

# Building a Web Site with Visual Studio .NET

---

Build a web application consisting of multiple projects:

1. Web Site
2. Web Service
3. Standalone Library (optional, to be discussed)

**Note:** This lab will be graded as Assignment #2 and is worth 15 points.

You will have 2 lab periods available to work on this assignment. It is due by the end of the last class.

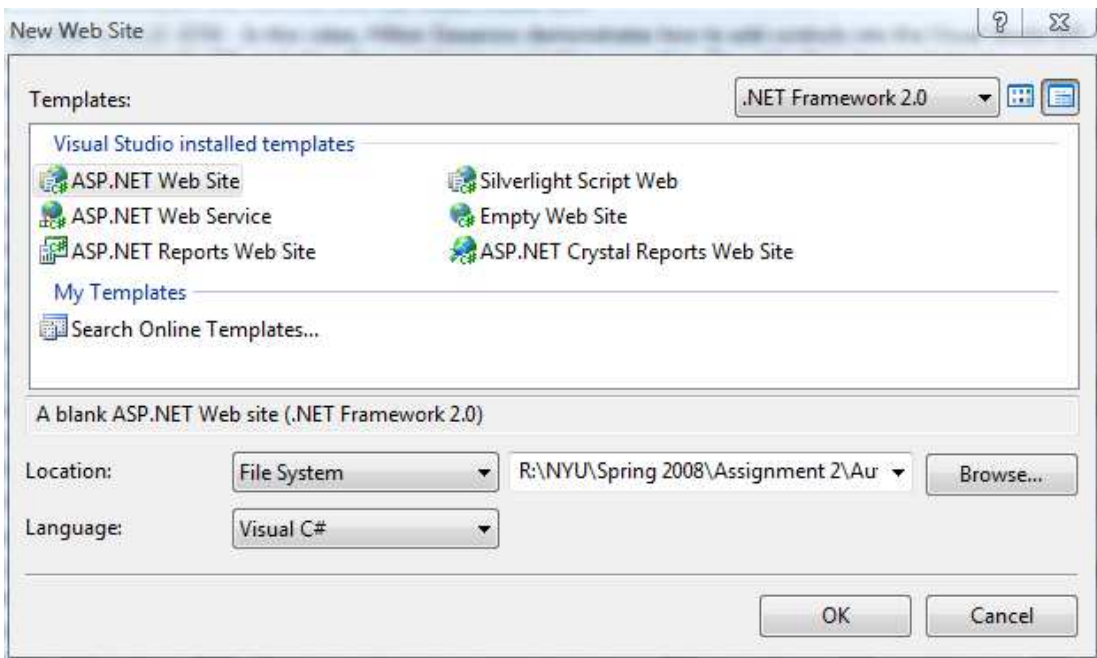
**Get Creative:** Creativity counts! Get up to 5 points for making your site extra special (these points are totally subjective based on perceived effort and up to the instructor's discretion).

## Exercise 1

### Building a Web Site

#### Scenario

Your employer wants you to create a web site that manages the business of its used car sales lot.

Tasks	Detailed Steps
1. Setting up the lab	<p>a. Open Visual Studio and create a C# web site named AutoSales.Web.UI. <b>Note:</b> For convenience, you should create the project on your flash drive.</p>  <p>b. Notice that the web site project model creates a page named default.aspx. This is the default page that is served if your site is addressed with an URL that does not specify a document.</p> <p>c. Create a reference to the autosales library assembly. This is where the Vehicle and ColorEnum types live along with the static helper class VehicleDB.</p>

<p><b>2.</b> Create the nav bar</p>	<p>a. Starting with 2.0, ASP.NET supports Master-Page templates. This allows you to abstract away common elements of a page. To your project, add a Master Page named MasterPage.Master.</p> <p>b. Notice that the MasterPage is an webform that defines the common HTML tags such as &lt;html&gt;, &lt;head&gt;, &lt;body&gt;, &lt;form&gt;.</p> <p>c. Notice that this page has an ASP.NET ContentPlaceholder control. This is where other pages are rendered.</p> <p>d. Add two more web forms named “AddVehicle” &amp; “ListInventory”. Be sure to check the “Select master page” option and choose MasterPage.Master when prompted.</p> <p>e. Initially, you are viewing the source of a web page. Here you can write HTML that affects the visual rendering of the page. Let’s do some RAD work using the familiar drag-and-drop approach.</p> <p>f. Notice that all open documents are listed as tabs along the top of Visual Studio. Click the MasterPage.master tab then click the Design tab at the bottom.</p> <p>g. Add a global header. Drag a label onto the form. Set its Text property to something meaningful such as “Student’s Auto Sales”. Set its font to X-Large, bold.</p> <p><b>Note:</b> Unlike winform applications, you cannot freely place controls according to the X-Y coordinate system. You must insert a carriage return (&lt;br/&gt; tag) to move down the page. If you require horizontal positioning, use a table or div tags + CSS style.</p> <p>h. Switch to Source view and notice that VisualStudio has created an ASP.NET label control and set its properties:  <code>&lt;asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="X-Large" Text="Student's Auto Sales"/&gt;</code></p> <p>i. Switch back to Design view and add a carriage return after this label.</p> <p>j. Add a table by selecting “Layout -&gt; Insert table” from the menu (or just “Table” if using visual studio 2008).</p> <p>k. Create a table that has 1 row and 3 columns.</p> <p>l. Drag a byperlink control into the first cell and set its Text property to “Home”. Set the NavigateUrl property to “Default.aspx”</p> <p><b>Note:</b> use the ellipse button (...) to set this property using a designer window.</p> <p>m. Drag a hyperlink control into the second cell and set its Text property to “Add Vehicle”. Set the NavigateUrl property to “AddVehicle.aspx”</p> <p>n. Drag a hyperlink control into the third cell and set its Text property to “List Inventory”. Set the NavigateUrl property to “ListInventory.aspx”.</p> <p>o. Below the navigation table, had a horizontal rule HTML control from the HTML section of the toolbox.</p>
-------------------------------------	--

**3. Design the AddVehicle page**

- a. Click on the AddVehicle form. Notice in Source view that this page does not have <html> or <body> tags. This is because it uses a master page template to define the common HTML. We will create our code within a Content placeholder control.
- b. Switch to design view and add a label. This will be our page header, set its text property to “Add Vehicle” and set its font properties to render big and bold.
- c. Next insert a table of 6 rows x 3 columns to layout the data collection.

- d. Layout the user input controls. Be sure to give each textbox and the Save button meaningful names as we'll reference them later.

**4. Add page validation**

- a. Drag a RequiredFieldValidator control into the last column. Set its ControlToValidate property equal to the vehicle make textbox control (txtMake). Set its Text property to an astrik (\*) and its ErrorMessage property to “Make is required!”.
- b. Repeat for each of the other inputs, setting each property appropriately.
- c. After the table, insert a ValidationSummary control. This will provide a summary of any errors that occur on a page.

**5. Wire up the Save button**

- a. Double-click on the save button. Visual Studio will take you to the code window and automatically create the save button's click event handler. This is where you add the code to create a vehicle object and save it to the DB by calling the Save() method of VehicleDB class.

<p>6. Design the ListInventory page</p>	<ul style="list-style-type: none"> <li>a. Click the ListInventory.aspx page.</li> <li>b. Switch to design view and add a header label. Set its text property to "Inventory List". Set its font weight and size appropriately.</li> <li>c. Place a GridView control on the page and name it gridInventory.</li> <li>d. Create a Page_Load event handler. Here you will assign gridInventory's datasource to the results of the static helper method VehicleDB.GetInventory(). In ASP.NET you must always data-bind a control by setting its DataSource property and then calling its DataBind() method.</li> </ul>
<p>7. Design the Default page</p>	<ul style="list-style-type: none"> <li>a. The default.aspx page will serve as a landing page. Here you can tell users about current discounts and marketing campaigns. Get creative, this page doesn't actually do anything. If your out of ideas, just throw in some Lorem Ipsum (<a href="http://www.lipsum.com">http://www.lipsum.com</a>).</li> </ul>
<p>8. Store the connection string in a configuration file</p>	<ul style="list-style-type: none"> <li>a. Add a Web Configuration File to your web site project. Name this file Web.config.</li> <li>b. Replace the <code>&lt;connectionStrings/&gt;</code> element with this: <pre> &lt;connectionStrings&gt;   &lt;add name="autosales" connectionString="Data Source= DataDirectory AutoSales.sdf" providerName="System.Data.SqlClient" /&gt; &lt;/connectionStrings&gt; </pre> </li> </ul> <p><b>Note:</b> Storing your database connection string in a configuration file is a better alternative to hard coding it into your project. If this were a winforms application, the configuration file would be called app.config.</p>