大家一定用过Profiler工具，我们可以用它来对**[SQL](javascript:;" \t "_self)** [**Server**](javascript:;)建立**[trace](javascript:;" \t "_self)**来**[监测](javascript:;" \t "_self)**某些感兴趣的事件，也可以replay抓到的trace来**[诊断](javascript:;" \t "_self)**是哪些SQL语句的执行造成你的SQL Server耗费了大量的CPU资源。但Profiler是个GUI程序，有没有办法通过程序来抓trace和重放trace呢？也许有些读者会想到用SQLCMD.exe执行sp\_trace\_create等存储过程来操作，但那毕竟还是有些麻烦，这里我们要介绍的Trace Management [**Object**](javascript:;)（TMO）则是.NET对象，你可以把它理解成trace/replay的API，你可以非常方便地在你的.NET程序中使用。请注意由于SQL Express版本不支持trace，因而TMO对象也无法在SQL Express版本上运行，即使是SQL Server [**2008**](javascript:;) Express with Advanced Services也不支持。

在SQL Server [**2005**](javascript:;)里TMO对象被实现在了Microsoft.SqlServer.ConnectionInfo.dll里，在SQL Server 2008里TMO对象则被移到了Microsoft.SqlServer.ConnectionInfoExtended.dll里，但仍然在Microsoft.SqlServer.Management.Trace命名空间里。下面我们将以SQL Server 2008为例。

用VS2005新建一个Visual C#的Console Application工程，在Project菜单里点击Add Reference…增加对下表几个组件的引用：

|  |  |
| --- | --- |
| 组件 | 描述 |
| Microsoft.SqlServer.ConnectionInfo | 需要SqlServerInfo类来建立对SQL的连接 |
| Microsoft.SqlServer.ConnectionInfoExtended | TMO对象在这个Assembly里 |
| Micrososft.SqlServer.Management.Sdk.Sfc | SQL Server 2008里很多组件是基于它建立，所以必须增加这个引用 |

组件描述Microsoft.SqlServer.ConnectionInfo 需要SqlServerInfo类来建立对SQL的连接 Microsoft.SqlServer.ConnectionInfoExtended TMO对象在这个Assembly里 Micrososft.SqlServer.Management.Sdk.Sfc SQL Server 2008里很多组件是基于它建立，所以必须增加这个引用

<http://www.codeplex.com/>上SQL Server 2008的Samples里有个Readme\_Tracer的例子（<http://www.codeplex.com/MSFTEngProdSamples/Release/ProjectReleases.aspx?ReleaseId=18651>），这个例子使用Standard.tdf模板启动一个live trace，trace的内容将打印在Console窗口上，但把结果打印到Console窗口上非常乱，而且也没有太大的实用价值，大家有兴趣可以去参考一下。本文第一部分将介绍一个capture trace的示例，和Readme\_Tracer有点类似，但我们会把trace结果输出到trace文件，第二部分将介绍一个replay trace的示例，这也是Profiler最常用的两个功能。

**Capture trace示例**

这个例子模仿你使用Profiler工具监测SQL Server操作的过程，程序启动一个trace，将抓到的trace event输出到文件中，等待60秒后退出。读者可以尝试将TraceFile类改为TraceTable类来输出到**[数据库](javascript:;" \t "_self)**表中。下面是详细的步骤和描述：

1、TraceServer类代表连接到SQL Server Instance的一个trace，下面的**[代码](javascript:;" \t "_self)**演示了如何创建一个TraceServer对象。代码前两行使用Windows认证方式建立一个SqlConnectionInfo对象，你也可以通过提供用户名/密码的方式建立这个对象，InitializeAsReader函数的第二个参数是trace模板，这里使用的TSQL\_Replay模板，你需要根据你的SQL Server安装目录进行修改。

SqlConnectionInfoconnInfo=newSqlConnectionInfo(".");  
connInfo.UseIntegratedSecurity=true;  
TraceServertraceServer=newTraceServer();  
traceServer.InitializeAsReader(connInfo,@"C:\Program Files\Microsoft SQL Server\100\Tools\Profiler\Templates\Microsoft SQL Server\100\TSQL\_Replay.tdf");

2、TraceFile类代表一个trace文件，它既可以是capture trace的输出文件，也可以是replay trace的输入文件，下面的代码将traceFile对象设为traceServer所代表的trace的输出文件，最后一行为traceFile增加了一个WriteNotify事件的event handler，通过它我们将对输出做一些过滤。

TraceFiletraceFile=newTraceFile();  
traceFile.InitializeAsWriter(traceServer,@"d:\tracefile.trc");  
traceFile.WriteNotify +=newWriteNotifyEventHandler(WriteHandler);

3、这两个对象建立完毕，此时trace已经开始了，但tracefile.trc却始终是0字节，为什么呢？因为你还需要调用TraceFile类的Write函数来输出，但Write函数有两个问题，一是调用一次只输出一个trace event，你需要不停地调用它；二是Write函数是同步的，如果没有可以输出的内容的话它会阻塞，所以你需要另起一个线程来调用Write函数。下面的代码启动WriteTraceProc线程并在60秒后结束capture trace。请注意traceServer.Close()必须在thread.Join()之前调用，否则WriteTraceProc线程可能会一直阻塞在Write函数调用上。

Threadthread =newThread(WriteTraceProc);  
thread.Start(traceFile);//pass traceFile as parameter  
Thread.Sleep(60000);  
lock(flagLock)  
{  
 exitFlag=true;  
}  
traceServer.Close();  
thread.Join();  
traceFile.Close();

4、接着是flagLock，exitFlag的定义及WriteTraceProc的代码，WriteTraceProc将持续调用Write直到exitFlag被主线程置为true。

privatestaticobjectflagLock=newobject();  
private staticboolexitFlag =false;  
privatestaticvoidWriteTraceProc(objectobj)  
{  
 TraceFiletraceFile = (TraceFile)obj;  
 while(true)  
 {  
 lock(flagLock)  
 {  
 if(exitFlag)  
 break;  
 }  
 traceFile.Write();  
 }  
}

5、最后是WriteHandler的代码，我们将使用EventClass列来过滤所有Audit Login及Audit Logout事件，你可以根据需要设置你的过滤条件。由于Books Online没有提供详细的文档，你也许需要使用IDataRecordChanger接口的GetName()来枚举所有你能使用的列。

privatestaticvoidWriteHandler(objectsender,TraceEventArgsargs)  
{  
 IDataRecordChangerrecordChanger = args.CurrentRecord;  
 stringeventClass = (string)recordChanger["EventClass"];  
 if(eventClass.StartsWith("Audit"))  
 args.SkipRecord =true;  
}

在这篇文章中我们将介绍一个replay [**trace**](javascript:;)的示例，通过重放抓到的trace文件来**[诊断](javascript:;" \t "_self)**应用程序在[**SQL**](javascript:;) [**Server**](javascript:;)上运行是否有问题。

**Replay trace示例**

这个例子模仿你使用Profiler工具对抓到的trace文件进行重放，从而对SQL Server及你的应用程序进行诊断的过程。下面是详细的步骤和描述。

1、TraceReplay类是对trace file或trace table进行重放的关键类，你需要设置TraceReplay的三个属性，Connection属性代表你将在哪个**[数据库](javascript:;" \t "_self)**实例上进行重放，Source属性代表重放哪个文件或数据表，OutputFile代表重放的结果将输出到哪个文件，你也可以使用OutputTable属性指定将重放结果输出到数据表中。下面的**[代码](javascript:;" \t "_self)**将使用capture trace例子里抓到的trace file作为重放的输入并将结果存为traceoutput.trc文件。

TraceReplayreplay =newTraceReplay();  
SqlConnectionInfoconnInfo =newSqlConnectionInfo(".");  
connInfo.UseIntegratedSecurity =true;  
replay.Connection = connInfo;  
TraceFiletraceFile =newTraceFile();  
traceFile.InitializeAsReader(@"c:\tracefile.trc");  
replay.Source = traceFile;  
TraceFiletraceFileOutput =newTraceFile();  
traceFileOutput.InitializeAsReplayOutputWriter(@"c:\replayoutput.trc");  
replay.OutputFile = traceFileOutput;

2、TraceReplay类还有个比较重要的属性Options，你可以通过设置Options属性来控制如何重放。Options属性是个TraceReplayOptions对象，Profiler工具中重放配置对话框上的很多配置项都可以通过Options属性来设置。下面的代码中设置了3个属性，Mode设置重放模式为连接层面上同步（另一个模式为SequentialReplay，指在所有连接上完全同步），NumberOfReplayThreads设置重放的线程数为2，KeepResults将控制在重放结果文件中是否出现SQL等的执行结果，比如你的trace里有select 1语句，那设置KeepResults为false将避免在结果文件中出现select 1返回的result set。

replay.Options.Mode =ReplayMode.ConnectionLevelSync;  
replay.Options.NumberOfReplayThreads = 2;  
replay.Options.KeepResults=false;

3、TraceReplay类提供了5个事件，你可以通过提供你自己的event handler来进一步控制重放操作，其中最有用的event是ReplayEvent，它将在每个event被重放之前调用，你可以控制是否跳过该event的重放。下面的代码在设置了ReplayEvent的处理函数后启动重放。请注意Start函数是同步操作，它将一直阻塞直到重放结束，所以如果你需要在中途停掉的话，你可以在event handler里或者其它线程中调用Stop函数。

replay.ReplayEvent +=newReplayEventHandler(ReplayHandler);  
replay.Start();  
traceFile.Close();

4、这里我们的ReplayHandler处理函数将对SPID进行过滤，所有SPID不是54的事件都将被忽略。代码中空的catch是为了避免某些event不含有SPID列会造成读取该列失败。

privatestaticvoidReplayHandler(Objectsender,ReplayEventArgsargs)  
{  
 IDataRecordChangerrecordChanger = args.CurrentRecord;  
 intspid = 0;  
 try  
 {  
 spid = (int)recordChanger["SPID"];  
 }  
 catch{ }  
 if(spid != 54)  
 args.SkipRecord =true;  
}

5、Replay的结果文件可以用Profiler工具打开查看，当然你仍然可以通过TraceFile打开，只要简单的调用InitializeAsReader然后循环调用Read即可，下面的代码将打出TextData列的内容。

TraceFiletraceFileOutput=newTraceFile();  
traceFileOutput.InitializeAsReader(@"c:\replayoutput.trc");  
while(traceFileOutput.Read())  
{  
 Console.WriteLine(traceFileOutput["TextData"]);  
}

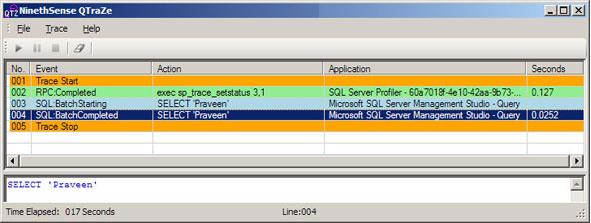
**MS SQL Server Profiler with .NET**

By [**NinethSense**](http://www.codeproject.com/script/Membership/View.aspx?mid=3045214) | 23 Aug 2007 Part of [The SQL Zone](http://www.codeproject.com/Zones/SqlServer/). http://www.codeproject.com/Zones/SqlServer/Images/Red_Gate_Logo_48x24.gif

**This article describes how to develop a Microsoft SQL Profiler with .NET.**

Top of Form





**Introduction**

Microsoft SQL Server Profiler is a helper for developers which is a client tool that comes with SQL Server. MS SQL Server Express edition does not come with SQL Profiler bundled. We use this tool to trace through queries. I mostly use this for peer testing as well as when the customer reports some bugs. Well, if you are a 'standard' SQL coder, you won't need to use these tools. May be you have already seen my article on [SQL good practices](http://www.codeproject.com/KB/architecture/SQLGoodPractices.aspx).

This article is an introduction to the implementation of a 'profiler like thing' with .NET. I would like to call it 'SQL Tracer' since it is out of the scope of this page to develop all the functionalities of a SQL Profiler. I have chosen C# for the demonstration.

**Background**

I was very much satisfied with the SQL Profiler which is available with Microsoft SQL Server 2000. But the one that comes with SQL Server 2005 seems a little bit slow. It inspired me to develop a fast query tracer tool.

**Important**

You must have Microsoft SQL Profiler components installed in your machine. You may be asking why we need this new tool if we already have the MS SQL Profiler. Note that the one I explain here is not an alternative for MS SQL Profiler. This is a handy tool with very basic functionalities. As a result, this tool gives you fast results. More than that, this article is for educational purposes.

**Using the code**

First... *Add Reference* to **Microsoft.SqlServer.ConnectionInfo.**

From this, we will get two namespaces:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/database/TSQL-profiler.aspx)

using Microsoft.SqlServer.Management.Trace;

using Microsoft.SqlServer.Management.Common;

For this example, I would recommend a ListView control since it gives the look and feel of the real Microsoft SQL Server Profiler.

**TraceServer**

The TraceServer class acts as a representation of a new SQL Server Trace. More information is available [here](http://technet.microsoft.com/en-us/library/microsoft.sqlserver.management.trace.traceserver.aspx).

**Trace file - Trace Definition File - .tdf**

You need to create a *.tdf* file, which is a template file. You can either create a new *.tdf* by using *Save as* option from the SQL Server Profiler itself, or you can use the default ones available on your installation folder, which is usually - *E:\Program Files\Microsoft SQL Server\90\Tools\Profiler\Templates\Microsoft SQL Server\80\\*.tdf*.

**ConnectionInfoBase**

With this class, we will initialize the server host, username, etc. It usually looks like this:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/database/TSQL-profiler.aspx)

ConnectionInfoBase conninfo = new SqlConnectionInfo();

((SqlConnectionInfo)conninfo).ServerName = "MyComputerNameOrIP";

((SqlConnectionInfo)conninfo).UserName = "PraveenIsMyUsername";

((SqlConnectionInfo)conninfo).Password = "MyPassword";

((SqlConnectionInfo)conninfo).UseIntegratedSecurity = false;

More information about this class is available [here](http://msdn2.microsoft.com/en-us/library/microsoft.sqlserver.management.common.connectioninfobase.aspx).

**InitializeAsReader**

This method is used to initialize an object for reading from the trace log file or server. E.g.:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/database/TSQL-profiler.aspx)

TraceServer trace = new TraceServer();

trace.InitializeAsReader(conninfo, "mytracetemplate.tdf");

InitializeAsReader causes the initialization and *start*ing of the tracing operation.

**Reading trace information**

trace.Read() is used to read trace information from SQL Server. You can put a *loop* to fetch all the trace information. Like this:

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/database/TSQL-profiler.aspx)

while (trace.Read()) {

*//Statements;*

}

Inside this loop, you can *display* status information in a ListView. The trace object contains all the needed properties.

trace["EventClass"] contains information like *ExistingConnection, Audit Login, Audit Logout, RPC:Completed, Trace Start* etc. If you are a SQL Profiler user, then you are already familiar with these messages.

trace["TextData"] is the element which contains the queries which are being executed.

Like this, we have trace["ApplicationName"], trace["Duration"] etc. also available. These elements are defined in your *.tdf* file. So investigate it. trace.FieldCount will give you the number of fields available. Since this article is for intermediate users and you know about fetching the values from collections etc., I will not mention it here.

**Use threading**

Since trace.Read() will not give you control to do your other tasks, there is a chance you will feel like your application died. So, use Thread.

**Start, Pause, and Stop**

You can control the tracing by applying the trace.start(), trace.pause(), and trace.stop() methods.

Do not forget to use trace.close() after use. Standard practice anyway.

**Need a sample application?**

Unfortunately, I do not have a stable sample application to provide. I will upload it once I get one.

**License**

This article, along with any associated source code and files, is licensed under [The Code Project Open License (CPOL)](http://www.codeproject.com/info/cpol10.aspx)

**About the Author**

Bottom of Form