## Lab: Specifying the Data to Include in the Grades Report

### Lab Setup

Estimated Time: **75 minutes**

### Preparation Steps

1. Initialize database:
   * In the **Apps** list, click **File Explorer**.
   * Navigate to the **E:/Allfiles/Mod12/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: Creating and Applying the IncludeInReport attribute

#### Task 1: Write the code for the IncludeInReportAttribute class

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/Mod12/Labfiles/Starter/Exercise 1**, click **Grades.sln**, and then click **Open**.
4. In **Solution Explorer**, right-click Solution **‘Grades’**, and then click **Properties**.
5. On the **Startup Project** page, 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 **IncludeInReport.cs**.
7. On the **View** menu, click **Task List**.
8. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1a: Specify that IncludeInReportAttribute is an attribute class** task.
9. In the code editor, below the comment, click at the end of the public **public class IncludeInReportAttribute** code, and then type the following code:

: Attribute

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1b: Specify the possible targets to which the IncludeInReport attribute can be applied** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* [AttributeUsage(AttributeTargets.Field | AttributeTargets.Property,
  + AllowMultiple = false)]

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1c: Define a private field to hold the value of the attribute** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* private bool \_include;

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1d: Add public properties that specify how an included item should be formatted** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* public bool Underline { get; set; }
* public bool Bold { get; set; }

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1e: Add a public property that specifies a label (if any) for the item** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* public string Label { get; set; }

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1f: Define constructors** task.
2. In the code editor, click at the end of the comment, press Enter, and then type the following code:

* public IncludeInReportAttribute()  
  {  
   this.\_include = true;  
   this.Underline = false;  
   this.Bold = false;  
   this.Label = string.Empty;  
  }  
    
  public IncludeInReportAttribute(bool includeInReport)  
  {  
   this.\_include = includeInReport;  
   this.Underline = false**;**  
   this.Bold = false**;**  
   this.Label = string.Empty;  
  }

#### Task 2: Apply the IncludeInReportAttribute attribute to the appropriate properties

1. In **Solution Explorer**, expand **Grades.WPF**, and then double-click **Data.cs**.
2. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2: Add the IncludeInReport attribute to the appropriate properties in the LocalGrade class** task.
3. In the **LocalGrade** class, expand the **Properties** region, and then expand the **Readonly Properties** region.
4. Above the **public string SubjectName** code, click in the blank line, and then type the following code:

[IncludeInReport(Label="Subject Name", Bold=true, Underline=true)]

1. Above the **public string AssessmentDateString** code, click in the blank line, press Enter, and then type the following code:

[IncludeInReport (Label="Date")]

1. Expand the **Form Properties** region.
2. Above the **public string Assessment** code, click in the blank line, press Enter, and then type the following code:

[IncludeInReport(Label = "Grade")]

1. Above the **public string Comments** code, click in the blank space, press Enter, and then type the following code:

[IncludeInReport(Label = "Comments")]

#### Task 3: Build the application and review the metadata for the LocalGrades class

1. On the **Debug** menu, click **Start Debugging**.
2. Close the application.
3. Open **File Explorer** and browse to the **C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.7 Tools** folder.
4. Right-click **ildasm.exe,** and then click **Open**.
5. In the **IL DASM** window, on the **File** menu, click **Open**.
6. In the **Open** dialog box, browse to **E:\AllFiles\Mod12\Labfiles\Starter\Exercise 1\Grades.WPF\bin\Debug**, click **Grades.WPF.exe**, and then click **Open**.
7. In the **IL DASM** application window, expand **Grades.WPF,** expand **Grades.WPF.LocalGrade**, and then double-click **Assessment : instance string()**.
8. In the **Grades.WPF.LocalGrade::Assessment : instance string()** window, in the **Assessment** method, verify that the **.custom instance void [Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor()** code is present, and then close the window.
9. In the **IL DASM** application window, double-click **AssessmentDateString : instance string()**.
10. In the **Grades.WPF.LocalGrade::AssessmentDateString : instance string()** window, in the **AssessmentDateString** method, verify that the **.custom instance void [Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor()** code is present, and then close the window.
11. In the **IL DASM** application window, double-click **Comments : instance string()**.
12. In the **Grades.WPF.LocalGrade::Comments : instance string()** window, in the **Comments** method, verify that the **.custom instance void [Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor()** code is present, and then close the window.
13. In the **IL DASM** application window, double-click **SubjectName : instance string();**.
14. In the **Grades.WPF.LocalGrade::SubjectName : instance string()** window, in the **SubjectName** method, verify that the **.custom instance void [Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor()** code is present, and then close the window.
15. Close the **IL DASM** application.
16. Close **File Explorer**.
17. Stop the debugger.
18. In **Visual Studio**, on the **File** menu, click **Close Solution**.

**Result:** After completing this exercise, the **Grades.Utilities** assembly will contain an **IncludeInReport** custom attribute and the **Grades** class will contain fields and properties that are tagged with that attribute.

### Exercise 2: Updating the Report

#### Task 1: Implement a static helper class called IncludeProcessor

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\Mod12\Labfiles\Starter\Exercise 2**, click **Grades.sln**, and then click **Open**.
3. In **Solution Explorer**, right-click Solution **‘Grades’**, and then click **Properties**.
4. On the **Startup Project** page, click **Multiple startup projects**, set **Grades.Web** and **Grades.WPF** to **Start**, and then click **OK**.
5. In **Solution Explorer**, expand **Grades.Utilities**, and then double-click **IncludeInReport.cs**.
6. Below the **Output** window, click **Task List**.
7. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1a: Define a struct that specifies the formatting to apply to an item** task
8. In the code editor, click in the blank line in the **FormatField** struct, and then type the following code:

public string Value;

public string Label;

public bool IsBold;

public bool IsUnderlined;

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1b: Find all the public fields and properties in the dataForReport object** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* Type dataForReportType = dataForReport.GetType();
* fieldsAndProperties.AddRange(dataForReportType.GetFields());
* fieldsAndProperties.AddRange(dataForReportType.GetProperties());

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1c: Iterate through all public fields and properties, and process each item that is tagged with the IncludeInReport attribute** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* foreach (MemberInfo member in fieldsAndProperties)
* {

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1d: Determine whether the current member is tagged with the IncludeInReport attribute** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* object[] attributes = member.GetCustomAttributes(false);
* IncludeInReportAttribute attributeFound = Array.Find(attributes,
* a => a.GetType() == typeof(IncludeInReportAttribute)) as IncludeInReportAttribute;

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1e: If the member is tagged with the IncludeInReport attribute, construct a FormatField item** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* if (attributeFound != null)
* {
* // Find the value of the item tagged with the IncludeInReport attribute
* string itemValue;
* if (member is FieldInfo)
* {
* itemValue = (member as FieldInfo).GetValue(dataForReport).ToString();
* }
* else
* {
* itemValue = (member as PropertyInfo).GetValue(dataForReport).ToString();
* }

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1f: Construct a FormatField item with this data** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* FormatField item = new FormatField()
* {
* Value = itemValue,
* Label = attributeFound.Label,
* IsBold = attributeFound.Bold,
* IsUnderlined = attributeFound.Underline
* };

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1g: Add the FormatField item to the collection to be returned** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

* items.Add(item);  
   }  
  }

#### Task 2: Update the report functionality for the StudentProfile view

1. In **Solution Explorer**, expand **Grades.WPF**, expand **Views**, expand **StudentProfile.xaml**, and then double-click **StudentProfile.xaml.cs**.
2. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2a: Use the IncludeProcessor to determine which fields in the Grade object are tagged** task.
3. In the code editor, click in the blank line below the comment, and then type the following code:

List<FormatField> itemsToReport = IncludeProcessor.GetItemsToInclude(grade);

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2b: Output each tagged item, using the format specified by the properties of the IncludeInReport attribute for each item** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

foreach (FormatField item in itemsToReport)

{

wrapper.AppendText(item.Label == string.Empty ? item.Value : item.Label + “:”

+ item.Value, item.IsBold, item.IsUnderlined);

* wrapper.InsertCarriageReturn();
* }

#### 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. In the **Username** text box, type **vallee**, and in the **Password** text box, type **password99**, and then click **Log on**.
4. In the **Class 3C** view, click **Kevin Liu**.
5. Verify that the student report for **Kevin Liu** appears, and then click **save report**.
6. In the **Save As** dialog box, browse to the **E:\AllFiles\Mod12\Labfiles\Starter\Exercise 2** folder.
7. In the **File name** text box, type **KevinLiuGradesReport**, and then click **Save**.
8. Close the application.
9. In **Visual Studio**, on the **File** menu, click **Close Solution**.
10. Open **File Explorer**, browse to **E:\AllFiles\Mod12\Labfiles\Starter\Exercise 2**, and then verify that **KevinLiuGradesReport.docx** has been generated.
11. Right-click **KevinLiuGradesReport.docx**, and then click **Open**.
12. Verify that the document contains the grade report for **Kevin Liu** and that it is correctly formatted, and then close**Microsoft Word**.
13. Close File Manager.

**Result:** After completing this exercise, the application will be updated to use reflection to include only the tagged fields and properties in the grades report.

### Exercise 3: Storing the Grades.Utilities Assembly Centrally (if time permits)

#### Task 1: Sign the Grades.Utilities assembly and deploy it to the General Assembly Cache (GAC)

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\Mod12\Labfiles\Starter\Exercise 3**, click **Grades.sln**, and then click **Open**.
3. In **Solution Explorer**, right-click **Solution ‘Grades’**, and then click **Properties**.
4. On the **Startup Project** page, click **Multiple startup projects**, set **Grades.Web** and **Grades.WPF** to **Start**, and then click **OK**.
5. Click on the **Windows Start** icon and begin typing **Developer Command Prompt for VS 2017**.
6. Right click on the **Developer Command Prompt for VS 2017** icon and click **Run as administrator**.
7. In the **User Account Control** dialog box, click on **Yes**.
8. In the **Command Prompt** window, type the following code, and then press Enter:

E:

cd E:\AllFiles\Mod12\Labfiles\Starter

1. In the **Command Prompt** window, type the following code, and then press Enter:

* sn -k GradesKey.snk

1. Verify that the text **Key pair written to GradesKey.snk** is displayed.
2. In **Visual Studio**, in **Solution Explorer**, right-click **Grades.Utilities**, and then click **Properties**.
3. On the **Signing** tab, select **Sign the assembly**.
4. In the **Choose a strong name key file** list, select **Browse** ftrom the drop down control.
5. In the **Select File** dialog box, browse to **E:\AllFiles\Mod12\Labfiles\Starter**, click **GradesKey.snk**, and then click **Open**.
6. On the **Build** menu, click **Build Solution**.
7. Switch to the **Command Prompt** window, type the following code, and then press Enter:

* cd E:\AllFiles\Mod12\Labfiles\Starter\Exercise 3\Grades.Utilities\bin\Debug

1. In the **Command Prompt** window, type the following code, and then press Enter:

* gacutil -i Grades.Utilities.dll

1. Verify that the text **Assembly successfully added to the cache** is displayed, and then close the **Command Prompt** window.

#### Task 2: Reference the Grades.Utilities assembly in the GAC from the application

1. In **Visual Studio**, in **Solution Explorer**, expand **Grades.WPF**, expand **References**, right-click **Grades.Utilities**, and then click **Remove**.
2. Right-click **References**, and then click **Add Reference**.
3. In the **Reference Manager – Grades.WPF** dialog box, click the **Browse** button.
4. In the **Select the files to reference** dialog box, browse to **E:\AllFiles\Mod12\Labfiles\Starter\Exercise 3\Grades.Utilities\bin\Debug**, click **Grades.Utilities.dll**, and then click **Add**.
5. In the **Reference Manager – Grades.WPF** dialog box, click **OK**.
6. On the **Build** menu, click **Build Solution**.
7. On the **Debug** menu, click **Start Without Debugging**.
8. In the **Username** text box, type **vallee**, and in the **Password** text box, type **password99**, and then click **Log on**.
9. In the **Class 3C** view, click **Kevin Liu**.
10. Verify that the student report for **Kevin Liu** appears, and then click **save report**.
11. In the **Save As** dialog box, browse to the **E:\AllFiles\Mod12\Labfiles\Starter\Exercise 3** folder.
12. In the **File name** text box, type **KevinLiuGradesReport**, and then click **Save**.
13. Close the application.
14. In **Visual Studio**, on the **File** menu, click **Close Solution**.
15. Open **File Explorer**, browse to **E:\AllFiles\Mod12\Labfiles\Starter\Exercise 3**, and then verify that **KevinLiuGradesReport.docx** has been generated.
16. Right-click **KevinLiuGradesReport.docx**, and then click **Open**.
17. Verify that the document contains the grade report for **Kevin Liu** and that it is correctly formatted, and then close **Word**.
18. Close all open windows.

**Result:** After completing this exercise, you will have a signed version of the **Grades.Utilities** assembly deployed to the GAC.