
SWsoft, Inc.

SWsoft SiteBuilder for Windows/Unix Template Creation Guide



ISBN: N/A
SWsoft, Inc.
13755 Sunrise Valley Drive
Suite 325
Herndon
VA 20171 USA
Phone: +1 (703) 815 5670
Fax: +1 (703) 815 5675

Copyright © 1999-2006 by SWsoft, 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.
Solaris is a registered trademark of Sun Microsystems, Inc.
X Window System is a registered trademark of X Consortium, Inc.
Intel, Pentium, and Celeron are registered trademarks of Intel Corporation.
MS Windows, Windows 2003 Server, Windows XP, Windows 2000, Windows NT are registered trademarks of Microsoft Corporation.
IBM DB2 is a registered trademark of International Business Machines Corp.
SSH and Secure Shell are trademarks of SSH Communications Security, Inc.
MegaRAID is a registered trademark of American Megatrends, Inc.
PowerEdge is a trademark of Dell Computer Corporation.
Request Tracker is a trademark of Best Practical Solutions, LLC
ActiveSite Compiler v4 is a trademark of Intorel, Inc.
XML-RPC Library under MIT License

Contents

Preface	6
Documentation Conventions.....	6
Typographical Conventions.....	6
Feedback.....	7
Introduction	8
Creating Design	9
Header.....	11
Top-Level Menu.....	12
Lower-Level Menu.....	13
Content Area.....	14
Footer.....	15
General Requirements.....	15
Variations.....	15
Converting Design to HTML	16
Requirements for Templates.....	17
Converting HTML Code to SiteBuilder Format.....	18
Creating info.xml File.....	19
Creating Base Document Template Master.page.....	21
Creating Menus.....	24
Creating Modules Design	26
Modules.css.....	27
Input Controls.....	28
Horizontal Rules.....	28
Buttons.....	28
Strong Buttons.....	29
Item Body Blocks.....	29
Alternating Item Body Blocks.....	30
Strong Links in Item Body Blocks.....	30
Custom Links in Any Item Body Blocks.....	31
Titles in Item Body Blocks.....	31
Horizontal Rules in Item Body Blocks.....	32
Buttons in Items Body Blocks.....	32
Strong Buttons in Item Body Blocks.....	33
Highlighted Text in Item Body Blocks.....	33
Item Header Blocks.....	34
Item Footer Blocks.....	34
Link in Item Footer Blocks.....	35
Category Body Blocks.....	35
Alternating Category Body Blocks.....	36
Custom Links in any Category Body Blocks.....	36
Title in Any Category Body Blocks.....	37

Category Header Blocks.....	37
Links in Category Header Blocks.....	38
Comment Body Blocks.....	38
Alternating Comment Body Blocks.....	39
Custom Links in Any Comment Body Blocks	39
Titles in Comment Body Blocks	40
Comment Header Blocks.....	40
Links in Comment Header Blocks.....	41
Search Form Blocks	41
Buttons in Search Form Blocks	42
Form Blocks	42
Titles in Form Blocks	43
Horizontal Rules in Form Blocks	43
Buttons in Form Blocks.....	43
Links in Form Blocks	44
Information Blocks.....	44
Information Header Blocks	45
Information Footer Blocks.....	45
Links Information Footer Blocks.....	46
Error Message Blocks.....	46
Information Message Blocks.....	47
Successful Message Blocks	47
Pager Blocks.....	48
Links in Pager Blocks.....	48
Additional Abilities	49
StatusBar	49
Pager.....	51
Form	53
Blog: ListPosts Page.....	55
Blog: ShowPost Page	58
Guestbook: MessageList Page.....	64

Changing Path to Image Files 69

Making Thumbnails 70

Making Thumbnails for Templates.....	70
Making Thumbnails for Menus	71
Making Thumbnails for Headers	71

Previewing Templates 72

Compiling Template into Installable Package 73

Installing Template Package 74

Appendix A 75

Container	75
TextDiv	76
List.....	77
Link.....	79

TextInput	80
ValidationText	80
Button	81

CHAPTER 1

Preface

In This Chapter

Documentation Conventions.....	6
Typographical Conventions	6
Feedback	7

Documentation Conventions

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

Typographical Conventions

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 http://docs/common/licenses 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 spot a typo in this guide, or if you have thought of a way to make this guide better, we would love to hear from you!

If you have a suggestion for improving the documentation (or any other relevant comments), try to be as specific as possible when formulating it. If you have found an error, please include the chapter/section/subsection name and some of the surrounding text so that we could find it easily.

Please submit a report by e-mail to userdocs@swsoft.com.

CHAPTER 2

Introduction

Template is a web page design created according to a certain standard and converted into the SiteBuilder format. This format is applicable to SiteBuilder 3.2 for Windows and SiteBuilder 3.0 for Unix.

By default, SiteBuilder is shipped with a certain number of templates. If you want to increase the list of available templates, you can create your own ones and import them to SiteBuilder.

This document provides detailed description, along with examples, of the process of creating new templates and transporting them to SiteBuilder.

The process of creating SiteBuilder templates is divided into five steps:

- 1** Creating template design
- 2** Converting the design to HTML code
- 3** Customizing modules styles
- 4** Creating template thumbnail
- 5** Converting HTML code into the SiteBuilder format

CHAPTER 3

Creating Design

The recommended software for creating template design is *Adobe Photoshop*.

Template layout consists of the following parts:

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

Footer Example:



Figure 1: Component Parts of a Template



Figure 2: Component Parts of a Template

In This Chapter

- Header 11
- Top-Level Menu 12
- Lower-Level Menu 13
- Content Area 14
- Footer 15
- General Requirements 15
- Variations 15

Header

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

- Header banner (image)
- Company logo
- Company name
- Company slogan
- Company URL

Note that it is recommended to place logo and name close to each other. The slogan should be placed under the company's name.

Example:

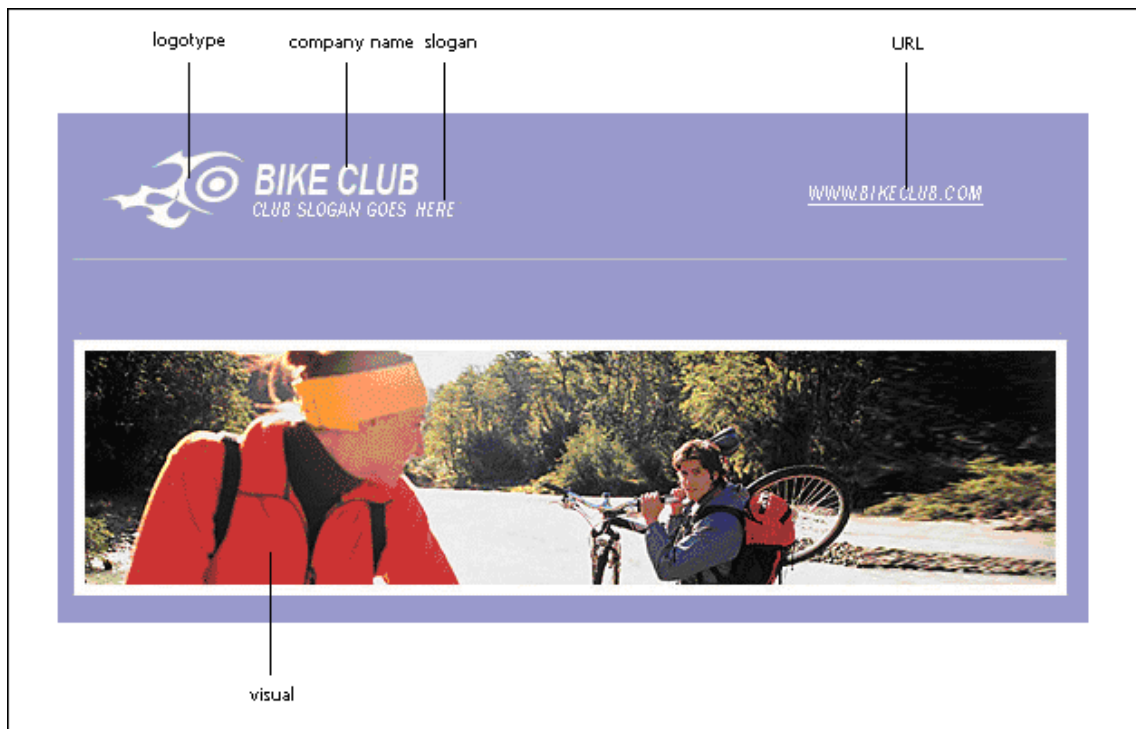


Figure 3: Header

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:

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

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

It is recommended to place the main menu area under the header. Besides, the page design should provide place for adding main menu items.

Example:

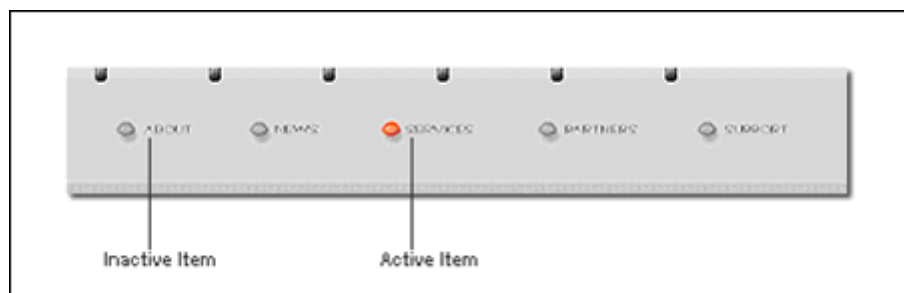


Figure 4: First-level menu

Lower-Level Menu

Lower-level menu (submenu) is a list of links to subpages of a section which was chosen from the main menu. Like the main menu, lower-level menu contains both active and inactive items. A lower-level menu can be created only for those site sections which have subpages.

Example:

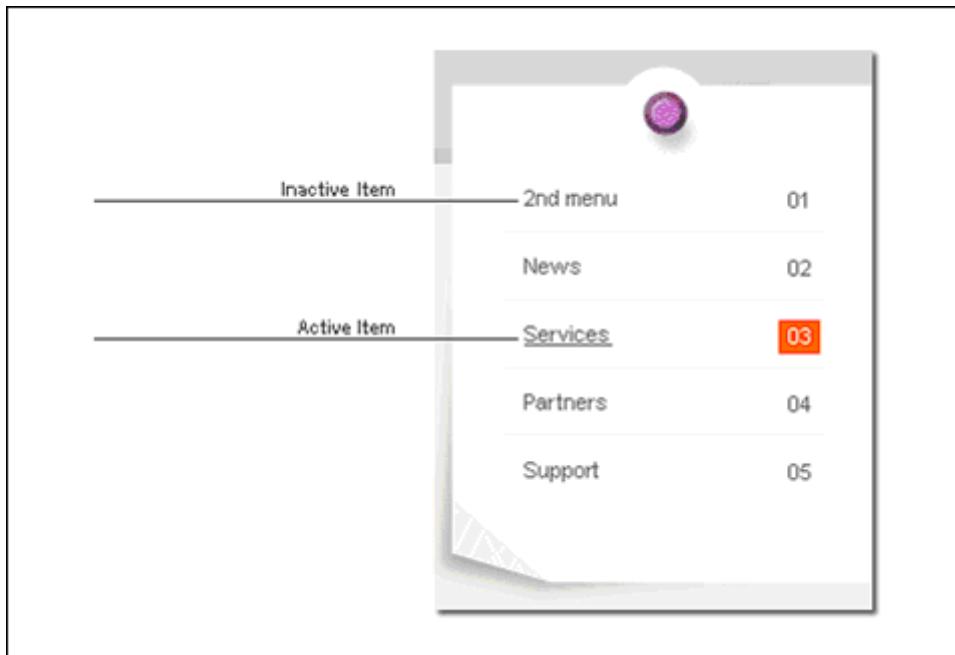


Figure 5: Second-level navigation bar

Content Area

Content area contains the principal substance (such as text, illustrations, etc.) of a site. 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).

Example:

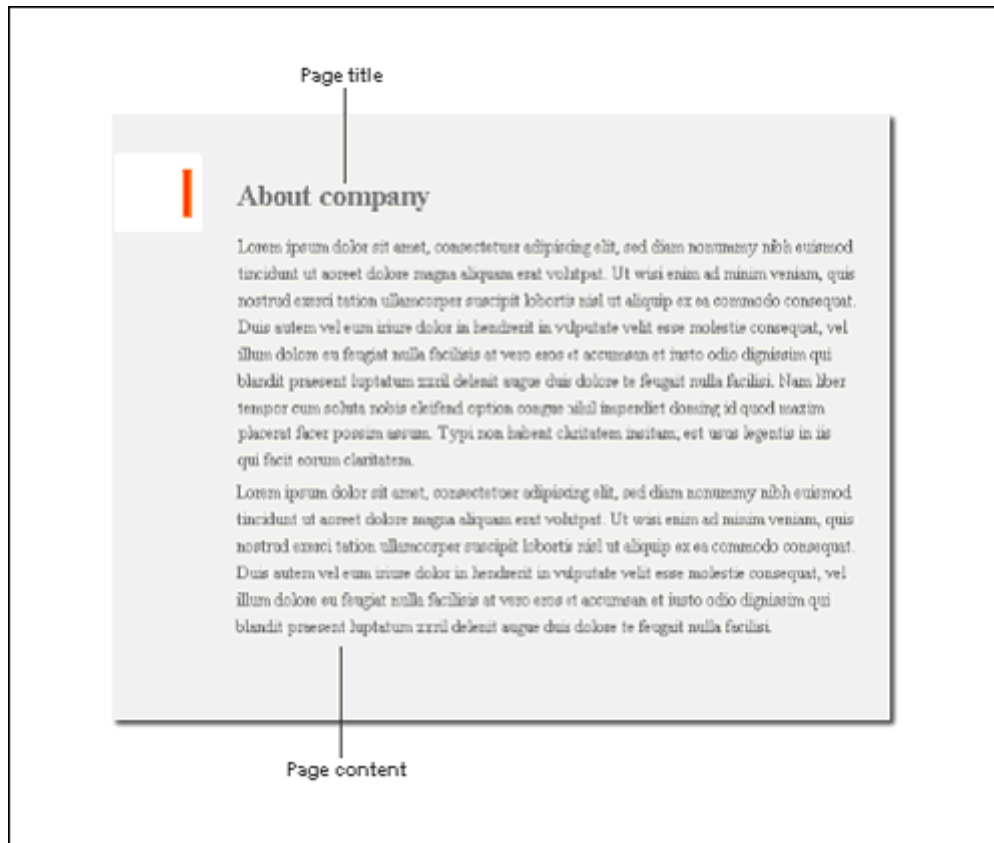


Figure 6: Content area

Footer

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

«Copyright © 2004, Company Name. All rights reserved.»

Footer may also duplicate the top-level menu.

Example:

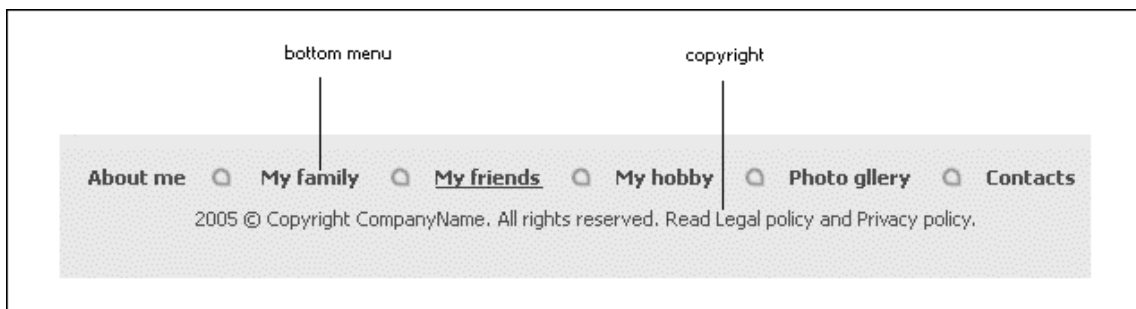


Figure 7: Footer

General Requirements

There are some technical requirements applied to templates design:

- Templates should be stretchable. That is a designer should provide the possibility of stretching a template both horizontally and vertically.
- The font for the company name, slogan, page title elements, top and lower level menus should be one of the standard "HTML fonts" from the system set: Tahoma, Arial, Helvetica, Times, Verdana, Sans.

Variations

By default, all the templates have three color themes, three header styles, and three top-level menu styles. In case of need, header and menu styles can be adapted for every color theme.

CHAPTER 4

Converting Design to HTML

When converting a template design into HTML code, pay special attention to the structure of the directories. The sample of a correct structure is given below:

```
Template
  Themes
    Color1
      Menus
        Menu1
        Menu2
        Menu3
      Headers
        Header1
        Header2
        Header3
      Images
    Color2
      Menus
        Menu1
        Menu2
        Menu3
      Headers
        Header1
        Header2
        Header3
      Images
    Color3
      Menus
        Menu1
        Menu2
        Menu3
      Headers
        Header1
        Header2
        Header3
      Images
```

The `Template` directory is the root template directory.

The `Themes` directory contains all color themes (`Color1`, `Color2`, `Color3`). By default, there are three color themes, but you can vary this number. Also, you can rename the color themes (for example, red, blue, green).

The `Menus` directory is divided into the subdirectories `Menu1`, `Menu2`, `Menu3`. Similar to color themes, there could be one or more menu themes. These directories contain the graphic files with the menu items images (usually, these are active and inactive bullets).

The `Headers` directory is divided into the subdirectories `Header1`, `Header2`, `Header3`. By default, there are three variants of design, but this number is not fixed. These directories contain the graphic files with the headers images.

The `Images` directory contains the graphic files with the images specific to a corresponding color theme. The default logo template is stored in this directory under the name of `logo.gif`. (it should be in GIF format).

The color theme directory contains the `styles.css` file. `styles.css` is a file containing CSS styles of a template. It should contain only two selectors: `pageContent` and `pageContent a`.

Example:

```
pageContent { font-size: 8pt; font-family: Tahoma, sans-serif; color: #818181; }
pageContent a { font-size: 8pt; font-family: Tahoma, sans-serif; color: #818181; }
```

This class must be applied to the content area.

In This Chapter

Requirements for Templates	17
Converting HTML Code to SiteBuilder Format	18
Creating info.xml File	19
Creating Base Document Template Master.page	21
Creating Menus	24

Requirements for Templates

- 1 Exterior**
 1. Template should have background – all visible space should be filled up with a color (including white).
 2. Active and inactive items of the top-level menu should be different.
- 2 Liquid**
 1. All templates should be liquid (occupy 100% of width and height of screen).
 2. All templates should stretch within the content area.
- 3 CSS**
 1. Fonts sizes should be in points (for example: `font-size: 8pt;`).
 2. HTML code should not have absolute positioning elements and scripts.
 3. `styles.css` should not have styles for tags and styles for id (for example: `a{...}` or `#item {...}`) – only classes.
 4. A template should not have negative margins and padding.
- 4 Compatibility**
 1. A converted HTML template should look as similar to the original layout as possible in all popular browsers: Microsoft Internet Explorer, Opera, Mozilla, FireFox.

Converting HTML Code to SiteBuilder Format

After you complete converting a design to HTML code, you should transfer it to the SiteBuilder format.

Creating info.xml File

The `info.xml` file, which contains important technical information about the template, should be placed into the Template folder.

The structure of the `info.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<template id="yourCompany-business-001" version="1.0"
caption="Business site" keywords=" business books" category="Business"
>
  <themes default="yellow">
    <theme id="green" caption="Green theme"/>
    <theme id="yellow" caption="Yellow theme"/>
    <theme id="orange" caption="Orange theme"/>
  </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
- Version
- Caption
- Category
- Keywords

The `id` attribute contains the name of a template. The name of a template must start with the name of your company, 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.

The `version` attribute contains the template version.

Version number consists of 2 digits:

- First (1) – the number of times a template was recoded with the same design.
- Second (0) – the number of hot fixes.

The `caption` attribute contains the description of a template. This can be any text of your choice, e.g. "Travel company".

The `keywords` attribute contains the list of keywords of a template. This can be any text of your choice, e.g. "business books". This attribute is optional.

The `category` attribute specifies the category a template belongs to, e.g. "Business". It may contain blank value or be absent.

The `template` element has child elements. A child element of the `themes` element is an element that has the same number as a color theme in the template.

The theme elements should contain two attributes:

- `id`
- `caption`

The `caption` attribute contains a description of the color theme. This can be any text of your choice, e.g. "Green color theme".

The `template` element has a child element - `compatibility`.

The child element of the `compatibility` is `screen`, and its child element is `minwidth`, which contains minimal width of a template without horizontal scrolling.

The `compatibility` element has child element - `browsers`. The `browsers` element contains the list of browsers compatible to the template, and their versions.

The `compatibility` element has child element - `modules`. The `modules` element contains the list of modules supported in this template.

List of possible values:

- Blog
- eShop
- Forum
- ImageGallery
- Guestbook
- Login
- Feedback
- RssReader
- Voting
- AreaMap
- FlashIntro

Creating Base Document Template Master.page

The `Master.page` file forms the basis of a template. This file is created for all the color themes, captions and menu types. Therefore, it must not contain information related only to a particular color theme, caption, or menu type.

The first step of converting an HTML template into SiteBuilder template is creating the `Master.page` file in the root directory of the created template.

The example of the `Master.page` file:

```
<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;">
                    
                  </td>
                  <td style="padding-left:
10px;" class="company">$CompanyName$</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">
                $CompanySlogan$
                <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>
    </tr>
  </table>
</body>
</html>
```

```

        </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>

```

To create the Master .page file, perform the following steps:

- 1 Copy the content of the HTML file.
- 2 Replace the HTML header with the following:

```

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>

```

- 3 Replace the fixed company name, slogan, copyright, page title, and textual content with the following constructs:
 - \$CompanyName\$
 - \$CompanySlogan\$
 - \$Copyright\$
 - \$PageTitle\$
 - <SiteBuilder:Container ID="Content" />
- 4 Put the \$LogoPath\$ construct instead of the name of the file with the logo image.
- 5 Put the call of the top-level menu <SiteBuilder:Container ID="TopMenu" /> to the place where the menu is stored.
- 6 Put the call of the top-level menu <SiteBuilder:Container ID="SubMenu" /> to the place where the menu is stored.
- 7 Put the call of the top-level menu <SiteBuilder:Container ID="BottomMenu" /> to the place where the menu is situated.

Creating Menus

SiteBuilder has two-level navigation system. Top-level menu item can have two statuses: active and inactive. Active items are items that are currently being used (highlighted). Inactive items are items available for selecting. A lower-level menu item can have two statuses: link status and active status (it is marked according to the design and does not contain link).

Typically, a template contains both the top-level menu (situated at the top of the page) and the lower-level menu (situated at the left/right part of the page). Sometimes the top-level menu is duplicated at the bottom of the page.

Top-level menu is described in the `TopMenu.skin` file.

Lower-level menu is described in the `SubMenu.skin` file.

Top-level menu at the bottom of the page is described in the `BottomMenu.skin` file.

These skin files should be placed into the following folders:

- `\Template\Theme_name\Menus\Menu1\TopMenu.skin`
- `\Template\Theme_name\Menus\Menu2\TopMenu.skin`
- `\Template\Theme_name\Menus\Menu3\TopMenu.skin`

Note: Menu(1,2,3) - for different menus, there can be one or more folders for one or more types of top-level menu.

- `\Template\Theme_name\Menus\SubMenu.skin`
- `\Template\Theme_name\Menus\BottomMenu.skin`

All these files have the same structure.

Example:

```
<SiteBuilder:SiteMenu>
  <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>
</SiteBuilder:SiteMenu>
```

Where:

`ItemTemplate` – a required template that provides the content and layout for the `SiteMenu` item.

`AlternatingItemTemplate` (if defined) - provides the content and layout for the `SiteMenu` item. If it is not defined, `ItemTemplate` is used.

`SelectedItemTemplate` (if defined) – provides the content and layout for the `SiteMenu` item. If it is not defined, `ItemTemplate` or `AlternatingItemTemplate` is used.

`SeparatorTemplate` (if defined) – provides the content and layout for the separator between the `SiteMenu` items. If it is not defined, the separator will not be displayed.

`HeaderTemplate` (if defined) – provides the content and layout for the header section of the `SiteMenu`. If it is not defined, the header section will not be displayed.

`FooterTemplate` (if defined) – provides the content and layout for the footer section of the `SiteMenu`. If it is not defined, the footer section will not be displayed.

Summary:

`ItemTemplate`, `SelectedItemTemplate`, `AlternatingItemTemplate`

Variables:

- `Url` – a link a menu item leads to.
- `Title` – a title of a menu item.

`SeparatorTemplate`, `HeaderTemplate`, `FooterTemplate`

No special variables or controls.

Insert the code generating a menu item (typically, a cell or several cells of a table) to the needed places of `Item/SeparatorTemplate` (active, just a link).

Perform the same operations for the lower-level menu and bottom menu (if any).

CHAPTER 5

Creating Modules Design

Each template may include SiteBuilder modules. By default, the modules have gray color scheme. You can change the modules colors and make them similar to the template style. To do so, put the `modules.css` file into the color theme folder. See the structure of the `modules.css` file below.

In This Chapter

Modules.css.....	27
Additional Abilities.....	49

Modules.css

In order to clearly understand the structure of the `modules.css` file, you should know that all elements of each module belong to the following categories:

- *Category* – style of a category block
- *Item* – style of list items
- *Comment* – style of comments block
- *Search* – style of search block
- *Form* – style of form block.
- *Information* – style of information block.
- *Pager* – style of paging block.
- *Statuses* – style of information messages.
- *Main* – other, when element is at the template background.

The modules describe their entities design using the classes which contain the words *Category*, *Item*, and *Comment*, selecting the semantically closest classes. Thus, for example, the Blog module uses the *Category* classes to represent categories, the *Item* classes for posts, and the *Comment* classes for comments to posts. In the eShop module products are described with the *Item* classes. The same classes are used to describe lines in the cart and on the Checkout page, while they are actually products.

Thus, when you change, for example, the *Item* classes, 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 72) tool was developed. This tool allows you to quickly review the applied changes.

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

There are also classes, which have to be defined in the very beginning of the `modules.css`. These are such classes as `.mod-item-body a`, which define the the links displaying rules in the `mod-item-body` block. They have to appear prior to a `.mod-item-body-a-strong` declaration, due to the specificity of css classes processing by browsers. The simplest way is to define them right in the beginning of the document. All such classes will be marked with a special note.

The elements can have one of the following characteristics: *normal*, *alternative*, or *strong*. These characteristics enable you to visually mark the elements on the page.

Input Controls

Defines the style for all input controls within the 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 the page.

Selector

```
.mod-hr
```

Sample

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

Buttons

Defines the style for simple buttons within the page.

Selector

```
.mod-hr
```

Sample

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

Strong Buttons

Defines the styles for strong buttons within the page.

Selector

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

Sample

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

Item Body Blocks

Defines the style for item body block.

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;
}
```

Alternating Item Body Blocks

Defines the alternating style for item body block.

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 Blocks

Defines the style for link within item body block.

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 Any Item Body Blocks

Defines the common style for all links without specific class in item body block.

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 30) and **Link in Item Footer Blocks** (see page 35).

Titles in Item Body Blocks

Defines the style for text titles within item body block.

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 Blocks

Defines the style for hr-elements within item body block.

Selector

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

Sample

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

Buttons in Items Body Blocks

Defines the style for simple buttons within item body block.

Selector

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

Sample

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

Strong Buttons in Item Body Blocks

Defines the style for strong buttons within item body block.

Selector

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

Sample

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

Highlighted Text in Item Body Blocks

Defines the style for highlighted text within item body block.

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 Blocks

Defines the style for item header block.

Selector

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

Sample

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

Item Footer Blocks

Defines the style for item footer block.

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 Blocks

Defines the style for item footer block.

Selector

a.mod-item-footer-a

Sample

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

Category Body Blocks

Defines the style for category body block.

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;
}
```

Alternating Category Body Blocks

Defines the alternating style for category body block.

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 any Category Body Blocks

Defines the common style for links within any category body block.

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 38).

Title in Any Category Body Blocks

Defines the style for title text within category body block.

Selector

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

Sample

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

Category Header Blocks

Defines the style for category header block.

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 Blocks

Defines the specific style for links within category header block.

Selector

`a.mod-category-header-a`

Sample

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

Comment Body Blocks

Defines the style for comment body block.

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;
}
```

Alternating Comment Body Blocks

Defines the alternating style for comment body block.

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 Any Comment Body Blocks

Defines the common style for links within any comment body block.

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 41).

Titles in Comment Body Blocks

Defines the style for title text within comment body block.

Selector

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

Sample

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

Comment Header Blocks

Defines the style for comment header block.

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 Blocks

Defines the specific style for links within comment header block.

Selector

`a.mod-comment-header-a`

Sample

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

Search Form Blocks

Defines the style for search form block.

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 Blocks

Defines the style for buttons within search form block.

Selector

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

Sample

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

Form Blocks

Defines the style for form block.

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 Blocks

Defines the style for title text within form block.

Selector

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

Sample

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

Horizontal Rules in Form Blocks

Defines the style for hr-elements within form block.

Selector

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

Sample

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

Buttons in Form Blocks

Defines the style for buttons within form block.

Selector

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

Sample

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

Links in Form Blocks

Defines the style for links within form block.

Selector

`a.mod-form-a`

Sample

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

Information Blocks

Defines the style for information block.

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;
}
```

Information Header Blocks

Defines the style for information header block.

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;
}
```

Information Footer Blocks

Defines the style for information footer block.

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 Information Footer Blocks

Defines the specific style for links within information footer block.

Selector

`a.mod-info-footer-a`

Sample

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

Error Message Blocks

Defines the style for error status in StatusBar control in default layout.

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 Blocks

Defines the style for information status in StatusBar control in default layout.

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;
}
```

Successful Message Blocks

Defines the style for successful status in StatusBar control in default layout.

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;
}
```

Pager Blocks

Defines the style for Pager control block in default layout.

Selector

```
.mod-pager
```

Sample

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

Links in Pager Blocks

Defines the specific style for links within pager blocks in default layout.

Selector

```
a.mod-pager-a
```

Sample

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

Additional Abilities

The guidelines provided in this document enable you to change the HTML code of basic visual elements of a site. The next version of this guide will contain the instructions on how to completely change the HTML code of the SiteBuilder modules. You will be able to create your custom templates that are absolutely different from standard SiteBuilder templates.

Note: This paragraph is applicable to SiteBuilder for Windows only.

To apply the provided guidelines to a certain color theme, place the files with the corresponding names into a corresponding color theme directory. For example, if you want to change the design of the *StatusBar* in the green color scheme, place the *StatusBar.skin* file into the Green folder. Similar to the *Master.page* file, you can use the `$Theme$` variable to write the path to an image.

More detailed information is provided below.

StatusBar

Status bar is a control that displays status messages on the page.

File Location

[Template path] / Themes / [Theme name] / *StatusBar.skin*

Sample

```

<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>

```

The *Statuses* control is used for displaying messages containing the results of users' actions. This control can have 3 values: *error*, *success* and *info*.

Image and text displayed after successful actions is described in the *ErrorTemplate*. You can input here any layout you like. To provide output for template variables you can use the following variables:

DefaultError – it is the path to a default image. You can specify a different path.

MessageType – it is the type of a message, for example, *Success*.

Message – it is the text of a message.

The procedure of changing *InfoTemplate* and *SuccessTemplate* is identical to the above-described procedure. Use *DefaultInfo* and *DefaultSuccess* respectively for default image paths.

Summary

ErrorTemplate

Variables:

- `DefaultError` – it is the path to a default image. You can specify a different path.
- `MessageType` – it is the type of a message, for example, *Success*. (Can be *TextDiv* (see page 76).)
- `Message` – it is the text of a message. (Can be *TextDiv*.)

InfoTemplate

Variables:

- `DefaultError` – it is the path to a default image. You can specify a different path.
- `MessageType` – it is the type of a message, for example, *Success*. (Can be *TextDiv*.)
- `Message` – it is the text of a message. (Can be *TextDiv*.)

SuccessTemplate

Variables:

- `DefaultError` – it is the path to a default image. You can specify a different path.
- `MessageType` – it is the type of a message, for example, *Success*. (Can be *TextDiv*.)
- `Message` – it is a text of a message. (Can be *TextDiv*.)

Pager

Pager is a control that displays paging for modules.

File Location

[Template path] / Themes / [Theme name] / Pager.skin

Sample

```
<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:Button ID="FirstButton" Type="Link" class="mod-
pager-a" />&nbsp;
          <SiteBuilder:Button ID="PreviousButton" Type="Link"
class="mod-pager-a" />&nbsp;
          <SiteBuilder:Container ID="PagesContainer" />&nbsp;
          <SiteBuilder:Button ID="NextButton" Type="Link" class="mod-
pager-a" />&nbsp;
          <SiteBuilder:Button ID="LastButton" Type="Link" class="mod-
pager-a" />
        </td>
      </tr>
    </table>
  </Template>
</SiteBuilder:Pager>
```

In the `<Template>` tag of this file, you can change the pager layout. The `${TotalInfo$}` text variable contains information about the total number of pages. Changing location of this variable, you can control the place of its output.

The `<Template>` tag includes two controls:

- Button
- Container

The *Button* controls define the design of the navigation links of a site. For example, a control with `ID="FirstButton"` redirects a user to the first page of a site. A control with `Type="Link"` is displayed as a link. The alternative value is `Button`. If you specify this type of control, the tag `<input type="button">` will be rendered.

Besides, you can set a CSS class for the *Button* control. By default, the class is `mod-pager-a`. You also can set a style attribute, which will define a style of a link or button depending on the `Type` value.

The procedure of configuring the other *Button* controls is similar to the above-described procedure.

The *Container* controls with `ID=PagesContainer` define the position of the list of navigation links to pages (i.e., "1 2 3 ..."). The *PagesConatiner* control has two attributes: `PageNumber-class` and `CurrentPageNumber-class`. These attributes define the style of the navigation links in active and inactive statuses respectively.

All controls inside the <Template> tag are optional. For example, you can leave only `PagesContainer`, and remove all the other elements.

Summary

Template

Variables:

- `TotalInfo` – it is information about total number of pages. (Can be `TextDiv` (*see page 76*).

Controls:

ID	Type	Is required	Description
<code>PagesContainer</code>	Container (see page 75)	Optional	Defines the place where the list of navigation links will be displayed.
<code>FirstButton</code>	Button (see page 81)	Optional	Represents the control that leads to the first page of the list.
<code>PreviousButton</code>	Button	Optional	Represents the control that leads to the previous page of the list.
<code>NextButton</code>	Button	Optional	Represents the control that leads to the next page of the list.
<code>LastButton</code>	Button	Optional	Represents the control that leads to the last page of the list.

Attributes

- `PageNumber-class` – it is a name of CSS class applied to the inactive navigation links.
- `CurrentPageNumber-class` – it is a name of CSS class applied to active navigation links.

Form

Description

Form is a control that displays a block of input controls (fields) on a page. The example of this control you can see on **Login** and **Feedback** pages. This control serves for creating forms where the number of fields can change dynamically.

File Location

[Template path] / Themes / [Theme name] / Form.skin

Sample

```

<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>

```

In the `<HeaderTemplate>` tag of this file, you can change the top part of a form. Usually, it contains the start-tag of the `TABLE` element. You can change the styles through the `mod-form` class or specify another class.

To configure the design of fields, use the `FieldTemplate` and `AlternatingFieldTemplate` tags. These tags are used to set the 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 `ID=IsRequired` defines the 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 space for the validation message will be allocated in the page layout. If you set `Display="Dynamic"`, the space for the validation message will be dynamically added to the page.

In the `<FooterTemplate>` tag of this file, you can change bottom part of a form. Usually, it contains the end-tag of the `TABLE` element.

Summary

HeaderTemplate

No special variables or controls.

FieldTemplate, AlternatingFieldTemplate

Variables:

- `caption` – it is used for text description of the form fields. (Can be `TextDiv` (*see page 76*).

Controls:

ID	Type	Is required	Description
IsRequired	ValidationText (see page 80)	Mandatory	Place for “is required” asterisk.
InputHolder	Container (see page 75)	Mandatory	Place for input controls.

FooterTemplate

No special variables or controls.

Attributes:

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

Blog: ListPosts Page

This file describes content for the **Blog** module page that shows post entries. Changing its HTML-content you can create your own special blog design.

Note: **Blog** module allows customizing content starting with 3.2.1 version of SiteBuilder for Windows.

File Location

[Template path] / Themes / [Theme 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" style="width: 100%;">
  <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>
</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 monthes and years -->
  <SiteBuilder:Link ID="BackLink" />
</div>
```

This sample shows default layout of `ListPosts` page. Let's look what controls are described here.

- `CategoryDescription`. It is a variable. It can be replaced with `TextDiv` (see page 76) control. This variable contains description of **Blog** module category. And when a user chooses one, it appears on the screen with some visual block that provides `TextDiv` control.

`List` (see page 77) with ID `EntryList` describes the list of blog post entries.

`ItemTemplate` describes how each post must be displayed.

`AlternatingItemTemplate` describes odd posts in this list. In each of this template you can customize variables `Time`, `Title` and `Entry` that represent post time, post title and post body. Also here you can customize `Link` (see page 79) controls with IDs `ReadMore` and `ToComments` that lead to another blog page with full post body and post comments.

`Container` (see page 75) with ID `Pager` reserves place for pager of `EntryList`.

`Link` (see page 79) with ID `BackLink` describes back link that appears when a user navigates through months and years.

Summary

Page

Variables

- `CategoryDescription` – it is the description of a selected category. (Can be `TextDiv` (see page 76)).

Controls

ID	Type	Is required	Description
EntryList	List (see page 77)	Mandatory	List of blog posts.
Pager	Container (see page 75)	Mandatory	Place for pager of <i>EntryList</i> controls.
BackLink	Link (see page 79)	Mandatory	Link to navigate through dates.

EntryList: ItemTemplate, AlternatingItemTemplate

Variables

- `Time` – time of post entry. (Can be `TextDiv`.)
- `Title` – title of post entry. (Can be `TextDiv`.)
- `Entry` – body of post entry. (Can be `TextDiv`.)

Controls

ID	Type	Is required	Description
ReadMore	Link (see page 79)	Mandatory	Displays if a blog entry was too big. Navigates to page with full entry body.
ToComments	Link	Mandatory	Navigates to comments of blog entry.

Blog: ShowPost Page

This file describes content of a page that contains full body of a blog entry, list of entry comments, and form for posting new comment.

Note: Blog module allows customizing content starting with 3.2.1 version of SiteBuilder for Windows.

File Location

[Template path] / Themes / [Theme 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>
      $Entry$
    </td>
  </tr>
  <tr>
    <td colspan="2" height="10">
      <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>
          $Comment$
        </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>
<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 page displays full body of a blog entry at the top of the page and uses posted time as a title of visual block (entry title on this page used as selected menu item title, so you do not need this variable). After entry body navigation link goes: back to list of post entries and to “add new comment” form.

The next item is **List** (see page 77) of entry comments with ID `CommentList` and pager for this list. `CommentList` templates allow you to describe how each item in list must arrange `Author`, `Time`, `Title` and `Comment` variables.

And the last item is form for adding new comment. This form is divided into separate controls so you can provide much more flexible layout than with `Form` control. (**Form** (see page 53) control is still required for pages where the number of field is a variable. For example, the **Feedback** module can be described only with `Form` control).

“Add new comment” form describes: variable `AddCommentTitle` that displays the name of the form; three fields for entering *Author of comment*, *Subject and Message* and **Button** (see page 81) with ID `AddComment`. Each field describes variable with suffix `Caption` – it is a name of the field (for example, `AuthorCaption`), control `TextInput` (see page 80) that displays HTML input and control `ValidationText` (see page 80) that displays validation message if a user tries to enter invalid data into a field.

Summary

Page

Variables

- `Entry` – full body of a blog entry. (Can be `TextDiv` (*see page 76*.)
- `PostedOn` – posted time of an entry. (Can be `TextDiv`.)
- `AddCommentTitle` – name of a form. (Can be `TextDiv`.)
- `AuthorCaption` – title for comment author input. (Can be `TextDiv`.)
- `SubjectCaption` – title for comment subject input. (Can be `TextDiv`.)
- `MessageCaption` – title for comment body input. (Can be `TextDiv`.)

Controls

ID	Type	Is required	Description
<code>BackLink</code>	Link (<i>see page 79</i>)	Mandatory	Navigates back to list of blog entries.
<code>AddCommentLink</code>	Link	Mandatory	Navigates to comments of blog entry.
<code>CommentList</code>	List (<i>see page 77</i>)	Mandatory	List of entry comments.
<code>Pager</code>	Container (<i>see page 75</i>)	Mandatory	Reserves place for pager of comment list.
<code>AuthorInput</code>	<code>TextInput</code> (<i>see page 80</i>)	Mandatory	Input control for author of comment.
<code>AuthorIsRequired</code>	<code>ValidationText</code> (<i>see page 80</i>)	Mandatory	Validation text about author name for comment is required to be entered.
<code>SubjectInput</code>	<code>TextInput</code>	Mandatory	Input control for subject of comment.
<code>SubjectIsRequired</code>	<code>ValidationText</code>	Mandatory	Validation text about subject of comment is required to be entered.
<code>MessageInput</code>	<code>TextInput</code>	Mandatory	Input control for body of comment.
<code>MessageIsRequired</code>	<code>ValidationText</code>	Mandatory	Validation text about body of comment is required to be entered.
<code>AddComment</code>	Button (<i>see page 81</i>)	Mandatory	Button for posting form.

`CommentList`: `ItemTemplate`, `AlternatingItemTemplate`

Variables

- `Author` – author of a comment. (Can be `TextDiv` (*see page 76*.)
- `Time` – posted time of a comment. (Can be `TextDiv`.)
- `Title` – subject of a comment. (Can be `TextDiv`.)
- `Comment` – body of a comment. (Can be `TextDiv`.)

Guestbook: MessageList Page

This page describes the only **Guestbook** module page that displays all messages in site guestbook and form for posting new message.

Note: **Guestbook** module allows customizing content starting from 3.2.1 version of **SiteBuilder** for Windows.

File Location

[Template path] / Themes / [Theme name] / Guestbook /
MessageList.page


```

</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>
      ${AuthorCaption}
      <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;">
      ${MessageCaption}
      <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. At the top of the page content **Container** (see page 75) **StatusBar** reserves the place for server messages. The next item is the list of guestbook messages – here you can see interesting using of **Link** (see page 79) control, it is displayed like an image link.

The next items are pager for list and form with controls for two fields: *Author* and *Message*.

Summary

Page

Variables

- `AddMessageTitle` – it is a title of “add message” form. (Can be `TextDiv` (see page 76).)
- `AuthorCaption` – it is a title for message author input. (Can be `TextDiv`.)
- `MessageCaption` – title for message body input. (Can be `TextDiv`.)

Controls

ID	Type	Is required	Description
StatusBar	Container (see page 75)	Mandatory	Reserves place for StatusBar control.
MessageList	List (see page 77)	Mandatory	List of guestbook messages.
Pager	Container	Mandatory	Reserves place for pager of message list.
AuthorInput	TextInput (see page 80)	Mandatory	Input control for the author of a message.
AuthorIsRequired	ValidationText (see page 80)	Mandatory	Validation text about the author name of a message is required to be entered.
MessageInput	TextInput	Mandatory	Input control for body of message.
MessageIsRequired	ValidationText	Mandatory	Validation text about body of message is required to be entered.
AddMessage	Button (see page 81)	Mandatory	Button for posting form.

MessageList: ItemTemplate, AlternatingItemTemplate

Variables

- `Message` – body of a message. (Can be `TextDiv` (see page 76).)
- `Time` – it is posted time of a message. (Can be `TextDiv`.)
- `Author` – author of a message. (Can be `TextDiv`.)

Controls

ID	Type	Is required	Description
MailTo	Link (see page 79)	Mandatory	Displays mailto-link if a message was posted by an authorized user.

HomePage	Link	Mandatory	Displays link to user home page if a message was posted by an authorized user and he entered it in the profile.
----------	------	-----------	-----------------------------------------------------------------------------------------------------------------

CHAPTER 6

Changing Path to Image Files

The next step is to replace the paths in all the skin files:

- 1 From “images/image_name” to “\$Theme\$/images/image_name”
- 2 From “menus/menu1(2,3)/image_name” to “\$Theme\$/images/image_name”
- 3 From “headers/header1(2,3)/image_name” to “\$Theme\$/images/image_name”.

CHAPTER 7

Making Thumbnails

Thumbnails are reduced-size preview images of templates. The recommended formats of these images are `gif` or `jpg`.

In This Chapter

Making Thumbnails for Templates.....	70
Making Thumbnails for Menus.....	71
Making Thumbnails for Headers	71

Making Thumbnails for Templates

You should create a thumbnail for every color theme. The width of a thumbnail should be 780px. The name of a thumbnail file should be `themename.jpg(gif)`, where `themename` is a name of the theme directory. Place the created thumbnail file into the root directory of a template.

Other preview images will be converted automatically. But to increase their quality, you can make it yourself. These images should be placed into the root directory of a template.

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

- `themename_thumb.jpg(gif)` (118,89)
- `themename_icon.jpg(gif)` (70x53)
- `themename_selected.jpg(gif)` (252x189)

Making Thumbnails for Menus

General rules for making thumbnails for menus:

- The number of thumbnail files depends on the number of color themes and buttons forms. A preview file should be created for each pair of color theme and button type.
- The maximum height of a thumbnail is 26px, and the maximum width is 69px.
- The recommended file formats are `jpg` or `gif`.
- The name of a thumbnail file should be `menuN.jpg(gif)`, where N is the button index number. The thumbnail files should be placed in the `/menus` subdirectory of each color theme.
- If the design (background and foreground colors) of buttons in different color themes is identical, you can create thumbnails only for one color theme and then copy them into the other themes.

Making Thumbnails for Headers

General rules for making thumbnails for headers:

- The number of thumbnail files depends on the number of color themes and header forms. A preview file should be created for each pair of color theme and header type.
- The maximum height of a thumbnail is 56px, and the maximum width is 230px.
- The recommended file formats are `jpg` or `gif`.
- The name of a thumbnail file should be `headerN.jpg(gif)`, where N is the button index number. The file `headerN.jpg(gif)` should be placed in the `/headers` subdirectory.
- Perform these operations for every color theme.
- If the design of headers in different color themes is identical, you can create thumbnails only for one color theme and then copy them into the other themes.

CHAPTER 8

Previewing Templates

SiteBuilder Templates SDK includes a program that enables you to preview a created template as it is displayed on a site.

➤ *To preview a template*

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

You will be prompted for selecting a template theme, menu style, and header.

Note: Optionally, you can open this window with the following command line:
`Preview.exe [Full path to template root directory]`

- 2 Click **Preview**.

The system will launch the check of the template structure. The results of the check are displayed in the **Template Preview** window. If the structure is correct, the preview of the template will open in the Internet Explorer window.

Note: This check does not require the template to contain preview images of theme, menus, and headers. 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.

Compiling Template into Installable Package

Template pack is an `msi` file ready-to-install in SiteBuilder. Compilation of installable template packages is performed by the SiteBuilder Template Pack Compiler program.

Prior to launching the compilation process, make sure that:

- The template has a unique ID (that is why the ID should start with your company name)
- The ID of the template in the `info.xml` file coincides with the name of the directory where the template is stored.

➤ *To compile an installable template pack*

- 1** In the Windows Explorer, right-click the template pack directory.
- 2** Select the SiteBuilder Template Pack Compiler option.

A window displaying the results of the compilation process opens. Besides, the log file `compilation.log` will be saved on your local drive in the root directory.

If the compilation process has been completed successfully, the `[template directory name].msi` file appears in the same level as the template directory.

Installing Template Package

Prior to starting the 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 package
- You have the correct version of SiteBuilder installed on the computer. (Please refer to the *SiteBuilder 3.2 for Windows SDK readme* file for compatibility details.)

➤ *To install a template package*

- 1 Start the compiled template package (.msi file)

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

Appendix A

This chapter contains full description for controls mentioned in the **Additional Abilities** (see page 49) section.

In This Chapter

Container.....	75
TextDiv	76
List	77
Link	79
TextInput.....	80
ValidationText	80
Button.....	81

Container

This control is intended to reserve place where any html can be rendered. You cannot customize this html through this control, just change the place where it will be displayed.

Sample

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

TextDiv

This control is very useful when you need to display a text on some div with background and this div must be hidden when the text is empty. (This behavior has `$CategoryDescription$` variable in start page of the **Blog** module. If we use simple template variable then white block will not disappear when no category is chosen.) You can use this control instead of simple template variable – just use variable name as value for `ID` attribute.

Attributes

These attributes will be applied to div that wraps text.

Name	Value
class	Name of Cascading Style Sheet (CSS) class applied to div tag
style	Style attribute for div tag

Sample

Instead of variable `$VariableName$` you can use (some variables with engine values, like `Url`, do not allow it):

```
<SiteBuilder:TextDiv ID="VariableName" class="css-class" style="padding: 10px; margin: 10px 0 0 0;"/>
```

List

This control describes the layout of list of some elements using templates. With these templates you can define how header of the list must be displayed, what html must be placed in a footer of the list, how an item must be displayed, etc. To clearly understand how it works see the sample below.

Attributes

Usually, the `List` control is wrapped with some tag – table or div. You can change the design of this wrapping using the following attributes:

Name	Value
class	Name of Cascading Style Sheet (CSS) class applied to this control.
style	Style attribute for control html tag.

Control Templates

If you are new to the idea of list template, please see the sample below.

Name	Value
ItemTemplate	Required template that provides the content and layout for items in the <code>List</code> .
AlternatingItemTemplate	If defined, provides the content and layout for alternating items in the <code>List</code> . If not defined, <code>ItemTemplate</code> is used.
SeparatorTemplate	If defined, provides the content and layout for the separator between items in the <code>List</code> . If not defined, a separator will not be displayed.
HeaderTemplate	If defined, provides the content and layout for the header section of the <code>List</code> . If not defined, a header section will not be displayed.
FooterTemplate	If defined, provides the content and layout for the footer section of the <code>List</code> . If not defined, a footer section will not be displayed.

Sample

Let us imagine: there are 3 different comments to display: “First comment”, “Second comment” and “Third comment”. We need to describe how these comments will be displayed on the page. So we write something like this:

```
<SiteBuilder:List ID="EntryList">
  <ItemTemplate>
```

```
<div>$Comment$</div>
</ItemTemplate>
</SiteBuilder:List>
```

And after processing it will be displayed like this:

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

Or we can display these comment in some table. Then we will write the following:

```
<SiteBuilder:List ID="EntryList">
  <HeaderTemplate>
    <table cellpadding="0" cellspacing="0" border="0"
width="100%">
  </HeaderTemplate>
  <ItemTemplate>
    <tr><td>$Comment$</td></tr>
  </ItemTemplate>
  <FooterTemplate>
    </table>
  </FooterTemplate>
</SiteBuilder:List>
```

After processing it will be displayed like this:

```
<table cellpadding="0" cellspacing="0" border="0" width="100%">
  <tr><td>First comment</td></tr>
  <tr><td>Second comment</td></tr>
  <tr><td>Third comment</td></tr>
</table>
```

Link

This control represents html anchor. It could be displayed as 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 anchor tag.
style	Style attribute for anchor tag.
Type	Type of button control. Can be <code>Link</code> or <code>Image</code> . Default is <code>Link</code> .
ImageUrl	Path to image.
ImageAlign	One of the following values: <code>Left</code> , <code>Right</code> , <code>Baseline</code> , <code>Top</code> , <code>Middle</code> , <code>Bottom</code> , <code>AbsBottom</code> , <code>AbsMiddle</code> , <code>TextTop</code> .

Sample

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

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

Or you can define image link with ID `SomeID` and URL to some theme image:

```
<SiteBuilder:Link ID="SomeID" Type="Image"
ImageUrl="$Theme$/images/someimage.gif" ImageAlign="AbsMiddle" />
```

TextInput

This control displays html text input on some form.

Attributes

Name	Value
class	Name of Cascading Style Sheet (CSS) class applied to html text input control.
style	Style attribute for html text input 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 some form.

Attributes

Name	Value
Display	Display behavior of the error message. Default value is <code>Static</code> . Can be one of following values: <code>Static</code> - Space for the validation message is allocated in the page layout. <code>Dynamic</code> - Space for the validation message is dynamically added to the page if validation fails.
class	Name of Cascading Style Sheet (CSS) class applied to this control.
style	Style attribute for control html tag.

Sample

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

Button

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

Attributes

Name	Value
class	Name of Cascading Style Sheet (CSS) class applied to this control.
Type	Type of button control. Can be Button or Link. Default is Button.
style	Style attribute for control html tag.

Sample

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