Usage Guide

Contents

Introduction

Egg Basket is a website framework for eCommerce development. The framework implements many of the standard features found on online shopping websites and offers a complete, working checkout system with a selection of built-in payment gateways. Egg Basket is essentially a complete online store ready to be skinned and populated.

The framework also implements a fully working administration panel for end users that offers many of the features found in other eCommerce frameworks. Both the administration panel and website front-end have been built to be as easy as possible for web developers to build new features into and therefore serves as an ideal base system to develop your own, bespoke eCommerce websites.

Egg Basket is a continuous development, built and maintained by QWeb Ltd and released under the terms of the GNU GPL as of version 1.9.

System Requirements

Egg Basket requires PHP 5 and MySQL 5. The framework makes use of the Apache 2 mod_rewrite and mod_negotiation modules as well as various htaccess rules.

As of version 1.2, Egg Basket includes an IIS module and has been tested to work on IIS servers running PHP as an ASAPI extension. See **settings.php** for more information.

Various PHP modules may also be required for certain functionality, such as the GD libraries for thumbnail generation. All modules, libraries and software required by Egg Basket are freely available.

Features

Egg Basket implements an array of core features including the following. As of version 1.01, a changelog is also included in **version.txt**.

- Well structured, widget based procedural code with MVC style methods.
- Responsive wireframe for mobile friendly eCommerce.
- Plugin system for additional, modular functionality.
- Local tree for safely amending core files without preventing updates.
- Unlimited categories, sub categories, products and product variants.
- Unlimited product images.
- Support for physical and downloadable products.
- Stock management and intelligent stock reservation system.
- Google product feed.
- Advanced offers system.
- VAT calculation and per-product exemption.
- Intelligent delivery calculation matrix.
- Built in regions and countries database.
- Built in tracking links in dispatch e-mails for most popular couriers.
- Sagepay Direct, Sagepay inFrame, Barclaycard ePDQ and PayPal payment gateways.
- Real-time currency conversion built in for most countries.
- Quick toggle maintenance mode.
- Automatically generated breadcrumb navigation.
- Automatically generated, multi-mode thumbnail generation.
- Content managed pages.
- Per-account order history.
- CSV mailing list.
- Search functionality.
- Search engine optimised.

Installation

To get started with Egg Basket, follow these steps:

1. Download the latest version from egg-basket.com.
2. Extract Egg Basket and copy the contents of the extracted httpdocs folder to your web space. Be sure to include any hidden **.htaccess** files.
3. Import the extracted SQL file into a new database on your server.
4. Set **cms-files**, **product-files**, **product-images** and **category-images** to be writeable by your web server (chmod 777 on Linux/Unix).
5. Copy **required/settings.php** to **local/required/settings.php** and populate all variables, using the included comments for guidance. Most importantly, populate the database connection variables, and if you've extracted Egg Basket in to a sub directory, be sure to populate the urlPrefix variable.
6. If populating the urlPrefix variable, also remember to amend the RewriteBase lines of both **.htaccess** and **admin/.htaccess**.
7. For security reasons, you should rename **hack-maintenance.php** to something else. Navigating to this file in your web browser gets around maintenance mode, allowing you to see and use the website as if it were live.
8. By default there will be one administrator log-in set up in tblAdmin of the database. Amend the username and password of this account to something of your choice. The password should be encrypted with MySQL's SHA1() function.

Your new website should at this point be functional. Unless you have disabled maintenance mode in **settings.php**, you'll need to navigate to your hack-maintenance file before you can see anything. It is recommended that you replace **maintenance.html** with your own holding page, remembering that you may need to put the website back into maintenance mode at a later date during updates.

To learn about administration, skip to the administration section of this guide. Alternatively, skip to the section about themes.

Administration

By default, the administration panel is located at **/admin**. If you haven't already done so, amend or create a log-in via the tblAdmin table of the database. Passwords must be stored as a SHA1 hash.

The administration panel has been built purposefully to look and feel quite minimal, making it both easy for the end user to navigate, and easy for developers to brand up or build upon. The panel expects to find **images/logo.png** within your chosen theme folder for use in the header.

Orders

Orders are split in to four categories. New, complete, refunded and failed. When a new order is placed and payment is made, this order will appear in the new orders section. If payment failed for some reason, it will appear instead in the failed section.

Each section shows a table of orders. Clicking on the order numbers shows more detail about each order, and clicking on the customer name shows more detail about each customer. If the customer is a registered member, you can also navigate to that members account information and order history from this page.

The order details shown include an itemised list of the products ordered, the payment method used and any errors or messages returned by the payment gateway, and the customers billing and delivery addresses.

If the order has been paid for you can update the dispatched quantities and send a dispatch notification e-mail to the customer from here. E-mails show which products have been dispatched so multiple dispatch notifications can be triggered if the order is to be shipped in parts. If the delivery courier has provided a tracking code, be sure to add this to the system before sending any dispatch notifications so that the notification can provide the customer with tracking links. Multiple tracking links are supported for each order. You can also update the dispatch quantities without sending a dispatch notification e-mail, which is useful for marking orders as complete that don't contain deliverable goods.

Once all items have been marked as dispatched, the order will automatically be moved to the completed orders section.

Depending on the payment gateway used, refunding an order via the gateways own administration interface may cause Egg Basket to automatically move the order into the refunded section. You can also manually mark orders as refunded, or undo the move.

Customers

The customers menu links to a list of registered user accounts, and a downloadable CSV export of customers who have opted in to the mailing list.

The accounts list shows how many orders each customer has placed, inclusive of orders placed before registration. Clicking on a customers name shows more information about each account, including their current billing and delivery addresses and their order history. It is also possible from here to resend the activation e-mail if the customer had trouble receiving one during registration.

The CSV export contains the name and e-mail address of each customer that has opted in during checkout or registration. Most e-mail campaign software supports CSV import.

Content

Egg Basket implements a complete content management system, separated into two sections – Pages, and Files.

By default, Egg Basket includes a few pages ready to be populated that cannot be deleted. Pages can be flagged as exempt from editing/deleting via the tblPages table within the database.

When creating new pages, the url automatically populates based on the page title and parent, although it can still be modified to something more specific.

The content of the page is populated via the TinyMCE WYSIWYG editor. TinyMCE is provided under an open source license and is very easy to build upon. In addition to the standard editor features, there are three further drop-down fields on the editor toolbar for inserting thumbnails, images and files into the page. These lists are populated by the files administration section and can be used to insert image tags, download links, or pseudo thumbnail tags.

Thumbnail tags are converted into thumbnails on the website front-end, which can be clicked to display a larger view of the image in a modal window. This is most useful for gallery pages.

The editor uses **styles/cms.css** within your chosen theme folder for text formatting, also used by the website front-end. The first style block within this file is used to tell TinyMCE how wide the content area is. Provided this is set correctly, how a page looks in the editor should be the same as how it looks on the website.

The files section is essentially a file manager for the **cms-files** folder. Multiple files can be uploaded asynchronously but the page should not be closed before all uploads are complete. Any type of file can be uploaded and Egg Basket will automatically decide which list to add them to in the page editor.

It is advised that large images are reduced to a canvas size of about 500px before upload, to ensure faster page load times when used on the front-end. To help end users with this, Egg Basket includes a link to batchimageresize.com.

Stock

Egg Basket supports an unlimited number of categories, sub categories, products, and product variants.

When creating categories, a description can be set with the same editor as page creation. Description output can be disabled via **settings.php** but the text is still used for SEO, so it is advised all categories are given this description.

Like with categories, product descriptions are used for SEO too. Creating products has a few more fields to categories, including an image field, used to set the primary product image, and a weight, used for shipping calculation.

From the products list, in addition to editing and deleting the products, there are also links to amend the stocks of each product and to add more images. You can upload an unlimited number of images to each product and they'll be used on the front-end as a gallery.

Product stocks are where variations can be set. New products are given a default variation record so that a price and stock level can be set. If the product has options, these should be added as new records with individual stock counts, prices and SKU's, and the option text of the default record should be changed. For example a shirt may have small, medium and large variations, each with a stock count and a price. On the front end, these variations are offered to customers as a drop menu.

Offers

Egg Basket implements an advanced offers system. This system supports voucher codes, seasonal reductions, bundled item discounts, and combinations of all types. You could for example have a discount that is only applied if you enter a specific voucher code during the Christmas period and have chosen to add 5 specific products to your basket.

Instructions on creating offers are included in the administration section.

Shipping

Egg Basket implements an intelligent shipping calculation matrix. The matrix already knows of all ISO recognised countries and regions and it is advised that at least one cost is input for each of the recognised countries. It is best to then input costs for each region of the local country, ideally using weight bands.

When an order is placed, the combined weight of each product in the basket is used, together with the destination country and region, to compute an accurate shipping cost based on all of the rules defined in this section.

For a fixed delivery cost to a country or region, regardless of shipment size, simply leave the weight as 0.

Themes

Egg Basket is provided with a responsive wireframe to give you something to design around, but to get up and running with a complete website quickly, it is recommended that you purchase one of the official Egg Basket themes from egg-basket.com. Simply extract the theme to your **themes** folder and amend the theme variable of **local/required/settings.php**. With this approach, it is possible to have a fully working, skinned shop up and running within hours of extracting the main Egg Basket framework. You may also build upon these themes to create a more unique looking website with minimal effort.

Alternatively, to create a new theme simply copy the **default** folder within **themes** to a name of your choice, and amend the theme variable of **local/required/settings.php**. A theme is essentially a collection of images and styles used to define how the website looks. There is also a **header.php** file which gets injected in to the head of each website page allowing you to include theme specific Javascripts, for example.

Inside the **styles** folder are, by default, two CSS files. The **cms.css** file is used both by the website front-end and the page editor within the administration panel, so text formatting should be defined here. The first style block within this file is used to tell the page editor how wide the content area is, so provided this is set correctly, how a page looks in the editor should be the same as how it looks on the website. The **screen.css** is the last CSS file to be loaded so rules defined in here have priority over **cms.css** and the wireframe itself.

Inside the **images** folder is the logo file. This is used both by the website front-end and the administration panel for branding, and should therefore be replaced with a file of the same name. The images used by the e-mail template are also found here. You should amend **emails/header.jpg** for branding. To edit the e-mail template itself, please refer to **required/email-settings.php** found outside of the theme package.

Further Development

Egg Basket is a framework designed to be built upon. From the ground up, it has been developed to be quick and easy to work with.

The core of Egg Basket is **required/settings.php**. The variables defined here control the entire website and should be populated before any other development. It is advised that you copy this file to **local/required/settings.php** before population.

Egg Basket has been built using procedural code, but following an MVC style methodology. Almost all requests are handled by the root files, acting as controllers. These controllers first set everything up by calling the core files in **required**, and then call request-specific script files in **includes**. These are effectively the models. Finally, with all the processing completed, the **header.php** and **footer.php** files are called from within **structure** along with a page-specific include file from **includes**. These are the views.

The view (include) files are really just layout templates. These call widget files, also found in **includes**, which host the bulk of the HTML. Widgets can be called from just about anywhere, and in just about any order.

All database communication in Egg Basket uses wrapper functions defined in **required/database.php**. This makes it easy to switch between database engines provided the chosen engine follows the standard SQL command set.

Required/globals.php contains all of the functions that need to be made available to all, or most of the pages in Egg Basket. There are many powerful functions in this file, including currency conversion and shipping calculation so it's a good idea to learn what's available.

Requests that don't directly translate to a root file, or to a file in **scripts** or **gateways**, are either handled by the **.htaccess** file and rerouted, or by **engine.php**. The engine file determines whether the requested page is a product, category, sub-category or a CMS page and calls the appropriate controller for each.

Egg Basket version 1.3 and above includes a **local** directory in the root. Rather than editing any existing PHP files, you should create a clone of the file you wish to rework inside this local tree, inclusive of it's full directory path, and then only edit the new copy. The framework will use this file instead and updates will never overwrite anything within **local**. This new mechanism allows you to make almost any changes you like and still be free to merge new version releases.

Egg Basket version 1.4 and above includes a **plugins** directory in the root. You can add new front-end functionality and/or administration panel functionality by creating or installing plugins into this tree. See the Plugin Development section below for further details.

You should also check out the comments within the framework code for more technical information.

Plugin Development

In Egg Basket, a plugin is a block of code that can literally be copy + pasted right in to add additional functionality without needing to modify any core files. Egg Basket includes a sample plugin that can be enabled to help illustrate how this system works. This sample should always be left disabled, or deleted entirely, from production websites.

Plugins are located in the **plugins** directory. All plugins must be kept in their own directory within this tree and the directory name must be used as an ID when hooking in to the framework, so developers should name these directories appropriately.

A plugin consists of at least a **plugin.php** for front-end functionality, or an **admin.php** for administration panel functionality. One plugin can implement either or both of these files and to help you understand their purpose, the **sample-plugin** plugin includes both.

Throughout the framework there are various 'hooks' for assigning plugins to. When the engine reaches one of these hooks, if your plugin has been registered to that hook ID, $initPlugin is created and your **plugin.php** gets launched. All **plugin.php** files found within the **plugins** directory are launched during the initial page request, by **globals.php**, and $initPlugin does not exist at this time. Effectively your **plugin.php** should first check for the existence of $initPlugin and then either run pre-output code such as loading data from the database or processing a form submission, or should output something visual such as a form.

Refer to hooks.txt for a list of standard hook ID's.

To register a plugin to a hook ID, your **plugin.php** should call registerPlugin(). This function takes 3 parameters. The first is a string and should be the same as the directory name your plugin resides in. The second is a string and should be the hook ID you want your plugin to be called at. The third is an optional array for your processing code to pass data back to the output code, for example success/error messages or options from the database. You can call registerPlugin() multiple times to assign your plugin to multiple hook locations.

Similarly, all **admin.php** files are called during the initial admin panel requests to allow processing scripts to run, and you can call registerAdmin() to create an admin menu tab. When **admin.php** is called from a menu tab, $initAdmin is created for you to check the existence of and either run a processing script or something visual such as a form.

Egg Basket implements a database table, tblPlugins, specifically for plugin developers to store settings in. This table consists of these fields:

- fldID, an auto-increment field you can ignore completely.
- fldPluginID, a varchar field you should use to identify your plugin records. You should use your plugin directory name for this.
- fldCategory, an integer field you can optionally use to reference a category ID.
- fldProduct, an integer field you can optionally use to reference a product ID.
- fldPage, an integer field you can optionally use to reference a page ID.
- fldData, a longtext field you should use for storing your settings in. It's recommended that you store serialized arrays using serialize() or json_encode().

Refer to the **sample-plugin** plugin for a technical example of how to build plugins that implement all of this, and check out the collection of official plugins at egg-basket.com to quickly add additional functionality to your store.

Upgrading

To get started with Egg Basket, follow these steps:

1. Download the latest version from egg-basket.com.
2. Before upgrading, ensure you have an up to date back-up of your website files and database, and ensure any modifications have been applied to files inside **local** rather than the core otherwise your changes may be overwritten.
3. Toggle maintenance mode to on via **settings.php**
4. Extract Egg Basket and copy the contents of the extracted httpdocs folder to your web space allowing files to be overwritten. Be sure to include any hidden **.htaccess** files.
5. It's possible that the database structure has altered in the newer version. You should import the extracted SQL file into a new database on your server and compare the tables. Alternatively, extract a dump file of your existing database and compare the SQL via a text editor or diff utilities.
6. Make sure **cms-files**, **product-files**, **product-images** and **category-images** are writeable by your web server (chmod 777 on Linux/Unix).
7. If you didn't create **local/required/settings.php** file before upgrading, re-populate your usual settings now. You should check the new **required/settings.php** file for additional variables too.
8. If you previously renamed **hack-maintenance.php**, you should delete the new file.
9. Check that everything is working properly and if so, toggle maintenance mode again via **settings.php**