Essential ASP.NET Web Forms Development Chapter Assignments by Robert E. Beasley

Chapter 1

1. Install Visual Studio.
2. Start a new project called *Lastname.Firstname.Chapter01*.
3. Look around the Visual Studio environment and become familiar with it.
4. Tear off the Solution Explorer and move it around the page. Dock it at different places in the environment. Then, dock it in its original location.

Chapter 2

1. Start a new project called *Lastname.Firstname.Chapter02*.
2. Add a Page class to the project called *DisplayBiography*.
3. Add a title to your page that says *Biography*.
4. Add a single paragraph to the body of your page that includes your name, where you are from, and something about you that most people probably don't know.
5. Test the page and make any necessary corrections.

Chapter 3

Part 1

1. Start a new project called *Lastname.Firstname.Chapter03*.
2. Add a Page class to the project called *CopyInformationPart1*.
3. Add seven Label controls that display *First Name*, *Middle Initial*, *Last Name*, *Address*, *City*, *State*, and *Zip Code*, respectively. Arrange these control vertically on the page.
4. To the right of each Label control, add a corresponding TextBox control.
5. To the right of each TextBox control, add a corresponding Label control.
6. Add a Button control that says *Copy*. When the button is clicked, copy the information in the text boxes to their corresponding labels on the right.
7. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *CopyInformationPart2*.
2. Add a Table control with 7 rows and 3 columns. Change the background color of each row so they alternate between white and light gray. You may need to use a search engine to find out how to set the background color of a control in ASP.Net.
3. In the first column, add a Label control to each row that displays *First Name*, *Middle Initial*, *Last Name*, *Address*, *City*, *State*, and *Zip Code*, respectively.
4. In the second column, add a corresponding TextBox control for each Label control.
5. In the third column, add a corresponding Label control for each TextBox control.
6. Add a Button control that says *Copy*. When the button is clicked, copy the information in the text boxes to the corresponding labels in the third column of the table.
7. Add a Button control that says *Change Color*. When the button is clicked, change the background color of the table to yellow.
8. Test the page and make any necessary corrections.

Chapter 4

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

1. Start a new project called *Lastname.Firstname.Chapter04*.
2. Add a Page class to the project called *PerformServerControlOperations*.
3. Add a Calendar control in column one of the table. Add an associated Label control in column two of the table. Experiment with the look of this calendar (e.g., change its background color, change its header text). Default the selected date to today's date. When a date is clicked, display the selected date in the label.
4. Add three CheckBox controls in column one of the table. Add an associated Label control in column two of the table. The check boxes should say *Red*, *White*, and *Blue*, respectively. When any one of the check boxes is checked or unchecked, display *all* of the *checked* items in the label.
5. Add two RadioButton controls in column one of the table. Add an associated Label control in column two of the table. These radio buttons should say *Check All* and *Uncheck All*, respectively. When either of the radio buttons is clicked, either check *all* or uncheck *all* of the check boxes in the previous problem and display *Checked!* or *Unchecked!* in the label.
6. Add a FileUpload control and a Button control that says *Upload* in column one of the table. Add an associated Label control in column two of the table. Add a folder to the project called *Files*. When the *Upload* button is clicked, upload the file selected using the *Browse…* button and display a confirmation message in the label. If a file hasn't been selected, don't attempt the upload and display an appropriate message in the label.
7. Add a HyperLink control in column one of the table. This link should say *Go to Facebook*. When the link is clicked, navigate to Facebook.com. Open the *Facebook* page in a new browser tab.
8. Add an Image control in column one of the table. Display a picture of yourself in the Image control. Make sure that the image is 100 pixels high. Adjust the width of the image so that the picture is not distorted.
9. Add an ImageButton control in column one of the table. Add an associated Label control in column two of the table. Display a picture of yourself in the Image control. Make sure that the image is 100 pixels high. Adjust the width of the image so that the picture is not distorted. When the image button is clicked, display your full name in the label.
10. Add an ImageMap control in column one of the table. Add an associated Label control in column two of the table. Using a graphics program like *Microsoft Paint*, create an image with four quadrants. Make each quadrant a different color. When a quadrant within the image map is clicked, display the name of the color of the quadrant in the label.
11. Add a LinkButton control that says *Copy* and a TextBox control in column one of the table. Add an associated Label control in column two of the table. When the link button is clicked, copy the information in the text box to the label.
12. Add a DropDownList control in column one of the table. Add an associated Label control in column two of the table. The drop down list should display three options: *Red*, *White*, and *Blue*. The *values* for these options should be *R*, *W*, and *B*, respectively. When an option is selected from the drop down list, display "The selected index is: _, the selected value is: _, and the selected text is: _." in the label, where each blank is replaced with the appropriate value.

13. Repeat Step 12 but use a ListBox control.
14. Add three Button controls that say *One*, *Two*, and *Three*, respectively in column one of the table. Add three Panel controls in column two of the table. On the first panel, add a Label control that says *This is panel one*. Set the grouping text of this panel to *One*. On the second panel, add a Label control that says *This is panel two*. Set the grouping text of this panel to *Two*. On the third panel, add a Label control that says *This is panel three*. Set the grouping text of this panel to *Three*. When button one is clicked, display the first panel. When button two is clicked, display the second panel. When button three is clicked, display the third panel.
15. Test the page and make any necessary corrections.

Chapter 5

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

1. Start a new project called *Lastname.Firstname.Chapter05*.
2. Add a Page class to the project called *EnterProduct*.
3. Add a 10 row by 2 column table.
4. In the first column, add 10 Label controls. From top to bottom, these labels should say *Category*, *Supplier*, *Product*, *Description*, *Image*, *Price*, *Number in Stock*, *Number on Order*, *Expected Delivery Date*, and *Reorder Level*.
5. In the second column, from top to bottom, add a category DropDownList control (with these options: <*Select a category*> [note: this is the default], *basketballs*, *footballs*, *volleyballs*), supplier DropDownList control (with these options: <*Select a supplier*> [note: this is the default], *Nike*, *Head*, *Wilson*), product TextBox control, description TextBox control (multiline), image TextBox control, price TextBox control, number in stock TextBox control, number on order TextBox control, expected delivery date Calendar control, and reorder level TextBox control.
6. Add validator controls (with appropriate error messages) to the input controls on the page to ensure that the end user
    a. Selects a category (i.e., something other than the default)
    b. Selects a supplier (i.e., something other than the default)
    c. Enters a product (must be four letters followed by four numbers)
    d. Enters a description
    e. Enters an image file name (.jpg or .png only) (use a CustomValidator control for this)
    f. Enters a price (greater than $0.00)
    g. Enters a number in stock (between 0 and 100, inclusive)
    h. Enters a number on order (between 0 and 100, inclusive)
    i. Selects an expected delivery date greater than today's date (place the selected date into a text box and then validate the selected date in the text box)
    j. Enters a reorder level (between 0 and 100, inclusive)
7. Add a Button control that says *Save* below the table. When the button is clicked, validate the data in all of the input controls.
8. Add a ValidationSummary control below the button. Set the HeaderText property of the control to "The following errors have occurred on this page. Please correct them." Experiment with different DisplayMode properties. Make sure that an asterisk (*) is displayed next to any invalid data.

9. Change the ValidationSummary control so that the error messages are displayed in a message box instead of on the page itself.
10. Test the page and make any necessary corrections.

Chapter 6

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

This assignment requires that you think ahead about how you will name the controls on the page in a meaningful and consistent way. Thus, you should read through Part 1 before you begin coding.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter06*.
2. Add a Page class to the project called *ComputeItPart1*.
3. Add a Button control that says *Perform +=*. Add two TextBox controls. Add a Label control. In the button click event handler method, declare two variables of type Byte. When the button is clicked, assign the value entered into the first text box to the first variable and assign the value entered into the second text box to the second variable. Then, *add* the first variable to the second variable and assign the result to the first variable *using a compound assignment statement*. Finally, display the value of the first variable in the label.
4. Test the page and make any necessary corrections.
5. Repeat Step 4 for the compound assignments -=, *=, /=, and %=.

Part 2

1. Make a copy of the *ComputeItPart1* Page class and rename it *ComputeItPart2*.
2. Convert each compound assignment statement to its equivalent *simple assignment statement*.
3. Test the page and make any necessary corrections.
4. Think about the difference between simple assignment statements and compound assignment statements. Add a comment at the bottom of your code that articulates your thoughts on which is a better way to code assignment statements and why.

Part 3

1. Make a copy of the *ComputeItPart2* Page class and rename it *ComputeItPart3*.
2. For each simple assignment statement performed in the code behind, write the code necessary to catch and handle any exception that is possible. (Only consider the exceptions discussed in the chapter.) Do *not* code for any exceptions that are not possible. For each assignment statement, put all exception tests into a single Try-Catch-Finally structure.
3. Test the page with *good* data and make any necessary corrections.
4. Test the page with *bad* data to make sure that any potential exceptions are caught and handled gracefully, which includes displaying an appropriate error message in the label.

Part 4

1. Add a Page class to the project called *CreateEnumeration*.

2. Create a helpful enumeration of your choosing in the code behind of the page.
3. Use the enumeration in an assignment statement.
4. Test the page and make any necessary corrections.

Chapter 7

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter07*.
2. Add a Page class to the project called *PerformWideningConversions*.
3. Add a Button control that says *Int16 to Int32*. Add a TextBox control. Add a Label control. When the button is clicked, convert the value entered into the text box to an Int16, *implicitly* convert the Int16 to an Int32, and display the result in the label.
4. Repeat Step 3 for Int32 to Double and Single to Double. For the latter, enter 12345.56789 and describe the result in a comment.
5. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *PerformNarrowingConversions*.
2. Add a Button control that says *Int32 to Int16*. Add a TextBox control. Add a Label control. When the button is clicked, convert the value entered into the text box to an Int32, *explicitly* convert (i.e., cast) the Int32 to an Int16, and display the result in the label.
3. Repeat Step 2 for Decimal to SByte. Enter a Decimal value (i.e., a value that includes digits after the decimal point) into the text box that can be accommodated by the SByte variable (e.g., 12.5) and describe the result in a comment. Enter a Decimal value into the text box that *cannot* be accommodated by the SByte variable (e.g., 1234.5) and describe the result in a comment.
4. Repeat Step 2 for Single to SByte. Enter a Single value (i.e., a value that includes digits after the decimal point) into the text box that can be accommodated by the SByte variable (e.g., 12.5) and describe the result in a comment. Enter a Single value into the text box that *cannot* be accommodated by the SByte variable (e.g., 1234.5) and describe the result in a comment.
5. Test the page and make any necessary corrections.

Part 3

1. Add a Page class to the project called *UseConvertClass*.
2. Add a Button control that says *Int16 to Int32*. Add a TextBox control. Add a Label control. When the button is clicked, convert the value entered into the text box to an Int16 *using the Convert class*, convert the Int16 to an Int32 *using the Convert class*, and display the result in the label. Note that this is a widening conversion.
3. Repeat Step 2 for Int32 to Double and Single to Double. Note that these are widening conversions.
4. Repeat Step 2 for Int32 to Int16, Decimal to SByte, and Single to SByte. Note that these are narrowing conversions.
5. Test the page and make any necessary corrections.

Chapter 8

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter08*.
2. Add a Page class to the project called *MakeDecisions*.
3. Add a ListBox control. The Value properties of the items in the list box should be *BB*, *FB*, *SB*, *TR*, and *VB*, respectively. The associated Text properties of the items in the list box should be *Basketball*, *Football*, *Soccer Ball*, *Tennis Racket*, and *Volleyball*, respectively. Add a Label control. When a list box option is selected, evaluate the Value property of the item selected. If BB is selected, display *Michael Jordan* in the label. If FB is selected, display *Peyton Manning* in the label. If SB is selected, display *Mia Hamm* in the label. If TR is selected, display *Roger Federer* in the label. If VB is selected, display *Jenny Ping* in the label. Use a *Nested-If* structure to solve this problem.
4. Repeat Step 3 but use an *If-Else-If* structure to solve the problem.
5. Repeat Step 3 but use a *Switch* structure to solve the problem.
6. Test the page and make any necessary corrections.
7. Add a comment at the bottom of your code that articulates your thoughts on which is the better way to code the solution and why.

Part 2

1. Add a Page class to the project called *PerformLoops*.
2. Add a Button control that says *Count Sports Using While*. Add a TextBox control. Default the Text property of this control to *Basketball, Football, Soccer, Tennis, Volleyball*. Add a Label control. When the button is clicked, loop through the string in the text box counting the number of items in the list and display in the label the number of items counted. Use a *While* structure to solve this problem.
3. Repeat Step 2 but use a Do-While structure to solve the problem.
4. Repeat Step 2 but use a *For* structure to solve the problem.
5. Test the page and make any necessary corrections.

Chapter 9

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter09*.
2. Add a Page class to the project called *FormatAddress*.
3. Add seven Label controls arranged vertically. These labels should display *First Name*, *Middle Initial*, *Last Name*, *Address*, *City*, *State*, and *Zip Code*, respectively.
4. Add seven corresponding TextBox controls next to each label.

5. Add another TextBox control to the right of these text boxes. Make sure that this text box is a multiline text box with three rows.
6. Add a Button control that says *Format Address* underneath the multiline text box. When the button is clicked, format the values entered into the seven text boxes so that they look like a three-line address label on a postal mail envelope and display the result in the multiline text box.
7. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *DisplayMessages*.
2. Add a Button control that says *Display Message*. Add a Label control. When the button is clicked, display in the label *You just clicked the "Display Message" button*. Make sure to include the double quotes in the message. Use escape characters to solve this problem.
3. Repeat Step 2 but use a verbatim literal to solve the problem.
4. Test the page and make any necessary corrections.

Part 3

Use the properties and/or methods of the String class to solve these problems. Some solutions may require you to combine more than one method.

1. Add a Page class to the project called *UseStringClass*.
2. Add a Button control that says *Check for Period*. Add a TextBox control. Add a Label control. Enter a sentence into the text box. When the button is clicked, display in the label whether or not the sentence ends with a period.
3. Add a Button control that says *Check for Hello*. Add a TextBox control. Add a Label control. Enter a sentence into the text box. When the button is clicked, display in the label whether or not the sentence contains the word *hello*.
4. Add a Button control that says *Make Uppercase*. Add a TextBox control. Add a Label control. Enter an email address into the text box. When the button is clicked, display in the label the same email address but in all upper case letters.
5. Add a Button control that says *Locate @ Sign*. Add a TextBox control. Add a Label control. Enter an email address into the text box. When the button is clicked, display in the label the character number of the email address's @ sign, where the first character in the email address is 1, the second character in the email address is 2, and so on.
6. Add a Button control that says *Check for The*. Add a TextBox control. Add a Label control. Enter a sentence into the text box. When the button is clicked, display in the label whether or not the sentence begins with the word *The*.
7. Add a Button control that says *Split List*. Add a TextBox control. Add a Label control. Enter four last names into the text box. When the button is clicked, split the list of names into an array and display in the label the second and third names in the array.
8. Add a Button control that says *Locate Last Period*. Add a TextBox control. Add a Label control. Enter an email address into the text box that contains more than one period (e.g., firstname.lastname@school.edu). When the button is clicked, display in the label the character number of the email address's last period, where the first character in the email address is 1, the second character in the email address is 2, and so on.

9. Add a Button control that says *Remove Spaces*. Add a TextBox control. Add a Label control. Enter a sentence into the text box. When the button is clicked, display in the label the same sentence with all of its spaces removed beginning with the tenth character in the sentence.
10. Add a Button control that says *Replace Single Quotes*. Add a TextBox control. Add a Label control. Enter a sentence into the text box that contains at least two single quotes (’). When the button is clicked, display in the label the same sentence with all of its single quotes replaced with grave accents (`). The grave accent key is often found under the Esc key on your keyboard.
11. Add a Button control that says *Format Dollar Amount*. Add a TextBox control. Add a Label control. Enter the number 123456 into the text box. When the button is clicked, display in the label the same number formatted as a dollar amount with commas.
12. Add a Button control that says *Get Number of Characters*. Add a TextBox control. Add a Label control. Enter a sentence into the text box. When the button is clicked, display in the label the number of characters in the sentence.
13. Add a Button control that says *Display Top-Level Domain*. Add a TextBox control. Add a Label control. Enter a valid email address into the text box. When the button is clicked, display in the label the top-level domain (e.g., com, edu, us) of the email address.
14. Add a Button control that says *Clean Up Sentence*. Add a TextBox control. Add a Label control. Enter a sentence into the text box that contains some unwanted spaces at the beginning and end of the sentence (after the period). When the button is clicked, display in the label the same sentence without any unwanted spaces at the beginning and end of the sentence.
15. Add a Button control that says *Format SSN*. Add a TextBox control. Add a Label control. Enter an unformatted social security number (i.e., an SSN without dashes) into the text box. When the button is clicked, display in the label the equivalent formatted social security number (i.e., an SSN with dashes).
16. Add a Button control that says *Remove Vowels*. Add a TextBox control. Add a Label control. Enter a sentence into the text box. When the button is clicked, display in the label the same sentence but without any vowels.
17. Test the page and make any necessary corrections.

Chapter 10

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter10*.
2. Add a Page class to the project called *ComputeNumbers*.
3. Add a Button control that says *Compute Total*. Add two TextBox controls. Add a Label control. Enter a subtotal into the first text box and enter a sales tax rate into the second text box. When the button is clicked, compute the total and display it in the label. Add parentheses to the computation to produce the correct total or to clarify the intent of the computation.
4. Add a Button control that says *Perform Modulo*. Add two TextBox controls. Add a Label control. Enter an integer into the first text box and enter another integer into the second text box. When the button is clicked, compute the modulus and display it in the label.

5. Add a Button control that says *Perform Postfix Increment*. Add a TextBox control. Add two Label controls. In the button click event handler method, create two Byte variables called bytNumber1 and bytNumber2. When the button is clicked, assign to bytNumber1 the value in the text box. Then, assign to bytNumber2 the value of bytNumber1 after performing a *postfix* increment of it. And finally, display in the first label the value of bytNumber1 and display in the second label the value of bytNumber2.
6. Repeat Step 5 but change the entire problem to a *prefix* increment. Note in a code behind comment how the results of the two computations (i.e., #5 and #6) differ and why.

Part 2

Use the properties and/or methods of the Math class to solve these problems. Some solutions may require you to combine more than one method.

1. Add a Page class to the project called *UseMathClass*.
2. Add a Button control that says *Display Nearest Integers*. Add a TextBox control. Add two Label controls. Enter a decimal number into the text box. When the button is clicked, display in the first label the nearest *larger* integer and display in the second label the nearest *smaller* integer.
3. Add a Button control that says *Display Positive or Negative*. Add a TextBox control. Add a Label control. Enter a number into the text box. When the button is clicked, display in the label a 1 (if the number is positive) or a -1 (if the number is negative).
4. Add a Button control that says *Display Larger Number*. Add two TextBox controls. Add a Label control. Enter different numbers into the text boxes. When the button is clicked, display in the label the larger of the two numbers.
5. Add a Button control that says *Display Smallest Number*. Add three TextBox controls. Add a Label control. Enter different numbers into the text boxes. When the button is clicked, display in the label the smallest of the three numbers.
6. Add a Button control that says *Raise to the Power*. Add two TextBox controls. Add a Label control. Enter numbers into the text boxes. When the button is clicked, display in the label the first number raised to the power of the second number.
7. Add a Button control that says *Display Integer*. Add a TextBox control. Add a Label control. Enter a decimal number into the text box. When the button is clicked, display in the label the integer portion of the number.
8. Add a Button control that says *Display without Sign*. Add a TextBox control. Add a Label control. Enter a negative number into the text box. When the button is clicked, display in the label the number without a sign.
9. Add a Button control that says *Display Square Root*. Add a TextBox control. Add two Label controls. Enter 110.25 into the text box. When the button is clicked, display in the first label the square root of the number rounded to the nearest integer (i.e., rounded using standard rounding) and display in the second label the square root of the number rounded to the nearest *even* integer (i.e., rounded using banker's rounding).
10. Test the page and make any necessary corrections.

Chapter 11

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Use the properties and/or methods of the DateTime structure to solve these problems. Some solutions may require you to combine more than one property or method.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter11*.
2. Add a Page class to the project called *ManipulateDates*.
3. Add a Button control that says *Compute Days*. Add a TextBox control. Add a Label control. Enter a date into the text box. When the button is clicked, display in the label the number of days between the date in the text box and the first day of this year. Do *not* use the TimeSpan structure to solve this problem.
4. Add a Button control that says *Compute Days*. Add a TextBox control. Add a Label control. Enter a date into the text box. When the button is clicked, display in the label the number of days between the date in the text box and the first day of this year. Use the TimeSpan structure to solve this problem.
5. Add a Button control that says *Check for Leap Year*. Add a TextBox control. Add a Label control. Enter a year into the text box. When the button is clicked, display in the label *It is a leap year* if the year is in a leap year or *It is not a leap year* if the year is not in a leap year.
6. Add a Button control that says *Compute Years*. Add two TextBox controls. Add a Label control. Enter dates into the text boxes. When the button is clicked, display in the label the difference between the two dates in years.
7. Add a Button control that says *Display Today 1*. Add a Label control. When the button is clicked, display in the label today's date in this form: month-day-year.
8. Add a Button control that says *Display Today 2*. Add a Label control. When the button is clicked, display in the label today's date in this form: year-month-day.
9. Add a Calendar control. Add a label control. When a date is selected, display in the label the associated day of the week.
10. Add a Calendar control. Add a label control. When a date is selected, display in the label the date that is 30 days after the selected date.
11. Add a Calendar control. Add a label control. When a date is selected, display in the label *Earlier* if the selected date is before today's date, *Same* if the selected date is equal to today's date, and *Later* if the selected date is after today's date.
12. Add a Button control that says *Parse Date*. Add a TextBox control. Add a Label control. Enter a date into the text box. When the button is clicked, display in the label the parsed date if the date successfully parses or *Invalid Date* if the date does not successfully parse. Test dates of the form 1/1/2000, 01-01-2000, 2000-01-01, and January 1, 2000.
13. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *ManipulateTimes*.

2. Add a Button control that says *Display Hours Minutes Seconds*. Add a Label control. When the button is clicked, display in the label the current time in this form: It is now ## hours, ## minutes, and ## seconds.
3. Add a Button control that says *Compare Times*. Add two TextBox controls. Add a Label control. Enter times into the text boxes in this form: hour, minute, second. Vary the seconds only. When the button is clicked, display in the label *Earlier* if the first time is before the second time, *Same* if the first time is equal to the second time, and *Later* if the first time is after the second date/time. Hint: Append both times to a date to get valid DateTime values.
4. Add a Button control that says *Compute Minutes*. Add two TextBox controls. Add a Label control. Enter times into the text boxes in the form hour:minute:second. Vary the minutes only. When the button is clicked, display in the label the number of minutes between the first time and the second time. Use the TimeSpan structure to solve this problem. Hint: Append both times to a date in the form year-month-day and then parse them to get valid DateTime values.
5. Add a Button control that says *Display Long Time*. Add a Label control. When the button is clicked, display in the label the current hour, minute, second, and whether it is AM or PM.
6. Add a Button control that says *Display Short Time*. Add a Label control. When the button is clicked, display in the label the current hour, minute, and whether it is AM or PM.
7. Add a Button control that says *Display Custom Time*. Add a Label control. When the button is clicked, display in the label the current time in this form: hour:minute O'clock AM (or PM).
8. Test the page and make any necessary corrections.

Chapter 12

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Use the properties and/or methods of the Array class to solve these problems. However, you may use indices instead of the SetValue and GetValue methods if you like. Some solutions may require you to combine more than one property or method.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter12*.
2. Add a Page class to the project called *PerformOneDimensionalArrayOperations*.
3. Declare a 5-element one-dimensional array called *strEquipmentArray* and populate its elements with the following values: *Basketball*, *Football*, *Soccer Ball*, *Tennis Racket*, and *Volleyball*.
4. Add a Button control called *Lookup Equipment*. Add a TextBox control. Add a Label control. Enter a number from 1 to 5 (not 0 to 4) into the text box. When the button is clicked, display in the label the associated equipment item (if the item is found) or *Not Found* (if the item is not found).
5. Add a Button control called *Display Equipment*. Add a TextBox control. Make this text box a read-only, multiline text box with five rows. When the button is clicked, display in the text box all of the equipment items—one item per line.
6. Add a Button control called *Display Equipment in Reverse*. Add a TextBox control. Make this text box a read-only, multiline text box with five rows. When the button is clicked, display in the text box all of the equipment items in reverse order—one item per line.

7. Add a Button control called *Display Equipment Array Properties*. Add a TextBox control. Make this text box a read-only, multiline text box with three rows. When the button is clicked, display in the text box the total number of elements in the array, the index of the first element of the array, and the index of the last element of the array—one item per line. Use labels (not Label controls) to indicate what each data item represents (i.e., *Total Number of Elements*, *Index of First Element*, and *Index of Last Element*, respectively).
8. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *PerformTwoDimensionalArrayOperations*.
2. Declare a 4x3 12-element two-dimensional array called *strClothingArray* and populate its elements with the following values: (Row 1) *Red Cap*, *White Cap*, *Blue Cap*, (Row 2) *Red Pants*, *White Pants*, *Blue Pants*, (Row 3) *Red Socks*, *White Socks*, *Blue Socks*, (Row 4) *Red T-Shirt*, *White T-Shirt*, and *Blue T-Shirt*.
3. Add a Button control called *Lookup Clothing*. Add two TextBox controls. Add a Label control. Enter a number from 1 to 4 (not 0 to 3) into the first text box and enter a number from 1 to 3 (not 0 to 2) into the second text box. When the button is clicked, display in the label the associated clothing item (if the item is found) or *Not Found* (if the item is not found).
4. Add a Button control called *Display Clothing*. Add a TextBox control. Make this text box a read-only, multiline text box with three rows. When the button is clicked, display in the text box all of the clothing items by color. The first line should list all of the red items, the second line should list all of the white items, and the third line should list all of the blue items.
5. Add a Button control called *Display Clothing in Reverse*. Add a TextBox control. Make this text box a read-only, multiline text box with five rows. When the button is clicked, display in the text box all of the clothing items in reverse order separated by commas.
6. Add a Button control called *Display Clothing Array Properties*. Add a TextBox control. Make this text box a read-only, multiline text box with seven rows. When the button is clicked, display in the text box the total number of elements in the array, the number of elements in the *first* dimension of the array, the index of the first element in the *first* dimension of the array, the index of the last element in the *first* dimension of the array, the number of elements in the *second* dimension of the array, the index of the first element in the *second* dimension of the array, and the index of the last element in the *second* dimension of the array—one item per line. Use labels (not Label controls) to indicate what each data item represents (i.e., *Total Number of Elements*, *Number of Elements in 1st Dimension*, *Index of First Element in 1st Dimension*, *Index of Last Element in 1st Dimension*, *Number of Elements in 2nd Dimension*, *Index of First Element in 2nd Dimension*, and *Index of Last Element in 2nd Dimension*, respectively).
7. Test the page and make any necessary corrections.

Chapter 13

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

To complete this project, you will need to "maintain the state" of the data structures required so that they can be used between postbacks to the server. A technique for doing this is described in detail in the section titled *Maintaining the State of a Data Structure* in the chapter titled *State Maintenance*.

Part 1

Use the properties and/or methods of the Stack class to solve these problems. Some solutions may require you to combine more than one property or method.

1. Start a new project called *Lastname.Firstname.Chapter13*.
2. Add a Page class to the project called *PerformStackOperations*.
3. Add a Button control that says *Add to Stack*. Add a TextBox control. Add a Label control. Enter a product (e.g., *Tennis Racquet*) into the text box. When the button is clicked, save the product to the stack and display in the label *[Name of Product] successfully added*. Add three more products to the stack.
4. Add a Button control that says *Display Size of Stack*. Add a Label control. When the button is clicked, display in the label the number of products in the stack.
5. Add a Button control that says *View Top of Stack*. Add a Label control. When the button is clicked, display in the label the product at the top of the stack (if one exists) or *Stack is Empty* (if one doesn't exist). However, do *not* remove the product from the stack.
6. Add a Button control that says *Remove from Stack*. Add a Label control. When the button is clicked, remove the product from the top of the stack (if one exists). In addition, display in the label the product that was removed from the stack (if one exists) or *Stack is Empty* (if one doesn't exist).
7. Add a Button control that says *Display Stack*. Add a TextBox control. Make this text box a read-only, multiline text box with five rows. When the button is clicked, display in the text box all of the products in the stack (one product per line) or *Stack is Empty* (if no products exist).
8. Test the page and make any necessary corrections.

Part 2

Use the properties and/or methods of the Queue class to solve these problems. Some solutions may require you to combine more than one property or method.

1. Add a Page class to the project called *PerformQueueOperations*.
2. Add a Button control that says *Add to Queue*. Add a TextBox control. Add a Label control. Enter a product (e.g., *Tennis Racquet*) into the text box. When the button is clicked, save the product to the queue and display in the label *[Name of Product] successfully added*. Add three more products to the queue.
3. Add a Button control that says *Display Size of Queue*. Add a Label control. When the button is clicked, display in the label the number of products in the queue.
4. Add a Button control that says *View Beginning of Queue*. Add a Label control. When the button is clicked, display in the label the product at the beginning of the queue (if one exists) or *Queue is Empty* (if one doesn't exist). However, do *not* remove the product from the queue.
5. Add a Button control that says *Remove from Queue*. Add a Label control. When the button is clicked, remove the product from the beginning of the queue (if one exists). In addition, display in the label the product that was removed from the queue (if one exists) or *Queue is Empty* (if one doesn't exist).

6. Add a Button control that says *Display Queue*. Add a TextBox control. Make this text box a read-only, multiline text box with five rows. When the button is clicked, display in the text box all of the products in the queue (one product per line) or *Queue is Empty* (if no products exist).
7. Test the page and make any necessary corrections.

Part 3

Use the properties and/or methods of the LinkedList class to solve these problems. Some solutions may require you to combine more than one property or method.

1. Add a Page class to the project called *PerformLinkedListOperations*.
2. Add a Button control that says *Add to Linked List*. Add a TextBox control. Add a Label control. Enter a product (e.g., *Tennis Racquet*) into the text box. When the button is clicked, save the product to the linked list and display in the label *[Name of Product] successfully added*. Add three more products to the linked list.
3. Add a Button control that says *Display Size of Linked List*. Add a Label control. When the button is clicked, display in the label the number of products in the linked list.
4. Add a Button control that says *View Start of Linked List*. Add a Label control. When the button is clicked, display in the label the product at the start of the linked list (if one exists) or *Linked List is Empty* (if one doesn't exist).
5. Add a Button control that says *View End of Linked List*. Add a Label control. When the button is clicked, display in the label the product at the end of the linked list (if one exists) or *Linked List is Empty* (if one doesn't exist).
6. Add a Button control that says *Search Linked List*. Add a TextBox control. Add a Label control. Enter a product into the text box. When the button is clicked, display in the label *Product Found* (if the product is found) or *Product Not Found* (if the product is not found).
7. Add a Button control that says *Remove from Linked List*. Add a TextBox control. Add a Label control. Enter a product into the text box. When the button is clicked, remove the product from the linked list (if it exists). In addition, display in the label the product that was removed from the linked list (if it exists) or *Product Not Found* (if it doesn't exist).
8. Add a Button control that says *Display Linked List*. Add a TextBox control. Make this text box a read-only, multiline text box with five rows. When the button is clicked, display in the text box all of the products in the linked list (one product per line) or *Linked List is Empty* (if no products exist).
9. Test the page and make any necessary corrections.

Part 4

Use the properties and/or methods of the SortedList class to solve these problems. Some solutions may require you to combine more than one property or method.

1. Add a Page class to the project called *PerformSortedListOperations*.
2. Add a Button control that says *Add to Sorted List*. Add two TextBox controls. Add a Label control. Enter a product *key* (e.g., *TR*) into the first text box and enter a product *value* (e.g., *Tennis Racquet*) into the second text box. When the button is clicked, save the product to the sorted list and display in the label *[Product Key] [Product Value] successfully added*. Add three more products to the sorted list.

3. Add a Button control that says *Display Size of Sorted List*. Add a Label control. When the button is clicked, display in the label the number of products in the Sorted list.
4. Add a Button control that says *Search Sorted List*. Add a TextBox control. Add a Label control. Enter a product key into the text box. When the button is clicked, display in the label the associated product value (if the product is found) or *Product Not Found* (if the product is not found).
5. Add a Button control that says *Remove from Sorted List*. Add a TextBox control. Add a Label control. Enter a product key into the text box. When the button is clicked, remove the product from the sorted list (if it exists). In addition, display in the label the product key and product value that was removed from the sorted list (if it exists) or *Product Not Found* (if it doesn't exist).
6. Test the page and make any necessary corrections.

Chapter 14

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Use the properties and/or methods of the File class to solve these problems. Some solutions may require you to combine more than one property or method.

1. Start a new project called *Lastname.Firstname.Chapter14*.
2. Add a Page class to the project called *PerformFileSystemOperations*.
3. Add a folder to the project called *Files*.
4. Add two TextBox controls. Make the second text box a multiline text box with five rows. Add a Label control.
5. Add a Button control that says *Create File*. Enter a file name (with a file extension of .txt) into the first text box. When the button is clicked, create the file in the folder and display in the label *File successfully created*. If the file already exists, display in the label *File already exists*.
6. Add a Button control that says *Update File*. Enter a file name into the first text box and enter a sentence of your choosing into the second text box. When the button is clicked, *overwrite* the contents of the file and display in the label *File successfully updated*. If the file does not exist, display in the label *File does not exist*.
7. Add a Button control that says *Read File*. Enter a file name into the first text box and delete anything from the second text box. When the button is clicked, read the contents of the file into the second text box and display in the label *File successfully read*. If the file does not exist, display in the label *File does not exist*.
8. Add a Button control that says *Protect File*. Enter a file name into the first text box. When the button is clicked, make the file read-only and display in the label *File is read-only*. If the file does not exist, display in the label *File does not exist*.
9. Add a Button control that says *Unprotect File*. Enter a file name into the first text box. When the button is clicked, make the file not read-only and display in the label *File is not read-only*. If the file does not exist, display in the label *File does not exist*.
10. Add a Button control that says *Delete File*. Enter a file name into the first text box. When the button is clicked, delete the file from the folder and display in the label *File successfully deleted*. If the file does not exist, display in the label *File does not exist*.
11. Test the page and make any necessary corrections.

Chapter 15

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter15*.
2. Add a folder to the project called *Classes*.
3. Add a *non-static* C# class to the folder called *Product*.
4. Add the following properties to the class: *Description*, *Price*, and *NumberInStock*. Make sure to define these with appropriate types.
5. Add a *non-static* method to the class called *UpdateNumberInStock*. This method should receive a single input parameter of type SByte called *bytNumberSold* and should return a value of type SByte. When the method is invoked, it should subtract the value in the bytNumberSold parameter from the value in the NumberInStock property giving a new NumberInStock property value. It should then return the value in the updated NumberInStock property. You may only use the bytNumberSold and the NumberInStock in this method. In other words, do *not* declare and use any local variables in the method.

Part 2

1. Add a Page class to the project called *InvokeNonStaticClass*.
2. Add four Label controls that say *Description*, *Price*, *Number in Stock, and Number Sold*. Arrange these labels vertically.
3. Add four associated TextBox controls.
4. Add a Button control that says *ModifyProduct*. Enter some data into the text boxes. When the button is clicked, create a Product object from the Product class, set the object's properties to the corresponding values entered into the text boxes, invoke the UpdateNumberInStock method with the value entered into the number sold text box, and display in the label the updated values of the object's properties—*not* the values in the text boxes.
5. Test the page and make any necessary corrections.

Part 3

1. Add a *static* C# class to the folder called *PhoneNumber*.
2. Add a *static* method to the class called *Translate*. This method should receive a single input parameter of type String called *strPhoneNumber* and should return a value of type String. When the method is invoked, it should take the phone number (e.g., 1-800-the-bomb) and translate it into its equivalent numeric number (i.e., 1-800-843-2662).
3. Add a *static* method to the class called *Clean*. This method should receive a single input parameter of type String called *strPhoneNumber* and should return a value of type String. When the method is invoked, it should take the phone number (e.g., 1-800-843-2662) and remove any non-numeric data from it (i.e., 18008432662).
4. Add a *static* method to the class called *Standardize*. This method should receive a single input parameter of type String called *strPhoneNumber* and should return a value of type String. When the

method is invoked, it should take the phone number (e.g., 1.800.843.2662) and replace any non-numeric data with dashes (i.e., 1-800-843-2662).

Part 4

1. Add a Page class to the project called *InvokeStaticClass*.
2. Add a TextBox control. Add a Label control.
3. Add a Button control that says *Translate*. Enter a phone number into the text box. When the button is clicked, invoke the Translate method of the PhoneNumber class and display in the label the translated phone number.
4. Add a Button control that says *Clean*. Enter a phone number into the text box. When the button is clicked, invoke the Clean method of the PhoneNumber class and display in the label the cleaned phone number.
5. Add a Button control that says *Standardize*. Enter a phone number into the text box. When the button is clicked, invoke the Standardize method of the PhoneNumber class and display in the label the standardized phone number.
6. Test the page and make any necessary corrections.

Chapter 16

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter16*.
2. Add a Page class to the project called *EnterOrderCookies*.
3. Add a Label control at the top of the page that says *Enter Order (Cookies)*. Make the text bold and increase the font to a tasteful size.
4. Add 13 Label controls. From top to bottom, these labels should say *Order ID*, *Customer*, *Employee*, *Shipper*, *Order Date*, *Last Name*, *First Name*, *Middle Initial*, *Address*, *City*, *State*, *Zip Code*, and *Phone*.
5. Next to these, add an order ID TextBox control, a customer DropDownList control (with three customer names of your choosing whose customer IDs are *1*, *2*, and *3*), an employee DropDownList control (with three employee names of your choosing whose employee IDs are *1*, *2*, and *3*), a shipper DropDownList control (with three shipper names *FedEx*, *UPS*, and *USPS* whose shipper IDs are *1*, *2*, and *3*), an order date Calendar control, a last name TextBox control, a first name TextBox control, a middle initial TextBox control, an address TextBox control, a city TextBox control, a state TextBox control, a ZIP code TextBox control, and a Phone TextBox control.
6. Make a copy of this Page class and rename it *ConfirmOrderCookies*.
7. Modify the Label control at the top of the *ConfirmOrderCookies* page so that it says *Confirm Order*.
8. Modify the *ConfirmOrderCookies* page so that all of the controls on the page are disabled.
9. At the bottom of the *EnterOrderCookies* page, add a Button control that says *Confirm*. When the button is clicked, save all of the drop down list items selected, all of the values entered into the text boxes, and the date selected from the calendar to *cookies*, redirect the end user to the *ConfirmOrderCookies* page, and display all of the selected and entered information for confirmation. Hint: You will need to set the VisibleDate property of the Calendar control on the

ConfirmOrderCookies page to get the correct month to display when a day is selected from a month other than the default month. Important: On the *EnterOrderCookies* page, do *not* save the selected customer's name, the selected employee's name, or the selected shipper's name. Instead, save the selected customer's ID, the selected employee's ID, and the selected shipper's ID. However, on the *ConfirmOrderCookies* page, display the selected customer's name, the selected employee's name, and the selected shipper's name in the drop down lists.

10. Test the pages and make any necessary corrections.

Part 2

1. Make a copy of the *EnterOrderCookies* Page class and rename it *EnterOrderQueryStrings*.
2. Make a copy of the *ConfirmOrderCookies* Page class and rename it *ConfirmOrderQueryStrings*.
3. Modify the code so that state is maintained using *query strings*.
4. Test the pages and make any necessary corrections.

Part 3

1. Make a copy of the *EnterOrderCookies* Page class and rename it *EnterOrderSessionState*.
2. Make a copy of the *ConfirmOrderCookies* Page class and rename it *ConfirmOrderSessionState*.
3. Modify the code so that state is maintained using *session state*.
4. Test the pages and make any necessary corrections.

Chapter 17

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Note: This project expands on the project created in Chapter 16. Thus, you should make sure that that project is in excellent working condition before proceeding.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter17*.
2. Add a MasterPage class to the project called *MasterPage*.
3. Code the master page as it is shown in the book.
4. Add your own graphic to the header of the master page and adjust it so that it is an appropriate size.

Part 2

1. Add a Page class with a MasterPage to the project called *EnterOrder*.
2. Copy the Aspx code from the *EnterOrderSessionState* page of the previous project and paste it into the Aspx code of the *EnterOrder* page. Copy the code behind code from the *EnterOrderSessionState* page of the previous project and paste it into the code behind code of the *EnterOrder* page.
3. Add a Page class with a MasterPage to the project called *ConfirmOrder*.
4. Copy the Aspx code from the *ConfirmOrderSessionState* page of the previous project and paste it into the Aspx code of the *ConfirmOrder* page. Copy the code behind code from the *ConfirmOrderSessionState* page of the previous project and paste it into the code behind code of the *ConfirmOrder* page.

5. Remove the Label controls at the top of the *EnterOrder* and *ConfirmOrder* pages that were used in the previous project as page titles. These are no longer necessary since the master page should have a place for page titles.
6. Modify the code in both content pages so that the exposed properties of the master page are set appropriately as each page loads.
7. Test the pages and make any necessary corrections.

Chapter 18

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Note: This project expands on the project created in Chapter 17. Thus, you should make sure that that project is in excellent working condition before proceeding.

1. Start a new project called *Lastname.Firstname.Chapter18*.
2. Add to this project the master page (i.e., *MasterPage*) and the two content pages (i.e., *EnterOrder* and *ConfirmOrder*) from the previous project. Hint: Use the *Add Existing Item…* option in Visual Studio.
3. Remove any formatting properties that you may have set on the individual server controls in the master page and in both content pages.
4. Add a theme to your project along with an associated *skin* file and *css* file. Make sure to modify your Web.config file so that your theme will be applied to all of your application's pages.
5. In your skin file, add a skin for each type of server control you are using in your application (i.e., Calendar control, DropDownList control, Label control, and TextBox control). Modify these skins (e.g., back color, fore color, font, font size) so that your pages reflect a holiday theme (e.g., Christmas, Fourth of July, Halloween) and view the result. Modify the skins again to reflect a different holiday theme and view the result. Set the EnableTheming property of one of your controls to *false* and view the result.
6. In your css file, add a *body* element selector (if one is not already there). Set the background color of your pages to a holiday-appropriate, yet aesthetically-pleasing, color and view the result. Add a holiday-appropriate background image to your pages and view the result. You may remove the background image if you don't like it. Hint: If a css selector is not having an effect when you run your application, you might need to clear your browser's cache.
7. Test the pages and make any necessary corrections.

Chapter 19

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Note: This project expands on the project created in Chapter 18. Thus, you should make sure that that project is in excellent working condition before proceeding.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter19*.
2. Add to this project the master page (i.e., *MasterPage*) from the previous project. Hint: Use the *Add Existing Item…* option in Visual Studio.
3. Add the following Page classes (each with a MasterPage class) to the project: *Home*, *MaintainCategories*, *MaintainCustomers*, *MaintainEmployees*, *MaintainOrders*, *MaintainProducts*, *MaintainShippers*, *MaintainSuppliers*, *DisplayProducts*, *ContactUs*, *SiteMap*, *About*, and *DisplayWilsonVolleyballs*.
4. Add a SiteMap class to your project called *Web.sitemap*. Code the site map similar to the site map shown in the book. However, under the *Products* option, add options for three *brands* of sports equipment (i.e., *Head*, *Spalding*, and *Wilson*). Then, under each of these items, add options for three *types* of sports equipment (i.e., *Baseballs*, *Footballs*, and *Volleyballs*).
5. Modify the code behind in these pages so that *all* of the exposed properties in the master page are set appropriately as each page loads.
6. Add a Menu control to the master page. Make sure that the end user's mouse pointer looks like a mouse pointer, instead of an I-beam, as it hovers over the Menu control. Experiment with the various display properties of the menu.
7. Test the menu and make any necessary corrections.

Part 2

1. Add a TreeView control to the *SiteMap* page.
2. Experiment with the various display properties of the tree view.
3. Test the tree view and make any necessary corrections.

Chapter 20

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

If you have not done so already, install SQL Server and SQL Server Management Studio (SSMS) on your computer. You will need both of these technologies to perform the tasks associated with this section of the book. Use SSMS to install the SportsPlay database supplied by your instructor.

Part 1

In SSMS, create a new query file called *Lastname.Firstname.Chapter20.sql* and save it. After coding and testing each SQL call below, comment it out so that you can code and test the other SQL calls independently.

1. Write a query that displays all customers.
2. Write a query that displays the same as above but display the last name, first name, middle initial, and email address only.
3. Write a query that displays the same as above but display the data in order by last name, first name, and middle initial.
4. Write a query that displays the same as above but only display the data for CustomerID 3.
5. Write a query that displays all products.

6. Write a query that displays the same as above but only display those products that have a price between $50 and $150, inclusive. Display the data in order by price.
7. Write a query that displays the same as above but only display tennis racquets.
8. Write a query that displays all customers and their associated orders. Do not include the order lines. Only display the customer's last name, first name, middle initial, order number (i.e., OrderID), and date.
9. Write a query that displays the same as above but display the last name, first name, and middle initial in this format: Jimmy J Johnson.
10. Write a query that displays all of the suppliers and their associated products and categories. Only display the supplier, product, price, and category. Display the data in order by supplier and product.
11. Write a statement that inserts an employee into the database. Insert data into *all* of the attributes.
12. Write a statement that updates the employee that was just inserted into the database. Change the data in *all* of the attributes.
13. Write a statement that deletes the employee that was just updated in the database.

Part 2

1. Start a new project called *Lastname.Firstname.Chapter20*.
2. Add a Page class to the project called *PopulateListBox*.
3. Add a Label control that says *Product*.
4. Add a ListBox control below the *Product* label. When the page loads, populate the list box with all of the products in the Product table.
5. Test the page and make any necessary corrections.

Part 3

1. Make a copy of the *PopulateListBox* Page class and rename it *FilterByDropDownList*.
2. Add a Label control that says *Supplier* above the *Product* label.
3. Add a DropDownList control below the *Supplier* label. When the page loads, populate the drop down list with all of the suppliers in the Supplier table and populate the list box with all of the products in the Product table that are associated with the supplier displayed in the drop down list. Make sure the products in the list box change when a different supplier is selected from the drop down list.
4. Test the page and make any necessary corrections.

Part 4

1. Make a copy of the *PopulateListBox* Page class and rename it *FilterBySessionVariable*.
2. Add a Page class to the project called *SetSessionVariable*.
3. Add a TextBox control.
4. Add a Button control that says *Set Session Variable*. When the button is clicked, display the *FilterBySessionVariable* page. When the *FilterBySessionVariable* page loads, populate the list box with all of the products in the Product table that contain the word entered into the text box on the *SetSessionVariable* page.
5. Test the page and make any necessary corrections.

Chapter 21

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

If you have not done so already, install SQL Server and SQL Server Management Studio (SSMS) on your computer. You will need both of these technologies to perform the tasks associated with this section of the book. Use SSMS to install the SportsPlay database supplied by your instructor.

1. Start a new project called *Lastname.Firstname.Chapter21*.
2. Add to this project the master page (i.e., *MasterPage*) you created in Chapter 17. Hint: Use the *Add Existing Item…* option in Visual Studio.
3. Add a Page class to the project called *MaintainOrders*. Make sure that the exposed properties of the master page are set appropriately as this page loads.
4. Add a DropDownList control that displays all of the Order IDs in the Order table.
5. Add a FormView control that maintains the Order table. You do *not* need to maintain the OrderLine table. Make sure that any messages regarding the status of an insert, update, or delete operation are displayed in the master page.
6. Add all the necessary validation controls so that no bad data can find its way into the database.
7. Test the page and make any necessary corrections.

Chapter 22

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

If you have not done so already, install SQL Server and SQL Server Management Studio (SSMS) on your computer. You will need both of these technologies to perform the tasks associated with this section of the book. Use SSMS to install the SportsPlay database supplied by your instructor.

1. Start a new project called *Lastname.Firstname.Chapter22*.
2. Add to this project the master page (i.e., *MasterPage*) you created in Chapter 17. Hint: Use the *Add Existing Item…* option in Visual Studio.
3. Add a Page class to the project called *MaintainOrderLines*. Make sure that the exposed properties of the master page are set appropriately as this page loads.
4. Add a DropDownList control that displays all of the Order IDs in the Order table.
5. Add a ListView control that maintains the OrderLine table. Make sure that any messages regarding the status of an insert, update, or delete operation are displayed in the master page.
6. Add all the necessary validation controls so that no bad data can find its way into the database.
7. Test the page and make any necessary corrections.

Chapter 23

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

If you have not done so already, install SQL Server and SQL Server Management Studio (SSMS) on your computer. You will need both of these technologies to perform the tasks associated with this section of the book. Use SSMS to install the SportsPlay database supplied by your instructor.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter23*.
2. Add to this project the master page (i.e., *MasterPage*) you created in Chapter 17. Hint: Use the *Add Existing Item…* option in Visual Studio.
3. Add a Page class to the project called *DisplayProducts*. Make sure that the exposed properties of the master page are set appropriately as this page loads.
4. Add a TextBox control. Make this text box a multiline text box with 15 rows that is wide enough to comfortably accommodate the output to be generated.
5. Add a Button control that says *DisplayProducts*. When the button is clicked, display all of the products in the Product table—one product per line. For each product, display *only* the Product, Price, Number in Stock, and the inventory value of the product (i.e., Price x Number in Stock). At the bottom of the report, display the total value of all the products. When the report completes, display in the master page a message that says *Report complete*.
6. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *LoginNonParameterizedQuery*. Make sure that the exposed properties of the master page are set appropriately as this page loads.
2. Add two Label controls. The first label should display *Email Address*, and the second label should display *Password*. Add a corresponding TextBox control next to each label.
3. Add a Button control that says *Login*. When the button is clicked, the values entered into the email address and password text boxes should be looked up in the Customer table using a *non-parameterized query*. If a match is found, display in the user section of the master page the customer's name and display in the message section of the master page a welcome message that contains the customer's name.
4. Test the page and make any necessary corrections.
5. Do some research about how you can use SQL injection to trick your application into thinking the end user has entered valid login credentials and give it a try.

Part 3

1. Make a copy of the *LoginNonParameterizedQuery* Page class and rename it *LoginParameterizedQuery*.
2. Modify the page so that a *parameterized query* is used to perform the login.
3. Test the page and make any necessary corrections.
4. Try using SQL injection to trick your application into thinking the end user has entered valid login credentials.

Part 4

1. Using SSMS, create a stored procedure called *CustomerLogin*.
2. Make a copy of the *LoginParameterizedQuery* Page class and rename it *LoginStoredProcedure*.

3. Modify the page so that the *Stored Procedure* is used to perform the login.
4. Test the page and make any necessary corrections.

Part 5

1. Using SSMS, create a stored procedure called *SupplierAdd*.
2. Add a Page class to the project called *AddSupplierStoredProcedure*. Make sure that the exposed properties of the master page are set appropriately as this page loads.
3. Add the appropriate Label controls and their associated TextBox controls so that the end user can enter all of the information for a new supplier.
4. Add a Button control that says *Save*. When the button is clicked, save the values entered into the text boxes to the Supplier table via the stored procedure. Display in the master page a message that says *Supplier successfully added*.
5. Test the page and make any necessary corrections.

Chapter 24

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

If you have not done so already, install the Papercut server on your computer. You will need this technology to perform the tasks associated with this chapter. If you receive an error message stating that the presence of HTML in your email message could be a security risk, you will need to add this information to the <system.web> section of your Web.config file: <pages validateRequest="false" />.

1. Start a new project called *Lastname.Firstname.Chapter24*.
2. Add a Page class to the project called *SendCustomerEmail*. In the steps that follow, you will be laying out the page so that it looks like a modern email client. Make sure to label each item appropriately.
3. Add a TextBox control to specify the email address of the sender of the message.
4. Add a TextBox control to specify the email address of the recipient of the message.
5. Add a TextBox control to specify the email address of a person who should receive a *carbon copy* of the message.
6. Add a TextBox control to specify the email address of a person who should receive a *blind carbon copy* of the message.
7. Add a TextBox control to specify the subject of the message.
8. Add a TextBox control to specify the body of the message. Make sure that this text box is a multiline text box with 15 rows and that its width is acceptable for entering an email message.
9. Add a CheckBox control to specify whether or not the body of the message is in HTML form. If the check box is checked, any HTML markup (e.g.,
, <b>, <i>) in the message should be used to format the message appropriately. If the check box is not checked, any HTML markup in the message should be displayed as is.
10. Add a RadioButton control to specify the priority (i.e., low, medium, high) of the message.
11. Add a Button control that says *Send*. When the button is clicked, send the email message to the Papercut server for display. Evaluate the message to make sure that it was sent to the correct recipients, that it was formatted properly, and that it was sent with the correct priority.

12. Test the page and make any necessary corrections.

Chapter 25

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

If you have not done so already, install the Ajax Control Toolkit on your computer. You will need this technology to perform the tasks associated with this chapter.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter25*.
2. Add a Page class to the project called *CountToOneThousand*.
3. Add a TextBox control. Make sure that this text box is a multiline text box with 15 rows.
4. Add a Button control that says *Count*. When the button is clicked, display in the text box all of the numbers between 1 and 1,000.
5. Add an UpdatePanel control so that, when the button is clicked, an asynchronous postback and partial-page rendering occurs.
6. Add an UpdateProgress control so that, when the button is clicked, *Please wait…* is displayed to the right of the button, if the process takes more than 1 second. If the process doesn't take more than 1 second (and thus *Please wait…* is not displayed), reduce the value of the DisplayAfter property or increase the loop counter (e.g., to 5,000).
7. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *DisplayFriends*.
2. Add an Accordion control that displays the names and bios of three friends. The names should be displayed in the headers of the control, and the bios should be displayed in the contents of the control.
3. Experiment with the control's properties and CSS formatting.
4. Test the page and make any necessary corrections.

Part 3

1. Add a Page class to the project called *UploadFile*.
2. Add a folder to the project called *Documents*.
3. Add an AjaxFileUpload control that permits you to select a file (.docx and .pdf files only) and upload it to the folder.
4. Experiment with the control's properties and CSS formatting.
5. Test the page and make any necessary corrections.

Part 4

1. Add a Page class to the project called *DisplayBalloonPopupExtender*.
2. Add a TextBox control.

3. Add a BalloonPopupExtender control that extends the text box by displaying a message when you hover over the text box with your pointer.
4. Experiment with the control's properties and CSS formatting.
5. Test the page and make any necessary corrections.

Part 5

1. Add a Page class to the project called *SelectDate*.
2. Add a TextBox control. Add a Label control.
3. Add a CalendarExtender control that extends the text box by permitting you to select a date. Default the selected date to today's date. When a date is clicked, display the selected date in the label.
4. Experiment with the control's properties and CSS formatting.
5. Test the page and make any necessary corrections.

Part 6

1. Add a Page class to the project called *AskQuestion*.
2. Add a Button control.
3. Add a ModalPopupExtender control that extends the button. When the button is clicked, ask the end user a question that has at least two possible responses, accept his or her answer, and display an appropriate message based their answer.
4. Experiment with the control's properties and CSS formatting.
5. Test the page and make any necessary corrections.

Part 7

1. Add a Page class to the project called *CreateUsernameAndPassword*.
2. Add two Label controls. The first label should say *Username*, and the second label should say *Password*. Add two corresponding TextBox controls. Add a Button control that says *Create*.
3. Add a PasswordStrength (extender) control that extends the password text box.
4. Experiment with the control's properties and CSS formatting.
5. Test the page and make any necessary corrections.

Chapter 26

Make sure all server controls are given a proper ID and all variables are given a proper name based on the naming standards discussed in the book. However, do *not* give a control an ID if it is not referenced in the code behind file. Use a table to cleanly arrange all of the controls on the page.

Use JavaScript to implement the functionality in this project.

Part 1

1. Start a new project called *Lastname.Firstname.Chapter26*.
2. Add a Page class to the project called *DuplicateEntries*.
3. Add a CheckBox control that says *Duplicate Entries*. Default this control to unchecked.
4. Under this, add two RadioButton controls (*Yes* and *No*, respectively), a TextBox control, and a DropDownList control (with the options *Red*, *White*, and *Blue*). The corresponding values of the drop down list options should be *R*, *W*, and *B*, respectively.

5. Create exact duplicates of these controls on the page.
6. When the *Duplicate Entries* check box is checked, copy the values/selections in the first set of controls to the second set controls and disable all of the controls on the page, except for the *Duplicate Entries* check box.
7. When the *Duplicate Entries* check box is unchecked, reset the second set of controls to their respective defaults and enable all of the controls on the page.
8. Test the page and make any necessary corrections.

Part 2

1. Add a Page class to the project called *AskQuestion*.
2. Add a Label control.
3. Add a Button control that says *Ask Question*. When the button is clicked, display a confirm dialog that asks the end user this question: *Are you sure you want to learn more about JavaScript? Click OK if you do. Click Cancel if you don't.*
4. If the end user clicks *OK*, display this message in the label: *Great!* If the end user clicks *Cancel*, display this message in the label: *Come on! It's fun!*
5. Modify your code so that the messages also appear in an alert dialog.
6. Test the page and make any necessary corrections.

Part 3

1. Add a Page class to the project called *LoadTextBox*.
2. Add a TextBox control. Make sure that this text box is a multiline text box with 15 rows.
3. Add a Button control that says *PrintNumbers*. When the button is clicked, display in the text box all of the numbers between 1 and 100.
4. Add two RadioButton controls. The first should say *Odd*, and the second should say *Even*.
5. Modify the code so that, when the button is clicked, only the odd or even numbers are displayed in the text box depending on which radio button is selected.
6. Test the page and make any necessary corrections.

Part 4

1. Add a Page class to the project called *DisplayDateAndTime*.
2. Add a Label control.
3. When the page loads, display the current date and time on the page.
4. Test the page and make any necessary corrections.