Project 4

T-213-VEFF, Vefforritun, 2015-1

Assignment

Note: you are allowed (and encouraged!) to work on this assignment in pairs.

In this assignment you should write a simple web application with one link on the front page of the web. If the link is clicked, the user will simply be routed to the front page again (using RedirectToAction). However, occasionally, the action method might throw an exception (but not always). Occasionally (one in every 5 occasions) the method should throw an ArgumentException, and occasionally (again, in maybe 1 in a 5), it should throw a custom Exception object you should declare yourself, called MyApplicationException.

Your application should then handle these errors "gracefully", i.e. it should catch these errors, and display a nice error page specifying what happened. Also, the application should log the error, and this should be handled by a class Logger that takes care of handling errors. This is the main part of this application.

In your application, errors should be routed to these three locations:

- the output window (using System.Diagnostics.Debug.WriteLine)
- a text file, the location of the textfile should be specified in web.config, The default location should be C:\Temp\<username>_logfile.txt, where <username> is your RU username. For instance, this path would be located in web.config in my case: C:\Temp\dabs_logfile.txt. However, your code should not rely on this particular location, i.e. there should be no hardcoded path in the C# code itself.
- sent to email. The email address of the receiver should be stored in web.config, plus information about the sender.

Each output method should be handled by a separate class, and they should all inherit from a common base class called LogMedia (you need to create all those classes yourself!). The Logger class must then own a collection of LogMedia instances, initialized in the constructor of the Logger class.

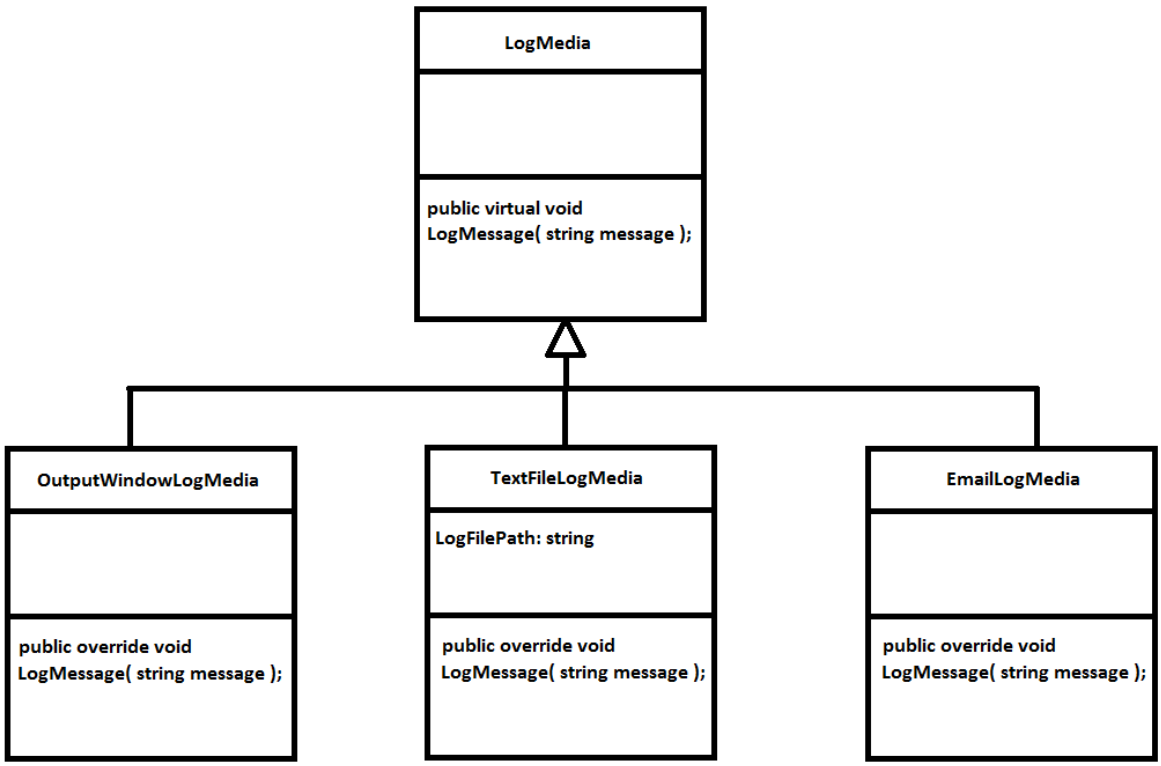
Implementation

Here are a number of hints on how to implement this:

- You could use System.Random to generate a random number inside the action method, and then check if the number % 5 == 0. If that is the case, an error should be thrown. You can find examples on how to use the System.Random class here.

- The class Logger should declare a collection of LogMedia as a member variable (perhaps List<LogMedia> m_loggers = new List<LogMedia>();), and only needs to declare a single function: void LogException(Exception ex). Inside this function, you should loop through all LogMedia instances, and call a virtual function defined in each one (see below). This class should also "prepare" the actual message sent, i.e. it should for instance append the current date/time to the message itself.

- The LogMedia class hierarchy could look like this:



I.e. you should declare the class LogMedia, and a single virtual function inside it. The implementation of this function would be empty, i.e. by default it doesn't do anything. You might even declare it as abstract, in which case there will be no implementation whatsoever; there will just be a semicolon after the function declaration. Each of the derived classes should then override this function, each providing their own implementation.

FAQ

Some FAQ that are floating around. If I start seeing patterns in students questions - they will end on this list.

Where should I put my code? I.e. the custom exception class, the logger classes, etc.?

In an ASP.NET MVC application, you don't really have to place your non-MVC code in any particular place. Personally, I prefer to create a separate folder for my code in the root of the project, call it (surprise, surprise) "Code", and place all my non-MVC classes either there directly or organize them into some hierarchy if the number of classes becomes quite large. Alternatively, I would create a separate project (a "Class library") containing all the stand-alone code, place these classes there, and add a reference to this library in my web project.

Hand in

Please hand in a single archive file (.zip, .rar, 7z), containing the entire solution.
It should be possible to unzip the archive, and run it instantaneously without any changes to the code.

Again: you may work on this in pairs. (and this is encouraged, particularly if you are not sure how this would be implemented).