

Parallels[®] Plesk Sitebuilder

Developer's Guide

Plesk Sitebuilder Custom Templates

Plesk Sitebuilder 4.5

Copyright Notice

ISBN: N/A

Parallels

660 SW 39th Street

Suite 205

Renton, Washington 98057

USA

Phone: +1 (425) 282 6400

Fax: +1 (425) 282 6444

© Copyright 1999-2008,

Parallels, Inc.

All rights reserved

Distribution of this work or derivative of this work in any form is prohibited unless prior written permission is obtained from the copyright holder.

Product and service names mentioned herein are the trademarks of their respective owners.

Contents

Preface	5
About This Guide	5
Who Should Read This Guide	5
Typographical Conventions	6
Feedback	7
Templates Overview	8
How to Create Template	9
Creating Template Design	10
Overview of Template Components	11
Header	12
Top-Level Menu	13
Lower-Level Menu	14
Content Area	15
Footer	16
Requirements to Templates Design	16
Creating Template Logical Structure	17
Before Creating Template Source Code	19
Requirements to HTML Code	20
Sitebuilder Controls and Variables	21
Previewing Templates	22
Converting HTML Code to Template Source Code	23
Adding Images to Template	24
Creating Master.page File	24
Master.page Example	26
Creating Customizable Content	29
Creating Menu Style	30
Adding Header Banners	33
Creating Thumbnails for Templates, Menu Styles, and Headers	34
Creating Thumbnails for Templates	34
Creating Thumbnails for Menu Styles	35
Creating Thumbnails for Header Banners	35
Specifying General Information About Template	36

Compiling Template	39
---------------------------	-----------

Installing Template	41
----------------------------	-----------

Changing Appearance of Sitebuilder Modules	42
---------------------------------------------------	-----------

Customizing Styles Used in Modules	43
"Categories" Group	44
"Items" Group	47
"Comments" Group	53
"Search" Group	56
"Form" Group	58
"Information" Group	60
"Paging" Group	62
"Status" Group	63
"Main" Group	65
Customizing Dynamic Content of Modules	67
StatusBar Control	67
Paging Control	69
Form Control	71
Customizing Parameters Specific for Blog Module	73
Customizing Parameters Specific for Guestbook Module	82

Appendix I. Controls Reference	87
---------------------------------------	-----------

Container	87
TextDiv	87
List	88
Link	90
TextInput	90
ValidationText	91
Button	91

Preface

In this section:

About This Guide.....	5
Who Should Read This Guide	5
Typographical Conventions	6
Feedback	7

About This Guide

This document provides you with the guidelines for creating your own Sitebuilder *design templates* and installing them to Sitebuilder.

Who Should Read This Guide

This guide is intended for the following users:

- Sitebuilder vendors who want to add custom Sitebuilder templates to their templates pool, or customize the existing set of templates.
- Web-designers who wish to distribute their own Sitebuilder templates.

This guide requires knowledge of Cascading Style Sheets (CSS) and Hypertext Markup Language (HTML), and understanding of how content templates work.

Typographical Conventions

Before you start using this guide, it is important to understand the documentation conventions used in it.

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information	Example
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Go to the System tab.
	Titles of chapters, sections, and subsections.	Read the Basic Administration chapter.
<i>Italics</i>	Used to emphasize the importance of a point, to introduce a term or to designate a command line placeholder, which is to be replaced with a real name or value.	The system supports the so called <i>wildcard character</i> search.
Monospace	The names of commands, files, and directories.	The license file is located in the <code>http://docs/common/licenses</code> directory.
Preformatted	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	<pre># ls -al /files total 14470</pre>
Preformatted Bold	What you type, contrasted with on-screen computer output.	<pre># cd /root/rpms/php</pre>
CAPITALS	Names of keys on the keyboard.	SHIFT, CTRL, ALT
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another.	CTRL+P, ALT+F4

Feedback

If you have found a mistake in this guide, or if you have suggestions or ideas on how to improve this guide, please send your feedback using the online form at <http://www.parallels.com/en/support/usersdoc/>. Please include in your report the guide's title, chapter and section titles, and the fragment of text in which you have found an error.

Templates Overview

In Sitebuilder, a *design template* (hereafter called simply *template*) is an installable file that dictates the overall look and feel of a site. Later on, the site can be filled with any content, provided by Sitebuilder site owners. A site template is compiled from *template source code* - files that contain resources (stylesheet files, images, etc.) and instructions necessary for installing the template to an existing Sitebuilder instance. Source code of one or more templates can be compiled into *template pack* - a package that contains one or more templates. It is considered that a single template is also a template pack.

To create custom template packs, you need to install Sitebuilder SDK. It contains tools for compiling template packs and samples of template source code. Sitebuilder SDK is available at the following URL: <http://swdn.swsoft.com/en/download/sdk/>.

There are two versions of Sitebuilder templates: 3 and 4 (depending on the version of SDK being used). Sitebuilder templates of version 3 are applicable to Sitebuilder 3.2 and later. Sitebuilder templates of version 4 are supported by Sitebuilder 4.0 and later.

How to Create Template

The process of creating a Sitebuilder template is divided into the following steps:

- 1 Create a template design using a graphics editor. Note, that Sitebuilder imposes limitations on template design. For details on how to create a proper design, refer to the **Creating Template Design** (on page 10) section.
- 2 Create template logical structure.
- 3 Convert the design to HTML code using software of your choice.
- 4 Convert the HTML code to template source code. This enables Sitebuilder to properly render dynamic content added by end users. For details on how to perform the step, refer to the **Converting HTML Code to Template Source Code** (see page 23) section.
- 5 Create template's thumbnails. These images will represent the template at the Design step of Sitebuilder Wizard. Note, that Sitebuilder imposes limitations on template's thumbnails. For details on how to create proper thumbnails, refer to the **Creating Thumbnails for Templates, Menus and Headers** (on page 34) section.
- 6 Specify general information about the template. The information describes the template to end users. For details on how to specify the information, refer to the **Specifying General Information About Template** (on page 36) section.
- 7 Compile the template source code to template. For details, refer to the **Compiling Template** (on page 39) section.

After you have compiled the template, it can be installed to Sitebuilder. For details on how to do it, refer to the **Installing Template** (on page 41) section.

Creating Template Design

A typical template can be visually divided into components. This chapter describes components for which the design must be created. The chapter also contains general requirements your custom template design should meet.

In this chapter:

Overview of Template Components	11
Requirements to Templates Design	16

Overview of Template Components

A Sitebuilder template consists of the following parts:

- Header
- Top-level menu
- Lower-level menu
- Content area
- Footer

For example:

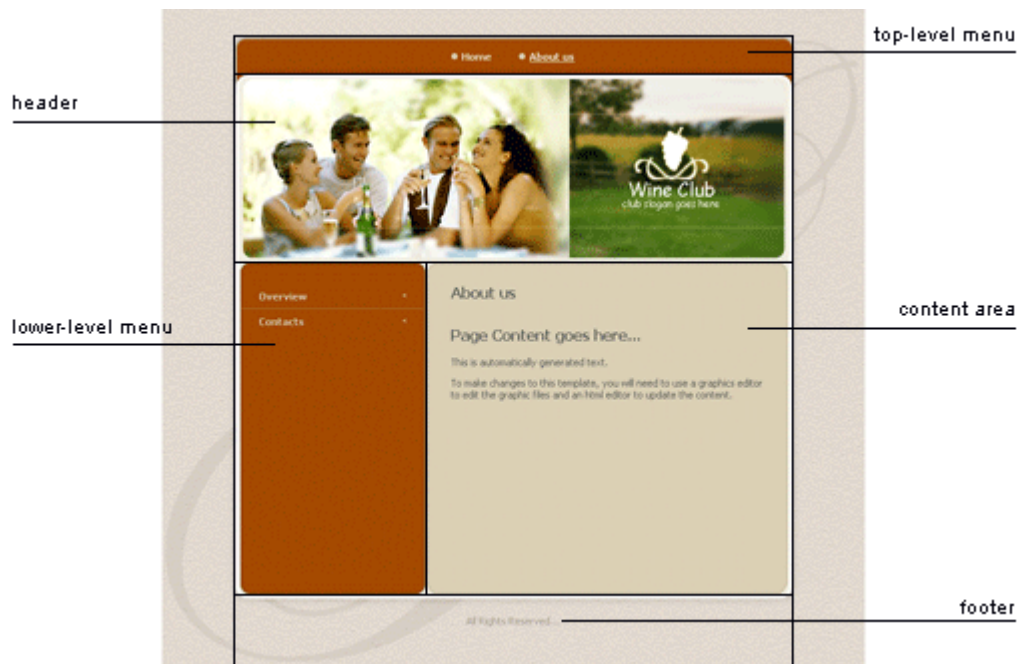


Figure 1: Component Parts of a Template

In this section:

Header	12
Top-Level Menu	13
Lower-Level Menu.....	14
Content Area	15
Footer.....	16

Header

Header is an image located at the top of a web page. Header consists of the following elements:

- Logotype
- Site title
- Site subtitle
- Banner

Note that it is recommended to place logo and site title close to each other. The subtitle should be placed under the title.

For example:



Figure 2: Header

Top-Level Menu

Top-level menu (main menu) contains links to the top level pages of a site. There are two types of top-level menu:

- A set of button links consisting of a background and a foreground. The foreground consists of the characters and pictures (menu item title) that appear on the screen. The background is the uniform canvas behind the menu item title.
- A set of links with small icons (bullets).

A menu item can have two statuses:

- Active (a selected item)
- Inactive (item available for selecting)

So, when creating a top-level menu, you should provide two designs for each menu item: one for active status and another for inactive status.

The main menu area is typically placed under the header. Besides, the page design should provide place for adding main menu items.

For example:



Figure 3: First-level menu

Lower-Level Menu

Lower-level menu (sub-menu) is a list of links to subpages of a section that was chosen from the main menu. Like the main menu, lower-level menu contains both active and inactive items.

For example:

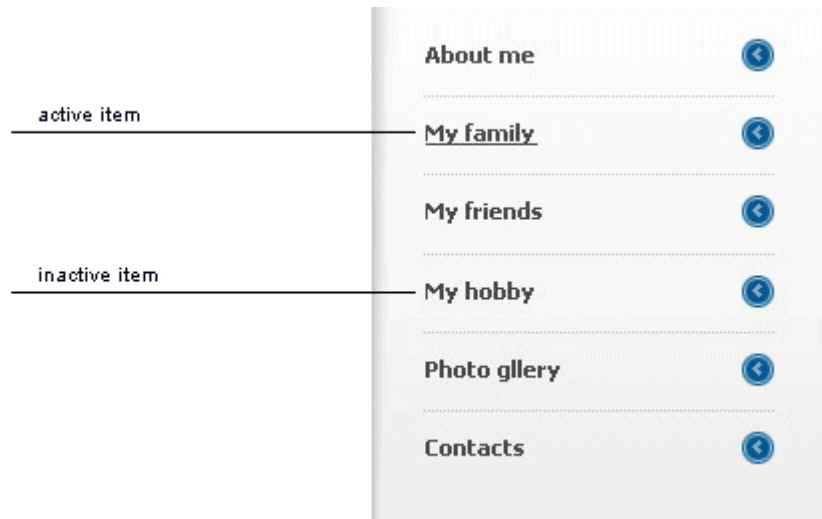


Figure 4: Second-level navigation bar

Content Area

Content area contains the site content (such as text, illustrations, and so on). Content area is divided into page content and page title. As a rule, page title is visually separated from the other text (usually, it is written with a bigger font).

For example:



Figure 5: Content area

Footer

Footer is a text that appears at the bottom of every page. Usually, footer contains copyright information, for example:

«Copyright © 2007, Company Name. All rights reserved.»

Footer may also duplicate the top-level menu.

For example:

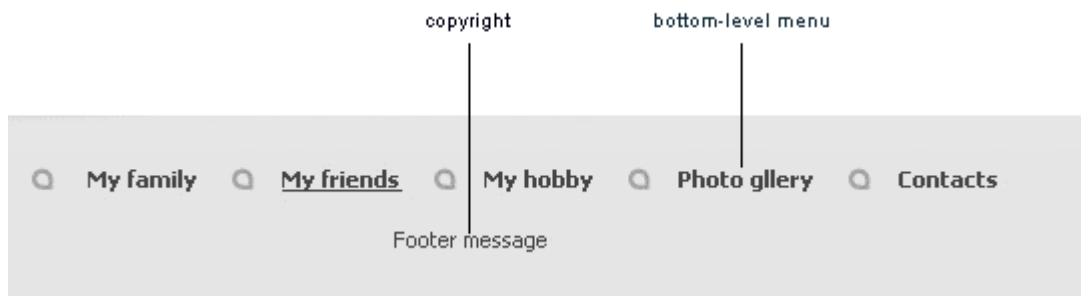


Figure 6: Footer






Requirements to Templates Design






There are some technical requirements applied to templates design:

- Template should be stretchable. That is a designer should provide the possibility of stretching a template both horizontally and vertically. Vertical and horizontal margins should remain the same after the template is filled with content.
- Template background should not be transparent.
- A template should use standard HTML fonts from the system set: Tahoma, Arial, Helvetica, Times, Verdana, Sans.
- All text files included in template must be saved with UTF-8 encoding.

Creating Template Logical Structure

A template must have a specific structure of directories. The sample of a correct structure is given below:

	<Template>/	The root template directory. You can rename it.
	Themes /	Contains files pertaining to template colour schemes. By default, Sitebuilder Wizard supports three different colour schemes for a specific template, but you can vary this number by changing number of sub-directories nested into this directory. Note that if you use more than three color schemes, this may break the appearance of a site based on the template at the "Design" step of the Wizard.
	Colour 1/	Contains files pertaining to the template colour scheme named <i>Colour 1</i> . The name is displayed at the "Design" step of the Wizard. You can rename the color themes (by renaming the sub-directories).
	Menus /	<p>Contains files that define menu style (active and inactive bullets, <i>.skin</i> files, etc). There can be one or more different menu styles depending on a number of sub-directories nested inside this directory. A specific colour scheme may use different sets of menu styles.</p> <p>Note that if you use more than three menu styles, this may break the appearance of a site based on the template at the "Design" step of the Wizard.</p> <p>The subdirectories of this directory must be named <i>MenuN</i>, where <i>N</i> stands for the number of a menu style.</p> <p>If the directory does not contain any subdirectories, menu style files are retrieved from the directory.</p>
	Menu1/	Contains files of the menu style named <i>Menu1</i> .

		Headers/	<p>Contains template header banners for a specific colour scheme. There can be one or more different header banners depending on a number of sub-directories nested into this directory.</p> <p>The subdirectories this directory of must be named <code>HeaderN</code>, where <i>N</i> stands for the number of a header banner.</p> <p>If the directory does not contain any subdirectories, header banners are retrieved from the directory.</p>
		<i>Header1/</i>	Contains header banner named <i>Header1</i> .
		Images/	Contains other graphics files related to a corresponding color scheme. For instance, the directory contains the default logo image (the <code>logo.gif</code> file).
		<code>styles.css</code>	Contains style sheets of a template. It may contain any number of selectors, but the recommended ones are <code>pageContent</code> and <code>pageContent a</code> . These selectors are responsible for presentation of the content area.
	<code>master.page</code>		Defines design and layout of a site page.

The color theme directory can also contain the modules design rules. For more details on this subject, see the **Changing Appearance of Sitebuilder Modules** (on page 42) section.

Before Creating Template Source Code

When the template design is ready, it should be converted into HTML code. Use software of your choice to perform the operation. Before starting the conversion, make sure that the HTML code will meet the Sitebuilder requirements. For details, refer to the **Requirements to HTML Code** (see page 20) section.

Template source code files responsible for site layout use HTML extended with Sitebuilder-specific elements. For details on the elements, refer to the **Sitebuilder Controls and Variables** (on page 21) section.

To see preliminary results of editing template source code, use the Sitebuilder Templates SDK tool. For details on the tool, refer to the **Previewing Templates** (see page 22) section.

In this chapter:

Requirements to HTML Code	20
Sitebuilder Controls and Variables	21
Previewing Templates	22

Requirements to HTML Code

When creating HTML code from existing design, fulfill the following requirements:

1 Exterior

- HTML pages should have a background – all visible space should be filled up with a color (including white).
- Active and inactive items both in top-level menu and lower-level menu should be different.

2 CSS

- Font sizes should be measured in points (for example: `font-size: 8pt;`).
- There should not be absolute positioning of elements in the HTML code.
- `Styles.css` should not have type or ID selectors (for example: `a{...}` or `#item{...}`). Nevertheless, you can use classes for content customization. The file should not contain images references.
- There should be no negative margins, paddings and the line-height property in declaration blocks.

3 Compatibility

- A converted code should look as similar to the original layout as possible in all popular browsers.

Sitebuilder Controls and Variables

Sitebuilder template source code may include plain HTML code, special elements of the `SiteBuilder` namespace and Sitebuilder variables. For example, a source code file may contain the following element:

```
<SiteBuilder:MyControl ID="TopMenu" />
```

Elements of the Sitebuilder namespace are called *controls*. They are responsible for dynamic content of sites. When the template source code is rendered by Sitebuilder, controls are substituted by specific HTML code. The code depends on control name and its attributes.

Each element type of a Sitebuilder control consists of a start tag, content (optional) and end tag (optional). Start tag contains `SiteBuilder` string and the control name separated by the colon (:) character. The tag can also contain control attributes. For instance, control `MyControl` (from the above example) has the `ID` attribute. For the complete list of the controls, refer to the **Appendix** (see page 87).

Template source code includes Sitebuilder *variables*. A variable is represented as a string bounded by \$ characters. It is used for defining dynamic parameters of the template (like paths to images, a page title text, copyright information, etc.). When Sitebuilder parses the template code, it substitutes a variable for its value. For instance, variable `$HomePageUrl$` is substituted for the URL of a site home page. Variables give those end users an opportunity to customize site's title, subtitle, copyright notice, etc. at the Design step of the Wizard.

Previewing Templates

Sitebuilder Templates SDK includes a utility that enables you to preview a site based on a specific template. The utility can be used only in MS Windows.

➤ *To preview a template, do the following:*

- 1 In the **Windows Explorer**, right-click a template directory and select the **Preview Sitebuilder Template** option.

You will be prompted to select a template scheme, menu style, and header banner.

You can open this window with the following command line:

```
Preview.exe [Full path to template root directory]
```

Or you can start `Preview.exe` and choose template root directory using program menu.

The `Preview.exe` is stored in *[Sitebuilder SDK folder]/Tools*. By default, the Sitebuilder SDK folder is `C:\Program Files\Parallels\Plesk Sitebuilder 4.5\SDK\`.

- 2 Click **Preview**.

The system will launch the check of the template structure. The results of the check are displayed in the **Logs** tab. If the structure is correct, the preview of the template is displayed in the **Browser** tab.

Note: This check does not require the template to contain preview images of a color scheme, menus, and header banners. Also, at this stage you do not have to provide a description of the template in the `info.xml` file. You can do it later on, when the template is completely ready.

Converting HTML Code to Template Source Code

After a site design is converted to HTML, it includes plain-text files and images. The files cannot be used as template source code for two reasons. First, they contain static content and cannot be customized by Sitebuilder end users. Secondly, Sitebuilder requires a site page file to be of a specific format. To make template source code from the HTML files, you should do the following:

- 1 Change paths to all images within the HTML files to Sitebuilder-specific paths. For details, refer to the **Adding Images to Template** (see page 24)section.
- 2 Convert HTML code representing a site page to a Sitebuilder-specific format. For details, refer to the **Creating Master.page File** (on page 24) section.
- 3 Create customizable content. This includes menu styles and header banners. For details, refer to the **Creating Customizable Content** (on page 29)section.

Note: You can optionally change appearance of Sitebuilder modules elements. For details, refer to the **Creating Appearance of Sitebuilder Modules** (on page 42)section.

In this chapter:

Adding Images to Template.....	24
Creating Master.page File	24
Master.page Example	26
Creating Customizable Content.....	29

Adding Images to Template

To add an image to a template, copy it to the `[Template path]/Themes/[Colour scheme name]/Images/` directory and add the `IMG` element to the corresponding place of a template source code file. The full path to the `images` directory (from the root site folder to the folder of the currently selected color theme) is stored in the `$Theme$` variable.

For example, if an image file named `bullet.gif` is located in the `images` directory of the red color scheme (`[Template path]/Themes/red/images/bullet.gif`), then the path to this file in the HTML code of the template should look as follows:

```

```

Note that paths to specific header banners or menu images (and therefore located not in the `images` directory) should be still specified in the same way as if the images were located in the `images` directory. For example, the `IMG` elements of the following images:

```
[Template path] / Themes / red / headers / header1 /  
headerBullet.gif
```

or

```
[Template path] / Themes / red / menus / menu1 / menuBullet.gif
```

should look as follows:

```

```

and

```

```

Creating Master.page File

The first step of converting HTML code into Sitebuilder template source code is creating the `Master.page` file in the root directory of the created template. The `Master.page` file contains code of a site page. This file is used with all color schemes (it must not contain information related only to a particular color scheme).

File `Master.page` must be comprised of HTML code, Sitebuilder controls and variables. HTML code is used to create a site layout. Controls and variables are used to fill the site with dynamic content. The standard site layout defined in the `Master.page` file can be schematically represented in the following way:

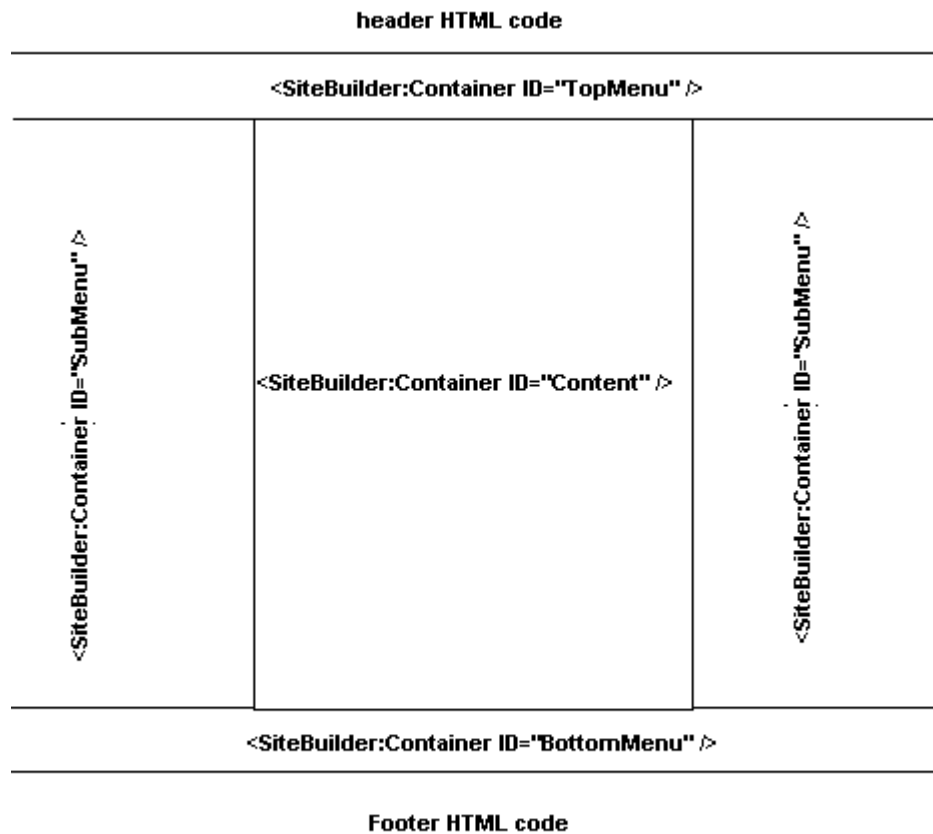


Figure 7: Schema of a site page.

To create the `Master.page` file, create a new text document in the template root directory and paste the HTML code generated from a template design into it. Then perform the following actions on the text document:

- 1 Find HTML code of the top-level menu and replace it with the `<Sitebuilder:Container ID="TopMenu"/>` control.
- 2 Find HTML code of the lower-level menu and replace it with the `<Sitebuilder:Container ID="SubMenu"/>` control.
- 3 Find HTML code of the content area and replace it with the `<Sitebuilder:Container ID="Content"/>` control.
- 4 (Optional) Find HTML code of the bottom-level menu and replace it with the `<Sitebuilder:Container ID="BottomMenu"/>` control.

Note: If the design did not have the bottom-level menu, you can omit the step.

- 5 Find HTML code of the page title and replace it with the `$PageTitle$` variable.

- 6 (Optional) Find URL to the site logo image and replace it with the `$LogoPath$` variable. If you are not using the logo image on the site, insert the element `` into any place of the file, and put the transparent 1 x 1 pixel file called `logo.gif` into the `Images` directory.
- 7 Save the text document as `Master.page`

Note: All controls and variables added to `master.page` at steps 1,2,3,4,7 are required.

Variable `$HomePageUrl$` stores URL to a site home page. Optionally, you can substitute the path to a site home page with the variable.

For details on how to create custom menus style, refer to the **Creating Menu Style** (see page 30) section.

Sometimes end-users want to customize the site title, subtitle and copyright information. To give the users an opportunity to customize additional site elements, open the `Master.page` in the text editor and do the following:

- Find HTML code of the site title and replace it with the `$SiteTitle$` variable.
- Find the site subtitle and replace it with the `$SiteSubTitle$` variable.
- Find the copyright details and replace them with the `$Copyright$` variable.

Note: These steps are optional, so you can omit them.

Master.page Example

File `Master.page` can look as in the following sample. Sitebuilder controls and variables are bolded.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body marginheight="0" marginwidth="0" topmargin="0" rightmargin="0"
bottommargin="0"
  leftmargin="0">
  <table cellpadding="0" cellspacing="0" style="width: 100%; height:
100%;" class="main-bg">
    <tr>
      <td valign="top" height="264">
        <table cellpadding="0" cellspacing="0" border="0"
width="237">
          <tr>
            <td width="237" height="132" class="img_top_header_bg">
              <table cellpadding="0" cellspacing="0" border="0">
                <tr>
                  <td style="padding-left: 34px;">
                    <a href ="$HomePageUrl$"></a>
              </td>
            </tr>
          </table>
        </td>
      </tr>
    </table>
  </td>
</tr>
</table>
```

```

        <td style="padding-left:
10px;"class="company">$$SiteTitle$</td>
        </tr>
    </table>
    </td>
</tr>
<tr>
    <td width="237" height="2" bgcolor="#525051"></td>
</tr>
<tr>
    <td width="237" height="130" valign="middle"
class="slogan img_bottom_header_bg"
    align="center">
        $$SiteSubTitle$
        <div style="width: 237px; height:
0px;"><span></span></div>
    </td>
</tr>
</table>
</td>
<td valign="top" bgcolor="#525051">
    <table cellpadding="0" cellspacing="0" border="0" width="2">
        <tr>
            <td width="2" height="100%" bgcolor="#525051"></td>
        </tr>
    </table>
</td>
<td width="100%" valign="top" class="img_header_bg">
    <table cellpadding="0" cellspacing="0" border="0"
width="304">
        <tr>
            <td width="302" height="263" class="img_header"></td>
        </tr>
    </table>
    <div style="width: 100%; height: 0px;">
        <span></span>
    </div>
</td>
<td valign="top" bgcolor="#525051">
    <table cellpadding="0" cellspacing="0" border="0" width="2">
        <tr>
            <td width="2" height="100%" bgcolor="#525051"></td>
        </tr>
    </table>
</td>
<td valign="middle">
    <table cellpadding="0" cellspacing="0" width="200"
align="center">
        <SiteBuilder:Container ID="TopMenu" />
    </table>
    <div style="width: 230px; height: 0px;">
        <span></span>
    </div>
</td>
</tr>
<tr>
    <td colspan="5" height="2" bgcolor="#525051"></td>
</tr>
<tr>

```

```
<td colspan="3" height="100%" valign="top" style="padding-top:
26px; padding-right: 28px; padding-left: 28px;" class="pageContent"
bgcolor="#1d1d1d">
    <table cellpadding="0" cellspacing="0" border="0">
        <tr>
            <td class="text-header">$PageTitle$</td>
            <td style="padding-left: 5px;">
                </td>
        </tr>
    </table>
    <div style="width: 0px; height: 15px;">
        <span></span>
    </div>
    <SiteBuilder:Container ID="Content" />
</td>
<td width="2" bgcolor="#525051"></td>
<td valign="top" style="padding-top: 28px;">
    <table cellpadding="0" cellspacing="0" border="0" width="200"
align="center">
        <SiteBuilder:Container ID="SubMenu" />
    </table>
    <div style="width: 180px; height: 0px;">
        <span></span>
    </div>
</td>
</tr>
<tr>
    <td colspan="5" height="2" bgcolor="#525051"></td>
</tr>
<tr>
    <td colspan="3" height="100%" valign="top" style="padding-top:
26px; padding-left: 28px;">
        <table cellpadding="0" cellspacing="0" align="center">
            <tr>
                <SiteBuilder:Container ID="BottomMenu" />
            </tr>
        </table>
    </td>
    <td width="2" bgcolor="#525051"></td>
    <td valign="middle" bgcolor="#1D1D1D">
        <div style="width: 0px; height: 10px;">
            <span></span>
        </div>
        <div align="center" class="footer">$Copyright$</div>
    </td>
</tr>
</table>
</body>
</html>
```

Creating Customizable Content

Each template colour scheme can have specific set of menu styles and header banners. At the Design step of Sitebuilder Wizard, end users pick a combination of a colour scheme, menu style and header banner. This section describes how to create custom menu styles and how to add header banners to a template.

In this section:

Creating Menu Style	30
Adding Header Banners	33

Creating Menu Style

A template must contain a top-level menu (usually situated at the top of the page) and a lower-level menu (usually situated at the left/right part of the page). Some templates use a bottom-level menu.

Files Defining Menu Style

The appearance of each menu must be defined in at least one color scheme. It is defined by menu images and the following plain-text files:

- Top-level menu is described in the `TopMenu.skin` file.
- Lower-level menu is described in the `SubMenu.skin` file.
- Bottom-level menu is described in the `BottomMenu.skin` file. (Optional. If not defined, bottom-level menu is not displayed.)

The files along with the images must be put into the following directory:

[Template path]/Themes/[Colour scheme name]/Menus

If you define only one menu style, you can place menu style files into the root of the `Menus` directory. If you want to define more than one menu style, create the needed number of directories (`Menu1`, `Menu2`, and so on) and put there the corresponding skin files and images. For example:

[Template path]/Themes/[Colour scheme name]/Menus/Menu1/TopMenu.skin

[Template path]/Themes/[Colour scheme name]/Menus/Menu1/someImage.gif

[Template path]/Themes/[Colour scheme name]/Menus/Menu2/SubMenu.skin

Elements and Variables that Can Be Used in .skin Files

`TopMenu.skin`, `SubMenu.skin` and `BottomMenu.skin` files contain special elements and variables. Each element stores HTML code of a specific menu part. Variables store menu items data.

They are as follows:

Elements

- `ItemTemplate`. Stores HTML code of a regular menu item.
- `AlternatingItemTemplate`. (Optional) Stores HTML code of an odd menu item. If it is not defined, `ItemTemplate` content is used.
- `SelectedItemTemplate`. (Optional) Stores HTML code of a selected menu item. If it is not defined, `ItemTemplate` or `AlternatingItemTemplate` is used.
- `SeparatorTemplate`. (Optional) Stores HTML code of a separator between menu items. If it is not defined, the separator is not displayed.

- `HeaderTemplate`. (Optional) Stores HTML code of a header section of the menu. If it is not defined, the header section is not displayed. For example, it could be a table start tag `<table>`.
- `FooterTemplate` (Optional) Stores HTML code of a footer section of the menu. If it is not defined, the footer section is not displayed. For example, it could be a table end tag `</table>`.

Variables (only for `ItemTemplate`, `SelectedItemTemplate`, `AlternatingItemTemplate` elements)

- `Url`. Link to which a menu item leads. **(Required)**
- `Title`. Title of a menu item. **(Required)**
- `Index`. Number of menu items. **(Optional)**

Rendering Algorithm

The algorithm is similar for top-level, lower-level and bottom-level menus. It is described by the example of the top-level menu rendering.

1. Sitebuilder opens `master.page` file and searches for the `<SiteBuilder:Container ID="TopMenu" />` control.
2. If the control is found, Sitebuilder searches for the `TopMenu.skin` file (first in the `MenuX` sub-directory, and if not found, in the `Menus` sub-directory of a template colour scheme directory).
3. If the file is found, Sitebuilder generates HTML code of the menu layout basing on `TopMenu.skin` variables and elements content . For instance, for each regular menu item it adds the content of the `ItemTemplate` element to the HTML code.
4. Sitebuilder replaces the `<SiteBuilder:Container ID="TopMenu" />` string in the `master.page` file with the generated HTML code.

Example of a .skin File

```
<SiteBuilder:SiteMenu>
  <HeaderTemplate>
    <table width="250px" border="0">
  </HeaderTemplate>
  <ItemTemplate>
    <tr>
      <td><a href="$Url$"></a></td>
      <td style="padding-left: 30px;">
        <a class="menu" href="$Url$" style="width: 100%;">$Title$</a>
      </td>
    </tr>
  </ItemTemplate>
  <SelectedItemTemplate>
    <tr>
      <td><a href="$Url$"></a></td>
```

```

        <td style="padding-left: 30px;">
            <a class="menu" href="$Url$" style="width: 100%;">$Title$</a>
        </td>
    </tr>
</SelectedItemTemplate>
<SeparatorTemplate>
    <tr>
        <td colspan="2" height="20">
            <div style="height: 1px; background-color: #252324;">
                <span></span>
            </div>
        </td>
    </tr>
</SeparatorTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</SiteBuilder:SiteMenu>

```

To add a menu style, place this code into the corresponding `.skin` file: `TopMenu.skin`, `SubMenu.skin`, or `BottomMenu.skin`.

➤ **To create a menu style from scratch:**

- 1 Convert the design of the menu into HTML code.

Note. It is recommended to use the following CSS minimum values for menus: 6pt for top-menu text size, 10pt for lower-level menu text size.

- 2 Place the HTML code defining the appearance of a regular menu item into the `ItemTemplate` section.
- 3 If you have created a special design for the selected menu items, place the HTML code defining the layout of a selected menu item into the `SelectedItemTemplate` section.
- 4 If you have created a special design for odd menu items, place the HTML code defining the layout of an odd menu item into the `AlternatingItemTemplate` section.
- 5 If you want the menu items to be visually separated from each other, place the HTML code defining the layout of the separator into the `SeparatorTemplate` section.
- 6 If your menu has a header and a footer, place the HTML code defining their layout into the `HeaderTemplate` and `FooterTemplate` sections respectively (for example, if the menu is a table, the `table` start tag will be the header, and the `table` end tag will be the footer).

Note: The paths to images are specified with the use of the `$Theme$` variable. For more details, see the **General Rules for Specifying Paths to Image Files** (see page 24) section.

Adding Header Banners

The files representing header banners must be placed into the following location:

[Template path]/Themes/[Colour scheme name]/Headers/Header[Number]

To provide end users with an opportunity to change a template's banner through the Sitebuilder Wizard interface, do the following:

- 1 Create as many sub-directories of the `Headers` directory as the number of different header banners to be displayed in Wizard.
- 2 Copy header banners to the created directories (one banner per directory). The images must share the same name (say, `banner_image.gif`).
- 3 Append the `master.page` file with the following HTML code of the banner image:

```

```

- 4 Save and close the `master.page` file.

After the template is imported into Sitebuilder, the alternative header images will be available for selection at the Design step of the Wizard. To make sure of this, you can preview the template (see page 22) using the Preview tool.

In Sitebuilder for Linux/Unix, end users are enabled to upload their own banner images through the Sitebuilder interface only if the *[Template path]/Themes/[Colour scheme name]/Headers/Header[Number]* directory has only one image.

In Sitebuilder for Windows, to give users access to uploading their own banner images, edit the `info.xml` file. The `detachableHeader` attribute of the `themes` element should contain the image name. To learn more about `info.xml`, refer to the **Specifying General Information About Template** (on page 36) section.

If you edit source code of existing template that contains different banners, you can ignore the 3 and 4 steps of the above guidelines and simply copy custom images into the corresponding directories.

Creating Thumbnails for Templates, Menu Styles, and Headers

Thumbnails are reduced-size preview images of templates. They are shown at the Design step of Sitebuilder Wizard to help end users select proper combination of a colour scheme, menu and header banner. When an end user clicks a thumbnail, the preview image is displayed. The recommended format of thumbnails and preview images is `gif` or `jpg`.

Thumbnails of different color schemes must be of the same size and scale. If after scaling the image does not fit the proportions, it should be placed in the center of the canvas with required proportions and the `#cccccc` background color. Then the data is to be saved and used instead of the image. For instance, if you have a `themename_icon.jpg` which size is 70 x 43 px, create a 70 x 53 px canvas, fill it with the `#cccccc` color, put the `themename_icon.jpg` in the center of the canvas (vertically and horizontally), and save the merged image as `themename_icon.jpg`.

In this chapter:

Creating Thumbnails for Templates.....	34
Creating Thumbnails for Menu Styles.....	35
Creating Thumbnails for Header Banners	35

Creating Thumbnails for Templates

Create preview images and thumbnails that will best represent available color schemes. The width of a preview image should be 780px. The name of the image file should be `themename.jpg(gif)`, where `themename` is a name of the scheme directory. Place the created file into the `Themes` directory. For details on the location of the directory, refer to the **Creating Template Logical Structure** (on page 17) section.

If you add a preview image, Sitebuilder automatically creates thumbnails of this image. However, to increase the quality of thumbnails, you can make it yourself. These images should be placed into the `Themes` directory.

The names of these images and their maximal sizes are the following:

- `themename_thumb.jpg(gif)` (118x89)
- `themename_small.jpg(gif)` (86x89)
- `themename_selected.jpg(gif)` (252x189)

Creating Thumbnails for Menu Styles

Create images that will best represent available menu styles.

General rules for making thumbnails for menu styles:

- The number of thumbnail files depends on the number of color schemes and menu styles. A thumbnail must be created for each menu style in each color scheme.
- The maximum height of a thumbnail is 26px, and the maximum width is 69px.
- Recommended file format is `jpg` or `gif`.
- The name of a thumbnail file must be `menuN.jpg(gif)`, where N is the menu index number. The thumbnail files must be placed in the `/menus` subdirectory of a color scheme.
- If the design (background and foreground colors) of menu buttons in different color schemes is identical, you can create thumbnails only for one color scheme and then copy them into other schemes.
- Menu item thumbnail should be the same size as menu item. If the thumbnail is bigger than 69 px, cut off the item thumbnail text.

Note: Creating thumbnails for menu styles is required for Sitebuilder for Linux/Unix templates and optional for Sitebuilder for Windows templates.

Creating Thumbnails for Header Banners

Create images that will best represent available header banners.

General rules for making thumbnails for header banners:

- The thumbnail must not contain text
- The number of thumbnail files depends on the number of color schemes and banners. A thumbnail file must be created for each pair of color scheme and banner.
- The maximum height of a thumbnail is 56px, and the maximum width is 230px.
- Recommended file format is `jpg` or `gif`.
- The name of a thumbnail file must be `headerN.jpg(gif)`, where N is the header index number. The file `headerN.jpg(gif)` must be placed in the `Headers` directory. For details on physical structure of a template, refer to the **Creating Template Logical Structure** (on page 17) section.
- If the design of banners in different color schemes is identical, you can create thumbnails only for one color scheme and then copy them into the other schemes.

Note: Creating thumbnails for header banners is required for Sitebuilder for Linux/Unix templates and optional for Sitebuilder for Windows templates.

Specifying General Information About Template

To compile a template, you must create a file where basic technical information about a template is stored. The file is named `info.xml`. It should be located in the template root directory.

The structure of the `info.xml` file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<template id="yourCompany-business-001" version="1.0"
caption="Business site" formatVersion="4" keywords=" business books"
category="Business" >
  <themes default="yellow">
    <theme id="Green" caption="Green scheme"/>
    <theme id="Yellow" caption="Yellow scheme"/>
    <theme id="Orange" caption="Orange scheme"/>
  </themes>
  <compatibility>
    <screen>
      <minwidth>800</minwidth>
    </screen>
    <browsers>
      <browser>MS IE 6.0</browser>
      <browser>Mozilla FireFox 1.5</browser>
      <browser>Opera 8</browser>
    </browsers>
    <modules>
      <module>Blog</module>
      <module>ImageGallery</module>
    </modules>
  </compatibility>
</template>
```

The first line, called XML declaration, defines the version of XML and the charset used in the document.

The root element of the document is `template`. It has the following attributes:

- `id`. This attribute contains the name of the template. The name of the template should start with a unique name, so that the template would be easily identified in Sitebuilder. The template name should coincide with the name of the directory containing the template content. If the template pack will be used in Sitebuilder for Linux/Unix, then the templates' names should contain only Latin characters, numerical digits, underscores (`_`), dots, and dashes.
- `version`. This attribute contains the template version.

Version number consists of 2 digits:

- First (1). The number of times the template code was changed retaining the same design.

- Second (0). The number of hot fixes.
- `caption`. This attribute contains the description of the template. This description is displayed to end users. For example, "Travel company".
- `formatVersion`. (*Optional*) This attribute contains the version of SDK that is used to compile the template. If you create the template using Sitebuilder SDK v. 4.5, then put "4" into this section. If you use SDK v. 3.2, then there is no need to include the `formatVersion` section in the `info.xml` file, because the format's version will be automatically defined as "3" during template compilation.
- `keywords`. This attribute contains the list of keywords that a user might enter to find the template at the Design step of the Wizard. Therefore, provide keywords which best describe you template.
- `category`. This attribute specifies the category the template belongs to, for example, "Business". It may contain a blank value or be absent.

The `template` element has the following child elements:

- `themes`. The element contains information about colour schemes used in the template. Its attributes are as follows:
 - `default`. (*Optional*) It defines default color scheme.
 - `detachableHeader`. (*Optional*) Defines if the custom header banner can be uploaded in Sitebuilder Wizard. For details, refer to the **Customizing Header Banners** (on page 33) section.
- `theme`. The element represents a specific colour scheme. Its name must be *the same as the color scheme directory name*. This element has two following attributes:
 - `id`. The `id` attribute contains the name of a color scheme.
 - `caption`. The `caption` attribute contains the description of a color scheme. This can be any text of your choice, e.g. "Green color scheme".
- `compatibility`. (*Optional*) This element describes properties of the template. Currently, these properties are just assigned to the template, but in the next version of Sitebuilder, the properties will be displayed to the end users selecting a template for their site.

The `compatibility` element has its own child elements:

- `screen`. The element has the child element `minwidth`, which contains minimal width of a template without horizontal scrolling.
- `browsers`. The `browsers` element contains the list of browsers' names and versions compatible to the template.
- `modules`. If you create the template specifically for certain Sitebuilder modules, specify names of the intended modules in this section.

The `modules` element can have the following values:

- Blog
- eShop
- Forum
- ImageGallery

- Guestbook
- Login
- Feedback
- RssReader
- Voting
- AreaMap
- FlashIntro
- SiteMap
- FileDownload
- ExternalPage
- OnlineStatusIndicator
- SitePal

Compiling Template

To be installed and operated in Sitebuilder, created template source files must be compiled into a template pack (the package that comprises one or more templates). This is an `.msi` file for Windows OS and `.bin` file for Linux/Unix OS.

Templates can be compiled only in MS Windows. Linux/Unix compiler will be available in next versions of the Sitebuilder SDK.

Prior to launching the compilation process, make sure that:

- You have installed Sitebuilder SDK
- The template directory contains the `info.xml` file
- The template has a unique ID and this ID coincides with the name of the template directory in the `info.xml` file. For details on where the ID is specified, refer to the **Specifying General Information About Template** (on page 36) section.

➤ **Use the following procedure to compile a template:**

- 1 Anywhere on your hard drive, create a new `compile.sbtpr` file containing the following lines:

```
<?xml version="1.0" encoding="utf-8"?>
<templatePack name="Sample pack" type="Windows">
<directory path="example-01" />
</templatePack>
```

- 2 Substitute values of the following arguments with your own.

TemplatePack element.

`name`. (*Optional*) Name of a template pack.

`type`. Type of operational system, for which the template is compiled. Values: *Windows, Unix*.

Directory element.

`path`. Path to the directory where template source code is located. The path can be absolute or relative (to the directory where `compile.sbtpr` file is located).

Example: `D:\Templates\my-template-01`.

Note: Use multiple `directory` elements to create a template pack.

- 3 Double click the `compile.sbtpr` file, or right click the file and choose "Compile template pack" option.

A window displaying the results of the compilation process opens.

The compiled template pack will look as follows: *[name of the .sbtpr file].msi* or *[name of the .sbtpr file].bin* depending on a template type.

Note: You can change name of `.sbtp` file. For instance, instead of `compile.sbtp` you can use `sample.sbtp`.

Remarks (Sitebuilder for Windows)

When you compile template source code to an `.msi` file for the first time, content of the `.sbtp` file is affected. It is true even if the attempt to compile the source code has failed. The following arguments are added automatically to the `templatePack` element:

`version`. Stores number of times a template was recompiled (starting from 1).

`uid`. Identifier of a template pack. The argument value is unaffected on recompilation.

Note: It is not recommended to add or edit the `version` or `uid` values without necessity.

Note: After each successful compilation of a given source code, the `version` value is automatically increased by 1.

Installing Template

Prior to starting template installation process, make sure that the following conditions are met:

- You have administrator rights on the computer where you are going to install a template pack
- The template packs you want to install are compatible with Sitebuilder installed on the computer.

➤ ***To install a template (pack) under MS Windows:***

Run the template (pack) `.msi` file

➤ ***To install a template (pack) under Linux/Unix:***

Run the template (pack) `.bin` file:

```
# sh template_name.bin
```

After the installation process is completed, the newly installed template appears in the list of available templates in the Sitebuilder Wizard (**Design** step).

Changing Appearance of Sitebuilder Modules

Each template's color scheme may include additional rules for displaying Sitebuilder modules. To customize look and feel of such modules, you need source code of modules presentation. The code can be copied from template source code samples included in Sitebuilder SDK. Use this code to customize presentation of Sitebuilder modules.

In this chapter:

Customizing Styles Used in Modules	43
Customizing Dynamic Content of Modules	67

Customizing Styles Used in Modules

The presentation style of modules is customized by the `modules.css` file located in the `[Template path]/Themes/[Colour scheme name]` directory. Each colour scheme can have different `modules.css` file.

Rules of the CSS file are divided into groups by selectors name. Selector name of a specific group contains a specific substring (say *category*). For instance, the following rule belongs to the Category group:

```
.mod-category-body {font-color: #000000}
```

A module style can be comprised of one or more groups of rules. Thus, for example, the Blog module categories are customized by the Category group, posts are customized by the Item group, and comments to posts are customized by the Comment group.

The style of each module is defined by the following groups:

- *Category*. Style of a category block
- *Item*. Style of list items
- *Comment*. Style of a comments block
- *Search*. Style of a search block
- *Form*. style of a form block
- *Information*. Style of an information block
- *Pager*. Style of a paging block
- *Statuses*. Style of information messages
- *Main*. Style of template text boxes and fields, horizontal rules, and buttons

Thus, when you change, for example, the Item group rules, you modify a series of module design elements at the same time. To simplify the process of debugging, for each template the **Previewing Templates** (see page 22) tool was developed. This tool allows you to quickly preview the applied changes.

When you change, for example, the background color for `mod-item-body` rule, remember to make sure that all the interface elements - text, links, buttons - are at least visible against the new background.

In this section:

"Categories" Group	44
"Items" Group	47
"Comments" Group	53
"Search" Group	56
"Form" Group	58
"Information" Group	60
"Paging" Group	62
"Status" Group	63
"Main" Group	65

"Categories" Group

Some Sitebuilder modules enable site owners to create categories to group similar objects together (for example, products, blog posts, images and so on). Customize the appearance of categories by editing the Category group of rules.

In this section:

Odd Category Body	44
Even Category Body	45
Custom Links in Category Body	45
Title in Category Body	46
Category Header	46
Links in Category Header	47

Odd Category Body

Defines the style for an odd category body area.

Selector

```
.mod-category-body
```

Sample

```
.mod-category-body
{
  border: 1px solid #969696;
  background-color: #ffffff;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Even Category Body

Defines the style for an even category body area.

Selector

```
.mod-category-body-alter
```

Sample

```
.mod-category-body-alter
{
  border: 1px solid #969696;
  background-color: #F9F9F9;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Custom Links in Category Body

Defines the style for links within a category body area.

Selector

```
.mod-category-body a, .mod-category-body-alter a
```

Sample

```
.mod-category-body a, .mod-category-body-alter a
{
  color: #2752A9;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Remarks

This style rule must be defined above the rule **Links in Category Header Blocks** (see page 47).

Title in Category Body

Defines the style for title text within a category body area.

Selector

```
.mod-category-body-title
```

Sample

```
.mod-category-body-title
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Category Header

Defines the style for a category header area.

Selector

```
.mod-category-header
```

Sample

```
.mod-category-header
{
  border: 1px solid #969696;
  background-color: #E5E5E5;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Links in Category Header

Defines the style for links within a category header area.

Selector

`a.mod-category-header-a`

Sample

```
a.mod-category-header-a
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

"Items" Group

Customize the appearance of items by editing the Item group of rules.

In this section:

Odd Items Body	48
Even Items Body	48
Strong Links in Item Body.....	49
Custom Links in Item Body.....	49
Titles in Item Body.....	50
Horizontal Rules in Item Body	50
Buttons in Items Body	50
Strong Buttons in Item Body.....	51
Highlighted Text in Item Body.....	51
Item Header	52
Item Footer.....	52
Link in Item Footer.....	53

Odd Items Body

Defines the style for an odd item body area.

Selector

```
.mod-item-body
```

Sample

```
.mod-item-body
{
  border: 1px solid #969696;
  background-color: #ffffff;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Even Items Body

Defines the style for an even item body area.

Selector

```
.mod-item-body-alter
```

Sample

```
.mod-item-body-alter
{
  border: 1px solid #969696;
  background-color: #F9F9F9;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Strong Links in Item Body

Defines the style for link within an item body area.

Selector

```
a.mod-item-body-a-strong
```

Sample

```
a.mod-item-body-a-strong
{
  color: #003399;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Custom Links in Item Body

Defines the common style for all links within an item body area.

Selector

```
.mod-item-body a, .mod-item-body-alter a
```

Sample

```
.mod-item-body a, .mod-item-body-alter a
{
  color: #666666;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Remarks

This style rule must be defined above the rules **Strong Links in Item Body Blocks** (see page 49) and **Link in Item Footer Blocks** (see page 53).

Titles in Item Body

Defines the style for text titles within an item body area.

Selector

```
.mod-item-body-title
```

Sample

```
.mod-item-body-title
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Horizontal Rules in Item Body

Defines the style for hr-elements within an item body area.

Selector

```
.mod-item-body-hr
```

Sample

```
.mod-item-body-hr
{
  background-color: #CECECE;
}
```

Buttons in Items Body

Defines the style for simple buttons within an item body area.

Selector

```
.mod-item-button
```

Sample

```
.mod-item-button
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Strong Buttons in Item Body

Defines the style for strong buttons within an item body area.

Selector

```
.mod-item-button-strong
```

Sample

```
.mod-item-button-strong
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
  font-weight: bold;
}
```

Highlighted Text in Item Body

Defines the style for highlighted text within an item body area.

Selector

```
.mod-item-highlight, a.mod-item-highlight
```

Sample

```
.mod-item-highlight, a.mod-item-highlight
{
  color: #cc0000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Item Header

Defines the style for an item header area.

Selector

```
.mod-item-header
```

Sample

```
.mod-item-header
{
  border: 1px solid #969696;
  background-color: #E5E5E5;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Item Footer

Defines the style for an item footer area.

Selector

```
.mod-item-footer
```

Sample

```
.mod-item-footer
{
  background-color: #F3F3F3;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Link in Item Footer

Defines the style for links in an item footer area.

Selector

`a.mod-item-footer-a`

Sample

```
a.mod-item-footer-a
{
  color: #666666;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

"Comments" Group

Sitebuilder Wizard enables its users to add blogs to their sites. After the blog owner posts some messages to the blog, site visitors can leave their comments to these posts. Customize the appearance of blog comments by editing the Comments group of rules.

In this section:

Odd Comment Body	54
Even Comment Body	54
Custom Links in Comment Body	55
Titles in Comment Body	55
Comment Header	56
Links in Comment Header	56

Odd Comment Body

Defines the style for an odd comment body area.

Selector

```
.mod-comment-body
```

Sample

```
.mod-comment-body
{
  border: 1px solid #969696;
  background-color: #F7F7F7;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Even Comment Body

Defines the style for an even comment body area.

Selector

```
.mod-comment-body-alter
```

Sample

```
.mod-comment-body-alter
{
  border: 1px solid #969696;
  background-color: #F7F7F7;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Custom Links in Comment Body

Defines the common style for links within a comment body area.

Selector

```
.mod-comment-body a, .mod-comment-body-alter a
```

Sample

```
.mod-comment-body a, .mod-comment-body-alter a
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Remarks

This style rule must be defined above the rule **Links in Comment Header Blocks** (see page 56).

Titles in Comment Body

Defines the style for title text within a comment body area.

Selector

```
.mod-comment-body-title
```

Sample

```
.mod-comment-body-title
{
  color: #666666;
  font-family: Arial, sans-serif;
  font-size: 10pt;
}
```

Comment Header

Defines the style for comment header area.

Selector

```
.mod-comment-header
```

Sample

```
.mod-comment-header
{
    border: 1px solid #969696;
    background-color: #E5E5E5;
    color: #000000;
    font-family: Arial, sans-serif;
    font-size: 8pt;
}
```

Links in Comment Header

Defines the specific style for links within a comment header area.

Selector

```
a.mod-comment-header-a
```

Sample

```
a.mod-comment-header-a
{
    color: #000000;
    font-family: Arial, sans-serif;
    font-size: 8pt;
}
```

"Search" Group

Customize the appearance of a module search box (say, for eShop module) by editing the Search group of rules.

In this section:

Search Form	57
Buttons in Search Form.....	57

Search Form

Defines the style for a search form area.

Selector

```
.mod-search
```

Sample

```
.mod-search
{
  border: 1px solid #969696;
  background-color: #F3F3F3;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Buttons in Search Form

Defines the style for buttons within a search form area.

Selector

```
.mod-search-button
```

Sample

```
.mod-search-button
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
  font-weight: bold;
}
```

"Form" Group

Some Sitebuilder modules involve filling out various forms, for example, form for adding a forum topic or a comment to blog post. Customize the appearance of a module form area by editing the Form group of rules.

In this section:

Form	58
Titles in Form	58
Horizontal Rules in Form	59
Buttons in Form	59
Links in Form	59

Form

Defines the style for a form area.

Selector

```
.mod-form
```

Sample

```
.mod-form
{
  border: 1px solid #969696;
  background-color: #F7F7F7;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Titles in Form

Defines the style for title text within a form area.

Selector

```
.mod-form-title
```

Sample

```
.mod-form-title
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Horizontal Rules in Form

Defines the style for HR elements within a form area.

Selector

```
.mod-form-hr
```

Sample

```
.mod-form-hr
{
  background-color: #CECECE;
}
```

Buttons in Form

Defines the style for buttons within a form area.

Selector

```
.mod-form-button
```

Sample

```
.mod-form-button
{
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
  font-weight: bold;
}
```

Links in Form

Defines the style for links within a form area.

Selector

```
a.mod-form-a
```

Sample

```
a.mod-form-a
{
  color: #2752A9;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

"Information" Group

Certain site visitors' actions are accompanied with information messages, for example, if a site contains an online store and a site visitor clicks the Checkout button, a message with details about his or her order is displayed. Customize the appearance of a module message area by editing the Information group of rules.

In this section:

Message Body	60
Message Header	61
Message Footer	61
Links in Message Footer	62

Message Body

Defines the style for a message body area.

Selector

mod-info-body

Sample

```
.mod-info-body
{
  border: 1px solid #969696;
  background-color: #FFFFFF;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Message Header

Defines the style for a message header area.

Selector

```
.mod-info-header
```

Sample

```
.mod-info-header
{
  border: 1px solid #969696;
  background-color: #E5E5E5;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Message Footer

Defines the style for a message footer area.

Selector

```
.mod-info-footer
```

Sample

```
.mod-info-footer
{
  border: 1px solid #969696;
  background-color: #F3F3F3;
  color: #000000;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Links in Message Footer

Defines the style for links within a message footer area.

Selector

a.mod-info-footer-a

Sample

```
a.mod-info-footer-a
{
  color: #666666;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

"Paging" Group

Customize the appearance of bottom link bar (called *paging*) on modules' pages by editing the Paging group of rules.

In this section:

Links	62
Paging.....	63

Links

Defines the style for links within a paging.

Selector

a.mod-pager-a

Sample

```
a.mod-pager-a
{
  color: #969696;
  font-family: Arial, sans-serif;
  font-size: 8pt;
}
```

Paging

Defines the style for a paging.

Selector

```
.mod-pager
```

Sample

```
.mod-pager
{
  color: #969696;
  font-family: Arial, sans-serif;
  font-size: 8pt;
  border: 1px solid #969696;
  background-color: #ffffff;
}
```

"Status" Group

Customize the appearance of messages with the results of site visitors' actions (for example, messages about successful completion of a certain operation) by editing the Status group of rules.

In this section:

Error Message.....	64
Information Message.....	64
Success Message	65

Error Message

Defines the style for error messages.

Selector

```
.statuses-error
```

Sample

```
.statuses-error
{
  background-color: #FFF0F0;
  border: 2px solid #CC0303;
  padding: 7px 9px;
  font-size: 8pt;
  font-family: Arial, sans-serif;
  color: #CC0000;
}
```

Information Message

Defines the style for information messages.

Selector

```
.statuses-info
```

Sample

```
.statuses-info
{
  background-color: #E3EFFF;
  border: 2px solid #1C7CF1;
  padding: 7px 9px;
  font-size: 8pt;
  font-family: Arial, sans-serif;
  color: #3333CC;
}
```

Success Message

Defines the style for success messages.

Selector

`.statuses-success`

Sample

```
.statuses-success
{
  background-color: #E8FFE1;
  border: 2px solid #039A03;
  padding: 7px 9px;
  font-size: 8pt;
  font-family: Arial, sans-serif;
  color: #017001;
}
```

"Main" Group

Customize the appearance of text boxes and fields, horizontal rules, and buttons.

In this section:

Input Elements	65
Horizontal Rules.....	66
Buttons.....	66
Strong Buttons	66

Input Elements

Defines the style for all form input elements within a page.

Selector

`.mod-input`

Sample

```
.mod-input
{
  font-size: 8pt;
  font-family: Arial, sans-serif;
  color: #000000;
}
```

Horizontal Rules

Defines the style for all HR elements within a page.

Selector

```
.mod-hr
```

Sample

```
.mod-hr
{
    background-color: #CECECE;
}
```

Buttons

Defines the style for buttons within a page.

Selector

```
.mod-button
```

Sample

```
.mod-button
{
    color: #000000;
    font-family: Arial, sans-serif;
    font-size: 8pt;
}
```

Strong Buttons

Defines the style for strong buttons within a page.

Selector

```
.mod-button-strong
```

Sample

```
.mod-button-strong
{
    color: #000000;
    font-family: Arial, sans-serif;
    font-size: 8pt;
    font-weight: bold;
}
```

Customizing Dynamic Content of Modules

The guidelines provided in this section enable you to change presentation of template dynamic content. It is described by controls. For more information on controls, refer to the **Sitebuilder Controls and Variables** (on page 21) section. When the template source code is rendered by Sitebuilder, controls are substituted by HTML code that is generated according to rules defined in the plain-text `.skin` file of a control. To apply custom presentation of a control in a certain color scheme, place the corresponding `.skin` file into a corresponding color theme directory. For example, if you want to change the design of the `StatusBar` control in the green color scheme, place the `StatusBar.skin` file into the `Green` folder.

This section also provides the instructions on how to customize Sitebuilder Blog and Guestbook modules. In future versions of this guide, similar instructions for other modules will be given.

Note: Sitebuilder modules can be customized only in Sitebuilder for Windows.

In this section:

StatusBar Control.....	67
Paging Control	69
Form Control	71
Customizing Parameters Specific for Blog Module	73
Customizing Parameters Specific for Guestbook Module	82

StatusBar Control

`StatusBar` is a control responsible for the HTML layout of status messages on sites.

File Location

[Template path]/Themes/[Colour scheme name]/StatusBar.skin

Sample

The `StatusBar` control is a template of messages containing results of user actions. There are three action statuses: *error*, *success* and *info*.

Image and text displayed after successfully performed actions are described in the `SuccessTemplate`. The content of the element must contain HTML code of a success message. To provide user with more details on the action, use the following variables:

`DefaultSuccess`. Path to a default icon displayed on the message window. You can specify a different path.

`MessageType`. Message header.

`Message`. Text of a message.

`InfoTemplate` and `SuccessTemplate` structure is similar to the `SuccessTemplate` element's structure. Use `DefaultInfo` and `DefaultError` respectively for the elements' default icon paths.

```
<SiteBuilder:StatusBar>
  <ErrorTemplate>
    <div class="statuses-error">
      
      <b>$MessageType$</b>$Message$
    </div>
  </ErrorTemplate>
  <InfoTemplate>
    <div class="statuses-info">
      
      <b>$MessageType$</b>$Message$
    </div>
  </InfoTemplate>
  <SuccessTemplate>
    <div class="statuses-success">
      
      <b>$MessageType$</b>$Message$
    </div>
  </SuccessTemplate>
</SiteBuilder:StatusBar>
```

Strict Definition

ErrorTemplate element

Variables:

- `DefaultError` (*Optional*). Stores the path to a default image.
- `MessageType`. (*Optional*). Stores the message header.
- `Message`. Stores the message text.

InfoTemplate element

Variables:

- `DefaultError` (*Optional*). Stores the path to a default image.
- `MessageType` (*Optional*). Stores the message header.
- `Message`. Stores the message text.

SuccessTemplate element

Variables:

- `DefaultError` (*Optional*). Stores the path to a default image.
- `MessageType` (*Optional*). Stores the message header.
- `Message`. Stores the message text.

Paging Control

Paging control is responsible for HTML layout of bottom links bar.

File Location

[Template path]/Themes/[Colour scheme name]/Pager.skin

Sample

The Paging control is a template of a paging.

The control has two attributes: `PageNumber-class` and `CurrentPageNumber-class`. These attributes define the style of navigation links in active and inactive statuses respectively.

In the `Template` element content defines the paging presentation.

The `$TotalInfo$` text variable contains information about total number of pages.

The `Template` element content includes two kinds of controls:

- Link
- Container

Note: Starting from the 4 version of templates, you can customize HTML layout of Link controls. For more details, refer to the [Link \(see page 91\)](#) section.

The `Container` control with the `ID=PagesContainer` attribute defines the position of the paging.

All controls inside the `<Template>` tag are optional.

```
<SiteBuilder:Pager
  pagenumber-class="mod-pager-a">
  <Template>
    <table cellpadding="0" cellspacing="0" border="0" width="100%"
class="mod-pager" style="margin: 10px 0;">
      <tr>
        <td style="padding: 5px 10px;">$TotalInfo$</td>
        <td align="right" style="padding: 5px 10px;">
          <SiteBuilder:Link ID="FirstPage" class="mod-pager-a" />&nbsp;
          <SiteBuilder:Link ID="PreviousPage" class="mod-pager-a"
/>&nbsp;
          <SiteBuilder:Container ID="PagesContainer" />&nbsp;
          <SiteBuilder:Link ID="NextPage" class="mod-pager-a" />&nbsp;
          <SiteBuilder:Link ID="LastPage" class="mod-pager-a" />
        </td>
      </tr>
    </table>
  </Template>
</SiteBuilder:Pager>
```

Strict Definition

Template element

Variables:

- `TotalInfo` (*Optional*). Stores total number of pages.

Controls:

ID	Type	Is required	Description
PagesContainer	Container (see page 87)	Optional	Position of navigation links code
FirstPage	Link	Optional	Link that leads to the first page of the list
PreviousPage	Link	Optional	Link to the previous page of the list
NextPage	Link	Optional	Link to the next page of the list
LastPage	Link	Optional	Link to the last page of the list

Pager control attributes

- `PageNumber`. Name of CSS class applied to the inactive navigation links.
- `CurrentPageNumber`. Name of CSS class applied to active navigation links.

Form Control

Form control is responsible for HTML layout of input fields on a page. The example of this control you can see on "Login" and "Feedback" pages.

File Location

`[Template path] / Themes / [Colour scheme name] / Form.skin`

Sample

The Form control is a template of a form.

The `HeaderTemplate` element is responsible for form header HTML layout. You can define a form header style by specifying a proper class selector as the `mod-form` attribute value.

To configure HTML layout of form fields, use the `FieldTemplate` and `AlternatingFieldTemplate` elements. These tags are used to modify HTML layout of even and odd fields respectively. For example, you can set different backgrounds for even and odd fields.

The `$_Caption$` variable contains the description of a field. The `ValidationText` control with attribute `ID=IsRequired` defines a place where the "is required" asterisk appears on the screen. You can set `class` or `style` attributes for this control. Also, you can change the way the validation messages appear on the page: if you set `Display="Static"`, the form page will not be resized when the validation message is displayed. If you set `Display="Dynamic"`, the page will be automatically resized when the message is displayed.

The `FooterTemplate` element is responsible for form footer HTML layout.

```
<SiteBuilder:Form input-class="mod-input">
  <HeaderTemplate>
    <table cellpadding="5" cellspacing="0" width="100%" class="mod-
form" style="margin: 10px 0; border: 0;">
  </HeaderTemplate>
  <FieldTemplate>
    <tr>
      <td width="30%">$_Caption$<SiteBuilder:ValidationText
Display="Static" ID="IsRequired" /></td>
      <td width="70%"><SiteBuilder:Container ID="InputHolder" /></td>
    </tr>
  </FieldTemplate>
  <AlternatingFieldTemplate>
    <tr>
      <td width="30%">
        $_Caption$<SiteBuilder:ValidationText Display="Static"
ID="IsRequired" />
      </td>
      <td width="70%"><SiteBuilder:Container ID="InputHolder" /></td>
    </tr>
  </AlternatingFieldTemplate>
  <FooterTemplate>
</table>
</FooterTemplate>
</SiteBuilder:Form>
```

Strict Definition

HeaderTemplate element

No special variables or controls.

FieldTemplate, AlternatingFieldTemplate elements

Variables:

- `Caption`. Stores text description of the form fields. Can be wrapped into `TextDiv` (see page 87).

Controls:

ID	Type	Is required	Description
IsRequired	ValidationText (see page 91)	Required	Defines if the “is required” asterisk is set for a form field
InputHolder	Container (see page 87)	Required	Position of input controls code

FooterTemplate element

No special variables or controls.

Form control attributes:

- `input-class` – it is a name of the CSS class applied to the input controls.

Customizing Parameters Specific for Blog Module

You can change HTML layout of pages specific for the Blog module, such as page listing all blog posts and page displaying content of a certain post.

Note: The Blog module can be customized only in Sitebuilder 3.2.1 for Windows and later versions of Sitebuilder for Windows.

List of Blog Posts

To create HTML layout of a Blog page displaying a list of posts, the `ListPosts.page` file is used. Modify this file to create a special Blog design.

File Location

[Template path] / Themes / *[Colour scheme name]* / Blog / ListPosts.page

Sample

```

<!-- Category description block - white div with padded text. If no
category chosed - all div will be hidden.
Use simple $CategoryDescription$ variable if you need just output
text. -->
<SiteBuilder:TextDiv ID="CategoryDescription" class="mod-category-
body" style="padding: 10px; margin: 10px 0 0 0;"/>
<SiteBuilder:List ID="EntryList">
  <ItemTemplate>
    <table cellpadding="0" cellspacing="0" border="0" width="100%"
style="border-collapse: collapse; margin-top: 10px;">
      <tr>
        <td class="mod-item-header" style="padding: 5px 10px;">
          <div style="float:left;">
            <b>$Title$</b>
          </div>
          <div style="float:right;">$Time$</div>
        </td>
      </tr>
      <tr>
        <td class="mod-item-body" style="padding: 5px 10px;">
          <div id="truncationEnvelope">$Entry$</div>
          <div align="right">
            <!-- this link is visible if entry is too big -->
            <SiteBuilder:Link ID="ReadMore" class="mod-item-body-a-
strong" />&nbsp;
            <SiteBuilder:Link ID="ToComments" class="mod-item-body-a-
strong" />
          </div>
        </td>
      </tr>
    </table>
  </ItemTemplate>
  <AlternatingItemTemplate>
    <table cellpadding="0" cellspacing="0" border="0" width="100%"
style="border-collapse: collapse; margin-top: 10px;">
      <tr>
        <td class="mod-item-header" style="padding: 5px 10px;">
          <div style="float:left;">
            <b>$Title$</b>
          </div>
          <div style="float:right;">$Time$</div>
        </td>
      </tr>
      <tr>
        <td class="mod-item-body-alter" style="padding: 5px 10px;">
          <div id="truncationEnvelope">$Entry$</div>
          <div align="right">
            <!-- this link is visible if entry is too big -->
            <SiteBuilder:Link ID="ReadMore" class="mod-item-body-a-
strong" />&nbsp;
            <SiteBuilder:Link ID="ToComments" class="mod-item-body-a-
strong" />
          </div>
        </td>
      </tr>
    </table>
  </AlternatingItemTemplate>

```

```

</AlternatingItemTemplate>
</SiteBuilder:List>
<!-- Here Pager for EntryList will be displayed. Customize Pager you
can in Pager.skin file. -->
<SiteBuilder:Container ID="Pager" />
<div align="right">
  <!-- visible when user navigates through months and years -->
  <SiteBuilder:Link ID="BackLink" />
</div>

```

This sample shows default layout of the `ListPosts.page` file.

The `CategoryDescription` variable stores description of a Blog module category. It is wrapped inside the `TextDiv` control. For details on the control, refer to the **TextDiv** section (see page 87).

The `List` control with the `ID=EntryList` attribute defines a blog post entries template.

`ItemTemplate` element defines presentation of a blog post.

`AlternatingItemTemplate` element defines presentation of even blog posts.

Variables `Time`, `Title` and `Entry` that represent post time, post title and post body can reside inside the `ItemTemplate` and `AlternatingItemTemplate` elements.

In addition, you can customize `Link` controls with IDs `ReadMore` and `ToComments`. The controls define the position of links to "full posts list" and "post comments" pages (correspondingly).

`Container` (see page 87) with ID `Pager` reserves place for a paging control. Refer to the **Paging Control** section (on page 69) for details on how to customize the control.

`Link` with ID `BackLink` defines the position of a link to posts archive.

Strict Definition

ListPosts.page file

Variables:

- `CategoryDescription`. The description of a selected category. Can be wrapped into `TextDiv` (see page 87).

Controls:

ID	Type	Is required	Description
EntryList	List (see page 88)	Required	Template of a blog post entry

Pager	Container (see page 87)	Required	Position of paging code
BackLink	Link	Required	Position of link to posts archive
RSSLink	Link	Required	Position of link to blog posts rss feed

ItemTemplate, AlternatingItemTemplate elements

Variables

- `Time`. Time when an entry was posted. Can be wrapped into `TextDiv` (see page 87).
- `Title`. Title of the post entry. Can be wrapped into `TextDiv` (see page 87).
- `Entry`. Body of the post entry. Can be wrapped into `TextDiv` (see page 87).

Controls:

ID	Type	Is required	Description
ReadMore	Link	Required	Link to list of all posts
ToComments	Link	Required	Link to comments on a blog entry

Post Comments

The `ShowPost.page` file defines the layout of a page that displays content of a certain blog post, including list of comments to the post and form for posting new comments.

File Location

[Template path] / Themes / *[Colour scheme name]* / Blog / ShowPost.page

Sample

```

<table cellpadding="0" cellspacing="0" border="0" width="100%"
class="mod-item-body" style="padding: 10px; margin-top: 10px;">
  <tr>
    <td colspan="2">
      <div class="mod-item-body-title" style="margin-bottom: 10px;">
        <b>${PostedOn}</b>
      </div>
      <b>${Entry}</b>
    </td>
  </tr>
  <tr>
    <td colspan="2" style="height: 10px;">
      <div class="mod-item-body-hr" style="height: 1px;">
        <span></span>
      </div>
    </td>
  </tr>
  <tr>
    <td>
      <SiteBuilder:Link class="mod-item-body-a" ID="BackLink" />
    </td>
    <td align="right">
      <SiteBuilder:Link id="AddCommentLink" class="mod-item-body-a-
strong" style="cursor: hand;" />
    </td>
  </tr>
</table>
<!-- Anchor for quick navigation to list of blog entry comments -->
<a name="Comments" />
<SiteBuilder:List ID="CommentList" style="width: 100%;">
  <ItemTemplate>
    <table cellpadding="0" cellspacing="0" border="0" width="100%"
style="border-collapse: collapse; margin-top: 10px;">
      <tr>
        <td class="mod-comment-header" style="padding: 5px 10px;">
          <div style="float:left;">
            <b>${Author}</b>
          </div>
          <div style="float:right;">${Time}</div>
        </td>
      </tr>
      <tr>
        <td class="mod-comment-body" style="padding: 5px 10px;">
          <div class="mod-comment-body-title" style="margin-bottom:
10px;">
            <b>${Title}</b>
          </div>
          <b>${Comment}</b>
        </td>
      </tr>
    </table>
  </ItemTemplate>
  <AlternatingItemTemplate>
    <table cellpadding="0" cellspacing="0" border="0" width="100%"
style="border-collapse: collapse; margin-top: 10px;">
      <tr>

```

```

        <td class="mod-comment-header" style="padding: 5px 10px;">
            <div style="float:left;">
                <b>$Author$</b>
            </div>
            <div style="float:right;">$Time$</div>
        </td>
    </tr>
    <tr>
        <td class="mod-comment-body-alter" style="padding: 5px
10px;">
            <div class="mod-comment-body-title" style="margin-bottom:
10px;">
                <b>$Title$</b>
            </div>
            $Comment$
        </td>
    </tr>
</table>
</AlternatingItemTemplate>
</SiteBuilder:List>
<!-- Here Pager for CommentList will be displayed. Customize Pager you
can in Pager.skin file. -->
<SiteBuilder:Container ID="Pager" />
<!-- Anchor for quick navigation to 'add new comment' form -->
<a name="AddComment" />
<table cellpadding="10" cellspacing="0" border="0" width="100%"
class="mod-form" style="margin: 10px 0;">
    <tr>
        <td colspan="2" class="mod-form-title">
            <b>$AddCommentTitle$</b>
        </td>
    </tr>
    <tr>
        <td colspan="2" style="padding: 0 10px;">
            <div class="mod-form-hr" style="height:1px;">
                <span></span>
            </div>
        </td>
    </tr>
    <tr>
        <td width="50%">
            $AuthorCaption$
            <SiteBuilder:ValidationText Display="Static"
ID="AuthorIsRequired" /><br />
            <SiteBuilder:TextInput ID="AuthorInput" class="mod-input"
style="width: 100%;" />
        </td>
        <td width="50%">
            $SubjectCaption$
            <SiteBuilder:ValidationText Display="Static"
ID="SubjectIsRequired" /><br />
            <SiteBuilder:TextInput ID="SubjectInput" class="mod-input"
style="width: 100%;" />
        </td>
    </tr>
    <tr>
        <td colspan="2" style="padding: 0 10px;">
            $MessageCaption$
            <SiteBuilder:ValidationText Display="Static"
ID="MessageIsRequired" /><br />

```

```
<SiteBuilder:TextInput ID="MessageInput" class="mod-input"
style="width: 100%; height: 87px;" />
</td>
</tr>
<tr>
<td colspan="2" align="right">
<SiteBuilder:Button ID="AddComment" class="mod-form-button" />
</td>
</tr>
</table>
```

This sample shows default layout of the `ShowPost.page` file.

The first table section content contains a Blog entry template.

The table is followed by a **List** control (see page 88) with ID `CommentList` that defines the template for comments on an entry. The **Paging** control defines the position of a paging for this list. For details on how to customize the control, refer to the **Paging Control** section (on page 69).

The paging is followed by a form for adding a new comment. This form is divided into separate controls so you can provide much more flexible layout than with `Form` control. The **Form** (see page 71) control is still required for pages with variable number of fields. For example, the Feedback module can be described only with the `Form` control.

Variable `AddCommentTitle` stores the form name. The form includes the following elements: *Author of comment*, *Subject and Message fields* and **Button** (see page 91) with ID `AddComment`. Each field name is defined by a variable with suffix `Caption` (for example, `AuthorCaption`)

Control **TextInput** (see page 90) displays HTML form input fields. Control **ValidationText** (see page 91) displays a validation message if a user entered invalid data into a field.

Strict Definition

ShowPost.page file

Variables:

- `Entry`. A Blog entry. Can be wrapped into **TextDiv** (see page 87).
- `PostedOn`. Time when the entry was posted. Can be wrapped into **TextDiv** (see page 87).
- `AddCommentTitle`. (*Optional*) Name of a form. Can be wrapped into **TextDiv** (see page 87).
- `AuthorCaption`. (*Optional*) Caption for the author input field. Can be wrapped into **TextDiv** (see page 87).
- `SubjectCaption`. (*Optional*) Caption for the comment subject input field. Can be wrapped into **TextDiv** (see page 87).
- `MessageCaption`. (*Optional*) Caption for the comment input field. Can be wrapped into **TextDiv** (see page 87).

Controls:

ID	Type	Is required	Description
BackLink	Link	Required	Link to list of Blog entries
AddCommentLink	Link	Required	Link to comments on a Blog entry
CommentList	List (see page 88)	Required	Template for comments on a Blog entry
Pager	Container (see page 87)	Required	Position of a paging code
AuthorInput	TextInput (see page 90)	Required	Input field "author of comment"
AuthorIsRequired	ValidationText (see page 91)	Required	Validation text for input field "author of comment"
SubjectInput	TextInput (see page 90)	Required	Input field "subject of comment"
SubjectIsRequired	ValidationText (see page 91)	Required	Validation text for input field "subject of comment"
MessageInput	TextInput (see page 90)	Required	Input field "body of comment"
MessageIsRequired	ValidationText (see page 91)	Required	Validation text for input field "body of comment"
Captcha	Container (see page 87)	Optional	CAPTCHA image and input field
AddComment	Button (see page 91)	Required	Button for submitting form.

ItemTemplate, AlternatingItemTemplate elements

Variables

- `Author`. Author of a comment. Can be wrapped into `TextDiv` (see page 87).
- `Time`. Posted time of a comment. Can be wrapped into `TextDiv` (see page 87).
- `Title`. Subject of a comment. Can be wrapped into `TextDiv` (see page 87).
- `Comment`. Body of a comment. Can be wrapped into `TextDiv` (see page 87).

Customizing Parameters Specific for Guestbook Module

You can change HTML layout of pages specific for the Guestbook module. For instance, you can customize page listing all guestbook messages and page showing the form for composing a new message.

Note: The Guestbook module can be customized only in Sitebuilder 3.2.1 for Windows and later versions of Sitebuilder for Windows.

File Location

[Template path] / Themes / *[Colour scheme name]* / Guestbook /
MessageList.page

Sample

```

<!-- Here StatusBar will be displayed. Customize it you can in
StatusBar.skin file. -->
<SiteBuilder:Container ID="StatusBar" />
<SiteBuilder:List ID="MessageList">
  <ItemTemplate>
    <table cellpadding="0" cellspacing="0" border="0" width="100%"
style="border-collapse: collapse; margin-top: 10px;">
      <tr>
        <td class="mod-item-header" style="padding: 5px 10px;">
          <div style="float:left;">
            <b>${Author}</b>
            <!-- This link is visible when user, that left message,
was logged in -->
            <SiteBuilder:Link ID="MailTo" style="margin-
left:10px;"></SiteBuilder:Link>
            <!-- This link is visible when user, that left message,
was logged in and filled his homepage url -->
            <SiteBuilder:Link ID="HomePage" style="margin-
left:10px;"></SiteBuilder:Link>
          </div>
          <div style="float:right;">${Time$}</div>
        </td>
      </tr>
      <tr>
        <td class="mod-item-body" style="padding: 5px
10px;">${Message$}</td>
      </tr>
    </table>
  </ItemTemplate>
  <AlternatingItemTemplate>
    <table cellpadding="0" cellspacing="0" border="0" width="100%"
style="border-collapse: collapse; margin-top: 10px;">
      <tr>
        <td class="mod-item-header" style="padding: 5px 10px;">
          <div style="float:left;">
            <b>${Author}</b>
            <!-- This link is visible when user, that left message,
was logged in -->
            <SiteBuilder:Link ID="MailTo" style="margin-left:
10px;"></SiteBuilder:Link>
            <!-- This link is visible when user, that left message,
was logged in and filled his homepage url -->
            <SiteBuilder:Link ID="HomePage" style="margin-left:
10px;"></SiteBuilder:Link>
          </div>
          <div style="float:right;">${Time$}</div>
        </td>
      </tr>
      <tr>
        <td class="mod-item-body-alter" style="padding: 5px
10px;">${Message$}</td>
      </tr>
    </table>

```

```

</AlternatingItemTemplate>
</SiteBuilder:List>
<!-- Here Pager for MessageList will be displayed. Customize Pager you
can in Pager.skin file. -->
<SiteBuilder:Container ID="Pager" />
<table cellpadding="10" cellspacing="0" border="0" width="100%"
class="mod-form" style="margin: 10px 0;">
  <tr>
    <td class="mod-form-title">
      <b>${AddMessageTitle}</b>
    </td>
  </tr>
  <tr>
    <td style="padding: 0 10px;">
      <div class="mod-form-hr" style="height:1px;">
        <span></span>
      </div>
    </td>
  </tr>
  <tr>
    <td>
      <SiteBuilder:ValidationText Display="Static"
ID="AuthorIsRequired" />
      <br/>
      <SiteBuilder:TextInput ID="AuthorInput" class="mod-input"
style="width: 100%" />
    </td>
  </tr>
  <tr>
    <td style="padding: 0 10px;">
      <SiteBuilder:ValidationText Display="Static"
ID="MessageIsRequired" />
      <br />
      <SiteBuilder:TextInput ID="MessageInput" class="mod-input"
style="width: 100%; height: 87px;" />
    </td>
  </tr>
  <tr>
    <td align="right">
      <SiteBuilder:Button ID="AddMessage" class="mod-form-button" />
    </td>
  </tr>
</table>

```

This is a default layout of the Guestbook module. **Container** (see page 87) `StatusBar` reserves the place for server messages. The next item is a template of a guestbook message. The template is followed by a messages list paging and form for adding a guestbook message. The form includes fields (*Author* and *Message*) and a submit button.

Strict Definition

MessageList.page file

Variables:

- `AddMessageTitle`. (Optional) Form title. Can be wrapped into **TextDiv** (see page 87).
- `AuthorCaption`. (Optional) Caption for the author input field. Can be wrapped into **TextDiv** (see page 87).
- `MessageCaption`. (Optional) Caption for the comment input field. Can be wrapped into **TextDiv** (see page 87).

Controls:

ID	Type	Is required	Description
StatusBar	Container (see page 87)	Required	Position to a status bar code
MessageList	List (see page 88)	Required	Template of a guestbook message.

Pager	Container (see page 87)	Required	Position of paging code
AuthorInput	TextInput (see page 90)	Required	Input field "author of comment"
AuthorIsRequired	ValidationText (see page 91)	Required	Validation text for input field "author of comment"
MessageInput	TextInput (see page 90)	Required	Input field "subject of comment"
MessageIsRequired	ValidationText (see page 91)	Required	Validation text for input field "subject of comment"
Captcha	Container (see page 87)	Optional	CAPTCHA image and input field
AddMessage	ValidationText (see page 91)	Required	Button for submitting a form.

ItemTemplate, AlternatingItemTemplate elements

Variables:

- `Message`. Body of a message. Can be wrapped into **TextDiv** (see page 87).
- `Time`. Time when the message was posted. Can be wrapped into **TextDiv** (see page 87).
- `Author`. Author of the message. Can be wrapped into **TextDiv** (see page 87).

Controls:

ID	Type	Is required	Description
MailTo	Link	Required	Mailto-link if a message was posted by an authorized user
HomePage	Link	Required	Link to user home page if a message was posted by an authorized user and he entered his homepage URL in the profile

Appendix I. Controls Reference

This chapter contains full description of controls mentioned in the **Customizing HTML Code of Sitebuilder Modules** (see page 67) section.

In this chapter:

Container	87
TextDiv	87
List	88
Link	90
TextInput	90
ValidationText	91
Button.....	91

Container

This control is intended to insert HTML code of a control specified in the ID attribute to a specific place in HTML code of a page.

For details on how to customize HTML code of a control specified in the ID attribute, refer to the **Customizing Presentation of Controls** (on page 67) section.

Sample

```
<SiteBuilder:Container ID="Pager" />
```

TextDiv

This control is used to display a variable value in an HTML layer. This layer is hidden when the value is empty.

Attributes

The control has the following attributes:

Name	Value
ID	Variable name
class	SCC class selector applied to the layer
style	Element defining the layer style

Variables

Text. Defines place in HTML code where the value of a variable specified by the ID argument is inserted.

Sample

To create a layer containing variable `$CategoryName$`, you can use the following code:

```
<SiteBuilder:TextDiv ID="CategoryName" class="category-block" style="padding: 10px; margin: 10px 0 0 0;"/>
```

For instance, `$CategoryName$` is equal to *"My category"*. After processing by Sitebuilder, the page will contain the following code instead of the control:

```
<div class="category-block" style="padding: 10px; margin: 10px 0 0 0;">My category</div>
```

If `$CategoryName$` is equal to `""`, the layer will not be created. It means that Sitebuilder processor simply removes the control when rendering HTML code.

The following example shows the `$Text$` variable usage. The following code:

```
<SiteBuilder:TextDiv ID="CategoryName">
  <table>
    <tr><td>$Text$</td></tr>
  </table>
</SiteBuilder:TextDiv>
```

will look as follows after rendering:

```
<div>
  <table>
    <tr><td>My category</td></tr>
  </table>
</div>
```

Note: Some variables cannot be wrapped into a layer. For instance, `Url` variable of the `SiteMenu` control.

List

This control is used to create layout for a complex object that includes header, one or more similar items and footer. For instance, a list of Blog comments is presented using this control.

The list control is rendered by Sitebuilder according to the following schema:

- 1 The code nested inside the `HeaderTemplate` is inserted into HTML code of a page instead of the list control.
- 2 For each list item (depending on its evenness), the code nested inside the `ItemTemplate` or `AlternatingItemTemplate` elements is inserted.
- 3 The code nested inside the `FooterTemplate` is inserted into HTML code after the code fragment pertaining to a last list element code.

If the inserted code contained Sitebuilder variables, they are substituted by their values.

Control elements

Name	Value
<code>ItemTemplate</code>	Contains HTML code of a list item.
<code>AlternatingItemTemplate</code>	Contains HTML code of an even list item. If not defined, <code>ItemTemplate</code> content is used.
<code>SeparatorTemplate</code>	Contains HTML code of a separator between list items. If not defined, a separator is not displayed.
<code>HeaderTemplate</code>	Contains HTML code of a list header. If not defined, the header is not displayed.
<code>FooterTemplate</code>	Contains HTML code of a list footer. If not defined, the footer is not displayed.

Sample

Say, there is a blog with three comments: "First comment", "Second comment" and "Third comment". To create layout for the comments, add the following fragment to the HTML code of the blog page.

```
<SiteBuilder:List ID="EntryList">
  <ItemTemplate>
    <div>$Title$</div>
  </ItemTemplate>
</SiteBuilder:List>
```

After processing, the control will be substituted by the following HTML fragment:

```
<div>First comment</div>
<div>Second comment</div>
<div>Third comment</div>
```

For details on variables used in the sample, refer to the **List of Blog posts>Strict Definition (on page 76) section**.

Link

This control represents HTML anchor. It could be displayed as a simple text link or some image. By default, it is a text link.

Attributes

Name	Value
class	Name of Cascading Style Sheet (CSS) class applied to the link.
style	Style attribute for the control.

Variables

`Text`. Use it to specify the place in HTML code where the link caption is to be displayed.

Sample

The simplest way to describe the link with ID `SomeID`:

```
<SiteBuilder:Link ID="SomeID" class="top-link" />
```

To specify an arbitrary caption for the link, use the following fragment:

```
<SiteBuilder:Link ID="SomeID">
  $Text$
</SiteBuilder:Link>
```

TextInput

This control displays a form input field.

Attributes

Name	Value
class	Name of Cascading Style Sheet (CSS) class applied to HTML text input control.
style	Style attribute for the control.

Sample

```
<SiteBuilder:TextInput ID="AuthorInput" class="mod-input"
style="width: 100%" />
```

ValidationText

This control displays error message if a user enters invalid data into a form input field.

Attributes

Name	Value
Display	Can be one of the following values: if <code>Display="Static"</code> , a form page will not be resized when a validation message is displayed. If <code>Display="Dynamic"</code> , the page will be automatically resized when the message is displayed.
class	Name of Cascading Style Sheet (CSS) class applied to HTML text input control.
style	Style attribute for the control.

Sample

```
<SiteBuilder:ValidationText Display="Static" ID="AuthorIsRequired" />
```

Button

This control displays a button control on a page. It can be simple or link-styled.

Attributes

Name	Value
class	Name of Cascading Style Sheet (CSS) class applied to this control.
Type	Type of button control. Allowed values: Button or Link. Default is Button.
style	Style attribute for the control.

Variables

`Text` . Use this variable to specify where the button text will be displayed.

Sample

```
<SiteBuilder:Button ID="AddMessage" class="mod-form-button" />
```

To specify the place in HTML code where for the button text is to be displayed, use the following fragment:

```
<SiteBuilder:Button ID="AddMessage">  
  <table>  
    <tr><td>${Text$}</td></tr>  
  </table>  
</SiteBuilder:Button>
```