

---

## Session 1: Exercise

- 1 Open *Exercise1* from the *Exercises* folder in your sample files folder.
- 2 Attempt to build the project. A build error will occur.
- 3 Use the *Error List* window to find and fix the build error.
- 4 Switch to the *Release* build profile and build the project.
- 5 Add a reference to the *AdvancedMath.dll* file from your sample files folder (click *Yes* when prompted to confirm as you'll change the ASP.NET version later in this exercise).
- 6 Enable XML Documentation for the *Release* build profile.
- 7 Switch back to the *Debug* profile and completely rebuild the project.
- 8 The project is currently set to use ASP.NET version 3.5. Change the project's settings so that it uses ASP.NET version 4.0.
- 9 Open the code-behind file of *Default.aspx*.
- 10 Use *Surround With* to add a *try, catch* statement around the entire contents of the *ButtonCalculate\_Click* event handler.
- 11 Use the *Extract Method* refactoring utility to extract the code inside the comment markers into a new method named: **AddRowToTable**
- 12 Set a breakpoint at the start of the *for* loop, start the project in *Debug* mode and click the *Calculate* button.  
Your code is paused at the breakpoint.
- 13 Use the *Immediate* window to change the value of the *NumberOfYears* variable to: **10**
- 14 Remove the breakpoint and resume debugging.

Year	Value
0	3217.89
1	1608.95
2	804.47
3	402.24
4	201.12
5	100.56
6	50.28
7	25.14
8	12.57
9	6.28
10	3.14

Exercise1 - Start

Exercise1 - End

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 13	Q 11	Q 10	Q 3
<p>1. Click inside the <i>Immediate</i> window at the bottom of the Visual Studio screen.</p> <p>2. Type: <b>NumberOfYears = 10</b></p> <p>3. Press the &lt;Enter&gt; key.</p> <p>This was covered in: <i>Lesson 1-11: Use the Immediate window.</i></p>	<p>1. Click and drag to highlight all of the code between the comment markers.</p> <p>2. Right-click the highlighted code and click Refactor→ Extract Method... from the shortcut menu.</p> <p>3. Type the name: <b>AddRowToTable</b> and click OK.</p> <p>This was covered in: <i>Lesson 1-14: Use the Extract Method refactoring utility.</i></p>	<p>1. Click and drag to highlight all code inside the <i>ButtonCalculate_Click</i> event handler.</p> <p>2. Right-click the highlighted code and click <i>Surround With...</i> from the shortcut menu.</p> <p>3. Find <i>try</i> in the list and double-click it.</p> <p>This was covered in: <i>Lesson 1-13: Use Surround With.</i></p>	<p>1. Double-click the error in the <i>Error List</i> window.</p> <p>The code-behind file of <i>Default.aspx</i> is opened and the error is highlighted.</p> <p>The problem is that <i>TextBoxNumberOfYears</i> has been misspelled.</p> <p>2. Change <i>TextBoxNumberOfYear</i> to <b>TextBoxNumberOfYears</b>.</p> <p>This was covered in: <i>Lesson 1-9: Use the Error List window.</i></p>

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

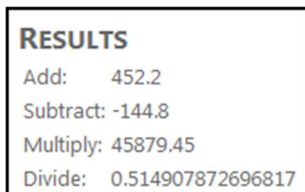
- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Lesson 1-4: Build, Rebuild and Clean a project.
- 4** Refer to: Lesson 1-3: Use the Debug and Release profiles.
- 5** Refer to: Lesson 1-2: Add references to a project.
- 6** Refer to: Lesson 1-8: Create an XML documentation file.
- 7** Refer to: Lesson 1-3: Use the Debug and Release profiles, Lesson 1-4: Build, Rebuild and Clean a project.
- 8** Refer to: Lesson 1-7: Change the version of ASP.NET used by the project.
- 9** Refer to: Essential Skills Session 1.
- 12** Refer to: Essential Skills Session 3.
- 14** Refer to: Essential Skills Session 3.

---

## Session 2: Exercise

- 1 Open *Exercise2* from your sample files folder.
- 2 Open *MultiMath.aspx* in *Design* view.

This page displays the results of adding, subtracting, multiplying and dividing the numbers entered in the text boxes.
- 3 Open the code-behind file of *MultiMath.aspx*.
- 4 Create a new method called *DoAddSubtractMultiplyDivide*. The new method should return a *Tuple* with four *double* properties. It should accept two *double* arguments named *Number1* and *Number2*.
- 5 Add code to the method to add, subtract, multiply and divide *Number1* and *Number2* and return a *Tuple* containing the results of each calculation.
- 6 Add code to the *ButtonDoMath\_Click* event handler to call the *DoAddSubtractMultiplyDivide* method and display the results in the *Label* controls on the page.



**RESULTS**  
Add: 452.2  
Subtract: -144.8  
Multiply: 45879.45  
Divide: 0.514907872696817

- 7 Open *Upload.aspx* in *Design* view.

This page allows you to upload files, which should then be stored in the database.
- 8 Open the code-behind file of *Upload.aspx*.
- 9 Add code to the *ButtonUploadFile\_Click* event handler to retrieve a *byte* array from the *FileUploadFileToUpload* control. Note that you can use the *FileBytes* property of the control to do this.
- 10 Add code to make the *Data* object lazily instantiated.
- 11 Add code to the *ButtonUploadFile\_Click* event handler to set the properties of the *NewFile* object using your *byte* array and the *FileName* property of the *FileUpload* control.
- 12 Add code to tell the Garbage Collector to clean up memory at the end of the event handler.
- 13 Open *Grid.aspx* in *Design* view.

This page will populate the *Table* control's cells with random background colors.
- 14 Open the code-behind file of *Grid.aspx*.
- 15 Add code to the *ButtonGetColors\_Click* event handler to create a 3x3 array of *Color* objects named: **ColorArray**
- 16 Add code to cycle through your array and populate each element using the *GetRandomColor* method.
- 17 Add code to cycle through your array and set the *BackColor* property of each cell in the *TableGrid* table with the value from your array. Do this using the following property:  
*TableGrid.Rows[X].Cells[Y].BackColor*

Exercise2 - Start

Exercise2 - 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 17	Q 10	Q 9	Q 4
<p>Use the following code:</p> <pre>for (int X = 0; X &lt; 3; X++) {     for (int Y = 0; Y &lt; 3; Y++)     {         TableGrid         .Rows[X].Cells         [Y].BackColor =         ColorArray[X, Y];     } }</pre> <p>This was covered in: <i>Lesson 2-7: Iterate through a multidimensional array.</i></p>	<p>Use the following code:</p> <pre>Lazy &lt;Exercise2Data Context&gt; Data = new Lazy &lt;Exercise2Data Context&gt;();</pre> <p>This was covered in: <i>Lesson 2-2: Use the Lazy class.</i></p>	<p>Use the following code:</p> <pre>byte[] FileBytes = FileUploadFileToUpload .FileBytes;</pre> <p>This was covered in: <i>Lesson 2-4: Use the Byte class.</i></p>	<p>Use the following code:</p> <pre>private Tuple &lt;double,double, double,double&gt; DoAddSubtract MultiplyDivide (double Number1, double Number2) { }</pre> <p>This was covered in: <i>Lesson 2-1: Use the Tuple class.</i></p>

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

- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Essential Skills Session 1.
- 3** Refer to: Essential Skills Session 1.
- 5** Refer to: Lesson 2-1: Use the Tuple class.
- 6** Refer to: Lesson 2-1: Use the Tuple class.
- 7** Refer to: Essential Skills Session 1.
- 8** Refer to: Essential Skills Session 1.
- 11** Refer to: Essential Skills Session 10 (particularly Lesson 10-8).
- 12** Refer to: Lesson 2-8: Send commands to the Garbage Collector.
- 13** Refer to: Essential Skills Session 1.
- 14** Refer to: Essential Skills Session 1.
- 15** Refer to: Lesson 2-7: Iterate through a multidimensional array.
- 16** Refer to: Lesson 2-7: Iterate through a multidimensional array.

---

## Session 3: Exercise

- 1 Open *Exercise3* from your sample files folder.
- 2 Open *SecretFiles.aspx* in Design view.

This page shows a list of files stored in the project's database and allows you to view them.

When it is finished, you will be able to encrypt and decrypt every file in the database by clicking the buttons.
- 3 Open the code-behind file of *SecretFiles.aspx*.

Most of the code is already in place. All that is needed is to complete the *EncryptBinary* and *DecryptBinary* methods.
- 4 Add *using* lines for the *System.Security.Cryptography*, *System.IO* and *System.Threading* namespaces.
- 5 Add code to the *EncryptBinary* method to create a new *DES* object from the *System.Cryptography* namespace (you'll need to use the *DES.Create* method to do this).

The *DES* object represents the *Data Encryption Standard*. It works identically to the *Aes* object you used in this session.
- 6 Add code to set the *Key* and *IV* properties of your *DES* object to the values provided in the *EncryptionKey* and *InitializationVector* properties.
- 7 Add code to create a *MemoryStream* object to receive the encrypted data. Name it: **EncryptedStream**
- 8 Add code to create a *CryptoStream* object to encrypt the data. Remember to use the *CreateEncryptor* method of your *DES* object when supplying the arguments.

Ensure that a *using* statement is in place to automatically dispose of the *CryptoStream* object.
- 9 Add code to tell the *CryptoStream* object to *Write* the binary data that you want to encrypt.
- 10 Return the encrypted data using the *ToArray* method of the *EncryptedStream* object.
- 11 Copy and paste your code from the *EncryptBinary* method to the *DecryptBinary* method and modify it to decrypt the binary data provided to it.
- 12 Modify the code in the *ButtonEncryptDatabase\_Click* event handler so that each call to the *EncryptSecretFile* method is handled by its own thread (use the *ThreadPool* class).
- 13 Add an *int* property to track how many files have been encrypted and add a *while* loop to the *ButtonEncryptDatabase\_Click* event handler to wait until it matches the value returned by *FilesToEncrypt.Count()*.
- 14 Add a *lock* statement to the *EncryptSecretFile* method to prevent two threads from calling *Data.SubmitChanges* at the same time.

Exercise3 - Start

Exercise3 - 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 14	Q 13	Q 12	Q 8, 9, 10
<p>1. Add the following code outside any methods:</p> <pre><b>object Locker = new object();</b></pre> <p>2. In the <i>EncryptSecretFile</i> method, replace the line:</p> <pre><i>Data.SubmitChanges();</i></pre> <p>...with:</p> <pre><b>lock (Locker)</b> {     <i>Data</i>     .<i>SubmitChanges();</i> }</pre> <p>This was covered in: <i>Lesson 3-7: Use the lock statement to prevent threads from conflicting.</i></p>	<p>1. Add the following code outside any methods:</p> <pre><b>int FilesEncrypted = 0;</b></pre> <p>2. Add the following code to the end of the <i>EncryptSecretFile</i> method:</p> <pre><b>FilesEncrypted++;</b></pre> <p>3. Add the following code to the end of the <i>using</i> statement in the <i>ButtonEncryptDatabase_Click</i> event handler:</p> <pre><b>while (FilesEncrypted &lt; FilesToEncrypt.Count())</b> {     <b>Thread.Sleep(1000);</b> }</pre> <p>This was covered in: <i>Lesson 3-6: Use the ThreadPool class to manage multiple threads.</i></p>	<p>Replace the line:</p> <pre><i>EncryptSecretFile</i> (<i>FileToEncrypt</i> .<i>SecretFileID</i>);</pre> <p>...with:</p> <pre><b>ThreadPool</b> .<b>QueueUserWorkItem</b> (<b>new WaitCallback</b> (<b>EncryptSecretFile</b>), <b>FileToEncrypt</b> .<b>SecretFileID</b>);</pre> <p>This was covered in: <i>Lesson 3-6: Use the ThreadPool class to manage multiple threads.</i></p>	<p>Use the following code:</p> <pre><b>using (CryptoStream</b> <b>EncryptionStream =</b> <b>new CryptoStream</b> (<b>EncryptedStream</b>, <b>DESAlgorithm</b>. <b>CreateEncryptor()</b>, <b>CryptoStreamMode.Write</b>)) {     <b>EncryptionStream.Write</b> (<b>BinaryToEncrypt</b>, 0, <b>BinaryToEncrypt</b> .<b>Length</b>); }</pre> <pre><b>return EncryptedStream.</b> <b>ToArray();</b></pre> <p>This was covered in: <i>Lesson 3-2: Encrypt data using the AES standard.</i></p>

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

- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Essential Skills Session 1.
- 3** Refer to: Essential Skills Session 1.
- 4** Refer to: Essential Skills Session 6.
- 5** Refer to: Lesson 3-2: Encrypt data using the AES standard.
- 6** Refer to: Lesson 3-2: Encrypt data using the AES standard.
- 7** Refer to: Lesson 3-2: Encrypt data using the AES standard.
- 11** Refer to: Lesson 3-3: Decrypt data using the AES standard.

---

## Session 4: Exercise

- 1 Open *Exercise4* from your sample files folder.
- 2 Open *MailSender.cs*.  
This is a simple class that sends e-mail messages.
- 3 Add an overload to the *SendEmail* method which makes the *BodyText* value into an argument.
- 4 Make the *ToAddress* argument of the first overload optional by giving it a default value of: **info@ASPNETCentral.com**
- 5 Make the first *SendEmail* overload *virtual*.
- 6 Create a new class called: **SalesMailSender.cs**
- 7 Make the new *SalesMailSender* class *extend* the *MailSender* class.
- 8 Override the *SendEmail* method in the *SalesMailSender* class and make the new method use the body text: **Thank you for your purchase.**
- 9 Create another new class called: **MarketingMailSender.cs**
- 10 Make the new *MarketingMailSender* class *extend* the *MailSender* class.
- 11 Add a new method to the *MarketingMailSender* class named: **SendPromotionalEmail**
- 12 Add code to the *SendPromotionalEmail* method that will send an e-mail message with the body text: **Here are some products that may interest you.**  
You can do this by copying and pasting the code from one of the other methods.
- 13 Create a new *interface* named: **IMailSender.cs**
- 14 Add a method to your new *interface* named: **SendEmail**
- 15 Ensure that the *SendEmail* method in the *IMailSender* interface returns *void* and requires a single *string* argument.
- 16 Make the *MailSender* class implement the *IMailSender* interface.

Exercise4 - Start

Exercise4 - 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 14 & 15	Q 8	Q 4	Q 3
<p>Use the following code:</p> <pre>void SendEmail( string FromAddress );</pre> <p>This was covered in: <i>Lesson 4-16: Create an interface.</i></p>	<p>Use the following code:</p> <pre>public override void SendEmail(string ToAddress = "info@ASPNETCentral.com") {     using (System.Net.Mail. SmtplibClient Client = new System.Net.Mail. SmtplibClient())     {         string Subject = "Exercise 4";         string BodyText = "Thank you for your purchase.";         Client.Send( "info@ASPNETCentral.com", ToAddress, Subject, BodyText);     } }</pre> <p>This was covered in: <i>Lesson 4-13: Understand the virtual and override keywords.</i></p>	<p>Change the line:</p> <pre>public virtual void SendEmail(string ToAddress) ...to: public virtual void SendEmail( string ToAddress = "info@ASPNET Central.com")</pre> <p>This was covered in: <i>Lesson 4-7: Add optional arguments to a method.</i></p>	<p>Add the following code below the original <i>SendEmail</i> method (copying and pasting will speed this up):</p> <pre>public void SendEmail( string ToAddress, string BodyText) {     using (System.Net.Mail. SmtplibClient Client = new System.Net.Mail. SmtplibClient())     {         string Subject = "Exercise 4";         Client.Send( "info@ ASPNETCentral.com", ToAddress, Subject, BodyText);     } }</pre> <p>This was covered in: <i>Lesson 4-6: Overload a method.</i></p>

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

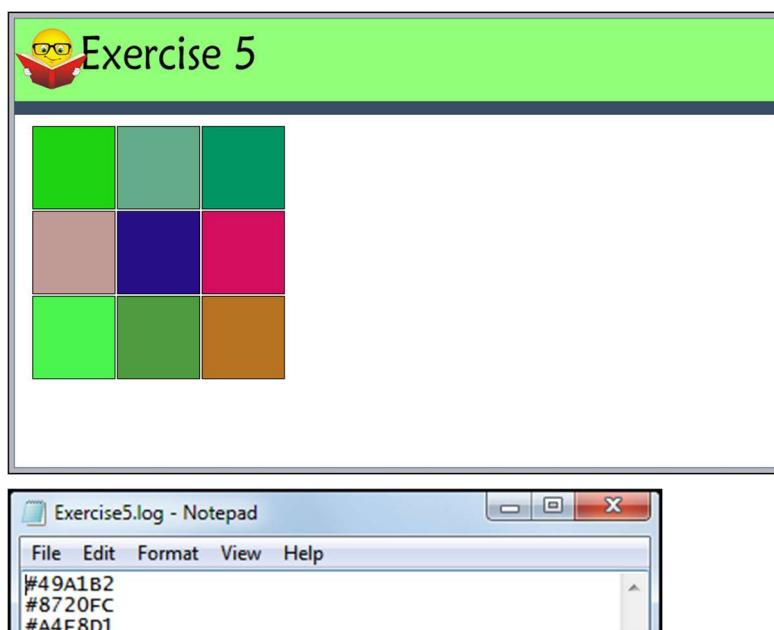
- 1, 2** Refer to: Essential Skills Session 1.
- 5** Refer to: Lesson 4-13: Understand the virtual and override keywords.
- 6** Refer to: Essential Skills Session 6.
- 7** Refer to: Lesson 4-10: Extend a class.
- 9** Refer to: Essential Skills Session 6.
- 10** Refer to: Lesson 4-10: Extend a class.
- 11** Refer to: Essential Skills Session 6.
- 12** Refer to: Lesson 2-9: Send e-mail messages using the *SmtplibClient* class.
- 13** Refer to: Lesson 4-16: Create an interface.
- 16** Refer to: Lesson 4-15: Implement an interface.



---

## Session 5: Exercise

- 1 Open *Exercise5* from your sample files folder.
- 2 Open the code-behind file of *Site.Master*.
- 3 Add an *#if* directive to the *Page\_Load* event handler to display *DEBUG* in the *LabelCompileType* control if the *DEBUG* constant is found.
- 4 Add an *#elif* directive to display *FINAL VERSION* in the *LabelCompileType* control if a constant named *FINALVERSION* is found.
- 5 Add an *#else* directive to display *STANDARD* in the *LabelCompileType* control if neither *DEBUG* nor *FINALVERSION* constants are found.
- 6 Use the *#define* directive to define the *FINALVERSION* constant on this page.
- 7 Open the code-behind file of *RandomColors.aspx*.
- 8 Use the *#region* directive to enclose the *Page\_Load* method in a region named: **Event Handlers**
- 9 Use the *#region* directive to enclose the *ColorTable* and *GetRandomColor* methods in a region named: **Random Color Generator**
- 10 Add a *using* line for: **System.Diagnostics**
- 11 Add code to the end of the *GetRandomColor* method to output the *ColorCode* variable using the *Debug.WriteLine* method.
- 12 Open *RandomColors.aspx* in Design view and enable tracing.
- 13 Open *Web.config* and add a *TextWriterTraceListener* to log debug information to:  
**C:\Practice\ASP.NET Expert\Exercise5.log**



Exercise5 - Start

Exercise5 - 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 11	Q 8	Q 6	Q 3
<p>Add the following line of code on the line after <i>string ColorCode...</i>:</p> <pre>Debug.WriteLine( ColorCode);</pre> <p>This was covered in: <i>Lesson 5-8: Use System.Diagnostics to write Debug messages.</i></p>	<p>Enclose the <i>Page_Load</i> event handler in the tags:</p> <pre>#region Event Handlers ...and: #endregion</pre> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <pre>#region Event Handlers protected void Page_Load(object { } #endregion</pre> </div> <p>This was covered in: <i>Lesson 5-1: Create code regions with the #region directive.</i></p>	<p>Add the following code to the very top of the page, before the <i>using</i> lines:</p> <pre>#define FINALVERSION</pre> <p>This was covered in: <i>Lesson 5-4: Use the #define directive to create a new constant.</i></p>	<p>Use the following code:</p> <pre>#if DEBUG LabelCompileType .Text = "DEBUG"; #endif</pre> <p>This was covered in: <i>Lesson 5-2: Use the #if directive to selectively compile code.</i></p>

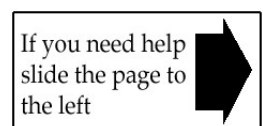
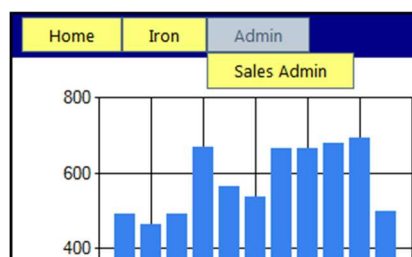
If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Essential Skills Session 1.
- 4** Refer to: Lesson 5-5: Use the #else and #elif directives.
- 5** Refer to: Lesson 5-5: Use the #else and #elif directives.
- 7** Refer to: Essential Skills Session 1.
- 9** Refer to: Lesson 5-1: Create code regions with the #region directive.
- 10** Refer to: Lesson 5-8: Use System.Diagnostics to write Debug messages.
- 12** Refer to: Lesson 5-10: Enable tracing.
- 13** Refer to: Lesson 5-9: Use a listener to save diagnostic information to a file.

---

## Session 6: Exercise

- 1 Open *Exercise6* from your sample files folder and open *SalesAdmin.aspx* in Design view.
- 2 In the *GridView* control, replace the *Product* field with a *TemplateField* called: **Product**
- 3 Add a *Label* control to the *ItemTemplate* of your new *TemplateField* and configure it to display the *Product.ProductName* data field using the *Eval* method.
- 4 In the *EditItemTemplate* of your *TemplateField*, add a *LinqDataSource* control that retrieves all records from the *Product* table. Set an appropriate ID property for the control for easy identification.
- 5 Add a *DropDownList* control to the *EditItemTemplate* and link it to the *LinqDataSource* control you created in the previous step.
- 6 Configure your *DropDownList* control to use *ProductName* as its display field and *ProductID* as its value field.
- 7 Bind the *DropDownList* control to the *ProductID* data field using the *Bind* method.
- 8 Open *IronData.aspx* in Design view and add a *ListView* control to the page, linked to the *LinqDataSourceIronSales* data source.
- 9 Generate templates for your new *ListView* control using the *Configure ListView* option with the default settings.
- 10 Open *IronChart.aspx* in Design view and add a *Chart* control to the page, linked to the *SqlDataSourceIronSales* data source.
- 11 In the *QuickTasks* menu of the *Chart* control, set the *X Value Member* property to **SaleMonth** and the *Y Value Member* to **SaleValue**.
- 12 Open the *Web.sitemap* file and add a new pair of *siteMapNode* tags (similar to, and at the same level as, the *Iron* tag) with a *title* property of: **Admin**
- 13 Add a new *siteMapNode* tag inside the *Admin* tag with a *url* property of: **SalesAdmin.aspx** and a *title* property of: **Sales Admin**
- 14 Open *Site.Master* in Design view and add a *SiteMapDataSource* control in the blue area near the top of the page.
- 15 Configure the new *SiteMapDataSource* control to hide the starting node.
- 16 Add a *Menu* control in the same area and link it to the *SiteMapDataSource* control.
- 17 Set the *CssClass* property of the new *Menu* control to: **menu**
- 18 Configure the *Menu* control to display horizontally instead of vertically.
- 19 Configure the *Menu* control not to generate any CSS code of its own.



## Session 6: Exercise Answers

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

Q 12 & 13	Q 7	Q 3	Q 2
<p>Use the following code:</p> <pre>&lt;siteMapNode title="Admin"&gt;   &lt;siteMapNode url="SalesAdmin.aspx" title="Sales Admin" /&gt; &lt;/siteMapNode&gt;</pre> <p>This was covered in: <i>Lesson 6-14: Add a sitemap file to a project.</i></p>	<ol style="list-style-type: none"> <li>1. Open the <i>QuickTasks</i> menu of the <i>DropDownList</i> control and click <i>Edit Databindings...</i></li> <li>2. Click the <i>Bound to</i> drop-down menu and select <i>ProductID</i>.</li> <li>3. Click <i>OK</i>.</li> </ol> <p>This was covered in: <i>Lesson 6-9: Use the GridView EditItemTemplate.</i></p>	<ol style="list-style-type: none"> <li>1. Open the <i>QuickTasks</i> menu of the <i>GridView</i> control and click <i>Edit Templates</i>.</li> <li>2. Drag a <i>Label</i> control into the <i>ItemTemplate</i> area.</li> <li>3. Open the <i>QuickTasks</i> menu of the <i>Label</i> control and click <i>Edit DataBindings...</i></li> <li>4. Click <i>Custom binding</i>.</li> <li>5. In the <i>code expression</i> box, type: <b>Eval("Product.ProductName")</b></li> <li>6. Click <i>OK</i>.</li> </ol> <p>This was covered in: <i>Lesson 6-8: Use the GridView TemplateField.</i></p>	<ol style="list-style-type: none"> <li>1. Open the <i>QuickTasks</i> menu of the <i>GridView</i> control and click <i>Edit Columns...</i></li> <li>2. Click the <i>Product</i> column in the <i>Selected fields</i> pane and click the delete button.</li> <li>3. Click <i>TemplateField</i> in the <i>Available fields</i> pane and click <i>Add</i>.</li> <li>4. Set the <i>HeaderText</i> property to: <b>Product</b></li> <li>5. Click <i>OK</i>.</li> </ol> <p>This was covered in: <i>Lesson 6-8: Use the GridView TemplateField.</i></p>

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

- 1** Refer to: *Essential Skills Session 1*.
- 4** Refer to: *Lesson 6-9: Use the GridView EditItemTemplate*.
- 5** Refer to: *Lesson 6-9: Use the GridView EditItemTemplate*.
- 6** Refer to: *Lesson 6-9: Use the GridView EditItemTemplate*.
- 8** Refer to: *Lesson 6-10: Create a ListView control*.
- 9** Refer to: *Lesson 6-10: Create a ListView control*.
- 10** Refer to: *Lesson 6-4: Create a Chart control*.
- 11** Refer to: *Lesson 6-4: Create a Chart control*.
- 14** Refer to: *Lesson 6-16: Use the TreeView and SiteMapDataSource controls*.
- 15** Refer to: *Lesson 6-16: Use the TreeView and SiteMapDataSource controls*.
- 16** Refer to: *Lesson 6-17: Use the Menu control*.
- 17** Refer to: *Lesson 6-17: Use the Menu control*.
- 18** Refer to: *Lesson 6-17: Use the Menu control*.
- 19** Refer to: *Lesson 6-17: Use the Menu control*.

---

## Session 7: Exercise

- 1 Open *Exercise7* from your sample files folder.
- 2 Open the ASP.NET Configuration utility and add a new application setting with the *Name*: **ApplicationName** and the *Value*: **Exercise7**
- 3 Create a new *Web User Control* named: **ApplicationName.ascx**
- 4 Add a *Label* control to your new *Web User Control* with the *ID*: **LabelApplicationName**
- 5 Open the code-behind file of your *Web User Control* and add code to the *Page\_Load* event handler to retrieve the *ApplicationName* application setting and display its value in the *LabelApplicationName* control.  
  
To retrieve application settings, use the *System.Configuration.ConfigurationManager* class.
- 6 Open *Default.aspx* in Design view and add your *Web User Control* to the page.
- 7 Add a new *skin* file to the project, named: **VioletGridViews.skin**
- 8 Add code to your new *skin* file to set the *BackColor* property of *GridView* controls to: **Violet**
- 9 Open *Sator.aspx* and apply the *VioletGridViews* theme to the page.
- 10 Open *LuckyNumber.aspx* and enable caching on the page with a duration of 3600 seconds.
- 11 Add a new HTTP Handler (also called an ASP.NET Handler) to the project, named: **FeaturesHandler.cs**
- 12 Add code to your HTTP Handler to look for the word *LuckyNumber* using the *context.Request.Path.Contains()* method and redirect the user to *LuckyNumber.aspx* if it is found.
- 13 Add a *Web.config* file to the *Features* folder and add code to apply the *FeaturesHandler* HTTP handler to the folder.
- 14 Add a new HTTP Module (also called an ASP.NET Module) to the project, named: **UserLogModule.cs**
- 15 Create a new method in your HTTP Module with the following code:  

```
public void LogUser(object sender, EventArgs args)
{
}
```
- 16 Add code to the *LogUser* method to retrieve the *HttpApplication* object from the *sender* argument.
- 17 Add code to the *LogUser* method to log the *HttpApplication.User.Identity.Name* property using the *System.Diagnostics.Debug.WriteLine* method.
- 18 Remove any existing code from the *Init* method and then add code to attach your *LogUser* method to the *context.AuthorizeRequest* event.
- 19 Add code to the project's main *Web.config* file to apply the HTTP Module to the project.

Exercise7 - Start

Exercise7 - 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 16, 17, 18	Q 13	Q 12	Q 5
<p>1. Add the following code to the <i>LogUser</i> method:</p> <pre><b>HttpApplication</b> <b>CurrentApplication =</b> <b>(HttpApplication)sender;</b></pre> <p>2. Add the following code on the next line:</p> <pre><b>System.Diagnostics.Debug.</b> <b>WriteLine</b> <b>(CurrentApplication.User.</b> <b>Identity.Name);</b></pre> <p>3. Add the following code to the <i>Init</i> method:</p> <pre><b>context.AuthorizeRequest</b> <b>+= new EventHandler</b> <b>(LogUser);</b></pre> <p>This was covered in: <i>Lesson 7-7: Create an HTTP Module.</i></p>	<p>Add the following code to the <code>&lt;system.web&gt;</code> tag in your new <i>Web.config</i> file:</p> <pre><b>&lt;httpHandlers&gt;</b> <b>&lt;add verb="*"</b> <b>path="*"</b> <b>type=</b> <b>"Exercise7</b> <b>.FeaturesHandler"</b> <b>/&gt;</b> <b>&lt;/httpHandlers&gt;</b></pre> <p>This was covered in: <i>Lesson 7-6: Apply an HTTP Handler to a folder.</i></p>	<p>There are a number of ways you could accomplish this, but this simple code will suffice:</p> <pre><b>if (context.Request.</b> <b>Path.Contains</b> <b>("LuckyNumber"))</b> <b>{</b> <b>context.Response.</b> <b>Redirect</b> <b>("~/</b> <b>LuckyNumber.aspx</b> <b>");</b> <b>}</b></pre> <p>This was covered in: <i>Lesson 7-5: Create an HTTP Handler.</i></p>	<p>Use the following code:</p> <pre><b>LabelApplicationName.</b> <b>Text =</b> <b>System.Configuration.</b> <b>ConfigurationManager.</b> <b>AppSettings</b> <b>["ApplicationName"];</b></pre> <p>This was covered in: <i>Lesson 7-1: Work with application settings.</i></p>

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

- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Lesson 7-1: Work with application settings.
- 3** Refer to: Lesson 7-4: Create a Web User Control.
- 4** Refer to: Lesson 7-4: Create a Web User Control.
- 6** Refer to: Lesson 7-4: Create a Web User Control.
- 7** Refer to: Lesson 7-2: Create a skin.
- 8** Refer to: Lesson 7-2: Create a skin.
- 9** Refer to: Lesson 7-2: Create a skin.
- 10** Refer to: Lesson 7-9: Enable caching.
- 11** Refer to: Lesson 7-5: Create an HTTP Handler.
- 14** Refer to: Lesson 7-7: Create an HTTP Module.
- 15** Refer to: Lesson 7-7: Create an HTTP Module.
- 19** Refer to: Lesson 7-8: Implement custom security with an HTTP Module.

---

## Session 8: Exercise

- 1 Open *Exercise8* from your sample files folder.
- 2 Add a new web service to the project, named: **TestService.asmx**
- 3 Add a new Service Reference to the project, connecting to *TestService.asmx*. Name it: **TestServiceReference**
- 4 Open *Time.aspx* in Design view.
- 5 Add a *ScriptManager* control to the page and name it: **ScriptManagerTime**
- 6 Add an *UpdatePanel* control to the page and name it: **UpdatePanelTime**
- 7 Add a *Label* control inside the *UpdatePanel* control and name it: **LabelTime**
- 8 Add a *Timer* control inside the *UpdatePanel* control and name it: **TimerTime**
- 9 Set the *Interval* property of the new *Timer* control to: **1000**
- 10 Set the *Triggers* property of the *UpdatePanel* control to refresh when the *Tick* event of the *TimerTime* control is fired.
- 11 Add a *Tick* event handler to the new *Timer* control with the following code:  
**LabelTime.Text = GetTime();**
- 12 Open the code-behind file of *TimeManual.aspx* and make the *GetTime* method into a web method.
- 13 Open *TimeManual.aspx* in Design view and configure the *ScriptManager* control to enable Page Methods.
- 14 Switch to *Source* view and add code to the *GetTime* function to call the *GetTime* web method and display the resulting value in the *LabelTime* *<div>* tag.  
You can change the value in *LabelTime* by using the following code:  
**\$('#LabelTime').html(ValueToSet);**
- 15 Modify your code to directly call the *GetTime* web method using the JQuery *\$.ajax* function.



Exercise8 - Start

Exercise8 - 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 15	Q 14	Q 12	Q 3
<p>Use the following code:</p> <pre>function GetTime() { \$.ajax({ type: "POST", url: "TimeManual.aspx/ GetTime", contentType: "application/json", success: function(data) { \$('#LabelTime'). html(data.d); }, }); }</pre> <p>This was covered in: <i>Lesson 8-11: Directly call a web method that returns a value using JQuery.</i></p>	<p>Use the following code:</p> <pre>function GetTime() { PageMethods. GetTime (function (response) { \$('#LabelTime') .html(response); }); }</pre> <p>This was covered in: <i>Lesson 8-8: Call a web method that returns a value using JavaScript.</i></p>	<p>Add the following code just above the <i>GetTime</i> method:</p> <p><b>[WebMethod]</b></p> <p>This was covered in: <i>Lesson 8-6: Create a web method.</i></p>	<ol style="list-style-type: none"> <li>1. Right-click <i>References</i> in the <i>Solution Explorer</i> and click <i>Add Service Reference...</i> from the shortcut menu.</li> <li>2. Click <i>Discover</i>.</li> <li>3. In the <i>Namespace</i> box, type: <b>TestServiceReference</b></li> <li>4. Click <i>OK</i>.</li> </ol> <p>This was covered in: <i>Lesson 8-2: Use a Service Reference to connect to a web service.</i></p>

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

- 1 Refer to: Essential Skills Session 1.
- 2 Refer to: Lesson 8-1: Create a web service.
- 4 Refer to: Essential Skills Session 1.
- 5 Refer to: Lesson 8-3: Create ScriptManager and UpdatePanel controls.
- 6 Refer to: Lesson 8-3: Create ScriptManager and UpdatePanel controls.
- 7 Refer to: Lesson 8-3: Create ScriptManager and UpdatePanel controls.
- 8 Refer to: Lesson 8-5: Use the Timer control for scheduled updates.
- 9 Refer to: Lesson 8-5: Use the Timer control for scheduled updates.
- 10 Refer to: Lesson 8-5: Use the Timer control for scheduled updates.
- 11 Refer to: Lesson 8-5: Use the Timer control for scheduled updates.
- 13 Refer to: Lesson 8-6: Create a web method.



---

## Session 9: Exercise

- 1 Open *Exercise9* from your sample files folder.
- 2 Open the code-behind file of *TotalSales.aspx*.
- 3 Add code to the *Page\_Load* event handler to extract the total of the *SaleValue* fields from the *Sale* database table using the LINQ *Sum* method. Display the result in the *LabelTotalSales* control.

TOTAL SALES
899280.14

- 4 Open the code-behind file of *Sales.aspx*.
- 5 Modify the *Page\_Load* event handler to use an *anonymous type* which extracts the following fields:

**Sale.SaleID**  
**Sale.SaleQuantity**  
**Sale.SaleValue**  
**Sale.Customer.CustomerName**  
**Sale.Product.ProductName**

Customer	Product	Quantity	Value
Bill Washington	Expert Skills Book	289	8667.1100

- 6 Open the code-behind file of *SalesByProduct.aspx*.
- 7 Modify the *Page\_Load* event handler to return the total of all sales values grouped by product name. To do this, use the *GroupBy* and *Sum* methods with an *anonymous type*.

Key	TotalSales
Essential Skills Book	312945.6500
Essential Skills Video	104755.0700

- 8 Open the code-behind file of *Customers.aspx*.
- 9 Add code to the *ButtonSaveAsCSV* event handler to save the data from the *Customers* collection into a CSV file.  
Save the file to: **C:\Practice\ASP.NET Expert\Exercise9.csv**
- 10 Add code to the *ButtonSaveAsXML* event handler to save the data from the *Customers* collection into an XML file.  
Save the file to: **C:\Practice\ASP.NET Expert\Exercise9.xml**
- 11 Open the code-behind file of *ReadDataFiles.aspx*.
- 12 Add code to the *ButtonReadCSV\_Click* event handler to read the CSV file you created and display it in the *LabelResult* control as an HTML table.
- 13 Add code to the *ButtonReadXML\_Click* event handler to read the XML file you created and display it in the *LabelResult* control as an HTML table.

**Exercise9 - Start**

**Exercise9 - End**

If you need help  
slide the page to  
the left



## Session 9: Exercise Answers

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

Q 13	Q 10	Q 7
<p>Use the following code:</p> <pre> <b>XmlDocument Document = new XmlDocument(); Document.Load( FileUploadDataFile.FileContent); XmlNodeList CustomerNodes = Document. GetElementsByTagName ("Customer"); string HTMLOutput = "&lt;table&gt;"; foreach (XmlNode CustomerNode in CustomerNodes) {     HTMLOutput += "&lt;tr&gt;&lt;td&gt;" +     CustomerNode.Attributes     ["CustomerID"].Value + "&lt;/td&gt;";     HTMLOutput += "&lt;td&gt;" +     CustomerNode.Attributes     ["CustomerName"].Value + "&lt;/tr&gt;"; } HTMLOutput += "&lt;/table&gt;"; LabelResult.Text = HTMLOutput;</b></pre> <p>This was covered in: <i>Lesson 9-10: Read data from an XML file.</i></p>	<p>Use the following code:</p> <pre> <b>XmlDocument Document = new XmlDocument(); XmlElement RootElement = Document.CreateElement("Customers"); foreach (Customer CustomerToXML in Customers) {     XmlElement CustomerElement =     Document.CreateElement("Customer");     CustomerElement.     SetAttribute("CustomerID", CustomerToXML.CustomerID. ToString());     CustomerElement.     SetAttribute("CustomerName", CustomerToXML.CustomerName);     RootElement.     AppendChild(CustomerElement); } Document.AppendChild(RootElement); Document.Save("C:\\Practice\\ASP.NET Expert\\Exercise9.xml");</b></pre> <p>This was covered in: <i>Lesson 9-9: Write data to an XML file.</i></p>	<p>Use the following code:</p> <pre> <b>var Result = Data.Sales. GroupBy(Sale =&gt; Sale.Product. ProductName). Select(Group =&gt; new {Group.Key, TotalSales= Group.Sum (Sale =&gt; Sale.SaleValue) });</b></pre> <p>This was covered in: <i>Lesson 9-3: Use the Sum and GroupBy LINQ methods.</i></p>

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

- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Essential Skills Session 1.
- 3** Refer to: Lesson 9-1: Use the Average LINQ method.
- 4** Refer to: Essential Skills Session 1.
- 5** Refer to: Lesson 9-2: Use LINQ anonymous types.
- 6** Refer to: Essential Skills Session 1.
- 8** Refer to: Essential Skills Session 1.
- 9** Refer to: Lesson 9-7: Write data to a CSV file.
- 11** Refer to: Essential Skills Session 1.
- 12** Refer to: Lesson 9-8: Read data from a CSV file.

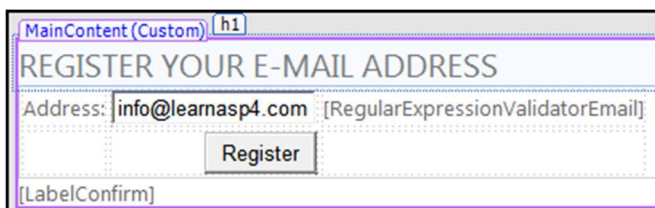
## Session 10: Exercise

- 1 Open *Exercise10* from your sample files folder.
- 2 View *EmailRegex.aspx* in your web browser.
- 3 Enter a regular expression for e-mail addresses in the *Regular Expression* box and click *Test Expression*.

The correct regular expression should match the two valid e-mail addresses but not match the two invalid e-mail addresses.

E-MAIL		
Valid Address 1:	<input type="text" value="info@learnasp4.com"/>	MATCH
Valid Address 2:	<input type="text" value="info@learnasp4.co.uk"/>	MATCH
Invalid Address 1:	<input type="text" value="infolearnasp4.com"/>	NO MATCH
Invalid Address 2:	<input type="text" value="info@learnasp4"/>	NO MATCH

- 4 Close your web browser, return to Visual Studio and open *EmailForm.aspx* in Design view.
- 5 Add a *RegularExpressionValidator* control to the page that will ensure that the *TextBoxEmailAddress* control contains a valid e-mail address.



- 6 Open the code-behind file of *EmailForm.aspx* and add C# code to confirm your *RegularExpressionValidator* control's validation even if the user has disabled JavaScript.
- 7 Open the code-behind file of *CustomerNumber.aspx*.
- 8 Add C# code to the *ButtonConfirm\_Click* event handler to confirm that the *CustomerNumber* variable contains a valid 6-digit number.
- 9 Open *Web.config* and add code to allow security validation to be disabled on the *LinkExtractor.aspx* page.
- 10 Open *LinkExtractor.aspx* in Source view and disable security validation on the page.
- 11 View *LinkExtractor.aspx* in your web browser.
- 12 In the *Regular Expression* box, enter a regular expression that will extract HTML links.
- 13 Click *Extract Links*.

EXTRACTED LINKS
<a href="#">Learn ASP</a>
<a href="#">Learn Excel</a>
<a href="#">Learn Access VBA</a>
<a href="#">The Smart Method</a>

Exercise10 - Start

Exercise10 - 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 12	Q 10	Q 9	Q 8
<p>Use the following regular expression:</p> <pre>&lt;a.+&gt;</pre> <p>This was covered in: <i>Lesson 10-5: Create a regular expression for HTML links.</i></p>	<p>Add the following property to the &lt;%@ Page&gt; tag:</p> <pre>ValidateRequest="false"</pre> <p>This was covered in: <i>Lesson 10-5: Create a regular expression for HTML links.</i></p>	<p>Add the following code inside the &lt;configuration&gt; tag:</p> <pre>&lt;location   path="LinkExtractor.aspx"&gt;   &lt;system.web&gt;     &lt;httpRuntime       requestValidationMode         ="2.0" /&gt;     &lt;/system.web&gt;   &lt;/location&gt;</pre> <p>This was covered in: <i>Lesson 10-5: Create a regular expression for HTML links.</i></p>	<p>1. Add the following <i>using</i> line:</p> <pre>using System.Text. RegularExpressions;</pre> <p>2. Add the following code:</p> <pre>bool IsValid = Regex. IsValid(   CustomerNumber,   "[0-9]{6}");</pre> <p>3. Enclose the remaining code in the following <i>if</i> statement:</p> <pre>if (IsValid) { }</pre> <p>This was covered in: <i>Lesson 10-7: Work with regular expressions in C#.</i></p>

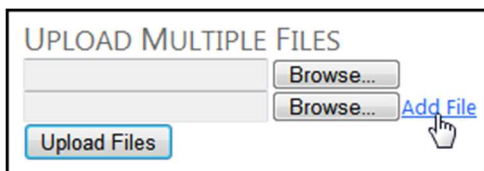
If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Essential Skills Session 1.
- 3** Refer to: Lesson 10-4: Create a regular expression for e-mail addresses.
- 4** Refer to: Essential Skills Session 1.
- 5** Refer to: Lesson 10-6: Use the RegularExpressionValidator control.
- 6** Refer to: Lesson 10-6: Use the RegularExpressionValidator control.
- 7** Refer to: Essential Skills Session 1.
- 11** Refer to: Essential Skills Session 1.

---

## Session 11: Exercise

- 1 Open *Exercise11* from your sample files folder.
- 2 Open *CompressFile.aspx* in Design view and add a *Click* event handler to the *ButtonCompress* control.
- 3 Add code to the new event handler to compress the file from the *FileUploadCompress* control.
- 4 Add code to output the compressed file to the user's web browser.
- 5 Open *Global.asax*.
- 6 Add a new method to *Global.asax* that runs the *CleanUp.bat* program.
- 7 Add code to the *Application\_Start* event handler that will run the new method every hour.
- 8 Open *MultiUpload.aspx* in Source view.
- 9 Add code to the *AddFile* JavaScript function to add a new upload control to the *Uploads* *<span>* tag.



- 10 Open the code-behind file of *XMLConfiguration.aspx*.
- 11 Add code to the *ButtonLoadConfiguration\_Click* event handler to deserialize the *Configuration.xml* file into the *LoadedConfiguration* object.



- 12 Open *BannerHandler.cs* from the *Banner* folder.
- 13 Add code to the *ProcessRequest* method to create and output an image.  
The image should be 700 pixels wide and 75 pixels high.  
The image should use the **Arial** font at size 60  
The image should contain the text: **The Smart Method**
- 14 View *Banner.aspx* in your web browser to view the image.



Exercise11 - Start

Exercise11 - 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 13	Q 6, 7	Q 3, 4
<p>Use the following code:</p> <pre> Bitmap CaptchaImage = new Bitmap(700, 75); Graphics GraphicsObject = Graphics.FromImage (CaptchaImage); GraphicsObject.FillRectangle (Brushes.White, new Rectangle(0, 0, 700, 75)); Font CaptchaFont = new Font("Arial", 60); GraphicsObject.DrawString (BannerText, CaptchaFont, Brushes.Black, new PointF(0, 0)); CaptchaImage.Save (context.Response. OutputStream, ImageFormat.Jpeg); context.Response.End(); </pre> <p>This was covered in: <i>Lesson 11-9: Use the System.Drawing classes to manipulate images and Lesson 11-10: Create a Captcha image system.</i></p>	<p>1. Create the method using the following code:</p> <pre> static void RunCleanUp(object sender) {     Process CleanUp =     Process.Start(         ((HttpContext)sender).         Server.MapPath         ("~/CleanUp.bat")); } </pre> <p>2. Add the following code to the <i>Application_Start</i> event handler:</p> <pre> Timer CleanUpTimer = new Timer(new TimerCallback (RunCleanUp), HttpContext.Current, 0, 3600000); </pre> <p>This was covered in: <i>Lesson 11-3: Run an external program and Lesson 11-4: Create a scheduled event using a Timer object.</i></p>	<p>Use the following code:</p> <pre> MemoryStream CompressedStream = new MemoryStream(); GZipStream Compressor = new GZipStream(CompressedStream, CompressionMode.Compress); Compressor.Write( FileUploadFileToCompress. FileBytes, 0, FileUploadFileToCompress.FileBytes. Length); byte[] CompressedBytes = CompressedStream.ToArray(); Response.AddHeader ("content-disposition", "attachment;filename=" + FileUploadFileToCompress. FileName); Response.BinaryWrite (CompressedBytes); Response.End(); </pre> <p>This was covered in: <i>Lesson 11-7: Compress a file and Lesson 11-2: Output a file to the user's web browser</i></p>

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

- 1** Refer to: Essential Skills Session 1.
- 2** Refer to: Essential Skills Session 3.
- 5** Refer to: Essential Skills Session 2.
- 8** Refer to: Essential Skills Session 1.
- 9** Refer to: Lesson 11-11: Create a page that can upload multiple files.
- 10** Refer to: Essential Skills Session 1.
- 11** Refer to: Lesson 11-6: Deserialize an XML file back into an object.
- 12** Refer to: Essential Skills Session 1.
- 14** Refer to: Essential Skills Session 1.