.NET Best Practices
Part 1 Master Pages Setup

Version 2.0

Table of Contents

Introduction **3**

What is a Master Page? 3

Getting Started **4**

Scenario -A-: Create a new Template Folder via Volte 4

Scenario –B-: Create a new Template Folder via CDC 5

Create a new C# Template for Master Page 5

Configure the input.aspx template file 6

Create an asset in CMS and link it to the MasterPage template you just created. 9

Configure the Output.aspx template file 10

Create filename.aspx template file: 12

Create OutputHelper.MainHeaderSetup 13

Create a C# Bridge template 14

Build page template 15

Configure input.aspx 15

Create an asset in the Example Folder and link it to the ContentPage template. 16

Configure output.aspx 16

.NET Publishing **18**

Next Steps **18**

Appendix **19**

Inputhelper.SetSelectlink 19

Outputhelper.MetaStringBuilder 19

TDW.MakeFilename 23

TDW.FormatFilename 23

Introduction

This document describes the structure, process, and best practices to configure CrownPeak to publish an ASP.NET site to your web host. It will show ASP.NET Master Page's basic concepts in order to demonstrate how the pages work, but this document assumes experience in building and configuring .NET web sites. This document also assumes knowledge of the CrownPeak API and template best practices.

What is a Master Page?

Master Page(*.master) is a feature that enables you to define common structure and interface markup elements for the Web site, including headers, footers, style definitions, or navigation bars. The master page consists of two pieces, the master page itself and one or more content pages (*.aspx).

Getting Started

Create a new Template Folder to store the templates for this site. Developers can use the Volte UI or CrownPeak Desktop Connection (CDC) to create templates

Scenario -A-: Create a new Template Folder via Volte

Browse to System >Templates, then click New > Template Folder as shown in figure -1-. Name your project and press the OK button.

Example: {YourName}_{FolderCreationDate}_Templates.

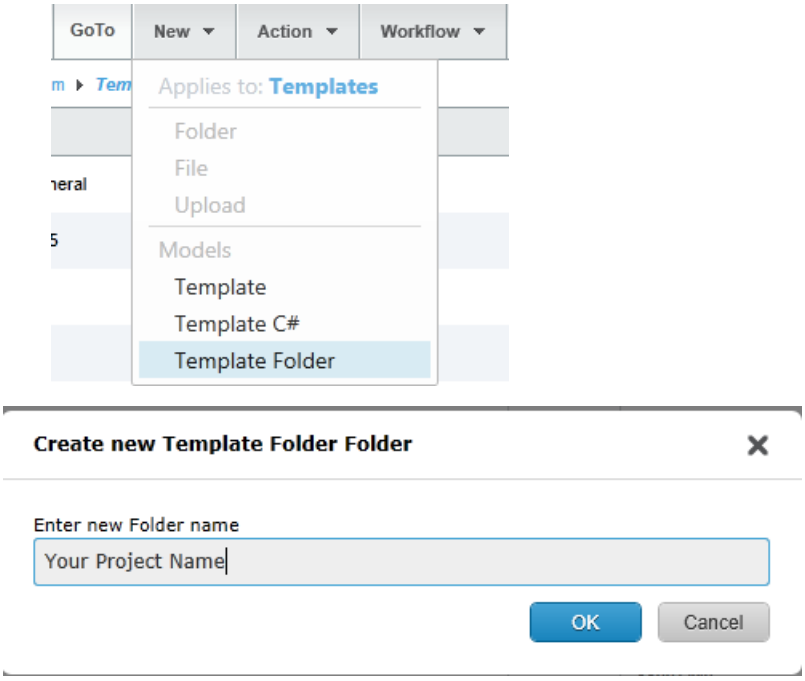

Figure 1: Create a Template Folder

Scenario –B-: Create a new Template Folder via CDC

Browse to System >Templates, then right-click on Templates folder and click New>Template Folder as shown in figure -2-.

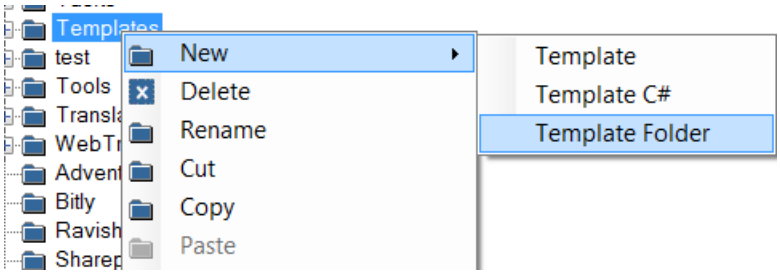

Figure 2: CDC scenario to create a Template Folder

Create a new C# Template for Master Page

Create a C# template in Template folder you just created and name it “MasterPage” as shown in figure -3-. This will be the masterpage file published to the web host and the wrapper used by CrownPeak to when pages are previewed.

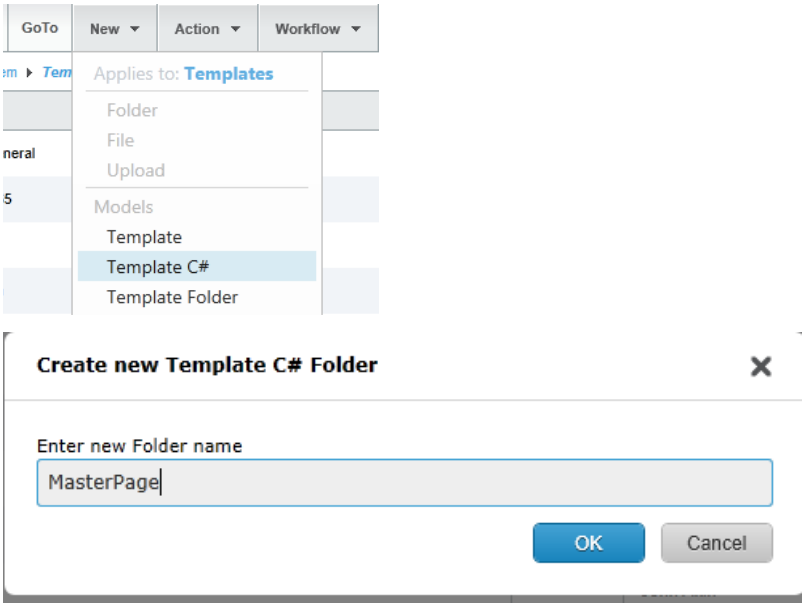

Figure 3: Create a new C# template

You will see the following window:

Figure 4: Template Folder content view

When you double-click on MasterPage template, you will see figure -5-:

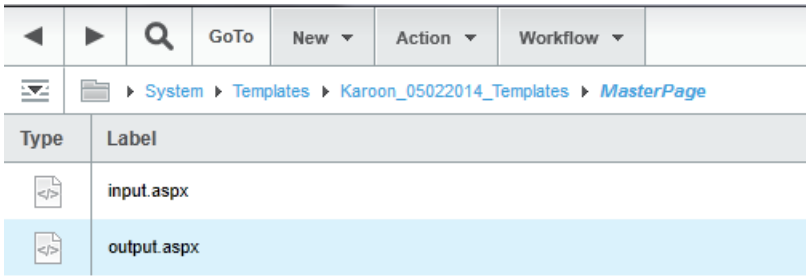

Figure 5: Template Folder with template input and output files

Configure the input.aspx template file

Consider the following example:

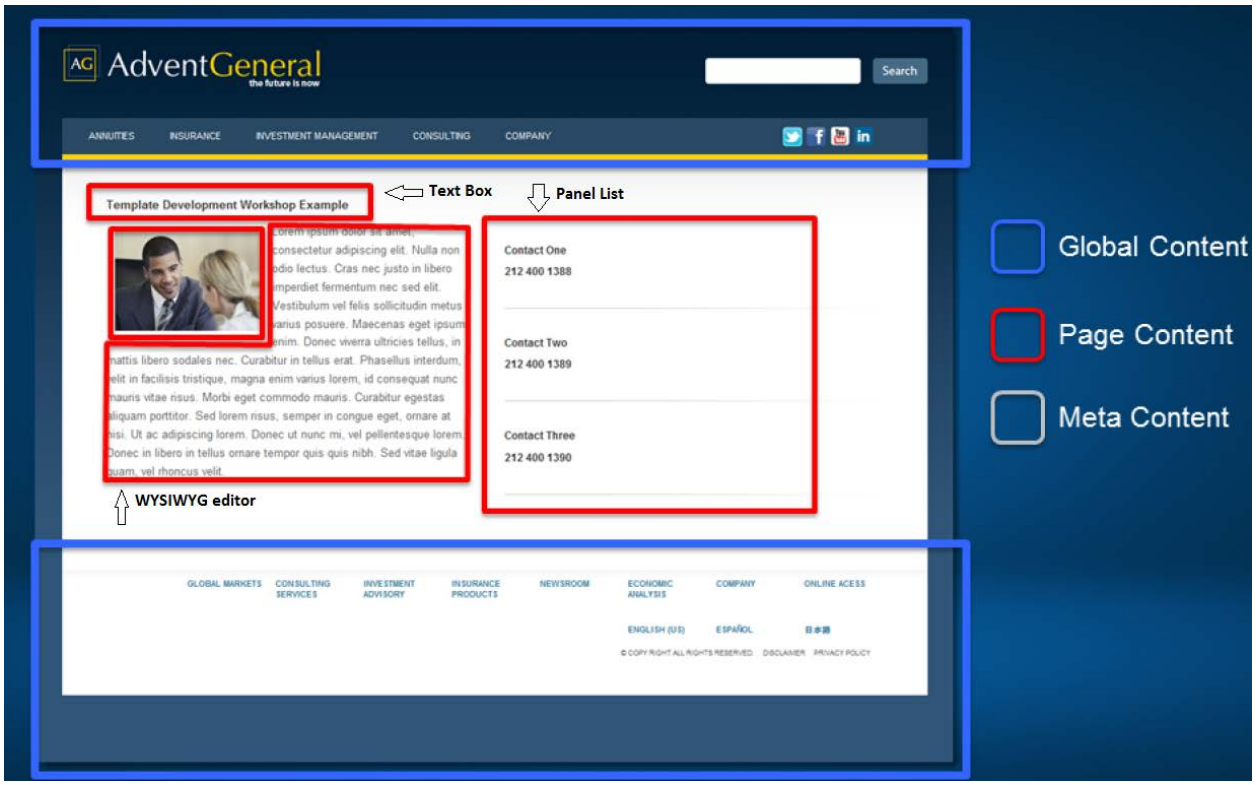

Figure 6: Web Page Example

This file is able to manage header/footer menu and specify JavaScript/CSS files to be loaded on every page. In general, we enable user to create lists of header and footer links using panels. We enable the same feature for CSS and JS files. We also include text fields so code can be added directly to the page.

There may be some differences in how you enable users to manage the content based on the site requirements, but here is an example of the input.aspx that represents our best practices –

```
<%@ Page Language="C#" Inherits="CrownPeak.Internal.Debug.InputInit" %>
        <%@ Import Namespace="CrownPeak.CMSAPI" %>
        <%@ Import Namespace="CrownPeak.CMSAPI.Services" %>
        <%@ Import Namespace="CrownPeak.CMSAPI.CustomLibrary" %>
        <!--DO NOT MODIFY CODE ABOVE THIS LINE-->
        <%//This plugin uses InputContext as its context class type%>
        <%
        // input.aspx: template file to specify content entry input form

            Input.StartTabbedPanel("Header", "Footer");
            //Logo
                Input.StartControlPanel("Logo");
                ShowAcquireParams sapImg = new ShowAcquireParams();
                sapImg.ShowBrowse = true;
                sapImg.ShowUpload = true;
                sapImg.AddAdditionalImage("image_thumbnail", 20, 20, 100);
                Input.ShowAcquireImage("Image", "logo_image");
              //  Input.ShowAcquireImage("Logo Upload", "logo_upload");
                Input.ShowTextBox("Image Alt", "logo_image_alt");
                Input.EndControlPanel();


                Input.ShowHeader("Main Navigation Menu");
                while (Input.NextPanel("main_nav_panel"))
                {
                    InputHelper.SetSelectLink(
                    "Main Menu",
                    "main_nav_menu",
                    "Link Type",
                    "main_nav_menu_link_type",
                    "Select a link",
                    "main_nav_menu_link_internal",
                    "URL",
                    "main_nav_menu_link_external");

                }
                Input.NextTabbedPanel();

                Input.ShowHeader("Footer Menu");
                Input.ShowMessage("List");
                    while (Input.NextPanel("footer_column_menu_panel"))
                    {
                        InputHelper.SetSelectLink(
                            "Link Text",
                            "footer_column_menu_link_text",
                            "Link Type",
                            "footer_column_menu_link_type",
                            "Select a link",
                            "footer_column_menu_link_internal",
                            "URL",
                            "footer_column_menu_link_external");
                    }
```

```
    Input.EndTabbedPanel();
%>
```

Create an asset in CMS and link it to the MasterPage template you just created.

This asset will be used later in output.aspx of both Master Page and Bridge template to retrieve user input and display it on the output. See figure -7- and follow the steps:

1. Browse to Sandbox, then click New > Folder, name it, and click “Save”.

   Example of Folder name: {YourName}_{Date}_Example Folder.

2. Inside the folder, create a new folder to include the asset (File) and link it to the MasterPage template.
3. Click New > Folder > must name it “_Master”.
4. Inside “_Master”, New > File:
5. Label: name it “Master Page”.
6. Select a Template: Click “Select” to browse to MasterPage template you created.
7. Select a Workflow: Choose “Basic Workflow” from the drop-down menu.

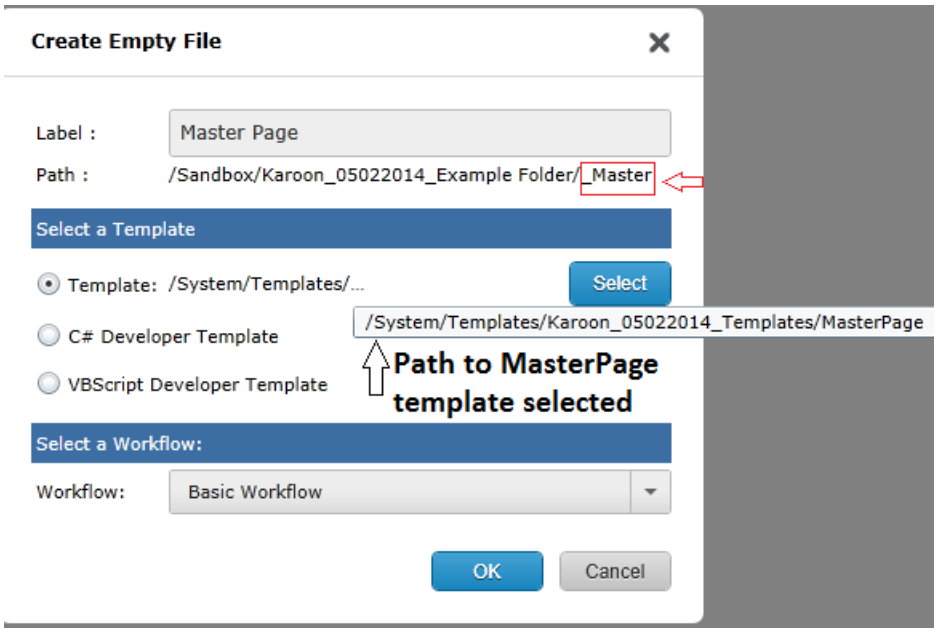

Figure 7: Create a Master Page asset

You will see figure -8-:

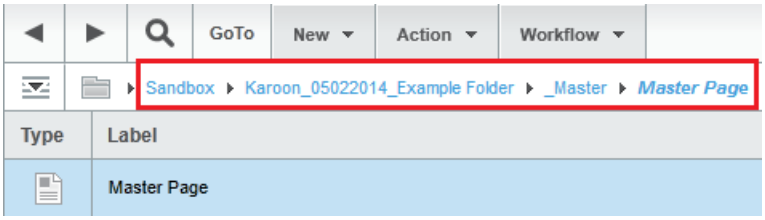

Figure 8: Master Page asset

Configure the Output.aspx template file

On an ASP.NET site, the syntax used to denote a masterpage is <%@ Master Language="C#" %>. In the output.aspx template file, include this code by using the context object to determine if the page is being published or previewed. This will prevent the ASP.NET from being included in the CMS preview, where it is not used.

Escape the % symbol to a $, so it looks like this <$@ Master Language="C#" $>. This will prevent the CMS running this code. The dollar sign will be converted back to a percent symbol when published to the Web host.

You need to create a CSS file for the example. You can create anywhere. Here it's created as follows:

1. Navigate to your Example Folder > New > File >
2. Label: name it "CSS.css".
3. Select a C# Developer Template.
4. Select a Workflow: Choose "Basic Workflow" from the drop down menu
5. Then paste the code displayed in Appendix.

```
<%@ Page Language="C#" Inherits="CrownPeak.Internal.Debug.OutputInit" %>
<%@ Import Namespace="CrownPeak.CMSAPI" %>
<%@ Import Namespace="CrownPeak.CMSAPI.Services" %>
<%@ Import Namespace="CrownPeak.CMSAPI.CustomLibrary" %>
<!--DO NOT MODIFY CODE ABOVE THIS LINE-->
<%//This plugin uses OutputContext as its context class type%>
<%
// output.aspx: template file to specify the published content in site HTML
// if no preview.aspx exists, then this is used by default for preview

Out.WriteLine(context.IsPublishing ? "<$@ Master Language=\"C#\" AutoEventWireup=\"true\" $>"
: "");
bool bMasterPage = true;
Asset aMasterPage = Asset.Load("/" + asset.AssetPath[0] + "/Karoon_05022014_Example
Folder/_Master/Master Page");

if (aMasterPage.IsLoaded)
{
    if (!int.Equals(asset.Id, aMasterPage.Id))
        bMasterPage = false;
}
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <%=OutputHelper.LoadCSS("/Sandbox/Karoon_05022014_Example Folder/CSS.css")%>
       <%Out.WriteLine(context.IsPublishing ? "<asp:ContentPlaceHolder id=\"cphHead\"
runat=\"server\"/>" : ""); %>
    </head>

    <body>
        <div id="wrapper">
            <div class="w1">
                <div id="header">
                    <div class="user-panel">
                        <% //Out.DebugWriteLine(aMasterPage.Label);%>

                                <%Out.Write("<h1 class=\"logo\" style=\"background-
image:url('{0}');\" />AdventGenaral</h1>",aMasterPage["logo_image"]); %>
                                <div class="search-form" id="searchForm" runat="server"><!--
action="#" -->
                                    <fieldset>
                                        <span class="text"><input type="text"
id="search"/></span>
                                        <input type="submit" class="submit" value="Go"/>
                                    </fieldset>
                                </div>
                        </div>

                        <div id="navigation" class="navigation">
                          <%List<PanelEntry> menu = aMasterPage.GetPanels("main_nav_panel");%>

                           <ul id="nav">
                            <li id="homepage"><!-- No Page Loaded from Sandbox/Homepage--
></li><!--Annuities-->
                        <%
                            foreach (PanelEntry peMenu in menu)
                            {
                                if (!string.IsNullOrWhiteSpace(peMenu.Raw["main_nav_menu"]))
                                {
                                    Out.Write("<li>{0}</li>", peMenu["main_nav_menu"]);
                                }
                            }
                        %>
                        </ul>
                        <div class="menu">
                            <ul class="social-networks">
                                <li><a href="#" class="twitter">twitter</a></li>
                                <li><a href="#" class="facebook">facebook</a></li>
                                <li><a href="#" class="youtube">youtube</a></li>
                                <li><a href="#" class="linkedin">linkedin</a></li>
                            </ul>
                        </div>

                    </div><!--Closing navigation-->
                </div><!--Closing header-->
                <%Out.WriteLine(context.IsPublishing ? "<asp:ContentPlaceHolder id=\"cphMain\"
runat=\"server\"/>" :
                           (bMasterPage ? "" : Out.GetWrapContentPlaceholder())); %>


                <div id="footer">
                      <%List<PanelEntry> footerMenu =
aMasterPage.GetPanels("footer_column_menu_panel");%>
```

```
                        <ul>
                    <%
                        foreach (PanelEntry peMenu in footerMenu)
                        {
                            if
(!string.IsNullOrWhiteSpace(peMenu.Raw["footer_column_menu_link_text"]))
                            {
                                Out.Write("<li><h5>{0}</h5></li>",
peMenu["footer_column_menu_link_text"]);
                            }
                        }
                    %>
                    </ul>
                    <div style="clear:both;"></div>
                    <ul>
                        <li><h5>English (US)</h5></li>
                        <li><h5>Japanese</h5></li>
                        <li><h5>Espa&#241;ol</h5></li>
                    </ul>
                    <div style="clear:both;"></div>
                     <div class="disclaimer">
                        <ul>
                            <li>&#169; Copy Right All Rights Reserved.</li>
                            <li><a href="#">Disclaimer</a></li>
                            <li><a href="#">Privacy Policy</a></li>
                        </ul>
                    </div>
                </div><!--Closing footer div-->
            </div><!--Closing w1 div-->
        </div><!--Closing wrapper div-->
    </body>
</html>
```

Create filename.aspx template file:

As mentioned earlier, master page has extension (.master), so the published filename of master page must be (.master) instead of (.aspx).

Navigate to MasterPage template folder > New > filename.aspx

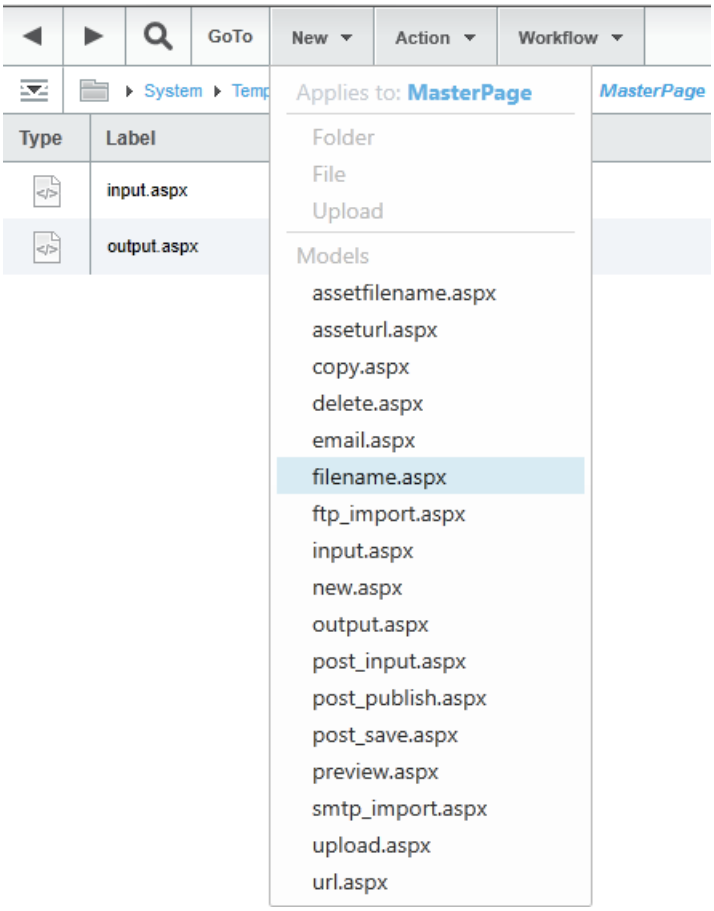

Figure 9:filename.aspx

Insert the following line of code:

```
<%context.PublishPath = TDW.MakeFilename (context, asset);%>
```

"MakeFilename" method in the library class, TDW, checks if the master page asset is in a folder called, "_Master", then replaces the extension with ".master".

On Volte, after publishing all assets to "Live", navigate to the master page asset > Properties > General, you should see all published URL's and filenames with (.master) extension.

Create OutputHelper.MainHeaderSetup

In /System/Library, create or edit the OutputHelper class. Create a method called MainHeaderSetup. This will manage the use of the bridge template. If the page is being published to the webhost, the bridge template will be used. If being previewed in the CMS, it will just use the Master page template to assemble the preview.

```
public static void MainHeaderSetup(OutputContext context)
{
    if (context.IsPublishing)
        Out.Wrap("System/Templates/{YourTemplatesFolder}/BridgePage");
    else
        Out.Wrap("/System/Templates/{YourTemplatesFolder}/MasterPage");
}
```

Create a C# Bridge template

Create a bridge template inside the folder you created in step -1- and call it **"**BridgePage**"**. This is a helper template to insert ASP.NET code into all the page templates. Follow the same steps and naming convention to create the template; you do not need to create an asset and link it to Bridge template. Bridge template separates the code needed for publishing from what is need to just preview the page in CrownPeak. In other words, the bridge template includes all the code necessary to make the published page work with the master page on the webhost. See figure -10.

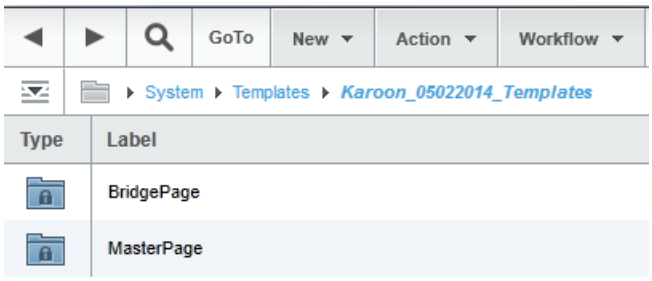

Figure 10: Bridge Template

Note the use of placeholders to include meta content and page specific scripts when published. Here we are populating the cphHead placeholder with content specific to the page using the OutputHelper.MetaStringBuilder method.

```
<%@ Page Language="C#" Inherits="CrownPeak.Internal.Debug.OutputInit" %>
<%@ Import Namespace="CrownPeak.CMSAPI" %>
<%@ Import Namespace="CrownPeak.CMSAPI.Services" %>
<%@ Import Namespace="CrownPeak.CMSAPI.CustomLibrary" %>
<!--DO NOT MODIFY CODE ABOVE THIS LINE-->
<%//This plugin uses OutputContext as its context class type%>

<%
// output.aspx: template file to specify the published content in site HTML
// if no preview.aspx exists, then this is used by default for preview

Asset aMasterPage = Asset.Load("/" + asset.AssetPath[0] + "/Karoon_05022014_Example
Folder/_Master/Master Page");
Out.WriteLine("<$@ Page Language=\"C#\" MasterPageFile=\"{0}\" $>", "~"
+aMasterPage.GetLink());
Out.DebugWriteLine(aMasterPage.GetLink());
Out.WriteLine("<asp:Content Id=\"ctHead\" ContentPlaceHolderId=\"cphHead\"
runat=\"Server\" >");
Out.WriteLine(OutputHelper.MetaStringBuilder(asset, aMasterPage));

if (!String.IsNullOrWhiteSpace(asset.Raw["custom_script"]))
{
```

```
            Out.WriteLine(asset["custom_script"]);
        }

        Out.WriteLine("</asp:Content>");
        Out.WriteLine("<asp:Content Id=\"ctMain\" ContentPlaceHolderId=\"cphMain\"
        runat=\"Server\" >");
        Out.Write(Out.GetWrapContentPlaceholder());
        Out.WriteLine("</asp:Content>");

%>
```

Build page template

Create a page content template.

Browse to your Templates folder you created on step -1-. Click New > Template C# and name it, "ContentPage". See figure -11.

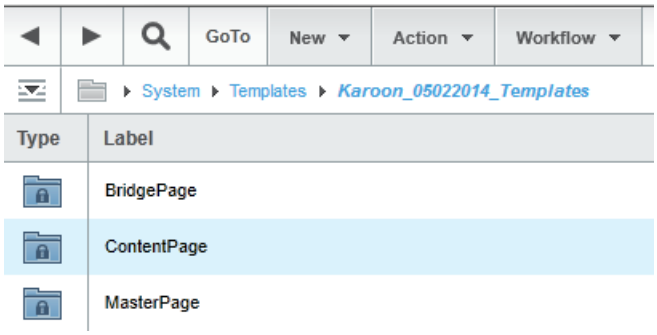

Figure 11: Content Page Template

Configure input.aspx

In this template file, the user can input the desired content by editing the fields that will be added in the code as follows: See figure -6-.

```
<%@ Page Language="C#" Inherits="CrownPeak.Internal.Debug.InputInit" %>
<%@ Import Namespace="CrownPeak.CMSAPI" %>
<%@ Import Namespace="CrownPeak.CMSAPI.Services" %>
<%@ Import Namespace="CrownPeak.CMSAPI.CustomLibrary" %>
<!--DO NOT MODIFY CODE ABOVE THIS LINE-->
<%//This plugin uses InputContext as its context class type%>
<%
// input.aspx: template file to specify content entry input form
    Input.ShowTextBox("Page Title", "page_title",width:50,defaultValue:"Title");

    ShowAcquireParams imgParams = new ShowAcquireParams();
```

```
imgParams.DefaultFolder = "/Sanbox/Assets";
imgParams.AddAdditionalImage("image_thumbnail",20,20,100);
imgParams.ShowUpload = false;
Input.ShowAcquireImage("Upload Image", "upload_image");

WysiwygParams wysiParams = ServicesInput.FullWYSIWYG();
wysiParams.Stylesheet = "/Sandbox/KaroonContentPage040814/CSSFile.css";
Input.ShowWysiwyg("Text Area", "text_area", wysiParams);

while (Input.NextPanel("counter_panel",displayName:"Contact information"))
{
    Input.ShowTextBox("Name", "name");
    Input.ShowTextBox("Telephone", "telephone");
}
%>
```

Create an asset in the Example Folder and link it to the ContentPage template.

Name it, “Content Page” as explained in step -4.2- on page 10. When you double-click on this asset and click Form tab, you can edit the fields as you desire. See figure -12-.

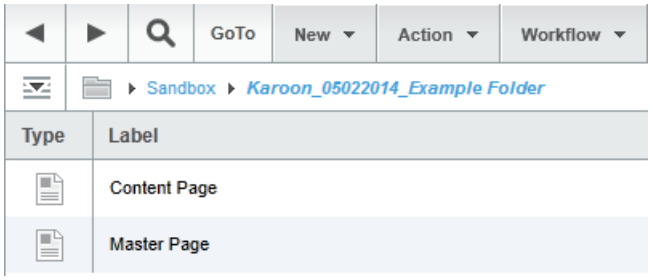

Figure 12: Content Page Asset

Configure output.aspx

This template file should include the wrapper as well as the content that will be retrieved from user input. As usual, the output files of your page templates will include the front end code to render the page layout and place content. In this case, they will also call the OutputHelper.MainHeaderSetup method.

Add the following text to output.aspx:

```
<%@ Page Language="C#" Inherits="CrownPeak.Internal.Debug.OutputInit" %>

<%@ Import Namespace="CrownPeak.CMSAPI" %>
<%@ Import Namespace="CrownPeak.CMSAPI.Services" %>
<%@ Import Namespace="CrownPeak.CMSAPI.CustomLibrary" %>
<!--DO NOT MODIFY CODE ABOVE THIS LINE-->
<%
  //  Asset gConfigAsset = OutputHelper.GetGlobalConfig(asset);
```

```
        OutputHelper.MainHeaderSetupTest(context);

    %>

    <div id="main">
        <div class="container" id="contentpage">
            <div class="container-holder">
                <div class="container-frame">
                    <div id="twocolumns">
                        <div id="content">
                            <h1><%=asset["page_title"]; %></h1>
                            <% Out.WriteLine("<img height=\"113\" width=\"168\"
    style=\"float:left;padding:10px;\"src=\"{0}\" />", asset["upload_image"]);%>
                            <%=asset["text_area"]; %>
                            <p>This is some more content</p>
                        </div>
                        <div id="sidebar">
                            <div class="box">
                                <div class="box-content">

                                    <%
                                    List<PanelEntry> sectionItems =
    asset.GetPanels("counter_panel");
                                    foreach (PanelEntry sectionItem in sectionItems)
                                        {
                                        %>

                                        <%Out.WriteLine("<h1>{0}</h1>", sectionItem["name"]);
                                        Out.WriteLine("<h1>{0}</h1>",
    sectionItem["telephone"]); %>
                                        <div class="box_divider"></div>
                                        <%} %>
                                </div><!--Closing box-content-->
                            </div><!--Closing box-->
                        </div><!--Closing sidebar-->
                    </div><!--Closing twocolumns-->
                </div><!--Closing container-frame-->
            </div><!--Closing container-holder-->
        </div><!--Closing contentpage-->
</div><!--Closing main-->
```

.NET Publishing

To test your URL's, browse to Sandbox > YourExampleAssets folder, click on the asset linked to the page content; on the right panel, click on "General"; on the left side, scroll down to "Current URL(s)", then click on the Stage Site URL. You should see the complete web page.

Next Steps

Now you have a master page and bridge template and you should have created at least one page template. The next step will be to start publishing to the staging webhost in order to validate the publishing configuration and the template code. The next document will outline these steps and others including

- .NET Site structure in the CMS
- .NET publishing configuration
- Configuring different filename extension for different .NET file type.
- Advanced topics like implementing user controls

Appendix

Inputhelper.SetSelectlink

```
//This method enables the user to create list(s) of menus for the navigation bar of header and
footer
    public static void SetSelectLink(string szLinkLabel, string szLinkTextFieldName, string
szLinkTypeLabel, string szLinkTypeFieldName, string szInternalLinkLabel, string
szInternalLinkFieldName, string szExternalLinkLabel, string szExternalLinkFieldName, string
szTargetFieldName = "")
    {
        Dictionary<string, string> dtLinkType = new Dictionary<string, string>();
        dtLinkType.Add("None", "");
        dtLinkType.Add("Internal", "internal");
        dtLinkType.Add("External", "external");

        if (!string.IsNullOrWhiteSpace(szLinkLabel) &&
!string.IsNullOrWhiteSpace(szLinkTextFieldName))
            Input.ShowTextBox(szLinkLabel, szLinkTextFieldName);

        Input.StartDropDownContainer(szLinkTypeLabel, szLinkTypeFieldName, dtLinkType);
            //None
        Input.NextDropDownContainer();
            Input.ShowAcquireDocument(szInternalLinkLabel, szInternalLinkFieldName);
        Input.NextDropDownContainer();
            Input.ShowTextBox(szExternalLinkLabel, szExternalLinkFieldName);
        Input.EndDropDownContainer();

        if (!string.IsNullOrWhiteSpace(szTargetFieldName))
        {
            Dictionary<string, string> dtTarget = new Dictionary<string, string>();
            dtTarget.Add("No", "_self");
            dtTarget.Add("Yes", "_blank");
            Input.ShowDropDown("New Window", szTargetFieldName, dtTarget);
        }
    }
```

Outputhelper.MetaStringBuilder

```
       //This method enables the user to retrieve meta content from the content page asset as
well as MasterPage asset


        public static String MetaStringBuilder(Asset asset, Asset globalConfigAsset)
        {
            StringBuilder metaOutput = new StringBuilder(String.Empty);
            Asset config = Asset.Load("/" + asset.Parent.AssetPath + "/_Config");
            try
            {
                String myPageTitle = "";
                if (config.IsLoaded)
                {
                    myPageTitle = config["html_title"];
                }

                myPageTitle = String.IsNullOrWhiteSpace(myPageTitle) ? asset["html_title"] :
myPageTitle;
```

```
                myPageTitle = String.IsNullOrWhiteSpace(myPageTitle) ? asset["page_title"] :
myPageTitle;
                myPageTitle = String.IsNullOrWhiteSpace(myPageTitle) ? asset.Label :
myPageTitle;
                if (!string.IsNullOrWhiteSpace(myPageTitle))
                {
                    metaOutput.AppendFormat("<title>{0}</title>", myPageTitle);
                    metaOutput.AppendLine();
                }
            }
            catch { }

            string metaTag = "";

            try
            {
                String metaTitle = "";
                if (config.IsLoaded)
                {
                    metaTitle = config["meta_title"];
                }
                metaTitle = String.IsNullOrWhiteSpace(metaTitle) ? asset["meta_title"] :
metaTitle;
                metaTitle = (String.IsNullOrWhiteSpace(metaTitle)) ?
globalConfigAsset["global_meta_title"] : metaTitle;
                if (!string.IsNullOrWhiteSpace(metaTitle))
                {
                    metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "title",
metaTitle);
                    metaOutput.AppendLine();
                }
            }
            catch { }

            try
            {
                String metaDescription = (String.IsNullOrWhiteSpace(asset["meta_desc"])) ?
globalConfigAsset["global_meta_desc"] : asset["meta_desc"];
                if (!string.IsNullOrWhiteSpace(metaDescription))
                {
                    metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />",
"description", StripQuotes(StripChars(metaDescription)));
                    metaOutput.AppendLine();
                }
            }
            catch { }

            try
            {
                //more concise way of writing it for keywords...
                String metaKeywords = String.IsNullOrWhiteSpace(asset["meta_keywords"]) ?
globalConfigAsset["global_meta_keywords"] : asset["meta_keywords"];
                if (!string.IsNullOrWhiteSpace(metaKeywords))
                {
                    metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />",
"keywords", metaKeywords);
                    metaOutput.AppendLine();
                }
            }
            catch { }
```

```
            metaTag = String.IsNullOrWhiteSpace(asset["meta-page-topic"]) ?
globalConfigAsset["meta-page-topic"] : asset["meta-page-topic"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "page-topic",
metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-language"]) ?
globalConfigAsset["meta-language"] : asset["meta-language"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "Language",
metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-robots"]) ?
globalConfigAsset["meta-robots"] : asset["meta-robots"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "robots",
metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-URL"]) ? globalConfigAsset["meta-
URL"] : asset["meta-URL"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "URL",
metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-author"]) ?
globalConfigAsset["meta-author"] : asset["meta-author"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "author",
metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-geo.region"]) ?
globalConfigAsset["meta-geo.region"] : asset["meta-geo.region"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "geo.region",
metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-geo.position"]) ?
globalConfigAsset["meta-geo.position"] : asset["meta-geo.position"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />",
"geo.position", metaTag);
                metaOutput.AppendLine();
```

```
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-classification"]) ?
globalConfigAsset["meta-classification"] : asset["meta-classification"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />",
"classification", metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-distribution"]) ?
globalConfigAsset["meta-distribution"] : asset["meta-distribution"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />",
"distribution", metaTag);
                metaOutput.AppendLine();
            }

            metaTag = String.IsNullOrWhiteSpace(asset["meta-copyright"]) ?
globalConfigAsset["meta-copyright"] : asset["meta-copyright"];
            if (!string.IsNullOrWhiteSpace(metaTag))
            {
                metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "copyright",
metaTag);
                metaOutput.AppendLine();
            }


            //standard asset information
            metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "asset_id",
asset.Id.ToString());
            metaOutput.AppendLine();

            metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "asset_label",
asset.Label.ToString());
            metaOutput.AppendLine();

            metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "template_label",
asset.TemplateLabel.ToString());
            metaOutput.AppendLine();

            metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />", "site_id",
Asset.Load("/" + asset.AssetPath[0] + "/").Id.ToString());
            metaOutput.AppendLine();


            try
            {
                String sectionFolder = asset.AssetPath[1].ToLower();
                //if (String.IsNullOrWhiteSpace(sectionFolder)) { sectionFolder = "foo"; }
                if (!string.IsNullOrWhiteSpace(sectionFolder))
                {
                    metaOutput.AppendFormat("<meta name=\"{0}\" content=\"{1}\" />",
"sectionfolder", sectionFolder);
                    metaOutput.AppendLine();
                }
            }
            catch { }
```

```
            return metaOutput.ToString();
        }
```

TDW.MakeFilename

```
public static String MakeFilename(OutputContext context, Asset asset)
        {
            context.PublishPath = FormatFilename(context.PublishPath, context, asset);
            return context.PublishPath;
        }
```

TDW.FormatFilename

```
public static String FormatFilename(String path, OutputContext context, Asset asset)
        {
            String ext = context.RemotePath.Extension;
            //if (asset.TemplateLabel.StartsWith("xml", StringComparison.OrdinalIgnoreCase))
            //{
            //    path = path.Replace(ext, "xml");
            //}
            //string[] arrPath = new string[2];
            string szPath1 = "", szPath2 = "";
            try { szPath1 = asset.AssetPath[1]; }
            catch { }
            try { szPath2 = asset.AssetPath[2]; }
            catch { }

             if (asset.AssetPath.Count > 0 && (szPath1.Equals("_Master",
             StringComparison.OrdinalIgnoreCase)
                 || szPath2.Equals("_Master", StringComparison.OrdinalIgnoreCase) ||
                 asset.Parent.Label.Equals("_Master")))

            {
                path = path.Replace(ext, "master");
            }
            if (context.LayoutName.Equals("rss_output.aspx") ||
                context.LayoutName.Equals("rss_output.asp") ||
                context.LayoutName.Equals("xml_output.aspx") ||
                context.LayoutName.Equals("xml_output.asp") ||
                context.LayoutName.Equals("output_rss.aspx") ||
                context.LayoutName.Equals("output_xml.aspx"))
            {
                path = path.Replace(ext, "xml");
            }

            if (context.LayoutName.Equals("output_exacttarget.aspx"))
            {
               path = path.Replace(context.RemotePath.Label, context.RemotePath.Label + "_et");
                //path = path.Replace(ext, "aspx");
            }

            if (asset.Label.Equals("Web.Config", StringComparison.OrdinalIgnoreCase))
            {
                path = path.Replace(ext, "config");
                path = path.Replace(context.RemotePath.Label, "web");
            }

            if (asset.Label.Equals("Homepage", StringComparison.OrdinalIgnoreCase) ||
```

```
                asset.Label.Equals("Index", StringComparison.OrdinalIgnoreCase))
                path = path.Replace(context.RemotePath.Label,
(context.LayoutName.Equals("output_mobile.aspx")) ? "index_mobile" : "index");

            if (asset.AssetPath.Count > 0 && asset.AssetPath[0].Equals("_Analytics and SEO",
StringComparison.OrdinalIgnoreCase))
            {
                path = path.Replace("/_Analytics and SEO", "");
            }

            if (asset.AssetPath.ToString().IndexOf("/Assets/") != -1)
            {
                path = path.Replace("/Assets", "");
            }

            if (string.Equals(asset.AssetPath[1], "RSS Feeds"))
                path = path.Replace(ext, "xml");


            if (context.LayoutName.Equals("output_mobile.aspx"))
                path = path.Replace(context.RemotePath.Label, context.RemotePath.Label +
"_mobile");

            path = path.ToLower().Replace("#47;", "");

            return path;
        }
```