Lab 06: Extending the Out-Of-The-Box Authoring Experience

**Lab Time:** 60 minutes

**Lab Overview:** In this lab you will practice extending the out-of-the-box authoring experience in a Publishing site. First you will leverage WSS' custom actions and augment the SharePoint navigation user experience. Next, you will utilize the Edit Mode Panel within a page layout in order to customize the experience for both edit and display modes. Finally, you will extend the Page Editing Toolbar by adding new menu items to the Page Edit Menu and Quick Access Buttons areas.

Exercise 1: Adding new items to SharePoint menus using custom actions

In this exercise you will create a Feature that will add menu items to various SharePoint menus using custom action definitions within an element manifest file within a feature.

Since you have now created and customized Visual Studio projects that compile then package the contents of the project into a WSS solution package, as well as creating a project that doesn't compile anything but simply packages the contents into a WSS solution package, we have given you a project to get started with. This project already has the directory structure created, the deployment files added and the changes to the project file to run the automated WSS solution package creation.

1. In Visual Studio, open the **Lab6Exercise1** solution located in the following directory:

c:\Student\Labs\06\_Extensibility\Lab\Lab6Exercise1\Lab6Exercise1.sln

1. The first thing you need to do is create the Feature definition file. Create a new XML file named **feature.xml** in the **Lab6Exercise1** Feature folder in the project, filling it with the following markup:

<?xml version="1.0" encoding="utf-8" ?>

<Feature xmlns="http://schemas.microsoft.com/sharepoint/"

Id="27966B78-C2E9-4C80-A538-E8AEF0356BDB"

Title="Lab 6 - Adding New Items To SharePoint Menus Using Custom Actions"

Hidden="FALSE"

Scope="Site"

Version="1.0.0.0">

<ElementManifests>

<ElementManifest Location="elements.xml"/>

</ElementManifests>

</Feature>

1. Next, create an element manifest files listed in the Feature definition file. Create a new XML file in the **Lab6Exercise1** Feature folder in the project named **elements.xml**.
2. Open the **elements.xml** file, add the following XML markup (you can omit the XML comments, they are here just as a comment to what the XML is doing):

<?xml version="1.0" encoding="utf-8" ?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<!-- create a new custom action group in Site Settings -->

<CustomActionGroup

Id="95B43F4D-E66D-4969-809E-D189E7BB388A"

Location="Microsoft.SharePoint.SiteSettings"

Title="Lab 6"

Sequence="1000" />

<!-- add a new link to site settings -->

<CustomAction

Id="834219CB-2595-493D-BA2F-A715A64CBC29"

GroupId="95B43F4D-E66D-4969-809E-D189E7BB388A"

Location="Microsoft.SharePoint.SiteSettings"

Sequence="10"

Title="Lab 6"

Description="Takes the user to wcm.litwareinc.com">

<UrlAction Url="http://wcm.litwareinc.com" />

</CustomAction>

<!-- add a new link to the site actions menu -->

<CustomAction

Id="4B10BFFB-4E66-4872-88C9-C557F8A2B2A5"

Location="Microsoft.SharePoint.StandardMenu"

GroupId="SiteActions"

Sequence="1000"

Title="Lab 6"

Description="Takes the user to wcm.litwareinc.com">

<UrlAction Url="http://wcm.litwareinc.com" />

</CustomAction>

<!-- add a new link to the document library ECB menu of all sites -->

<CustomAction

Id="D917A1CF-9F9C-4FCA-855A-910814E38759"

RegistrationType="List"

RegistrationId="101"

ImageUrl="/\_layouts/images/plslforw.gif"

Location="EditControlBlock"

Sequence="100"

Title="Lab 6">

<UrlAction Url="~site/default.aspx?listid={ListId}&amp;itemid={ItemId}"/>

</CustomAction>

</Elements>

1. Next, add the manifest file for the Feature and necessary entries in the DDF file to package up the required files. Create a new XML file named **manifest.xml** to the **DeploymentFiles** folder in the project and add the following XML markup:

<?xml version="1.0" encoding="utf-8" ?>

<Solution xmlns="http://schemas.microsoft.com/sharepoint/"

SolutionId="F24B6CA8-56D5-4500-9E6D-110F8DB3B5BB"

DeploymentServerType="WebFrontEnd"

ResetWebServer="FALSE">

<FeatureManifests>

<FeatureManifest Location="Lab6Exercise1\feature.xml"/>

</FeatureManifests>

</Solution>

1. Finally, add the following lines to the **BuildSharePointPackage.ddf** file between the comments:

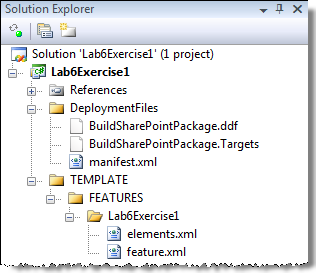
DeploymentFiles\manifest.xml

.Set DestinationDir=Lab6Exercise1

TEMPLATE\FEATURES\Lab6Exercise1\feature.xml

TEMPLATE\FEATURES\Lab6Exercise1\elements.xml

1. When you save everything, your project should look like the following image:



1. Now it's time to build deploy the WSS solution package. Build the project in Visual Studio to create the WSP file.
2. First the WSS solution package must be deployed. Open a command prompt and navigate to the following directory:

c:\Program Files\Common Files\Microsoft Shared\web server extensions\12\BIN

1. Enter the following command into the command line window and hit Enter:

stsadm -o addsolution -filename c:\Student\Labs\06\_Extensibility\Lab\Lab6Exercise1\wsp\Debug\Lab6Exercise1.wsp

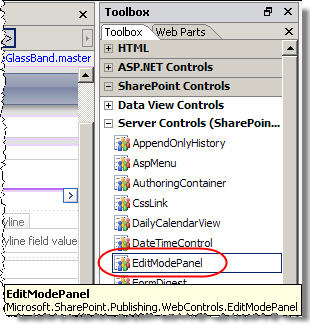
1. Launch Central Administration by selecting **Start » All Programs » Microsoft Office Server » SharePoint 3.0 Central Administration**.
2. From the **Central Administration** site, select the **Operations** tab and then select **Solution management** under the **Global Configuration** section.
3. On the **Solution Management** page, click the link on **lab6exercise1.wsp**.
4. On the **Solution Properties** page, select **Deploy Solution**.
5. On the **Deploy Solution** page, specify **Now** in the **Deploy When?** section and click **OK**.
6. Test the Feature by browsing to the **http://wcm.litwareinc.com/** site and select **Site Actions » Site Settings » Modify All Site Settings**.
7. On the **Site Settings** page, select **Site collection features** under the **Site Collection Administration** section.
8. On the **Site Collection Features** page, click **Activate** on the **Lab 6 - Adding New Items To SharePoint Menus Using Custom Actions Feature**.
9. With the Feature activated, click through the site to see the actions that were added. You should see a new group on the site's Site Settings page, a new item in the Site Actions menu, and a new item in the ECB menu in document libraries (hint: use the Site Collection Images library which contains a few default images).

In this exercise you saw how to build a feature that created new menu items and menu groups. You can use this technique to add links to the SharePoint navigation menus for your own functionality & administration pages.

Exercise 2: Utilizing the Edit Mode Panel

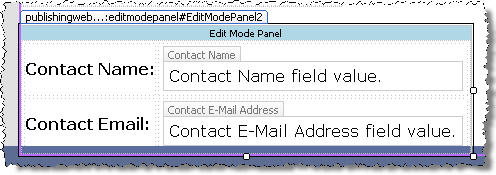
In this exercise you will see how to leverage the edit mode panel to provide content authors a way to display controls to change the page's contact person when in edit mode, but to not show this when in display mode.

1. Launch **SharePoint Designer** and open the site **http://wcm.litwareinc.com**.
2. Open the page layout **ArticleLeft.aspx** from within the **\_catalogs/masterpage** folder within the **Folder List** tool window. When prompted, check the file out.
3. While this file currently leverages an existing edit mode panel, you will create a new instance of one. From the **Toolbox** tool window, select **EditModelPanel** within the **SharePoint Controls / Server Controls** section and drag it onto the design surface just below the existing edit mode panel. The following image shows where you'll find the **EditModePanel**:

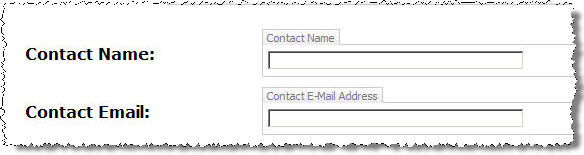


*If you have problems dragging the* ***EditModePanel*** *control onto the design surface, you may need to create some space for it first. The easiest way to do this is to switch to the* ***Code*** *view, add some text at the end of the* ***PlaceHolderMain*** *content placeholder section (usually at the end of the file), switch back to the Design view, drag the* ***EditModePanel*** *where the text is that you just added, and then remove the text.*

1. Next, add a HTML table to the edit mode panel you just added to the page with two rows and two cells in each row. In the first row, add the label **Contact Name:** in the first cell and drag the **Contact Name** control from the **Toolbox** tool window within the **SharePoint Controls / Page Fields** section in the second cell. Then, do the same for the **Contact E-Mail** Address field in the second row. You should have something that looks similar to the following image on the **ArticleLeft.aspx** page layout:



1. Save all changes, check-in the file, publish a major version when prompted and approve the publishing request file.
2. Now it's time to test the edit mode panel. Browse to the **http://wcm.litwareinc.com/PressReleases** site and create a new page using the page layout **(Article Page) Article page with image on left**. Notice that when the page is created, you now have some contact information the content author can add to the page, as shown in the following image:



1. Select **Page » Page Settings and Schedule** from the **Page Editing Toolbar**. When the **Page Settings** page loads, scroll down to the **Page Contact** section and select **Enter** **contact information**, but don't enter any information... just click **OK**.
2. Switch over to edit mode if you aren't already. Enter a name & contact email address, along with other content data on the page and select Check In to Share Draft from the Page Editing Toolbar.
3. Notice that the contact table you created in the page layout is not rendered when the page is not in edit mode; it is not visible in display mode.
4. After the page refreshes, select **Page » Page Settings and Schedule**. Scroll down to the Page Contact section and notice that the information you entered when editing the page is now shown.

In this exercise you saw how you can use the edit mode panel to show a customized edit mode experience for the content author. This same control can be used to show content when in display mode as well using the PageDisplayMode attribute.

Exercise 3: Adding custom menu items to the Page Editing Toolbar's Quick Access Button section

In this exercise you will add a new item to the Quick Access Button section within the Page Editing Toolbar. This item will always be visible and will take the user to the list item

Since you have now created and customized Visual Studio projects that compile then package the contents of the project into a WSS solution package, as well as creating a project that doesn't compile anything but simply packages the contents into a WSS solution package, we have given you a project to get started with. This project already has the directory structure created, the deployment files added, the changes to the project file to run the automated WSS solution package creation and the project has been signed with a key.

1. In **Visual Studio**, open the **Lab6Exercise3** solution located in the following directory:

c:\Student\Labs\06\_Extensibility\Lab\Lab6Exercise3\Lab6Exercise3.sln

1. The first thing you need to do is add a few references to the project. Add the following references to the project by right-clicking the References folder within the Lab6Exercise3 project in the **Solution Explorer** tool window:
   * **Microsoft.SharePoint** (hint: Component Name = Windows SharePoint Services)
   * **Microsoft.SharePoint.Publishing** (hint: Component Name = Microsoft Content Publishing and Management)
   * **System.Web**
2. Now, with the necessary references, create a new class named **PagePropertiesMenuItem.cs** in the root of the **Lab6Exercise3** project. This class will contain the logic for the button that will be placed on the Quick Access Buttons area within the Page Editing Toolbar.
3. Open the **PagePropertiesMenuItem.cs** file. The first thing you need to do is add a few namespace references to make typing a little easier as we add the necessary code. Add the following references to the top of the file:

using Microsoft.SharePoint;

using Microsoft.SharePoint.Publishing.WebControls;

using Microsoft.SharePoint.Publishing.WebControls.EditingMenuActions;

1. Next, configure the class to inherit from the **Microsoft.SharePoint.Publishing.WebControls.EditingMenuActions.ConsoleAction** class by changing the class declaration to the following:

public class PagePropertiesMenuItem : ConsoleAction

While you are about to add a few public properties to the **ConsoleAction**, you could alternatively set these properties declaratively with XML which you'll get to later in this exercise. While setting properties via XML is easier as you have no assembly to deploy, it doesn't provide as much control in your custom control. You'll notice that all properties you override only implement the **get{}** portion of the property... not the set**{}**. This gives you much more control over what you want to display on your **ConsoleAction**.

1. Next, you need add code to define who can see the menu item and when it is visible in the Page Editing Toolbar. Add the following two property overrides to the **PagePropertiesMenuItem.cs** file which will show the button regardless of the user's permissions (**UserRights = EmptyMask**) and to be shown whenever the Page Editing Toolbar is visible (**RequiredStates = EditingMenuEnabled**):

public override SPBasePermissions UserRights {

get { return SPBasePermissions.EmptyMask; }

}

public override AuthoringStates RequiredStates {

get { return AuthoringStates.EditingMenuEnabled; }

}

1. With the permissions and conditions set, the next thing you should set is the icon to be shown in the Quick Access Button. Add the following code to the following code to the **PagePropertiesMenuItem.cs** file to use the info icon that's included in a typical WSS v3 installation:

public override string ImageUrl {

get {

return "~/\_layouts/images/info16by16.gif";

}

}

1. Finally, you need to set the URL so the button will actually do something when clicked. In this case, you want to specify the URL of the View Properties page within the Pages list and pass in the ID of the page within the list. Add the following code to the following code to the **PagePropertiesMenuItem.cs** file to do this:

public override string NavigateUrl {

get {

return String.Format("javascript:window.location='{0}/Pages/Forms/DispForm.aspx?ID={1}';",

SPContext.Current.Web.Url.ToString(),

SPContext.Current.ListItem.ID.ToString());

}

}

1. Next, you need to add the manifest file for the WSP and add the necessary entries in the DDF file to package up the required files. Create a new XML file named **manifest.xml** in the **DeploymentFiles** folder and add the following XML markup:

<?xml version="1.0" encoding="utf-8" ?>

<Solution xmlns="http://schemas.microsoft.com/sharepoint/"

SolutionId="00EE4F8F-0C15-4F3C-B5E7-F2D5A75BB79E"

DeploymentServerType="WebFrontEnd"

ResetWebServer="FALSE">

<Assemblies>

<Assembly DeploymentTarget="GlobalAssemblyCache" Location="Lab6Exercise3.dll">

<SafeControls>

<SafeControl Namespace="Lab6Exercise3" TypeName="\*" Safe="True" />

</SafeControls>

</Assembly>

</Assemblies>

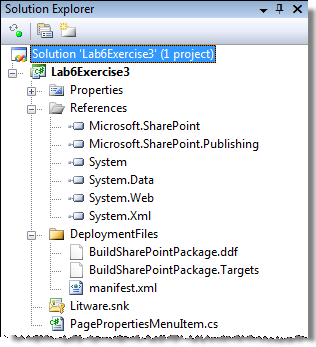
</Solution>

1. Finally, add the following lines to the BuildSharePointPackage.ddf file between the comments:

DeploymentFiles\manifest.xml

bin\debug\Lab6Exercise3.dll

1. When you save everything, your project should look like the following image:



1. Build the **Lab6Exercise3** project to compile the code and package everything into a solution package.

Now it's time to deploy the WSS solution package.

1. First the WSS solution package must be deployed. Open a command prompt and navigate to the following directory:

c:\Program Files\Common Files\Microsoft Shared\web server extensions\12\BIN

1. Enter the following command into the command line window and hit **Enter**:

stsadm -o addsolution -filename c:\Student\Labs\06\_Extensibility\Lab\Lab6Exercise3\wsp\Debug\Lab6Exercise3.wsp

1. Launch Central Administration by selecting **Start » All Programs » Microsoft Office Server » SharePoint 3.0 Central Administration**.
2. From the **Central Administration** site, select the **Operations** tab and then select **Solution management** under the **Global Configuration** section.
3. On the **Solution Management** page, click the link on **lab6exercise1.wsp**.
4. On the **Solution Properties** page, select **Deploy Solution**. On the **Deploy Solution** page, specify **Now** in the **Deploy When?** section. In the **Deploy To?** section, select **http://wcm.litwareinc.com** and click **OK**.
5. With the assembly added to the GAC and SafeControl entry added, you need to now make SharePoint aware of the ConsoleAction. Launch **SharePoint Designer** and open the site **http://wcm.litwareinc.com**.
6. Open the XML file **CustomQuickAccess.xml** from within the **\_catalogs/masterpage/Editing Menu** within the **Folder List** tool window. When prompted, check the file out.
7. Add the following XML to the **CustomQuickAccess.xml** file to declare and add the button to the page:

<?xml version="1.0" encoding="utf-8" ?>

<Console>

<references>

<reference TagPrefix="wcm401"

assembly="Lab6Exercise3, Version=1.0.0.0, Culture=neutral, PublicKeyToken=d4e5777b16a5749f"

namespace="Lab6Exercise3" />

</references>

<structure>

<ConsoleNode Sequence="1"

ConfigMenu="Add"

DisplayText="View Page Properties"

UseResourceFile="false"

Action="wcm401:PagePropertiesMenuItem"

ID="wcm401PagePropertiesMenuItemQuickAccess" />

</structure>

</Console>

1. Save your changes, check-in the **CustomQuickAccess.xml** file, publish and approve it.
2. Open a browser and navigate to any page within the **http://wcm.litwareinc.com** site and turn on the Page Editing Toolbar if it isn't visible by selecting **Site Actions » Show Page Editing Toolbar**. You should see the new button in the far left part of the **Quick Access Button** section of the **Page Editing Toolbar** as shown in the following image. Click the button to watch it take you to the display page for the page list item in the Pages list.

C:\Development\Writing\Courses\WCM401\trunk\Student Download\Labs\06_Extensibility\Figures\QuickAccessButton.jpg

In this exercise you created and added a new button to the Quick Access Button area within the Page Editing Toolbar.