

---

# Microsoft Content Management Server 2002 Hands-On Lab: Building the Site Framework

---

## **Objectives**

After completing this lab, you will be able to:

- Create a new channel.
- Create an ASP.NET template.
- Define placeholders for a template.
- Add and bind Microsoft® Content Management Server (MCMS) 2002 placeholder server controls.
- Test your new template.
- Create new server-based actions and status controls for Content Management Server consoles.
- View and use new actions in the WoodgroveNet sample site.

**Estimated time to complete this lab: 120 minutes**

## Exercise 1

### Creating a New Channel

In this exercise, you will use the Site Manager to create a new channel in which news items can be placed.

#### ✦ To create a new channel

1. On the **Start** menu, point to **Programs | Microsoft Content Management Server** and click **Site Manager**.
2. In the **Site Manager** dialog box, ensure the option **Log on as Administrator** is selected and click **Start**.
3. Ensure that **Channels** is selected in the left-hand navigation bar.
4. In the **Channels** tree-view, expand the **WoodgroveNet** channel.
5. Expand the **About Us** channel.

Note that two channels within **About Us** exist — **Careers** and **Press Releases**.

6. Select the **About Us** channel.
7. From the **File** menu, point to **New** and click **Channel**.  
The **New Channel** dialog box appears.
8. Type **News\_Items** into the **Name** text box.
9. Type **News Items** in the **Display Name** text box.
10. Click **OK**.

The **News\_Items** channel appears beneath **About Us** in the channel hierarchy.

11. Close Site Manager.

## Exercise 2

### Creating a New Template

In this exercise, you will use the Microsoft Visual Studio® .NET development environment to create a new template file for use with the News\_Items channel created in the previous exercise.

You will first create a template in the MCMS database, which is done through the MCMS Template Explorer in the Visual Studio .NET development environment. You will then create the corresponding template file, and associate it with the template in the CMS database. Finally, you will create and bind placeholders in the new template. These will be HTML placeholders.

#### ⚡ To add a template to the MCMS database

1. On the **Start** menu, point to **Programs | Microsoft Visual Studio .NET**, and click **Microsoft Visual Studio .NET**.
2. From the **File** menu, point to **Open** and click **Project**.
3. In the **Open Project** dialog box, navigate to the **C:\Program Files\Microsoft Content Management Server\Sample Data\WoodgroveNet** folder.
4. Select **WoodgroveNet.sln** and click **Open**.
5. From the **View** menu, point to **Other Windows** and click **MCMS Template Explorer**.
6. In the **MCMS Template Explorer** window, expand the **Templates**, the **WoodgroveNet**, and the **About Us Net** nodes.
7. Right-click **About Us Net** and select **New Template**.
8. Rename the new template as **News Items ASPX**.

#### ⚡ To create the ASP.NET template file

1. In the **Solution Explorer** window, expand the **WoodgroveNet** project and select the **Templates** folder.
2. From the **Project** menu, select **Add New Item**.
3. In the **Add New Item** dialog box, select **Content Management Server** in the **Categories** tree-view.
4. Select **MCMS Template File** in the **Templates** list.
5. In the **Name** text box, type **News\_Items.aspx** and click **Open**.  
A new document is added.
6. Right-click the background of the document and select **Properties**.
7. In the **DOCUMENT Property Pages** dialog box, change the **Page Layout** property to **Flow Layout** and click **OK**.

**⚡ To associate the template file with the CMS template**

1. In the **MCMS Template Explorer** window, select **News Items ASPX**.
2. In the **Properties** window, click the **TemplateFile** property.
3. Click the ellipsis button [...] in the **TemplateFile** property.  
The Select File dialog box appears.
4. Select the **Templates** folder in the tree-view.
5. Select the **News\_Items.aspx** file in the list box.
6. Click **Select**.

**⚡ To create placeholder definitions**

1. Click the **News Items ASPX** template in the **MCMS Template Explorer** window.
2. In the **Properties** window, click the **PlaceholderDefinitions** property.
3. Click the ellipsis button [...] in the **PlaceholderDefinitions** property.  
The Placeholder Definition Collection Editor appears.
4. Click the down arrow next to the **Add** button, and select **HtmlPlaceholderDefinition**.
5. In the **Properties** list on the right-hand side of the dialog, apply the following properties:
  - AllowHyperlinks : **False**
  - AllowLineBreaks : **False**
  - Formatting : **NoFormatting**
  - Description : **Enter Date Here**
  - Name : **NewsDate**
  - AllowAttachments : **False**
  - AllowImages : **False**
  - MustUseResourceGallery : **False**
  - UseGeneratedIcon : **False**
6. Click the down arrow next to the **Add** button, and select **HtmlPlaceholderDefinition**.
7. In the **Properties** list on the right-hand side of the dialog, apply the following properties:
  - AllowHyperlinks : **False**
  - AllowLineBreaks : **False**
  - Formatting : **NoFormatting**
  - Description : **Enter Headline Here**
  - Name : **Headline**
  - AllowAttachments : **False**
  - AllowImages : **False**

- MustUseResourceGallery : **False**
  - UseGeneratedIcon : **False**
8. Click the down arrow next to the **Add** button, and select **HtmlPlaceholderDefinition**.
  9. In the **Properties** list on the right-hand side of the dialog, apply the following properties:
    - AllowHyperlinks : **True**
    - AllowLineBreaks : **True**
    - Formatting : **NoFormatting**
    - Description : **Enter News Text Here**
    - Name : **NewsText**
    - AllowAttachments : **False**
    - AllowImages : **False**
    - MustUseResourceGallery : **False**
    - UseGeneratedIcon : **False**
  10. Click **OK**.
  11. In the **MCMS Template Explorer** window, right-click **News Items ASPX**, and select **Check In**.
  12. From the **File** menu, select **Save All**.

#### 🔍 To design the template layout and add HtmlPlaceholder controls

1. Click the **HTML** tab at the base of the **News\_Items.aspx** document.
2. Place the cursor immediately before the closing **</form>** tag, and press **ENTER**.
3. In the space you have just created, type the following HTML code to create a table:

```
<table width="100%" border="0">
  <tr>
    <td colspan="2"></td>
  </tr>
  <tr>
    <td rowspan="4" width="27%" valign="top"></td>
    <td></td>
  </tr>
  <tr>
    <td></td>
  </tr>
  <tr>
    <td></td>
  </tr>
  <tr>
    <td></td>
  </tr>
</table>
```

4. Click the **Design** tab at the base of the **News\_Items.aspx** document.

5. In the Solution Explorer window, expand the **UserControls** folder.
6. Drag the **HeadTags.ascx** item and drop it in the first row of the table in the **News\_Items.aspx** document.
7. Drag the **Header.ascx** item and drop it to the right of the **HeadTags** control you have just added in the first row of the table.
8. Drag the **Footer.ascx** item from the **UserControls** folder and drop it in the last cell of the table in the **News\_Items.aspx** document.
9. In the **Solution Explorer** window, expand the **Console** folder.
10. Drag the **DefaultConsole.ascx** item and drop it in the cell that spans four rows of the table in the **News\_Items.aspx** document.
11. From the **View** menu, select **Toolbox**.
12. Click the **Content Management Server** tab on the toolbox.
13. Drag and drop an **HtmlPlaceholder** control into the first empty table cell in the **News\_Items.aspx** document.
14. In the **Properties** window, set the following properties for the **HtmlPlaceholder** control that you have just added:
  - (ID) : **plDate**
  - EditControlHeight : **50**
  - EditControlWidth : **300**
  - PlaceholderToBind : **NewsDate**
15. Drag and drop an **HtmlPlaceholder** control into the next empty table cell in the **News\_Items.aspx** document.
16. In the **Properties** window, set the following properties for the **HtmlPlaceholder** control that you have just added:
  - (ID) : **plHeadline**
  - EditControlHeight : **50**
  - EditControlWidth : **300**
  - PlaceholderToBind : **Headline**
17. Drag and drop an **HtmlPlaceholder** control into the remaining empty table cell in the **News\_Items.aspx** document.
18. In the **Properties** window, set the following properties for the **HtmlPlaceholder** control that you have just added:
  - (ID) : **plText**
  - EditControlHeight : **600**
  - EditControlWidth : **300**
  - PlaceholderToBind : **NewsText**
19. From the **File** menu, select **Save All**.
20. From the **Build** menu, select **Build Solution**.

## Exercise 3

### Testing and Using the New Template

In this exercise, you will browse the modified WoodgroveNet site to confirm the News Items channel is displayed. You will also create a new posting in this new channel, based on the template you created in the previous exercise.

#### ⏪ To confirm the news items channel exists

1. Using Microsoft Internet Explorer, navigate to **http://localhost/woodgrovenet**.

If the Authoring console is not visible, click **Login**, and then log in with the details below.

If you are presented with the login screen, log in using the following credential:

- Domain: **CMSWorkshop**
- User Name: **Administrator**
- Password: **P&ssw0rd**
- Remember my credentials: **Checked**.

2. Click the **About Us** hyperlink.

The default Web page for the About Us channel appears, with a navigation tree on the left-hand side. This tree now displays three channels: Careers, News Items and Press Releases. News Items is the channel you created in the first exercise.

#### ⏪ To create a new page based on the news item template


1. In the navigation tree, click the **News Items** hyperlink.

As there is no default page for the News Items channel, a general welcome page is displayed.

2. Click the **Switch To Edit Site** hyperlink.
3. Click the **Create New Page** hyperlink.

The Create New Page dialog box appears.

4. In the **Template Gallery** list, click **About Us Net**.

5. In the **Template** list, click the  symbol for the **News Items ASPX** template.

6. If you are prompted with a Security Warning dialog box, click **Yes**.
7. In the first text box, select the **Enter Date Here** text and type today's date.
8. In the second text box, select the **Enter Headline Here** text and type **Woodgrove News Channel Goes Live!**
9. In the third text box, select the **Enter News Text Here** text and type **In response to popular demand, Woodgrove Bank has now implemented a news channel for customers, partners and employees.**

10. Click the **Save New Page** hyperlink.

The Create New Page dialog box appears.

11. In the **Name** text box, type **GoingLive**.

12. In the **Display Name** text box, type **News channel goes live**.

13. Click **OK**.

14. Click the **Submit** hyperlink.

15. Click the **Approve** hyperlink.

16. Click the **Switch To Live Site** hyperlink.

Your new page is displayed on the site.

17. Close Internet Explorer.

## Exercise 4

### Creating a Class that Retrieves and Displays Posting Information

In this exercise, you will create a new class that retrieves and displays posting information about the current page.

#### ✦ To add a new class to the `McmsWebControlLibrary` project

1. Switch to Visual Studio .NET.
2. In the Solution Explorer window, select the `McmsWebControlLibrary` project.
3. From the **Project** menu, click **Add Class**.  
The Add New Item dialog appears.
4. In the **Name** textbox, type `PostingSummary.cs`.
5. Click **Open**.

A new class is added to the project.

#### ✦ To add required declarations to the class

1. At the top of the `PostingSummary` class that you have just created, locate the `using System;` statement.
2. Directly below this statement, add the following `using` statements:

```
using System.Web.UI;  
using System.Web.UI.WebControls;  
using System.ComponentModel;  
using Microsoft.ContentManagement.Publishing;  
using  
Microsoft.ContentManagement.WebControls.ConsoleControls;
```

3. Locate the `public class PostingSummary` declaration directly below the commented summary block.
4. Position the cursor at the end of this line and press the spacebar on the keyboard.
5. Type the following code (including the colon):

```
: PostingStatus
```

`PostingStatus` is a class from the `Microsoft.ContentManagement.WebControls.ConsoleControls` namespace. This statement means that your class will inherit from the `PostingStatus` class.

#### ✦ To create the `Text` property

In this procedure, you will provide a string value for the `Text` property of the base `PostingStatus` class from which your class inherits.

1. Place the cursor at the beginning of the `public PostingSummary()` function declaration line, and press ENTER to create a new line.
2. In the space you have just created, enter the following code to create the `Text` property:

```
new public string Text
{
    get
    {
        string sOutput;
        Posting thisPosting;
        CmsHttpContext thisContext;
        thisContext = CmsHttpContext.Current;
        thisPosting = thisContext.Posting;
        sOutput = "<B>Page Name: </B>"
            + thisPosting.DisplayName
            + "<HR>"
            + "<B>Created By: </B>"
            + thisPosting.CreatedBy.ClientAccountName
            + "<BR>"
            + "<B>Created On: </B>" + thisPosting.CreatedDate
            + "<HR>";
        return sOutput;
    }
}
```

3. From the **File** menu, click **Save All**.

## Exercise 5

### Creating a Class that Provides Channel Creation Actions

In this exercise, you will create a new class that provides channel creation capabilities for the WoodgroveNet site.

#### ✦ To add a new class to the McmsWebControlLibrary project

1. In the Solution Explorer window, select the **McmsWebControlLibrary** project.
2. From the **Project** menu, click **Add Class**.  
The Add New Item dialog appears.
3. In the **Name** textbox, type **ChannelCreator.cs**.
4. Click **Open**.  
A new class is added to the project.

#### ✦ To add required declarations to the class

1. At the top of the ChannelCreator class that you have just created, locate the **using System;** statement.
2. Directly below this statement, add the following **using** statements:

```
using System.Web.UI;  
using System.Web.UI.WebControls;  
using System.ComponentModel;  
using Microsoft.ContentManagement.Publishing;  
using  
Microsoft.ContentManagement.WebControls.ConsoleControls;
```

3. Locate the **public class ChannelCreator** declaration directly below the commented summary block.
4. Position the cursor at the end of this line and press the spacebar on the keyboard.
5. Type the following statement:

```
: BasePostBackAction
```

BasePostBackAction is a class from the Microsoft.ContentManagement.WebControls.ConsoleControls namespace. This statement means that your class will inherit from the BasePostBackAction class.

#### ✦ To declare private variables for use in the ChannelCreator class

1. Place the cursor at the beginning of the **public ChannelCreator()** function declaration line, and press ENTER to create a new line.
2. In the space you have just created, declare the following private string variables:

```
private string sQuery;  
private string sName;  
private string sDisplay;
```

You will use these variables in the next procedure.

### ✎ To develop the createChan function

In this procedure, you will develop a function named createChan. The function will parse the query string for information about the new channel to be created, and will switch to *Update* mode if necessary (since you can only create channels in update mode). If all conditions are met for creating a new channel, the function will use the MCMS Publishing API to create a channel in the current CMS context, and will ensure that it is created permanently.

1. Directly below the variables you have declared in the previous procedure, type the following code to implement the createChan() function:

```
private void createChan()
{
    int iPos;
    CmsHttpContext thisCTX;
    Channel thisChannel;
    Channel newChannel;
    try
    {
        sQuery = this.Page.Request
            .QueryString.Get("chan").ToString();
        if (sQuery!="")
        {
            iPos = sQuery.IndexOf("^d");
            sName = sQuery.Substring(0,iPos);
            sDisplay = sQuery.Substring(iPos+2);
            thisCTX = CmsHttpContext.Current;
            thisChannel = thisCTX.Channel;
            if (thisCTX.Mode!=PublishingMode.Update)
            {
                this.Page.Response.Redirect
                    (thisChannel.UrlModeUpdate
                    +"&chan=" +sQuery,true);
            }
            newChannel = thisChannel.CreateChannel();
            newChannel.Name = sName;
            newChannel.DisplayName = sDisplay;
            thisCTX.CommitAll();
            this.Page.Response.Redirect
                (newChannel.UrlModeUnpublished,true);
        }
    }
    catch
    {
    }
}
```

2. From the **File** menu, select **Save All**.

### ⚡ To override the Text property of the BasePostBackAction base class

In this procedure, you will override the Text property of the BaseAction base class, so that you can specify what text appears in the console on the WoodgroveNet Web site. As well as providing the display text, the property will also call the createChan function that you created in the previous procedure. Although, this means that the createChan function will be called every time the text of your control is rendered, this function only creates a new channel when the site is in update mode and when an appropriate querystring is found by the function.

1. Directly below the createChan() function that you have already written, type the following code to override the Text property of the base class:

```
public override string Text
{
    get
    {
        createChan();
        return "Create Sub-Channel";
    }
}
```

2. From the **File** menu, select **Save All**.

### ✦ To override the ActionJavascript property of the BasePostBackAction base class

In this procedure you will override the ActionJavascript property of the BaseAction base class, so that you can specify what happens when the user clicks your action in the console. The property will return a string that contains client-side JavaScript for displaying a custom dialog box that retrieves the name and display name of the channel to be created. Note: You will import the dialog box in the next procedure.

1. Directly below the Text property procedure that you have already written, type the following code to override the ActionJavascript property of the base class:

```
public override string ActionJavascript
{
    get
    {
        string sJavaScript;
        sJavaScript = "var sDLGResult;"
            + "sDLGResult = window.showModalDialog"
            +
            "('\/WoodgroveNet\/CustomDialogs\/CreateChannel.aspx',"
            + "'', 'scroll:no;dialogHeight:250px;"
            + "dialogWidth:400px;edge:sunken;"
            + "center:Yes;help:No;resizable:No;"
            + "status:No;');"
            + "if (sDLGResult!='Cancelled')"
            + "{"
            + "window.navigate(window.location + '&chan='"
            + "+sDLGResult);"
            + "}";
        return sJavaScript;
    }
}
```

2. From the **File** menu, select **Save All**.

### ✦ To import the custom dialog

In this procedure you will import the custom dialog box that will be used to retrieve the channel name and display name from the user. The dialog box is a simple web form with some client-side JavaScript for returning values to the calling Web page.

1. In the Solution Explorer window, select the **WoodgroveNet** project.
2. From the **Project** menu, select **New Folder**.
3. Rename the new folder to **CustomDialogs**.
4. Select the **CustomDialogs** folder.
5. From the **Project** menu, select **Add Existing Item**.

The **Add Existing Item** dialog appears.

6. Navigate to the **C:\Labfiles\Hands-On\StarterFiles\** folder.
7. In the **Files of type** dropdown list, select **Common Web Files**.

8. Select the **CreateChannel.aspx** file and click **Open**.  
The .aspx file is added to your project.
9. Double-click the **CreateChannel.aspx** file to open it.
10. Click the **HTML** tab at the base of the document, and review the following JavaScript functions:
  - function btnClose\_onclick()
  - function btnOK\_onclick()
11. From the **File** menu, choose **Save All**.

## Exercise 6

### Adding your Classes as Actions to the Default Console

In this exercise, you will add the classes you have just created as actions to the default console used in the Woodgrove site.

#### ✦ To open the console

1. In the Solution Explorer window, expand the **Console** folder in the **WoodgroveNet** project.
2. Double-click **DefaultConsole.ascx**.  
The console opens in Design View.
3. Click the **HTML** tab at the bottom of the **DefaultConsole.ascx** page.

#### ✦ To register a prefix for the WoodgroveWebControlLibrary

1. Position the cursor at the top of the DefaultConsole.ascx page.
2. Type the following code:

```
<%@ Register TagPrefix="wg"  
Namespace="McmsWebControlLibrary"  
Assembly="McmsWebControlLibrary" %>
```

Note: The above code should be typed on one line, not as it appears in this document.

3. Press ENTER.

#### ✦ To add the TemplateSummary class as an action to the Default Console

1. From the **Edit** menu, click **Go to**.
2. In the **Line number** textbox, type **25** and click **OK**.

Your cursor should be on the line that reads:

```
<CmsConsole:PostingStatus id="PostingStatus"  
runat="server">
```

3. Press ENTER three times to create three new lines.
4. On the new lines that you have created, enter the following code:

```
<wg:PostingSummary id="PostingSummary" runat="server">  
  <%# Container.Text %>  
</wg:PostingSummary >
```

#### ✦ To add the ChannelCreator class as an action to the Default Console

1. From the **Edit** menu, click **Go to**.
2. In the **Line number** textbox, type **78** and click **OK**.

Your cursor should be on the line that reads:

```
<CmsConsole:CreateNewPageAction id="CreateNewPageAction"  
runat="server">
```

Note: The above code will appear on a single line on your computer screen.

3. Press ENTER six times to create six new lines.
4. On the new lines that you have created, enter the following code:

```
<wg:ChannelCreator id="ChannelCreator" runat="server">
  <A style="cursor:hand" onclick="<##
Container.ActionJavascript %>;">
  <## Container.Text %>
  </A>
  <BR>
</wg:ChannelCreator>
```
5. Expand the **WoodgroveNet** project in the Solution Explorer window.
6. Expand the **References** folder.
7. Right-click the **McmsWebControlLibrary** reference and click **Remove**.
8. From the **File** menu, click **Save All**.
9. Right-click the **References** folder, and choose **Add Reference**.
10. Ensure you are viewing the **.NET** tab and click **Browse**.
11. Navigate to the **C:\Program Files\Microsoft Content Management Server\Sample Data\MCMSWebControlLibrary\bin\** folder.
12. Click **McmsWebControlLibrary.dll** and then click **Open**.
13. Click **OK**.
14. From the **File** menu, click **Save All**.
15. From the **Build** menu, click **Build Solution**.

## Exercise 7

### Testing your Actions

In this exercise, you will browse the Woodgrove site to verify your actions function as expected.

#### ↙ To test the PostingSummary action

1. Using Internet Explorer, navigate to **http://localhost/woodgrovenet**.  
If the Authoring console is not visible, click **Login**, and then log in with the details below.

If you are presented with the login screen, log in using the following credential:

- Domain: **CMSWorkshop**
- User Name: **Administrator**
- Password: **P&ssw0rd**
- Remember my credentials: **Checked**

2. Click the **Switch To Edit Site** link in the console on the Web page.

The Authoring actions appear in the console.

3. Note the entries for **Page Name**, **Created By**, and **Created On**.

These entries are produced by the **Text** property of the **PostingSummary** class that you created.

#### ↙ To test the ChannelCreator action

Note the entry for **Create Sub-Channel**.


This entry is produced by the **Text** property of the **ChannelCreator** class that you created. Note this item appears as a hyperlink—this is because you included the Anchor tags (<A></A>) in the Default Console for this action.

1. Click **Create Sub-Channel**.

The custom Web dialog that you imported in Exercise 2 appears. This is because you used an override for the ActionJavascript property in the ChannelCreator class, and returned the client-side JavaScript for displaying the CreateChannel.aspx Web page as a modal dialog.

2. In the Channel Name textbox, type **Employee\_Benefits**.
3. In the Display Name textbox, type **Employee Benefits**.
4. Click **OK**.

The Employee\_Benefits channel is created. The Web dialog box returned the name and the display name to the ChannelCreator class that you developed in an earlier Exercise. The class then switched the current page to update mode and created the channel programmatically, using the MCMS Publishing API. Since the new channel does not yet have any postings, you are presented with the default welcome page.

5. Click **Create New Page**.
6. In the **Create New Page** dialog, click **General Use Net**.
7. In the **Template** list, click the  symbol for the **Home ASPX** template.

8. In the placeholder control, type **Woodgrove Bank offers full medical care and insurance for all employees, along with corporate membership of a number of health clubs.**
9. Click **Save New Page**.
10. In the **Name textbox**, type **Health**.
11. In the **Display Name** textbox, type **Health**.
12. Click **OK**.
13. Click **Approve**.
14. Click **Switch to Live Site**.

Your new channel and posting appears in the site navigation controls.

15. Close all open windows and applications.

Congratulations! You have now completed this lab.