

Developer Manual

Tradingeye php v6 - Developer Manual

Contents

- 1) [Template system](#)
- 2) [Our Approach](#)
- 3) [Structure used](#)
- 4) [Configuration](#)
- 5) [FAQ](#)
 - a. [Can we edit the html templates?](#)
 - b. [How to change the layout of the site?](#)
 - c. [How to add a new department template?](#)
 - d. [How to add a new product template?](#)
 - e. [How to add a new content template](#)
 - f. [How to add new module?](#)
 - g. *How to install Tradingeye in a subfolder?*
 - h. *Integrating the style sheet switcher as seen on the Tradingeye demo*
- 6) [Database structure](#)
- 7) [Programmatic Flow](#)

1) We use PHPLib templates.

[Top](#)

Reason:

PHPLib templates separates business logic and data presentation. PHPLib templates are based on php function preg_replace which is faster than ereg_replace used by Fast templates.

2) Our approach: A modular approach following the MVC structure

[Top](#)

There are two sections:

- a) Admin section
- b) Front end

Admin section is divided into six modules.

- i) Ecom – This is the shop module. I.e. adding departments, products, articles, menus, options for products, creating packages are handled in this module.
- ii) User – This module is related to user management, supplier management etc.
- iii) Sales – All the promotions, email campaigns, froogle feeds, and reports related to sales, products are handled in this section.
- iv) Admin – All the site settings, admin accounts are handled in this section.
- v) Order – All the sold product invoices are handled in this section.
- vi) Default – All the common functions are handled in this section

Front end is divided into three modules.

- i) Ecom – In this module all the listing of products, articles, departments, cart display, checkout process, payment gateways are handled here.
- ii) User – In this module user registration, user account (profile), login etc. are handled here.
- iii) Default – All the common functions are handled in this section

3) Structure Used

[Top](#)

- i) There is directory for each module at the root level, with an "index.php" file if the module is used in the frontend & "adminindex.php" file, if the module is used in the admin section. There are also message files for both admin and frontend in the directory. You can edit all the messages for any particular module in the message files.
- ii) Except the module directories other directories in the root are
 - a. /config – This directory must have writable permissions. As Tradingeye writes a dbConfig.php in this directory during the installation and when we update the site settings.
 - b. /css – This directory has all css files used in admin and front end section.
 - c. /FCKeditor – has all the content editor component files.
 - d. /froogle – this directory must have writable permissions if Froogle functionality is used.
 - e. /graphics – This directory has all the graphics related to the site.
 - f. /images – This directory must have writable permissions. All the images from departments, products, articles, menus, options are uploaded in respective sub directories within the images directory. For example: department related images will uploaded to department sub-directory, product related images will uploaded to sub-directory product etc. This folder will not have sub-directories for Tradingeye 6.
 - g. /installs – This includes all the installation related files. Please remove this directory after installation.
 - h. /ioncube – This directory has library files to decode all the encoded files in the /libs directory.
 - i. /jscrip – has all the JavaScript files.
 - i. This directory file has commonly used JavaScript files. Like calendar.js, calendar.html used to display popup date picker calendar which is used a number of times.

- ii. Admin.js for admin related common JavaScript.
 - iii. Validations.js – has all the common functions to validate data.
- j. /libs – This directory has all the encoded files. These files should not be edited. This directory has files for the database, templates, image uploading, image resizing, mail sending etc.
- k. /modules – This directory has all the modules of Tradingeye. This directory has six subdirectories i.e. six modules we discussed earlier and two main controller files. One for the front end and one for the admin section.
 - i. Every module except the default module has two module controller files one for the admin and one for the the front end.
 - ii. Every module also has two subdirectories: for classes and templates.
 - iii. Each subdirectory is further divided into two subdirectories one is for the the admin and one is for the main. Admin subdirectory includes all the files related to the admin section and main has all the files related to the front end.
 - iv. Every module has a number of classes each handles the functionality for one link. For each link there are two classes one handles all the display related functionality and the other handles all the functionality related to inserting, updating, deleting etc. For example ecom module (builder) in admin has four links i.e. Shop builder, Menu builder, Option builder, Package builder. There are 8 classes to handle the functionality for this module. Class appended with “_interface.php” will handle all functionality related to the display and class file appended with “_db.php” will handle all inserting, Updating etc.
 - v. All the functions are called from the module controller file.
- l. /scheduler – This directory has a file that can be set as a cronjob to automatically update products in froogle.
- m. / UserFiles – UserFiles directory and its subdirectories must be writable as all the images added from FCKeditor are uploaded to this directory.

4) Configuration

[Top](#)

Most of the configuration is admin managed. There are few variables that are changed rarely and are managed in the configuration.php.

DEFAULTVATTAX – This constant is defined for default vat tax. This vat tax is used to display prices if a user is not logged in. Otherwise the price is displayed according to the country the user belongs to. The current value for this constant is United Kingdom tax value i.e. 15%

DEFAULT_CSS – This constant is used to specify the default CSS file name. You can change the CSS by simply changing the file name and uploading your file to the /css folder.

ABSOLUTE_IMAGE_PATH – This constant is added in new version of Tradingeye. You can specify its value to use an absolute path for images. Otherwise images are displayed relative to the root.

DEPT_PRODUCT_LIMIT – This constant is also added in new versions. You can change the value to change the number of products displayed in the department template.

5) FAQs

i) Can we edit html templates?

[Top](#)

You can edit templates according to your requirements but make sure that commented lines in the templates with keywords **BEGIN / END** are not deleted or the page will crash. These commented lines are required. Yes, you can modify and delete all the text between these commented lines.

ii) How to change the layout of the site?

[Top](#)

The path where the site layout template exists is

/httpdocs/modules/default/templates/main/layout/

The default template for the site is layout.htm. Please don't delete this file. This file provides the layout for the entire front end of the site. You can modify this. Yes, you can create new templates and upload them to the layout path i.e. **/httpdocs/modules/default/templates/main/layout/** and these templates can be used while displaying departments, products, contents (articles) as you can see the selection option for the layout template while adding new products, departments, content (articles).

Code for layout.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-Transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
```

-----SEO-----

```
<title>{TPL_VAR_METATITLE}</title>
<meta name="description" content="{TPL_VAR_METADESCRIPTION}" />
<meta name="keywords" content="{TPL_VAR_METAKEYWORDS}" />
<meta name="title" content="{TPL_VAR_METATITLE}" />
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="Content-Language" content="en-gb" />
<meta name="robots" content="all" />
<meta name="Revisit-After" content="7 days" />
<meta http-equiv="imagetoolbar" content="false" />
<meta name="MSSmartTagsPreventParsing" content="true" />
<meta name="author" content="dpivision.com Ltd" />
```

The above code is for search engine optimization you can modify it according to your requirement. You may have noted the template variables

- 1) {TPL_VAR_METATITLE}
- 2) {TPL_VAR_METADESCRIPTION}
- 3) {TPL_VAR_METAKEYWORDS}

These three variables value can be changed from the admin section. Otherwise you can replace them here in the template with your hard-coded value.

-----PATHS-----

```
<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />
<link rel="stylesheet" type="text/css" href="/css/{TPL_VAR_CSSFILE}" media="screen" />
<link rel="stylesheet" type="text/css" href="/css/inline_editor.css" media="screen" />
<link rel="stylesheet" type="text/css" href="/css/print.css" media="print" />
<script language="Javascript" type="text/javascript" src="/jscript/validations.js"></script>
```

Above are the paths for CSS and common JavaScript files. Mainly you need to modify them when you have to run the shopping cart from subdirectory. Then you need to have an absolute path. The only modification you have to do is copy {TPL_VAR_SITEURL} in front of all relative urls.

e.g. <link rel="stylesheet" type="text/css" href="/css/inline_editor.css" media="screen" /> can be modified as

```
<link rel="stylesheet" type="text/css" href="{TPL_VAR_SITEURL}css/inline_editor.css" media="screen" />
```

Note: There is no slash after {TPL_VAR_SITEURL}

```
</head>
```

```

<body>
-----INLINE EDITOR-----
-----

<!-- BEGIN TPL_VAR_INLINE_EDITOR_BLK -->
<div id="inline_editor" class="null">

    <!-- [DRK] -->
    <dl class="inline_menu">
        <dt>Menu</dt>
        <dd><a href="#" onclick="document.getElementById('inline_editor').id =
'inline_editor_hidden'; return false;">Hide Side Menu</a></dd>
    </dl>
    <!-- [/DRK] -->

    <dl class="inline_menu">

        <dt id="builder">Editor</dt>

        <dd><a href="{TPL_VAR_EDITLINK}">Edit This Page</a></dd>

        <dd><a href="{TPL_VAR_ADMINURL}" target="_top">Load Admin</a></dd>

        <dd><a href="{TPL_VAR_ADMINLOGOUT}" target="_top">Admin Logout</a></dd>

        <dd><a href="javascript:history.go(-1)" target="_top">Previous Page</a></dd>

        <dd><a href="{TPL_VAR_ASSOCIATE_PRODUCTLINK}" target="_top">Associate
Product</a></dd>

        <dd><a href="{TPL_VAR_ASSOCIATE_ARTICLELINK}" target="_top">Associate
Article</a></dd>

    </dl>

    <!-- [DRK] -->
    <p id="show_menu"><a href="#" onclick="document.getElementById('inline_editor_hidden').id
= 'inline_editor'; return false;"><span>S<br />h<br />o<br />w<br /><br />M<br />e<br />n<br
/>u</a></span></p>
    <!-- [/DRK] -->

</div>
<!-- END TPL_VAR_INLINE_EDITOR_BLK -->
-----

```

Above code is to display the inline editor when the admin is logged in. If you don't need this then please remove this except these two lines

```

<!-- BEGIN TPL_VAR_INLINE_EDITOR_BLK -->
<!-- END TPL_VAR_INLINE_EDITOR_BLK -->

```

Note: Please don't delete these two lines otherwise your template will not work.

```

<p id="skip">[<a href="#main">skip to main content</a>]</p>

<div id="container">
<div class="inner">
-----HEADER-----
-----

```

```

<div id="header">
  <div class="inner">

    <h1 id="logo"><a href="{TPL_VAR_HOMEURL}"
title="{TPL_VAR_SITENAME}">{TPL_VAR_SITENAME}</a></h1>

    <h2 id="tagline">{TPL_VAR_SLOGAN}</h2>

    <p id="info"><a href="{TPL_VAR_CARTLINK}">Items in Basket</a>:
{TPL_VAR_TOTALQTY} Total: {TPL_VAR_CURRENCY}{TPL_VAR_GRANDTOTAL}</p>

  </div>
</div>

```



HEADER Code displays the Company name, Company slogan, total quantity in basket and total price etc. You can modify the above code to change the header area of the site.

-----MIDDLE-----

```

<div id="mid">
  <div class="inner">
    <!--Bread crumb trail start-->
    <p id="breadcrumbs">You are here: <a href="{TPL_VAR_HOMEURL}">Home</a>
{TPL_VAR_BREDCRUMBS}
    </p>
    <!--Bread crumb trail end-->

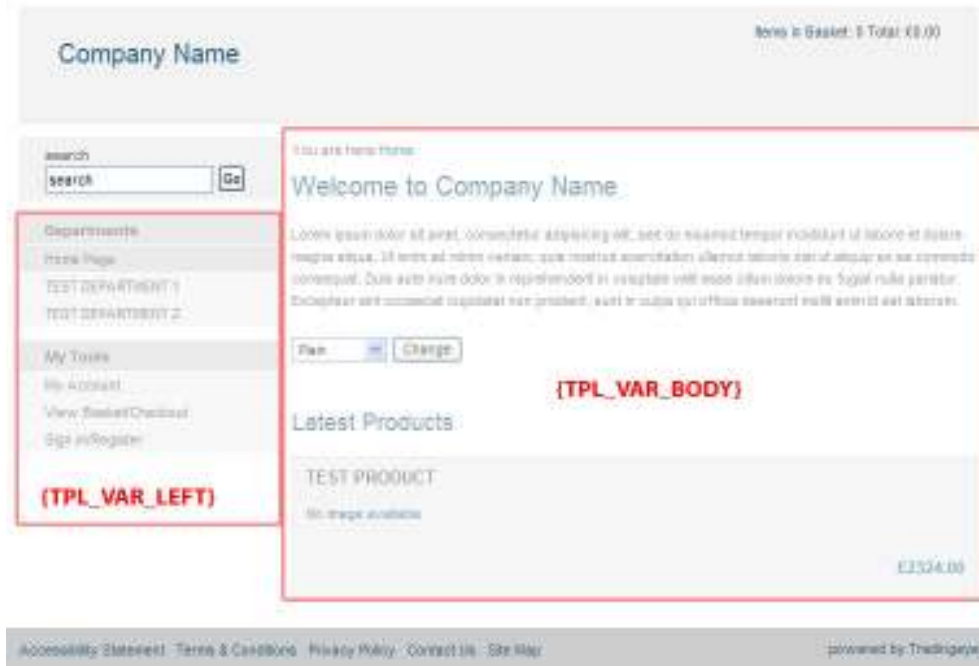
    <div id="side">
      {TPL_VAR_LEFT}
    </div><!-- end side -->

    <div id="main">
      <div class="inner">

        {TPL_VAR_BODY}
      </div><!-- end main/inner -->
    </div><!-- end main -->

  </div><!-- end body/inner -->
</div><!-- end body -->

```



This is further divided into two portions

- 1) Left portion – rendered by variable {TPL_VAR_LEFT} and can be modified from the template **/htdocs/modules/default/templates/main/leftMenu.tpl.htm**
- 2) Body portion – This portion is rendered by {TPL_VAR_BODY} and the template changes according to action we are performing. i.e. This is the only area that changes with every action.

Code from leftMenu.tpl.htm



- Departments
- Home Page
- TEST DEPARTMENT 1
- TEST DEPARTMENT 2

- My Tools
- My Account
- View Basket/Checkout
- Sign in/Register

```
<div class="inner">
*****Search
Box*****
  <form id="search" method="post" name="search" action="{TPL_VAR_SEARCHURL}">
    <label for="searchKeyword">search</label>
    <input type="text" name="mode" id="searchKeyword"
onfocus="if(this.value=='search')this.value='';" onblur="if(this.value=='')this.value='search';"
value="search" maxlength="50" />
    <input type="submit" name="go" id="searchSubmit" value="Go" />
  </form>
```

Above code displays this search box. You can modify / delete this portion

```
*****Departments*****

  <h3 id="navDeptTitle">Departments</h3>
  <ul id="navDept">
    <li class="first"><a href="{TPL_VAR_HOMEURL}" {TPL_VAR_HOMECLASS}
title="Home Page"><span>Home Page</span></a></li>
    <!-- BEGIN TPL_DEPARTMENT_BLK -->
    <li><a href="{TPL_VAR_DEPTURL}"
{TPL_VAR_DEPTCLASS}><span>{TPL_VAR_DNAME}</span></a></li>
    <!-- END TPL_DEPARTMENT_BLK -->
  </ul>
```

Above code displays the department list. While modifying please don't delete these two commented lines.
<!-- BEGIN TPL_DEPARTMENT_BLK --> and <!-- END TPL_DEPARTMENT_BLK -->
*****My Tools*****

```
<h3 id="navToolsTitle">My Tools</h3>
```

```

        <ul id="navTools">
            <li class="first"><a href="{TPL_VAR_MYACCOUNT}"
{TPL_VAR_ACTCLASS}><span>My Account</span></a></li>
            <!-- BEGIN TPL_WISHLINK_BLK -->
            <li><a href="{TPL_VAR_WISHLIST}" {TPL_VAR_WISHLISTCLASS}><span>My
Wish List</span></a></li>
            <!-- END TPL_WISHLINK_BLK -->
            <li><a href="{TPL_VAR_VIEWCART}"
{TPL_VAR_VIEWCARTCLASS}><span>View Basket/Checkout</span></a></li>
            <li><a href="{TPL_VAR_SIGNUP}" {TPL_VAR_SIGNUPCLASS}>
<span>{TPL_VAR_LABEL}</span></a></li>
        </ul>
*****
*****

```

You can modify the above portion but make sure you do not delete the commented lines.

```

<!-- BEGIN TPL_WISHLINK_BLK --> and <!-- END TPL_WISHLINK_BLK -->

*****MENU
SECTION*****
        <!-- BEGIN TPL_MENUHEAD_BLK -->
        <h3 class="navStaticTitle">{TPL_VAR_MENUHEAD}</h3>
        <ul class="navStatic">
            <!-- BEGIN TPL_MENU_BLK -->
            <li><a href="{TPL_VAR_LINK}"
{TPL_VAR_HREFATTRIBUTES}><span>{TPL_VAR_MENUTITLE}</span></a></li>
            <!-- END TPL_MENU_BLK -->
        </ul>
        <!-- END TPL_MENUHEAD_BLK -->

```

Above code can be modified except for the commented lines. If you don't want the menu section then please delete all the code except for these commented lines.

Note: There should be only one space between BEGIN and block name. Similarly for END and block name.

```

</div>

```

```

-----FOOTER-----
-----
<div id="footer">
    <div class="inner">
        <ul>

            <li class="first"><a href="/accessibility/">Accessibility Statement</a></li>

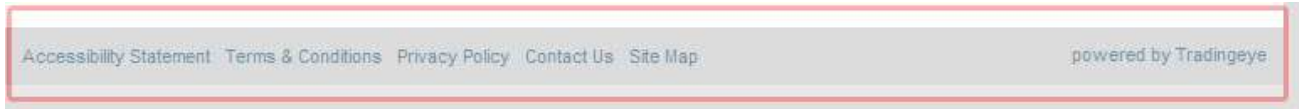
            <li><a href="/terms/">Terms &amp; Conditions</a></li>

            <li><a href="/privacy/">Privacy Policy</a></li>

```

```
<li><a href="/contactus/">Contact Us</a></li>
<li><a href="/sitemap/">Site Map</a></li>
</ul>
```

```
<p id="credits"><a href="http://www.tradingeye.com/">powered by
Tradingeye</a></p>
</div>
</div>
```



To edit the selected portion please modify the code in the FOOTER

```
</div>
</div><!-- end container -->

<div id="extra"></div>

<!-- Google Analytics -->
{TPL_VAR_GOOGLEANALYTICS}
```

Above template variable is for google analytics. You can remove it if you don't want google analysis.

```
</body>
</html>
```

iii) How to add a new department template?

[Top](#)

You have to create new html template file and upload it in **/httpdocs/modules/ecom/templates/main/department/** then this template will be listed in the template selection box in the admin section.

Just copy the code from any existing template and modify the new template according to your requirements.

Screenshot for the department when department.htm is used:

Welcome to Company Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Latest Products

No image available	The Princess Sleeps Here ★★★★★ 0 reviews Add to Compare £12.55
No image available	The Stare9 ★★★★★ 0 reviews Add to Compare £53.20
No image available	mine ★★★★★ 0 reviews Add to Compare £88.00

Here is the code for the department.htm

```
<h2 id="pageTitle">{TPL_VAR_DEPARTMENTTITLE}</h2>
<!-- BEGIN TPL_MAINDEPT_BLK -->
  <ul id="navSubDept">
    <!-- BEGIN TPL_DEPT_BLK -->
      <li><a href="{TPL_VAR_DEPTURL}"><span>{TPL_VAR_TITLE}</span></a></li>
    <!-- END TPL_DEPT_BLK -->
  </ul>
<!-- END TPL_MAINDEPT_BLK -->
```

TEST DEPARTMENT 1

TEST SUB 1

TEST SUB 2

The above code handles the display of the following part of the department.htm template.

First line displays the department heading. You can use the template variable {TPL_VAR_DEPARTMENTTITLE} to display the department title anywhere in the department template.

After that there are blocks to display the sub departments. While editing please don't delete these commented lines.

```
<!-- BEGIN TPL_MAINDEPTDESC_BLK -->
<div id="content">
{TPL_VAR_DEPARTMENTDESC}
</div>
<!-- END TPL_MAINDEPTDESC_BLK -->
```

TEST DEPARTMENT DESCRIPTION

Products

Above code will handle the department description.

```
<!-- BEGIN TPL_MAINPRODUCT_BLK -->
<div class="products">
  <h2>Products</h2>
  {PagerBlock1}
  <ul>
    <!-- BEGIN TPL_PRODUCT_BLK -->
      <li>
        <h3><a href="{TPL_VAR_PRODUCTURL}">{TPL_VAR_TITLE}</a></h3>
        <p class="image"><a
href="{TPL_VAR_PRODUCTURL}">{TPL_VAR_IMG}</a></p>
        <!-- BEGIN TPL_PDESC_BLK -->
        <p class="desc">{TPL_VAR_DESC}</p>
        <!-- END TPL_PDESC_BLK -->
        <p class="price">{TPL_VAR_CURRENCY}{TPL_VAR_PRICE}
{TPL_VAR_ONSALE}</p>
      </li>
    <!-- END TPL_PRODUCT_BLK -->
  </ul>
</div>
<!-- END TPL_MAINPRODUCT_BLK -->
```



Above code will display the products in a loop.

If you need to modify the way the above template is displayed. Then please don't delete

```
<!-- BEGIN TPL_MAINPRODUCT_BLK -->
  <!-- BEGIN TPL_PRODUCT_BLK -->
    <!-- BEGIN TPL_PDESC_BLK -->

      <!-- END TPL_PDESC_BLK -->
    <!-- END TPL_PRODUCT_BLK -->
  <!-- END TPL_MAINPRODUCT_BLK -->
```

You have to write the product related code inside

```
<!-- BEGIN TPL_MAINPRODUCT_BLK -->
```

1. {PagerBlock1} will display the paging this should be outside TPL_PRODUCT_BLK as this variable renders the paging and if you use this variable inside TPL_PRODUCT_BLK then it will be displayed for every product.

```
<!-- BEGIN TPL_PRODUCT_BLK -->
```

This code will be displayed for every product.

You can use another template variable anywhere according to your design.

e.g.

1) {TPL_VAR_PRODUCTURL} - This variable gives the path to the product page. You can use this variable for page reference **href="{TPL_VAR_PRODUCTURL}"**.

2) {TPL_VAR_TITLE} - This variable is used to display the product name (title).

3) {TPL_VAR_IMG} - This variable is used to display the product image.

```
4) <!-- BEGIN TPL_PDESC_BLK -->
```

```
  <p class="desc">{TPL_VAR_DESC}</p>
```

```
<!-- END TPL_PDESC_BLK -->
```

- This code is used to display the product description.

5) {TPL_VAR_CURRENCY}{TPL_VAR_PRICE} {TPL_VAR_ONSALE} - These three variables are used to display the price. First displays the currency, second displays the price and third variable to indicates whether the product is on Sale.

Note: All these template variables are used within

<!-- BEGIN TPL_PRODUCT_BLK --> and <!-- END TPL_PRODUCT_BLK -->

<!-- END TPL_PRODUCT_BLK -->

<!-- END TPL_MAINPRODUCT_BLK -->

/*CONTENT BLOCK */

<!-- BEGIN TPL_MAINCONTENT_BLK -->

<div id="articles">

<h2>Latest News</h2>

<!-- BEGIN TPL_CONTENT_BLK -->

<a

href="{TPL_VAR_CONTENTURL}">{TPL_VAR_TITLE}

<!-- END TPL_CONTENT_BLK -->

</div>

<!-- END TPL_MAINCONTENT_BLK -->

Above code is used to display contents (articles) associated with departments.

Please write every thing related to content (articles) inside:

<!-- BEGIN TPL_MAINCONTENT_BLK --> and <!-- END TPL_MAINCONTENT_BLK -->

<!-- BEGIN TPL_CONTENT_BLK --> and <!-- END TPL_CONTENT_BLK --> are traversing all the content associated with departments.

Template variables

4) {TPL_VAR_CONTENTURL} – Variable provides full path to the content page.

5) {TPL_VAR_TITLE} - Common variable for the image and title. If an image exists then it will display the image or the content (article) title will be displayed where this variable is used.

6)

Note: These variables are used inside

<!-- BEGIN TPL_CONTENT_BLK --> and <!-- END TPL_CONTENT_BLK -->

Please don't delete any commented line with the keywords **BEGIN/ END** or your template will not work. The best way is to create a copy of working template and modify the html according to your requirement.

iv) How to add a new product template?

[Top](#)

You have to create a new template file and upload it to

/httpdocs/modules/ecom/templates/main/product/ when a new template is uploaded to the specified location then you can apply that template from the admin section.

Now how to modify the template, we will discuss this step by step:

- 1) Copy the existing template and edit it according to your needs.
- 2) To edit the html code, we will discuss the "product.htm"

Cashmerino DK



[View Larger Image](#)

Price: £4.25

Options

Cashmerino DK

01 Black

Qty:

[Add to Basket](#)

[Add to My Wish List](#)

Product Details

Added to the cashmerino range due to popular demand this DK will quickly become a classic like the others in the Debbie Bliss stable.

Extremely versatile, as it will knit to all DK patterns, Cashmerino DK is available in an initial 20 colourways.

Already one of our most popular yarns.

Customer Reviews

Be the first to review this product

3)

Following code is used to display the product title

```
<h2 id="pageTitle">{TPL_VAR_PRODUCTTITLE}</h2>
```

All the content below is enclosed in the div with an id "content"

```
<div id="content">
```

```
<!-- product display start -->
```

```
<div id="product">
```

```
*****Display image and link to large image*****
```

```
<div class="image">
```

```
{TPL_VAR_IMAGE}
```

```
<!-- BEGIN TPL_LARGEIMG_BLK -->
```

```
<p><a href="{TPL_VAR_LARGEIMGLINK}">View Larger Image</a></p>
```

```
<!-- END TPL_LARGEIMG_BLK -->
```

```
</div>
```

```
*****
```

```
*****
```

```
*****Display product information div with class='info'
```

```
*****
```

```
<div class="info">
```

```
<form id="productDisplay" action="{TPL_VAR_CARTLINK}"
```

```
method="post" name="productDisplay">
```

```
<input type="hidden" name="mode" value="add" />
```

```
<input type="hidden" name="productid" value="{TPL_VAR_MAINID}" />
```

```
<!-- BEGIN TPL_SUPPLIERIMG_BLK -->
```

```
<p>{TPL_VAR_SUPPLIERIMAGE}</p>
```

```
<!-- END TPL_SUPPLIERIMG_BLK -->
```

```
<!-- BEGIN TPL_ONSALELBL_BLK -->
```

```
<p class="highlight">{TPL_VAR_ONSALE}</p>
```

```
<!-- END TPL_ONSALELBL_BLK -->
```

```
<p class="price">Price: {TPL_VAR_CURRENCY}{TPL_VAR_PRICEMAIN} </p>
```

```
<!-- BEGIN TPL_RRPLBL_BLK -->
```

```
<p>{TPL_VAR_RRP}</p>
```

```
<!-- END TPL_RRPLBL_BLK -->
```

```
{TPL_VAR_FREEPOSTAGE}
```

```

        {TPL_VAR_SHIPNOTES}
        *****Qty display block*****
        <!-- BEGIN TPL_QTY_BLK -->
        <p>Quantity available: {TPL_VAR_QTY}</p>
        <!-- END TPL_QTY_BLK -->

        *****
        *****Template variable to display options*****
        {TPL_VAR_MAINOPTIONS}
        *****Template variable to display choices*****
        {TPL_VAR_MAINCHOICES}
        *****Template variable to volume discount *****

        {TPL_VAR_VOLDISCOUNTS}
        Note: You can delete these template variables or can move then anywhere in the template. But to
        move the template block i.e. code that begins with the commented lines having a keyword
        BEGIN and ends with commented line with the keyword END, move the whole block from one
        place to other in the template.
        <br />
        *****To display the add to basket
        button*****
        <!-- BEGIN TPL_BASKET_BLK -->
        <p id="addBasket">
            <label for="quantity">Qty</label>
            <input class="formFieldShort" type="text" name="qty"
            id="quantity" maxlength="3" value="1" size="3" onfocus="this.select()" />
            <input class="formButton" value="Add to Basket" type="submit"
            />
        </p>
        <!-- END TPL_BASKET_BLK -->
        *****To display wish lis
        *****
        <!-- BEGIN TPL_WISHLIST_BLK -->
        <p id="addWishlist"><a href="{TPL_VAR_WISHLISTLINK}">Add to My
        Wish List</a></p>
        <!-- END TPL_WISHLIST_BLK -->
        *****End wishlist
        link*****
        </form>
        *****Enquiry button
        *****
        <!-- BEGIN TPL_ENQUIRY_BLK -->
        <form name="frmenquire" action="{TPL_VAR_ENQUIRYLINK}" method="post">
            <input type="hidden" name="id" value="{TPL_VAR_ID}" />
            <input type="submit" value="Enquire now" class="formButton" />
        </form>
        <!-- END TPL_ENQUIRY_BLK -->
        ***** Enquiry
        button*****
        </div>
    </div>
    *****End
    info*****
    *****Product
    details*****
    <!-- BEGIN TPL_PDETAILS_BLK -->
        <h2>Product Details</h2>
        {TPL_VAR_LONGDESCMAIN}
    <!-- END TPL_PDETAILS_BLK -->

```

```

*****End product
details*****
</div>
<!-- end content -->

```

Following block is used display attached products. You can edit the way attached products are displayed. Currently we are displaying the product image, title, price, description etc.

```

<!-- BEGIN TPL_MAINPRODUCT_BLK -->
<div class="products">
  <h2>May we suggest</h2>
  <ul>
    <!-- BEGIN TPL_PRODUCT_BLK -->
      <li>
        <h3><a href="{TPL_VAR_PRODUCTURL}">{TPL_VAR_TITLE}</a></h3>
        <p class="image"><a
href="{TPL_VAR_PRODUCTURL}">{TPL_VAR_IMG}</a></p>
        <!-- BEGIN TPL_PDESC_BLK -->
        <p class="desc">{TPL_VAR_DESC}</p>
        <!-- END TPL_PDESC_BLK -->
        <p class="price">{TPL_VAR_CURRENCY}{TPL_VAR_PRICE}
{TPL_VAR_ONSALE}</p>
      </li>
    <!-- END TPL_PRODUCT_BLK -->
  </ul>
</div>
<!-- END TPL_MAINPRODUCT_BLK -->

```

Following code is used to display the content (articles) attached to products. Image or title will be displayed of the attached content. i.e. if attached content has an image uploaded then the image is displayed otherwise the title will be displayed.

```

<!-- BEGIN TPL_MAINCONTENT_BLK -->
<div id="articles">
  <h2>Latest News</h2>
  <ul>
    <!-- BEGIN TPL_CONTENT_BLK -->
      <li><a
href="{TPL_VAR_CONTENTURL}"><span>{TPL_VAR_TITLE}</span></a></li>
    <!-- END TPL_CONTENT_BLK -->
  </ul>
</div>
<!-- END TPL_MAINCONTENT_BLK -->

```

Following code displays the form to add new reviews and displays added reviews.

```

<div id="reviews">
<!-- BEGIN TPL_REVIEWLINK_BLK -->
  <h2>Customer Reviews</h2>
  <ul>
    <li><a href="{TPL_VAR_REVIEWFORM}">{TPL_VAR_LBLREVIEW}</a></li>
  </ul>
<!-- END TPL_REVIEWLINK_BLK -->
  <!-- BEGIN TPL_REVIEW_BLK -->
  <p>
    {TPL_VAR_DELETE_REVIEWURL}
    Rank: {TPL_VAR_RANK}<br />
    {TPL_VAR_BY}
    {TPL_VAR_TITLE}<br />
    {TPL_VAR_COMMENT}<br />
  </p>
</div>

```

```

Submitted on: {TPL_VAR_DATE}<br />

<!-- BEGIN TPL_LINK_BLK -->
Was this review helpful? <a href="{TPL_VAR_HELPURL}">Yes</a> | <a
href="{TPL_VAR_NOHELPURL}">No</a><br />
<!-- END TPL_LINK_BLK -->
{TPL_VAR_STATUS}
</p>
<!-- END TPL_REVIEW_BLK -->
<!-- BEGIN TPL_REVIEWFORM_BLK -->
<form class="global-form" action="{TPL_VAR_REVIEWPOST}" method="post">
<input type="hidden" name="productid" value="{TPL_VAR_MAINID}" />
<input type="hidden" name="seotitle" value="{TPL_VAR_SEOTITLE}" />
<table summary="Product purchase options">
<tr>
<td class="first"><label for="title">Review Title</label></td>
<td><input class="formField" type="text" name="title" id="title"
onFocus="if(this.value=='review title')this.value='';" onBlur="if(this.value=='')this.value='review
title';" value="review title" maxlength="40" /></td>
</tr>
<tr>
<td><label for="rank">Ranking</label></td>
<td>
<select class="formField" id="rank" name="rank">
<option value="Excellent"
selected="selected">Excellent</option>
<option value="Good">Good</option>
<option value="Fair">Fair</option>
<option value="Poor">Poor</option>

<option value="Horrible">Horrible</option>
</select></td>
</tr>
<tr>
<td><label for="comment">Your Review</label></td>
<td><textarea class="formField" id="comment" name="comment"
rows="5" cols="30" onFocus="this.select()">Your review</textarea></td>
</tr>
<tr>
<td><label for="checkbox">Show my Name</label></td>
<td><input type="checkbox" class="formRadio" id="display"
name="display" value="1" /></td>
</tr>
</table>
<input type="submit" class="formButton" value="Add Review" />
</form>
<!-- END TPL_REVIEWFORM_BLK -->
</div>

```

v) How to add a new content template?

[Top](#)

You have to create new template file and upload it to **/httpdocs/modules/ecom/templates/main/content/** when the new template is uploaded at the specified location, you can apply that template from the admin section.

Now how to modify template, we will discuss this step by step:

- 1) Copy the existing template and edit it according to your needs.
- 2) To edit the html code, we will discuss the "content.htm"

3) Displays the Content title

```
<h2 id="pageTitle">{TPL_VAR_PRODUCTNAME}</h2>
```

1) Displays the Content long description

```
<div id="content">
  {TPL_VAR_LONGDESC}
</div>
```

2) Displays the Last updated date

```
<p><em>Last Updated: {TPL_VAR_DATE}</em></p>
</div>
```

3) Following code displays Associated Products with this article. This code is the same as the product template

```
<!-- BEGIN TPL_MAINPRODUCT_BLK -->
<div class="products">
<h2>May we suggest</h2>
<ul>
<!-- BEGIN TPL_PRODUCT_BLK -->
  <li>
    <h3><a href="{TPL_VAR_PRODUCTURL}">{TPL_VAR_TITLE}</a></h3>
    <p class="image"><a
href="{TPL_VAR_PRODUCTURL}">{TPL_VAR_IMG}</a></p>
    <!-- BEGIN TPL_PDESC_BLK -->
    <p class="desc">{TPL_VAR_DESC}</p>
    <!-- END TPL_PDESC_BLK -->
    <p class="price">{TPL_VAR_CURRENCY}{TPL_VAR_PRICE}
    {TPL_VAR_ONSALE}</p>
  </li>
<!-- END TPL_PRODUCT_BLK -->
</ul>
</div>
<!-- END TPL_MAINPRODUCT_BLK -->
```

4) Following code displays the associated Content. Same as for the product display template

```
<!-- BEGIN TPL_MAINCONTENT_BLK -->
<div id="articles">
  <h2>Latest News</h2>
  <ul>
    <!-- BEGIN TPL_CONTENT_BLK -->
    <li><a
href="{TPL_VAR_CONTENTURL}"><span>{TPL_VAR_TITLE}</span></a
></li>
    <!-- END TPL_CONTENT_BLK -->
  </ul>
</div>
<!-- END TPL_MAINCONTENT_BLK -->
```

vii) **How to add a new module?**

[Top](#)

A new module can be added in two ways:

- 1) Directly into the working structure (Recommended when the module is large)
- 2) As a plug-in. We can use this option to integrate a module with limited and independent functionality.

1) Directly into the working structure: To integrate directly, an understanding of the working structure is required. As discussed earlier we are following the MVC structure. This means view is handled separately from logic.

A few steps which need to be followed.

- i. Create a folder with the name of the module.

- ii. Add index.php file to set a global variable for the mode. (Please refer to existing modules).
- iii. For the module to be used in the admin section adminindex.php needs to be created.
- iv. Place the folder with a structure similar to the existing modules.
- v. In the module folder there are two files index.php & adminindex.php. As the name implies, index.php is used in the front end & adminindex.php is used in the admin section.
- vi. In this file we call the required classes according to modules. Calling a class means calling an appropriate function from that class according to the action passed in the URL. We just call the constructor of the module controller file. Then this controller file will call the appropriate functions. In this file we pass three objects to the constructor of that class.
 - 1. First is the database object. The same object will be used throughout the module.
 - 2. Second is the template object. The same object will be used for the outer template but for the inner templates a separate object is created.
 - 3. Third argument is the array of all "post and get" variables.
- vii. The entire logic is handled in the controller class. All the common objects are created in the constructor. All the actions are handled in switch cases and in cases we call the required functions from the classes inside the class's folder within the specific module. In the classes folder there are two subfolders "main" and "admin". "main" is used for the front end and "admin" is used for the admin section. In each folder there are classes for each sub module. Submodulename_inteface.php handles the view of the site. Submodulename_db.php handles insertion, deleting, updating etc. Name of the sub module will be derived from the URL or switch case.

Integrate as a plug-in:

Step1. Create a folder(folder name should be according to plugin name) in the /plugins/ folder.

Step2. Create the index.php file in the new plugin folder and include in the php file `include_once("../plugins_common.php");` Then extend plugin class to your new class eg. `class c_newpluginname extends c_commonPlugin.`

Step3. Create the newplugin.tpl.html for the html in the new folder.

Step4. Write your code in `plugins/pluginname/index.php` and include further files if needed.

Step5. When the plugin has been created, add the plugin link in plugin modules to give a direct link to your plugin.

Step6. Plugins can be accessible by this link

<http://sitename.com/plugins/pluginname/index.php>

Installing Tradingeye in a subfolder:

In order to for Tradingeye to function properly in a sub folder a few changes must be made to the following files.

First you must edit the layout HTML template. This can be found in **/modules/default/templates/main/layout/layout.htm**

Changes need to be made to the following code within layout.html

```

<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />
  <link rel="stylesheet" type="text/css" href="/subfolder/css/{TPL_VAR_CSSFILE}"
media="screen" />
  <link rel="stylesheet" type="text/css" href="="/subfolder/css/inline_editor.css"
media="screen" />
  <link rel="stylesheet" type="text/css" href="="/subfolder/css/print.css"
media="print" />
  <link rel="stylesheet" type="text/css" href="="/subfolder/css/menu.css" />
  <script language="Javascript" type="text/javascript"
src="/jscript/validations.js"></script>
  <script type="text/javascript" src="="/subfolder/jscript/menu.js"></script>

```

Changes must then be made to the CSS file image paths. The CSS files can be found in the CSS directory.

You must make changes to the following CSS files

plain.css
admin_blue.css
inline_editor.css

You must again put the subfolder directory in front of any lines that call in the graphics directory.

After this then make the following changes to the configuration.php file. This can be found at the root of the directory.

Image path needs to be set: At approx line 32 change (ABSOLUTE_IMAGE_PATH) variables from 0 to 1

Changing the .htaccess file

Just two changes required in htaccess:

1) There is a line in the .htaccess:

```
RewriteBase /
```

We just need to append the folder name to:

```
RewriteBase /Foldername/
```

2) And the second change is on last line where it says:

```
ErrorDocument 404 /index.php?action=error&mode=content
```

append the folder name to /index.php

```
ErrorDocument 404 /Foldername/index.php?action=error&mode=content
```

Changing the FCK configuration.php file

You need to change two files in FCKeditor to make the FCKeditor work when tradingeye is stored in a sub-folder. Within these two files you must append the subfolder name to the /UserFiles path. For example /subfolder/userfiles

The two files are

1. /FCKeditor/editor/filemanager/browser/default/connectors/p

hp/config.php
2. /FCKeditor/editor/filemanager/upload/php/config.php

Integrating the style sheet switcher

You must make changes to two files within Tradingeye.

Firstly change the following file.

\modules\default\classes\main\cmsContent_interface.php

Within the m_showHomePage() function after the following line

\$this->ObTpl->set_var("content_blk",""); APPROX LINE No 38

Copy the following code:

```
$this->ObTpl->set_block("TPL_VAR_CMS","TPL_VAR_THEME","theme_blk");
    $this->ObTpl->set_var("theme_blk","");
    #TO DISPLAY CSS SELECTOR
    $objHTML = new mosHTML();
    $arr = array(
    $objHTML->makeOption( 'clothing.css', 'Clothing' ),
    $objHTML->makeOption( 'jewellery.css', 'Jewellery' ),
    $objHTML->makeOption( 'business.css', 'Business' ),
    $objHTML->makeOption( 'plain.css', 'Plain' ),
    $objHTML->makeOption( 'plainfluid.css', 'Plain Fluid' )
    );

    if($this->libFunc->ifSet($this->request,"cssfile","")
    {
    $_SESSION['cssSelectedFile']=$this->m_checkCss($this-
>request['cssfile']);
    $cssFile=$_SESSION['cssSelectedFile'];
    }
    elseif($this->libFunc->ifSet($_SESSION,"cssSelectedFile","")
    {
    $cssFile=$_SESSION['cssSelectedFile'];
    }
    else
    {
    $cssFile=DEFAULT_CSS;
    }
    $selTheme = $objHTML->selectList( $arr, 'cssfile', 'class="textbox"',
    'value', 'text', $cssFile);

    $this->ObTpl->set_var("TPL_VAR_ACTION",$this->libFunc-
>m_safeUrl(SITE_URL));
    $this->ObTpl->set_var("TPL_CSS_THEME", $selTheme);
    $this->ObTpl->parse("theme_blk","TPL_VAR_THEME",true);
```

The second file to change is:

\modules\default\templates\main\home.tpl.htm

After the following line {TPL_VAR_TEXT} **APPROX LINE 4**

Copy the following code:

```
<!-- BEGIN TPL_VAR_THEME -->
<form method="post" action="{TPL_VAR_ACTION}"
name="frmTheme">{TPL_CSS_THEME}
<input type="submit" value="Change" name="change" id="change" />
</form>
<!-- END TPL_VAR_THEME -->
```

Database Structure

[Top](#)

- 1) Prefix_tbAdmin_users
- 2) Prefix_tbModules
- 3) Prefix_tbAdmin_Security
Above three tables are related
"Prefix_tbAdmin_users" table stores the admin username and encrypted password with the admin's email address for the forgot password functionality.
"Prefix_tbModules" stores names for all modules.
"Prefix_tbAdmin_Security" stores information of the modules authorizations for every user.
- 4) Prefix_tbUser_Customers
Stores information of all customers registered on the website.
- 5) Prefix_tbUser_vendors
Stores information for vendors (supplier) registered with the website.
- 6) Prefix_tbUser_contactus
Stores information posted by users in the contact us form.
- 7) Prefix_tbShop_Departments
Stores all the information related to departments.
- 8) Prefix_tbShop_Products
Stores information about products.
- 9) Prefix_tbShop_content
Stores all the data related to contents. (Articles)
- 10) Prefix_tbfusion
This table is used to build relations between two departments, department & product, department & article, product to product, article to article etc. This tables has foreign keys to other tables, identified by type for the owner & child i.e. ownertype is (department, product or article) and similarly for child records.
- 11) Prefix_tbCountry
Stores all the country names with the short name and tax associated with each country.
- 12) Prefix_tbState
Stores all the states against country the id (foreign key), shortname, tax associated with the specific state.
- 13) Prefix_tbMenuheaders
Stores menu the header name, image link, sorting, active/ not active information.
- 14) Prefix_tbMenuItems
Stores menu items against the menu header. Menu header is a foreign key to the menu headers table. This table holds menu item details like name, image link, link attributes etc.
- 15) Prefix_tbShop_Options
Stores option names, description etc.
- 16) Prefix_tbShop_OptionValues
Stores standard option items names, images etc.
For example if we create an option for colour. Then the name like colour and description (colour of shirt) will be stored in the options. In option value names of the option values are stored like "red, blue, green" etc.
- 17) Prefix_tbShop_choices
This table stores information related to custom options.

- 18) Prefix_tbShop_vdiscounts
All the information related to volume discounts are stored in this table.
- 19) Prefix_tbShop_ProductOptions
All the associations of products and standard options are stored in this table. i.e. when we associate an option with a product then unique keys for the product and option are stored in this table.
- 20) Prefix_tbShop_ProductChoices
All the associations of products and custom options are stored in this table. i.e. when we associate options with product then the unique keys for product and custom options are stored in this table.
- 21) Prefix_tbShop_ProductKits
All the package related information is stored in this table.
- 22) Prefix_tbShop_Orders
When new orders are placed the general information (not product specific) related to the order is stored in this table. i.e. billing address, shipping address, payment gateway, transaction id, order total, way of shipping etc.
- 23) Prefix_tbShop_OrderProducts
This table stores the product specific details. Like product name, description, price, taxable/non taxable etc.
- 24) Prefix_tbShop_OrderOptions
This table has information of all the standard options associated with products that are ordered.
- 25) Prefix_tbShop_OrderChoices
This table has information of all the custom options associated with products that are ordered.
- 26) Prefix_tbShop_Orderkits
This table has information of all the packages that are ordered.
- 27) Prefix_tbShop_Creditcards
This table holds information of credit cards if an order is placed using an onsite gateway. But for Protx this information is not stored.
- 28) Prefix_tbShop_OrderShip
This table stores shipping information when an order is approved and dispatched by the admin.
- 29) Prefix_tbCompanySettings
This table holds all the information related to the company i.e. company name, slogan, address, vat number etc.
- 30) Prefix_tbsettings
This is the most important table of the site. In this table all the settings related to the website are saved. You can modify siteurl, sitepath directly from this table.
- 31) Prefix_tbPostageMethods
This table stores the name of postage methods, base rates etc.
- 32) Prefix_tbPostageMethodDetails
This table stores various options available under the postage methods. Like various ranges, postage codes, special methods.
- 33) Prefix_tbPlugin_apps
This table holds all the details of the plug-ins i.e directory name, help file name, plugin name, authorization for plugin.
- 34) Prefix_tbShop_discounts
This table stores all the discounts available.
- 35) Prefix_tbShop_giftcerts
This table stores all the gift certificates available.
- 36) Prefix_tbShop_giftwrap
This table stores all the gift wrap options available.
- 37) Prefix_tbShop_Promotions
This table stores all the promotional discounts like free postage, flat discount etc.

Following tables are used in the email campaigner.

- 38) Prefix_tbMailroom
- 39) Prefix_tbMailroomLead

- 40) Prefix_tbMailroomLeadList
- 41) Prefix_tbMailroomLeadProducts
 - "Prefix_tbMailroom" stores all the composed emails.
 - "Prefix_tbMailroomLead" This table holds information of leads. i.e. name, create date, modified date etc.
 - "Prefix_tbMailroomLeadList" stores leads and customer associations.
 - "Prefix_tbMailroomLeadProducts" stores leads and product associations.
- 42) Prefix_tbShop_Wishlist
 - Stores wish list data i.e. product and customer association
- 43) Prefix_tbWish_emails
 - Stores email addresses added by the customer to send their wish list.
 - Following tables are used for product reviews
- 44) Prefix_tbUser_CustomerReviews
- 45) Prefix_tbUser_CustomerReviewsHelpful
- 46) Prefix_tbUser_CustomerRatings
 - "Prefix_tbUser_CustomerReviews" stores reviews added by customers.
 - "Prefix_tbUser_CustomerReviewsHelpful" stores information whether the review added by the customer is helpful to others or not.
 - "Prefix_tbUser_CustomerRatings" stores customer ratings for reviews.
- 47) Prefix_tb_frooglesettings
 - This table stores FTP details for froogle.
 - Following are temporary tables to store information of the shopping cart stored against the sessionid.
- 48) Prefix_temp_cart
 - Stores information of products added into the shopping cart.
- 49) Prefix_temp_options
 - Stores standard options associated with the products in the shopping cart.
- 50) Prefix_temp_choices
 - Stores custom options associated with products in the shopping cart.
- 51) Prefix_temp_images
 - Stores images associated with the products in shopping cart..

7) **Programmatic Flow:**

[Top](#)

On the root there are two main files which initiates the way the programming is handled in tradingeye.
The index.php and adminindex.php are the root files for front end and admin side respectively.

In index.php file the line:
`include_once(MODULES_PATH."index.php");`
 allocates index.php file for the "MODULES_PATH" .

For the very first time it includes ../modules.
So, the file "modules/index.php" file is called.

Further in modules/index.php
i> In this file the sessions are started to be maintained further.

ii> All the important files are included here.
`include_once(MODULES_PATH."default/prevNext.php");`
`include_once MODULES_PATH."user/user_controller.php";`
`include_once MODULES_PATH."ecom/ecom_controller.php";`
`include_once(MODULES_PATH."default/commonFunctions.php");`
`include_once(MODULES_PATH."default/homeDisplay.php");`
`include_once(MODULES_PATH."default/leftmenu.php");`

iii> The "\$sModule" is a global variable used in the coding to identify the module. At the very first time, the default case is executed and the class c_homeDisplay is instantiated as listed below.

```
global $sModule;
switch($sModule)
{
    case "ecom":
        $obEcomAdmin=new c_ecomController($obDatabase,&$obMainTemplate,$attributes);
        break;
    case "user":
        $obUserAdmin=new c_userController($obDatabase,&$obMainTemplate,$attributes);
        break;
    default:
        $obHomeDisplay=new c_homeDisplay($obDatabase,&$obMainTemplate,$attributes);
        break;
}
```

So, the class : "c_homeDisplay" is instantiated and the constructor function is called from this class in the included file 'include_once(MODULES_PATH."default/homeDisplay.php");'.

Now, in file modules/default/homeDisplay.php the constructor function is:

```
function c_homeDisplay($obDatabase,$obTemplate,$attributes)
{
    $this->obDb=$obDatabase;
    $this->obTpl=&$obTemplate;
    $this->request=$attributes;
    $this->templatePath=MODULES_PATH."default/templates/main/";
    $this->printMainBlock();
}
```

All the required variables are assigned here ie:
\$this->obDb, \$this->obTpl, \$this->request, \$this->templatePath
and the function "\$this->printMainBlock()" is called from the class.

Now you will notice in function printMainBlock() the following code:

```
if(!isset($this->request['action']))
{
    $this->request['action']="";
}
```

"\$this->request" contains the list of all the post,get,request variable as passed from the module/index.php file. "\$this->request['action']" contains all the get variables as passed in the URL.

We now just to switch the cases based on the value of the variables passed form the URL.

For the very first time the default: case is managed.

ie approximately as code listed below:

```
default :
if(isset($this->request['sid']) && !empty($this->request['sid']))
{
    $value=$this->request['sid'];
    setcookie("sourceid", $value, time()+3600, "/");
}
$obCms->cmsTemplate = $this->templatePath."home.tpl.htm";
$this->obTpl->set_var("TPL_VAR_BREDCRUMBS","");
```

```
$this->obTpl->set_var("SiteUrl",SITE_URL);  
$this->obTpl->set_var('TPL_VAR_BODY', $obCms->m_showHomePage());  
break;
```

In this case the template file (.tpl file) called is "templatePath."home.tpl.htm"" as listed in the code: `$obCms->cmsTemplate = $this->templatePath."home.tpl.htm";` Whereas the php function responsible to display the output for tpl file is "m_showHomePage()" as listed in the code line: `$this->obTpl->set_var('TPL_VAR_BODY', $obCms->m_showHomePage());`

Depending on the objects created and the functions called, the functionality flows from one file to another and so on.

The main work is basically done through:

- i> All the variables which are to be used in the .tpl files are set using `set_var()`.
- ii> The blocks are initialized using `set_block()`.
- iii> The files are set using `set_file()`.
- iv> The blocks are parsed using `parse()`.
- v> The `pparse()` flushes the entire data and returns the complete output to the screen.

Same is the case with `adminindex.php` file used at the admin side.

The whole functionality is in the php files, the variables, blocks to be displayed are set in the php files, parsing is also done here. Only these variables and blocks set here are used in the template files (.tpl files) for displaying it to the screen giving it a complete MVC structure (separating the html from the logic and having a single controller, which refers to which functions of php are to be called and the output will be assigned to which template files).