Session 1: Exercise

1 Create a new *ASP.NET Web Forms Application* project in your sample files folder called: **Exercise1**

2 Add a new *Web Form* item to the project called: **mypage.aspx**

3 Add a *Calendar* control to the *mypage.aspx* page.

4 Use *QuickTasks* to *Auto Format* the Calendar control to the *Colorful 1* scheme.

5 Change the *ID* property of the Calendar control to: **CalendarColorful**

6 Delete the existing *Images* folder.

7 Add a new **Images** folder.

8 Add the *pattern.jpg* file from the *Images* folder in your sample files folder to your new *Images* folder.

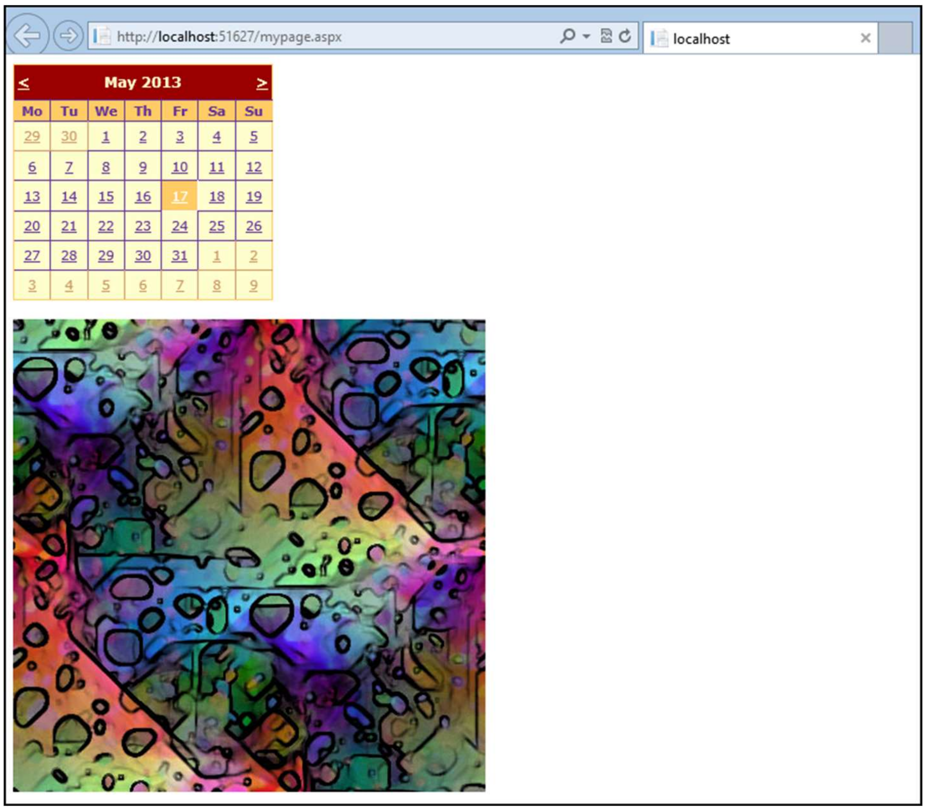
9 Add a HTML *Image* control to the page using the *HTML* category of the *ToolBox*.

10 Set the *Src* property of the new *Image* control to: **Images/pattern.jpg**

11 Set *mypage.aspx* to be the project's start page.

12 Start the project in debug mode.

13 Save your work.



If you need help slide the page to the left

Session 1: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 8	Q 7	Q 5	Q 3
1. Right-click on the *Images* folder in the *Solution Explorer*. 2. Click: Add→ Existing Item... from the shortcut menu. 3. Browse to the *C:\Practice\ASP.NET45\Images* folder. 4. Click on *pattern.jpg* and then click *Add*. This was covered in: *Lesson 1-8: Add and remove files from a project.*	1. Right-click on *Exercise1* in the *Solution Explorer*. 2. Click Add→ New Folder from the shortcut menu. 3. Type the name: **Images** This was covered in: *Lesson 1-8: Add and remove files from a project.*	1. Click on the calendar in *Design* view. 2. Scroll down in the *Properties* window until you see the *ID* property. 3. Click in the box that currently says *Calendar1* and change the text to: **CalendarColorful** This was covered in: *Lesson 1-13: Change properties in Design view.*	1. Double-click on *mypage.aspx* in the *Solution Explorer*. 2. Click on the *Design* button at the bottom of the main panel. 3. Drag a *Calendar* control from the *ToolBox* to the page. This was covered in: *Lesson 1-15: Add controls to a page with the Toolbox.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1	**Refer to: Lesson 1-5: Create an ASP.NET Web Forms Application project.**
2	**Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**
4	**Refer to: Lesson 1-16: Use the QuickTasks menu.**
6	**Refer to: Lesson 1-8: Add and remove files from a project.**
9	**Refer to: Lesson 1-15: Add controls to a page with the Toolbox.**
10	**Refer to: Lesson 1-13: Change properties in Design view.**
11	**Refer to: Lesson 1-9: Run a project in debug mode.**
12	**Refer to: Lesson 1-9: Run a project in debug mode.**
13	**Refer to: Lesson 1-10: View .aspx pages in Source and Design views.**

Session 2: Exercise

1 Open *exercise.aspx* within the *HTMLTest* sample project in *Source* view.

2 Set the page title in the head section to: **Session 2 Exercise**

3 Add a link to the CSS file called *layout.css*. It can be found in the *styles* folder.

4 Add a pair of *div* tags to the page (between the *form* tags).

5 Type the text **Site Name** between the *div* tags.

6 Set the *class* property of the *div* tag to the CSS class: **header**

7 Switch to *Design* view and add an HTML table to bottom of the page using the default settings.

8 Remaining in *Design* view, merge the bottom two cells of the HTML table.

9 In the first cell of the HTML table, type the text: **Site**

10 Switch to *Source* view and make the *Site* text bold using HTML tags.

11 Switch to *Design* view and type the text: **ASPNETCentral Website** into the top-right table cell.

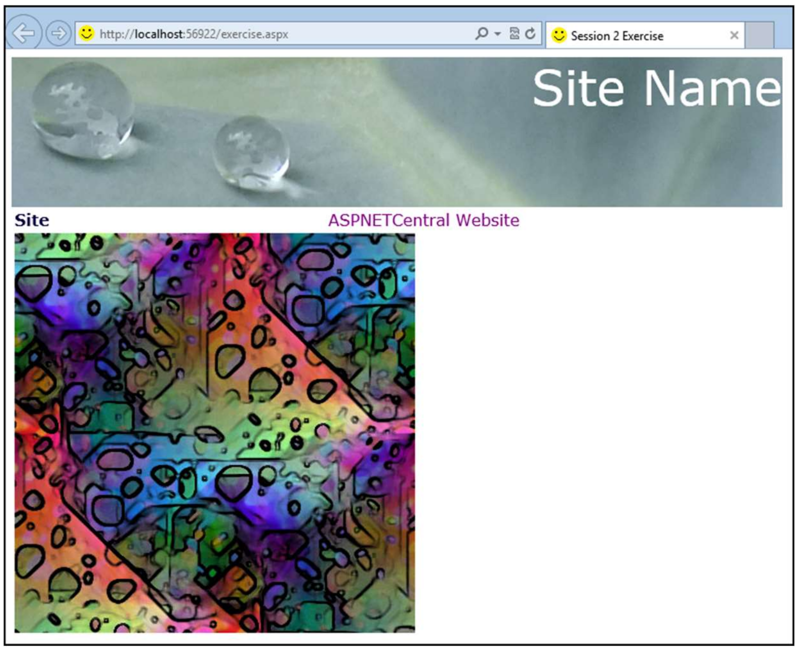
12 Make the text you have just typed into a hyperlink to: **http://www.ASPNETCentral.com**

13 Add an HTML Image control to the bottom row of the table and configure it to display the *pattern.jpg* image from the *images* folder.

14 Using the *CSS Properties* window, set the *color* CSS property of the *Site Name* text to: **White**

15 Add a link to the JavaScript file *exercise.js*. It can be found in the *scripts* folder.

16 Add JavaScript code to *exercise.js* to display a pop-up message.



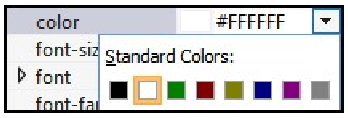
HTMLTest - start

HTMLTest - end

If you need help slide the page to the left

Session 2: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 14	Q 8	Q 7	Q 6
1. Switch to *Design* view. 2. Click on the *header* div so it is highlighted. *<DIV>* should appear as the selected item in the *Properties* window. 3. Click View→ CSS Properties. 4. Open the drop-down menu next to *color* in the *CSS Properties* window and click the white box.  color #FFFFFF font-siz Standard Colors: font font-fa This was covered in: *Lesson 2-9: Use the CSS Properties window.*	1. Switch to *Design* view. 2. Click and drag from the bottom-left cell of the table to the bottom right, so they are both highlighted. 3. Click Table→Modify→ Merge Cells. This was covered in: *Lesson 2-5: Create an HTML table.*	1. Switch to *Design* view. 2. Click below the *header* div. 3. Click Table→ Insert Table. 4. Click OK on the dialog that appears. This was covered in: *Lesson 2-5: Create an HTML table.*	1. Switch to *Source* view. 2. Modify the *div* tag to: **<div class="header"> Site Name </div>** This was covered in: *Lesson 2-10: Use the div and span tags.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

2 **Refer to: Lesson 2-4: Use the title, meta, link and script tags.**

3 **Refer to: Lesson 2-4: Use the title, meta, link and script tags.**

4 **Refer to: Lesson 2-10: Use the div and span tags.**

5 **Refer to: Lesson 2-10: Use the div and span tags.**

9 **Refer to: Lesson 2-5: Create an HTML table.**

10 **Refer to: Lesson 2-1: Understand HTML bold, italic and heading tags.**

11 **Refer to: Lesson 2-5: Create an HTML table.**

12 **Refer to: Lesson 2-7: Display images and links on a page.**

13 **Refer to: Lesson 2-7: Display images and links on a page.**

15 **Refer to: Lesson 2-4: Use the title, meta, link and script tags.**

16 **Refer to: Lesson 2-11: Work with JavaScript.**

Session 3: Exercise

1 Open the *CSharpTest* sample project and open *exercise.aspx*.

2 Disable *ViewState* on the *TextBoxText* control by setting its *EnableViewState* property to: **False**

3 Add a *Click* event handler to the *ButtonChangeText* control.

4 Add code to the new *Click* event handler to set the *Text* property of the *TextBoxText* control to: **The Smart Method**

5 Add a *Click* event handler to the *ButtonSendData* control.

6 Add code to the *ButtonSendData* control's *Click* event to move to *passdata2.aspx* using *Server.Transfer*.

7 Set a breakpoint in the *Click* event of *ButtonSendData*.

8 Run *exercise.aspx* in Debug mode and type some text into the text box.

9 Click *Send Data* and then use the *Watch* window to get the value of *TextBoxText.Text*.

10 Stop debugging and add code to the *ButtonSendData* control's *Click* event handler to store the *Text* of the *TextBoxText* control in *Session* under the key of *Text*.

11 Change the *ButtonSendData* control's *Click* event handler to redirect the user to *passdata4.aspx* using *Response.Redirect* instead of *Server.Transfer*.

CSharpTest - start

CSharpTest - end

If you need help slide the page to the left

Session 3: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 9	Q 7	Q 6	Q 3
1. Run *exercise.aspx* in Debug mode by clicking: Debug→Start Debugging. 2. Click on the *Send Data* button. Your code will be paused. 3. Return to the code-behind file of *exercise.aspx* if you aren't automatically sent there. 4. Click on the *Watch* button at the bottom of the screen. 5. Click in an empty box in the *Watch* window and type: **TextBoxText.Text** 6. Press <**Enter**>. This was covered in: *Lesson 3-3: Use Breakpoints.*	1. Open the code-behind file of *exercise.aspx*. 2. Right-click on the *Page.Server.Transfer* line in the *ButtonSendData_Click* event handler. 3. Click: Breakpoint→ Insert Breakpoint from the shortcut menu. This was covered in: *Lesson 3-3: Use Breakpoints.*	1. Open the code-behind file of *exercise.aspx*. 2. Add the following code to the *ButtonSendData_Click* event handler: **Page.Server. Transfer("passdata2.aspx");** This was covered in: *Lesson 3-10: Move between pages using C#.*	1. Open *exercise.aspx* in *Design* view. 2. Select the *ButtonChangeText* control by clicking on it. 3. Click on the *Events* button in the *Properties* window. ⚡ 4. Double-click in the empty box next to *Click*. This was covered in: *Lesson 3-2: Add event handlers to controls.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

2 **Refer to: Lesson 3-9: Work with ViewState.**

4 **Refer to: Lesson 3-1: Change properties with C#.**

5 **Refer to: Lesson 3-2: Add event handlers to controls.**

8 **Refer to: Lesson 1-9: Run a project in debug mode.**

10 **Refer to: Lesson 3-11: Send data between pages.**

11 **Refer to: Lesson 3-10: Move between pages using C#.**

Session 4: Exercise

1 Open the *ShiningStone* sample project and open *buy.aspx* in *Design* view.

2 Set the maximum length of each of the address text box controls to 50.

3 Make each of the address text boxes 50 columns wide.

4 Add a *CheckBox* control in the space before the *Submit Order* button.

5 Set the *Text* property of the *CheckBox* control to: **I accept the terms and conditions**

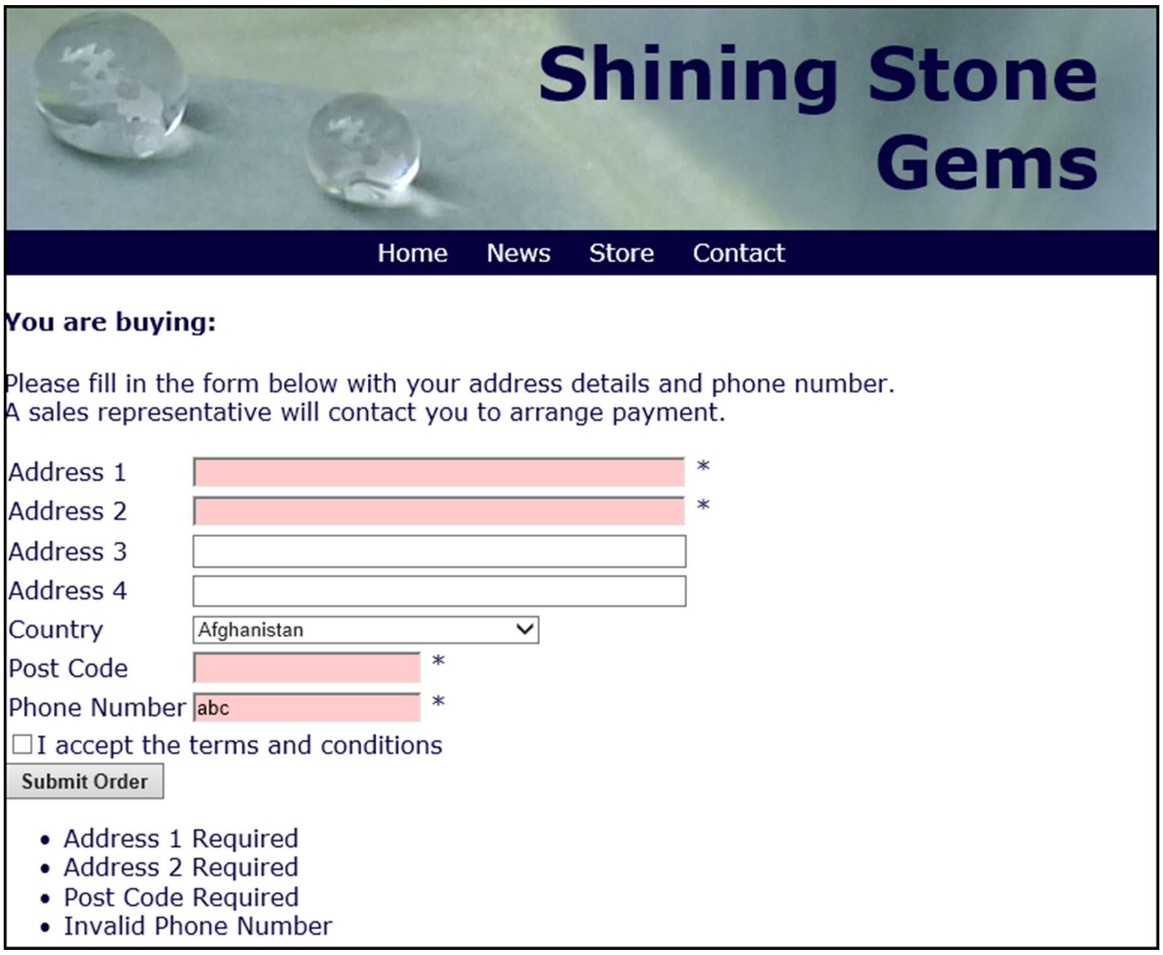
6 Set the *CheckBox* control's ID property to: **CheckBoxAcceptTerms**

7 Add a *RequiredFieldValidator* control next to the *Address 2* text box and configure it appropriately.

8 Add a *RequiredFieldValidator* next to the *Post Code* text box and configure it appropriately.

9 Make the background color of the *Post Code* text box match the background color of the *Address 1* text box.

10 Make the font of the *Submit Order* button bold.



You are buying:

Please fill in the form below with your address details and phone number.
A sales representative will contact you to arrange payment.

Address 1		*
Address 2		*
Address 3		
Address 4		
Country	Afghanistan	
Post Code		*
Phone Number	abc	*

☐ I accept the terms and conditions

Submit Order

- Address 1 Required
- Address 2 Required
- Post Code Required
- Invalid Phone Number

ShiningStone - start

ShiningStone - end

If you need help slide the page to the left

Session 4: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 9	Q 7	Q 6	Q 2
1. Open *buy*.aspx in *Design* view. 2. Click one of the pink text boxes. 3. Examine the *BackColor* property. You will see that it is set to: *#FFCCCC* 4. Click the *Post Code* text box. 5. Set the *BackColor* property to: **#FFCCCC** This was covered in: *Lesson 1-13: Change properties in Design view.*	1. Open *buy*.aspx in *Design* view. 2. Drag a *RequiredFieldValidator* from the *Validation* category of the *Toolbox* to the space after *TextBoxAddress2*. 3. Select the *RequiredFieldValidator*. 4. Set the *ID* property to: **RequiredFieldValidatorAddress2** 5. Set the *Text* property to: ***** 6. Set the *ErrorMessage* property to: **Address 2 Required** 7. Set the *ControlToValidate* property to: **TextBoxAddress2** This was covered in: *Lesson 4-8: Use the RequiredFieldValidator control.*	1. Open *buy*.aspx in *Design* view. 2. Select *CheckBox1* and set its *ID* property to: **CheckBoxAcceptTerms** This was covered in: *Lesson 4-1: Name controls correctly.*	1. Open *buy*.aspx in *Design* view. 2. Select each of the address text box controls by clicking on them. 3. Set the *MaxLength* property of each text box control to: **50** This was covered in: *Lesson 4-4: Use the TextBox control.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

3 **Refer to: Lesson 4-4: Use the TextBox control.**

4 **Refer to: Lesson 4-5: Use the CheckBox control.**

5 **Refer to: Lesson 4-5: Use the CheckBox control.**

8 **Refer to: Lesson 4-8: Use the RequiredFieldValidator control.**

10 **Refer to: Lesson 4-10: Use common properties.**

Session 5: Exercise

1 Open the *My Project* sample project and open *calculator.aspx* in Design view.

2 Add a new *Button* control to the page called: **ButtonCalculate2**

3 Add a *Click* event handler to the *ButtonCalculate2* control.

4 Create a *string* variable called **PIString** in the *ButtonCalculate2_Click* event handler with a value of: "**3.14159265**"

5 Create a *double* variable called **PIDouble** in the same event handler and set its value to the value of the *PIString* variable by using the *Convert* method.

6 Create an *int* variable in the same event handler called **CircleRadius** with a value of: **19**

7 Create a *double* variable in the same event handler called **CircleCircumference** with a value of: **PIDouble * CircleRadius**

8 Use the *Pow* function from the *Math* library to raise the *CircleCircumference* variable to the power of 2.

9 Convert the *CircleCircumference* variable to a *string* using the *ToString* method. Call the *string*: **OutputCircumference**

10 Create a *DateTime* variable called **TodaysDate** containing today's date.

My Project - start

My Project - end

If you need help slide the page to the left

Session 5: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 9	Q 8	Q 5	Q 4
Use the following line of code: **string OutputCircumference = CircleCircumference .ToString();** This was covered in: *Lesson 5-8: Convert variables using Convert and Parse.*	Use the following line of code: **CircleCircumference = Math.Pow (CircleCircumference, 2);** This was covered in: *Lesson 5-11: Use the Math library for advanced mathematics.*	Use the following line of code: **double PIDouble = Convert.ToDouble(PIString);** This was covered in: *Lesson 5-8: Convert variables using Convert and Parse.*	Use the following line of code: **string PIString = "3.14159265";** This was covered in: *Lesson 5-3: Use string variable properties and methods.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

2 **Refer to: Lesson 1-15: Add controls to a page with the Toolbox.**

3 **Refer to: Lesson 3-2: Add event handlers to controls.**

6 **Refer to: Lesson 5-4: Use integer variables.**

7 **Refer to: Lesson 5-10: Perform basic mathematical operations.**

10 **Refer to: Lesson 5-7: Use DateTime variables.**

Session 6: Exercise

1 Open the *My Project* sample project and add a new class called: **Circle.cs**

2 Add a public *double* property to the *Circle* class called: **CircleCircumference**

3 Add a public method to the *Circle* class called: **CalculateDiameter**

4 Make the *CalculateDiameter* method return a *double* value.

(Don't worry about the indicated error, this will be overcome in question 6).

5 Make the *CalculateDiameter* method ask for a *double* argument called: **Radius**

6 Add code to the *CalculateDiameter* method to multiply the *Radius* argument by 2 and return the result.

7 Add a constructor method to the *Circle* class.

8 Make the constructor method require a *double* value as an argument called: **Circumference**

9 Make the constructor method set the *CircleCircumference* property to the value of the *Circumference* argument.

10 Make the *CalculateDiameter* method into a static method.

11 Add a new Web Form to the project called: **circlecalculator.aspx**

12 Open the code-behind file of *circlecalculator.aspx*.

13 Add code to the *Page_Load* event handler to create an instance of the *Circle* class named **MyCircle** using a *Circumference* argument of: **50**

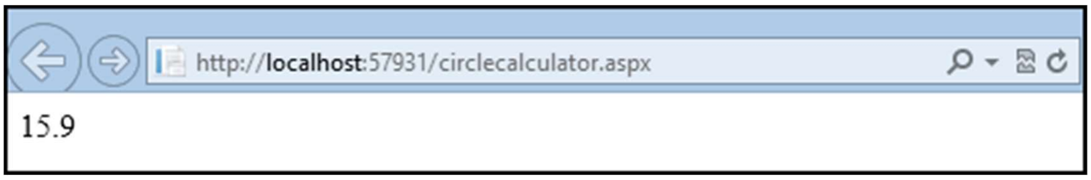
14 Add code on the next line to create a new *double* variable called: **MyCircleDiameter**

15 Add code on the next line to call the static *CalculateDiameter* method of the *Circle* class with a *Radius* argument of **7.95**, storing the resulting value in the *MyCircleDiameter* variable.

(Remember that *CalculateDiameter* is a static method and is called in a different way to normal methods).

16 Add code to output the value of *MyCircleDiameter* using *Response.Write*.

17 View *circlecalculator.aspx* in your browser.



```
15.9
```

My Project - start

My Project - end

If you need help slide the page to the left

Session 6: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 10	Q 7	Q 6	Q 3
Change the line that starts the *CalculateDiameter* method to: **public static double CalculateDiameter (double Radius)** This was covered in: *Lesson 6-9: Create a static method.*	Use the following code: **public Circle()** **{** **}** This was covered in: *Lesson 6-11: Create a class constructor method.*	Use the following line of code: **return Radius * 2;** This was covered in: *Lesson 6-7: Create methods that return a value.*	Use the following code to add the public method: **public void CalculateDiameter()** **{** **}** This was covered in: *Lesson 6-5: Create and use methods.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 6-1: Create a class.**

2 **Refer to: Lesson 6-1: Create a class.**

4 **Refer to: Lesson 6-7: Create methods that return a value.**

5 **Refer to: Lesson 6-6: Create methods with arguments.**

8 **Refer to: Lesson 6-11: Create a class constructor method.**

9 **Refer to: Lesson 6-11: Create a class constructor method.**

11 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

12 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

13 **Refer to: Lesson 6-11: Create a class constructor method.**

14 **Refer to: Lesson 5-5: Use floating point variables.**

15 **Refer to: Lesson 6-9: Create a static method.**

16 **Refer to: Lesson 3-7: Understand Request and Response.**

17 **Refer to: Lesson 1-9: Run a project in debug mode.**

Session 7: Exercise

1 Open the *Spark* sample project and open *viewtransactions.aspx* in *Design* view.

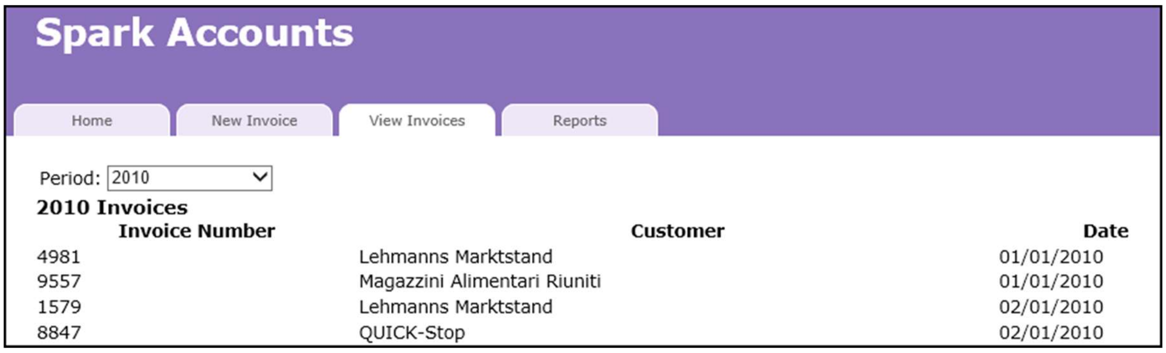
2 Add a *SelectedIndexChanged* event handler to the *DropDownListSelectedPeriod* control.

3 Add an *if* statement to the event handler that checks whether the value of the *DropDownListSelectPeriod* control's *SelectedValue* property is equal to: "**2010**"

4 If the value of the property is "**2010**", make your *if* statement change the *Panel2010.Visible* property to **true** and the *Panel2011.Visible* property to **false.**

5 Use *else if* to check whether the value of *DropDownListSelectPeriod.SelectedValue* is "**2011**". If it is, set the *Panel2011.Visible* property to **true** and the *Panel2010.Visible* property to **false**.

6 View *viewtransactions.aspx* in your browser and test your code.



2010 Invoices

Invoice Number	Customer	Date
4981	Lehmanns Marktstand	01/01/2010
9557	Magazzini Alimentari Riuniti	01/01/2010
1579	Lehmanns Marktstand	02/01/2010
8847	QUICK-Stop	02/01/2010

7 Close your browser and open the code-behind file of *newtransaction.aspx*.

8 Add an *if* statement to the start of the *ButtonSubmit_Click* event handler to check whether the value of the *DropDownListCustomer* control's *SelectedValue* property is "**6**", "**9**" or "**11**". If so, set the *Text* property of the *LabelError* control to:
That customer is currently out of use

9 Add an *else* statement to the *ButtonSubmit_Click* event handler which will run if the value of the property is not "**6**", "**9**" or "**11**".

10 Add *try* and *catch* statements to the *ButtonSubmit_Click* event handler and place any error messages in the *Text* property of the *LabelError* control.

11 Add a comment to the *CalculateVAT* method to explain what it does. (VAT or Value Added Tax is a sales tax levied in Europe).

12 Add a summary to the *CalculateVAT* method and populate it with useful descriptions.

Spark - start

Spark - end

If you need help slide the page to the left

Session 7: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 10	Q 9	Q 8	Q 4

Add the code:
 	
try
{
 	
…at the very beginning of the event handler.
 	
At the very end of the event handler, add:
 	
}
catch (Exception Ex)
{
   LabelError.Text = Ex.Message;
}
 	
This was covered in: *Lesson 7-6: Use try and catch to handle errors.*
	After the end of your last *if* statement, add the code:
	
	else
	{
	}
	
	This was covered in: *Lesson 7-2: Use else and else if.*
		Use the following lines of code:
		
		string CustomerID = DropDownListCustomer .SelectedValue;
		if (CustomerID == "6"
		|| CustomerID == "9"
		|| CustomerID == "11")
		{
		   LabelError.Text = "That customer is currently out of use.";
		}
		
		This was covered in: *Lesson 7-3: Use basic logical operators.*
			Use the following lines of code:
			
			if (DropDownListSelectPeriod .SelectedValue == "2010")
			{
			   Panel2010.Visible = true;
			   Panel2011.Visible = false;
			}
			
			This was covered in: *Lesson 7-1: Use the if statement.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

2 **Refer to: Lesson 3-2: Add event handlers to controls.**

3 **Refer to: Lesson 7-1: Use the if statement.**

5 **Refer to: Lesson 7-2: Use else and else if.**

6 **Refer to: Lesson 1-8: Add and remove files from a project.**

7 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

11 **Refer to: Lesson 7-7: Use comments.**

12 **Refer to: Lesson 7-8: Use summaries.**

Session 8: Exercise

1 Open the *My Project* sample project and create a new class called: **MyData.cs**

2 Add a new public method called **GetNumbers**, which returns an array of *int* variables.

(You'll see an error at this stage as you have not yet created code that returns a value).

3 Create an array of *int* variables called **Numbers** in the *GetNumbers* method containing the numbers: **1, 1, 3, 5, 8** and make the method return the array.

(The previously flagged error should disappear as soon as you specify the return value).

4 Add a new public method called **GetNames**, which returns a *List* of *string* variables.

(You'll see an error at this stage as you have not yet created code that returns a value).

5 Create a *List* of *string* variables called **Names** in the *GetNames* method containing the names: "**Mike**", "**Simon**", "**Emily**" and make the method return it.

(The previously flagged error should disappear as soon as you specify the return value).

6 Add a new public method called **ProcessNames**, which doesn't return a value.

7 Create a *List* of *string* variables called **NamesToProcess** in the *ProcessNames* method and populate it with the *List* collection returned by the *GetNames()* method.

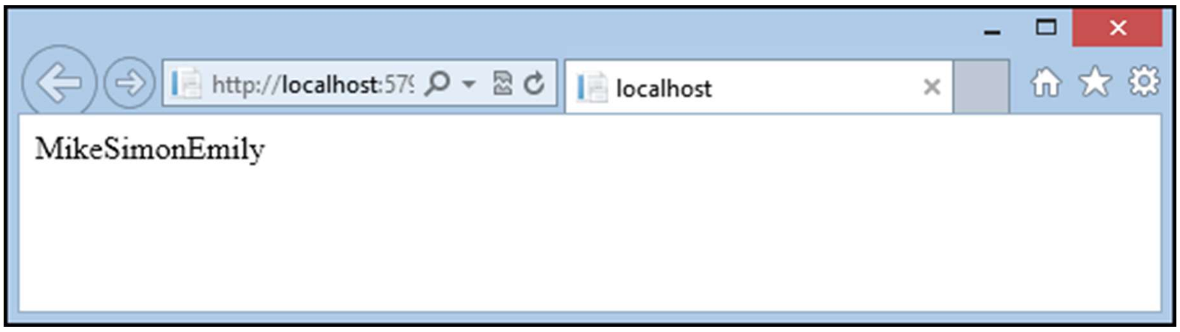
8 Use a *for* loop to loop through the list of names and make each one upper case using the *ToUpper* method of the *string* variable type.

9 Add a new public method called **AppendNames** which returns a *string* value.

(You'll see an error at this stage as you have not yet created code that returns a value).

10 In the new method, add a *foreach* loop which loops through the names returned by the *GetNames* method and appends them all to a single *string* variable. Make the method return the *string*.

11 Add a new page called **test.aspx** and use the *Page_Load* event handler to call the *AppendNames* method of the *MyData* class and display the return value at the top of the web page.



```
MikeSimonEmily
```

My Project - start

My Project - end

If you need help slide the page to the left

Session 8: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 10	Q 8	Q 5	Q 3
Use the following code: **public string AppendNames() { string AppendedNames = ""; foreach (string Name in GetNames()) { AppendedNames = AppendedNames + Name; } return AppendedNames; }** This was covered in: *Lesson 8-3: Iterate through a collection using foreach.*	Use the following code: **public void ProcessNames() { List<string> NamesToProcess = GetNames(); for (int Counter = 0; Counter < NamesToProcess.Count; Counter++) { NamesToProcess [Counter] = NamesToProcess [Counter].ToUpper(); } }** This was covered in: *Lesson 8-4: Iterate through a collection using a for loop.*	Use the following code: **public List<string> GetNames() { List<string> Names = new List<string>(); Names.Add("Mike"); Names.Add("Simon"); Names.Add("Emily"); return Names; }** It is also possible to do this using less code. This was covered in: *Lesson 8-2: Create a collection.*	Use the following code: **public int[] GetNumbers() { int[] Numbers = new int[5]; Numbers[0] = 1; Numbers[1] = 1; Numbers[2] = 3; Numbers[3] = 5; Numbers[4] = 8; return Numbers; }** It is also possible to do this using less code. Both this and the alternative technique were covered in: *Lesson 8-1: Create an array.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 6-1: Create a class.**

2 **Refer to: Lesson 6-7: Create methods that return a value, Lesson 8-1: Create an array.**

4 **Refer to: Lesson 6-7: Create methods that return a value, Lesson 8-2: Create a collection.**

6 **Refer to: Lesson 6-5: Create and use methods.**

7 **Refer to: Lesson 8-2: Create a collection.**

9 **Refer to: Lesson 6-7: Create methods that return a value.**

11 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer, Lesson 6-2: Create an instance of a class, Lesson 3-7: Understand Request and Response.**

Session 9: Exercise

1 Create a new *ASP.NET Web Forms Application* project in your sample files folder, named: **Session9**

2 Start the project in *Debug* mode, view its pages and then close your web browser.

(This is necessary because the project must be built before the *ASP.NET Configuration* utility will work properly. Starting debugging causes the project to be built).

3 Open the *ASP.NET Configuration* utility for your new project.

4 Enable roles for the application.

5 Add a new role called: **Moderator**

6 Add a new folder to the project called: **Moderate**

7 Add a new *aspx* page to the *Moderate* folder called: **default.aspx**

8 Add a *Calendar* control to your new page.

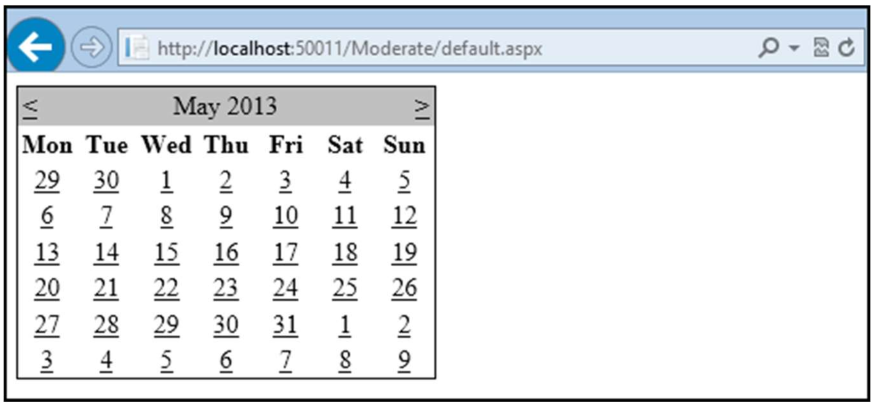
9 Use the *ASP.NET Configuration* utility to add access rules to allow only users with the *Moderator* role to access the *Moderate* folder.

10 Create a new user account and assign it to the *Moderator* role.

11 Attempt to view the new *default.aspx* page in the *Moderate* folder in your browser.

12 Log in when prompted using the user account that you created in step 10.

If all of the above questions were completed correctly you will now see the new *default.aspx* file in the *Moderate* folder.



<			May 2013			>
Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Session9 - end

If you need help slide the page to the left

Session 9: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 10	Q 9	Q 5	Q 3
1. Open the *ASP.NET Configuration* utility (if it isn't open already). 2. Click the *Security* tab. 3. Click *Create user*. 4. Complete the form. 5. Check the *Moderator* box. 6. Click *Create User*. This was covered in: *Lesson 9-1: Use .NET's built-in security features.*	1. Open the *ASP.NET Configuration* utility. 2. Click the *Security* tab. 3. Click *Manage Access Rules*. 4. Click the *Moderate* folder on the left. 5. Click A*dd new access rule*. 6. Click *Allow*. 7. Click *OK*. 8. Click *Add new access rule*. 9. Click *Anonymous Users*. 10. Click Deny. 11. Click *OK*. This was covered in: *Lesson 9-8: Add folder-level security.*	1. Open the *ASP.NET Configuration* utility. 2. Click the *Security* tab. 3. Click *Create or Manage roles*. 4. Type **Moderator** into the *New role name* text box. 5. Click *Add Role*. This was covered in: *Lesson 9-9: Set up roles.*	Click Project→ ASP.NET Configuration. This was covered in: *Lesson 9-2: Manage a site with ASP.NET Configuration.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-5: Create an ASP.NET Web Forms Application project.**

2 **Refer to: Lesson 1-9: Run a project in debug mode.**

4 **Refer to: Lesson 9-9: Set up roles.**

6 **Refer to: Lesson 1-8: Add and remove files from a project.**

7 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

8 **Refer to: Lesson 1-15: Add controls to a page with the Toolbox.**

11 **Refer to: Lesson 9-1: Use .NET's built-in security features.**

12 **Refer to: Lesson 9-1: Use .NET's built-in security features.**

Session 10: Exercise

1 Open the *Session10* project from your sample files folder.

2 Add *LINQ to SQL Classes* to the project. Call the file: **Session10.dbml**

3 Add the *Customer* table from the *Spark* database to the *LINQ to SQL Classes*.

4 Add the *SpGetLastInvoiceNumber* stored procedure from the *Spark* database to the *LINQ to SQL Classes*.

5 Open the code-behind file of *Default.aspx*.

6 Add code to the *Page_Load* event handler to retrieve a *Customer* object with the *CustomerID* of *7* and display the object's *CustomerName* property in the *TextBoxEditCustomerName* control.

7 Add *Click* event handlers to the *ButtonAddCustomer* and *ButtonSaveCustomer* controls on the page.

8 Add code to the *ButtonSaveCustomer_Click* event handler to retrieve the customer with the *CustomerID* of *7* and set its *CustomerName* property to the value entered in the *TextBoxEditCustomer* control.

9 Add code to the *ButtonSaveCustomer_Click* event handler to commit the changes to the *CustomerName* property to the database by calling the *SubmitChanges* method.

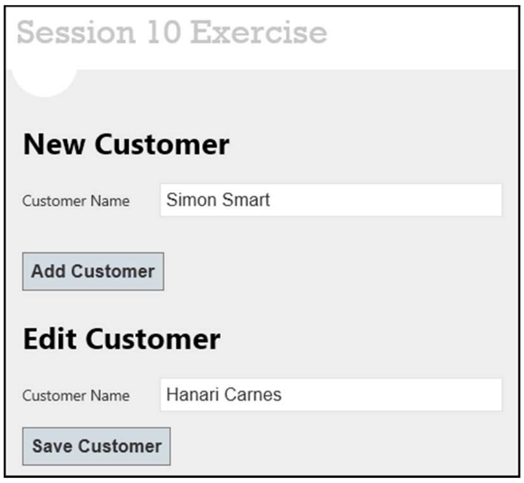
10 Add code to the *ButtonAddCustomer_Click* event handler to add a new record to the *Customer* table in the database.

Set the new record's *CustomerName* property to the value of the *TextBoxNewCustomerName.Text* property.

(Remember to use the *InsertOnSubmit* method before the *SubmitChanges* method).

11 Add *try* and *catch* code to all three event handlers and put the *Message* property of any exceptions into the *LabelError.Text* property.

12 View and test the *Default.aspx* page in your browser.


Session 10 Exercise

New Customer

Customer Name: Simon Smart

Edit Customer

Customer Name: Hanari Carnes

Session10 - start

Session10 - end

If you need help slide the page to the left

Session 10: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 11	Q 10	Q 8	Q 6

1. Enclose your code in the following:
try
{
[Code]
}
2. Add the following:
catch (Exception Ex)
{
LabelError .Text = Ex .Message;
}
This was covered in: *Lesson 7-6: Use try and catch to handle errors.*
	Use the following code:
	using (Session10DataContext Data = new Session10DataContext())
	{
	Customer NewCustomer = new Customer();
	NewCustomer .CustomerName = TextBoxNewCustomer Name.Text;
	Data.Customers .InsertOnSubmit (NewCustomer);
	Data.SubmitChanges();
	}
	This was covered in: *Lesson 10-8: Insert database records using LINQ.*
		Use the following code:
		using (Session10DataContext Data = new Session10DataContext())
		{
		Customer MyCustomer = Data.Customers.Single (Customer => Customer.CustomerID == 7);
		MyCustomer .CustomerName = TextBoxEditCustomer Name.Text;
		}
		This was covered in: *Lesson 10-7: Update database records using LINQ.*
			Use the following code:
			if (!Page.IsPostBack)
			{
			using (Session10DataContext Data = new Session10DataContext())
			{
			Customer MyCustomer = Data.Customers .Single (Customer => Customer .CustomerID == 7);
			TextBoxEditCustomer Name.Text = MyCustomer .CustomerName;
			}
			}
			This was covered in: *Lesson 10-3: Retrieve a single row of data using LINQ.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

2 **Refer to: Lesson 10-2: Add LINQ data classes to a project.**

3 **Refer to: Lesson 10-2: Add LINQ data classes to a project.**

4 **Refer to: Lesson 10-2: Add LINQ data classes to a project.**

5 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

7 **Refer to: Lesson 3-2: Add event handlers to controls.**

9 **Refer to: Lesson 10-7: Update database records using LINQ.**

12 **Refer to: Lesson 1-9: Run a project in debug mode.**

Session 11: Exercise

1 Open the *Spark* project from your sample files folder.

2 Open *customer.aspx* in *Design* view.

3 Add a *LinqDataSource* control and configure it to retrieve records from the *Customer* table, sorted by *CustomerName*. Name your new control: **LinqDataSourceCustomer**

4 Add a *GridView* control and attach it to the *LinqDataSource* control.

5 Enable sorting and paging for the *GridView* control.

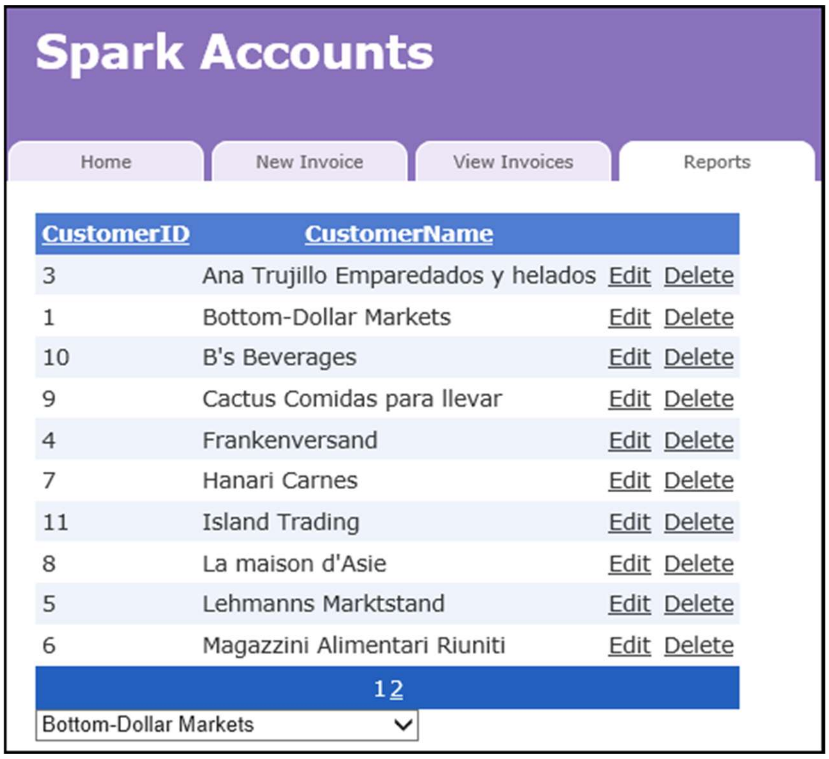
6 Add *Command fields* to the *GridView* control to allow records to be edited and deleted.

7 Use *Auto Format* to make the *GridView* control more presentable.

8 Add a *DropDownList* control to the page. Name your new control: **DropDownListCustomer**

9 Add C# code to the *Page_Load* event handler of *customer.aspx* to retrieve the contents of the *Customer* table and place it in the *DropDownList* control.

10 Set the *DropDownList* control's *DataTextField* property to **CustomerName** and the *DataValueField* property to **CustomerID**.



CustomerID	CustomerName		
3	Ana Trujillo Emparedados y helados	Edit	Delete
1	Bottom-Dollar Markets	Edit	Delete
10	B's Beverages	Edit	Delete
9	Cactus Comidas para llevar	Edit	Delete
4	Frankenversand	Edit	Delete
7	Hanari Carnes	Edit	Delete
11	Island Trading	Edit	Delete
8	La maison d'Asie	Edit	Delete
5	Lehmanns Marktstand	Edit	Delete
6	Magazzini Alimentari Riuniti	Edit	Delete
1 2			

Bottom-Dollar Markets

Spark - start

Spark - end

If you need help slide the page to the left

Session 11: Exercise Answers

These are the three questions that students find the most difficult to answer:

Q 9	Q 6	Q 3

Use the following code:
using (SparkDataContext Data = new SparkDataContext())
{
DropDownListCustomer .DataSource = Data.Customers;
DropDownListCustomer .DataBind();
}
This was covered in: *Lesson 11-8: Bind data to a control using C# code.*
	1. Click *Edit Columns…* in the QuickTasks menu of the *GridView* control.
	2. Expand the *CommandField* category in the *Available Fields* list.
	3. Click *Edit, Update, Cancel* from the *CommandField* category.
	4. Click *Add*.
	5. Click *Delete* from the *CommandField* category.
	6. Click *Add*.
	7. Click *OK*.
	This was covered in: *Lesson 11-5: Add editing features to a GridView.*
		1. Add a *LinqDataSource* control to the page.
		2. Set the *ID* property of the new control to: **LinqDataSourceCustomer**
		3. Click *Configure Data Source…* from the QuickTasks menu of the control.
		4. Ensure that *Spark.SparkDataContext* is selected and click *Next*.
		5. Ensure that *Customers(Table<Customer>)* is selected in the *Table* drop-down.
		6. Click *OrderBy…*
		7. Ensure that *CustomerName* is selected in the *Sort by* drop-down.
		8. Click *OK*.
		9. Click *Finish*.
		This was covered in: *Lesson 11-1: Use the LinqDataSource control.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

2 **Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**

4 **Refer to: Lesson 11-3: Use the GridView control.**

5 **Refer to: Lesson 11-4: Add sorting and paging to a GridView.**

7 **Refer to: Lesson 1-16: Use the QuickTasks menu.**

8 **Refer to: Lesson 1-15: Add controls to a page with the Toolbox.**

Session 12: Exercise

1 Open the *SmartMethodStore* project from your sample files folder.

2 Open *products.aspx* from the *admin* folder.

3 Add a *LinqDataSource* control to the page which retrieves all entries from the *Product* table.

4 Add a *GridView* control and link it to the *LinqDataSource.*

5 Add the ability to update and delete products to the new *GridView* control.

6 Add a *DetailsView* control linked to the same *LinqDataSource* control.

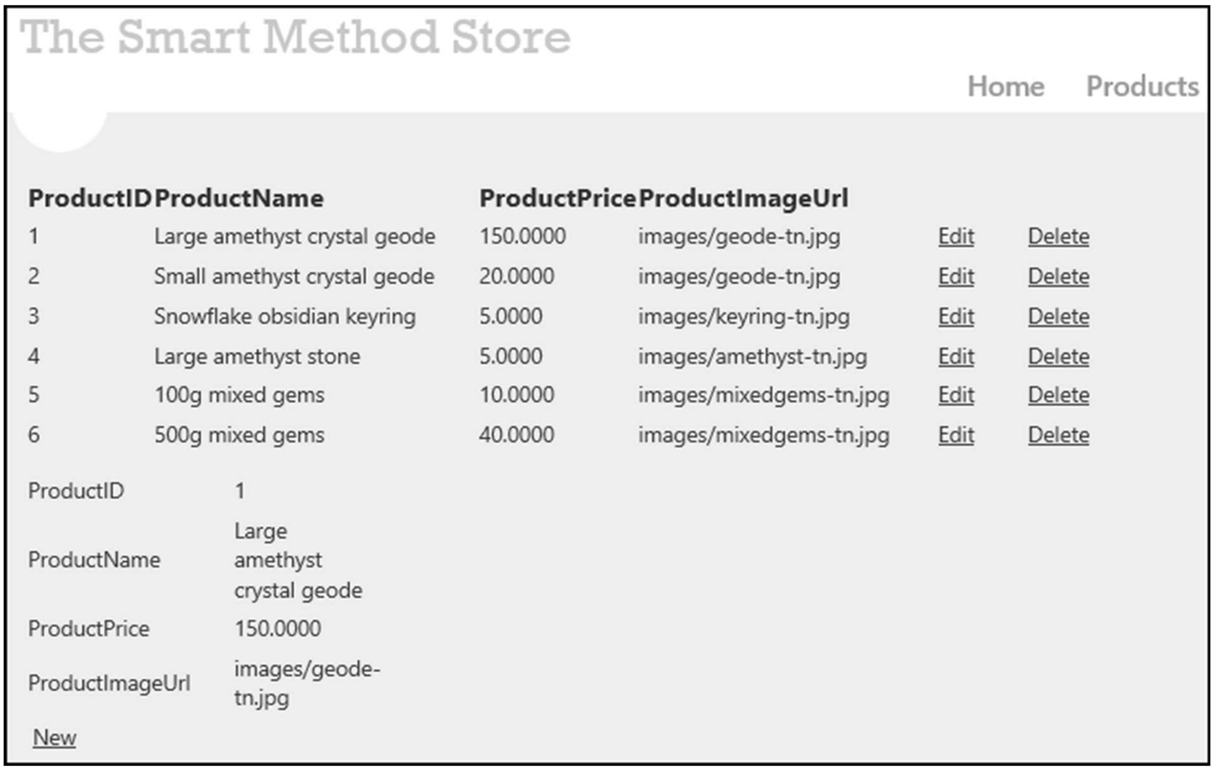
7 Add the ability to insert a new product to the *DetailsView* control.

8 Open *orders.aspx* from the *admin* folder.

9 Add *LinqDataSource* and *GridView* controls to display all records from the *Order* table where *OrderSent* is *false* and *OrderPaid* is *true.*

10 Add a *ButtonField* to the *GridView* control and set its *Text* property to: **Send Order**

11 Add a *RowCommand* event handler to your *GridView* control that will set the selected order's *OrderSent* property to *true* when the *Send Order ButtonField* is clicked.



ProductID	ProductName	ProductPrice	ProductImageUrl		
1	Large amethyst crystal geode	150.0000	images/geode-tn.jpg	Edit	Delete
2	Small amethyst crystal geode	20.0000	images/geode-tn.jpg	Edit	Delete
3	Snowflake obsidian keyring	5.0000	images/keyring-tn.jpg	Edit	Delete
4	Large amethyst stone	5.0000	images/amethyst-tn.jpg	Edit	Delete
5	100g mixed gems	10.0000	images/mixedgems-tn.jpg	Edit	Delete
6	500g mixed gems	40.0000	images/mixedgems-tn.jpg	Edit	Delete

ProductID	1
ProductName	Large amethyst crystal geode
ProductPrice	150.0000
ProductImageUrl	images/geode-tn.jpg
New	

SmartMethodStore - start

SmartMethodStore - end

If you need help slide the page to the left

Session 12: Exercise Answers

These are the four questions that students find the most difficult to answer:

Q 11	Q 9	Q 7	Q 5
1. Add a *RowCommand* event handler to your *GridView* control. 2. Add the following code: **int RowClicked = Convert.ToInt32 (e.CommandArgument);** **int OrderID = Convert.ToInt32 (GridViewOrder.DataKeys[RowClicked].Value);** **using (StoreDataContext Data = new StoreDataContext()) { Order OrderToSend = Data.Orders .Single(Order => Order.OrderID == OrderID); OrderToSend .OrderSent = true; Data.SubmitChanges(); } GridViewOrder.DataBind();** This was covered in: *Lesson 12-6: Create a Products page.*	1. Add a new *LinqDataSource* to the page. 2. Click *Configure Data Source* from the *QuickTasks* menu of the *LinqDataSource*. 3. Click *Next*. 4. Choose *Orders* from the *Table* drop-down. 5. Click *Where…* 6. Choose *OrderSent* from the *Column* drop-down. 7. Choose == from the *Operator* drop-down. 8. Choose *None* from the *Source* drop-down. 9. Type **False** into the *Value* box. 10. Click *Add* and repeat the process for the *OrderPaid* property with a value of: **True** 12. Add a *GridView* control and link it to the *LinqDataSource*. This was covered in: *Lesson 11-1: Use the LinqDataSource control.*	1. Open the *Edit Columns* dialog from the *QuickTasks* menu of the *DetailsView* control. 2. Add a *New, Insert, Cancel* field from the *CommandField* category. 3. Click OK. 4. Set the *EnableInsert* property of your *LinqDataSource* to: **True** This was covered in: *Lesson 11-6: Use the DetailsView control.*	1. Open the *Edit Columns* dialog from the *QuickTasks* menu of the *GridView* control. 2. Add an *Edit, Update, Cancel* field from the *CommandField* category. 3. Add a *Delete* field from the *CommandField* category. 4. Set the *EnableUpdate* and *EnableDelete* properties of your *LinqDataSource* to: **True** This was covered in: *Lesson 11-5: Add editing features to a GridView.*

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

1	**Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**
2	**Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**
3/4	**Refer to: Lesson 11-3: Use the GridView control.**
6	**Refer to: Lesson 11-6: Use the DetailsView control.**
8	**Refer to: Lesson 1-7: Manage a project with the Solution Explorer.**
10	**Refer to: Lesson 12-6: Create a Products page.**