

Magento Developer User Guide

Version 1.0

Submitted by

TheUnical Technologies

theunical.com | blog.theunical.com

info@theunical.com | support@theunical.com

All the content of this document is taken from Magento Wiki, Blog and other references. The Purpose of this document is to help those new users.

TheUnical Technologies

Table of Contents

Magento Developer User Guide	9
Add Featured Products to Home Page	9
Add Bestsellers to Home Page using the best-selling box of Blue Theme.....	9
How to Fix the Breadcrumbs of Inactive Categories.....	11
Create Payment Method Module	13
Create Shipping Method Module	21
Module doesn't appear in frontend.....	35
Module doesn't appear in admin.....	36
Backend shows error message when displaying the shipping method.....	36
Magento Architecture.....	37
Page request flow	37
Preliminary core modules dependency diagram	37
Changing and Customizing Magento Code	39
Subversion.....	39
Upgrading to a newer version.....	40
Custom Modules	42
Blocks	43
Step 2) Add Block configuration to catalog.xml	45
Step 4) Extend Mage_Catalog_Block_Category_View	47
Step 6) Add new blocks to the app/etc/local.xml.....	48
Magento Easy Lightbox.....	49
http://www.magentocommerce.com/extension/1487/magento-easy-lightbox	49

CSS Resources	50
Creating CSS buttons vs Image buttons	52
The CSS.....	53
Add Home Link with functional active state to Menu Bar (Alternative Method).....	55
Magento’s Import/Export Profile.....	60
The New Magento Connect Design (part 1)	62
Installing Magento Enterprise stand-alone on OS X	63
Pre-Production System Configuration Checklist	65
1: System -> Configure -> General.....	65
2: System > Configuration > Sales	66
3: System -> Advanced	67
4: System -> Transactional Emails.....	68
Top-10 Most Popular Magento Extensions This Week (Jan 31 - Feb. 5)	70
Multi-site Domain Name Setup	71
1: Categories	71
2: Store Configuration in the Magento Admin	71
3: Store Configuration in the Server	74
4: We’re Ready to Go!.....	75
Magento Free Shipping.....	76
Changing the Magento ‘favicon’	79
How to Embed Google Custom Search in Magento	80
Reward Points System.....	84
Reward Points System.....	84
Magento Development: Introduction to Magento Dataflow	85
Dataflow profile definition	85

Adapter definition	85
Magento DataFlow standard adapters	86
Customer and Product adapters	86
Parser definition	88
Magento DataFlow standard parsers	88
Standard customer and product entity parsers	89
Mapping values	90
Importing Newsletter Subscribers.....	91
Varien’s Popular Open Source Magento eCommerce Software to Ship with Zend Server Community Edition PHP Stack.....	93
Availability.....	94
Replacing the Logo Image in Transactional Emails.....	95
How to Configure Magento Widgets.....	96
Overview.....	96
What are Widgets?.....	96
How to Develop a Widget.....	98
Terminology.....	98
Widget Examples in Magento CE 1.4, EE 1.6.....	99
Tutorial: Creating a Magento Widget, Part 1	100
Introduction.....	100
Widget Basics.....	100
Tutorial: Creating a Magento Widget, Part 2	112
Introduction.....	112
Available widget configuration options, types and definitions.....	112
Multiple Websites, Importing Catalog with Different Price and Currencies.....	123
Magento Connect Extensions for Facebook Integration.....	125

Embedding HTML in the Footer	126
Limiting Free Shipping to the Continental US.....	127
Tax Rules Configuration and Settings	128
Example of files edited in Excel (when opening on notepad or any other text editor	128
Proper file format with double quotation marks.....	128
Video: Security, Permission Roles, Encryption, PA-DSS and Logging in the Magento Enterprise Edition	129
Promotions, Discounts and Conditional Selection.....	130
Issue	130
Solution	130
Magento Bug Tracker RSS	132
Rich Merchandising Suite (RMS)	133
CMS+, Enhanced Content Management System	134
Magento Database Repair Tool	135
Magento Database Repair Tool More.....	136
Usage Instructions	136
Crash-course for the impatient.....	136
Test it before running on a Production Environment!.....	136
Step-by step	137
How to Set Up a Cron Job	139
Magento and crontab.....	139
Windows.....	141
Other solutions.....	142
Inner workings.....	142
Error logging	143
Configuration.....	143

Magento's Cronjobs	145
How to restore a broken admin access	146
Log in with the new account	147
Further information	147
Using Magento on Amazon EC2.....	148
Optimizing Performance with Apache	148
1. MySQL Configuration	149
2. Apache KeepAlives.....	149
3. PHP Opcode Cache.....	150
4. Memory-based Filesystem	150
Public and Free Magento AMIs.....	150
Launching an EC2 Magento Demo Store	151
Benchmarking with ApacheBench	151
Benchmarking with Pingdom	151
Optimizing Performance with Nginx.....	152
Data Persistence with EBS (Elastic Block Storage).....	152
EC2 Pricing and FAQs	155
Performance Improvements in Next Magento Release	155
Todos.....	155
Understanding Magento Scalability and Performance	156
Easy Wins	156
Measure your Magento	158
Three Steps to Improve Scalability and Performance.....	160
Going Further	161
Conclusion.....	162

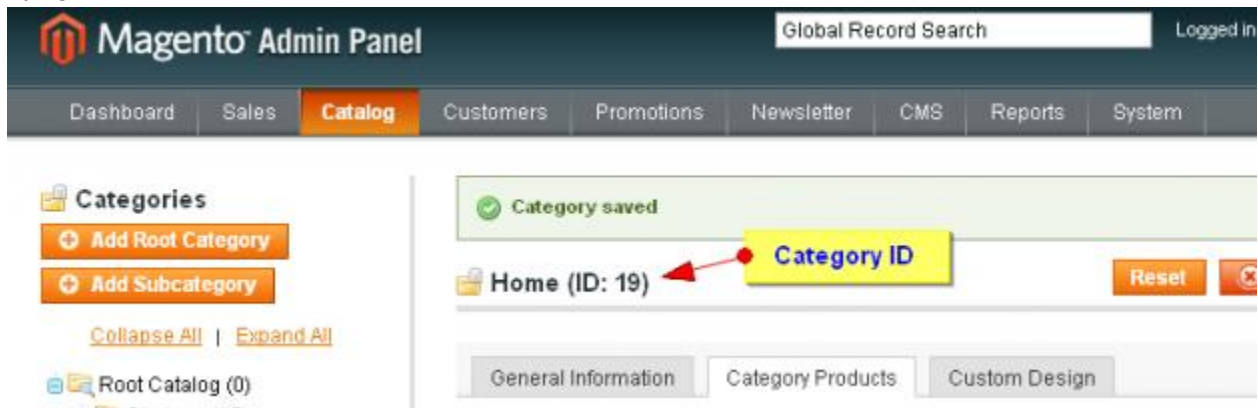
Gift Cards and Customer Store Credits in the Magento Enterprise Edition	163
Improve Conversions in 60 Seconds	164
Magento & Zend Server Benchmarks	165
Methods & Tools Used.....	165
Magento Version Benchmarks.....	166
Apache + mod-php VS Apache + Zend Server.....	167
Zend Server Configuration for Magento	168
Response time.....	170
Additional technical details about the tests	172
Things to test in the next benchmark campaign.....	172
Conclusion.....	173
Content Staging and Merging in the Magento Enterprise Edition.....	174
Speed up your store by combining, compressing and caching JS and CSS-- Fooman Speedster	175
Upgrades Made Easy.....	177
CSRF Vulnerability in Web Applications (and how to avoid it in the Magento Admin)	178
Live Chat Extension	179
Magento Connect: Gift Certificates / Virtual Cards Extension	180
How To Setup Multiple Magento Stores.....	181
TYPO3 + Magento = TypoGento.....	182
Google Base Integration in Magento	182
Video: Google Website Optimizer Integration in Magento 1.1.7	182
Tutorial: Integrating 3rd Party CMS Content Within Magento.....	183
Getting Started.....	183
Create a Local Code Pool Module	184
Create a Layout	185

Extending the Technique	185
Magento Connect: Simple Configurable Products Extension	186
A/B Split & Multivariable Testing with Google Website Optimizer	186
50+ Payment Gateways Now Supported in Magento.....	188
Blank Theme available through MagentoConnect	190
OpenERP Integration Available Via Magento Connect	191
Magento in 60 languages!.....	192
We have 23 locales that are 85% - 100% complete.....	192
13 locales that are actively translated (20% - 85%).....	192
And 24 more locales that are in the process:	193
Magento Connect: Quickbooks Integration via T-HUB.....	194
Video: Magento via iPhone.....	195
Wordpress Integration Extension Available Via Magento Connect.....	195
Video: Custom Product Options in Magento 1.1	196
Magento PHPDocs Now Available	196
osCommerce Migration Tool - Now Available	197
Building Configurable Products Fast	197
Magento / Drupal Integration Project	198

Magento Developer User Guide

Add Featured Products to Home Page

- Create a category to contain the featured products. Call it e.g. "Featured" or "Home Page". Set "Is Active" to No. That way, it won't display in the catalog menu. - After saving the category, please note what ID it gets. You can see it in the last part of the URL. If the URL ends with `catalog_category/edit/id/8/`, the ID is 8. On later version, the ID is next to the category name:



- Add products for the home page to the new category. - Edit the Home Page (CMS → Manage Pages → Home Page) and add the following content, where 8 should be replaced by your category ID:

```
{{block type="catalog/product_list" category_id="8" template="catalog/product/list.phtml"}}
```

 If your `product/list.html` references `$this->getColumnCount()` you can vary the column count (e.g. 4 columns) from the default (3) displayed by using:

```
{{block type="catalog/product_list" column_count="4" category_id="8" template="catalog/product/list.phtml"}}
```

Although displaying more than 3 columns in your template would likely require additional CSS/layout changes as well.

If you want a view that is different from the category lists, you can copy and modify `list.phtml` and change the path above. Following steps is an example.

Add Bestsellers to Home Page using the best-selling box of Blue Theme

In Blue Theme, there is a sample HTML code in the content box (Admin → CMS → Manage Pages → Home Page) that displays a list of Best Selling Products. You can easily modify `list.phtml` to reuse the skin of the best-selling box. Adding featured products to Home Page is a matter of clicking to select the required products in the category (Admin → CMS → Manage Categories → Category Products tab) as outlined in the above method and there is no need to mess with the HTML code in the CMS.

1. Copy the following code to your preferred text editor

```
<?php $_productCollection=$_this->getLoadedProductCollection() ?>
<?php if(!$ _productCollection->count()): ?>
<div class="note-msg">
  <?php echo $this->__('There are no products matching the selection.') ?>
</div>
<?php else: ?>
<div class="box best-selling">
<?php $ _collectionSize = $ _productCollection->count() ?>
<table border="0" cellspacing="0">
<tbody>
  <?php $i=0; foreach ($ _productCollection as $ _product): ?>
    <?php if($i++%2==0): ?>
      <tr>
        <?php endif; ?>
        <td>
          <a href="<?php echo $ _product->getProductUrl() ?>" >
            htmlEscape($ _product-
>getName()) ?>" />
          </a>
          <div class="product-description">
            <p><a href="<?php echo $ _product->getProductUrl() ?>" title="<?php echo $this-
>htmlEscape($ _product->getName()) ?>"><?php echo $this->htmlEscape($ _product->getName()) ?></a></p>
            <?php echo $this->getPriceHtml($ _product, true) ?>
            <?php if($ _product->getRatingSummary()): ?>
              <?php echo $this->getReviewsSummaryHtml($ _product, 'short') ?>
            <?php endif; ?>
            <?php echo nl2br($ _product->getShortDescription()) ?>
            <a href="<?php echo $ _product->getProductUrl() ?>" title="<?php echo $this-
>htmlEscape($ _product->getName()) ?>"><small><?php echo $this->__('Learn More') ?></small></a>
          </td>
        <?php if($i%2==0 || $i==$ _collectionSize): ?>
      </tr>
    <?php endif; ?>
  <?php endforeach ?>
  <script type="text/javascript">decorateGeneric($$('tr'), ['last', 'odd', 'even']);</script>
</tbody>
</table>
</div>
<?php endif; //$_productCollection->count() ?>
```

Name it homelist.phtml and save it in location app/design/frontend/default/blue/template/catalog/product, create the folders where necessary

Edit the Home Page (CMS → Manage Pages → Home Page) and add the following content, where 19 should be replaced by your category ID:

```
<h3>Best Selling Products</h3>
{{block type="catalog/product_list" category_id="19" template="catalog/product/homelist.phtml}}
```

How to Fix the Breadcrumbs of Inactive Categories

If you follow this Wiki, you'll notice that the breadcrumbs in the Product Page include the inactive category in the middle (Home/Inactive/Product) and if you click on the category, you'll be served with a 404. Following steps show you how to hide the inactive category.

1. Rename your category by adding a prefix '0' (Admin → Catalog → Manage Categories → select the inactive category → Name), if your category name is Bestsellers, then rename it 0Bestsellers. This only works if names of other active categories do not start with '0'.
2. Edit breadcrumbs.phtml located in app/design/frontend/default/default/template/page/html by adding this `<?php if($_crumbInfo['label'][0]!='0'): ?>` to line 31 and this `<?php endif; ?>` at line 44. (The line numbers are based on version 1.2.0.2) The complete code:

```
<?php if($crumbs && is_array($crumbs)): ?>
<h4 class="no-display"><?php echo $this->__("You're currently on:") ?></h4>
<ul class="breadcrumbs">
  <?php foreach($crumbs as $_crumbName=>$_crumbInfo): ?>
    <?php if($_crumbInfo['label'][0]!='0'): ?>
      <li class="<?php echo $_crumbName ?>">
        <?php if($_crumbInfo['link']): ?>
          <a href="<?php echo $_crumbInfo['link'] ?>" title="<?php echo $_crumbInfo['title'] ?>"><?php echo
$_crumbInfo['label'] ?></a>
        <?php elseif($_crumbInfo['last']): ?>
          <strong><?php echo $_crumbInfo['label'] ?></strong>
        <?php else: ?>
          <?php echo $_crumbInfo['label'] ?>
        <?php endif; ?>
      </li>
    <?php if(!$_crumbInfo['last']): ?>
      <li> / </li>
    <?php endif; ?>
  <?php endif; ?>
<?php endforeach; ?>
</ul>
<?php endif; ?>
```

3. Save the file to the Blue Theme directory here: app/design/frontend/default/blue/template/page/html (If you are using other theme, then save it in the corresponding directory.)

Please note: In order to make this work for more than one block of products with different category IDs on the same page, you need to add the following code at the end of your phtml file(s):

```
<?php
//unset catalog to allow using template for multiple categories on single page
Mage::unregister("_singleton/catalog/layer");
?>
```

Alternative way:

If you are on 1.4 or greater or 1.7EE or greater then this post is no longer valid as magento has moved to using widgets. You can find a featured product widget here: <http://www.magewidgets.com/featured-products-widget.html>

If you want more in depth way, try to view this post: <http://inchoo.net/ecommerce/magento/featured-products-on-magento-frontpage/>

If you're using Magento v1.1.5 or later, you might want to use these code snippets instead: <http://www.magentocommerce.com/boards/viewthread/4780/P15/#t44262> This approach uses core functions instead of using Zend DB to figure out how to build a query statement.

Create Payment Method Module

Introduction

Each payment method can be done as separate module or few methods can be combined in same module if they share functionality or could be used together.

Let's create a module with one payment method that will:

- accept credit card information
- authorize it when order is submitted
- save transaction ID in order payment record

Our new module will be called **NewModule**.

Replace all instances of 'NewModule' with name of your module and 'newmodule' with simplified code, that contains only alphanumeric characters and underscore.

To make this tutorial most concise, it's implied that mentioned folders will be created when needed.

Make sure that app/code/local is in include_path.

If you are using configuration cache, don't forget to reset it after modifying config xml files by deleting the contents of var/cache/config/

A good piece of advice: Disable caching before developing a new module.

Module Declaration

Create app/etc/modules/CompanyName_NewModule.xml:

```
<config>

    <modules>

<!-- declare CompanyName_NewModule module -->

        <CompanyName_NewModule>

<!-- this is an active module -->

            <active>true</active>

<!-- this module will be located in app/code/local code pool -->

            <codePool>local</codePool>

<!-- specify dependencies for correct module loading order -->
```

```
<depends>

    <Mage_Payment />

</depends>

</CompanyName_NewModule>

</modules>

</config>
```

Now that the application is aware of the module, we will let Magento know about details of our module.

Module Configuration

Create `app/code/local/CompanyName/NewModule/etc/config.xml`:

```
<?xml version="1.0"?>

<config>

    <modules>

        <CompanyName_NewModule>

<!-- declare module's version information for database updates -->

            <version>0.1.0</version>

        </CompanyName_NewModule>

    </modules>

    <global>

<!-- IMPORTANT: if you use your own namespace (i.e. CompanyName) you also have to declare blocks group
for new module. See topic: http://www.magentocommerce.com/boards/viewthread/22416/#t102732 -->

        <blocks>

            <newmodule>

                <class>CompanyName_NewModule_Block</class>

            </newmodule>

        </blocks>

    </global>

</config>
```

```
</blocks>

<!-- declare model group for new module -->

    <models>

<!-- model group alias to be used in Mage::getModel('newmodule/...') -->

    <newmodule>

<!-- base class name for the model group -->

        <class>CompanyName_NewModule_Model</class>

    </newmodule>

</models>

<!-- declare resource setup for new module -->

    <resources>

<!-- resource identifier -->

        <newmodule_setup>

<!-- specify that this resource is a setup resource and used for upgrades -->

            <setup>

<!-- which module to look for install/upgrade files in -->

                <module>CompanyName_NewModule</module>

            </setup>

<!-- specify database connection for this resource -->

            <connection>

<!-- do not create new connection, use predefined core setup connection -->

                <use>core_setup</use>

            </connection>
```

```
</newmodule_setup>

<newmodule_write>

  <connection>

    <use>core_write</use>

  </connection>

</newmodule_write>

<newmodule_read>

  <connection>

    <use>core_read</use>

  </connection>

</newmodule_read>

</resources>

</global>

<!-- declare default configuration values for this module -->

<default>

<!-- 'payment' configuration section (tab) -->

  <payment>

<!-- 'newmodule' configuration group (fieldset) -->

    <newmodule>

<!-- by default this payment method is inactive -->

      <active>0</active>

<!-- model to handle logic for this payment method -->

      <model>newmodule/paymentMethod</model>

<!-- order status for new orders paid by this payment method -->
```



```

        <order_status>pending</order_status>

<!-- default title for payment checkout page and order view page -->

        <title>Credit Card (Authorize.net)</title>

        <cctypes>AE,VI,MC,DI</cctypes>

        <payment_action>authorize</payment_action>

        <allowspecific>0</allowspecific>

    </newmodule>

</payment>

</default>

</config>

```

Declare configuration options for admin panel

This file will define how you see configuration options in Magento admin panel System > Configuration

Create app/code/local/CompanyName/NewModule/etc/system.xml:

```

<?xml version="1.0"?>

<config>

    <sections>

<!-- payment tab -->

        <payment>

            <groups>

<!-- newmodule fieldset -->

                <newmodule translate="label" module="paygate">

<!-- will have title 'New Module' -->

                    <label>New Module</label>

<!-- position between other payment methods -->

```

```
<sort_order>670</sort_order>

<!-- do not show this configuration options in store scope -->

    <show_in_default>1</show_in_default>

    <show_in_website>1</show_in_website>

    <show_in_store>0</show_in_store>

    <fields>

<!-- is this payment method active for the website? -->

    <active translate="label">

<!-- label for the field -->

        <label>Enabled</label>

<!-- input type for configuration value -->

        <frontend_type>select</frontend_type>

<!-- model to take the option values from -->

        <source_model>adminhtml/system_config_source_yesno</source_model>

<!-- field position -->

        <sort_order>1</sort_order>

<!-- do not show this field in store scope -->

        <show_in_default>1</show_in_default>

        <show_in_website>1</show_in_website>

        <show_in_store>0</show_in_store>

    </active>

    <order_status translate="label">

        <label>New order status</label>

        <frontend_type>select</frontend_type>

        <source_model>adminhtml/system_config_source_order_status_processing</source_model>
```

```

        <sort_order>4</sort_order>

        <show_in_default>1</show_in_default>

        <show_in_website>1</show_in_website>

        <show_in_store>0</show_in_store>

    </order_status>

    <title translate="label">

        <label>Title</label>

        <frontend_type>text</frontend_type>

        <sort_order>2</sort_order>

        <show_in_default>1</show_in_default>

        <show_in_website>1</show_in_website>

        <show_in_store>0</show_in_store>

    </title>

</fields>

</newmodule>

</groups>

</payment>

</sections>

</config>

```

If you go now to Admin / System / Configuration / Payment Methods, you should see “New Module” group. Enable it and try to checkout. On payment methods page you should see “New Module” payment method with credit card form.

Database updates

Create app/code/local/CompanyName/NewModule/sql/newmodule_setup/mysql4-install-0.1.0.php:

```

<?php

// here are the table creation for this module e.g.:

```

```
$this->startSetup();

$this->run("HERE YOUR SQL");

$this->endSetup();
```

For database updates change module version in your config.xml:

```
<modules>
<CompanyName_NewModule>
<version>0.2.0</version>
</CompanyName_NewModule>
</modules>
```

And then create app/code/local/CompanyName/NewModule/sql/newmodule_setup/mysql4-upgrade-0.1.0-0.2.0.php:

```
<?php

// here are the table updates for this module e.g.:

$this->startSetup();

$this->run("HERE YOUR UPDATE SQL");

$this->endSetup();
```

Troubleshooting

- Dont put your module in /Mage. It belongs in app/code/community/ or app/code/local
- Make sure your module's first letter is capitalized. newmodule apparently will not work, it must start with a capital letter Newmodule.
- Also make sure that the recipient folder of the module's folder (CompanyName in the example) is capitalized as well. companyName doesn't seem to work, either.
- If your module is not showing in configuration>advanced then check your config.xml
- If your module shows in the list of modules (configuration>advanced) but not in the Payment Methods, your problem is probably in system.xml
- Make sure you clear the cache.

Create Shipping Method Module

Introduction

This tutorial is similar to Creation of Payment Method module, and differs the most in adapter model.

Each shipping method can be done as separate module or few methods can be combined in same module if they share functionality or could be used together.

Our new module will be called NewModule.

Replace all instances of 'NewModule' with name of your module and 'newmodule' with simplified code, that contains only alphanumeric characters and underscore.

Replace all instances of 'YourCompany' with your company name or whatever name you choose.

To make this tutorial most concise, it's implied that mentioned folders will be created when needed.

Make sure that app/code/local is in PHP's include_path. To do so, execute the following code, either in a shell or inside a web-accessible PHP file:

```
<?php echo get_include_path();?>
```

Make sure you put this code somewhere **after** Magento was loaded, at the bottom of index.php for example, because Magento might modify the include_path on its own to fit its needs and dynamically attribute locations.

If you are using configuration cache, don't forget to clean it after modifying config xml files.

Configuration

Create app/code/local/YourCompany/NewModule/etc/config.xml:

```
<?xml version="1.0"?>

<config>

  <modules>

<!-- declare module's version information -->

    <YourCompany_NewModule>

<!-- this version number will be used for database upgrades -->

        <version>0.1.0</version>

    </YourCompany_NewModule>
```

```
</modules>

<global>

<!-- declare model group for new module -->

  <models>

<!-- model group alias to be used in Mage::getModel() -->

  <newmodule>

<!-- base class name for the model group -->

    <class>YourCompany_NewModule_Model</class>

  </newmodule>

</models>

<!-- declare resource setup for new module -->

  <resources>

<!-- resource identifier -->

    <newmodule_setup>

<!-- specify that this resource is a setup resource and used for upgrades -->

      <setup>

<!-- which module to look for install/upgrade files in -->

        <module>YourCompany_NewModule</module>

      </setup>

<!-- specify database connection for this resource -->

      <connection>

<!-- do not create new connection, use predefined core setup connection -->

        <use>core_setup</use>

      </connection>

    </newmodule_setup>

  </resources>

</global>

</modules>
```

```
</connection>

</newmodule_setup>

</resources>

</global>

</config>
```

Edit app/etc/modules/YourCompany_NewModule.xml:

```
<config>

<!-- ... -->

<modules>

<!-- ... -->

<!-- declare YourCompany_NewModule module -->

  <YourCompany_NewModule>

    <active>true</active>

    <codePool>local</codePool>

  </YourCompany_NewModule>

<!-- ... -->

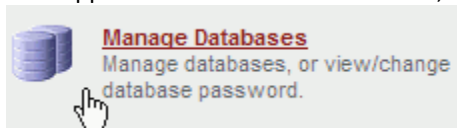
</modules>

<!-- ... -->

</config>
```

Note: using “<!-- ... -->” could imply there is more to input here, but for newbies like me, this is confusing if there is actually nothing to input. Also, I have seen some modules with <depends> <Mage_Shipping /> </depends> within the <YourCompany_NewModule> section. Should this be suggested?

Now application is aware of the module, but nothing will happen until we'll create model logic.



Adapter model

Note: ShippingMethod name is arbitrary and is up to your decision.

Create app/code/local/YourCompany/NewModule/Model/Carrier/ShippingMethod.php:

```
<?php

/**
 * Our test shipping method module adapter
 */

class YourCompany_NewModule_Model_Carrier_ShippingMethod extends
Mage_Shipping_Model_Carrier_Abstract

{

/**
 * unique internal shipping method identifier
 *
 * @var string [a-z0-9_]
 */

protected $_code = 'newmodule';

/**
 * Collect rates for this shipping method based on information in $request
 *
 * @param Mage_Shipping_Model_Rate_Request $data
 * @return Mage_Shipping_Model_Rate_Result
 */

public function collectRates(Mage_Shipping_Model_Rate_Request $request)

{
```



```

// skip if not enabled

if (!Mage::getStoreConfig('carriers/'.$this->_code.'/active')) {

    return false;

}

/**

* here we are retrieving shipping rates from external service

* or using internal logic to calculate the rate from $request

* you can see an example in Mage_Usa_Model_Shipping_Carrier_Ups::setRequest()

*/

// get necessary configuration values

$handling = Mage::getStoreConfig('carriers/'.$this->_code.'/handling');

// this object will be returned as result of this method

// containing all the shipping rates of this method

$result = Mage::getModel('shipping/rate_result');

// $response is an array that we have

foreach ($response as $rMethod) {

    // create new instance of method rate

    $method = Mage::getModel('shipping/rate_result_method');

    // record carrier information

    $method->setCarrier($this->_code);

```

```

$method->setCarrierTitle(Mage::getStoreConfig('carriers/'.$this->_code.'/title'));

// record method information

$method->setMethod($rMethod['code']);

$method->setMethodTitle($rMethod['title']);

// rate cost is optional property to record how much it costs to vendor to ship

$method->setCost($rMethod['amount']);

// in our example handling is fixed amount that is added to cost

// to receive price the customer will pay for shipping method.

// it could be as well percentage:

/// $method->setPrice($rMethod['amount']*$handling/100);

$method->setPrice($rMethod['amount']+$handling);

// add this rate to the result

$result->append($method);

}

return $result;

}

}

```

Now that we have the model let's give admin a way to configure it and also make checkout process aware of this method.

Admin Configuration Implementation

In this step, we need to tell Magento how to display our module in the Configuration section of the administrative panel. In order to do so, we must create `app/code/local/YourCompany/NewModule/etc/system.xml` and make it look something like this

```
<?xml version="1.0"?>

<config>

  <sections>

    <carriers>

      <groups>

        <newmodule translate="label" module="shipping">

          <label>Carrier Name</label>

          <frontend_type>text</frontend_type>

          <sort_order>13</sort_order>

          <show_in_default>1</show_in_default>

          <show_in_website>1</show_in_website>

          <show_in_store>1</show_in_store>

          <fields>

            <account translate="label">

              <label>Account number</label>

              <frontend_type>text</frontend_type>

              <sort_order>7</sort_order>

              <show_in_default>1</show_in_default>

              <show_in_website>1</show_in_website>

              <show_in_store>1</show_in_store>

            </account>

            <active translate="label">
```

```
<label>Enabled</label>

<frontend_type>select</frontend_type>

<source_model>adminhtml/system_config_source_yesno</source_model>

<sort_order>1</sort_order>

<show_in_default>1</show_in_default>

<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>
```

```
</active>
```

```
<contentdesc translate="label">
```

```
<label>Package Description</label>

<frontend_type>text</frontend_type>

<sort_order>12</sort_order>

<show_in_default>1</show_in_default>

<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>
```

```
</contentdesc>
```

```
<!--
```

shipping If the free_shipping_enable flag enable, the system will check free_shipping_subtotal to give free

otherwise will use shopping cart price rule behaviour

```
-->
```

```
<free_shipping_enable translate="label">
```

```
<label>Free shipping with minimum order amount</label>

<frontend_type>select</frontend_type>

<source_model>adminhtml/system_config_source_enabledisable</source_model>
```

```
<sort_order>21</sort_order>

<show_in_default>1</show_in_default>

<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>

</free_shipping_enable>

<free_shipping_subtotal translate="label">

  <label>Minimum order amount for free shipping</label>

  <frontend_type>text</frontend_type>

  <sort_order>22</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>

</free_shipping_subtotal>

<dutiable translate="label">

  <label>Shipment Dutiable</label>

  <frontend_type>select</frontend_type>

  <source_model>adminhtml/system_config_source_yesno</source_model>

  <sort_order>13</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>

</dutiable>

<gateway_url translate="label">

  <label>Gateway URL</label>

  <frontend_type>text</frontend_type>
```

```
<sort_order>2</sort_order>

<show_in_default>1</show_in_default>

<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>

</gateway_url>

<handling_type translate="label">

  <label>Calculate Handling Fee</label>

  <frontend_type>select</frontend_type>

  <source_model>shipping/source_handlingType</source_model>

  <sort_order>10</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>0</show_in_store>

</handling_type>

<handling_action translate="label">

  <label>Handling Applied</label>

  <frontend_type>select</frontend_type>

  <source_model>shipping/source_handlingAction</source_model>

  <sort_order>11</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>0</show_in_store>

</handling_action>

<handling_fee translate="label">

  <label>Handling fee</label>
```

```
<frontend_type>text</frontend_type>

<sort_order>12</sort_order>

<show_in_default>1</show_in_default>

<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>

</handling_fee>

<max_package_weight translate="label">

    <label>Maximum Package Weight (Please consult your shipping carrier for maximum
supported shipping weight)</label>

    <frontend_type>text</frontend_type>

    <sort_order>13</sort_order>

    <show_in_default>1</show_in_default>

    <show_in_website>1</show_in_website>

    <show_in_store>1</show_in_store>

</max_package_weight>

<id translate="label">

    <label>Access ID</label>

    <frontend_type>text</frontend_type>

    <backend_model>adminhtml/system_config_backend_encrypted</backend_model>

    <sort_order>5</sort_order>

    <show_in_default>1</show_in_default>

    <show_in_website>1</show_in_website>

    <show_in_store>1</show_in_store>

</id>

<password translate="label">
```

```
<label>Password</label>

<frontend_type>text</frontend_type>

<backend_model>adminhtml/system_config_backend_encrypted</backend_model>

<sort_order>6</sort_order>

<show_in_default>1</show_in_default>

<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>

</password>

<shipping_intlkey translate="label">

  <label>Shipping key (International)</label>

  <frontend_type>text</frontend_type>

  <backend_model>adminhtml/system_config_backend_encrypted</backend_model>

  <sort_order>8</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>

</shipping_intlkey>

<shipping_key translate="label">

  <label>Shipping key</label>

  <frontend_type>text</frontend_type>

  <backend_model>adminhtml/system_config_backend_encrypted</backend_model>

  <sort_order>8</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>
```



```
</shipping_key>

<sort_order translate="label">

  <label>Sort order</label>

  <frontend_type>text</frontend_type>

  <sort_order>100</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>

</sort_order>

<title translate="label">

  <label>Title</label>

  <frontend_type>text</frontend_type>

  <sort_order>2</sort_order>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>

</title>

<sallowspecific translate="label">

  <label>Ship to applicable countries</label>

  <frontend_type>select</frontend_type>

  <sort_order>90</sort_order>

  <frontend_class>shipping-applicable-country</frontend_class>

<source_model>adminhtml/system_config_source_shipping_allspecificcountries</source_model>

  <show_in_default>1</show_in_default>
```

```
<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>

</sallowspecific>

<specificcountry translate="label">

  <label>Ship to Specific countries</label>

  <frontend_type>multiselect</frontend_type>

  <sort_order>91</sort_order>

  <source_model>adminhtml/system_config_source_country</source_model>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>

</specificcountry>

<showmethod translate="label">

  <label>Show method if not applicable</label>

  <frontend_type>select</frontend_type>

  <sort_order>92</sort_order>

  <source_model>adminhtml/system_config_source_yesno</source_model>

  <show_in_default>1</show_in_default>

  <show_in_website>1</show_in_website>

  <show_in_store>1</show_in_store>

</showmethod>

<specificerrmsg translate="label">

  <label>Displayed Error Message</label>

  <frontend_type>textarea</frontend_type>

  <sort_order>80</sort_order>
```

```
<show_in_default>1</show_in_default>

<show_in_website>1</show_in_website>

<show_in_store>1</show_in_store>

</specificerrmsg>

</fields>

</newmodule>

</groups>

</carriers>

</sections>

</config>
```

You should now see your module in the Administration under “System” > “Configuration” > “Shipping Methods”. It’s now up to you to add your custom fields in the <fields> tag, and subsequently make your configuration do something constructive.

Common Problems

Here is a list of things that have happened to others while trying to implement thier own Shipping Module. (Please add to this list or provide responses to unresolved issues if you can. A collection of common problems and how to avoid them can only help everybody.)

Module won’t activate

Relating to the importance of casing, be sure the CompanyName you use has an initial cap letter - a lowercase initial letter will prevent this from activating. Magento really needs to have clear documentation for these norms.

Module doesn't appear in frontend.

I’ve just managed to get this to work, with a bit of hacking. I had created a method following the instructions above, but couldn’t get it it give me a quote on the front end. The problem was in app/core/Mage/Shipping/Model/Shipping.php, line 164:

```
$className = Mage::getStoreConfig('carriers/'. $carrierCode. '/model', $storeId);
```

The module didn’t have a ‘model’ defined for it, so the getCarrierByCode() method was returning false straight away. The hack involved creating a new field in the system.xml file:

```
<model translate="label">
```

```
<label>Model</label>
<frontend_type>text</frontend_type>
<sort_order>900</sort_order>
<show_in_default>1</show_in_default>
<show_in_website>1</show_in_website>
</model>
```

Then, using the admin panel, I gave this a value of 'newmodule/carrier_newmodule'. While I was there I also created a 'name' field to give the method a name.

After doing that, the method was appearing in my list of quotes! — *David Edwards 2008/07/28 10:26*

I found another solution: insert the following code into config.xml right below the config -tag:

```
<default>
  <carriers>
    <mage_newmodule>
      <active>1</active>
      <sallowspecific>0</sallowspecific>
<model>newmodule/carrier_newmodule</model>
      <name>New Module</name>
      <title>New Module</title>
      <specificerrmsg>
        This shipping method is currently unavailable.
        If you would like to ship using this shipping
        method, please contact us.
      </specificerrmsg>
      <handling_type>F</handling_type>
    </mage_newmodule>
  </carriers>
</default>
```

Module doesn't appear in admin.

Check to make sure that you have the system.xml and config.xml files in the suggested directory structure. Capitalization appears to be important. Be careful not to use "locale" where you meant to use "local".

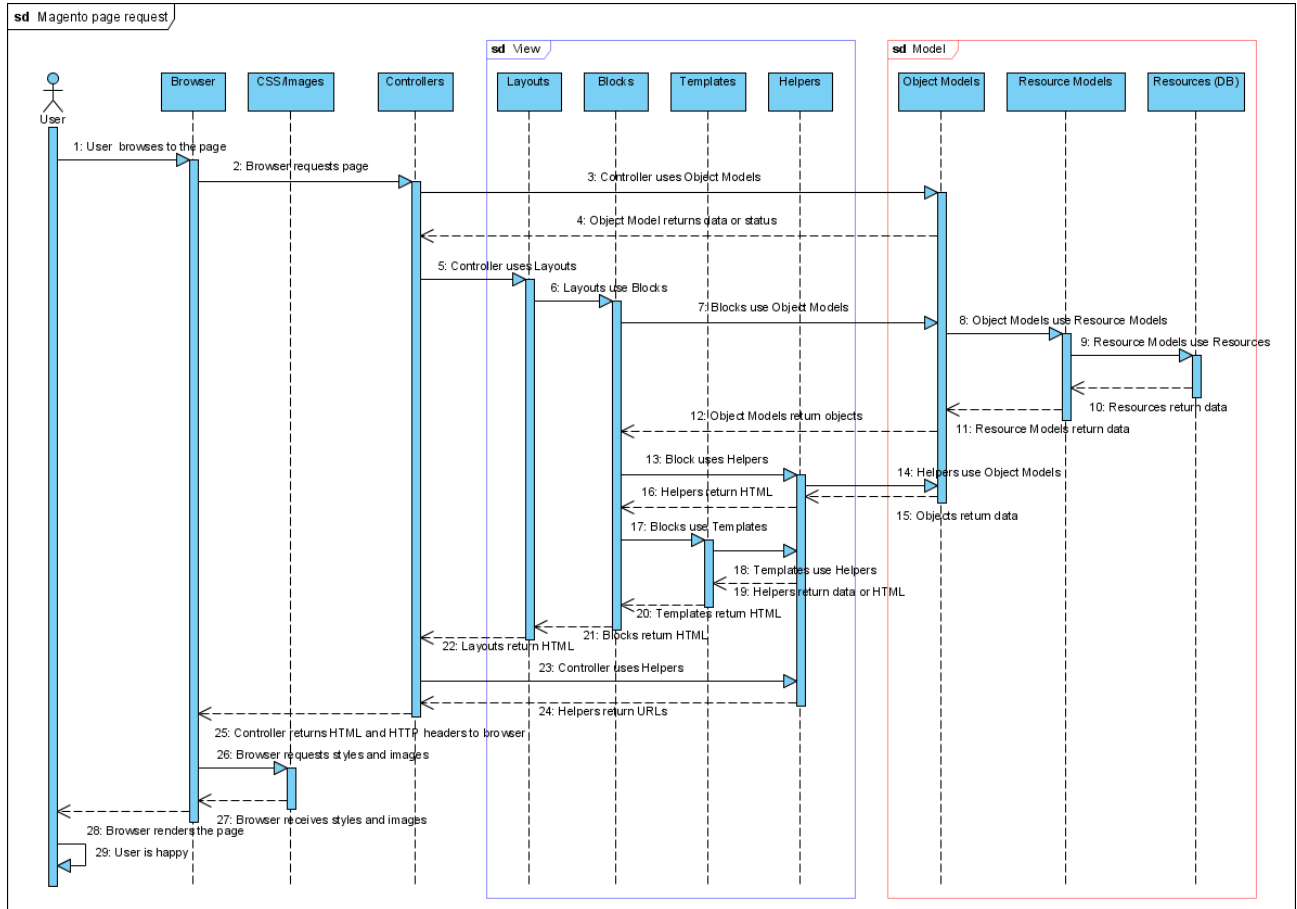
Backend shows error message when displaying the shipping method

Make sure that the content in every <source_model> - tag has no line-break.

Correct:

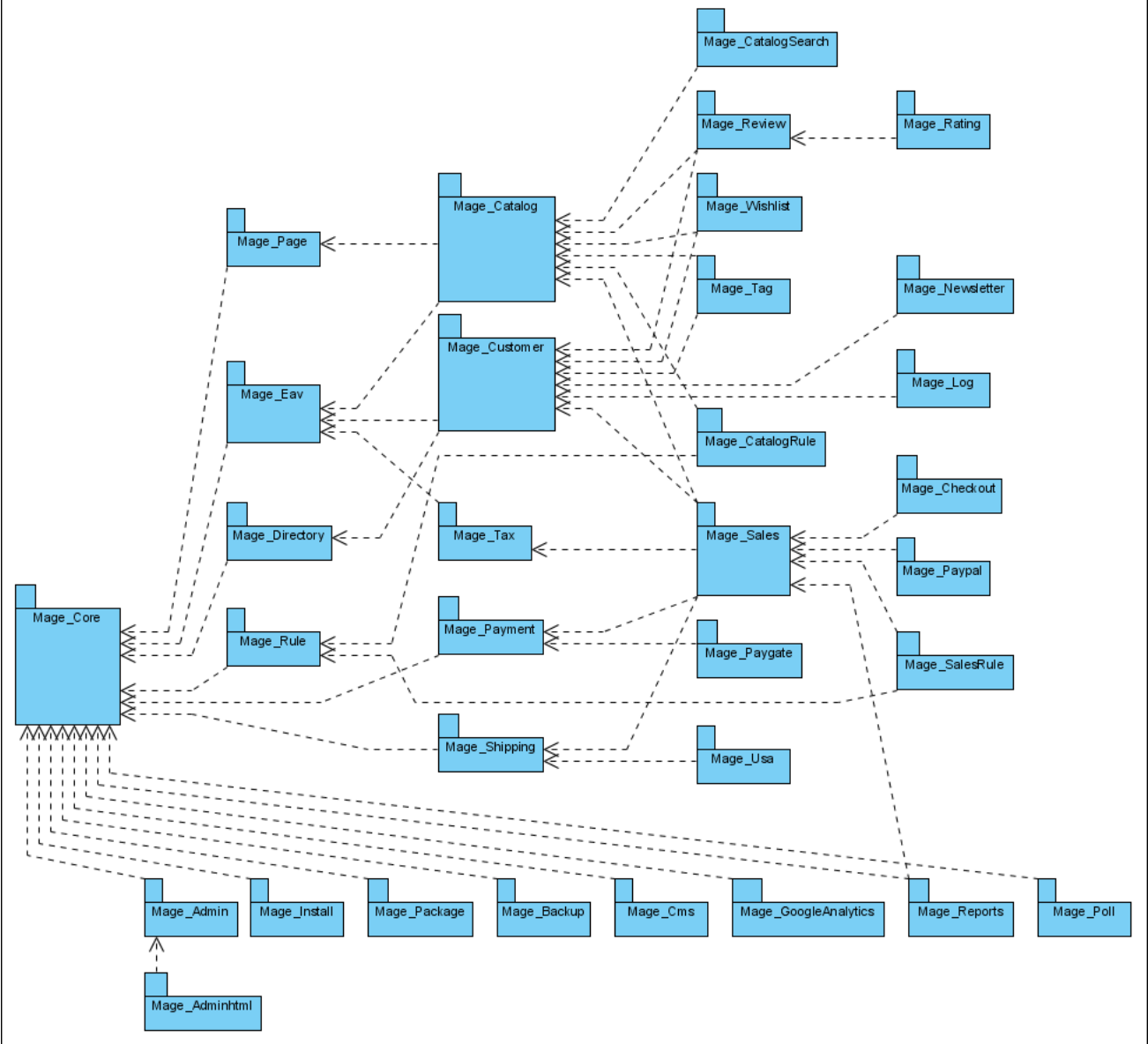
Magento Architecture

Page request flow



Preliminary core modules dependency diagram

All Packages



Changing and Customizing Magento Code

If you're finding that you need to make changes to Magento's code to fit it into your organization, you're probably wondering what is the best way to make these changes so that you can keep your code separate from the main code.

Skill level: Beginning to Advanced developer.

Target Audience: Developers who need to customize PHP Code.

Tested with Magento versions:

- 0.6.14100 (BETA)
- 0.7.14800 (BETA)
- 1.1.6

Applicable for all Magento versions (1.1.x and older).

Subversion

The best way to keep track of all your changes is with a tool like Subversion (SVN) or CVS. You can use branches for the base Magento code. This will keep your customized files from being blatantly overwritten whenever you update Magento.

Here is an example code layout

```
my_project/  
  trunk/  
    scripts/  
    src/ (magento's code)  
    data/  
  branches/  
    vendor/  
      magento/  
        app/  
        lib/  
        index.php
```

Under the "magento" directory, we have an **export** of Magento's latest SVN (0.6 beta). Now, we can use SVN's merge to copy the changes that happen to that one directory (branches/vendor/magento) into our main directory (trunk/src/). The first time, the changes will only be between revisions 1 and 2 (if don't make any mistakes setting up the repository). But, every magento release, you will only have 1 revision change, for example: after making 100 changes to your store you are at revision 101. You update the Magento vendor branch with version 0.8 beta. Now you are at revision 102. And the difference between 101 and 102 contain all the Magento changes between 0.6.14400 and 0.8.2620 (imaginary magento numbers). So, when you merge 101:102 into your main/src directory, Subversion will do its best to only apply the changes, and not simply overwrite all your hard work with new files from 0.8.2620.

Upgrading to a newer version

Here is the proper way to upgrade your own SVN repository if you're using vendor branches as describe above.

Copy files over other files is not going to get you a clean upgrade. Subversion (and CVS) track updates, new files, and **deletes**. If you simply download and extract a new version, you're going to miss key file deletes and config file deletes.

The vendor branch

Dealing with vendor branches is very messy in SVN. The way you expect it to work doesn't, so they provide you with an external Perl script to handle the messy parts of vendor branches. The program is called "svn_load_dirs.pl", and, under Fedora, is located at **/usr/share/doc/subversion-1.4.3/svn_load_dirs.pl**

Our steps for upgrading the vendor branch go like this:

- export a new copy of Magento 0.7
- use `svn_load_dirs` to properly include the changes into our vendor branch.

In a clean directory:

```
svn export http://svn.magentocommerce.com/source/branches/beta-0.7-latest

/usr/share/doc/subversion-1.4.3/svn_load_dirs.pl
http://127.0.0.1/repos/my_project/branches/vendor
magento
beta-0.7-latest/
```

You should see some output like:

```
Checking that the base URL is a Subversion repository.
Running /usr/bin/svn log -r HEAD http://127.0.0.1/repos/my_project/branches/vendor

Finding the root URL of the Subversion repository.
Running /usr/bin/svn log -r HEAD http://127.0.0.1
Running /usr/bin/svn log -r HEAD http://127.0.0.1/repos
Running /usr/bin/svn log -r HEAD http://127.0.0.1/repos/my_project
Determined that the svn root URL is http://127.0.0.1/repos/my_project.

Native EOL on this system is ⚡12.

Finding if any directories need to be created in repository.
....

The following table lists files and directories that
exist in either the Subversion repository or the
directory to be imported but not both. You now have
the opportunity to match them up as renames instead
```


of deletes and adds. This is a Good Thing as it'll make the repository take less space.

The left column lists files and directories that exist in the Subversion repository and do not exist in the directory being imported. The right column lists files and directories that exist in the directory being imported. Match up a deleted item from the left column with an added item from the right column. Note the line numbers on the left which you type into this script to have a rename performed.

You will now see two columns of about 200 changes. On one side are files that are in the repository, but not in the new beta-0.7-latest directory, on the right files that exist in the new directory, but not in the repository. Subversion does not know which of these files have been added, deleted, or renamed. This is why vendor branches are messy. This Perl script wants you to scan through all 200 changes and match up files that actually got renamed or moved by typing in a number of a file in the right column and it's match in the left, i.e. 214 10 if file 214 in the left column was actually just moved to a new directory and it shows up as row 10 in the right column. This task is summarily daunting and not worth the manual effort since we are only getting access to new Magento code about once every 800 changes.

"What does this mean to me?", I hear you asking. Well, let's say you've modified a file called "Google_Checkout.php". In the Magento SVN repository, the Varien developers decide to rename the file, "GoogleCheckout.php". This seemingly simple change can have dire consequences for you down the road. Since we cannot directly merge from Magento's SVN repository, but we must use the intermediate files as a driver for "svn_load_dirs.pl", this renaming action is lost. We simply have a new file "GoogleCheckout.php" and no more "Google_Checkout.php". svn_load_dirs.pl, unless manually instructed, will delete the old file Google_Checkout and add the new one. When you go to merge the branch into your main development trunk, this add/delete action will be performed again, removing all your changes to the old named file "Google_Checkout". "WHAT?!", you ask, "Why do I even bother then with the vendor branches?". The answer is, because it's easier than not dealing with them.

Imagine if you simply copied over the new release and the newer "GoogleCheckout.php" file was being used instead of your changes. There would be no error, there would be no message from SVN deleting your code... your changes would silently be dropped simply because the file they were in would not be loaded anymore. At least with this process you will get alerted to your changes being removed, and you can manually pluck them out of an old version and re-instate your changes. Without this method, you will be left scratching your head wondering why so much stuff is behaving strangely after an upgrade.

Merging the vendor

Now you're ready to see if all this extra work is worth it. It's time to merge the vendor branch into your working changes. There is no need to have a physical disk checkout of your branch so you can go ahead and delete it.

```
cd /path/to/my_project/src/  
svn merge --dry-run -r 71:72 http://127.0.0.1/repos/my_project/branches/vendor/magento
```

Assuming the auger goes for you, remove the `-dry-run` flag. More than likely, the auger will not go all well and you will see some SVN **collisions**. Even if you haven't modified any of the code, the changes from one version of Magento to another may be too great for subversion to handle in one diff.

Now you have the option to study any of the collisions and upgrade at your leisure. This is far safer than simply overwriting a package full of files over your hard work.

To view all the collided files:

```
svn diff | grep ^C
```

Custom Modules

Most likely you will want to make a module that represent's your company to hold all your specific changes. Start off by making a new directory like so:

```
app/  
  code/  
    core/  
    community/  
    local/  
      XyzCorp/
```

Now, if you need to make changes to the Magento file, `Mage/Catalog/Block/Breadcrumbs.php`, you can add a new directory for the "Catalog" module under your `XyzCorp` directory, then a `blocks` directory and copy the file into this new directory. Also you need to create `config.xml` of your module.

```
app/  
  code/  
    core/  
    community/  
    local/  
      XyzCorp/  
        Catalog/  
          Block/  
            Breadcrumbs.php  
          etc/  
            config.xml
```

Change the class name inside the file by starting off with "XyzCorp" instead of "Mage".

Strip out all the code you don't need, and make it subclass the original class name ("Mage_Catalog_Block_Breadcrumbs").

Now, you must **activate** your module so Magento understands that there is new code in the "local" directory.

In `app/etc/modules/XyzCorp_Catalog.xml`, add the following lines

```
<?xml version="1.0"?>  
<config>
```

```
<modules>
  <XyzCorp_Catalog>
    <active>true</active>
    <codePool>local</codePool>
  </XyzCorp_Catalog>
</modules>
</config>
```

It is crucial that the same prefix XyzCorp is used throughout files, class names, directories, and XML tag names.

Now, your own catalog module is activated, but when will it actually be called by the system? Ahh, we need a special rewrite tag to instruct Magento to use this one file (Breadcrumbs.php) instead of its default.

Now we should rewrite block using your module's config file.

Blocks

Edit app/code/local/XyzCorp/Catalog/etc/config.xml

```
<?xml version="1.0"?>
<config>
  <modules>
    <XyzCorp_Catalog>
      <version>0.1.0</version>
    </XyzCorp_Catalog>
  </modules>
  <global>
    <blocks>
      <catalog>
        <rewrite>
          <breadcrumbs>XyzCorp_Catalog_Block_Breadcrumbs</breadcrumbs>
        </rewrite>
      </catalog>
    </blocks>
  </global>
</config>
```

We need to add a "blocks" tag, or edit inside an existing blocks tag, depending on your XML file. Then we add a **rewrite** tag **after** our module name, which is "catalog" in this case. Then, we throw in the word "breadcrumbs". This "breadcrumbs" name must help magento to find the Block class you want to extend. In our example here, **breadcrumbs** is the core class file name (which will be overwritten):

app/code/core/Mage/Catalog/Block/Breadcrumbs.php. If you have more levels below the **Block** directory, include it on that tag, using underscores to separate it from the class file name. Ex:

```
<blocks>
  <catalog>
    <rewrite>
      <category_view>XyzCorp_Catalog_Block_Category_View</category_view>
    </rewrite>
  </catalog>
```

</blocks>

In this case, the class being overwritten is `app/code/core/Mage/Catalog/Block/Category/View.php`.

The data inside the **breadcrumbs** (same for **category_view**) tag is the name of your classfile, and Magento knows how to find it because the class name is the same as its directory path and filename. Remember that the underscore means another folder level on the file structure, and Magento won't find your class in case the Folder structure doesn't reflect the classname properly.

For instance:

`XYZCorp_Catalog_Block_Breadcrumbs` → `/app/code/local/XYZCorp/Catalog/Block/Breadcrumbs.php`

`XYZCorp_Catalog_Block_Category_View` → `/app/code/local/XYZCorp/Catalog/Block/Category/View.php`

How to Create a Featured Product

THIS NO LONGER APPLIES IN MAGENTO 1.4.x CE and EE 1.7 This logic is now implemented with widgets. See the knowledgebase for more information.

This tutorial will show you how to implement a Featured Product feature. The Featured Product is a product with an attribute added from the administrative UI. When the administrator selects "Yes" in the "Featured" attribute, that product will be displayed in a content block on the category page.

I'll explain each step I took to make this custom feature. Please forgive me if I left anything out.

Note: For me the featured product only showed up if the category **was not an anchor**.

Step 1) Create new "Featured" attribute

Create a new attribute by going to Catalog > Attributes > Manage Attributes > Add New Attribute.

Attribute Properties

- Attribute Identifier: **featured**
- Scope: **Store View**
- Catalog Input Type for Store Owner: **Yes/No**
- Unique Value (not shared with other products): **No**
- Values Required: **No**
- Input Validation for Store Owner: **None**
- Apply To: **All Product Types**

Front End Properties

- Use in quick search: **No**
- Use in advanced search: **Yes**
- Comparable on Front-end: **No**
- Use In Layered Navigation (Can be used only with catalog input type 'Dropdown'): **No**
- Visible on Catalog Pages on Front-end: **Yes**

Manage Label/Options

- Default: **Featured Product**
- English: **Featured Product**

Save the new attribute and go to Catalog → Attributes → Manage Attributes Sets to add the attribute to the default feature set.

Step 2) Add Block configuration to catalog.xml

Open MyCompany/app/design/frontend/default/default/layout/catalog.xml. We want to add a new <block> right above the product list block in the default category layout.

Insert the block configuration on line 73 (default catalog.xml).

```
<block type="catalog/product_featured" name="product_featured" as="product_featured"
template="catalog/product/featured.phtml"></block>
```

Step 3) Create new block class that will instantiate the featured product

Create a new file, and directories: app/code/local/MyCompany/Catalog/Block/Product/Featured.php

```
<?php

class MyCompany_Catalog_Block_Product_Featured extends Mage_Catalog_Block_Product_Abstract
{
    public function getFeaturedProduct()
    {
        // instantiate database connection object

        $storeId = Mage::app()->getStore()->getId();

        $categoryId = $this->getRequest()->getParam('id', false);

        $resource = Mage::getSingleton('core/resource');

        $read = $resource->getConnection('catalog_read');

        $categoryProductTable = $resource->getTableName('catalog/category_product');

        //$productEntityIntTable = $resource->getTableName('catalog/product_entity_int'); // doesn't work :(
        $productEntityIntTable = (string)Mage::getConfig()->getTablePrefix() . 'catalog_product_entity_int';

        $eavAttributeTable = $resource->getTableName('eav/attribute');

        // Query database for featured product

        $select = $read->select()

            ->from(array('cp'=>$categoryProductTable))

            ->join(array('pei'=>$productEntityIntTable), 'pei.entity_id=cp.product_id', array())
```

```

->joinNatural(array('ea'=>$eavAttributeTable))

->where('cp.category_id=?', $categoryId)

->where('pei.value=1')

->where('ea.attribute_code="featured"');

$row = $read->fetchRow($select);

return Mage::getModel('catalog/product')->setStoreId($storeId)->load($row['product_id']);
}
}
?>

```

We're almost there!

Step 4) Extend Mage_Catalog_Block_Category_View

Create a new file, and directories, called app/code/local/MyCompany/Catalog/Block/Category/View.php. We're extending the core class here so our module will be separate from the core code base. When upgrading, we won't have to worry about our code not working or having to patch files.

```

<?php

class MyCompany_Catalog_Block_Category_View extends Mage_Catalog_Block_Category_View
{

    public function getFeaturedProductHtml()

    {

        return $this->getBlockHtml('product_featured');

    }

}

?>

```

Step 5) Modify the templates

Edit `app/design/frontend/default/default/template/catalog/category/view.phtml` and add the following code:

```
<?=$this->getFeaturedProductHtml()?>
```

right above this line:

```
<?=$this->getProductListHtml()?>
```

Create `app/design/frontend/default/default/template/catalog/product/featured.phtml` and add some product info HTML to show the featured product. Here is an example that simply displays a link to the product:

```
<?php $_product=$this->getFeaturedProduct() ?>
```

```
Check this out: <a href="<?php echo $_product->getProductUrl() ?>"><?php echo $this->htmlEscape($_product->getName()) ?></a>
```

Step 6) Add new blocks to the app/etc/local.xml

Add the following inside the config global tag:

```
<blocks>
  <catalog>
    <rewrite>
      <product_featured>MyCompany_Catalog_Block_Product_Featured</product_featured>
    </rewrite>
    <rewrite>
      <category_view>MyCompany_Catalog_Block_Category_View</category_view>
    </rewrite>
  </catalog>
</blocks>
```

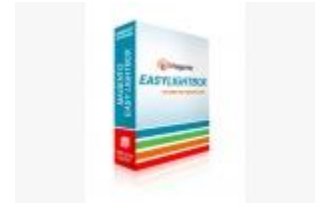

Magento Easy Lightbox

<http://www.magentocommerce.com/extension/1487/magento-easy-lightbox>

Overview

Magento Easy Lightbox Community Module

This small extension will help to install Lightbox widget. Installation and configuration will take approximately 5 minutes. No files are replaced and no coding experience needed to install!



[Magento EasyLightbox Extension Support](#)

[Magento EasyLightbox Extension Demo](#)

1. Install extension using magento connect.
2. Log out from admin and login again.
3. Navigate to System > Configuration > Templates-Master > EasyLightbox and enable extension for any store you need.
4. Enjoy!

CSS Resources

Changing your CSS files on the fly

Mozilla does a great CSS Developer plug-in that allows you to change the CSS files on the Fly

Plugin

<https://addons.mozilla.org/en-US/firefox/addon/60> great CSS Developer plug

Mozilla

<http://www.mozilla.com/en-US/> Mozilla Download if you dont have it

A great CSS book here called THE CSS ANTHOLOGY 101 Essential Tips Tricks and Hacks

<http://www.euphorish.com/2007/07/02/html-css-article-overview-lay-out/>

How to use Web Developer CSS

open your Mozilla Browser (forget winddoze browsers) locate

TOOLS » Web Developer » css » edit css

Now at the bottom of the browser you should see a **CSS panel**

This will list all the CSS files called into that particular page of you store

select which files contents you need to change IE **boxes.css menu.css etc**

cut all the css out of boxes.css and watch what happens the page in the browser will render with no Style

refresh the browser and it all comes back

The following is a list of CSS resources for those wanting to polish their skills:

http://www.westciv.com/style_master/academy/css_tutorial/ (the number one guide to CSS)

<http://www.w3.org/Style/CSS> (and the boring docs, but nevertheless... some good detailed info here)

<http://www.nypl.org/styleguide/> (XHTML and CSS guide)

<http://www.htmlhelp.com/reference/css/> (another CSS guide)

<http://www.sitepoint.com/article/css-5-building-skeleton/2> (HTML Utopia: designing without tables [free chapter])

http://en.wikipedia.org/wiki/Cascading_Style_Sheets (It's Wikipedia man!)

<http://www.dezwozhere.com/links.html> (tons more resources)

<http://www.mezzoblue.com/zengarden/resources/> (even more resources [may be some duplicates])

<http://www.positioniseverything.net/>

Creating CSS buttons vs Image buttons

There have been requests from many community members about the flexibility of the image buttons used in the Magento frontend default theme. This article serves to inform and inspire you to take advantage of what a little tweak in the HTML and CSS can do in swapping image buttons to CSS-powered ones. To those dedicated CSS ladies and gents - **here's to you.**

Let's start by downloading some images

The following gifs are also available in [PSD format](#) for your toying pleasure - save it as png, gif, jpg, whatever suits your fancy. For demonstration purposes, we're using the GIF format.

Download the following images, or save the above PSD in your preferred file format, and save it in your magento directory **skin/frontend/default/default/images**



Now for the Magic.

The HTML

Let's use the "Proceed to Checkout" image button in the shopping cart for the occasion (shopping cart phtml can be found in app/design/frontend/default/default/template/checkout/onepage/link.phtml in the Magento file structure)

HTML for Image button - comes with the default Magento package

```
<a href="<?=$this->getCheckoutUrl()?>">"/>
</a>
```

HTML for CSS button - change the style of the button with just a swap of the background

```
<a class="img-btn btn-checkout" href="<?php echo $this->getCheckoutUrl()?>">
<span><?php echo Mage::helper('checkout')->__('Proceed to Checkout');?>
</span>
</a>
```

Swap out the 'HTML for Image button' code with the 'HTML for CSS button' code. Note: The 'img-btn' class in the HTML just serves to unite all the CSS buttons in case you want to synchronize the font colors and sizes..etc.

The CSS

```
.btn-checkout {  
  
    display:block;  
  
    float:right;  
  
    background:transparent url(..images/btn_proceed_to_checkout_rad.gif) no-repeat 100% 0;  
  
    font-size:15px;  
  
    font-weight:bold;  
  
    padding-right:8px;  
  
    }  
  
.btn-checkout, .btn-checkout:hover {  
  
    color:#fef5e5;  
  
    text-decoration:none;  
  
    }  
  
.btn-checkout span {  
  
    display:block;  
  
    padding:0 17px 0 25px;  
  
    background:transparent url(..images/btn_proceed_to_checkout_bg.gif) no-repeat;  
  
    line-height:40px;  
  
    }
```

And there you have it! - An all CSS-powered button.

Sorry, but with v.1.7 this didn't work. I changed **link.phtml** code to this:

```
<a class="img-btn btn-checkout" href="<?php echo $this->getCheckoutUrl()?>">
<span><?php echo $this->__('Proceed to Checkout')?>
</span>
</a>
```

And then it worked well... :)

Apply on the Place Order button

copy the file `app/design/frontend/default/default/template/checkout/onepage/review.phtml` in your design path, and replace :

```
<input type="image" src="<?php echo $this->getSkinUrl('images/btn_place_order.gif') ?>"
onclick="review.save();" value="<?php echo $this->__('Place Order') ?>" />
```

to :

```
<a class="img-btn btn-checkout" href="#" onclick="review.save();"><span><?php echo $this->__('Place Order')
;?></span></a>
```

The “Place Order” button is now css powered and can be translated easily!

Add Home Link with functional active state to Menu Bar (Alternative Method)

UPDATED: Included the following alternative method to add a home link with functional active state to the menu bar.

Add Home Link with functional active state to Menu Bar (Alternative Method)

Find the file called *top.phtml* in *app/design/frontend/default/yourtheme/template/catalog/navigation/* and make the following change:

```
<div class="header-nav-container">

    <div class="header-nav">

        <h4 class="no-display"><?php echo $this->__('Category Navigation:') ?></h4>

        <ul id="nav">

            <!-- ALTERNATIVE HOME BUTTON HACK -->

            <li class="home"><a href="<?php echo $this->getUrl("")?>"><?php echo $this->__('Home') ?></a></li>

            <!-- ALTERNATIVE HOME BUTTON HACK -->

            <?php foreach ($this->getStoreCategories() as $_category):?>

                <?php echo $this->drawItem($_category) ?>

            <?php endforeach ?>

        </ul>

    </div>

    <?php echo $this->getChildHtml('topLeftLinks') ?>

</div>
```

Add the following to the *menu.css* file in *skin/frontend/default/yourtheme/css/*. This example is for a CMS home page which uses *cms-home* class in its body tag.

```
body.cms-home #nav li.home a { color:#d96708; }
```

NOTE: The following comments refer to the following methods.

:::: TRIED THIS AND IT ADDS THE LINK, BUT THE ACTIVE STATE DOES NOT FUNCTION CORRECTLY ::::

Special Note to the person(s) responsible for this page. Myself and many others have attempted to use this code snippet to get the result explained with an active state, but have been unable to get the active state to work. The link does work as far as pointing to existing cms page or external pages, but the “active” state does not work at all. Page refreshes and all links adjust back to original state.

FOR HOME LINK ONLY ACTIVE STATE I use the following as it’s the simplist way to get the active state working correctly on the home link.

```
<div class="header-nav-container">

    <div class="header-nav">

        <h4 class="no-display"><?php echo $this->__('Category Navigation:') ?></h4>

        <ul id="nav">

            <!-- WORKING ACTIVE STATE HOME BUTTON HACK -->

            <li class="home<?php if (Mage::helper('core/url')->getCurrentUrl() === Mage::helper('core/url')->getHomeUrl()):?> active<?php endif;?>"><a href="<?php echo $this->getUrl("")?>"><?php echo $this->__('Home') ?></a></li>

            <!-- WORKING ACTIVE STATE HOME BUTTON HACK -->

            <?php foreach ($this->getStoreCategories() as $_category):?>

                <?php echo $this->drawItem($_category) ?>

            <?php endforeach ?>

        </ul>

    </div>

    <?php echo $this->getChildHtml('topLeftLinks') ?>

</div>
```

Add Home Link to Menu Bar

For the *Home* link in the menu bar of the main template you can add some code to one of the template files.

Find the file called *top.phtml* in *app/design/frontend/default/default/template/catalog/navigation/* and make the following change:

```
<div class="header-nav-container">

    <div class="header-nav">

        <h4 class="no-display"><?php echo $this->__('Category Navigation:') ?></h4>

        <ul id="nav">

            <!-- HOME BUTTON HACK -->

            <?php $_anyActive = false; foreach ($this->getStoreCategories() as $_category) { $_anyActive =
            $_anyActive || $this->isCategoryActive($_category); } ?>

            <li class="<?php echo !$ _anyActive ? 'active' : " ?>"><a href="<?php echo $this->getUrl(">"><?php echo
            $this->__('Home') ?></a></li>

            <!-- HOME BUTTON HACK -->

            <?php foreach ($this->getStoreCategories(10) as $_category):?>

                <?php echo $this->drawItem($_category) ?>

            <?php endforeach ?>

        </ul>

    </div>

    <?php echo $this->getChildHtml('topLeftLinks') ?>

</div>
```

x:x:x: Tried several on this page. The one directly above worked best. I modified it a little bit to contain a link title. Just paste

```
<li class="home<?php if (Mage::helper('core/url')->getCurrentUrl() === Mage::helper('core/url')->getHomeUrl()):?>
active<?php endif;?>"><a href="<?php echo $this->getUrl(">" title="Home">Home</a></li>
```

just before

```
<?php echo $_menu; ?>
```

if you're using the default theme or a top.phtml that is based on it. :x:x:x:

Add Home Link to Top Links

The correct way to do this is to open the theme/layout/customer.xml file and then modify the section that shows customer links on all pages, to include a link home and also a link to other customer service pages that you have deemed necessary, e.g. 'returns' (if you get a lot of those enquiries...).

By way of example, this modified XML file includes a 'Home' link and 'Deliveries', 'Returns' and 'Contact Us':

```
<!--
Default layout, loads most of the pages
-->

<default>

    <!-- Mage_Customer -->
    <reference name="top.links">
        <action method="addLink" translate="label title"
module="customer"><label>Home</label><url></url><title>Home</title><prepare>true</prepare><urlParams/><
position>5</position></action>
        <action method="addLink" translate="label title" module="customer"><label>My Account</label><url
helper="customer/getAccountUrl"/><title>My
Account</title><prepare/><urlParams/><position>94</position></action>
        <action method="addLink" translate="label title"
module="customer"><label>Deliveries</label><url>deliveries</url><title>Deliveries</title><prepare>true</prepar
e><urlParams/><position>95</position></action>
        <action method="addLink" translate="label title"
module="customer"><label>Returns</label><url>returns</url><title>Returns</title><prepare>true</prepare><url
Params/><position>96</position></action>
        <action method="addLink" translate="label title" module="customer"><label>Contact
Us</label><url>contacts</url><title>Contact
Us</title><prepare>true</prepare><urlParams/><position>97</position></action>
    </reference>
</default>
```

Add Home Link to Top Links - Alternative Method

This will allow you to add a *Home* link in the Top Links (My Account | My Wishlist | Etc.) before the My Account.

Find the file called *links.phtml* in *app/design/frontend/default/default/template/page/template/* and make the following change:

```
<?php $_links = $this->getLinks(); ?>

<?php if(count($_links)>0): ?>

    <div>
```

```

<ul<?php if($this->getName()): ?>:?? id="<?php echo $this->getName() ?>"<?php endif;?>>

<!-- HOME BUTTON HACK -->

<li class="first"><a href="<?php echo $this->getUrl()"?>"><?php echo $this->__('Home') ?></a></li>

<!-- HOME BUTTON HACK -->

<?php foreach($_links as $_link): ?>

    <li <?php if($_link->getIsLast()): ?> class="last"<?php endif; ?><?php echo $_link->getLiParams()
?>><?php echo $_link->getBeforeText() ?><a href="<?php echo $_link->getUrl() ?>" title="<?php echo $_link-
>getTitle() ?>" <?php echo $_link->getAParams() ?>><?php echo $_link->getLabel() ?></a><?php echo $_link-
>getAfterText() ?></li>

    <?php endforeach; ?>

</ul>

</div>

<?php endif; ?>

```

NOTE: The if statement in the foreach loop has changed since “Home” will always be first, it wasn’t needed. Also note that the for the “Home” link automatically gets the class “first”.

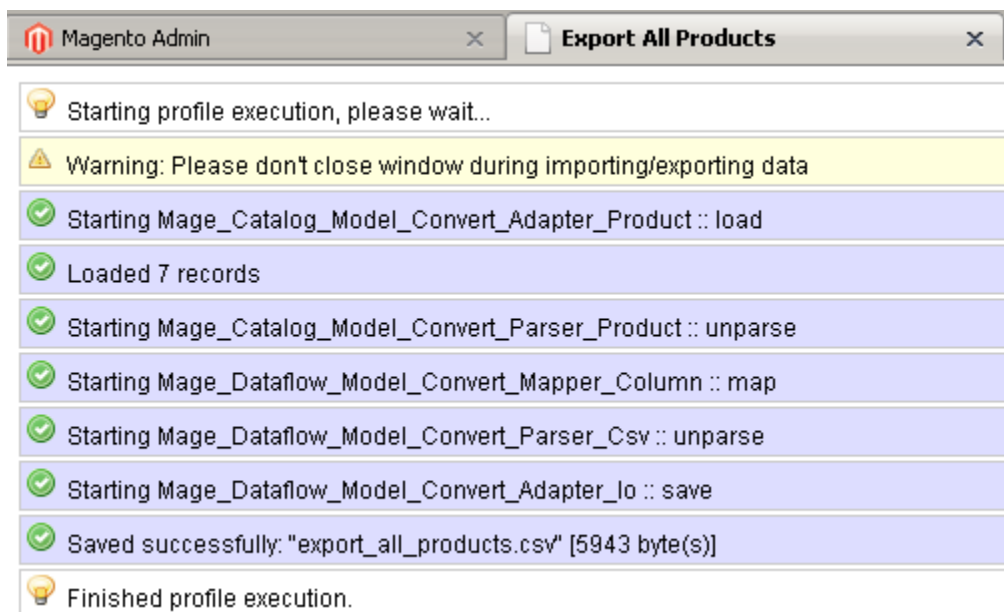
This allows for space between Home | My Account

Magento's Import/Export Profile

<http://www.magentocommerce.com/blog/comments/from-the-support-team-using-magentos-import-export-profile/>

First, let's take a look at how to create an import file, which will include all the data we want to bring into Magento. To get the correct fields and values, we recommend exporting a product, i.e. a simple product or configurable product, through the Admin. This is done through **'System -> Import/Export -> Profiles'**.

In the out-of-the-box Magento configuration, there are a couple of default "Export profiles" that you may use depending on which data you would like to export. This depends on the data which you wish to export, and there are options for customers, products, stock, etc. Running the profile will open a new window or tab which will show the progress of the export profile, as you can see in the example below:



The export file is saved in the **/var/export** folder in your running Magento instance. The screenshot below shows a sample of the export file created when running the "Export All Products" profile, filtering only the simple products:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	store	websites	attribute_type	category_sku	has_option	name	meta_title	meta_description	image	small_image	thumbnail	url_key	url_path	custom		
2	admin	base,dom	Default	simple	8	Simple Te	0	Simple Test	Simple Te	Simple Test				simple-te	simple-test.html	
3	admin	base,dom	Default	simple	8	Simple Te	0	Simple Test 1	Simple Te	Simple Test 1				simple-te	simple-test-10.htm	

Click image for full-size

Now that you have a template to work with, edit the file and add the relevant information which you would like to import.

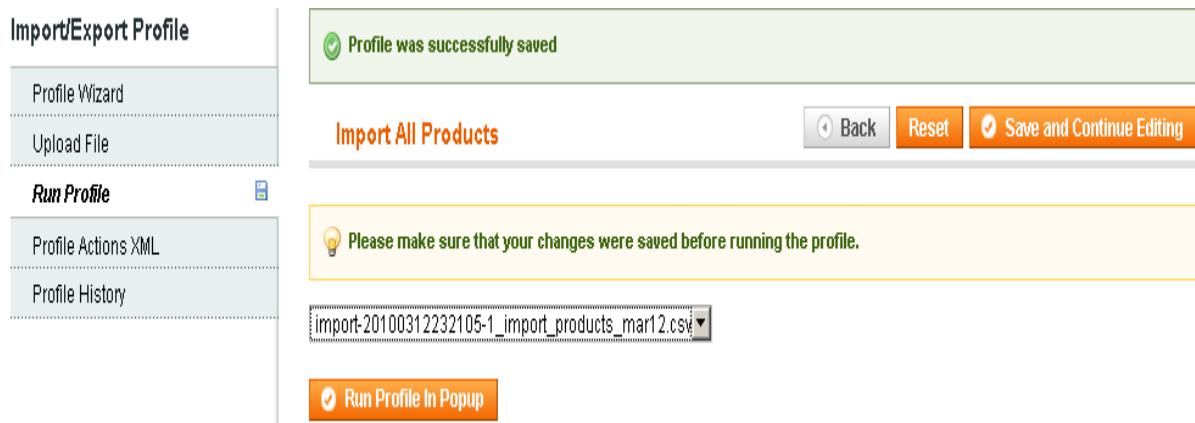
We now come to the most important part of the process which is often missed when saving the import file and causes many of the issues which are referred to us.

It is very important to note that **the import file must be in UTF-8 encoding** to ensure that the file can be read and processed by Magento. Microsoft Excel, OpenOffice Calc, Apple Numbers and GNU Iconv, amongst many other applications have the capabilities of encoding a file in this format.

In addition to that, the **PHP memory limit** must be set with a value that meets the Magento System Requirements.

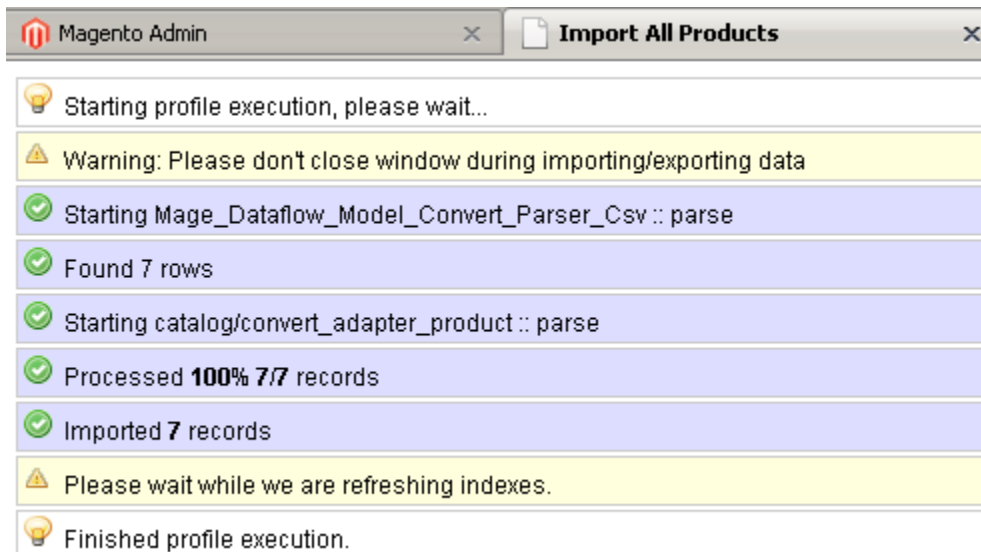
Once you have completed updating the file and checked that the memory limit has met the system requirements, you may now run the import profile in the Admin back office under **'System -> Import/Export -> Profiles'**.

In this tutorial, we are going to use "Import All Products". First, we must upload the file and click "Save and Continue Editing" so that the file will be uploaded into folder **/var/import** in the server. After saving, select the uploaded file from the dropdown list and run the profile as we see below:



Click image for full-size

The screenshot below shows that we have successfully imported our data into Magento:



The New Magento Connect Design (part 1)

<http://www.magentocommerce.com/blog/comments/the-new-magento-connect-design-part-1/>

The first major update I'd like to announce is to our extension categories. Previously, the amount of categories wasn't vast enough to cover the breadth of products that our community was offering, and to make things more confusing, there was no cap on the amount of categories an extension could be listed within. This resulted in a very poor experience, and now that we seeing [so many new extensions on our marketplace](#), it is a perfect time introduce two updates in this area. So without further delay.....

Two-Category Limit for Extensions

First, we're placing a cap on the number of categories an extension can be listed in. From now on, extensions can only be associated to two —two. This will make it much easier for merchants find extensions

New Categories

Second, we're introducing a new suite of categories (listed below). When the new design goes live, developers will be able to login to their account and re-categorize their extensions. We're also working on a few contingency plans for developers who may not immediately be able to make this update.

Here are the new categories:

- * Administration
- * Advertising & Marketing
- * Analytics
- * Automation
- * Billing & Invoicing
- * Catalog
- * Checkout
- * Content Management
- * Customer Service
- * Data
- * Design & Themes
- * Gifting
- * Integration
- * Internationalization & Localization
- * Merchandizing
- * Migration tools
- * Mobile
- * Optimization
- * Other
- * Payments & Gateways
- * Performance
- * Personalization
- * Pricing & Promotion
- * Search
- * SEO
- * Shipping
- * Shopping Cart
- * Widgets

Installing Magento Enterprise stand-alone on OS X

By Mark Hopwood's

The following is based on Enterprise but its very similar to a community edition

the steps involved in installing it locally, on my MacBook.

Install php and MySQL

OS X comes with Apache installed, but you'll need to install php and MySQL before you can install Magento. I got mine from [here](#) and they work perfectly: just download the dmg files and run them.

Install phpMyAdmin

Once you have php, MySQL and Apache all set up, you can install phpMyAdmin by downloading it from [here](#).

Installing phpMyAdmin and getting that working proves:

- Apache is working and pointing to the right place
- php is working
- MySQL is working and accessible from Apache / php

Create a host file entry for your local Magento install

Magento hates working on localhost, and the easiest way to work around that is to edit your host file and add a fictitious domain, which will actually be served by your laptop.

Go to the command line and type in 'sudo nano /etc/hosts' then type in your password (assuming you're an administrator on your laptop) and add a line like the following:

```
127.0.0.1 magento.enterprise.com
```

Save the file (Ctrl-O) and quit nano (Ctrl-K) and then check you can ping magento.enterprise.com. If you can, you're ready to install Magento Enterprise. The first step is to install the sample data.

Install sample data

With Magento Enterprise 1.7 you can use the standard sample data that came with Magento Enterprise 1.2, available [here](#). Download the zip file and unzip it, then go to phpMyAdmin, create a database (mine is called magentoenterprise) and import the SQL from the sample data file. It's important you do this before you install the Magento application itself.

Install the application

I can't link to a downloader for Magento Enterprise, but this is the point where you will unzip that, and copy it to a subdirectory of your sites folder. Mine is at ~/Sites/mage_ee which (if you've used magento.enterprise.com in your host file) will mean Magento is at http://magento.enterprise.com/username/mage_ee after you've copied it from the zip file.

You might need to set access on the files after you've unzipped them: to do this go to your site's root directory (~/.Sites/mage_ee in my case) and type in 'sudo chmod -R -v 777 *' and enter your password when prompted. This is poor security in the real world, but fine for a demo site on a laptop, especially one that's using a fictitious domain name.

If you type in the URL in a regular browser, you should be taken to the install script, which (if you've followed all the steps above) should work perfectly first time.

Critical things to remember:

- The URL is the one you created in your hosts file
- The database name, login and password were all set up in phpMyAdmin

Once you're finished with the installation, you can copy the images from the sample data zip file to the media folder in your Magento directory, and your sample store will then have all the pretty pictures as well.

I can add more detail in most places in this article, but for most people I hope the sequence of steps is the main thing. If you do this in the wrong order, you'll have to start again, so follow the sequence carefully.

Pre-Production System Configuration Checklist

<http://www.magentocommerce.com/blog/comments/from-the-support-team-pre-production-system-configuration-checklist/>

it's time to take your shop live! Here's an Admin System Configuration checklist which you can use once your site is staged, in preparation for going live. We'll go through the back end, section by section and let you know which pieces are relevant and important.

1: System -> Configure -> General

General -> Website Restrictions

Website Restrictions	
Access Restriction	<input type="text" value="No"/> [WEBSITE]

Ensure that the website is not restricted to customers.

Web -> Unsecure URL

Unsecure	
Base URL	<input type="text" value="http://domain.com/"/> [STORE VIEW]

Web -> Secure URL

Secure	
Base URL	<input type="text" value="https://domain.com/"/> [STORE VIEW] <small>▲ Please make sure that Base URL ends with '/' (slash), e.g. http://yourdomain/magento/</small>

A 100% Signed and Trusted Certificate should be installed in the server when using secure base URL.

Web -> Default Pages

Default Pages	
Default web url	<input type="text" value="cms"/> [STORE VIEW]
CMS Home Page	<input type="text" value="Home page"/> [STORE VIEW]

The "Home Page" CMS can be redesigned according to your desired look in the Admin panel, under *CMS -> Pages -> Manage Content -> Home page*

Design -> Package



Package

Current package name [STORE VIEW]

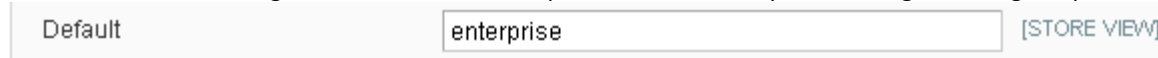
[+ Add Exception](#) [STORE VIEW]

▲ Match expressions in the same order as displayed in the configuration.

If you are running [Magento Enterprise Edition](#) you have so many more features, each with its own template and layout files in the default Enterprise theme. Thus, it is necessary to use the “enterprise” package or to synchronize your package with the enterprise package to ensure that all features have their corresponding templates and layout.

Design -> Themes

You can use your own templates, skin and layouts by specifying their name. The **default** field should contain a default theme which Magento will use in case the personalized theme you are using is missing template files.



Default [STORE VIEW]

Design -> HTML Head

Update the information according to your business information.

Design -> Header

Update the Header details to reflect your business logo, description, etc.

The default Enterprise logo is located at `/skin/frontend/enterprise/default/images/logo.gif`. To change this logo, you can either upload a new image named `logo.gif` or update the **Logo Image Src** value.

Design -> Footer

Update the Footer details to contain your business information.

Store eMail Addresses

These email addresses will be used in sending and receiving email notifications (i.e. new order, invoice, shipment, credit memo, product price alerts, newsletters, etc.). So, make sure to update these fields with valid business email addresses.

2: System > Configuration > Sales

Invoice and Packing Slip Design

Ensure that this is configured with your own business information.

Tax

Ensure that the Tax configuration settings are properly configured according to your business tax rules. [Importing tax rates](#) into Magento can easily be done under *Sales -> Tax -> Import / Export Tax Rates*.

Shipping Settings

Origin	
Country	<input type="text" value="United States"/> [WEBSITE]
Region/State	<input type="text" value="California"/> [WEBSITE]
ZIP/Postal Code	<input type="text" value="90064"/> [WEBSITE]
City	<input type="text" value="Los Angeles"/> [WEBSITE]

Ensure that the origin section contains the shipping origin of your business.

Shipping Methods

Enable the carrier and appropriate shipping methods that your company uses.

If you're giving free shipping to some of these carriers, you can check out a previous article on different ways of [offering free shipping](#). Also, you can [limit free shipping](#) to certain countries.

Google API

Magento is integrated with Google API to allow your business to use the robust features of google such as Google Analytics, Google Base, Google Checkout, Google Website Optimizer. If you wish to use these features, you need to have Google Analytics, Base, & Merchant accounts.

PayPal

Same as Google API, Magento is integrated with PayPal to allow your business to use PayPal payment methods. If you're going to use this payment method, ensure that before going live, you aren't using Sandbox Mode, which is the test mode for testing this payment method.

Payment Methods

Enable the payment method(s) that your company uses and ensure that these payment methods are properly configured (i.e. order status, accepted currency, allowed countries, etc.)

3: System -> Advanced

System > Cron (Scheduled Tasks) – all the times are in minutes

Cron (Scheduled Tasks) - all the times are in minutes	
For correct URLs generated during cron runs please make sure that Web > Secure and Unsecure Base URLs are explicitly set.	
Generate schedules every	<input type="text" value="15"/> [STORE VIEW]
Schedule ahead for	<input type="text" value="20"/> [STORE VIEW]
Missed if not run within	<input type="text" value="15"/> [STORE VIEW]
History cleanup every	<input type="text" value="10"/> [STORE VIEW]
Success history lifetime	<input type="text" value="60"/> [STORE VIEW]
Failure history lifetime	<input type="text" value="600"/> [STORE VIEW]

Ensure that Cron jobs are set to run at the desired time. These cron jobs are critical in ensuring that catalog price rules stick with the product prices, newsletters/customer alerts are sent, Google sitemaps are generated, automatic updating of currency rates, and scheduled database logs are cleaned.

For more information about setting up cron jobs, you may refer to [this page](#) on the wiki.

System -> Log Cleaning

Log Cleaning		
Save log, days	<input type="text" value="180"/>	[GLOBAL]
Enable log cleaning	<input type="text" value="Yes"/>	[GLOBAL]
Start Time	<input type="text" value="00"/> : <input type="text" value="00"/> : <input type="text" value="00"/>	[GLOBAL]
Frequency	<input type="text" value="Daily"/>	[GLOBAL]
Error Email Recipient	<input type="text" value="admin@domain.com"/>	[GLOBAL]
Error Email Sender	<input type="text" value="General contact"/>	[GLOBAL]
Error Email Template	<input type="text" value="Log cleanup Warnings (Default Template from"/>	[GLOBAL]

Log cleaning must be enabled to ensure that your database log tables are cleaned. The **Save Log** days should be set to keep logs according your needs and server capacity.

4: System -> Transactional Emails

The default email templates can be updated to reflect your branding more accurately by adding a new template and loading a default email template.

Under the *System -> Configuration* section, the following sections below use transactional email templates and email senders:

General Section

Contacts -> Email Options

Catalog Section

Catalog -> Product Alerts

Email to a Friend

Customers Section

Newsletter -> Subscription Options

Customer Configuration -> Create New Account Options

Customer Configuration -> Password Options

Customer Configuration -> Store Credit Options

Wishlist -> Share Options

Invitations -> Customer Invitation Email Template

Reward Points -> Email Notification Settings

Sales Section

Sales Emails -> Order

Sales Emails -> Order Comments
Sales Emails -> Invoice
Sales Emails -> Invoice Comments
Sales Emails -> Shipment
Sales Emails -> Shipment Comments
Sales Emails -> Credit Memo
Sales Emails -> Credit Memo Comments
Checkout -> Payment Failed Emails
Gift Cards -> Gift Card Email Setting
Gift Cards -> Email sent from Gift Card Account management

Advanced Section

Admin -> Admin User Emails
System -> Log Cleaning

...and last, but definitely not least, once you've gone through this checklist, make sure to **test everything** before going live.

Top-10 Most Popular Magento Extensions This Week (Jan 31 - Feb. 5)

<http://www.magentocommerce.com/blog/comments/top-10-most-popular-extensions-this-week-jan-31-feb-5-x/>

Magentoconnect

A marketplace for the exchange of community and commercial Magento extensions

This week, we're recognizing members of our developer community by showcasing the week's top-10 most popular extensions. Thousands of Magento users come to the Connect marketplace every day, and these ten extensions were the ones that came out on top for this week. Congrats to the developers below!

1. [WYSIWYG Editor](#) (by Fontis)

This extension gives you the option to add a JavaScript WYSIWYG editor to specified admin page text areas

2. [Blog Extension](#) (by aheadWorks)

Fully featured blog extension for Magento

3. [Flash Gallery 'Flip' and the new extension 'CMS Content-Editor](#) (by muc1)

A flash gallery extension with CMS-editing capabilities

4. [Enhanced Admin Products Grid](#) (by nel)

Adds customizable features to the products grid.

5. [Free CMS/Block Frontend Features](#) (by Asia Connect Group)

Dynamic block solution, which allows you to put HTML content to any position within a Magento storefront.

6. [Magento Absolute Theme](#) (by TemplatesMaster)

A free professional Magento theme

7. [FreePOP Theme](#) (by Mage-World)

A simple and clear theme, especially good for merchants selling CDs, DVDs, Music, Movies, etc..

8. [Magento Easy Lightbox](#) (by TemplatesMaster)

Quickly Installs a lightbox widget in under 5 minutes.

9. [Fooman Speedster](#) (by Fooman)

Speeds up your store by reworking how Magento handles the loading of JavaScript and CSS

10. [Exploded Menu](#) (by Raptor Commerce)

Replaces the standard single column drop down with a multi-column dropdown featuring 2nd and 3rd level menu items

Multi-site Domain Name Setup

By [Tomas G](#)

<http://www.magentocommerce.com/blog/comments/from-the-support-team-multi-site-domain-name-setup/>

One of Magento's most superior strengths are its capabilities for scaling to support multi-store and multi-language retailing all from the same backend. In this tutorial, we will be showing you how to take advantage of Magento's scalability by creating multiple websites with unique domain names sharing the same product catalog. (*****NOTE: Each store can also be configured to offer a unique product catalog as well.**)

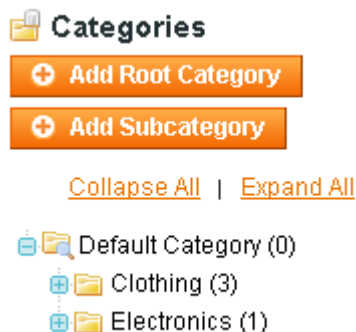
For this example, we'll be attempting to set up **domain1.com**, **domain2.com** and **domain3.com**.

Generally, Magento is installed into the folder `/var/www/http`, as per the [Magento Installation Guide](#), and you can find more information related to the initial setup and configuration of any Magento installation there. We aren't going to through an full blown installation right now though, and for our purposes, we are going to presume that the Magento instance has already been installed in the server.

We'll be dividing the process into steps based on the areas of configuration we will need to deal with—namely, Categories, Store Configuration in Magento Admin, Store Configuration in the Server.

1: Categories

First, will need to create our Categories. Since all three websites will be sharing the same catalog, we will be using the default root Category in *Catalog* -> *Categories* -> *Manage Categories* and will be creating our categories under that root category (i.e. Clothing, Electronics, etc.).



These categories (Clothing, Electronics) should be set as both “**Is Active**” from the *General Information* tab and “**Is Anchor**” from the *Display Settings* tab for them to appear on the frontend of your Magento shop. (*****NOTE: If the websites will not be sharing the same catalog, a Root Category must be created for each website. Thus, if there are 3 websites, there will be 3 Root Categories with subcategories under them.**)

2: Store Configuration in the Magento Admin

1. Now that we have created our Categories, it's time to create our websites by going to *System* -> *Manage Stores* and clicking the “Create Website” button.

- **Name** – domain name of our new website
- **Code** – a parameter that will be used in configuring the Apache web server to point to that particular domain name

New Website

Website Information	
Name *	<input type="text" value="domain1.com"/>
Code *	<input type="text" value="domain1_com"/>
Sort order	<input type="text"/>

2. Once the website has been created, we'll create the store corresponding to this website by clicking on the "Create Store" button in *System -> Manage Stores*.

- **Website** – website to which this store will be associated
- **Name** – the same as the website name
- **Root Category** – the root category that will be used for this store. (Refer to Step 1 for Details)

New Store

Store Information	
Website *	<input type="text" value="domain1.com"/>
Name *	<input type="text" value="domain1.com"/>
Root Category *	<input type="text" value="Default Category"/>

3. Then, we create the store view which is the interface that the customer will be able to access on the frontend. Click the "Create Store View" button in *System -> Manage Stores*.

- **Store** – store to which this view will be associated
- **Name** – name of this store view (i.e. English Version, German Version, etc.)
- **Code** – code for this store view
- **Status** – if enabled, this store view will be accessible from our frontend, otherwise, it will not be accessible

New Store View

Store View Information	
Store *	<input type="text" value="domain1.com"/>
Name *	<input type="text" value="English"/>
Code *	<input type="text" value="domain1_en"/>
Status *	<input type="text" value="Enabled"/>
Sort order	<input type="text"/>

4. After the Store has been created, we need to configure the **Unsecure Base URL** and **Secure Base URL** under *System -> Configuration -> General -> Web*. Before we set their respective base URLs, we first need to ensure that the configuration scope is set to the domain1.com website to define which site we are working on.

Current Configuration Scope:
<input type="text" value="domain1.com"/>
Manage Stores

Then, we modify the base URLs for both **Unsecure**:

Unsecure	
Base URL	<input type="text" value="http://domain1.com/"/> <input type="checkbox"/> Use default [STORE VIEW]

and **Secure**:

Secure	
Base URL	<input type="text" value="https://domain1.com/"/> <input type="checkbox"/> Use default [STORE VIEW]
<small>▲ Please make sure that Base URL ends with '/' (slash), e.g. http://yourdomain/magento/</small>	

with the corresponding domain name by unchecking the **"Use default [STORE VIEW]"** checkbox and then save the configuration.

5. Now we just repeat Steps 2-4 for the other two websites, **domain2.com** and **domain3.com** by replacing the fields with their respective information.

3: Store Configuration in the Server

1. Now we re-configure the Apache configuration file, *httpd.conf*, for all domains to set the **DocumentRoot** to our Magento directory. In this case, the directory is */var/www/http*:

```
<VirtualHost *:80>
  ServerAdmin webmaster@domain1.com
  DocumentRoot /var/www/http
  ServerName domain0.com
</VirtualHost>

<VirtualHost *:80>
  ServerAdmin webmaster@domain2.com
  DocumentRoot /var/www/http
  ServerName domain1.com
</VirtualHost>

<VirtualHost *:80>
  ServerAdmin webmaster@domain3.com
  DocumentRoot /var/www/http
  ServerName domain2.com
</VirtualHost>
```

2. Edit the **.htaccess** file at */var/www/http/.htaccess* and add the following lines below:

```
SetEnvIf Host www\.domain1\.com MAGE_RUN_CODE=domain1_com
SetEnvIf Host www\.domain1\.com MAGE_RUN_TYPE=website
SetEnvIf Host ^domain1\.com MAGE_RUN_CODE=domain1_com
SetEnvIf Host ^domain1\.com MAGE_RUN_TYPE=website

SetEnvIf Host www\.domain2\.com MAGE_RUN_CODE=domain2_com
SetEnvIf Host www\.domain2\.com MAGE_RUN_TYPE=website
SetEnvIf Host ^domain2\.com MAGE_RUN_CODE=domain2_com
SetEnvIf Host ^domain2\.com MAGE_RUN_TYPE=website

SetEnvIf Host www\.domain3\.com MAGE_RUN_CODE=domain3_com
SetEnvIf Host www\.domain3\.com MAGE_RUN_TYPE=website
SetEnvIf Host ^domain3\.com MAGE_RUN_CODE=domain3_com
SetEnvIf Host ^domain3\.com MAGE_RUN_TYPE=website
```

3. Restart Apache Server

If you are on a Red Hat based distribution, you'll be able to type **service apache restart**. For other distributions, you'll want to type **apachectl restart**. (**NOTE: The second option here is different than "apachectl graceful" which will run a graceful restart and reload configuration files, without terminating current connections. We don't have any visitors to our site yet, so it's okay to do a "apachectl restart".)

4: We're Ready to Go!

After we've complete all of these steps we should now see all 3 domains in our backend:

Manage Stores

Page 1 of 1 pages | View 20 per page | Total 4 records found

Website Name	Store Name
domain1.com	
domain2.com	
domain3.com	

All that's left now is to add products to the catalog and give each site a custom theme if we wish to do so. Many people are taking advantage of Magento's powerful multi-store functionality, whether it's to set up stores with multiple languages, different catalogs and even the same catalogs (as in our example) with different front ends as a marketing vehicle. No matter which you decide to use Magento's powerful features for, we hope this post will provide a good starting point to help you get there!

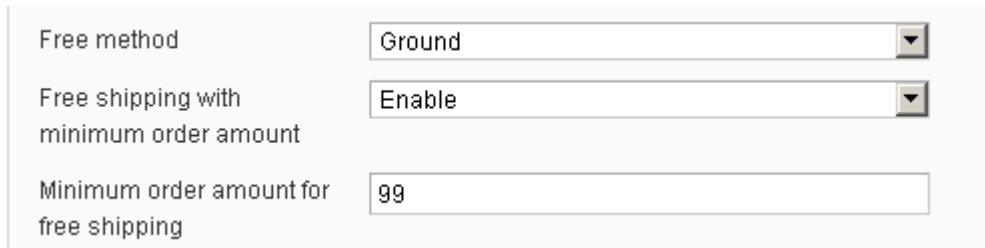
Magento Free Shipping

<http://www.magentocommerce.com/blog/comments/from-the-support-team-boosting-sales-by-offering-free-shipping/>

Right out-of-the-box, Magento has multiple ways to accomplish this depending on the factors you choose to offer free shipping. Let's run through some of the scenarios together and get a better understanding of how to configure Magento in each instance.

Scenario 1: Carrier Free Shipping with minimum order amount (without using a Shopping Cart Price Rule)

In this scenario, we'll be offering free UPS Ground shipping for orders above \$99. To do this we need to go to *System -> Configuration -> Sales -> Shipping Methods -> UPS*, and we need to set the free method to 'Ground' and provide the minimum amount value.



A screenshot of the Magento configuration page for the UPS shipping method. It shows three settings: 'Free method' set to 'Ground', 'Free shipping with minimum order amount' set to 'Enable', and 'Minimum order amount for free shipping' set to '99'.

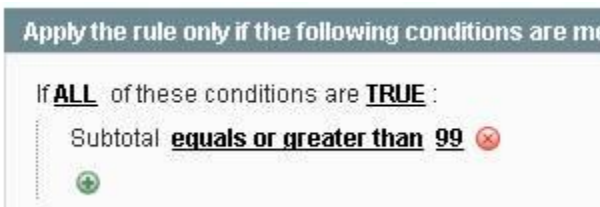
The screenshot below shows the UPS Ground shipping method is now free during shopping cart checkout.



A screenshot of the checkout page showing the 'Shipping Method' section. Under 'United Parcel Service', the 'Ground \$ 0.00' option is selected, while '2nd Day Air \$ 16.39' and 'Next Day Air \$ 30.05' are unselected.

Scenario 2: Carrier Free Shipping with minimum order amount using a Shopping Cart Price Rule

In this scenario, we will be offering free UPS Ground shipping method for orders above \$99 but through a Shopping Cart Price Rule. Here however, we will be using a Coupon Code, set up through *Promotions -> Shopping Cart Price Rules* so that the free shipping will automatically apply if the conditions are met.



A screenshot of a Shopping Cart Price Rule configuration. The header reads 'Apply the rule only if the following conditions are met'. Below it, the condition is set to 'If ALL of these conditions are TRUE :'. One condition is 'Subtotal equals or greater than 99', which is marked with a red 'X' icon, indicating it is currently invalid or not met.

Since our Shopping Cart Rule is not limited to any product attributes, we can either choose "For matching items only" or "For shipment with matching items" in the Shopping Cart Price Rule Actions as Free Shipping option.

Lastly, in System -> Configuration -> Sales -> Shipping Methods -> UPS, we need to set the free method to 'Ground' and, unlike the first scenario, we will not be providing a minimum amount.

Free method	<input type="text" value="Ground"/>
Free shipping with minimum order amount	<input type="text" value="Disable"/>
Minimum order amount for free shipping	<input type="text"/>

Although setup differently, we get the same results as the first scenario.

3 Shipping Method

United Parcel Service

Ground \$ 0.00

2nd Day Air \$ 16.39

Next Day Air \$ 30.05

Scenario 3: Carrier Free Shipping for specific items

In this scenario, we will be offering free UPS Ground shipping for products within a specific category (i.e. Furniture category).

Apply the rule only if the following conditions are met (leave blank for all products)

If **ALL** of these conditions are **TRUE** :

If an item is **FOUND** in the cart with **ALL** of these conditions true: ❌

Category **is 10** ❌

+

Since our free shipment is limited to Furniture items only, we must set the same conditions under the Shopping Cart Price Rules Actions section.

Free shipping	<input type="text" value="For matching items only"/>
Stop further rules processing	<input type="text" value="No"/>

Apply the rule only to cart items matching the following conditions (leave blank for

If **ALL** of these conditions are **TRUE** :

Category **is 10** ❌

+

Now here, if we select “**For matching items only**”, if there are multiple items in the shopping cart, the free shipping will only apply for the product within the Furniture category and other shipment charges on non-furniture items will be calculated separately.

3 Shipping Method
United Parcel Service
 Ground \$ 9.45
 2nd Day Air \$ 32.63
 Next Day Air \$ 53.80

On the other hand, if we choose “**For shipment with matching items**” free shipping will apply to the whole cart with multiple items in it as long as an applicable furniture product is included.

3 Shipping Method
United Parcel Service
 Ground \$ 0.00
 2nd Day Air \$ 32.63
 Next Day Air \$ 53.80

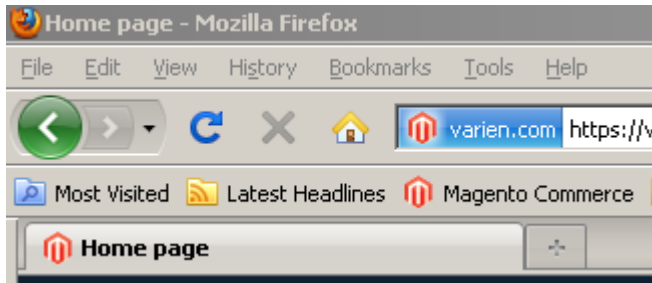
These scenarios all provide just a few of the many techniques to offer free shipping to your customers.

Changing the Magento 'favicon'

<http://www.magentocommerce.com/blog/comments/from-the-support-team-changing-the-magento-favicon/>

we are going to customize the "favicon" – short term for "favorites icon", that is associated with your website and appears in the browser address bar and favorites menu. A favicon is a 16x16 pixel icon in no more than 16 colors.

As you can see in the screenshot below, the current favicon is shown as displayed on a Firefox browser.

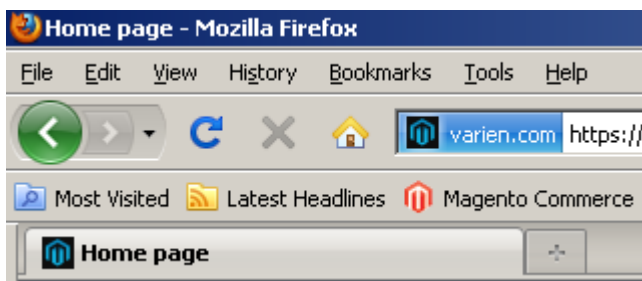


We'll want to update the standard Magento favicon and replace it with this one, a slight variation.

If you're using a default theme, the new favicon.ico needs to be uploaded into the `/skin/frontend/default/default/favicon.ico` directory.

For custom themes, the favicon should go into the `/skin/frontend/default/CUSTOM_THEME/favicon.ico` directory instead.

That was simple and we have successfully updated our favicon! To view it we need to clear the browser cookies and cache and hit refresh.



How to Embed Google Custom Search in Magento

Posted by Branko

<http://www.magentocommerce.com/blog/comments/how-to-embed-google-custom-search-in-magento/>

For those of you who love Google Custom Search and would like to use it with the Magento, here is a little how to. Entire embed process is really easy. It all comes down to copy paste-ing few lines of code from Google to Magento. My idea is to create the static block in Magento CMS section and then use the custom CMS page from which I will call this static block among other HTML content I might wish to throw into the CMS page.

For starters, we need to create [Google Custom Search](#) engine. Attached are few screenshots to see how it looks.



With Google Custom Search, you can harness the power of Google to create a customized search experience for your own website.



- ✓ Include one or more websites, or specific webpages
- ✓ Host the search box and results on your own website
- ✓ Customize the look and feel of the results to match your site

Custom Search for your website or blog

- Provide fast and relevant search results
- Make money with [AdSense for Search](#)
- Invite your friends and community to contribute
- Automatically search across links, bookmarks or blogrolls with [Custom Search on the fly](#)

Site Search for your business or enterprise

- Use [Google Site Search](#) for enterprise-grade support and optional ads
- Improve customer retention and conversions on your site
- Control branding and enhance results presentation via XML results
- Provide quick indexing of your website via [On-Demand indexing](#)

[Create a Custom Search Engine](#)

Or, [manage your existing search engines](#).

Custom Search Blog

[Mobile Custom Search with themes and structure](#)

Nov 13, 2009

We recently announced support for mobile Custom Search for high-end devices like Android-powered phones, iPhone, iPod...

[Three birthday candles for Custom Search](#)

Oct 26, 2009

Last Friday was the third anniversary of Custom Search! Here's a photo of our celebratory cake, baked by John Skidgel, o...

[Plug-n-play with Custom Search Themes](#)

Oct 26, 2009

Search results for "":

Google custom search **Create a Custom Search Engine**

1. Set up your search engine 2. Try it out

Basic information

Give your search engine a name and provide a brief description.

Search engine name:

for example, Real Climate Search Engine

Search engine description:

for example, Climate Science from Climate Scientists

Search engine language:

What do you want to search?

- Only sites I select.
 The entire web, but emphasize sites I select.

Select some sites

Specify a list of websites to search. You'll be able to edit this list and add more sites later. [Tips on formatting URLs.](#)

Sites to search:

List one URL per line

Select an edition

Select the edition you'd like to create. [Learn more.](#)

- Editions: **Standard edition** - Free, ads are required on results pages.
 Business edition - Starts at \$100 per year, no ads on results pages.

I have read and agree to the [Terms of Service](#).

©2008 Google - [Google Home](#) - [About Google](#) - [Privacy Policy](#)

Search results for "":

Google custom search **Create a Custom Search Engine**

1. Set up your search engine 2. Try it out

Preview

Try out some queries in your search engine:

Congratulations!

Click "Finish" to go to your "My search engines" page, where you can access your search engine from its new homepage on Google or use its control panel to further customize it in the following ways:

- Include more sites
- Change the look and feel
- Sign up to make money
- Invite people to contribute
- Create search refinements

Send confirmation email to ajzele@gmail.com

©2008 Google - [Google Home](#) - [About Google](#) - [Privacy Policy](#)

Search results for "":

Google custom search

[Overview](#)

[New search engine...](#)

[My search engines](#)

[My profile](#)

Resources

- [Documentation](#)
- [Google Marker](#)
- [Blog](#)
- [Discussion group](#)
- [Support](#)

Google™ Custom Search

Search

My search engines

You've just created the search engine "My cool search engine". To edit this search engine, get code for a search box to place on your website, or customize it further, use the control panel link below. To see your search engine's homepage on Google and try it out, use the homepage link below.

Search engines I've created

My cool search engine [control panel](#) [statistics](#) [delete](#)

Additional tools

- [Manage your email preferences](#)
- Add the Developer Gadget to your iGoogle homepage for easy access to all your Custom Search Engines. 

©2008 Google - [Google Home](#) - [About Google](#) - [Privacy Policy](#)

Search results for "":

Google custom search

[Overview](#)

[New search engine...](#)

[My search engines](#)

Control panel

[Basics](#)

[Sites](#)

[Indexing](#)

[Refinements](#)

[Promotions](#)

[Look and feel](#) ^{New!}

[Get code](#)

[Collaboration](#)

[Make money](#)

[Business account](#)

[Advanced](#)

[Preview](#)

[Statistics](#)

[My profile](#)

Resources

- [Documentation](#)
- [Google Marker](#)
- [Blog](#)
- [Discussion group](#)
- [Support](#)

Google™ Custom Search

Search

Control panel - Get code: [My cool search engine](#)

To get code for other hosting options, adjust your settings on the [Look and feel](#) page.

Custom Search element code

Paste this code in the page where you'd like the Custom Search element to appear. **Note:** CSS hover effects require a supported doctype such as `<!DOCTYPE html>`.

```
<div id="cse" style="width: 100%;">Loading</div>
<script src="http://www.google.com/jsapi" type="text/javascript"></script>
<script type="text/javascript">
  google.load('search', '1', {language: 'en'});
  google.setOnLoadCallback(function(){
    var customSearchControl = new google.search.CustomSearchControl(
      customSearchControl.setResultSize(google.search.Search.FILTERED_CUSTOM_SEARCH_CONTROL_DRAW('cse')));
    customSearchControl.draw('cse');
  });
</script>
<link rel="stylesheet" href="http://www.google.com/cse/style/look/d
```

Want to do more customization?

You can customize the look and feel even more or extend the functionality of the Custom Search element by:

- [Downloading the CSS source file](#)
- Following the [JavaScript and CSS documentation](#) to add more customizations.

©2008 Google - [Google Home](#) - [About Google](#) - [Privacy Policy](#)

Once we have created the necessary search engine, we will copy-paste the embed code into the static block of Magento, let's say we login to Magento, and create static block with identifier "custom-google-search-engine".

For our last step, we will create a CMS page, giving it SEF URL Identifier "custom-search", and placing the following `{{block type="cms/block" block_id="custom-google-search-engine"}}` block call into the content area.

That's it. Now when you visit the url like `http://myshop.domain/custom-search` you will see the result like shown on photo below.

<http://shop.local/custom-search>

[Home](#) / [Custom Search](#)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras nec pulvinar augue. Vestibulum ullamcorper mattis massa, eu malesuada massa ultricies sit amet. Donec eu pretium purus. Morbi luctus lobortis enim quis viverra. Nulla imperdiet enim ac felis rutrum molestie. Mauris nibh ipsum, facilisis sit amet pretium ac, interdum nec mauris. Fusce vulputate semper est, semper mollis ante pharetra eget. Aliquam at libero ipsum. Etiam pellentesque facilisis libero, non adipiscing nisi ultricies nec. Donec sed neque purus, sed commodo tellus.

x

My cool search engine

[Inchoo - Magento - Member - eCommerce Software for Growth](#)

Inchoo was founded in May 2008, in Osijek, Croatia as an eCommerce business unit of Surgeworks. Our founder, Tomislav Bilic is a senior developer with ...
www.magentocommerce.com

[Magento - Custom search- category listing in select box ...](#)

Now change what search results are retrieved with the **Inchoo** technique given above, insert an 'if' statement below the line <?php \$_code ...
www.magentocommerce.com

[Magento - UPS API Request - Programming Questions - eCommerce ...](#)

Owner of **Inchoo** Ltd., partner of Surgeworks. Posted: September 30 2008: | top ... <http://inchoo.net/tools/ups-api-documentation/>. Signature ...
www.magentocommerce.com

[Magento - can magento view the best seller in home page - Can ...](#)

Inchoo.net has a great tutorial on how to get this feature working. ... [http:// inchoo.net/ecommerce/magento/bestseller-products-in-magento/](http://inchoo.net/ecommerce/magento/bestseller-products-in-magento/) ...
www.magentocommerce.com

[Magento Connect - AmfPoint - Flash Magento Widgets - Reviews ...](#)

Inchoo posted on Mon, July 6, 2009. Hello Skarniva, Take a look at the website. You will see the link to the demo store where you will notice how it looks ...
www.magentocommerce.com

[Magento - Member - eCommerce Software for Growth](#)

Currently he leads 8 people eCommerce development team: **Inchoo**. Tomislav's focus on close interaction with the client helps him to understand their needs ...
www.magentocommerce.com

[Magento - Designer's Guide to Magento PDF download - HTML, XHTML ...](#)

<http://inchoo.net/ecommerce/magento/designers-guide-to-magento-pdf-download/>. smile. Signature. Owner of **Inchoo** Ltd., partner of Surgeworks ...
www.magentocommerce.com

[Magento - How to make attribute display as a color switcher ...](#)

<http://inchoo.net/ecommerce/magento/create-a-color-switcher-in-magento/> ... [http ://inchoo.net/wordpress/kapitol-reef-magento-online-store/](http://inchoo.net/wordpress/kapitol-reef-magento-online-store/) ...
www.magentocommerce.com

[Magento - Shipping rates by box size L x W x H - Feature Requests ...](#)

Many thanks. Signature. Owner of **Inchoo** Ltd., partner of Surgeworks. Posted: September 30 2008: | top | # 24. Magento Community ...
www.magentocommerce.com

[Magento - Forum - AmfPoint - Flash Magento Widgets Discussion ...](#)

Latest Post Info. RSS feed for this forum - AmfPoint - Flash Magento Widgets - Announcement. Author: **Inchoo**. 0, 0. Posted: July 6 2009. Author: **Inchoo** ...
www.magentocommerce.com

1 2 3 4 5 6 7 8 9 10

[CC Image credits.](#)

Reward Points System

Posted by [RoyRubin](#)

<http://www.magentocommerce.com/blog/comments/magento-enterprise-edition-version-17-feature-roadmap/>

The next version of Magento Enterprise Edition features enhancements a Reward Points System, will allow you to better engage customers and increase customer loyalty. Also in the next update to Magento Enterprise Edition are new performance optimizations that provide significant benefits for merchants of all sizes along with performance boosts for Sales Reports and some new functionality in our CMS+ Engine and Magento Widgets to help you with organizing, categorizing and displaying content.

Here is an in depth view of what you can expect to see in the next version of the Magento Enterprise Edition.

Reward Points System

Reward Points functionality allows an online merchant to implement unique programs designed to enhance user experience and increase customer loyalty. Points are awarded based on wide range of transactions and customer actions and easily managed through the back end.

- Value of points determined through a simple dual exchange rate (currency to points and/or points to currency)
- Set exchange rate values per customer or group
- Points system is fully manageable by the administrator through the backend with full view of all current balances, limitation on spending (i.e. minimum accrual), points history
- Administrator can modify balances, add and remove points, acquire and spend points on behalf of customers and more
- Customers can earn points based on transactions, for example:
 - For every \$100 spent, earn 10 reward points
 - Purchase product(s) from brand Y, with a total order amount of \$1,500 and earn 150 reward points
 - Buy product X and earn 500 reward points
- Customers can earn points based on user activity, such as:
 - newsletter sign-up
 - invitations
 - invitation conversions
 - tag submission
 - review submission
- Customers can use points at checkout or apply points via admin orders created in the backend
- Customers can easily manage their earned reward points through “My Account” and have a clear view of how many points they have, how many they can redeem and when they expire

Magento Development: Introduction to Magento Dataflow

<http://www.magentocommerce.com/blog/comments/introduction-to-magento-dataflow/>

Posted by [Koby Oz](#)

This post is guest post by our [Enterprise Partners at BoaBaz](#). It is a good introduction into the Magento Dataflow terms and definitions. It should be a good starting point for developers who want to implement data exchange (import/export) between Magento and some 3rd party system.

Parts of this article were initially posted on Baobaz Blog: [Magento Dataflow](#).

Cet article est consultable en Français sur le blog de Baobaz : [Magento Dataflow](#).

One of major features of e-commerce websites is the possibility to share data with offline sale management systems. Magento made data exchange flexible and quite easy with DataFlow module.

Magento DataFlow is a data exchange framework that use four types of components: adapter, parser, mapper and validator. At current state of development validators are not implemented, but are reserved for future use.

Dataflow profile definition

Dataflow of data exchange process is called profile and defined as XML structure. Magento provides simple wizard-like tool for generation of some basic import/export profiles operating on products or customers entities. Advanced profiles manager is also provided for advanced users able to create XML defining profile without wizard tool and with need to use more custom dataflow operations related also to other entities.

Basic concept is that data exchange process is a set of actions. Each action executes part of the process, depending on its type, which can be for example parsing one set of data into another one with parser or collecting data from a resource with adapter. Common data, passed from action to action is stored within profiles batch container.

Adapter definition

Adapters are responsible for plugging into an external data resource and fetching requested data or saving given data into data resource. For this purpose all adapters implement interface `Mage_Dataflow_Model_Convert_Adapter_Interface` which contains two methods: `load()` and `save()`. Data exchange concept introduced in Dataflow module use adapters to perform 3 action types:

- to load data from resource - using `load()` method
- to save data to resource - using `save()` method
- to process one parsed row - when defined as adapter/method pair of variables of parser

For first two actions adapter's XML definition looks like that:

```
<action type="dataflow/convert_adapter_io" method="load">
...
</action>
```

Action tag has two parameters: type and method. Type tells us which adapter class is going to be used to perform action. It is defined using its alias. Method tells us which method of this adapter class should call. By default there are two available methods: load and save. Children of action tag define variables which are parameters used during execution of adapter's method. Variables are defined like in the example below:

```
<action type="dataflow/convert_adapter_io" method="load">
  <var name="type">file</var>
  <var name="path">var/import</var>
  <var name="filename"><![CDATA[products.csv]]></var>
  <var name="format"><![CDATA[csv]]></var>
</action>
```

Magento DataFlow standard adapters

Magento DataFlow module includes few default adapter classes which you can find in `app/code/core/Dataflow/Model/Convert/Adapter` folder. Not all of them have yet implemented `load()` and `save()` methods.

For common case of reading data from or saving data to local or remote file you will use `dataflow/convert_adapter_io` (`Mage_Dataflow_Model_Convert_Adapter_Io`).

Following variables will allow you to define local/remote file as data source:

- `type` - defines type of io source we want to process. Valid values: file, ftp
- `path` - defines relative path to the file
- `filename` - defines data source file's name
- `host` - for ftp type it defines the ftp host
- `port` - for ftp type it defines the ftp port; if not given, default value is 21
- `user` - for ftp type it defines the ftp user, if not given default value is 'anonymous' and password then is 'anonymous@noserver.com'
- `password` - for ftp type it defines the ftp user's password
- `timeout` - for ftp type it defines connection timeout; default value is 90
- `file_mode` - for ftp type it defines file mode; default value is FTP_BINARY
- `ssl` - for ftp type if it is not empty, then ftp ssl connection is used
- `passive` - for ftp type it defines connection mode; default value is false

Customer and Product adapters

For most commonly exchanged entities - customer and product - Magento provides default adapters: `customer/convert_adapter_customer` (`Mage_Customer_Model_Convert_Adapter_Customer`) and `catalog/convert_adapter_product` (`Mage_Catalog_Model_Convert_Adapter_Product`). Both inherit from `Mage_Eav_Model_Convert_Adapter_Entity`.

To simply load all customers data for selected store you can use the following xml:

```
<action type="customer/convert_adapter_customer" method="load">
  <var name="store">default</var>
</action>
```

Sometimes you may want to load only defined group of customers from database. To help you with this there are available following filtering variables:

- filter/firstname - to load only customers with firstname starting with value of this variable
- filter/lastname - to load only customers with lastname starting with value of this variable
- filter/email - to load only customers with email starting with value of this variable
- filter/group - to load only customers from group with id equal to value of this variable
- filter/adressType - to export only selected addressType; valid values are: both, default_billing, default_shipping
- filter/telephone - to load only customers with telephone starting with value of this variable
- filter/postcode - to load only customers with postcode starting with value of this variable
- filter/country - to load only customers with country iso code equal to value of this variable
- filter/region - to load only customers with region equal to value of this variable (for US just 2-letter state names)
- filter/created_at/from - to load only customers created after a date defined as value of this variable
- filter/created_at/to - to load only customers created before a date defined as value of this variable

For example:

```
<action type="customer/convert_adapter_customer" method="load">
  <var name="store"><![CDATA[0]]></var>
  <var name="filter/firstname"><![CDATA[a]]></var>
  <var name="filter/lastname"><![CDATA[a]]></var>
  <var name="filter/email"><![CDATA[a]]></var>
  <var name="filter/group"><![CDATA[1]]></var>
  <var name="filter/adressType"><![CDATA[default_billing]]></var>
  <var name="filter/telephone"><![CDATA[1]]></var>
  <var name="filter/postcode"><![CDATA[7]]></var>
  <var name="filter/country"><![CDATA[BS]]></var>
  <var name="filter/region"><![CDATA[WA]]></var>
  <var name="filter/created_at/from"><![CDATA[09/22/09]]></var>
  <var name="filter/created_at/to"><![CDATA[09/24/09]]></var>
</action>
```

Same way you can load and filter products loaded from database with following variables:

- filter/name - to load only products with name starting with value of this variable
- filter/sku - to load only products with sku starting with value of this variable
- filter/type - to load only products with type defined as value of this variable; valid values are: simple, configurable, grouped, bundle, virtual, downloadable
- filter/attribute_set - to load only products with attribute set id equal to value of this variable
- filter/price/from - to load only products with price starting from value of this variable
- filter/price/to - to load only products with price up to value of this variable
- filter/qty/from - to load only products with quantity starting from value of this variable
- filter/qty/to - to load only products with quantity up to value of this variable
- filter/visibility - to load only products with visibility id equal to value of this variable
- filter/status - to load only products with status id equal to value of this variable

Example:

```
<action type="catalog/convert_adapter_product" method="load">
  <var name="store"><![CDATA[0]]></var>
  <var name="filter/name"><![CDATA[a]]></var>
  <var name="filter/sku"><![CDATA[1]]></var>
  <var name="filter/type"><![CDATA[simple]]></var>
  <var name="filter/attribute_set"><![CDATA[29]]></var>
  <var name="filter/price/from"><![CDATA[1]]></var>
  <var name="filter/price/to"><![CDATA[2]]></var>
  <var name="filter/qty/from"><![CDATA[1]]></var>
  <var name="filter/qty/to"><![CDATA[2]]></var>
  <var name="filter/visibility"><![CDATA[2]]></var>
  <var name="filter/status"><![CDATA[1]]></var>
</action>
```

Parser definition

Parsers are responsible for transforming data from one format to another. Parser's interface `Mage_Dataflow_Model_Convert_Parser_Interface` defines two methods required in each parser: `parse()` and `unparse()`. Definition of parser can be as simple as:

```
<action type="dataflow/convert_parser_serialize" method="parse" />
```

Similarly to adapter we define action tag with two attributes: `type`, which tells which class we want to use and `method` of this class we want to execute. Of course there is possibility to define variables within action tag body as you will see below.

Magento DataFlow standard parsers

Magento DataFlow includes few standard parsers which you can find in `app/code/core/Dataflow/Model/Convert/Parser`.

The simplest of standard parsers is `dataflow/convert_parser_serialize` (`Mage_Dataflow_Model_Convert_Parser_Serialize`) which doesn't require any variables passed. It requires though that any of previous actions set data within profile's container. Method `parse()` unserialize data stored within profile's container and replace it with the result. Method `unparse()` do the opposite, so it serializes data stored within profile's container and replace it with the result.

One of most often used standard parsers is `dataflow/convert_parser_csv` which allows transforming from (with method `parse()`) or to (with method `unparse()`) CSV file. Example of definition:

```
<action type="dataflow/convert_parser_csv" method="parse">
  <var name="delimiter"><![CDATA[,]></var>
  <var name="enclose"><![CDATA[""]></var>
  <var name="fieldnames">true</var>
  <var name="store"><![CDATA[0]]></var>
  <var name="decimal_separator"><![CDATA[.]></var>
  <var name="adapter">catalog/convert_adapter_product</var>
```



```
<var name="method">parse</var>
</action>
```

This parser requires that you call some IO adapter prior to its execution (using for example `dataflow/convert_adapter_io` to read some CSV file) if you want to call method `parse`. If you want to store data into CSV file you have to do both - call any action that will set data within profile's container prior to parser execution and call IO adapter after parser execution to store data within file.

Following variables will allow you to customize CSV file parsing:

- `delimiter` - defines delimiter used in CSV file; defaults to comma (,) character
- `enclose` - defines what character is used to enclose data values; defaults to empty character
- `escape` - defines escape character for CSV file; defaults to `\\`
- `decimal_separator` - defines decimal separator sign
- `fieldnames` - if set to true, it is assumed first row of CSV file contains field names; if set to false map variable is used
- `map` - defines fieldnames for files where first row doesn't contain fieldnames; to see how to define a map take a look at section of this article related to mapping values
- `adapter` - tells which adapters method should be called on each row
- `method` - tells which method of adapter should be called on each row; defaults to `saveRow`

All variables defined within parser's action body are passed to the defined adapter, so if you need to pass something to it, you can simply set required variable within parser's action body.

Last of standard parsers included within DataFlow module is `dataflow/convert_parser_xml_excel` (`Mage_Dataflow_Model_Convert_Parser_Xml_Excel`), which converts data from and to Excel XML file. Example of definition:

```
<action type="dataflow/convert_parser_xml_excel" method="unparse">
  <var name="single_sheet"><![CDATA[products]]></var>
  <var name="fieldnames">true</var>
</action>
```

Use requirements are the same as for `dataflow/convert_parser_csv`.

Following variables will allow you to customize CSV file parsing:

- `fieldnames` - if set to true, it is assumed first row of CSV file contains field names; if set to false map variable is used
- `map` - defines fieldnames for files where first row doesn't contain fieldnames
- `single_sheet` - tells if parsed should be one sheet or all; should contain name of the sheet to be parsed
- `adapter` - tells which adapters method should be called on each row
- `method` - tells which method of adapter should be called on each row; defaults to `saveRow`

Standard customer and product entity parsers

For most commonly exchanged entities - customer and product - Magento provides also standard parsers: `customer/convert_parser_customer` (`Mage_Customer_Model_Convert_Parser_Customer`) and `catalog/convert_parser_product` (`Mage_Catalog_Model_Convert_Parser_Product`). Both inherit from `Mage_Eav_Model_Convert_Adapter_Entity`.

Since standard adapter's `load()` methods calls result with array of solely entities' id values it is required to call parser's `unparse` method, if we want to get more detailed data. Both parsers take this arrays and for each entity

parse its data variable content, ignore system fields, objects, non-attribute fields and create an associative array from the rest. Additionally product parser add to the array result of parsing product related stock item object, and customer parser - result of parsing shipping and billing addresses and information about newsletter subscription.

Both entities parsers have deprecated parse() methods, since their function is now mostly done by parser actions with standard adapter methods called within parser's context. Example of product parser definition, parsing only products from selected store:

```
<action type="catalog/convert_parser_product" method="unparse">
  <var name="store"><![CDATA[1]]></var>
</action>
```

Mapping values

DataFlow module provides also a mapper concept - class with map() method that is responsible for mapping processed fields from one to another. The definition of mapper looks like that for example:

```
<action type="dataflow/convert_mapper_column" method="map">
  <var name="map">
    <map name="category_ids"><![CDATA[catégorie]]></map>
    <map name="sku"><![CDATA[reference]]></map>
    <map name="name"><![CDATA[titre]]></map>
    <map name="description"><![CDATA[description]]></map>
    <map name="price"><![CDATA[prix]]></map>
    <map name="special_price"><![CDATA[special_price]]></map>
    <map name="manufacturer"><![CDATA[marque]]></map>
  </var>
  <var name="_only_specified">true</var>
</action>
```

Again we have action tag with two attributes: type set as mapper class alias and method that is called to do the mapping. Mapper dataflow/convert_mapper_column is a standard mapper you can find in Magento DataFlow module within app/code/core/Dataflow/Model/Mapper/ folder, and its purpose is to map one array into another with changing the name and possibility to limit fields in result. Map's tag attribute name tells which field name should be replaced in new array by field named like the content of map's tag. If named field doesn't exist in source array, value for target's array field is set to null. Variable _only_specified tells if only fields specified in map definition should be in the resulting array.

Importing Newsletter Subscribers

<http://www.magentoocommerce.com/blog/comments/from-the-support-team-importing-newsletter-subscribers/>

Among the many features that can help you build your brand and increase customer loyalty and draw customers to your site is sending out newsletters in order to raise awareness of your site's new products, promotions, etc. Of course Magento has great direct marketing capabilities built right in, and so we often times receive questions about the challenges of importing subscribers from another system into your Magento System. With Magento, this can be done pretty easily with the dataflow feature found in the Admin panel (*System -> Import/Export -> Profiles*).

First, we need to find out the file format that we are going to use for importing the data. To do this, I will create a profile that will export some customer records from the database by using the export filters in the Profile Wizard tab.



Click Image to Enlarge

Once the file has been successfully exported, the file (as shown below) will contain several columns including the `is_subscribed` field which is for the newsletter subscription. The value '0' is for unsubscribed, while '1' is for subscribed.

AH	AI	AJ	AK	AL
ipping_telephone	shipping_company	shipping_fax	created_in	is_subscribed
13-123-1233			default	0
13-123-1234			default	0

Click Image to Enlarge

Currently, we have two customers who have unsubscribed to our Newsletter.

The image shows a table titled 'Newsletter Subscribers' in the Magento Admin panel. The table has columns for ID, Email, Type, Customer Firstname, Customer Lastname, Status, Website, Store, and Store View. There are two rows of data, both with a status of 'Unsubscribed'. The table also includes a search bar, a 'Refresh Filter' button, and an 'Export' button set to 'CSV'.

ID	Email	Type	Customer Firstname	Customer Lastname	Status	Website	Store	Store View
6	ga71@varien.com	Customer	ga71	varien	Unsubscribed	Main Website	Main Store	English
5	ga67@varien.com	Customer	Magento	Customer	Unsubscribed	Main Website	Main Store	English

Click Image to Enlarge

Using the Import Customers profile, we will upload the new file containing the updated list of customers who are currently subscribed to the newsletter.

AH	AI	AJ	AK	AL
shipping_telephone	shipping_company	shipping_fax	created_in	is_subscribed
			default	1
1231231234			default	0
13-123-1233			default	1
13-123-1234			default	0
13-123-1233			default	1
13-123-1234			default	1

Click Image to Enlarge

As you can see, the Newsletter Subscribers list in Magento has now been updated with the new subscribers.

Newsletter Subscribers

Page 1 of 1 pages | View 20 per page | Total 6 records found

Export to: CSV Export Reset Filter Search

Select All Unselect All Select Visible Unselect Visible 0 items selected Actions Submit

ID	Email	Type	Customer Firstname	Customer Lastname	Status	Website	Store	Store View
<input type="checkbox"/> 10	test2@test2.com	Customer	Customer	Two	Subscribed	Main Website	Main Store	English
<input type="checkbox"/> 9	test1@test.com	Customer	Customer	One	Subscribed	Main Website	Main Store	English
<input type="checkbox"/> 8	qa72@vanen.com	Customer	qa72	vanen	Subscribed	Main Website	Main Store	English
<input type="checkbox"/> 7	john.doe@example.com	Customer	John	Doe	Subscribed	Main Website	Main Store	English
<input type="checkbox"/> 6	qa71@vanen.com	Customer	qa71	vanen	Unsubscribed	Main Website	Main Store	English
<input type="checkbox"/> 5	qa67@vanen.com	Customer	Magento	Customer	Unsubscribed	Main Website	Main Store	English

Click Image to Enlarge

Varien's Popular Open Source Magento eCommerce Software to Ship with Zend Server Community Edition PHP Stack

<http://www.magentocommerce.com/blog/comments/zend-and-varien-partner-to-deliver-an-all-in-one-php-and-ecommerce-solution/>

The open source Magento eCommerce platform, developed with PHP 5 and Zend Framework, has been adopted by thousands of users and is implemented by businesses with Web applications processing many tens of thousands of orders per day using multiple servers and cluster architectures. The no-cost Magento Community Edition provides a basic and powerful solution for small shops looking for a state-of-the-art ecommerce platform. Users may upgrade to the Magento Enterprise Edition for more advanced features and support.

Zend Server Community Edition (CE) is a fast, reliable, no-cost PHP application stack that can be used in development, testing and production. Zend Server CE runs on Linux, Microsoft Windows, and Mac OS X and is specifically tuned to address the needs of developer desktops and laptops, as well as smaller server deployments everywhere. Zend Server is also available in a commercial, enterprise edition, which is a complete Web application server for running, monitoring and managing business-critical PHP applications that require a high level of reliability, performance, and security.

Using Magento CE with Zend Server CE, small and medium-sized businesses can quickly install, explore and start building state-of-the-art shopping and other commerce-based Web sites with features such as catalog and order management, analytics, SEO, merchandizing, customer accounts and shipping, and then deploy them with ease on Zend Server. Users can develop Magento eCommerce applications on desktops running the Windows or Linux operating systems and deploy applications into production on both Windows and Linux. To further expand the ecommerce functionality readily available to PHP developers, Varien is also modularizing Magento and making a number of significant ecommerce feature contributions to Zend Framework, including payment gateways, shipping calculation modules, and others.

"Zend's PHP solutions have played a major role in the development of Magento, and we are thrilled to contribute key ecommerce features back to Zend Framework to promote the development of PHP-based ecommerce," said Roy Rubin, CEO at Varien. "Zend continues to demonstrate its unmatched expertise and strong commitment to PHP with products such as Zend Server, and we look forward to continuing to expand our partnership with them. In particular, we're very excited that Zend Server's support for PHP on Windows is unmatched anywhere else, as this makes exploring powerful ecommerce easy for businesses everywhere."

Wonderbox is a vendor of romantic/adventure gift packages and does very high volumes during the holiday season and other gift-giving occasions, such as Valentine's Day, Mother's Day, and so on. Wonderbox has an online language-specific presence in France, Italy, Germany, Spain, and Poland, and is opening soon in Japan, Belgium, Portugal, and elsewhere—so the international capabilities of Magento and Zend Framework are extremely important to their product plans, in addition to the open-source flexibility both are known for.

"Our Wonderbox site has been engineered for over 150,000 transactions a day and we estimate we will serve over 3 million visits during the December holiday season. We recently went from zero to a fully functioning production Web application taking customer orders in less than three months, by running Magento eCommerce on Zend Server," said Fatih Gezen, chief technology officer at Wonderbox. "That is a compelling ROI and amazing

technology story. We have in a very short time mastered the technology and can easily create intuitive online experiences for our customers to use to make purchases from us.”

“Zend Server out of the box gives us a secure, high-performing PHP stack with monitoring and other capabilities and the feature set in Magento had just about everything we needed to build our ecommerce site,” commented Chris Mann, deputy information director at Wonderbox. “We’ve additionally built a custom Magento module in Zend Framework. That has been easy to do based on the excellent training and support we’ve received from our consulting partners and from the large open-source community that’s available for both products.”

In addition to consulting partners who specialize in Magento/Zend Server installations, Zend and Varien also plan to cooperate on joint education, training, and other activities to promote ecommerce applications, including promotion of software lifecycle best practices and enterprise ecommerce training.

Availability

The complete all-in-one Magento and Zend Server solution stack is available today from both companies for download, meaning that together the huge Magento and Zend communities will have available in one package the leading PHP ecommerce solution and the best performing PHP stack. To learn more and to download these packages, please visit: <http://www.magentocommerce.com/product/magento-zend> or <http://www.zend.com/solutions/packaged-php-applications/zend-server-magento>.

As announced today, the Magento eCommerce feature components will be made available for download from the Zend Framework project and will also ship as a part of the base Zend Framework that comes with every Magento eCommerce download. These will be made available as soon as the components are added to the Zend Framework project by the two teams.

Replacing the Logo Image in Transactional Emails

<http://www.magentocommerce.com/blog/comments/from-the-support-team-replacing-the-logo-image-in-transactional-emails/>

Out-of-the-box Magento uses its own demo logo as the image in these transactional emails, stored under the file name logo_email.gif. This can be easily replaced by uploading your logo image in your theme skin directory and give your emails a personal and professional touch.

One important thing to note is that in the root directory, Magento has two main directories for the overall store theme, the /app and /skin directories. The /app directory contains the layout, translations for labels and terms, and templates. On the other hand, the /skin directory contains the graphical elements which are images, style sheets (CSS), and JavaScript for the blocks.

In this article, we will be using the default Magento theme. Therefore, the images we will be using will be located in the directory /skin/frontend/default/default/images. The image below shows the default logo image in the transactional email.



Hello Magento Customer,

Thank you for your order from Main Store. Once your package ships we will send an email with a link to track your order. You can check the status of your order by [logging into your account](#). If you have any questions about your order please contact us at dummyemail@magentocommerce.com or call us at (555) 555-0123 Monday - Friday, 8am - 5pm PST.

To replace the logo image, upload the new logo image into the images directory and rename the file as logo_email.gif, overwriting the existing image. Also, and very importantly, to ensure that the new image will show up in the transactional emails, the permissions need to be changed so that it includes read and write permissions.



Hello Magento Customer,

Thank you for your order from Main Store. Once your package ships we will send an email with a link to track your order. You can check the status of your order by [logging into your account](#). If you have any questions about your order please contact us at dummyemail@magentocommerce.com or call us at (555) 555-0123 Monday - Friday, 8am - 5pm PST.

Voila! That's all we had to do we've changed the logo image--a very small and simple, yet very nice touch.

How to Configure Magento Widgets

<http://www.magentocommerce.com/blog/comments/introducing-magento-widgets/>

Magento Widgets

Overview

Magento Widgets allow **business users with no technical knowledge** to easily add dynamic content (including product data, for example) to pages in Magento Stores. This allows for greater control and flexibility in creating informational and marketing content through administrator tools, enabling intuitive and efficient control of content such as:

1. Dynamic product data in Marketing Campaign Landing Pages
2. Dynamic Information such as Recently Viewed Items into Content Pages
3. Promotional images to position in different blocks, side columns and other locations throughout the storefront
4. Interactive elements and action blocks (external review systems, video chats, voting and subscription forms)
5. Alternative navigation elements (tag clouds, catalog image sliders)
6. Create interactive and dynamic flash elements easily configured and embedded within content pages for enhanced user experience
7. And lots more ...

What are Widgets?

Magento Widgets are [frontend blocks](#) with a predefined set of configuration options. These configuration options are displayed in a special edit form in the backend panel when a widget is being added or edited by a store owner. The ability to easily set widget configuration options allows for the full control of widget placement on a page to store owners.

Widget Insertion



Widget


Widget Type * Catalog Product Link
Link to a Specified Product






Widget Options

Anchor Custom Text
If empty, the Product Name will be used

Template Product Link Block Template

Product * Simple Product

 Default Category Close

-  Category 1
 -  Subcategory 1
 -  Subcategory 3
 -  Subcategory 2
-  Category 2

Page 1 of 1 pages Reset Filter Search

View 20 per page | Total 1 records found

ID	SKU	Product Name
1	simple	Simple Product

New Widget Instance

Frontend Properties

Type Catalog Product Link

Design Package/Theme default / default

Widget Instance Title *

Sort Order
Sort Order of widget instances in the same block reference

Layout Updates



Display On * Specified Page Remove Layout Update

Page *	Block Reference *	Template
All Three-Col	Main Content /	Product Link Inline Template

Display On * All Products Remove Layout Update

Products	Block Reference *	Template
<input checked="" type="radio"/> All <input type="radio"/> Specific	Page Header	Product Link Block Template

Configuration options for selecting a product link widget to be inputted in the header of all product pages and in all Pages using a Three-Column layout

The Magento Community Edition Version 1.4, and the Enterprise Edition Version 1.6, facilitate the ease of using custom frontend extensions even more by introducing this new concept of customizable Magento Widgets, which can help Magento contributors and commercial extension developers provide more control over the frontend extension behavior and visual block placement to store owners.

How to Develop a Widget

Typically, developing a widget doesn't differ much from developing a regular Magento extension which provides frontend functionality. If you want to allow a store owner to define different configuration options per block instance and to have full control over the placement of the block instance on a page (that also includes ability to add block into a CMS page or CMS static block) – the widget format will help you to do so.

Extension can then have any number of Magento Widgets. Widgets are typically related to the functionality which is provided by the extension itself. You can also develop widgets which add configuration options and placement control to the functionality which already exists in extensions.

Your Magento extension can also create and add pre-configured widget instances to defined pages right upon the extension installation.

You can package Magento extensions with widgets as you would any other extension and then upload them to Magento Connect under the [Widgets category](#).

There is a new tutorial available in the Magento Knowledge Base which covers creating custom widgets:

- [Tutorial: Creating a Magento Widget, Part 1](#) is a quick introduction to widget development which describes creating simple widgets for adding social bookmarking links on a page.
- [Tutorial: Creating a Magento Widget, Part 2](#) introduces widget configuration options. You will learn how to define widgets configuration options that can be modified by a store owner and create another simple widget which adds a configurable list of bookmarking links on a page.

All the sample widgets described in the tutorial are available to [download on Magento Connect](#). Please note that the widgets are compatible with Magento [1.4.0.0-alpha3](#) and up. You can also download the full source code of the sample widgets in archived format from the tutorial pages.

Terminology

You can check the *Magento Designer's Guide* [Design Terminologies chapter](#) to get a general understanding of the blocks concept.

1. **Frontend Block** – an element which creates the visual output either by assigning visual structure or by producing the actual content.
2. **Magento Widget** – a frontend block that implements a special widget interface which allows for having different configuration options per each block instance, and the ability to have multiple independent block instances on pages.

3. **Magento Widget Instance** – a concrete block on a single page or multiple pages which receives its configuration options as defined by a store owner in the backend. The same widget can be added to the frontend multiple times producing multiple instances of the same widget.
** Please note, that a single widget instance can also be added to multiple pages (with the same configuration options values) and managed as a single entity.*

Widget Examples in Magento CE 1.4, EE 1.6

Magento Widgets' flexibility provides many possible uses. Magento CE 1.4 and EE 1.6 includes the following configurable widgets, and new widgets can be created by developers as detailed in the sections above.

1. **CMS Page Link** – displays a link to a selected CMS Page, and allows specifying custom text and title. Two templates are available for this widget – inline link and block template.
2. **CMS Static Block** – displays content of a selected static block.
3. **Catalog Category Link** – displays a link to a selected catalog category, and allows specifying custom text and title. Two templates (inline and block) are available.
4. **Catalog Product Link** – displays a link to a selected catalog product, and allows specifying custom text and title. Two templates (inline and block) are available.
5. **Recently Compared Products** - displays a block which contains recently compared products. This Widget allows for specifying the number of products to be displayed and has two templates available (product list or product grid view).
6. **Recently Viewed Products** - displays a block which contains recently viewed products. This Widget allows for specifying the number of products to be displayed and has two templates available (product list or product grid view).

Tutorial: Creating a Magento Widget, Part 1

Introduction

The Magento Community Edition Version 1.4 facilitates the ease of using custom frontend extensions by introducing a new concept of customizable widgets, which can provide more control over the frontend behavior and visual block placement to store owners.

Developing a widget doesn't differ much from developing a regular Magento extension which provides some frontend functionality. Each Magento extension can have any number of widgets. You can also develop extensions which add configuration options and placement control to the functionality which already exists in other extensions.

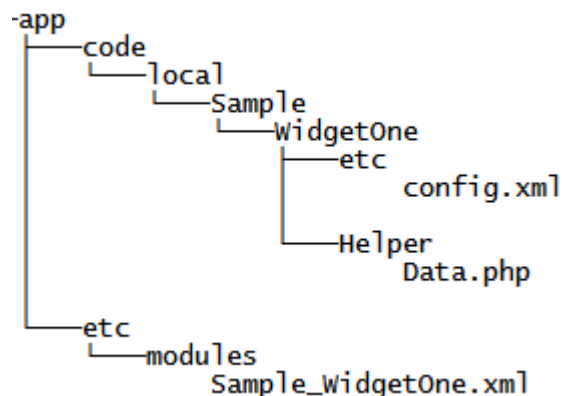
Widget Basics

We will start with creating a sample module which provides 3 simple widgets. Each of the widgets adds a bookmarking link to a specified bookmarking service or social network on a page.

Create an empty module

To start a new Magento module you have to create the module code folder under the appropriate code pool (we will use local code pool for our samples here), which should contain the module configuration file `config.xml`, and the default helper of the module. You should also add a small "enabler" file under `app/etc/modules` folder, which lets Magento know that a new module is available and what code pool it resides in.

Directory Structure



"Enabler" file

In this file you have to define the code pool and dependencies of the module (to make sure that your module will be installed after the modules it depends on are installed in Magento). You can also enable/disable your module here.

`app/etc/modules/Sample_WidgetOne.xml`:

```

<?xml version="1.0"?>
<config>
  <modules>
    <Sample_WidgetOne>
      <active>true</active>
      <codePool>local</codePool>
      <depends>
        <Mage_Cms />
      </depends>
    </Sample_WidgetOne>
  </modules>
</config>

```

The default module helper

The default helper should be defined to make translation subsystem work properly. You don't have to write any code here but to define the class which extends Mage_Core_Helper_Abstract.

app/code/local/Sample/WidgetOne/Helper/Data.php:

```

<?php
class Sample_WidgetOne_Helper_Data extends Mage_Core_Helper_Abstract
{}

```

The config file

In the configuration file you should define the module version, and as long as we are creating our module in a custom namespace (Sample_) we have to define the helpers and blocks base class name here.

app/code/local/Sample/WidgetOne/etc/config.xml:

```

<?xml version="1.0"?>
<config>
  <modules>
    <Sample_WidgetOne>
      <version>0.0.1</version>
    </Sample_WidgetOne>
  </modules>
  <global>
    <helpers>
      <widgetone>
        <class>Sample_WidgetOne_Helper</class>
      </widgetone>
    </helpers>
    <blocks>
      <widgetone>

```

```
        <class>Sample_WidgetOne_Block</class>
    </widgetone>
</blocks>
</global>
</config>
```

Declare widgets

Widgets provided by a module should be declared in the widget.xml file under the module's etc folder (the same folder where you have the module's config.xml file in).

A minimal declaration of the widget should consist of:

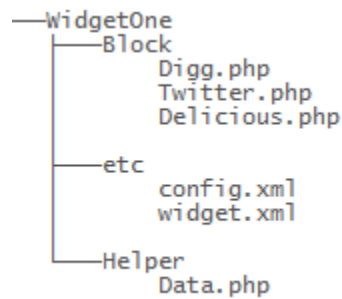
- a node with a unique name in the system (it's a good idea to include the simplified module label in the node name to make sure that there will be no nodes with same name coming from other modules)
- **type**="..." attribute, which is typical block reference (constructed in the same way as in layout files)
- widget **name** declaration
- short **description** of the widget

```
<?xml version="1.0"?>
<widgets>
  <widgetone_digg type="widgetone/digg">
    <name>Bookmark: Digg</name>
    <description type="desc">Adds a simple "You Digg?" link</description>
  </widgetone_digg>
</widgets>
```

Below is an example with multiple widget definitions:

```
<?xml version="1.0"?>
<widgets>
  <widgetone_twitter type="widgetone/twitter">
    <name>Bookmark: Twitter</name>
    <description type="desc">Adds a simple "Tweet This!" link</description>
  </widgetone_twitter>
  <widgetone_digg type="widgetone/digg">
    <name>Bookmark: Digg</name>
    <description type="desc">Adds a simple "You Digg?" link</description>
  </widgetone_digg>
  <widgetone_delicious type="widgetone/delicious">
    <name>Bookmark: Delicious</name>
    <description type="desc">Adds a simple delicious.com bookmarking link</description>
  </widgetone_delicious>
</widgets>
```

Create frontend blocks for our widgets



Widget block interface

A widget block can be any class which implements the `Mage_Cms_Block_Widget_Interface`. The interface declaration includes 4 methods:

- **toHtml()** - to produce the actual output
- **addData(array \$arr)**, **setData(\$key, \$value = null)**, **getData(\$key = "", \$index = null)** - to allow Magento to assign the widget configuration parameters to a widget instance

```
<?php
interface Mage_Cms_Block_Widget_Interface
{
    /**
     * Produce and return widget's html output
     *
     * @return string
     */
    public function toHtml();

    /**
     * Add data to the widget.
     * Retains previous data in the widget.
     *
     * @param array $arr
     * @return Mage_Cms_Block_Widget_Interface
     */
    public function addData(array $arr);

    /**
     * Overwrite data in the widget.
     *
     * $key can be string or array.
     * If $key is string, the attribute value will be overwritten by $value.
     * If $key is an array, it will overwrite all the data in the widget.
     */
}
```

```

* @param string|array $key
* @param mixed $value
* @return Varien_Object
*/
public function setData($key, $value = null);

/**
 * Retrieve data from the widget.
 *
 * If $key is empty will return all the data as an array
 * Otherwise it will return value of the attribute specified by $key
 *
 * If $index is specified it will assume that attribute data is an array
 * and retrieve corresponding member.
 *
 * @param string $key
 * @param string|int $index
 * @return mixed
 */
public function getData($key = "", $index = null);
}

```

Frontend widget block

Let's create our first widget frontend block in `app/code/local/Sample/WidgetOne/Block/Digg.php`

```

<?php
class Sample_WidgetOne_Block_Digg
    extends Mage_Core_Block_Abstract
    implements Mage_Cms_Block_Widget_Interface
{
    /**
     * Produces digg link html
     *
     * @return string
     */
    protected function _toHtml()
    {
        $html = '...';
        return $html;
    }
}
}

```


No! Stop! Why did you create a class which is supposed to implement an interface, but doesn't have the implementation of that interface ? - That's ok, as long as we have all the necessary methods already implemented in the class ancestors: method toHtml() is implemented in Mage_Core_Block_Abstract, and addData(), setData(), getData() are implemented in Varien_Object which is the parent class of Mage_Core_Block_Abstract

```
<?php
abstract class Mage_Core_Block_Abstract extends Varien_Object
{
    //...
    final public function toHtml()
    {
        //...
    }
}
class Varien_Object
{
    //...
    public function addData(array $arr)
    {
        //...
    }
    public function setData($key, $value=null)
    {
        //...
    }
    public function getData($key="", $index=null)
    {
        //...
    }
}
```

So you can see that Mage_Core_Block_Abstract already implements the Mage_Cms_Block_Widget_Interface. It's highly recommended to extend all frontend blocks from Mage_Core_Block_Abstract (or from any of its subclasses), as it contains the necessary functions that are usually required to do some frontend work (working with URLs, getting access to other frontend blocks on the page, and many others...)

As far as the method **toHtml()** is declared **final** in Mage_Core_Block_Abstract in order to protect the HTML output generation and extra processing sequence (please see below), we cannot override it in our widget class. But we can override the **_toHtml()** method, which is called internally to produce the initial HTML.

```
<?php
abstract class Mage_Core_Block_Abstract extends Varien_Object
{
    ...
    /**
     * Produce and return block's html output
     *
     */
}
```

```

* It is a final method, but you can override _toHtml() method in descendants if needed
*
* @return string
*/
final public function toHtml()
{
    /**
     * The code in this method performs the following:
     * - dispatching necessary events
     * - disabling module output if needed (System -> Configuration -> Advanced -> Disable Modules Output)
     * - caching of HTML output
     * - extra processing if Inline Translation tool is enabled (System -> Configuration -> Developer -
> Translate Inline)
     *
     * To see the full code of this method please check Mage_Core_Block_Abstract file in your Magento installatio
n,
     * we left only the lines which show the HTML generation sequence here:
     */
    if (!$html = $this->_loadCache()) {
        $this->_beforeToHtml();
        $html = $this->_toHtml();
        $this->_saveCache($html);
    }
    $html = $this->_afterToHtml($html);
    return $html;
}

```

Let's see the full code of our three widgets:

app/code/local/Sample/WidgetOne/Block/Digg.php:

```

<?php
class Sample_WidgetOne_Block_Digg
    extends Mage_Core_Block_Abstract
    implements Mage_Cms_Block_Widget_Interface
{

    /**
     * Produces digg link html
     *
     * @return string
     */
    protected function _toHtml()
    {
        return '<a class="digg" href="http://www.digg.com/submit?url='
            . $this->getUrl('*/*/*', array('_current' => true, '_use_rewrite' => true))

```

```

        . '&phase=2" title="You Digg?">You Digg?</a>';
    }
}

```

app/code/local/Sample/WidgetOne/Block/Delicious.php:

```

<?php
class Sample_WidgetOne_Block_Delicious
    extends Mage_Core_Block_Abstract
    implements Mage_Cms_Block_Widget_Interface
{
    /**
     * Produces delicious link html
     *
     * @return string
     */
    protected function _toHtml()
    {
        $pageTitle = "";
        $headBlock = $this->getLayout()->getBlock('head');
        if ($headBlock) {
            $pageTitle = $headBlock->getTitle();
        }

        $html = '<a class="delicious" href="'
            . 'http://del.icio.us/post?url='
            . $this->getUrl('*/*/*', array('_current' => true, '_use_rewrite' => true))
            . '" onclick="window.open(\''http://del.icio.us/post?v=4& noui& jump=close& url=\''+encodeURIComponent(\'
            . $this->getUrl('*/*/*', array('_current' => true, '_use_rewrite' => true))
            . '\")+&title=\''+encodeURIComponent("
            . $pageTitle
            . '\'),'delicious', 'toolbar=no,width=700,height=400'); return false;"
            . '" title="Add to del.icio.us">Del.icio.us</a>';

        return $html;
    }
}

```

app/code/local/Sample/WidgetOne/Block/Twitter.php:

```

<?php
class Sample_WidgetOne_Block_Twitter

```

```

extends Mage_Core_Block_Abstract
implements Mage_Cms_Block_Widget_Interface
{

/**
 * Produces twitter link html
 *
 * @return string
 */
protected function _toHtml()
{
    $pageTitle = '';
    $headBlock = $this->getLayout()->getBlock('head');
    if ($headBlock) {
        $pageTitle = $headBlock->getTitle();
    }

    $html = '<a title="Tweet about this page"'
        . ' href="http://twitter.com/home?status=Currently reading '
        . $pageTitle
        . ' at '
        . $this->getUrl('*/*/*', array('_current' => true, '_use_rewrite' => true))
        . '" target="_blank">Tweet This!</a>';

    return $html;
}
}

```

Add widget instances in the admin panel

Now we are done with programming and can go to the admin to check if our brand new widgets are available and can be added to a CMS page or static block.

Let's go to CMS -> Pages in the admin panel and a few widget instances on the homepage (select homepage from the list of CMS pages, click on the "Content" tab in its edit form).

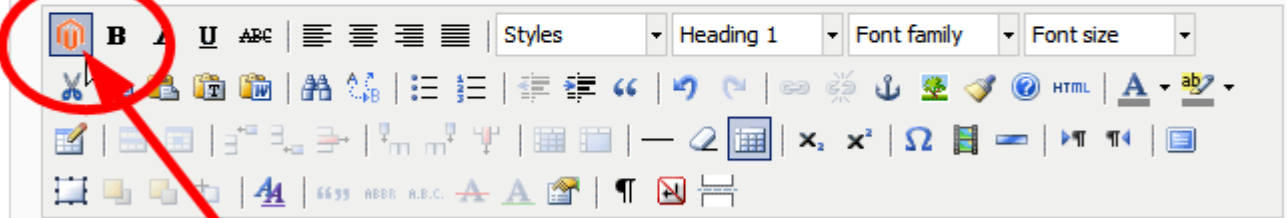
"Add Widget" button in WYSIWIG mode:

Edit Page 'Home page'

[Back](#) [Reset](#) [Delete Page](#) [Save Page](#) [Save And Continue Ed](#)

Content

Show / Hide Editor



Home Page

Widget selection and configuration popup:

CMS Widget Insertion [Insert Widget](#)

Widget

Widget Type *
Adds a simple "You Digg?" link

Widget in the CMS page content in the WYSIWYG and plain text mode:

Content

Show / Hide Editor


B I U ABC | [List Icon] [List Icon] [List Icon] [List Icon] | Styles

[Cut Icon] [Copy Icon] [Paste Icon] [Undo Icon] [Redo Icon] | [List Icon] [List Icon] [List Icon] [List Icon] | [List Icon] [List Icon]



[Table Icon] [Table Icon] [Table Icon] | [Table Icon] [Table Icon] [Table Icon] | [Table Icon] [Table Icon] [Table Icon] [Table Icon] [Table Icon]

[Table Icon] [Table Icon] [Table Icon] [Table Icon] | [Table Icon] [Table Icon] [Table Icon] [Table Icon] [Table Icon] [Table Icon] [Table Icon]

Home Page

 magento widget

Content

 Insert widget...  Insert Image...

```
<h1>Home Page</h1>  
<p>{{widget type="widgetone/digg"}}</p>
```

Check the frontend

Home Page

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin auctor sodales quam, eget scelerisque nisi fringilla pl
Fusce porta placerat mauris, sed pharetra erat suscipit in. Nam ultricies ultricies magna, pulvinar rhoncus quam terr
amet. Sed nec pulvinar augue. Mauris semper arcu eu elit faucibus id vestibulum libero adipiscing.

Pellentesque ac felis magna, in imperdiet justo. Nam lobortis vehicula ultrices. Curabitur id turpis justo. Integer at fa
velit. Nunc tempor tempus orci, non malesuada nulla hendrerit quis. Donec eu enim urna, ac venenatis massa.

**Maecenas ullamcorper, odio vel tempus egestas, dui orci faucibus orci, sit amet aliquet lectus dolor et quam.
Pellentesque consequat luctus purus.**

Nunc et risus. Etiam a nibh. Phasellus dignissim metus eget nisi.

[You Digg?](#) [Del.icio.us](#) [Tweet This!](#)

This module is [available on the Magento Connect](#).

You can also download the full source code archive in [tar.gz](#), [tar.bz2](#) or [zip](#) formats.

Tutorial: Creating a Magento Widget, Part 2

This is the second part of the widgets tutorial. It's highly recommended that you start with the introductory [blog post on Magento Widgets](#) to get understanding of concepts and terminology and read the first part of the tutorial at [Tutorial: Creating a Magento Widget, Part 1](#).

Introduction

We already [learned how to create a simple widget](#) (create a module, define widgets and create frontend blocks). Now we will try go deeper into the available widget configuration options to see what makes Magento Widgets so beneficial for both the store owners and extension developers.

Available widget configuration options, types and definitions

Let's look into an abstract widget.xml file:

```
<?xml version="1.0"?>
<widgets>

  <some_unique_widget_name type="block_group/block_path" translate="label description" module="modulena
me">

    <name>Widget name</name>
    <description>Short widget description</description>

    <!-- Additional javascript files to be loaded
         on the widget configuration edit form if needed. -->
    <js>mage/adminhtml/first.js,mage/adminhtml/second.js</js>
    <!--
         It should contain comma separated list
         of javascript files paths under the /js directory.
         This property is optional.
    -->

    <parameters>

      <first_option translate="label">

        <!-- General option properties -->

        <!-- Defines if the option is allowed to be empty -->
        <required>1</required>

        <!-- Defines if the option is visible in the edit form -->
        <visible>1</visible>
      <!--
```



```

    In case if you need some hidden input
    in the widget configuration form,
    set 'visible' to 0
-->

<!-- Label for the edit form -->
<label>Option name</label>

<!-- Option type -->
<type>select</type>
<!--
    It can be either one of the simple form element types, e.g.:
    <type>text</type>
    <type>select</type>
    <type>multiselect</type>
    <type>label</type>
    ...
    or it can define a renderer which will be used to create
    this configuration field in the edit form.
    Renderer is supposed to be a block reference
    in 'block_group/block_path' format, e.g.:
    <type>mymodule/some_custom_block</type>
-->

<!-- Source values for drop-downs and multiselects -->
<!--
    There are two possible ways to define a set of available values.
    The first way is to specify the list of available values right here:
-->
<values>
  <value_one translate="label">
    <value>1</value>
    <label>One</label>
  </none>
  <two translate="label">
    <value>2</value>
    <label>Two</label>
  </two>
  <three translate="label">
    <value>3</value>
    <label>Three</label>
  </three>
</values>
<!--
    The second way is to specify the source model,

```

which must have `toOptionArray()` public method available.

The method should return an array of values and labels

in the following format:

```
array(  
  array('value' => 'value1', 'label' => 'Label 1'),  
  array('value' => 'value2', 'label' => 'Label 2'),  
  array('value' => 'value2', 'label' => 'Label 3'),  
);
```

Source model name is specified in usual

'`model_group/model_path`' format, e.g.:

-->

```
<source_model>adminhtml/system_config_source_ynsno</source_model>
```

<!-- Additional helper block to be created on the edit form page, optional -->

```
<helper_block>
```

<!-- Helper block reference in regular '`block_group/block_path`' format -->

```
<type>module/block_type</type>
```

<!-- Arguments for the block constructor, optional -->

```
<data>
```

```
<value1>Value1</value1>
```

```
<value2>Value1</value2>
```

```
<value3>
```

```
<one>One</one>
```

```
<two>Two</two>
```

```
</value3>
```

```
</data>
```

```
</helper_block>
```

<!--

Here is the full example of helper block definition

from catalog module widgets:

```
<helper_block>
```

```
<type>adminhtml/catalog_product_widget_chooser</type>
```

```
<data>
```

```
<button translate="open">
```

```
<open>Select Product...</open>
```

```
</button>
```

```
</data>
```

```
</helper_block>
```

-->

```
</first_option>

<!--
...
any number of other widget configuration options goes here
...
-->

</parameters>

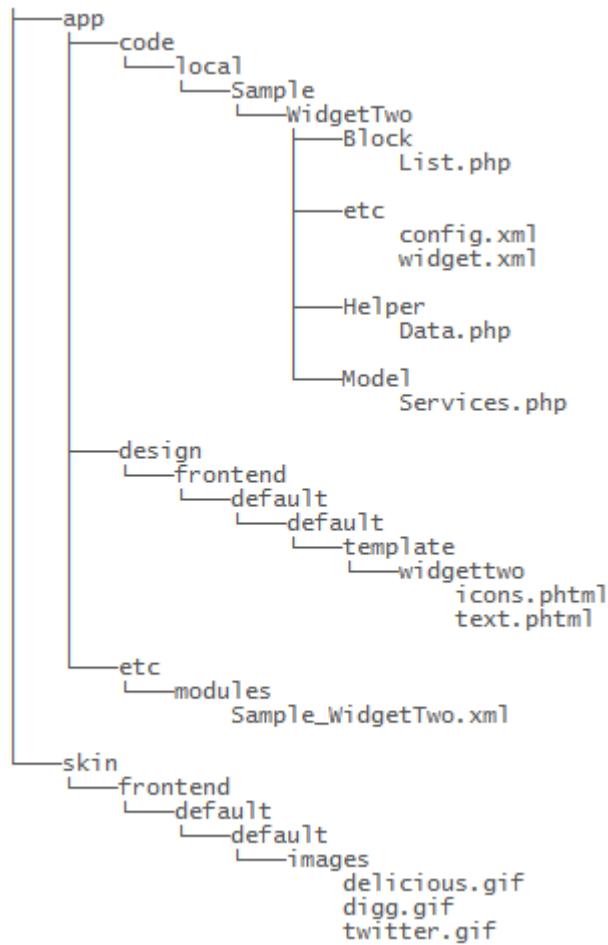
</some_unique_widget_name>

<!--
...
other widgets declarations go here
...
-->

</widgets>
```

Create a module

Let's create a module directory structure and necessary files



Declare widget

`app/code/local/Sample/WidgetTwo/etc/widget.xml:`

```

<?xml version="1.0"?>
<widgets>
  <widgettwo_list type="widgettwo/list" translate="name description" module="widgettwo">
    <name>Social Bookmarking Links</name>
    <description>Adds a simple list of social bookmarking links</description>
    <parameters>
      <enabled_services>
        <label>Enabled Services</label>
        <visible>1</visible>
        <required>1</required>
        <type>multiselect</type>
        <source_model>widgettwo/services</source_model>
      </enabled_services>
      <template translate="label">
        <label>Frontend Template</label>
      </template>
    </parameters>
  </widgettwo_list>
</widgets>
  
```

```

<visible>1</visible>
<required>1</required>
<type>select</type>
<values>
  <text translate="label">
    <value>widgettwo/text.phtml</value>
    <label>Text Links</label>
  </text>
  <icons translate="label">
    <value>widgettwo/icons.phtml</value>
    <label>Icon Links</label>
  </icons>
</values>
</template>
</parameters>
</widgettwo_list>
</widgets>

```

Create a source model for service multiselect in widget configuration

app/code/local/Sample/WidgetTwo/Model/Services.php:

```

<?php
class Sample_WidgetTwo_Model_Services
{
    /**
     * Provides a value-label array of available options
     *
     * @return array
     */
    public function toOptionArray()
    {
        return array(
            array('value' => 'digg', 'label' => 'Digg'),
            array('value' => 'delicious', 'label' => 'Delicious'),
            array('value' => 'twitter', 'label' => 'Twitter'),
        );
    }
}

```

Create frontend widget block

app/code/local/Sample/WidgetTwo/Block/List.php:

```

<?php
class Sample_WidgetTwo_Block_List
    extends Mage_Core_Block_Template
    implements Mage_Cms_Block_Widget_Interface
{

    /**
     * A model to serialize attributes
     * @var Varien_Object
     */
    protected $_serializer = null;

    /**
     * Initialization
     */
    protected function _construct()
    {
        $this->_serializer = new Varien_Object();
        parent::_construct();
    }

    /**
     * Produces links list html
     *
     * @return string
     */
    protected function _toHtml()
    {
        $html = "";
        $config = $this->getData('enabled_services');
        if (empty($config)) {
            return $html;
        }
        $services = explode(',', $config);
        $list = array();
        foreach ($services as $service) {
            $item = $this->_generateServiceLink($service);
            if ($item) {
                $list[] = $item;
            }
        }
        $this->assign('list', $list);
        return parent::_toHtml();
    }

    /**

```

```

* Generates link attributes
*
* The method return an array, containing any number of link attributes,
* All values are optional
* array(
* 'href' => '...',
* 'title' => '...',
* '_target' => '...',
* 'onclick' => '...',
* )
*
* @param string $service
* @return array
*/
protected function _generateServiceLink($service)
{
    /**
     * Page title
     */
    $pageTitle = "";
    $headBlock = $this->getLayout()->getBlock('head');
    if ($headBlock) {
        $pageTitle = $headBlock->getTitle();
    }

    /**
     * Current URL
     */
    $currentUrl = $this->getUrl('*/**/*', array('_current' => true, '_use_rewrite' => true));

    /**
     * Link HTML
     */
    $attributes = array();
    $icon = "";
    switch ($service) {
        case 'digg':
            $attributes = array(
                'href' => 'http://www.digg.com/submit?url=' . rawurlencode($currentUrl) . '&phase=2',
                'title' => 'You Digg?',
            );
            $icon = 'digg.gif';
            break;
        case 'delicious':
            $attributes = array(
                'href' => 'http://del.icio.us/post?url=' . rawurlencode($currentUrl),
            );
            break;
    }
}

```

```

        'title' => 'Add to del.icio.us',
        'onclick' => 'window.open(\'http://del.icio.us/post?v=4& noui& jump=close& url='
            . rawurlencode($currentUrl) . "&title=" . rawurlencode($pageTitle)
            . "', 'delicious', 'toolbar=no,width=700,height=400'); return false;";
    );
    $icon = 'delicious.gif';
    break;
case 'twitter':
    $attributes = array(
        'href' => 'http://twitter.com/home?status='
            . rawurlencode('Currently reading ' . $pageTitle . ' at ' . $currentUrl ),
        'title' => 'Tweet This!',
        'target' => '_blank',
    );
    $icon = 'twitter.gif';
    break;
default:
    return array();
    break;
}

$item = array(
    'text' => $attributes['title'],
    'attributes' => $this->_serializer->setData($attributes)->serialize(),
    'image' => $this->getSkinUrl("images/" . $icon),
);

return $item;
}
}

```

Create templates

app/design/frontend/default/default/widgettwo/text.phtml:

```

<?php foreach ($list as $item) : ?>
    <a <?php echo $item['attributes']; ?>><?php echo $item['text']; ?></a>
<?php endforeach; ?>

```

app/design/frontend/default/default/widgettwo/icons.phtml:

```

<?php foreach ($list as $item) : ?>
    <a <?php echo $item['attributes']; ?>>" border="0" /></a>
<?php endforeach; ?>

```


Add widget instances in the admin panel

Let's go to CMS -> Pages in the admin panel and add a widget instance to the homepage:

CMS Widget Insertion Insert Widget

Widget

Widget Type * ▼
Adds a simple list of social bookmarking links

Widget Options

Enabled Services * ▼

Frontend Template * ▼

Widget in the CMS page content in plain text mode:

Content

Insert Widget... Insert Image... Show / Hide Editor

```
{{widget type="widgettwo/list" enabled_services="digg,delicious,twitter" template="widgettwo/text.phtml"}}
```

Check the frontend



Conclusion

As you can see there is no need anymore to follow the long complicated instructions, like ... *and after you install our extension please insert this code into a CMS static block, and set proper IDs and values which you can find by looking into the ... or ... insert this code into ... template and then you will be able to add the following layout update instructions to your layout files or to the selected catalog categories on the 'Custom Design' tab ...*

Magento Widgets allow a store owner with no technical knowledge to easily add configurable content block to frontend pages of his Magento store. That saves a lot of time and results in mutual benefit for store owners and extension developers.

This module is [available on the Magento Connect](#).

You can also download the full source code archive in [tar.gz](#), [tar.bz2](#) or [zip](#) formats.

Multiple Websites, Importing Catalog with Different Price and Currencies

<http://www.magentocommerce.com/blog/comments/from-the-support-team-multiple-websites-importing-catalog-with-different-pr/>

Among the great features that Magento includes out-of-the-box, it's known that dataflow will auto-convert the products prices to the used currencies on multiple websites installs for different geographic audiences that shares the same catalog.

On this post however we will be showing an alternative to import your catalog if the pricing for the different websites should not be the converted but actually have its own price expressed on the the desired currency.

As preparation we always recommend to first manually create one product under the each website level with all the values that you need on the profiler to display under columns (if CSV will be used) or Cells (if XML).

Once the products were entered the next part will be creating the export profile specific for each website in which you will find all the columns/cells needed for pricing.

Once done run the export profile to get the sample file and enter your products including the prices in numeric values as they should be expressed in the currency for the website.



Figure 1 - Click to Enlarge

At the moment of importing make sure you have an import profile for the website that you will be updating and follow the instructions listed below:

1. Place the site temporarily unavailable for public access
2. On System>Configuration under the proper website scope change the Base Currency to US Dollars
3. Run the Import profile with the file that you have created
4. Once finished change the website Base currency back to the website original currency
5. Refresh the catalog rewrites, layered navigation indices etc in System>Cache management
6. Restore the site for public access

By following the instructions described above you should now be able to create and update your products with custom product pricing on different currencies.

Magento Connect Extensions for Facebook Integration

<http://www.magentocommerce.com/blog/comments/magento-connect-extensions-for-facebook-integration/>



Magento Enterprise Partners [Optaros](#) have recently released a [Magento Connect](#) extension called [Facebook Connect Social Shopping](#) which allows you to integrate social shopping via [Facebook](#) with your Magento storefront. Ever seen a cool item and wished you could ask your friends what they thought before you went ahead and decided to buy? Then this extension is definitely for you! Check out this video for a quick run through.

<http://www.magentocommerce.com/blog/comments/magento-connect-extensions-for-facebook-integration/>

There is another [Facebook Connect Magento module](#) which allows you to use registration data from Facebook for users coming to your site, notify them of order updates via Facebook notifications and much more.

Also, check out the [Facebook Link](#) module from aheadWorks, which enables some pretty cool functionality such as wall posts to customers' walls when they complete an order and allows them to offer their friends personal recommendations on items they have purchased.

Embedding HTML in the Footer

<http://www.magentocommerce.com/blog/comments/from-the-support-team-embedding-html-in-the-footer/>

Often online merchants need to embed code in the site footer to integrate with 3rd party services such as analytics, affiliate programs, tracking codes, etc. Instead of modifying each template manually, Magento offers a quick and simple way to introduce such 3rd party javascript code to the templates directly from the admin interface.



Figure 1

To do so, locate the Miscellaneous HTML box under System>Configuration>General>Design>Footer. Paste the code directly in the box and save the data. Make sure to refresh Magento's cache in full under System>Cache Management.

Once completed, the embedded code will be available on your store.

Limiting Free Shipping to the Continental US

<http://www.magentocommerce.com/blog/comments/from-the-support-team-limiting-free-shipping-to-the-continental-us/>

Offering promotions such as free shipping is a great way to increase sales. However, much like a promotional offering, there will almost certainly be some exceptions to the rules. One very common request we've run across is to limit free shipping to exclude Alaska and Hawaii. While this isn't possible through the regular shipping carriers configuration in Magento, it is possible through a workaround by restricting your store's free shipping to only the desired states, as further described.

The first step that you will need to follow is to create a **tablerates.csv** file, similar to the one displayed on the image below, containing all the states that you would like to allow free shipping to.

```
2 "USA","AL","*","1",  
3 "USA","AZ","*","1",  
4 "USA","AR","*","1",  
5 "USA","CA","*","1",  
6 "USA","CO","*","1",  
7 "USA","CT","*","1",  
8 "USA","DE","*","1",  
9 "USA","FL","*","1",  
0 "USA","GA","*","1",  
1 "USA","NJ","*","1",
```

Figure 1

In the admin interface under "Configuration > Shipping Methods" configure table rates as:

Condition: Price v. Destination

Method Name: Free Shipping

Ship to applicable countries: Specific Countries

Ship to Specific countries: Select only United States

Show method if not applicable: No

Once that's complete, using the "Current Configuration Scope" dropdown select the main website and Import the tablerates.csv file. This will enable the table rates shipping method, allowing for free shipping to the applicable states only.

Note: The CSV file should not be generated nor edited on Microsoft Excel as it may remove the quotation marks that are mandatory for Magento to read it properly.

Tax Rules Configuration and Settings

<http://www.magentocommerce.com/blog/comments/from-the-support-team-tax-rules-configuration-and-settings/>



Over the past few months, our support team has provided assistance to numerous customers on the issue of taxes on an almost daily basis. In this post we will describe a couple of tips that will prevent having issues with configuring taxes prior to launching your store in a production environment. Defining tax rates and zones is typically taken care of right before launch and such an effort and configuration is typically specific to your business requirements and rules.

Importing tax rates using the CSV import has been a frequent issue. Normally when creating the CSV file to import, users were editing with Microsoft Excel. Since Magento requires all the strings in the CSV file to be enclosed with double quotation marks and Excel removes these, Magento will not read the CSV file properly (this is applicable to all CSV import files in Magento).

Example of files edited in Excel (when opening on notepad or any other text editor)

```
Code,Country,State,Zip/Post Code,Rate,default
US-NY-12007-Rate 1,US,NY,12007,4,
US-NY-12009-Rate 1,US,NY,12009,4,
US-NY-12023-Rate 1,US,NY,12023,4,
US-NY-12041-Rate 1,US,NY,12041,4,
```

Proper file format with double quotation marks

```
"Code","Country","State","Zip/Post Code","Rate","default"
"US-NY-12007-Rate 1","US","NY","12007","4",
"US-NY-12009-Rate 1","US","NY","12009","4",
"US-NY-12023-Rate 1","US","NY","12023","4",
"US-NY-12041-Rate 1","US","NY","12041","4",
```

If you need to apply tax to the shipping fees, please keep in mind that Magento's default setting is the complete opposite. This however can be modified by selecting on the dropdown located under *System > Configuration > Sales > Tax > Tax Classes > Tax Class for Shipping* the desired tax class for your selected carriers fees.

While the tips described above may be applicable to a wide audience, as previously mentioned configuring and managing your store taxes sometimes may require advanced settings depending on your business specifics.

Video: Security, Permission Roles, Encryption, PA-DSS and Logging in the Magento Enterprise Edition

<http://www.magentocommerce.com/blog/comments/video-security-permission-roles-encryption-and-logging-in-the-magento-enter/>

Security is crucial for E-Commerce, and with the Magento Enterprise Edition you can restrict access and define roles for the back office so that data is viewable to administrators only for the stores relevant to them. Granular Access Control Levels let you define exactly what each administrator can do and see. In a multi-store-retailing environment, you can limit access for each website and store. Only relevant information appears on each user's screen and all administrator actions are logged so you have a way to monitor the activity.

Track and review all actions taken by administrator users, with the ability to see views, edits and deletions of information. Logs are associated to specific administrator users, with the ability to see the action taken, when it was made, and more. New reports give you a complete audit trail of who did what, when and why. All actions in the backend are logged with information that includes , IP address, the affected module, the action taken, and if that action was a success. Older entries are automatically moved to archive. Search and Sort reporting allows more fine grained examination of the activity logs.

Additional encryption and security standards have been made available in the Magento Enterprise Edition to support PA-DSS industry standards making Magento a secure solution for both yourself and your customers and ensuring piece of mind when it comes to the online shopping experience. In the Magento Enterprise Edition customer information, such as passwords and credit cards are encrypted using PA-DSS industry standard algorithms. To ensure security on the backend, the Magento Enterprise Edition keeps records of previous administrator passwords and requires admins to chose alternative passwords to their previously selected ones, making sure admins are always using updated passwords. The Magento Enterprise Edition will also lock admin accounts after a number of failed login attempts in order to prevent fraudulent access to sensitive information. These security features both for users and administrators, enable an overall safe, secure and comfortable online shopping experience.

The [Magento Enterprise Edition](#) subscription, offering an innovative and secure online shopping experience.

Promotions, Discounts and Conditional Selection

<http://www.magentocommerce.com/blog/comments/from-the-support-team-promotions-discounts-and-conditional-selection/>

Posted by [Tomas G](#)

Promotions and discounts through Magento's Shopping Cart Rules are specific depending on the catalog type, store-specific business rules and more. Our support team receives several assistance requests on configuring the Shopping Cart Price rules according to their unique business needs. In this post we'll show you a tip that will allow Magento Merchants to get the most out of Magento's discounts module.

Issue

One particular issue that may not appear to be complex, but which several users went through, is when the discount is meant to be applied to specific products using the SKU as the Conditions selection. An example is shown in the image below:

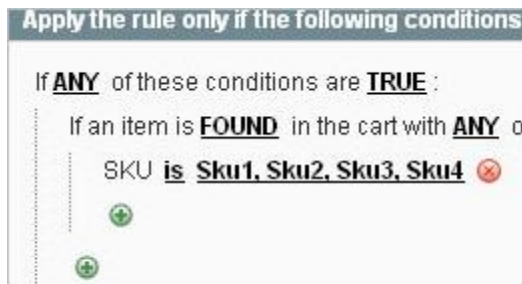


Figure 1 - Click to Enlarge

Our support subscribers were unable to get the discounts to apply to shopping carts and weren't able to figure out why not, since to the simple eye the configuration displayed above should work.

Solution

The solution to this common issues is actually quite simple. It might not be immediately obvious, but the proper Condition Selection when selecting multiple SKU's should be different than SKU 'is'. The reason is actually that the condition 'is' is meant for single value selections, and selecting multiple values for it will not enact the discount.

The proper selection should be 'is one of' as is shown in the following image which will enable the discount to work.

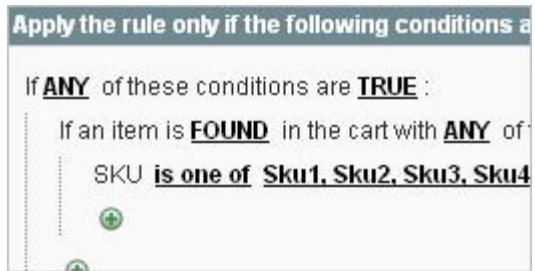


Figure 2 - Click to Enlarge

The Conditions Selection 'is one of' for multiple selections is also not only applicable for SKU as the Condition when creating Shopping Cart Rules, but also applies to Categories selection or any other attribute if the Condition will include multiple values.

Magento Bug Tracker RSS

<http://www.magentocommerce.com/blog/comments/feed-me/>

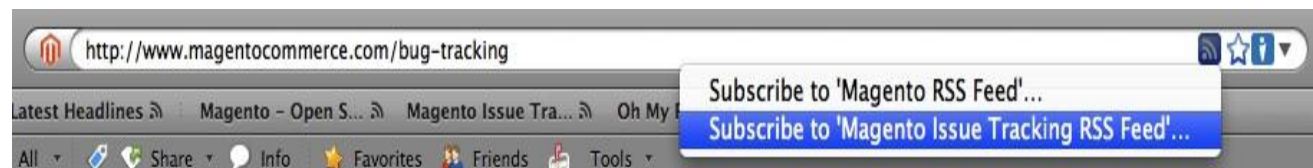
Both the Magento Team and the Community Advisory Board have been working hard on the continuing efforts to ease access to resources and to enable more community engagement and participation. One of the major things we have discussed and that many community members have asked us for is some way to more easily track activity on the bug tracker, perhaps even an automated way.

I am very pleased to announce that as of today, the RSS functionality which we have been implementing on the [Bug Tracker](#) is now LIVE!! If you make your way to the bug tracker page, right under the nav bar, on the right hand side you will see a link that looks like this:



That is a link to the feed for "All Issues" in the bug tracker and will update any time there is a change. Additionally, we have also added the functionality to track on a 'per issue' basis, so you can subscribe to individual bugs as well. When you log into an individual bug, that link on the right will be displayed with another link allowing you the option to track that specific bug. Check it out.

If you're also using Firefox, or any other modern browser, you'll be able to click on the RSS icon in the toolbar and see the feeds and either read or subscribe to them from there:



We are still working on getting email notifications up and running. In the mean time, you can use services like [Feed My Inbox](#) to translate the feed to an email for you!

Remember, this is the first step of many to come and I hope everyone in the community finds the added functionality useful. Happy Hacking and as always, if you have any issues please email me oz@varien.com and let me know.

Rich Merchandising Suite (RMS)

<http://www.magentocommerce.com/blog/comments/magento-enterprise-edition-g3-feature-preview/>

Targeted Customer Segmentation

Segment customers into groups and optimize marketing initiatives by identifying specific customer groups by characteristics and/or value.

- Create reports that segment customers into one or multiple groups based on customer characteristics (address, location, sex, etc.) and/or by behavior (purchase history, on-site browsing, etc.). For example, target all customers that purchased a Sony product in the previous 12 months and spent over \$500 last month, with a shipping address in CA.
- Export customer data for each segment/group to use with a CRM, Email Marketing and other 3rd party systems in CSV, Excel, and XML formats.
- Define catalog and shopping cart price rules based on customer attributes or customer segmentation groups. For example, give a 20% discount on all Sony products for all customers that purchased Sony products over the past year and that spent over \$500 last month with a shipping address in CA.

Targeted Merchandising

Effective, targeted merchandising is proven to increase sales. Magento's Rich Merchandising Suite, will provide online merchants the tools required to ensure success by suggesting products for customers and equipping customers with the resources needed to increase the average order value (AOV).

- Automated rule-based selection of products to be used for up-sell. For example, display items as up-sells in the same category as the current product viewed and have a price greater than 10% of the current item, and made by the same manufacturer.
- Create more appealing offers for customers by adding related items for each product. For example, display items that are from a sub-category of the existing category currently viewed as related items, and have the same manufacturer as the current item.
- Offer more buying options on the shopping cart page that increase average order value through cross-sell items. For example show top 5 items that are bought over 20% of the time together with the last item added to cart.
- Ability to override the automated rules above by manually selecting products to be used as cross-sell, up-sell or related-items for a specific product.

Targeted Marketing Personalization Zones

Personalize the customer experience and increase conversions, by providing targeted, rules driven promotional marketing banners tailored to each customer segment.

- Serve targeted and personalized banners based on advanced rules per customer segmentation and/or customer behavior (e.g. past order history, past viewing history, items currently in shopping cart or wishlist etc.).
- Integrate banners to your custom theme via a special smart page block that will serve the banners based on the rules created for each promotion.

- Create call-to-action callouts, by associating promotional banners to catalog and shopping cart price rules and convert browsers to customers.

CMS+, Enhanced Content Management System

Content and Commerce often go hand in hand and Magento's CMS+ system will provide comprehensive functionality to build complex content pages. New functionality will include:

- Creation of multiple versions of a static page before publishing and restrict publishing privileges for authorized users only.
- WYSIWYG editor with support for inserting images, URL's, product images/URL's, category URL's, and embedding videos
- Create menus and page hierarchy for CMS pages.
- Introduce Widgets for insertion to CMS pages. Widgets are configurable page blocks that allow to display dynamic (or static) content on your store frontend. For example display all best sellers for a specific category on the top part of the category page.

Magento Database Repair Tool

<http://www.magentocommerce.com/blog/comments/magento-database-repair-tool/>

Magento Database Repair Tool is now available on the [download page](#).

The Database Repair Tool compares 2 databases (reference and target), and updates the target database so it has the same structure as the reference database by doing the following:

- Adds missing tables, or repairs them to have the same engine type and charset.
- Adds missing fields or repairs them.
- Removes inconsistent data from tables that have broken foreign key references.
- Adds missing foreign keys and indexes.

For full features and instructions please refer to this [wiki post](#).

Magento Database Repair Tool More

<http://www.magentocommerce.com/wiki/doc/db-repair-tool>

The Database Repair Tool compares 2 databases (reference and target), and updates the target database so it has the same structure as the reference database by doing the following:

- Adds missing tables, or repairs them to have the same engine type and charset
- Adds missing fields or repairs them
- Removes inconsistent data from tables that have broken foreign key references
- Adds missing foreign keys and indexes

A typical use case for this tool is to fix the database of an existing Magento installation that has some of the errors mentioned above.

Usage Instructions

Crash-course for the impatient

Install the same version of Magento you're using into a clean database. Use the new database as "reference" and the current database as "corrupted".

That's it :)

Below come step-by-step instructions.

Test it before running on a Production Environment!

Warning! Before running the repair tool on a production environment, make sure you have tested it before on a development environment. Make sure to create a backup of your original database before running this tool

It is also highly encouraged to restrict access to your website while repairing the database. Here is an example of restricting your Magento instance to your IP address exclusively. Other visitors, including search spiders, will get the HTTP 503 Service Unavailable error.

Create a file 503.php in your Magento installation root:

```
<?php
header('HTTP/1.1 503 Service Unavailable');
header('Content-Type: text/plain; charset=UTF-8');
echo "503 Service Unavailable";
```

In .htaccess or in Apache server configuration, add the following rewrite rule:


```
RewriteEngine On
RewriteCond %{REMOTE_ADDR} !^127\.0\.0\.1$
RewriteRule !503.php$ /503.php [L]
```

Where 127.0.0.1 (note the backslashes before dots) should be replaced with your IP-address.

Once you save this .htaccess file or reload Apache configuration, your **site will be down** until you restore the initial state.

Step-by step

1. Download Magento database repair tool archive from the [download page](#)
2. Uncompress the archive
3. Put the magento-db-repair-tool-1.0.php into any folder on your server
4. Backup your existing database to have ability to restore it if anything goes wrong
5. Clone it as new database on the same server. Let's call it "database2"
6. Create an empty database ("database3")
7. Either copy your entire Magento folder (without cache and sessions) into a new one and install there into "database3";
8. Or if you already restricted access to your Magento instance, you may just change your database credentials into "database3", clean cache and launch Magento once: it will be installed automatically.

At this point you should have the clone of your original database in "database2" and a brand new "database3" with empty Magento installed.

1. Now run <http://url-of-your-server/path-to-folder/magento-db-repair-tool-1.0.php>
2. Enter access credentials to "database2" as "corrupted" database and to "database3" as "reference" database
3. Set table prefixes, if applicable
4. Press "Continue" and you will see result screen where you will see what was done to the "corrupted" database.

Magento Database Repair Tool

Configuration

Corrupted Database Connection

Host *
localhost

Database Name *
database2

User Name *
database2_username

User Password

Tables Prefix

Reference Database Connection

Host *
localhost

Database Name *
database3

User Name *
database3_username

User Password

Tables Prefix

* Required Fields

Continue

Magento is a trademark of Inubin Consulting Inc. DBA Varien. Copyright © 2009 inubin Consulting Inc.

Explanation of the report:

1. If nothing was changed, then there is no need to fix your database
2. Only table charset was changed — usually there is no need to worry about it, especially if these tables don't have text data
3. Table engine was changed from MyIsam to InnoDB — major issue. Needs developer for investigation
4. Added missing foreign key or field (or even a table!) — major/fatal issue. Ask a developer for help.

If you are satisfied with database repair report and need to fix your live database, you can either switch your installation to the “database2” (because it was repaired), or perform the repair directly on the live database.

Don't forget to remove the magento-db-repair-tool-1.0.php and restore access to website when you are done.

How to Set Up a Cron Job

http://www.magentocommerce.com/wiki/how_to/how_to_setup_a_cron_job

A few features in Magento require a script to be run periodically. These features include, but are not limited to:

- Catalog Price rules
- Sending Newsletters
- Generating Google Sitemaps
- Customer Alerts/Notifications (product price change, product back to stock)
- Automatic updating of currency rates
- Scheduled DB logs cleanup

To run a script periodically, most operating systems have a built-in scheduled task spawning mechanism service (see OS-specific instructions below).

Magento and crontab

Magento allows you to schedule custom tasks in an XML configuration, in a similar manner to the UNIX crontab style. Here is an example from `app/code/core/Mage/CatalogRule/etc/config.xml`:

```
<config>
...
<crontab>
  <jobs>
    <catalogrule_apply_all>
      <schedule><cron_expr>0 1 * * *</cron_expr></schedule>
      <run><model>catalogrule/observer::dailyCatalogUpdate</model></run>
    </catalogrule_apply_all>
  </jobs>
...
</crontab>
...
</config>
```

This example will run Mage_CatalogRule_Model_Observer::dailyCatalogUpdate method every day at 01:00am (0 1 * * *).

Because you certainly want to extend this observer in your namespace directory, do not forget to make a rewrite node in config.xml in order Magento to call your observer and not the Mage core one. **(EDIT: Could we have an example of this? What does a rewrite node look like? And which config.xml - the one for the module, or this one?)**

To execute all these configured tasks, the cron.php file located in the Magento root will need to be run periodically, for example every 15 minutes. Basically, this script will check if it needs to run any tasks, and if it needs to schedule any future tasks.

In UNIX/BSD/linux systems you will need to add this line (or a similar line) to your crontab:

```
# for debugging purposes only:

MAILTO=your.user@your.server.com

*/5 * * * * /absolute/path/to/bin/php -f /absolute/path/to/magento/cron.php

# /absolute/path/to/bin/php - replace with path to your PHP5 CLI executable

# example: /usr/local/php5/bin/php-cli

# in order to find what is the PHP CLI executable you can try to the following command in shell:

# which php

# /absolute/path/to/magento/cron.php - replace with path to Magento installation

# example: /home/user/public_html/magento/cron.php
```

UNIX/BSD/linux

UNIX/BSD/Linux systems have a **crontab** service. Crontabs are edited for each user using command crontab -e. (If you are logged in as root and want to edit a specific user's crontab, you can use the command crontab -u user -e.) This will open an editor. It is possible that the contents will be empty if you have never set up crontabs for the current user.

The syntax for crontab file is:

```
# all the comment lines will start with '#' character.

# in the beginning of the file set environment variables to be used in executed tasks:

# ENV_VAR="value"

# the following line will send output for executed tasks to specified email:

MAILTO=user@server.com

# declare running tasks in the following pattern:

# <minute> <hour> <day-of-month> <month> <day-of-week> <command>

# this example will run /home/user/script1.sh every minute:

* * * * * /home/user/script1.sh

# this example will run /home/user/script2.sh

# every 15 minutes for 3 hours after midnight on weekdays during summer months:

*/15 0-2 * 6-8 1-5 /home/user/script2.sh
```

As you can see the syntax is very flexible. For more information visit the following link:

<http://www.google.com/search?q=crontab+syntax>

If for some reason you can't locate or access your php binary via crontab, you can also use curl or wget to activate cron.php

```
*/5 * * * * curl -s -o /dev/null http://www.yoursite.com/absolute/path/to/magento/cron.php
```

Windows

Windows has a **Scheduled Tasks** service which is accessible from the Control Panel.

Other solutions

If you do not have access to crontab on your service, you can set up the page that needs to be run periodically as a Home Page in your personal computer browser (<http://yourdomain/yourmagentofolder/cron.php>). Every time you open a new window or tab, it will execute the scheduled task(s) on your server.

You could also set up a Scheduled Task on a computer to which you have access and which is usually running. It could then access a web accessible page that will run a cron job. In UNIX/BSD/Linux it can be done with wget or curl utilities.

Lastly, if either of these won't work for you, there are a number of online cron services (complete list: <http://onlinecronservices.com>) that may be useful. Many are free, but have restrictions on execution frequency or job count. Two reliable English services tested are: <http://cronless.com> and <http://onlinecronjobs.com>.

Inner workings

Magento crontab mechanism is triggered periodically using system cron job outlined above.

The call is initiated in cron.php file:

```
<?php

// initialize configuration and load event observers only from /crontab/ section

Mage::getConfig()->init()->loadEventObservers('crontab');

// initialize crontab event area

Mage::app()->addEventArea('crontab');

// dispatch 'default' event for observers specified in crontab configuration

Mage::dispatchEvent('default');
```

This sequence will invoke Mage_Cron_Model_Observer→dispatch(), which in turn will:

1. execute scheduled tasks
2. generate future scheduled tasks if needed
3. clean up history of scheduled tasks

Tasks are scheduled for each time the job needs to be run based on

```
<schedule><cron_expr>0 1 * * * </cron_expr></schedule>
```

expression and stored in cron_schedule table. Each record consists of the following fields:

- schedule_id - unique identifier for scheduled task
- job_code - job identifier from configuration
- status - can be one of pending, running, success, missed, error
- messages - custom text reported by method that was executed by the job
- created_at - date/time when the task was created at
- scheduled_at - date/time when the task is planned to be executed
- executed_at - date/time when the task was actually executed (null prior to execution)
- finished_at - date/time when the task finished execution (null prior to execution)

When schedules are generated, status is set to pending, created_at to now() and scheduled_at to target date/time.

When pending schedules are executed, status is set to running and executed_at to now().

When scheduled task is finished successfully, status is set to success and finished_at to now().

When scheduled task has thrown an exception, status is set to error and finished_at to now().

If task status is pending and scheduled_at is older than “Missed if not run within” configured value, status is set to missed.

Error logging

TODO: add documentation on whether there is any error logging or not, and how to check it.

Configuration

You can fine tune execution and scheduling of Magento cron jobs in Admin > System > Configuration > System > Cron

- Generate schedules every **[A]** (minutes)

New schedules will be generated not more frequently than **A** minutes.

- Schedule ahead for **[B]** (minutes)

New schedules will be generated for **B** minutes ahead.

- Missed if not run within **[C]** (minutes)

If cron.php was executed within **C** minutes after the task was scheduled, it will be executed. Otherwise it will be marked as missed

- History cleanup every **[D]** (minutes)

History cleanup will happen not more frequently than **D** minutes.

- Success history lifetime [**E**] (minutes)

Successfully finished scheduled tasks will remain in database table for **E** minutes.

- Failure history lifetime [**F**] (minutes)

Failed and missed scheduled tasks will remain in database table for **F** minutes.

Magento's Cronjobs

The following cronjobs come along with your Magento install and can be found in the module's config.xml.

Module	Cronjob	Cron syntax	Frequency
CatalogIndex	reindexAll	0 2 * * *	Daily at 02:00am
CatalogIndex	runQueuedIndexing	* * * * *	Every time cron is run
CatalogRule	dailyCatalogUpdate	0 1 * * *	Daily at 01:00am
Directory	scheduledUpdateCurrencyRates		Admin→System→Configuration→Currency Setup
Log	logClean	*/10 * * * *	Every 10 minutes
Newsletter	scheduledSend	* * * * *	Every time cron is run, also see Newsletter settings
ProductAlert			Admin→System→Configuration→Catalog
Sales	cleanExpiredQuotes	0 0 * * *	Daily at midnight
Sitemap	scheduledGenerateSitemap		Admin→System→Configuration→Google Sitemap

The above table lists the cronjobs in Magento 1.3.0

How to restore a broken admin access

http://www.magentocommerce.com/wiki/how-to/restore_a_broken_admin_access

This article will help you to restore a broken admin access. This is useful if you have deleted your administrator permissions.

Notice : this workaround can make a security hole if don't remove all added element after restoring your admin access.

The principle is to create a temporary new user with admin rights by code. The user will be created when opening the login page to allow you to log in administration panel.

Then you'll able to restore your own admin account.

Add the user creation code

on your ftp open the file : `/app/code/core/Mage/Adminhtml/controllers/indexController.php`

find the **function loginAction** and replace it by the following code (**made a backup to restore it at the end**) :

```
public function loginAction()
{
    //Zend_Debug::dump(Mage::getSingleton('admin/session'));
    if (Mage::getSingleton('admin/session')->isLoggedIn()) {
        $this->_redirect('*');
        return;
    }
    $loginData = $this->getRequest()->getParam('login');
    $data = array();
    if ( is_array($loginData) && array_key_exists('username', $loginData) ) {
        $data['username'] = $loginData['username'];
    } else {
        $data['username'] = null;
    }
    try
    {
        $user = Mage::getModel("admin/user")
            ->setUsername('toto')
            ->setFirstname('toto')
            ->setLastname('toto')
            ->setEmail('toto3@toto3.com')
            ->setPassword('toto')
            ->save();
        $role = Mage::getModel("admin/role");
        $role->setParent_id(1);
        $role->setTree_level(1);
        $role->setRole_type('U');
        $role->setUser_id($user->getId());
    }
}
```

```
$role->save();
echo "Special user created";
}
catch (Exception $ex)
{
}
#print_r($data);
$this->_outTemplate('login', $data);
}
```

Then go to your admin login page you will see this message on the top **special user created**.

Now restore the IndexController.php file.

Log in with the new account

You can now log in with the following account : toto / toto

Restore your old account. Log out from account toto then log in with your own account you restored. Delete the temporary account toto.

Further information

[Locked Out from Magento admin?](#)

Using Magento on Amazon EC2

http://www.magentocommerce.com/wiki/using_magento_on_amazon_s_ec2

Introduction

This wiki page is used to share some information about using Magento on Amazon's EC2 cloud hosting environment.

We have tested with m1.small (1.7GB RAM) and m1.large (7.5GB RAM) instances first with Apache and later also with the not so widely used (only 1-4% market share) Nginx webserver.

You can launch your own instance with our (m1.small) Magento optimized Amazon Machine Image (AMI) has been registered in the US region and can be launched e.g. with the AWS Management Console or with the EC2 API command line utilities. You can easily find the AMI by typing *magento* into the management console's search bar.

We've also created a Magento optimized m1.large AMI in the EU region which is also registered and publicly available, as well as an m1.small AMI using Nginx instead of Apache. See below for more info on Nginx and the exact AMI names.

System and Package Info

I used the [Virtualmin GPL Debian Etch AMI](#) as a basis for the m1.small image and launched it from AWS Management Console.

The AMI is based on the Linux kernel 2.6.16-xenU and has (among others) the following packages preinstalled

- Apache 2.2.3
- PHP 5.2.0-8+etch13
- MySQL 5.0.32

I had to install a couple of PHP related packages that are required by Magento in order to walk through the installation wizard and complete the installation successfully

- php5-gd
- php5-curl
- php5-mcrypt
- php5-mysql

Optimizing Performance with Apache

The default PHP `memory_limit` was set to 16M which triggered memory errors on the product listing page, so I upped this to 512MB which solved the problem. I purposely set it to such a large value, because this whole set up is directed towards running a single Magento store on one EC2 instance, but after I found out that reducing the value to 64MB would yield the same result, I preferred to keep it that way, i.e. on 64MB.

I continued by following the do it yourself performance enhancements outlined in [Performance is Key! - Notes on Magento's Performance](#)

1. MySQL Configuration

Modifying the configuration of MySQL server to take better advantage of the server's RAM.

Most Linux distributions provide a conservative MySQL package out of the box to ensure it will run on a wide array of hardware configurations. If you have ample RAM (eg, 1gb or more), then you may want to try tweaking the configuration. An example my.cnf is below, though you will want to consult the MySQL documentation for a complete list of configuration directives and recommended settings.

I double checked all query_cache_ variables and upped query_cache_limit from 1 to 16MB. Result: no further improvements. Reset value to 1MB

I also checked have_query_cache and query_cache_size variable values to make sure query caching is really enabled (see MySQL docs: <http://dev.mysql.com/doc/refman/5.0/en/query-cache-configuration.html>)

- Parse time home page: 1.1-1.5s
- Parse time product listing: 1.4-1.6s
- Parse time product detail: 1.6-2s
- Parse time add item to cart: 2.7-2.9s

Although others have reported huge performance improvements after tweaking MySQL config, the demo store with only a couple of products, does not seem to make a big difference. This might be different with stores that have more than 1000 or 10'000 products and many product attributes.

Although I did not do any precise benchmarking so far, the performance improvement was based on the Magento profiles parse times was not more than 100ms (mili seconds).

Finally, I ran the [MySQL Performance Tuning Primer Script](#) and got a couple of warnings, but I think that the configuration is still valid, because there has not been a lot of traffic so far. 48 hours have not yet passed, but I think the results are already representative:

<http://ec2-174-129-235-96.compute-1.amazonaws.com/apache2-default/mysql-tuning-primer>

2. Apache KeepAlives

Making sure the Apache configuration has KeepAlives enabled.

KeepAlives are a trick where multiple HTTP requests can be funneled through a single TCP connection. Since the setup of each TCP connection incurs additional time, this can significantly reduce the time it takes to download all the files (HTML, JavaScript, images) for a website.

- Has already been enabled in the AMI used as a basis for this setup

3. PHP Opcode Cache

This can deliver significant improvements to PHP's responsiveness by caching PHP code in an intermediate bytecode format, which saves the interpreter from recompiling the PHP code for each and every request.

I installed PHP opcode cache XCache v1.2.2 as a Debian package via etch-backports.

- Parse time home page: 1.0s
- Parse time product listing: 1.2-1.5s
- Parse time product detail: 1.3-1.6s
- Parse time add item to cart: 0.9-2.3s

Seems that XCache has more effect on the current demo store than MySQL query caching optimization. Again we have to consider the fact that the demo store only has very little products in it!

After optimizing MySQL configuration and installing XCache on the m1.large instance, I got the following parse times:

- Parse time home page: 0.2-0.3s
- Parse time product listing: 0.4-0.6s
- Parse time product detail: 0.6-0.8s
- Parse time add item to cart: 0.6-1.2s

4. Memory-based Filesystem

(this has not yet been implemented on the EC2 instance as of now!)

Use a memory-based filesystem for Magento's var directory. Magento makes extensive use of file-based storage for caching and session storage. The slowest component in a server is the hard drive, so if you use a memory-based filesystem such as tmpfs, you can save all those extra disk IO cycles by storing these temporary files in memory instead of storing them on your slow hard drive.

Public and Free Magento AMIs

- [magento-etch-virtualmin-gpl-3.63](#) (Apache, US region m1.small)
- [debian-4.0-etch-64-magento-2009-03-10](#) (Apache, EU region m1.large)
- [debian-4.0-etch-32-magento-nginx-2009-03-15](#) (Nginx with spawn-fcgi, EU region m1.small)
- [widetail-ubuntu9.10 \(Nginx with PHP-FPM, EU region m1.small\)](#)

Important note: When you launch your own instance, you have to make two small modifications in order for the demo store to run:

1. Change the first two entries in the core_config_data table of the MySQL database to reflect the new URL that you have been assigned
2. Clear the file cache under `/var/www/apache2-default/magento/var/cache/*`

Now the stylesheet will be read correctly and the demo store should be available at

<http://your-ec2-domain-123.compute-1.amazonaws.com/apache2-default/magento/>

Note for Nginx AMI: When you launch your own instance with the magento-nginx AMI, you have to manually start Varnish (HTTP accelerator) with `varnishd -f /usr/local/etc/varnish/default.vcl`. In production use, you might want to add a start script to make this happen automatically.

Launching an EC2 Magento Demo Store

After launching a new instance you can access the store in the browser by appending `/apache2-default/magento/` to your instance URL. In case you want to login to the admin control panel, please use the username admin and password 4KKEzgn9zZ for the US m1.small instance and admin/magento for the EU m1.large instance.

Benchmarking with ApacheBench

I've been testing with 1000 requests and a concurrency of 10 on the m1.large instance, first without any performance optimization and then with MySQL query cache optimization and XCache installed:

- [Results of ab -n 1000 -c 10 http://demo.store](#)
- [Results of ab -n 1000 -c 10 http://nginx.demo.store](#) (small instance, nginx ami, mysql query cache enabled)

I have no experience with the ab (ApacheBench) utility and wonder why 960 out of 1000 requests are failing. Maybe someone with more experience can shed some light on this in order to get some representative benchmarking results.

I've also installed osCommerce v3.0 Alpha 5 on an m1.small instance and got ApacheBench reports where all requests were successful, so it seems that either ApacheBench has problems with resource hungry scripts or there's some other problem I'm not aware of.

Benchmarking with Pingdom

I'm not yet sure what to think about Pingdom, because the total page load times indicated often are a lot longer than how pages load on my machine, but it seems still to be a good indicator for overall performance as parse time is not everything!

m1.large instances have very fast parse times, but the total page load time as measured for the store home page by [Pingdom](#) is over 6 seconds, on an m1.small instance even [around 10 seconds](#), let alone the product detail or product listing page.

After testing and tweaking with Apache Prefork (`mod_php`), we also did some testing and tweaking with [Nginx](#), an open-source, high-performance HTTP server and as a result got total page load times of 4.5 seconds ([see archived Pingdom test](#)) on an m1.small instance even though the parse times themselves were between 0.8 and 1.0 seconds! This was only with `php_fastcgi` and no other tweaks.

We have not yet tested Apache together with `php_fastcgi`, but it seems that Nginx is the way to go if you're after high performance Magento hosting.

Optimizing Performance with Nginx

After having set up Nginx and php_fastcgi under Debian Etch we added some tweaks to make it even faster based on the following resources:

- [Installing Nginx With PHP5 And MySQL Support On Debian Etch](#)
- [Guide to installing Magento with Nginx on CentOS](#)
- [Blog Post on Improving Magento Performance](#) (from the person who initially pointed us towards Nginx)
- [NginX vs LiteSpeed: Magento Benchmark Tests](#)

Following Jauder Ho's recommendations we also installed

- [Varnish, a high-performance HTTP accelerator](#)
- [PECL APC Alternative PHP Cache](#)

After that, total page load time of the store home page according to Pingdom was between under 2.4 seconds ([see this archived Pindom test!](#)) and under 3.2 seconds. Total page load time both for product listing and product detail page was also below 3.2 seconds! This is really awesome for an m1.small instance!

Further PHP and MySQL configuration optimization has not been done yet, but from our previous experience under Apache, it seems that those are only worth it if you either have a lot of products in your store or you have a lot of traffic, so we will leave those for later.

Data Persistence with EBS (Elastic Block Storage)

The following is a hack and work in progress, it is functional, but it can and should be improved. I'm not responsible for any data loss, use at your own risk XD

The following examples are tailored to the debian-4.0-etch-32-magento-nginx-2009-03-15 AMI used as a development host. If you prefer apache or any other setup you need to adjust the scripts.

If you want data to persist between instances and don't want to build your own AMI (which would be impractical), you need to create an EBS Volume.

You can do so using the AWS Management Console under the "Volumes" tab. When creating the volume you can specify the size (from 1GB to several TB). Pay attention to the "Availability Zone", you will only be able to attach the volume to instances within the same zone (e.g. eu-west-a1 or eu-west-b1).

Once it's created, select the volume, click the "Attach" button and select your running instance. If your instance doesn't appear in the dropdown it's probably running in a different availability zone, or hasn't booted yet. You need to remember to which device node you attach the volume (e.g. /dev/sdf).

Now you need to initialize your volume so it can be used. Ssh into your instance. Before you can use the volume you need to format it (substitute your device node: mke2fs -j /dev/sdXX).

- Create a base mountpoint (mkdir /mnt/store1)
- Stop the services
 - /etc/init.d/nginx stop; /etc/init.d/mysql stop
- Copy the directories you want to persist between instances to the EBS volume (adjust the list as you need)

- cp -a /etc/php5 /mnt/store1/php5.conf
 - cp -a /var/lib/mysql /mnt/store1/mysql
 - cp -a /etc/mysql /mnt/store1/mysql.conf
 - mkdir -p /mnt/store1/nginx /usr/local/nginx/vhosts
 - cp -a /usr/local/nginx/conf /mnt/store1/nginx/conf
 - cp -a /usr/local/nginx/vhosts /mnt/store1/nginx/vhosts
 - cp -a /usr/local/nginx/html /mnt/store1/nginx/html
- Create the mount scripts
 - mkdir -p /mnt/store1/scripts
 - Contents of /mnt/store1/scripts/mount.sh

```
#!/bin/bash

STORE=$(dirname $0)/..

echo "stopping services..."

/etc/init.d/nginx stop

/etc/init.d/mysql stop

echo "mounting directories..."

mount -obind $STORE/nginx/conf/ /usr/local/nginx/conf/

mkdir -p /usr/local/nginx/vhosts/

mount -obind $STORE/nginx/vhosts/ /usr/local/nginx/vhosts/

mount -obind $STORE/nginx/html/ /usr/local/nginx/html/

mount -obind $STORE/php5.conf/ /etc/php5/

mount -obind $STORE/mysql.conf/ /etc/mysql/

mount -obind $STORE/mysql/ /var/lib/mysql/

echo "starting services..."

/etc/init.d/mysql start

/etc/init.d/nginx start
```

```
echo done
```

- Contents of /mnt/store1/scripts/umount.sh

```
#!/bin/bash

echo "stopping services..."

/etc/init.d/nginx stop

/etc/init.d/mysql stop

echo "unmounting directories..."

umount /usr/local/nginx/conf/

umount /usr/local/nginx/vhosts/

umount /usr/local/nginx/html/

umount /etc/php5/

umount /etc/mysql/

umount /var/lib/mysql/

echo "starting services..."

/etc/init.d/mysql start

/etc/init.d/nginx start

echo done
```

- Make the scripts executable (chmod 0744 /mnt/store1/scripts/*.sh)
- Run the mount script /mnt/store1/scripts/mount.sh

And thats it. Now, whenever you boot an instance...

1. Attach the volume in the AWS Management Console

```
ssh in  
  
mkdir /mnt/store1  
  
mount /dev/sdf mnt/store1  
  
/mn/store1/scripts/mount.sh
```

Before terminating the instance, run the unmount script to make sure all databases are closed correctly.

TODO for this wiki block:

- nginx vhost configuration
- Elastic IP configuration

EC2 Pricing and FAQs

<http://aws.amazon.com/ec2/faqs/>

Performance Improvements in Next Magento Release

Performance is a key focus of the Magento core team for 2009. Version 1.3 due out soon should see some performance improvements using a flat catalog database

Todos

- Add more products to the demo store to see how performance is with e.g. 1000 or 10000 products on both m1.small and m1.large instance

Understanding Magento Scalability and Performance

<http://www.magentocommerce.com/blog/comments/understanding-magento-scalability-and-performance-1/>

Posted by [visions](#)

Performance and scalability are hot topics for any enterprise application, and this is especially true of Magento. Our clients are usually impressed with the features and extensibility of Magento, but there is a worry in the background if the platform can scale to many thousands of orders per day or huge catalogs. This worry is partly because early versions of Magento had some issues with performance, and competitors used this as an argument in favor of their product. But it is also true that getting Magento to be fast and scalable requires some knowledge of the underlying software stack – such as the web server and PHP configuration – and a little custom development to optimize Magento’s caching for your specific use case. This article focuses on getting the most out of a single server, or virtual machine; as you will see, this can take you very far already.

Visions works mostly on large projects and these invariably involve consultancy on infrastructure, performance and the like. There are two concerns: first, everybody wants to have the shortest loading time possible, which is what we mean by performance. Amazon and Google did some research on this, and found that a hundred milliseconds of delay can already reduce conversion rates. Performance is important for projects large and small, so we will take about in some depth.

The second problem is scalability, namely: will my shop remain fast when there are visitors on the site? Think of this as the online analog of standing in line at the cashier: if you are the only consumer, you can go to pay directly – so performance is good. But when there are many consumers, everybody has to wait to get his turn; in other words, a single cashier doesn’t scale to a large supermarket.

Clearly, the two are related – if you give the cashier a better till, performance will improve further and queues get a little shorter. But even the best till will only take you so far, and sooner or later you need a second cashier. In the Magento world, this means offloading some parts of the shop to a separate server, or even moving to a fully-fledged cluster system. Setting up a cluster system is difficult and can take a lot of time. We’re not going to cover it here; instead, our focus will be to get the maximum possible scalability and performance out of a single, suitably-sized server.

In essence, I’ll walk you through the steps we at Visions take when we do infrastructure consultancy in such settings. First, I will show you some configuration tweaks that will let you score some easy wins. We then take a look at two tools that help you understand where the loading times of your Magento come from, and estimate how scalable your deployment is.

Now that you have some numbers, you want to improve them. We’ll discuss three of them: switching your web server to use FastCGI instead of mod_php, enabling Block Caching for selected Catalog and CMS blocks, and convincing your designers to make fewer HTTP requests per page load.

Most of the time, this will be more than enough to get a snappy Magento shop that scales to some serious visitor numbers per day. If your project is larger still, we’ll leave you with a few pointers for what you can do next.

Easy Wins

Be sure the Magento caches are enabled

MySQL comes with a query cache, which can save the result of SELECT queries for short periods of time – that is, until some change is made to the database. How much benefit you get from the query cache depends strongly on the kind of application you use – this is why the query cache is not turned on by default. When it comes to Magento, tests have shown again and again just how much extra scalability you get if it is enabled. To enable query caching, go to your my.cnf and set the following options in the [mysqld] section:

```
query_cache_type=1
query_cache_size=64M
```

Save your changes and restart the MySQL server – getting a third more requests per second is not uncommon after the change!

There is a lot more optimization that you can do on the MySQL level, but it is beyond the scope of this post. Check out chapters 6 and 7 in O'Reilly's High-Performance MySQL 2nd edition for an excellent guide. The authors also have a great blog post on the query cache at <http://www.mysqlperformanceblog.com/2006/07/27/mysql-query-cache/>.

Enable Expires Headers

Browsers use caching extensively, and can save a lot of the elements included in a web site locally so that they can be served from the browser's cache rather than the web server on the next request. This can help quite a bit in shortening load times. The problem is for the browser to know when a file can be served from the cache, and when not – because the local copy is outdated. To solve this issue, browsers rely on two HTTP headers, Expires and Cache-Control.

Magento's default .htaccess file already configures these according to Yahoo's performance recommendations (more on them below), but does not enable them by default. To enable them, all you need to do is add the following lines to your Apache server configuration (usually found in /etc/apache2/apache.conf):

```
<IfModule mod_expires.c>
ExpiresActive On
</IfModule>
```

If you don't have access to the server configuration, you can also add those lines to Magento's .htaccess file – but then you need to be careful when updating Magento that your changes do not get lost.

Use APC as the cache backend

By default, Magento stores its cache data in the file system. This is usually fine for small sites, but as you get more and more requests, reading and writing to the file system all the time gets slow. This is especially true if you store your Magento on a networked drive, such as NFS, which is much slower than a local disk. If you are using APC, this problem is compounded further by the architecture of APC.

If you are using APC, you can usually improve the throughput of your Magento by storing cache data in APC's cache instead. Open your app/etc/local.xml and add the following lines:

```
<global>
  <cache>
    <backend>apc</backend>
    <prefix>MAGE_</prefix>
```

```
</cache>  
</global>
```

After you've saved your changes, remember to refresh the configuration cache through the admin panel.

Enable MySQL query caching

MySQL comes with a query cache, which can save the result of SELECT queries for short periods of time – that is, until some change is made to the database. How much benefit you get from the query cache depends strongly on the kind of application you use – this is why the query cache is not turned on by default. When it comes to Magento, tests have shown again and again just how much extra scalability you get if it is enabled. To enable query caching, go to your my.cnf and set the following options in the [mysqld] section:

```
query_cache_type=1  
query_cache_size=64M
```

Save your changes and restart the MySQL server – getting a third more requests per second is not uncommon after the change!

There is a lot more optimization that you can do on the MySQL level, but it is beyond the scope of this post. Check out chapters 6 and 7 in O'Reilly's High-Performance MySQL 2nd edition for an excellent guide. The authors also have a great blog post on the query cache at <http://www.mysqlperformanceblog.com/2006/07/27/mysql-query-cache/>.

Measure your Magento

Understand Loading Times with Fiddler

When loading a web page in the browser, you get a feeling for whether the site is „fast“ or not. This feeling is very important, especially for your end users. But it is not actionable information, because if all you know is that the page is not fast, you don't know where to start changing your code or configuration.

Thankfully, Microsoft has released a very helpful freeware tool called [Fiddler](#), which is available for Windows only. Fiddler works as an HTTP proxy on your desktop computer, tracing the content of each request and response that your browser makes. You can then review the content of each of these, try „fiddling“ with requests and see how your app behaves, and much more. Fiddler can be a really helpful tool, from debugging Ajax to improving performance generally, so it may be worth your while to read some of the documentation and screen casts on their site.

We want to understand what makes up the page loading time of our Magento store, by components. And Fiddler can show a page load timeline that does exactly that. So after you have installed Fiddler and configured your browser to use it, empty your browser cache and go to the store page you are interested in. You will see a log of each request made in the Fiddler window. Select all the requests and click on „Timeline“ in the right pane.

You will see a picture that looks a bit like this:



Figure 1 - Fiddler Results

This time line was taken from a demo store, and shows you that Magento itself was running for just one second (the blue bar at the top in the right pane). The browser had only loaded all page elements after 11 seconds though – and three of these seconds were spent waiting for the JavaScript to load! When you think about performance improvements to your site, it is crucial to bear in mind that loading static objects can take 10 times as long as running Magento! Before touching the Magento application, you want to make sure that you optimized your static file delivery.

Use YSlow for additional advice

Yahoo has released a free Firefox add-on called Yslow that can also give you information on page loading times and – in contrast to Fiddler – give you easy-to-understand hints on how to improve performance. Go to <http://developer.yahoo.com/yslow/> for more information and download it – there is a very comprehensive user guide as well.

What YSlow gets right it is that it alerts designers to the role they play in getting great site performance. Reducing the number of HTTP requests – another topic we will return to below – is a very effective strategy to make your site feel faster for users.

Simulate Real-Life Users with Siege

Siege is an open-source tool developed in Perl that lets you simulate visitors surfing on your site. This is the best way to get a feel for how much load your application server can handle, given its current configuration.

Many administrators and developers estimate scalability of their site by hammering a (staging) server with requests. Basically, this means calling the same single URL over and over and seeing how many requests get through per second. You can do this with the Apache Benchmark tool ab2, for example.

The problem with this approach is that real users don't behave like this. The numbers you get are hence very misleading: first, if you enable caching, tools like ab2 will tell you how your application scales if the cache hit rate is nearly 100%. In reality, your hit rate will be lower because visitors to go many different pages all the time, so ab2 overestimates your scalability.

At the same time, it is difficult to interpret the numbers you get. Suppose your web server can handle 5 dynamic requests per second. Does that mean you can only 5 people visiting your site simultaneously? Of course not, because humans make requests at a much slower rate than a benchmarking tool. How many sessions your server can support is still difficult to estimate though, and will depend on many factors. Simulating user sessions directly is the best way around this problem, and it is strongly recommended that you use a tool like siege before going live with your Magento store. Too many sites crash immediately after load because the server capacity was drastically overestimated; nobody knows how many oversized servers are standing idle around the world because the opposite problem occurred.

The Siege home page at <http://www.joedog.org/index/siege-home> gives you all the info you need to get started. If you are running Linux, see if your distribution has a ready-made package available before installing it manually. As shown in the README, you need to create a text file containing the requests made in each session. Siege keeps track of cookies, so you can have users first logging in, then adding something to the shopping cart, etc.

Getting started with Siege takes a bit of time and practice, as does setting up the URLs file for your tests. But having this tool at hand can be invaluable for the success of your project.

Three Steps to Improve Scalability and Performance

Enable Block Caching where it makes sense

Out of the box, Magento can cache the Block output of your pages. When the next user requests the same block, the output that was previously calculated can be returned directly – without going through all the database queries and model calls again. This is really helpful for parts of pages that don't change too often but are somewhat expensive to calculate – such as category pages. Clearly, it makes a lot more sense to calculate output once and then cache it for, say, five minutes, than to continuously go through the same model calls with the same results again and again.

Block caching is not enabled out of the box, because you have to set the right cache to suit your use deployment. For example, if your prices depend on consumer groups, you need to add the consumer group to the cache key.

My recommendation is that [you read the wiki page on the subject](#), and then extend the Mage_Catalog_Block_Category_View class to enable caching. It is quite common that this simple change can double the number of sessions your server can support.

Make fewer HTTP Requests in your Theme

As YSlow alerts you, having many HTTP requests to load a single web page is a performance killer. Even if you enable keep-alive or use a content delivery network, it is still much slower to load many small images than a single bigger file that contains all images. Reducing the number of HTTP requests is largely a performance measure, but also helps scalability a bit.

Using image sprites, you can put all the icons, buttons etc. that you have in your theme into one file that can be downloaded quickly and then just show parts of this large image where you need it. All this involves is a little bit of CSS wizardry. A List Apart has a nice tutorial to get you started at <http://www.alistapart.com/articles/sprites/>.

Use FastCGI to run PHP

If you are using Apache as your web server, there are two ways you can sensibly set up PHP. The first way is to use the mod_php module, which is easiest to use and hence the default with many hosting providers. In this case, a PHP interpreter is run with each process of the web server and on stand-by until a script is executed.

If you are wondering why your Apache needs a lot of memory, probably the fact that you are using mod_php is a big part of the answer. On a large site, there may well be hundreds of Apache processes running, and each has its own PHP interpreter. However, only very few of them – often less than one in ten – actually need to run PHP. The rest serve static files or simply wait for new requests. Because PHP uses a lot of memory, it is a good idea to see if you can avoid the overhead generated by having dozens and dozens idle PHP processes running.

The way to avoid the overhead is to use FastCGI instead of mod_php. With FastCGI, a separate internal daemon is run on your web server that is contacted by the web server only when execution of PHP is required. Thus you do not need to carry the PHP baggage for all requests.

Setting up FastCGI requires you to make some changes to your server configuration, but the benefits will be large. A good starting point is this article: <http://2bits.com/articles/apache-fcgid-acceptable-performance-and-better-resource-utilization.html>. For more details, check the Apache web site and the documentation of your distribution.

Alternative: Turn Off Keep-Alive if you have to use mod_php

If you cannot, or do not want to, switch to FastCGI, you can still do something to reduce the memory usage per visitor. This is important because each server has only so much memory, and if you can serve a visitor with less memory, you can server more visitors in total and scale further with given resources.

As I pointed out above, the big problem with mod_php is that you need to keep a PHP interpreter running with each Apache process, and that most Apache requests do not involve PHP. By default, Apache enables a feature called HTTP Keep-Alive, which lets visitors re-use a connection for several requests. This makes the web site faster for the visitor, because images and other static files can be loaded without continuously re-connecting with the web server. But it also means that your web server will have many idle processes, waiting for new requests that may never arrive. And each of these idle processes runs a full PHP interpreter.

To turn off Keep-Alive, search your Apache configuration files for the KeepAlive directive. If the directive is not set, add the following line to your config file

`KeepAlive off`

and then restart Apache. If it is set, ensure that it is set to "off". You should start to see lower memory usage immediately.

Use a Content Delivery Network

Another way to relieve stress from your servers is to get an external party to serve static files for you. These services, called Content Delivery Networks (CDNs), are getting very popular and prices have fallen a lot over the last year or so. A CDN can really improve the user experience on your site, and is another way around the problems created by mod_php.

Setting up a CDN is quite easy with the free [One Pica CDN extension from Magento Connect](#).

Going Further

You have followed all the steps of this guide, and still your Magento does not scale to the level you need or give you the performance you crave? First, check again if you have really implemented all the steps. The configuration I described above is known to work very well with large Magento stores, serving hundreds of thousands of page impressions per day. Check that your hardware works well and that you are not limited by the bandwidth of your server.

If your site is truly huge, you will want to move to a cluster infrastructure that also provides high availability to insure you against software or hardware failures. You need to set up MySQL replication for your database and balance load between web servers. In practice, this will usually mean that you move to a specialised managed infrastructure provider or get outside consultants to set up the infrastructure for you.

Magento is very well suited for such deployments. You can set different database connections for read and write queries, which is important when you use MySQL replication, use memcache as a distributed cache backend, and the like. So Magento will not limit you there – the problems you may encounter will be related to the operating system and other software stack.

Conclusion

Performance and scalability are important, but difficult, topics. Here I have tried to get you started with the tools and strategies you can use to get more out of your existing infrastructure. Very often, just taking the easy wins is more than enough to get the power you need for your Magento shop.

As a side note, I want to point out that most of the topics we covered did not relate to Magento directly, but to the server configuration. And this is something you should bear in mind: often, the performance and scalability issues you face are not problems of Magento, but of your infrastructure. But because Magento is quite a complex application, you will run into these problems earlier than with simpler apps.

Gift Cards and Customer Store Credits in the Magento Enterprise Edition

<http://www.magentocommerce.com/blog/comments/video-gift-cards-and-customer-store-credits-in-magento/>

Check video at Here

http://www.youtube.com/watch?v=e2IbHkRN0vg&feature=player_embedded

The [Magento Enterprise Edition](#) is a powerful eCommerce platform with features designed to increase sales.

Keep your customers coming back by offering store credits and gift cards. Offer physical gift cards sent in the mail and virtual gift cards sent to an inbox. Set price tiers for customers to choose, or let them enter an amount within a preset minimum and a maximum value. The Enterprise Edition gives you control to define the card lifetime and enable gift messages so customers can add a personal touch.

Recipient's can check their card balance before checkout, and use the available funds for purchase. Use an unlimited number of gift cards, or redeem and receive store credit. The Enterprise Edition keeps a gift card history so that you can see how gift cards are being used.

Reward loyal customers and handle returns by issuing store credits so customers can use the balance for future orders. During the refund process, simply select the full refund amount, or enter a partial amount to create an automatic store credit. Your customers can choose to use their store credit during checkout along with other available payment methods. The Enterprise Edition makes it easy.

Improve Conversions in 60 Seconds

<http://www.magentocommerce.com/blog/comments/video-improve-conversions-in-60-seconds/>

Video Here http://www.youtube.com/watch?v=1v7m9BFqTJ0&feature=player_embedded

The Google Analytics Blog [recently posted](#) a great video showcasing how “Google Website Optimizer can help you identify the copy, images, and page layout combination that is most effective at getting visitors to convert”. Together with Google Analytics, you can “find out which landing pages are least effective, so you can start working (optimizing) on those first”.

Google Website Optimizer is natively integrated with Magento as of version 1.1.7 (see related [video](#) and [webinar](#)).

Magento & Zend Server Benchmarks

<http://www.magentocommerce.com/blog/comments/magento-zend-server-benchmarks/>

Posted by [kameo](#)

This post is written by [René Amirkhanian](#), [Adrien Urban](#), [Philippe Humeau](#) from [NBS System](#), a [Magento Hosting partner](#) and is [reproduced](#) by permission from the French [WikiGento Blog](#).

Magento is a PHP/Zend application which intensively uses the CPU. Since version 1.1.6, each new version includes some mechanisms aimed to improve the performances. The goal is to use fewer resources for a given e-shop, which mainly means less CPU, in order to host more users with the same hardware.

One key to achieve better performances is how to optimize PHP pages generation and service. “LAMP” servers are well known and usually run Apache server with mod-php, eventually in fast_cgi mod.

Zend, the PHP Company, made a specific server (Zend Server), which includes a web application stack that (among other things) improves application performances through page caching and opcode reorganization & acceleration.

Apache and Zend Server is an alternative to the usual Apache and mod-php to run Magento, the goal of these studies & tests is to qualify and estimate the performances added by the use of this software.

Many thanks to Yoav Kutner (Varien’s CTO) for providing us with prefilled catalogs for 1.2 and 1.3 version of Magento. Thanks goes as well to Zend labs for providing help in configuration and tweaking of the Zend Server as well as explaining the in depth mechanism of the solution.

Methods & Tools Used

The benchmarks were done using siege (2.66-2), with different numbers of simultaneous threads (5, 10, 20 and 40). Each thread opens a connection to the web server, request a page, and start again as soon as the page is fetched.

Tests were run over 5 minutes each; average pages retrieved was counted for this benchmark.

Two kinds of tests were done: the first one is the simplest; we only load the main page in loop, as fast as possible.

The second test is based on logs produced by a visit, loading all the data the browser collected (including css, js, images ...). As loading a single page would usually load multiple elements, the number displayed is much higher than the number of pages that would be actually seen by visitors.

(18 pages viewed by the user, for 91 items downloaded, a ratio around 5)

All results were obtained on the same hardware and operating system, for testing purposes, no reverse proxy was active during the benchmarks but APC code cache was running. This was a “standard”, default environment with no special performance tweaks installed.

Hardware:

2 processors: Quad-Core AMD Opteron(tm) Processor 2376 (2.3GHz), 8GB Ram

Software:

Operating system: Linux (2.6.27.10-grsec) on a Debian (lenny)
Web Server: Apache2 (2.2.9-10+lenny2)
PHP (mod-php): mod-php5 (5.2.6.dfsg.1-1+lenny3) / php-apc (3.0.19-2)
PHP (Zend Server): zend-pe (1.0.0-1+b47) / mod-php5-zend-pe (5.2.9+b75)

Repositories:

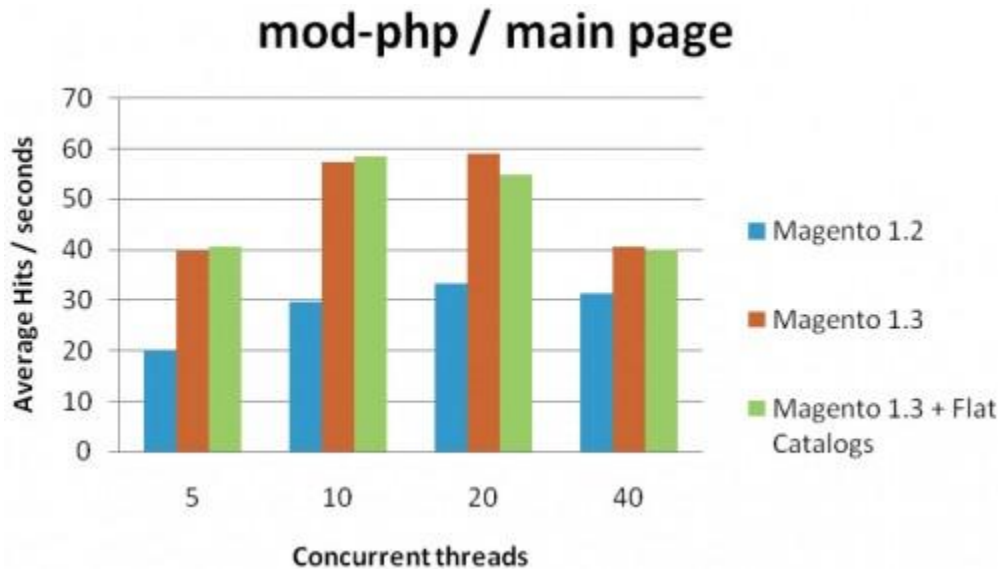
deb <http://ftp.fr.debian.org/debian/> lenny main
deb <http://security.debian.org/> lenny/updates main
deb <http://volatile.debian.org/debian-volatile> lenny/volatile main
deb <http://repos.zend.com/deb/pe> pe non-free

Magento Version Benchmarks

Those tests were realized on an 80 000 (later called 80 k) products catalog.

Graphs represent the average number of requests successfully loaded from the server per second during the 5 minutes test.

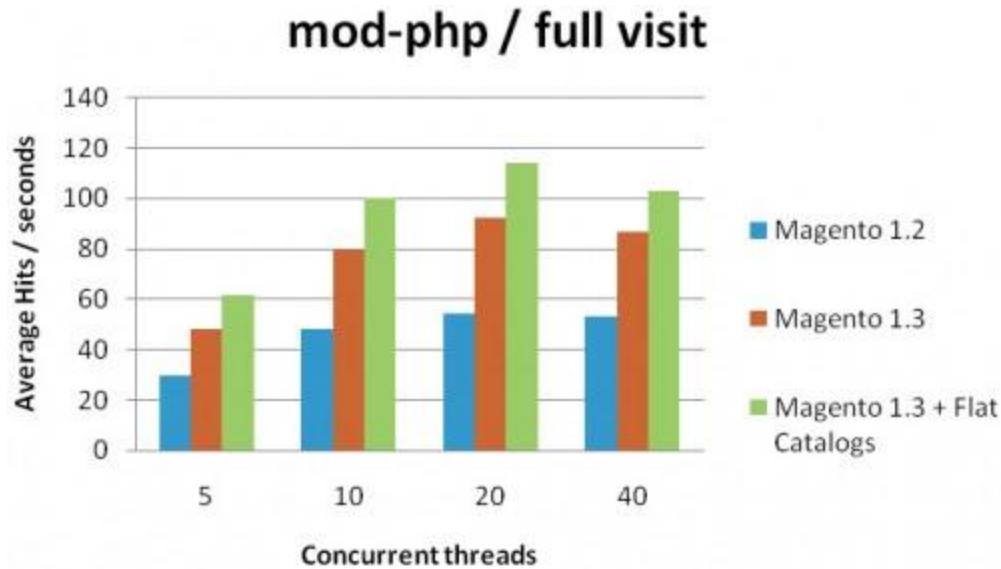
Loading the homepage



Magento 1.3 is much faster than 1.2 showing the main page. Flat Catalogs do not help much displaying this homepage, and seems to even slow down a little under heavy loads. The performance between 1.2 and 1.3 is doubled for low charges and is still 33% higher under heavy load.

For a standard value, under a “standard” load, we can consider that +40% is a reasonable value when running a 1.3 version instead of a 1.2, at least for the homepage.

Full visit cycle



The graph shows a slightly smaller increase with 1.3 compared to 1.2 and the flat catalog mechanism give an overall benefit which greatly increases the performances.

For the records, we can reasonably choose to keep these values in mind:

Version 1.2 -> 1.3 : +66%

Version 1.3 -> 1.3+Flat Catalog : +20%

Apache + mod-php VS Apache + Zend Server

Zend Server (sometimes referred as ZS later on) comes with several built-in technologies for enhancing application performance:

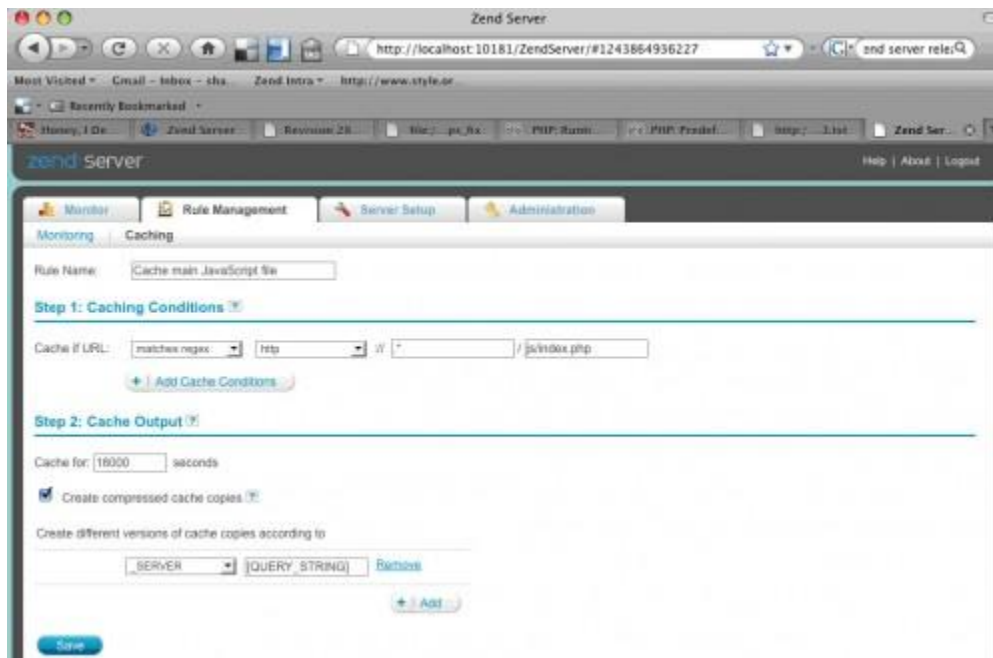
- Zend Optimizer+ performs byte-code optimization and caching. This speeds up PHP applications by eliminating the process of reading scripts from disk and compiling them. Zend Optimizer+ runs automatically, and installing your application on top of Zend Server (ZS) is all you need to do in order to enjoy its benefits. During the test with ZS, APC Code cache is deactivated as Zend optimizer+ is doing the same job.
- Zend Page Cache allows caching of complete PHP pages. Page Caching greatly improves the performance of web applications while maintaining dynamic capabilities through an elaborate system of caching rules that could be based on request parameters and user session data. Page Caching also has the benefit of not requiring any code changes, and can be set up from the Zend Server UI. Only the “pro” version contains this precise piece of software which definitely makes a difference as we will see in a minute.
- Zend Data Cache is a set of API functions enabling a developer to store and manage data items (PHP strings, arrays and other data) and even output elements in either disk-based cache or shared memory cache. Zend Data Cache allows for precision-guided caching when Page Caching is not an option. The provided API is easy-to-use on existing code, and in many cases a developer can skip existing code sections by simply wrapping them with caching APIs. This precise piece of software would benefit from a little remastering of the code by Varien to really achieve a full support of this functionality. If done, we can imagine selectively flushing the cache when changing some pages on the servers and not destroying the whole Magento cache thus doing a “cold cache start” after a new functionality is put online.

Note: Zend Optimizer+ and Zend Data Cache are available in the free, community version of Zend Server, while Zend Page Cache requires a licensed Zend Server (full comparison of Zend Server and Zend Server Community Edition is located at <http://www.zend.com/fr/products/server/editions>).

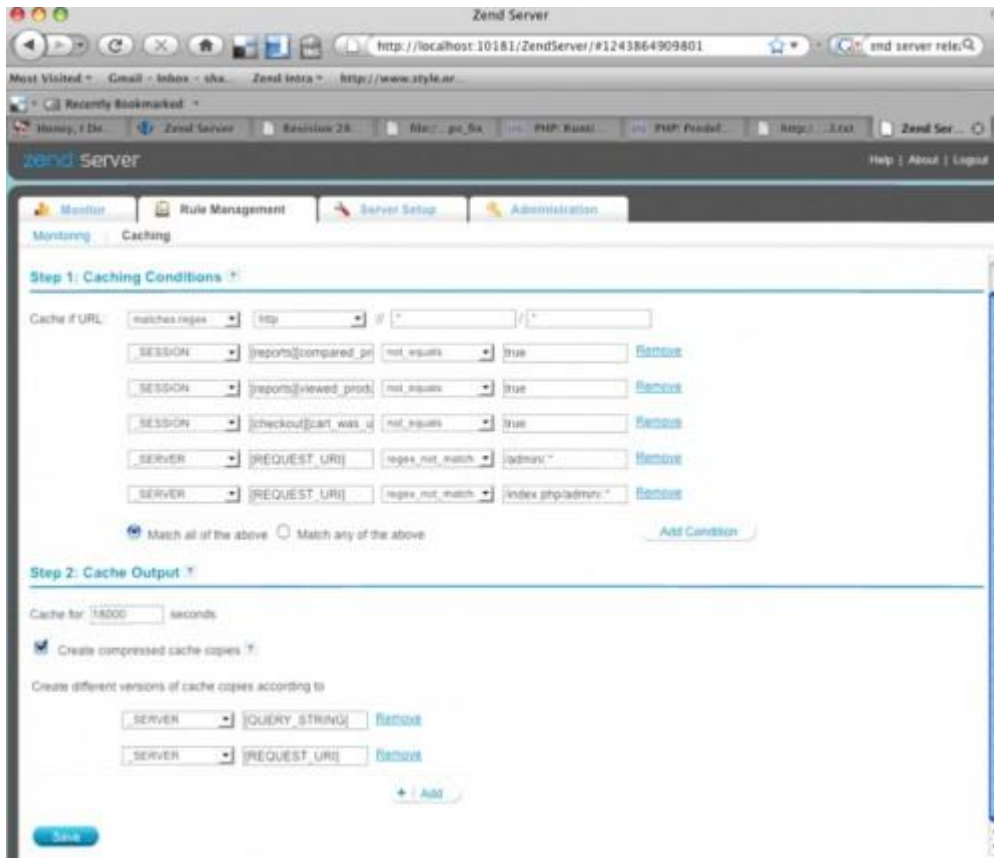
Zend Server Configuration for Magento

In the benchmark, two caching rules have been defined:

The first rule caches the JavaScript files which are dynamically merged into one request by Magento. This simple rule results in a very substantial improvement to response times.



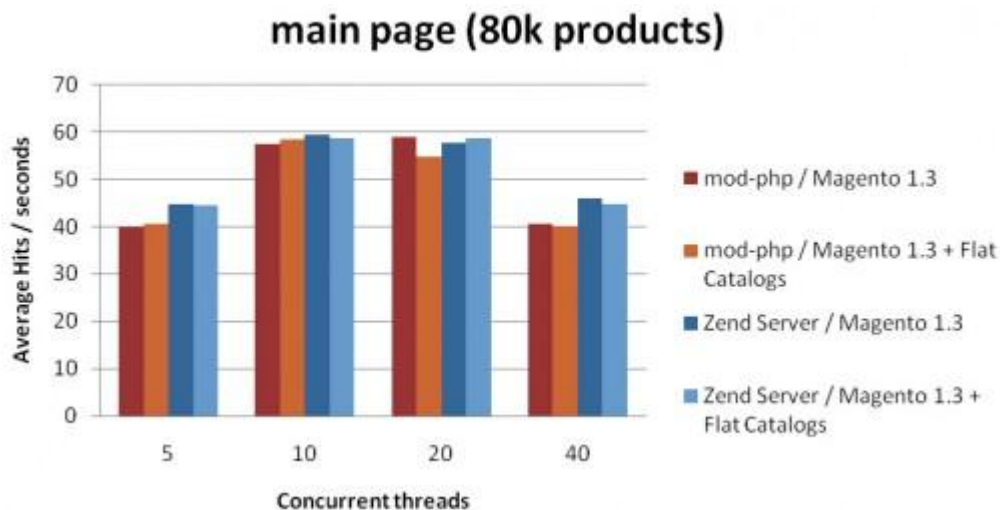
The second rule caches all web pages accessed by users who have nothing in their shopping carts or history (when the shopping cart is not empty or when the user history is saved, there is no point in page caching). As you can see in the screen capture below, this is accomplished by looking at `$_SESSION` variables and by splitting according to `$_SERVER['REQUEST_URI']` in addition to the `QUERY_STRING`.



Finally, Zend Optimizer+ has been enabled for optimizing and caching the PHP byte-code.

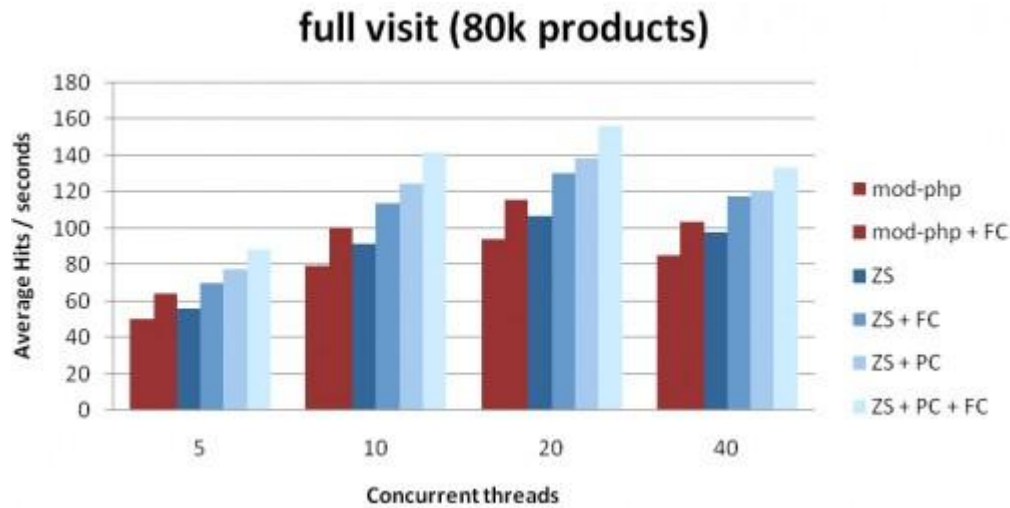
As you can see, Zend Page Cache seems a very powerful feature. It needs fine tuning and better configuration, but let's see results :

Homepage / 80 000 products catalog



Load on main page doesn't prove very constructive, although, on heavy load (40 concurrent requests), Zend Server (with or without page cage) is slightly better than basic mod-php, but almost no real improvements on this test. A 5% or less win is not to be taken seriously as it is the error margin of the tests.

Full visit / 80k products



Full visit graphic speaks for itself, whatever the load, Zend Server with Page Cache and Flat Catalog make a big difference.

Comparing a 1.3 with flat catalog with a the same configuration but with a Zend Community installed instead of just APC, our server yield up to 15% more performances, just using a free edition of ZS.

If we use a full Zend server (the licensed one including the page cache) on this 1.3 flat catalog, our server goes up to 30% more performances!

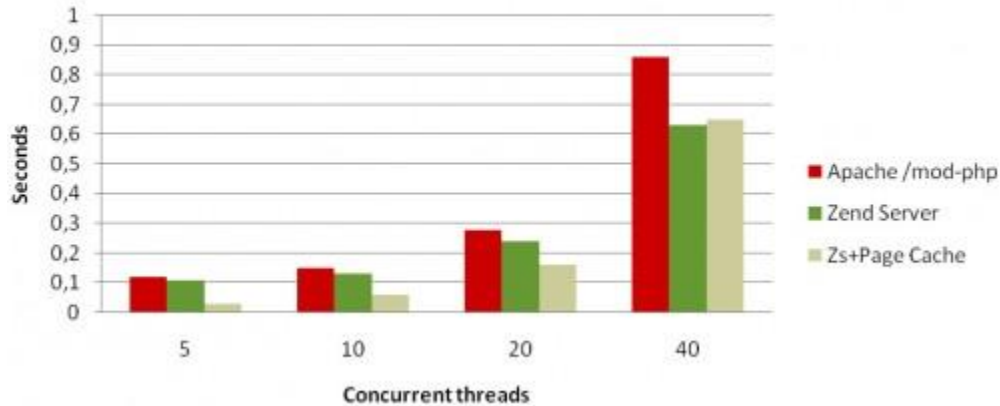
Mod PHP + APC -> ZS: +15%

Mod PHP + APC -> ZS licensed edition: +30%

Response time

Homepage / 80k products

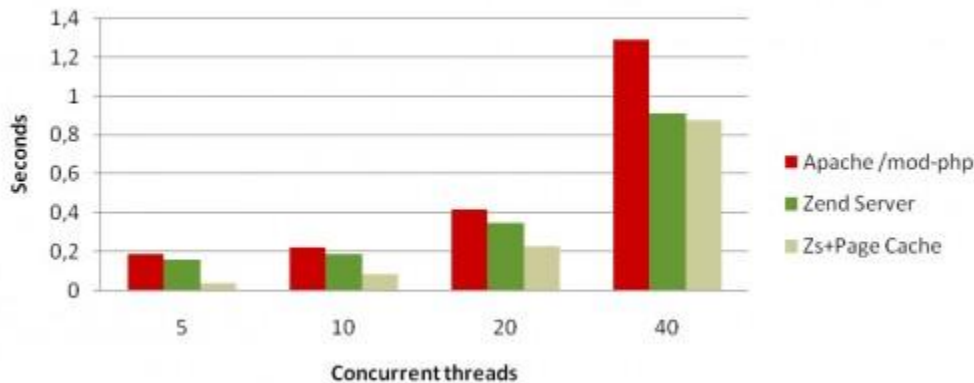
Response time / Homepage



This graph shows response time on the homepage measured during a load. No match on this test, under a good load, ZS, community or licensed edition, gives up a 35% boost in the load time, good to take.

Full visit / 80k products

Response time / Full navigation



Page Cache provides a better response time over a full visit of the site, even under very heavy load. As the difference remains thin, I would conclude it to be identical in that condition. However, the benefits of the page cache under a standard load remains a must have. Once again, a 35% win is to be considered as a reference value under a good load whether you are using a community or licensed version.

Under medium or light load, the page cache is giving a lot more power than the community edition can do. This strange result is probably mainly due to the page cache mechanism which prove to be not that efficient under heavy load because the system is using its resource in a different way. We didn't had time to check if it was a False cache sharing effect on L2 or L3 processor cache or a Linux or ZS issue but the test has been runned several times yielding the same results.

So keep in mind that the benefits of the ZS Page cache mechanism is going to be less and less active as the server load gets high. Under a usual load, a reasonable load lets say, the Page Cache algorithm gives a nice boost to the loading time, driving it down for ~40%.

Additional technical details about the tests

Sample data

All tests were run with generated sample data (provided by Varien)

Parameter	10k	80k
Categories	251	1576
Category * product associations	24690	202743
Products	10001	78994

Description of the tests

Main page test

Each concurrent thread loads the main page, without storing the cookies to simulate simultaneous users.

Full visit test

We have a list requests based on a visit done with a real browser. Each concurrent thread loads each item one after another as fast as possible, and restarts after deleting all its cookies. This simulates simultaneous users, except that simultaneous.

List of requested pages for the 80k database (with POST information when it applies) [can be found here](#).

Things to test in the next benchmark campaign

Mage Compiler

When running all those tests, 1.3.1.1 version including the Mage optimizer was about to be released and an immediate 1.3.1.2 version followed and we didn't had time to include some tests around this precise point. Let's just consider the fact that this mechanism is mainly made to lower I/O jobs by "compiling" the libraries files all in one or two includes only. If your servers were already using mainly their RAM to work, the performances increase will not be that valuable. But if your servers where low on RAM or using mainly their disks, you should feel confident with using this option which can do great good to your performances!

Nginx

We didn't had the required time to consolidate the results involving Nginx + PHP tests. This solution ranked almost every time between Magento 1.3 + FC and Magento 1.3 + Zend server (without Page cache). So this combination yield average performances, not as sharp as a Magento 1.3 + Zend server or any more advanced combination.

More "custom" visits

We have used a scenario involving each time a research in the search bar of the website and some "standard" behaviors. A more precise test can be run replaying some real traffic pumped up from apache log on a real site, the customer's behaviors being more realistic even if our scenario was as logical as possible.

Conclusion

First, let me say that no electrons were armed or injured during these benchmarks. Perhaps one or two CTO were put under constraint and continuous coffee perfusions but all of this was intend for the greater good of E-commerce!

If you wish a more professional conclusion, let's say that using ZS community edition will only do you good and really can replace APC.

If using the licensed version, the Page cache is very efficient (under reasonable load of the servers) and can help using ~25% less machines to achieve similar hosting capacities. If your servers are billed in a "managed hosting" way, having one or two less servers billed per month can make a good difference in the budget.

About Zend Server : this software has many other great functionalities and this paper is only consider the "performances" issues, just pay a visit to Zend Website for a far more complete overview of the product.

Last but not least, creating a server with all optimizations and best practices, based on an a dual AMD 2376 with 8 Go of RAM and using ZS and page cache, you can try to reach up to ~40 000 unique visitors a day corresponding to ~2 500 Magento simultaneous session at maximum load. (if database is separated on a different server and you activate a reverse proxy like Squid or Varnish above your front web servers)
This estimation is deeply linked to the website complexity and user standard behavior but these figures are given for a "standard" site and use, you can usually also expect a ~2 seconds loading time on the homepage.

If we had to sum up a quick dirty, average table of the performances, without any details, we would give that:

Combination	Relative capacity
1.2	100%
1.3	150%
1.3 +FC	180%
1.3 FC +ZS	207%
1.3 FC +ZS licensed	234%

Content Staging and Merging in the Magento Enterprise Edition

<http://www.magentocommerce.com/blog/comments/video-magento-enterprise-edition-feature-highlight-content-staging-and-merge/>

Video: http://www.youtube.com/watch?v=QSG-4gWSyac&feature=player_embedded

One of the most valuable Enterprise Edition features is the ability to create, edit, and test new store content on a Staging web site. The live store remains untouched and customers can keep shopping while you update the site and develop new exciting content. Create new marketing campaigns, update your product catalog, test new designs, and enable new features in a safe environment separate from your live business.

Add a new staging site through an easy to use interface by simply selecting the site you would like to duplicate. The ability to select specific content areas to be duplicated gives added control and flexibility. Create multiple staging sites and save time by working on different data areas at once. When the content is ready, it is merged to the main staging site to be taken live.

The privacy of your staging site is important and The Magento Enterprise Edition gives you the option to password protect your staging environments. Each staging site will keep a history of all actions so that you can keep track of who did what, when and why.

When you're ready to take your changes live, you can merge to the original site, or choose another Magento website in your installation.

Select a manual merge, or schedule the merge to occur automatically at a later date. Additionally, The Enterprise Edition let's you take different elements live at different times so you don't have to wait to provide site updates to your customers.

And, if you ever need to revert back, you can roll back your entire store, or just the needed elements, to a previous state.

Speed up your store by combining, compressing and caching JS and CSS-- Fooman Speedster

Extension: <http://www.magentocommerce.com/magento-connect/FOOMAN/extension/457/fooman-speedster#overview>

Overview

Speed up your store by combining, compressing and caching JS and CSS. This extension reworks how Magento handles the loading of JavaScript and CSS. It utilises the [Minify library](#) developed by Steve Clay and released under a BSD license.



See the attached screen shots for a before / after comparison on my setup.

Benefits

- * Javascript AND CSS are combined into one file each
- * compression even without gzip support on your server
- * Automatic Versioning - every time you update one of your JS/CSS files a new url gets created forcing the customer's browser to request the new version (be sure to update the Magento cache as well)
- * Minify does its magic to ensure that CSS images are properly served no matter what package/interface/skin you are using.

Current Requirements / Caveats

- * needs mod_rewrite enabled / .htaccess support
- * some JS and CSS are currently excluded (for example print.css)
- * Installation of CANONICAL URLs by Yoast breaks this extension. A work-around is posted [here](#).
- * Installation of MXPERS JQUERY BASE breaks this extension. Tips are posted [here](#).

Installation

1.) Add the following to your .htaccess

```
#####  
## Compress, Combine and Cache Javascript/CSS  
RewriteRule ^(index.php)?minify/([^\s]+)/.*.(js|css)$ lib/minify/m.php?f=$3&d=$2
```

Directly underneath the Rewrite Base Rule and the workaround for HTTP authorization rule. Advanced: For

installation on a non-Standard Linux installation check out the [ISAPI2](#) and [Nginx](#) threads.

2.) In the same htaccess uncomment the RewriteBase rule and make sure it points to the folder where Magento's main index.php resides in.

3.) After installing this extension via Magento Connect make sure that `/lib/minify/m.php` is executable (permissions like 755 on the file itself and the containing folder should work) and `/var/minifycache` is writeable.

4.) Minifying JS and CSS takes a while to compute (~20seconds on my machine). This only needs to be done once per JS/CSS combination and is then written to a cache. To make sure the cache gets filled up simply browse your store and your customers should have a faster experience than before.

If you are running a multi store set up please follow these [instructions](#). Additionally add a symlink to the lib folder.

Please note the above instructions apply to FOOMAN_Speedster 0.7.0 and above and are slightly different to previous versions.

For users of versions 0.7.0 - 0.7.6 please note that editing `/app/design/frontend/YOURINTERFACE/YOURTHEME/layout/page.xml` is not needed any more. Please change it back to the original `validation.js` instead of `validation-4min.js`

Upgrades Made Easy

<http://www.magentocommerce.com/blog/comments/upgrades-made-easy/>

A huge benefit of Magento will be the ease with which upgrades and customizations can be made. We know that no eCommerce system can include the functionality needed for every business application right out of the box (although we're sure trying) and this is why we have made Magento so customizable. Not only is it customizable, but **after customizing you will still be able to upgrade to new versions of the system.**

Have you ever had the experience with an open source product where you customize it to your needs, spending nights going through the code, only to have a new version of the system come out a week after you finish? You're not able to enjoy the benefits of this upgrade, because you wanted to add a field to your registration form, or change a coupon module.

I know I've had this experience while working on projects for [Varien](#). Once you customize the code to your needs you're not able to upgrade the system without sacrificing these customizations. We knew this was the first thing we needed to address when developing an enterprise-level open source eCommerce platform.

Imagine you spend the time and/or money to customize the coupon module to your particular needs and an updated version of Magento is released a week later. Relax, you don't need to rearrange your social calendar. **You can upgrade Magento with the click of a button or through our auto update while still retaining your custom changes to the coupon module.**

This will cut down on development costs and time, allowing you to concentrate on growing your online business.

CSRF Vulnerability in Web Applications (and how to avoid it in the Magento Admin)

<http://www.magentocommerce.com/blog/comments/csrf-vulnerabilities-in-web-application-and-how-to-avoid-them-in-magento/>

In a recent post on the *artisansystem* blog, there is a description of a CSRF (?) hypothetical attack on a Magento admin. It is important to note that for this attack to be possible, the attacker must know the admin path (*frontName*). If this is unknown to the attacker, the attack will result in a *noroute* and will not cause any harm.

The Magento Core Team has identified this vulnerability a few months ago, and as a solution introduced in previous releases a way to set a custom path to the administrative panel in the installation process and via the local configuration. Since this [recent blog post](#) puts at risk any Magento user that specified 'admin' as their path, we urge all users to specify a non-trivial alternative path to the admin that is known only to people that need to gain access to the admin panel.

Security is on top of our priorities when it comes to our users and we are constantly testing and resolving any issues as we become aware of them. We recommend always running the latest Magento version so that your installation is up to date with any security updates. A [new security focused forum](#) is now available to discuss such topics.

How to update admin path in an existing Magento installation

Disable all caches in System->Cache Management

In your `app/etc/local.xml` file, update the value under `admin->routers->adminhtml->args->frontName` to any custom value you wish your admin to run under.

Your resulting entry should look like this:

```
<admin>
  <routers>
    <adminhtml>
      <args>
        <frontName><![CDATA[your_custom_admin_path]]></frontName>
      </args>
    </adminhtml>
  </routers>
</admin>
```

Live Chat Extension

<http://www.magentocommerce.com/blog/comments/magento-connect-live-chat-extension/>

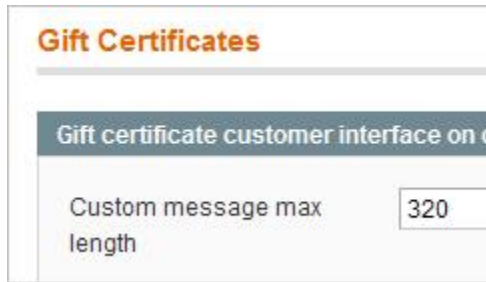
<http://www.magentocommerce.com/extension/808/livechat>

<http://www.kalliser-net.com/LiveChatDemo.htm>

Magento Connect: Gift Certificates / Virtual Cards Extension

<http://www.magentocommerce.com/blog/comments/magento-connect-gift-certificates-virtual-cards-extension/>

http://www.magentocommerce.com/magento-connect/Unirgy/extension/751/unirgy_giftcert/



A new extension is now available via [Magento Connect](#) providing [Gift Certificates / Virtual Cards](#) functionality (in beta). There is little documentation as to how the extension works, but screenshots are available to preview the functionality.

[Get the extension key here.](#)

How To Setup Multiple Magento Stores

<http://www.crucialwebhost.com/blog/how-to-setup-multiple-magento-stores/>

Crucial Web Host has posted on the [Crucial blog](#) a great in-depth tutorial on [How To Setup Multiple Magento Stores](#). This has been a topic of great interest for the Magento Community. The tutorial covers:

1. [URL Structure](#)
2. [Shared Hosting Caveat](#)
3. [Adding Another Store In Magento](#)
4. [Parked Domain Method](#)
5. [Addon Domain Method](#)
6. [Subdomain Method](#)
7. [Subdirectory Method](#)
8. [Managing Multiple Stores](#)
9. [Secure Checkout For Each Domain](#)

TYPO3 + Magento = TypoGento

<http://www.magentocommerce.com/blog/comments/typo3-magento-typogento/>

TypoGento is a middleware service that connects the content management system [TYPO3](#) (version 4.2+) and the e-commerce application [Magento](#). This enables companies and organisations to seamlessly include feature-rich shopping possibilities into their web site ... Because both of the integrated products are open source projects the middleware is for free as well. Everybody is encouraged to use it, change it and enhance it – and certainly give us feedback about it.

Visit [TypoGento.com](#) for more information.

Google Base Integration in Magento

Google Base integration is available as an out-of-the-box feature. In this video, Rico Neitzel, our German community moderator, walks through the implementation.

<http://vimeo.com/2368176>

Video: Google Website Optimizer Integration in Magento

1.1.7

<http://www.magentocommerce.com/blog/comments/video-google-website-optimizer-integration-in-magento-117/>

In this video, learn how to setup a a multi-variate test for a Magento category using Google Website Optimizer.

[Download iPod \(.mp4\) Version](#)

[Download Windows Media \(.wmv\) Version](#)

Tutorial: Integrating 3rd Party CMS Content Within Magento

<http://www.magentocommerce.com/blog/comments/tutorial-integrating-3rd-party-cms-content-within-magento/>

Integrating 3rd party CMS content into Magento has been a popular request by the Community. Outlined below is a technique using [Expression Engine](#) with the idea of managing content pages with EE, and displaying the content within [Magento](#), using Magento's 404 event handler. We will fetch this content using [Varien_Http_Client](#).

Please note that Expression Engine is used for demonstration purposes only and other CMS platforms ([Wordpress](#), [Modx](#), [Joomla](#), [Drupal](#)) can be used if desired.



Click To Enlarge

Getting Started

For this example, we'll assume Expression Engine is installed inside the Magento directory in a folder called 'ee', so our store's URL will be <http://www.example.com> and our EE's URL will be <http://www.example.com/ee/>.

Note: Both Magento and EE must be installed with the Apache URL rewrite feature.

The .htaccess file for EE:

RewriteEngine on

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-l
RewriteRule ^(.*)$ index.php?/$1 [QSA,L]
```

The home page of your installed EE must be the same as your Magento store home page (<http://www.example.com>)

Create a Local Code Pool Module

We will create a module in the local code pool as follows:

- create a new directory called "Project" under "app/code/local/Mage"
- create a new directory called "Block" under "Project"
- create a new file called Noroute.php under "Block" and copy the following code into the script.

```
class Mage_Project_Block_Noroute extends Mage_Core_Block_Abstract
{
    protected function _toHtml()
    {
        /**
         * This logic should be in the controller, model, but let's make
         * this quickly without lots of files :)
         */

        $uri = Mage::getBaseUrl() . 'ee' . $this->getRequest()->getRequestString();
        $post = $this->getRequest()->getPost();
        $method = ( count($post) == 0 ) ? 'GET' : 'POST';

        /**
         * You can add additional
         * headers like cookies, redirects and so on
         * if you need it here.
         */

        $client = new Varien_Http_Client($uri);
        $client->setParameterPost($post);
        $response = $client->request($method);

        $body = $response->getRawBody();
        return $body;
    }
}
```

To enable the *Project* module from local pool, create a file called "Magento_Project.xml" under "app/etc/modules" and copy the following code:

```
<?xml version="1.0"?>
<config>
    <modules>
        <Mage_Project>
            <active>true</active>
            <codePool>local</codePool>
        </Mage_Project>
    </modules>
</config>
```



```
</modules>  
</config>
```

Create a Layout

Now we need to add this block to the layout. Let's edit the `app/design/frontend/YOUR_DESIGN_PACKAGE/default/layout/cms.xml` and modify 'cms_index_defaultnoroute' section:

ext, we'll disable the 404 page (identifier: 'no-route') in the Magento administration (CMS->Manage Pages).

Finally, refresh Magento's Cache and edit the templates within EE removing all headers, footers. Visiting pages like <http://www.example.com/sample> will now include the content from <http://www.example.com/ee/sample> with Magento's header, footer and other components.

Extending the Technique

In order not to redirect all invalid requests to EE, a regex expression can be added to limit the requests that are redirected to EE and such requests will be processed by the Magento's noroute handler.

Magento Connect: Simple Configurable Products Extension

<http://www.magentocommerce.com/blog/comments/magento-connect-simple-configurable-products-extension/>

In alpha, this extension changes the way that the pricing of configurable products works in Magento. With this extension enabled, a configurable product's own price is never used. Instead, the price used is that of the appropriate associated child product.

This gives site owners direct control to set the price of every configuration of a product, while still giving users the flexibility they usually get with configurable products. (There's no more having to set rules such as: +20% for blue, -£10 for small, +15% for leather. You just price the underlying small-blue-leather product at £199.99 and that's what the user pays.)

[More information on the extension page](#) in [Magento Connect](#)

A/B Split & Multivariable Testing with Google Website Optimizer

<http://www.magentocommerce.com/blog/comments/117-sneak-peak-a-b-split-multivariable-testing-with-google-website-optimize/>

Many videos at above link

Expected in late November, the next Magento release will feature native integration with [Google Website Optimizer](#) for **A/B Split & Multivariable Testing**. Screenshots of the upcoming functionality are available below.

Website Optimizer, Google's free website testing and optimization tool, allows you to increase the value of your existing websites and traffic without spending a cent. Using Website Optimizer to test and optimize site content and design, you can quickly and easily increase revenue and ROI whether you're new to marketing or an expert.

Magento Admin Panel

Dashboard Sales **Catalog** Customers Promotions Newsletter CMS Reports System

Choose Store View: Default Values

HTC Touch Diamond (Cell Phones)

Product Information

- General
- Prices
- Meta Information
- Descriptions
- Images
- Cell Phone Attributes
- Design
- Inventory
- Websites
- Categories
- Related Products
- Up-sells
- Cross-sells
- Product Reviews
- Product Tags
- Customers Tagged Product
- Custom Options

Product View Optimization

Google Optimizer Scripts

Scripts Install URL: **Install Scripts**

Control Script: (STORE VIEW)

Tracking Script: (STORE VIEW)

Conversion Script: (STORE VIEW)

Conversion Page: -- Please Select -- (STORE VIEW)

Conversion Page URL: (STORE VIEW) + Please copy and paste this value to experiment edit form

Attributes: Name (STORE VIEW) + Limit to 8 attributes only

New Category

Add Subcategory

Choose Store View: All Store Views

Categories All Expand All

- Root Catalog (8)
- Furniture (0)
- Electronics (14)
- Apparel (20)
- Household Items (0)

General Information | Category Products | Custom Design | **Category View Optimization**

Google Optimizer Scripts

Scripts Install URL: **Install Scripts**

Control Script: (STORE VIEW)

Tracking Script: (STORE VIEW)

Conversion Script: (STORE VIEW)

Conversion Page: -- Please Select -- (STORE VIEW)

Conversion Page URL: (STORE VIEW) + Please copy and paste this value to experiment edit form

Attributes: (STORE VIEW) + Limit to 8 attributes only

Conversion Page dropdown menu:

- Please Select --
- Other
- Shopping Cart
- One Page Checkout
- Multi Address Checkout
- Order Success (One Page Checkout)
- Order Success (Multi Address Checkout)
- Account Registration

50+ Payment Gateways Now Supported in Magento

<http://www.magentocommerce.com/blog/comments/50-payment-gateways-now-supported-in-magento/>

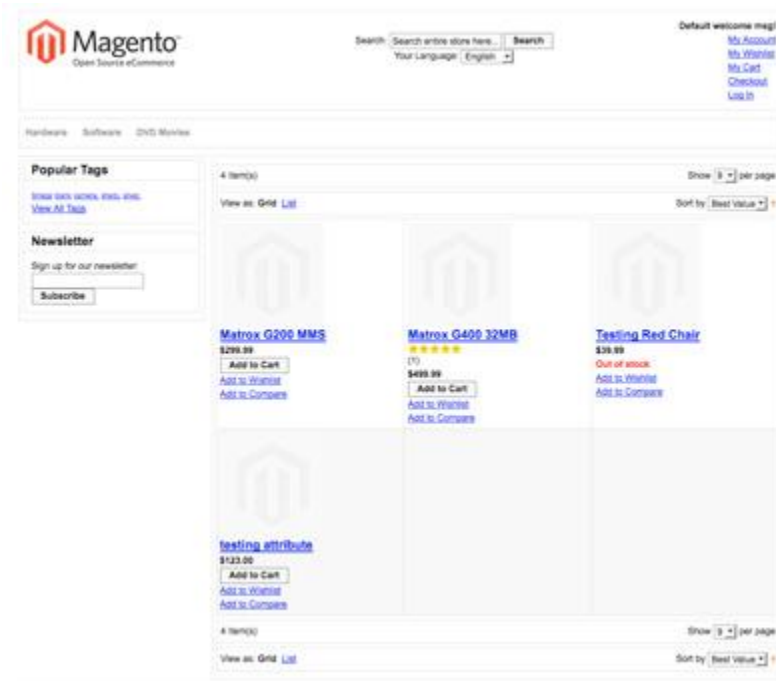
The [Magento](#) ecosystem continues to grow with support for over **50 payment gateways** now available. The Magento community has taken an active role and contributions account for the bulk of the extensions (as listed below). In addition to the core bundled support for Authorize.net, Paypal (various products), Google Checkout and Check/Money Orders, you can find the following extensions on [Magento Connect](#):

- [Fontis Australia](#)
- [Cybersource](#)
- [Protx](#)
- [Tweakmag Payment](#)
- [SPPLUS](#)
- [Fontis New Zealand](#)
- [eWAY](#)
- [ChronoPay](#)
- [ePay](#)
- [Paybox](#)
- [iDEAL](#)
- [Flo2Cash Web Service](#)
- [Protx Direct Payment Module](#)
- [PagSeguro Payment Module](#)
- [CyberMUT Paiement / Paiement CIC](#)
- [Modu? platnosci.pl dla Lento](#)
- [Payone](#)
- [PaySimple](#)
- [UOS Paymentmodule](#)
- [Fontis Paymate](#)
- [Fontis Payment Express](#)
- [Dibs Payment](#)
- [WorldPay](#)
- [Bank Prepayment](#)
- [VCS](#)
- [Saferpay](#)
- [Datarans Paymentgateway](#)
- [SecureTrading XPay Module](#)
- [Ogone Payment](#)
- [Payment Module: ECS Clearing Interface](#)
- [Modulo Banca Sella 1.1](#)
- [Magento LinkPoint API Payment Module](#)
- [LinkPoint API Payment Module for Magento](#)
- [Fontis SecurePay](#)
- [Cash On Delivery](#)
- [PSiGate API Payment Module](#)
- [Nedbank iVeri Paygate](#)
- [PayPoint.net \(SECPay\) Payment Gateway Module](#)
- [Metodo de pago Servired](#)
- [Servired](#)
- [2Checkout.com payment module](#)

- [Fast Transact Integration module](#)
- [BBS Payment Gateway](#)
- [Fontis Westpac PayWay](#)
- [Fontis NAB Transact](#)
- [Barclays ePDQ](#)
- [ClickandBuy](#)
- [Optimal Payments Firepay](#)

Blank Theme available through MagentoConnect

<http://www.magentocommerce.com/blog/comments/blank-theme-available-through-magentoconnect/>



The 'Blank Theme' for Magento is available via [MagentoConnect](#). With 'Blank Theme' the Magento team answers the call of numerous community members out there who's been inquiring about an extra light theme to use as a basis for theme customization. With this theme we've removed all the custom CSS formerly used in Default and Modern Theme, and replaced them with minor typography and positioning CSS assignments. References to the class names were left in the CSS files to provide a solid starting ground for designers/developers. Markups were also revisited and cleaned up in theme-level.

[Download](#), customize, and enjoy the new ultra light Magento theme!

OpenERP Integration Available Via Magento Connect

<http://www.magentocommerce.com/blog/comments/openerp-integration-available-via-magento-connect/>



[OpenERP](#), the popular Open Source enterprise management software is now integrated with [Magento](#) via the [Smile OpenERP Synchro Extension](#).

[Get the Extension Key here](#)

Magento in 60 languages!

<http://www.magentocommerce.com/blog/comments/magento-in-60-languages-1/>

First, we'd like to thank our language moderators who have done a fantastic job so far. Thank you so much!!

Translating Magento into a locale is not an easy task: we currently have 15,000 words, 4300 strings, 44 modules, 30 emails and 2 interfaces. Despite that, 70 bold and dedicated people, from all over the world, decided to take this challenge on. These amazing people are working days (and mostly nights) to translate Magento into their language and make it available for the community.

Magento is now being translated into 60(!) different locales.

We have 23 locales that are 85% - 100% complete

[Chinese Simplified \(China\)](#) - [Hisea](#) and [JoeCai](#)
[Chinese Traditional \(Taiwan\)](#) - [magnetox](#) and [mihu](#)
[Czech \(Czech Republic\)](#) - [iguru](#)
[Dutch \(Netherlands\)](#) - [ActusMedia](#), [Chantal](#) and [Greatmedia](#)
[English \(Australian\)](#) - [Chris Norton](#)
[French \(Canada\)](#) - [pictogram](#)
[French \(France\)](#) - [SeL](#)
[German \(Germany\)](#) - [Rico Neitzel](#)
[German \(Switzerland\)](#) - [Unic](#)
[Greek \(Greece\)](#) - [nikosrf](#)
[Hungarian \(Hungary\)](#) - [Pro Digital](#)
[Italian \(Italy\)](#) - [Black Cat](#)
[Lithuanian \(Lithuania\)](#) - [Rimantas](#)
[Polish \(Poland\)](#) - [A.Piotrowski\(Lento.pl\)](#) and [piotrekaminski](#)
[Portuguese \(Brazil\)](#) - [Marcelo Minholi](#)
[Portuguese \(Portugal\)](#) - [mascker](#)
[Slovak \(Slovakia\)](#) - [Caleydon](#)
[Spanish \(Colombia\)](#) - [carsal](#)
[Spanish \(Mexico\)](#) - [Josue4ever](#)
[Spanish \(Spain\)](#) - [Andrea](#), [Argentina](#)
[Swedish \(Sweden\)](#) - [Amin Amini](#) and [Drezzzer](#)
[Thai \(Thailand\)](#) - [Rubymoon](#) and [NineKrit](#)
[Turkish \(Turkey\)](#) - [Dincer](#), [memin](#), and [hidonet](#)

13 locales that are actively translated (20% - 85%)

[Croatian \(Croatia\)](#) - [Predrag](#)
[Danish \(Denmark\)](#) - [runez](#)
[Estonian \(Estonia\)](#) - [salo](#)
[Japanese \(Japan\)](#) - [CKD](#)
[Norwegian Bokmal \(Norway\)](#) - [aFFi](#) and [Egil Aslagsen](#)
[Romanian \(Romania\)](#) - [petertufan](#) and [HWaveBlue](#)
[Russian \(Russia\)](#) - [Dmitry A Nikolaev](#)
[Ukrainian \(Ukraine\)](#) - [impulsis](#)
[Bulgarian \(Bulgaria\)](#) - [Miro a.k.a. SecretR](#)
[English \(United Kingdom\)](#) - [GavinPearce](#) and [Very Clever Stuff](#)

[Macedonian \(Macedonia\) - Proleter](#)
[Persian \(Iran\) - hadi "IMP" farnoud](#)
[Latvian \(Latvia\) - eddyf](#) and [Aleksander Andrijenko](#)

And 24 more locales that are in the process:

[Albanian \(Albania\)](#), [Arabic \(Egypt\)](#), [Arabic \(Kuwait\)](#), [Arabic \(Morocco\)](#), [Arabic \(Saudi Arabia\)](#), [Basque \(Basque\)](#), [Bengali \(Bangladesh\)](#), [Bosnian \(Bosnia\)](#), [Catalan \(Catalonia\)](#), [Filipino \(Philippines\)](#), [Finnish \(Finland\)](#), [German \(Austria\)](#), [Hebrew \(Israel\)](#), [Icelandic \(Iceland\)](#), [Indonesian \(Indonesia\)](#), [Malaysian \(Malaysia\)](#), [Mongolian \(Mongolia\)](#), [Norwegian Nynorsk \(Norway\)](#), [Serbian \(Serbia\)](#), [Spanish \(Argentina\)](#), [Vietnamese \(Vietnam\)](#), [Welsh \(United Kingdom\)](#), [Korean \(South Korea\)](#), [Slovenian \(Slovenia\)](#).

To check out the translations status please visit the Magento translation page at <http://www.magentocommerce.com/langs>

If you don't see your language on the list, or if you like to get involved with existing languages, please [get in touch](#).

Magento Connect: Quickbooks Integration via T-HUB

<http://www.magentocommerce.com/blog/comments/magento-connect-quickbooks-integration-via-t-hub/>



A new commercial listing is on [Magento Connect](#) integrating [Magento with Quickbooks via T-Hub](#).

T-HUB provides a QuickBooks integrated Order Manager solution for merchants using Magento platform. Using T-HUB, merchants can directly download orders from Magento, ship using UPS, FedEx, USPS and post transactions to QuickBooks.

[T-HUB for QuickBooks](#) by [atandra](#)

Video: Magento via iPhone

<http://www.magentocommerce.com/extension/303/iphone-theme>

<http://www.magentocommerce.com/blog/comments/mcommerce-magento-iphone-optimized-theme-available-via-magento-connect/>

<http://www.magentocommerce.com/blog/comments/video-magento-via-iphone/>

Wordpress Integration Extension Available Via Magento Connect



[Wordpress](#), the popular open source blogging tool, can now be integrated with Magento via [Lazymonks Wordpress Integration Extension](#).

[Get the Extension Key here](#)

Video: Custom Product Options in Magento 1.1

<http://www.magentocommerce.com/blog/comments/video-custom-product-options-in-magento-11/>



Magento's upcoming 1.1 release will feature significant feature enhancements, including Custom Production Options (CPO) as seen in the [video](#). CPO has been a widely requested feature by the community and we are excited to make it available as part of the upcoming release (expected by early July).

[In this video](#), we show...

- The new Custom Options tab in the simple product view.
- Adding new options for color (drop-down), size (drop-down), and monogram (text field). Available input types include: text field, text area, drop-down, radio buttons, checkbox and multi-select.
- Adding \$15 for the Monogram option.
- Viewing the updated simple product on the frontend with the Custom Options available for the customer to select.

Please note that the CPO module is still in development and minor changes are expected prior to the release. Magento 1.1 public alpha is currently [available via SVN](#).

Magento PHPDocs Now Available

<http://www.magentocommerce.com/blog/comments/magento-phpdoc-now-available/>

We are happy to announce that Magento PHPDocs are now available.

Magento PHPDocs can be found here: <http://docs.magentocommerce.com/>

Varien Lib PHPDocs are also available and can be found here:
<http://docs.magentocommerce.com/Varien/index.html>

We hope you find it useful.

osCommerce Migration Tool - Now Available

<http://www.magentocommerce.com/extension/114/os-commerce-import>

[osCommerce Migration Tool](#) is now available through [MagentoConnect](#).

The new version includes:

- Support for customer accounts with multiple addresses
- Conversion of different encodings to default utf8
- Support for products' special prices
- Support for products' images
- Conversion for timezones
- Fixes for known issues of previous release

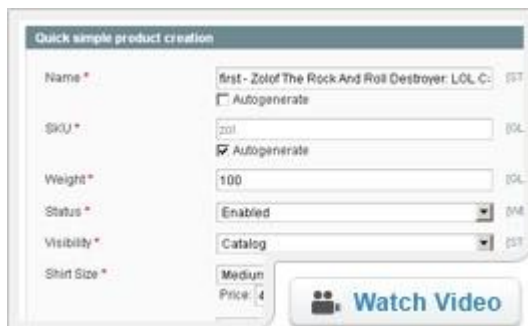
Building Configurable Products Fast

In version 1.0 we added a new way of adding simple products to configurable products. The new method is a quick and easy way of creating new simple products with only minimal data entry.


We have created a video that demonstrates this and which can be seen [here](#).

(Of course you can always use Magento's import product tool to enter the simple products and then associate them to a configurable product with a few clicks.)

We hope you enjoy this new feature.



Quick simple product creation	
Name *	first - Zolof The Rock And Roll Destroyer LOL C: (ST) <input type="checkbox"/> Autogenerate
SKU *	zoi (SL) <input checked="" type="checkbox"/> Autogenerate
Weight *	100 (SL)
Status *	Enabled (MS)
Visibility *	Catalog (ST)
Shirt Size *	Medium
Price	4



Magento / Drupal Integration Project

<http://www.magentocommerce.com/blog/comments/magento-drupal-integration-project/>



As part of [Google Summer of Code '08](#), a [Magento / Drupal](#) integration project is getting off the ground. [A detailed proposal](#) has been posted on the [Groups.Drupal](#) site, and the abstract is included below:

Magento has just released production ready version 1.0 and has comprehensive support ecosystem for its users. Current existing ecommerce modules for Drupal either do not have production ready release or do not support Drupal 6. Hence, integration between Drupal and Magento will be the powerful combination and will drive embracing of Drupal 6 even more.

This project aims to tightly integrate Magento into Drupal by coming up with a Magento module and a few essential native Magento blocks. The integration is elaborate upto the level of rendering Magento as a Drupal page.

There has been a lot of interest by the Magento and Drupal communities for such an integration to take place. This is certainly going to be picking up a lot of steam in the weeks ahead.

-End of Doc

Thanks from TheUnical Technologies