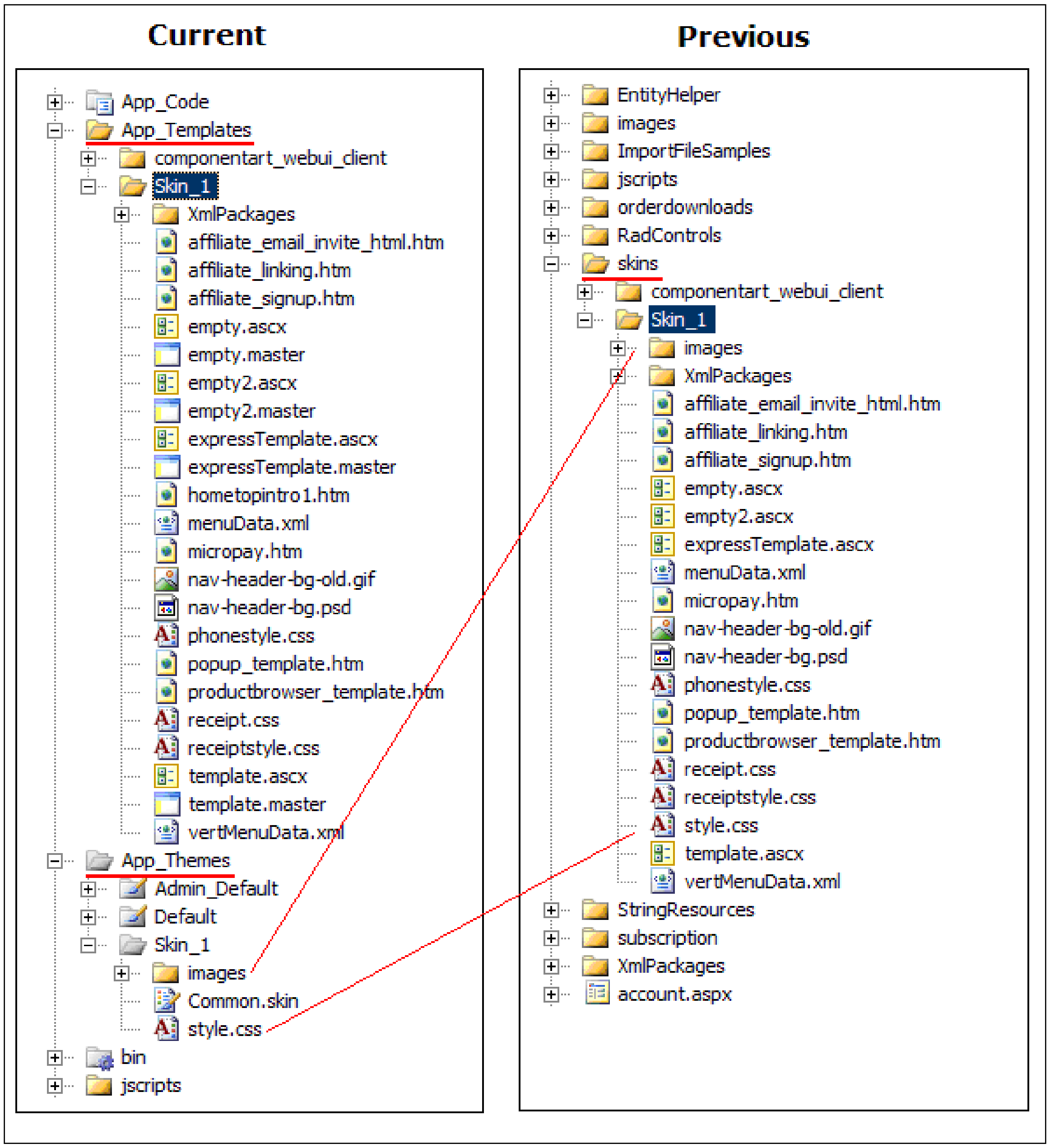
This guide explains the new structure of skins/themes in version 9 of the AspDotNetStorefront software, and goes over how to convert from an older skin to an asp.net master page.

Asp.net Themes and Directory Structure Changes

Along with the move to master pages, we have also added support for Asp.net Themes. Themes allow for easier skinning of an entire site and give better 'design' support in Visual Studio than our previous skinning engine.

Asp.net Themes require a predefined folder named App_Themes, which is a reserved folder for asp.net that can only contain style.css, images, and .skin files. The other files that we need (such as XML packages, .htm files, etc) go in another folder called App_Template.



```
Current
App_Code
App_Templates
    componentart_webui_client
    Skin_1
        XmlPackages
        affiliate_email_invite_html.htm
        affiliate_linking.htm
        affiliate_signup.htm
        empty.ascx
        empty.master
        empty2.ascx
        empty2.master
        expressTemplate.ascx
        expressTemplate.master
        hometopintro1.htm
        menuData.xml
        micropay.htm
        nav-header-bg-old.gif
        nav-header-bg.psd
        phonestyle.css
        popup_template.htm
        productbrowser_template.htm
        receipt.css
        receiptstyle.css
        template.ascx
        template.master
        vertMenuData.xml
App_Themes
    Admin_Default
    Default
    Skin_1
        images
        Common.skin
        style.css
bin
jscripts
```

```
Previous
EntityHelper
images
ImportFileSamples
jscripts
orderdownloads
RadControls
skins
    componentart_webui_client
    Skin_1
        images
        XmlPackages
        affiliate_email_invite_html.htm
        affiliate_linking.htm
        affiliate_signup.htm
        empty.ascx
        empty2.ascx
        expressTemplate.ascx
        menuData.xml
        micropay.htm
        nav-header-bg-old.gif
        nav-header-bg.psd
        phonestyle.css
        popup_template.htm
        productbrowser_template.htm
        receipt.css
        receiptstyle.css
        style.css
        template.ascx
        vertMenuData.xml
StringResources
subscription
XmlPackages
account.aspx
```

In the image above, you will notice that new files have been added to the skins folder. Empty.master, empty2.master, expresstemplate.master, and template.master are the master page equivalent of the old skinning engine's empty.ascx, empty2.ascx, expresstemplate.ascx, and template.ascx.

Creating a New Skin

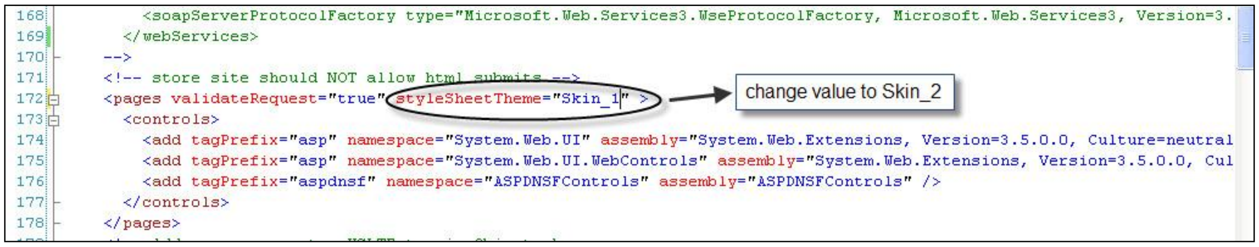
If you're going to design a new master page skin for the first time, it's best to start out by creating a new skin, rather than modifying the 'stock' skin. This way there is always a backup to recover from if necessary, and you can easily compare changes you make to how the original working skin is laid out. To create a new 'skin 2', simply follow these steps:

1- Open AspDotNetStorefront in Visual Studio and go to the App_Themes folder in the root of the site you installed. Create a new folder named Skin_N where N is the number of the skin you want to create (example: Skin_2).
2- Copy the contents of the App_Themes/Skin_1 folder to your new folder. Make sure that you copy over the images folder, the style.css file, and the common.skin file.
3- Next, create a folder within the App_Templates folder with the same name as the folder you created in step 1.
4- Copy the contents of App_Templates/Skin_1 to your new folder.
5- You can now modify the site's skin in the template.master file. As that is a standard .NET master page with asp.net themes, much of the skin design can be done in Design view in Visual Studio.

Using a New Skin

After installation, skin 1 is the default skin assigned to the store. To change that so all customers see the skin you created, there are 2 steps:

- Change the DefaultSkinID AppConfig to the number of the skin you created
- In the web.config file, change the page styleSheetTheme attribute to the theme you created:



```
        <soapServerProtocolFactory type="Microsoft.Web.Services3.WseProtocolFactory, Microsoft.Web.Services3, Version=3.
      </webServices>
    -->
    <!-- store site should NOT allow html submits -->
    <pages validateRequest="true" styleSheetTheme="Skin_1" >
      <controls>
        <add tagPrefix="asp" namespace="System.Web.UI" assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral
        <add tagPrefix="asp" namespace="System.Web.UI.WebControls" assembly="System.Web.Extensions, Version=3.5.0.0, Cul
        <add tagPrefix="aspdnsf" namespace="ASPDNSFControls" assembly="ASPDNSFControls" />
      </controls>
    </pages>
```

It is also possible to have multiple skins functioning on the site at one time. The simplest way to do this is by manually invoking the skin in a link, like this:

http://www.yoursite.com/default.aspx?skinid=#

(where # is the number of the skin you want to invoke)

You can also have the same customer view different parts of the site in different skins, using the directions at http://manual.aspdotnetstorefront.com/p-831-varying-the-skin-within-a-site.aspx

Tokens

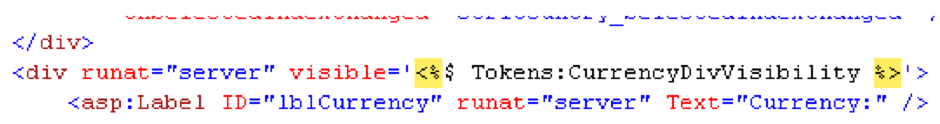
Skin tokens are tags that can be inserted into the site's skin, and are parsed out at runtime and replaced with common dynamic elements like the minicart, the customer's name, common text/hyperlinks, etc. Customers familiar with AspDotNetStorefront's older versions may remember the old (!TOKENNAME!) format for skin tokens. Beginning in version 9, tokens are implemented via asp.net's custom expression style in this format:

<%$ Tokens:{token name} {optional parameters} %>

As an example:

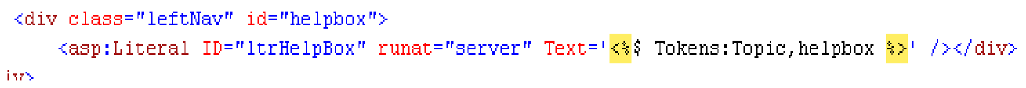
(!PAGEINFO!) would become <%$ Tokens:PAGEINFO %>

This new parsing method allows developers to view dynamic tokens in design view in Visual Studio which was not possible in older builds. Some of the new tokens that return strings can also be used to set attributes on page elements. For example:



```
</div>
<div runat="server" visible='<%$ Tokens:CurrencyDivVisibility %>'>
    <asp:Label ID="lblCurrency" runat="server" Text="Currency:" />
```

Or…



```
<div class="leftNav" id="helpbox">
    <asp:Literal ID="ltrHelpBox" runat="server" Text='<%$ Tokens:Topic,helpbox %>' /></div>
iv>
```

There is a complete list of the supported skin tokens in the online manual at http://www.manual.com/p-1064-skin-tokens.aspx

Converting a Skinbase Template to Master Pages

To convert an older AspDotNetStorefront skin to a master page, follow these steps:

Head:

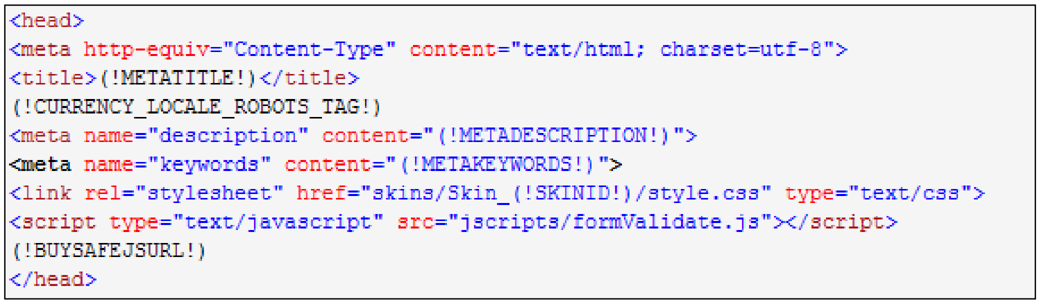
1 – The <head> tag of the template file must have the runat="server" attribute.

2 – Remove any stylesheet references. They will be added automatically by asp.net to match the theme used

3 – Add a javascript reference to core.js, located in the root/jscripts folder.

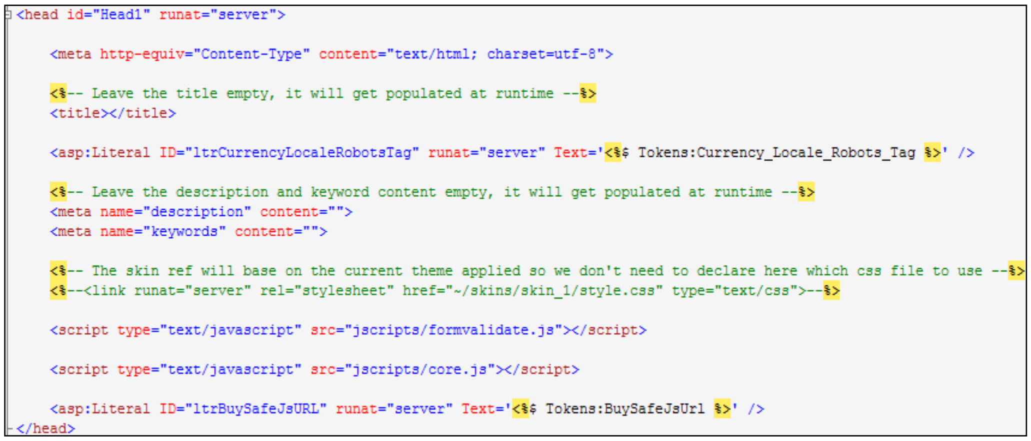
4 – Replace old skinbase parser tokens with the new expression format (see the Tokens section of this document for more information)

After these steps, your <head> section should go from looking something like this:



```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>(!METATITLE!)</title>
(!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>
(!BUYSAFEJSURL!)
</head>
```

To more like this:

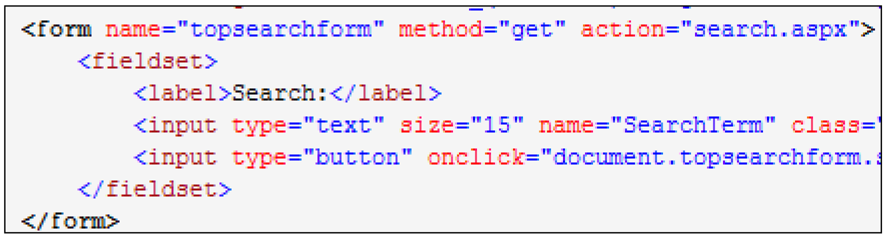


```
<head id="Head1" runat="server">

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

    <%-- Leave the title empty, it will get populated at runtime --%>
    <title></title>

    <asp:Literal ID="ltrCurrencyLocaleRobotsTag" runat="server" Text='<%$ Tokens:Currency_Locale_Robots_Tag %>' />

    <%-- Leave the description and keyword content empty, it will get populated at runtime --%>
    <meta name="description" content="">
    <meta name="keywords" content="">

    <%-- The skin ref will base on the current theme applied so we don't need to declare here which css file to use --%>
    <%--<link runat="server" rel="stylesheet" href="~/skins/skin_1/style.css" type="text/css">--%>

    <script type="text/javascript" src="jscripts/formvalidate.js"></script>

    <script type="text/javascript" src="jscripts/core.js"></script>

    <asp:Literal ID="ltrBuySafeJsURL" runat="server" Text='<%$ Tokens:BuySafeJsUrl %>' />
</head>
```

Body:

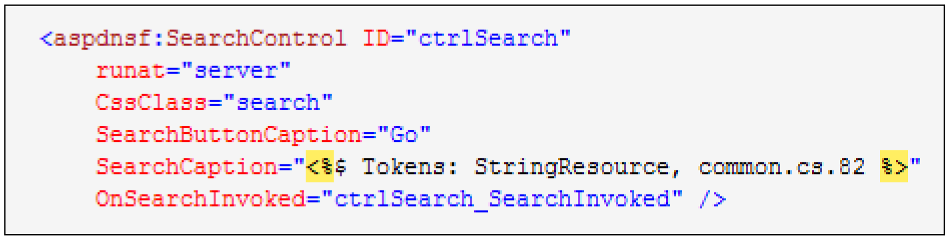
1 – Add a <form> tag to the page, preferably right after the opening <body> tag. Master pages only allow one form per page, so duplicates have to be removed. These are generally replaced with controls. One example would be the search function.

Instead of this:



```
<form name="topsearchform" method="get" action="search.aspx">
    <fieldset>
        <label>Search:</label>
        <input type="text" size="15" name="SearchTerm" class="
        <input type="button" onclick="document.topsearchform.s
    </fieldset>
</form>
```

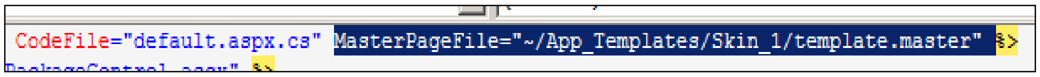
You would use something like this:



```
 <aspdnsf:SearchControl ID="ctrlSearch"
     runat="server"
     CssClass="search"
     SearchButtonCaption="Go"
     SearchCaption="<%$ Tokens: StringResource, common.cs.82 %>"
     OnSearchInvoked="ctrlSearch_SearchInvoked" />
```

2 – Convert any custom .aspx pages you've created into master page child pages. To do this:

- ○ Add the MasterPageFile directive:

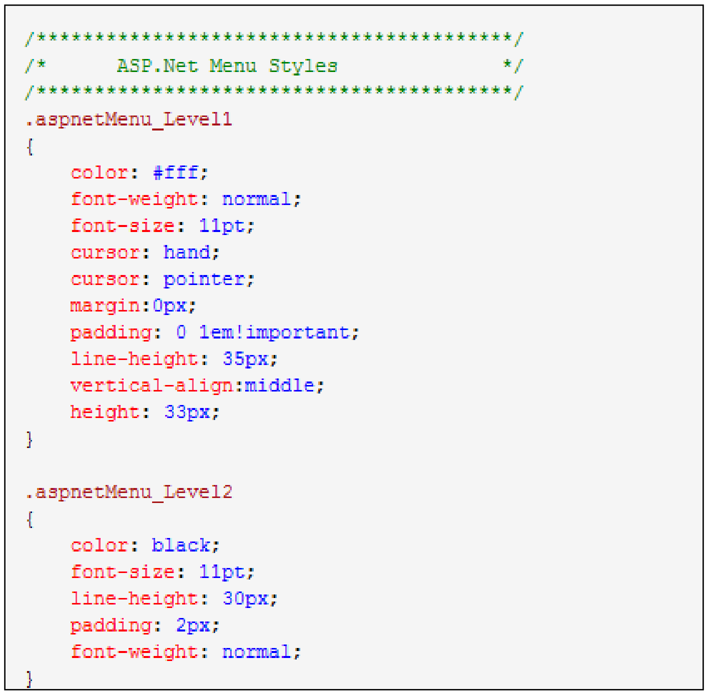


```
CodeFile="default.aspx.cs" MasterPageFile="~/App_Templates/Skin_1/template.master" %>
```

- ○ Remove any HTML other than what should be present in the content area itself.
- ○ Add an asp.net *Content* control as the root control on the page. Make sure that ContentPlaceHolderID is "PageContent".
- ○ Add a panel with an ID of "pnlContent" as the only child control of the Content control.
- ○ Insert the contents of the page inside the *pnlContent* control.

3 – Replace the ComponentArt menu with the asp.net Menu control. All references to ComponentArt will need to be removed from the template, as support for them is being dropped.

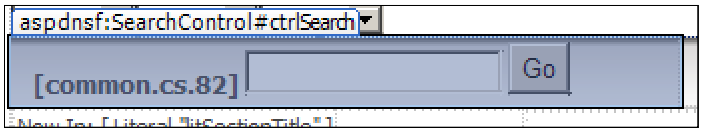
The default look and feel of the asp.net menu is the same as the old ComponentArt menu, but it uses a different CSS style. The asp.net menu control uses its own styles, which can be found in the theme’s style.css file. The class names have an ‘aspnetMenu’ prefix to make finding them easier:



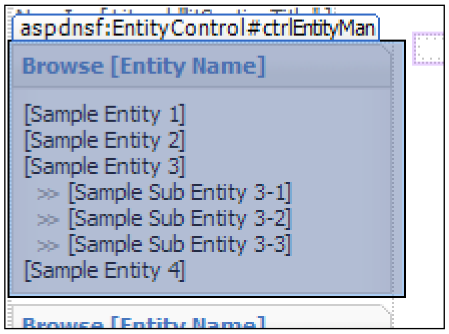
```
/*****************************************/
/*      ASP.Net Menu Styles              */
/*****************************************/
.aspnetMenu_Level1
{
    color: #fff;
    font-weight: normal;
    font-size: 11pt;
    cursor: hand;
    cursor: pointer;
    margin:0px;
    padding: 0 1em!important;
    line-height: 35px;
    vertical-align:middle;
    height: 33px;
}

.aspnetMenu_Level2
{
    color: black;
    font-size: 11pt;
    line-height: 30px;
    padding: 2px;
    font-weight: normal;
}
```

4 – Incorporate new controls where necessary. Many of the old parser tokens have been replaced by controls, which can be used in the master page. See the template.master file for skin 1 for examples of how these controls can be used.

Search Control:

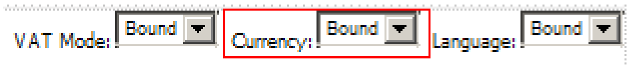


Entity Control:


aspdnsf:EntityControl#ctrlEntityMan
Browse [Entity Name]
[Sample Entity 1]
[Sample Entity 2]
[Sample Entity 3]
>> [Sample Sub Entity 3-1]
>> [Sample Sub Entity 3-2]
>> [Sample Sub Entity 3-3]
[Sample Entity 4]
Browse [Entity Name]

- Displays the vertical menu for entities like Categories, Manufacturers, and Departments
- Replaces the rev XML packages

Select List Controls:



- These controls provide options to change settings like:
    - Locale
    - Currency
    - VAT Included/Excluded

Frequently Asked Questions

Q: Am I allowed to add another content placeholder on my default MasterPage?

> A: Yes. You will need to handle that per page if you want to introduce content.

Q: Can I use my existing ComponentArt menu?

> A: As of version 9, the ComponentArt menus will no longer be supported.

Q: Can I use my customized ComponentArt menu styles?

> A: The old styles have been deprecated with the end of support for the ComponentArt menus. Use the new asp.net styles prefixed by 'aspnetMenu' in the style sheet.

Q: Will Parser tokens still be supported?

> A: Some parser tokens will still function, though many have been removed as they can be replaced by much more flexible controls. See the manual for a full list of functioning tokens at http://www.manual.com/p-1064-skin-tokens.aspx

Troubleshooting Guide

- The master page won't show a preview in DreamWeaver, FrontPage, etc
  - Dreamweaver doesn't support preview mode on all file types, including master pages. The preferred editor is Visual Studio.

- Pages just refresh after postback actions
  - Ensure that there are no nested <form> tags within the page, as master pages don't support multiple forms per page. If a custom page requires multiple forms, it should not be set to inherit the master page.

- You receive an error: "Cannot set property value to property name for a user control" when using design mode in Visual Studio.
  - This is a VS 2008 bug. You may have to restart the IDE to view the control. See https://connect.microsoft.com/VisualStudio/feedback/ViewFeedback.aspx?FeedbackID=361826