Autor: Felipe Monteiro

Base de Conhecimento: Configurando logs(log4net) em aplicação .net core

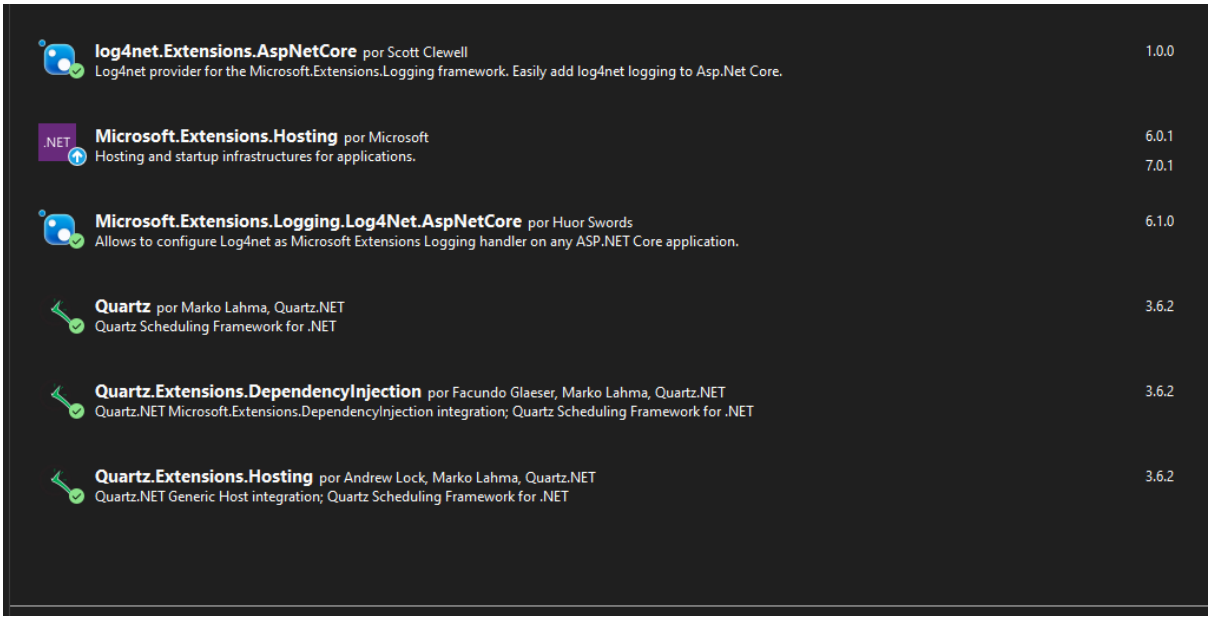
- **Objetivo:** Adicionar Log com o objetivo de registrar todos a execução das tarefas de uma aplicação

1. **Para que servem os logs?**
   São extrema importância, porque permitem monitorar uma aplicação, registrando momentos de execução de tarefas ou funcionalidades de um sistema, como também ajuda a identificar problemas que está ocorrendo, registrando exceções em que podem ser mapeadas por meio dos logs, sendo salvos em arquivo, permitindo assim a visualização e identificação do problema.

   Níveis para uso do log:

Log Level	Severidade	Método	Descrição
Trace	0	LogTrace()	Loga mensagens apenas para fins de rastreamento para os desenvolvedores
Debug	1	LogDebug()	Loga mensagens para fins de depuração de curto prazo
Information	2	LogInformation()	Loga mensagens para o fluxo do aplicativo.
Warning	3	LogWarning()	Loga mensagens para eventos anormais ou inesperados no fluxo do aplicativo.
Error	4	LogError()	Loga mensagens de erros.
Critical	5	LogCritical()	Registra mensagens para as falhas que exigem atenção imediata

2. **Como implementar os logs em uma aplicação asp.net core?**
   Passo 1: Instalar a biblioteca log4net no gerenciador Nuget no visual studio.
   Instalando a biblioteca:
   Microsoft.Extensions.Logging.Log4Net.AspNetCore



Passo 2: Adicionar um arquivo(log4net.config) contendo todas as configurações do log, sendo a pasta de salvamento, tamanho do arquivo, entre outros.

```
<log4net>
    <appender name="RollingFile" type="log4net.Appender.RollingFileAppender">
        <file value="../../../logs/arquivo.log" />
        <appendToFile value="true" />
        <maximumFileSize value="100KB" />
        <maxSizeRollBackups value="2" />
        <layout type="log4net.Layout.PatternLayout">
            <conversionPattern value="%date %5level %logger.%method [%line] - MESSAGE: %message%newline %exception" />
        </layout>
    </appender>
    <root>
        <level value="TRACE" />
        <appender-ref ref="RollingFile" />
    </root>
</log4net>
```

Passo 3: Adicionar o uso do log4net na aplicação no arquivo Program.cs

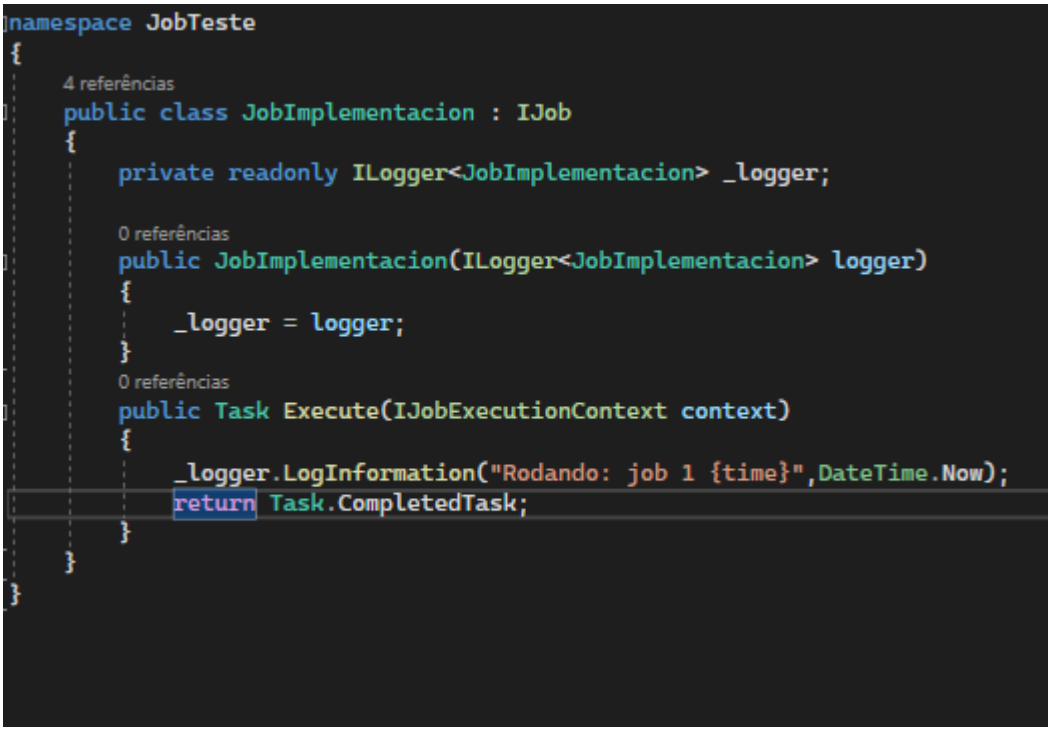
```
IHost host = Host.CreateDefaultBuilder(args)
    .ConfigureServices((hostContext, services) =>
    {
        XmlConfigurator.Configure(new FileInfo("log4net.config"));
        //provedor
        services.AddLogging(loggingBuilder =>
        {
            loggingBuilder.AddLog4Net(); // Adiciona o provedor de logging do Log4Net
        });

        services.AddQuartz(q =>
        {
            q.UseMicrosoftDependencyInjectionJobFactory();
            q.AddJobAndTrigger<JobImplementacion>(hostContext.Configuration);
            q.AddJobAndTrigger<JobImplementacion2>(hostContext.Configuration);
        });
        services.AddQuartzHostedService(q => q.WaitForJobsToComplete = true);
    })
    .Build();

await host.RunAsync();
```

Passo 4: Nessa última etapa só precisamos fazer a injeção da classe ILogger e utilizar de acordo com a necessidade.

Referências:

https://www.macoratti.net/18/12/aspn_logging1.html

codebehind - Gravando logs com log4Net Net 6 C#


namespace JobTeste
{
4 referências
public class JobImplementacion : IJob
{
private readonly ILogger<JobImplementacion> _logger;
0 referências
public JobImplementacion(ILogger<JobImplementacion> logger)
{
_logger = logger;
}
0 referências
public Task Execute(IJobExecutionContext context)
{
_logger.LogInformation("Rodando: job 1 {time}",DateTime.Now);
return Task.CompletedTask;
}
}
}