Лекция 3. Трассировка и отладка в .NET

С помощью трассировки можно просматривать сообщения из работающего приложения. Чтобы получить информацию о работающем приложении, его можно запустить в отладчике. Во время отладки приложение можно выполнять пошагово и помещать точки останова на определенных строках и по достижению определенных условий. Проблема, связанная с отладкой, состоит в том, что программа с рабочим кодом может вести себя иначе, чем программа с отладочным кодом.

Например, когда выполнение программы прерывается в точке останова, другие потоки приложения также приостанавливаются. Кроме того, в рабочей сборке вывод компилятора оптимизирован, и потому могут проявляться другие эффекты. Поэтому существует потребность в получении информации и от рабочей сборки. Трассировочные сообщения записываются как отладочным, так и рабочим кодом.

Ниже описан сценарий, демонстрирующий пользу трассировки. После того, как приложение развернуто, оно успешно выполняется на одной системе, в то время как на другой системе периодически возникают проблемы. После включения подробной трассировки система с проблемами предоставит детальную информацию о том, что происходит внутри приложения. Система, работающая без проблем, поддерживает трассировку, сконфигурированную только на перенаправление сообщений об ошибках в систему протоколирования событий Windows. Критичные ошибки просматриваются системным администратором.

Накладные расходы, связанные с трассировкой, очень невелики, потому что уровень трассировки конфигурируется только при необходимости.

Архитектура трассировки состоит из четырех основных частей:

Источник (source)

Источник информации трассировки. Вы используете его для отправки трассировочной информации.

Переключатель (switch)

Определяет уровень информации для протоколирования. Например, можно запрашивать только информацию об ошибках или детальную многословную информацию.

Слушатели (listener)

Определяют, куда должны быть записаны сообщения трассировки.

Фильтры (filter)

Слушатели могут иметь присоединенные фильтры (filter). Фильтр определяет, как трассировочные сообщения должны быть записаны слушателем. Таким

образом для одного и того же источника можно иметь различные слушатели, которые пишут различные уровни информации.
Класс TraceSource использует переключатель для определения протоколируемой информации. У TraceSource есть коллекция TraceListenerCollection, ассоциированная с ним, куда отправляются сообщения трассировки. Коллекция состоит из объектов TraceListener, и каждый слушатель имеет подключенный к нему TraceFilter.
Многие технологии .NET используют источники трассировки, которые понадобится просто включить, чтобы увидеть, что происходит. Например, в WPF определяет помимо прочих источники с именами System.Windows.Data, System.Windows.RoutedEvent, System.Windows.Markup, System.Windows.Media.Animation. Однако для включения трассировки WPF необходимо не только сконфигурировать слушателей, но также установить внутри ключа реестра HKEY_CURRENT_USER\Software\MicrosoftTracing\WPF новый ключ DWORD по имени ManagedTracing со значением 1. Классы из пространства имен System.Net используют источник трассировки System.Net, а WCF — источники трассировки System.ServiceModel и System.ServiceModel.MessageLogging.
Источники трассировки
Трассировочные сообщения записываются с помощью класса TraceSource. Трассировка требует установки флага Trace компилятора. В проекте Visual Studio флаг Trace по умолчанию установлен для отладочных и рабочих сборок, но это можно изменить через свойства Build (Сборка) проекта.
При записи трассировочных сообщений класс TraceSource использовать труднее по сравнению с классом Trace, однако он предлагает больше опций. Для записи трассировочных сообщений потребуется создать экземпляр TraceSource. В конструкторе должно быть определено имя источника трассировки. Метод TraceInformation() пишет информационное сообщение в вывод трассировки. Вместо простой записи информационных сообщений метод TraceEvent() требует значения перечисления TraceEventType, которое определяет тип трассировочного сообщения.
TraceEventType.Error указывает, что сообщение является сообщением об ошибке. Определив переключатель трассировки, можно видеть только сообщения об ошибках. Второй аргумент метода TraceEvent() требует идентификатора. Идентификатор может использоваться внутри самого приложения.
Например, идентификатор 1 может служить для обозначения входа в метод, а 2 — для выхода из метода. Метод TraceEvent() перегружен, поэтому TraceEventType и идентификатор — это единственные обязательные параметры.

В третьем параметре перегруженного метода можно передать сообщение, записанное в трассировочную информацию. TraceEvent() также поддерживает передачу форматной строки с любым количеством параметров, подобно тому, как это делается в Console.WriteLine(). Метод Transformation() не делает ничего помимо вызова TraceEvent() с идентификатором 0. Transformation() — лишь упрощенная версия TraceEvent().

В методе TraceData() можно передать вместо сообщения любой объект, например, экземпляр исключения. Чтобы убедиться, что данные записаны слушателями и не остались в памяти, необходимо выполнить Flush(). Если источник больше не нужен, можно вызвать метод Close(), который закроет все слушатели, ассоциированные с источником трассировки. Метод Close() также вызывает Flush().

```
internal static TraceSource trace =
     new TraceSource("MyConsoleApplication");

static void TraceSourceDemo1()
{
    trace.TraceInformation("Info Message");
    trace.TraceEvent(TraceEventType.Error, 3, "Error Message");
    trace.TraceData(TraceEventType.Information, 2, "data1", 4, 5);
    trace.Flush();
    trace.Close();
}
```

Внутри приложения можно использовать различные источники трассировки. Для различных библиотек имеет смысл определить разные источники, чтобы для разных частей приложения можно было включать различные уровни трассировки. Чтобы использовать источник трассировки, вы должны знать его имя. Обычно в качестве такого имени используется имя сборки.

Перечисление TraceEventType, значение которого передается в качестве аргумента TraceEvent(), определяет следующие уровни для указания серьезности проблемы: Verbose, Information, Warning, Error и Critical. Уровень Critical определяет фатальную ошибку или крах приложения, а Error — восстановимую ошибку. Сообщения трассировки на уровне Verbose предоставляют детализированную отладочную информацию.

TraceEventType также определяет уровни действия Start, Stop, Suspend и Resume. Эти уровни определяют временные события внутри логической операции. В приведенном виде код не отображает никаких трассировочных сообщений, поскольку переключатель, ассоциированный с источником трассировки, выключен.

Переключатели трассировки

Для включения или отключения трассировочных сообщений можно сконфигурировать переключатель трассировки. Переключатели трассировки — это классы, унаследованные от абстрактного базового класса Switch. Производными классами являются BooleanSwitch, TraceSwitch и SourceSwitch. Класс BooleanSwitch может быть включен и выключен, а другие два предоставляют ранговый уровень. Один ранг определен перечислением SourceLevels. Чтобы сконфигурировать переключатели трассировки, необходимо знать значения, ассоциированные с перечислением SourceLevels. Класс SourceLevels определяет значения Off, Error, Warning, Info и Verbose.
Переключатель трассировки можно ассоциировать программно, установив свойство Switch объекта TraceSource. Здесь ассоциированный переключатель типа SourceSwitch имеет имя MyConsoleApplication и уровень Verbose:

```
internal static TraceSource trace =
    new TraceSource("MyConsoleApplication")
            {
                Switch = traceSwitch
            };

internal static SourceSwitch traceSwitch =
    new SourceSwitch("MyConsoleApplication")
       {
          Level = SourceLevels.Verbose
        };
```

Установка уровня Verbose означает, что записываться должны все трассировочные сообщения. Если установить значение Error, то будут фиксироваться только сообщения об ошибках. Установка значения Information означает, что должны записываться сообщения об ошибках, предупреждающие и информационные сообщения. Трассировочные сообщения можно просматривать в окне Output (Вывод) отладчика.

Обычно уровень переключателя нужно изменять не перекомпиляцией приложения, а внесением изменений в конфигурацию. Трассировка может быть настроена в элементе <system.diagnostics> конфигурационного файла приложения. Источник трассировки определен элементом <source>, который является дочерним для <sources>. Имя источника в конфигурационном файле должно в точности соответствовать имени источника в коде. Здесь источник трассировки имеет переключатель типа System.Diagnostics.SourceSwitch, ассоциированный с именем MySourceSwitch. Сам переключатель определен в разделе <swithes> и уровень переключателя установлен в verbose:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
```

```
  <system.diagnostics>
    <sources>
      <source name="MyConsoleApplication" switchName="MySourceSwitch"
              switchType="System>diagnostics.SourceSwitch"></source>
    </sources>
    <switches>
      <add name="MySourceSwitch" value="Verbose"/>
    </switches>
  </system.diagnostics>
</configuration>
```

Теперь можно изменить уровень трассировки, просто изменяя конфигурационный файл, без необходимости перекомпиляции кода. После изменения конфигурационного файла приложение должно быть перезапущено.

Сейчас трассировочные сообщения пишутся в окно Output отладчика Visual Studio. Добавляя слушателей трассировки, это можно изменить.

Слушатели трассировки

По умолчанию трассировочная информация пишется в окно Output отладчика Visual Studio. Изменяя конфигурацию приложения, этот вывод можно направить в другое место. Куда будет записан вывод трассировки — зависит от слушателей. Слушатель наследуется от абстрактного базового класса TraceListener.

В .NET поставляется несколько слушателей трассировки для записи событий трассировки в различные места. Для файловых слушателей трассировки используется базовый класс TextWriterTraceListener, а также производные классы XmlWriterTraceListener — для записи XML-файлов и DelimitedListTraceListener — для записи файлов с разделителями. Запись в журнал событий осуществляется с помощью либо EventLogTraceListener, либо EventProviderTraceListener.

Слушатель EventProviderTraceListener использует формат файла событий, который появился в Windows Vista. Можно также комбинировать веб-трассировку с трассировкой System.Diagnostics и записывать сведения в файл trace.axd.

.NET Framework предлагает множество слушателей, в которые может быть записана трассировочная информация. В случае если слушатели не отвечают существующим требованиям, можно написать собственный слушатель, наследуя специальный класс от базового класса TraceListener. С помощью такого специального слушателя можно, например, записывать трассировочную информацию в веб-службу, отправлять сообщения на мобильный телефон и т.д.

Слушатель трассировки можно сконфигурировать программно, создав объект слушателя и присвоив его свойству Listeners класса TraceSource.

Однако обычно намного интереснее просто изменить конфигурацию, чтобы определить другой слушатель.

Слушатели можно сконфигурировать как дочерние элементы элемента <source>. В этом случае определяется тип класса слушателя и с помощью initializeData указывается направление вывода слушателя. В показанной ниже конфигурации определяются слушатели XmlWriterTraceListener для записи в файл demotrace.xml и DelimitedListTraceListener для записи в файл demotrace.txt:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.diagnostics>
    <sources>
      <source name="MyConsoleApplication" switchName="MySourceSwitch"
              switchType="System>diagnostics.SourceSwitch">
        <listeners>
          <add name="xmlListener"
               type="System.Diagnostics.XmlWriterTraceListener"
               traceOutputOptions="None"
               initializeData="c:/logs/mytrace.xml"></add>
          <add name="delimitedListener"
               delimiter=":"
               type="System.Diagnostics.DelimitedListTraceListener"
               traceOutputOptions="DateTime, ProcessId"
               initializeData="c:/logs/mytrace.txt"></add>
        </listeners>
      </source>
    </sources>
    <switches>
      <add name="MySourceSwitch" value="Verbose"/>
    </switches>
  </system.diagnostics>
</configuration>
```

С таким слушателем можно указывать дополнительную информацию, которая должна быть записана в журнал трассировки. Эта информация задается в XML-атрибуте traceOutputOptions и определена перечислением TraceOptions. Доступными значениями перечисления являются Callstack, DateTime, LogicalOperationStack, ProcessId, Threadld и None. В XML-атрибут traceOutputOptions можно добавлять сразу несколько необходимых значений, разделяя их запятыми, как в слушателе delimitedListener из предыдущего примера.