Log4Net, how to add a custom field to my logging

Ask Question

88

35

I use the log4net.Appender.AdoNetAppender appender.
My log4net table are the following fields `[Date],[Thread], [Level],[Logger],[Message],[Exception]`

I would need to add another field to the log4net table (e.g SalesId), but how would I specify in my xml and in code to log the "SalesId" when logging a Error or Info message?

e.g. `log.Info("SomeMessage", SalesId)`

Here's the log4net xml

```
  <appender name="SalesDBAppender" type="log4net.Appender.AdoNetAppender">
    <bufferSize value="1" />
    <connectionType value ="System.Data.SqlClient.SqlConnection" />
    <connectionString value="Data Source=..." />
    <commandText value="INSERT INTO Log4Net ([Date],[Thread],[Level],[Logger],[Message],
[Exception]) VALUES (@log_date, @thread, @log_level, @logger, @message, @exception)" />
    <parameter>
      <parameterName value="@log_date" />
      <dbType value="DateTime" />
      <layout type="log4net.Layout.RawTimeStampLayout" />
    </parameter>
    <parameter>
      <parameterName value="@thread" />
      <dbType value="String" />
      <size value="255" />
```

```
    </parameter>
    <parameter>
      <parameterName value="@log_level" />
      <dbType value="String" />
      <size value="50" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%level" />
      </layout>
    </parameter>
    <parameter>
      <parameterName value="@logger" />
      <dbType value="String" />
      <size value="255" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%logger" />
      </layout>
    </parameter>
    <parameter>
      <parameterName value="@message" />
      <dbType value="String" />
      <size value="4000" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%message" />
      </layout>
    </parameter>
    <parameter>
      <parameterName value="@exception" />
      <dbType value="String" />
      <size value="2000" />
      <layout type="log4net.Layout.ExceptionLayout" />
    </parameter>
  </appender>
```

c# log4net appender

asked Aug 27 '12 at 9:36

Eminem

3,169 8 38 73

3 Answers

178

```
[Thread],[Level],[Logger],[Message],[Exception],
[MyColumn]) VALUES (@log_date, @thread, @log_level,
@logger, @message, @exception, @CustomColumn)
```

2) Add the parameter definition for the custom column:

```
<parameter>
   <parameterName value="@CustomColumn"/>
   <dbType value="String" />
   <size value="255" />
   <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%property{CustomColumn}" /
  </layout>
</parameter>
```

3) Then use one of log4net's contexts to transfer values to the parameter:

```
// thread properties...
log4net.LogicalThreadContext.Properties["CustomColumn"] =
log.Info("Message");

// ...or global properties
log4net.GlobalContext.Properties["CustomColumn"] = "Custom
```

edited Aug 4 '15 at 11:24

answered Aug 27 '12 at 10:27

Marcelo De Zen
8,195 3 26 43

2 Can someone suggest which Context is best for logging browser capabilities. – VivekDev Mar 10 '15 at 12:20

1 @DumbDev, usually you will use the Thread context. The GlobalContext is useful to set properties that do not change very often. – Marcelo De Zen Mar 10 '15 at 19:06

the ThreadContext will not be moved along with it. – Robba Jan 26 '16 at 8:59

3 @theberserker `LogicalThreadContext` is ok to use with Tasks, It's not ok to use `ThreadContext` though, because it's always bounded to a particular thread. – Marcelo De Zen Jun 9 '16 at 16:07

5 It's my opinion, but it seems a bit weird this way, I expected an overload that takes more arguments to add to the new LoggingEvent instance.. – A77 Jul 12 '17 at 10:08

2

Three types of logging context available in Log4Net.

1. Log4Net.GlobalContext :- This context shared across all application threads and domains.If two threads set the same property on GlobalContext, One Value will override the other.
2. Log4Net.ThreadContext :- This context scope limited to calling thread. Here two threads can set same property to different values without overriding to each other.
3. Log4Net.ThreadLogicalContext :- This context behaves similarly to the ThreadContext. if you're working with a custom thread pool algorithm or hosting the CLR, you may find some use for this one.

Add the following code to your program.cs file:

```
static void Main( string[] args )
{
    log4net.Config.XmlConfigurator.Configure();
    log4net.ThreadContext.Properties[ "myContext" ] = "Log
    Log.Info( "this is an info message" );
    Console.ReadLine();
}
```

2) Add the parameter definition for the custom column:

```
  <log4net>
    <appender name="ConsoleAppender" type="log4net.Appende
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%logger (%property{myCon
%message%newline" />
      </layout>
    </appender>
  </log4net>
```

answered Jul 26 '18 at 16:18

piyush gupta
21 1

0

Here is a working version with some personalized preferences. I added a custom column for storing a generated exception code.

1) Add your custom column(exceptionCode here) to Log4net config:

```
<commandText value="INSERT INTO Log([Date],[Thread],[Level
[Exception],[ExceptionCode])
VALUES (@log_date, @thread, @log_level, @logger, @message,
/>
```

```
    <size value="11" />
    <layout type="Common.Utils.LogHelper.Log4NetExtentedLo
        <conversionPattern value="%exceptionCode{Code}" />
    </layout>
</parameter>
```

2) Log4NetExtentedLoggingCustomParameters.cs

```
namespace Common.Utils.LogHelper
{
    public class Log4NetExtentedLoggingCustomParameters
    {
        public string ExceptionCode { get; set; }

        public string Message { get; set; }

        public override string ToString()
        {
            return Message;
        }
    }
}
```

3) Log4NetExtentedLoggingPatternConverter.cs

```
namespace Common.Utils.LogHelper
{
    public class Log4NetExtentedLoggingPatternConverter : l
    {
        protected override void Convert(TextWriter writer,
        {
            if (state == null)
            {
                writer.Write(SystemInfo.NullText);
                return;
            }

            var loggingEvent = state as LoggingEvent;
            var messageObj = loggingEvent.MessageObject as
Log4NetExtentedLoggingCustomParameters;

            if (messageObj == null)
```

```
            {
                switch (this.Option.ToLower()) //this.Opti
                {
                    case "code": //config conversionPatter
%exceptionCode{Code}
                        writer.Write(messageObj.ExceptionC
                        break;
                    default:
                        writer.Write(SystemInfo.NullText);
                        break;
                }
            }
        }
    }
}
```

4) Log4NetExtentedLoggingPatternLayout.cs

```
namespace Common.Utils.LogHelper
{
    public class Log4NetExtentedLoggingPatternLayout : Pat
    {
        public Log4NetExtentedLoggingPatternLayout()
        {
            var customConverter = new log4net.Util.Convert
            {
                Name = "exceptionCode",
                Type = typeof(Log4NetExtentedLoggingPatter
            };

            AddConverter(customConverter);
        }
    }
}
```

5) Logger.cs // Enjoy your logger with new column! :)

```
namespace Common.Utils.LogHelper
{
    public class Logger
    {
        static ILog Logger =
LogManager.GetLogger(System.Reflection.MethodBase.GetCurre
```

```
            var logWithErrCode = GetLogWithErrorCode(messa
            Logger.Error(logWithErrCode, exception);
            return logWithErrCode.ExceptionCode;
        }

        private static Log4NetExtentedLoggingCustomParamet
message)
        {
            var logWithErrCode = new Log4NetExtentedLoggin
            logWithErrCode.ExceptionCode = GenerateErrorCo
for simplicity. Use your own implementation
            logWithErrCode.Message = message;
            return logWithErrCode;
        }
    }
}
```

references:

http://blog.stvjam.es/2014/01/logging-custom-objects-and-fields-with

edited May 25 '18 at 7:38

answered May 25 '18 at 7:33

Ali Karaca

972 14 25