Server Framework – 4

(tracing and logging)

Peter Levinsky IT, Roskilde

03.02.2025

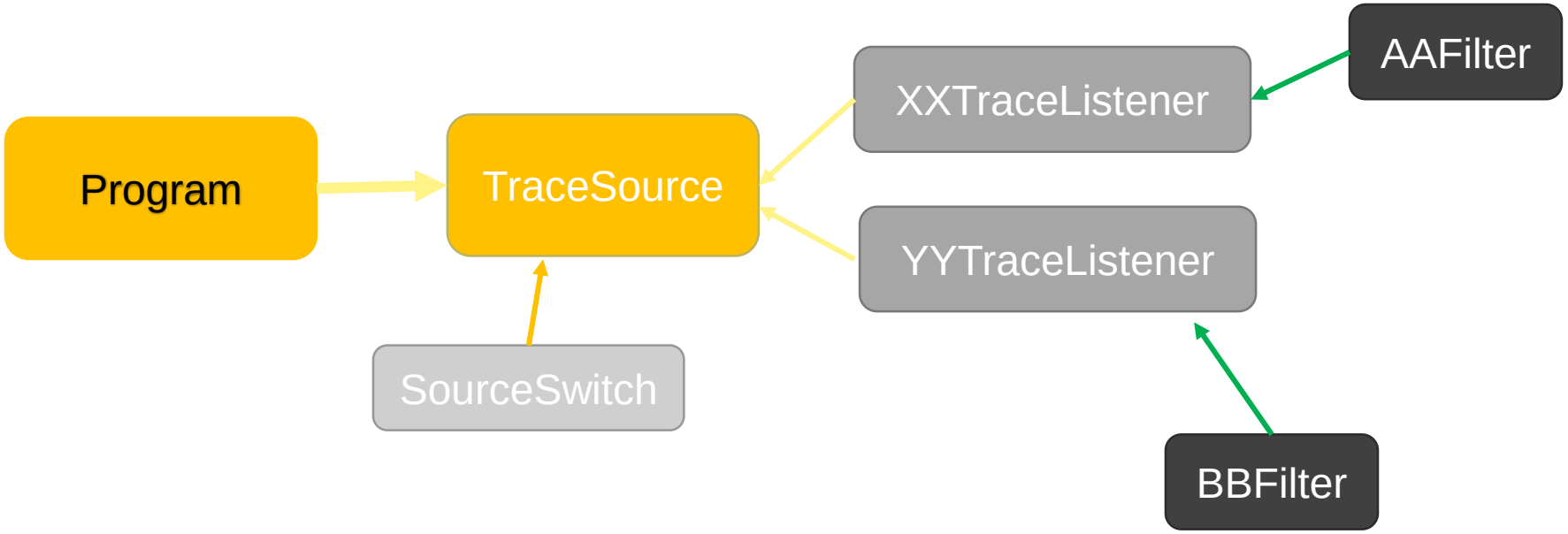
Tracing / logging information

- Instead of using Console.WriteLine **use** tracing / logging for released Systems.

- You can setup the log to write to:
  - The Console
  - A File, in different formats
  - (Windows Event Log)

- The Tracing can have several output channels
- The Trace level can be changed (actual at runtime)

Overview Tracing / logging



How to Choose Output Chanel

- The **TraceSource** class can write to “TraceListener”
- The ”TraceListener” is an abstract class
  i.e. you need concrete TraceListener class.

- They work like observers
  i.e. you can add them to a TraceSource (ts) object like:
  ```
  ts.Listeners.Add(objOfTraceListener);
  ```

- C# have some buildt in classes like:
  - TextWriterTraceListener
  - XmlWriterTraceListener
  - EventLogTraceListener

- Customer Created Listener

Make your own TraceListener class

- You can design and implement you own Listener by Inherits from **`TraceListener`** and override:
  - **public override void Write(string message)**
  - **public override void WriteLine(string message)**

Trace Level

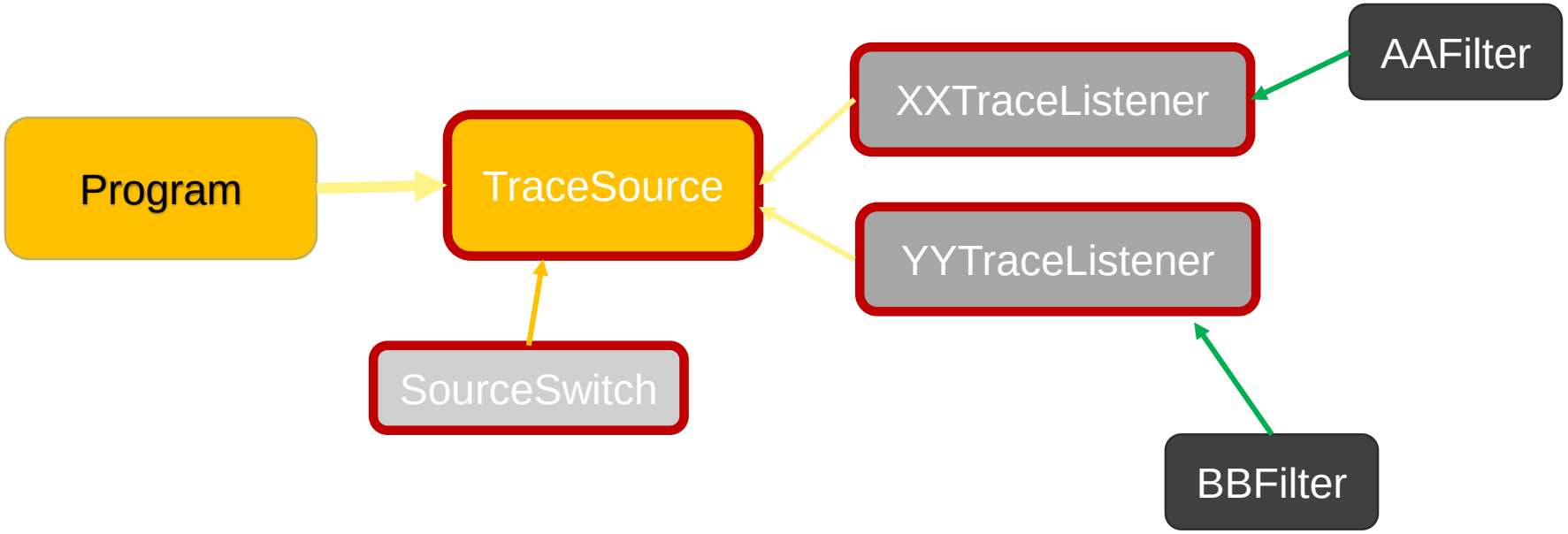
- TraceSource works with diff. Levels of logging
  - Verbose
  - Info
  - Warning
  - Error
  - Critical

- Setting actual levels of logging ex:
  ts.Switch = new SourceSwitch("Peters","All");

- You specify the level when you logging like:
  ts.TraceEvent(TraceEventType.Error, <<ID>>, <<Object/string to log>>);

  Example:
  ts.TraceEvent(TraceEventType.Error, 333, ”This is an Error");

Overview Tracing / logging



Trace Filters

- TraceSource can take filters to configure the individual TraceListener

- Types of filters:
  - SourceFilter (build in) -- for configure which part of the system to log
  - EventTypeFilter (build in) -- for configure level of logging messages to log

  - Customer Created Filters

- Example of filter setting:

```
xxListener.Filter = new EventTypeFilter(SourceLevels.All);
```

Make your own Filter

- You can design and implement you own Filter by Inherits from **`TraceFilter`** and override:
  - **public override bool ShouldTrace(**

```
TraceEventCache cache,                  -- some metadata
string source,                          -- where does it come from
TraceEventType eventType,               -- the level error,warning…
int id,                                 -- some id
string formatOrMessage,                 -- the text string – can be null
object[] args,                          -- additional inf.
object data1,                           -- additional inf.
object[] data)                          -- additional inf.
```

Special TraceListener - EventLog

- **The `EventLogTraceListener` will log to the system Event Log System (use EventViewer to lookup the logging)**
- **Need to get NuGet Package (System.Diagnostics.EventLog)**
- **Example:**

```
TraceListener logListener = new EventLogTraceListener("Application");
ts.Listeners.Add(logListener);
```

Your turn

Thursday – Comments / soft closing