EBook Gratuito

APPRENDIMENTO magento

Free unaffiliated eBook created from
Stack Overflow contributors.

#magento

Sommario

Di

Capitolo 1: Iniziare con Magento

Osservazioni

Magento è una piattaforma di e-commerce open source scritta in PHP; una piattaforma di e-commerce altamente personalizzabile e un sistema di gestione dei contenuti che può essere utilizzato per creare negozi online per la vendita di merci.

Fornisce funzionalità comuni di e-commerce, come carrelli della spesa e gestione dell'inventario, e incoraggia l'ampia personalizzazione per soddisfare gli obiettivi specifici dell'organizzazione.

Magento è anche un framework PHP orientato agli oggetti che può essere utilizzato per sviluppare applicazioni web moderne e dinamiche che sfruttino le funzionalità di eCommerce di Magento.

Le principali caratteristiche della piattaforma Magento sono:

- espandibilità
- scalabilità
- flessibilità
- personalizzazione
- open-source

Versioni

Edizione comunitaria

Versione	Data di rilascio
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
1.0	2008-03-31

Edizione Enterprise

Versione	Data di rilascio
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

Installazione e configurazione

Prerequisiti e requisiti per Magento Community Edition 1.9

Ospitando

- Apache 2.x (con mod_rewrite) o Nginx 1.7.x
- A causa delle esigenze di elaborazione delle operazioni di Magento, si consiglia di installare Magento su un server con almeno 2 GB di RAM. Ciò garantirà che tutto il software coinvolto nella gestione dello store disponga di memoria sufficiente per funzionare.
- Possibilità di eseguire lavori programmati (crontab) con PHP 5.
- Possibilità di sovrascrivere le opzioni nei file .htaccess.

PHP

- PHP 5.4, PHP 5.5
- Estensioni richieste: PDO_MySQL, simplexml, mcrypt, hash, GD, DOM, iconv, curl, SOAP (per API Webservices)
- memory_limit non meno di 256 MB (consigliati 512 MB)

Banca dati

- MySQL 5.6 (Oracle, Percona, MariaDB)

SSL

- Per HTTPS è richiesto un certificato di sicurezza valido.
- I certificati SSL autofirmati non sono supportati

Installazione:

Scarica e configura i file Magento

Stiamo usando openMage mirror come download diretto per il ramo 1.9.2.4 disabilitato e il sito Web di Magento richiede un account. Ma sei incoraggiato a scaricare la copia da 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
```

Creare un database e un utente MySQL

accedere alla console mysql

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

nella 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
```

Completa l'installazione tramite l'interfaccia web

Per accedere all'interfaccia web con il browser, accedere al nome di dominio o all'indirizzo IP

pubblico del server:

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

Quindi seguire le istruzioni sullo schermo

Risoluzione dei problemi comuni

Funziona solo la homepage, tutte le altre pagine restituiscono 404

Assicurati che il modulo `mod_rewrite` sia stato installato in Apache e sia stato abilitato per il caricamento. Vedere il **passaggio 2** per informazioni su come farlo qui: https://www.digitalocean.com/community/tutorials/how-to-set-up-mod_rewrite-for-apache-on-ubuntu-14-04

Assicurati di consentire le modifiche nel file .htaccess abilitandolo nel tuo sito conf. Vedere la **fase 3** : https://www.digitalocean.com/community/tutorials/how-to-set-up-mod_rewrite-for-apache-on-ubuntu-14-04

Il tuo file `.htaccess` potrebbe essere configurato in modo errato o mancante: vai alla pagina di download di Magento: https://www.magentocommerce.com/download - scarica la versione pertinente ed estrai il file .htaccess da inserire nella root di installazione di Magento.

Il sito funziona ma non vengono caricati stili o script

Assicurati di aver impostato le autorizzazioni e la proprietà rilevanti: **vedi qui** per maggiori informazioni - http://devdocs.magento.com/guides/m1x/install/installer-privileges_before.html

Soluzione comune: prova a reindicizzare e svuotare la cache manualmente (nel caso in cui l'amministratore sia troppo difficile da navigare). Reindex tramite la riga di comando: https://www.atwix.com/magento/process-magento-indexes-from-command-line/ Svuota cache (tramite admin o riga di comando): https://www.properhost.com/ supporto / kb / 23 / How-To-Clear-La-Magento-cache

seguito

Leggi Iniziare con Magento online: https://riptutorial.com/it/magento/topic/812/iniziare-con-magento

Capitolo 2: Aggiungi un prezzo diverso per più negozi utilizzando l'API SOAP Magento

Examples

Magento SOAP V1

Devi modificare l'ambito di prezzo **"Globale" in "Sito web"** (Sysytem-> Configurazione-> Catalogo-> Catalogo-> Prezzo)

```
$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>";
```

Leggi Aggiungi un prezzo diverso per più negozi utilizzando l'API SOAP Magento online: https://riptutorial.com/it/magento/topic/7410/aggiungi-un-prezzo-diverso-per-piu-negozi-utilizzando-l-api-soap-magento

Capitolo 3: Attributi personalizzati

introduzione

Attributi personalizzati per vendite, categoria, ecc.

Examples

Attributo di vendita

Attributo personalizzato nelle tabelle relative alle vendite come: sales_flat_quote, sales_flat_order_item, sales_flat_order, tabella ecc.

Nel file di installazione 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();
 ?>
```

Un altro modo per farlo è:

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

Assicurati di svuotare la cache dopo questo.

Leggi Attributi personalizzati online: https://riptutorial.com/it/magento/topic/9267/attributi-personalizzati

Capitolo 4: Collezione completa di prodotti

introduzione

Filtri completi per la raccolta dei prodotti in base a visibilità, negozio, E OR, stato delle scorte, stato, ecc

Examples

Filtraggio della collezione di prodotti

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

Filtro basato sul negozio:

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

Filtro basato sul tipo di prodotto:

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

Filtro basato sullo stato:

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

Filtra usando null e notnull:

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

Filtra usando maggiore di e minore di:

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

Filtra usando maggiore di e uguale a:

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

Filtra usando meno di e uguale a:

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

Filtra usando in e non in:

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

Filtra i prodotti in base a un intervallo di ID entità:

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

Filtro basato sulla visibilità del prodotto:

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

Filtra usando come e non come:

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

Filtro utilizzando uguale e non uguale a:

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

Filtrare i prodotti in magazzino:

```
$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')
```

Imposta ordine per:

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

Imposta dimensioni della pagina:

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

Leggi Collezione completa di prodotti online:
https://riptutorial.com/it/magento/topic/9261/collezione-completa-di-prodotti

Capitolo 5: collezioni

Examples

Ottieni collezioni modello

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

Ottieni ulteriori attributi dell'oggetto

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

filtraggio

```
// 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));
```

Ordinamento

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

Accesso

```
// 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();
```

Ottieni oggetto di raccolta

```
    // 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');
```

Leggi collezioni online: https://riptutorial.com/it/magento/topic/5937/collezioni

Capitolo 6: collezioni

Examples

Collezione di prodotti

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

Selezione dell'attributo specifico

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

Selezione di tutti gli attributi

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

Aggiungi filtro alla raccolta

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

Aggiungi filtro attributo prodotto alla raccolta

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

Imposta ordine

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

Imposta limite

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

Imposta la pagina corrente

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

Categoria Raccolta di un negozio specifico e livello specifico

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

Leggi collezioni online: https://riptutorial.com/it/magento/topic/6763/collezioni

Capitolo 7: collezioni

Examples

Collezione di prodotti

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

Selezione dell'attributo specifico

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

Selezione di tutti gli attributi

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

Aggiungi filtro alla raccolta

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

Imposta ordine

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

Imposta limite

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

Imposta la pagina corrente

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

Leggi collezioni online: https://riptutorial.com/it/magento/topic/7206/collezioni

Capitolo 8: Come filtrare le raccolte

Parametri

Parametro	Dettagli
$ addFieldToFilter (**$ campo** , $ condizione = null)	{string} Il campo che stiamo aggiungendo al filtro.
$ addFieldToFilter ($ campo, **$ condizione = null**)	{misto} La definizione del filtro che useremo.
addAttributeToFilter (**$ attr** , $ condizione = null, $ join = 'inner')	{string} Il campo che stiamo aggiungendo al filtro.
addAttributeToFilter ($ attr, **$ condizione = null** , $ join = 'inner')	{misto} La definizione del filtro che useremo.
addAttributeToFilter ($ attr, $ condizione = null, **$ join = 'inner'**)	{('inner', 'left')} Il tipo di join SQL da utilizzare quando si entra nella tabella EAV.

Osservazioni

Argomenti di confronto dei filtri

Magento offre anche un modo flessibile di filtraggio utilizzando anche operatori di confronto. Ecco una lista di operatori validi e la loro sintassi:

Tutti gli argomenti di confronto possono essere passati al secondo parametro dei `addFieldToFielter()` o `addAttributeToFilter()` .

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

Confronto	Array di argomenti	Snippet SQL risultante
È uguale a	array ("eq" => $ var)	DOVE (`my_field` = $ var)
Non uguale	array ("neq" => $ var)	DOVE (`my_field`! = $ Var)
Piace	array ("come" => $ var)	DOVE (`my_field` LIKE $ var)
Non come	array ("Nlike" => $ var)	DOVE (`my_field` NOT LIKE $ var)
È	array ("è" => $ var)	DOVE (`my_field` IS $ var)
Nel	array ("a" => $ var)	DOVE (`my_field` IN ($ var))

Confronto	Array di argomenti	Snippet SQL risultante
Non in	array ("nin" => $ var)	DOVE (`my_field` NOT IN ($ var))
Nullo	array ("nullo" => true)	DOVE (`my_field` È NULL)
Non nullo	array ("notnull" => true)	DOVE (`my_field` NON È NULL)
Più grande di	array ("GT" => $ var)	DOVE (`my_field`> $ var)
Meno di	array ("LT" => $ var)	DOVE (`my_field` <$ var)
Maggiore o uguale	array ("gteq" => $ var)	DOVE (`my_field`> = $ var)
Meno o uguale	array ("lteq" => $ var)	DOVE (`my_field` <= $ var)
Trova nel set	array ("finset" => array ($ var))	DOVE (find_in_set ($ var, `mio_field`)
Da e a	array ("from" => $ var1, "a" => $ var2)	DOVE (`my_field`> = $ var1 AND` my_field` <= $ var2)

Examples

Collezioni di filtri

Magento ha un potente set di metodi per filtrare le collezioni. Dato che ci sono due tipi di oggetti che possono essere contenuti nelle collezioni, dobbiamo prima determinare quale tipo di dati stiamo lavorando prima di poterlo filtrare. Magento implementa un modello di dati EAV per entità come prodotti e categorie. C'è un diverso insieme di metodi da usare se stiamo filtrando una collezione di oggetti EAV.

In Magento, gli ordini non sono memorizzati come oggetti EAV. Ciò rende la raccolta ordini un buon esempio per il filtraggio di una raccolta di base.

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

Se osserviamo la collezione di prodotti, possiamo vedere che i prodotti sono memorizzati in un modello di dati EAV. Possiamo facilmente filtrare anche per attributi EAV.

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

Gestione di AND e OR nei filtri

Quando interrogiamo i nostri dati, spesso abbiamo bisogno di più di un filtro per ottenere il set di

dati esatto che stiamo cercando. In SQL, gestiamo questo con le clausole AND e OR. Possiamo ottenere la stessa cosa con le collezioni.

Per aggiungere una clausola AND alla query, è sufficiente aggiungere un'altra chiamata al metodo. Questo aggiungerà il secondo filtro all'istruzione WHERE originale che lo unisce ad un AND.

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

La clausola WHERE risultante sarà simile a questa:

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

Ora diciamo che vogliamo tutti gli skus che iniziano con 'a' O terminano con 'b'. Come aggiungiamo una clausola OR? Grazie alle collezioni di Magento, è abbastanza semplice. Aggiungiamo il filtro come un secondo elemento nell'array del filtro.

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

Ora, la clausola WHERE risultante sarà simile a questa:

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

Leggi Come filtrare le raccolte online: https://riptutorial.com/it/magento/topic/5849/come-filtrare-le-raccolte

Capitolo 9: Comprensione dei tipi di prodotto

Osservazioni

Esistono sei diversi tipi di prodotto integrati in Magento.

- Semplice

Una singola unità di magazzino

- configurabile

Primo dei prodotti compositi. Consenti ai clienti di configurare il proprio prodotto e aggiungere un singolo prodotto semplice al carrello.

- raggruppate

Il secondo prodotto composito, un prodotto raggruppato mette in relazione prodotti semplici e offre ai clienti la possibilità di scegliere le quantità di ciascun articolo.

- impacchettare

Il terzo tipo di prodotto composito, un pacchetto collega prodotti semplici insieme per l'acquisto come singolo articolo.

- Virtuale

Nessun articolo fisico richiesto per la consegna, ad es. Servizi

- Scaricabile

Un prodotto digitale piuttosto che fisico. La maggior parte dei tipi di prodotto sono implementati come parte del modulo Mage_Catalog, oltre a Mage_Bundle e Mage_Downloadable.

I prodotti raggruppati, bundle e configurabili implementano una relazione genitore-figlio in cui un numero di altri prodotti (di default, semplici, virtuali o scaricabili) vengono assegnati a un prodotto principale. In questo modo vengono gestiti i dati del prodotto per l'intera raccolta (ad es. Gruppo, pacchetto o nome prodotto, prezzo e stato configurabili).

I prodotti scaricabili e bundle hanno tabelle aggiuntive nel database, mentre il resto sono condivisi tra tutti gli altri tipi di prodotto. I prodotti configurabili dispongono di una tabella aggiuntiva per il collegamento ai prodotti secondari, catalog_product_super_link.

Tipo di prodotto personalizzato

Per creare un tipo di prodotto che estenda uno dei tipi di prodotto integrati, è necessario estendere il modello del tipo di prodotto corrispondente. In caso contrario, il nuovo tipo di prodotto dovrebbe estendere la classe Mage_Catalog_Model_Product_Type_Abstract.

È richiesta anche una voce nel config.xml del modulo:

Prodotti più complicati possono richiedere altre aree personalizzate come il modello di prezzo e il retriever dei dati dell'indice.

Calcolo del prezzo

Quando si tratta di un singolo prodotto, il prezzo viene sempre calcolato al volo. L'attributo EAV di prezzo viene caricato con il prodotto e il prezzo finale viene calcolato dal modello di prezzo, Mage_Catalog_Model_Product_Type_Price.

Alcuni tipi di prodotti si occupano in modo diverso. In tal caso estendono questa classe e implementano la loro logica. Ad esempio, il prodotto configurabile sovrascrive getFinalPrice () e aggiunge logica aggiuntiva. Questo modello personalizzato può quindi essere specificato in config.xml con un tag <price_model>.

Le raccolte di prodotti, tuttavia, utilizzano l'indice dei prezzi per recuperare i prezzi precalcolati, eliminando la necessità di calcolarlo per ciascun prodotto.

Il prezzo finale può essere regolato dagli osservatori dell'evento catalog_product_get_final_price. Per impostazione predefinita, solo il modulo Mage_CatalogRule osserva questo evento.

Un altro metodo per scavalcare il prezzo di produzione è semplicemente impostarlo sul prodotto. Se il prezzo è impostato, il prodotto non lo ricalcola.

Il prezzo del livello del prodotto è separato dal prezzo normale (sebbene preso in considerazione nel calcolo del prezzo). È implementato come un elenco con un gruppo di clienti e qualificatori di quantità minima per ogni livello. I prezzi di livello vengono visualizzati in una tabella, utilizzando il modello catalogo / prodotto / vista / tierprices.phtml.

Le opzioni di prodotto personalizzate vengono elaborate durante il calcolo del prezzo finale. Ogni opzione ha il suo prezzo definito, che viene aggiunto al prezzo finale.

I prezzi di gruppo, di livello e speciali vengono presi in considerazione contemporaneamente `($priceModel->getBasePrice())` e il più piccolo dei tre (o quattro se si include il prezzo normale) viene scelto come prezzo base del prodotto.

Imposta

Il modulo Mage_Tax utilizza la classe di tasse di un prodotto e indipendentemente dal fatto che il prezzo del prodotto sia inclusivo o esente da tasse al fine di identificare l'aliquota corretta da applicare.

I seguenti fattori vengono utilizzati per calcolare le imposte sui prodotti:

- Classe fiscale del prodotto
- L'importo della tassa già incluso
- Indirizzi di fatturazione e spedizione
- Classe fiscale cliente
- Memorizza le impostazioni

Navigazione a strati

Le classi responsabili del rendering della navigazione a più livelli sono:

- Mage_Catalog_Block_Layer_View

  - Gestisce i filtri e le opzioni

- Mage_Catalog_Block_Layer_State

-Controlla ciò che viene attualmente filtrato da

Per implementare la navigazione a più livelli sugli attributi con modelli di origine personalizzati, il metodo Mage_Catalog_Model_Layer_Filter_Abstract :: apply () dovrebbe essere sovrascritto per dettare come deve essere filtrata la raccolta di prodotti.

La navigazione stratificata viene resa dai blocchi Mage_Catalog_Block_Layer_View e Mage_Catalog_Block_Layer_State, che utilizzano i blocchi filtro per i singoli filtri.

La navigazione stratificata utilizza la tabella degli indici per la maggior parte dei filtri, ad esempio prezzo, indice degli attributi dei prodotti, indice dei prodotti decimali.

categorie

Categorie nel database

La gerarchia di categorie viene gestita memorizzando l'ID genitore di una categoria. La gerarchia completa è mostrata nella colonna percorso (ID barra separata). C'è una categoria speciale con parent_id di 0. Questa è la vera categoria radice e ognuna delle altre categorie di root come definite in Magento usa questo come un genitore condiviso.

Per leggere e gestire un albero di categorie dal database vengono utilizzate due classi diverse a seconda che il catalogo sia attivato, Mage_Catalog_Model_Resource_Category_Tree e Mage_Catalog_Model_Resource_Category_Flat.

Il vantaggio delle categorie piatte è che è più veloce da interrogare. Tuttavia, deve essere ricostruito dalle tabelle EAV ogni volta che viene apportata una modifica.

```
getChildren()
```

restituisce una stringa separata da virgole di ID bambini immediati

```
getAllChildren()
```

restituisce una stringa o una matrice di tutti gli ID figli

```
getChildrenCategories()
```

restituisce una raccolta di categorie di bambini immediati NB Se il catalogo flat è abilitato, le sole

categorie di figlio restituite saranno quelle con include_in_menu = 1. In entrambi i casi, vengono restituite solo le categorie attive.

Regole del prezzo del catalogo

Le regole del prezzo del catalogo applicano sconti ai prodotti in base alla data, al prodotto, al sito Web e al gruppo di clienti.

Quando viene chiamato `getFinalPrice()` su un prodotto, viene `getFinalPrice()` l'evento catalog_product_get_final_price. Questo è osservato da Mage_CatalogRule_Model_Observer che cercherà quindi qualsiasi regola del prezzo di catalogo applicabile al prodotto. Se applicabile, esamina quindi la tabella dei prezzi del database e scrive il prezzo sul modello del prodotto come un campo di dati Varien final_price.

All'interno del database, la tabella catalogrule descrive le regole, le loro condizioni e le loro azioni. catalogrule_product contiene i prodotti abbinati e alcune informazioni sulle regole. Nel frattempo catalogrule_product_price contiene il prezzo dopo l'applicazione della regola.

Indicizzazione e tabelle piatte

Le tabelle di catalogo semplici sono gestite dagli indicizzatori di catalogo. Se la ricostruzione automatica degli indici è attivata, gli indicizzatori del catalogo vengono ricreati ogni volta che un prodotto, una categoria o qualsiasi entità correlata vengono aggiornati. Il metodo _afterSave () chiama il processo dell'indicizzatore. Altrimenti devono essere reindicizzati manualmente tramite admin.

Il tipo di prodotto influisce sull'indice dei prezzi e sull'indice azionario in cui i prodotti possono definire i propri indicizzatori personalizzati (in config.xml) per gestire i loro dati per questi indici.

Il modulo Mage_Index fornisce il framework con il quale è possibile creare indici personalizzati per ottimizzare le prestazioni del sito. La classe Mage_Index_Model_Indexer_Abstract deve essere estesa per creare un nuovo indice, implementando i metodi _registerEvent () e _processEvent (). Senza dimenticare di registrarlo in 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;
    }
}
```

Descrivere tipi di prodotti standard (semplici, configurabili, in bundle)

Semplice

Il tipo di prodotti semplici dovrebbe essere usato in quanto generalmente hanno una configurazione singola (taglia unica). Questo potrebbe includere elementi come:

- Una scatola di pastelli, piccola (24 colori)
- Una scatola di pastelli, grande (64 colori)
- Monitor per computer HD da 26 "SuperHighTech
- Barrack Obama Action Figure (6 ")

raggruppate

I prodotti raggruppati ti consentono di creare un nuovo prodotto utilizzando uno o più prodotti

esistenti nel tuo negozio. Ad esempio, supponiamo di avere già una "figura d'azione di Barrack Obama" e una "figura d'azione di George W. Bush" nel tuo negozio e volevi venderle come un pacchetto. Dovresti semplicemente creare un nuovo prodotto raggruppato (chiamiamolo "Obama + Bush (prendi entrambi e spendi due volte tanto!)", Quindi aggiungi entrambe le figure di azione al gruppo tramite la scheda "Prodotti associati".

Nota: Purtroppo, non è possibile impostare un prezzo speciale per "gruppo" direttamente dalla pagina del prodotto. Per offrire uno sconto per l'acquisto di articoli, dovrai creare una nuova regola del prezzo del carrello.

configurabile

Prodotto configurabile: questo prodotto consente ai clienti di selezionare la variante che desiderano scegliendo le opzioni. Ad esempio, puoi vendere magliette in due colori e tre taglie. Dovresti creare sei prodotti semplici come singoli prodotti (ciascuno con i propri SKU) e quindi aggiungere questi sei a un prodotto configurabile in cui i clienti possono scegliere la dimensione e il colore, quindi aggiungerlo al carrello. Funzionalità molto simile è possibile utilizzando le opzioni personalizzate per i prodotti semplici. La differenza tra un prodotto configurabile e un prodotto che include opzioni personalizzate è che l'inventario non viene controllato o aggiornato per le singole opzioni durante l'acquisto delle opzioni personalizzate.

Virtuale

I prodotti virtuali sono quelli che non hanno una controparte fisica o digitale. Non spediscono, né hanno un link per il download. Questo tipo di prodotto potrebbe essere utilizzato per servizi come:

- Pulizie di casa
- Iscrizione alla newsletter di 1 anno

Nota: se si utilizzano prodotti virtuali per "abbonamenti", è importante notare che non esiste un modo integrato per gestire gli abbonamenti con rinnovo automatico. Tutti gli acquisti effettuati in Magento, indipendentemente dal tipo di prodotto, sono acquisti una tantum.

impacchettare

Questo tipo di prodotto è anche noto come "kit" in altri software eCommerce. Questo tipo di prodotto è ideale per le circostanze in cui l'utente deve selezionare un numero di opzioni configurabili, ma almeno un'opzione. Questo potrebbe includere prodotti come:

- Sistemi informatici personalizzabili
- Smoking / tute personalizzabili
- Fai clic qui per un'esercitazione video sull'utilizzo dei pacchetti

Scaricabile

I prodotti scaricabili sono simili ai prodotti virtuali, tranne che includono la possibilità di aggiungere uno o più file digitali per il download. I file possono essere caricati tramite l'interfaccia di amministrazione o caricati direttamente sul server tramite FTP e quindi aggiunti tramite URL. Quando un cliente acquista un prodotto scaricabile, Magento genererà un collegamento

crittografato sicuro (in modo che i clienti non possano vedere la posizione reale del file) affinché quel cliente possa scaricare il proprio file.

Questa categoria potrebbe includere prodotti come:

- Musica / Mp3
- Software per il computer

Nota : se hai SSL abilitato per il tuo sito, i download potrebbero non riuscire in tutte le versioni di IE poiché IE contiene un bug che impedisce il download su connessioni sicure se è impostata l'intestazione no-cache. Questo può essere facilmente risolto in un file htaccess rimuovendo le intestazioni no-cache e no-store o forzando i link di download a non essere sicuri.

Leggi Comprensione dei tipi di prodotto online:
https://riptutorial.com/it/magento/topic/7142/comprensione-dei-tipi-di-prodotto

Capitolo 10: Crea carte regalo aziendali a livello di programmazione

Examples

Crea una singola carta regalo

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

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

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

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

Leggi Crea carte regalo aziendali a livello di programmazione online: https://riptutorial.com/it/magento/topic/2387/crea-carte-regalo-aziendali-a-livello-di-programmazione

Capitolo 11: Dati del negozio e del sito web

introduzione

Ottieni informazioni su Magento Store e sul sito web

Examples

Ottieni i dati del negozio corrente

```
$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();
```

Leggi Dati del negozio e del sito web online: https://riptutorial.com/it/magento/topic/9266/dati-del-negozio-e-del-sito-web

Capitolo 12: EAV (Valore attributo entità)

Osservazioni

Entità

Memorizza informazioni sul tipo di dati memorizzati. Nel caso di Magento questo è cliente, prodotto, categoria, ecc.

Attributo

Le singole proprietà di ciascuna entità, ad es. Nome, peso, indirizzo e-mail, ecc.

Valore

Il valore di una determinata entità e attributo. Ad esempio, potremmo specificare l'entità cliente e l'attributo e-mail e dargli il valore ciao@esempio.com.

Schema del database

eav_entity

La tabella delle entità.

eav_entity_attribute

La tabella degli attributi.

eav_entity_ {tipo}

Le tabelle dei valori. I tipi sono datetime, decimals, int, text e varchar.

È importante notare che la tabella eav_entity_varchar ha il tipo varchar per i valori anche se la data o il numero intero si adattano meglio al valore.

Modelli contro modelli di risorse

Tutti i modelli all'interno di Mage / Eav / Model / Resource sono Mysql4 e sono modelli di risorse.

Inoltre Entity / Abstract.php e Entity / Setup.php.

Versus EAV piatto

I modelli EAV sono più complessi, forniscono la logica per salvare e caricare da più tabelle, mentre i modelli piatti o standard sono relativamente semplici (tradizionali).

I modelli standard gestiscono principalmente le loro proprietà con data setter e getter che lavorano con una singola tabella. I modelli EAV gestiscono principalmente i loro modelli di attributi. I modelli standard salvano i loro dati solo su una tabella e li caricano. I modelli EAV caricano tutti gli attributi

(o un set specifico) dopo aver caricato i dati di base e salvato gli attributi dopo aver salvato i dati (incluso l'inserimento, l'aggiornamento e l'eliminazione degli attributi).

Esempi di modelli di risorse EAV

Per trovare le entità che utilizzano lo schema di archiviazione EAV alla ricerca della stringa, è possibile utilizzare Mage_Eav_Model_Entity_Abstract. Questo dovrebbe rivelare tutti i modelli di risorse basati sulla struttura EAV. Tuttavia, l'elenco risultante includerà molte classi obsolete dal modulo Mage_Sales che non vengono più utilizzate.

Gli unici moduli contenenti entità che utilizzano lo schema di archiviazione EAV sono Mage_Catalog (categorie e prodotti) e Mage_Customer (clienti e indirizzi).

I gruppi di clienti utilizzano lo schema di archiviazione della tabella flat. Tutte le entità di vendita in cui sono state convertite in entità di tabella flat con il rilascio di Magento 1.4.

La ragione per cui EAV viene utilizzato è che le entità possono avere un numero indeterminato di proprietà e quindi rimanere flessibili. Ad esempio, quando aggiungi un nuovo attributo a un'entità Cliente (che è un'entità EAV), la tabella del database non deve essere modificata per poter aggiungere questo nuovo attributo.

vantaggi

Flessibilità Lo schema del database non deve essere modificato con il modello Quick da implementare

svantaggi

- Inefficiente

Una query che restituisce normalmente 20 colonne consisterebbe in 20 self join in EAV. Tuttavia, il modulo Mage_Eav generalmente non utilizza i join per caricare i dati del valore dell'attributo. Invece vengono usati i sindacati. I join vengono utilizzati solo per filtrare le raccolte EAV.

- Nessun meccanismo per le relazioni tra sottotipi
- Nessun raggruppamento di sottotipi di entità

Siti Web e negozi

Per gestire i valori degli attributi del sito Web e dello spazio di archiviazione all'interno di EAV, esiste un valore store_id sull'entità catalogo per mostrare l'ambito che collega nuovamente a core_store. Insieme ai normali negozi (punti vendita) c'è anche un negozio '0' che è il valore globale. Quando si trova su un determinato negozio, il sistema prima controlla il valore di un'entità nel negozio corrente e quindi torna all'entità globale. Le entità EAV di Mage_Customer non hanno una colonna dello scope store_id.

Inserisci, Aggiorna e Elimina

Per determinare se un inserto, l'aggiornamento o l'eliminazione devono essere eseguiti su un attributo, viene eseguito un confronto con l'oggetto originale. L'oggetto originale è

fondamentalmente un duplicato dell'oggetto dati quando l'entità è stata richiamata dal database.

- Se l'attributo esiste originariamente e il suo nuovo valore non è vuoto; si aggiorna.
- Se l'attributo esiste in origine ma il suo nuovo valore è impostato su vuoto; cancella. - Se l'attributo non esiste in origine e il suo nuovo valore non è vuoto; inserisce.

Gestione degli attributi

Modelli di attributi

Modello di attributo *Rappresenta l'attributo nel modulo del database, la sua logica è standard su tutti gli attributi ed è difficile da modificare* .

Modello di Frontend

L'interfaccia dell'attributo al frontend e fornisce qualsiasi logica richiesta dall'attributo sul frontend, ad esempio il metodo getUrl () sulle immagini.

Modello di backend

Questi eseguono la convalida sull'attributo prima che venga salvato nel database. Ad esempio, il modello di backend della password converte la password in un hash prima che venga salvata. Controlla inoltre che la password e la conferma della password corrispondano prima di salvare.

Modelli di origine

Utilizzato per popolare le opzioni disponibili per un attributo, ad es. Catalog / product_status è abilitato e disabilitato.

Metodi richiesti

Un modello sorgente richiede:

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

Di solito, solo getAllOptions () deve essere implementato anche se un'implementazione per getOptionText () esiste già nel modello sorgente astratto Mage_Eav_Model_Entity_Attribute_Source_Abstract.

Un modello di frontend non richiede il metodo getValue ().

Un modello di back-end richiede:

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

Tutti questi metodi sono implementati nel modello di backend astratto Mage_Eav_Model_Entity_Attribute_Backend_Abstract. Per i modelli di backend personalizzati, solo i metodi che richiedono la personalizzazione devono essere sovrascritti.

Modelli di origine della configurazione del sistema

Non può essere utilizzato per gli attributi EAV. I modelli di origine EAV implementano il metodo getAllOptions mentre i modelli di origine adminhtml implementano il metodo toOptionArray ().

I modelli di origine della configurazione di sistema di default sono disponibili in Mage / Adminhtml / Model / System / Config / Source /.

Attributi Modelli di origine

Lo scopo di Attribute Source Models è fornire l'elenco di opzioni e valori per gli attributi select e multiselect. Forniscono inoltre le informazioni della colonna all'indicizzatore della tabella flat del catalogo, se necessario.

Per ottenere un elenco di tutte le opzioni per un attributo, effettuare le seguenti operazioni:

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

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

Modelli di attributi predefiniti

Se nessuna classe viene specificata come frontend, backend o - per gli attributi select o multiselect - source models, viene utilizzata una classe predefinita.

Il modello di frontend degli attributi predefinito è Mage_Eav_Model_Entity_Attribute_Frontend_Default.

Il modello di back-end dell'attributo predefinito dipende dal codice dell'attributo ed è determinato nel metodo 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();
    }
?>
```

Se il metodo passa all'ultima riga, viene utilizzato Mage_Eav_Model_Entity_Attribute_Backend_Default.

Il modello di origine predefinito è impostato in Mage_Eav_Model_Entity_Attribute_Source_Table. Questo è impostato nel modello di attributo dei moduli del catalogo. Il modello di origine di configurazione predefinito specificato nel modulo eav non viene mai utilizzato.

Aggiungi attributo

Per aggiungere gli attributi EAV, utilizzare Mage_Eav_Model_Entity_Setup estendendo nella classe di installazione.

addAttribute () Crea gli attributi, aggiungilo a gruppi e insiemi (incluso quello predefinito) o aggiorna se già esiste updateAttribute () Aggiorna solo i dati degli attributi. Le classi di installazione personalizzate possono essere utilizzate per estendere questi metodi, aggiungendo dati aggiuntivi o semplificando gli argomenti necessari.

Tavoli piatti

Gli attributi di catalogo semplici sono gestiti dagli indicizzatori:

Mage_Catalog_Model_Resource_Product_Flat_Indexer :: updateAttribute () Mage_Catalog_Model_Resource_Category_Flat :: synchronize () Gli attributi del prodotto vengono aggiunti alla tabella flat se lo sono (vedere Mage_Catalog_Model_Resource_Product_Flat_Indexer :: getAttributeCodes ()):

Statico (tipo di back-end) Filtrabile Utilizzato nell'elenco dei prodotti Utilizzato per le regole promozionali Utilizzato per l'ordinamento in base agli attributi di sistema Esiste una tabella piatta diversa per ogni negozio, ognuno dei quali contiene un valore di attributo dell'entità con ambito negozio differente. I valori multilingue sono gestiti avendo diversi negozi per ogni lingua.

Examples

implementare l'interfaccia dei modelli di frontend, source e backend degli attributi

Interfaccia di frontend

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

}
```

Interfaccia di origine

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

Interfaccia di 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);
}
```

Leggi EAV (Valore attributo entità) online: https://riptutorial.com/it/magento/topic/7121/eav--valore-attributo-entita-

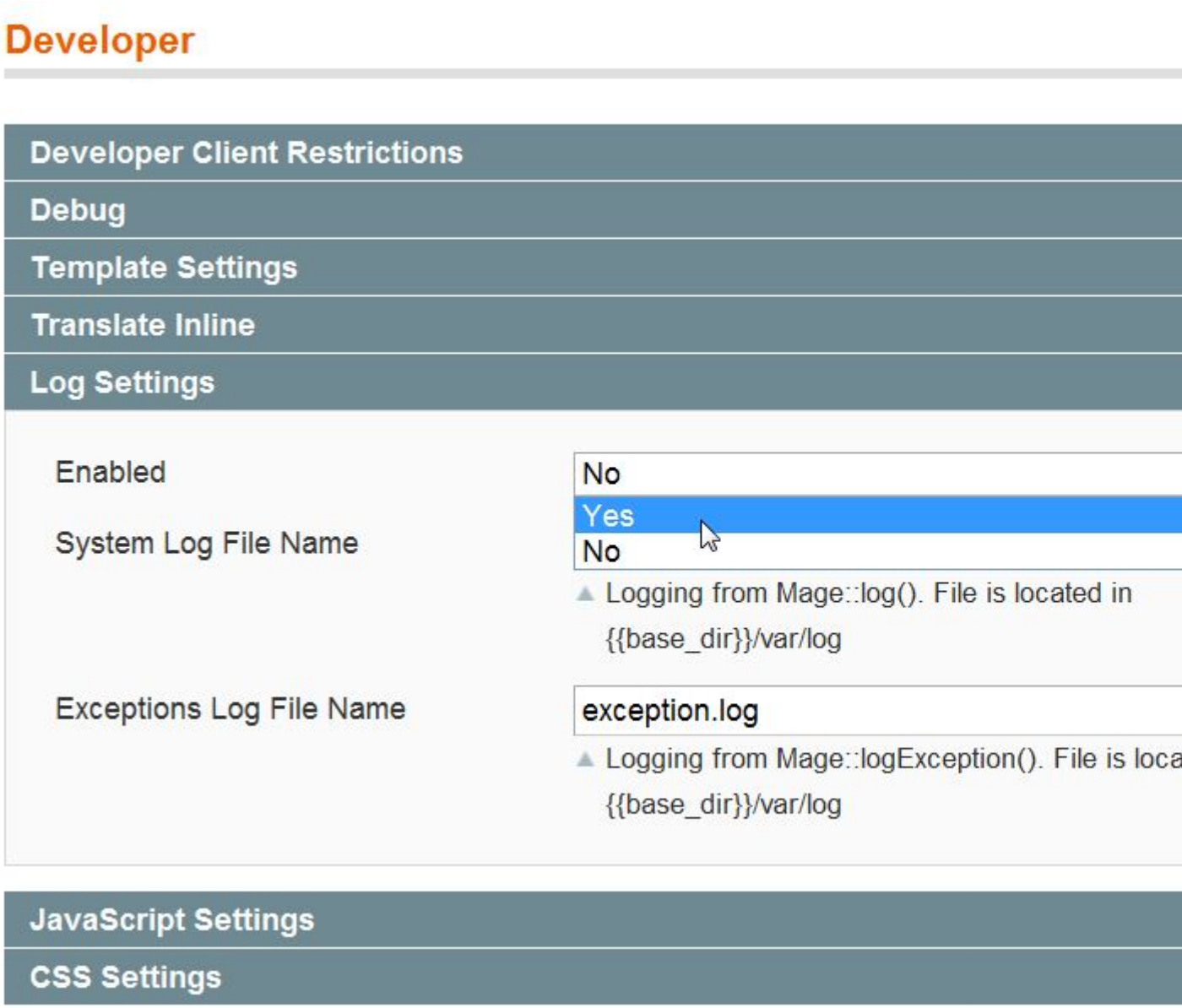
Capitolo 13: Gestione degli errori Magento, messaggi e report

Osservazioni

Posizioni del registro errori

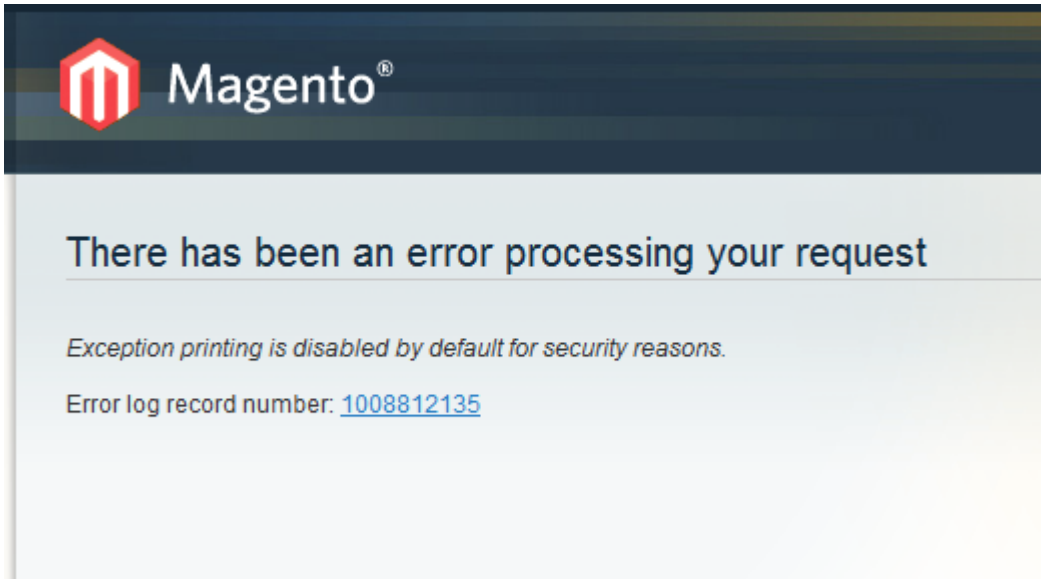
`/var/log/`

In genere il file system.log e exception.log esiste nella cartella `/var/log/` . Questi contengono la maggior parte delle informazioni di cui avrai bisogno. È possibile verificare se questi sono abilitati e quali sono i nomi dell'eccezione e del registro di sistema andando su `System > Configuration > System > Developer > Log Settings` .



/var/report/

I file di report vengono generati in questa cartella dopo che un utente ha riscontrato un errore. Ogni file include solo i dettagli per un errore. Questi sono usati per nascondere i dettagli dell'errore dal pubblico. Sulla pagina di errore ci sarà un numero di rapporto che è un riferimento al file corrispondente con lo stesso nome nella cartella `/var/report/` .



Examples

Abilita la visualizzazione della segnalazione degli errori

Nella pagina Indice cambia il seguente:

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

a

```
error_reporting(E_ALL);
```

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

e decommenta questa riga (rimuovi il `#`)

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

Puoi anche impostare la modalità Dev utilizzando `SetEnv` nel tuo file `.htaccess`

Per rendere l'errore leggibile e più facile da capire, effettuare le seguenti operazioni:

1. Apri la tua directory di installazione di Magento. Vai alla cartella degli errori.
2. Rinominare `local.xml.sample` file `local.xml` .

3. Aggiorna la pagina di errore nel browser.

Leggi Gestione degli errori Magento, messaggi e report online:
https://riptutorial.com/it/magento/topic/2369/gestione-degli-errori-magento--messaggi-e-report

Capitolo 14: Helpers

Examples

Creare un aiuto

Gli `Mage_Core_Helper_Abstract` dovrebbero estendersi da `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;
    }
}
```

Per poter accedere tramite `Mage::helper` è necessario definire un alias helper in un file `config.xml` per consentire al caricatore automatico Magento di trovare la classe:

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

Supponendo che il tuo modulo sia configurato correttamente e tu abbia svuotato la tua cache, dovresti essere in grado di usare il tuo helper in questo modo:

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

Nota: se stai usando una classe Data, il suo nome helper è implicito se non ne specifichi uno. Ad esempio, i seguenti due esempi sono identici:

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

Leggi Helpers online: https://riptutorial.com/it/magento/topic/5904/helpers

Capitolo 15: Magento nella cache

Examples

Come memorizzare dati personalizzati in 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;
```

Pulisci la cache per ID della cache

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

Svuota tutte le voci della cache Magento

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

o:

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

Usa Redis come backend cache

Configurazione Redis:

1. Installare redis (richiesto più di 2.4)
2. Installa phpredis
3. Installa l'estensione Magento `Cm_Cache_Backend_Redis` (solo per Magento 1.7 e sotto)

4. Modifica la tua `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>
```

Leggi Magento nella cache online: https://riptutorial.com/it/magento/topic/4902/magento-nella-cache

Capitolo 16: Modello

Examples

Carica modello

È possibile caricare il modello Magento utilizzando il seguente codice: Mage :: getModel ('nomedipo / nome modello')

Esempio: Mage :: getModel ('catalog / product') Questo caricherà Mage_Catalog_Model_product

Crea un modello vuoto

Per creare un nuovo modello nel modulo Aggiungi un `Model` cartella nella cartella principale del modulo e crea un file Nomename.php in questa cartella. per esempio Rick / Demo / Model / Modelname.php

Il nome della classe del tuo modello è importante chiamalo così:

```
<?php
class Rick_Demo_Model_Modelname {

}
```

assicurati che il tuo modello sia definito nel tuo `config.xml` nella cartella `etc` del tuo modulo

Ecco un esempio:

2.0.4 Rick_Demo_Model

Per caricare il tuo modulo usa il seguente codice:

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

Leggi Modello online: https://riptutorial.com/it/magento/topic/7665/modello

Capitolo 17: Ordini

Examples

Ottieni ordine per ID

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

Il codice sopra riportato è approssimativamente analogo alla seguente query SQL.

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

Ottieni l'ordine in base all'ID dell'incremento

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

Il codice sopra riportato è approssimativamente analogo alla seguente query SQL.

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

Il valore `increment_id` è l'identificativo dell'ordine del cliente, mentre `entity_id` è l'identificatore del livello del database per l'ordine.

Aggiungi commento alla cronologia degli ordini

È possibile aggiungere commenti e stato all'ordine. Ottieni l'ordine:

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

E aggiungi commento:

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

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

Leggi Ordini online: https://riptutorial.com/it/magento/topic/1556/ordini

Capitolo 18: Ottenere gli URL Magento

Sintassi

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

Parametri

percorso delle immagini	dettagli
esempio: *'images / my-images.jpg'*	percorso per immagine

Osservazioni

Ottieni l'URL delle immagini formattate ed evita le dipendenze del tema.

Examples

Nell'interfaccia / tema corrente

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

Ottieni Url della pelle

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

Ottieni l'Url di base

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

Secure Skin Url

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

Ottieni l'URL dei media

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

Url di pelle non sicuro

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

Ottieni l'URL del negozio

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

Ottieni Js Url

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

Ottieni l'URL corrente

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

Leggi Ottenere gli URL Magento online: https://riptutorial.com/it/magento/topic/2148/ottenere-gli-url-magento

Capitolo 19: Ottieni il nome del negozio e altri dettagli dalla configurazione del sistema

Examples

Ottieni il nome del frontend per la vista corrente del negozio

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

Ottieni l'ID del negozio corrente

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

Ottieni il codice negozio attuale

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

Questo restituisce il codice del negozio, ad esempio 'en' per un negozio che è impostato per l'inglese e chiamato 'en', e non l'id numerico.

Determina se la vista negozio è abilitata

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

Ottieni l'ID del sito web per il negozio corrente

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

Ottieni il modello attuale del negozio

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

Restituisce un'istanza di `Mage_Core_Model_Store`

Ottieni il nome del gruppo per il negozio

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

Ottieni tutti i negozi Magento

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

Restituisce una matrice di modelli `Mage_Core_Model_Store` .

Leggi Ottieni il nome del negozio e altri dettagli dalla configurazione del sistema online: https://riptutorial.com/it/magento/topic/1876/ottieni-il-nome-del-negozio-e-altri-dettagli-dalla-configurazione-del-sistema

Capitolo 20: Ottieni il nome della categoria dalla pagina del prodotto

Examples

Ottieni la categoria padre

```
$_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();
```

Ottieni la categoria corrente

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

Leggi Ottieni il nome della categoria dalla pagina del prodotto online: https://riptutorial.com/it/magento/topic/6078/ottieni-il-nome-della-categoria-dalla-pagina-del-prodotto

Capitolo 21: Ottieni l'utente corrente

Examples

Ottieni l'utente amministratore corrente

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

Ottieni cliente corrente

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

o

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

Controlla se l'utente è loggato

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

Leggi Ottieni l'utente corrente online: https://riptutorial.com/it/magento/topic/2157/ottieni-l-utente-corrente

Capitolo 22: Ottieni prodotti dal database

Examples

Ottieni il prodotto da sku

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

Ottieni il prodotto per ID

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

Raccolta del prodotto - query LIKE

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

Ottieni raccolta prodotti per attributo

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

Ottieni dati dall'oggetto del prodotto

```
// First load a product object

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

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

Ottieni raccolta di dati dalla forma del prodotto

```
// First load a collection object

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

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

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

Collezione di prodotti - con attributi

```
//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');
```

Controlla se il prodotto è stato caricato correttamente

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

Ottieni l'ID prodotto per SKU

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

Ottieni raccolta prodotti da un elenco di SKU

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

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

O

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

Imposta limite nella raccolta del prodotto

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

```
            ->setPageSize(20)
            ->setCurPage(1);
```

Leggi Ottieni prodotti dal database online: https://riptutorial.com/it/magento/topic/1102/ottieni-prodotti-dal-database

Capitolo 23: Ottimizzazione di Magento per la velocità

Examples

Ottimizzazione di Magento Modifica del file .htaccess

Magento è un'applicazione eCommerce molto popolare. Offre una grande quantità di personalizzazione e abilità dall'installazione iniziale. Ecco alcuni suggerimenti per l'ottimizzazione di un'installazione di Magento.

Abilitazione della compressione di output

Nel tuo file .htaccess per Magento troverai una sezione di testo che inizia con la linea,

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

Questa sezione di codice può essere utilizzata per attivare il modulo mod_deflate di Apache, che fornisce la compressione per testo, css e javascript. Dovrai decommentare (rimuovere il simbolo #) più righe in modo che assomigli a questo:

##

abilita la compressione dei file serviti da 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>
```

Abilitare le intestazioni di scade

I visitatori per la prima volta a qualsiasi pagina Web devono fare diverse richieste HTTP. Utilizzando l'intestazione "Expires" si rendono i componenti delle richieste memorizzabili nella cache. Ciò evita richieste HTTP non necessarie nelle successive visualizzazioni di pagina.

Vuoi trovare l'area del file .htaccess che inizia con <IfModulemod_expires.c> e finisce con il primo che vedi dopo, e fallo apparire così:

```
<IfModule mod_expires.c>

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

</IfModule>
```

Impostazioni amministratore

Unisci file JS e CSS

Questa particolare modifica ridurrà il numero di richieste HTTP sul tuo sito eCommerce. [box type = "alert" border = "full"] Nota: questo a volte può rompere alcune applicazioni. Dopo aver eseguito i seguenti passaggi, assicurati che il sito continui a funzionare come prima di abilitare questa funzione. [/ Box]

1. Accedi all'area di amministrazione e vai a - Sistema> Configurazione> Sviluppatore
2. In "Impostazioni JavaScript", modifica "Unisci file JavaScript" su Sì.
3. In "Impostazioni CSS", modifica "Unisci file CSS" su Sì.
4. Finalmente vorrai svuotare il tuo cache Magento.

Abilita cataloghi piatti

Il modello utilizzato da Magento per archiviare i risultati dei dati dei clienti e dei prodotti in query SQL più lunghe della media e più letture. L'abilitazione dell'opzione Catalogo semplice per Categorie e Prodotti unirà i dati di prodotto in un'unica tabella, migliorando quindi le prestazioni.

Accedi all'area di amministrazione e vai a - Sistema> Configurazione> Catalogo in "Frontend", cambia "Usa categoria catalogo flat" su sì. Sotto "Frontend", cambia "Use Flat Catalog Product" su yes - questo è opzionale. Successivamente, vorrai svuotare il tuo cache Magento. Infine, dovrai reindicizzare i tavoli. Abilita la compilazione

[box type = "alert" border = "full"] Nota: questo a volte può rompere alcune applicazioni. Dopo aver eseguito i seguenti passaggi, assicurati che il sito continui a funzionare come prima di abilitare questa funzione. [/ Box]

1. Accedi all'area di amministrazione e vai a - Sistema> Strumenti> Compilazione

2. Quindi, fai semplicemente clic sul pulsante Esegui processo di compilazione

3. Dopo che la compilazione è stata eseguita, dovrebbe attivarsi automaticamente

Abilita cache di sistema

1. Accedi all'area di amministrazione e vai a - Sistema> Cache Gestione
2. Quindi, fare clic sul collegamento Seleziona tutto
3. Infine, assicurati che le azioni siano impostate su Abilita e fai clic su Invia

Disabilita registrazione errori

Accedi all'area di amministrazione e vai a - Sistema> Configurazione> Sviluppatore Nella sezione Impostazioni registro, assicurati che Abilitato sia impostato su Nessun suggerimento per la manutenzione del database

Ci sono diverse tabelle usate da Magento per la registrazione. Mentre la registrazione è molto importante per sapere cosa ha e sta succedendo con il tuo negozio, i registri possono diventare grandi molto rapidamente, quindi una manutenzione regolare può essere di grande aiuto.

Ecco le tabelle per la registrazione:

```
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
```

Leggi Ottimizzazione di Magento per la velocità online:
https://riptutorial.com/it/magento/topic/8010/ottimizzazione-di-magento-per-la-velocita

Capitolo 24: Quick Cheat Sheet

Osservazioni

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

Molti molti frammenti sono molto utili

Examples

Ottieni la quantità di magazzino del prodotto

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

Leggi Quick Cheat Sheet online: https://riptutorial.com/it/magento/topic/7060/quick-cheat-sheet

Capitolo 25: Registrazione su file

Sintassi

- log delle funzioni statiche pubbliche ($ message, $ level = null, $ file = '', $ forceLog = false)

Parametri

Parametro	Dettagli
stringa $ messaggio	Il messaggio che verrà registrato
numero intero $	Livello di registro
stringa $ file	Percorso e nome con estensione del file che verrà salvato in `var/log/` . Se NULL o non specificato, verrà utilizzato `system.log` .
bool $ forceLog	Se impostato su `TRUE` log verrà scritto anche se la modalità sviluppatore è disattivata e la registrazione non è attiva.

Osservazioni

La registrazione è disattivata per impostazione predefinita a meno che la modalità sviluppatore sia attiva.

Tutte le eccezioni sono registrate in `exceptions.log` indipendentemente dal fatto che la registrazione sia abilitata nella configurazione.

La registrazione può essere abilitata accedendo a Magento Admin e procedendo a:

- Sistema> Configurazione (barra superiore)
- Sviluppatore (menu a sinistra)
- Sezione Impostazioni registro
- Selezionare Sì dall'elenco a discesa `Enabled` .
- Salva la configurazione nell'angolo in alto a destra.

Tipo di variabile del messaggio

Anche se la documentazione definisce che il messaggio dovrebbe essere una stringa, se viene passato un array c'è un blocco di codice in quel metodo per occuparsene con `print_r` :

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

Livello di registro

Se il parametro di livello è impostato su null, viene preso il livello DEBUG.

`$level = is_null($level) ? Zend_Log::DEBUG : $level;` I livelli sono dichiarati nel file: `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
```

Le costanti sotto forma di `Zend_Log::INFO` o il numero intero nell'intervallo specificato sopra possono essere passati come parametro del livello di registro.

Examples

File di registro personalizzato

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

Questo log accede a

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

Registrazione predefinita

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

Questo registrerà su `/var/log/system.log`

Oggetti e matrici vengono automaticamente scritti tramite una direttiva `print_r()` . Fai attenzione quando usi gli oggetti poiché questi possono avere dimensioni considerevoli.

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

Questo registrerà la stringa di traccia di eccezione su `/var/log/exception.log`

Leggi Registrazione su file online: https://riptutorial.com/it/magento/topic/2363/registrazione-su-file

Capitolo 26: Rendering

Osservazioni

Personalizzazione della funzionalità principale con temi

I temi hanno file di layout che, tra le altre cose, possono essere usati per cambiare i blocchi visualizzati sulla pagina. Il modello di blocco può anche essere cambiato e vengono chiamati diversi metodi.

Design a livello di negozio

I temi strutturati gerarchici di Magento indicano che un tema di base può essere esteso e assegnato a livello di negozio.

Registrazione di temi personalizzati

I temi possono essere configurati in tre modi:

1. Per archivio in Sistema> Configurazione> Progettazione.
2. Modifica del design con limiti di tempo Sistema> Design.
3. Le eccezioni ai temi possono anche essere impostate su una categoria e un livello di prodotto.

Pacchetto contro tema

Un pacchetto ha più temi. Ciascuno dei temi in un pacchetto eredita dal tema predefinito all'interno di un pacchetto.

Design Fallback

La procedura di fallback del tema per individuare i file di modelli è:

1. {Package} / {} tema
2. {Package} / default
3. Base / default

Per aggiungere ulteriori directory al meccanismo di fallback del tema, è necessario riscrivere il metodo Mage_Core_Model_Design_Package :: getFilename

Per l'area di amministrazione, il fallback è predefinito / predefinito.

Percorsi modello e layout

blocchi

I blocchi sono usati per l'output. Il root block è il genitore di tutti i blocchi ed è di tipo Mage_Page_Block_Html.

I blocchi Mage_Core_Block_Template utilizzano i file modello per il rendering del contenuto. Il nome del file modello è impostato all'interno di setTemplate () o addData ('modello') con percorsi relativi.

I modelli sono solo parti di PHP incluse in Mage_Core_Block_Template. Pertanto $ questo in un modello si riferisce al blocco.

Mage_Core_Block_Template utilizza un buffer prima di includere un modello per impedire l'output prematuro.

Il metodo Mage_Core_Model_Layout :: createBlock crea istanze di blocchi.

La classe Mage_Core_Model_Layout_Update considera quali blocchi devono essere creati per ciascuna pagina esaminando gli handle del layout.

Tutti i blocchi di output sono resi, ad esempio chiamando aHtml (), che a sua volta può scegliere di rendere i propri figli.

Text e Text_List bloccano automaticamente il rendering del loro contenuto.

Ci sono due eventi che vengono attivati attorno al rendering del blocco che può essere utilizzato per modificare il blocco prima e dopo il rendering dell'HTML:

core_block_abstract_to_html_before core_block_abstract_to_html_after Un blocco figlio verrà reso automaticamente solo se appartiene alla classe Mage_Core_Block_Textlist, altrimenti il metodo getChildHtml deve essere chiamato.

È possibile accedere alle istanze di blocco tramite il layout, ad es. Mage :: app () -> getLayout () e $ controller-> getLayout (). L'output del blocco è controllato dalla funzione _toHtml ().

I modelli sono resi dai metodi renderView () / fetchView () all'interno di un blocco template. Il buffering dell'output può essere disabilitato con $ layout-> setDirectOutput.

È possibile aggiungere un blocco al layout corrente, ma è necessario farlo prima che venga chiamato il metodo renderLayout ().

Layout XML

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

I file di layout possono essere registrati in config.xml:

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

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

L'output della pagina può essere personalizzato nei seguenti modi:

- Modifiche al modello
- Modifiche al layout
- Sovrascrivere i blocchi
- Osservatori Le variabili sui blocchi possono essere impostate nei seguenti modi:
- Layout: azioni o attributi
- Controller - $ this-getLayout () -> getBlock ()
- Blocchi figlio - $ this-> getChild ()
- Altro -Mage :: app () -> getLayout ()

Capi di testa

Le risorse JavaScript e CSS sono gestite nel blocco Mage_Page_Block_Html_head. Questo blocco gestisce la fusione delle risorse in un singolo file per ridurre al minimo le richieste HTTP. Il file unito si basa sul tempo di modifica dei file di origine.

Quando si fondono i CSS, viene richiamata una funzione di callback su Mage_Core_Model_Design_Package per aggiornare le direttive @import o url () con gli URL corretti

Examples

Diversi meccanismi per disabilitare l'output dei blocchi

- Se la risposta è già stata creata e impostata sull'oggetto risposta al di fuori del normale processo di rendering (ad esempio, in un osservatore), il flag 'no-renderLayout' può essere impostato sul controller di azione usando

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

- Ciò impedisce a `renderLayout()` di elaborare i blocchi di output.
- Lo stesso può essere ottenuto chiamando setNoRender (true) sul front controller: `Mage::app()->getFrontController()->setNoRender(true);`
- L'impostazione del `isDispatched()` sull'oggetto risposta potrebbe essere più efficiente per ottenere un effetto simile.

diversi tipi di blocchi

- 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 Tutti i blocchi sono blocchi **strutturali** o Tutti i blocchi sono blocchi strutturali o blocchi di contenuto. Esempio:
- **core / text_list**

Esempio di un **blocco strutturale** .

Non utilizza modelli; è semplicemente usato per generare il contenuto di tutti i suoi blocchi figlio uno dopo l'altro.

- **core / template**

Esempio di un **blocco di contenuti** .

L'output di questo tipo di blocco dipende dal modello assegnato. I suoi blocchi figlio vengono visualizzati nel suo modello tramite il metodo getChildHtml ('block_name'). Esempio: core / text_list - Esempio di un blocco strutturale. Non utilizza modelli; è semplicemente usato per generare il contenuto di tutti i suoi blocchi figlio uno dopo l'altro. core / template - Esempio di un blocco di contenuti. L'output di questo tipo di blocco dipende dal modello assegnato. I suoi blocchi figlio vengono emessi all'interno del suo modello tramite il metodo getChildHtml ('block_name').

È possibile accedere alle istanze di blocco dal controller

Da un controller di azioni:

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

Leggi Rendering online: https://riptutorial.com/it/magento/topic/7140/rendering

Capitolo 27: Script Sql per cancellare i dati del test

introduzione

Script Sql per eliminare i dati di test di prodotti, clienti, registri e vendite.

Examples

Elimina i dati del test del cliente

```
SET FOREIGN_KEY_CHECKS=0;

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

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

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

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

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

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

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

SET FOREIGN_KEY_CHECKS=1;
```

Elimina i dati del test del prodotto

```
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;
```

Elimina i dati del test di vendita

```
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;
```

Elimina i dati dei test dei registri

```
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;
```

Leggi Script Sql per cancellare i dati del test online:
https://riptutorial.com/it/magento/topic/9263/script-sql-per-cancellare-i-dati-del-test

Capitolo 28: Shell, CLI

Osservazioni

Nozioni di base

- È necessario avere una riga di comando Linux o connettersi utilizzando SSH al server per poter utilizzare gli script di shell.
- Vai alla tua `MAGENTO_ROOT/shell`
- Lo script può essere eseguito digitando ie

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

Metodi di shell core per file

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

Script di shell php personalizzati

A volte abbiamo bisogno di accedere a Magento al di fuori di un browser web per omettere i tempi di esecuzione o impostare cose diverse che non influenzeranno il frontend.

Ci sono 2 modi per avviare Magento, ma solo uno è il modo Magento. Leggi di più sopra nella sezione degli esempi.

Examples

Utilizzo della shell senza estendere Mage_Shell_Abstract

Avvio automatico di Magento chiamando:

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

Questo è il modo più semplice ma non proprio il modo Magento perché non stiamo usando la classe che estende `Mage_Shell_Abstract` - la classe che quando estesa ci fornisce gli strumenti per analizzare gli argomenti della riga di comando, chiama `__applyPhpVariables()` nel suo costruttore (la funzione analizza i file .htaccess e applica le impostazioni php allo script di shell).

Usando la shell in modo Magento - estendi Mage_Shell_Abstract

Magento way

Il file risiede in `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();

}
```

Esecuzione di Reindex dalla CLI

Visualizza stato:

```
php indexer.php status
```

Reindex All

```
php indexer.php reindexall
```

Indice specifico di reindicamento

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

Elenco dei singoli codici

Indice	Codice
Caratteristiche del prodotto	catalog_product_attribute
Prezzi del prodotto	catalog_product_price
L'URL del catalogo riscrive	catalog_url
Dati piatti del prodotto	catalog_product_flat
Categoria dati piatti	catalog_category_flat
Prodotti di categoria	catalog_category_product
Indice di ricerca del catalogo	catalogsearch_fulltext
Stato delle scorte	cataloginventory_stock

Leggi Shell, CLI online: https://riptutorial.com/it/magento/topic/5387/shell--cli

Capitolo 29: Struttura del modulo

Osservazioni

I moduli esistono da estendere. Non è possibile modificare `the app/code/` file senza proibire aggiornamenti futuri. Invece, aggiungiamo un modulo `app/code/local` directory (la directory locale potrebbe mancare, in tal caso, deve essere creata manualmente. Questo è comune nelle versioni successive di Magento) per aggiungere funzionalità locali personalizzate.

Tutti i file di configurazione del modulo iniziano con un tag `<config>` . Il nuovo modulo è dichiarato all'interno del tag `<modules>` . Chiameremo un modulo chiamato YOUR_COMPANY_HelloWorld, quindi vengono utilizzati i tag `<YOUR_COMPANY_HelloWorld>` . Qui definiamo la versione (molto prima = 0.0.1)

Un elenco di **eventi XML** può essere trovato su: http://www.magentocommerce.com/wiki/5_-_modules_and_development/reference/module_config.xml

Se hai problemi, dai un'occhiata a: http://coding.smashingmagazine.com/2012/11/30/introducing-magento-layout/

Examples

Creazione di un modulo da zero (Hello World)

Lo sviluppo di moduli personalizzati Magento è una parte fondamentale di qualsiasi sviluppo Magento o progetto Magento, perché in qualsiasi fase potresti voler integrare la tua funzionalità / modulo nel tuo progetto Magento esistente.

La prima cosa che gli sviluppatori dovrebbero disabilitare è la cache di sistema. In caso contrario lo sviluppo diventerà un dolore poiché qualsiasi cambiamento richiederà un lavaggio. Dal pannello di amministrazione di Magento: vai su `System > Cache Management > Select All > Actions : Disable.` Utilizzare la seguente guida per creare un nuovo modulo:

- Crea una cartella in `app/code/local/` - le convenzioni di denominazione di solito prendono il nome della tua azienda, ad esempio `app/code/local/<YOUR_COMPANY>` .
- Ora abbiamo una posizione personalizzata per i nostri moduli. Crea un'altra directory, chiamala qualcosa relativa al tipo di modulo che vuoi creare, ad esempio app / code / local / <TUA_COMPANY> / HelloWorld / - "HelloWorld" è ciò che chiamerò questo modulo.
- Questa directory ha bisogno di un `config.xml` quindi Magento lo riconosce come un nuovo modulo. Crea un'altra cartella denominata `etc` . Seguito da un file xml chiamato config.xml. La directory dovrebbe apparire come `app/code/local/<YOUR_COMPANY/HelloWorld/etc/config.xml` Questa è la struttura del file xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<config>
   <modules>
      <YOUR_COMPANY_HelloWorld>
         <version>0.0.1</version>
      </YOUR_COMPANY_HelloWorld>
   </modules>
</config>
```

- Successivamente, i moduli devono essere dichiarati a Magento. Procedi con l' `app/etc/modules` . Crea un altro documento XML e `YOUR_COMPANY_HelloWorld` nomi scelti dei tuoi tag: `YOUR_COMPANY_HelloWorld` nel mio caso. In questo documento scrivi:

```
<?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>
```

- Di nuovo i tag config e module sono usati per dichiarare un nuovo modulo a Magento. Attivo è il valore predefinito a cui è possibile accedere nel `Admin Panel under System > Configuration > Advanced` . `codePool` dice a Magento quale directory cercare. `local` nel nostro caso

- Questo modulo è stato ora impostato, quindi il modello della nostra struttura MVC. dovresti essere in grado di vedere il tuo nuovo modulo nel Pannello di amministrazione in `System > Configuration > Advanced` . **Tuttavia, non fa ancora niente!** Dovrai tornare al nostro file config.xml e definire gli elementi XML.

- Seguendo il tutorial; Useremo alcuni di questi Elementi XML per creare classi e manipolare tutte le pagine nel frontend del nostro sito. Torna al file `config.xml` scrivi quanto segue sotto il `</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>
```

- Come puoi vedere, stiamo estendendo costantemente anziché manipolare i file principali. Il tag `helloworld` è in minuscolo perché indicherà un handle e per la continuità lo `helloworld` più vicino possibile. Quindi colleghiamo questo al modulo `YOUR_COMPANY_HelloWorld` .

- Stiamo cambiando il layout. Pertanto, dobbiamo creare questa maniglia nella directory di layout. procedere ad `app/design/frontend/base/default/layout` . Abbiamo detto al modulo di cercare il file `helloworld.xml` . Pertanto dobbiamo crearlo in questa directory. Che cosa state aspettando. Fallo!! e popolarlo con:

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

- Ora, quelli di voi che hanno una piccola esperienza su Magento, o che hanno letto altri tutorial Magento degni di nota, potrebbero essere senza fiato sul fatto che stiamo apportando delle modifiche in base / default dato che qui si trovano i file principali di Magento. Tuttavia, non stiamo modificando alcun file qui, ne stiamo creando di nuovi, e inoltre stiamo prefiggendo il nostro nome di file con "helloworld", quindi ci sono pochissime possibilità che questo sia in conflitto con altri moduli o che causi problemi con l'aggiornamento di Magento nel futuro. Giorni felici!

- Poiché vogliamo influenzare tutte le pagine, utilizziamo il tag predefinito e lo riferimento al blocco strutturale `before_body_end` . Questo agirà nel ruolo dell'Action e attiverà la sezione View della nostra struttura MVC.

- Ora capiamo che ci stiamo imbullonando sul blocco `before_body_end` . e collegandolo al nostro blocco personalizzato. Questo è chiamato un riferimento ed è un **gancio** . Al momento non lo stiamo collegando a nulla esistente, quindi dobbiamo creare i file necessari.

- In `helloworld.xml` abbiamo indicato nel template un `footer.phtml` . Procedere ad `app/design/frontend/base/default/template` e creare una directory `helloworld` .

- All'interno di questa directory crea il file `footer.phtml` e `footer.phtml` HTML, questo tutorial lo

scrive semplicemente per mostrare alcune funzionalità PHP collegate al nostro file PHTML:

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

- Ora dobbiamo creare il nostro oggetto di blocco per accoppiare il modello con la nostra funzionalità di blocco. Crea l'app di directory / code / local / YOUR_COMPANY / HelloWorld / Block / e crea il file `Footer.php` all'interno di questo. Questo è stato fatto riferimento nel nostro zeta_layout.xml nel tipo "helloworld / footer". Compila questo file con:

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

- Questa è la funzionalità che popolerà la nostra chiamata `getDate()` abbiamo chiamato dal nostro file `.phtml` . Estendiamo il `Mage_Core_Block_Template` .
- Questa funzionalità è ora completa. Prova questo fuori andando alla tua home page dove dovresti vedere il tuo modulo nel piè di pagina di ogni pagina!

Leggi Struttura del modulo online: https://riptutorial.com/it/magento/topic/3962/struttura-del-modulo

Capitolo 30: Struttura MVC

Osservazioni

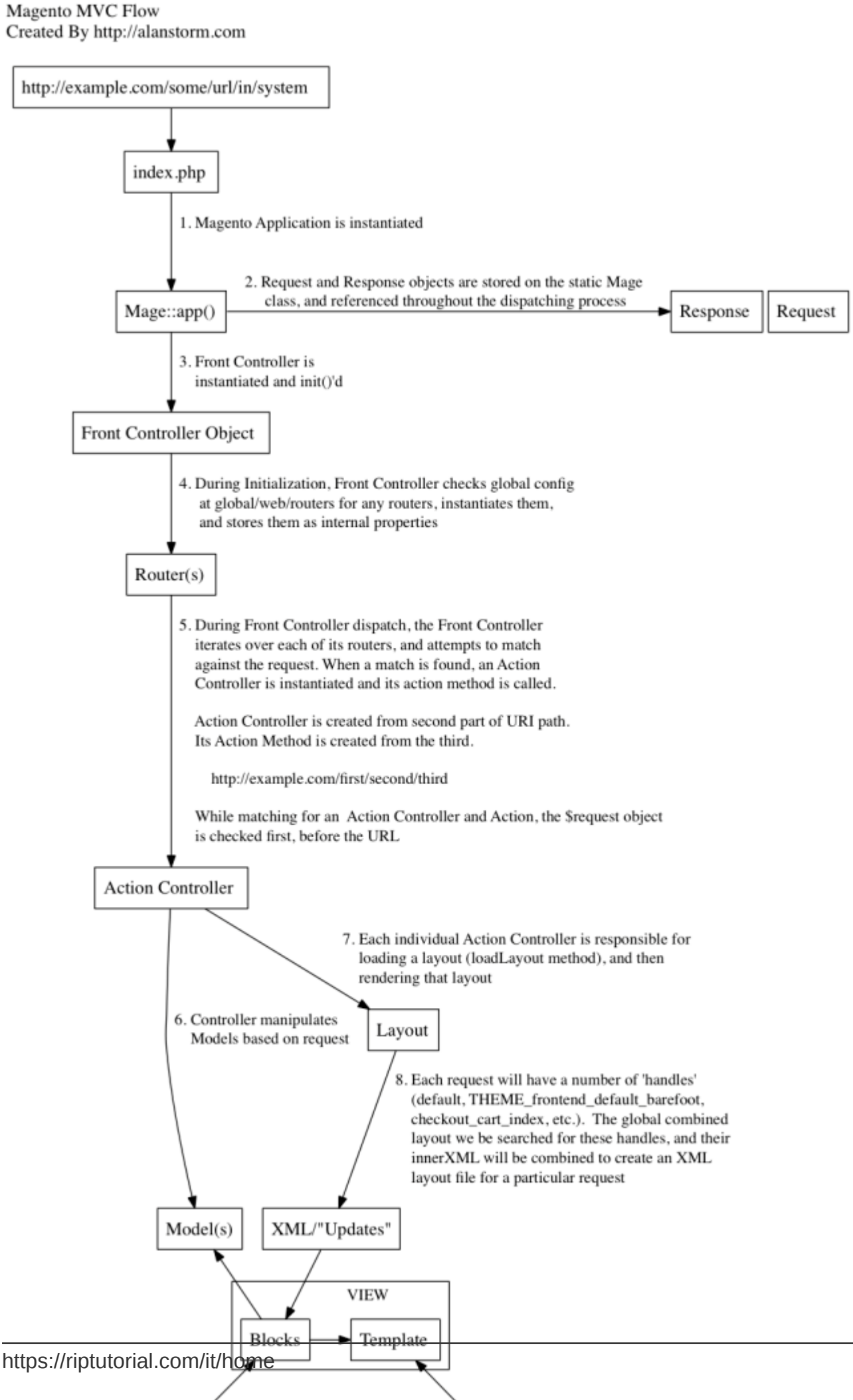
MVC è l'acronimo di Model-View-Controller. Qualsiasi applicazione che separa l'accesso ai dati, la logica aziendale e l'interfaccia utente è chiamata MVC. Ci possono essere due tipi di MVC: basati sulla convenzione e basati sulla configurazione. Ad esempio, cakePHP è basato sulla convenzione, ovvero basta seguire le istruzioni del sistema principale per preparare il modulo in poche righe. Magento è basato sulla configurazione, ovvero devi specificare ogni cosa nel file di configurazione del tuo modulo per farlo funzionare. Magento ha un controller (per il routing), un blocco, un modello e un file modello. Come funziona MVC di Magento:

1. Quando inserisci l'URL (qualcosa come http://mysite.com/frontname/controller/method/param1/value1/param2/value2) , questo URL viene intercettato da un file PHP chiamato index.php che istanzia l'applicazione Magento
2. L'applicazione Magento crea un'istanza dell'oggetto Front Controller
3. Inoltre, il front controller istanzia oggetti Router (specificati nel modulo config.xml, tag globale)
4. Ora, il router è responsabile per "abbinare" il nome che è nel nostro URL
5. Se viene trovata la "corrispondenza", vede il nome del controller e il nome del metodo nell'URL, che viene infine chiamato.
6. Ora, a seconda di ciò che è scritto nel nome dell'azione (nome del metodo), viene eseguito. Se vengono chiamati tutti i modelli, il metodo del controller creerà un'istanza di quel modello e chiamerà il metodo in esso richiesto.
7. Quindi l'azione del controller (metodo) crea un'istanza dell'oggetto Layout, che chiama il blocco specificato per questo nome di azione (metodo). Ogni nome di azione del controller ha un blocco e un file modello associato, che può essere trovato in app / design / frontend o adminhtml / namespace / module / layout / module.xml file, il nome del file di layout (module.xml) può essere trovato in config.xml di quel modulo, nel tag di aggiornamento del layout).
8. Il file di modello (.phtml) ora chiama il blocco corrispondente per qualsiasi richiesta di metodo. Quindi, se scrivi $ this-> methodName nel file .phtml, controllerà "methodName" nel file di blocco associato nel file module.xml.
9. Il blocco contiene la logica PHP. Fa riferimento a Modelli per qualsiasi dato dal DB.
10. Se Block, Template o Controller devono ottenere / impostare alcuni dati da / a database, possono chiamare Model direttamente come Mage :: getModel ('nomedipo / nome modello').

Examples

Comprendi MVC in Magento

MVC Flow in 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

Capitolo 31: URL corrente

Sintassi

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

Examples

Homepage

ritorno: http://www.example.com/

Pagina del prodotto

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

Controlla se l'url corrente è sicuro

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

Questo restituirà true se l'url corrente è sicuro.

Leggi URL corrente online: https://riptutorial.com/it/magento/topic/2150/url-corrente

Capitolo 32: URL immagine prodotto

introduzione

Ottieni gli URL delle immagini del prodotto per miniature, immagini di piccole dimensioni e immagini di base. Ricevi anche gli URL dei media diretti nella cache e non nella cache.

Examples

URL di immagini memorizzate nella 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 immagine non memorizzati nella cache dal supporto

```
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
```

Leggi URL immagine prodotto online: https://riptutorial.com/it/magento/topic/9262/url-immagine-prodotto

Capitolo 33: URL specifici

Examples

Carrello url

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

O

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

URL di verifica

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

O

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

Login url

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

O

Mage :: helper ('cliente / dati') -> getLoginUrl ();

URL di disconnessione

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

O

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

Hai dimenticato l'URL della password

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

O

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

URL cliente account

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

O

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

Media, JS, Skin URL

Per recuperare il percorso URL in STATIC BLOCK o pagine CMS

Per ottenere l'URL SKIN

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

Per ottenere l'URL del supporto

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

Per ottenere l'URL del negozio

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

Per ottenere l'URL di base

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

PER Recuperare il percorso dell'URL in PHTML

URL skin non sicuro

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

Secure Skin URL

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

Ottieni l'URL corrente

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

Ottieni l'URL della tua home

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

Ottieni l'URL multimediale di Magento

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

Ottieni l'Url Skin Magento

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

Ottieni l'URL del negozio Magento

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

Ottieni Magento Js Url

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

Leggi URL specifici online: https://riptutorial.com/it/magento/topic/2144/url-specifici

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con Magento	7ochem, Chris Rogers, Community, Gabriel Somoza, goutam, Henry's Cat, Luke Rodgers, Marek Skiba, Pawel Dubiel, RamenChef, Robbie Averill, Stephen Leppik
2	Aggiungi un prezzo diverso per più negozi utilizzando l'API SOAP Magento	Harsha Sampath
3	Attributi personalizzati	Twinkal
4	Collezione completa di prodotti	Twinkal
5	collezioni	gebrial, gulshan maurya
6	Come filtrare le raccolte	wesleywmd
7	Comprensione dei tipi di prodotto	lalit mohan
8	Crea carte regalo aziendali a livello di programmazione	JPMC, RamenChef
9	Dati del negozio e del sito web	Twinkal
10	EAV (Valore attributo entità)	lalit mohan
11	Gestione degli errori Magento, messaggi e report	7ochem, Charles, Pankaj Pareek, RamenChef
12	Helpers	Robbie Averill
13	Magento nella cache	RamenChef, Robbie Averill, Yogendra - eCommerce Developer
14	Modello	Rickert

15	Ordini	bpoiss, Hemant Sankhla, Luke Rodgers
16	Ottenere gli URL Magento	Nolwennig, Robbie Averill, Steven Church
17	Ottieni il nome del negozio e altri dettagli dalla configurazione del sistema	Henry's Cat, Robbie Averill
18	Ottieni il nome della categoria dalla pagina del prodotto	user2925795
19	Ottieni l'utente corrente	bpoiss, Rickert
20	Ottieni prodotti dal database	Akif, Alex, baoutch, bpoiss, ctrimm, gulshan maurya, jignesh prajapati, mnoronha, Olavi Sau, pce, SH-
21	Ottimizzazione di Magento per la velocità	Lemon Kazi
22	Quick Cheat Sheet	Chris Richardson
23	Registrazione su file	Akif, gulshan maurya, versedi
24	Rendering	lalit mohan
25	Script Sql per cancellare i dati del test	Twinkal
26	Shell, CLI	djdy, versedi
27	Struttura del modulo	Chris Rogers, RamenChef
28	Struttura MVC	lalit mohan
29	URL corrente	Nolwennig, Twinkal
30	URL immagine prodotto	Twinkal
31	URL specifici	Mayank Pandeyz, Nolwennig, Qaisar Satti, Yogendra - eCommerce Developer