

# 6

## Creating Templates and Postings

Before content management systems, websites had teams of webmasters who were in charge of putting up content. There were some problems in getting all pages to adopt a common look. Typically, a single file would be passed around as the "template" and programmers filled in the parts marked for content. As the site expanded, the task of updating pages was usually delegated to the people who provided the content. Maintaining the look and feel of the site became increasingly difficult as more people got involved in the content publishing process.

Microsoft Content Management Server provides a solution to this problem by fixing the look and feel across a website in predefined templates, which cannot be changed by authors. Pages based on the same template are alike in both design and layout. They serve as special web forms that support editable text regions for authors to enter content. In this way, developers concentrate on the programming aspect of the job, leaving the task of providing content to the subject experts.

### Something About Templates

Templates predetermine the layout, design, and behavior of a page.

You can use a single template to create multiple pages that share the same design elements. In our TropicalGreen website, we will have a template called `Plant`. There are thousands of species of plants in the tropics. Each plant's web page will use the `Plant` template and so although the content for each plant's page is different, the look and feel will be the same because they share the same template. And if you decide to remove the plant's picture from each and every single web page, all you have to do is to amend that single template and thousands of plant postings are updated.

In MCMS, templates are made up of:

- MCMS template files on the web server
- Template gallery items in the CMS repository

Each template file is linked to a related template gallery item.

## MCMS Template Files

MCMS template files dictate the overall look and feel of the web pages, how the content is positioned, the colors and fonts used, etc. They can be either ASP.NET web forms (\*.aspx) or ASP files (\*.asp). More importantly, they give developers full flexibility in programming the behavior of the web page. Since they are regular \*.aspx or \*.asp files, they enjoy the full benefits of ASP.NET/ASP programming. All the tools that ASP.NET/ASP provides can be used to create powerful templates in MCMS.

## Template Gallery Items

Template gallery items are stored in the MCMS repository. They contain additional information about the template such as its name, its description, what placeholder definitions have been defined for the template, which custom properties should be available to authors of postings, and which gallery it belongs to. In addition, the template gallery item contains the information about which template file it is bound to.

## Before We Begin

In this chapter, we will create the Plant template. We will create the template file and the template gallery item. Our first template will simply be used to generate web pages. At the end of this chapter we will use the template to generate the first plant fact sheet (or posting). The material in Chapter 7 extends our work to include editable regions (placeholders) for authors to enter content.

The Plant template will be used later to create fact sheets about plants used as a resource for hobbyists wishing to find out more about certain plants. Club members are avid gardeners. They often exchange gardening tips with fellow members. Younger green thumbs look up the information posted in the plant fact sheet to tap into the wealth of experience that has been posted by fellow gardeners.

Typically, before you begin creating templates, you would have planned the entire site structure and storyboard of the site. The complete draft of the blueprint would have been discussed with stakeholders and printed out. As our focus is on the technical aspect of the project, we won't be discussing that here.

At this point, we do not need to worry about the content (information about plants). We will concentrate only on the skeletal structure of the template.

The resulting skeleton is shown on the next page. We will use this shell to provide several key lessons to give us a solid framework on which to build the rest of the site.

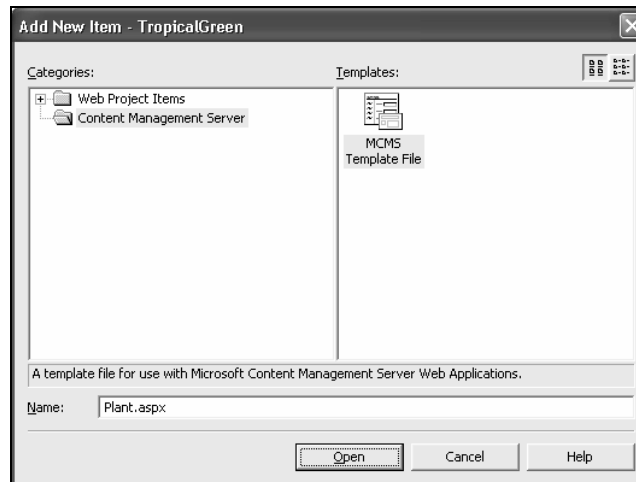


## Creating the Plant Template File

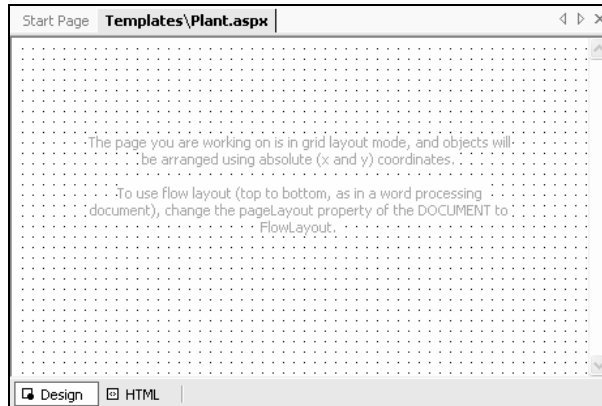
In Chapter 2, we installed the Developer Tools that integrate MCMS with Visual Studio.NET. We will use the add-ons to create the template file in Visual Studio .NET.

1. Open the TropicalGreen Solution in Visual Studio .NET. Point to Start | Programs | Microsoft Visual Studio .NET | Microsoft Visual Studio .NET. On the Start Page, click Open Project. Browse to the location of the TropicalGreen.sln file and click Open.
2. In Solution Explorer, right-click on the Templates folder created in Chapter 5. Select Add | Add New Item from the pop-up menu. The Add New Item dialog opens.
3. In the Categories column, select Content Management Server. Select the MCMS Template File option that appears in the Templates column.

In the Name input field enter Plant.aspx. Click Open. A new template file is created.



4. The MCMS template file you added looks just like a regular ASP.NET web form.



## Editing the Template File

On adding the MCMS template file to the project, `Plant.aspx` file opens and is ready for editing.

1. The dots indicate gridlines. The page is currently using grids to position objects—the default setting for web forms created using Visual Studio .NET to allow positioning of controls using drag and drop.

Using `GridLayout` means that all controls and text will be at fixed positions. This can cause problems in combination with MCMS when content entered by authors into Placeholder controls is longer than expected and overlaps with other content later. To prevent this from happening, change the layout to `FlowLayout`.

Right-click anywhere within the page. Select `Properties` in the pop-up menu. Change the value of `Page Layout` from `GridLayout` to `FlowLayout`. The grid disappears.

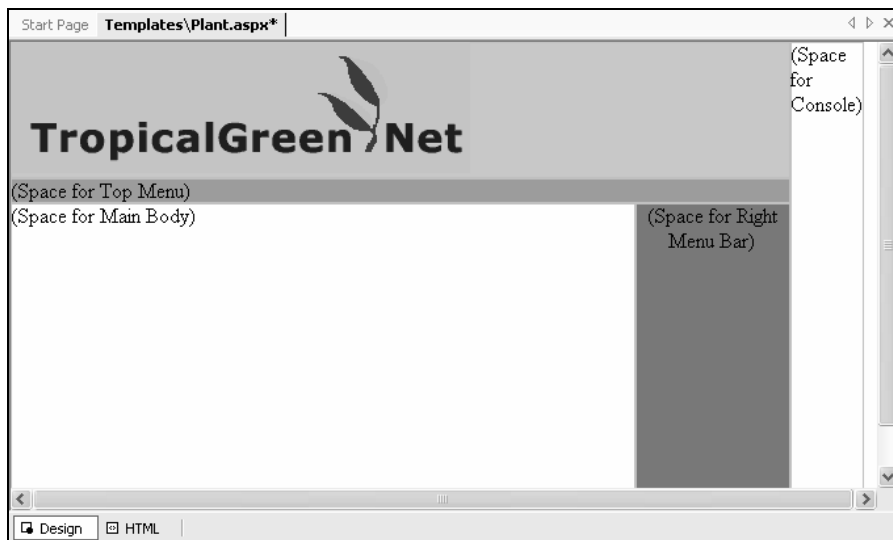
2. Switch to HTML view by right-clicking anywhere on the page and selecting `View HTML Source` in the pop-up menu (or press `Ctrl + Page Down`). Enter the HTML table below as you see it between the `<form>` and `</form>` tags. The table provides the basic format for the page with the logo and horizontal menu at the top of the page and the vertical menu on the right-hand side. The rest of the space is for the main body or content. Text markers for content (e.g. `(Space for Console)`) will be replaced later in this chapter and in subsequent chapters.

```
<table width="100%" border="0" cellpadding="0" cellspacing="0" height="100%">
<tr>
  <td width="100%" colspan="2" valign="top" bgcolor="#FFCC00">
    
  </td>
  <td rowspan="10" valign="top">
    (Space for Console)
  </td>
</tr>
<tr bgcolor="#66CC33">
  <td colspan="2">(Space for Top Menu)</td>
```

```
</tr>
<tr>
  <td valign="top">
    (Space for Main Body)
  </td>
  <td class="RightMenuBar" width="20%" valign="top" height="100%"
    align="center" rowspan="2" bgcolor="#669900">
    (Space for Right Menu Bar)
  </td>
</tr>
</table>
```

3. The logo of the TropicalGreen club can be obtained from the code download section in the book's companion site. Download the `logo.gif` file and add it to the solution in the Images folder of the TropicalGreen project
4. Your developer's instinct is probably prompting you to test the template code. Switch over to Design View and see what you have created. Click on the Design tab at the bottom of the Visual Studio designer, or right-click anywhere on the page and select View Design. Or, for those of us who like shortcuts, press *Ctrl + Page Down*.

This is plain for a template file but it covers the essence of it. We'll add in the cosmetic effects later.



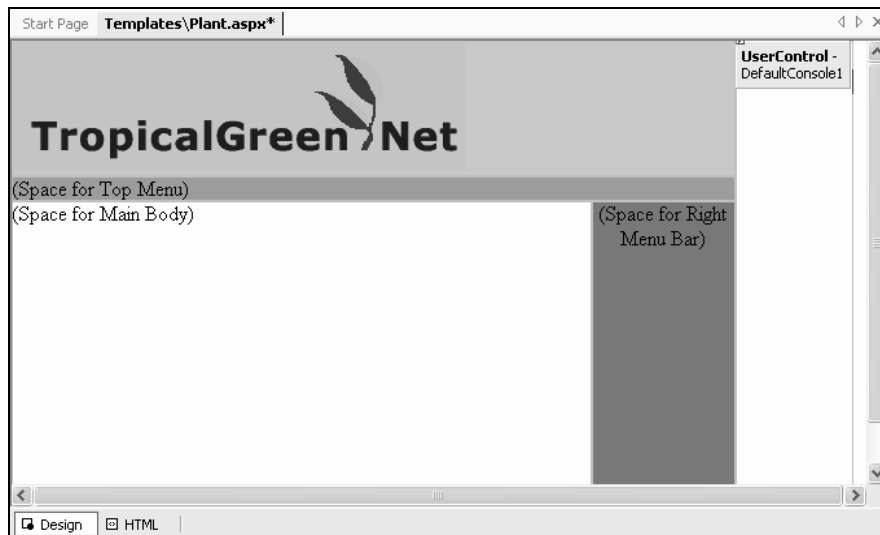
## Adding the Default Console

Note that we have reserved a cell on the right of the table for the console. We will be introduced to the Web Author console in the next few chapters. For now, add it into the template file. To add the Default Console:

1. In Design view, drag and drop the `DefaultConsole.ascx` file into the table cell that contains the words `(Space for Console)`. You can find `DefaultConsole.ascx` in the

## Creating Templates and Postings

- /Console/ folder of the solution. This widget was added by the wizard when you created the TropicalGreen Project.
2. Delete the words (Space for Console). In Design View, your template file should now look like the screenshot below.
  3. Make it a habit to press *Ctrl + S* or press the Save button once in a while when writing your code to save your work. If you have not done so, save now.
  4. Build the solution by pressing *Ctrl + Shift + B*.



We're halfway there. Let's create the second part of a template—the template gallery item (or the part that resides in the content repository).

## Creating a Template Gallery Item

Creating an ASPX template file as we did in the previous section is only half the story. We still need to create an entry in the MCMS repository (or the template gallery item) to let MCMS know about the template file we've created.

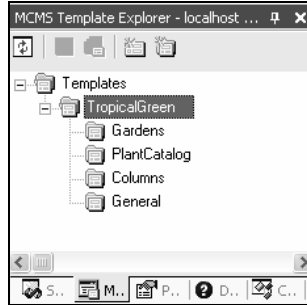
## Viewing the MCMS Template Explorer

An additional window appears in Visual Studio .NET for projects that are MCMS enabled: the MCMS Template Explorer.

The MCMS Template Explorer (shown on the next page) provides a graphical interface to manage template galleries and template gallery items. In Chapter 4, we created template galleries using Site Manager. We can do the same using the MCMS Template Explorer in Visual Studio .NET.

This is a good feature that nicely integrates Visual Studio .NET with MCMS. It gives template designers the ability to use a single tool to maintain templates. They don't have to use Site

Manager to maintain template galleries; all required tasks to create and update templates can be performed from within Visual Studio .NET.



If the MCMS Template Explorer Window does not appear on the screen, you can bring it to the surface by pointing to View | Other Windows | MCMS Template Explorer from the Visual Studio .NET toolbar.

Should the option to open the MCMS Template Explorer not be available, chances are, your project is not MCMS-enabled. You can choose to:

1. Select Project | Enable as MCMS Project from the toolbar.
2. Delete the project and follow the steps outlined in Chapter 2.

## Creating the Plant Template Gallery Item

In the MCMS Template Explorer, right-click on the template gallery named PlantCatalog that was created in Chapter 5.

Select New Template from the pop-up menu, and name the new template Plant.

Look at the icon next to the newly created template. It looks like a broken object with a red check next to it:



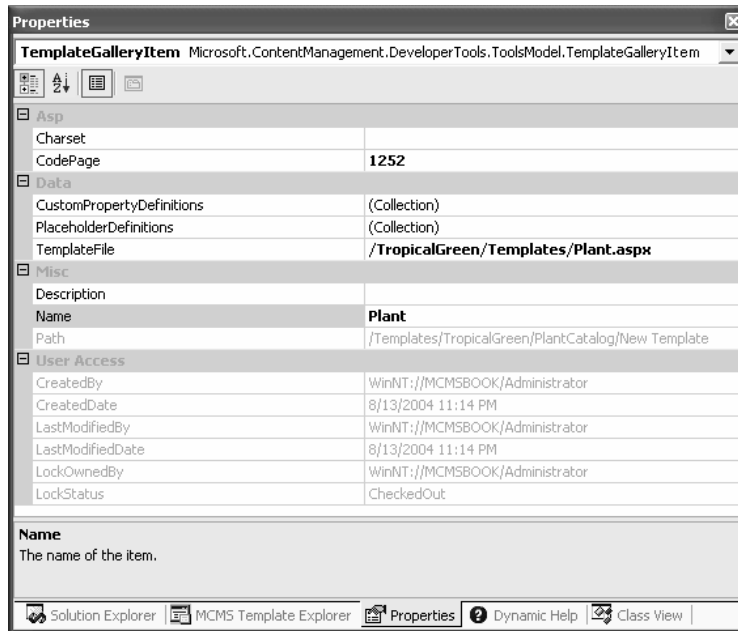
The broken template indicates that the template gallery item is not linked to a physical template file. The red check means that you have checked out the template gallery item. No one else can change the template gallery item while it's checked out to you.

## Linking the Template Gallery Item to the ASPX Template

We need to link the template gallery item to the template file created earlier. This is an important step. Miss it and your template would not have a physical form.

## Creating Templates and Postings

Right-click on the newly created Plant template in MCMS Template Explorer and select Properties. The Properties window of the template gallery item appears. For now, zoom in to the TemplateFile field and set it to point to the /TropicalGreen/Templates/Plant.aspx file we have just created, as shown below. To save typing effort, click on the ellipsis and browse to the Plant.aspx template file.



Look at the icon that represents the template gallery item—it is no longer showing as broken:



That's because the template gallery item is now linked to a template file.

## Saving the Template Gallery Item

Your changes are not committed to the database yet. To do so, right-click on the Plant template and select Save. Alternatively, click the save icon in the MCMS Template Explorer. Notice that the name of the template changes from bold text to regular text. If the template name is in bold, it means recent changes haven't been saved to the database yet.

Use the save button to commit changes to the database while the template is checked out to you.

## Checking In the Template Gallery Item

In the Template Explorer, right-click the Plant template and select Check In. The red check beside the icon of the template gallery item disappears after you have checked in the template.



The Check-In and Check-Out functions of the MCMS Template Explorer are useful when you are working with other developers on the same project. When a developer needs to amend the properties of a template gallery item, he or she has to check it out first. When the template gallery item is checked out, no one else can change any of its properties. In this way, the developer can safely make any changes to the object without worrying that someone else is unwittingly overwriting any work. Keep in mind this does not reserve the template ASPX file, just the template gallery item in the MCMS database.

Remember to check in the template gallery item when you are done with the changes, or the template will be checked out indefinitely. You don't want to get urgent phone calls from co-workers when you go on vacation as no one else can work on the template!

Checking in a template gallery item automatically saves it to the database.

And we are done! The template is now complete. We are ready to create our first posting.

## Creating the First Posting

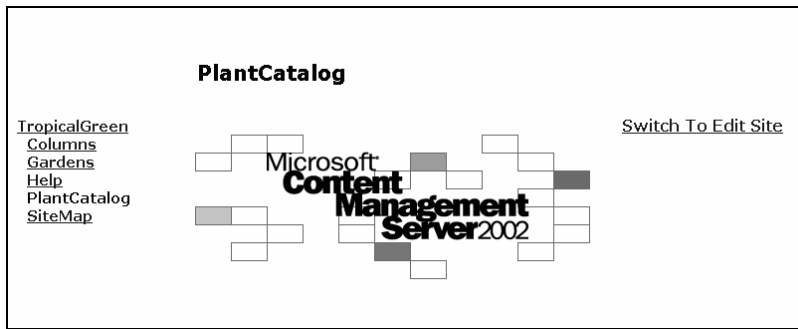
Creating the first posting is a chicken-and-egg problem. Postings are created by clicking the Create New Posting option in the Web Author Console. If we don't have any postings within the channel in the first place, we don't have a posting with a Console to use. How then, do we create the first posting?

The good news is that channels can be accessed from a browser just like postings. For any channel that does not contain a posting MCMS renders a default channel rendering script, called the channel cover page. This cover page also contains a console.

## Opening the Built-In Channel Rendering Script

To see this in action, open Internet Explorer and enter `http://localhost/TropicalGreen/PlantCatalog` in the address bar. If you get a login screen, type in the username and password of the Windows account you are using. You will get the page as shown overleaf. This is the built-in channel rendering script.

The built-in channel rendering script is just a regular old-style Active Server Page called `Cover.asp`. You can find it at `(Installation)\Microsoft Content Management Server\Server\IIS_NR\Shared`.

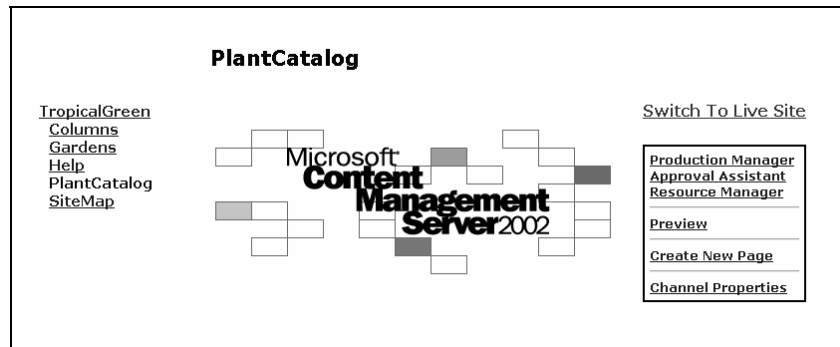


The Channel Cover Page can also be viewed by accessing <http://localhost/channel.s>.

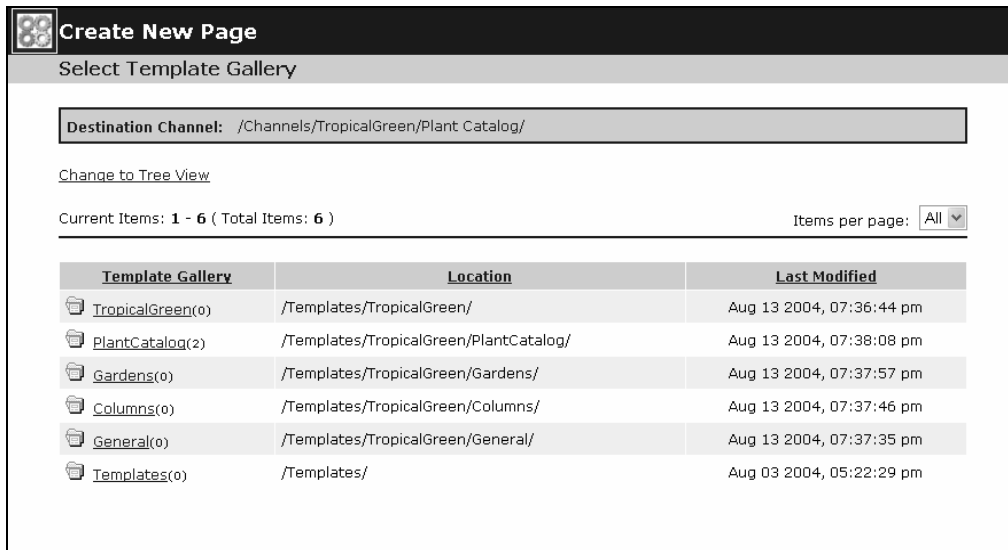
### Creating a New Page

Look at the Web Author Console on the right-hand side of the built-in channel rendering script. We use the buttons presented in the Console to create a new page.

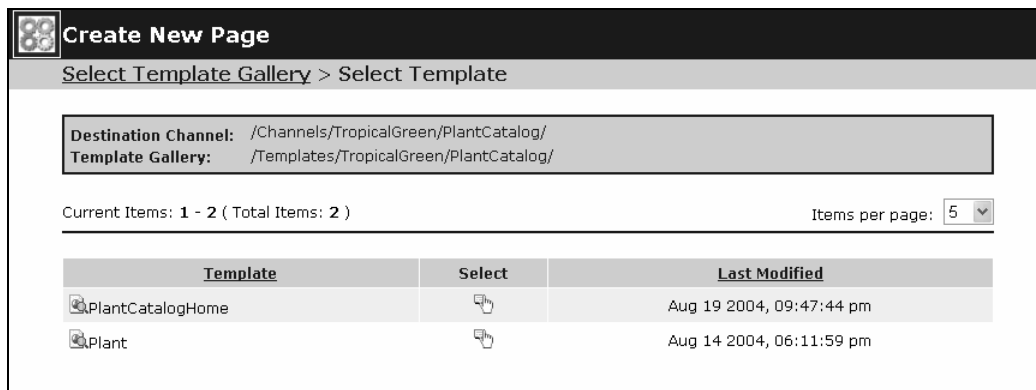
1. Click Switch to Edit Site in the Web Author Console. You are now in edit mode. The Web Author shows you several options that are available to you.
2. In the Web Author Console, click Create New Page.



3. The Select Template Gallery dialog appears. Select the template gallery PlantCatalog. If you can't see the PlantCatalog template gallery, expand the number of items in the page to All.

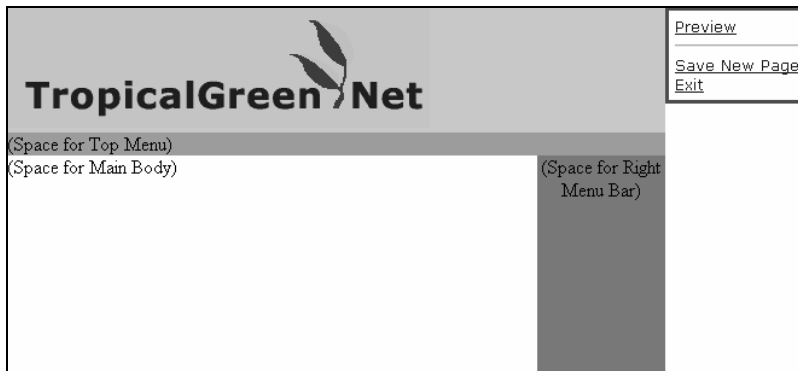


- Select the Plant template by clicking on the hand icon () in the second column.

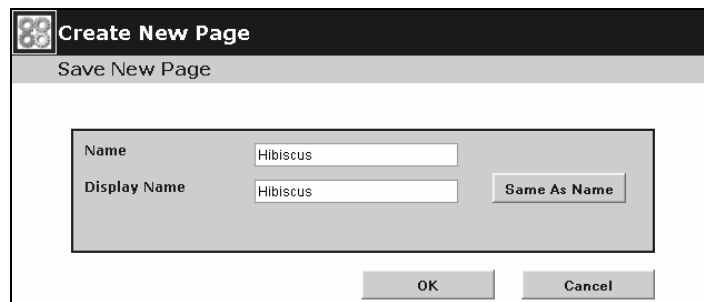


- A posting based on the Plant template is displayed. For now, we have not added any editable regions to the template so there is nothing we can do to change its appearance. Click Save New Page in the Web Author Console.

Creating Templates and Postings



6. The Save New Page dialog appears. Set the Name and Display Name of the new page to Hibiscus.



7. Towards the bottom of the Web Author Console, click Submit. The posting is immediately published to the live site. Clicking Switch To Live Site allows you to view the published posting, which should look something like that shown below.



Observe the URL in the address bar: `http://localhost/NR/exeres/AD955D2B-4B33-4AEA-AF96-B611B2622304_frameless.htm?NRMODE=Published`. The GUID in the middle may be different but no matter how you look at it, it's ugly. In an earlier chapter, we discussed how MCMS adds instructions to a URL. This is the URL with the added instructions. MCMS converts all those clean URLs (like `http://localhost/tropicalgreen/plantcatalog`) to this ugly URL for its own use.

## The Difference between a Template File and a Regular ASP.NET Web Form

When you add an MCMS template to an MCMS template project, the wizard does a few things behind the scenes. Back in Visual Studio .NET, open `Plant.aspx` and toggle to HTML View by selecting `View | HTML Source` from the toolbar or pressing `Ctrl + Page Down`.

At the top of the HTML code, you will see a page-level Register directive to register the `Microsoft.ContentManagement.WebControls` namespace and assign it an alias of `cms`. `Microsoft.ContentManagement.WebControls` contains classes that are required to support all out-of-the-box Placeholder controls.

```
<%@ Register TagPrefix="cms"
    Namespace="Microsoft.ContentManagement.WebControls"
    Assembly="Microsoft.ContentManagement.WebControls,
    Version=5.0.1200.0,
    Culture=neutral,
    PublicKeyToken=31bf3856ad364e35" %>
```

Further down, in the HTML head section, you will find the `RobotMetaTag`:

```
<cms:RobotMetaTag runat="server" id="RobotMetaTag1"></cms:RobotMetaTag>
```

To understand what the `RobotMetaTag` does, look at the source code of a posting:

1. View the posting created earlier by entering its URL into the address bar of your browser (e.g. `http://localhost/tropicalgreen/plantcatalog/hibiscus.htm`).
2. Right-click on the browser window and select `View Source` from the pop-up menu.

The `RobotMetaTag` writes out two lines of code within the `<HEAD>` tags of the generated posting.

The first line is:

```
<meta name="ROBOTS" content="FOLLOW, INDEX">
```

By default, most Search Crawlers will index all pages of a site. You can prevent a page from being indexed by setting the content attribute of this tag to `"noindex"`. A value of `"nofollow"` prevents pages that are linked to this page from being indexed. The value of the content attribute takes its cue from the `Web Robots Can Crawl Links` (`IsRobotFollowable` in the PAPI) and `Web Robots can Index this Channel's Navigation` (`IsRobotIndexable` in the PAPI) properties of the posting.

The second line looks like this:

```
<base href="(fully qualified path of the template file with querystring)">
```

A live example of that would be:

## Creating Templates and Postings

---

```
<base href="http://www.tropicalgreen.net/Templates/Plant.aspx?
NRORIGIURL=%2fNR%2fexeres%2f994840E6-823F-4B7C-824F-
237CD81B28FA%2cframeless%2ehtm&FRAMELESS=true&
NRNODEGUID=%7bF3658973-8735-484E-845D-22C7A70D4070%7d&
NRCACHEHINT=ModifyLoggedIn">
```

The `<base>` tag provides support for relative links within the template. For example, you may have an image that is coded as ``. Without the `<base>` tag, the browser will look for the image based on the path of the generated web page. In our Plants example, this could be anything from `/shrubs/myimage.gif` to `/Cactus/myimage.gif`. As we can't predict the path of the page, it makes more sense to look for `myimage.gif` based on the path of the template file.

**Warning:** The `<base>` tag causes internal bookmarks to go awry.

**Secret:** Any regular ASP.NET web form can be used as an MCMS template file. If your template file hosts placeholder controls, add the above page level Register directive. Should you need to prevent a search engine from indexing the postings generated by this template or have relative links within your page, add both the `RobotMetaTag` and the page level registration.

## Summary

Congratulations—you've now created your first template! Templates ensure that a consistent look and feel is applied across pages that are generated from it. They also act as special kinds of web forms where authors can enter content in special editable regions without needing to know how to create a web page.

Templates are made up of two distinct parts:

- Template files
- Template gallery items

Template files are created and managed in Visual Studio .NET in MCMS-enabled projects. Developers define the layout, design, and behavior of the page by writing code in template files.

MCMS provides an add-on to Visual Studio .NET called MCMS Template Explorer. Developers use the MCMS Template Explorer to create MCMS template gallery items in the MCMS repository. Once created, template gallery items are linked to template files.

You created the first posting based on the Plant template and viewed it from the browser. Because this is the first posting in the website, we used the Web Author Console available from the built-in channel rendering script to access the Create New Page function.

The Plant template is at its bare minimum. In the next chapter, we will add placeholders, which are editable regions where authors can upload content.