Lab Answer Key: Module 6: Reading and Writing Local Data

Lab: Generating the Grades Report

Exercise 1: Serializing Data for the Grades Report as XML

Task 1: Prompt the user for a filename and retrieve the grade data

1. Start the MSL-TMG1 virtual machine if it is not already running.
2. Start the 20483B-SEA-DEV11 virtual machine.
3. Log on to Window 8 as **Student** with the password **Pa$$w0rd**. If necessary, click **Switch User** to display the list of users.
4. Switch to the Windows 8 **Start** window.
5. Click **Visual Studio 2012**.
6. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
7. In the **Open Project** dialog box, browse to **E:\Mod06\Labfiles\Starter\Exercise 1**, click **GradesPrototype.sln**, and then click **Open**.
8. In Solution Explorer, expand **GradesPrototype**, expand **Views**, and then double-click **StudentProfile.xaml**.
9. Note that this view displays and enables users to add grades for a student. The solution has been updated to include a **Save Report** button that users will click to generate and save the Grades Report.
10. On the **View** menu, click **Task List**.
11. In the **Task List** window, in the **Categories** list, click **Comments**.

12. Double-click the **TODO: Exercise 1: Task 1a: Store the return value from the SaveFileDialog in a nullable Boolean variable.** task.

13. In the code editor, click in the blank line below the comment, and then type the following code:

```
Nullable<bool> result = dialog.ShowDialog();
if (result.HasValue && result.Value)
{
```

14. Click at the end of the last comment in this method, press Enter, and then type the following code:

```
}
```

15. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1b: Get the grades for the currently selected student.** task.

16. In the code editor, click in the blank line below the comment, and then type the following code:

```
List<Grade> grades = (from g in DataSource.Grades
where g.StudentID == SessionContext.CurrentStudent.StudentID
select g).ToList();
```

17. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1c: Serialize the grades to a MemoryStream.** task.

18. In the code editor, click at the end of the comment, press Enter, and then type the following code:

```
MemoryStream ms = FormatAsXMLStream(grades);
```

Task 2: Serialize the grade data to a memory stream

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2a: Save the XML document to a MemoryStream by using an XmlWriter** task.

2. In the code editor, click in the blank line below the comment, and then type the following code:

```
MemoryStream ms = new MemoryStream();
XmlWriter writer = XmlWriter.Create(ms);
```

3. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2b: Create the root node of the XML document.** task.

4. In the code editor, click in the blank line below this and the next comment, and then type the following code:

```
writer.WriteStartDocument();
writer.WriteStartElement("Grades");
writer.WriteAttributeString("Student", String.Format("{0} {1}",
SessionContext.CurrentStudent.FirstName,
SessionContext.CurrentStudent.LastName));
```

5. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2c: Format the grades for the student and add them as child elements of the root node** task.

6. In the code editor, click in the blank line below this and the next comment, and then type the following code:

```
foreach (Grade grade in grades)
{
    writer.WriteStartElement("Grade");
    writer.WriteAttributeString("Date", grade.AssessmentDate);
    writer.WriteAttributeString("Subject", grade.SubjectName);
    writer.WriteAttributeString("Assessment",
grade.Assessment);
    writer.WriteAttributeString("Comments", grade.Comments);
    writer.WriteEndElement();
}
```

7. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2d: Finish the XML document with the appropriate end elements** task.
8. In the code editor, click in the blank line below the comment, and then type the following code:

```
writer.WriteEndElement();
writer.WriteEndDocument();
```

9. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2e: Flush the XmlWriter and close it to ensure that all the data is written to the MemoryStream** task.
10. In the code editor, click in the blank line below the comment, and then type the following code:

```
writer.Flush();
writer.Close();
```

11. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2f: Reset the MemoryStream so it can be read from the start and then return it** task.
12. In the code editor, click in the blank line below the comment, and then type the following code:

```
ms.Seek(0, SeekOrigin.Begin);
```

```
return ms;
```

13. Delete the following line of code from the end of the method:

```
throw new NotImplementedException();
```

Task 3: Debug the application

1. On the **Build** menu, click **Build Solution**.
2. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1c: Serialize the grades to a MemoryStream** task.
3. In the code editor, select the closing brace immediately below the following line of code:

```
MemoryStream ms = FormatAsXMLStream(grades);
```

4. On the **Debug** menu, click **Toggle Breakpoint**.
5. On the **Debug** menu, click **Start Debugging**.
6. In the **Username** box, type **vallee**.
7. In the **Password** box, type **password99**, and then click **Log on**.
8. In the main application window, click **Kevin Liu**.
9. In the **Report Card** view, click **Save Report**.
10. In the **Save As** dialog box, click **Save**.

Note: You will write the code to actually save the report to disk in Exercise 3 of this lab.

11. When you enter Break Mode, in the Immediate Window, type the following code, and then press Enter.

```
?(new StreamReader(ms)).ReadToEnd()
```

12. Review the grade data formatted as XML that is returned to the Immediate Window.
13. On the **Debug** menu, click **Stop Debugging**.
14. On the **Debug** menu, click **Delete All Breakpoints**
15. In the confirmation message box, click **Yes**.
16. On the **File** menu, click **CloseSolution**.

Results: After completing this exercise, users will be able to specify the location for the Grades Report file.

Exercise 2: Previewing the Grades Report

Task 1: Display the string to the user in a message box

1. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:\Mod06\Labfiles\Starter\Exercise 2**, click **GradesPrototype.sln**, and then click **Open**.
3. On the **View** menu, click **Task List**.

4. In the **Task List** window, in the **Categories** list, click **Comments**.

5. Double-click the **TODO: Exercise 2: Task 1a: Generate a string representation of the report data** task.

6. In the code editor, click in the blank line below the comment, and then type the following code:

```
string formattedReportData = FormatXMLData(ms);
```

7. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1b: Preview the string version of the report data in a MessageBox** task.

8. In the code editor, click in the blank line below the comment, and then type the following code:

```
MessageBox.Show(formattedReportData, "Preview Report",
MessageBoxButton.OK,
MessageBoxImage.Information);
```

Task 2: Build a string representation of the XML document

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2a: Use a StringBuilder to construct the string** task.

2. In the code editor, click in the blank line below the comment, and then type the following code:

```
StringBuilder builder = new StringBuilder();
```

3. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2b: Use an XmlTextReader to read the XML data from the stream** task.

4. In the code editor, click in the blank line below the comment, and then type the following code:

```
XmlTextReader reader = new XmlTextReader(stream);
```

5. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2c: Read and process the XML data a node at a time** task.

6. In the code editor, click in the blank line below the comment, and then type the following code:

```
while (reader.Read())
            {
                switch (reader.NodeType)
                {
                    case XmlNodeType.XmlDeclaration:
                        // The node is an XML declaration such
as <?xml
version='1.0'>
                        builder.Append(String.Format("<?{0}
{1}>\n", reader.Name,
reader.Value));
                        break;
                    case XmlNodeType.Element:
                        // The node is an element (enclosed
between '<' and '/>')
                        builder.Append(String.Format("<{0}",
reader.Name));
                        if (reader.HasAttributes)
                        {


                            // Output each of the attributes of
the element in the
form "name='value'"
                            while
(reader.MoveToNextAttribute())
                            {
                                builder.Append(String.Format("
{0}='{1}'",
reader.Name, reader.Value));
                            }
                        }
                        builder.Append(">\n");
                        break;
                    case XmlNodeType.EndElement:
                        // The node is the closing tag at the
end of an element
                        builder.Append(String.Format("</{0}>",
reader.Name));
                        break;
                }
            }
```

7. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2d: Reset the stream and return the string containing the formatted data** task.

8. In the code editor, click in the blank line below the comment, and then type the following code:

```
stream.Seek(0, SeekOrigin.Begin);
return builder.ToString();
```

9. Delete the following line of code from the end of the method:

```
throw new NotImplementedException();
```

Task 3: Run the application and preview the data.

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. In the **Username** box, type **vallee**.
4. In the **Password** box, type **password99**, and then click **Log on**.
5. In the main application window, click **Kevin Liu**.
6. In the **Report Card** view, click **Save Report**.
7. In the **Save As** dialog box, click **Save**.

   Note: You will write the code to actually save the report to disk in the next exercise of this lab.

8. Review the XML data displayed in the message box, and then click **OK.**
9. Close the application.
10. In Visual Studio, on the **File** menu, click **Close Solution**.

Results: After completing this exercise, users will be able to preview a report before saving it.

Exercise 3: Persisting the Serialized Grade Data to a File

Task 1: Save the XML document to disk

1. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:\Mod06\Labfiles\Starter\Exercise 3**, click **GradesPrototype.sln**, and then click **Open**.

3. On the **View** menu, click **Task List**.

4. In the **Task List** window, in the **Categories** list, click **Comments**.

5. Double-click the **TODO: Exercise 3: Task 1a: Modify the message box and ask the user whether they wish to save the report** task.

6. In the code editor, delete the line of code below the comment, and then type the following code:

```
MessageBoxResult reply = MessageBox.Show(formattedReportData,
"Save Report?",
MessageBoxButton.YesNo, MessageBoxImage.Question);
```

7. In the **Task List** window, double-click the **TODO: Exercise 3: Task 1b: If the user says yes, then save the data to the file that the user specified earlier** task.

8. In the code editor, click at the end of the comment, press Enter, and then type the following code:

```
if (reply == MessageBoxResult.Yes)
{
    // If the user says yes, then save the data to the file
that the user specified
earlier
    // If the file already exists it will be overwritten (the
SaveFileDialog box will
already have asked the user whether this is OK)
    FileStream file = new FileStream(dialog.FileName,
FileMode.Create,
FileAccess.Write);
    ms.CopyTo(file);
    file.Close();
}
```

Task 2: Run the application and verify that the XML document is saved correctly

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. In the **Username** box, type **vallee**.
4. In the **Password** box, type **password99**, and then click **Log on**.
5. In the main application window, click **Kevin Liu**.
6. In the **Report Card** view, click **Save Report**.
7. In the **Save As** dialog box, browse to the **Documents** folder, and then click **Save**.
8. Review the XML data displayed in the message box, and then click **Yes**.
9. Close the application.
10. Open Internet Explorer.
11. Press the Alt key, and then on the **File** menu, click **Open**.
12. In the **Open** dialog box, click **Browse**.
13. In the **Windows Internet Explorer** dialog box, browse to the **Documents** folder, click **Grades.xml**, and then click **Open**.
14. In the **Open** dialog box, click **OK**.
15. Verify that the file contains the expected grade data, and then close Internet Explorer.
16. In Visual Studio, on the **File** menu, click **Close Solution**.

Results: After completing this exercise, users will be able to save student reports to the local hard disk in XML format.