Internet Business
Applications & Services
94 El. Venizelou Av., 17671 Kallithea, Athens, Greece
Tel.: +30 210 9565421, +30 210 9589311, Fax: +30 210 9589647
Email: info@goup.gr • URL: www.goup.gr

Author: Ilias Antonopoulos (ia@goup.gr)
Date: 28 May 2007
Version: 1.0

Elxis CMS (2006.4) Templates Design Reference

This guide aims to present the key parts of templates design for Elxis CMS. Elxis CMS has three different kinds of templates.

Site templates. The templates used to render the site itself. Is what visitors view.
Administrator Templates. The templates used to render administration side.
Login Screens. The templates used to render the administration side login screen.

In this document we will focus only to Site Templates.

Compatibility between templates for Mambo/Joomla and Elxis

Generally templates written for Mambo/Joomla, are **compatible** with Elxis CMS. That is because Elxis and Joomla are forks of Mambo.

The **key differences** and something that you have to take into account when you design or adapt templates to Elxis is that Mambo/Joomla templates do not have the **"language" module position**, so you have to add it manually if you want the language selector to be visible.
Note: It is advisable to place the language module position inside a div container identified by id, in order to have finer control on its appearance.

Alternatively you can move the Language Selection module from the language module position to a module position that exists in the template you are using. Mambo and Joomla are not multilingual by default so they did not need this module position.

Elxis is a true multilingual CMS using natively the **UTF-8 character set**. Make sure that the template you use support this encoding.

Some advanced templates for Mambo/Joomla, use **custom PHP code** that queries the database. You have to make sure to rewrite the code in order to be compatible with Elxis CMS advanced features.

Template Structure

To normal users, a template is just a package file, usually a .zip file that they use to install the new template on their Elxis site. This .zip file contains all the files and directories that form the template.

You can install a new site template from the administration section, using one of the following paths:
Site -> Template Manager -> Site Templates -> Click New icon
Site -> Template Manager -> Install (under Site Templates)
Installers -> Templates - Site

If after installation, you check the file system of your Elxis site, you will find that under the /templates directory, appeared a new directory that has the name of the newly installed template, i.e. /templates/mynewtemplate.

If you go into the directory you will see the files and directories that form a typical Elxis template. These are:

templateDetails.xml: This file contains instructions to the templates installer and uninstaller regarding the files and directories forming the template.
index.php: Is the key file of the template. Is responsible for defining the structure of the template and module positions.
template_thumbnail.png: Is a small color image of type .png 8bit, that shows a screen shot of the template. It is used during preview or from the template chooser module.
/css/template_css.css: Inside the css directory there is the template_css.css file. This file holds all CSS classes used to render the template.
/images: Images directory holds all the image files used by the template.

These were the files forming a typical template. In advanced templates you may encounter more files and even directories.

In the next sections we will describe in detail the key elements of each file.

templateDetails.xml

This is an XML file describing the files that form the template. It is used from templates manager to display information regarding this template and from templates installer and uninstaller to install or uninstall this template.

templateDetails.xml code lines	Comments
<?xml version="1.0"?>	
<mosinstall type="template" version="2006">	Inform installer that this is a template for version 2006.x of Elxis CMS
<name>templateName</name>	Specify template's name. This name is going also to be used in index.php while forming paths to CSS file and images.
<creationDate>YYYY-MM-DD HH:II:SS</creationDate>	Specify the creation date of the template.
<author>authorName</author>	Specify the name of the author.
<copyright>license</copyright>	Specify the license under which this template is released. It can be GNU/GPL or other.
<authorEmail>authorEmail</authorEmail>	Specify author's email.
<authorUrl>authorUrl</authorUrl>	Specify author's web site URL, i.e. http://www.elxis.org
<version>1.0</version>	Specify the version of this template.
<description>templateDescription</description>	Give a short description for your template.
<files>	Start of files section.
<filename>index.php</filename>	Copy index.php to the root folder of the template, i.e. /templates/templateName
<filename>template_thumbnail.png</filename>	Copy template_thumbnail.png to the root folder of the template, i.e. /templates/templateName
<filename>css/index.html</filename>	Copy index.html file at /templates/templateName/css directory. Index.html is a blank html file. It is used to prohibit users for displaying the contents of this directory.
<filename>images/index.html</filename>	Copy index.html file at /templates/templateName/images directory. Index.html is a blank html file. It is used to prohibit users for displaying the contents of this directory.
. . .	Add here other files that you want do distribute.
</files>	End of files section
<images>	Start of images section. This section lists all images used by the template.
<filename>images/imageFilename.type</filename>	Please note the way image files are described. Note the use of "images/" preceding the file name.
. . .	Add here more image files.
</images>	End of images section
<css>	Start of CSS files section
<filename>css/template_css.css</filename>	Please note the way css files are described. Note the use of "css/" preceding the file name.
. . .	Add here more css files if they exist.
</css>	End of CSS files section
</mosinstall>	

Index.php

We start describing the key elements of index.php file. Index.php file is divided in three main parts. The pre-header section, the header section, and the body section. If you we would like to oversimplify things

we could say that is a typical xhtml file in which we have added extra PHP commands to convert it to a template file for Elxis CMS. Do not take this explanation as granted. If you are new to templates design, just use it as a guide.

Pre-Header section

Index.php code lines	Comments
`<?php`	PHP code opening tag
`defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );`	Do not allow the execution of this code if it is not called from Elxis CMS
`$iso = explode( '=', _ISO );`	Separate the ISO number from the language file constant _ISO
`echo '<?xml version="1.0" encoding="'. $iso[1] .'"?' .'>';`	XML Prolog
`?>`	PHP code closing tag
`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`	Public Text Identifier. appear at the very beginning of an HTML/XHTML document in order to identify the content of the document as conforming (theoretically) to a particular HTML DTD specification. Click here for information regarding DOCTYPE
`<html xmlns="http://www.w3.org/1999/xhtml">`	xmlns declaration for the XHTML namespace

<Head> Section

The head section is common among html and xhtml documents. In Elxis CMS template we use it to use or not use the editor, to define the documents codepage and to load the required css files.

Index.php code lines	Comments
`<head>`	Start of head section
`<?php mosShowHead(); ?>`	Calls the mosShowHead() function. During execution, generates the following tags: <title> <meta name="description" <meta name="keywords" <meta name="Generator" <meta name="robots" <link rel="shortcut icon"
`<?php if ( $my->id ) { initEditor(); } ?>`	Loads the WYSIWYG editor only when a user is logged in.
`<meta http-equiv="Content-Type" content="text/html; <?php echo _ISO; ?>" />`	Defines the value of the codepage used by this template.
`<link href="<?php echo $mosConfig_live_site;?> /templates/[template]/css/template_css.css" rel="stylesheet" type="text/css"/>`	Loads the CSS definitions store in the template_css.css file. [template] is replaced by the name of the template. Please note the use of `$mosConfig_live_site` variable. During execution is replaced by site's URL.
`<head>`	End of head section

Body Section

Inside the body section we **describe the layout** of our template and we **define the module positions** used by our template. The final look and feel of our template is governed by the layout described here, the rendering options defined in the template_css.css file, and the module positions used.
For example, we could define a   container like this: . The defines the skeleton. But how this skeleton will look like, will be governed by the #topmenu {} class defined in template_css.css file.

If the contained a module position like in the following example, then the way the modules displayed in this module position ("top" in our example) will look like, is governed by the CSS classes that control modules appearance, also defined in template_css.css file.


```
<div id="topmenu">
      <?php mosLoadModules('top',-1); ?>
</div>
```

In this guide we will not focus on how to design the skeleton of your template. This is something that has no difference with any other html design. You can use a plain text editor to do so or use more advanced applications like Macromedia Dreamweaver or even Adobe Photoshop. It is up to you to decide what is best for you.

In this guide we will focus to the differences that transform a normal page to a template for Elxis CMS. The key difference is that all content displayed in our web site is coming from Elxis. Our template should be prepared for that and define the appropriate placeholders, the places the content will be displayed. Our template has also to control how this content will be rendered. What font will be used, what size, what color, what will be the distances, the background images, etc.

Template PHP functions

We use the following PHP functions inside our templates. Parameters inside [ ] mean that they are optional.

mosLoadModules

Syntax: mosLoadModules($position_name [,style])
Example: <?php mosLoadModules ("left", 0); ?>
Description: Use this function to define module positions. During execution it will be replaced by the modules that you have assigned to the module position defined. Style parameter defines the method used to render modules. Possible values for "style" are:

0. Normal behavior (default)
1. Show modules horizontally
-1. Do not process module output. Module title is hidden
-2. Display using XHTML
-3. Display using extra divs. Used to display modules with curved corners.

Note: The final output, especially when a module is used to display a menu, is also affected by the "Menu Style" parameter of the module. This parameter is defined at: Modules -> Site Modules -> click a module title and scroll down to parameters.

mosShowHead

Syntax: mosShowHead()
Example: <?php mosShowHead(); ?>
Description: This function is placed inside the <head></head> section of the template. During execution it is replaced with:

```
<meta name="description" ...
<meta name="keywords" ...
<meta name="Generator" ...
<meta name="robots" ...
<link rel="shortcut icon" ...
```

mosCountModules

Syntax: mosCountModules($position_name)
Example:

```
<?php if ( mosCountModules("left") ) { ?>
        //HTML tags
        mosLoadModules("left");
        // HTML tags
<?php } ?>
```

Description: Returns the number of modules that have been assigned to the $position_name. It is used primarily by smart templates that display or not a part of the template. For example, it can be used to implement logic as: Display the right column only if there are modules assigned to the right module position.

mosMainBody

Syntax: mosMainBody()
Example: <?php mosMainBody(); ?>
Description: Wherever you place this function, it will be replaced by the main content of Elxis. It can be used only once.

mosLoadComponents

Syntax: mosLoadComponents($name)
Example: <?php mosLoadComponents("banners"); ?>

Description: You can use this function to display the output of a component. It is not recommended to use it, unless you know very well what you are doing.

mosPathWay

Syntax: mosPathWay()
Example: <?php mosPathWay(); ?>
Description: Place this function where you want to show Elxis pathway. Pathway displays the “You are here” feature.

Display Footer

If you want to display Elxis CMS footer (recommended), place the following code in your template:
<?php include_once($GLOBALS['mosConfig_absolute_path'] . '/includes/footer.php'); ?>

Debug Info

Although it is a special use of mosLoadModules function, we decided to reference it separately. You can use it if you want to display debug info when the relevant option is activated from Global Configuration.
<?php mosLoadModules('debug',0); ?>

CSS Classes

In this section, we will talk about the CSS classes defined by the Elxis. In your template_css.css file, you should provide declarations for all these, if you want your template to render properly. Your template_css.css file should also provide declarations for all custom CSS classes defined in your template.

BODY
Controls the body of your template. Used also to define the background of your page. As the same parameters are used by the embedded WYSIWYG editor of Elxis to render its background, if you define a dark colored background, you will see it also in the editor. You can override that by changing editor's parameters.

H1 – H7
The normal H1 to H7 headers of HTML

TD, P
As most of the text in Elxis 2006 versions are displayed inside td or p elements, actually control the appearance of your text.

UL & LI & OL
Define how ordered and unordered lists will render.

.pathway
.pathway a:link, .pathway a:visited, .pathway a:hover
Control the appearance of the pathway. See Figure css-01.

Figure css-01

.blog (table.blog)
Controls the table that surrounds content in blog view. See red outline in Figure css-02.

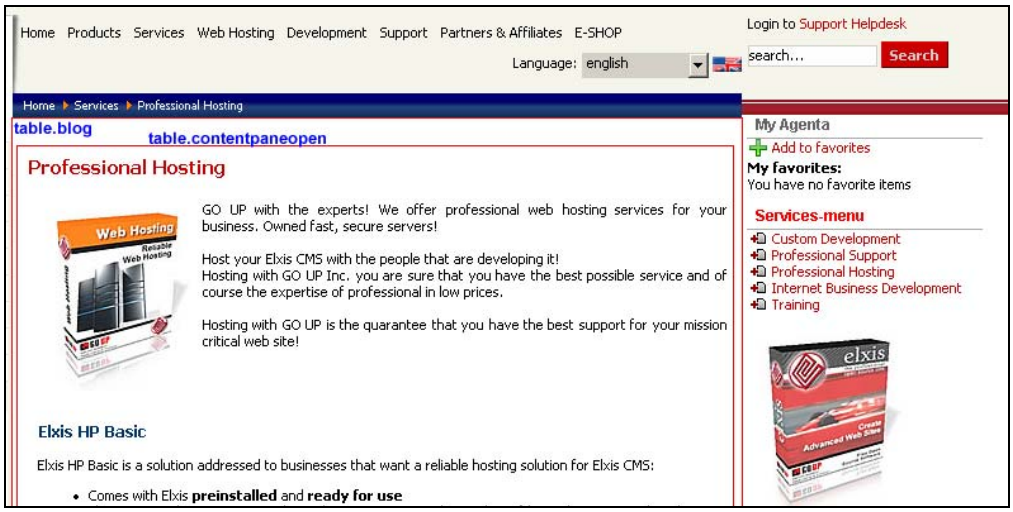

Figure css-02

.blog .contentpaneopen .contentheading

Controls the appearance of the title of an article when it is the only article displayed. For example, when we have clicked the "Read more" link or we have clicked article's title and we are viewing it in detail. See Figure css-03.

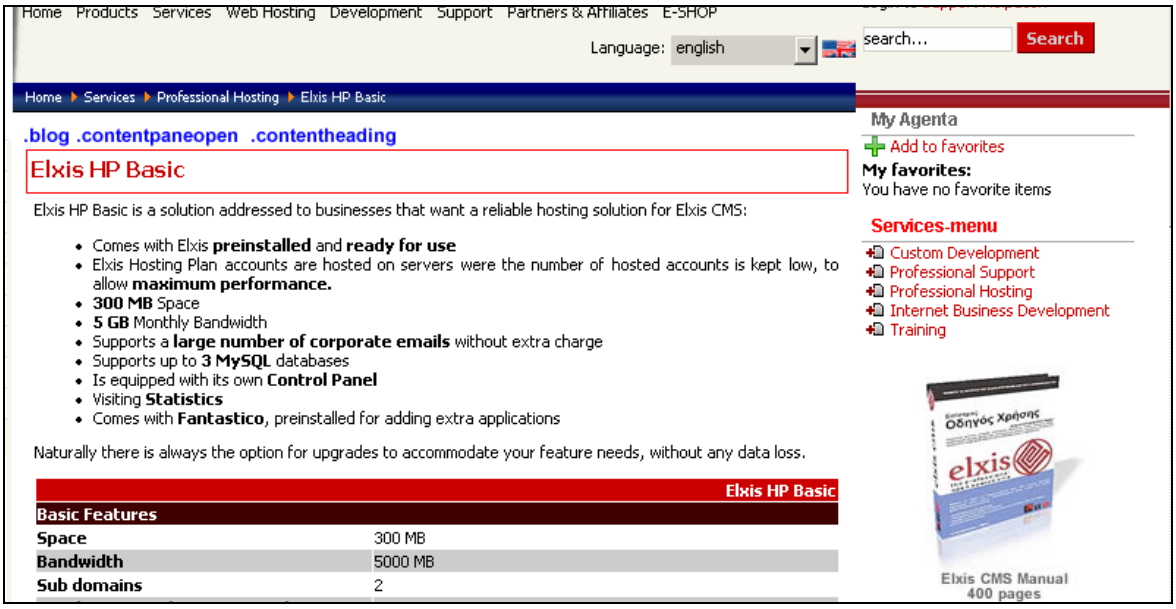

Figure css-03

.contentheading

Controls the appearance of the <td> that holds content item titles in blog view. See Figure css-04.

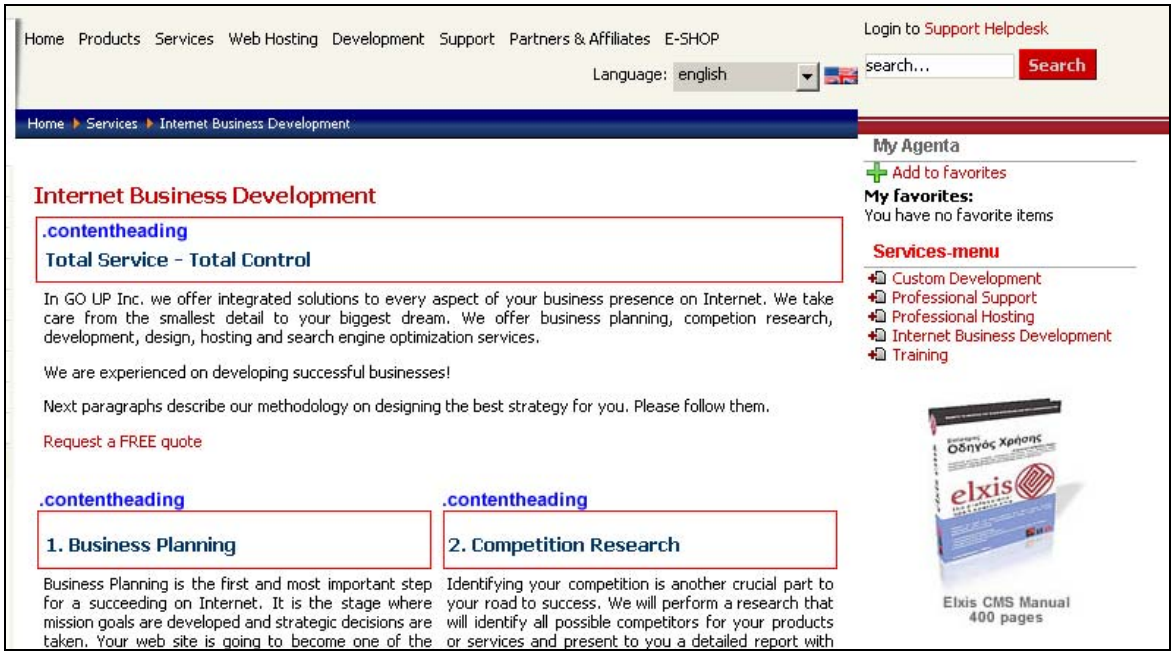

Figure css-04

a.contentpagetitle:link, a.contentpagetitle:visited, a.contentpagetitle:hover

From Global Configuration, you can configure content item titles to behave as links. You can click the title and read the full article. This class controls the behavior of linked titles. See Figure css-05

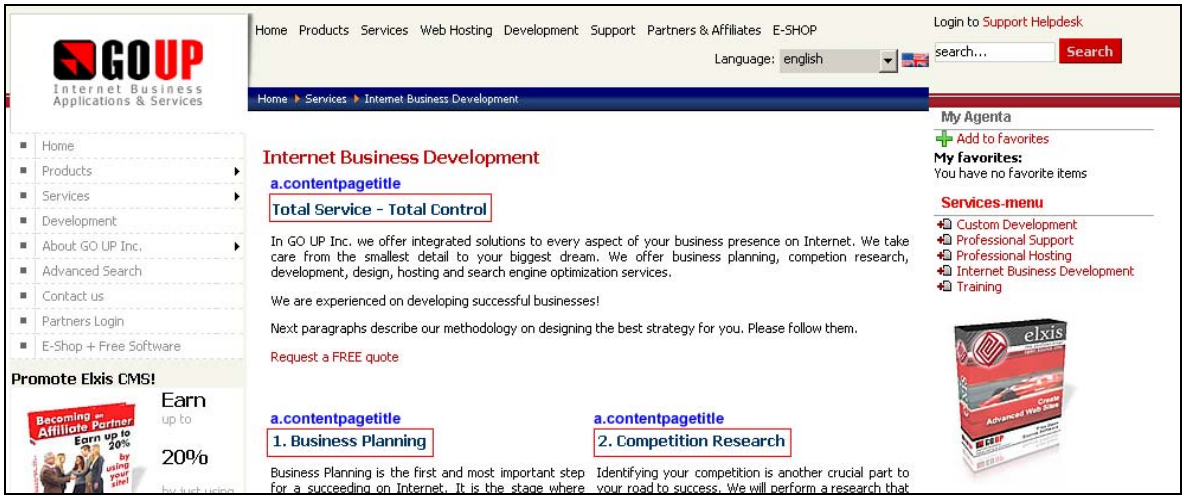

Figure css-05

table.contentpaneopen
Is a table that immediately follows table.blog (Figure css-02).

td.contentpaneopen_text
Controls the <td> that surrounds every content item or autonomous page displayed in Elxis CMS. See Figure css-06.

Figure css-06

.componentheading (div.componentheading)
Controls the appearance of component's title. You will see this title every time you click a menu item of type "Component". See Figure css-07.

Figure css-07

.contentdescription (td.contentdescription)
Controls the appearance of the category description when viewing content in table view. Table view is used when you have clicked a menu item of type: "Table - Content Category" and you have selected the category description text to be visible. See Figure css-08.

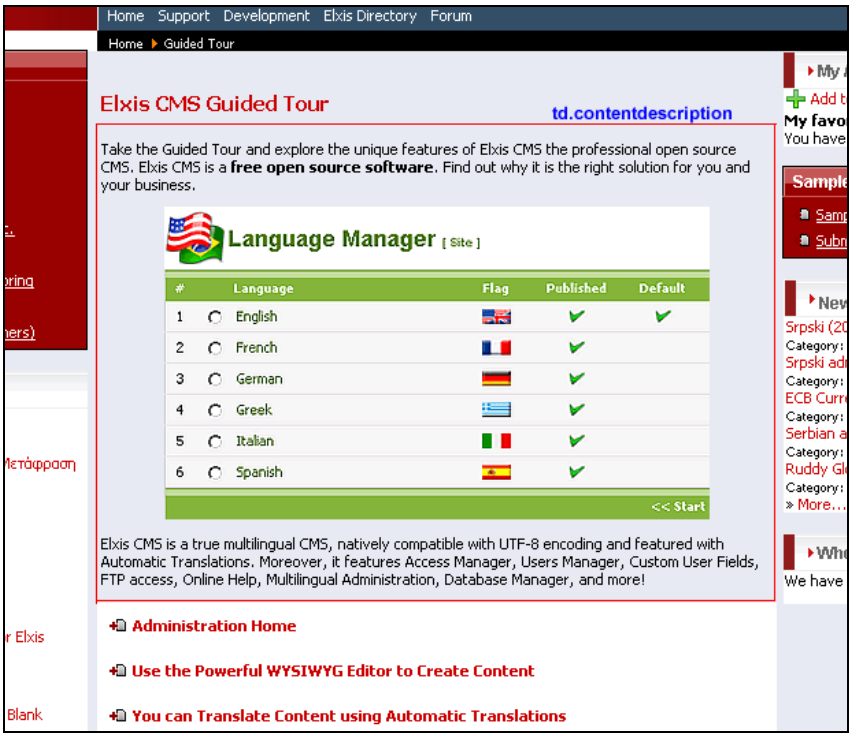

Figure css-08

a.readon:link, a.readon:visited, a.readon:hover
Controls the appearance of “Read more…” link. See Figure css-09.

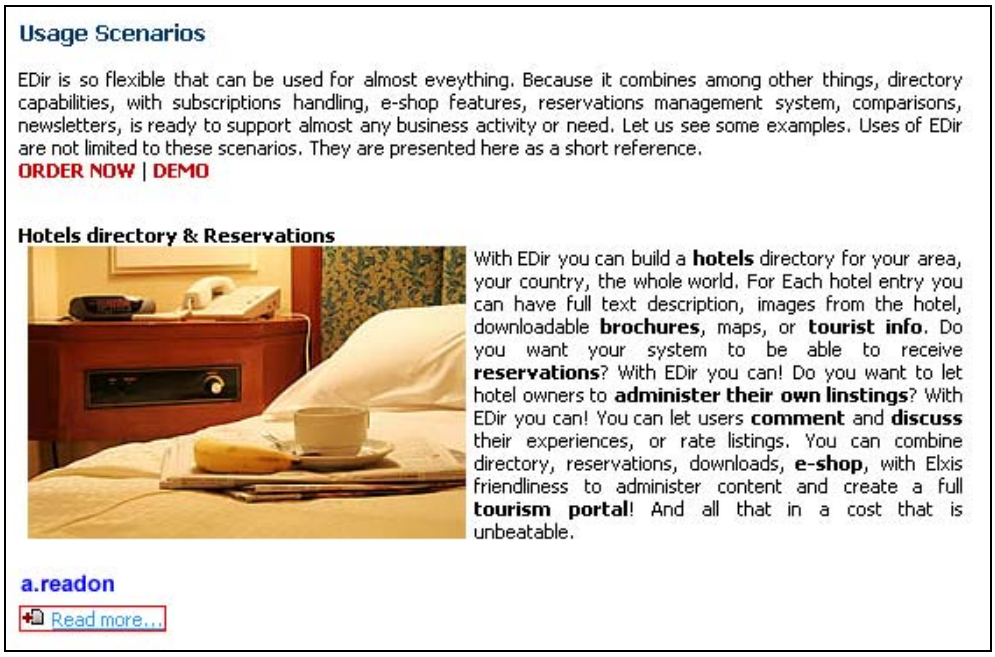

Figure css-09

A:link, A:visited, A:hover
Controls the behavior of all links used in your site, unless they are overridden by a special declaration, like the a:readon presented before.

.buttonheading (.buttonheading)
Controls the area that print, pdf and mail buttons are displayed. See Figure css-10.

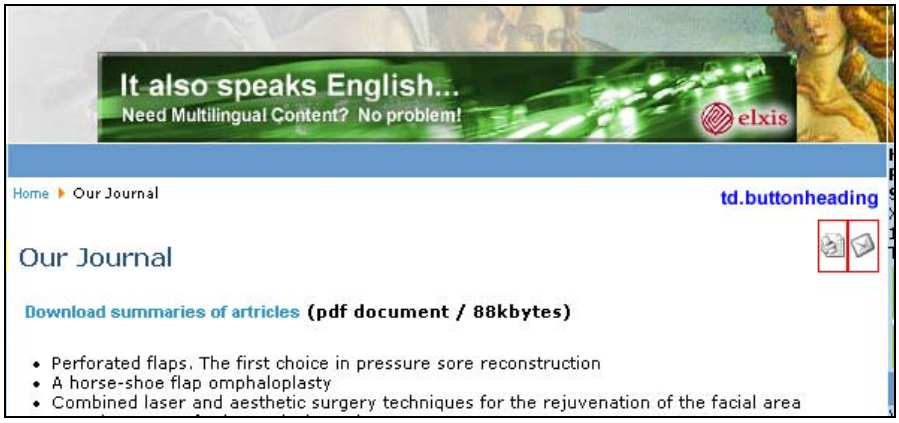

Figure css-10

.inputbox & .button
Controls the default appearance of all .inputbox and .button elements used in forms. Default appearance can be overridden by wrapping special forms, like language selection, inside container elements like . In these cases you can provide special classes.

#emailForm & #emailForm .inputbox & #emailForm textarea.inputbox
These definitions override the default behavior of forms, for the Contact us default form. You can use them to differentiate the appearance of Contact us form.

table.contenttoc
table.contenttoc th
table.contenttoc td
a.toclink:link, a.toclink:visited, a.toclink:hover
When we break an article in smaller sections using the {mospagebreak} mambot, appears a Table of Contents (TOC) table. These classes control its appearance. See Figure css-11.

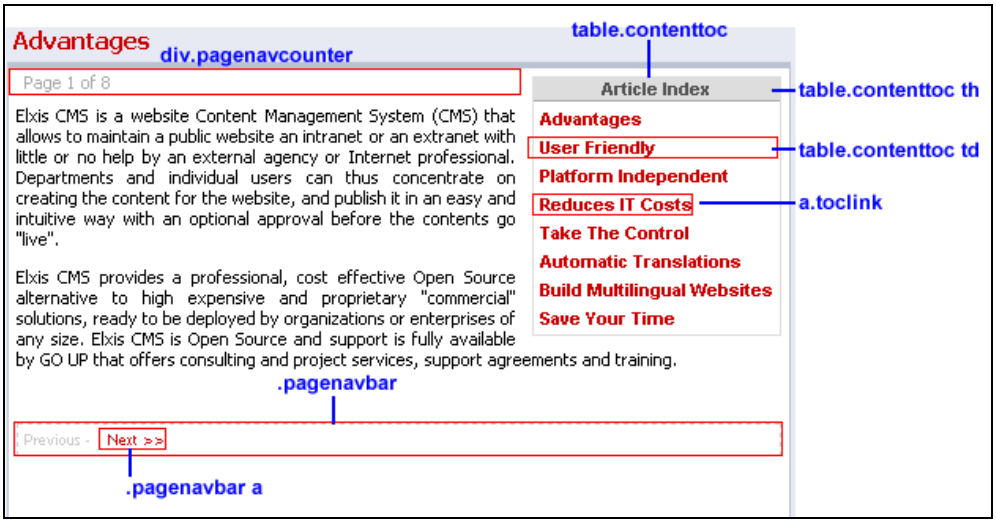

Figure css-11

.pagenavcounter
See Figure css-11.

.pagenavbar (div)
.pagenavbar a:link, .pagenavbar a:visited, .pagenavbar a:hover
See Figure css-11.

.small
Is used in several places where auxiliary information is presented, like author.

.createdate & .modifydate
As their names imply, they are used to render creation and modification dates for articles.

.back_button (div)
.back_button a:link, .back_button a:visited, .back_button a:hover
Control the appearance of the area the [back] link is displayed and the link itself. See Figure css-12.

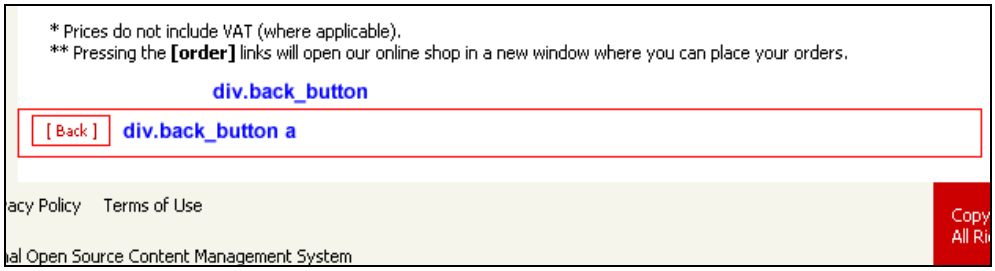

Figure css-12

table.moduletable
table.moduletable th
table.moduletable td
Control the appearance of modules. **table.moduletable** controls the whole table, **table.moduletable th** the module title, and **table.moduletable td** the menu item cell. See Figure css-13. Can be

overridden by specifying in module parameters the “Module Class Suffix” parameter. If for example you set for a module this parameter to the value of “-custom”, then to implement it you have to provide CSS classes for **table.moduletable-custom**, **table.moduletable-custom th**, **table.moduletable-custom td**.

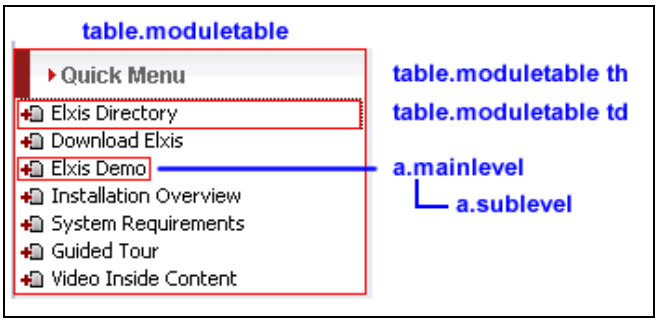

Figure css-13

a.mainlevel:link, a.mainlevel:visited, a.mainlevel:hover
a.sublevel:link, a.sublevel:visited, a.sublevel:hover
As you can see in Figure css-13, they control the appearance of menu item links. You can override them by specifying in module parameters the “Menu Class Suffix” parameter. If for example you set for a module this parameter to the value of “-m”, then to implement it you have to provide CSS classes for:
a.mainlevel-m:link, a.mainlevel-m:visited, a.mainlevel-m:hover
a.sublevel-m:link, a.sublevel-m:visited, a.sublevel-m:hover

ul.latestnews
li.latestnews
a.latestnews:link, a.latestnews:visited, a.latestnews:hover
ul.mostread
li.mostread
a.mostread:link, a.mostread:visited, a.mostread:hover
Latest News and **Popular** are two special modules. They display the latest published news and the most popular articles. The list of displayed articles is implemented as unordered lists. This classes control their appearance.

IMG
Controls the appearance of images (<img>) all over the site.

.mosimage
Control the appearance of images inserted using the {mosimage} mambot.

.mosimage_caption
Control the appearance of the caption of images inserted using the {mosimage} mambot.

a.pagenav:link, a.pagenav:visited, a.pagenav:hover
td.sectiontableheader
td.sectiontablefooter
.sectiontableentry1 td
.sectiontableentry2 td
.sectiontableentry1 a:link, .sectiontableentry1 a:visited, .sectiontableentry1 a:hover
.sectiontableentry2 a:link, .sectiontableentry2 a:visited, .sectiontableentry2 a:hover
Control the appearance of elements in a “Table – Content Category” view. See Figure css-14.

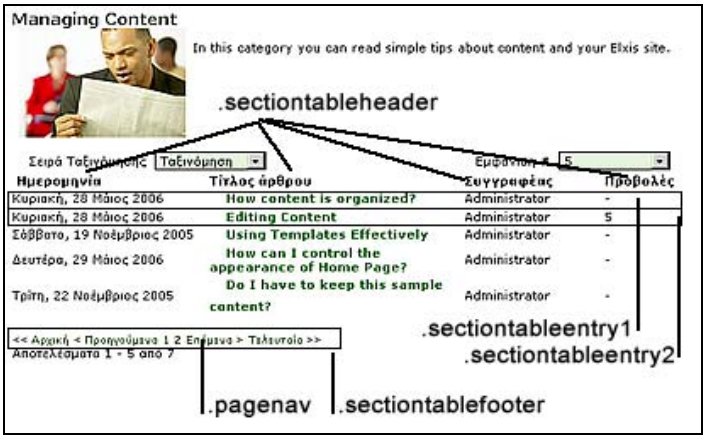

Figure css-14

a.category:link, a.category:visited, a.category:hover
Control the behavior of category links in "Table – Content Section" view. See Figure Figure css-15.

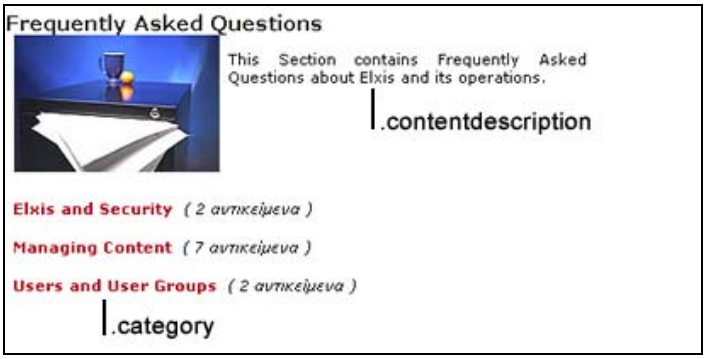

Figure css-15

table.contentpane ul
table.contentpane ul li
a.category
Control the appearance of output from web links component, affecting the defined web link categories. See Figure css-16.

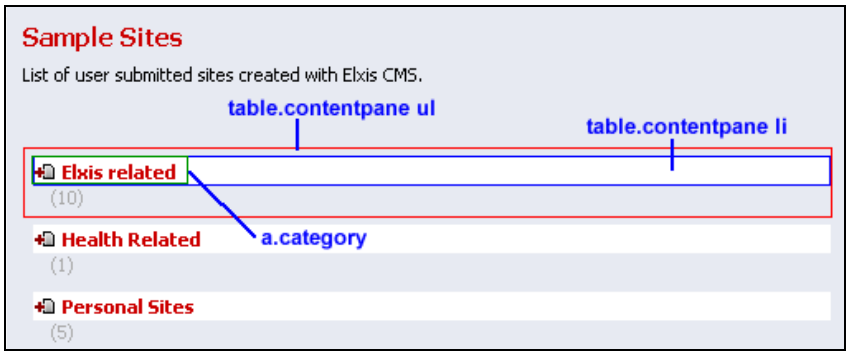

Figure css-16

Note: Some class definitions may be missing.

-- END --

