

The Web Pro Miami, Inc.
2963 NW 99th Pl.
Doral, FL 33172-1092

T: 786.273.7774
info@thewebpro.com
www.thewebpro.com



Web Pro Manager: Customization Guide

v.1.0

Web Pro Manager is an open-source website management platform that is easy to use, intuitive, and highly customizable.

Web Pro Manager can be used to easily create and maintain complete Web sites.

Table of Contents

1. [Introduction to Customizing WPM](#)
2. [Templating the Front-End](#)
3. [Customizing Your Components](#)
4. [Customizing the Back-End Manager](#)

Introduction to Customizing WPM

What do you mean by customizing Web Pro Manager?

When you install Web Pro Manager on your website, by default the design of the front-end (public area) of the site is generic and decently attractive (**Figure 1.1**).

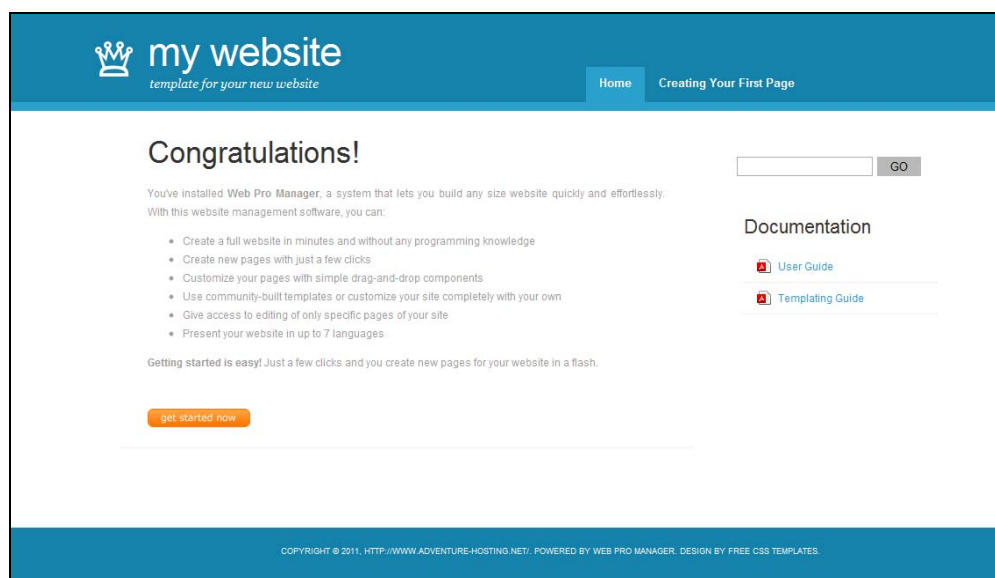


Figure 1.1 The default look of the front-end of your new Web Pro Manager-based website. Not bad, huh?

Although it looks nice, it's probably not what you had in mind when you envisioned your website. Or maybe you already have a pre-existing website or a design template. Either way, in order to make the site the way you want it, you'll need to customize your Web Pro Manager installation.

Customizing is easy!

Don't get scared away from building your website with Web Pro Manager because customizing is involved. The truth is, customizing WPM is very simple and the process is quicker than you think.

Here are the 3 simple steps to customizing WPM:

- 1) **Template the front-end**
- 2) **Customize your components**
- 3) **Brand the back-end manager**

Templating the Front-End

What's a template?

As far as websites go, a template is a base file that holds the design and layout of your web page, but offers little or no content itself. Think of a template like a blank sheet of company letterhead... it displays the logo, maybe the company address, and sets up the page to be used for new content without going through the hassle of adding the logo and address every time. A website template works the same way; the template sets up the page for content to be added to (**Figure 2.1**).

A website template is coded in HTML and is usually created from a design made in Photoshop or some other image editing program. A website template contains HTML, images, JavaScript and CSS styles, and it most likely is built-up from several of these types of files.

From this point on, I'll assume that you have your own template coded in HTML, and that all related files (images, stylesheets, etc...) are packaged with your template. If you do not have a template file yet, do not proceed until you create one.

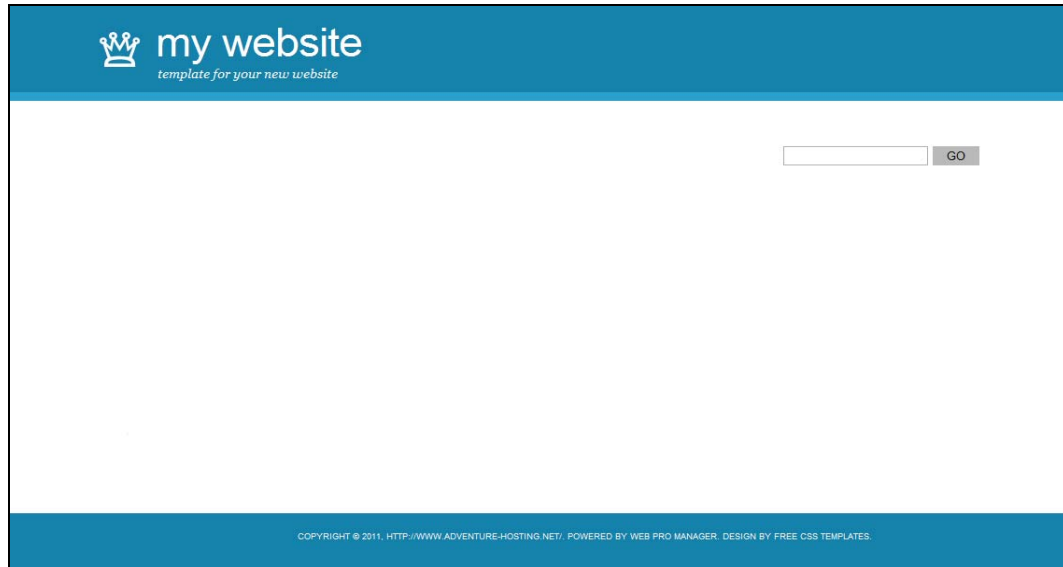


Figure 2.1 The template file for Web Pro Manager – basically the same web page, but without the content and navigation.

You'll need to install your template to get the site designed the way you'd like it to be. **Web Pro Manager gives you access to templates for 3 page types:**

- **Home**
- **Content**
- **Search**

Why three? Because it's pretty likely that certain pages, like the homepage of your site, look dramatically different than others. With separate template files, you can easily setup a unique design for your homepage versus that of a content page without any extra hassle.

Installing your template

1. First, make sure that your template files are all in one main folder. That folder can contain subfolders with images and stylesheets, but no file that makes up the template should be out of that one main folder.
2. Change the name of that main folder to “tmpl”.
3. If you have one HTML template file for your website, change the name of the file to *home.html*, and make 2 duplicates: *content.html* and *search.html*. If you actually have a separate HTML template file for the “inner” pages of your website, change the name of that to *content.html* and make a duplicate of it: *search.html*. In the end, your new “tmpl” folder should contain 3 template files and all of the other files that make up your templates (**Figure 2.2**).

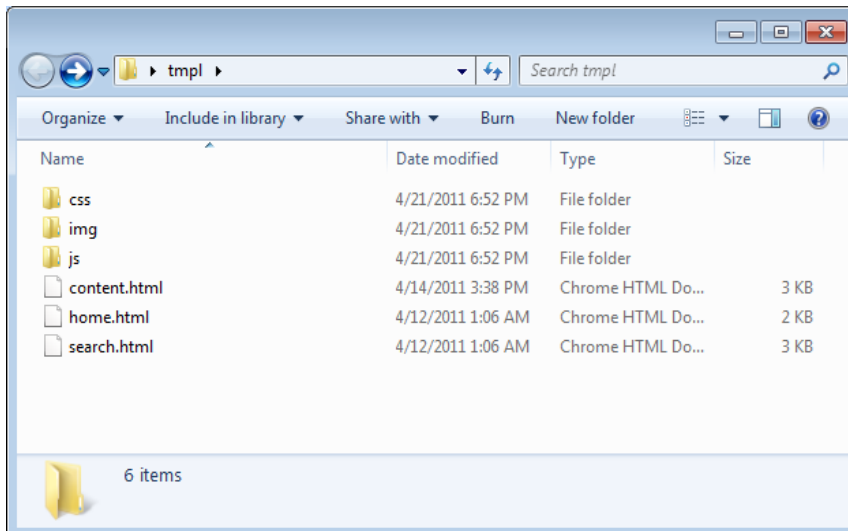


Figure 2.2 A sample file structure for your template folder.

Finalizing your template

Before you finish, you need to tell Web Pro Manager what parts of your template will have content that's editable by the user. In the case of WPM, this content will be created by one or more components that are dragged and dropped onto the page. We will call these parts of your template **Component Columns** and they will be defined in your template with a tag.

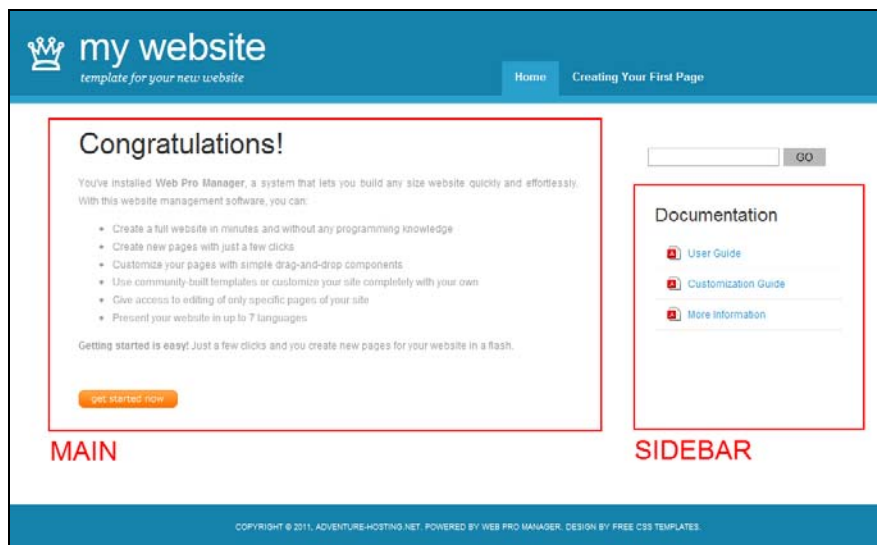


Figure 2.3 The default template has only two editable areas, or Component Columns: the main content section and the sidebar.

Here's how you update your templates to define your **Component Columns**:

1. Open your first template, *home.html*.
2. Find the place where your first **Component Column** begins. For instance, find an area of the template that contains changeable navigation, content or a sidebar.
3. Add the following tag in that space: **[#compColumnComps_COLID=1#]**
This tag should also replace all "dummy" content you have there, if needed. Here's a sample snippet from the default template to show you what it should look like after your edit:

```
<div id="content">  
  [#compColumnComps_COLID=1#]  
  <div style="clear: both;">&nbsp;  </div>  
</div>
```

4. Repeat this step for each **Component Column** in your template, but raising the number of the tag by one each time. For instance, your second **Component Column** tag added should be:
`[#compColumnComps_COLID=2#]`
5. Repeat steps 1-4 for each of your other templates.
6. Once all tags have been added, replace the “tmpl” folder in the root directory of your Web Pro Manager installation with the one you’ve been working on. You can now test out your template and see how it looks live on your website.

Special tags

The **Component Column** tag is only one of several tags available for use in your template. Following are the other tags that can be added:

`[#siteTitle#]`

This tag is replaced with the title of your website (e.g. “My Website”) as defined in System Settings.

`[#siteName#]`

This tag is replaced by the Site URL Name of your website (e.g. “Mywebsite.com”) as defined in System Settings.

`[#siteTplPath#]`

This tag is replaced with the full path to your site’s template and is a good idea to use in front of the SRC attribute to any linked files in your template... for example:

```
<link href="#"[#siteTplPath#]css/style.css" rel="stylesheet" type="text/css" media="screen" />
```

`[#showDateTime#]`

This tag is replaced by the date. You can provide an optional **FORMAT** parameter to the tag to specify how you would like the date to display (ex: `[#showDateTime_FORMAT=year#]`) with possible values of **year**, **standard**, **fullText**, or **time**.

`[#compPageNav#]`

This tag is replaced by a linked list of pages in the section (navigation) that you specify with optional parameters for display control. This tag works well for a main navigation that changes infrequently. It requires one parameter **NAVID** which specifies the section from which to pull the pages for the

navigation (ex: `[#compPageNav _NAVID=3#]`). Two other parameters can be used, **FILE** and **FILE-ACTIVE**, to specify the template files for the individual navigation items (ex: `[#compPageNav _FILE=nav/hdrnavitem.html _FILE-ACTIVE=nav/hdrnavitemactive.html#]`) where the file specified points to a file in the “`tmpl/comp`” folder.

`[#compLangNav #]`

This tag is replaced, if more than one language is present, by a separate mini-template file of your choosing to display a clickable list of links of available languages on the site (ex: [English](#) | [French](#) | [German](#)). It takes one parameter, **FILE**, which is used to specify the mini-template file in the “`tmpl/comp`” folder you want to show when multiple languages are present (ex:

`[#compLangNav _FILE=nav/navLang.html#]`).

`[#langChange #]`

This tag is replaced by a list of languages present in your WPM settings. It can be used by itself in your template, or be called by the tag `[#compLangNav #]` to allow for its display only when more than one language is present. This tag takes one parameter, **FORMAT**, with possible values of **slim**, **standard**, or **full** (ex: `[#langChange _FORMAT=slim#]`).

Additional templates for different languages

Your main templates (home.html, content.html, search.html) should use the **Default Language** you set for your website. If you have more than one language for your site, you’ll probably have different templates for each language.

To use a template for an additional language, name it like this: **[template]_[language-code].html**. So, if you have a Spanish language homepage template, you would call it **home_es.html**. If you have German language content page template, you would name it **content_de.html**. Web Pro Manager will search for these language-specific templates first and use them if found. If not, they will use the default template for that page type.

Here are the default languages supported by Web Pro Manager and their corresponding language codes:

- **English (en)**
- **German (de)**
- **Spanish (es)**

- **French (fr)**
- **Italian (it)**
- **Portuguese (pt)**
- **Serbian (sb)**

Adding more languages

If you have an additional language which is not listed above, you can easily add it to Web Pro Manager by adding a new file to the “lang/def” folder.

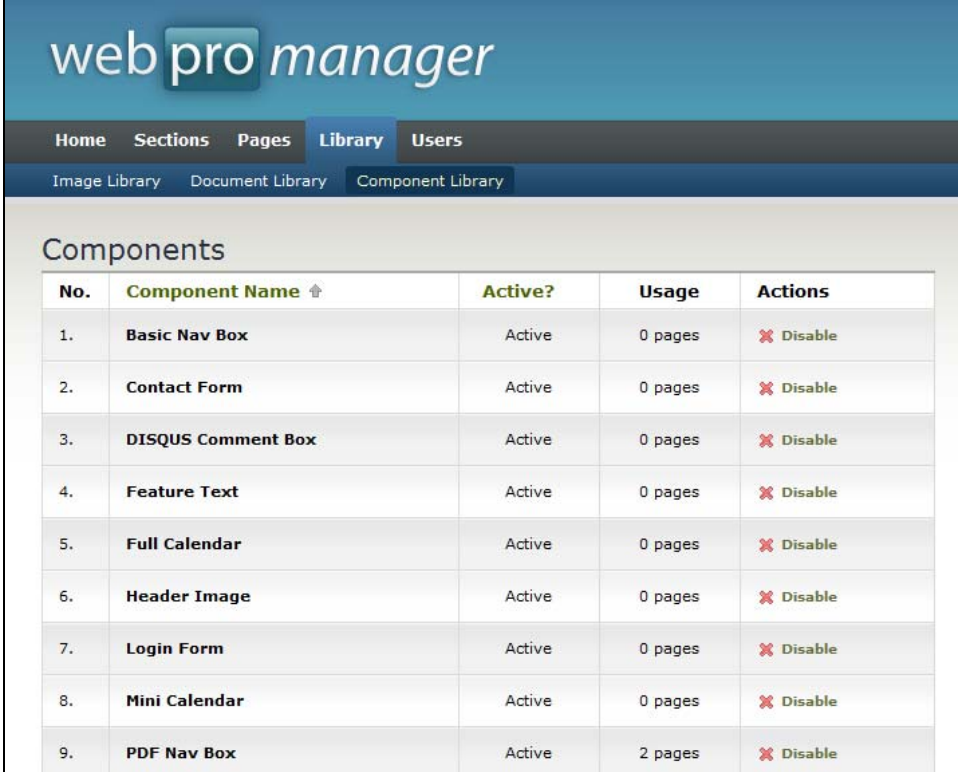
The name of the file should be its language code and it should have an extension of PHP (e.g. en.php, de.php, ...) It must have the same formatting as the other language files in that folder. To make your life easier, just make a duplicate of any language file that is already there. You’ll need to edit the variables defined in that file as well.

Customizing Your Components

What's a component?

A component is just a fancy name for a piece of content. Components are added to pages by a drag and drop editor and can be moved, edited, duplicated and deleted easily. Web Pro Manager comes pre-installed with many different components for placing text, images, navigation, forms and scripts onto your pages. You can also create your own components.

You can view the components installed in WPM in the **Component Library**. (Figure 3.1).



The screenshot shows the 'Component Library' section of the Web Pro Manager interface. It features a navigation menu with 'Home', 'Sections', 'Pages', 'Library', and 'Users'. Under 'Library', there are sub-menus for 'Image Library', 'Document Library', and 'Component Library'. The main content area is titled 'Components' and contains a table with the following data:

No.	Component Name ↑	Active?	Usage	Actions
1.	Basic Nav Box	Active	0 pages	✘ Disable
2.	Contact Form	Active	0 pages	✘ Disable
3.	DISQUS Comment Box	Active	0 pages	✘ Disable
4.	Feature Text	Active	0 pages	✘ Disable
5.	Full Calendar	Active	0 pages	✘ Disable
6.	Header Image	Active	0 pages	✘ Disable
7.	Login Form	Active	0 pages	✘ Disable
8.	Mini Calendar	Active	0 pages	✘ Disable
9.	PDF Nav Box	Active	2 pages	✘ Disable

Figure 3.1 The Component Library lists the components currently installed in your WPM.

Why do I want to customize a component?

The pre-installed components come coded with some very basic formatting for each. For instance, the Title component surrounds the title displayed with `<h1>` HTML tags.

This pre-set formatting may not work for your template, especially if you have more complicated styles behind the look of your navigation, images, titles or text. In this case, you'll need to override the pre-installed component's formatting with your own desired formatting.

Customizing a component by overriding its display

By far the fastest way to get a component to look the way you desire is to override it for your template without changing the component directly.

Here's how you do it:

1. Create a sub-folder in your "tmpl" folder named "comp", if it doesn't exist already.
2. Copy the component HTML file you wish to override from the "comp" folder in the root of your WPM installation to this folder. Note that the HTML file may be in a language sub-folder of "comp" (e.g. "comp/en"). Alternatively, you can also place the copied component in a language sub-folder in your "tmpl/comp" directory (e.g. "tmpl/comp/en").
3. You can now edit this file in any way you need and your component will display using the edited. Here's an example of the code we updated for the Title component:

BEFORE:

```
<!-- BEGIN COMP: Title -->
[#text_BEFORE=<h1>_AFTER=</h1>#]
```

AFTER:

```
<!-- BEGIN COMP: Title -->
[#text_BEFORE=<h2 class="special">_AFTER=</h2>#]
```

4. Upload your "tmpl/comp" directory to your WPM installation to see your changes.

Creating a new component

If customizing existing components just doesn't cut it for you, the next best alternative is to create a completely new component.

Before proceeding, you might want to browse the files currently in the "comp" folder and sub-folders to get a feel of the components already installed with Web Pro Manager and how they work.

Here's how you do it:

1. To create a new component, you must first make a **PHP configuration file** which defines what your component does. It can be named anything, but must end in a ".php" extension and reside in the root "comp" folder (e.g. "comp/mycomponent.php"). Without this file in place, the component will not install or run.
2. In your new **PHP configuration file**, you will set a **name** and **description** of the component. You will also set a **page-type limit** which allows the component only to run on a certain type of page (e.g. Content, Homepage, or Search type only), and you will set an **HTML template file** for display of your component on the front-end website. To make sure your **PHP configuration file** is correctly formatted, you should copy the code from a default component that most resembles your new component.

Here is an example taken directly from the **Simple Text** component's **PHP configuration file**:

```
<?php
// Required settings
$name = "Simple Text";
$description = "Adds a space to enter in any text content or HTML.";
$pageTypeLimit = ""; // Content, Homepage, Search
$templateFile = "simpletext.html";

addField("text", "Title", "(max. 200 characters)", "", "width:750px;");
addField("textarea", "Content", "", "", "width:750px; height:200px;");
addField("link-Page", "Page Link", "When 'more' is clicked on, choose a page to link to.", "", "width:300px;");
addField("image-Button", "Use This Button Image with Page Link", "", "", "width:300px;");
?>
```

3. With the **name**, **description**, **page-type limit**, and **template** set, you need to only add the fields that your component uses during editing and display. You do so with the "addField()" function. You can add as many fields as you like, of the same type or of different types.

The addField() function accepts 6 parameters in this order:

- {field ID} - determines how your field will be processed, choose ID from list below; field ID must be "text", "textarea", "linkid"(shows list of -Page or -Section), or "imgid" (shows list of images, add a dash and library name or default to "Content").
- {title} - title to display in the back-end during editing.
- {caption} - caption to display in the back-end during editing
- {default value} - default value in the back-end during editing
- {field style} - CSS style in the back-end during editing
- {template} - for lists (when field ID is 'linkid-Section') created during default processing, allows templating of individual list items using 2 tags: [#title#] and [#link#]

Here are 3 addField() examples:

```
// Adds a standard text field called Title
addField("text", "Title", "(max. 200 characters)", "", "width:750px;", "");

//Adds a page navigation from a Section chosen by the user
addField("linkid-Section", "Build navigation from this
Section", "", "", "width:300px;", "", "basicnavitem.html");

// Adds a large textarea field called Content
addField("textarea", "Content", "", "", "width:750px;", "");
```

4. Save your component **PHP configuration file** when finished.
5. Create a new component **HTML template file** with the same name that you set for your **template** in Step 2 and save it in the same folder as your **PHP configuration file**, or in a language sub-folder underneath it. In it, add the HTML that will display when this component is used. For the parts of the component that are user-driven (every field you added in the component PHP

configuration file), place a tag for that field.

Here is a list of available tags and their parameters which will be replaced by the user's chosen content:

TAG	AVAILABLE PARAMETERS
[#text#]	NUM={field number} BEFORE={text} AFTER={text}
[#textarea#]	NUM={field number} BEFORE={text} AFTER={text}
[#link#]	NUM={field number} BEFORE={text} AFTER={text} FILE={filename} FILE-ACTIVE={filename}
[#image#]	NUM={field number} BEFORE={text} AFTER={text}
[#linkedtext#]	NUM={field number for text} LINKNUM={field number for link} BEFORE={text} AFTER={text} SHOWUNLINKED={1/0} NOTEXT={text}
[#linkedtextarea#]	NUM={field number for textarea} LINKNUM={field number for link} BEFORE={text} AFTER={text} SHOWUNLINKED={1/0}

	NOTEXT={text}
[#linkedimage#]	NUM={field number for image} LINKNUM={field number for link} BEFORE={text} AFTER={text} SHOWUNLINKED={1/0} NOIMAGE={text}
[#pageid#]	
[#compid#]	
[#sitePath#]	

Following is an example taken directly from the **Simple Text** component's **HTML template file**:

```
<!-- BEGIN COMP: Simple text //-->
[#text_BEFORE=<h1>_AFTER=</h1>#]
<div id="contentBox">
    [#textarea#]
    [#linkedimage_NOIMAGE=more_SHOWUNLINKED=0_BEFORE=<BR><BR>#]
</div>
```

6. Save your component **HTML template file** when finished.

Customizing the Back-End Manager

Configuring the layout of your page editor

When you finalized your templates, you added **Component Columns** to specifying which areas contain content that's editable by the user (**Figure 4.1**).

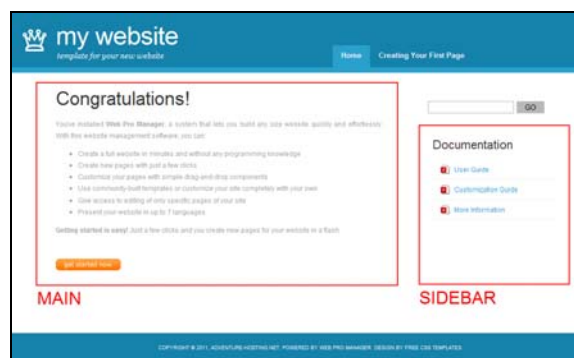


Figure 4.1 A quick reminder of what Component Columns look like in your front-end template.

Since the back-end allows for dropping in content into those **Component Columns**, it would make the most sense if the page editor had a similar layout to your front-end template. Well, it does and you have 100% control over that as well (**Figure 4.2**).

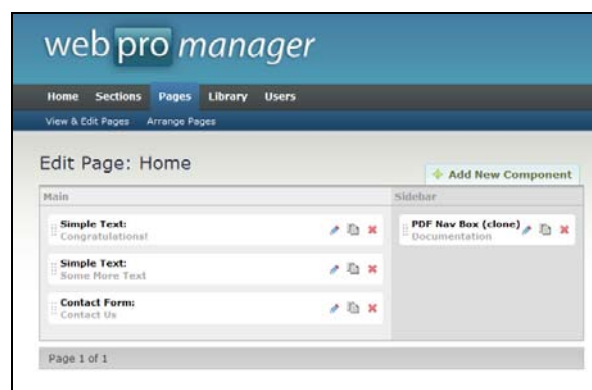


Figure 4.2 The page editor matches the template layout exactly, with a main content area and sidebar.

To change the layout of your page editor, just follow these quick steps:

1. Add a new folder under your “tmpl” folder called “layouts”.
2. Create a new HTML file for each of your templates in this folder (e.g. home.html, content.html, search.html). Add the HTML code of the layout you wish to appear in the page editor for that page type. Add the same tag you added to your templates to define your **Component Column**, but with an additional **TITLE** parameter to title your column: **[#compColumnComps#]**.

```
<table width="667" border="0" cellspacing="0" cellpadding="0" style="border:1px solid #AAA;">
  <tr>
    <td width="432" valign="top" style="background-color: #EEE;">
      [#compColumnComps_COLID=1_TITLE=Main#]
    </td>
    <td width="235" valign="top" style="background-color: #DDD;">
      [#compColumnComps_COLID=2_TITLE=Sidebar#]
    </td>
  </tr>
</table>
```

3. Upload your “tmpl/layouts” directory to your WPM installation to see your changes.

The final and simplest step

The back-end management system can also be branded with a logo of your choice to complete the customization of Web Pro Manager. To do so, just follow these simple steps:

1. Login to the Web Pro Manager (*see Chapter 2: Logging In of Web Pro Manager User Guide*).
2. From the *Welcome* page, Click on **System Settings** in the sub navigation to proceed to the **System Settings** page.
3. Click the **Edit** button to change the **System Settings**.
4. Under **Logo (for Manager only)**, choose a logo image from your local computer. This image must be no larger than 400 pixels in width and 80 pixels in height and must be in PNG format.

You should use a transparent background to make the logo feel like an integrated part of the system.

5. Click **Update** to save your new logo to the system.