



SKINNING TUTORIAL

For AspDotNetStorefront v6.2 or higher

Document Revision 1.2

Table Of Contents

- 1. Applicable Application Version 3
 - 1.1. Introduction 3
 - 1.2. Invoking a Skin 3
 - 1.3. Skins are Cookied! 3
 - 1.4. Changing Templates By Language 4
 - 1.5. How Skins Work (Generally) 4
- 2. Skin Structure..... 5
 - 2.1. Skin Files..... 5
 - 2.2. Template.ascx file 7
 - 2.3. Style.css file..... 8
 - 2.4. Skin Images..... 8
 - 2.5. Asp.Net Menu Control..... 8
 - 2.6. String Resources..... 9
- 3. Converting HTML file to skin template (template.ascx) 10
 - 3.1. Template.ascx definition 10
 - 3.2. Template.ascx control lines 10
 - 3.3. Skin Tokens 13
 - 3.4. Convert HTML file to template file..... 13
 - 3.5. Input HTML file from Designer. 13
 - 3.6. Create Template Steps 16
 - 3.7. Resulting Template.ascx 16

1. Applicable Application Version

1.1. Introduction

This guide covers creating skins for AspDotNetStorefront v6.2 or higher. All editions are supported: Standard, PRO, ML, DNN, and IS versions.

A "skin" is what gives your AspDotNetStorefront web site it's overall appearance. A skin is generally composed of an HTML file, stylesheet, and graphics.

Skins can also contain either one or two "template files". If using one template in a skin, that template will be used for all pages on the site. If using two templates in a skin, one template will be used for the home page, and the other template is used for all interior pages. This allows you to have a different layout for your home page, if desired, even within one "skin".

1.2. Invoking a Skin

Most sites have one skin, but it is possible to have multiple skins for a single site. Skins are identified as numbers (e.g. skin 1, skin 2, etc).

Skins are explicitly invoked via:

<http://www.yourdomain.com/default.aspx?skinid=1> (to load skin 1)

<http://www.yourdomain.com/default.aspx?skinid=2> (to load skin 2)

etc.

Upon installation, Skin 1 is the default skin.

1.3. Skins are Cookied!

When a visitor first comes to your site, the skin # defined by AppConfig:DefaultSkinID is shown to them. So if you want skin 2 to be the default, set that AppConfig to have a value of 2. We recommend that you leave the default skin at 1 in almost all cases.

Once you have visited a store with that skin, that skin is now cookied, so you will see that same skin again if you visit the site later. This applies to your site visitors also. If you want to force them to change a skin, use the explicit invocation url's as shown above for example.

NOTE: SITE VISITOR'S ARE COOKIED IMMEDIATELY AS TO THE ACTIVE SKIN WHEN THEY FIRST VISIT YOUR SITE!

When the user later returns to your site they will see their same skin EVEN IF you have later changed AppConfig:DefaultSkinID! Use explicit invocation url's to change prior visitor skins. The AppConfig:DefaultSkinID IS ONLY USED for the very first visit to the site that someone makes.

We cover how to convert a HTML file (like one you may get from your designer) into a skin template file in section three of this manual.

1.4. Changing Templates By Language

You can have a skin that changes content, layout, and even tokens by language. For example, if you have Japanese installed, you can provide a `template.ascx` file (for en-US, English users), and a `template.ja-JP.ascx` file, for Japanese language users. The storefront will automatically use the one which matches the current customer viewing language.

You do not have to provide separate template files for each language, but it is possible and supported.

If we do not find a specific override `template.locale.ascx` file for the customer's viewing language, we will use the `template.ascx` file anyway.

1.5. How Skins Work (Generally)

A skin template file is loaded by the storefront on every page. We then parse out the tokens in the template to figure out what goes where on the page.

The tokens basically say things like: put the horizontal menu here, put this text here, put the topic called "PDQ" here, put the page contents here, put the site navigation breadcrumb here, etc... For a complete list of tokens supported, consult the main manual.

This token replacement process happens transparently for you once the skin template files are created.

For developers, the "code behind" for the skin template files are in the `/app_code/templatebase.cs` class. The main store "skin" file is `/app_code/skinbase.cs`.

Both are used on almost every page. Designers do not generally have to care about these codebehind files.

`SkinBase.cs` is most often the one you interact with in asp.net code when making custom modifications to the storefront.

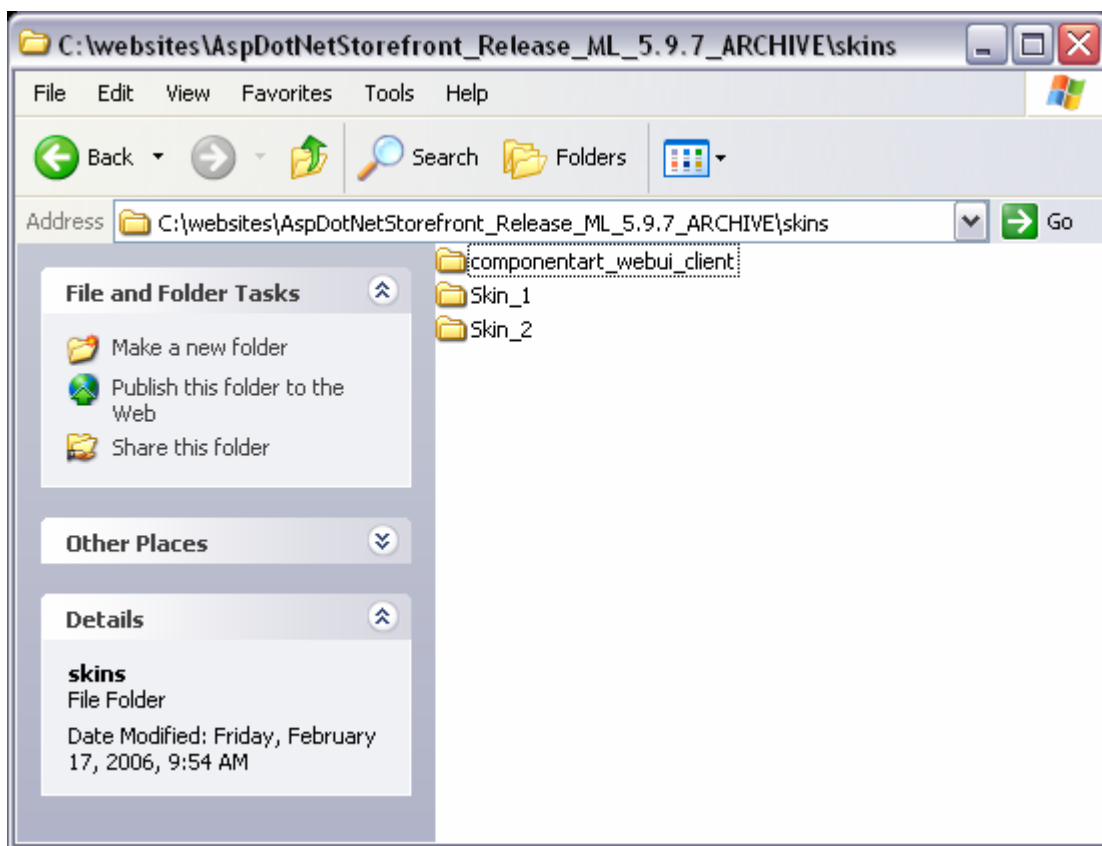
2. Skin Structure

2.1. Skin Files

To change the appearance of your storefront, you will be creating a new "skin". A skin is comprised of the following elements:

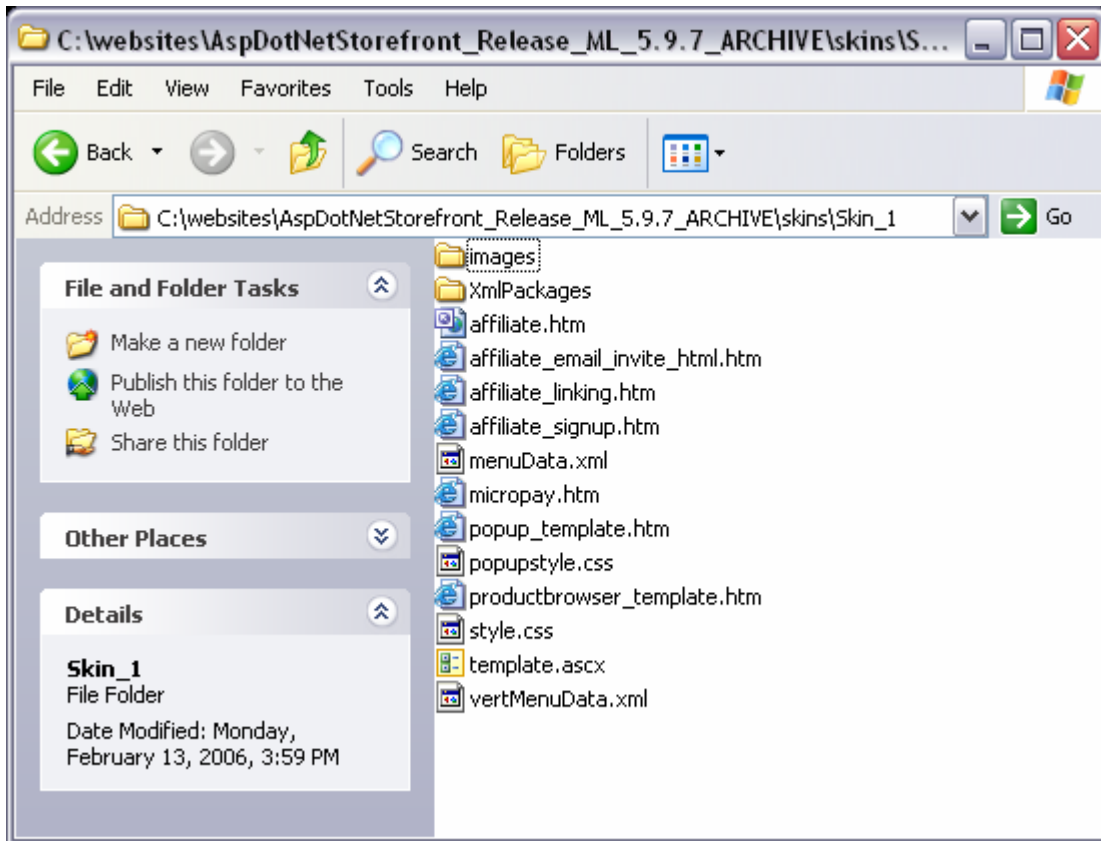
1. template.ascx file (this will be created from your html design file)
2. style sheet (e.g. style.css)
3. graphics used in the skin (e.g. images/logo.gif)

Skins are stored in the web site in the skins/skin_N directory (e.g. skins/skin_1). By default, your skins directory should look like the following:



Note that some of the files may change over time with various release updates. Each skin is stored in a folder called skin_N, and the componentart_webui_client contains some common javascript files for the menu controls. Just leave them as you see them. It will almost never be needed to edit them.

If you go into any skin folder (e.g. skin_1), it should look something like:



Note that some of the files may change over time with various release updates. Some of the files in this directory are:

- **template.ascx**: this is the main store skin file! It is really just an HTML file that has been turned into a user control (.ascx) for the .net platform. We will provide instructions later on for how to convert a html file into a template.ascx file. For some stores, you may also have a `hometemplate.ascx`, as the store supports a template for the home page, and for all interior pages. See `AppConfig:HomeTemplate` if you need a custom home page template file.
- **style.css**: This is the main store style sheet, usually included by the `template.ascx` file(s)
- **affiliate_*.htm**: these files are used for some of the affiliate page info on the site. You may or may not ever need to edit these.
- **popup_template.htm**: this file is used for popup windows (you should not have to edit this)
- **productbrowser_template.htm**: this file is used for pack products (you should not have to edit this)
- **menuData.xml**: this is the horizontal menu master control Xml file. To change your store horizontal main menu items, you can edit this file (using notepad). Many of the sub-menus are dynamically generated by the storefront code (`templatebase.cs` file). This file also contains strings in the syntax (!xxxxx!).

Those strings are localized at run-time, and the store looks for special strings (e.g. (!menu.Categories!)), for instance) to know WHERE to dynamically insert sub-menus at run time. Other main menus (e.g. menu.CustomerService) you can just edit to list whatever you want.

- vertMenuData.xml: this is the vertical menu master definition.
- Popupstyle.css: this file is used for some simple popup-up window css definitions
- Micropay.htm: used for pop-up info on the Micropay Payment Method (see main manual for more details)
- Receiptstyle.css: this option file can be used if you need a special stylesheet for your receipts. This is required for instance if your main store style is using a dark background (which doesn't print well), so then a separate receiptstyle is probably needed so that receipts are able to be printed on white background.

2.2. Template.ascx file

The template.ascx file is the main store skin layout file. This file defines the appearance of all store pages.

If you need a different template layout for your home page, create a hometemplate.ascx file, and set AppConfig:HomeTemplate=hometemplate.ascx. Now, your skin has two layout files, one for the home page, and one for all inner pages.

It is even possible to vary the skin template file by category, product, manufacturer, etc, but those types of situations usually also require a few code level mods to do based on your unique business rules.

NOTE: Once you have converted a html file into a "skin ascx file", you **cannot later edit template.ascx in FrontPage or DreamWeaver!** You can edit it in Notepad or Visual Studio.net if you want. You **can** use FrontPage or DreamWeaver to design a new skin file (.htm) and then convert that to .ascx. That conversion process is explained later in this document.

You will notice that the template.ascx file contains some things (tokens) which get replaced at run-time (e.g. name of user who is logged in, # of items in their cart, etc). Tokens in the template.ascx file are identified by (!...!) syntax, e.g. (!NUM_CART_ITEMS!) , or (!CARTPROMPT!), etc...

For a complete list of supported tokens, see our support forums, or consult the main manual. For example, if you wanted to include a topic called "MyStuff" into your skin at a specific location, you could use this token in your skin file:

```
(!Topic Name="MyStuff"!)
```

At runtime, the store will find that token, get the topic (from the db) and insert it there. Other dynamic tokens include invoking XmlPackages directly in the skin file, e.g:

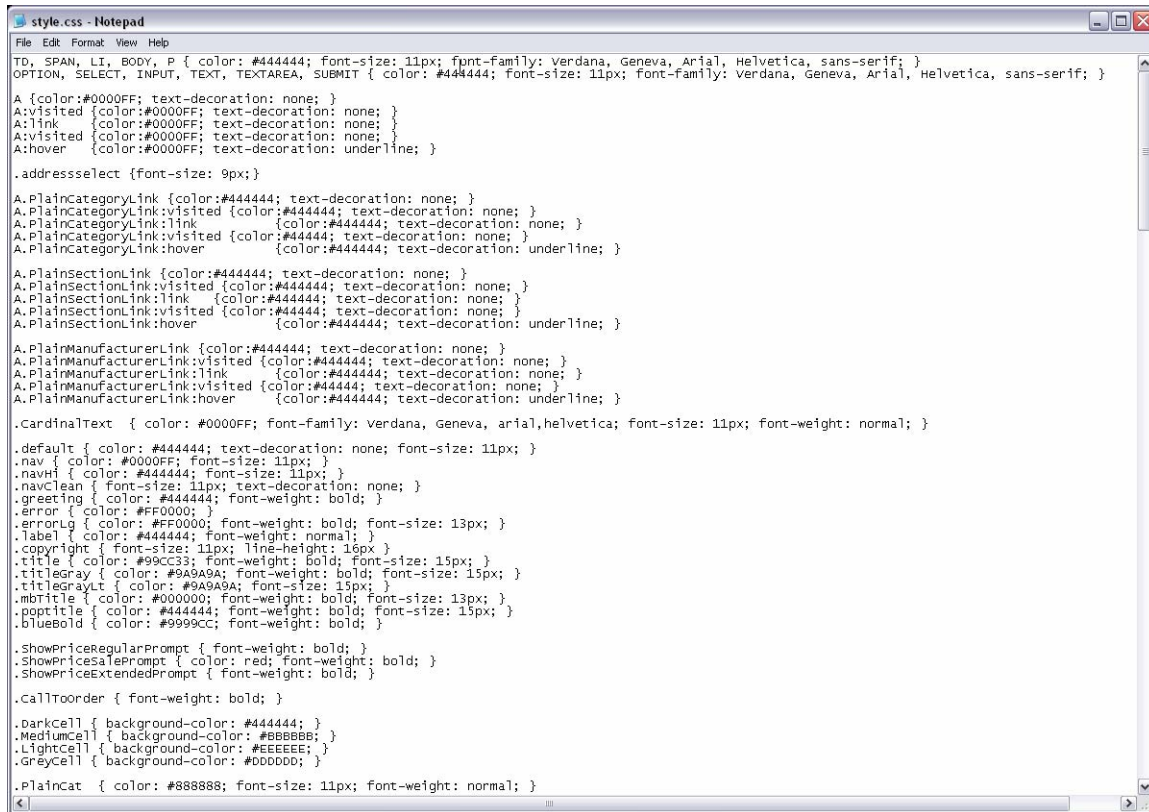
```
(!XmlPackage Name="mycoolpackage.xml.config"!)
```

This XmlPackage will be executed at RUN TIME on each page, and it can produce

completely dynamic data output, and make almost any database query (or set of queries) desired. Through this technique, you can inject almost any arbitrary dynamic data contents into your skin files.

2.3. Style.css file

The style.css file controls most (but not all) fonts/colors/etc used by the storefront. You can edit this style sheet in notepad (or your favorite editor like FrontPage) to make changes. A small sample of this file is shown below. We will not explain styles here, consult your favorite HTML/CSS tutorial to learn about css stylesheets.



```
style.css - Notepad
File Edit Format View Help
TD, SPAN, LI, BODY, P { color: #444444; font-size: 11px; font-family: Verdana, Geneva, Arial, Helvetica, sans-serif; }
OPTION, SELECT, INPUT, TEXT, TEXTAREA, SUBMIT { color: #444444; font-size: 11px; font-family: Verdana, Geneva, Arial, Helvetica, sans-serif; }

A {color:#0000FF; text-decoration: none; }
A:visited {color:#0000FF; text-decoration: none; }
A:link {color:#0000FF; text-decoration: none; }
A:visited {color:#0000FF; text-decoration: none; }
A:hover {color:#0000FF; text-decoration: underline; }

.addressselect {font-size: 9px;}

A.PlainCategoryLink {color:#444444; text-decoration: none; }
A.PlainCategoryLink:visited {color:#444444; text-decoration: none; }
A.PlainCategoryLink:link {color:#444444; text-decoration: none; }
A.PlainCategoryLink:visited {color:#444444; text-decoration: none; }
A.PlainCategoryLink:hover {color:#444444; text-decoration: underline; }

A.PlainSectionLink {color:#444444; text-decoration: none; }
A.PlainSectionLink:visited {color:#444444; text-decoration: none; }
A.PlainSectionLink:link {color:#444444; text-decoration: none; }
A.PlainSectionLink:visited {color:#444444; text-decoration: none; }
A.PlainSectionLink:hover {color:#444444; text-decoration: underline; }

A.PlainManufacturerLink {color:#444444; text-decoration: none; }
A.PlainManufacturerLink:visited {color:#444444; text-decoration: none; }
A.PlainManufacturerLink:link {color:#444444; text-decoration: none; }
A.PlainManufacturerLink:visited {color:#444444; text-decoration: none; }
A.PlainManufacturerLink:hover {color:#444444; text-decoration: underline; }

.CardinalText { color: #0000FF; font-family: verdana, Geneva, arial,helvetica; font-size: 11px; font-weight: normal; }

.default { color: #444444; text-decoration: none; font-size: 11px; }
.nav { color: #0000FF; font-size: 11px; }
.navH { color: #444444; font-size: 11px; }
.navClean { font-size: 11px; text-decoration: none; }
.greeting { color: #444444; font-weight: bold; }
.error { color: #FF0000; }
.errorLg { color: #FF0000; font-weight: bold; font-size: 13px; }
.label { color: #444444; font-weight: normal; }
.copyright { font-size: 11px; line-height: 16px }
.title { color: #99CC33; font-weight: bold; font-size: 15px; }
.titlegray { color: #9A9A9A; font-weight: bold; font-size: 15px; }
.titlegrayLt { color: #9A9A9A; font-size: 15px; }
.mbTitle { color: #000000; font-weight: bold; font-size: 13px; }
.poptitle { color: #444444; font-weight: bold; font-size: 13px; }
.blueBold { color: #9999CC; font-weight: bold; }

.showPriceRegularPrompt { font-weight: bold; }
.showPriceSalePrompt { color: red; font-weight: bold; }
.showPriceExtendedPrompt { font-weight: bold; }

.callToorder { font-weight: bold; }

.darkCell { background-color: #444444; }
.MediumCell { background-color: #BBBBBB; }
.LightCell { background-color: #EEEEEE; }
.GreyCell { background-color: #DDDDDD; }

.PlainCat { color: #888888; font-size: 11px; font-weight: normal; }
```

As a note, please see that the first 2 lines control nearly ALL of the text color & size used on your entire store web site.

2.4. Skin Images

Images used in the skin are in the images directory of that skin. We recommend that all skin related images be stored here. We do not recommend storing skin images in the /images dir on the root of your web site. We try to limit that directory to only having very generic images which are shared across any skin you might use. Keeping your skin images inside each skin directory is preferred.

2.5. Asp.Net Menu Control

The sample skin template files provided show sample invocations of the horizontal menu, vertical menu, tree menu, advanced category browse boxes (collapsing/expanding) for

categories, departments, and manufacturers. Section 3 of this manual covers how to create these from your HTML file.

2.6. String Resources

Skin template files may also contain string resources. These are used to call out strings from the storefront database, which can vary by user locale (language).

Nearly every string used by the storefront is now stored in the database. You can edit these strings (e.g. skinbase.cs.8) in the admin site under Misc -> String Resource Manager.

The master strings for the storefront are supplied in Excel format in the stringresources directory (e.g. strings.en-US.xls). You can maintain your master strings in excel, and re-upload a new excel file into the admin site at any time. Note that after doing this, you MUST restart the storefront (touch web.config, iisreset, etc) as all string resources are cached in memory to increase performance.

Each string resource has a CaSe SENSiTiVE name (e.g. account.aspx.1) and its corresponding string value in whatever language this resource file is for. Tokens in the string value (e.g. {0}, {1}, etc...) should be left there in all translations, as the store will be putting some run-time values in there.

To use a string resource inside a skin template file, just call it out in the syntax:

```
(!stringresourceName!)
```

The storefront will then parse these at run time, based on user language. An example would be:

```
(!shoppingcart.aspx.1!)
```

Consult the main manual for more information on string resources.

3. Converting HTML file to skin template (template.ascx)

3.1. Template.ascx definition

The template file, template.ascx, is really just a HTML file with a few special lines added to it's header, and then renamed to template.ascx. In the new form, it is an asp.net user control, which we then load in our code to provide true asp.net based page handling.

Most designers create "HTML" files. So this section shows the simple steps which are required to turn almost any HTML file, into a skin template.ascx file.

Please open the template.ascx file we provide with our default skins, to glance through their structure. Understanding that is helpful first, before you convert your HTML file to a template.ascx file.

3.2. Template.ascx control lines

In particular, there are a number of header lines in a template.ascx file. We document them here below (follow along in the default skin_1/template.ascx file we provide).

The header lines are:

1. `<%@ Control Language="c#" AutoEventWireup="false"`
`Inherits="AspDotNetStorefront.TemplateBase"`
`TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>`
2. `<%@ Register TagPrefix="ComponentArt" Namespace="ComponentArt.Web.UI"`
`Assembly="ComponentArt.Web.UI" %>`
3. `<html xmlns="http://www.w3.org/1999/xhtml">`
4. `<head>`
5. `<meta http-equiv="Content-Type" content="text/html; charset=utf-8">`
6. `<title>(!METATITLE!)</title>`
7. `(!CURRENCY_LOCALE_ROBOTS_TAG!)`
8. `<meta name="description" content="(!METADESCRIPTION!)">`
9. `<meta name="keywords" content="(!METAKEYWORDS!)">`
10. `<link rel="stylesheet" href="skins/Skin_(!SKINID!)/style.css" type="text/css">`
11. `<script type="text/javascript" src="jscripts/formValidate.js"></script>`
12. `<link rel="alternate" type="application/rss+xml" title="ROR"`
`href="rorindex.aspx" />`
13. `</head>`
14. `<body rightmargin="0" leftmargin="0" topmargin="0" marginwidth="0"`
`marginheight="0" bgcolor="#ffffff">`
15. `(!XmlPackage Name="skin.adminalert.xml.config!")`
16. `(!PAGEINFO!)`
17. *... rest of your html design file*
18. *... rest of your html design file*
19. *(!Placeholder Token!) This is where the main page contents are put!*
20. *... rest of your html design file*
21. *... rest of your html design file*
22. `</body>`
23. `</html>`

You will find these lines on almost every single template file. Your version MAY BE DIFFERENT! Explanations of each line (where appropriate) are given below that line in italics (the descriptions are NOT part of the actual template.ascx file).

1. `<%@ Control Language="c#" AutoEventWireup="false"`
`Inherits="AspDotNetStorefront.TemplateBase"`

```
TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
```

This line defines this file as a web control, for c# language. Your file may use Language="vb" depending on which development language you are using. All templates inherit from TemplateBase.cs class. The TemplateBase.cs class provides some core skinning functionality and works as a helper for SkinBase.cs class. Consult the code for what those classes do. A tutorial on the inner workings of our skinning engine is beyond the scope of this document.

2.

```
<%@ Register TagPrefix="ComponentArt" Namespace="ComponentArt.Web.UI" Assembly="ComponentArt.Web.UI" %>
```

This line tells asp.net to register the .net menu controls we are using on the page. If you are not using any ComponentArt menu or tree controls in your skin design, you can remove this line. Many skin designs do not need to have the menu on the page, so that is a design consideration for your site skin design person.

3.

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

The html declaration for this page, so really, when it comes down to it, a skin template.ascx control file looks almost exactly like a regular .htm file except it has the additional asp.net specifications from lines 1 and 2 above. For Developers: You will note that the template.ascx file DOES NOT have any code-behind! This allows you to add a new skin to the site at any time without recompiling anything. TemplateBase.cs and SkinBase.cs classes provide the skinning engine core logic. If you have to change the code-behind for the template, it is the TemplateBase.cs file.

4.

```
<head>
```

Html head tag.

5.

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

Use utf-8 for international language support (required).

6.

```
<title>(!METATITLE!)</title>
```

The title of the page. You can hard-code it or use the token shown above (recommended) so each page gets titles that match the page (e.g. category name, product name, specific Search Engine title/s you entered, etc). See AppConfig:SE_MetaTitle and the MetaTitle fields in the editcategory, editproduct, editvariant pages, etc on the admin site). You can have different values for many pages defined by the products you setup in your store.

7.

```
(!CURRENCY_LOCALE_ROBOTS_TAG!)
```

Parser tag that outputs the appropriate currency meta tag for your page (optional, you can delete this line if not needed).

8.

```
<meta NAME="description" CONTENT="(!METADESCRIPTION!)">
```

Description meta-tag for Search Engines, dynamic token replaced at run-time based on page specifications (defined by admin site, and read from the db).

9.

```
<meta NAME="keywords" CONTENT="(!METAKEYWORDS!)">
```

Keywords meta-tag for Search Engines, dynamic token replaced at run-time based on page specifications (defined by admin site, and read from the db). MetaKeywords come from various things in the db. On the home page, they

come from the AppConfig:SE_* parameters. For all other pages (e.g. categories, products, topics, etc...) they come from what you have defined in the database for those particular items. Every single category, section, product, manufacturer, topic can have it's own meta description, keywords, page title, etc... This is for search engine optimizations.

10. `<link rel="stylesheet" href="skins/Skin_(!SKINID!)/style.css" type="text/css">`

Required, this invokes the main store style sheet which defines how things appear (e.g. font sizes, colors, etc). Note that the location of this element is dynamic based on a relative path of skins/skin_(!SKINID!)/... That technique is used to locate ALL items referenced from this skin where the element can vary by skin number. This (!SKINID!) will be replaced at run-time with the number of the active skin (e.g. 1), so the resolution will work properly. Note that having this capability confuses most HTML editors, so editing in anything besides notepad can produce strange results!! You "can" hard-code skin numbers if you want, just realize what you are doing (e.g. skins/skin_27/style.css). If you change the "number" of this skin, this line won't work properly if you hard-code it. Note that the "location" of the skin when executed (in then browser) is store root (not skins/skin_N/...). This is because the browser is using the skin file as RENDERED from the root of the store where the .aspx files are.

11. `<script type="text/javascript" src="jscripts/formValidate.js"></script>`

Required on ALL pages for form validation routines. Does not need to be relative path, as file is always in root of store.

12. `<link rel="alternate" type="application/rss+xml" title="ROR" href="rorindex.aspx" />`

optional, if using Rorweb.com site maps, otherwise, you can delete this line.

13. `</head>`

html close body tag

14. `<body rightmargin="0" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0" bgcolor="#ffffff">`

Whatever you want for your body tag. Your skin designer will tell you this.

15. `(!PAGEINFO!)`

This line is optional, but EXTREMELY helpful for debug/run-time info. And we STRONGLY ask that you leave them in your skin, to help with any customer support we may need to provide!

16. The rest of the lines are your own skin html design. You can however embed string resource tokens, parser tokens, XmlPackage invocation tokens, and the page contents tokens in that HTML.

17. You will have one more critical line in your skin template. This line tells the store where to put the "contents" (the main page content block) for the page. This content varies by whatever is being shown. It could be a category listing, a product page, the shopping cart page, a checkout page, etc. So each page has a content block that is rendered into this line in the file:

```
<asp:placeholder id="PageContent" runat="server"></asp:placeholder>
```

If you forget to put that placeholder into your template, no store "content" will be displayed.

3.3. Skin Tokens

Please consult the full manual for a list of all "tokens" which can be included in your skin template files. There are about 150 tokens which are supported, for common dynamic elements like "put username here", "put logout link here", "show a root list of categories here", etc...

3.4. Convert HTML file to template file

Ok, this is the section you want. You just got a HTML file (with stylesheet/css files, graphics, etc) from your designer. Now you want to create a skin with it. This section shows how.

For illustrative purposes, we are going to use an EXTREMELY simple html file simulating the input you may get from the designer. Your HTML files will of course be more complicated.

NOTE: You can do CSS skin designs also, That is outside the scope of this skinning tutorial, but the exact same steps, concepts, and transformations apply, it's just that you'd be using a lot of CSS elements instead of table elements, etc...

3.5. Input HTML file from Designer.

Here is the input file we will start with (ignoring .css files and images):

```
<html>
<head>
<title>Cool Site Title</title>
<link rel="stylesheet" href="designer.css" type="text/css">
</head>
<body rightmargin="0" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
bgcolor="#ffffff">
    <table width="100%" align="center" cellpadding="0" cellspacing="0"
border="0">
    <tr>
        <td colspan="3" height="3" width="100%"></td>
    </tr>
    <tr>
        <td colspan="3" width="100%" align="left" valign="top" height="60">
            <table width="100%" cellpadding="0" cellspacing="0" border="0"
bgcolor="#ffffff">
                <tr>
                    <td width="316" height="60" align="left" valign="middle"><a
href="index.html"></a></td>
                    <td align="right" valign="middle"></td>
                    <td height="60" align="right" valign="middle">
                        <a href="index.html" class="head">Home</a> | <a
href="shoppingcart.html" class="head">
```



```

                                <!-- LEFT COL -->
                                <td valign="top" align="left">
DESIGNER)                                PUT LEFT COL STUFF HERE (NOTE FROM
                                </td>
                                <!-- CENTER COL: -->
                                <td width="100%" valign="top" align="left">
FROM DESIGNER)                            <!-- CONTENTS START -->
                                PUT PAGE CONTENST HERE (NOT
                                <!-- CONTENTS END -->
                                </td>
                                <!-- RIGHT COL: -->
                                <td valign="top" align="left">
                                </td>
                                </tr>
                                </table>
                                </td>
                                </tr>
                                </table>
                                </td>
                                </tr>
                                </table>
                                </td>
                                <td width="1" bgcolor="#cccccc"></td>
                                </tr>
                                <tr>
                                <td colspan="3" valign="middle" height="24" bgcolor="#cccccc"
align="center">
                                <a href="index.html" class="foot">Home</a> | <a href="contact.html"
class="foot">
                                Contact Us</a> | <a href="affiliate.html"
class="foot">Affiliates</a> | <a href="returns.html" class="foot">
                                Return Policy</a> | <a href="privacy.html" class="foot">Privacy
Policy</a> |
                                <a href="security.html" class="foot">Security Policy</a> | <a
href="sitemap.html" class="foot">
                                Site Map</a> | <a class="foot" href="copyright.html">Copyright
&copy; 1995-2006. All Rights Reserved.</a>
                                </td>
                                </tr>
                                <tr>
                                <td colspan="3" height="5" width="100%"
background="images/bottomgradient.jpg"></td>
                                </tr>
                                <tr>
                                <td colspan="3" height="5" width="100%"></td>
                                </tr>
                                <tr>
                                <td colspan="3" height="5" width="100%"></td>
                                </tr>
                                </table>
                                </div>
                                </div>
                                </body>
                                </html>

```

3.6. Create Template Steps

In pseudo-logic, these are the steps you will be taking (assume template.htm as input from designer):

1. Remove the HTML header lines provided by the designer, all the way up to the <body> tag. Replace them with the ones we show above (the ones we shipped you with our skins/skin_1/template.ascx file).
2. Put back the designer's body tag
3. Put the contents placeholder into the file where you want the "page guts" to be rendered.
4. Put in other control tags you want (e.g. horizontal menu, vertical tree, etc). See sample skin file for these.
5. Change all hrefs (links) in the html template to be valid AspDotNetStorefront page links (e.g. designer might have put contact.html in a href, you change that to t-contact.aspx, which is how the storefront will refer to that page, etc)
6. Remove any absolute href links, and make them relative (to the storefront, not hard-coded http://...etc)
7. Add links to shopping cart page, wish list, login/logout, etc...where you need. Hopefully your designer left you places for these, or put dummy links into your template.htm file design for these.
8. Rename file from template.htm to template.ascx
9. Create new skin folder (copy our current skin_1 folder to skin_5 for example)
10. Copy template.ascx to skin_5 dir (overwriting the one from skin_1)
11. Copy the stylesheet into the skin_5/style.css stylesheet. NOTE: You may have to MERGE the styles provided by your designer into the master style sheet. There may be conflicts also, it just depends on how your designer set things up. If your designer used style names that we had already used, you'll have to reconcile those conflicts.
12. Copy the designer skin images into your /skin_5/images dir
13. Rename all image hrefs to "skins/(!SKINID!)/images/..." (e.g. make them relative to the active skin images dir)
14. Save the skin, and try it out (default.aspx?skinid=5 in this case)

3.7. Resulting Template.ascx

When done the transformation, we now have:

```
<%@ Control Language="c#" AutoEventWireup="false"
Inherits="AspDotNetStorefront.TemplateBase"
TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
<%@ Register TagPrefix="ComponentArt" Namespace="ComponentArt.Web.UI"
Assembly="ComponentArt.Web.UI" %>
<html>
<head>
<title>(!METATITLE!)</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
(!CURRENCY_LOCALE_ROBOTS_TAG!)
<meta NAME="description" CONTENT="(!METADESCRIPTION!)">
<meta NAME="keywords" CONTENT="(!METAKEYWORDS!)">
<link rel="stylesheet" href="skins/Skin_(!SKINID!)/style.css" type="text/css">
<script type="text/javascript" src="jscripts/formValidate.js"></script>
<link rel="alternate" type="application/rss+xml" title="ROR" href="rorindex.aspx"
/>
</head>
```



```

        <td colspan="3" width="100%" height="24" align="left" valign="middle"
bgcolor="#cccccc">
        <table width="100%" cellpadding="0" cellspacing="0" border="0"
bgcolor="#ffffff">
        <tr>
        <td align="left" valign="middle" bgcolor="#cccccc" height="25">
<!-- START TOP MENU -->
    <ComponentArt:Menu id="PageMenu"
        ClientScriptLocation="skins/componentart_webui_client/"
        ImagesBaseUrl="skins/skin_1/images/"
        ScrollingEnabled="true"
        ScrollUpLookId="ScrollUpItemLook"
        ScrollDownLookId="ScrollDownItemLook"
        Orientation="horizontal"
        CssClass="TopMenuGroup"
        DefaultGroupCssClass="MenuGroup"
        DefaultItemLookID="DefaultItemLook"
        DefaultGroupItemSpacing="1"
        ExpandDelay="0"
        ExpandDuration="0"
        ExpandSlide="None"
        ExpandTransition="None"
        CascadeCollapse="false"
        CollapseDelay="0"
        CollapseSlide="None"
        CollapseTransition="None"
        EnableViewState="false"
        runat="server">
    <ItemLooks>
        <ComponentArt:ItemLook LookId="DefaultItemLook"
HoverCssClass="MenuItemHover" LabelPaddingTop="2px" ActiveCssClass="MenuItemDown"
LabelPaddingRight="15px" LabelPaddingBottom="2px" ExpandedCssClass="MenuItemDown"
LabelPaddingLeft="5px" CssClass="MenuItem" />
        <ComponentArt:ItemLook LookId="TopItemLook" CssClass="TopMenuItem"
HoverCssClass="TopMenuItemHover" LabelPaddingLeft="4" LabelPaddingRight="4"
LabelPaddingTop="2" LabelPaddingBottom="2" />
        <ComponentArt:ItemLook LookID="ScrollUpItemLook" ImageUrl="scroll_up.gif"
ImageWidth="15" ImageHeight="13" CssClass="ScrollItem" HoverCssClass="ScrollItemH"
ActiveCssClass="ScrollItemA" />
        <ComponentArt:ItemLook LookID="ScrollDownItemLook"
ImageUrl="scroll_down.gif" ImageWidth="15" ImageHeight="13" CssClass="ScrollItem"
HoverCssClass="ScrollItemH" ActiveCssClass="ScrollItemA" />
        <ComponentArt:ItemLook LookID="BreakItem" ImageUrl="break.gif"
ImageHeight="1" ImageWidth="100%" />
    </ItemLooks>
    </ComponentArt:Menu>
<!-- END TOP MENU -->
        </td>
    </tr>
</table>
</td>
</tr>
<tr>
    <td width="1" bgcolor="#cccccc"></td>
    <td width="100%" height="25" align="left"
        valign="middle">
        <div id="SectionTitle" style="DISPLAY: block; VISIBILITY: visible">
            <table width="100%" cellpadding="0" cellspacing="0" border="0">
                <tr>
                    <td align="left"><span class="SectionTitleText">(!SECTION_TITLE!)</span></td>

```

```

        <td align="right">&nbsp;</td>
    </tr>
</table>
</div>
</td>
<td width="1" bgcolor="#cccccc"></td>
</tr>
<tr>
<td width="1" bgcolor="#cccccc"></td>
<td valign="top" align="center" bgcolor="#ffffff">
<table width="100%" align="center" cellpadding="4" cellspacing="0">
<tr>
<td align="left" valign="top" width="100%" height="400">
<table border="0" cellpadding="4" cellspacing="0" width="100%">
<tr>
<td width="100%" valign="top" align="left">
width="100%">
<table border="0" cellpadding="4" cellspacing="0"
width="100%">
<tr>
<td align="left" colspan="2">
<!-- LEFT COL -->
<td valign="top" align="left">
(!SEARCH_BOX!)
(!SECTION_BROWSE_BOX!)
(!CATEGORY_BROWSE_BOX!)
(!HELPBOX!)<br>
</td>
<td align="center" colspan="2">
<!-- CENTER COL: -->
<td width="100%" valign="top" align="left">
<!-- CONTENTS START -->
<asp:placeholder id="PageContent"
runat="server"></asp:placeholder>
<!-- CONTENTS END -->
</td>
<td align="right" colspan="2">
<!-- RIGHT COL: -->
<td valign="top" align="left">
</td>
</tr>
</table>
</td>
</tr>
</table>
</td>
</tr>
</table>
</td>
<td width="1" bgcolor="#cccccc"></td>
</tr>
<tr>
<td colspan="3" valign="middle" height="24" bgcolor="#cccccc"
align="center">
<a href="default.aspx" class="foot">Home</a> | <a href="t-
contact.aspx" class="foot">
Contact Us</a> | <a href="t-affiliate.aspx"
class="foot">Affiliates</a> | <a href="t-returns.aspx" class="foot">
Return Policy</a> | <a href="t-privacy.aspx" class="foot">Privacy
Policy</a> |
<a href="t-security.aspx" class="foot">Security Policy</a> | <a
href="sitemap.aspx" class="foot">
Site Map</a> | <a class="foot" href="t-copyright.aspx">Copyright
&copy; 1995-2006. All Rights Reserved.</a>

```

```

        </td>
    </tr>
    <tr>
        <td colspan="3" height="5" width="100%"
background="skins/Skin_(!SKINID!)/images/bottomgradient.jpg"></td>
    </tr>
    <tr>
        <td colspan="3" height="5" width="100%"></td>
    </tr>
    <tr>
        <td colspan="3" width="100%" align="center"><small>Powered by <a
href="http://www.aspdotnetstorefront.com" target="_blank"
title="AspDotNetStoreFront.com">AspDotNetStorefront</a> <a
href="http://www.aspdotnetstorefront.com" target="_blank" title="e-Commerce
Shopping Cart"> E-Commerce Shopping Cart</a></small></td>
    </tr>
    <tr>
        <td colspan="3" height="5" width="100%"></td>
    </tr>
</table>
</div>
</div>

<noscript>

    Powered by <a href="http://www.aspdotnetstorefront.com"
target="_blank">AspDotNetStorefront E-Commerce Shopping Cart</a>

</noscript>
</body>
</html>

```

You will see all the header control lines added, the images are now relative to skins/skin_(!SKINID!)/images/, there is a contents placeholder added, and the top menu bar is now an active control, etc...

These transformation steps are very simple, and after you've done a skin, only take about 5-10 minutes to perform, if your designer gave you a solid starting HTML design to work from.