# Using Background Processing to Build Scalable Applications with Hangfire

## Lab 9

### Goal

As our application continues to grow and we create more and more jobs, the logging framework we put in place may not be sufficient. In this lab we will take a deeper dive into logging and update our logging to make it easier to find what we are looking for (if we were to ever have an error that we would need to dig up more information on).

In the first part of the lab we will be updating our logging with a MappedDiagnostics and GlobalDiagnostics Context so that we can easily find all the logs for each instance of a job.

1. Open the BaseLemmingJob in Core.Jobs and add the following protected variable:

protected readonly Guid Id = Guid.NewGuid();

1. Add a constructor to the BaseLemmingJob

protected BaseLemmingJob()

{

MappedDiagnosticsContext.Set("JobId", Id.ToString("N"));

}

1. Next open the Bootstrap class in Core.Configuration and add the following to the beginning of the BootstrapNLog method:

GlobalDiagnosticsContext.Set("ServiceInstanceId", Guid.NewGuid().ToString("N"));

1. Run your project by hitting F5 and queue some jobs at random.
2. Open SQL Server Management Studio or SQL Server Object Explorer (View -> SQL Server Object Explorer) in Visual Studio and look at the EventLog table with the following query:

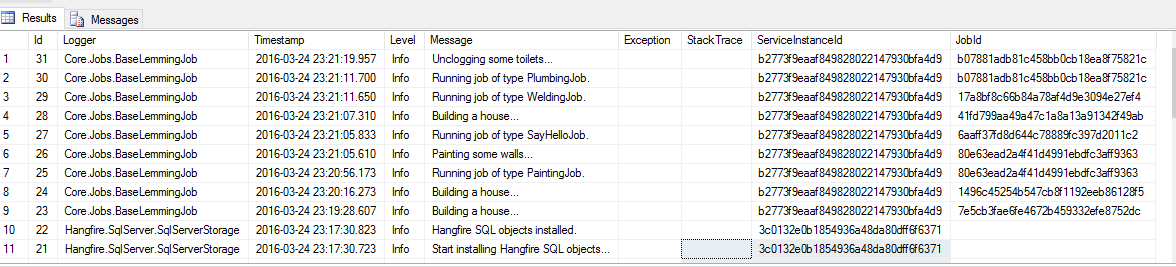
USE LemmingSchedulingSystem

SELECT TOP 100 \*

FROM dbo.EventLog

ORDER BY Timestamp DESC

1. Notice that can now identify all messages that were logged for a particular instance of a job using the JobId column and the server that the job ran on with the ServiceInstanceId column



Next we will log some messages to Slack. We have already setup a slack server for you to use and setup the channel. If you want more information on setting up a WebHook in Slack please review: <https://api.slack.com/incoming-webhooks>

1. Add a new folder to Core called Model
2. Add a new class to Core.Model call SlackPostModel

public class SlackPostModel

{

[JsonProperty("text")]

public string Text { get; set; }

}

1. Add the following using statement to this class:

using Newtonsoft.Json;

1. Add a new folder to Core called Extensions and add a new class to this called SlackTarget.

[Target("Slack")]

public sealed class SlackTarget : TargetWithLayout

{

[RequiredParameter]

public string MessagePrefix { get; set; }

[RequiredParameter]

public string ChannelUrl { get; set; }

protected override void Write(LogEventInfo logEvent)

{

var logMessage = Layout.Render(logEvent);

if (!string.IsNullOrWhiteSpace(MessagePrefix))

{

logMessage = $"{MessagePrefix} - {logMessage}";

}

var json = JsonConvert.SerializeObject(new SlackPostModel() { Text = logMessage });

var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.ExpectContinue = false;

httpClient.DefaultRequestHeaders.ConnectionClose = true;

var stringContent = new StringContent(json);

var response = httpClient.PostAsync(ChannelUrl, stringContent).Result;

try

{

response.EnsureSuccessStatusCode();

}

catch (Exception)

{

throw;

}

}

}

1. Add the following using statements to this class:

using System.Net.Http;

using Core.Model;

using Newtonsoft.Json;

using NLog;

using NLog.Config;

using NLog.Targets;

1. Open the Bootstrap class in Core.Configuration and add the following lines to the BootstrapNLogMethod prior to last line (LogManager.Configuration = config;)

var slackTarget = new SlackTarget() {ChannelUrl = "https://hooks.slack.com/services/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", MessagePrefix = Environment.MachineName};

config.AddTarget("slack", slackTarget);

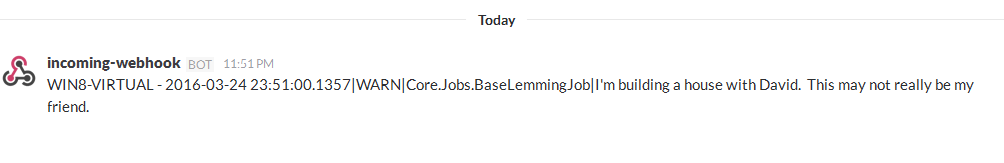
config.LoggingRules.Add(new LoggingRule("\*", LogLevel.Warn, slackTarget));

We have now successfully configured NLog to log Warnings to Slack!

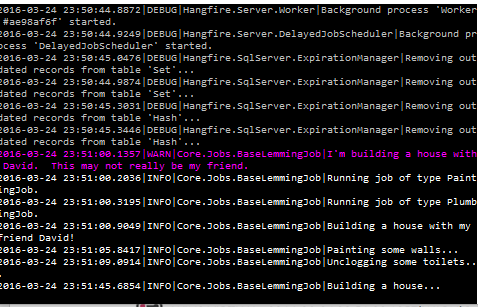
1. Open up your favorite job and use the Logger.Warn to add a custom warning message. I have chosen the BuildHouseWithFriendWithName job.

Example: Logger.Warn($"I'm building a house with {friendName}. This may not really be my friend.");

1. Run your project by pressing F5 and queue the job that you updated with a Warning message and take note of the Slack channel on the projector!



1. P.S. We have also configured a ColoredConsoleTarget for you. Take a look at the messages in the console of your Hangfire server application!



This completes Lab 09.