Programming 3 - Advanced

Laboratory 9 - Assembly, Attribute, Reflection, Embedded Resources

Agenda:

- Task 1 - Creating objects from the class
- Task 2 - Loading DLLs and invoking methods
- Project - Testing framework

Task 1

In this task, you are provided with two classes: `Author` and `Book`. Using Reflection, write a program that reads the class definition and asks the user for the data for each property in the class.

1) Read the `Author` class and create an instance based on its definition. To do this, implement the `TypeCrafter.TypeCraft` method. Fill the created instance with data provided via the command line. To parse input from the command line, use the `TryParse` Method from each member type.
2) Add a new exception, `ParseException`, and when `TryParse` returns false, throw the `ParseException` with an appropriate message. In the main program, catch the exceptions - if `ParseException` is caught, attempt to create object one again. Abandon the process otherwise.
3) Using the same function (`TypeCrafter.TypeCraft`) as above, read the `Book` class. As the `Author` type is not parsable, change the function implementation to **recursively** create objects for non-parsable properties.
4) Change `Author`'s property `int Age` to `DateTime BirthDate`. Add automatic age calculation to the `ToString()` method. Use documentation to get to know the date format that can be parsed into the `DateTime` class.

Task 2

In this task, you are given the window application that is supposed to apply image filters to the displayed image.

The layout of the application is already written for you. After running the `Task2_ImageFilters` project, you should be able to see a simple layout displaying:

- in the main area: modified image
- on the right side: control panel storing button to reset displayed image and 3 buttons for applying image filters:
  - Gaussian Blur filter
  - Laplace filter
  - emboss filter

Don't worry! All filters have already been implemented for you in the projects: `Task2_Filters.Emboss`, `Task2_Filters.GaussianBlur` and `Task2_Filters.Laplace`! The output files of those projects are dynamic-link libraries (DLLs) that can be loaded into your application dynamically and serve as a library of functions ready to be invoked.

Your task is to setup proper Post-Build events to save the DLLs in the proper location (to the folder in either Task2_Filters.Common or Task2_ImageFilters), compile the projects to generate DLLs, read DLLs into the application in the designated places, and invoke appropriate methods.

Crucial information:

- all filters implement interface `IImageFilter` that is located in the `Task2_Filters.Common` project,
- `IImageFilter` cannot be changed,
- all methods to be implemented are located in project `Task2_ImageFilters` in the `FiltersLibrary.cs` file:
  - for Gaussian Blur filter: `ApplyGaussianBlurFilter`,
  - for Laplace filter: `ApplyLaplaceFilterButton_Click`,

  - for emboss filter: `ApplyEmbossFilterButton_Click.` Other existing classes shouldn't be changed. Nevertheless, you can add your own helper classes to the `Task2_ImageFilters` project, and you can add helper methods to the `FiltersLibrary` class if you need to.

1) Setup proper Post-Build events in the `Task2_Filters.Emboss`, `Task2_Filters.GaussianBlur` and `Task2_Filters.Laplace` projects. This Post-Build event should copy the output DLLs into the proper folder. From this folder, you will later load DLLs into the main application.
2) Compile `Task2_Filters.Emboss`, `Task2_Filters.GaussianBlur` and `Task2_Filters.Laplace` to receive necessary DLL files. (Note that for all parts below, you can implement a helper method and call it with proper parameters.)
3) Implement the `ApplyGaussianBlurFilter` method in `FiltersLibrary.cs` to read `Task2_Filters.GaussianBlur.d` and invoke the `ApplyFilter()` method.
4) Implement the `ApplyLaplaceFilter` method in `FiltersLibrary.cs` to read `Task2_Filters.Laplace.dll` and invoke the `ApplyFilter()` method.
5) Implement the `ApplyEmbossFilter` method in `FiltersLibrary.cs` to read `Task2_Filters.Emboss.dll` and invoke the `ApplyFilter()` method.