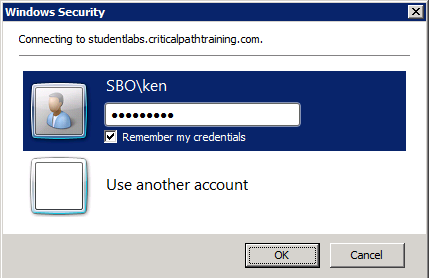
## Working with XSLT

**Lab Time**: 45 minutes

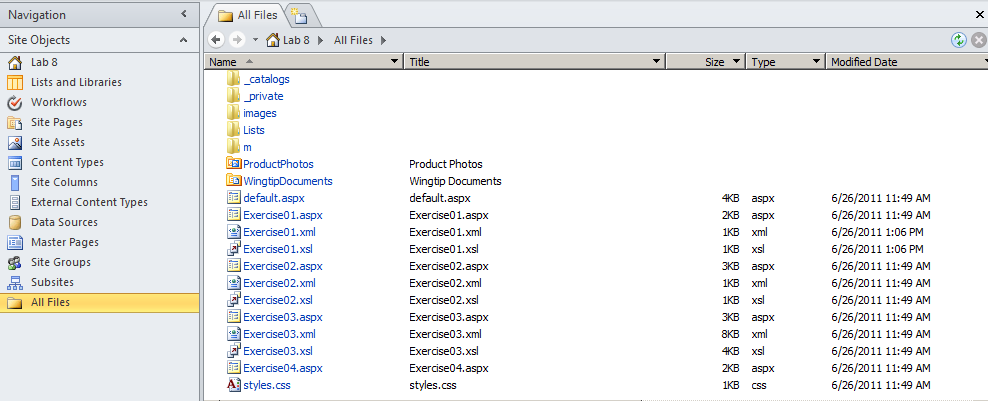
### Exercise 1: Performing Transforms with the XML Viewer Web Part

In this exercise you will work to integrate an XML document and a simple XSLT file into a SharePoint site using the XML Viewer Web Part. You will then make some edits to the XSLT file to customize its HTML output.

1. Launch SharePoint Designer 2010 and open the site at **http://[[COLLAB-SITE]]/Lab08**. If you are prompted to log in, enter the site collection owner as **[[AD\_DOMAIN]]\[[USERNAME]]**. For example, if you were supplied the user account login credentials for Ken Sanchez with a user account name **ken** and a domain name **SBO,** then theaccount name for logging in should be entered as **SBO\ken**. The password for all accounts should be set to **Password1**.



1. Click the **All Files** link at the bottom of the **Site Objects** section. This will allow you to see the virtual file system for the current site. You should be able to see the files for pages named **Exercise01.aspx**, **Exercise02.aspx**, **Exercise03.aspx** and **Exercise04.aspx**. You should also be able to see that the root folder of the site contains several XML documents and XSLT files such as **Exercise01.xml** and **Exercise01.xsl**.



1. Open **Exercise01.xml** and inspect its contents. You should be able to see that this XML document contains a root element named **PageContent** with two inner text elements named **TopHeading** and **PageBody**. When you are done looking at this file, leave it open because you will be asked to edit it in a later step.

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

<PageContent>

<TopHeading>Breaking News</TopHeading>

<PageBody>The quick brown fox jumped over the lazy dog</PageBody>

</PageContent>

1. Open the XSLT file named **Exercise01.xsl** and inspect its contents. You should be able to see that this simple XSL transform outputs with a single HTML **div** element. The div element content is generated using an **xsl:value-of** element which retrieves the text value inside the **PageBody** element nested inside the **PageContent** element. When you are done examining this file, leave it open because you will be asked to edit it in a later step.

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>

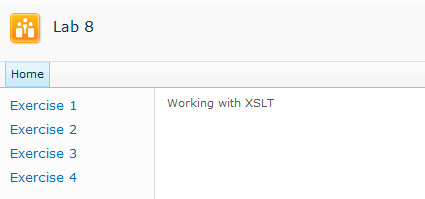
<xsl:template match="/">

<div><xsl:value-of select="PageContent/PageBody"/></div>

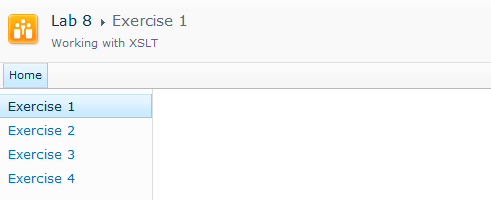
</xsl:template>

</xsl:stylesheet>

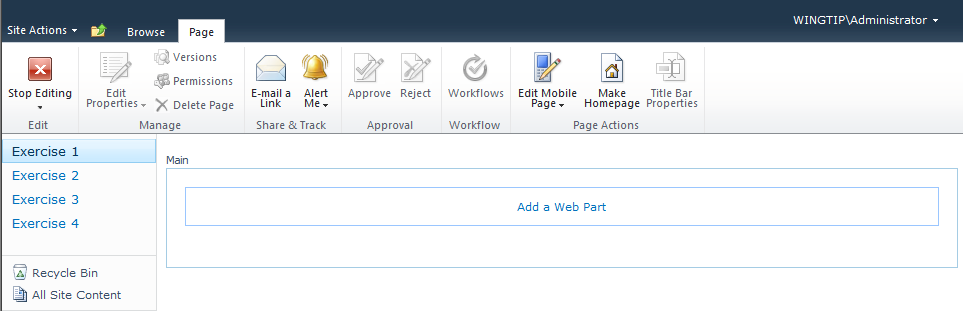
1. Launch a new session in the browser and navigate to the Collaboration Site Collection, which is located at **http://[[COLLAB-SITE]]**. Next, navigate to the child site for **Lab 8** by using the Top Link bar navigation menu of the top-level site. This will take you to the home page for the **Lab 8** site.



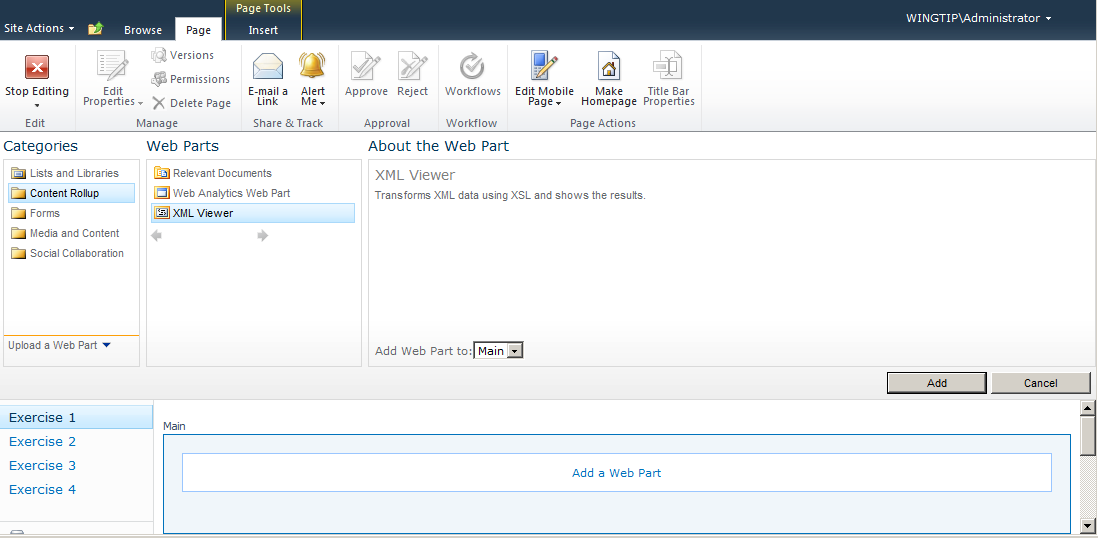
1. On the home page of the **Lab 8** site, you should be able to see that the Quick launch menu contains links to four pages that will be used throughout this lab. Click on the **Exercise 1** link which will take you to an empty Web Part page named **Exercise01.aspx**.



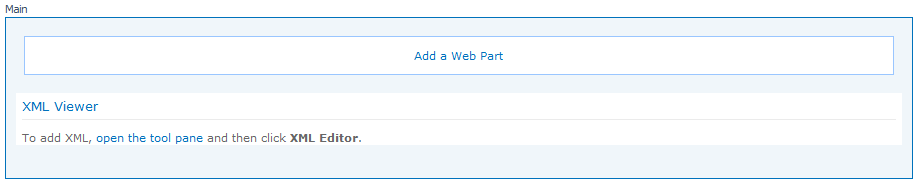
1. Click the **Edit Page** menu command from the **Site Actions** menu to transition the page into Edit Mode. You should see that this empty Web Part page consists of one Web part zone named **Main** that spans the entire content area of the page. Click the **Add a Web Part** link inside the **Main** zone, which will display the Web Part adder panel.



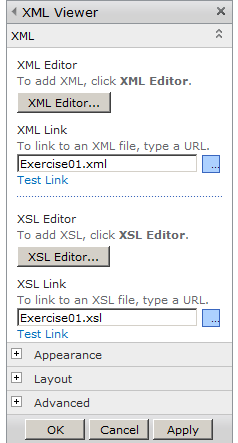
1. The Web Part adder panel should now be showing at the top of the page. Select the **Content Rollup** in the **Categories** section on the left-hand side of the Web Part add panel. Next, select the **XML Viewer** Web Part in the **Web Parts** section to the right. Add the Web Part to the page by clicking the **Add** button in the bottom right corner of the Web Part adder panel.



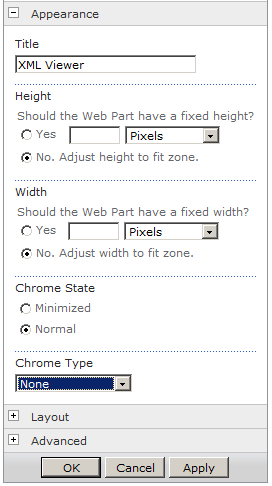
1. After you click the **Add** button you should see an empty XML Web Part. Click the link that reads **open the tool pane** to edit the properties of the new Web Part.



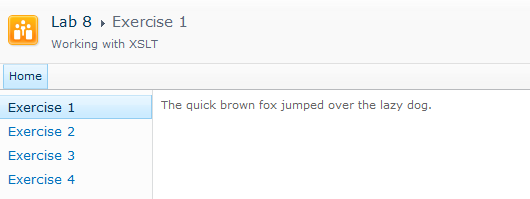
1. You should now be able to see the **XML Viewer** section in the task pane on the right-hand side of the page. Modify the value of the **XML Link** property to **Exercise01.xml**. Modify the value of the **XSL Link** property to **Exercise01.xsl**. Do not click the **OK** button until you complete the next step.



1. There is one more change you should make to configure the Web Part so that it does not display a title. Accomplish this by expanding the **Appearance** section and setting the drop down menu for the Chrome Type property to a value of **None**. When you are done, click the **OK** button to save the configuration changes you have made to the XML Viewer Web Part.



1. At this point, you should have configured the XML Viewer Web Part to perform an XSL transform on the contents of **Exercise01.xml** using **Exercise01.xsl**. You should be able to see the text from the **PageBody** element inside **Exercise01.xml** displaying on the page.



1. Now, return to **Exercise01.xml** in SharePoint Designer. Make a simple change to the text inside the **PageBody** element such as the one shown below. Next, save your changes in SharePoint Designer, which will update the content for the XML document in the content database.

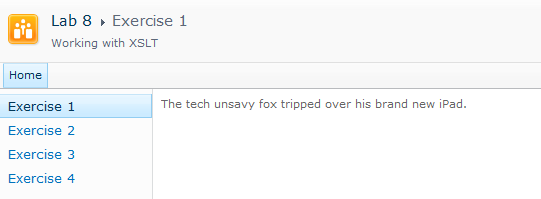
<PageContent>

<TopHeading>Breaking News</TopHeading>

<PageBody>The tech unsavy fox tripped over his brand new iPad.</PageBody>

</PageContent>

1. Return to **Exercise01.aspx** in the browser and refresh the page. You should be able to see your changes. From this you can conclude that SharePoint processes the XSL transform each time the page is loaded or refreshed.



1. Now it's time to make some changes to the XSLT file that is generating the HTML output. Return to **Exercise01.xsl** in SharePoint Designer and locate the **xsl:template** element which generates the HTML output.
2. Inside the **xsl:template** element, add a new line just above the HTML **div** element which displays the content from the **PageBody** element. Inside this new line, add a new HTML **h2** element containing an **xsl:value-of** element. Configure the **select** attribute of the **xsl:value-of** element with an XPath expression of **PageContent/TopHeading** to reference the XML **TopHeading** element nested inside the **PageContent** element. When you are done, save your changes in SharePoint Designer to update the content of **Exercise01.xsl** inside the content database.

<xsl:template match="/">

<h2><xsl:value-of select="PageContent/TopHeading"/></h2>

<div><xsl:value-of select="PageContent/PageBody"/></div>

</xsl:template>

1. Return to **Exercise01.aspx** in the browser and refresh the page. You should be able to see that your changes to the XSLT file have resulted in customizing the page output.

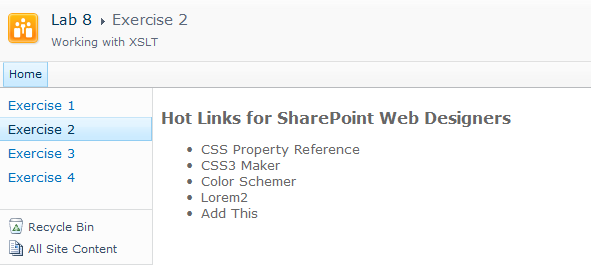


Now that you know how to create and configure an instance of the XML Viewer Web Part, you will not be required to do it again in the remaining exercises of this lab. For your convenience in the next two exercises, you will work with Web Part pages that already contain a preconfigured instance of the XML Viewer Web Part. You will only be required to modify XSLT files to complete your work.

### Exercise 2: Generating HTML Elements with Dynamic Attribute Values

In this exercise you will work to with an XSLT file which generates the HTML for a list of links. You will be required to create HTML attributes whose values are retrieved dynamically from an XML document. This will give you experience learning how to use the the **xsl:attribute** element.

1. In the browser, click the **Exercise 2** link in the Quick launch bar to navigate to the page named **Exercise02.aspx**. The HTML content on this page is being generated with an XML Viewer Web Part, which has been preconfigured to transform the XML document named **Exercise02.xml** with the XSLT file **Exercise02.xsl**. However, the items in the list do not contain links yet. Your work in this lab will involve you modifying **Exercise02.xsl** to add the links to each of these sites.



1. In SharePoint Designer, open **Exercise02.xml** and examine the XML content inside. Note that you will not be required to make any changes to this file. However, you should review the names of the XML elements inside because it will be helpful when you are modifying **Exercise02.xsl** in later steps. The following XML snippet shows the basic structure of **Exercise02.xml,** which contains a root element named **HotLinks** and three child elements named **Title**, **URL** and **Comments**.

<HotLinks>

<Link>

<Title>CSS Property Reference</Title>

<URL>http://www.w3.org/TR/CSS21/propidx.html</URL>

<Comments>Document which specifies CSS properties for version 2.1.</Comments>

</Link>

<Link>

<Title>CSS3 Maker</Title>

<URL>http://www.css3maker.com/</URL>

<Comments>Get CSS 3.0 Layouts.</Comments>

</Link>

<Link>

<Title>Color Schemer</Title>

<URL>http://www.colorschemer.com</URL>

<Comments>Site to help you pick a color scheme.</Comments>

</Link>

</HotLinks>

1. In SharePoint Designer, open **Exercise02.xsl** and examine the XSL code inside the **xsl:template** element. As you can see, the XSL code contains an HTML **ul** element with a nested **xsl:for-each** element. The **xsl:for-each** element enumerates through each XML **Link** element and creates an HTML **li** element containing the text value from the XML **Title** element.

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>

<xsl:template match="/">

<h2>Hot Links for SharePoint Web Designers</h2>

<ul style="font-size:10pt">

<xsl:for-each select="HotLinks/Link">

<li>

<xsl:value-of select="Title"/>

</li>

</xsl:for-each>

</ul>

</xsl:template>

</xsl:stylesheet>

1. Now it's time to begin your work on **Exercise02.xsl** to modify the HTML **li** elements to contain links. Add an HTML **a** element inside the HTML **li** element. This **a** element should contain the **target** attribute with a value of **\_blank**. Be sure to position the existing **xsl:value-of** element retrieving the XML **Title** element so that it is nested inside the new HTML **a** element.

<xsl:for-each select="HotLinks/Link">

<li>

<a target="\_blank" >

<xsl:value-of select="Title"/>

</a>

</li>

</xsl:for-each>

1. Now it's time to add the **href** attribute to the HTML **a** element so that it is properly configured with a link. Accomplish this by adding an **xsl:attribute** element with a **name** attribute of **href** inside the HTML **a** element. Inside the **xsl:attribute** element, add a **xsl:value-of** element which retrieves the text content inside the XML **URL** attribute. When you are done, save your changes to **Exercise02.xsl** in SharePoint Designer.

<xsl:for-each select="HotLinks/Link">

<li>

<a target="\_blank" >

<xsl:attribute name="href">

<xsl:value-of select="URL"/>

</xsl:attribute>

<xsl:value-of select="Title"/>

</a>

</li>

</xsl:for-each>

1. Return to **Exercise02.aspx** in the browser and refresh the page. You should be able to see that your changes to the XSLT file has resulted in modifying the **li** elements to contain links. And these links can be used to launch a new session of the browser which navigates to the intended website.



1. As a final step, you are going to take the text value of the XML **Comments** element for each link and use it to produce a tooltip for the link displayed on the page. Below the **xsl:attribute** element for the **href**, add a second **xsl:attribute** element which adds a **title** attribute. Inside the **xsl:attribute** element, add a **xsl:value-of** element which retrieves the text content inside the XML **Comments** element. When you are done, save your changes to **Exercise02.xsl**.

<xsl:for-each select="HotLinks/Link">

<li>

<a target="\_blank" >

<xsl:attribute name="href">

<xsl:value-of select="URL"/>

</xsl:attribute>

<xsl:attribute name="title">

<xsl:value-of select="Comments"/>

</xsl:attribute>

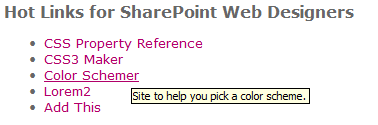
<xsl:value-of select="Title"/>

</a>

</li>

</xsl:for-each>

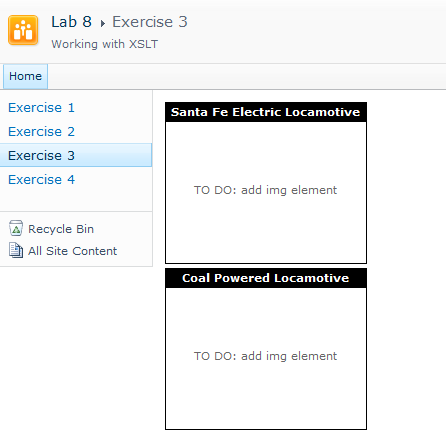
1. Return to **Exercise02.aspx** in the browser and refresh the page. You should be able to see your changes to the XSLT file by hovering the mouse over a link and seeing a tooltip appear.



### Exercise 3: Generating div Elements to Display Images

In this exercise you modify an XSLT file to produce a set of HTML div elements which displays a set of images that are stored in a Photo Library.

1. In the browser, click the **Exercise 3** link in the Quick launch bar to navigate to the page named **Exercise04.aspx**. The HTML content on this page is being generated with an XSLT List View Web Part, which is displaying content from the Photo Library named **ProductPhotos**. Currently, the page displays the title of each photo, but not the images themself.



1. In SharePoint Designer open **Exercise03.xml** and examine the XML content inside. Note that you will not be required to make any changes to this file. However, you should review the names of the XML elements inside because it will be helpful when you are modifying **Exercise03.xsl** in later steps. The following XML snippet shows the basic structure of **Exercise03.xml,** which contains a root element named **ProductPhotos** and an inner **Photo** element for each photo in the photo library. Each **Photo** element contains three child elements named **Name**, **Path** and **Title**. These three child elements contain the values for the photo file name, site-relative path and the photo title respectively.

<ProductPhotos>

<Photo>

<Name>Train1.jpg</Name>

<Path>ProductPhotos/</Path>

<Title>Santa Fe Electric Locamotive</Title>

</Photo>

<Photo>

<Name>Train2.jpg</Name>

<Path>ProductPhotos/</Path>

<Title>Coal Powered Locamotive</Title>

</Photo>

</ProductPhotos>

1. In SharePoint Designer, open **Exercise03.xsl** and examine the XSL code inside the **xsl:template** element. As you can see, the XSL code contains some CSS styles at the top of the **xsl:template** element. There is also an **xsl:for-each** element that generates an HTML div element with a class of **ProductPhoto** for each photo. Inside this div element, the existing XSL code also generates a child div element with a class of **ProductPhotoTitle** that contains the photo Title value. Under the div element, the XSL code created an HTML table element with a single table cell. In the following steps you will modify **Exercise03.xsl** to add an HTML **img** element inside the table cell to display the actual photo image.

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>

<xsl:template match="/">

<style type="text/css">

/\* styles used by XSLT omitted for brevity \*/

</style>

<xsl:for-each select="ProductPhotos/Photo">

<div class="ProductPhoto" >

<div class="ProductPhotoTitle">

<xsl:value-of select="Title"/>

</div>

<table class="ProductPhotoImageTable" >

<tr>

<td>

TO DO: add img element

</td>

</tr>

</table>

</div>

</xsl:for-each>

</xsl:template>

</xsl:stylesheet>

1. Now it's time to begin your work within **Exercise03.xsl** to modify the HTML **td** element to contain an image. Add an HTML **img** element inside the HTML **td** element. This **img** element should contain an inner **xsl:attribute** element to add the HTML **src** attribute which contains a full site-relative path to the image file for the photo. You can create the value for the source attribute by appending the text value of the XML **Path** element together with a text value of the XML **Name** element. Save your changes to **Exercise03.xsl**.

<table class="ProductPhotoImageTable" >

<tr>

<td>

<img>

<xsl:attribute name="src">

<xsl:value-of select="Path" />

<xsl:value-of select="Name" />

</xsl:attribute>

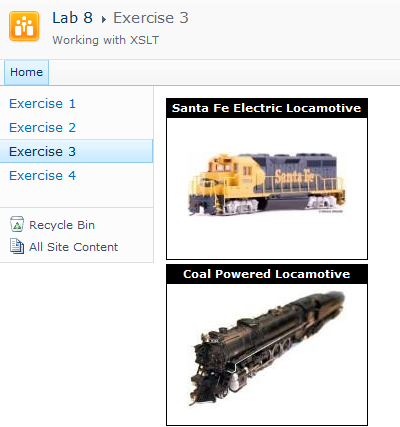
</img>

</td>

</tr>

</table>

1. Return to **Exercise03.aspx** in the browser and refresh the page. You should be able to see that your changes to the XSLT file now result in images being displayed for each photo.



1. The last step in this exercise will involve adding two new CSS styles to **Exercise03.xsl** to change the layout of the top-level div elements. Locate the CSS rule with the selector of **div.ProductPhoto**. Add a new line to change the **display** attribute value from the default value of **block** to a value of **inline-block**. Add a second line to set the **overflow** attribute to a value of **hidden**. Save your changes to **Exercise03.xsl**.

div.ProductPhoto {

margin: 4px; width:200px; height: 160px;

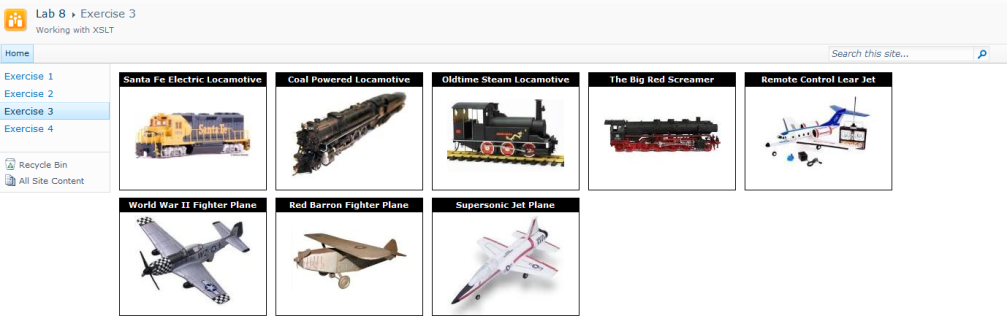
border:solid 1px black; padding: 0px; text-align:center;

display: inline-block;

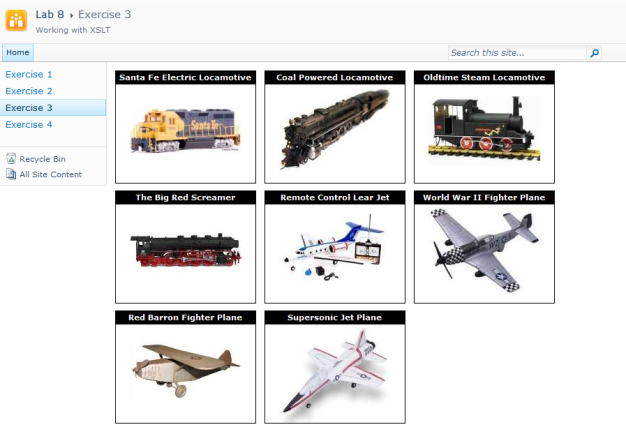
overflow: hidden;

}

1. Return to **Exercise03.aspx** in the browser and refresh the page. You should be able to see that your changes to the XSLT file now result in div elements that float across the page.



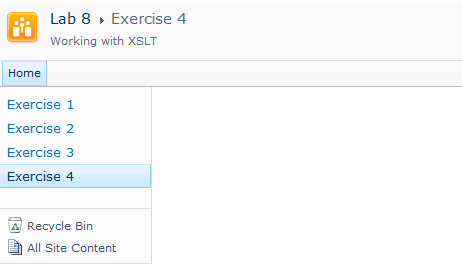
1. Note that the number of div elements displayed per row dynamically changes as you resize the width of the page.



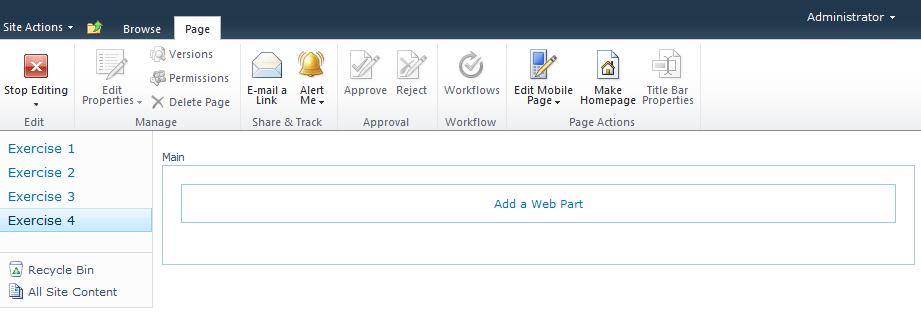
### Exercise 4: Working with the XSTL List View Web Part

In this exercise you add an XSLT List View Web Part to a Web Part Page that displays the contents of the photo library named ProductPhotos on the page. You will then use the SharePoint Designer to customize the output of this Web part by customizing the XSL code SharePoint uses to render the content.

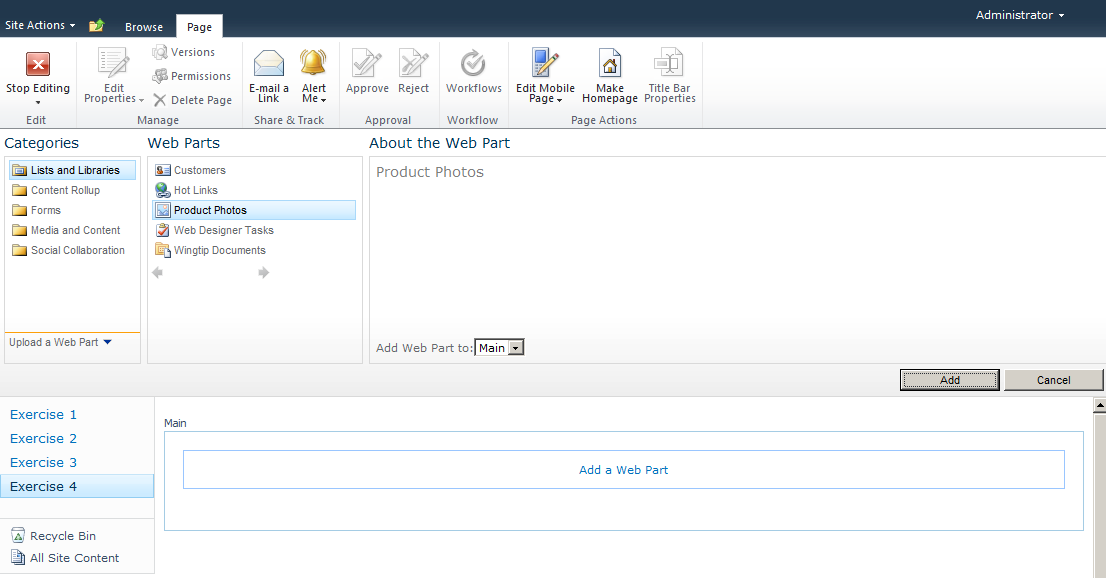
1. In the browser, click the **Exercise 4** link in the Quick launch bar to navigate to an empty Web Part page named **Exercise04.aspx**.



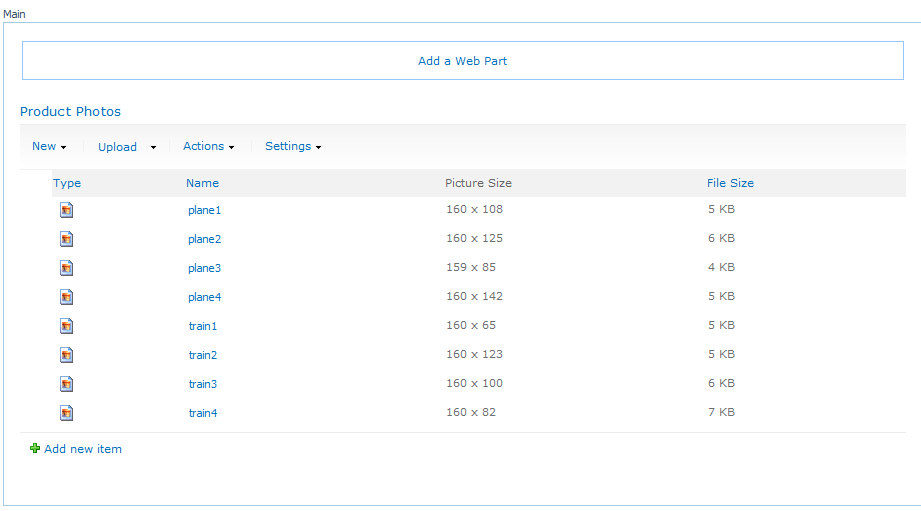
1. Click the **Edit Page** menu command from the **Site Actions** menu to transition the page into Edit Mode. You should see that this empty Web Part page consists of one Web part zone named **Main** that spans the entire content area of the page. Click the **Add a Web Part** link inside the **Main** zone that will display the Web Part adder panel.



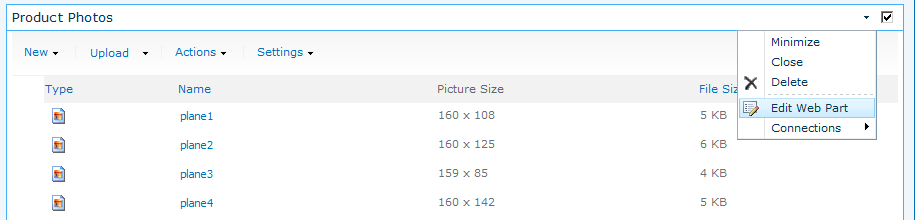
1. The Web Part adder panel should now be showing at the top of the page. Select the **Lists and Libraries** in the **Categories** section on the left-hand side of the Web Part add panel. Next, select the **Product Photos** list in the **Web Parts** section to the right. Add the XSLT List View Web Part to the page by clicking the **Add** button in the bottom right corner of the Web Part adder panel.



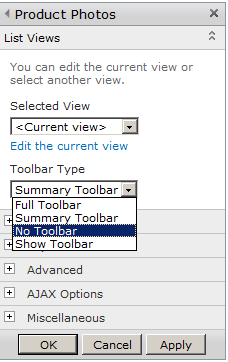
1. After you click the **Add** button you should see an empty XML Web Part. Click the link that reads **open the tool pane** to edit the properties of the new Web Part.



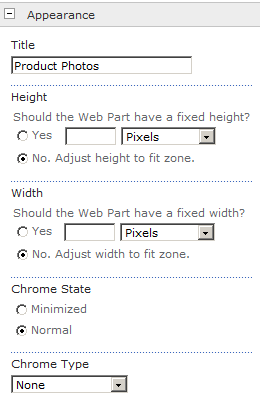
1. In the Web Part menu in the top right-hand corner of the Web Part, click the **Edit Web Part** menu to display the task pane with the editor parts that can be used to customize the Web Part.



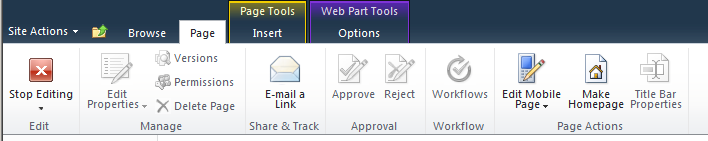
1. In the task pane, modify the **Toolbar Type** property to have the value of **No Toolbar**.



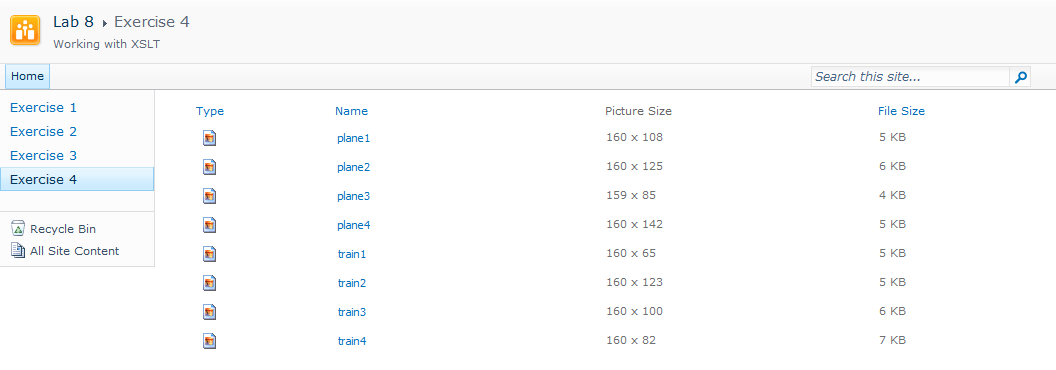
1. In the task pane, expand the **Appearance** section and configure the Chrome Type property to have a value of None. Click the **OK** button at the bottom of the task pane to save your changes and dismiss the task pane.



1. Click the **Stop Editing** button on the left-hand side of the ribbon to leave editing mode and return to display mode so you can see how the Web Part will appear to the users of the site.

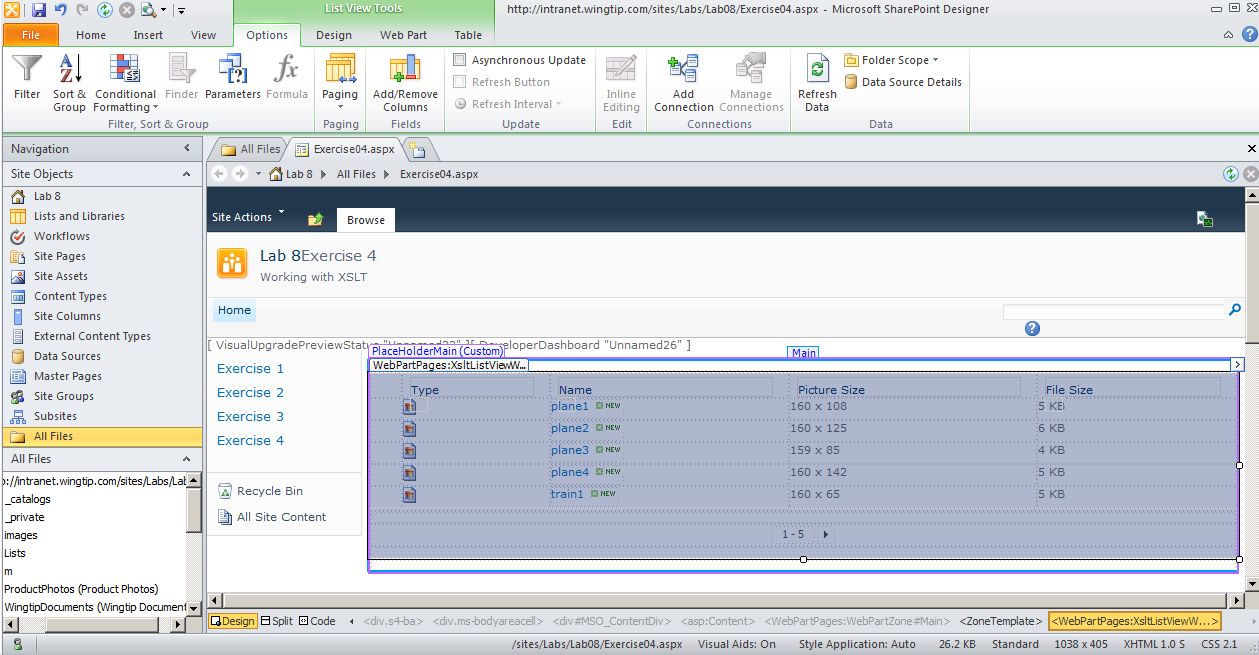


1. Your Web Part should look like the one shown in the following screenshot.

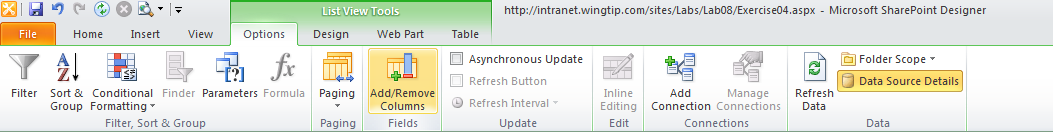


At this point you have made a few customization changes through the browser but the Web Part still does not display the actual images for the photos. Now you will use the SharePoint Designer to customize the XSL code that SharePoint will use to render the Web Part.

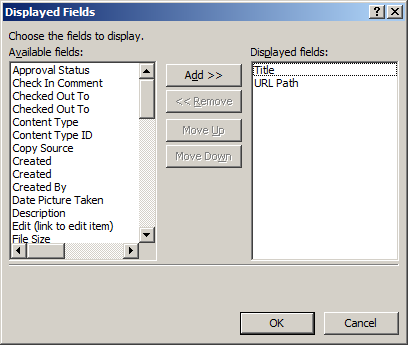
1. In SharePoint Designer open **Exercise04.aspx**. When the page opens, make sure the display mode for the page editor is set for **Design** view as opposed to **Code** view or **Split** view. Select the Web Part.



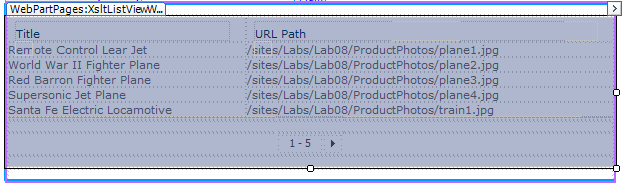
1. When the Web Part is selected in **Design** view, the Ribbon at the top of the page displays context-sensitive tabs that provide extra customization support for Web Parts. Click the **Options** tab and examine the set of controls in this tab. Click the button with the caption of **Add/Remove** Columns.



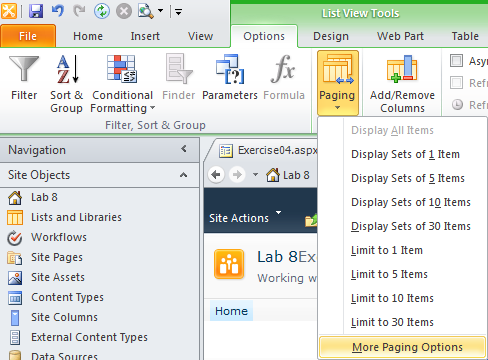
1. Remove all the existing columns (i.e. fields) and add the columns named **Title** and **URL Path**. Click OK to save your changes.



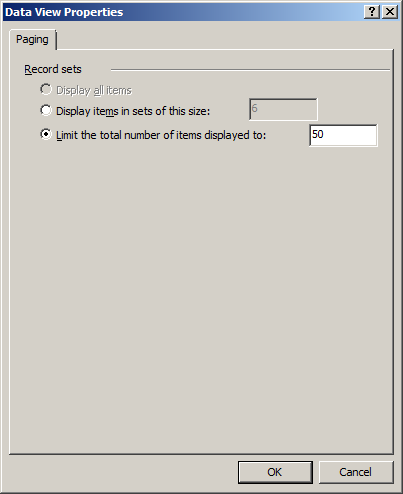
1. Your Web Part should now appear with the two required columns like the one in the following screenshot.



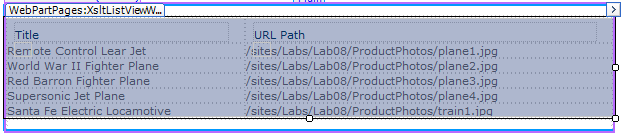
1. In some scenarios, it is valuable for the XSLT List View Web Part to provide automatic paging. However, in this scenario it doesn't make sense to provide paging controls because the number of images will also be relatively small. Drop down the Paging menu in the Options tab and click the **More Paging Options...** menu command.



1. When the **Data View Properties** dialog appear, limit the total number of items for display to **50** and click the **OK** button.



1. Your Web Part should now appear without the paging controls like the one in the following screenshot.



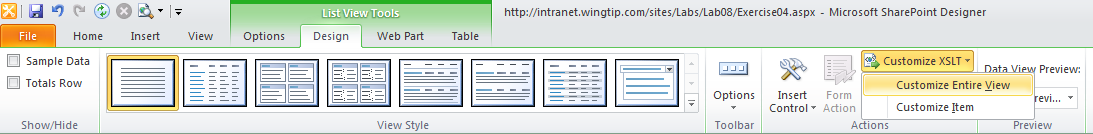
1. Click on the **Design** tab and examine the controls available for modifying a Web Part. There is a gallery of **View Styles** that allow you to change the layout of the Web Part display. Click the third view style from the left which has a tooltip of **Boxed, no labels**.

,

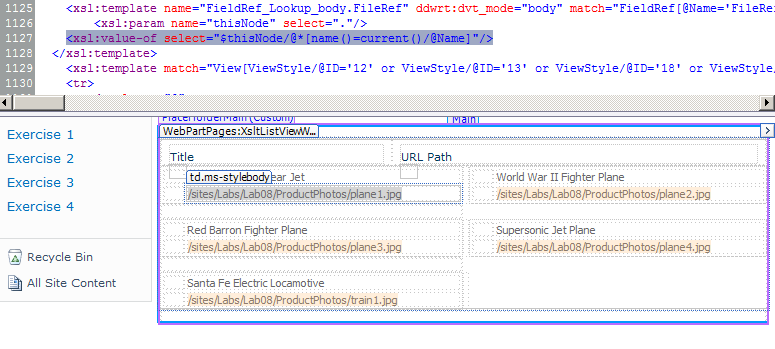
1. Your Web Part should now appear with a table like the one in the following screenshot.



1. Now it is time to add an HTML **img** element to display the actual .jpg file for each photo. In order to do this you will be required to add customized XSL code behind the Web Part. Begin by locating the Customize XSLT drop down menu on the right-hand side of the Design tab. Drop down this menu and select the **Customize Entire View** menu command.



1. Now move the page in **Split** view so you can see the code in Exercise04.aspx in addition to the visual layout of the page. If you scroll down, you should be able to see quite a bit of XSL code at the bottom of the page. Locate the **xsl:template** element with a name of **FieldRef\_Lookup\_body.FileRef** which should be around line 1125. Note that **FileRef** is the underlying name for the **Name** column.



1. Place your cursor in the Code view window inside the **xsl:template** element with the name of **FieldRef\_Lookup\_body.FileRef**.

<xsl:template name="FieldRef\_Lookup\_body.FileRef" ddwrt:dvt\_mode="body" ... >

<xsl:param name="thisNode" select="."/>

<xsl:value-of select="$thisNode/@\*[name()=current()/@Name]"/>

</xsl:template>

1. Add a new line after the xsl:param element. Inside the new line, add an HTML **img** attribute. This **img** element should contain an inner **xsl:attribute** element to add the HTML **src** attribute which contains the path to the image file for the photo. You can use the existing xsl:value-of element to provide the value for the photo image file.

<xsl:template name="FieldRef\_Lookup\_body.FileRef" ddwrt:dvt\_mode="body" ... >

<xsl:param name="thisNode" select="."/>

<img>

<xsl:attribute name="src">

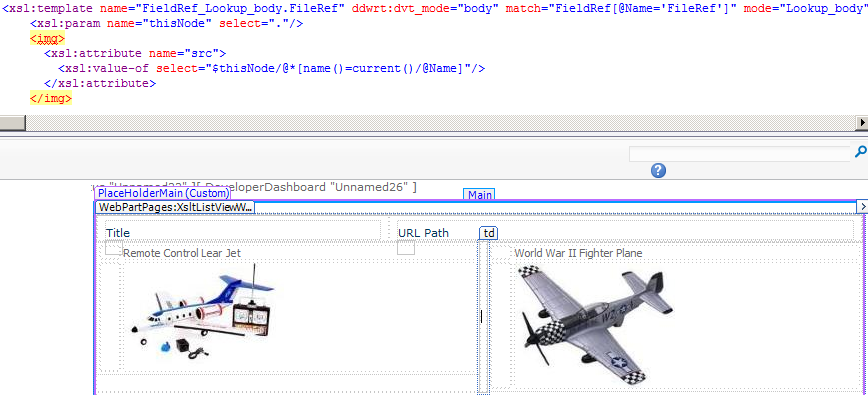
<xsl:value-of select="$thisNode/@\*[name()=current()/@Name]"/>

</xsl:attribute>

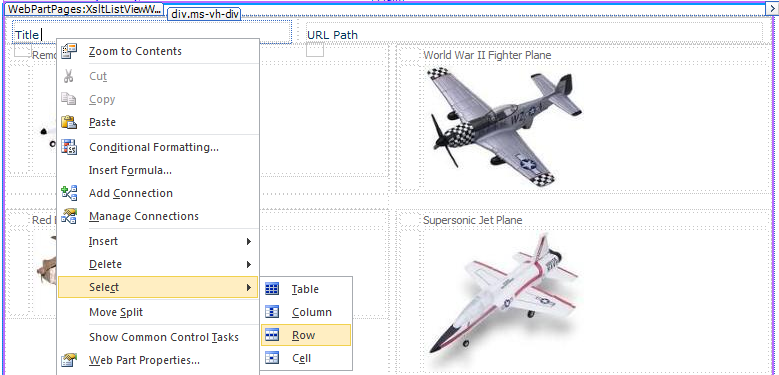
</img>

</xsl:template>

1. After you have made this edit in the Code View window, press the {F5} key to push your changes into the Design View window. Your Web Part should now be showing the photo images.



1. The final step will be to remove the headings from the top of the Web Part that displays **Title** and **URL Path**. In **Design** view, select the row with these two headings. Right click the Title heading.



1. Press the Delete key to remove the heading row.
2. In SharePoint Designer, save your changes to Exercise04.aspx.
3. Return to browser and refresh Exercise04.aspx. The page should now look like this:

