

# TIPS AND TRICKS

## OpenCart 2.0.x



# #1

*Special thanks to Daniel Kerr, his team and  
the OpenCart community.*

Copyright © 2015 by iSenseLabs Ltd.

All rights reserved. No part of this publication text may be uploaded or  
posted online without the prior written permission of iSenseLabs.

For permission requests, get in touch with us on  
[sales@isenselabs.com](mailto:sales@isenselabs.com).

# Authors



**GEORGE**



**MIHAIL**



**ALEXANDER**



**NIKOLA**



**SIMEON**



**IVAILO**



**STOYAN**



**TEYA**



**JULIAN**



**JONIDA**



**EVGENI**

# Index

## Introductory

1.	Step-by-step OpenCart Migration to a New Server	5
2.	How to migrate from OpenCart 1.5.1.3+ to OpenCart 2.x with ExcelPort	14
3.	How to solve OpenCart 2 issue when FTP support is disabled?	19
4.	How to install or delete modules in OpenCart 2.0	22
5.	How to configure modules and assign them to layouts in OpenCart 2.x?	30
6.	How to add Google Analytics to OpenCart 2.0.x	37
7.	How to set up multi-store in OpenCart 2.0.x	44

## Tweaks

8.	How to add a Notification bar in your store and admin panel in OpenCart 2.x	50
9.	How to display price of an item as FREE instead of 0.0	57
10.	How to remove 'Compare this Product' and 'Add to Wish List' in OpenCart 2.0	61
11.	How to add store logo and product images to invoices in OpenCart 2.x	67

12.	How to add a Scroll-to-top button in OpenCart 2.x	74
-----	---	----

## **Security**

13.	Restrict admin access to your IP addresses	83
14.	Password protect the OpenCart admin	85
15.	Tip for Making OpenCart More Secure	88

## **Dev**

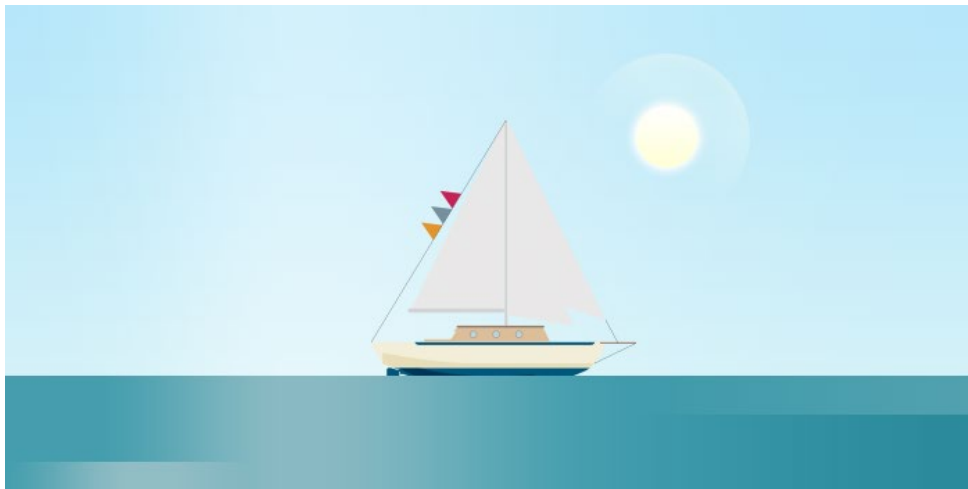
16.	Using the new API methods of OpenCart 2.x	94
17.	OpenCart 2.0.x Event System Tutorial	100

## **Bonus**

18.	Supercharge OpenCart with OpenPack	107
-----	------------------------------------	-----

# Step-by -step OpenCart Migration to a New Server

OpenCart is one of the simplest and loveliest e-commerce platforms. In this blogpost we want to present a detailed step-by-step guide of the necessary steps you need to take in order to migrate your OpenCart store to a new server.



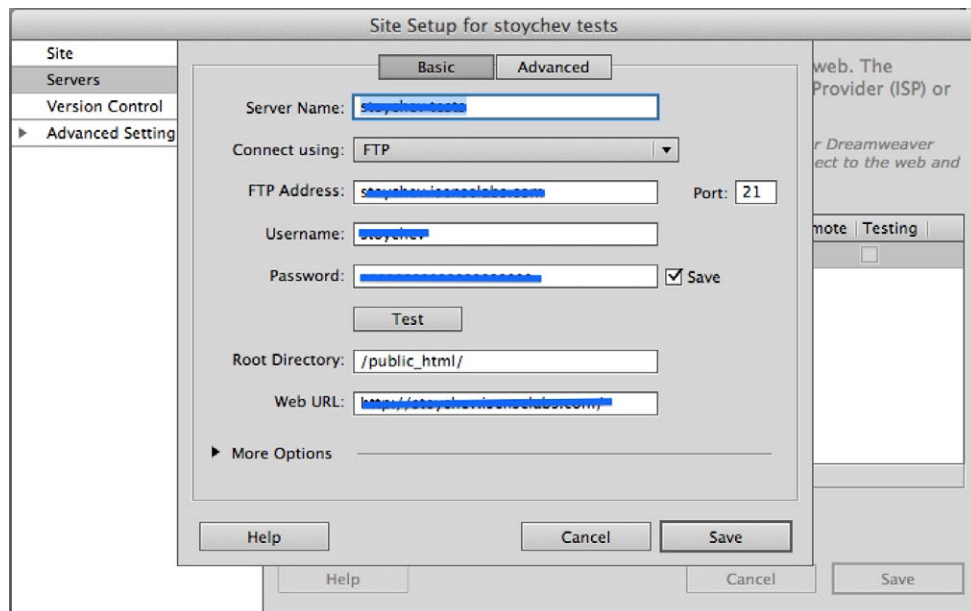
## Setting up our station

Before we begin with our guide, it is best to do some prep and set up our station. As a first, we always recommend backing-up of your OpenCart store and database. Some hostings offer automatic back-ups, so in case your hosting provides this feature, you can

fast forward to Importing your OpenCart database. If your hosting does not come with automatic back-up please read on.

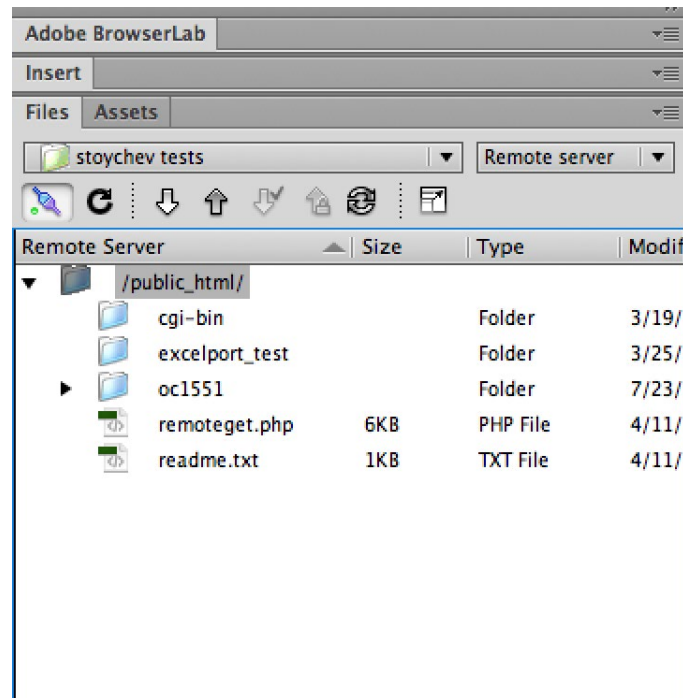
## Backing up our OpenCart store contents

As a first, you will need to connect to your host via FTP. In this example we will use Adobe Dreamweaver, but if you are looking for an alternative, Aptana, FileZilla or any other FTP client will do the job as well.



This is a familiar picture of how things look when connected to your FTP server. You will immediately spot the public\_html folder (at the top of the page). In case you are wondering what this public\_html is, it is the web root for your primary domain name. It serves as a master cabinet where you put all website files and folders. Double click on the public\_html so we reveal its contents. Now, we need to locate our OpenCart installation folder. In this example we have named it oc1551 folder.

\* Disclaimer: In some cases there may not be a public\_html folder or it may be named differently.



Double click on the OpenCart install folder we will select all of the files in the directory and get them. By “get” we mean that we will get them from your server and copy them to your local environment. To do this, can click on Ctrl + A/ CMD + A on a Mac and Get/ Copy the files. To have everything organized neatly, we recommend creating a folder and pasting the files there in the folder.

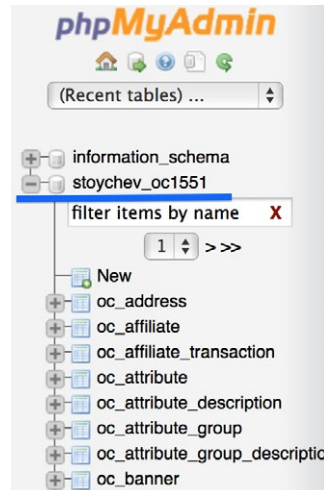
## Backing up our MySQL

Next, we will need to get a copy of your database. A database is an electronic filing system allowing a user to quickly pull out content out of it. We will use phpMyAdmin in this scenario, which is available in your Cpanel. Cpanel is a graphical user panel from where we are going to access phpMyAdmin. Your hosting should provide

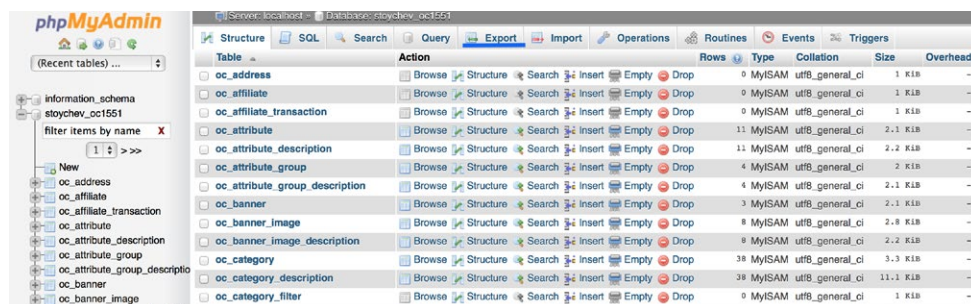


you with access information to your Cpanel.

Once you login at your Cpanel, you need to go to phpMyAdmin. After this, please select the database name from the left hand menu.



Next, you will see the tables of the database. Right after we verify that we have selected the correct DB (Server name and Database are listed on the top), we need to click on Export.



Then, we need to select Custom Export. We need to make sure all of the tables are selected and proceed to the next title Output:

\* Disclaimer: In some cases there may not be a public\_html folder or it may be named differently.

Exporting tables from oc\_product\_catalog database

Export Method:

☐ Quick - display only the minimal options  
☒ Custom - display all possible options

Table(s):

Select All / Unselect All

oc_product_description
oc_product_discount
oc_product_filter
oc_product_image
oc_product_option
oc_product_option_value
oc_product_related
oc_product_reward
oc_product_special
oc_product_to_category

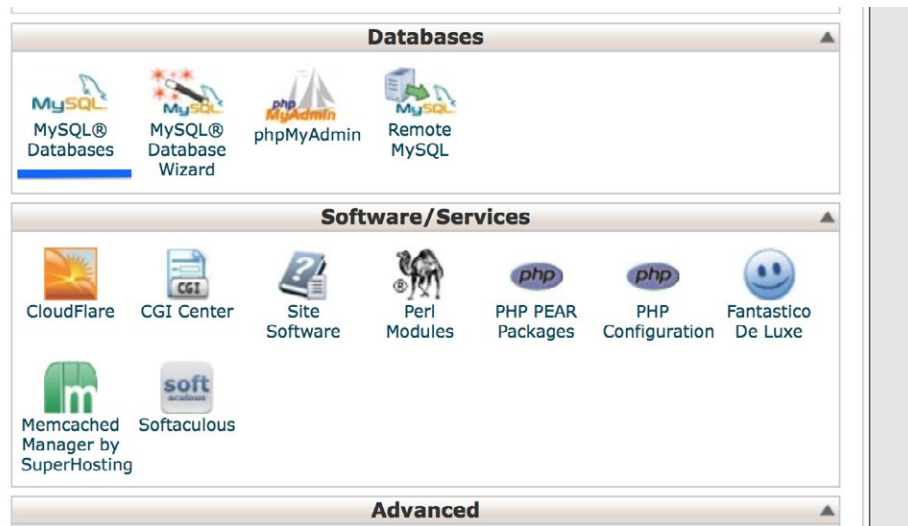
After we are done with the above listed steps, we need to go all the way down and click on Go. Disregard other options as we do not need them for now. This will finalize the back-up of your website and DBs so now we are ready to upload files to the new server and create a new database. Get ready.

## Uploading OpenCart to the new hosting

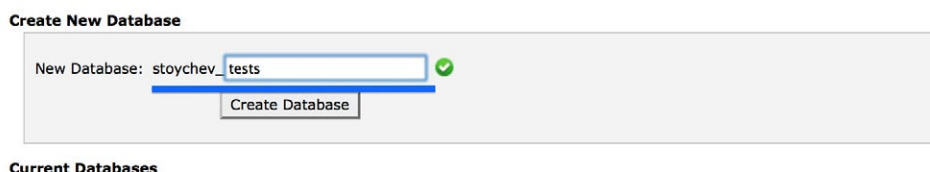
Remember the folder that we have stored your website? We need to open it up and upload the information back on the new hosting you have purchased. To do this we will need to connect to your new FTP server. Please fill in your username and password. Once we connect, we have to expand on the public\_html folder and use “Put” button to upload the files. Once all files are uploaded we may proceed to the next step.

## Creating MySQL Database

To create a MySQL Database we need to login to the Cpanel of the new hosting. Make sure you use the new hosting details. After we login we need to create a database name.



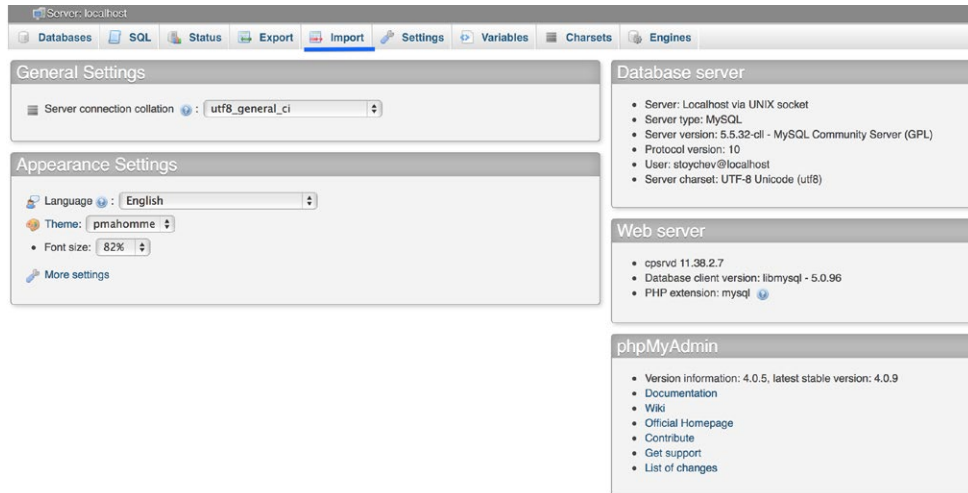
To do that click on the MySQL Databases icon under the Database section. Click on the Create new and give your database an unique name.



Once you do this we need to proceed to phpMyAdmin. There we will import the backed up database.

Once we are in phpMyAdmin, we need to make sure the database we created is present. To do that just make sure you check your left column. To make sure it is empty you can click on the database name. After we have made sure the database is created and empty

we need to go to the Import tab.



Next we need to click on Browse and select our backed up database and upload it.

#### Importing into the current server

**File to Import:**

File may be compressed (gzip, bzip2, zip) or uncompressed.  
A compressed file's name must end in `.{format}.[compression]`. Example: `.sql.zip`

Browse your computer:  no file selected (Max: 50MiB)

Character set of the file:

---

**Partial Import:**

☒ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. (This might be a good way to import large files, however it can break transactions.)

Number of rows to skip, starting from the first row:

Please be patient as the larger database, the more time it will require to upload.

✔ Import has been successfully finished, 183 queries executed. (stoychev\_oc1551.sql)

Importing into the database "stoychev\_tests"

## Editing Config files

Next stop, we need to edit the config.php and admin/config.php. To do so open Dreamweaver/Aptana, FileZilla or the FTP client you are using and connect to your new server. You will find config.php in the parent OpenCart directory. Open the config.php for editing.

Now, we will need to edit the following lines of code (9-18) in our example, located under // DIR:

```
define('DIR_APPLICATION', '/home/system username/public_html/oc1551/catalog/');
define('DIR_SYSTEM', '/home/system username/public_html/oc1551/system/');
define('DIR_DATABASE', '/home/system username/public_html/oc1551/system/database/');
define('DIR_LANGUAGE', '/home/system username/public_html/oc1551/catalog/language/');
define('DIR_TEMPLATE', '/home/system username/public_html/oc1551/catalog/view/theme/');
define('DIR_CONFIG', '/home/system username/public_html/oc1551/system/config/');
define('DIR_IMAGE', '/home/system usernamepublic_html/oc1551/image/');
define('DIR_CACHE', '/home/system username/public_html/oc1551/system/cache/');
define('DIR_DOWNLOAD', '/home/system username/public_html/oc1551/download/');
define('DIR_LOGS', '/home/stoychev/system username/oc1551/system/logs/');
```

**\*Notice:** Please put your website Cpanel username under system username.

Next, we need to edit the Database entries lines (21-26):

```
define('DB_DRIVER', 'mysql');
define('DB_HOSTNAME', 'localhost');
define('DB_USERNAME', 'stoychev_test');
define('DB_PASSWORD', '123456789');
define('DB_DATABASE', 'stoychev_oc1551');
define('DB_PREFIX', 'oc_');
```

Please fill in your USERNAME, PASSWORD and DATABASE under the 'DB\_USERNAME', 'DB\_PASSWORD' and "DB\_DATABASE".

After we are done editing the config file we need to right click on config.php and select Set Permissions. We recommend setting the permissions of config.php to 444. This will make the file as read-only.

Next stop, we need to repeat the same procedure with admin/config.php file. We will find the file under admin/config.php. Again we will edit the database settings and change the permissions of the admin/config.php file to 444.

## Update of Name Servers

As a last step we should update your name servers. Name servers configure the domain to point to the right host. Most hosting providers have their own name servers. The easiest way to go about this is to contact your hosting registrars and simply ask them to handle this for you. Please mind that this can take up to 48 hours to change.

So how did your OpenCart migration go? Please let us know if there is anything we can help you with or if you have any questions or comments.

# How to migrate from OpenCart 1.5.1.3+ to OpenCart 2.x with ExcelPort

[ExcelPort by iSenseLabs](#) is an excellent tool if you have just upgraded your OpenCart store and need a way to transfer your data from your old OpenCart 1.5.1.3+ store to your shiny new OpenCart 2.x. Follow this step-by-step tutorial to learn how to do it. Let's go!



Requirements for the old OpenCart 1.5.1.3+ store:

- A running installation of the latest ExcelPort **1.x** (be careful not to install ExcelPort **2.x**) Please make sure to upgrade to the latest available version of ExcelPort 1.x.

Requirements for the new OpenCart 2.x store:

- A running installation of the latest ExcelPort **2.x** (be careful not to install ExcelPort **1.x**) Please make sure to upgrade to the lat-

est available version of ExcelPort 2.x.

**Important:** This tutorial assumes that your new store is a fresh installation of OpenCart. If your new store is already populated with *Products, Categories, Options, Attributes, Attribute groups, Customers, Customer groups* and *Orders*, following this tutorial WILL delete this information in order to successfully import your data.

**Important:** This tutorial divides the migration progress in three main stages, which should be followed consecutively:

1. Migrating products
2. Migrating customers
3. Migrating orders

The idea is to **not** migrate your orders before you have first migrated your customers and products.

**Note:** if you wish to migrate **only** your products, you can follow only Stage 1.

**Note:** If you wish to migrate **only** your customers you can follow only Stage 2, skipping Stage 1

This tutorial assumes that you are already familiar with the interface of ExcelPort. If not, feel free to click around our ExcelPort demo page to get acquainted:

<http://excelport.demo.isenselabs.com/admin> (user: **demo** pass: **demo**)

## STAGE 1: Migrating products



To migrate your products, you will first need to migrate the *Categories, Attributes, Attribute Groups, Options* and *Customer Groups*.

Here is a video tutorial, which explains the full product migration: [watch video](#).

In your old website visit:

*Admin > Extensions > Modules > ExcelPort 1.x [Edit] > Export*

... and do separate exports of:

1. Categories
2. Customer Groups
3. Attributes and Attribute Groups
4. Options
5. Products (making sure to keep the “Quick Export” checkbox **unchecked**)

Next, in the OpenCart 2.x website, go to:

*Admin > Extensions > Modules > ExcelPort 2.x [Edit] > Import*

Mind the checkboxes above the “Import Data” button. Before doing your imports, please make sure to **check** the checkbox “*Delete the entries before doing the import*”

This will tell ExcelPort to first delete your existing data, after which it will do the import. Do separate imports in the following order:

1. Categories
2. Customer Groups
3. Attributes and Attribute Groups
4. Options
5. Products

If you wish to migrate customers and orders, you can move to the

next stages. Otherwise, you can skip to the end of the article.

## STAGE 2: Migrating customers

**Important:** Note that OpenCart have changed the encryption of the passwords after version 1.5.4.1. This means that if you transfer customers between a version of OpenCart, older than 1.5.4, and 1.5.4.1 – 2.x then the passwords will not work and the customers will need to use the “*Forgotten password*” feature. The passwords will work just fine if you transfer between OpenCart 1.5.4.1+ and OpenCart 2.x.

To migrate your customers, you will first need to migrate the *Customer Groups*.

Here is a video tutorial, which explains the full customers migration: [watch video](#).

In your old website visit:

*Admin > Extensions > Modules > ExcelPort 1.x [Edit] > Export*

... and do separate exports of:

1. Customer Groups (you can skip this export if you have already done it in the previous Stage)
2. Customers

Next, in the OpenCart 2.x website, go to:

*Admin > Extensions > Modules > ExcelPort 2.x [Edit] > Import*

Mind the checkboxes above the “*Import Data*” button. Before doing your imports, please make sure to check the checkbox “*Delete the entries before doing the import*” This will tell ExcelPort to first delete your existing data, after which it will do the import.

Do separate imports in the following order:

1. Customer Groups (you can skip this import if you have already done it in the previous Stage)
2. Customers

## STAGE 3: Migrating orders

If you have not already followed Stage 1 and Stage 2, please do it now.

Here is a video tutorial, which explains the migration of the orders: [watch video](#).

In your old website visit:

*Admin > Extensions > Modules > ExcelPort 1.x [Edit] > Export*  
... and export your Orders.

Next, in the OpenCart 2.x website, go to:

*Admin > Extensions > Modules > ExcelPort 2.x [Edit] > Import*

Mind the checkboxes above the “*Import Data*” button. Before doing your imports, please make sure to **check** the checkbox “*Delete the entries before doing the import*” This will tell ExcelPort to first delete your existing data, after which it will do the import.

Now import your Orders.

## Conclusion

Congratulations! You have successfully migrated your store. If you have any questions, feel free to open up a thread in our community forums <http://isenselabs.com/forum>, or in our support system <http://isenselabs.com/users/support>.

# How to solve OpenCart 2 issue when FTP support is disabled?

OpenCart 2.0 is out and there are a couple of things that need a little help to work as intended. For instance If you get an error below saying “Could not connect as .....” while uploading your zipped extensions via the Extension Installer, you probably have the FTP support disabled from your hosting. We will offer two ways to solve this - one for *tech-savvy people* who are comfortable with editing files on their FTP and one for *store owners* who can run it via the OC Installer.

## Quick Fix

### 1. Here is how to fix this? (Developer version)

Open up Notepad++, Dreaweaver or any other editor and go to *admin/controller/extension/installer.php* and before the line saying:

```
public function unzip() {
```

## paste the following code

```

public function localcopy() {
    $this->load->language('extension/installer');

    $json = array();

    if (!$this->user->hasPermission('modify', 'extension/installer')) {
        $json['error'] = $this->language->get('error_permission');
    }

    $directory = DIR_DOWNLOAD . str_replace(array('../', '..\\', '..'), '', $this->request->post['path']) . '/upload/';

    if (!is_dir($directory)) {
        $json['error'] = $this->language->get('error_directory');
    }

    if (!$json) {
        // Get a list of files ready to upload
        $files = array();
        $path = array($directory . '*');
        while (count($path) != 0) {
            $next = array_shift($path);
            foreach (glob($next) as $file) {
                if (is_dir($file)) {
                    $path[] = $file . '/*';
                }
                $files[] = $file;
            }
        }

        $root = dirname(DIR_APPLICATION) . '/';
        foreach ($files as $file) {
            // Upload everything in the upload directory
            $destination = $root.substr($file, strlen($directory));

            if (is_dir($file)) {
                $list = glob(rtrim($destination, '/') . '/*');
                if (!file_exists($destination)) {
                    if (!mkdir($destination)) {
                        $json['error'] = sprintf($this->language->get('error_ftp_directory'), $destination);
                    }
                }
            }
        }
    }
}

```

```
        if (is_file($file)) {  
            if (!copy($file, $destination)) {  
                $json['error'] = sprintf($this->language->get('error_ftp_file'), $file);  
            }  
        }  
    }  
    $this->response->addHeader('Content-Type: application/json');  
    $this->response->setOutput(json_encode($json));  
}
```

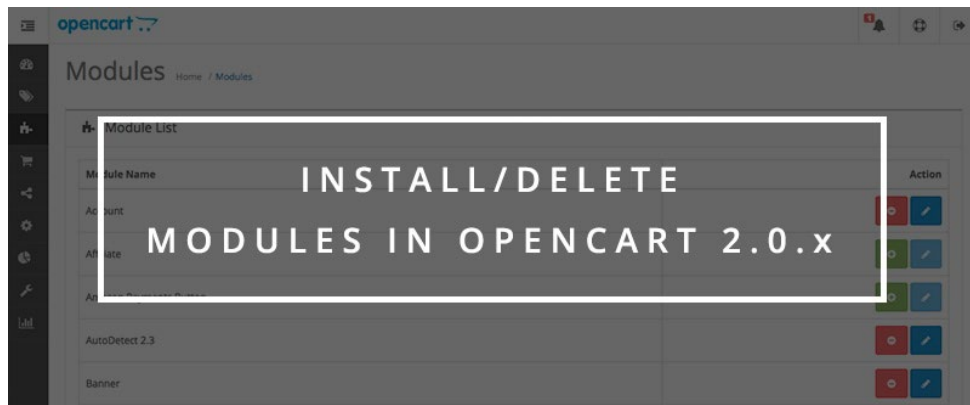
## 2. Here is how to fix this? (Store owner version, set up in less than 30 seconds)

As an alternative you can just download this OCMOD ready modification file, which we contributed to the OpenCart society and just follow the installation instructions : [http://www.opencart.com/index.php?route=extension/extension/info&extension\\_id=18892](http://www.opencart.com/index.php?route=extension/extension/info&extension_id=18892)

That's it, you can now enjoy the module installer! Sharing is caring so spread the word. Thanks!

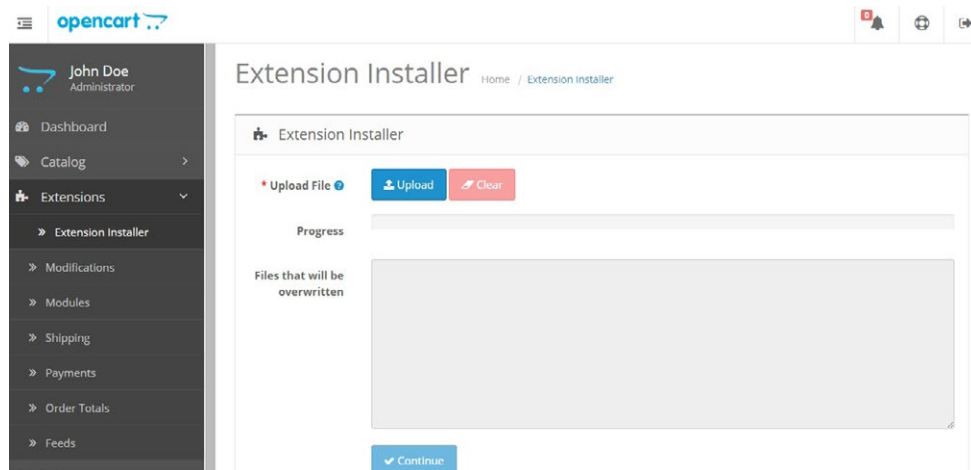
# How to install or delete modules in OpenCart 2.0

OpenCart modules are the tools which allow us to customize and extend our online store functionality. While in OpenCart 1.5.x you had to manually install the module via a FTP client, OpenCart 2.0x introduced the Extension installer, which allows you to complete the whole process through your dashboard with a few simple clicks. This tutorial will show you how to install and uninstall / delete a module on your OpenCart system.

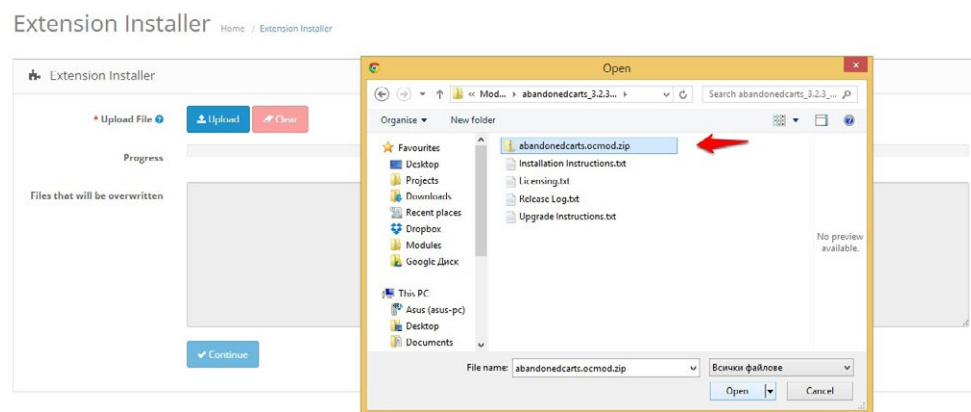


## Installing a module

Lets start by logging into your store admin panel. Navigate to Extensions -> Extension installer.



Click on the upload button. A dialog box should open.

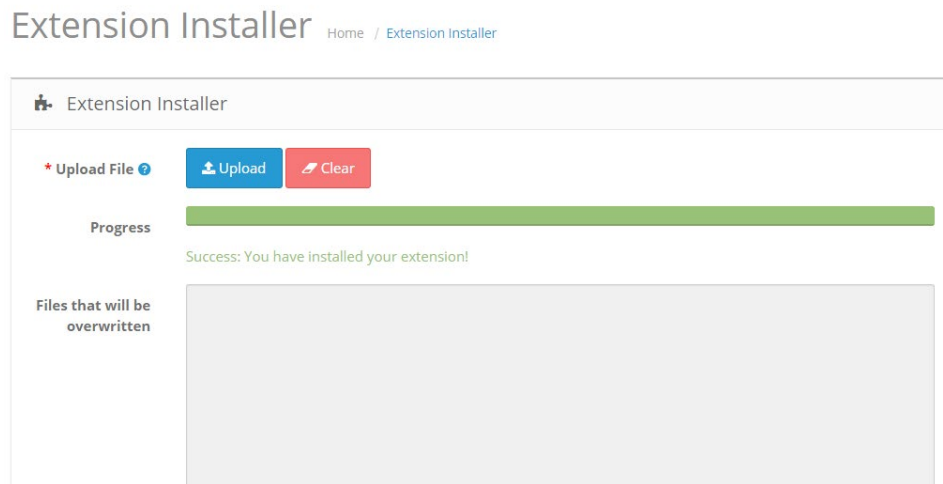


Locate the installation folder of the extension you are going to install and select it. For this tutorial I will be using [AbandonedCarts](#). Select the necessary .zip file and click “OK”

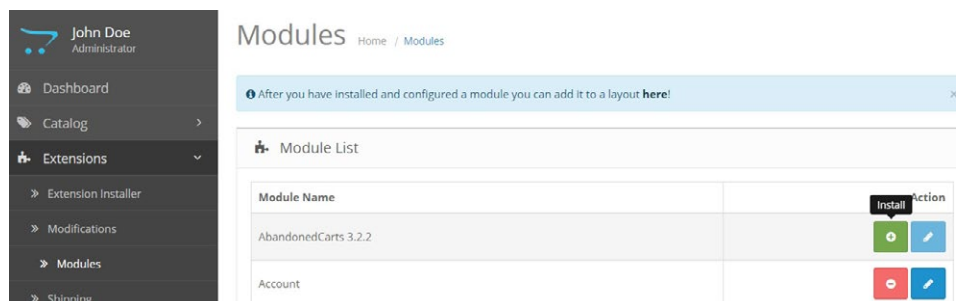
*Note: It is not uncommon for OpenCart users to receive an error “Could not connect as .....” while uploading files through the Extension installer. Most often this is due to a disabled FTP access from your hosting provider or FTP settings in general. Here is a nice blog-post which will help you resolve this issue - <https://isenselabs.com/posts/how-to-solve-opencart-2-issue-when-ftp-support-is-disabled>*



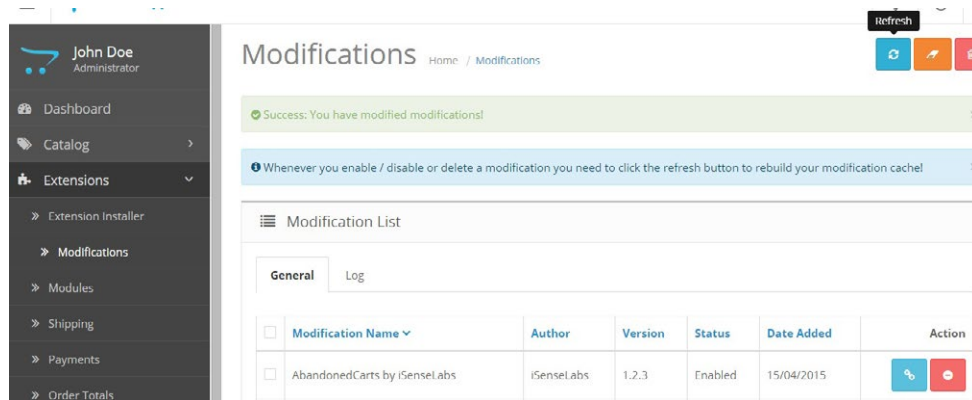
After clicking “OK” your extension will be uploaded and a “success” message should appear.



Now your module should be visible in **Extensions -> Modules**. After locating it in the Module list just click the install button (“+” sign)



The final step of the installation process is to apply the changes we have just made. In order to do so, go to **Extensions -> Modifications** and click the Refresh sign at the upper right corner of the page.



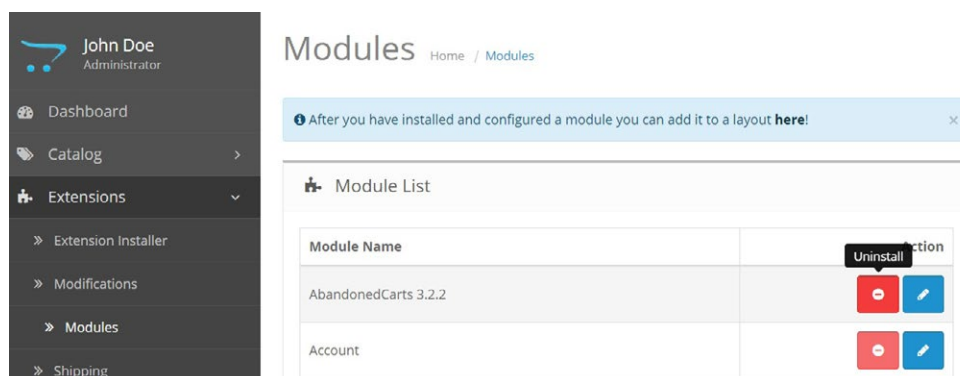
Installing a module in OpenCart 2.x using the Extension installer is easy and intuitive.

## Uninstalling / deleting a module

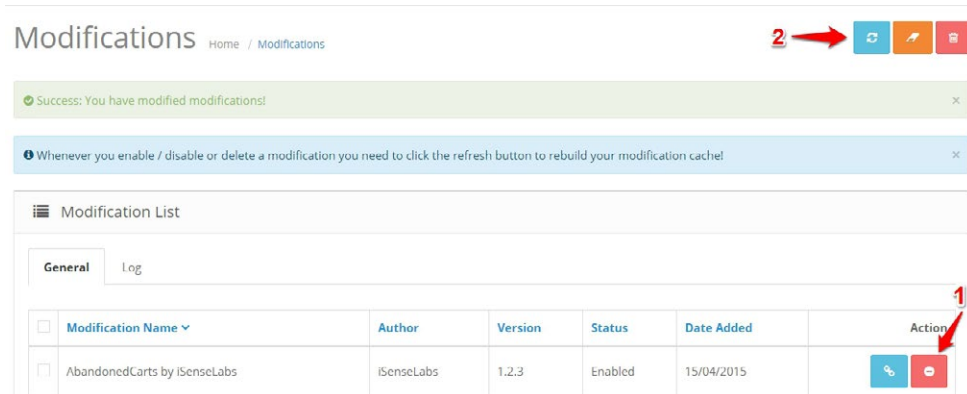
There is a difference between these two terms when it comes to OpenCart modules and we will cover them separately.

### Uninstalling a module

Navigate to **Extensions -> Modules**, locate the module you wish to uninstall in the Modules list and click on the uninstall button (“-” minus sign)



To apply the changes you have just made go to **Extensions** -> **Modifications**, click on the Uninstall button next to the module you are uninstalling and on the Refresh sign at the upper right corner of the page.



The module will be uninstalled.

## Deleting a module

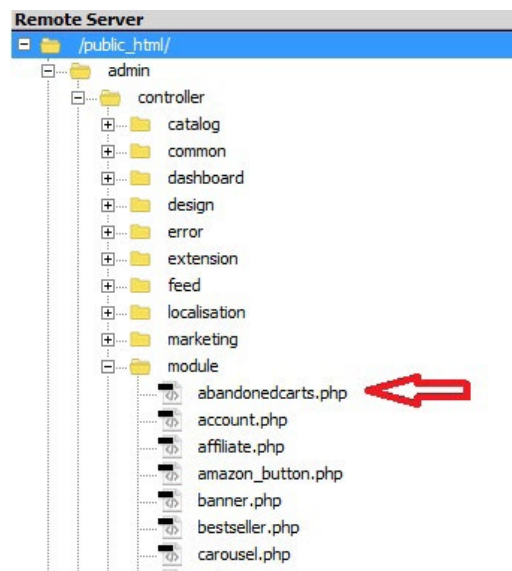
*Note: In this tutorial the module I will delete is AbandonedCarts. Please bear in mind that different modules have different file structures, so their files will most probably be located in other folders in your main installation directory. The basic rule is to delete the files corresponding to the ones you have uploaded during the installation process of the module (i.e the files in the upload folder in the .zip)*

Completely deleting a module from OpenCart requires a bit more complex actions which include deleting all of the module files from your OpenCart installation directory. You can do that either through your favorite FTP client or through the web hosting account control panel. Either way the process is identical and all you have to do is locate the files of the module you want to delete in

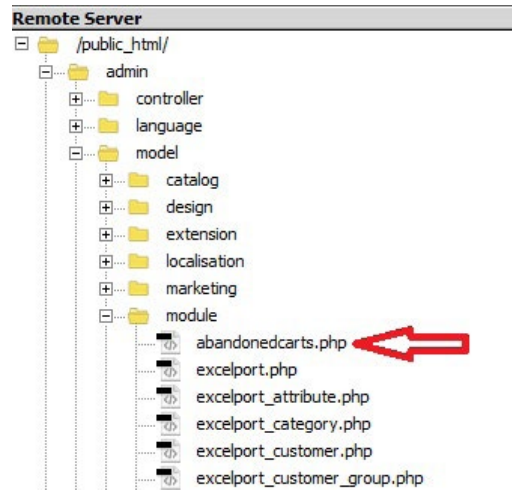
your OpenCart root directory.

By default, your OpenCart installation should be located in the **public\_html** directory. The files which we are looking for are located in the **admin** and **catalog** folders.

Let's start with the admin folder. Navigate to **public\_html/admin/controller/module** and locate the file which has your module's name and file extension .php and delete it. In our example that would be **abandonedcarts.php**.

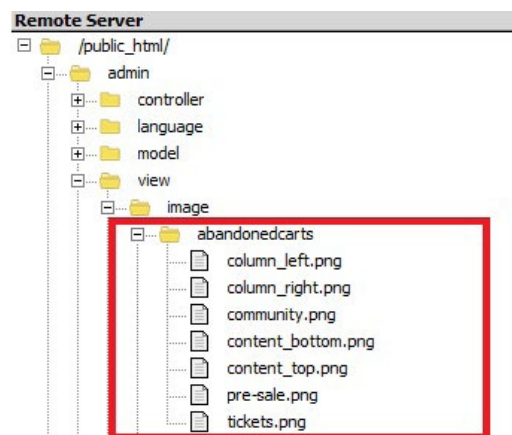


Next go to **public\_html/admin/model** and delete the corresponding to your module file.



We have covered the controller and model folders so now we have to do the same with the view folder.

Navigate to **public\_html/admin/view/image/modulename** and delete all of the files corresponding to the module.



Following these paths we will delete the rest of the files associated with the module -

**admin/view/template/module/; admin/view/template/module/modulename ; admin/view/javascript/modulename ; admin/view/stylesheet/; catalog/view/theme/yourtheme/template/module/; catalog/view/theme/default/template/module/.**

After you have located and deleted the corresponding to your module files in the admin folder, you should do the same in the catalog folder. The files should be located in a folders with similar to the following paths:

**catalog/controller/module/modulename.php**

**catalog/language/english/module/modulename.php**

**catalog/model/module/modulename.php**

After deleting all of the files used by the module it should be completely removed from your system.

While the steps in this tutorial can serve as a building ground, if you are unsure about deleting anything we would always recommend to do a backup of your site and/or get in touch with the module developer.

# How to configure modules and assign them to layouts in OpenCart 2.x

Since OpenCart 2.0x has completely redesigned admin panel, there are some UI changes that might make you feel not at home. While the visually new modules and layout system may seem a challenging concept for the OpenCart 2.x user, it is more or less the same as in the earlier versions. In this blog post we will walk you through the essentials and shed some light on how to configure modules and assign them to layouts.

This guide assumes that you already have a working OpenCart 2.x.

Amazon Payments Button	 
Banner	 
Banner > Category	 
Banner > Test	 
Bestsellers	 

## Module vs. Extension

In OpenCart terms, an extension is a collection of files which extend/add some OpenCart functionality. Extensions can usually be downloaded from the OpenCart extension store.

A module is a group of settings related to an extension. The modules are assigned to layouts and positions.


## Creating a new module

You can create any number of modules with specific settings. In OpenCart 2.x the modules can be assigned a name which is later on used to help you identify the different sets of settings when assigning the modules to the desired layouts and positions. We will walk you through the process of creating a module and assigning it to a layout. We will use the integrated Banner extension in OpenCart for this example.

In order to access the banner extension, go to the Admin panel of your store, click on ***Extensions*** > ***Modules*** > ***Banner*** edit button. Here is how it looks like:

**Banner** [Home](#) / [Modules](#) / [Banner](#)

---

 Edit Banner Module

Module Name	<input type="text" value="Test"/>
Banner	<input type="text" value="Home Page Slideshow"/>
Width	<input type="text" value="300"/>
Height	<input type="text" value="300"/>
Status	<input type="text" value="Enabled"/>



## Filling the fields

Next, we should populate the module's fields, but first here is a bit of info on what each of the fields purpose is:

**Module Name:** The name, which will be used for the newly created module. Later, you can see this name in the Modules page and when you add it to a layout. We suggest you to write a unique and descriptive name, which you can easily recognize later.

**Banner:** A dropdown list with the available banners. These banners are set from **System > Design > Banners**.

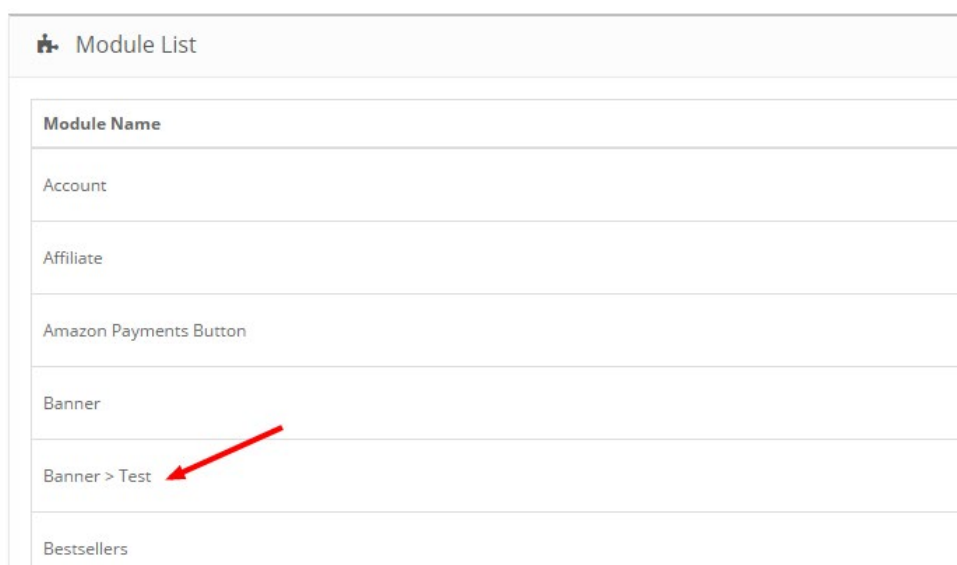
**Width:** The width in pixels that the banner slider will take.

**Height:** The height in pixels that the banner slider will take.

**Status:** Enable or disable the current module.

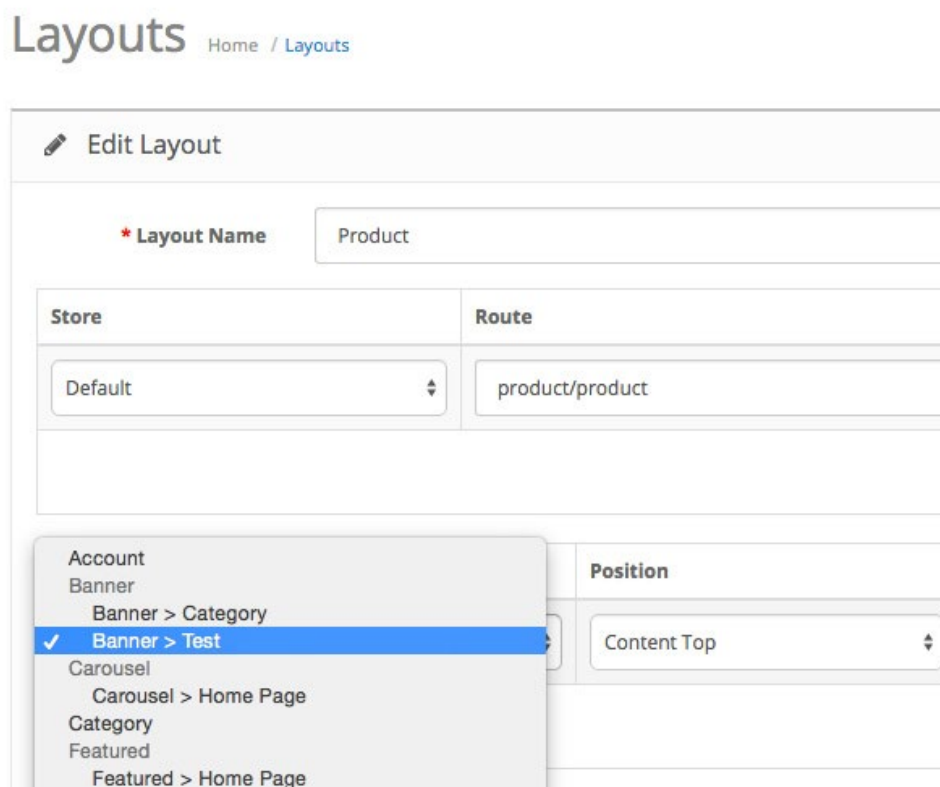
*Tip: Please mind, that these options are module specific and may vary across the different modules.*

When you are ready with the setup, click Save. Once you press Save, you will be redirected to the Modules page where a new entry has been added.



Module Name
Account
Affiliate
Amazon Payments Button
Banner
Banner > Test
Bestsellers

After you have set up the module correctly, you can assign it to a specific layout from **System > Design > Layouts**. This is done by choosing a layout and adding the module we created as presented in the screenshot below (notice how the module name is displayed next to the extension name). This will connect the module and the front-end, making your banner visible on the selected layout. After we are done here, we need to once again click Save.




## Layout Override

The layout override functionality is a great way for users looking to create a custom product layout. For example, let's say that we want to add a banner to one of the default products in OpenCart - Canon EOS 5D. To do that, we have to create a new layout or modify an existing one. In this example we will use the default Manu-

facturer layout. You can get to the Manufacturer layout from **System > Design > Layouts**.

In the first step we created a **Banner > Test module** that we will have to use now. To do this, click on the plus button, select the module and adjust the settings as shown in the screenshot below.

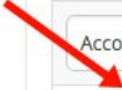
Layouts [Home](#) / [Layouts](#)

 Edit Layout

\* Layout Name

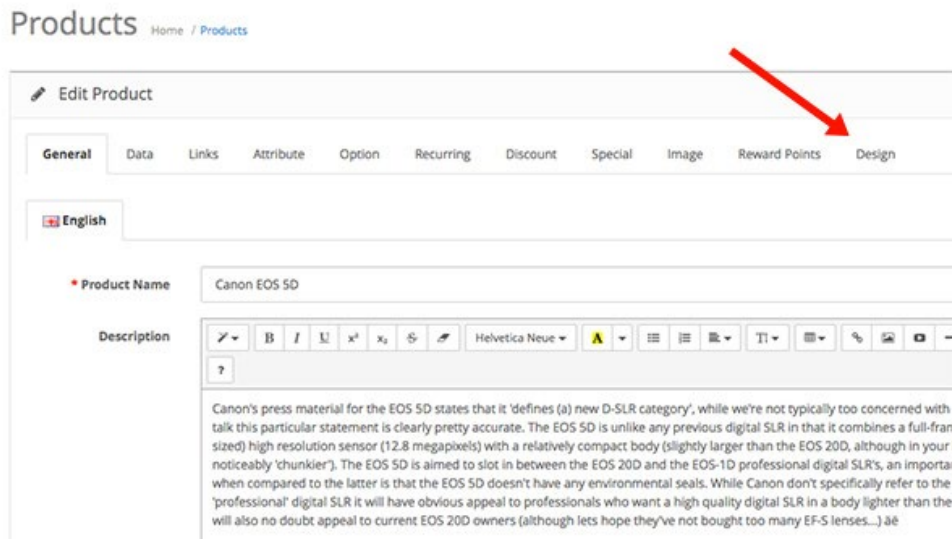
Store	Route
Default	product/manufacturer

Module	Position	
Account	Column Left	2
Banner > Test	Content Top	1

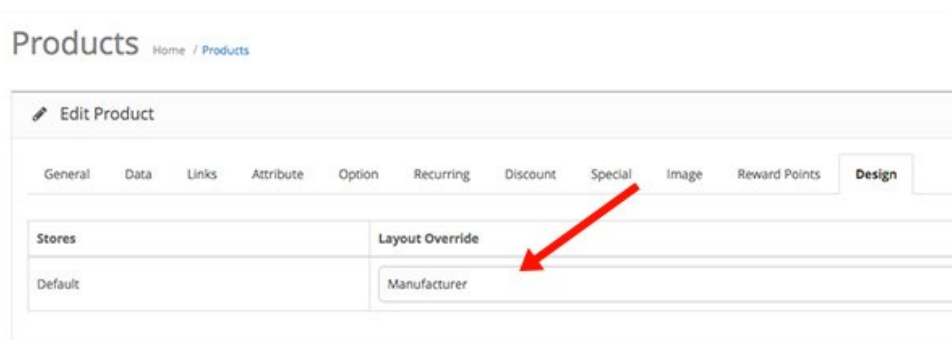


Now we need to save the layout settings. Next stop is editing the product's settings and overriding the standard product layout.

In order to do that, go to **Catalog > Products > Canon EOS 5D** and click the edit button.

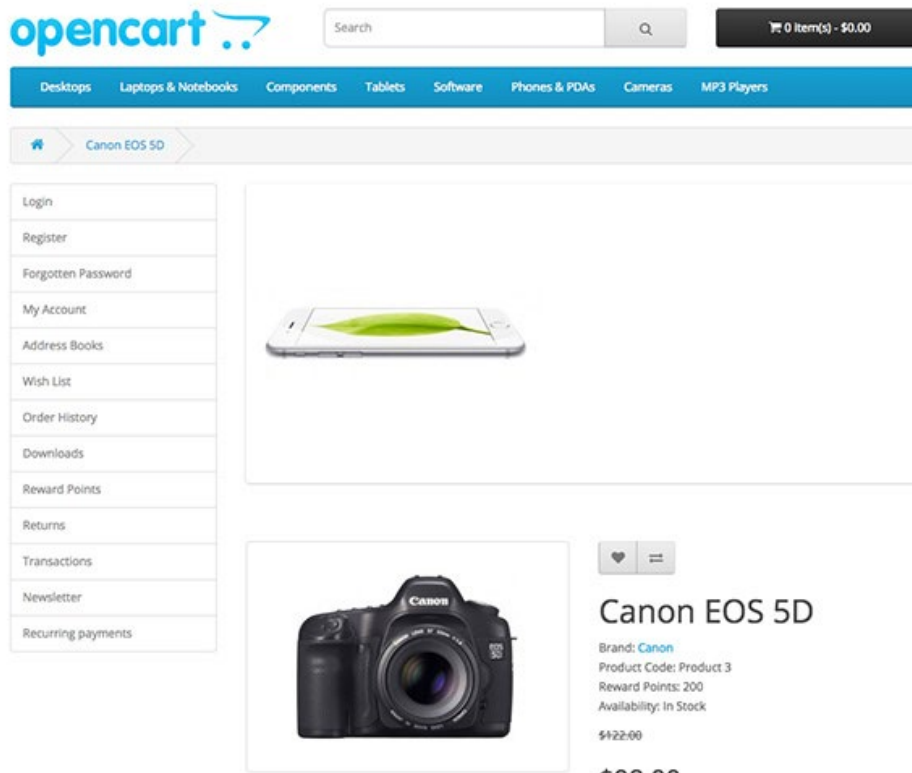


The product page will load. There we need to click on the Design tab.



OpenCart © 2009-2014 All Rights Reserved.  
Version 2.0.1.1

On the design tab we will see a list of all of our stores. In this example we have set up only 1 store, called Default. There we have to select which layout we wish to override the default one. For the sake of our example we will select the Manufacturer layout. Once we are done here, we need to click on Save in the top right corner. To see the changes, we will go to the store front and select the edited product, Canon EOS 5D.



Here is how the new overridden layout looks like. Conversely, you can spot the difference when compared to any other product using the standard layout.

## Conclusion

The layout system in OpenCart 2.0 provides the same powerful feature and the same level of flexibility as in OpenCart 1.5.6x. The only difference is how the modules are organized and how we assign them to layouts. We don't have to be afraid of the new things, right, so it is a matter of time to get used to them.

# How to add Google Analytics to OpenCart 2.0.x

Is your webstore as successful as you want it to be? Are you using marketing tactics to generate more traffic and customers in your store? In order to be successful you need to have the right information, analyze it and produce the adequate strategies to achieve the set goals. To ensure, that your website is achieving the results you are expecting you can use web analytics services. One of the most powerful tools for monitoring and analysing traffic on your website is the [freemium](#) Google Analytics. In this tutorial we will show how to add Google Analytics in your OpenCart 2.0.x webstore.

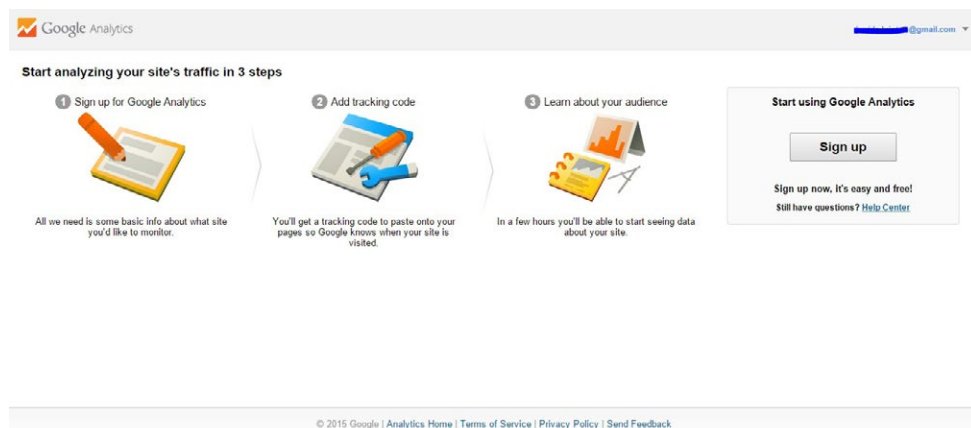


# Configuring Google Analytics in OpenCart

In order to integrate Google Analytics in OpenCart 2.0.x, we will go in a three steps process:

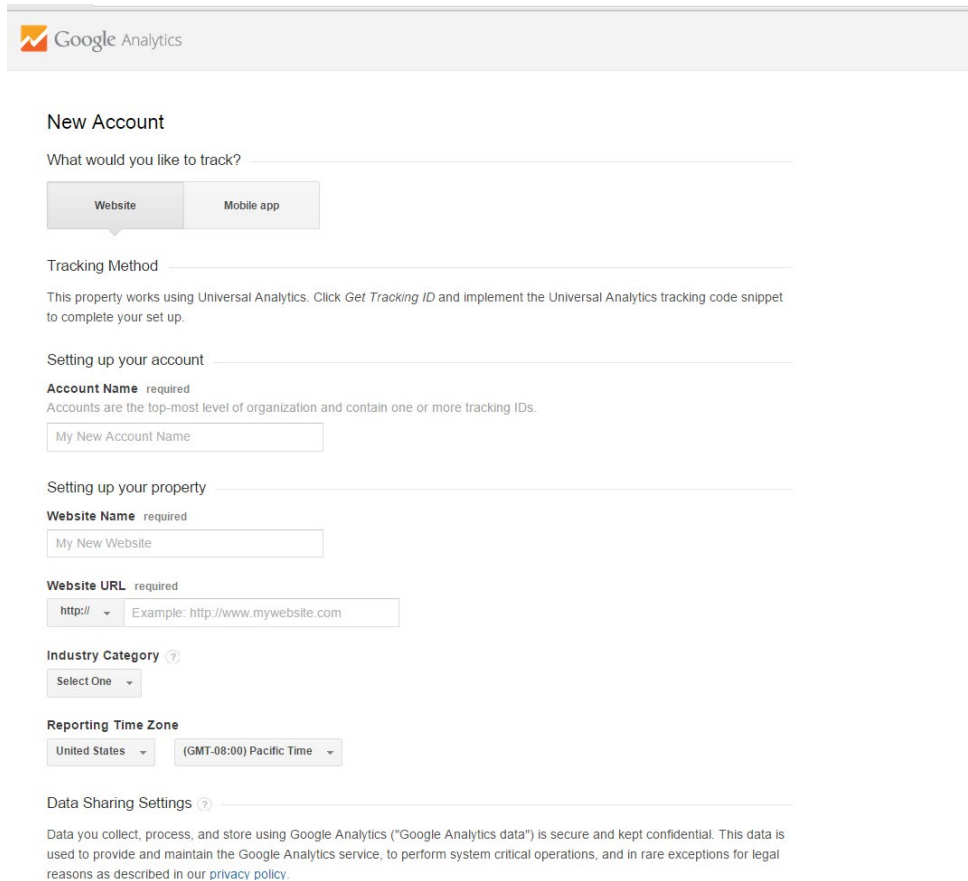
- Sign Up for a Google Analytics account
- Add Tracking Code
- Make sure you are collecting data

After this tutorial, you will be ready to start analyzing your tracked store data. Let's start with creation of Google Analytics account.



## 1. Sign up for Google Analytics

In order to access Google Analytics you need to have a Google account. With this account you can create a new Google Analytics account. Visit <https://www.google.com/analytics/>, click the Access Google Analytics button if you already have a Google Analytics account, otherwise click Create an account button (top right), and follow the on-screen instructions.



The screenshot shows the Google Analytics 'New Account' setup form. At the top is the Google Analytics logo. Below it is the 'New Account' heading. The first section, 'What would you like to track?', has two buttons: 'Website' (selected) and 'Mobile app'. The 'Tracking Method' section states that the property works using Universal Analytics and provides a link to 'Get Tracking ID'. The 'Setting up your account' section includes a required 'Account Name' field with the placeholder 'My New Account Name'. The 'Setting up your property' section includes a required 'Website Name' field with the placeholder 'My New Website', a required 'Website URL' field with a dropdown set to 'http://' and a placeholder 'Example: http://www.mywebsite.com', an 'Industry Category' dropdown set to 'Select One', and a 'Reporting Time Zone' dropdown set to 'United States' with a secondary dropdown set to '(GMT-08:00) Pacific Time'. At the bottom is the 'Data Sharing Settings' section with a note about data security and a link to the privacy policy.

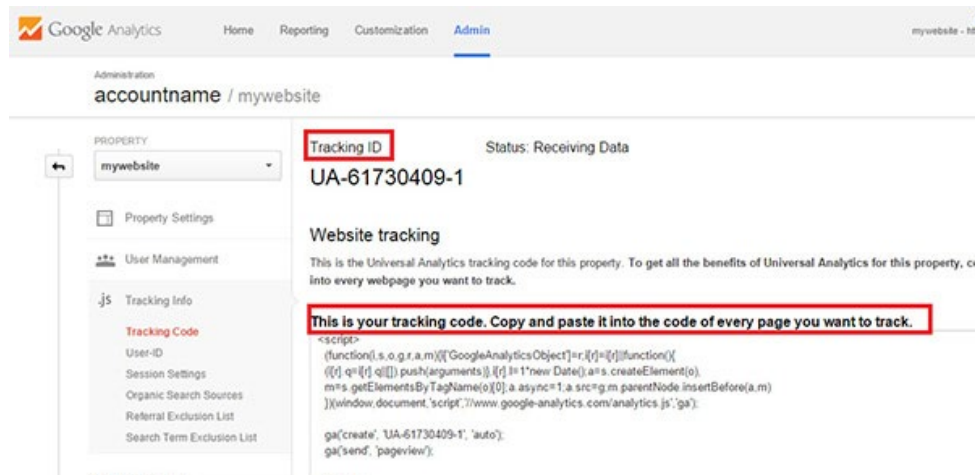
There are few fields you need to fill in order to create a new account. You can specify if you are tracking a website or a mobile application. You need to set your account and website name. When typing in your website URL, be careful to point the correct URL where your OpenCart store is installed. Optionally, you can enter the type of industry which represents your business and the time zone from where you are reporting. Furthermore, you can control your data sharing with other Google products and services by checking or not the four checkboxes: **Google products & services**, **Benchmarking**, **Technical support** and **Account specialist**.

## 2. Get Tracking Code

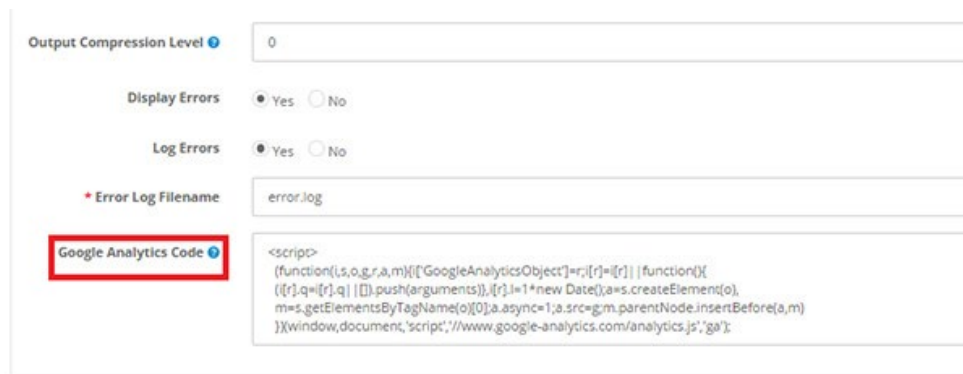
After completing the form and pressing Get Tracking ID button, yo-



u are redirected to admin page where the Tracking Code is located.



You can now copy the given code and paste it in your OpenCart admin panel in **Admin Panel => System => Settings => YourStore=> Server => Google Analytics Code**. Save changes and proceed to your Google Analytics account to see you website analytics.



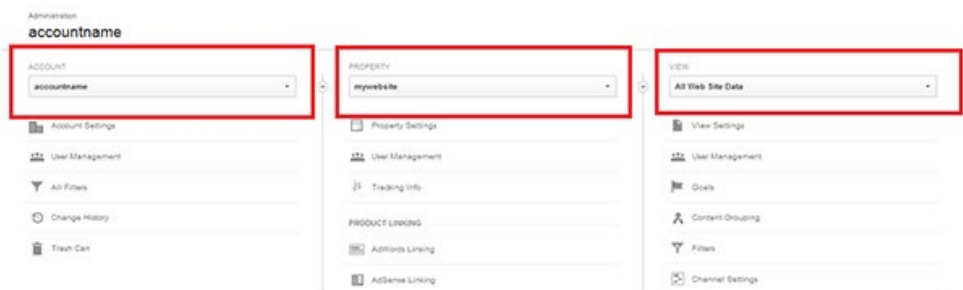
### 3. Make sure you are collecting data

Once you put the Tracking code into your web-store, you have to make sure it is working properly. Here we will show two simple ways to make sure your tracking is enabled. One possibility is to

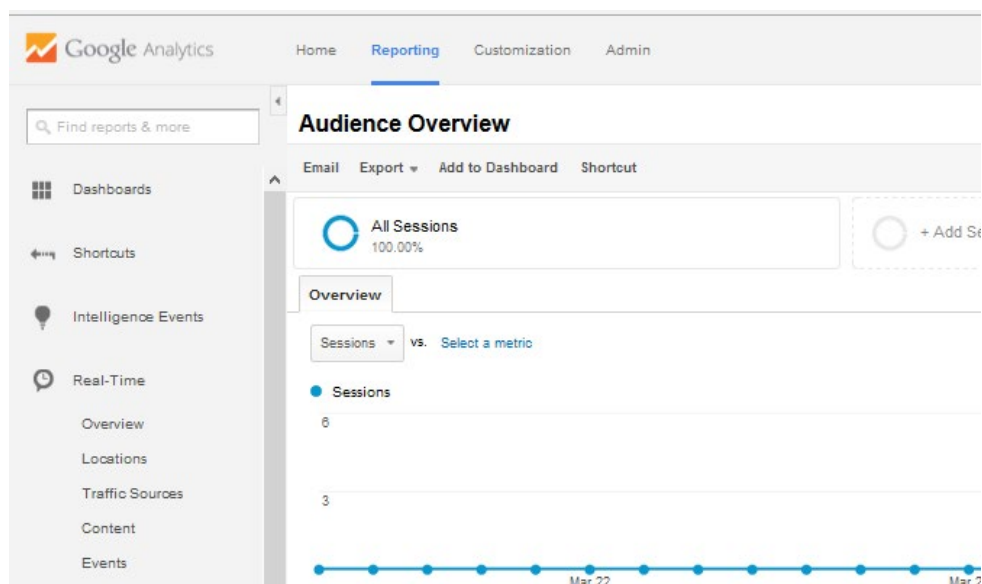
check the Real-Time reports which Google Analytics provides. You can do that by following these steps:

- 1. Sign in with your Google Analytics account**
- 2. Select the view of your property where you have added the tracking code.**

If you have recently added the tracking code to this property (website), it is likely that there will only be one view. In Google Analytics you are allowed to have up to 100 accounts with different properties and each property with different views.

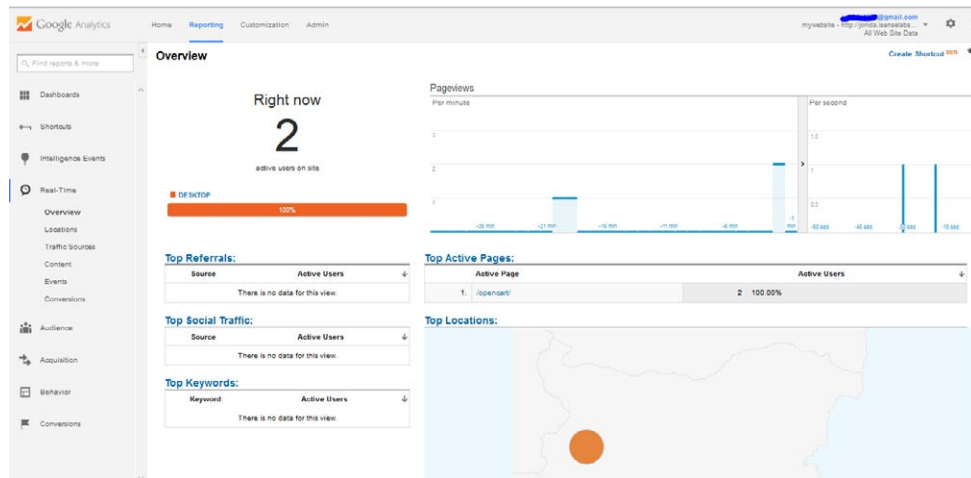


### 3. Select Reporting tab



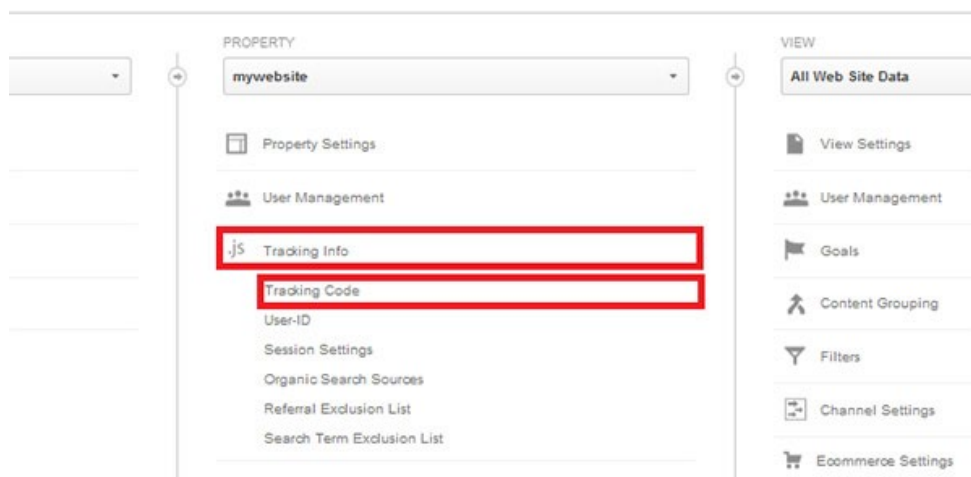
Go to **Real-Time -> Overview**. If there is data displayed in these re

ports it means that your tracking code is currently collecting data.



Another way to see if tracking code is working is by checking the Tracking Status in your property. These steps need to be followed in order to achieve it.

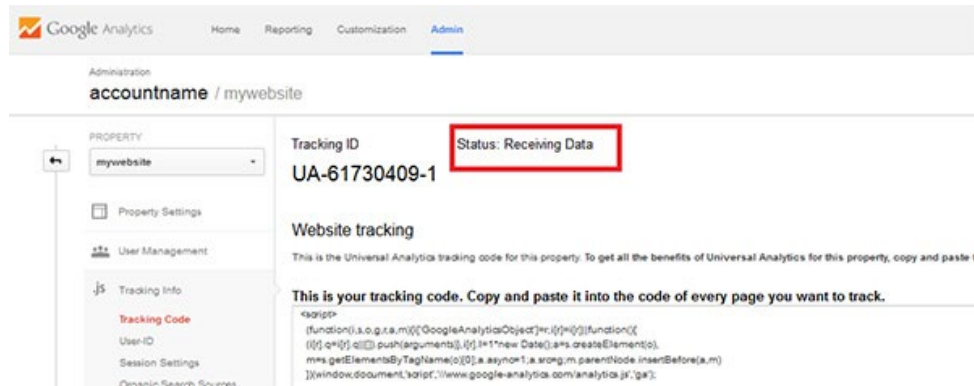
1. Sign in with your Google Analytics account
2. Navigate to the property by going into Admin tab.
3. Click **Tracking Info** -> **Tracking Code**.



4. Receive the tracking code status message.

There are four possible statuses of the tracking: **Waiting for Data**,

## Tracking Not installed or Not Verified ,Receiving Data,Unknown.



If your status is in 'Waiting for Data' or 'Receiving Data' that means that the configuration of the tracking code has been successful. Now you can explore all the analytics features that Google Analytics offers.

## In conclusion

By performing simple steps, you can easily add Google Analytics to your OpenCart store. In this way you can benefit from tracking its eCommerce features. You can track eCommerce conversions for each product, understand why engaged visitors are in your site and why some users bounce your site. Furthermore, you can see the success of email marketing campaign, track social engagement of users in your site and use event tracking to calculate the number of calls to action completed. Now it is all up to you to track the success of your store.

# How to set up multi-store in OpenCart 2.0.x

OpenCart is a powerful ecommerce platform and as such it comes with a built-in multi-store functionality. This means that you can set-up and manage as many stores as you like in just a single OpenCart installation.

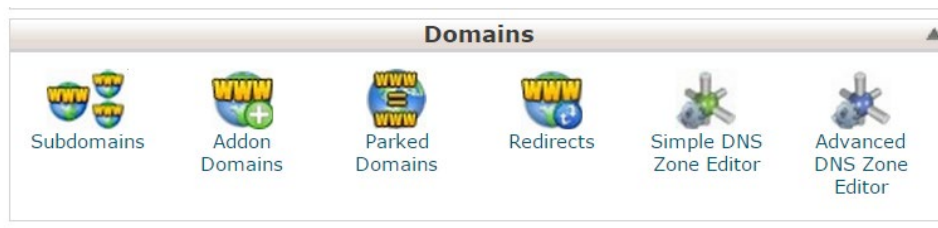
In this blog post we will guide you through the steps of setting up multi-store in OpenCart 2.0.x. In order to ease things up the tutorial will be split into two parts: part one - creating a domain or sub-domain for your new store and part two - creating the actual store in the OpenCart admin panel.



## Creating a new domain or subdomain

Login to your web hosting account control panel. In this tutorial

we will be using cPanel. Once you are logged in, navigate to the Domains section.



*Note: Registering a new domain requires purchasing the new domain name from your hosting provider.*

## 1. Create a new domain

Click on the *Addon Domains* icon in the *Domains* section. The following screen should appear:

Create an Addon Domain

New Domain Name:	<input type="text" value="mynewstore.com"/>	✓
Subdomain or FTP Username:	<input type="text" value="mynewstore"/>	✓
Document Root:	<input type="text" value="public_html/"/>	✓
Password:	<input type="password" value="....."/>	✓
Password (Again):	<input type="password" value="....."/>	✓
Strength (Why?):	<span style="background-color: #90EE90;">Very Strong (100/100)</span>	<input type="button" value="Password Generator"/>

The document root directory should be the same as your existing OpenCart installation

←

**IMPORTANT:** Your web host must enable this feature for your account before you can use it. Addon domains will not function unless you register your domain and configure it to point to the correct DNS servers.

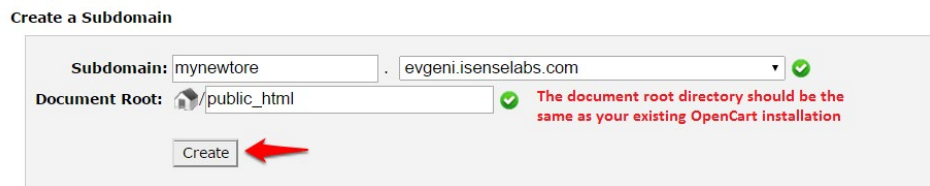
- **New Domain Name** - here you have to enter the website address of your new store.
- **Subdomain or FTP Username** - this should be filled automatically.

- **Document Root** - this is the key part of this step. The Document Root directory of your new domain should be exactly the same as the one of your existing OpenCart installation.

After clicking the Add domain button your new website address will be created.

## 2. Create a subdomain

If you choose to have your other store on a subdomain like my-newstore.mystore.com you just have to click on the Subdomain icon in the Domains section and the following screen will load up:



Create a Subdomain

Subdomain: mynewstore . evgeni.isenselabs.com ✓

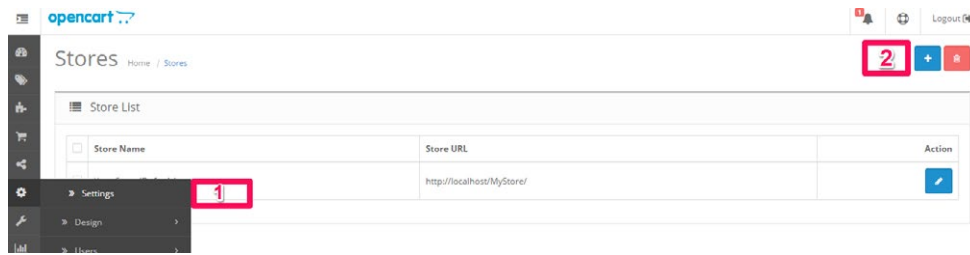
Document Root: /public\_html ✓ The document root directory should be the same as your existing OpenCart installation

Create

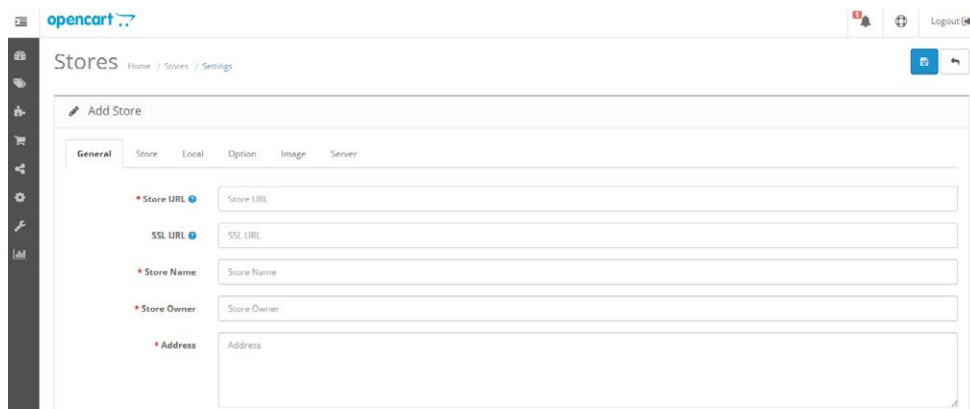
- **Subdomain field** - this is the place to fill in your subdomain name.
- **Document root** - the procedure here is the same as in the previous step. In order to make multi-store work your new subdomain should point to the same Document root directory as your existing OpenCart installation.

## Set up multi-store in OpenCart 2.0.x

Login to your admin panel and navigate to **System -> Settings** and click on the *Add new* button at the upper right corner of the page.



This will open a new store configuration window.



Here we can see several different tabs. Lets check each one of them in details.

- **General** - the most important setting in this tab is the Store URL. Here you have to specify your new store's url. In our example this is either the new domain name or subdomain we created in part one of this tutorial - mynewstore.com or mynewstore.your-store.com

The SSL URL setting is needed in order to provide secure check out in the front-end of your store. If you have a SSL certificate for this domain or subdomain you should enter your SSL URL here.

- **Store** - this is the place where you make the main configuration of your store . Here you can specify different theme and default layout.

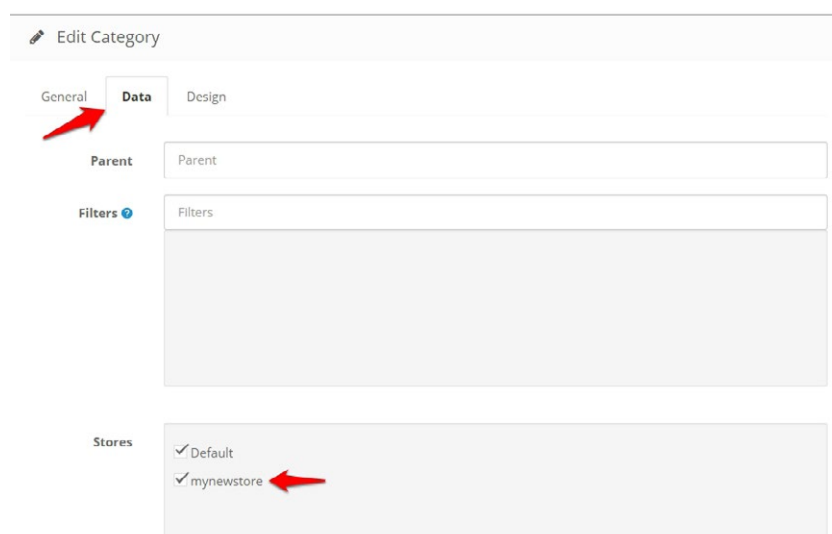


- **Local** - under this tab you can configure your store's localization settings. Most of the settings are self explanatory, but it is recommended to pay special attention to the Currency setting as it is used to set the specific currency your customers will be prompted with during their checkout.
- **Option** - here you can tweak different settings related to Items, Taxes, Account , Checkout and Stock.
- **Image** - in this section you can change your store logo, favicon and configure image dimensions.
- **Server** - the only option here is to enable or disable the SSL setting.

After configuring all the settings according to your preferences just click the Save button at the upper right corner of the page.

## Assigning specific categories to your new store

Whether you want to create a new category or add an existing one to your new store all you have to do is navigate to the Data tab of the selected category and at the Store section select your newly created store.



The screenshot shows the 'Edit Category' interface. At the top, there are three tabs: 'General', 'Data', and 'Design'. The 'Data' tab is selected, indicated by a red arrow. Below the tabs, there are three main sections: 'Parent', 'Filters', and 'Stores'. The 'Parent' section has a text input field with 'Parent' entered. The 'Filters' section has a text input field with 'Filters' entered. The 'Stores' section contains a list of stores with checkboxes. The 'Default' store is checked, and the 'mynewstore' store is also checked, with a red arrow pointing to the 'mynewstore' checkbox.

Lets see the equivalent settings regarding the products' management.

Go to **Catalog -> Products**. Select an existing product or create a new one. Navigate to the Links tab. Here you will find the same Store section where all the available stores are listed. Just select the one which you want the product to appear on.

The same procedure of configuration applies to the other entities as *Manufacturers, Information, Layouts etc.*

## Configuring your store front

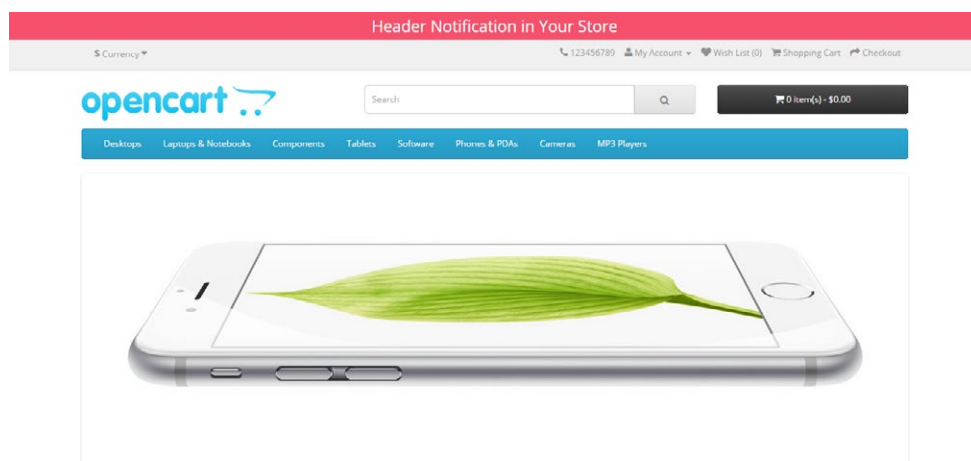
OpenCart will duplicate the default layout configuration of your main store to each of the newly created stores. In order to change that and make the stores look different you can go to **System -> Design -> Layouts** and configure them according to your preferences.

## Conclusion

OpenCart multi-store functionality is really easy to use and powerful. It gives you all the necessary settings to configure each one of the stores in the multi-store set up. You can configure different currencies, taxes and have different user groups. This is very useful when using the multi-store functionality to run different stores in different countries. Adding to these features the easy-to-use administration makes OpenCart a really good choice of an e-commerce platform for your next project.

# How to add a Notification bar in your store and admin panel in OpenCart 2.x

In this blog post we will demonstrate you how you can simply and quickly add a header notification to both your store-front and your administration panel in OpenCart 2.x.



In our previous tutorials we have suggested that modifying the core files is not advisable, which is why here we are going to concentrate only on the creation of an OCMOD file, which can be enabled and disabled at any time you want. Another cool thing about this simple modification is that you will be able to customize the notification depending on your own desire, need and imagination.

## Create an OCMOD File

So let's start with the creation of an empty text file using any text editor (the suggested options would be Notepad or Dreamweaver) and copying and pasting the following code:

```
<modification>

<name>Add Header Notification in the catalog and admin pages</name>

<version>1.0</version>

    <link>http://isenselabs.com</link>

    <author>iSenseLabs</author>

<code>isenselabs_header_notification</code>

<file path="admin/view/template/common/header.tpl">

    <operation>

        <search>

            <![CDATA[<header id="header" class="navbar navbar-static-top">]]>

        </search>

        <add position="replace">

            <![CDATA[

                <div id="HeaderNotification" style=" background-color: #F54661; z-index-
:99999;font-size:22px; text-align:center; color:#fff; position:fixed;width:100%;height: 40px-
;line-height:40px;top:0px;">Header Notification in Your Admin Panel</div><header id="header"
class="navbar navbar-static-top" style="margin-top:40px;">

                ]]>

            </add>

        </operation>

    </file>

<file path="catalog/view/theme/*/template/common/header.tpl">

    <operation>

        <search>

            <![CDATA[</head>]]>

        </search>

        <add position="after">

            <![CDATA[

                <div id="HeaderNotification" style=" background-color: #F54661; z-index
```

```
:99999;font-size:22px; text-align:center; color:#fff; position:fixed;width:100%;height:
40px;line-height:40px;top:0px;">Header Notification in Your Store</div><header id="header"
class="navbar navbar-static-top" style="margin-top:40px;">
```

After you have created the file, you will have to save it with the following name: *namebyyourchoice.ocmod.xml*. Please, keep in mind that the **.ocmod.xml** extension is compulsory if you want your file to be in the correct recognizable format.

Then, go to your OpenCart Store **Administration Page** => **Extensions** => **Extension Installer** and upload the file. After you get the Success Message go to **Extensions** => **Modifications** and click on the Refresh button in order to apply the new changes.

## Customize your Notification

Now we are going to show you how you can customize the layout and the text in the notification. The code in the first snippet between the `<file>``</file>` tags applies the notification in the admin panel and the second one in the store-front. If you would like to put a notification bar only in either of the admin panel or the store-front, you can just delete the snippet than you do not need starting from the opening `<file>` tag and ending with the closing `</file>` tag. If we suppose that you would like to display a notification only in your store-front, your code should look like that:

```
<modification>
<name>Add Header Notification in the catalog and admin pages</name>
<version>1.0</version>
    <link>http://isenselabs.com</link>
    <author>iSenseLabs</author>
<code>isenselabs_header_notification</code>
```

```
<file path="catalog/view/theme/*/template/common/header.tpl">

<operation>

  <search>

    <![CDATA[</head>]]>

  </search>

  <add position="after">

    <![CDATA[

      <div id="HeaderNotification" style=" background-color: #F54661; z-index:99999;-
font-size:22px; text-align:center; color:#fff; position:fixed;width:100%;height: 40px-
;line-height:40px;top:0px;">Header Notification in Your Store</div><header id="header"
class="navbar navbar-static-top" style="margin-top:40px;">
```

Now we are going to concentrate on the code itself and explain each statement in the following snippet:

```
<div id="HeaderNotification" style=" background-color: #F54661; z-index:99999;font-size:22px;
text-align:center; color:#fff; position:fixed;width:100%;height:40px;line-height:40px-
;top:0px;">Header Notification in Your Store</div>
```

**1. <div></div>:** The <div> tags are the ones that are creating the container of the actual notification bar.

**2. style=""** component applies the styles for this container:

- **background-color :** the background-color property sets the color of the notification bar, which is usually selected by either its hex color code (which is the case in our example #F54661) or its RGBA equivalent (which for our case would be *rgba(245,70,97)* ). There are plenty of websites with color codes in all formats which you can easily find in Google and use for reference, but I would suggest <http://www.color-hex.com> .For more information on background-color you can visit: [http://www.w3schools.com/cssref/pr\\_background-color.asp](http://www.w3schools.com/cssref/pr_background-color.asp)
- **z-index:** specifies the stack order of the element. Elements with a greater z-index show on the top of elements with a smaller z-index. For more information check out

[http://www.w3schools.com/cssref/pr\\_pos\\_z-index.asp](http://www.w3schools.com/cssref/pr_pos_z-index.asp)

- **font-size:** specifies the size of the letters that you are using in your notification. For further details you can check [http://www.w3schools.com/cssref/pr\\_font\\_font-size.asp](http://www.w3schools.com/cssref/pr_font_font-size.asp)
- **text-align:** sets the alignment of the text within your container. The text can be either centered, justified, aligned to the left or to the right. For examples please refer to: [http://www.w3schools.com/cssref/pr\\_text\\_text-align.asp](http://www.w3schools.com/cssref/pr_text_text-align.asp)
- **color:** the color property sets the color of the text in your notification bar. For more details you can refer to the background-color information.
- **position:** specifies the positioning of an element. In our case the 'fixed' is what keeps the notification bar always at the top of the window even when we scroll down. Further explanation for all of the positions you can find here: [http://www.w3schools.com/cssref/pr\\_class\\_position.asp](http://www.w3schools.com/cssref/pr_class_position.asp)
- **width & height:** specify the width and the height of the notification bar. In our case the width is set to 100%, which allows the bar to stretch and be as wide as the screen. The height is set in pixels, which means that it will be the same regardless of the screen size. For more information check here: [http://www.w3schools.com/css/css\\_dimension.asp](http://www.w3schools.com/css/css_dimension.asp)
- **line-height:** specifies the line-height of the font. More information here: [http://www.w3schools.com/cssref/pr\\_dim\\_line-height.asp](http://www.w3schools.com/cssref/pr_dim_line-height.asp)
- **top:** the top property sets the top edge of an element. In our case it is set to 0 indicating that the notification bar will begin from the very top of the body. Here, you have to keep in mind that the position element is important with regard to how exactly

the top property will be applied. For more information refer to:

[http://www.w3schools.com/cssref/pr\\_pos\\_top.asp](http://www.w3schools.com/cssref/pr_pos_top.asp)

Some other properties that are not used in our example but can be implemented include **font-family**, **font-weight**, **border**, etc.

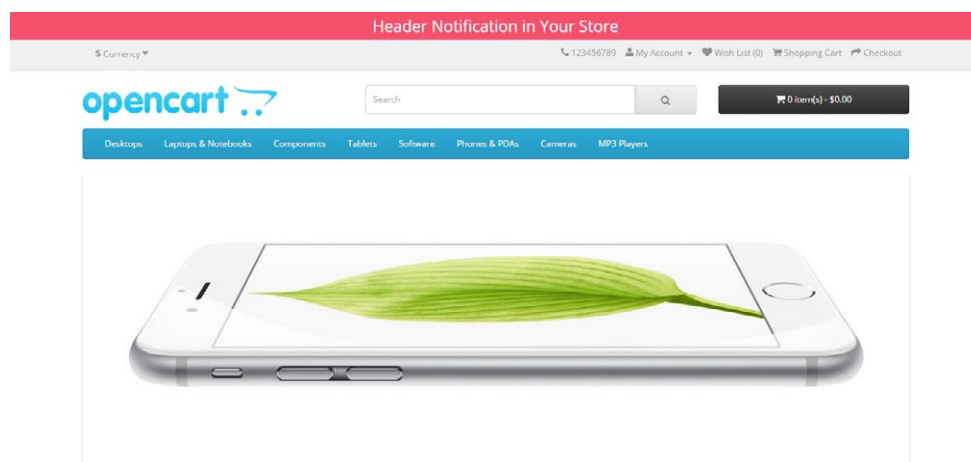
For more details, examples and ideas, check out the following link:

<http://www.w3schools.com/cssref/>

3. The last component in the div container is the **text** itself, in our case: *Header Notification in Your Store*, which you can easily replace with one by your choice.

**Important:** *Note that in the admin panel, you have the following code: `<header id="header" class="navbar navbar-static-top" style="margin-top:40px;">` . Here you have to set the **margin-top equal to the height** of the notification bar, because otherwise the notification bar might hide part of header in the admin panel screen.*

*The end result should be something similar to this one:*



Please, keep in mind that most of the CSS properties are related to each other and some might affect others (as the previously mentioned case with position and top properties). *If you happen to en-*









*counter anything like that or some unexpected behavior, please feel free to comment under the blog post.*

We hope that you will find this tutorial helpful and that it will give you some insights and ideas on how create and customize your own notification bars.

# How to display price of an item as FREE instead of 0.0

In this article we will show you a very easy way to change the 0.00 price for your products to a text string. In this case we will use the word “Free”, but you can change it to whatever you like. We will show two ways of implementing this - by changing the core files or by using the new OCMOD functionality in OpenCart 2.0.

## Shopping Cart (10.00kg)

Image	Product Name	Model	Quantity	Unit Price	Total
	Canon EOS 5D Select: Red Reward Points: 200	Product 3	1  	Free	Free
	iPhone	product 11	1  	\$123.20	\$123.20

## Edit the core files (Faster but not recommended)

The modification that we have to do is quite simple. What you have to do is open `system/library/currency.php` and search for this line:

```
$string = '';
```

Add this code before or after the searched line:

```
if ((float)$value == 0 && ($format)) {
    return 'Free';
}
```

We check if the price is 0 (zero) and if so, instead of placing a zeros for it, we change the string to **'Free'**. You can change the word to whatever you like.

Even though it looks not very challenging to implement, I suggest you to use the OpenCart 2.0 OCMOD system to apply this change.

That is because it is never good to directly change the core files of any system.

## Use OCMOD

Using OCMOD will not only keep your core files intact, but will give you the ability to easily enable or disable the modification whenever you like. You will also have no problems upgrading your OpenCart to a newer version using this kind of modification.

What you have to do is copy the code below and save it in a file called **new\_name.ocmod.xml**

```
<modification>
    <name>Show price FREE for zero amount</name>
    <version>1.0 (Initial)</version>
    <link>http://www.isenselabs.com</link>
    <author>iSenseLabs</author>
    <code>isenselabs_price_free</code>

    <file path="system/library/currency.php">
        <operation>
            <search><![CDATA[$string = ''];></search>
```

```
<add position="after"><![CDATA[
    if ((float)$value == 0 && ($format)) {
        return 'Free';
    }
}]></add>
</operation>
</file>
</modification>
```

*Note: Keep in mind that the file should end with **.ocmod.xml** otherwise the system will not implement it.*

When you are ready with the file, open your store administration and navigate to **Extensions -> Extension Installer**. After that, simply click upload the file using the upload button and wait for the success message. You are almost ready.

The file is uploaded, but the changes are not implemented. You should navigate to **Extensions -> Modifications** and click on the **Refresh** button. The page will refresh and you should receive another success message. The modification is now implemented and ready to use.

This is the end result:



### Apple Cinema 30"

The 30-inch Apple Cinema HD Display delivers an amazing 2560 x 1600 pixel resolution. Designed sp..

**\$110.00** ~~\$122.00~~

Ex Tax: \$90.00

 ADD TO CART



### Canon EOS 5D

Canon's press material for the EOS 5D states that it 'defines (a) new D-SLR category', while we'r..

Free

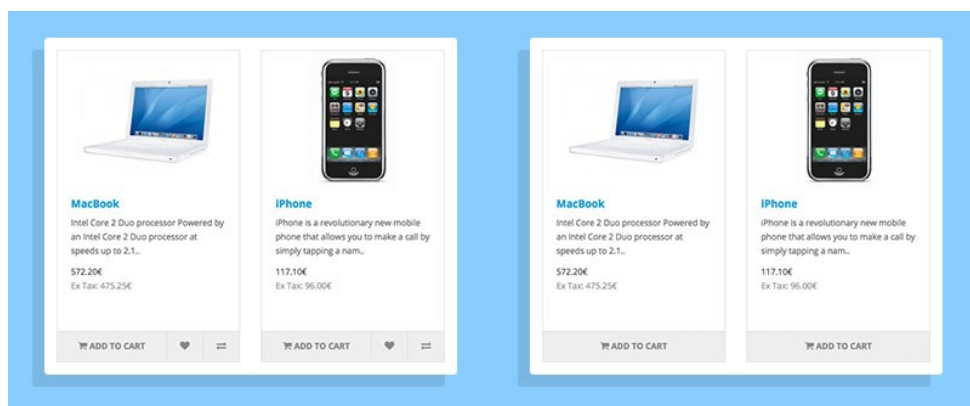
Ex Tax: Free

 ADD TO CART



# How to remove ‘Compare this Product’ and ‘Add to Wish List’ in OpenCart 2.0

‘Compare this Product’ and ‘Add to Wish List’ are great features in OpenCart however for some stores they are just not needed. Unfortunately there is no functionality or setting where you can disable these two features. In this tutorial I will show you how to remove the ‘Compare this Product’ and ‘Add to Wish List’ buttons from all places in your OpenCart store. For this tutorial I will use a clean installation of OpenCart 2.0.1.1 with the Default theme enabled.



In order to remove the ‘Compare this Product’ and ‘Add to Wish List’ buttons we have to modify several OpenCart templates. We can make the modifications directly in the template files, which I do not recommend, or we can make them as a new OCmod modification. Creating modifications using the OCmod modification sys-

em is much safer, because no core files are being directly modified plus we can revert back the default functionality at any time.

## Creating a new OCmod file

We can use almost any code or text editor and create a new blank XML file. Once we create the file we should create the skeleton of our OCmod file. You can copy/paste the code below, and change the name, version, link, author and code with your personal information.

```
<modification>

    <name>Type a name of this modification</name>

    <version>The version of the modification in numbers (ex. 1.0)</version>

    <link>http://yourwebsite.com</link>

    <author>Your Name</author>

    <code>unique_identifier_for_the_modification</code>

</modification>
```

Once we save the file we should add the '.ocmod' extension after the file name - ex. modification.ocmod.xml.

## Applying the changes

Now that the OCmod file is ready we start adding the modifications, one by one. The modifications are added before the closing </modification> tag.

### **1. Remove 'Compare this Product' and 'Add to Wish List' from Bestsellers, Featured, Latest, Specials.**

Let's start with removing the 'Compare this Product' and 'Add to

## Wish List' buttons from the default OpenCart modules - Bestsellers, Featured, Latest and Specials.

```
<file path="catalog/view/theme/default/template/module/{bestseller, featured, latest, special}*.
tpl">

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" title="<?php echo
$button_wishlist; ?>" onclick="wishlist.add('<?php echo $product['product_id']; ?>');"><i
class="fa fa-heart"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>

    </operation>

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" title="<?php echo $but-
ton_compare; ?>" onclick="compare.add('<?php echo $product['product_id']; ?>');"><i class="fa
fa-exchange"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>

    </operation>

</file>
```

## 2. Remove 'Compare this Product' and 'Add to Wish List' from Category Page.

There are 'Compare this Product' and 'Add to Wish List' buttons for each product in the Category Page, both for the Grid and List views. Let's remove them.

```
<file path="catalog/view/theme/default/template/product/category.tpl">

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" title="<?php echo
$button_wishlist; ?>" onclick="wishlist.add('<?php echo $product['product_id']; ?>');"><i
class="fa fa-heart"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>

    </operation>

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" title="<?php echo $but-
ton_compare; ?>" onclick="compare.add('<?php echo $product['product_id']; ?>');"><i class="fa
fa-exchange"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>
```



```

    </operation>
</file>

```

### 3. Remove 'Compare this Product' and 'Add to Wish List' from Product Page.

```

<file path="catalog/view/theme/default/template/product/product.tpl">

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" class="btn btn-de-
fault" title="<?php echo $button_wishlist; ?>" onclick="wishlist.add('<?php echo $product_id;
?>');"><i class="fa fa-heart"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>

    </operation>

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" class="btn btn-de-
fault" title="<?php echo $button_compare; ?>" onclick="compare.add('<?php echo $product_id;
?>');"><i class="fa fa-exchange"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>

    </operation>
</file>

```

### 4. Remove 'Compare this Product' and 'Add to Wish List' from Search Page.

```

<file path="catalog/view/theme/default/template/product/product.tpl">

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" class="btn btn-de-
fault" title="<?php echo $button_wishlist; ?>" onclick="wishlist.add('<?php echo $product_id;
?>');"><i class="fa fa-heart"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>

    </operation>

    <operation>

        <search><![CDATA[<button type="button" data-toggle="tooltip" class="btn btn-de-
fault" title="<?php echo $button_compare; ?>" onclick="compare.add('<?php echo $product_id;
?>');"><i class="fa fa-exchange"></i></button>]]></search>

        <add position="replace"><![CDATA[]]></add>

    </operation>
</file>

```

## 5. Remove 'Wish List' link from header.

After you have all the 'Add to Wish List' buttons removed from your store, there is no need of the 'Wish List' link in the header top bar. Let's remove it as well

```
<file path="catalog/view/theme/default/template/common/header.tpl">
    <operation>
        <search><![CDATA[<li><a href="<?php echo $wishlist; ?>" id="wishlist-total" ti-
title="<?php echo $text_wishlist; ?>"><i class="fa fa-heart"></i> <span class="hidden-xs hid-
den-sm hidden-md"><?php echo $text_wishlist; ?></span></a></li>]]></search>
        <add position="replace"><![CDATA[]]></add>
    </operation>
</file>
```

We have all the modifications added which will remove all the 'Compare this Product' and 'Add to Wish List' buttons. Now we have the save the file and install in on the store using the Extensions Installer.

However this modification will cause a small glitch for the places where we have the 'Add to cart' button grouped with the 'Compare this Product' and 'Add to Wish List' buttons. For example we have such button group in the Featured module and on the Category page. The glitch is caused by the fact that this button group has a certain width and the 'Add to cart' button width is set to 60%, which means it will not take the full width of this button group. In order to fix this glitch we should apply a small modification in the stylesheet.css file which is located in catalog/view/theme/default/stylesheet/stylesheet.css. We should search for the following CSS selector '.product-thumb .button-group button' and change its width from 60% to 100%.

```
.product-thumb .button-group button {  
    width: 100%;  
    ...  
}
```

## Conclusion

This tutorial is made especially for the Default OpenCart theme but I think the same result may be achieved for other custom themes after small changes in the modifications. However, 'Compare this Product' and 'Add to Wish List' are features which may have impact on your sales, so think twice before cutting them off.

You can find the OCmod modification file that I created for this tutorial [here](#).

# How to add store logo and product images to invoices in OpenCart 2.x

In this blog post we will show you how to add your store logo and product images to your invoices, changing the default OpenCart 2.x invoice template.

Unfortunately, OpenCart has not built-in editor for editing invoice template within the admin panel and we have to change it by modifying the code itself.

Note: This modification works only with OpenCart 2.x and we assume that you have already installed it.



## Invoice #1

Order Details	
<b>Your Store</b> Address 1  <b>Telephone:</b> 123456789 <b>E-Mail:</b> johndoe@testdomain.com <b>Web Site:</b> http://myteststore.com	<b>Date Added:</b> 10/02/2015 <b>Invoice No.:</b> INV-2013-001 <b>Order ID:</b> 1 <b>Payment Method:</b> Cash On Deliv <b>Shipping Method:</b> Flat Shipping f
<b>To</b> John Doe	<b>Ship To (if different address)</b> John Doe

## OCMod modification

Since modifying core files is not recommended, because of other

modules malfunctioning, we will use OCMOD modification to edit the template for the invoices.

This method does not modify the core functionality of OpenCart and this is why we are going to use this one.

First thing you should do is to create a new XML file and name it **yourcustommodificationname.ocmod.xml**. Please, make sure that your file has the extension .ocmod.xml, because otherwise it won't be recognized by the integrated Extension installer in OpenCart 2.x.

## Add logo in the header

If we want to include the logo in the header of the invoice, we need to add these lines in the file:

```
<modification>

  <name>Add images to invoice by iSenseLabs</name>

  <version>1.0 (Initial)</version>

  <link>http://isenselabs.com</link>

  <code>isense_invoice</code>

  <author>iSenseLabs</author>

  <file path="admin/controller/sale/order.php">

    <operation>

      <search><![CDATA[public function invoice() {}]]></search>

      <add position="after"><![CDATA[

          // ISENSELABS.COM CODE STARTS HERE

          $this->load->model('tool/image');

          // ISENSELABS.COM CODE ENDS HERE

        ]]></add>

    </operation>
```

## How to add store logo and product images to invoices in OpenCart 2.x

```
<operation>

    <search index="0"><![CDATA[$this->response->setOutput($this->load->view('sale/order_
invoice.tpl', $data));]]></search>

<add position="before"><![CDATA[

        // ISENSELABS.COM CODE STARTS HERE

        if (isset($this->request->server['HTTPS']) && (($this->request-
>server['HTTPS'] == 'on') || ($this->request->server['HTTPS'] == '1')) {

            $server = $this->config->get('config_ssl');

        } else {

            $server = $this->config->get('config_url');

        }

        if ($this->config->get('config_logo') && file_exists(DIR_IMAGE .
$this->config->get('config_logo'))) {

            $data['logo'] = $server . 'image/' . $this->con-
fig->get('config_logo');

        } else {

            $data['logo'] = '';

        }

        // ISENSELABS.COM CODE ENDS HERE

    ]]></add>

</operation>

</file>

<file path="admin/view/template/sale/order_invoice.tpl">

    <operation>

        <search><![CDATA[<div style="page-break-after: always;">]]></search>

        <add position="after"><![CDATA[

            <!-- ISENSELABS.COM CODE STARTS HERE -->

            <?php if(isset($logo) && !empty($logo)) { ?>

            <?php } ?>

            <!-- ISENSELABS.COM CODE ENDS HERE -->

        ]]></add>

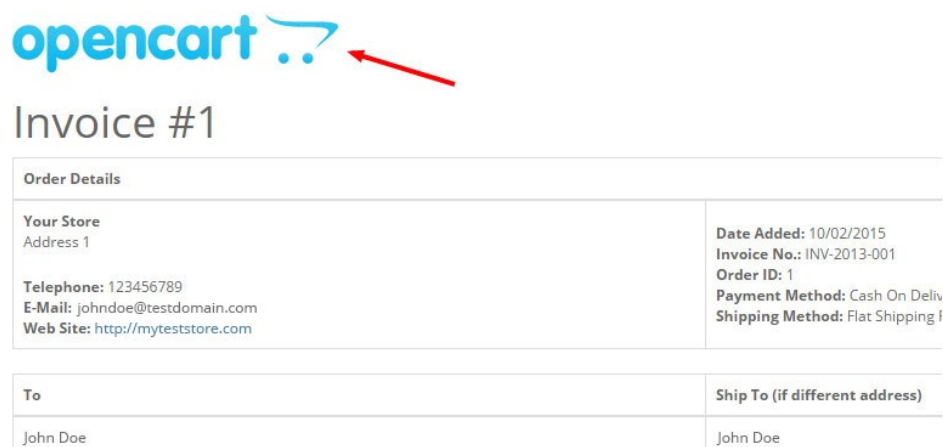
    </operation>

</file>

</modification>
```

Then, we should go to your **OpenCart Store Administration Page** > **Extensions** > **Extension Installer** and upload the file. After we get the message ‘*Success: You have installed your extension!*’, we need to go to **Extensions** > **Modifications** and click on the **Re-fresh** button to apply the changes.

The invoices should look like this:



## Add product images

If we want also to add product images to our invoices, we have to add this code in our current modification file or we can create a new one:

```
<file path="admin/controller/sale/order.php">

<operation>

<search><![CDATA[public function invoice() {}]]></search>

<add position="after"><![CDATA[

    // ISENSELABS.COM CODE STARTS HERE

    $this->load->model('catalog/product');

    // ISENSELABS.COM CODE ENDS HERE
```

```

]]></add>

</operation>

<operation>
    <search index="0"><![CDATA[$product_data[] = array()]]></search>

    <add position="before"><![CDATA[

        // ISENSELABS.COM CODE STARTS HERE

        $width = 100;

        $height = 100;

        $product_info = $this->model_catalog_product->getProduct($product['product_id']);

        if ($product_info['image']) {

            $image = $this->model_tool_image->resize($product_info['image'], $width, $height);

        } else {

            $image = $this->model_tool_image->resize('placeholder.png', $width, $height);

        }

        // ISENSELABS.COM CODE ENDS HERE

    ]]></add>

</operation>

<operation>

    <search index="0"><![CDATA[$product_data[] = array()]]></search>

    <add position="after"><![CDATA[

        'image'    => $image, // ISENSELABS.COM

    ]]></add>

</operation>

</file>

<file path="admin/view/template/sale/order_invoice.tpl">

    <operation>

        <search><![CDATA[<?php echo $product['name']; ?>]]></search>

        <add position="before"><![CDATA[

            <!-- ISENSELABS.COM CODE STARTS HERE -->

            <?php if(isset($product['image']) && !empty($product['image'])) { ?>

            <?php } ?>

        ]]></add>

    </operation>

</file>

```



```

        <!-- ISENSELABS.COM CODE ENDS HERE -->

    ]]></add>

</operation>

</file>

```

After that, we need to upload the modification file to our OpenCart store and the final result will be this:

Product	Model	Quantit
 iPhone	product 11	

## Change product image size

Here we will show you how to customize the code in the OCMod file. This modification is not big, but you can do some minor changes.

If you need smaller or bigger product images, you can change these two rows:

```

$width = 100;
$height = 100;

```

After you made the changes you need to save your .ocmod.xml file. Before uploading the modification, you need to delete the old one from Extensions > Modifications, check the modification 'Add images to invoice by iSenseLabs' and click on the red delete button. You need also to clear and refresh the modifications. After that you will be able to upload the new .ocmod.xml file, using the integrat

ed Extension Installer in OpenCart 2.x.

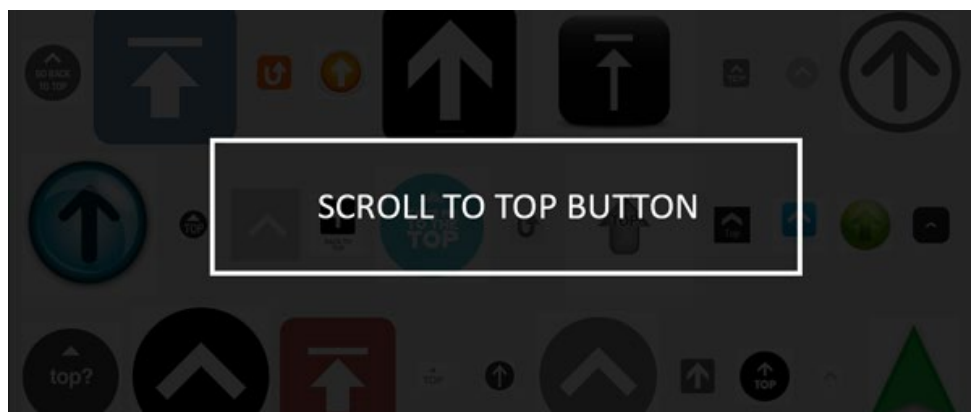
## Final words

We hope that you will find this tutorial helpful and that it will give you ideas how to freshen up your invoice. [Download Source](#).

# How to add a Scroll-to-top button in OpenCart 2.x

The scroll to top button improves your website navigation, brings a nice touch in the user interface and refines the overall design of your store. As you may have already realized, having a scroll to top button is one of these little things that your website doesn't necessarily need, but nonetheless, are enhancing the user experience and the slickness of your website.

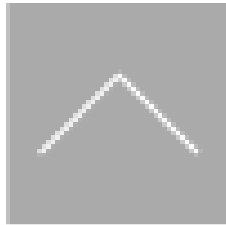
In this blog post we will show you how you can easily and quickly add a scroll button to your pages in your OpenCart 2.x web store.



## Choose Scroll-To-Top-Button Image

We shall start with choosing the image for the button. In fact, you

can use any image of your choice, but keep in mind that it should remind its purpose, otherwise the button will be quite useless. There are plenty of scroll-to-top and arrows icons in google.com which you can download. Alternatively, you can make one your own. After you have chosen your image, you will need to upload it in the following directory in your server: catalog/view/theme/your\_theme\_name/image/. The image chosen for this tutorial is named 'scroll.png' ([View Here](#)) and looks like this:



## Create scroll-to-top.js file

The next step after uploading the image, is to create a JavaScript file which will add the button to your pages and will 'tell' it what to do. For this purpose you will need to locate the following directory in your server: catalog/view/javascript and create a new file using a text editor by your choice. The file has to be named in the following way: nameofyourpreference.js. In our example, we will name the file scrolltotopdemo.js.

After creating the empty file you need to copy and paste the following code:

```

var scrolltotop={
    setting: {startline:200, scrollto: 0, scrollduration:1000, fadeduration:[800, 500]},
    controlHTML: '',
    controlattrs: {offsetx:20, offsety:75},
    anchorkeyword: '#top',
    //Enter href value of HTML anchors on the page that should also act as "Scroll Up"
links
    state: {isvisible:false, shouldvisible:false},
    scrollup:function(){
        if (!this.cssfixedsupport) //if control is positioned using JavaScript
            this.$control.css({opacity:0}) //hide control immediately after clicking it
        var dest=isNaN(this.setting.scrollto)? this.setting.scrollto : parseInt(this.setting.
scrollto)
        if (typeof dest=="string" && jQuery('#'+dest).length==1) //check element set by
string exists
            dest=jQuery('#'+dest).offset().top
        else
            dest=0
        this.$body.animate({scrollTop: dest}, this.setting.scrollduration);
    },
    keepfixed:function(){
        var $window=jQuery(window)
        var controlx=$window.scrollLeft() + $window.width() - this.$control.width() - this.
controlattrs.offsetx
        var controly=$window.scrollTop() + $window.height() - this.$control.height() - this.
controlattrs.offsety
        this.$control.css({left:controlx+'px', top:controly+'px'})
    },
    togglecontrol:function(){
        var scrolltop=jQuery(window).scrollTop()
        if (!this.cssfixedsupport)
            this.keepfixed()
        this.state.shouldvisible=(scrolltop>=this.setting.startline)? true : false
        if (this.state.shouldvisible && !this.state.isvisible){
            this.$control.stop().animate({opacity:0.5}, this.setting.fadeduration[0])
            this.state.isvisible=true
        } else if (this.state.shouldvisible==false && this.state.isvisible){

```

```

        this.$control.stop().animate({opacity:0}, this.setting.fadeduration[1])
this.state.isvisible=false
    }
},
init:function(){
    jQuery(document).ready(function($){
        var mainobj=scrolltotop
        var iebrws=document.all

        mainobj.cssfixedsupport=!iebrws || iebrws && document.compatMode=="CSS1Compat" &&
window.XMLHttpRequest

        mainobj.$body=(window.opera)? (document.compatMode=="CSS1Compat"? $('html') :
$('body')) : $('html,body')

        mainobj.$control=$('<div id="topcontrol">'+mainobj.controlHTML+'</div>')

        .css({position:mainobj.cssfixedsupport? 'fixed' : 'absolute', bottom:mainobj.
controlattrs.offsety, right:mainobj.controlattrs.offsetx, opacity:0, cursor:'pointer'})

        .attr({title:'Scroll to Top'})

        .mouseenter(function() { $(this).css("opacity", "1");})

        .mouseleave(function() { $(this).css("opacity", "0.5");})

        .click(function(){mainobj.scrollup(); return false})

        .appendTo('body')

        if (document.all && !window.XMLHttpRequest && mainobj.$control.text()!='')

            mainobj.$control.css({width:mainobj.$control.width()})

        mainobj.togglecontrol()

        $('a[href="'+ mainobj.anchorkeyword +'"]').click(function(){

            mainobj.scrollup()

            return false

        })

        $(window).bind('scroll resize', function(e){

            mainobj.togglecontrol()

        })

    })
}

scrolltotop.init()

```

\* Keep in mind that you should replace the **scroll.png** with the name of your image in the following code snippet:

```
scroll.
png</span>" style="width:50px; height:50px" />
```

## Add the file to your pages

In order to add the file to your pages, you need to make a references to it. To do that, you'll need to go to ***catalog/view/theme/your\_theme\_name/template/common/header.tpl*** and add the following code snippet **before** the closing **</head>** tag:

```
<script src="catalog/view/javascript/<span style="background-color:Yellow;">scrolltotopdemo.
js</span> " type="text/javascript"></script>
```

\* Keep in mind that you should add the name of the JavaScript file that you have just created instead of '***scrolltotopdemo.js***'. Make sure to SAVE and UPLOAD the file to your server.

## Modify the Code

In this section we will concentrate on the ways in which you can modify the JavaScript file and make the button look and behave in the way you desire.

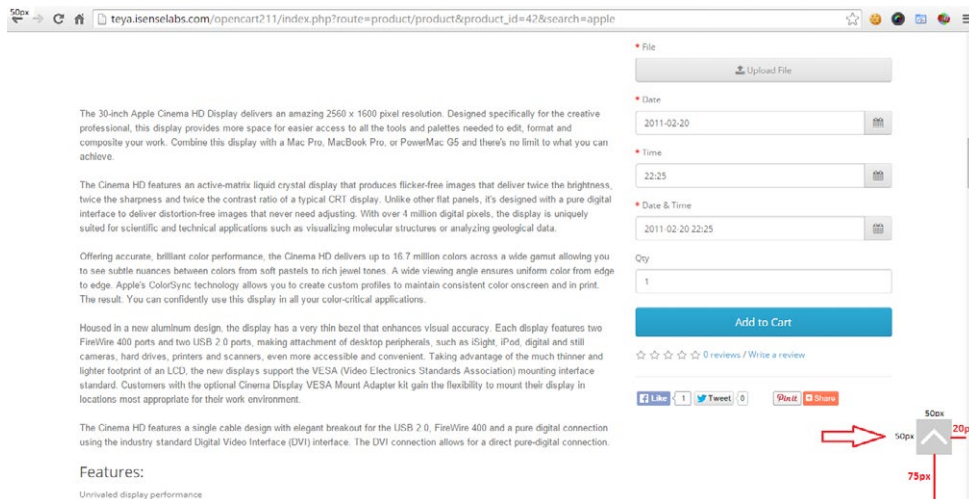
### Size

In order to change the size of your image you will need to replace the 50px with the desired image dimensions in the following code snippet:

```
controlHTML: `50px</span>; height:<span style="background-color:Yel-
low;">50px</span>" />
```

## Position

With the current set up, the button is with a fixed positioning and is displayed in the bottom left corner of your page.



If you want to change the positioning, you need to modify the numbers here:

```
controlattrs: {offsetx:20, offsety:75},
```

In our script, the *offsetx* is responsible for the distance between the button and the right border of your page and the *offsety* for the distance to the bottom of the page. If you want to stick the button to the very bottom right edge of the page, for example, you will need to set the *offsetx* and *offsety* to 0.

## Switch on/off

As you will notice, the button doesn't appear right when the page is loaded, but when you scroll down a bit. If you would like to increase or decrease the scrollable distance before the button ap



pears, you will need to increase or decrease, the number 200 set for startline in the following line:

```
setting: {startline:200, scrollto: 0, scrollduration:1000, fadeduration:[800, 500]},
```

## Scrolling

The same line contains the settings for the scrolling functionality.

- **Scrollto:0**

Sets the page to scroll back to the very top. If you replace the 0 with let's say 300, the page would not scroll to the top, but to 300px from the top.

- **Scrollduration:1000**

This sets the time for which the page will scroll to the top. The number 1000 stands for 1000 milliseconds or 1 second. If you increase the number, the scrolling will be slower, and vice versa.

- **Fadeduration:[800, 500]**

These numbers set how long the button fades in and out. The first number sets the time for fading in when the button is appearing and the second number, sets the fading out duration when the button is disappearing. As with the scrollduration setting, the numbers stand for time measured in milliseconds.

## Mouse over

Currently the button is fading in, but not to its full opacity, but to 0.5 as set in the following code:

```
if (this.state.shouldvisible && !this.state.isvisible){
    this.$control.stop().animate({opacity:0.5}, this.setting.fadeduration[0])
    this.state.isvisible=true }
```

This means that the button is half transparent. When you hover over it, however, it becomes fully solid and it creates an effect of darkening the color. When the mouse is removed from the button, it becomes half transparent again or it lightens. This is achieved with the following code:

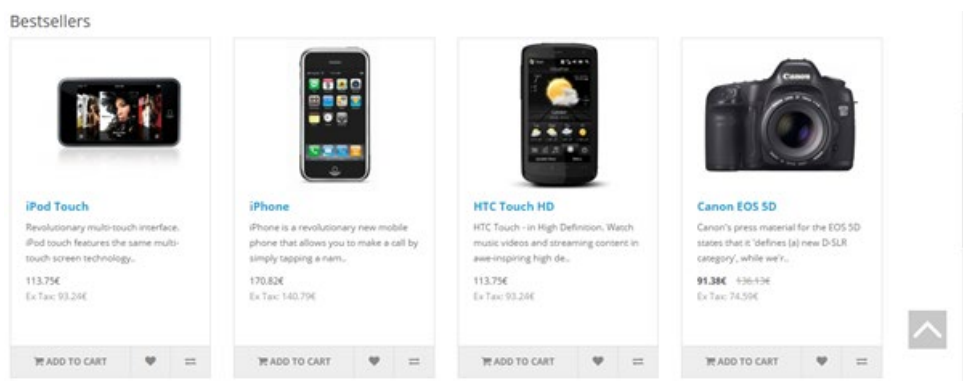
```
.mouseenter(function() { $(this).css("opacity", "1");})
.mouseleave(function() { $(this).css("opacity", "0.5");})
```

If you would like to change the transparency you can change the numbers *1* and *0.5*. Keep in mind that 1 stands for fully solid and 0 for fully transparent. Or in other words, the smaller the number, the more transparent the image is and vice versa. If you would like to remove the hover effect, you can go ahead and delete these two lines of code.

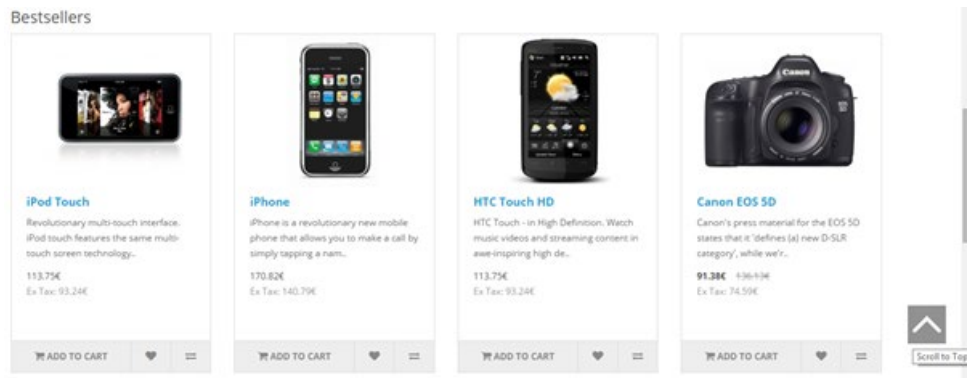
You will also notice that when the mouse is over the button, a text field appears: *"Scroll to Top"*. In order to change that, you will need to modify the text in the quotes in the following code snippet:

```
.attr({title:'Scroll to Top'})
```

## Default Display:



## Mouse Hover Effect:



# Restrict admin access to your IP addresses

*This article is written with OpenCart in mind. It is perfectly valid for other platforms as well, including Magento, WordPress, PrestaShop and Joomla.*

A quick and powerful security tip on your admin panel - secure it to be accessible to certain IP addresses only. This would ideally be you, and your other site administrators. In the next steps, I will explain how while keeping it simple.



## STEP 1 - Check your public IP address

First step is to find your public IP address you would want to later whitelist in your admin folder. You can do this [here](#), [here](#) or [here](#).

## STEP 2 - Put an .htaccess file

The objective here is to go to your admin folder on your server and to restrict the access to your admin panel to certain IP addresses only.

Connect to your server, where your site files are, using FTP client of your choice (like FileZilla). You will need the FTP credentials for your site in order to connect. If you are not sure how to find them, ask your hosting provider.

Once connected, navigate to your admin folder and create a new file there with name .htaccess. If such file exists already, simply open it for edit. Add the following code to this .htaccess file, by replacing 1.2.3.4 from the code below with your IP address (which you got from Step 1). You can allow multiple IP addresses by adding new *Allow from* lines with the other IPs.

```
Order deny,allow  
Deny from all  
Allow from 1.2.3.4
```

Then save, and upload the file.

## STEP 3 - Give it a try

Now, open your web browser and navigate to your admin panel. You should be seeing it the same way as before, without any issues. Now go back to Step 2 and change the allowed IP address to different than yours, and try again to reach the admin panel URL. Your access should be denied. You have tested it and it is working. Good job! Revert it back to your IP, and enjoy your new protection.

# Password protect the OpenCart admin

The following article will cover a simple security feature of the Apache server - password protected directories and how you can use them in your OpenCart setup.



This method is very useful, because it does not rely on any PHP verification, but it blocks all access to files and folders in the protected folder, until a user is logged in. To add one more layer of security to your OpenCart installation, we are going to protect the **admin** folder in OpenCart.

In order to protect a folder on your Apache server, you will need to have two files. The **.htaccess** file, which will prevent the user from accessing the folder, without logging in and the **.htpasswd** file, which holds the encrypted user passwords. The example we are

going to demonstrate is using the default MD5 algorithm encryption - an iterated (1,000 times) MD5 digest of various combinations of a random salt and the password. This is the default since Apache version 2.2.18.

## 1. Generate a password

We have created this little tool to help you generate your **.htaccess** and **.htpasswd** files easily, without having to write any commands in your terminal or command prompt. The tool is located here -

[https://isenselabs.com/external/htaccess\\_password/index.php](https://isenselabs.com/external/htaccess_password/index.php).

Fill in the **Username** and **Password** inputs, after that type in the **Server Path** to the folder. You will have to upload the generated **.htpasswd** file to that folder on your server. Example: **/home/my-website/public\_html/** or something similar to this.

Click **Generate** and your **.htaccess** and **.htpasswd** will be output in the content sections below the form. For your convenience, you can use the **Download Files** button, to download a zip containing the two files.

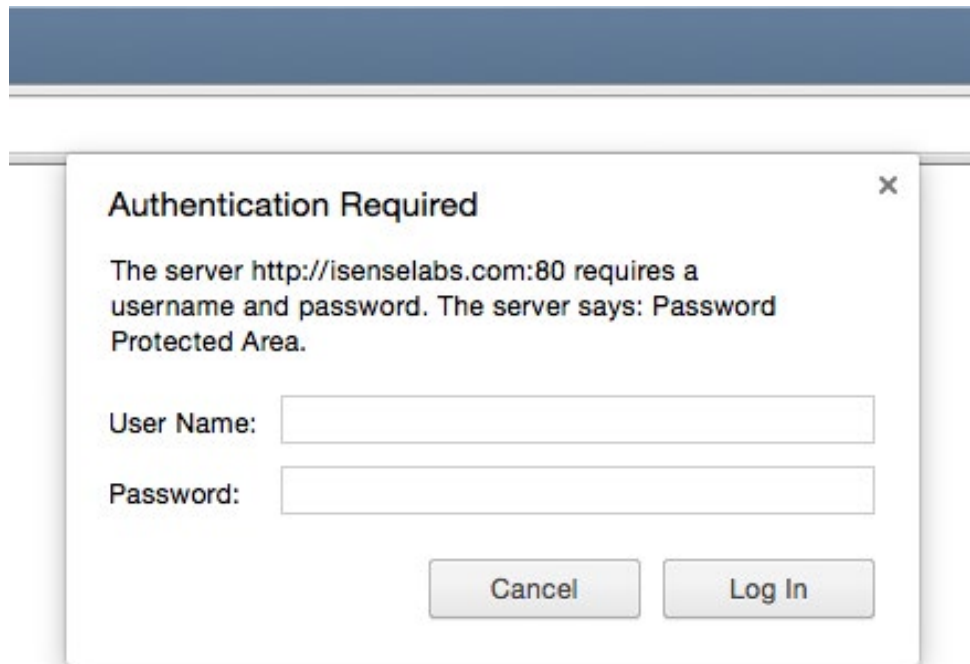
## 2. Upload the generated files

Now with the files generated the only thing left to do is upload them to your web server. In order to protect a certain directory upload the **.htaccess** file to that directory and after that upload the **.htpasswd** to the folder pointed in the **Server Path** field.

In our case we will have to upload the **.htaccess** file to the **/home/mywebsite/public\_html/admin/** directory to protect the OpenCart

admin section. Then we will upload the .htpasswd file to **/home/mywebsite/public\_html/** and we are done!

Now when a user tries to access a resource (file, image, document etc.) located in your **admin** directory they will see a screen similar to this one, where they have to login.



Thank you for reading!

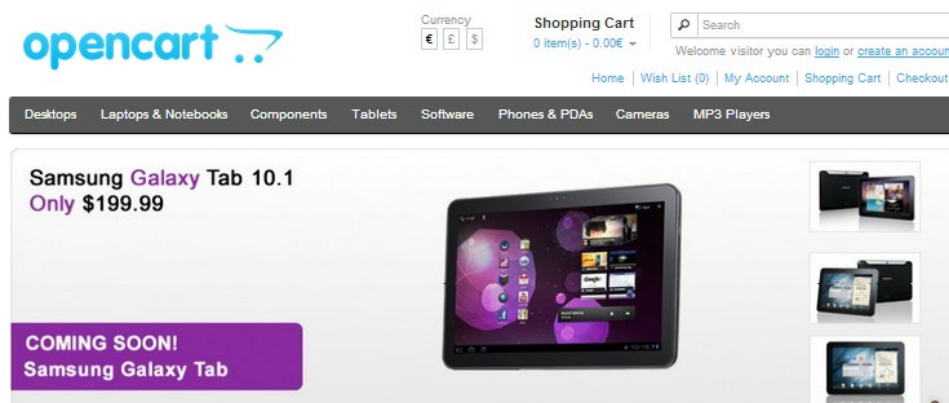
P.S. You can also rename the OpenCart default **admin** folder, read how here - <https://isenselabs.com/posts/tip-for-making-opencart-more-secure>.



# Tip for Making OpenCart More Secure

Changing your admin folder's name is not mandatory but a recommended step if you want to have better security for your online store. In this post I will show you how to do that as easy as possible in a few steps.

*This tutorial is written specifically for OpenCart 1.5.6.1, but the following steps can be applied to older versions of OpenCart as well.*



## 1. First things first

Rename the folder admin to the new name that you have chosen. In this case we will use **isenselabs**.

## 2. Open the file located in admin/config.php

It has to look like this:

```
<?php

// HTTP
define('HTTP_SERVER', 'http://example.com/opencart/admin/');
define('HTTP_CATALOG', 'http://example.com/opencart/');

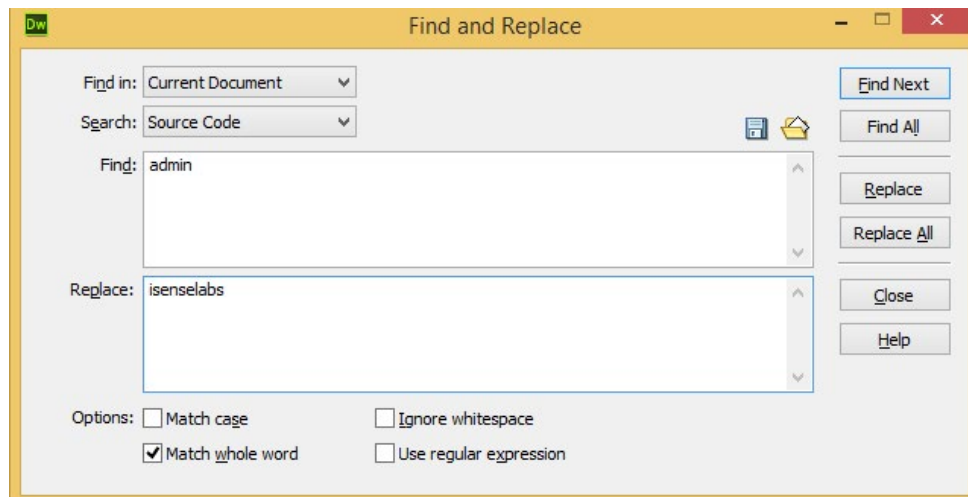
// HTTPS
define('HTTPS_SERVER', 'http://example.com/opencart/admin/');
define('HTTPS_CATALOG', 'http://example.com/opencart/');

// DIR
define('DIR_APPLICATION', '/home/user/public_html/opencart/admin/');
define('DIR_SYSTEM', '/home/user/public_html/opencart/system/');
define('DIR_DATABASE', '/home/user/public_html/opencart/system/database/');
define('DIR_LANGUAGE', '/home/user/public_html/opencart/admin/language/');
define('DIR_TEMPLATE', '/home/user/public_html/opencart/admin/view/template/');
define('DIR_CONFIG', '/home/user/public_html/opencart/system/config/');
define('DIR_IMAGE', '/home/user/public_html/opencart/image/');
define('DIR_CACHE', '/home/user/public_html/opencart/system/cache/');
define('DIR_DOWNLOAD', '/home/user/public_html/opencart/download/');
define('DIR_LOGS', '/home/user/public_html/opencart/system/logs/');
define('DIR_CATALOG', '/home/user/public_html/opencart/catalog/');

// DB
define('DB_DRIVER', 'mysql');
define('DB_HOSTNAME', 'localhost');
define('DB_USERNAME', 'user_1');
define('DB_PASSWORD', 'user_1');
define('DB_DATABASE', 'user_opencart');
define('DB_PREFIX', 'oc_');

?>
```

You have to change all the lines where the word **admin** is present. The easiest way to do this is by using the quick 'Find and Replace' tool which is available in almost every text editor (Notepad, Dreamweaver and along other editors).



Find all occurrences of the word **admin** and change it with the folder name that you have chosen. This is an example of how your **admin/config.php** should look like after you make the edits:

```
// HTTP
define('HTTP_SERVER', 'http://example.com/opencart/isenselabs/');
...

// HTTPS
define('HTTPS_SERVER', 'http://example.com/opencart/isenselabs/');
...

// DIR
define('DIR_APPLICATION', '/home/user/public_html/opencart/isenselabs/');
...

define('DIR_LANGUAGE', '/home/user/public_html/opencart/isenselabs/language/');
define('DIR_TEMPLATE', '/home/user/public_html/opencart/isenselabs/view/template/');
```

### 3. Fix your vQmod configuration and your vQmod modules.

If you are using vQmod, there are some things that you need to edit. First of all, you need to fix your vQmod configuration file. Open the **index.php** located in **vqmod/install/index.php** and change the following line:

```
$admin = 'admin';
```

To this:

```
$admin = 'isenselabs';
```

After that, you have to ensure that all of your modules are working correctly. As of vQmod 2.3.0 there is a file called **pathReplaces.php**. It is used to globally replace the admin folder name without having to modify the .xml files. You have to open the file and add the following line:

```
$replaces[] = array('~^admin\b~', 'isenselabs');
```

That's it!

Still, If you are using an older version of vQmod, you have to make the changes manually. Please read on, if you are running a version older than vQmod 2.3.0. Here is what you need to do:

Open all files in vqmod/xml and replace all occurrences of the string admin with the folder's name that you have chosen. For example, the line:

```
<file name="admin/view/template/sale/order_form.tpl">
```

Should be changed to:

```
<file name="isenselabs/view/template/sale/order_form.tpl">
```

## 4. Be careful with caching extensions.

It is necessary to clear the caches (browser cache, page cache, database cache and others) in order the changes to take effect.

If you are using caching extensions such as [NitroPack](#), you have to make sure that they don't cache your new admin folder. If you don't do that, you won't be able to change your web store settings, view orders and etc.

## 4.1 NitroPack

If you are using NitroPack, you will have to edit the following three lines:

Open the file `core.php` located in `system/nitro/core` and find this line of code:

```
$predefinedIgnoredUrls = array('/admin/', 'isearch');
```

Again, you have to replace `/admin/` with the new name:

```
$predefinedIgnoredUrls = array('/isenselabs/', 'isearch');
```

*The changes below should be made only if you are using OpenCart with vQmod < 2.3.0.*

Open the file `nitro.xml` located in `vqmod/xml/` and edit the following lines:

```
<file name="admin/controller/catalog/product.php"> (line 32)
```

```
<file name="admin/controller/catalog/product.php"> (line 47)
```

They have to look this:

```
<file name="isenselabs/view/template/common/header.tpl">
```

```
<file name="isenselabs/view/template/common/header.tpl">
```

If you are using another product for caching, you will have to make similar (depending on the product) changes.

## 5. That's all folks!

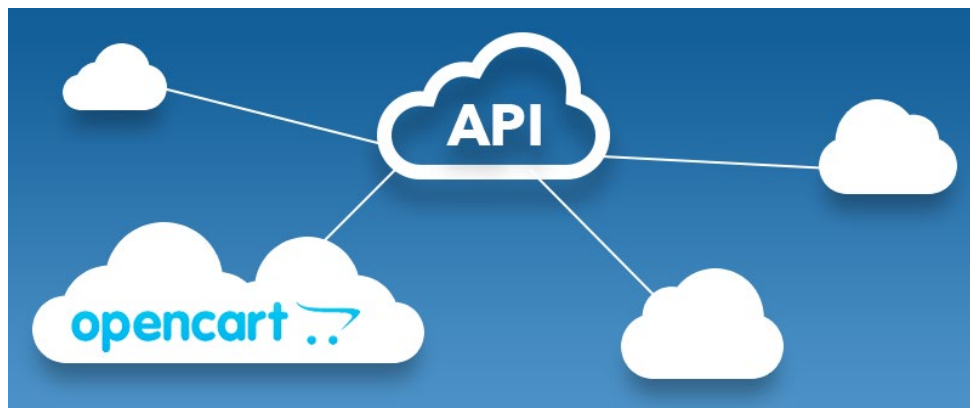
All you need to do now is to check if everything is working properly. If you don't see any errors or blank pages, you did a great job!

Enjoy your better secured OpenCart store!

# Using the new API methods of OpenCart 2.x

In OpenCart 2.x a new API system has been introduced. It should allow third party applications to talk to your OpenCart store for better integration capabilities. In this article, we will explain how the API works and how to use it. The API does not seem to be ready yet, but it is a good idea to get familiar with its current state and the way it works.

*This article is based on OpenCart 2.0.1.1*



## Introduction

In OpenCart 2.x a new API system has been introduced. It allows third party applications to talk to your OpenCart store for better

integration capabilities. In this article, we will explain how the API works and how to use it. A good progress has been made on the API and although it is by no means the final version, we believe it is a good idea to get familiar with its current state and the way it works.

## Configuration

In order to use any of the API methods, you will need to login first. Login credentials for the API users are managed from OpenCart's admin panel by going to **System > Users > API**. Creating a new pair of credentials is pretty straight-forward. After you have obtained your username and password, you are ready to start playing with the API.

## Logging in

All of the API controllers are found in the api directory so your base URL, should look similar to this: *yourdomain.com/index.php?route=api/controller/method*. Here *yourdomain.com* should be replaced with your actual domain name and *controller / method* with the needed controller and method from the API. Keep in mind that the *method* part is optional. If you want to call the default *index()* function of a controller, you can omit the *method* from your request URL.

In order to login you need to call the index function of the login controller, so the request URL will be: *yourdomain.com/index*.



*php?route=api/login*. The request type should be **POST** and the required **POST** data is “**username**” and “**password**”. Each API method returns JSON formatted data. In the case of the login method, it will either return a success message or an error message. So checking whether the returned object has a **success** or **error** member will tell you whether the login was successful or not. It will also return a **cookie**, which is the value of your session ID. You will most probably not need this value, since the cookie’s name is not returned. In order to maintain a session, you will need to extract the cookies from the raw HTTP response and send them with each consequent request.

## Adding a product to the cart

After a successful login, you can now interact with the API. One thing you can do is add a product to your cart using the *cart/add* method. The URL for this operation is: *yourdomain.com/index.php?route=api/cart/add*. In its most basic form, the method takes a **product\_id** and **quantity**. You can also feed an **option** array to add the desired product with specific options. If you want to add more than one product, you can pass the method a single **product** array where each element, should be an array with **product\_id**, **quantity**, and (not mandatory) **option** keys.

**Note:** The bolded words in this section are the names of the POST variables you need to supply.

## Explore the API

We cannot cover all of the available API methods here, but we encourage you to go inspect all the files in your *catalog/controller/api/directory*. This way you will have an idea of what you can do and what not with the API.

## What is missing

Ok, so I showed you how to add an item to the cart, but for that method I told you that you need to supply a **product\_id**. Now you may be wondering how to get a list of the available product IDs - well currently you cannot. This is not implemented yet, and we do not have a way to query the product database, or get a list of options for a particular product. This seems like a pretty solid feature, so we expect that it will be coming in one of the next versions of OpenCart.

## A helper library

We thought it would be nice if there was a library to help us work with the API without thinking of cookies, GET / POST parameters and their names - so we created one. You can get a copy from <https://github.com/iSenseLabs/OpenCartAPI>. The main class is called `OpenCart` and is in the namespace `OpenCart`. We tried to keep it consistent with the API URLs, so the *cart/add* method became the *add()* method of the *cart* object of an *OpenCart* object. Let me show you an example of logging in and adding a product to the

**cart:**

```
$oc = new OpenCart\OpenCart('yourdomain.com');
if ($oc->login('your_username', 'your_password')) {
    $oc->cart->add(40, 3);

    $cart = $oc->cart->products();

    foreach ($cart['products'] as $product) {
        echo $product['name']."\n";
    }
} else {
    echo $oc->getLastError();
}
```

You can also save your session for later use, just add a second parameter to the constructor of the OpenCart class. This should be a filename, where the cookie information will be stored. You can point this anywhere in your filesystem. Here is an example of the simplest form:

```
$oc = new OpenCart\OpenCart('yourdomain.com', 'cookiejar');
```

And here is a reference of all the methods in our library:

```
OpenCart->cart->add($product, $quantity = 1, $option = array());
OpenCart->cart->edit($key, $quantity);
OpenCart->cart->remove($key);
OpenCart->cart->products();

OpenCart->order->add($shipping_method = '', $comment = '', $affiliate_id = '', $order_status_id = '')
OpenCart->order->edit($order_id, $shipping_method = '', $comment = '', $affiliate_id = '', $order_status_id = '')
OpenCart->order->delete($order_id)
OpenCart->order->history($order_id, $order_status_id = '', $notify = '', $append = '', $comment = '')

OpenCart->payment->address($firstname = '', $lastname = '', $company = '', $address_1 = '',
```

```

$address_2 = '', $postcode = '', $city = '', $zone_id = '', $country_id = '')

OpenCart->payment->methods()

OpenCart->payment->method($payment_method)


OpenCart->shipping->address($firstname = '', $lastname = '', $company = '', $address_1 = '',
$address_2 = '', $postcode = '', $city = '', $zone_id = '', $country_id = '')

OpenCart->shipping->methods()

OpenCart->shipping->method($shipping_method)


OpenCart->reward->add($reward)

OpenCart->reward->maximum()

OpenCart->reward->available()


OpenCart->voucher->apply($voucher)

OpenCart->voucher->add($voucher_from_name = '', $from_email = '', $to_name = '', $to_email =
'', $voucher_theme_id = '', $message = '', $amount = '')


OpenCart($url, $sessionFile = '')

OpenCart->getUrl($method)

OpenCart->getCookie()

OpenCart->getLastError()

OpenCart->login($username, $password)

OpenCart->coupon($coupon)

OpenCart->customer($customer_id = 0, $customer_group_id = 0, $firstname = '', $lastname = '',
$email = '', $telephone = '', $fax = '')

```

## Final words

We hope this will help you get started with the OpenCart API. Feel free to suggest API functionality you think will be useful to the OpenCart project in GitHub: <https://github.com/opencart/opencart>

# OpenCart 2.0.x Event System Tutorial

Learn about the new event system in OpenCart 2.x and how to use it



```
1 <?php
2 class ControllerModuleMyModule extends Controller {
3     public function install() {
4         $this->load->model('extension/event');
5         $this->model_extension_event->addEvent('mymodule', 'admin.store.delete');
6         $this->model_extension_event->addEvent('mymodule', 'customer.add', 'mymodule');
7     }
8
9     public function uninstall() {
10        $this->load->model('extension/event');
11        $this->model_extension_event->removeEvent('mymodule');
12    }
13
14    public function on_store_delete($store_id) {
15        $this->load->model('setting/store');
16        $store_info = $this->model_setting_store->getStore($store_id);
```

## Overview

OpenCart 2.x introduces a number of new features for the module developers and one of them is the event system. This is going to be a quick tutorial teaching you how to make use of it in your modules.

The concept is simple, yet very effective. It allows you to write code which will get executed when something interesting happens in OpenCart. Like when an order is made, or a customer is created

without the need of using vQmod or OCMOD. By using the events system, you can avoid collisions which happen when you are using a modification system like vQmod or OCMOD. Additionally since you write the event handlers in PHP file, you get to enjoy proper code coloring/completion/linting from your editor/IDE.

## The Basics

Making a real use of events involves only two steps:

1. Register an event handler
2. Implement the event handler

The event handlers are simply methods defined in controller files. You can use one file for all your methods or you can create separate controller for your event handlers. To register your event handlers you need to use the extension/event model (OpenCart 2.0.1+) or the tool/event model for OpenCart 2.0.0.0. The extension/event model has 2 methods: `addEvent($code, $trigger, $action)` and `deleteEvent($code)`. As you can guess the `addEvent()` method is used to register event handlers and the `deleteEvent()` is used to unregister event handlers. It is a good practice to register your event handlers in the `install()` method of your module's controller in the `admin/` directory. It is also advised to unregister your event handlers when your module is being uninstalled, which can be done in the `uninstall()` method of the same controller file.

The **\$code** argument is used to group your event handlers. It is a good idea to set this to the name of your module, much like you have named your main controller file.

The **\$trigger** argument is the event name which you would like to set your handler to. A list of all available triggers can be found here: [https://github.com/opencart/opencart/wiki/Events-\(script-notifications\)](https://github.com/opencart/opencart/wiki/Events-(script-notifications)).

The **\$action** argument is the route to your handler function. It is written in the form of a standard route to a controller. For example `module/mymodule/on_user_created`.

## Example

We know that an example is worth a lot, so we will walk you through the steps of implementing an event handler with simple code examples. The example assumes that we are working with OpenCart 2.0.1+. Now let's assume that we are creating a module called "My Module". The admin controller file for the module will be *admin/controller/module/mymodule.php*. The catalog controller file for the module will be *catalog/controller/module/mymodule.php*.

The module will achieve 2 simple tasks - send e-mail messages to the administrator when a store is deleted and upon a customer registration. The triggers that we are going to use are **pre.admin.store.delete** and **post.customer.add**.

First we will register our event handlers in the **install()** method of our module:

```
public function install() {
    $this->load->model('extension/event');
```

```

$this->model_extension_event->addEvent('mymodule', 'pre.admin.store.delete', 'module/mymodule/on_store_delete');

    $this->model_extension_event->addEvent('mymodule', 'post.customer.add', 'module/mymodule/on_customer_add');
}

```

We should also take care of the uninstall process like this:

```

public function uninstall() {

    $this->load->model('extension/event');

    $this->model_extension_event->deleteEvent('mymodule');

}

```

Next, we need to implement the event handlers. The **pre.admin.store.delete** event is emitted in the admin area, so its handler must be implemented in the admin controller file for our module. We would like our event handler method to notify the administrator that a store has been deleted. Events starting with pre are emitted before the action is executed, and events starting with post are executed after the action has been taken. We would also like to include the store domain in our message, which is why we have chosen the “pre” event instead of the “post”. If we register for the **post.admin.store.delete** event, we will not be able to read the store’s domain.

Our event handler should look something like this:

```

public function on_store_delete($store_id) {

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

    $store_info = $this->model_setting_store->getStore($store_id);

    $admin_mail = $this->config->get('config_email');

    mail($admin_mail, "A store has been deleted", "The store " . $store_info['url'] . " was deleted.");

}

```



All that is left now is to implement the handler method for the **post.customer.add** event. We will do this in the catalog controller of our module. The method will notify the administrator when a new customer is registered. The method should look similar to this:

```
public function on_customer_add($customer_id) {
    $this->load->model('account/customer');
    $customer_info = $this->model_account_customer->getCustomer($customer_id);
    $admin_mail = $this->config->get('config_email');
    mail($admin_mail, "New Customer", "A new customer has just registered with the following
e-mail: " . $customer_info['email']);
}
```

**Note:** We are using the mail() function for sending e-mails for simplicity. In a real module you need to use OpenCart's Mail class for sending e-mails.

That's it! Our module is ready to rock. The final versions of our files is as follows:

*admin/controller/module/mymodule.php*

```
<?php
class ControllerModuleMyModule extends Controller {
    public function install() {
        $this->load->model('extension/event');
        $this->model_extension_event->addEvent('mymodule', 'pre.admin.store.delete', 'module/
mymodule/on_store_delete');
        $this->model_extension_event->addEvent('mymodule', 'post.customer.add', 'module/my-
module/on_customer_add');
    }

    public function uninstall() {
        $this->load->model('extension/event');
        $this->model_extension_event->deleteEvent('mymodule');
    }
}
```

```

public function on_store_delete($store_id) {
    $this->load->model('setting/store');
    $store_info = $this->model_setting_store->getStore($store_id);
    $admin_mail = $this->config->get('config_email');
    mail($admin_mail, "A store has been deleted", "The store " . $store_info['url'] . "
was deleted.");
}
}

```

### *catalog/controller/module/mymodule.php*

```

<?php
class ControllerModuleMyModule extends Controller {
    public function on_customer_add($customer_id) {
        $this->load->model('account/customer');
        $customer_info = $this->model_account_customer->getCustomer($customer_id);
        $admin_mail = $this->config->get('config_email');
        mail($admin_mail, "New Customer", "A new customer has just registered with the fol-
lowing e-mail: " . $customer_info['email']);
    }
}

```

## Advanced

Apart from the standard use, the event system can be used to create cross-module integrations. Using the **Event** object (`$this->event`), you can trigger any event at any point. You can use it to trigger your own events as well. Imagine that you are developing a module for customer comments. You can trigger events on a comment post, for example. This will allow other module developers to create handler methods for your events and execute code without the need of vQmod or OCMOD. This will ensure better system stability and better customer experience. You can also use the system

to register and trigger your own events at runtime.

The Event class is defined in the *system/engine/event.php* file, so you can take a look at it for further reference.

## Conclusion

Mastering the events in OpenCart is beneficial in many ways, both for the developers and the customers. When possible you should always opt for the events instead of a modification system like vQmod and OCMOD. It is less likely that your extensions will collide with the other developers' extensions which leads to better customer experience and less work for you to fix collisions. It also allows for inter-module communication, so you can make several modules working together nicely and elegantly.

# Supercharge OpenCart with OpenPack

One of the most common sales request we had over the last year was a very simple one - a bundle with all iSenseLabs extensions. As we used to accommodate this need individually for you over email and per request, I am thrilled to share that last week we released a publicly available product! Meet OpenPack - a full-blown bundle of all iSenseLabs OpenCart extensions that provides extra features previously not available.



## Unlimited access to all extensions

By purchasing OpenPack you get a full access to all of our extensions, themes and modules. What is more awesome is that each

time we release a new extension, this automatically get added to your account as well, and for free! It will not be a hassle anymore to hunt for new extensions or new versions, if released - you have it. We believe this offering of functionality base perfectly suits web agencies, freelancers, multi-store owners and all people building more than one store per year.

## **Big deal**

OpenPack final price is discounted with more than \$2200 of the total price of all extensions it includes. This is more than 72% of discount! What's even better is that the real discount will continue growing as each new extension we release is automatically added to the pack.

## **Lifetime ownership**

When you purchase OpenPack or any iSenseLabs module you own the extension forever. It won't stop or change with the expiration of your support and updates license. It will just work fine. The only thing that lasts is the access to premium support and future updates and releases after your license expires.

## **Fully open source**

All of our extensions are open source. A well written open source code with meaningful class names and methods.

## Premium support

The same great support you get for your individual modules is available for the OpenPack extensions. Our default premium plan features 48-hour ticket reply time as most of the tickets get answered within the first 8 hours. On benchmark this is way above the industry-standard first support reply of 22.2 hours ([benchmark source](#)).

## Try out

With OpenPack you have the option to try installing extensions on your store you would not normally purchase or not sure of how they would function. You can also easily experiment adding features to your store and measure conversion and user feedback.

## Half-price license renewals

The license renewal is optional and if you decide to go for it - it is half-priced! All existing customers enjoy 50% off on renewals, any-time.

## Existing customer? Enjoy upgrade discount!

All existing customers can upgrade to OpenPack at discounted price. The total price you would need to pay will deduct the already purchased active licenses you have, and you pay only the difference.