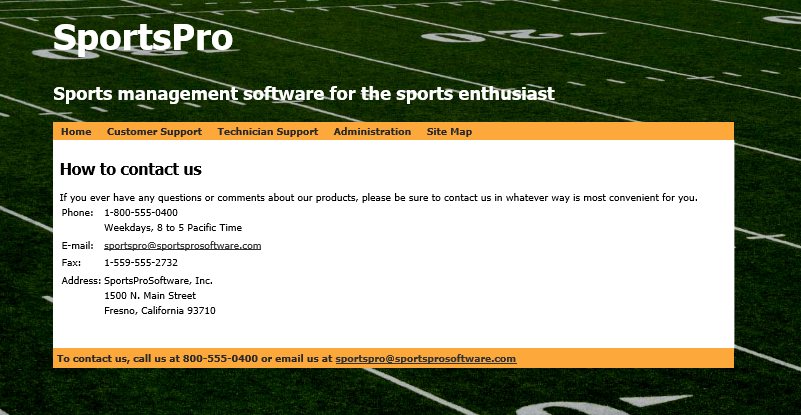
Master page and content pages

* Make a copy of your SportsPro1, name it SportsPro2—LastName. Fix any errors.
* Create a Master Page for the SportsPro application. Create a new Content Page named ContactUs.aspx that displays contact information. Also, replace all the pages you created in Case 1 with Content Pages. Your background image should appear on all pages, so include it in the master page. (*Required reading: Murach Chapter 9*)

The design of the *Master Page* and the *Contact Us* Content Page



Operation

* When the user clicks one of the links in the menu at the top of the page, the appropriate page is displayed.
* When the user clicks the E-mail link under “How to contact us” or at the bottom of the page, the user’s email client starts a new email for the specified address.

Specifications

* The Master Page consists of the two titles in a banner section, a **Menu** control with three items in a top\_nav section, and the contact information in a footer section. The footer should only appear on the Contact page. You can use any fonts, colors, and sizes you want for these elements **using a CSS style page** (be sure it is in a separate folder).
* To create the two email links, you can use a hyperlink with its NavigateUrl attribute set to “<mailto:sportspro@sportsprosoftware.com.>”
* Modify the design so that all pages are displayed centered horizontally on the page with empty space of the left and right sides, use either CSS or Bootstrap for creating the spacing. Add a CSS3 *box-shadow*. Be sure it works on IE, Chrome and Firefox.

Customer survey

Add a new page named **CustomerSurvey.aspx.** Set this page as the “Start Page”. This page surveys customers regarding recent incidents. Add a new page named **SurveyComplete.aspx** that displays a message when the survey is submitted. (*Required reading: Murach Chapters 6 and 7*)

The design of the *Customer Survey* and *Survey Complete* pages

Operation

* When the **Customer Survey** page is first displayed, all of the controls on this page, except for the Customer ID text box and the **Get** **Incidents** button, are disabled.
* When the user enters a customer ID and clicks the **Get Incidents** button, any closed incidents for that customer are displayed in the listbox and all of the controls on the page are enabled. If there aren’t any closed incidents, the controls are not enabled and an appropriate message is displayed below the customer ID.
* To complete the survey, the user selects an incident in the listbox, makes the appropriate selections and entries, and then clicks the **Submit** button, then the **Survey Complete** page is displayed with an appropriate message that depends on whether or not the user has asked to be contacted. The user can close this page or complete another survey by clicking the **Return to Survey** button.

Specifications

* Create a new class named **Incident** that consists of one public property for each field in the Incidents table and a method named CustomerIncidentDisplay that formats an incident for display on the Customer Survey page.
* Create another new class named **Survey** that consists of the following properties that store the data from a survey:

public int CustomerID { get; set; }  
public int IncidentID { get; set; }  
public int ResponseTime { get; set; }  
public int TechEfficiency { get; set; }  
public int Resolution { get; set; }  
public string Comments { get; set; }  
public bool Contact { get; set; }  
public string ContactBy { get; set; }

* Use Labels within the survey where needed so you can enable and disable them as required.
* Use RadioButtonLists to implement the ratings for the three categories in the survey. The Text properties for the list items in these lists should be set as shown on the page, and the Value returned by selecting a radio button should be 1, 2, 3, or 4.
* Include a RadioButton group for the “Contact by” radio buttons on the bottom of the page.
* Use Required Field Validators for the Customer ID text box and the Incidents list box, and use a Compare Validator for the Customer ID text box that checks for an integer value. In addition, use validation groups so that the validators for the Customer ID text box are executed only when the user clicks the Get Incidents button, and the validator for the Incidents listbox is executed only when the user clicks the Submit button.
* When the **Customer Survey** page is first displayed, set the focus to the customer ID text box. Then, if the user clicks the Get Incidents button and there are incidents for the specified customer, move the focus to the Incidents listbox.
* To get the data that’s displayed in the Incidents list box, create a **Sql Server Data Source** that retrieves all the rows and all the columns except the Description column from the Incidents table, sorted by the DateClosed column. Use a **DataView** to filter the rows so that only the closed incidents for the specified customer are displayed. To do that, you’ll need to set the RowFilter property of the DataView to a compound condition where the CustomerID column is equal to the customer ID the user enters **and** the DateClosed column is not null. Null DateClosed values will cause errors. You will probably want to use **foreach** to loop through the rows returned.
* Research the **ListItem** class, notice that it can have 0, 1, 2, or 3 parameters (i.e. method overloading). When populating the Incidents ListBox, create list items using two parameters. Set the first parameter (string text) to the value of the CustomerIncidentDisplay method of the appropriate incident. Set the second parameter (string value) to the incident ID. In addition, the Text of the first item in the list box should be set as shown on the **Customer Survey** page above, and the second parameter of this item should be set to “None” or “NA”. An appropriate error message should appear if the survey is submitted and no **Incident** is selected, as recognized by returning the value “None” or “NA”.
* When the survey is submitted,
* The application should create a Survey object and set its properties appropriately,
* Use a Session variable to pass the survey results between pages.
* The Survey Complete page should retrieve the Session variable value to determine the “Contact” message that is displayed on the page and to display the survey results.

Site navigation

Add addition navigation to the SportsPro application. Include a page that contains a **TreeView** of the site. (*Required reading: Murach Chapter 10*)

The design of the *Map* page



Specifications

* Add several new forms to the application. You should give these forms the following names: Default.aspx, CustomerSupport.aspx, ProductRegistration.aspx, TechnicianSupport.aspx, CustomerIncidentDisplay.aspx, IncidentUpdate.aspx, Administration.aspx, ProductMaintenance.aspx, CustomerMaintenance.aspx, TechnicianMaintenance.aspx, IncidentDisplay.aspx, TechnicianIncidentSummary.aspx, and Map.aspx. Include some text on these pages that indicates the name of the page. The only page you will actually implement in this part of the project is the Map page. The other pages will be used in later cases.
* After you create the necessary pages, create a **Sitemap** file with the structure shown in the **TreeView** above. Add a **TreeView** control to the Map page. Choose an auto format for the TreeView.
* Add a **Menu** control that will navigate to the four items as shown in the page immediately above.

# Case Submission

* If the project does not run, the highest score you can get is 50 pts.
* Create a .zip of your project, it should be named SportsPro2--LastName. Upload your file in Blackboard.