## Lab: Upgrading the Grades Report

### Lab Setup

Estimated Time: **60 minutes**

### Preparation Steps

1. Initialize database:
   * In the **Apps** list, click **File Explorer**.
   * In **File Explorer**, navigate to the **E:/Allfiles/Mod11/Labfiles/Databases** folder, and then double-click **SetupSchoolGradesDB.cmd.**
   * **>** Note : If a Windows protected your PC dialog appears, click More info and then click Run Anyway.
   * Close **File Explorer**.

### Exercise 1: Generating the Grades Report by Using Word

#### Task 1: Examine the WordWrapper class that provides a functional wrapper around the dynamic (COM) API for Word

1. Open **Visual Studio 2017**.
2. In **Visual Studio**, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **E:/Allfiles/Mod11/Labfiles/Starter/Exercise 1**, click **Grades.sln**, and then click **Open**.
4. In **Solution Explorer**, right-click **Solution ‘Grades’**, and then click **Properties**.
5. In the **Solutions ‘Grades’ Properties Pages** dialog box, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start**, and then click **OK**.
6. In **Solution Explorer**, expand **Grades.Utilities**, and then double-click **WordWrapper.cs**.
7. Examine the code that is currently contained within this class.
8. On the **View** menu, click **Task List**.
9. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1a: Create a dynamic variable called \_word for activating Word** task.
10. In the code editor, click in the blank line below the comment, and then type the following code:

* dynamic \_word = null;

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1b: Instantiate \_word as a new Word Application object** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* this.\_word = new Application { Visible = false };

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1c: Create a new Word document** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* var doc = this.\_word.Documents.Add();
* doc.Activate();

1. In the **Task List** window, double-click **TODO: Exercise 1: Task 1d: Save the document using the specified filename**task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* var currentDocument = this.\_word.ActiveDocument;
* currentDocument.SaveAs(filePath);

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1e: Close the document** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* currentDocument.Close();

#### Task 2: Review the code in the GeneratedStudentReport method to generate a Word document

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2a: Generate a student grade report as a Word document**task.
2. Examine the code that is in this method to generate the student report.
3. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2b: Generate the report by using a separate task** task.
4. In the code editor, click in the blank line below the comment, and then type the following code:

Task.Run(() => GenerateStudentReport(SessionContext.CurrentStudent, dialog.FileName));

#### Task 3: Build and test the application

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. When the application loads, in the **Username** text box, type **vallee**, and in the **Password** text box, type **password99**, and then click **Log on**.
4. Click **Kevin Liu**, and then click **save report**.
5. In the **Save As** dialog box, browse to **E:/Allfiles/Mod11/Labfiles/Starter/Exercise 1**.
6. In the **File name** text box, delete the existing contents, type **Kevin Liu Grades Report**, and then click **Save**.
7. Close the application, and then in **Visual Studio**, on the **File** menu, click **Close Solution**.
8. Open **File Explorer**, browse to the **E:/Allfiles/Mod11/Labfiles/Starter/Exercise 1** folder, and then verify that the report has been generated.
9. Double-click **Kevin Liu Grades Report.docx**.

**Note**: Close the panel that asks fothe r your product key, you can use **Office** on the trial plan for a few days. Accept the trial license.

1. Review the grade report, and then close **Microsoft Word**.

**Result:** After completing this exercise, the application will generate grade reports in the **docx** format.

### Exercise 2: Controlling the Lifetime of Word Objects by Implementing the Dispose Pattern

#### Task 1: Run the application to generate a grades report and view the Word task in Task Manager

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:/Allfiles/Mod11/Labfiles/Starter/Exercise 2**, click **Grades.sln**, and then click **Open**.
3. In **Solution Explorer**, right-click **Solution ‘Grades’**, and then click **Properties**.
4. In the **Solutions ‘Grades’ Properties Pages** dialog box, click **Multiple startup projects**, set **Grades.Web** and **Grades.WPF** to **Start**, and then click **OK**.
5. On the **Build** menu, click **Build Solution**.
6. On the **Debug** menu, click **Start Without Debugging**.
7. When the application loads, in the **Username** text box, type **vallee**, and in the **Password** text box, type **password99**, and then click **Log on**.
8. Click **Kevin Liu**, and then click **save report**.
9. In the **Save As** dialog box, browse to **E:/Allfiles/Mod11/Labfiles/Starter/Exercise 2**.
10. In the **File name** text box, delete the existing contents, type **Kevin Liu Grades Report**, and then click **Save**.
11. Close the application.
12. Open **File Explorer**, browse to the **E:/Allfiles/Mod11/Labfiles/Starter/Exercise 2** folder, and then verify that the report has been generated.

#### Task 2: Update the WordWrapper class to terminate Word correctly

1. In **Visual Studio**, in the **Task List** window, double-click the **TODO: Exercise 2: Task 2a: Specify that the WordWrapper class implements the IDisposable interface** task.
2. In the code editor, on the line below the comment, click at the end of the **public class WordWrapper** code, and then type the following code:

: IDisposable

Note: Add the following using statement:

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2b: Create the protected Dispose(bool) method** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

protected virtual void Dispose(bool isDisposing)

{

if (!this.isDisposed)

{

if (isDisposing)

{

// Release managed resources here

if (this.\_word != null)

{

this.\_word.Quit();

}

}

* // Release unmanaged resources here  
   if (this.\_word != null)  
   {  
   System.Runtime.InteropServices.Marshal.ReleaseComObject(this.\_word);  
   }  
    
   this.isDisposed = true;  
   }
* }

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2c: Create the public Dispose method** task.
2. In the code editor, click at the end of the comment, press Enter, and then type the following code:

public void Dispose()

{

this.Dispose(true);

GC.SuppressFinalize(this);

}

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2d: Create a finalizer that calls the Dispose method** task.
2. In the code editor, click in the blank line below the comment, and then type the following :

private bool isDisposed = false;

#### Task 3: Wrap the object that generates the Word doc in a using statement

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 3: Ensure that the WordWrapper is disposed when the method finishes** task.
2. Below the comment, modify the **WordWrapper wrapper = new WordWrapper();** code to look like the following:

using (var wrapper = new WordWrapper()) {

1. At the end of the method, after the **wrapper.SaveAs(reportPath);** line of code, add a closing brace to end the **using** block.
2. Your code should look like the following:

public void GenerateStudentReport(LocalStudent studentData, string reportPath)

{

// TODO: Exercise 2: Task 3: Ensure that the WordWrapper is disposed when the

// method finishes

using (var wrapper = new WordWrapper())

{

// Create a new Word document in memory

wrapper.CreateBlankDocument();

* // Add a heading to the document
* wrapper.AppendHeading(String.Format("Grade Report: {0} {1}",
* studentData.FirstName, studentData.LastName));
* wrapper.InsertCarriageReturn();
* wrapper.InsertCarriageReturn();
* // Output the details of each grade for the student  
   foreach (var grade in SessionContext.CurrentGrades)  
   {  
   wrapper.AppendText(grade.SubjectName, true, true);  
   wrapper.InsertCarriageReturn();  
   wrapper.AppendText("Assessment: " + grade.Assessment, false, false);  
   wrapper.InsertCarriageReturn();  
   wrapper.AppendText("Date: " + grade.AssessmentDateString, false, false);  
   wrapper.InsertCarriageReturn();  
   wrapper.AppendText("Comment: " + grade.Comments, false, false);  
   wrapper.InsertCarriageReturn();  
   wrapper.InsertCarriageReturn();  
   }  
    
   // Save the Word document  
   wrapper.SaveAs(reportPath);  
   }
* }

#### Task 4: Use Task Manager to observe that Word terminates correctly after generating a report

1. On the **Build** menu, click **Build Solution**.
2. Right-click the **taskbar**, and then click **Task Manager**.
3. In **Visual Studio**, on the **Debug** menu, click **Start Without Debugging**.
4. When the application loads, in the **Username** text box, type **vallee**, and in the **Password** text box, type **password99**, and then click **Log on**.
5. Click **Kevin Liu**, and then click **save report**.
6. In the **Save As** dialog box, browse to **E:/Allfiles/Mod11/Labfiles/Starter/Exercise 2**.
7. In the **File name** text box, delete the existing contents, and then type **Kevin Liu Grades Report**.
8. As you click **Save**, in the **Task Manager** window, watch the **Processes** and verify that **Microsoft Word** appears and then disappears from the list.
9. Close **Task Manager** and then close the application.
10. In **Visual Studio**, on the **File** menu, click **Close Solution**.

**Result:** After completing this exercise, the application will terminate **Word** correctly after it has generated a grades report.