JPMC, RamenChef
10	Des aides	Robbie Averill
11	Données du magasin et du site Web	Twinkal
12	EAV (Entity Attribute Value)	lalit mohan
13	Gestion des erreurs Magento, messages et rapports	7ochem, Charles, Pankaj Pareek, RamenChef
14	Image du produit	Twinkal

15	Le rendu	lalit mohan
16	Magento Caching	RamenChef, Robbie Averill, Yogendra - eCommerce Developer
17	Modèle	Rickert
18	Obtenir des produits de la base de données	Akif, Alex, baoutch, bpoiss, ctrimm, gulshan maurya, jignesh prajapati, mnoronha, Olavi Sau, pce, SH-
19	Obtenir des URL Magento	Nolwennig, Robbie Averill, Steven Church
20	Obtenir l'utilisateur actuel	bpoiss, Rickert
21	Obtenir le nom de la catégorie à partir de la page du produit	user2925795
22	Obtenir le nom du magasin et d'autres détails de la configuration du système	Henry's Cat, Robbie Averill
23	Optimiser Magento pour la vitesse	Lemon Kazi
24	Ordres	bpoiss, Hemant Sankhla, Luke Rodgers
25	Quick Task Cheat Sheet	Chris Richardson
26	Script SQL pour supprimer les données de test	Twinkal
27	Shell, CLI	djdy, versedi
28	Structure du module	Chris Rogers, RamenChef
29	Structure MVC	lalit mohan
30	URL actuelle	Nolwennig, Twinkal
31	URL spécifiques	Mayank Pandeyz, Nolwennig, Qaisar Satti, Yogendra - eCommerce Developer