In this lab you will create an XML application. It will consist of:

1. An XML data file
2. An XML style sheet that controls which data items are selected and how they appear on the web page
3. A web page to display the XML data

Development is done in two stages:

1. First the files are created and tested locally. This allows the product to be developed without impacting the outside world.
2. Then the completed and tested product is uploaded to a server (and retested) so the whole world has access to it.

**Part 1**. Create an XML data file.

The file will need to be accessible wherever you use MS Visual Studio. Unfortunately MS and Novell don’t play well together, so using the Novell H: network drive is not an option. Also, the C: drive on a lab machine is a poor choice for two reasons: 1) You may not get back onto VS using the same machine and 2) The C: drives are (in theory) cleaned every night of user files. There are two good choices of where to keep your development files: 1) on a “thumb” drive or 2) on a MS 2003 server directory. A thumb drive is the best choice, but using a MS server directory is less expensive - your choice.

Also note that there are several equivalent ways to do the same process in Visual Studio. The directions below are not the only way to finish this lab, but do work.

1. Decide where to keep your data:
	1. MS network drive: Open a Command Prompt and type the following command: **net use** and check whether there is a **V** drive on you lab machine.
		1. If there is not, type the following command:
		**net use v: \\coit-ts2003.uncc.edu\<yourusername>**
		Make sure you insert the space between the v: and \\ in the previous command and replace <yourusername> with your personal id. Enter your user name when prompted. Enter the last four digits of your student ID as your password. Go to Step 2.
		2. If there is a V: drive and it is not yours you need to remove it with the following command:
		**net use v: /d**then do step 1.a.i above.
	2. Thumb drive: plug it in and note which drive letter gets assigned to it. Go to Step 2.
2. Start Microsoft Visual Studio 2005 from the Start Menu, not from the Novell Launcher. If this is the first time you are using it, select *general developing environment*.
	1. Click **File** 🡪**New Website**.
	2. In the New Website window**,** select **ASP.NET Web Service** in the *Templates* pane. In the *Location* there will be a default file name already listed. On the lab machines it is most likely: **H:\My Documents\Visual Studio 2005\WebSites\WebSite1.** Change it to **V:\XML\_App.** Use V: if you are on the network drive. Otherwise use the drive letter for your thumb drive. You can use the *Browse...* button to find the correct directory to put the data in. Don’t forget to make the “tail end” directory name **XML\_App**. This will make life easier later on. If the directory doesn’t exist it will ask if you want to create it. Check your spelling before clicking **Yes**. Click **OK**.
	3. A bunch of code should pop up.
3. Close the *Default* web form. Right click **V:\XML\_App** in the Solution Explorer window. Click **Add New Item…**. In the *Add New Item* window choose **XML File** from the Templates. Change the default name to **XMLFile.xml**. Click **Add**.
4. A new view should be created and displayed. Modify the contents of the ***XMLFile.xml*** so that it matches the following data.
**Note:** **VB** has text assist features. Notice that when you finish a start tag, **VB** will usually auto-create a matching end tag for you. This cuts down significantly on the amount of typing you will need to do and also reduce typing errors. Also when you hit Enter between tag pairs **VB** will also auto-indent the code. **VB** also monitors for syntax errors and will red underscore errors it sees.

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

**<city>**

 **<street>**

 **<name>Main Street</name>**

 **<first\_address>123</first\_address>**

 **<last\_address>1210</last\_address>**

 **<length>5 miles</length>**

 **</street>**

 **<street>**

 **<name>Oak Avenue</name>**

 **<first\_address>201</first\_address>**

 **<last\_address>1534</last\_address>**

 **<length>7 miles</length>**

 **</street>**

 **<street>**

 **<name>Easy Street</name>**

 **<first\_address>564</first\_address>**

 **<last\_address>2987</last\_address>**

 **<length>12 miles</length>**

 **</street>**

**</city>**

1. Add a 4th street named after yourself with appropriate values in the other three labels. Click **File 🡪Save All**.

**Part 2**. In this exercise you will create an XML style sheet that will be used to display the data in the **XMLFile1** file. Note that when you enter a starting tag, VS.NET automatically supplies the proper ending tag. Ensure that the data you enter goes between the correct starting and ending tags.

1. If you are not already in Visual Studio .NET, start it and open the **XML\_App** project.
2. Right-click **XML\_App** in the Solution Explorer Window, then click **Add New Item**. Under *Templates*, click **Style Sheet**, in the *Name:* box type **cities.xsl**, then click **Add**.
3. Delete **body { }**
4. Enter the following data exactly as shown:

**<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">**VB may or may not create the matching end tag for you, so go down a few lines and, if necessary, type

**</xsl:stylesheet>**

1. Between the stylesheet tags enter the following data exactly as shown:

**<xsl:template match=”/”>**VB should create a matching end tag for you, if not add one.

1. Between the template tags enter the following data exactly as shown. This code indicates that for each **street** the **name**, **first\_address**, **last\_address**, and **length** elements will be processed.

**<xsl:for-each select="//street">**

 **<xsl:apply-templates select="name" />**

 **<xsl:apply-templates select="first\_address" />**

 **<xsl:apply-templates select="last\_address" />**

 **<xsl:apply-templates select="length" />**

**</xsl:for-each>**

1. This completes the main template for the street node.
2. Below the **</xsl:template>** tag enter the following code exactly as shown. This defines the formatting of the various elements in the browser window.

 **<xsl:template match="name">**

 **<hr size="1" width="100%" color="#004040">**

 **<b>Street </b><xsl:value-of select="." />**

 **</hr>**

 **</xsl:template>**

 **<xsl:template match="first\_address">**

 **<b> starts at </b><xsl:value-of select="." />**

 **</xsl:template>**

 **<xsl:template match="last\_address">**

 **<b>, ends at </b><xsl:value-of select="." />**

 **</xsl:template>**

 **<xsl:template match="length">**

 **<b> and is </b><xsl:value-of select="." /> miles long.<br/><br/>**

 **</xsl:template>**

Note the </hr> in the first template, HTML does not need it but it makes VS happy.

1. Click **Save All**. Click **Build** then click **Build Web Site**. (If you’re getting build errors, make sure the directory you’re saving your project to isn’t the **H**: drive, this will cause build errors.)

**Part 3**. In this exercise you will use the XML control to create a web page that displays street information.

1. If you are not already in Visual Studio .NET, start it and open the **XML\_App** project.
2. Right-click **XML\_App** in the Solution Explorer Window, point to **Add New Item**, and click **Web Form**. In the name text box enter **liststreets.aspx**, and click **Add**.
3. Ensure the panel is showing. If not click the *Design* button at the bottom. From the *Toolbox*, drag the **Panel** control from the **Standard Group** to the page. Position the Panel control close to the left side of the grid. Drag the lower right handle of the panel to enlarge it so it takes up almost the entire width of the grid.
4. From the Toolbox drag the **XML** control from the **Standard Group** and drop it in the **Panel** control.
5. While the XML control is still selected, in the *Properties* window change the *DocumentSource* property to **XMLFile.xml**.
	1. Click on *DocumentSource*, then click on the ellipses button (…). From *Contents of folder* select XMLFile.xml and click **OK**.
	2. Change the *TransformSource* property to **cities.xsl** using a similar procedure as you did to change the *DocumentSource* property.
6. Click **Save All**. Click **Build** then **Build Web Site**.
7. Right-click the page and then click **View in Browser**.
	1. There is a small error in the resulting web page. Identify and fix that error.

**Part 4**. Up to this point we have created and tested an application that runs on the local machine only. In this final part you will upload the files you have created to a MS IIS server and test your web form from the Internet. FTP will be used to move the files from your development environment to the MS IIS (Web) Server

The following steps will upload your files to the IIS server so they may be viewed by the outside world:

1. If you are not already in Visual Studio .NET, start it and open the **XML\_App** project.
2. On the action bar select: **Website** 🡪 **Copy Website**… 🡪 **Connect**
3. On the *Open Web Site* window select *FTP Site*
4. Server: ftp://coit-ts2003.uncc.edu
5. Directory:
6. Check Passive Mode
7. Uncheck Anonymous and enter your Username and Password
8. Upload your file(s).
	1. Select Source Web Site File(s)
	2. Click 🡪
9. Close Visual Studio .NET and return to your normal Desktop by logging off the Visual Studio .NET server.
10. Launch a browser and enter the following URL (replacing *username* with your user name) to view your web form:

**http://coit-ts2003.uncc.edu/*username*/liststreets.aspx**

1. Try browsing again from another workstation to ensure it is truly on the network and has not retrieved the data from the local copy somehow.