**Hosting Requirements**

* .NET Framework 4.5 +
* MS SQL Server 2012 +
* IIS 8 +

**Software Required**

* Visual studio 2015 Community Edition +
* MS SQL Server 2012 + ( Works perfectly fine with the Free Express Editions )

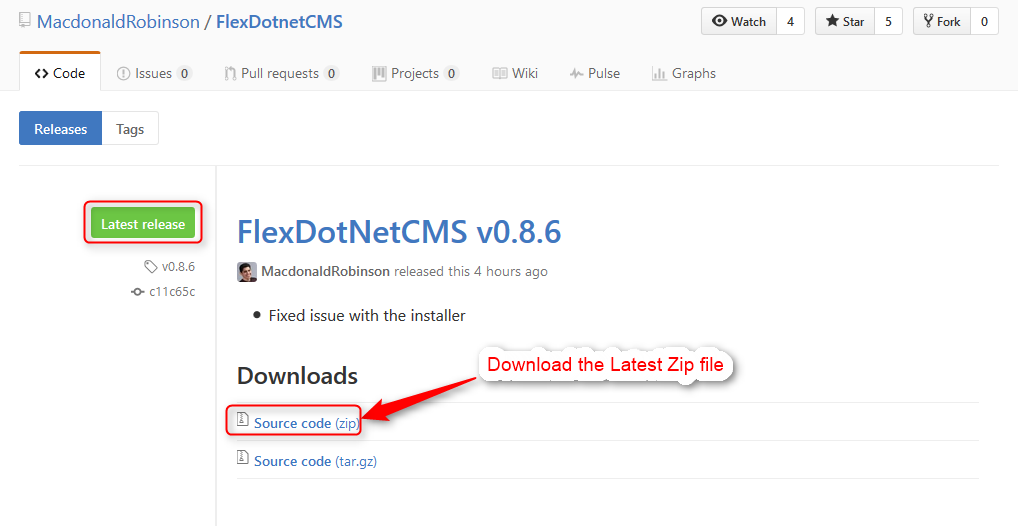
**Introductions**

This project is open source and hosted on Github, you can view a full list of features by going to: <https://github.com/MacdonaldRobinson/FlexDotnetCMS>

**Download the CMS**

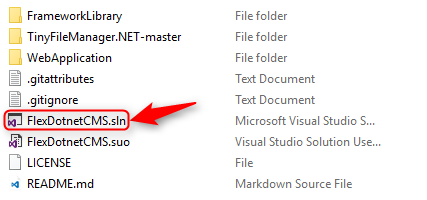
* Browse to: <https://github.com/MacdonaldRobinson/FlexDotnetCMS>
* Click on the “Releases tab”
* Download the Latest Release

****

****

**Open the FlexDotnetCMS Solution in Visual studio**

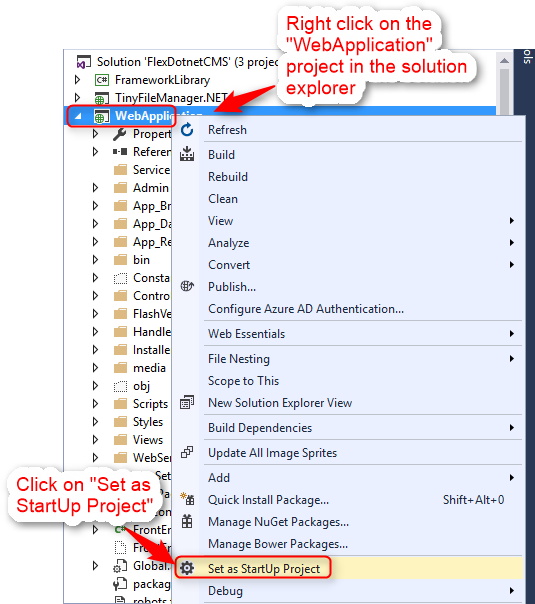
* Unzip the file
* Double click on the “FlexDotnetCMS.sln” file
* This will load the project in visual studio



**Set the “WebApplication ” project as the Startup Project**

Once the project finishes loading in visual studio, you will see 3 projects in the Solution Explorer. The “WebApplication” Project contains all the CMS files and needs to be set as the start up project so it would run this project by default.

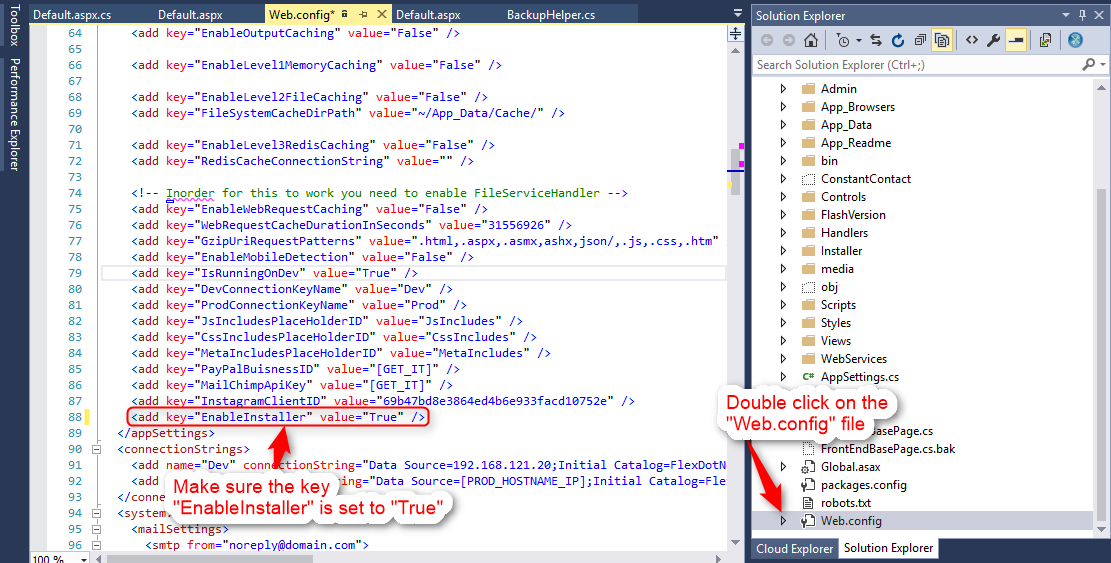
* Right click on the “WebApplication” project
* Select “Set as StartUp Project” from the menu that pops up



**Enable the Installer**

For security reasons the installer needs to be enabled in the “Web.config” file

* Search for the “Web.config” file under the “WebApplication” Project
* Double click on the “Web.config” file to open it
* Search for the key “EnableInstaller”
* Make sure the value for this key is set to “True”



**Create the database**

* Create a new MSSQL Database
* If you don’t already have a user, create one and give the user full permission to this database.

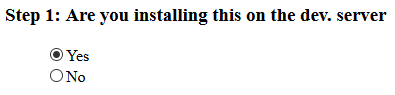
**Run the Project from Visual Studio**

You can now run the project by clicking on the Play button  in the visual studio toolbar. If the “EnableInstaller” key is set to “True” you will be taken to the Installer screen.

The installer provides step by step instructions on how to setup the database. It also updates the “Web.config” file based on the information you provide in the installer steps.

Below is how the Steps map to the web.config file.

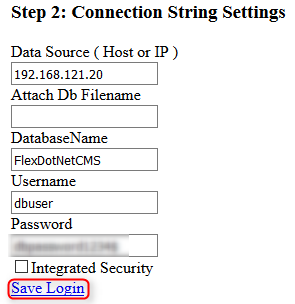




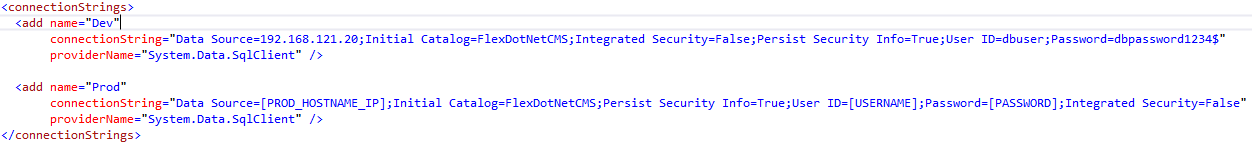
This step maps to the “IsRunningOnDev” AppSetting in the Web.config file.



If you select Yes, the value in the web.config will change to “True”. If you select “No” it will change to “False”



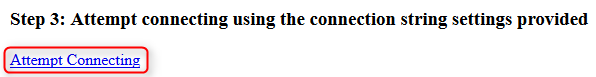
Clicking on the “Save Login” link will update the connection string in the Web.Config File.



Which connection string will be updated is based on the values set for the “IsRunningOnDev” AppSetting, which is set based on the previous step.

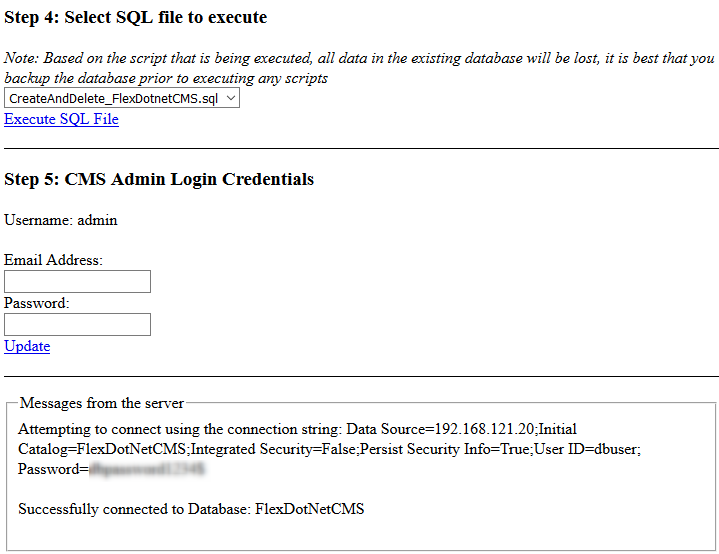
If “IsRunningOnDev” is set to “True”, it will update the “Dev” connection string.

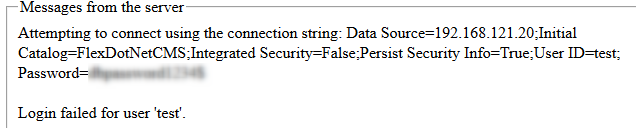
If “IsRunningOnDev” is set to “False”. It will update the “Prod” connection string.

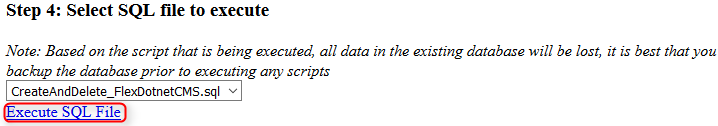


Clicking on the “Attempt Connecting” link will attempt to connect to the database using the connection string set in the previous step.

If the connection to the database is successful, you will see the next steps.



If the connection is not successful you will see the reason why in the “Messages from the server” area.   


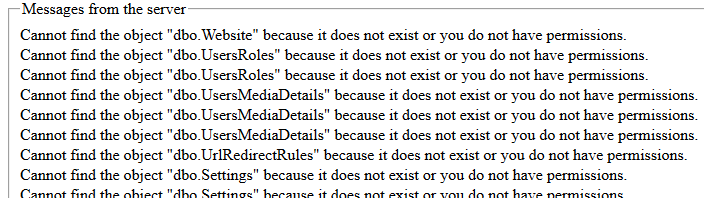


The dropdown list is populated with a list of all the sql files located in this folder: “/WebApplication/Installer/SqlFiles/”.

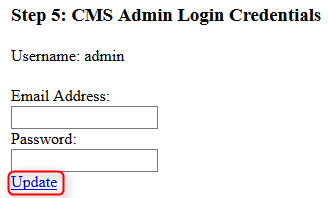
By default the file “CreateAndDrop\_FlexDotnetCMS.sql” is the sql for the CMS

Clicking on the “Execute SQL File” will execute the script from the selected file against the database.

If the database is empty then you will get the following errors in the “Messages from the server” file



This is expected, since the script also includes “Drop” commands.



Clicking on the “Update” link will update the default “admin” user’s Email Address and Password.

If the information was successfully updated you will see the following success message in the “Messages from the server” area, and you will be taken to the next step





Clicking on the “Disable Installer” link will update the “EnableInstaller” AppSetting in the web.config file and set its value to “False”.



This will disable the installer and take you to the CMS login.

You can also manually go to the login page by going to “[DOMAIN]/admin/”

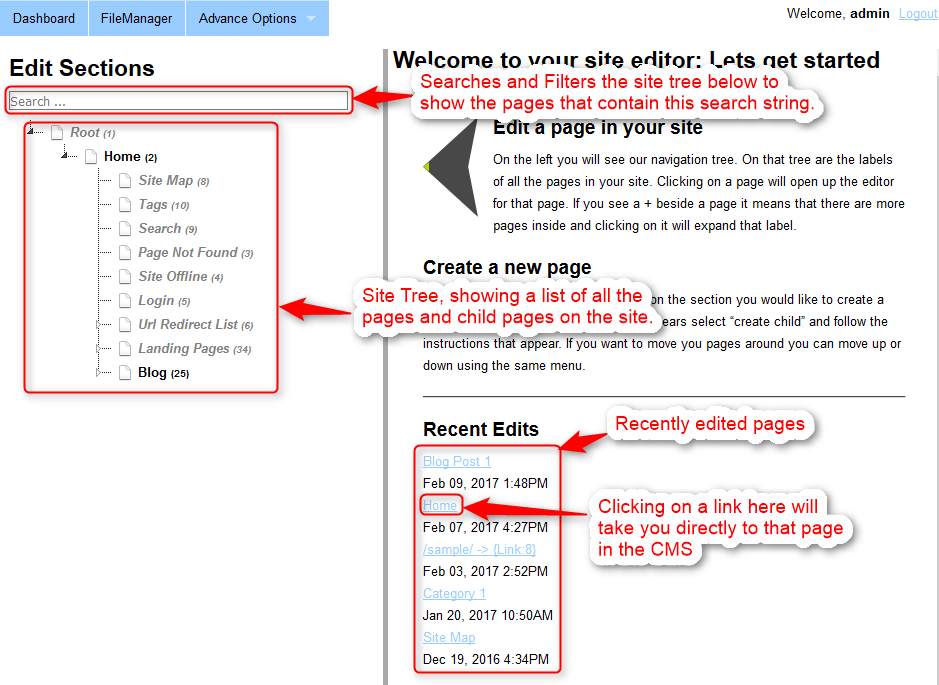


Enter the username ‘admin’ and password that you set in Step #5 and click the “Login” Link. You will now be logged into the CMS.

**CMS Overview**

The dashboard is the first screen you will see once you login to the CMS

The dashboard contains a list of recently edited pages. Clicking on the link will take you directly to that page in the CMS

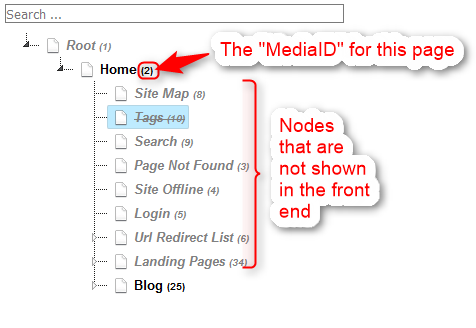


**Site Tree**

The Site Tree show’s you a list of all the pages on the website, including all the other nodes on the site that are not visible in the front end.

The “Home” node in the site tree is the home page of the website. All other pages are under this node.

The Search field will filter the site tree showing nodes that contain the search text.



Every node has a number beside it. This is the “MediaID” of the page. This will be important later.

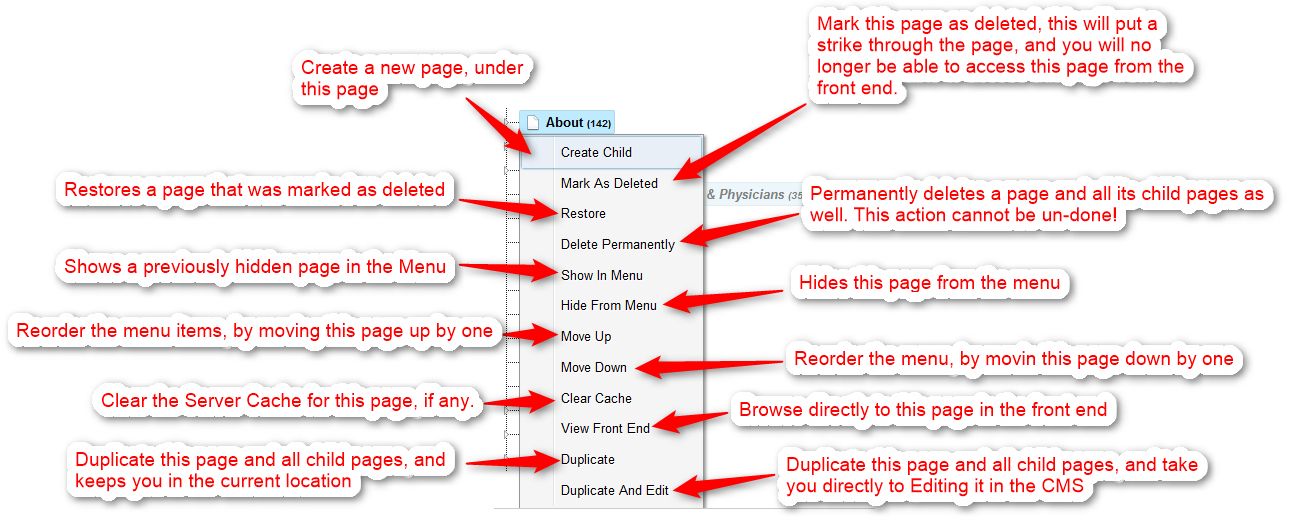
Nodes that are Grayed out are nodes that are not shown on the site.

Nodes that have a strike thru them are nodes that are marked as deleted.

Deleted nodes cannot be accessed from the front end of the site. However, nodes that are simply hidden can be accessed from the front end, provided you know the URL to that node.

**Site Tree Right Click Context Menu**

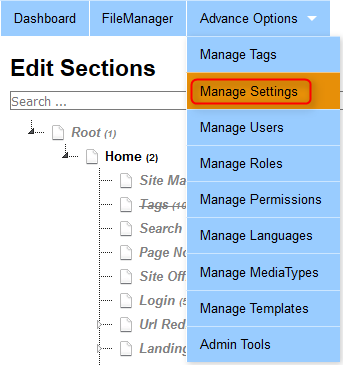
Right clicking on any node in the site tree will bring up a context menu, which provides a list of actions that can be taken on this node.



* **Create Child:** This will allow you to create a new page, under this node.
* **Mark As Deleted:** This will mark this page with a strike through it. This page will no longer be accessible from the front end.
* **Restore:** This will restore a page that has been marked as deleted, so you will once again be able to access this page from the front end.
* **Hide From Menu:** This will gray out this page, You can still access this page from the front end, however it will no longer be shown in the menu.
* **Show In Menu:** This will make a page that was hidden from the menu previously, show in the menu once again.
* **Duplicate:** This will duplicate a page and all its child pages.
* **Duplicate and Edit:** This will duplicate a page and all its child pages, once its done duplicating it will take you to that page in the CMS so you can edit it right away.
* **View Front End:** This will open a new tab and browse you to the page.
* **Clear Cache:** This will clear any server cache for the current page.

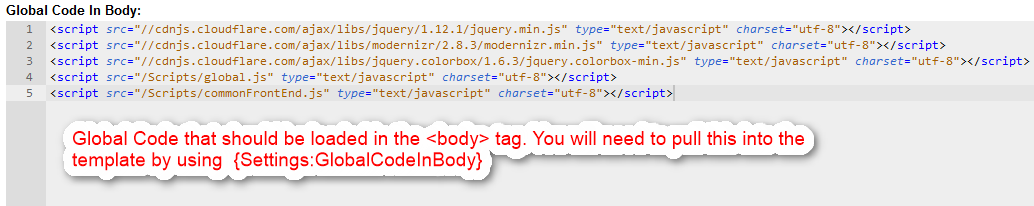
**Settings Section**

The settings section contains global settings that will be used by the CMS. You can access the settings section by going to “Advance Options -> ManageSettings”



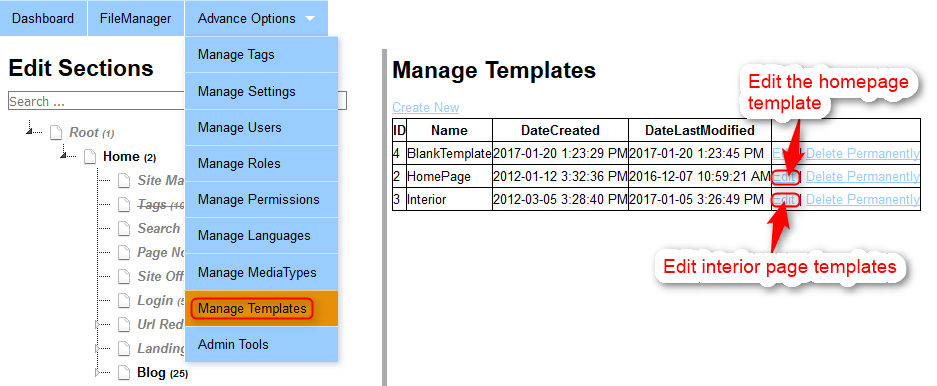






**Create / Edit a Template**

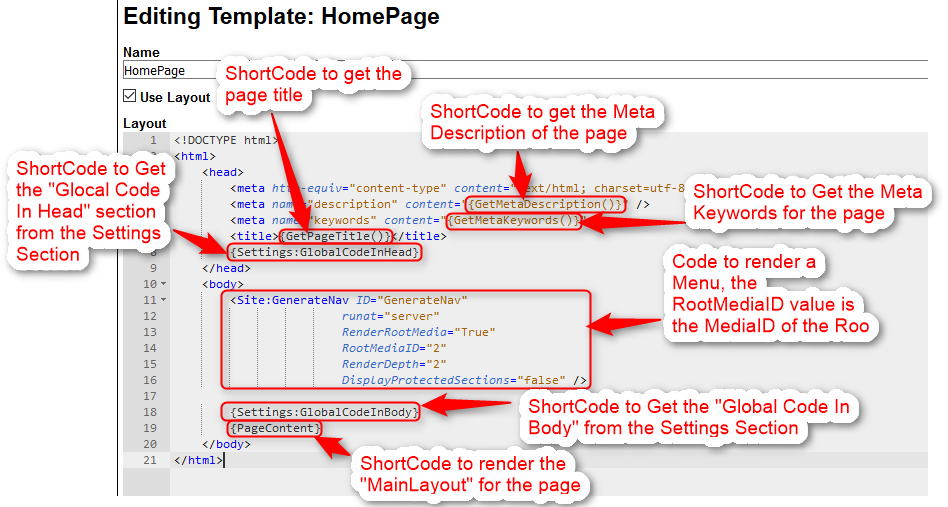
The first thing you would want to do is to create / edit a template, you can do this by going to “Advance Options-> Manage Templates”

****

I have created 3 different templates by default.

* **BlankTemplate**: This doesn’t have any layout to it.
* **HomePage**: This is used by the “Home” node
* **Interior**: This is used by all other pages.

Clicking on the “Edit” beside a template will popup a modal window allowing you to edit that template.



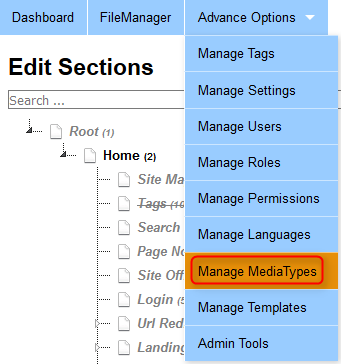
* **{GetMetaDescription()}:** Short Code that will load the Meta Description for the current page.
* **{GetMetaKeywords()}:** Short Code that will load the Meta Keywords for the current page.
* **{Settings:GlobalCodeInHead}:** Load the “Global Code In Head” field from the Settings Section.
* **{Settings:GlobalCodeInBody}:** Load the “Global Code In Body” field from the Settings Section.
* **{PageContent}:** Loads the “MainLayout” of the current page.
* **<Site:GenerateNav ID="GenerateNav" runat="server" RenderRootMedia="True" RootMediaID="2" RenderDepth="2" DisplayProtectedSections="false" />**
  + This code renders a menu, the “RootMediaID” property determines what the root page is going to be. You can find this from the SiteTree  
    

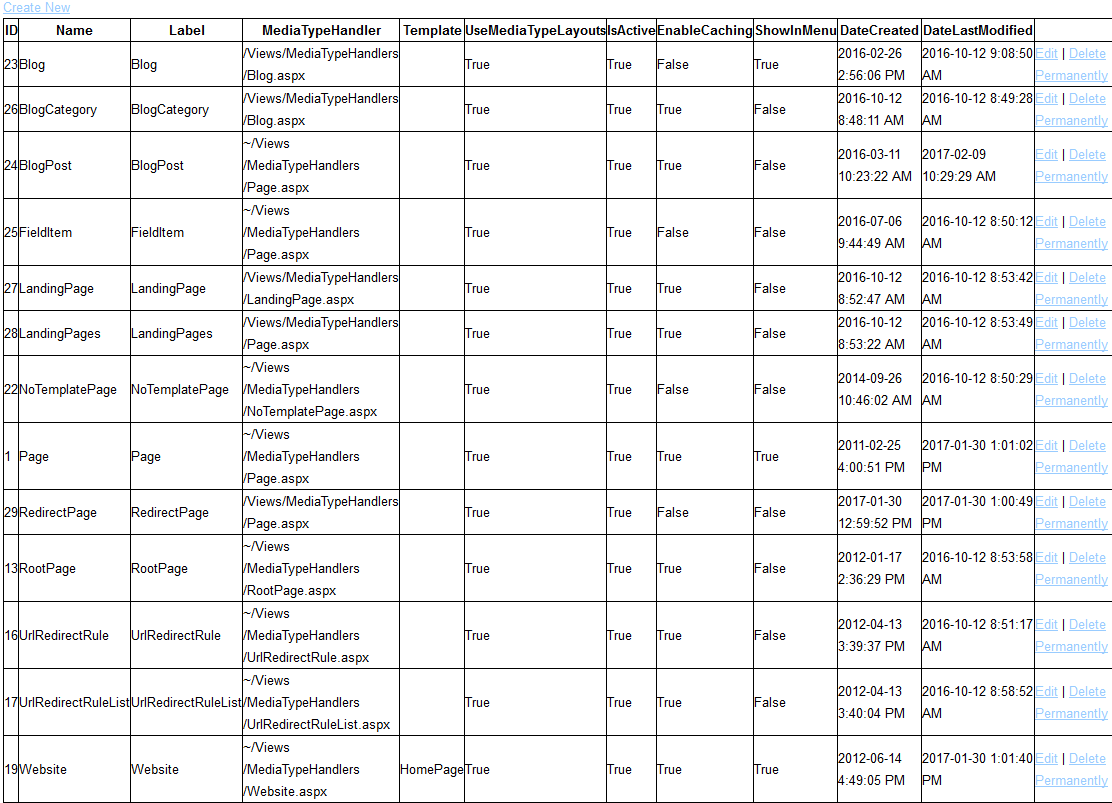
**Creating / Editing a Media Type**

Every node in the “Site Tree” has a “Media Type”, which is essentially the type of the page. Which means you can have multiple nodes that have the same “Media Type” and you can control the layout of all the nodes that use the same “Media Type” from 1 location i.e “MediaType” Layout.

For example, you can create a media type called “BlogPost” and have a layout associated with it, every page that uses the “BlogPost” Media Type automatically inherits that layout.

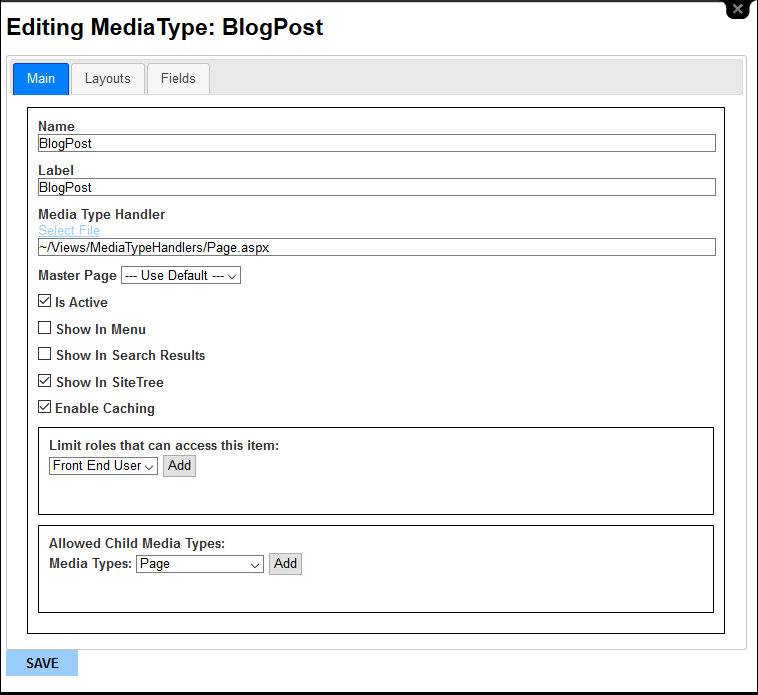
You can view a list of all the Media Types by going to “Advance Options -> Manage MediaTypes”





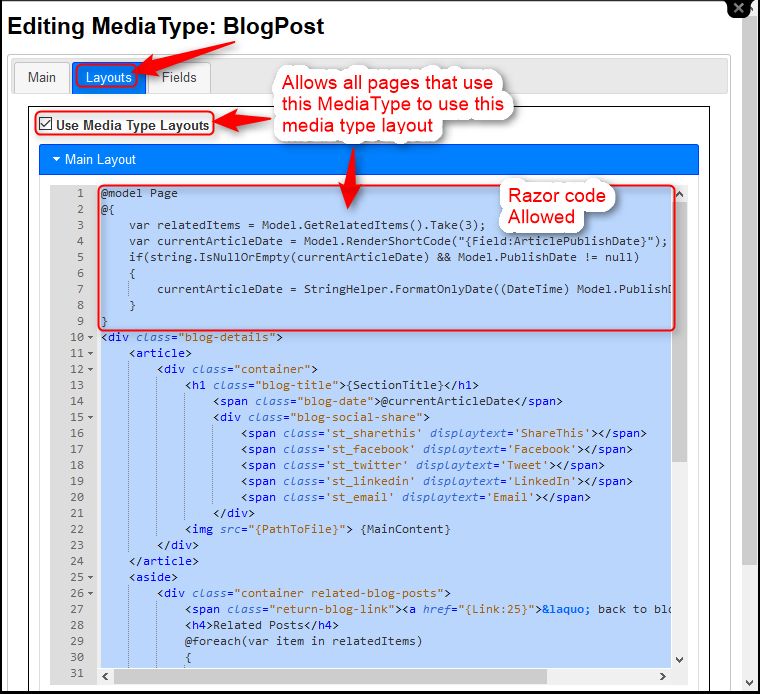
You can click on the “Create New” Link to create a new media type. Or click on the “Edit” Link to edit an existing media type.

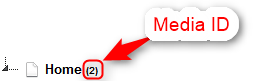
Below is how a MediaType Editor looks:

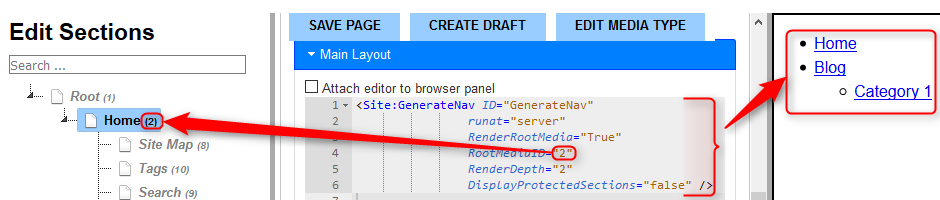


* **The MediaType Handler Field**: In general should be set to “~/Views/MediaTypeHandlers/Page.aspx”
* **Master Page field:** Is the “Template” for this “Media Type”, all pages that use this media type will automatically use the template specified here
* **Is Active:** Means that this MediaType is Active and will show up in the selector when creating a new page.
* **Show In Menu:** This page can show up in the Menu of the website.
* **Show In Site Tree:** Any pages created using this media type will show in the “SiteTree”
* **Enable Caching:** Pages that use this media type can be cached.
* **Allowed Child Media types:** Assign what type of children can be created under this page.

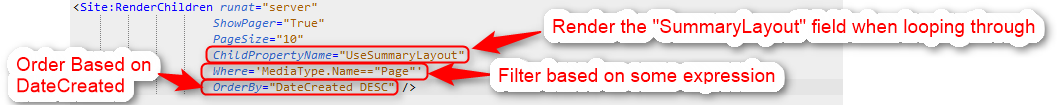
**MediaType Layouts Tab**



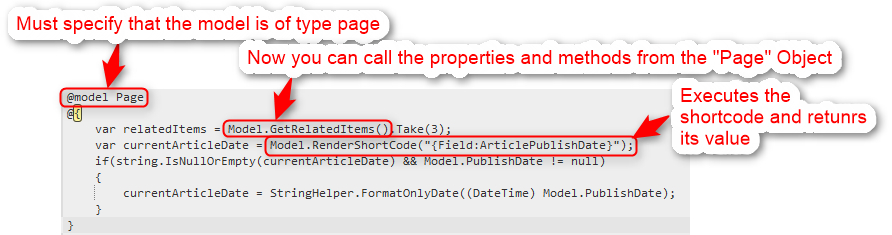
* **Use MediaType Layouts Field:** If Checked All pages that use this media type will by default get its layout from the media type layout field
* **Main Layout field:** The Main Layout section contains the HTML for the current media type. You can also add razor code in here which makes it extremely powerful. You can also add short codes in here which allow you to pull the different fields and perform other actions.
  + **{SectionTitle}:** This will pull the value of the “Section Title” Field and inject it into the location.
  + **{MainContent}:** This will pull the contents of the “Main Content” Field and inject it in this location
  + **{PathToFile}:** This will pull the value in the “PathToFile” field and inject it here
  + **ASP.NET Webform tags:** You can include ASP .NET Webform tags as well, this includes pulling in Custom User Controls and built in controls.  
    ****
  + **{Link:[MediaID]}:** This will create a link to the page. Replace “[MediaID]” with the MediaID of the page  
    ****
  + **{Field:[FieldCode]}:** You can load a custom field that you have create in the “Fields” tab. Replace “[FieldCode]” with the actual code of the field.
  + **<Site:GenerateNav ID="GenerateNav" runat="server" RenderRootMedia="True"RootMediaID="2" RenderDepth="2" DisplayProtectedSections="false" />**
    - Provides an easy way to render a Nav. This respects all the properties for the page such as Marked as Deleted, Show in Menu, Hide In Menu, Published or not etc

****

* + **<Site:RenderChildren runat="server" ShowPager="True" PageSize="10" ChildPropertyName="UseSummaryLayout" Where='MediaType.Name=="Page"' OrderBy="DateCreated DESC" />**
    - Provides an easy way to loop through all the children and render the “UseSummaryLayout” property of those children, also creates a pager if needed.

****

* + **Razor Code:** This allows you to program directly in the Layouts section its self. You can use @if, @foreach etc to write C# code which will be compiled on the fly using the Razor Engine Template Parser.

****

You can render a variable, by using the syntax “@VariableName”

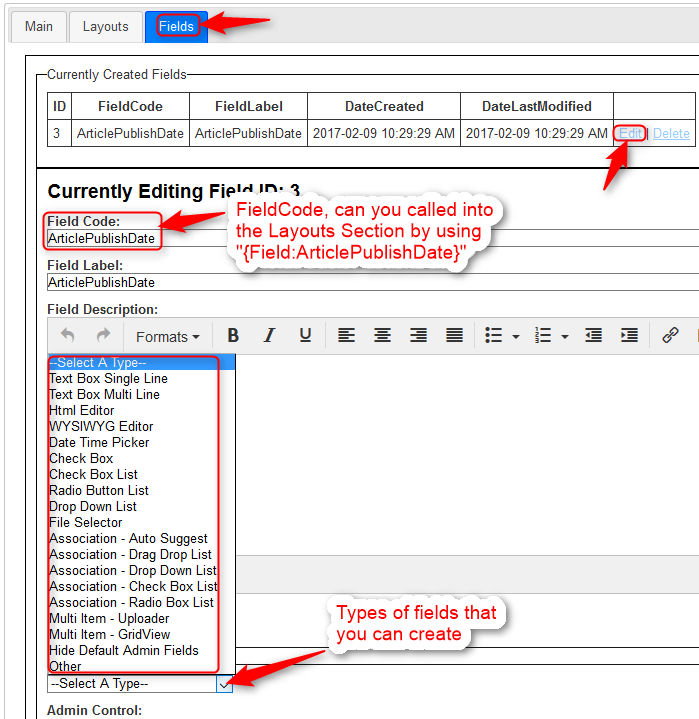
****

**MediaType Fields Tab**

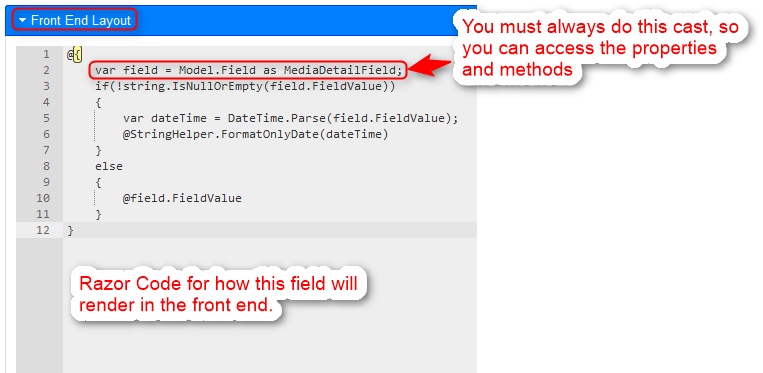
The Fields tab allows you to create custom fields that will show up in the “Main” Tab when editing a specific page with this type.

All pages that use this media type will automatically get these fields

Custom fields can be rendered in the front end by using: “{Field:[FieldCode]}” in the Main Layouts Section where “[FieldCode]” needs to be replaced by the actual field code.

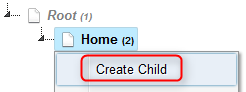


Every field has a “Front End Layout” section. This is what tells how to render this field in the front end when calling {Field:[FieldCode]}

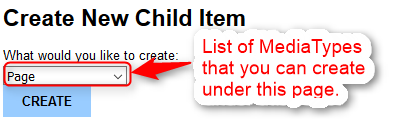
****

**Create a Page with a specific Media Type**

Right click on the “Home” Node in the “Site Tree” and select “Create Child”



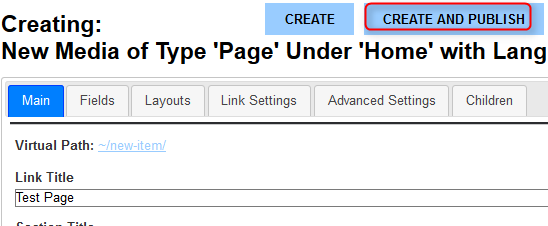
This will take you to a page where you will need to select the “Media Type” for this page.



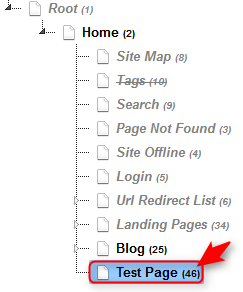
The dropdown list will show you a list of the MediaTypes that you can create under this page. This is based on the “Allowed Media Types” field in the MediaType Editor for the “Home” Page.

Now click on the “Create” button

You will be taken to the page editor, enter the name of the page in the “Link Title” section and click on the “Create And Publish” button

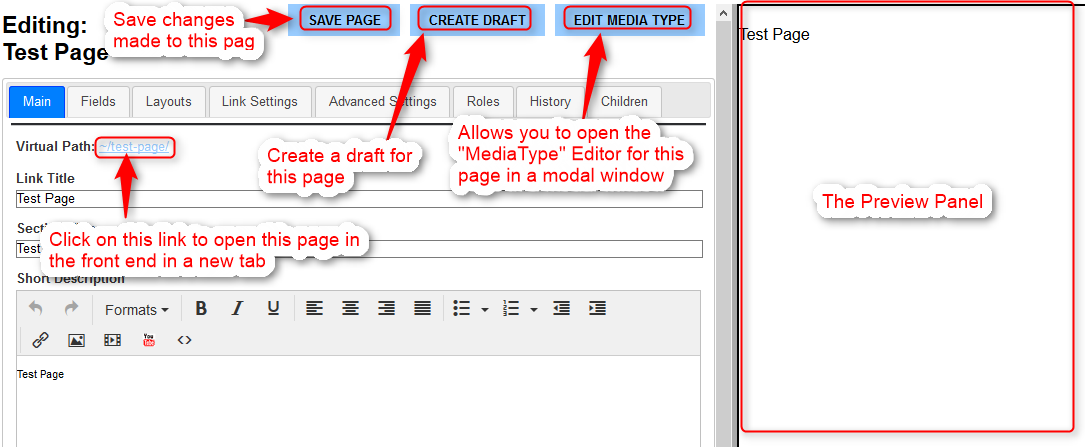


The page is now created and will show up in the site tree.

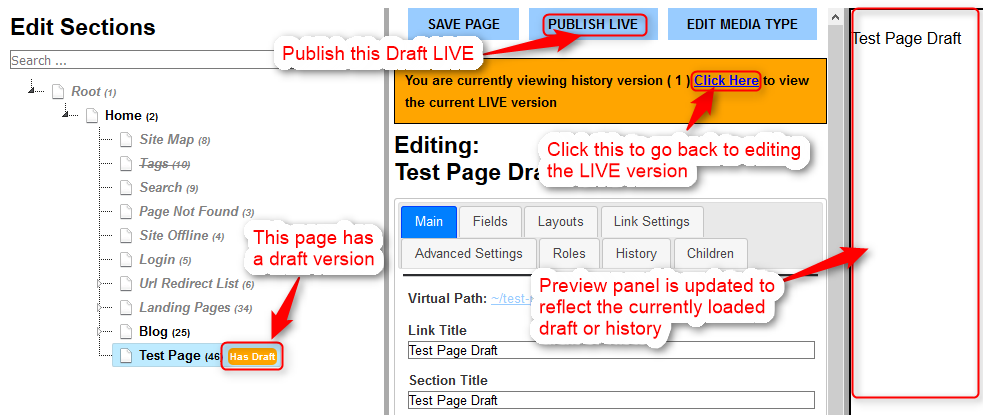


**Editing a Page**

Simply click on a node from the site tree that you would like to edit. This will load the Editor for that page.

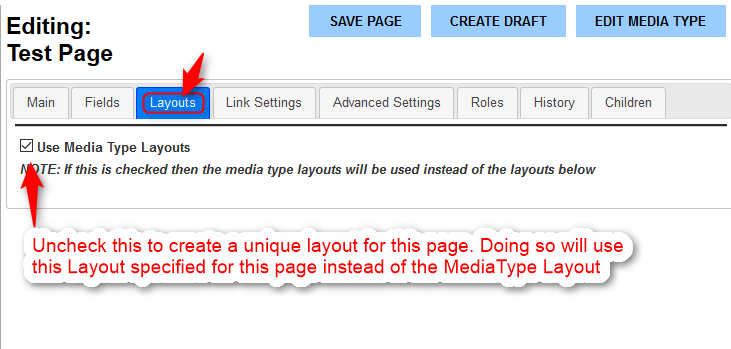


* **Save Page:** Clicking on this button will save any changes you have made to the fields
* **Create Draft:** Clicking on this button, will immediately create a snapshot of the current live version of the page and put you in an edit mode for the Draft

****

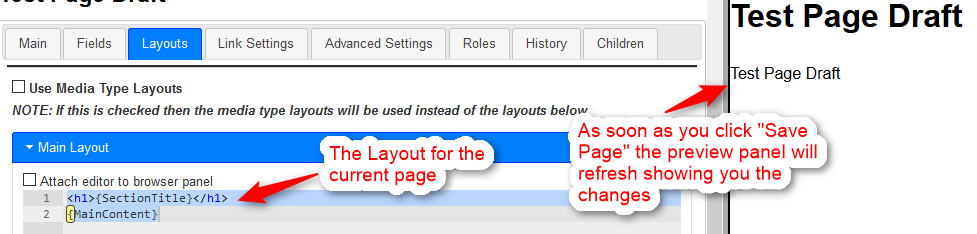
* **Edit Media Type:** Clicking on this button will open the MediaType Editor for the current page.

Click on the “Layouts” Tab. If the checkbox labeled “Use Media Type Layouts” is check then the layout for the page will come from the “Media Type” Layouts section.



If you uncheck the “Use Media Type Layout” field, it will show you the layout only specific to this 1 page.

It is recommended to first create all the media types for the site that you can identify and then create pages using the media types, so you can control the layout through the “MediaType” Layout section. So every page with a specific type can all use 1 MediaType layout.

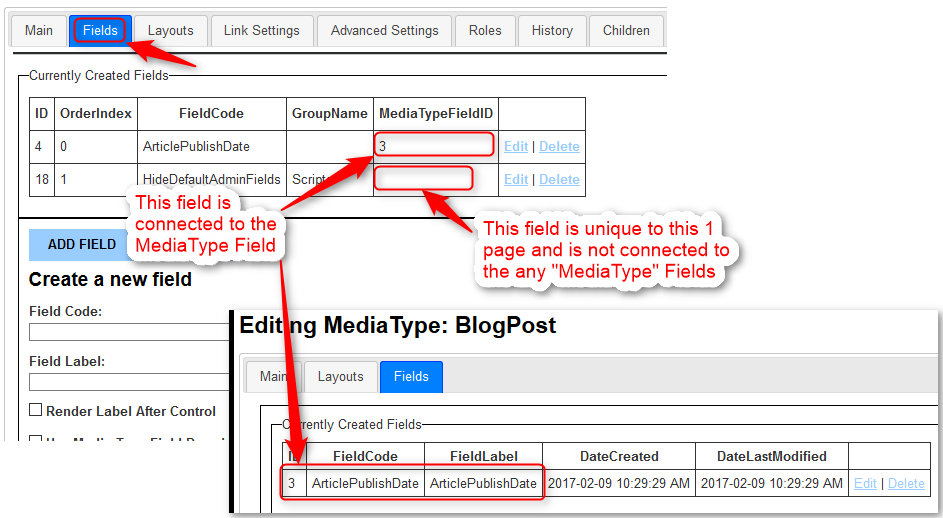


**Custom Fields for a specific page.**

Click on the “Fields” Tab. All custom fields for this page will show up in the “Currently Created Fields” section.

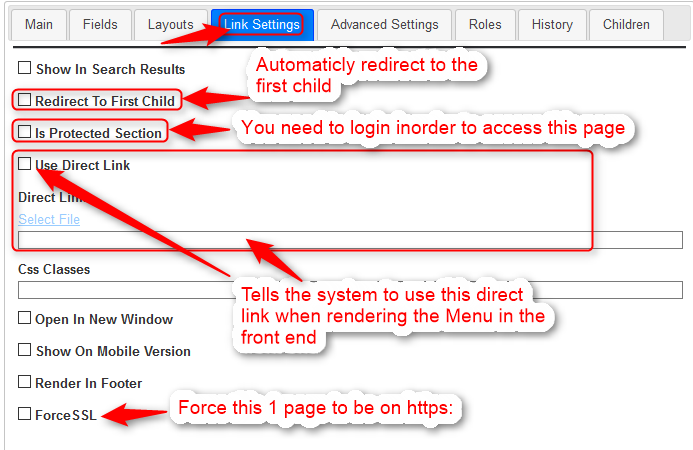
Fields that have a value in the “MediaTypeFieldID” column are associated with the Media Type Field with the same name in the MediaType Fields section. And must be Edited there.

Fields that don’t have a value in the “MediaTypeFieldID” column are not associated with the MediaType and are specific to this 1 page.



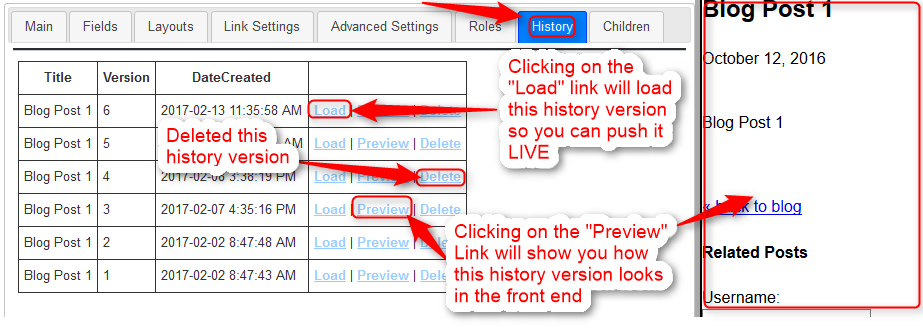
**Links Settings Tab**

Click on the “Link Settings” Tab.



**History Tab**

Click on the “History” Tab.



**Advance Settings Tab**

Click on the “Advance Settings” Tab.



**Using Handler Files**

If the functionality for a page is too complicated to simply program in the Layouts section, or in the MediaType layout section, you can link the page or the media type to a specific “aspx” file.

This allows you to use all the power of visual studio to layout the page as you need.

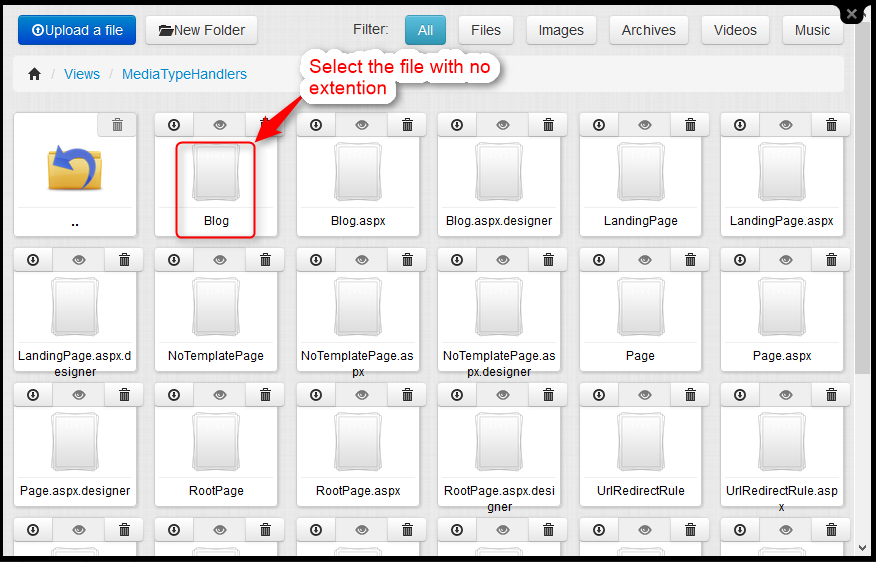
The “Blog” MediaType and the “BlogCategory” MediaType both use a handler file.

**Selecting a Handler File for the MediaType**

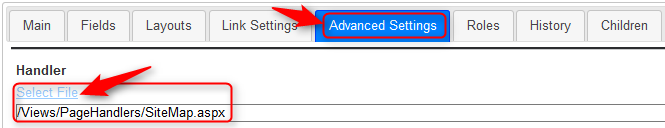


Clicking on the “Select File” in the MediaType will open the file manager, showing you all the files under the following folder “/Views/MediaTypeHandlers/”

Make sure to select the file without an extension in the file manager, this is just because the file manager does not show extensions for files.

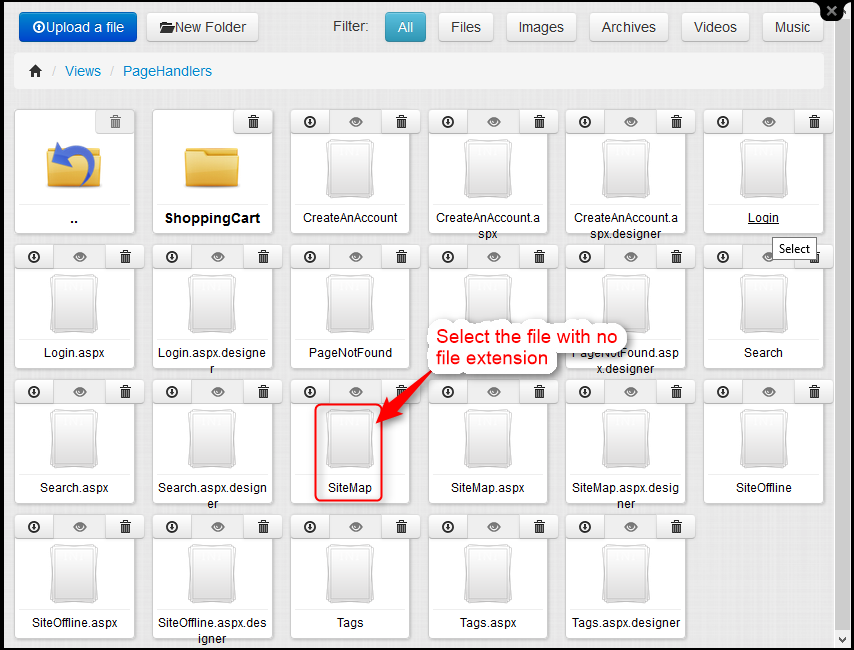


**Selecting a Handler File for an individual page**

****

Clicking on “Select File” will open the file manager showing you the files at the following location: “/Views/PageHandlers/”

Make sure to select the file without an extension in the file manager, this is just because the file manager does not show extensions for files.



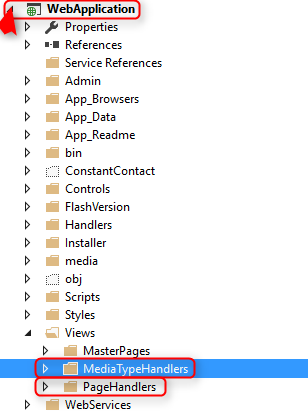
**Creating a handler file**

Every handler file is a webform page ( aspx ) which inherits from “FrontEndBasePage”. There are several handler files that are included, which you can use as a reference.

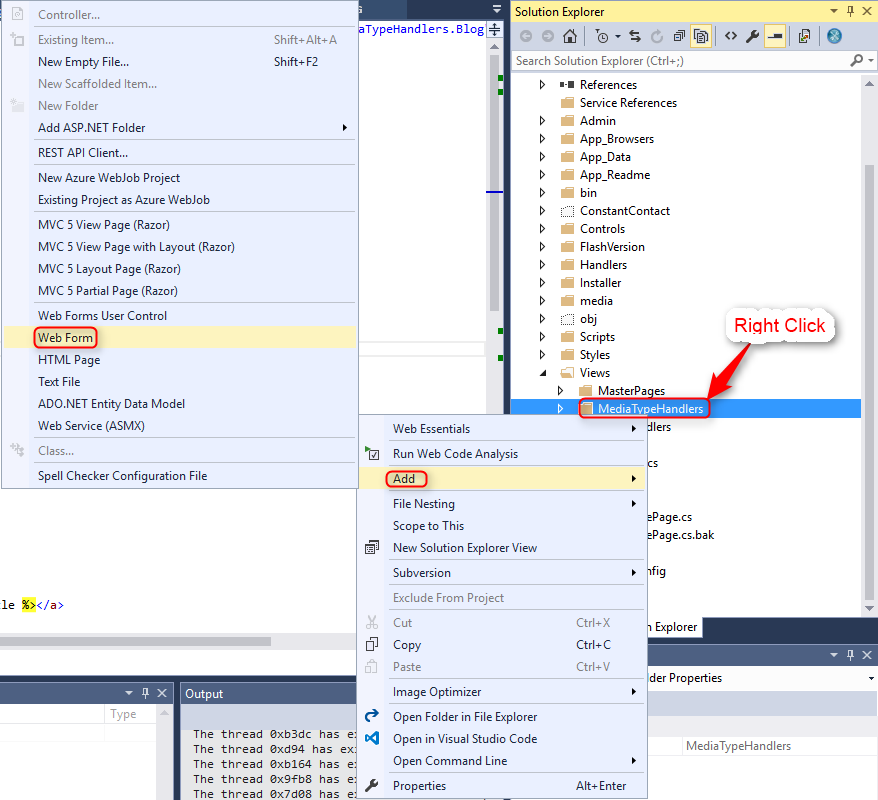
I have divided the handler files into 2 folders: “PageHandlers” and “MediaTypeHandlers”. A Page handler is the layout for 1 single page, where as a media type handler file is a layout for a type of page ( “MediaType” )

You can find these handler files in the following locations

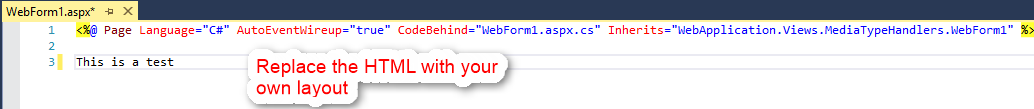
* PageHandlerFiles: “/Views/PageHandlers/”
* MediaTypeHandlers: “/Views/MediaTypeHandlers/”

****

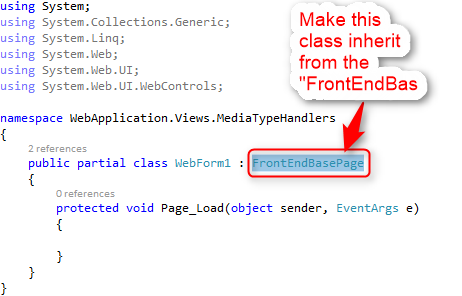
To create a handler file simply right click on the folder that you want to create the handler file under and select “Add”-> “Web Form”



Once the WebForm file has been created, you can simply remove all the HTML code that is placed in the “.aspx” file with what you want.



Open the Code Behind file“.aspx.cs” and change the Base Class from “System.Web.UI.Page” to “FrontEndBasePage”



For an advance reference of a handler file please open the “Blog.aspx” and the “Blog.aspx.cs” pages and view how the Blog Handler file is created.

**“Blog.aspx” file**



**“Blog.aspx.cs” file**



**Editing SEO Settings**

At the bottom of every page that you edit in the CMS you will find the “SEO Settings” tab



Clicking on this tab will expand the SEO sections, show the following fields:

* **Page Title**: This is what is shown in the browser title bar and is what Google shows as the name of the page in the search results
  + **Note**: The CMS automatically concatenates this title with the parent page titles
* **Meta Description:** This is what Google shows for the description of the page in the search results
* **Meta Keywords:** These are keywords that are relevant to this page