Abstract TCP Server – part 2

Mission

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

Background

Previous exercise Abstract TCP Server – part 1
Slides: ServerFramework1.pdf and ServerFramework2.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

XML files see

- https://www.w3schools.com/xml/default.asp
- https://support.microsoft.com/da-dk/help/307548/how-to-read-xml-from-a-file-by-using-visual-c

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 (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.

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.
- You server can log to the EventLog system

Assignment A – Make your own Tracelistener

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

Assignment B – 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.

Assignment C – 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.

Assignment D – 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.

Assignment 3 – Create Configuration file

Make an xml-configuration file, with **one** root e.g. ServerConfig and with values for

- ServerPort
- ShutDownPort
- Possible servername
- Possible debuglevel (inf, warning, error)
- …

Use any text editor.

Assignment 4 – Read Configuration File

In the AbstractTCPServer somewhere in the initialization (e.g. in the Main-method) read the config-file and use the values for setting up the server.

To open config-file use:

```
XmlDocument configDoc = new XmlDocument();
configDoc.Load( " << configFileName >> ");
```

To read a port number:

```
XmlNode xxNode = configDoc.DocumentElement.SelectSingleNode("NameOfTag");
if (xxNode != null)
{
    String xxStr = xxNode.InnerText.Trim();
    Int xx = Convert.ToInt32(xxStr);
}
```

Assignment E – Improve the Configuration

To be more free where to locate the configuration-file you are to read an environment variable, which value hold the location.

Step 1: create environment variable

Open file exploire, right click at my PC and chose properties, choose Advanced System Settings, and then click environment variable (danish – 'miljøvariable').
Now make a new entry for the system 'AbstractServerConf' with a value some path on your computer e.g. c:\conf\server, where you have placed your config-file.

Step 2: read the environment variable
see: https://docs.microsoft.com/en-us/dotnet/api/system.environment.getenvironmentvariable?view=net-5.0

```
String path = Environment.GetEnvironmentVariable(”AbstractServerConf”)
```

Step 3: use this path for open / load the xml config-file (see assignment 4)