Лабораторная 15

Код

namespace Microsoft.ApplicationInsights.Extensibility.Implementation.Tracing.DiagnosticsModule

{

using System;

using System.Collections.Generic;

using System.Diagnostics.Tracing;

using System.Linq;

internal class DiagnosticsEventListener : EventListener

{

private const long AllKeyword = -1;

private readonly EventLevel logLevel;

private readonly DiagnosticsListener listener;

private readonly List<EventSource> eventSourcesDuringConstruction = new List<EventSource>();

public DiagnosticsEventListener(DiagnosticsListener listener, EventLevel logLevel)

{

this.listener = listener;

this.logLevel = logLevel;

List<EventSource> eventSources;

lock (this.eventSourcesDuringConstruction)

{

eventSources = this.eventSourcesDuringConstruction;

this.eventSourcesDuringConstruction = null;

}

foreach (var eventSource in eventSources)

{

this.EnableEvents(eventSource, this.logLevel, (EventKeywords)AllKeyword);

}

}

protected override void OnEventWritten(EventWrittenEventArgs eventSourceEvent)

{

if (eventSourceEvent == null || this.listener == null)

{

return;

}

var metadata = new EventMetaData

{

EventSourceName = eventSourceEvent.EventSource?.Name,

Keywords = (long)eventSourceEvent.Keywords,

MessageFormat = eventSourceEvent.Message,

EventId = eventSourceEvent.EventId,

Level = eventSourceEvent.Level,

};

var traceEvent = new TraceEvent

{

MetaData = metadata,

Payload = eventSourceEvent.Payload?.ToArray(),

};

this.listener.WriteEvent(traceEvent);

}

protected override void OnEventSourceCreated(EventSource eventSource)

{

if (ShouldSubscribe(eventSource))

{

// If our constructor hasn't run yet (we're in a callback from the base class

// constructor), just make a note of the event source. Otherwise logLevel is

// set to the default, which is "LogAlways".

var tmp = this.eventSourcesDuringConstruction;

if (tmp != null)

{

lock (tmp)

{

if (this.eventSourcesDuringConstruction != null)

{

this.eventSourcesDuringConstruction.Add(eventSource);

return;

}

}

}

this.EnableEvents(eventSource, this.logLevel, (EventKeywords)AllKeyword);

}

base.OnEventSourceCreated(eventSource);

}

/// <summary>

/// This method checks if the given EventSource Name matches known EventSources that we want to subscribe to.

/// </summary>

private static bool ShouldSubscribe(EventSource eventSource)

{

#if REDFIELD

if (eventSource.Name.StartsWith("Redfield-Microsoft-A", StringComparison.Ordinal))

{

switch (eventSource.Name)

{

case "Redfield-Microsoft-ApplicationInsights-Core":

case "Redfield-Microsoft-ApplicationInsights-WindowsServer-TelemetryChannel":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-AppMapCorrelation-Dependency":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-AppMapCorrelation-Web":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-DependencyCollector":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-EventCounterCollector":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-PerformanceCollector":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-PerformanceCollector-QuickPulse":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-Web":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-WindowsServer":

case "Redfield-Microsoft-ApplicationInsights-WindowsServer-Core":

case "Redfield-Microsoft-ApplicationInsights-Extensibility-EventSourceListener":

case "Redfield-Microsoft-ApplicationInsights-AspNetCore":

case "Redfield-Microsoft-ApplicationInsights-LoggerProvider":

return true;

default:

return false;

}

}

if (eventSource.Name == "Microsoft-AspNet-Telemetry-Correlation")

{

return true;

}

#else

if (eventSource.Name.StartsWith("Microsoft-A", StringComparison.Ordinal))

{

switch (eventSource.Name)

{

case "Microsoft-ApplicationInsights-Core": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/BASE/src/Microsoft.ApplicationInsights/Extensibility/Implementation/Tracing/CoreEventSource.cs

case "Microsoft-ApplicationInsights-WindowsServer-TelemetryChannel": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/BASE/src/ServerTelemetryChannel/Implementation/TelemetryChannelEventSource.cs

// AppMapCorrelation has a shared partial class: https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/Common/AppMapCorrelationEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-AppMapCorrelation-Dependency": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/DependencyCollector/DependencyCollector/Implementation/AppMapCorrelationEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-AppMapCorrelation-Web": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/Web/Web/Implementation/AppMapCorrelationEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-DependencyCollector": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/DependencyCollector/DependencyCollector/Implementation/DependencyCollectorEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-EventCounterCollector": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/EventCounterCollector/EventCounterCollector/EventCounterCollectorEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-PerformanceCollector": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/PerformanceCollector/PerformanceCollector/Implementation/PerformanceCollectorEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-PerformanceCollector-QuickPulse": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/PerformanceCollector/PerformanceCollector/Implementation/QuickPulse/QuickPulseEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-Web": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/Web/Web/Implementation/WebEventSource.cs

case "Microsoft-ApplicationInsights-Extensibility-WindowsServer": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/WindowsServer/WindowsServer/Implementation/WindowsServerEventSource.cs

case "Microsoft-ApplicationInsights-WindowsServer-Core": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/WEB/Src/WindowsServer/WindowsServer/Implementation/MetricManager.cs

case "Microsoft-ApplicationInsights-Extensibility-EventSourceListener": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/LOGGING/src/EventSource.Shared/EventSource.Shared/Implementation/EventSourceListenerEventSource.cs

case "Microsoft-ApplicationInsights-AspNetCore": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/NETCORE/src/Microsoft.ApplicationInsights.AspNetCore/Extensibility/Implementation/Tracing/AspNetCoreEventSource.cs

case "Microsoft-ApplicationInsights-LoggerProvider": // https://github.com/microsoft/ApplicationInsights-dotnet/blob/develop/LOGGING/src/ILogger/ApplicationInsightsLoggerEventSource.cs

case "Microsoft-AspNet-Telemetry-Correlation": // https://github.com/aspnet/Microsoft.AspNet.TelemetryCorrelation/blob/master/src/Microsoft.AspNet.TelemetryCorrelation/AspNetTelemetryCorrelationEventSource.cs

return true;

default:

return false;

}

}

#endif

return false;

}

}

}

Работа программы





