Abstract TCP Server – part 4

Mission

To add logging to the Abstract Server using xml-files.

Background

Previous exercise Abstract TCP Server – part 3 (part1+part2 as well)
Slides: ServerFramework1.pdf, ServerFramework2.pdf, ServerFramework3.pdf and ServerFramework4.pdf

Logging see

- Trace: https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.trace?view=net-5.0
- TraceListeners: https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.tracelistener?view=net-5.0
- TraceFilter: https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.tracefilter?view=net-5.0

Assignment 1 – Logging Information of the Simple Framework

Add logging information to your TCP-server.

In the AbstractServer create a TraceSource and add at least two listeners (e.g. ConsoleListener and TextWriterTraceListener). Set the starting level of logging to 'All' (i.e. setting the Switch property to new SourceSwitch(…, …))

Insert appropriate logging information for different levels (verbose, information, warnings …). This could be when server is started, connected to a client etc. or if any errors occur.

Look at every place you have a Console.WriteLine, **then replace** this with logging (tracing) instead. Then perhaps add mode Tracing.

Assignment 2 – More Logging Information of the Simple Framework

Enhanced your server to support:

- That you could have different levels bounded to the different Listeners.
  i.e. adding filters
- You server can log to the EventLog system

Assignment 3 – Refactor your logging

You are to refactor your logging system to be in a singleton class, whereby it will be the same logging setup all over the server:

- Make a new class "MyLogger", which holds the TraceSource object.
- Make the class into a singleton, see https://www.dofactory.com/net/singleton-design-pattern (though you can make the constructor private instead of protected – i.e. do not inheritance)
- Add appropriated methods to add listeners, set values, and write logging information

Extra E1 – Make your own Tracelistener

Implement a new class JsonTraceListener as Inherit from TraceListener. Let the Listener write log information to a jsonFile.

Extra E2 – Make your own Filter

Implement a new class SubstringFilter as inherit from TraceFilter. Let the Filter return true if the 'formatOrMessage' contains a specified substring.

Extra E3 – Make a CompositeFilter

Implement a new class CompositeFilter as inherit from TraceFilter (see https://www.dofactory.com/net/composite-design-pattern). This filter can hold zero to many other filters and will return true only if all filters evaluates true.

Extra E4 – Make your own Tracelistener

Implement a new class RestTraceListener as Inherit from TraceListener. Let the Listener write log information to a Rest-service that stores loginformation – i.e. use post-calls.