PerformanceLogger

# Introduction

This document describes the PerformanceLogger .NET component that helps developers adding performance information to their applications.

This document refers to the PerformanceLogger usage done in the Center system. We will often refer to samples and concepts specific to the Center world (more specifically, the Itcs subsystem). Anyway, the concepts can be easily adapted and/or extended to other contexts.

# Overview

Performance information is stored using two simple concepts: **sessions** and **markers**.

A **session** defines a period during which the performance is controlled.

For each session, multiple **markers** are stored when the application reaches some notable “checkpoints” of the execution logic.

Using the given API it is possible to define the markers and their specific properties (see later for details).

Persistence of the collected information is delegated to NLog. Using its configuration it is possible to store the information in different useful ways (text, CVS, database).

The preferable way to store the performance information is a database, which keeps information in a well-defined structure and provides developers with the possibility to query data to retrieve statistics.

Additionally, we defined a PowerShell script that performs some queries to extract simple statistics from the stored markers.

# Database

The PerformanceLogger uses NLog to log performance information to a database.

By default the database is named ‘PerformanceLogger’ and contains two tables: Sessions and Markers.

## Sessions

The Sessions table stores information about different periods of time under control. Periods can be defined according to the machine under test, the version of the software, etc.

Each session defines the following properties:

* **Id**: the identifier of the session. This value is needed for configuration (see later)
* **StartDate**: the date when the session was started (created)
* **Machine**: the name of the machine where tests are run
* **Description**: an optional description for the session

## Markers

For each session, using the API of the component it is possible to store different Markers.

A marker is a “checkpoint” reached by the application(s) at a specific time.

* **Id**: value internally used to identify markers; not actually used outside the database
* **SessionId**: reference to the session the marker belongs to
* **Category**: property used to classify the markers in a given context; an example could be the category “Itcs” to group all Itcs-related markers
* **Tag**: a string describing the marker/checkpoint
* **TickCount**: the value of the tick count on the machine

Other additional columns can be added according to context-specific requirements. For instance, for the Itcs logging a “Unit” column was added to distinct messages sent to different units (but sharing the same Tag in code).

# Usage

## Session setup

To create a new session, a StoredProcedure must be manually used before starting the performance monitoring activity. Usage example:

DECLARE @RC int

DECLARE @MachineName varchar(160)

DECLARE @Description varchar(500)

-- TODO: Set parameter values here.

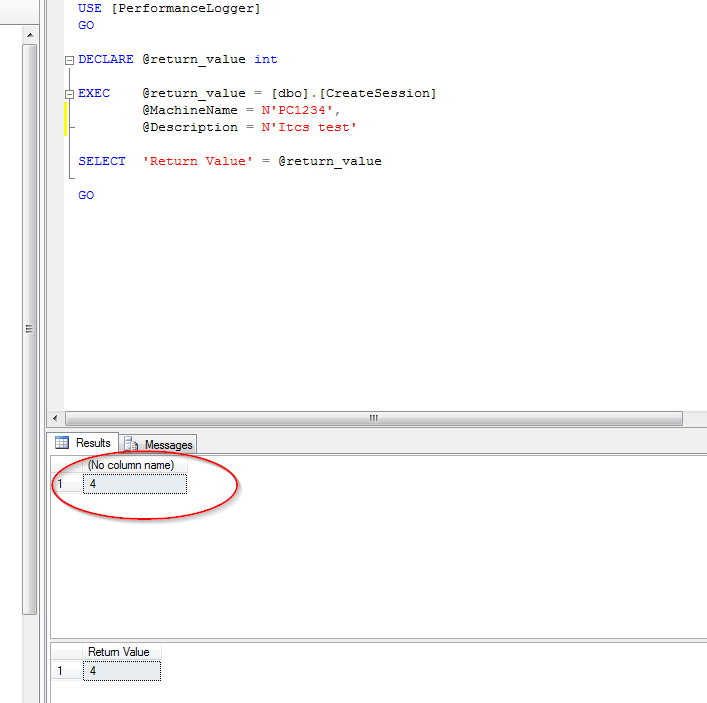
EXECUTE @RC = [PerformanceLogger].[dbo].[CreateSession]

@MachineName

,@Description

GO

The output of the execution gives the value of the Id assigned to the session. This value is needed later for the configuration of the system.



## Application configuration

The NLog configuration must be set to save information. When using the database, it is important that the application process has writer permissions on it. Please check the Installation section for more information.

The following fragment shows a possible target configure to log markers to the database.

  <nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <variable name="logDirectory" value="C:\Temp\Gorba\Center\Logs\BackgroundSystem"/>

    <targets async="true">

      <target

        name="file"

        xsi:type="File"

        fileName="${logDirectory}\BackgroundSystem\_${shortDate}.log"

        layout="${longdate} ${uppercase:${level}} ${logger} [${threadid}:${threadname}] ${message}${onexception: ${newline}${exception:format=tostring}}"

        createDirs="true"

        archiveAboveSize="5242880"

        archiveNumbering="Sequence"

        concurrentWrites="true"

        maxArchiveFiles="20"

        keepFileOpen="false"

        />

      <target xsi:type="Database" name="performance" connectionString="Data Source=.;Initial Catalog=PerformanceLogger;Integrated Security=True"

              commandText="[PerformanceLogger].[dbo].[AddMarker] @SessionId = 1, @Category = @Category, @MarkerId = @MarkerId, @TickCount = @TickCount, @Tag = @Tag, @Unit = @Unit"

            >

        <parameter name="Category" layout="${event-context:item=Category}" />

        <parameter name="MarkerId" layout="${event-context:item=MarkerId}" />

        <parameter name="TickCount" layout="${event-context:item=TickCount}" />

        <parameter name="Tag" layout="${event-context:item=Tag}" />

        <parameter name="Unit" layout="${event-context:item=Unit}" />

      </target>

      <target xsi:type="Debugger"

          name="debugger"

          layout="${time} ${uppercase:${level}} &lt;${logger:shortName=true}&gt; ${message}${onexception: ${newline}${exception:format=tostring}}"

 />

    </targets>

    <rules>

      <logger name="\*" minlevel="Trace" writeTo="file" />

      <logger name="\*" minlevel="Debug" writeTo="debugger" />

      <logger name="Gorba.Common.Utility.Core.Performance.NLogPerformanceLogInfoWriter" minlevel="Debug" writeTo="performance" />

    </rules>

  </nlog>

Important: the SessionId value in the Database target (set to 1 in this example) must be changed with the value for the session that should be used (see previous chapters for details).

Please also notice how this example uses a custom parameter (Unit) specific for the context (Center Itcs in this case).

## Add markers

Use the static method available on the **PerformanceLogger** class to add a new marker:

public static void Mark(string category, string tag, int id, params KeyValuePair<string, string>[] properties)

The **category** is used to logically group the markers

The **tag** is used to describe the entry. It is optional in the database, but anyway strongly suggested

The **id** is an integer that identifies the marker. It can be a hash code generated in the code according to the contextual information

The properties are string/string key value pairs to pass additional context items to NLog. By default, the following properties are already added by the Mark method itself:

* Category
* Tag (if not null or empty)
* MarkerId
* TickCount

The following example shows how to add an additional parameter:

PerformanceLogger.Mark(

                "Itcs", "TimetableDataBaseMessage Out", id, new KeyValuePair<string, string>("Unit", message.Address));

# Installation

One or two steps are needed to configure the PerformanceLogger:

1. Setup the NLog config. This step is required.
2. Setup the database. This step is needed only if you use the database to store the markers.

## Database setup

To setup the database, you need to create a database being able to store the information. A good example is the CreatePerformanceLoggerDatabase.sql script ($/Gorba/Main/Center/Common/Deploy/Source/CreateScripts/CreatePerformanceLoggerDatabase.sql).

There are a few important things to remark:

The user configured in NLog for the database connection must have the required permissions on the database.  
The script uses the AppPool user to login to the database, setting the Integrated Security for the connection string. This is the suggested way to be used when working with IIS application. Of course, the IIS process must run under that user.

Be sure to run the creation script with the correct permissions on the master database (check the file for further information).

Remark: when you create an AppPool, a new user is automatically created. For instance, for the AppPool BackgroundSystem, the user [IIS AppPool\BackgroundSystem] is created and used by default by the application (ApplicationPoolIdentity set in the advanced settings of the Application Pool).

# Appendix A – Remarks

## Database

All times stored in the database are in UTC format.

## NLog

There could be several issues while trying to store information into the database using NLog. In order to be able to understand what’s going wrong, please check http://nlog-project.org/wiki/Internal\_logging.