Table of Contents

System requirements

OpenCart requires certain technical requirements to be met for the store to operate properly. First, a web server must be created to make the OpenCart store publicly available on the web. Domain names and hosting services can easily be purchased for an affordable price.

When selecting a hosting service, you should check to see that these server requirements are provided and installed on their web servers:

- Web Server (preferably Apache)
- PHP (at least 5.2)
- MySQL
- Curl

These extensions must be enabled for OpenCart to install properly on the web server.

You can download the ZIP file containing the complete package from our downloads page.
If you are a developer and would like to obtain the source code including unit test scripts, this is available via GitHub

- Web Server (Apache suggested)
- PHP (at least 5.2)
- Database (MySQLi suggested)

Required PHP libraries / modules

- Curl
- ZIP
- Zlib
- GD Library
- Mcrypt
- Mbstrings

The above PHP extensions should be available by almost all hosting providers, during the install processa it will check you have them all enabled. You should contact your hosting provider if one is missing.

User Guide

Skip to end of metadata

Attachments:2
Added by Catherine [HostJars], last edited by Jennifer [HostJars] on Oct 13, 2013 (view change)

Go to start of metadata

None

10 Child Pages

Page: Getting StartedPage: Browsing the store frontPage: Admin interfacePage: CatalogPage: ExtensionsPage: SalesPage: ReportsPage: SystemPage: HelpPage: Miscellaneous

Contribute
Add to the documentation

Support
Get help from the community

PDF Version
Buy the User Guide PDF

Extensions
Get import tools and modules

Getting Started

Skip to end of metadata

Attachments:7
Added by Justin [HostJars], last edited by Jennifer [HostJars] on Oct 23, 2013 (view change)

Go to start of metadata

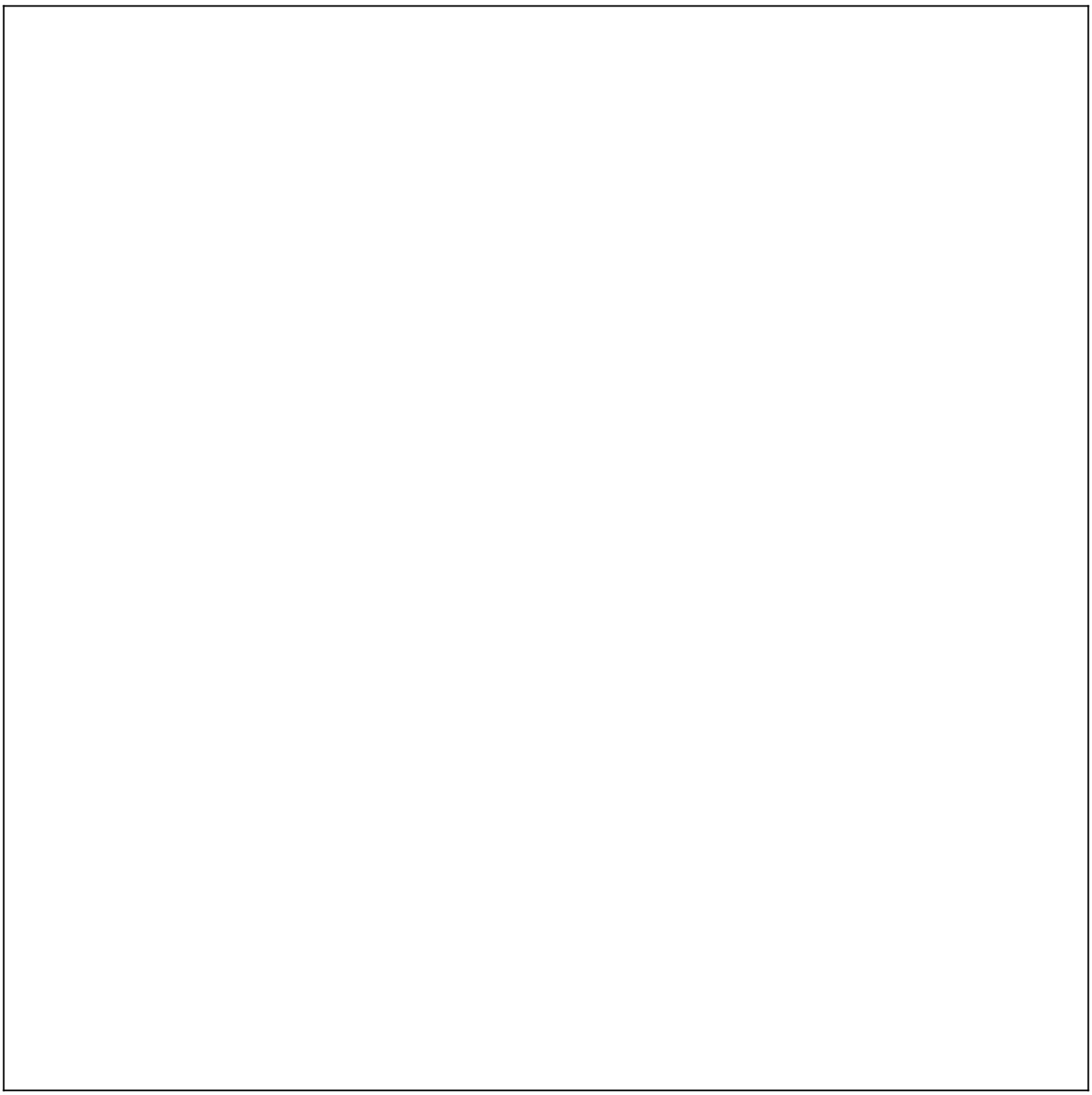
There are a number of steps required after installing OpenCart to get started selling from your online store. This page is a helpful guide to getting started. An OpenCart store is comprised of a Frontend and Admin interface. This guide will primarily focus on Admin interface functions and settings. These settings should be tested on the frontend before your site is launched. Following the steps in this guide will get your store to a ready to use state from installation. This guide assumes that you have admin login credentials and are able to login to your store's Admin interface via <yourdomain.com>/admin, for example: http://demo.hostjars.com/admin.

System Settings

The first thing a new OpenCart store requires is correct settings. Store settings include your store's logo, name, contact details, some tax, stock and product settings, and server settings. For a full coverage of these please see theSettings documentation page. It is recommended that you go through all these settings to make sure they are correct for your store's requirements. This guide will only cover those which will always need to be set for a new OpenCart store. Settings are under the System menu in your OpenCart admin, and are unique to each store in a multistore setup.

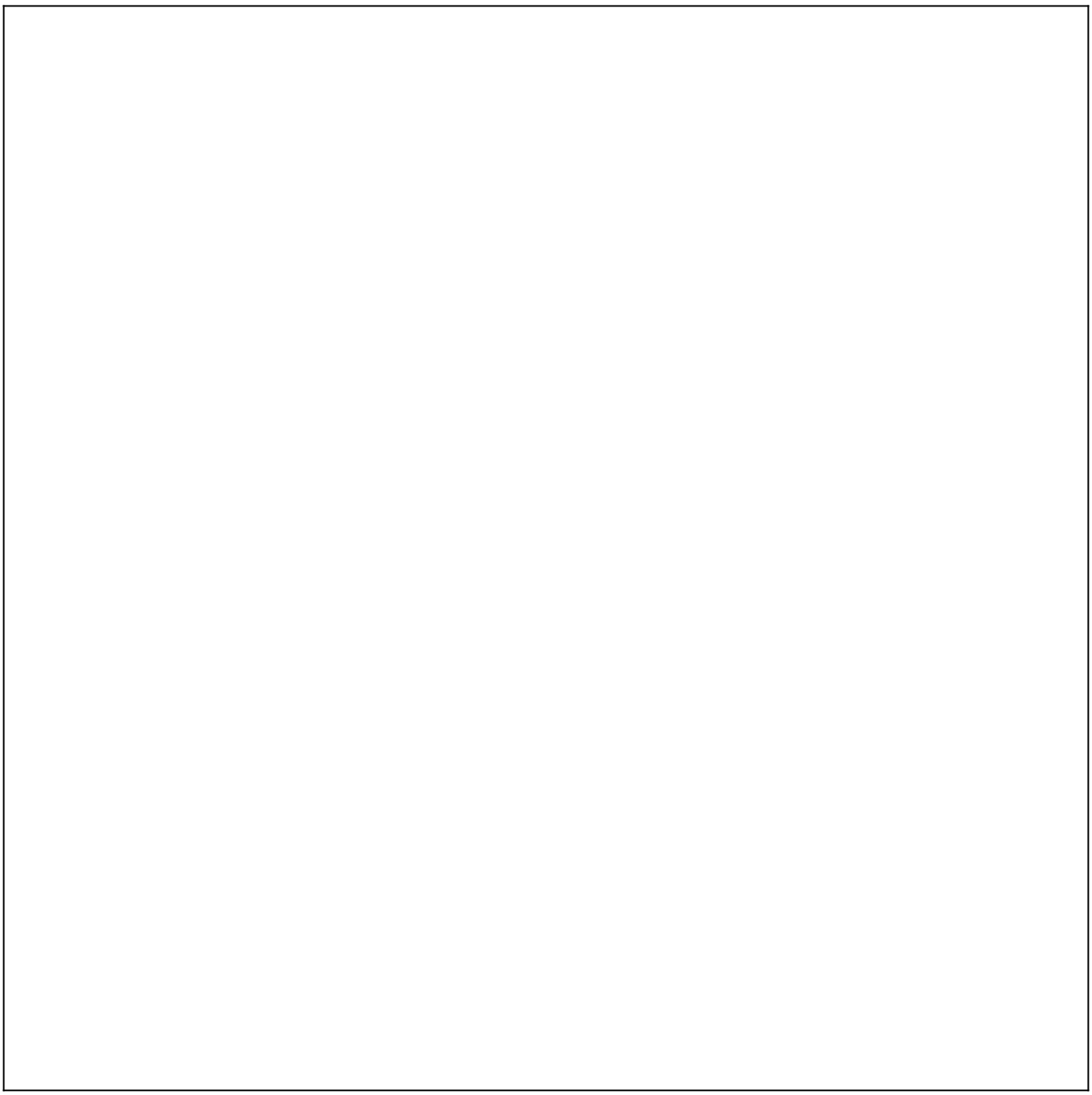
General Settings

Most General Settings are required. Under General Settings you will need to set your store's name, owner, address, email and phone number. These will be used for your store's contact page and order invoices.



Store Settings

Store Settings need to be changed from the default settings of "Your Store", etc., to your actual Store Name and a description of your store for search engines.



Local Settings

Local Settings allow you to choose the Country that your store is operating from. You can also choose the default currency for your store. If your store accepts multiple currencies, you will need to select whether OpenCart should automatically update based on current exchange rates.

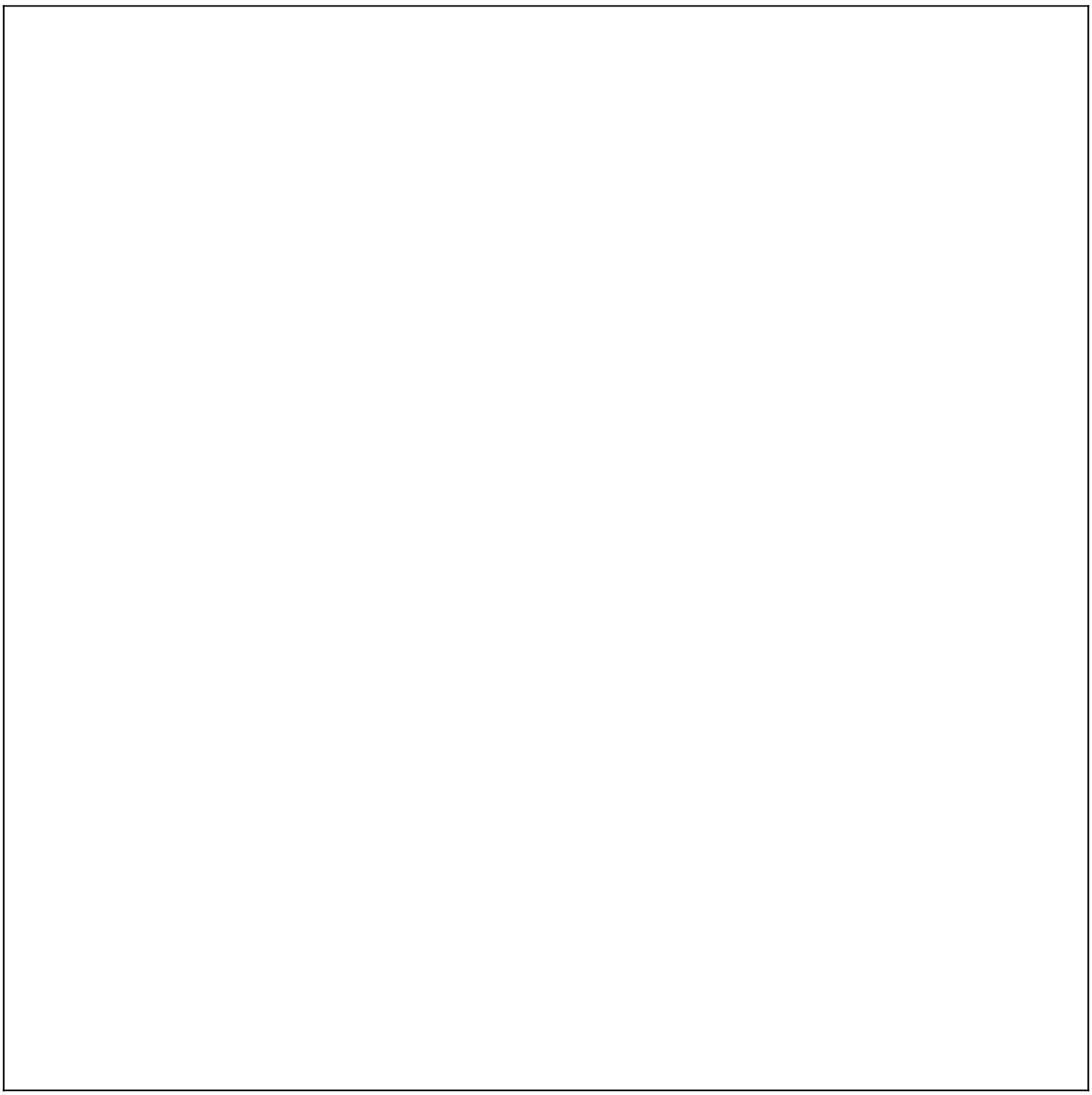
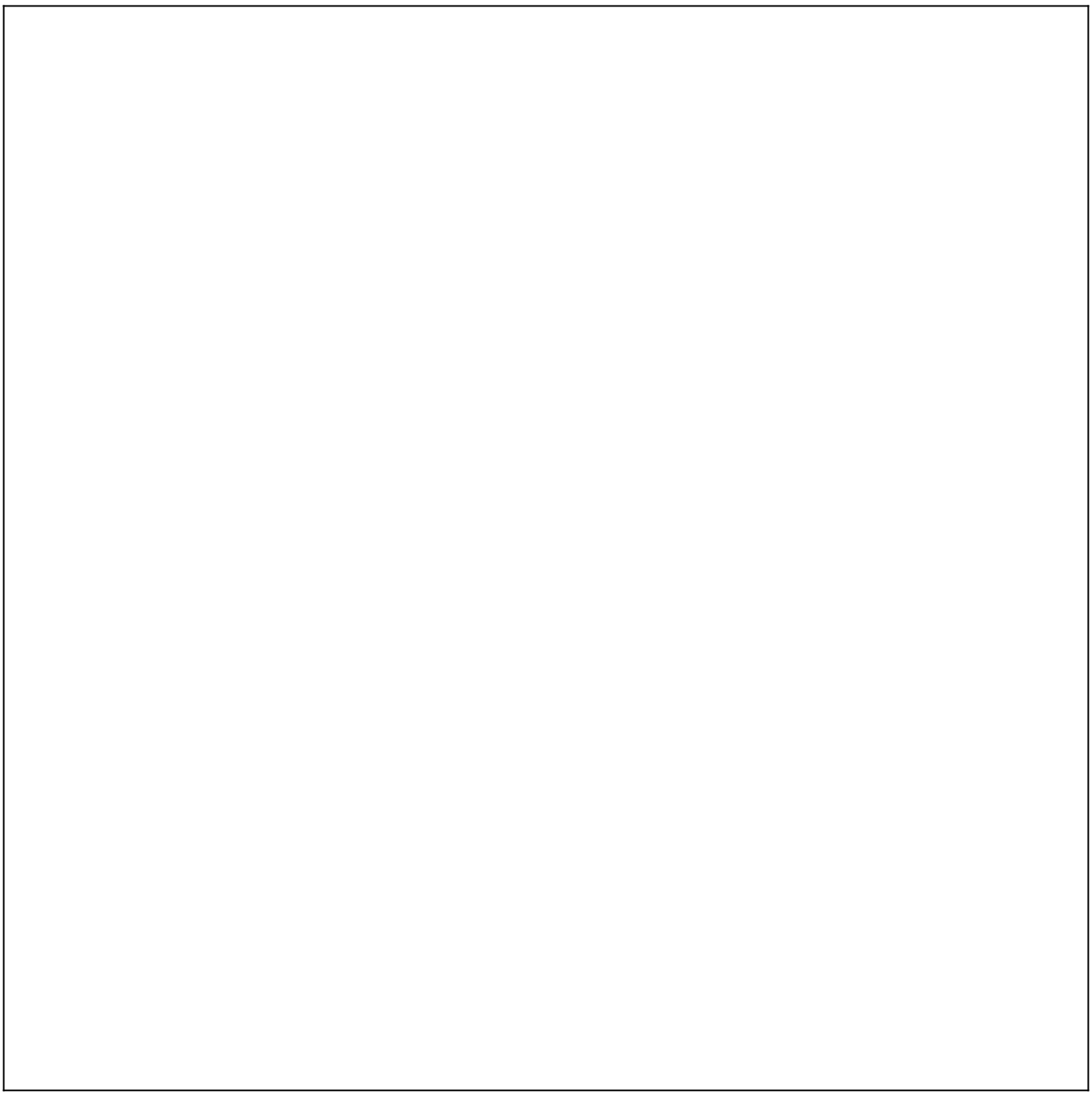


Image Settings

Next, you will need to select a logo for your store and upload it via the Image Settings tab. The logo must be a suitable size and shape for your theme, and should ideally be a jpg or png file. You can also choose a favicon at this point. This is the small image that displays in a browser when you visit a site. Typically, favicons are 16px by 16px, and OpenCart requires a png file for favicons.



Server Settings

There are two important settings under the Server tab of your OpenCart store's System Settings.

1. SEO URLs. If you wish to have search engine optimized URLs in your store then you will need to set this to yes. You will also need to follow the instructions on setting up SEO URLs to make sure these work correctly, but it is an important step for getting the best possible result for your store from search engines.
2. The Display Errors option should always be set to No for a live store. This will prevent your customers seeing any errors in your store's code while using your site. Your developer will still be able to diagnose and repair these issues using the error logs.

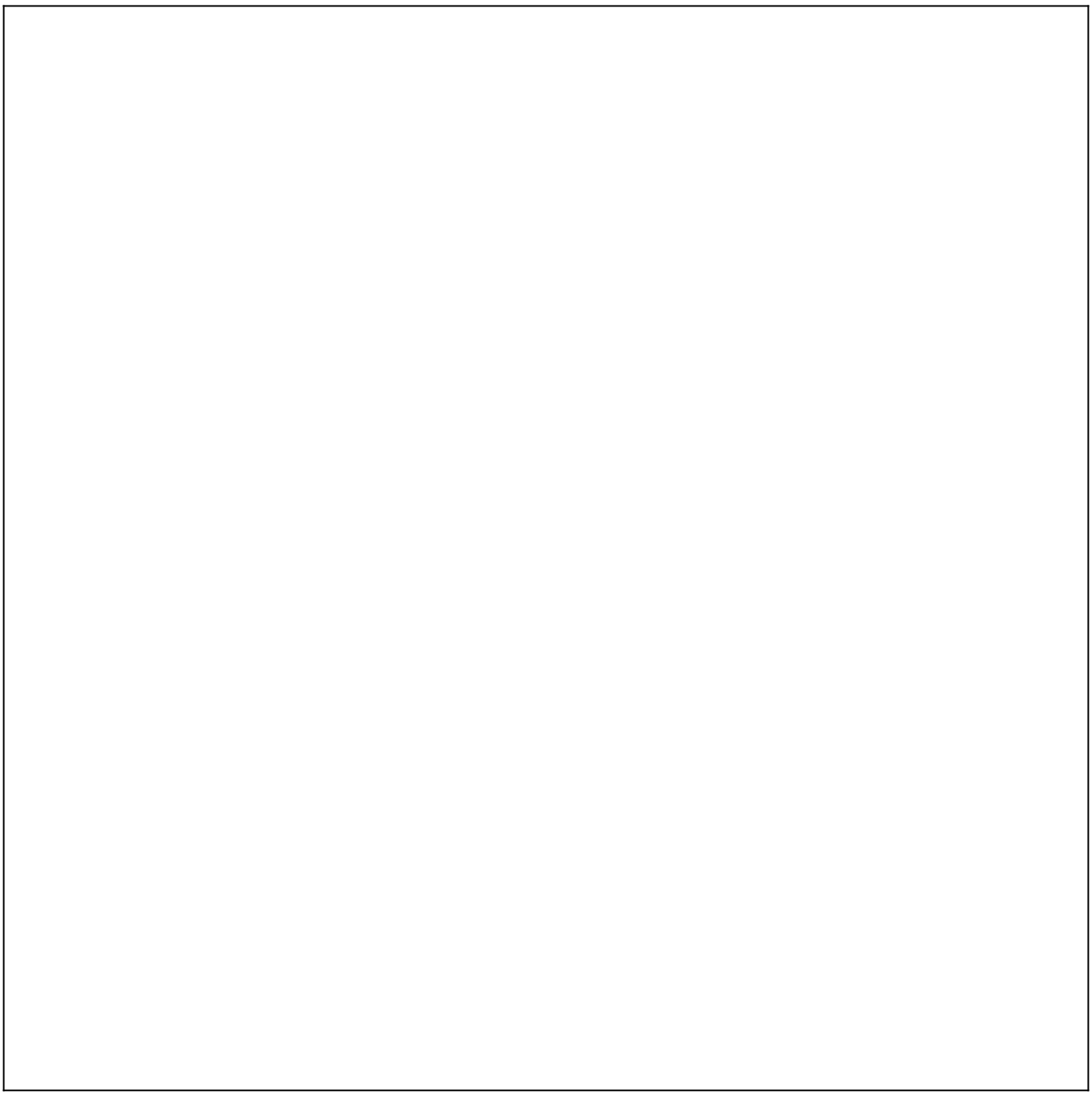
Accepting Payments and Adding Taxes

Once your settings are correct, you will need to select the payment gateway you wish to use. If your store needs to charge taxes on purchases, these should also be setup via your admin interface as described below.

Payment Gateways

OpenCart supports many Payment Gateways out of the box. If your payment gateway is not supported, you can find and install additional gateways from the OpenCart extension store. In order to choose the payment gateway your store will use, go to the Extensions menu, and the Payments submenu. This page will show you a list of all the available payment gateways. Your store can use more than one payment gateway.

In order to enable payment gateways first click the Install link corresponding to the gateway you wish to enable, then click the Edit link to set it up for your payment account. The Order field on each payment gateway's settings allows you to select the order in which payment options are displayed on your checkout page. Lower numbered payment options will always be shown above higher numbered options.



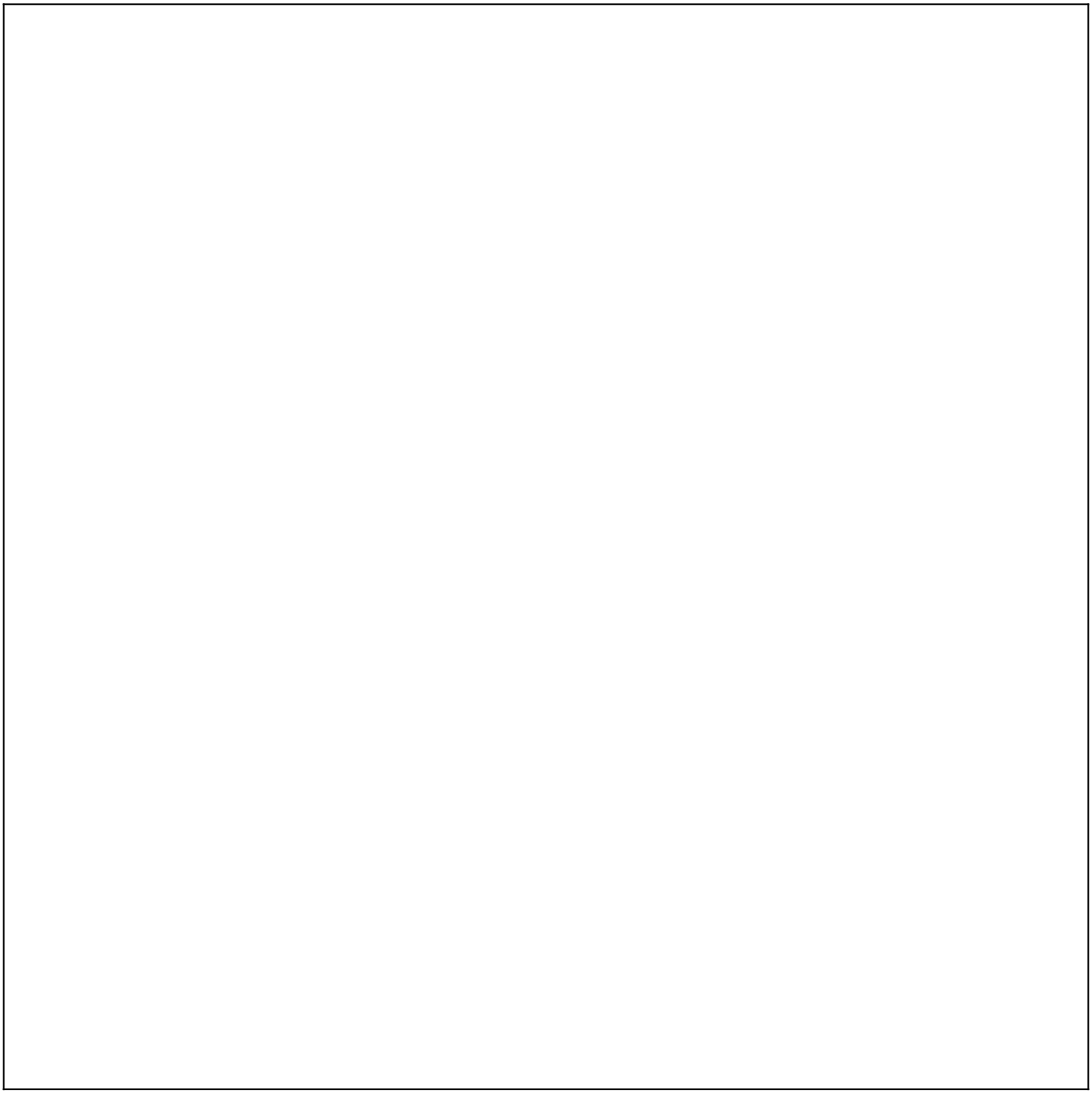
Taxes

OpenCart offers a number of different Tax options by default. Taxes are location dependent, so you can charge different tax rates to customers in different geographical locations, according to local tax rules. Tax rates are set up under System > Localisation > Taxes > Tax Rates. If your required tax rate is not already present, you will need to create a new tax rate and add it. For example, the Tax Rate applicable to New Zealand is GST which is currently at 15%. New Zealand stores should therefore create a new Tax Rate called GST, set to 15%, and applicable to customers in the Geo Zone of New Zealand.

If you have product specific tax rates, you need to define each of these tax rates under the Tax Rates page. You will later choose which tax rate applies to each product, so you will be able to manage your product specific taxes at the product level.

Shipping Method

Stores selling tangible goods will need to edit their shipping options. Shipping methods are selected under the Extensions > Shipping menu item. Just as with the Tax Rates above, you will need to Install and Edit the settings for the Shipping Methods your store will support. The documentation contains full instructions on setting your Shipping Method.



Inventory Management

OpenCart's installation includes demonstration data to help you see how to setup your OpenCart store's inventory. This includes Categories, Manufacturers, Options, Attributes and Products and some home page Banners. These are the Apple iPods that you see when you first visit your store's frontend after installation. In order to get started with your store, you will need to replace these demo items with the actual

categories, manufacturers and products your store will sell. There are two recommended ways to do this:

1. You can manually edit these under the Catalog menu in your OpenCart store's admin.
2. You can use an import tool to simplify the upload of your products in bulk. This is more practical for larger inventories or dropshippers, and will allow you to remove all existing items and replace them at the same time.

Extensions, Modules and Themes

OpenCart functionality, look and feel are all controlled by modules and themes. The final step to get started with OpenCart is to check your Extensions > Modules page to ensure the functionality you want is enabled. The default banners can be modified under System > Banners, or removed via the Slideshow module at this point. You can also choose other modules that you wish to display, and the pages you wish to display them. The OpenCart defaults are sensible and will not necessarily need editing to get started, except for the default home page banners, which are for demonstration purposes only.

If you do not find the functionality you need in your store, you can often add it as a 3rd Party Extension.

None

Browsing the store front

Skip to end of metadata

Go to start of metadata

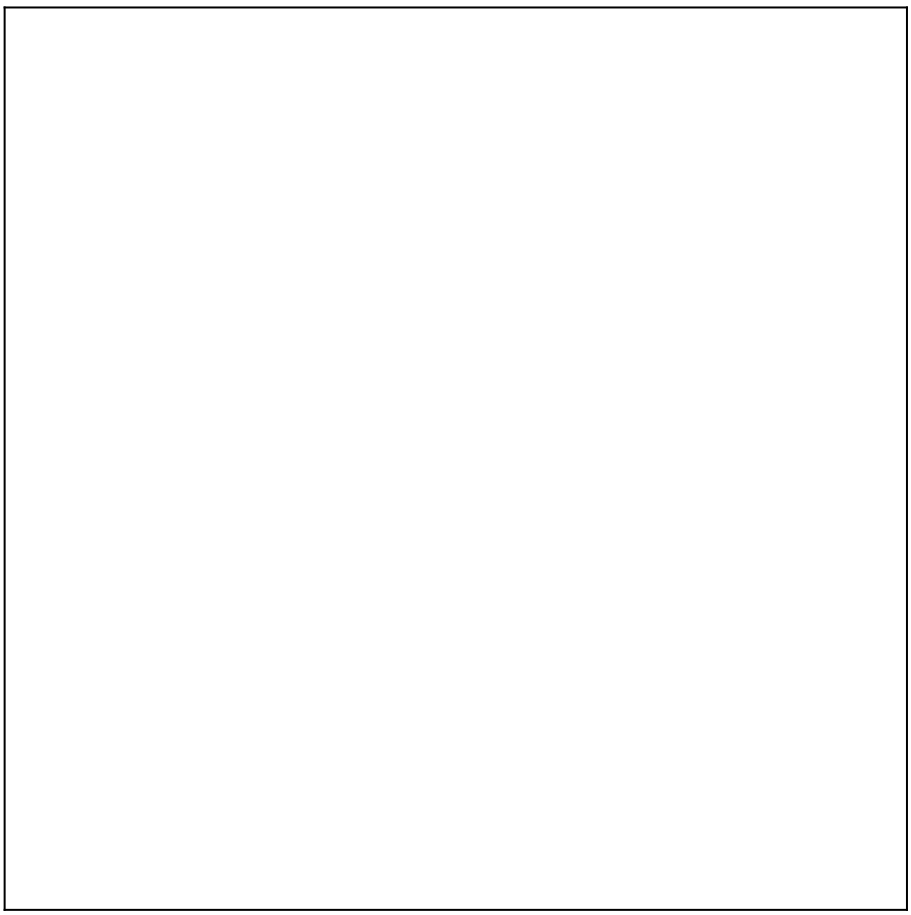
Table of Content

Browsing the store front

This guide is intended to be used as an introduction to the OpenCart default store front. The store front reveals how the customer views and interacts with the store.

The default theme

OpenCart comes with a default theme after a fresh installation:



The products seen above are included as sample data with the OpenCart installation. These products can easily be removed and replaced with the shop's products later.

This guide will cover the basics of browsing the store front with the default theme. You can find an ever expanding list of available themes at the OpenCart Extension Directory.

Navigating the shop

The OpenCart default theme makes navigating a shop's products easily accessible to its customers.

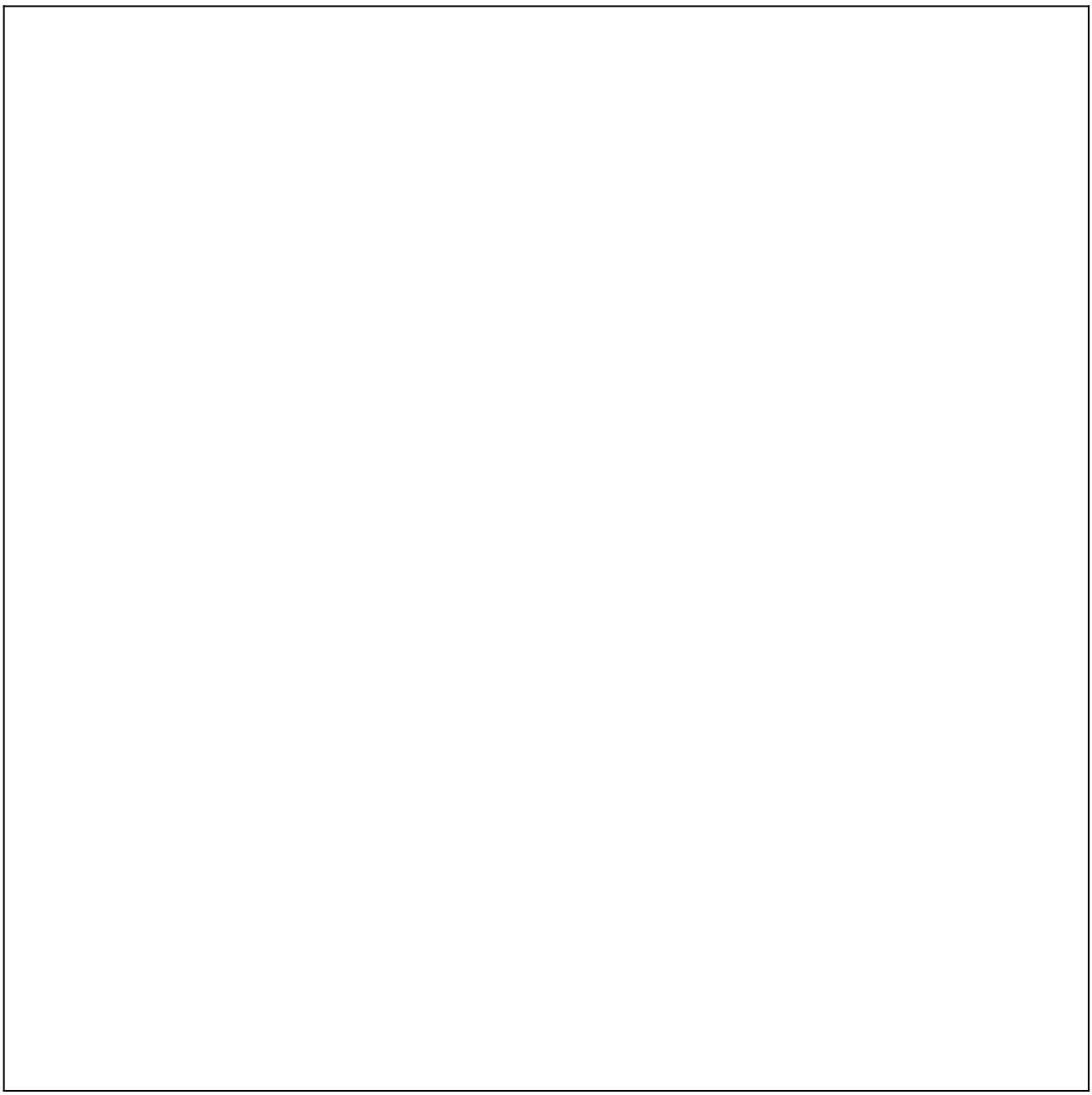
Anatomy of the home page

The home page is arguably the most important page in the shop, in terms of presentation. In most cases, this will be the first page that a customer interacts with (especially if they are directed to the store site from a search engine). The shop's homepage needs to be user-friendly, while at the same time highlighting the shop's products.

The first step in becoming familiar with the store front is understanding the anatomy of the OpenCart default homepage.

The header

The header will be displayed at the top of the page, on every page of the store; not just the home page.



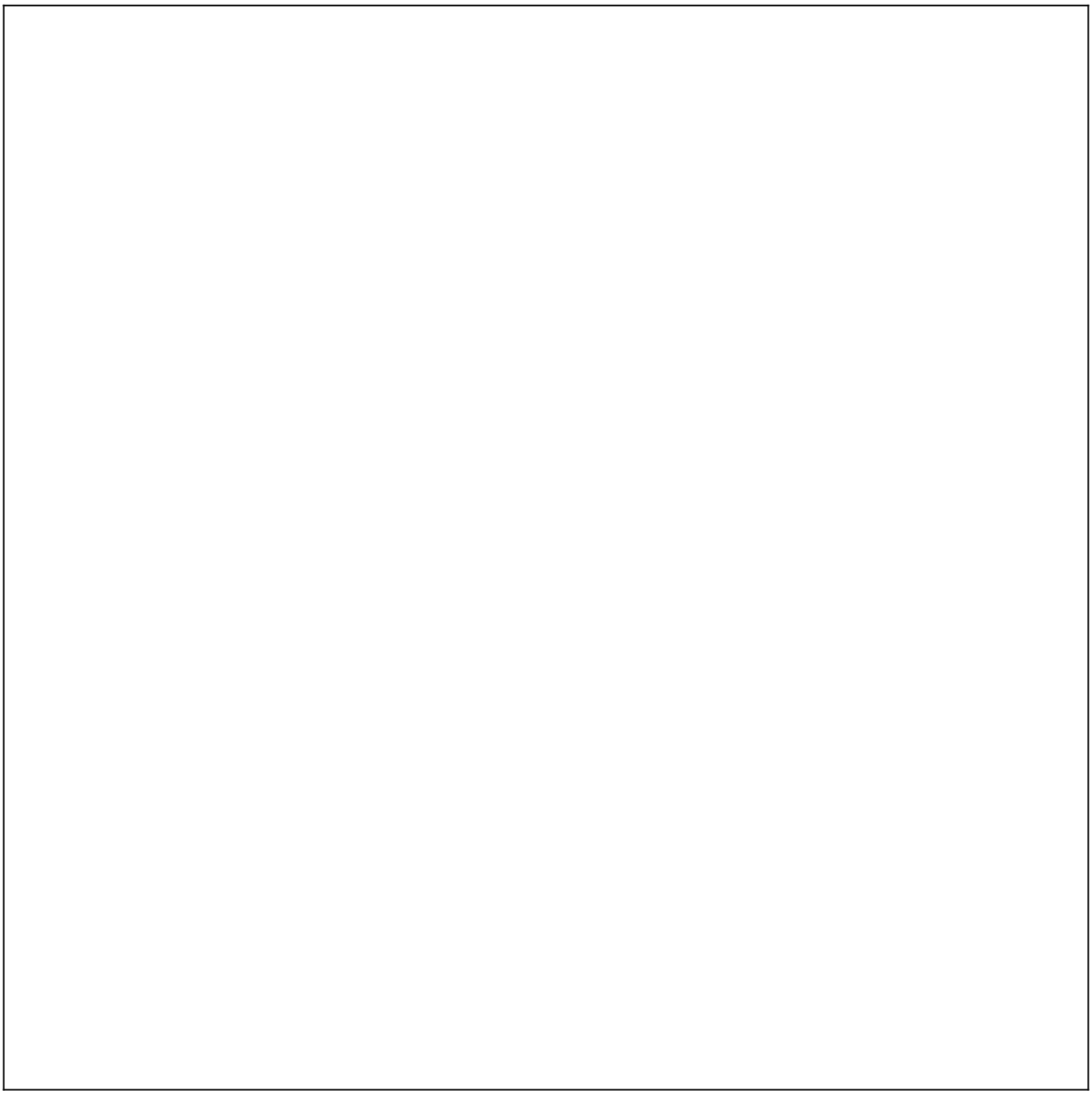
The header has the following navigation options:

- *Store logo*: Clicking on this logo will direct the customer back to the home page of the store.
- *Currency block*: The customer can select which currency the store's products will be in by clicking on any of the currency icons.
- *Shopping Cart:* Displays the number of items purchased, and the total price of the order. Clicking on the icon will display a drop-down box containing all of products added to the cart and an option to "View Cart" or "Checkout".
- *Search box:* The customers can type in the search box to search for a product within the store's product categories.
- *Welcome text:* Welcomes the visitor and gives them an option to "login" or "create an account". When logged in it will display "You are logged in as [first name]" and gives an option to log out.
- *Links:* Links the customer to the Home page, Wish List, My Account, Shopping Cart, and Checkout.

The top menu

The top menu category only displays the top parent categories of products. See Categories for more information on how to create and assign product categories.

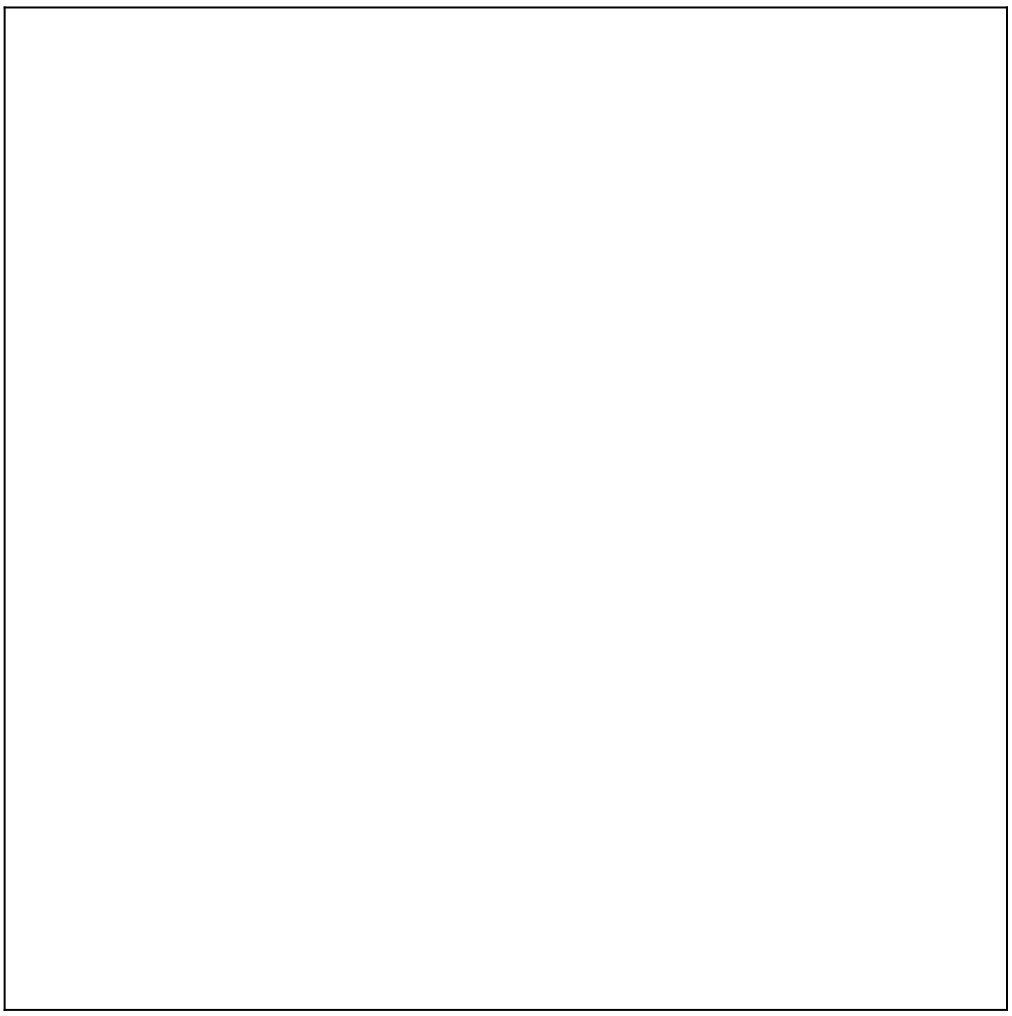
Like the header, the top menu will be displayed on every page. When the customer's mouse is dragged over a category, a drop down menu will display the sub-categories for that parent category.



When a parent category is clicked, the customer will be directed to the category page, which displays all the products within that category.

Slideshow

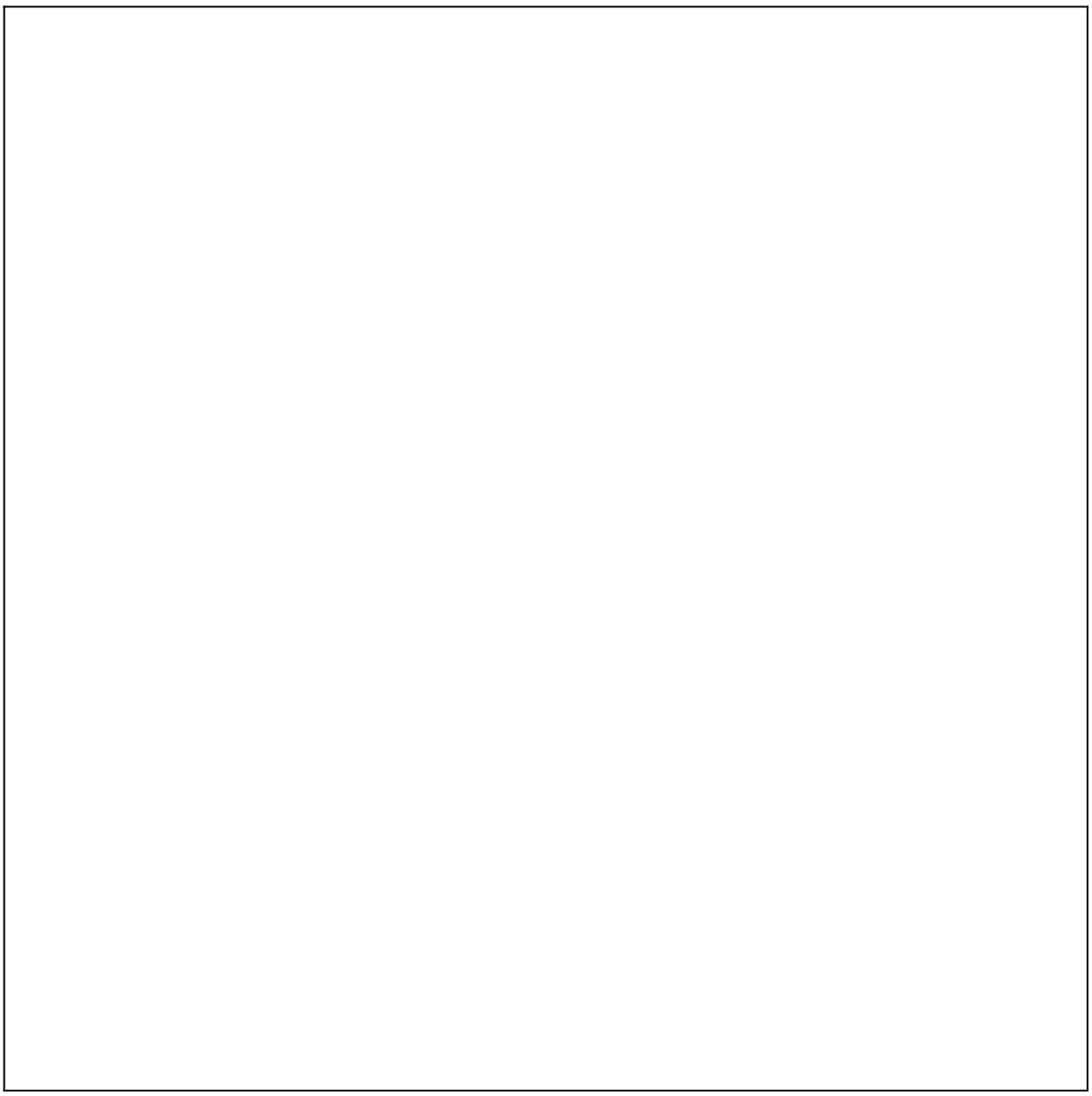
The slideshow displays several product banners of your choice by alternating the images in a slideshow. After a certain amount of time, one banner will shift to the the next banner. Banners in this slideshow are useful for highlighting certain products to be easily accessible by the customer. When the banner is clicked on, the customer will be directed to the product on the banner's page.



Unlike the top menu and header, the slideshow in the OpenCart default can only be viewed on the home page in this position.

Featured products

OpenCart gives you the option of featuring specific products of their choosing on the home page.

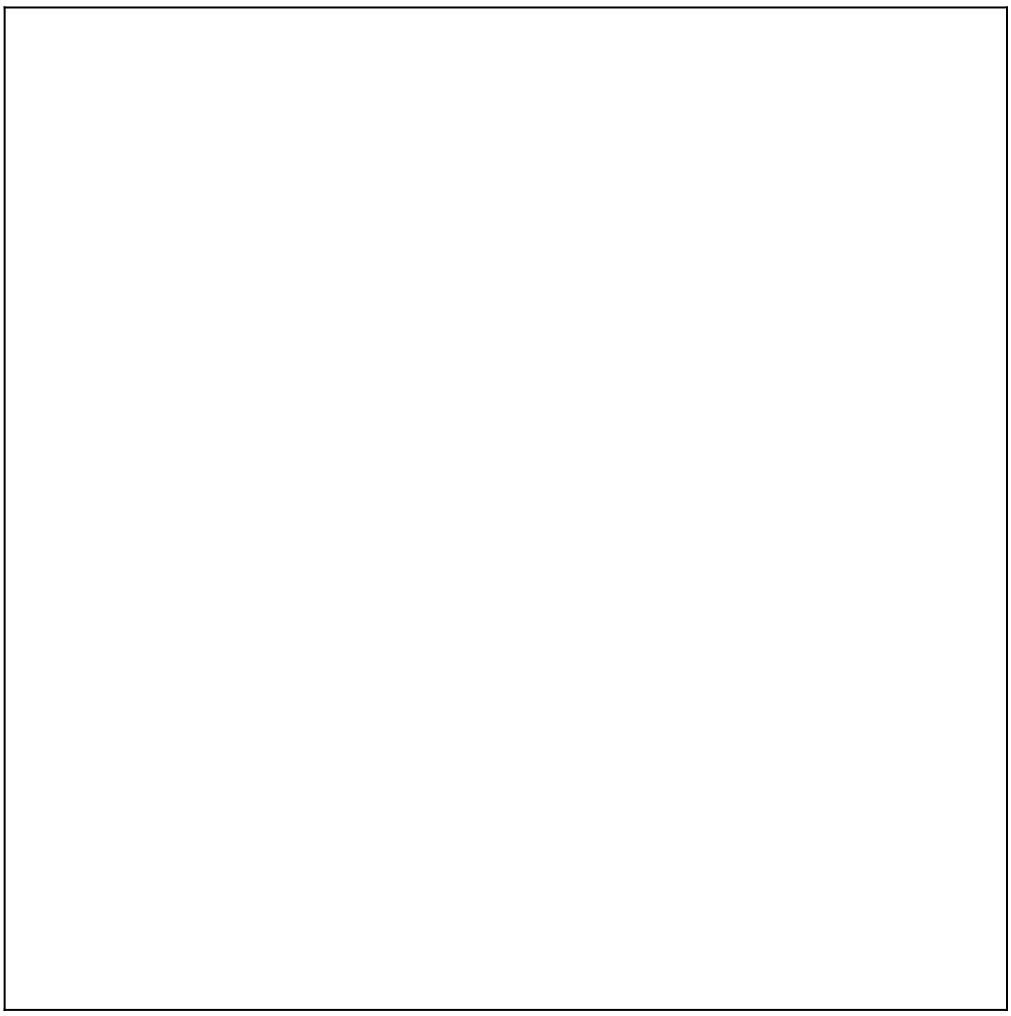


The Featured section includes the product image, name, price; and an option to add the product directly to the Shopping Cart.

The featured products is only located on the Home Page in the default.

Carousel

The carousel in the default theme displays the product manufacturers in the form of image icons. This feature lets customer's browse the manufacturers featured in the shop.



When a manufacturer icon is clicked, the customer will be directed to a page listing all the products within that manufacturer's category.

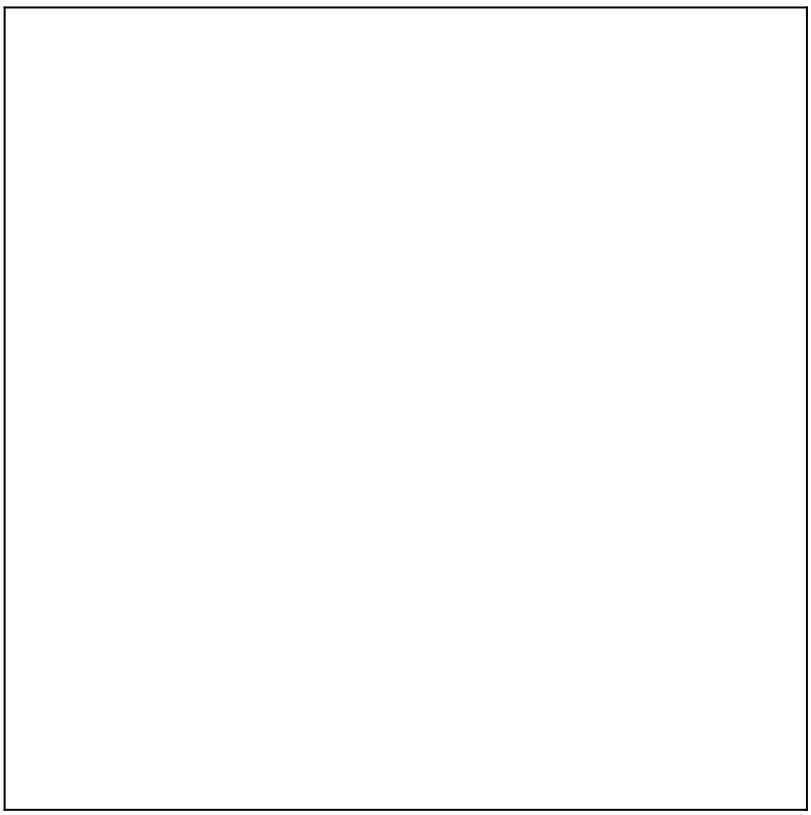
The carousel is only located on the Home Page in the default.

Footer

The footer is located at the bottom of every page, not just the Home Page. This block of miscellaneous links is useful in sorting relevant pages for the customer that may not logically sort anywhere else.

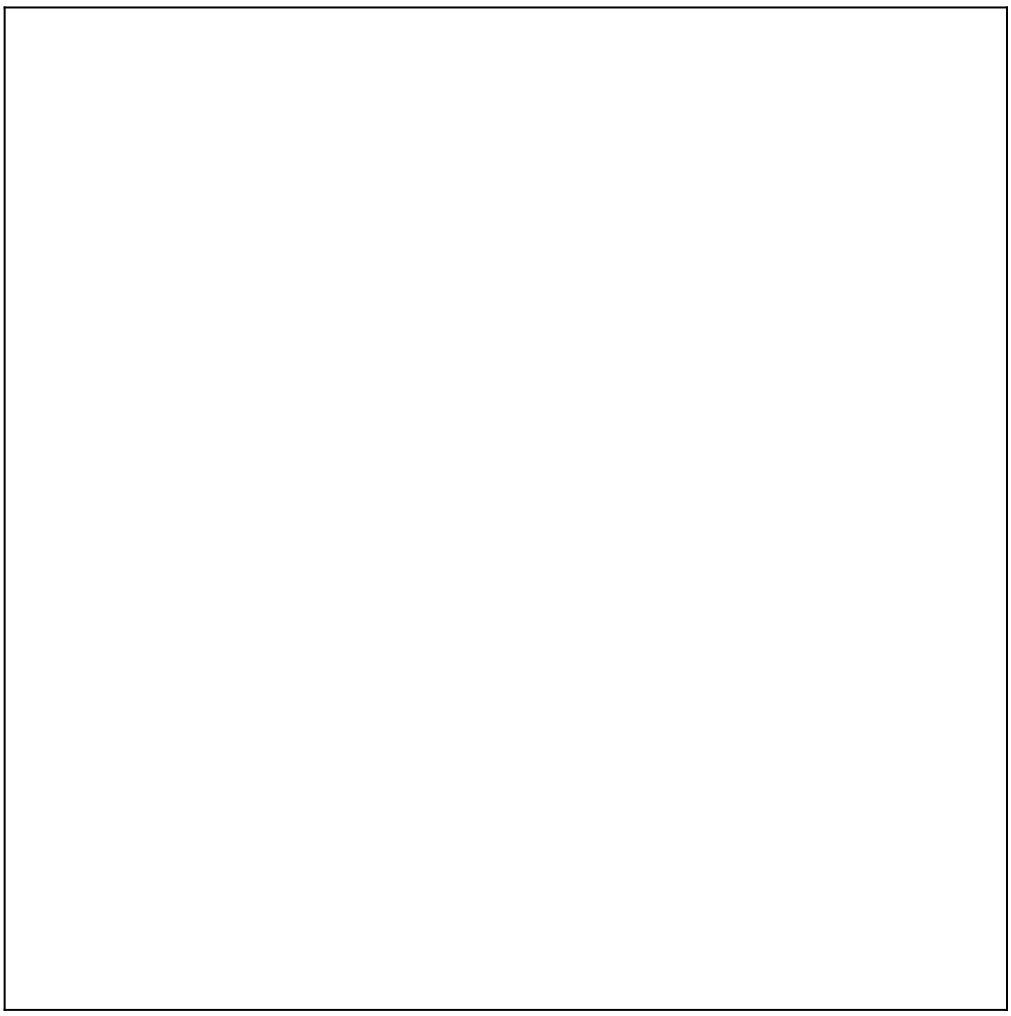
The organizational scheme of the footer can be divided into the following sections:

- **Information:** "About Us", "Delivery Information", "Privacy Policy", "Terms & Conditions"
- **Customer Service:** "Contact Us", "Returns", "Site Map"
- **Extras:** "Brands", "Gift Vouchers", "Affiliates", "Specials"
- **My Account:** "My Account", "Order History", "Wish List", "Newsletter"



Product pages

The OpenCart default product page will follow the structural format seen below (minus header/top menu/footer).

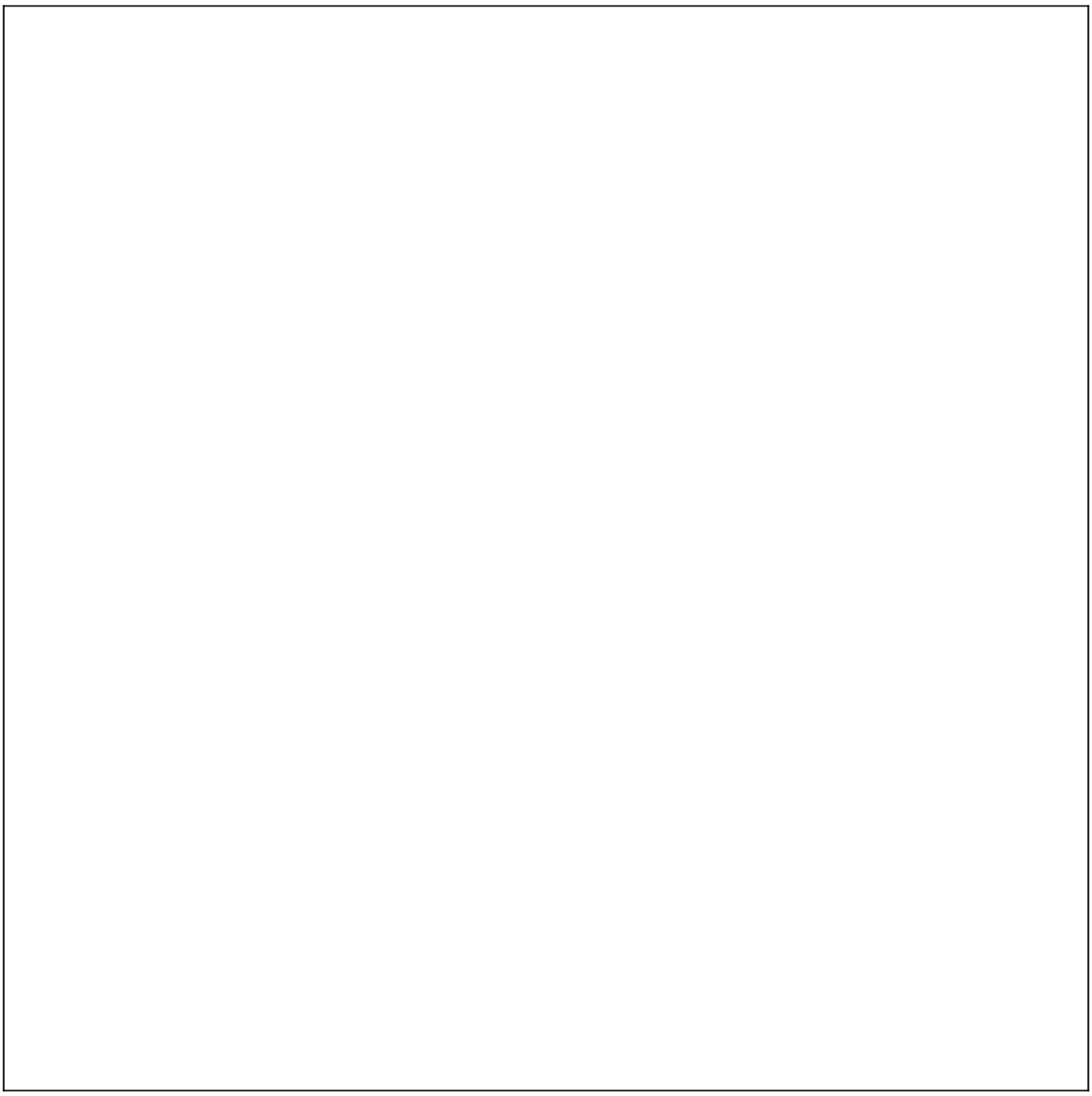


The product page can be divided into the following sections:

- **Categories**: All the products on the shop will be organized into designated categories. The category link block on the left-side of the product page will enable the customer to browse other products by category.
- **Product image**: The product image can be displayed under the title on the left-side, along with alternate views of the product underneath it in smaller box. Clicking on the main image will expand the image within the window for the customer to see it in greater detail.
- **Product details**: The product code, availability, and price are displayed just right to the product image.
- **Cart**: The customer can select a quanity and add the product to their cart, wishlist, or compare.
- **Rating/Sharing:** Underneath the cart can rate the product and/or share the product on different social media websites.
- **Description tab:** An area underneath the main product information to provide a detailed description of the product.
- **Review tab:** An area for the customer to write a review on the product.

Category product listings

Category product listings enable customers to browse products similar to other products within the same category. This is especially helpful for customers looking to compare products, a feature that will be explained under Categories. The category page can be accessed a number of ways. It can be accessed from the top menu, when a customer clicks on one of the parent categories. Also, on product pages a customer can access the category product listing page by clicking on a category on the left side category block.

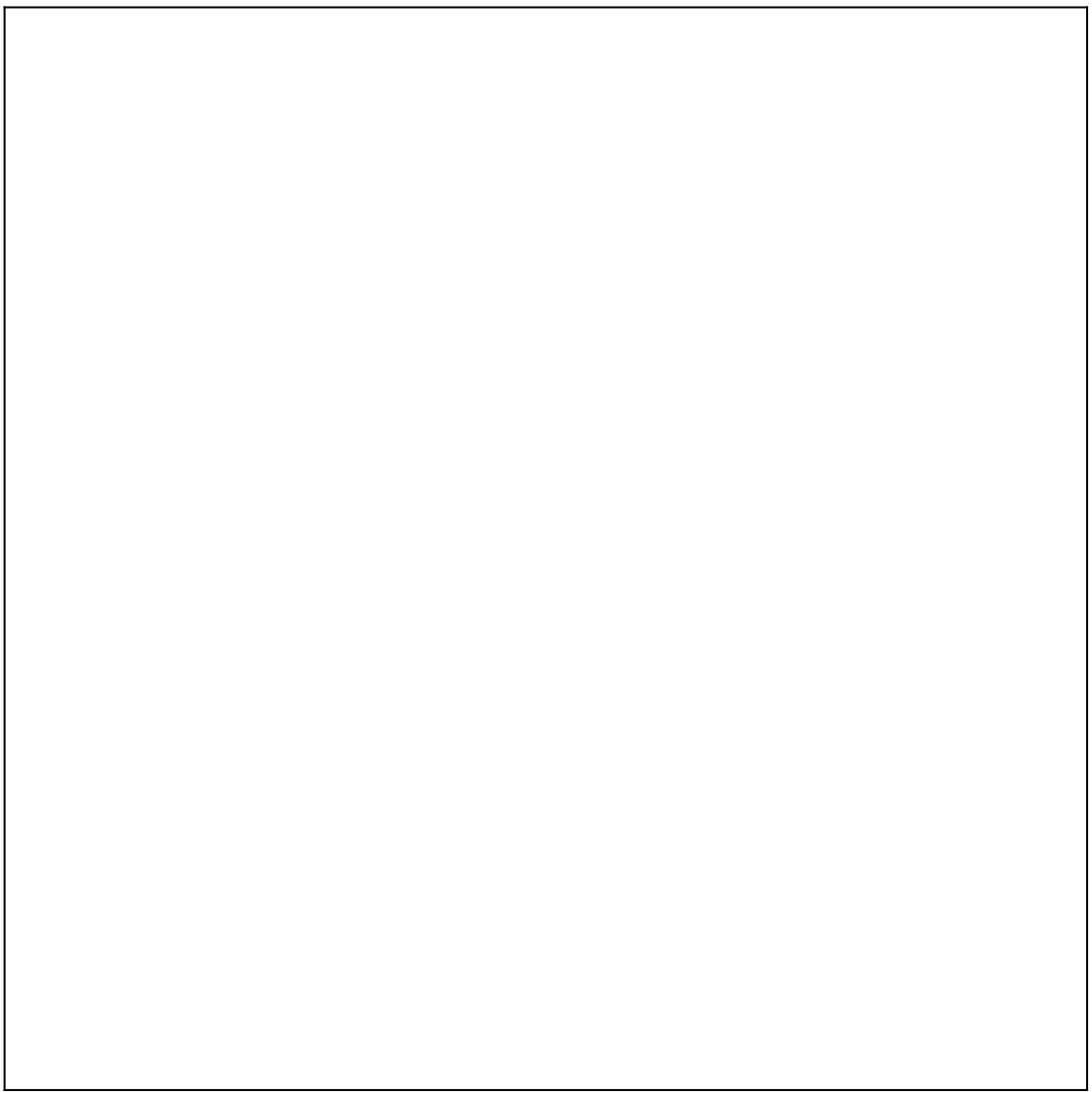


As seen above, the category block is displayed on the left-side like it is in the product page. There is space under the Category title at the top to add a description to the category. The "Refine Search" links to sub-categories of that category for the user to browse. The products can be displayed according to the customer's preference: in a list or grid. The above image is sorted in the listing format. The products can be sorted according to name, price, rating, or model in the "Sort By" box. The number of products displayed in the product listing can be changed in "Show" from 15 up to 100.

There is a section that gives space for each of the products within the category, providing a product image, description, price, and an Add to Cart option. There is an option to add the product to a wish list. Another option for the product is to "Add to Compare".

Product compare

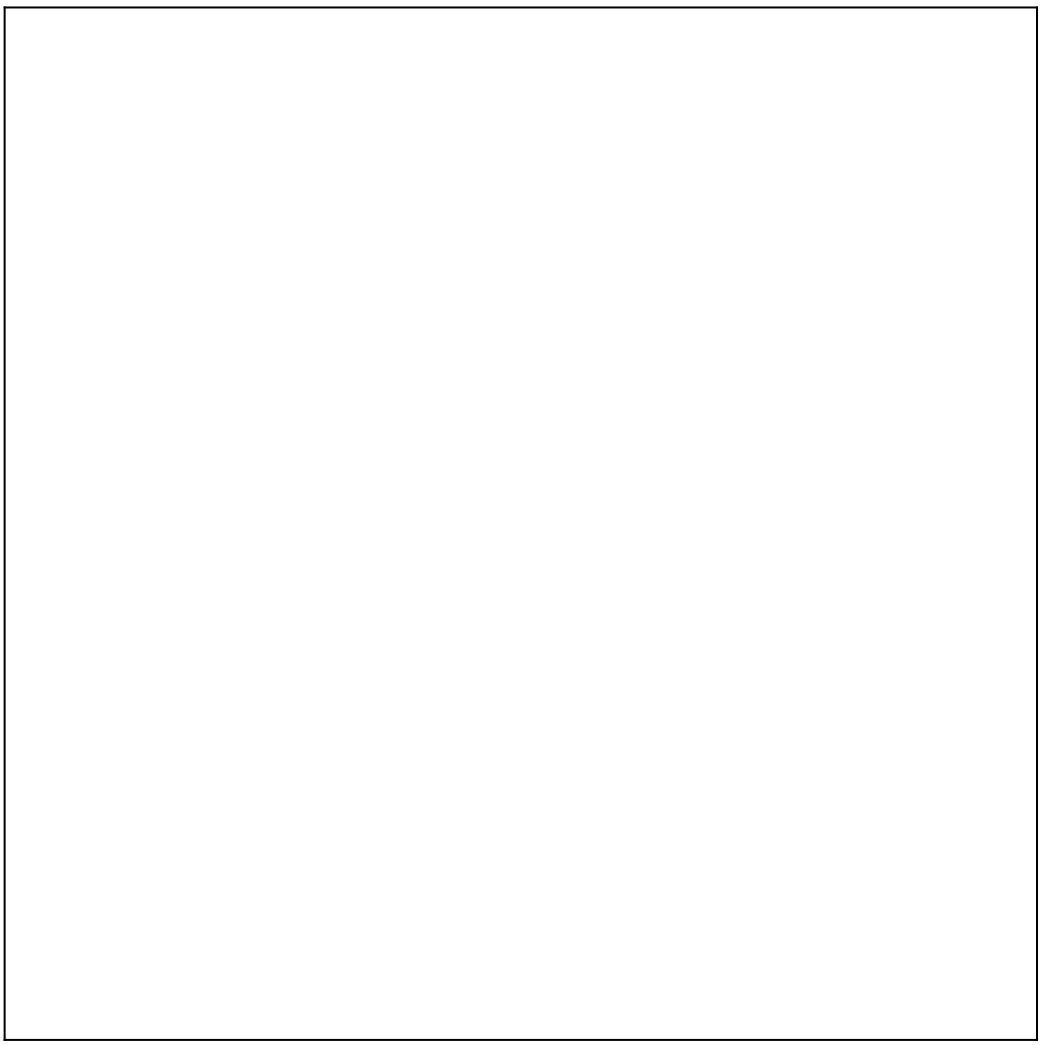
The "Add to Compare" feature in the product section allows the customer to compare the different specifications, features, and price of a number of products s/he might be interested in.



The customer is given the option to add one of the compared products to the cart if they want to. Pressing "Continue" will bring the user back to the home page.

Shopping Cart page

Once a customer adds a product to the cart, they can access the shopping cart in the header under "Shopping Cart".



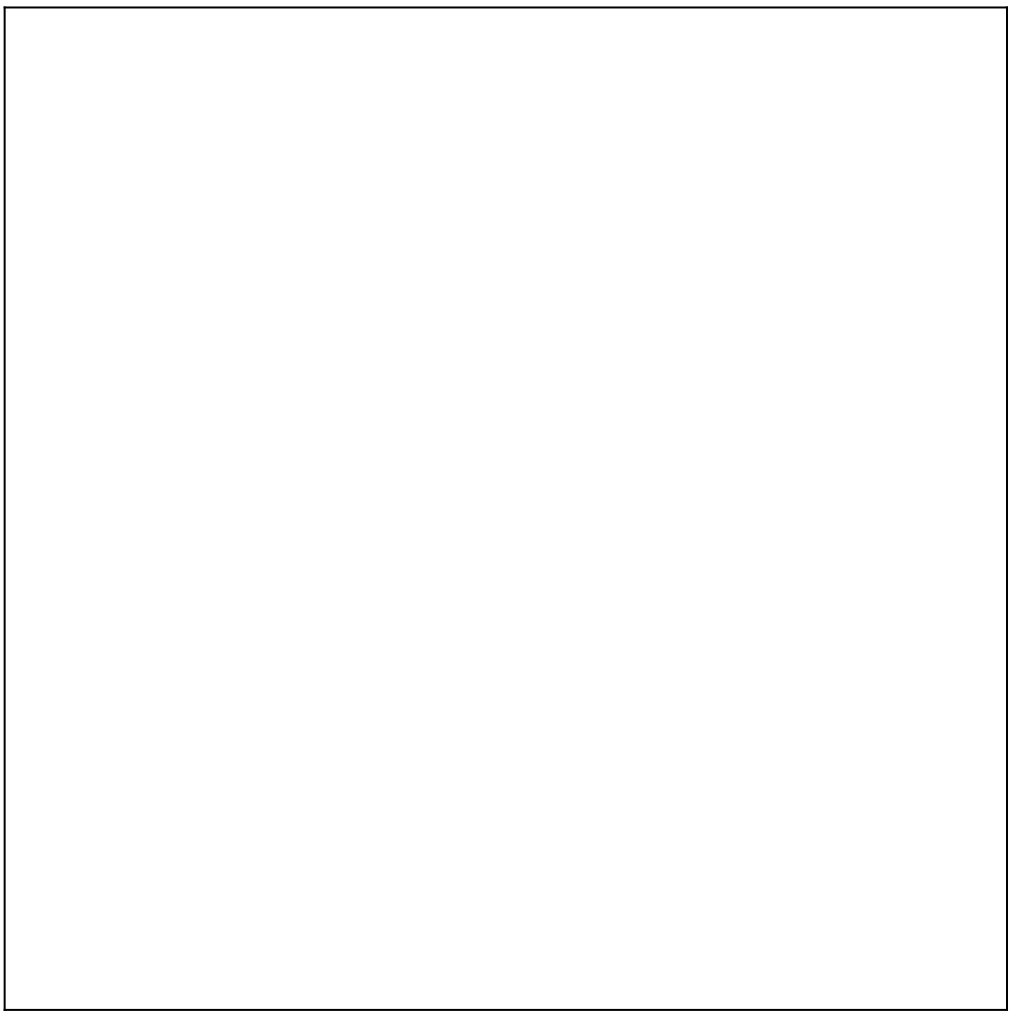
The shopping cart gives an overview of the product selected by including the categories "Image", "Product Name", "Model", "Quantity", "Unit Price", and "Total". The customer has an option to add a coupon code or gift voucher, or estimate shipping & taxes, before heading to the checkout. The "Continue Shopping" button links back to the homepage.

Creating an account

Before a customer can continue checking out a product from the shopping cart, the customer needs to select either guest checkout or log into their account. The guest checkout doesn't require log-in details. Returning customers may want to make an account with the store.

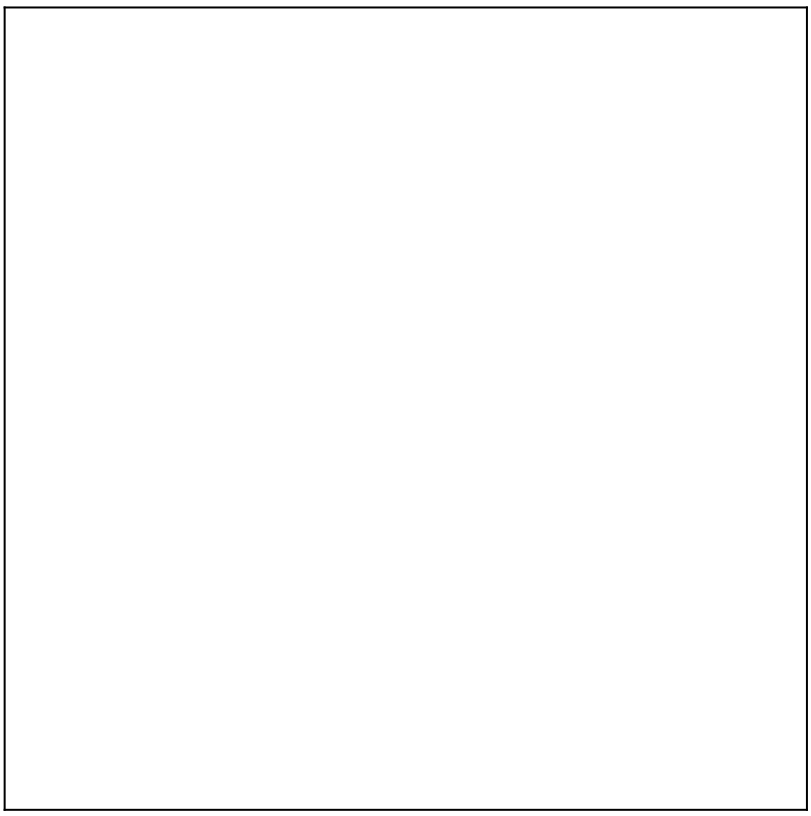
There are a few ways a customer can make an account:

1. **Checkout**



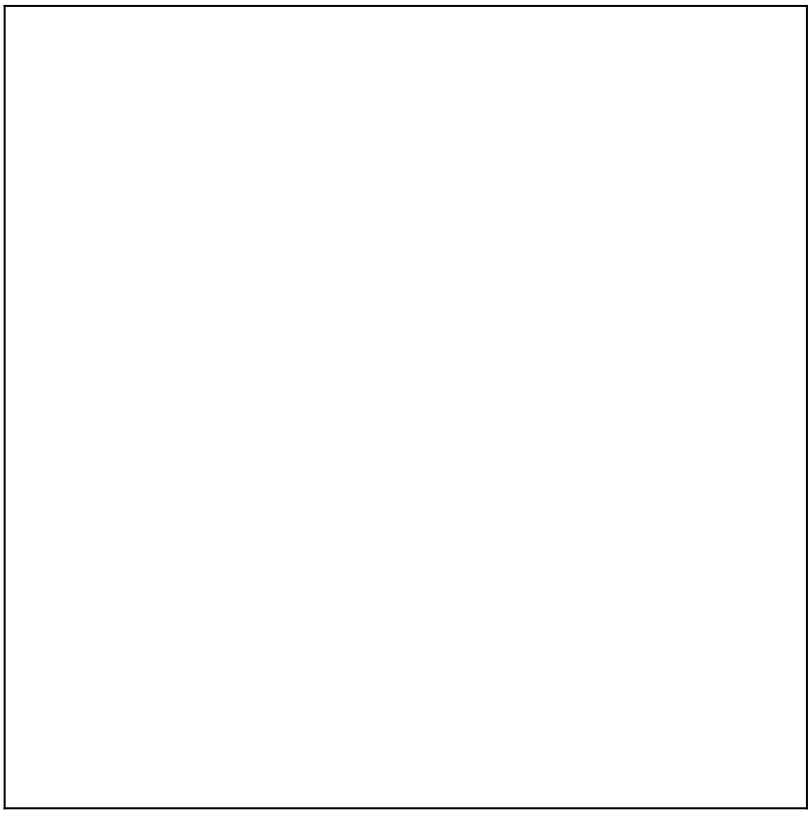
Step 1 of the check out process allows the user to make an account before continuing with payment. Selecting "Register Account" will change Step 2 of checkout from Billing to Account & Billing details. Account & Billing asks for the same personal details as Billing, except that it asks for the user to create a password for their account. After Step 2 is completed, the customer may continue with the checkout process.

2. **Header- "create an account"**



Clicking the "create an account" in the header will direct the customer to the "Register an Account" page. The same personal details included in the checkout will need to be filled in here.

3. **Header-** "My Account"



Clicking "My Account" in the header will direct the customer to the "Account Login" page. This page gives the customer an option to log in if they already have an account, or create a new account. In the "New Customer" section the customer can click "Continue" under Register Account to be directed to the "Register an Account" page.

Checkout

Once a product has been added to the cart, the customer can continue to the checkout to make their product purchase. The Checkout page can be accessed in the header section of every page (found under the search box). Customer checkout using OpenCart is a simple process that can be completed in 6 steps.

Step 1: Checkout options

The customer can log into or register their account (as explained above), or select guest checkout.

Step 2: Billing details

Personal details including "First Name", "Last Name", "E-mail", and "Telephone" are filled into a form. It also requires the customer's address details.

Step 3: Delivery details

In Billing Details, the user can check a box to indicate that the delivery details and billing details are the same. This will cause it to skip over this step to Delivery Method. If the delivery etails are different from the billing details the customer can enter this information in a form in this section.

Step 4: Delivery method

A method of shipping is selected here. A comment box is added for the customer to add comments about their order.

Step 5: Payment method

The customer selects their method of payment here and may add comments in the comment box.

Step 6: Confirm order

In this last step, the customer will see an overview of their purchase; including the product description, quantity, and price (with tax & shipping).

Store front customization using the admin dashboard

All of the features listed above can be customized to some degree in the admin panel. The administrator can change the position of certain products, disable categories, edit prices and descriptions, upload banners, etc. There is much work that can be done in the OpenCart admin to establish the shop's brand.

Admin interface

Skip to end of metadata

Attachments:4
Added by Catherine [HostJars], last edited by Jennifer [HostJars] on Oct 13, 2013 (view change)

Go to start of metadata

Table of Content

Connecting the OpenCart administration

OpenCart's administration side of the store is where you can modify features, upload images, add products, keep track of customers, manage payments, and much more. Customization in the admin affects how the customer will interact with a store: by modifying the look, structure, and content of the store front.

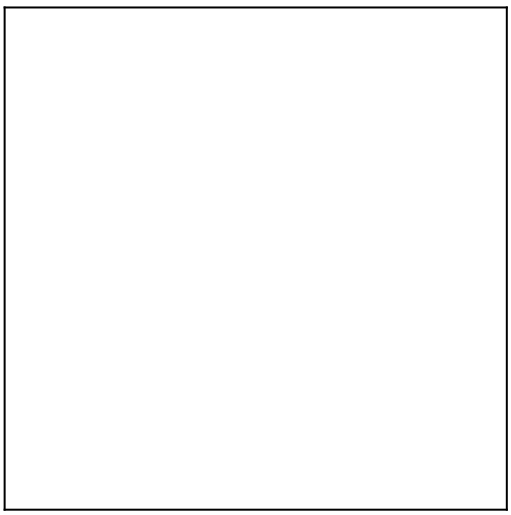
Accessing the admin panel

To access the admin panel, type in location of the store into the web browser followed by "/admin". For example, if your store is located at "www.chocolatechip.com", your admin panel is located at "www.chocolatechip.com/admin". Even if the store is located in a sub-folder or on a sub-domain of their site, adding "/admin" to the end of the store's path will lead you to the administration side.

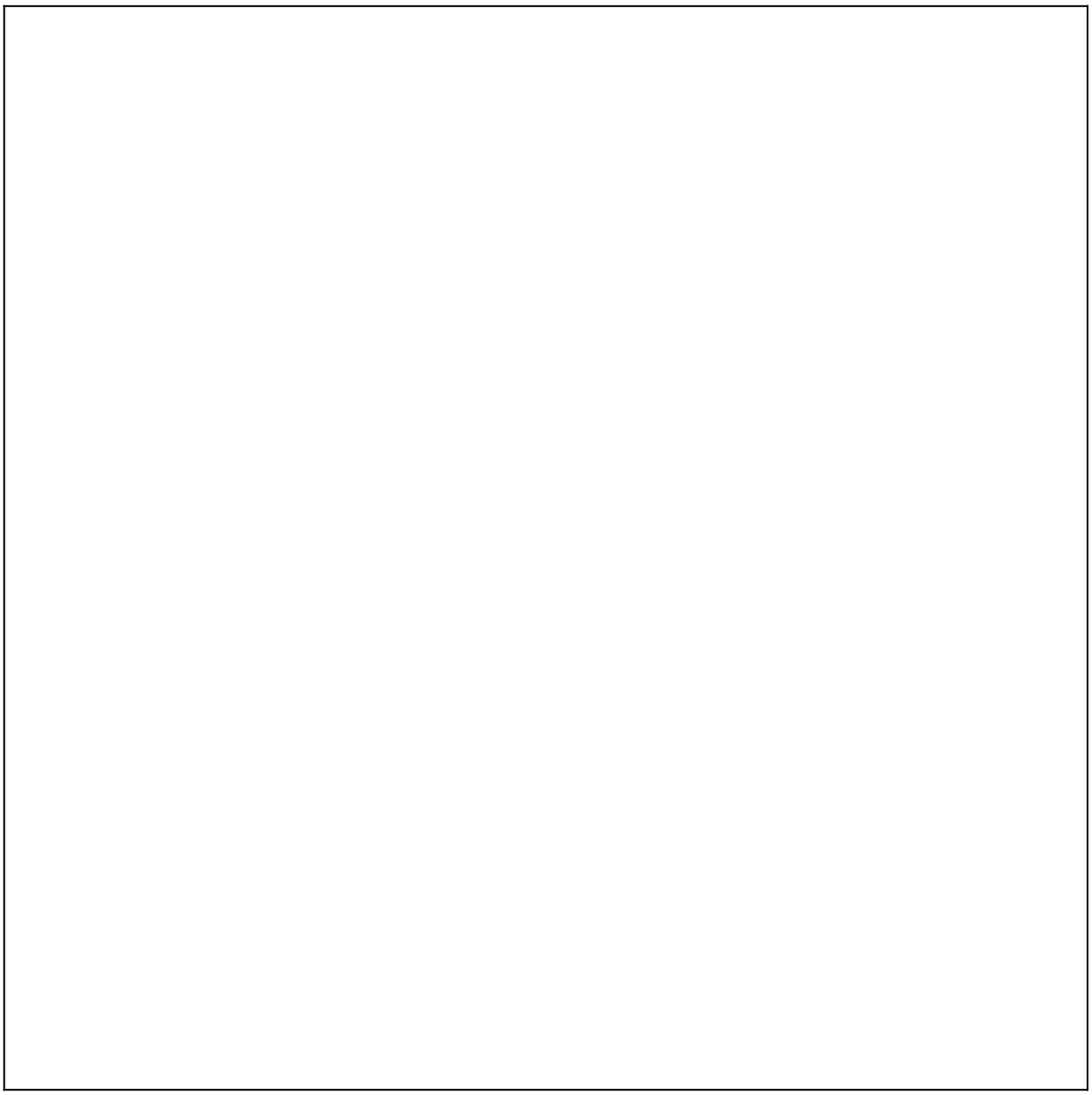
If the name of the folder is changed by you during installation, the location of the admin panel is changed to that new name. For example, if the name of the admin folder is changed to admin1, the new location of the admin panel is at "www.chocolatchip.com/admin1".

> Changing the name of the admin folder to another name is important for improving an online shop's security. The config.php file needs to be edited to indicate the correct file path and location.

A username and password is created in Step 3 of the auto-installer during installation. This information can now be used to fill in the administrator login details.



After filling in the correct username and password, pressing the "Login" button will direct you to the OpenCart dashboard. When you first login to your shop, the dashboard will be blank (as seen below), because there isn't any statistical data to be analyzed yet.



Entering the administration through the Dashboard

The dashboard is the first thing you will see when entering OpenCart's admin. The main function of the dashboard is to give the shop owner an overview of how the shop is performing. There are 3 sections of the dashboard that can help you understand the statistical data collected by your store:

- **Overview**: OpenCart calculates the numerical values for "Total Sales", "Total Sales This Year", "Total Orders", "No. of Customers", "Customers Awaiting Approval", "Reviews Awaiting Approval", "No. of Affiliates", "Affiliates Awaiting Approval", to alert you to approvals and keeping track of sales.
- **Statistics**: A graph is provided to track the chronological progress of the store relative to the amount of orders and customers over time. The x value is time; which can be hours, days, or months depending on the range selected. The y value displays the number of total orders(yellow) and total customers(blue).
- **Latest 10 Orders**: A list that displays the last 10 orders and their details ("Order ID", "Customer", "Status", "Date Added", "Total", and "Action")

Above the dashboard in the top menu is the administration navigation. You can navigate between the "Catalog", "Extensions", "Sales", "System", "Reports", and "Help". These sections will be explained in further detail in the following sections of the User Guide.

None

Catalog

A look inside the Catalog

Skip to end of metadata

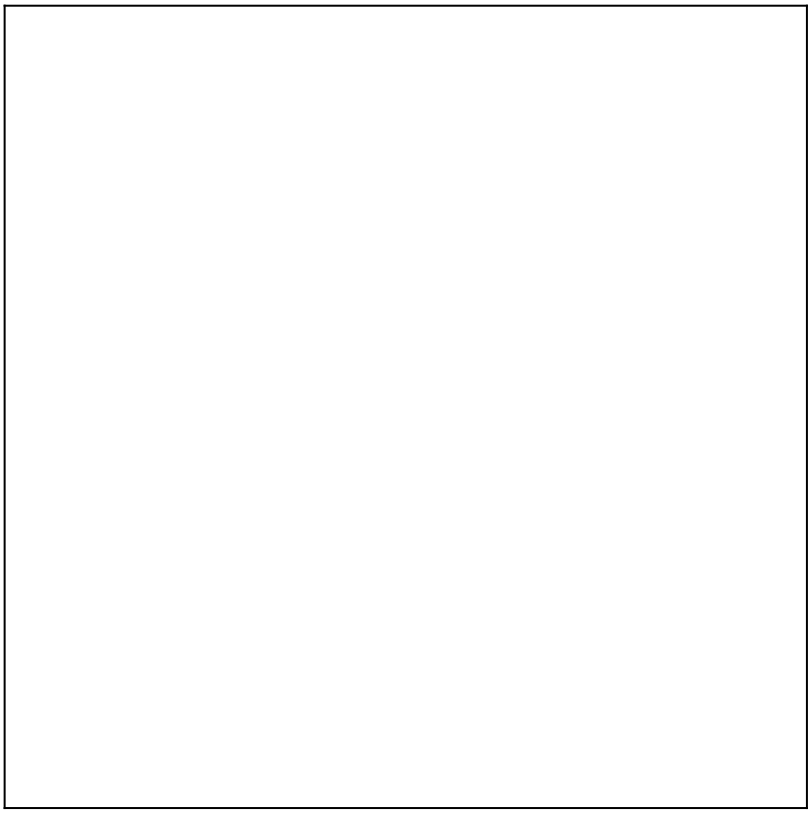
Attachments:1
Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

A look inside the Catalog

For stores managing a variety of product, organizing product data may seem like a daunting task. Fortunately, the OpenCart Catalog sections in the administration panel make it relatively simple to manage a store's inventory.

The Catalog section is located at the top menu of the admin panel. If you haven't accessed the admin panel yet, you may visit Admin interface for more information.



This guide will give an overview to the organizational features available in the Catalog section. We will walk you through how to add store product and product categories to the administration in Products and Categories.

None

10 Child Pages

Page: CategoriesPage: ProductsPage: FiltersPage: ProfilesPage: AttributesPage: OptionsPage: ManufacturersPage: DownloadsPage: ReviewsPage: Information

Extensions

Skip to end of metadata

Attachments:1
Added by Catherine [HostJars], last edited by Lars [HostJars] on Sep 12, 2012 (view change)

Go to start of metadata

Extensions

The Extensions section contains the following pages in the administration:

- Modules
- Shipping
- Payments
- Order Totals
- Product Feeds

Additional extensions can be downloaded to the store from Extension Directory.

None

5 Child Pages

Page: ModulesPage: ShippingPage: PaymentsPage: Order totalsPage: Product Feeds

Sales

Skip to end of metadata

Added by Catherine [HostJars], last edited by Catherine [HostJars] on Sep 10, 2012

Go to start of metadata

None

7 Child Pages

Page: OrdersPage: ReturnsPage: CustomersPage: AffiliatesPage: CouponsPage: Gift VouchersPage: Mail

Add to the documentation

Get help from the community

Buy the User Guide PDF

Get import tools and modules

Reports

Skip to end of metadata

Added by Catherine [HostJars], last edited by Catherine [HostJars] on Sep 10, 2012

Go to start of metadata

None

4 Child Pages

Page: Affiliate commission reportPage: Customer reportsPage: Products viewed reportPage: Sales report

Contribute
Add to the documentation

Support
Get help from the community

PDF Version
Buy the User Guide PDF

Extensions
Get import tools and modules

System

Skip to end of metadata

Added by Catherine [HostJars], last edited by Catherine [HostJars] on Sep 10, 2012

Go to start of metadata

None

6 Child Pages

Page: SettingsPage: DesignPage: UsersPage: LocalisationPage: Error logsPage: Backup/Restore

Add to the documentation

Get help from the community

Buy the User Guide PDF

Get import tools and modules

Help

Skip to end of metadata

Added by Catherine [HostJars], last edited by Lars [HostJars] on Sep 12, 2012 (view change)

Go to start of metadata

Help

The Help section links to additional resources when clicked on in the admin panel. The following pages are available for you if you need any help with or have any questions regarding the OpenCart administration side or store front:

- Homepage: the OpenCart main page can be visited to see if there are any updated versions available, access the Support section, play with the OpenCart demo, browse features, and more.
- Documentation: a beginner's guide to using OpenCart.
- Support Forum: a support forum where you can post your direct questions, receive support, report bugs, and contribute to the OpenCart community. If you have an issue, usually entering a keyword in the search box will bring up forum posts related to the issue.

None

Miscellaneous

Skip to end of metadata

Added by Catherine [HostJars], last edited by Catherine [HostJars] on Sep 10, 2012

Go to start of metadata

None

2 Child Pages

Page: FilterPage: Image manager

System administrator guide

Skip to end of metadata

Attachments:4
Added by Catherine [HostJars], last edited by Jennifer [HostJars] on Oct 13, 2013 (view change)

Go to start of metadata

System administrator guide

None

8 Child Pages

Page: Adding multiple languagesPage: Creating a multi-storePage: Image uploadPage: Moving OpenCart to a new serverPage: SEO keywordsPage: SSL Certificates and HTTPSPage: vQmodPage: Basic security practices

Adding multiple languages

Skip to end of metadata

Attachments:2
Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

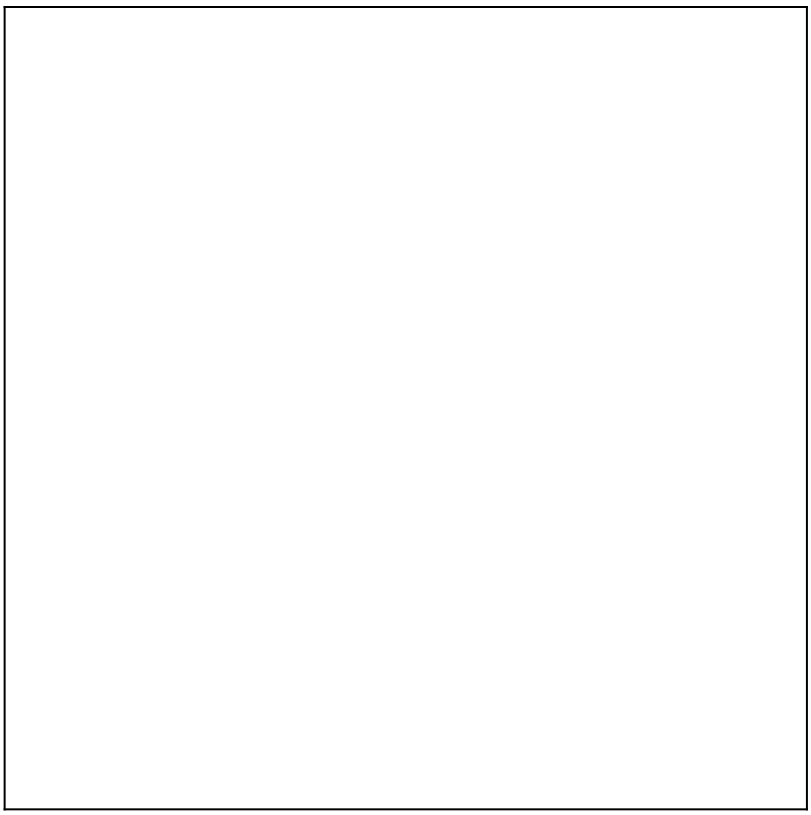
Go to start of metadata

Adding multiple languages

With shops serving a wider array of customers across multiple countries, it may become necessary to add multiple language options for your customers. The default language provided from the installation is English. Other available languages can be viewed on the OpenCart Language page.These language packs can be downloaded through the Extension Directory.

FTP a language pack to an OpenCart store

Any language pack that isn't English needs to be uploaded to OpenCart, post installation, using an FTP client like FileZilla. Before we continue, please make sure that you have downloaded your language pack from the Extension Directory and uncompressed the download contents to a location on your computer. Connect to your OpenCart store in the FTP client. Locate the root directory of where the OpenCart store was installed. From there, open the path Catalog>Language. If this is your first time here, you will see an "english" folder already in this location. In Catalog>Language, upload your new language pack to this location.

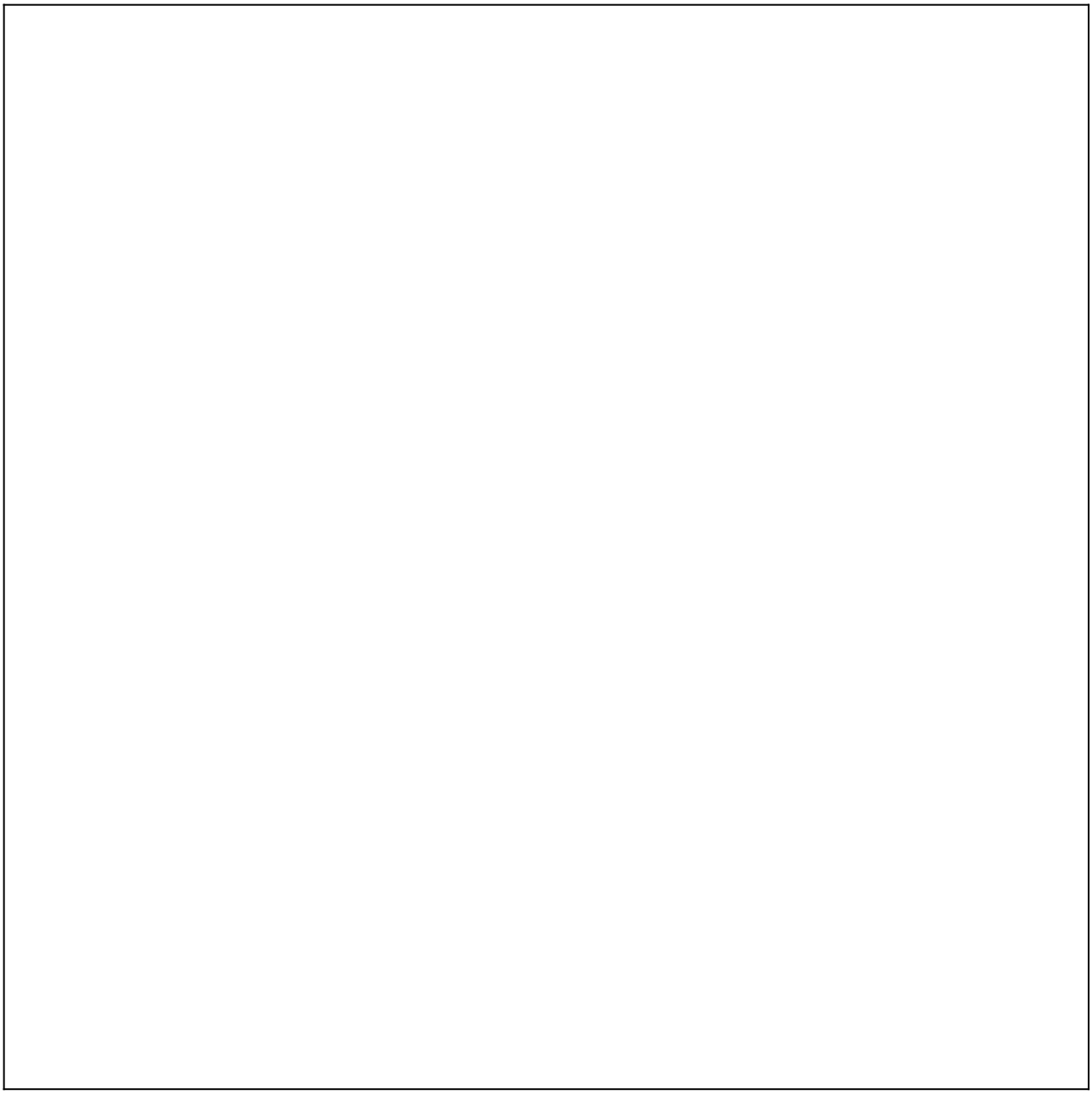


Adding a language to the administration

The OpenCart administration requires some specific information in the System area about the language after you FTP it. Visit Localisation to see what fields are required. After the language is saved there, the language name will appear in the language list under Localisation>Language.

Changing a language in the store front

Saving a language pack in Localisation will make it immediately available in the store front. The language area is located in the header of every page, next to currency. In achieve the example below, a German language pack was downloaded and FTP'd to OpenCart. By assigning German an assorting order of 2, it is displayed right of English in the footer of every page of our store. The customer can click on the German flag in the header to change the language.



None

Contribute
Add to the documentation

Support
Get help from the community

PDF Version
Buy the User Guide PDF

Extensions
Get import tools and modules

Creating a multi-store

Skip to end of metadata

Attachments:1
Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

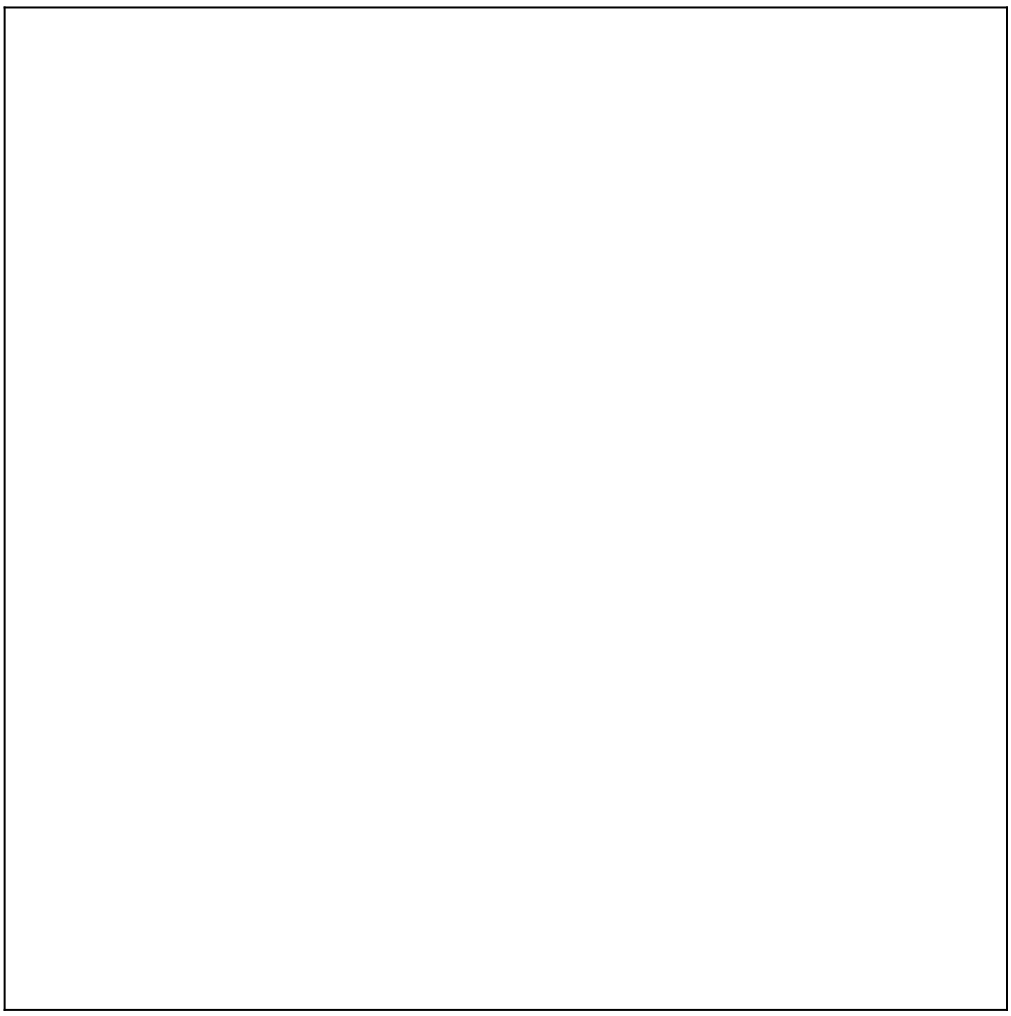
Go to start of metadata

Creating a multi-store

OpenCart allows for multi-store management using only one installation. If you have installed OpenCart into at least one store, you can add multiple stores to your admin panel without having to repeat the installation process. To add a new store you must first create a subdomain in your cPanel, then add the store in the Settings section of the admin panel.

Creating a subdomain in cPanel

You can login to the cPanel of their default store to create a subdomain. You should create a new folder for the subdomain, under Subdomain. In the root directory, link the subdomain to the path where the default store was installed to. For example, if the default store had been installed under "public_html/opencart", the new subdomain (opencart2) should be created under "public_html/opencart". We could add more subdomains here to create our multi-store by following this procedure.



When we visit our new subdomain in our browser, the default store is visible. To customize the new store at this subdomain we need to visit the admin panel for our multi-store.

> Don't worry if the default store is displayed where the new store should be, this is normal. The new store will not be visible at the subdomain's location until it has been added in the administration side of OpenCart.

Creating a new store in Settings

The admin panel of the new store can be visited through either store sites, at "opencart.mystore.com/admin" or "opencart2.mystore.com/admin". Just add a "/admin" to either of your stores' locations to access the admin panel. There is only one admin panel that controls all the stores in your multi-store.

To create a new store in the administration, visit System > Settings and press Insert. Adding a store requires information to be filled from the General, Store, Local, Option, Image, and Server tabs. In these sections you can add a new template, logo, currency, language, and layout. After pressing "Save", you will see that the default store is replaced by the new store in that subdomain's store front.

Customizing the store front

Products, product categories, customers, page layouts, and more, can be edited in the administration and customized for each store. You can individually select which products are available for each store in the Links tab when editing or creating a product. Checking the stores in this section makes the product only available in those specific stores. When adding or modifying a product category, you can select which stores display the category in the Data tab. Customer and order info will be automatically sorted into their appropriate store in the administration side when they create an account or buy a product at that store.

None

Contribute
Add to the documentation

Support
Get help from the community

PDF Version
Buy the User Guide PDF

Extensions
Get import tools and modules

Image upload

Skip to end of metadata

Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

Image upload

The image and file manager in OpenCart can often cause users grief. Noting a few things can greatly reduce difficulty:

- Make sure your system settings are set to not display errors. PHP errors in JSON responses required by the image manager break the JSON and therefore the image upload.
- Make sure your files are valid images of appropriate format (animated gif files are not supported).

None

Moving OpenCart to a new server

Skip to end of metadata

Added by Catherine [HostJars], last edited by Catherine [HostJars] on Sep 17, 2012

Go to start of metadata

Moving OpenCart to a new server

OpenCart can be easily migrated to a new web server. The steps are listed below:

1. Copy all the OpenCart files from your existing web server to the new web server.
2. Export the OpenCart database and import it into the new server.
3. Edit config.php and admin/config.php. These files contain the filesystem paths to OpenCart folders, URLs to OpenCart frontend and admin, and database access details. The paths and database details will most likely need to be updated, and the URLs will need to be changed only if the OpenCart store's domain has changed.

None

SEO keywords

Skip to end of metadata

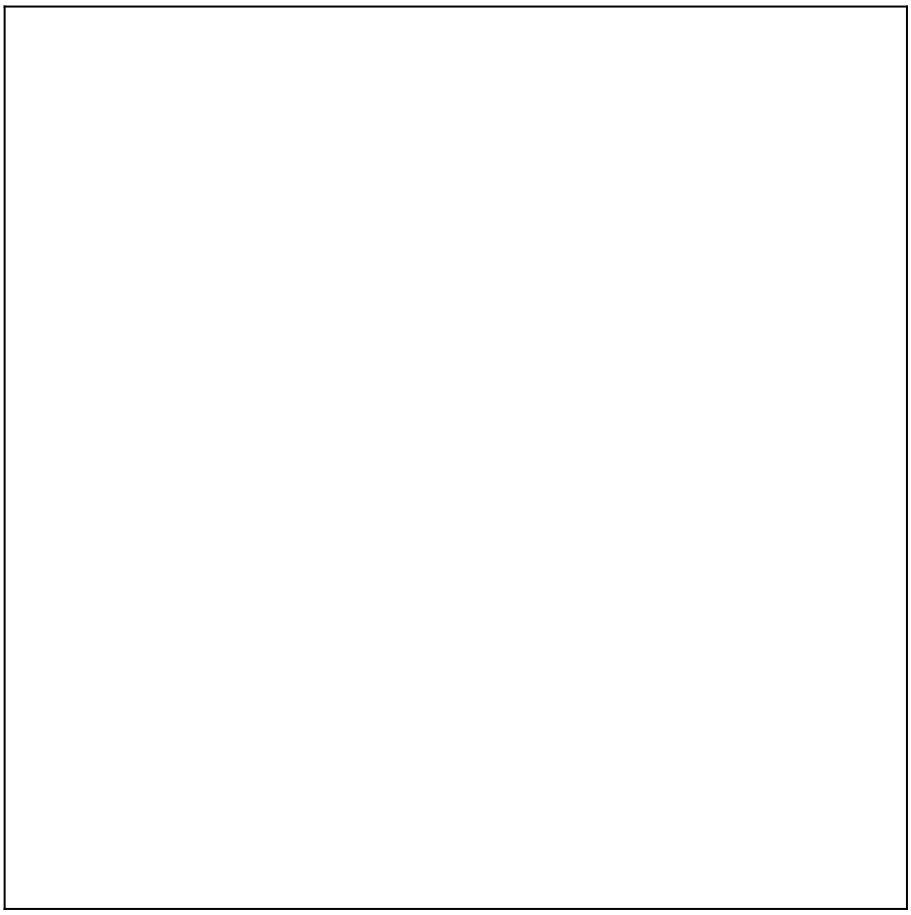
Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

SEO keywords

OpenCart allows the use of search engine optimized URLs
for product, category, manufacturer and information pages. SEO keywords are defined per product, category, manufacturer and information page, stored in a database table (url_alias), looked up at each page request, and translated into an internal URL.

In order to use SEO keywords, they must be enabled in the store settings page in the admin interface.

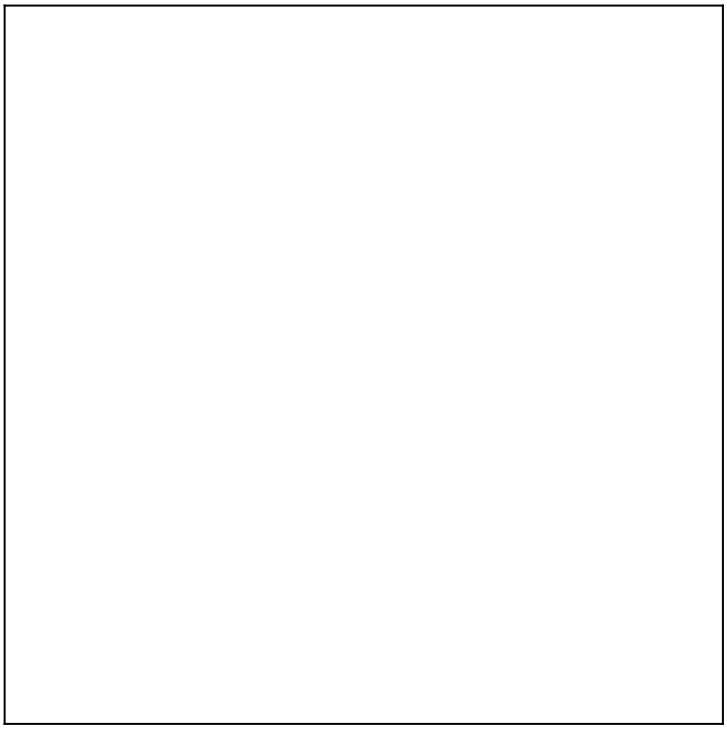


A correctly formatted .htaccess file must be present in the root of the OpenCart installation (beside the admin and catalog folders). The contents of the .htaccess file are shown below:

The .htaccess.txt file should be present in your store's root folder. If not, copy and paste the text above into a

text editor and save it as ".htaccess". If the .htaccess.txt file is present, please change its name to ".htaccess". With the .htaccess file changed, you should be able to add unique SEO keywords to individual products, manufacturers, categories, and information pages.

The SEO keywords you add will be displayed in the url of the page, so they must be unique for each product, product category, etc. Let's see what happens when we add the SEO keyword, "nikon-camera", to our Nikon D300 product under the Data tab:



If our store is located at "www.mystore.com", the new URL of the Nikon D300 product page will be located at "www.mystore.com/nikon-camera". Overall, creating SEO keywords will clean up your store page's URL and optimize your page for search engines.

None

SSL Certificates and HTTPS

Skip to end of metadata

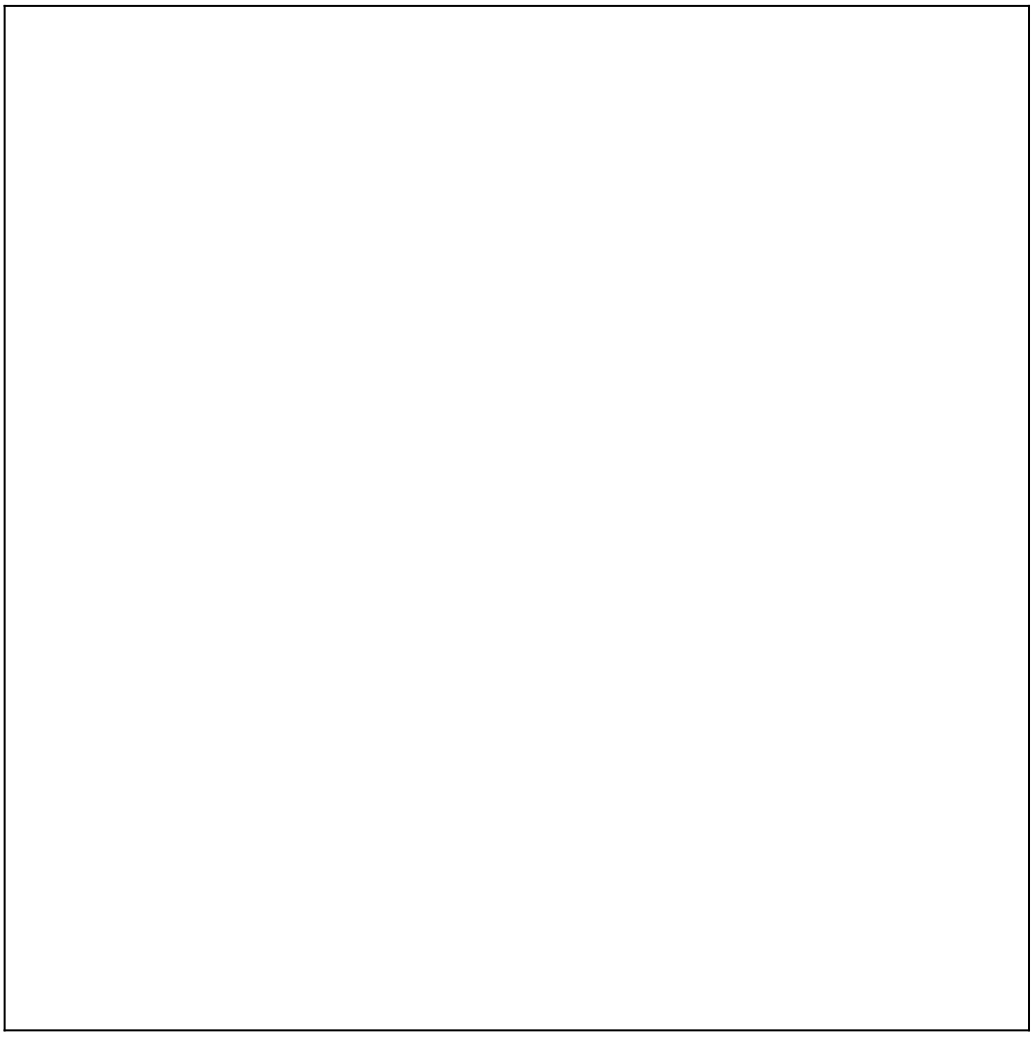
Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

SSL certificates and HTTPS

With sensitive customer and payment information being handled by your store, it is critical that you are able to secure that important information as it travels between web server and browser. As a result of this need, many store owners turn to HTTPS for additional security. A SSL certificate needs to be obtained and installed before you can enable HTTPS for your store.

Once the SSL certificate has been installed, go to the Server tabs under your store's settings in your admin panel. The first option in the Server tab lets you decide if you want to use SSL. Selecting "yes" will enable the SSL so that you can access your store through HTTPS.



None

Contribute
Add to the documentation

Support
Get help from the community

PDF Version
Buy the User Guide PDF

Extensions
Get import tools and modules

vQmod

Skip to end of metadata

Attachments:2
Added by Catherine [HostJars], last edited by Justin [HostJars] on Sep 21, 2012 (view change)

Go to start of metadata

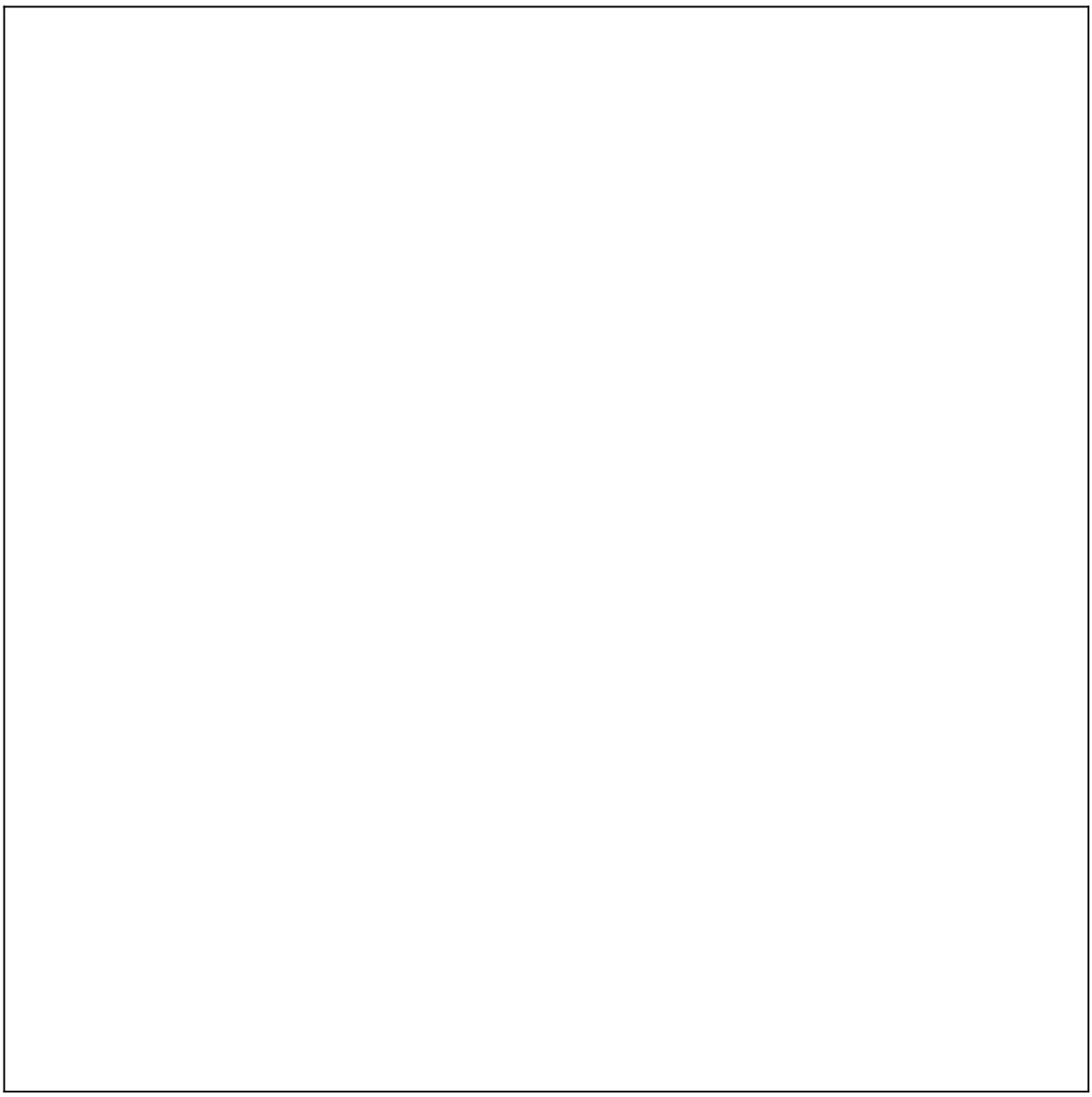
vQmod

While strictly modules, product feeds and OpenCart extensions should be standalone and modular, some require modifications to the OpenCart core. Modifying OpenCart core files greatly reduces your ability to upgrade your store to later versions, and can affect other modules and extensions. vQmod provides a mechanism by which modules requiring OpenCart core modifications can be installed without damaging the integrity of your core system for upgrades. vQmod keeps a list of filenames and changes required, in the form of one XML file per extension. These changes are then created as the core files are used, and the resulting files are stored as cached copies which are executed instead of the original, core PHP or TPL file.

Installing vQmod

If you wish to upgrade your store, or have a 3rd party extension that you have not used before, it is advisable that you use the vQmod system. The vQmod download and instructions for installation can be found here:http://code.google.com/p/vqmod/.

Once you have downloaded the appropriate vQmod zip file provided in the link, you need to uncompress the zip file and FTP the vQmod folder to your site's root directory. From the Filezilla screenshot below, we can see the root directory of this store is located at public_html/opencart in the Remote site side. Uploading the vQmod folder here will make the vqmod folder visible in public_html/opencart.



vQmod advises you to set the permissions to writable for the vqmod/vqcache folders, index.php, and admin/index.php. In Filezilla you can right click on a file or folder, select "File Permissions...", and enter 755 or 777 in the "Numeric value" to set it to writable.

The next step in installing vQmod is to visit the install page in your store. Enter your store's location in your browser, then "vqmod/install" afterwards. If your store is located at "www.mystore.com", the vQmod install page would be located at "www.mystore.com/vqmod/install". The following text will appear on the page if it was installed correctly: "VQMOD HAS BEEN INSTALLED ON YOUR SYSTEM!".

> Unlike the install folder after installing OpenCart, do not delete vQmod's install folder after the vQmod installation!

vQmod advises that you load your homepage to see if it appears and works correctly after the install. You would also need to check your vqmod/vqcache folder to see if there are new vq files.

None

Contribute
Add to the documentation

Support
Get help from the community

PDF Version
Buy the User Guide PDF

Extensions
Get import tools and modules

Basic security practices

OpenCart is not responsible for the securing your website, therefore it is up you to ensure your server's level of safety. The following suggestions aim to improve your OpenCart store's security.

These additional steps can be taken immediately after OpenCart is installed to your server; or whenever you store becomes active. See Installation for more information on how to install your OpenCart store.

Delete the install folder

Deleting the install folder is advised by OpenCart immediately after installation. OpenCart will warn you in the administration if the install folder is not deleted.

Directory protection

Admin folder

The admin directory is where you have access to your store's administration. People with access to your store administration will have access your editing your products, customer information, store settings, and more valuable information. Therefore, it is very important that the admin login be difficult to find and access.

Rename admin

Renaming the admin directory to something unrelated to the the admin is necessary to prevent unwanted eyes from discovering it's location. You can access your admin login through entering your store's location, then the path to the admin. For example, if the admin folder was changed to "cookiemonster", the admin login would be at "www.yourstore.com/cookiemonster".

.htaccess & .htpasswd

A .htaccess and .htpasswd file in the admin folder will prevent hackers from accessing your store, even if they discover the admin login location. Using .htaccess, you can deny all IP addresses from viewing your store, except the admin's IP address. A .htpasswd in the admin folder will require an additional password for the allowed administrator to access this directory.

Catalog

The catalog can be protected with the traditional .htaccess file. Using file match can be useful for protecting important file types for your store, such as php and txt, rather than all of them. The following code can be used for .htaccess in your catalog folder:

```
<FilesMatch "\.(php|tpl|txt)$">
Order Deny,Allow
Deny from all
Allow from "your ip address"
</FilesMatch>
```

This will deny access to all template, php, and txt files.

System folder

The system folder contains two files that need to be protected: logs/error.txt and start_up.php. The logs/error.txt can be renamed if necessary.

.Htaccess

The .htaccess will work to protect these files and the subfolders of System from being accessed by anyone except the designated administrator. To do so, insert the code below into your .htaccess:

```
<Files *.*>
Order Deny,Allow
Deny from all
Allow from "your ip address"
</Files>
```

File permissions

The following files need to be set to 644 or 444 to prevent anyone else from writing to them:

- config.php
- index.php
- admin/config.php
- admin/index.php
- system/startup.php

None

Developer guide

Skip to end of metadata

Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

Developer guide

None

4 Child Pages

Page: Introduction to MVC-LPage: Developing modulesPage: Developing new product feedsPage: Loading files in the controller

Add to the documentation

Get help from the community

Buy the User Guide PDF

Get import tools and modules

Introduction to MVC-L

Skip to end of metadata

Added by Justin [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

Getting started

OpenCart is an excellent platform for developers looking to understand PHP web frameworks in general. It is one of the easiest to follow MVC structured applications available. OpenCart allows you to learn the MVC framework while giving you access to the familiar PHP, mySQL and HTML technologies on which it is built. This guide will assume a basic understanding of HTML, CSS, Javascript, PHP (including classes and inheritance), and mySQL, and will describe how these are used in the OpenCart system.

MVC(L)

OpenCart is designed to follow an MVC design pattern. The components of MVC (Model View Controller) can be broken down as follows.

M - Model

This is where you will interact directly with your database, pulling data out and restructuring it to a format that is suitable for your frontend. This will usually mainly consist of DB queries, and little more. If you are used to writing mySQL queries, you will enjoy the way OpenCart provides access to continue to do just that. OpenCart does not use an ORM, but allows you to write direct database queries.

V - View

This is the display side of the MVC pattern. The idea of the M and C is to pull as much logic out of the view as possible, meaning simpler templates. In order to redesign your whole store, you simply modify the View component, the M, C and L would remain the same. The view files in OpenCart have the .tpl suffix.

C - Controller

This is where you will pull together the data from the Model, any config settings saved with your install or modules, and then render it by choosing the appropriate View file(s).

L - Language

OpenCart extends MVC to MVCL, providing an easy way of separating language specific information for internationalization. You can use language files to store any text like headings, titles, button text, etc., so that you only need to adjust one file per language to provide translations of your store.

Directory structure

The OpenCart directory structure is based around two important parts of the OpenCart application.

The frontend and the admin interface are each represented by a folder in the top level of your OpenCart installation. The frontend folder is called catalog/ and the admin folder is called admin/. If you are making modifications only to the admin of an OpenCart store, you should not expect to modify any file within the catalog/ folder. If you are working on the frontend only, you should not be modifying the admin/ folder at all.

Within each of the admin/ and catalog/ folders, you will find a folder for each of the Model, View, Controller and Language components of the OpenCart application.

Several other folders exist in your OpenCart installation.

The system folder contains classes and functions used by both the admin and catalog areas of your store. Within this folder are email helpers, database helpers, the core definition of controllers, models and other parts of the OpenCart engine and library classes. When modifying OpenCart functionality, you will seldom need to edit any of the system files.

The image folder contains all the images that are uploaded via the Image manager. These are your product images, additional images, and the cached versions of images that OpenCart has resized.

The download folder contains any downloads associated with products. Downloads are given a hashed suffix to avoid malicious users guessing the filenames correctly and accessing your downloads directly. This is why you will see random strings on the end of your download filenames in this directory.

None

Adding multiple languages

Skip to end of metadata

Attachments:2
Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

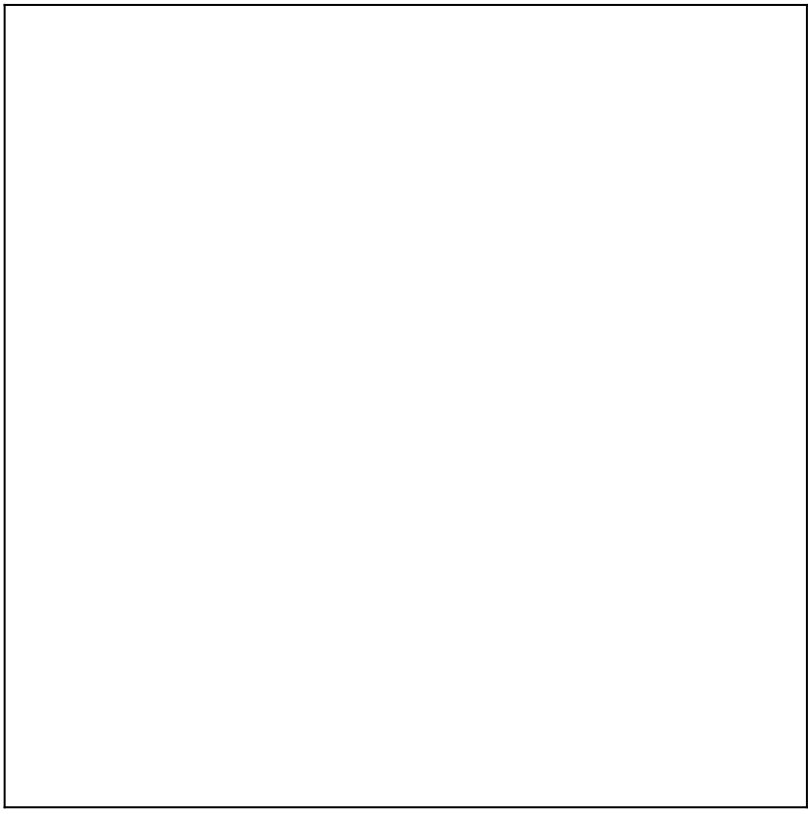
Go to start of metadata

Adding multiple languages

With shops serving a wider array of customers across multiple countries, it may become necessary to add multiple language options for your customers. The default language provided from the installation is English. Other available languages can be viewed on the OpenCart Language page.These language packs can be downloaded through the Extension Directory.

FTP a language pack to an OpenCart store

Any language pack that isn't English needs to be uploaded to OpenCart, post installation, using an FTP client like FileZilla. Before we continue, please make sure that you have downloaded your language pack from the Extension Directory and uncompressed the download contents to a location on your computer. Connect to your OpenCart store in the FTP client. Locate the root directory of where the OpenCart store was installed. From there, open the path Catalog>Language. If this is your first time here, you will see an "english" folder already in this location. In Catalog>Language, upload your new language pack to this location.

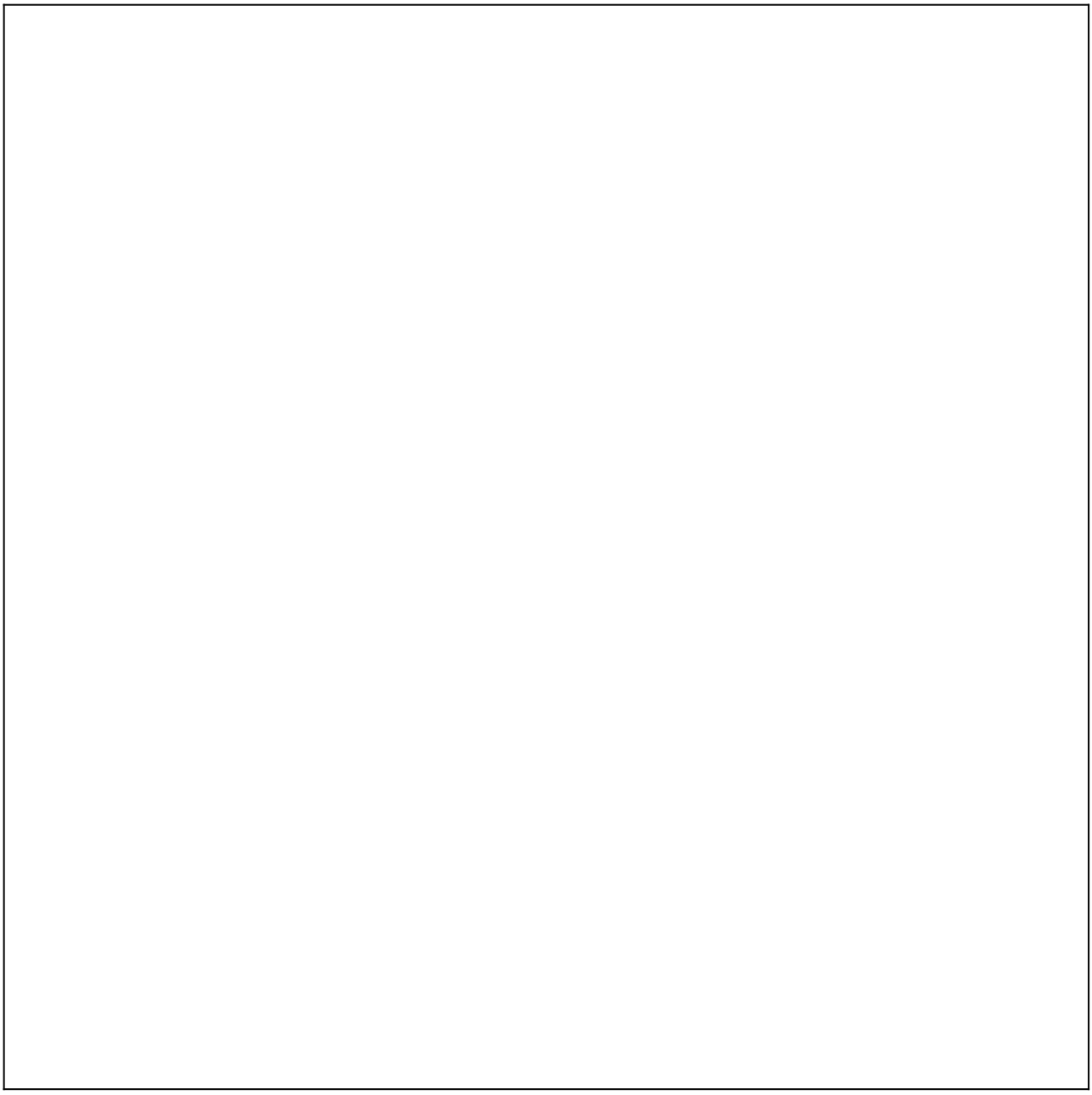


Adding a language to the administration

The OpenCart administration requires some specific information in the System area about the language after you FTP it. Visit Localisation to see what fields are required. After the language is saved there, the language name will appear in the language list under Localisation>Language.

Changing a language in the store front

Saving a language pack in Localisation will make it immediately available in the store front. The language area is located in the header of every page, next to currency. In achieve the example below, a German language pack was downloaded and FTP'd to OpenCart. By assigning German an assorting order of 2, it is displayed right of English in the footer of every page of our store. The customer can click on the German flag in the header to change the language.



None

Contribute
Add to the documentation

Support
Get help from the community

PDF Version
Buy the User Guide PDF

Extensions
Get import tools and modules

Developing modules

Skip to end of metadata

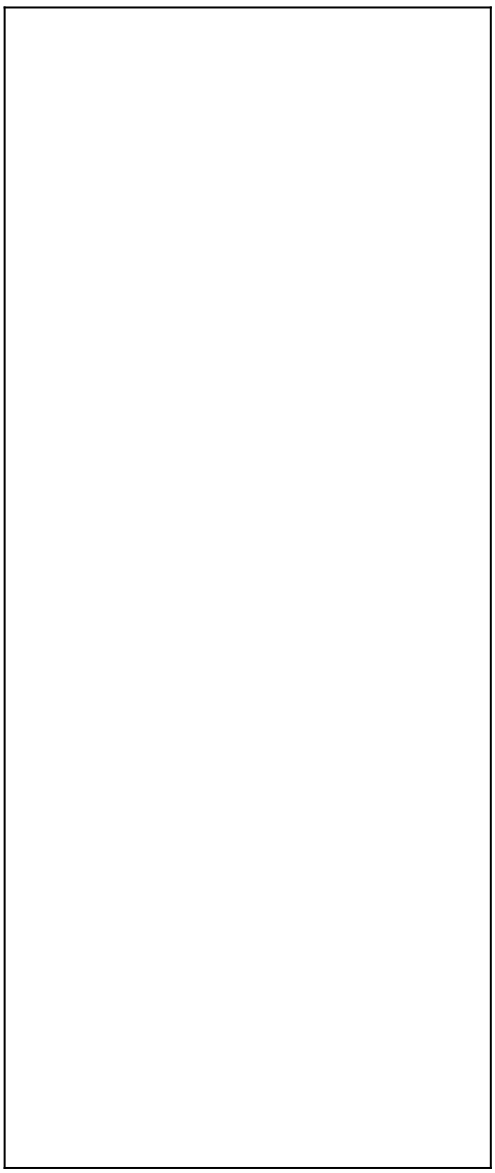
Attachments:2
Added by Justin [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

Writing OpenCart modules can be a very good way to learn how the fundamentals of OpenCart actually work. Just as the rest of OpenCart, modules follow the MVCL design pattern. This documentation guide will describe how you use each of the MVC-L components to create the admin and frontend parts of your module. The easiest way to create a module is to download the DIY Module Builder skeleton from HostJars. This module contains the directory structure, files, and instructions on how to understand and build your own modules. This page is a more theoretical guide.

Basic directory structure

The basic file structure for your module will be divided into two sections, the admin and the catalog folders. The contents of each folder will follow the MVC-L framework respectively, with the difference that the admin will deal solely with backend functionality, and the catalog with frontend functionality. Users of your module will interact and configure its settings in the administration side of the store. Therefore, the files in the admin folder will handle any changes to its settings, the way the module is displayed in the administration, install/uninstalling the module, etc. Likewise, the way the module is displayed and how it works in the front end of the store will be handled by the files in the catalog folder.



The image above displays a skeleton of the directory structure that your module should follow. A good way to get started with your module is to duplicate the folder structure and create the files above. What will go in those files is determined by what your module is trying to accomplish, but the basic functionality is detailed in the sections below.

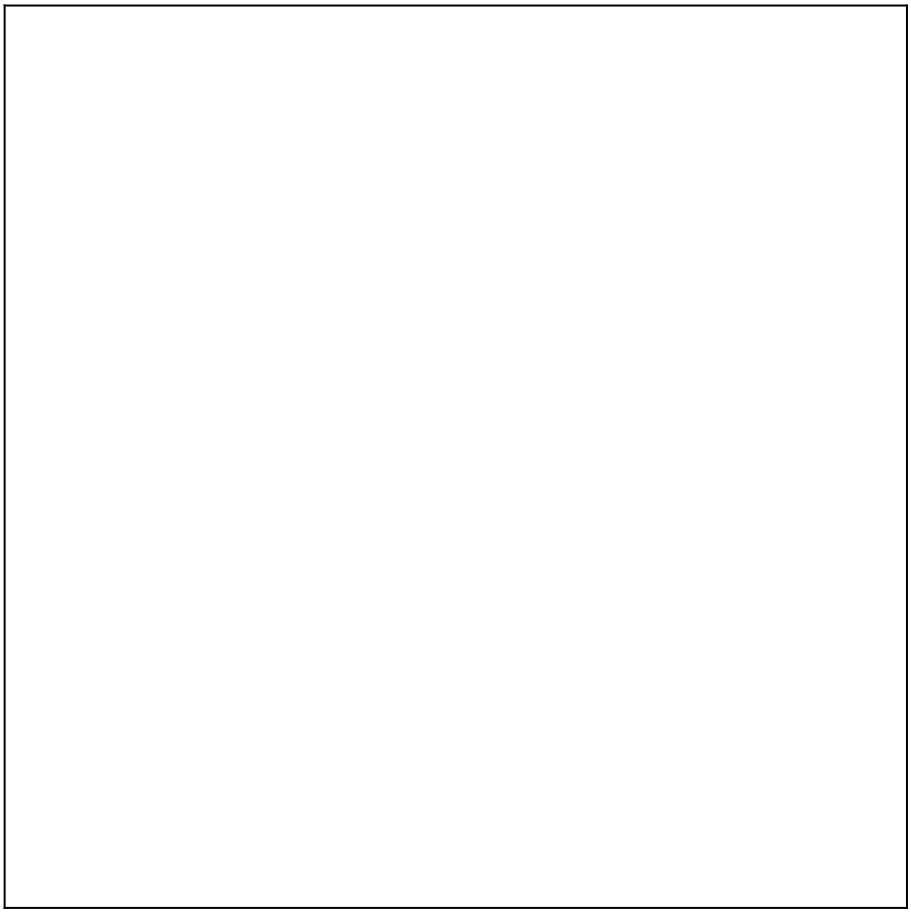
Admin module functionality

When a store owner uses your module, they will want to edit the module's configuration options in the admin in order to decide which layouts to display the module on, whether it is enabled or disabled, and any module specific options. As the module developer, you will need to create the admin page where the module may be edited and the configuration options added or adjusted.

All admin files are located in the admin/ folder. You will find four folders within the admin/ folder:

1. controller
2. view
3. language
4. model

All modules will require at least a single file in each of the view and controller folders. Most will require a file in each of the model and language folders. Usually the files have the same name, except the view file has a different suffix (.tpl). We will go through these files one by one.



Controller

The first file you make will be the controller for your module's admin interface page. OpenCart identifies existing modules automatically, simply by reading the admin/controller/module folder of your store. Any modules existing in this folder will automatically be shown on the Modules page, and on the User Permissions, page. You may call your controller file my_module.php.

The controller file is the place where you can load the language files to convert text into variables to be utilized in the template file. In the diagram above, you can see the $_['text'] variable being handled by the controller, then sent as $text to the view. You will also utilize multiple model files and their class functions here, including your module's model file if it has one. For more information on loading files see Loading files in the controller.

You may also have a function defined as public function install(). This function will be triggered when the install link is clicked on the Extensions > Modules page. Similarly, a function defined as public function uninstall() will be triggered when the uninstall link is clicked. You can use these functions to create and remove any structures (such as database tables or config settings) required by your module. It is good practice to create an uninstall function to clean up any changes your module has made. To see the specific code for an install(), uninstall(), visit Install/Uninstall a module.

Accessed via URL

The controller is the only file in the MVC-L framework to be accessed by URL in OpenCart. In the administration, the URL will look like /admin/index.php?route=module/my_module&token. The admin adds a token to the URL, whereas the link in the catalog will not have it. As a result, the controller file will have a function defined as public function index(). This is a publicly accessible 'page' that is loaded by the URL, which will be shown

when the Edit button is clicked, and where the view form will submit to. The submitted data will be processed in this function and saved to the `settings` database table through the controller's config object.

View

The second required file for your module's admin interface is the view file. This will be created in the admin/view/template/module folder, and will have the suffix .tpl. This is standard for OpenCart view files. In this file you will create a form for the user to fill out and submit. It will be submitted to the module controller's index function. The easiest way to create your view file is to copy and paste an existing, similar view file, and edit the form to contain the correct fields for your module's configuration options. You may call your view file my_module.tpl.

In the view, you will be able to access the text from the language that the controller file stored as a PHP variable. See Loading Files in the Controller for the code on how to do so.

Model

It is uncommon for modules to require a model file on the admin side. However, if your module relies on its own database table, or a custom query to create data of a particular format, then you may find yourself writing a model file. An example of this may be if you are writing a visitor counter module, where each visitor is stored in a database table with their IP address and number of visits. You may then create a model file, with a function to define and create this extra table in the OpenCart database. The model file will live in the admin/model/module folder. You may call your model file my_module.php, the same as your controller and language files.

Frontend module functionality

The frontend of your module follows the same pattern as the admin interface just described. What you will include in each of your frontend files will largely depend on what your module is supposed to do. A module can access any model files that already exist in OpenCart, you do not need to write your own database queries if the same query already exists. For example, the catalog/product model contains many useful queries for fetching products. Using these model functions should be preferred over reinventing the wheel.

A key difference in the frontend of your module, is that your view file will be in the catalog/view/theme/<themename>/template/module folder. This is a significantly deeper folder structure to the admin view file because of themes. An OpenCart store may have many different frontend themes available, but only one admin template.

On the frontend part of your module you will have access to the configuration options saved by your module, through both the controller's config object, and the $settings variable passed to the module controller's index function. You can control aspects of the frontend display on the basis of these settings.

None

1 Child Page

Page: Install/Uninstall a module

Add to the documentation

Get help from the community

Buy the User Guide PDF

Get import tools and modules

Developing new product feeds

Skip to end of metadata

Added by Justin [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

OpenCart includes several useful Product Feeds in the core, but you may find you require a custom format and decide to create your own. Writing OpenCart product feeds is very similar to writing OpenCart modules, and can be a very good way to learn how the fundamentals of OpenCart actually work. Just as the rest of OpenCart, feeds follow the MVCL design pattern. This documentation guide will describe how you use each of the MVCL components to create the admin andfrontend parts of your product feed.

Admin feed functionality

When a store owner uses your product feed, they will want to edit the product feed's configuration options in the admin in order to select whether it is enabled or disabled and any specific options you make available. As the developer, you will need to create the admin page where the product feed may be edited.

All admin files are located in the admin/ folder. You will find four folders within the admin/ folder:

1. controller
2. view
3. language
4. model

All product feeds will require at least a single file in each of the view and controller folders. Some will require a file in each of the model and language folders. Usually the files have the same name, except the view file has a different suffix (.tpl). We will go through these files one by one.

Controller

The first file you make will be the controller for your product feed's admin interface page. OpenCart identifies existing product feeds automatically, simply by reading the admin/controller/feeds folder of your store. Any product feed files existing in this folder will automatically be shown on the Product Feeds page, and on the User Permissions, page. You may call your controller file my_feed.php.

The controller file will have a function defined as public function index(). This is a publicly accessible 'page', which will be shown when the Edit button is clicked, and where the view form will submit to. The submitted data will be processed in this function and saved to the `settings` database table through the controller's config object.

You may also have a function defined as public function install(). This function will be triggered when the install link is clicked on the Extensions > feeds page. Similarly, a function defined as public function uninstall() will be triggered when the uninstall link is clicked. You can use these functions to create and remove any structures (such as database tables or config settings) required by your feed. It is good practice to create an uninstall function to clean up any changes your feed has made.

View

The second required file for your feed's admin interface is the view file. This will be created in the admin/view/template/feed folder, and will have the suffix .tpl. This is standard for OpenCart view files. In this file you will create a form for the user to fill out and submit. It will be submitted to the feed controller's index function. The easiest way to create your view file is to copy and paste an existing, similar view file, and edit the form to contain the correct fields for your feed's configuration options. You may call your view file my_feed.tpl.

Language

The third file you will usually create for any feed is the language file(s). You will need one language file per language that your feed is compatible with. The language file will live in the admin/language/<language name>/feed folder. It simply contains a PHP associative array called $_, which contains the internal name as the key and the translation as the value. You may call your language file my_feed.php, the same as your controller and model files.

Model

It is uncommon for feeds to require a model file on the admin side. However, if your feed relies on its own database table, or a custom query to create data of a particular format, then you may find yourself writing a model file. An example of this may be if you are writing a visitor counter feed, where each visitor is stored in a database table with their IP address and number of visits. You may then create a model file, with a function to define and create this extra table in the OpenCart database. The model file will live in the admin/model/feed folder. You may call your model file my_feed.php, the same as your controller and language files.

Frontend feed functionality

The frontend of your feed follows the same pattern as the admin interface just described. What you will include in each of your frontend files will largely depend on what your feed is supposed to do. A feed can access any model files that already exist in OpenCart, you do not need to write your own database queries if the same query already exists. For example, the catalog/product model contains many useful queries for fetching products. Using these model functions should be preferred over reinventing the wheel.

A key difference in the frontend of your feed, is that your view file will be in the catalog/view/theme/<themename>/template/feed folder. This is a significantly deeper folder structure to the admin view file because of themes. An OpenCart store may have many different frontend themes available, but only one admin template.

On the frontend part of your feed you will have access to the configuration options saved by your feed, through both the controller's config object, and the $settings variable passed to the feed controller's index function. You can control aspects of the frontend display on the basis of these settings.

None

Loading files in the controller

Skip to end of metadata

Added by Catherine [HostJars], last edited by Jennifer Creager on Jul 31, 2013 (view change)

Go to start of metadata

In OpenCart's MVC-L framework, your module's controller is the glue connecting your language, model, and template files to each other. The controller is responsible for grabbing the text contained in the language file and making them accessible as PHP variables in the view's template file. In addition to inheriting the functions available in the controller's parent class, Controller, you can also load any of OpenCart's default model files and their functions in the controller. In this documentation, we will show the PHP code needed to load language and model files and their functions.

Loading the language file

The controller brings the text stored in the language file, and turns them into variables that can be echoed in the template file to displayed text. This is especially useful for managing translations of your module. Instead of modifying your .tpl file every time you have a new translation to change each piece of text inside, you just need to modify the text in your language file, and the variables will remain the same in the controller and the template.

The piece of code below will load the language file inside in your module's controller. Inside the parentheses you will need to include the path to the language file from inside the language folder.

```
$this->load->language('module/my_module');
```

It is important to remember that the admin controller will only load the admin language file, but not the catalog language file; and likewise the catalog controller will only load the catalog language file. Once the language file is loaded into the controller, you can store its text into a php variable with the use of the $data array. The $this->language->get('text') will grab the text from the $_['text'] variable inside of the language file.

```
$this->data['text'] = $this->language->get('text');
```

The $this->language->get('text') will grab the text from the $_['text'] variable inside of the language file we just loaded above. Every element of the data array will be converted into its own variable. The $data['text'] will become $text for the template file inside view. The $text variable can be echoed in the view's .tpl file wherever needed:

```
<p><?php echo $text; ?></p>
```

Setting the heading title

The following code will set text from the language file as the heading title of the page:

```
$this->document->setTitle($this->language->get('heading_title'));
```

This will grab the text for the variable $_['heading_title'] stored in the module's language file.

If you need the text to be stored as a php session variable, use $this->session->data['text'] instead of $this->data['text'].

Loading model files

Loading model files into your controller file will allow your module to utilize OpenCart's built-in functions. The functions inside the model files interact with the store's database and to add/pull important information for your module. We recommended that you to take advantage of these functions, rather than making your own DB queries. Take some time to explore the model folders in both the admin and catalog files, to see which files may benefit your module's purpose. For example, if your module needs to pull product information from the store's database, it will be useful to load the admin/model/catalog/products.php file, since it already has a multitude of helpful, built-in functions that interact with the store's products in the database.

Your module can load any model file its controller file using the following code, granted that they are in the same admin or catalog folder as the controller.

```
$this->load->model('setting/setting');
```

You will need to specify the path to the file you want to load from the admin folder within the parentheses. The code above will load the settings class so we have access to the functions within the ModelSettingSetting class in our model's controller file. Use the following format in your code to call a function from a loaded model file:

```
$this->model_setting_setting->editSetting('my_module', $this->request->post);
```

The underscores refer to the file designations for model/setting/setting.php. If you have a model file included for your module your code would follow the format mentioned above, since the model file is uploaded to model folder.

```
$this->load->model(module/my_module.php);
$this->model_module_my_module->myFunction();
```

The code above will load the my_module.php stored in admin/model/module/my_module.php.

Instead of using spaces in file names for your module, use underscores.

Loading template files

In the controller you will need to load your module's template file in view. To do so, set $this->template as so:

```
$this->template = 'module/my_module.tpl';
```

Loading library files

The OpenCart directory contains a collection of library files that can be accessed by both the admin and catalog controller files. These files are located under system/library in the root folder of the OpenCart store. In the code examples seen in loading the language file, both $this->load->language and $this->document give access to functions within the language.php and document.php files in the library folder. If you want to access a function in a library file you need to call it using $this->[insert library file name]->function() in the controller class.

Additional information

For more information on the concept of how the controller behaves within the MVC framework or in the context of developing a module for OpenCart, see the Introduction and Developing Modules. To see how loaded files can be utilized in a basic module, see the DIY Module from HostJars.

None

Designer Guide

Skip to end of metadata

Added by Justin [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

Designer guide

None

1 Child Page

Page: Creating a custom theme

Creating a custom theme

Skip to end of metadata

Attachments:5
Added by Catherine [HostJars], last edited by Catherine [HostJars] on Feb 26, 2013 (view change)

Go to start of metadata

Table of Content

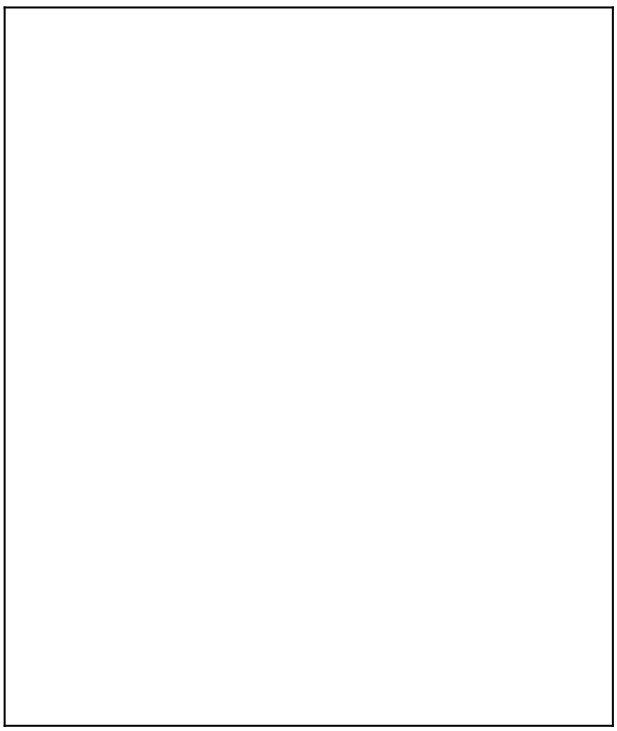
Creating a custom theme

For many people using OpenCart, the default theme provided with the installation meets their store's design needs. For users wanting added customization to their store's theme, such as a change of the color scheme, different borders, font, spacing, or any other style elements to enhance the look of their store, a custom theme needs to be created. The OpenCart directory architecture can be easily navigated to change the style and appearance of the store front. For those wanting to save themselves the trouble of making a theme, the OpenCart Extension Directory has about 1600+ themes available for purchase and download.

If you plan on constructing a custom theme for your store, you should have the functional knowledge of HTML andCSS inherent to web design. Knowledge of PHP and JavaScript will add more variety to what you can do with your theme, but it is not essential to the design element.

Default Theme folder structure

Before we can learn how to create our own theme, we need to become familiar with the layout of the default theme folders, because our new theme's layout must follow this file structure. The default theme is integrated into the Model-View-Controller pattern system of OpenCart, as explained in the Developer guide. View will contain all the files necessary to modify the appearance of your store. The default theme's stylesheet and template files are available in "/catalog/view/theme/default" of your store's root folder:



The "default" folder contains three folders:

- **template**
- **stylesheet**
- **image**

Template

The "template" folder contains template (.tpl) files. These .tpl files are a mixture of HTML and PHP, used strictly to create the structure of a web page. The PHP is ultimately converted into HTML in the browser, meaning you will not see the PHP when viewing the source in the browser. When you open the "template" folder you will see more folders organized into intuitive categories. Opening any of these folders will reveal individual .tpl files in that folder.

The way folders are organized are intuitive, files relating to product pages are under "product", files that control the modules are under "module", account pages are under "account", etc. The "common" folder contains the most used template files.The home page layout is determined by the home.tpl file. The other files in common, such as header.tpl, footer.tpl, column_left.tpl, column_right.tpl, make up individual sections of html on the home page, as well as other layout pages. As for the other folders, navigating them and determining which files to edit to get the desired results on a page can be tricky. You may encounter two .tpl files of the same name, but each will control the structure of different pages/sections. Template/module/category.tpl determines how the Category module box is structured in the store front, while template/product/category.tpl determines the HTML structure of the Category page where products are listed by category. When you are creating your own theme, you will need to view the source of the page you want to edit in your browser, and compare them to corresponding .tpl files to see which one matches up, and which file controls what section of the page. We will cover this further in depth in Modifying content in template files.

Stylesheet

The "stylesheet" folder contains all the .css files that control the style elements of the store. Stylesheet.css is the main style sheet used to change the style elements of every layout. When constructing the theme's stylesheets, this will be the stylesheet we will refer back to. The slideshow and carousel module require their own style sheets, so they are included as slideshow.css and carousel.css.

Image

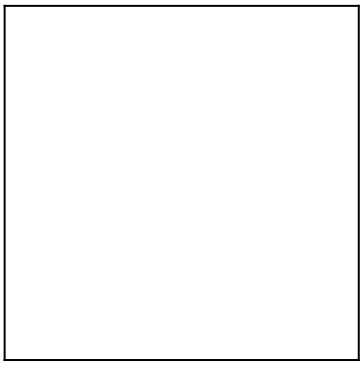
The image file is the location of all store images that are linked to in the .tpl files. These images include buttons, rating stars, and arrows seen in the store front. If you plan on adding your own buttons, this is the folder you would link them to in the HTML.

Creating our theme

Now that we are familiar with the general structure of the "default" folder, we are ready to create our own theme. Our strategy will be to grab and copy specific .tpl and .css files from the default theme, make our changes to the files to modify the store's style and structure, then upload them into our new theme folder in "catalog/view/theme". It is unlikely we will need to make a change to every .tpl or .css file to achieve the design we want, so only a small quantity of files will need to be uploaded and changed. Remember, the less files you modify to achieve your goal, the easier upgrades will be for your store.

Step 1: Create a new theme folder

Visit "/catalog/view/theme" in your store's root folder. You will see that the default theme is in this folder. You will need to create a new folder as the name of your theme. In this new theme folder, create a "template", "image", and "stylesheet" folder.



Whenever you add a file, you will need to mimic the correct file path as seen in the default folder. For example, if we chose to edit the "home.tpl" file, we would need to create a "template" folder, a "common" folder inside of that folder; then we can add the home.tpl to the "common" folder.Feel free to copy the stylesheet.css from the default folder and upload it to your new theme's "stylesheet" folder because you will need to edit this file at some point.

Override system

We will **not** modify any of the files in the actual "default" folder for our theme. These files need to be left untouched. When you are creating your theme you will need to download the file you want to modify, then upload it to its new file path in the new theme folder.

> When creating your theme, do not modify the default core files. This will make upgrades to your store more complicated and cause problems for later. Always download the file you need to modify from the "default" folder, then re-upload it to your new theme's location. Do not upload it to the default theme location or it will overwrite the default theme.

OpenCart can read your theme without needing to move or copy all the individual .tpl, .png, or .css core files to the New Theme folder. In other words, you only need to upload the files that you applied your theme changes with to your new theme folder. For example, if we only made changes to the stylesheet.css under default > stylesheet and the account.tpl under default > template > account, we would only need to upload those two files in our new theme folder. When we select the theme in the administration, OpenCart will override the

stylesheet.css and account.tpl files in the default with our modified files, but keep the default structure and style for the remaining files in the "stylesheet", "template", and "image" files.

Step 2: Link to New Theme's stylesheet.css in header.tpl

The header.tpl file will be the first file that needs to be modified. This file is originally located in the default theme under the default > template > common. If you open the file in a text editor, you will see that the the opening <html> tag, the complete <head></head> elements, and opening <body> tag is included. As you can see, the header.tpl is meant to be in virtually every page of your store. The <head></head> element currently provides a link to the stylesheet.css file in the default theme folder. If you want your theme to link to your new theme's stylesheet.css, you will need to change the file path from the default to your new theme here.

The header will need to be changed at line 19, specifically at "href=", to link to the new theme's stylesheet.css, from:

```
<link rel="stylesheet" type="text/css"
href="catalog/view/theme/default/stylesheet/stylesheet.css" />
```

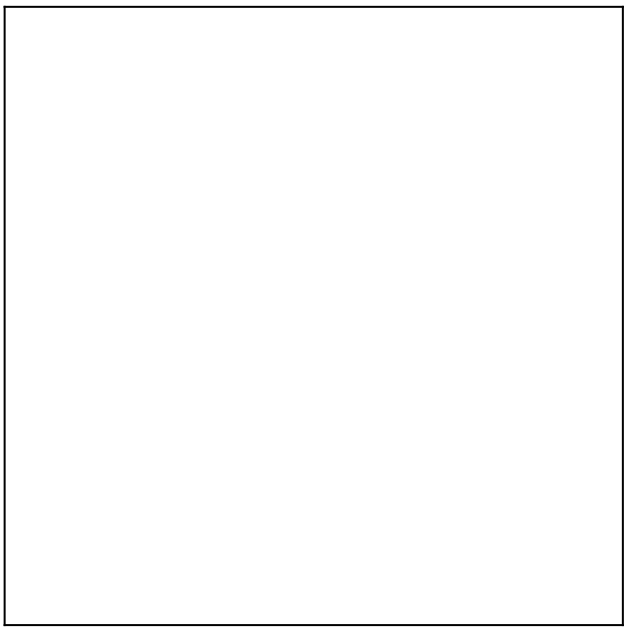
to:

```
<link rel="stylesheet" type="text/css"
href="catalog/view/theme/newtheme/stylesheet/stylesheet.css" />
```

"newtheme" will be replaced by the name of your custom theme's root folder. If you plan on editing the slideshow.css, ie7.css, ie6.css, or carousel.css, you will need to change those file path locations in the header.tpl as well.

Step 3: Enable the New Theme in the Administration

At this point we will want to enable our theme in the administration side of our store. Visit the Settings :: Store tab and select your new theme's folder name under template.



When we make our changes to .css or .tpl files in the new theme folder, we can view the results in our browser. When you refresh the home page for your store, you shouldn't see a difference from the default unless you

modified any files. This is the override system at work.

Step 4: Making Changes to the Stylesheet

It is important that we edit the style elements for our theme before we edit the HTML. To learn how to make changes to the stylesheet.css, see Making changes to the stylesheet.

Step 5: Modifying Content with Template Files (optional)

After we are finished with styling, we may want to change the structure or content of our HTML files. See Modifying content in template files for more information. This is the final step of creating a custom OpenCart theme. Remember, you can always revert back to the default theme by selecting it under the store settings under Settings in the administration.

None