PART 7

測試與除錯篇

- Q75：如何讓除錯器輸出自訂的訊息？
- Q76：如何在程式執行時紀錄特定資訊以協助除錯？
- Q77：如何收集能夠重現錯誤環境的資料？

Q75 如何讓除錯器輸出自訂的訊息？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我的 ASP.NET 程式碼在開發的過程中逐漸變大, 而且也變得比較難以維護, 我想要利用除錯器來幫助我除錯程式, 並且在除錯器執行到一些特定的條件時, 能夠在除錯器上顯示一些訊息, 請問要如何做？

問題說明

除錯器 (Debugger) 和開發人員其實有著很密切的關係, 在撰寫程式時, 如果沒有除錯器的幫忙, 想要很快的找出程式的錯誤在哪裡, 其實是很困難的, 尤其是在程式碼很多 (成千上萬行) 時。除錯器可以讓開發人員設定中斷點 (breakpoint) 或是執行除錯程式 (Visual Studio 的除錯器) 這樣的工作, 還可以輸出變數的資料內容, 讓開發人員可以掌握目前程式執行時的變數變化如何, 以及邏輯上的處理是否正確等等。

.NET Framework 又將 Visual Studio 除錯器拉到更高的層次, 在類別庫中直接支援除錯以及呼叫除錯器的功能, 分別由 System.Diagnostics 命名空間的 Debug 類別以及 Debugger 類別來負責。

Debug 類別以一般性的訊息輸出爲主, 像是輸出字串的 Debug.Write()；輸出字串及斷行符號的Debug.WriteLine()；條件式輸出的Debug.WriteIf()以及Debug.WriteLineIf() 函式；評估執行結果的 Debug.Assert() 函式等。Debugger 類別則是以一般性的除錯器通訊與執行爲主, 像是令程式中斷的 Debugger.Break() 以及啓動除錯器的 Debugger.Lanuch() 函式。

在應用程式中, 最實用的應算是 Debug 類別以及它的姐妹類別 Trace (在「如何在程式執行時紀錄特定資訊以協助除錯」問答中會說明 Trace 類別) , 它可以在除錯器或記錄檔中輸出資料與訊息, 讓開發人員可以判斷與識別問題的發生所在, 以及可能發生的原因, 對於加速程式品質的改良, 會有正面的幫助。

Debug.Write() 系列函式的操作方法比較簡單, 而 Debug.WriteIf() 也只是依條件來輸出資料而已。

函式宣告：

```
Debug.Write(msg, category);
Debug.WriteIf(condition, msg, category);
Debug.WriteLine(msg, category);
Debug.WriteLineIf(condition, msg, category);
```

參數	說明
msg	訊息內容，可以是字串或物件，如果是給物件的話，則會輸出物件的字串輸出結果。
category	訊息的分類，可有可無。
condition	輸出的條件，只要這個條件是 true，就會輸出指定的訊息。

範例：

```
Debug.Write("輸出訊息");
Debug.Write("輸出訊息", "輸出訊息分類");
Debug.WriteLine("輸出訊息");
Debug.WriteLine("輸出訊息", "輸出訊息分類");
Debug.WriteLineIf((myObject.Value == 1), "輸出訊息");
Debug.WriteLineIf((myObject.Value == 1), "輸出訊息", "輸出訊息分類");
```

比較特殊的是 Debug.Assert(), 它需要在 Web.config 中設定, 可以設定是否顯示訊息方塊, 以及寫入的記錄檔等等, 在 ASP.NET 中, Debug.Assert()預設是不會顯示任何訊息。

```
<configuration>
  <system.diagnostics>
    <assert assertuienabled="false" logfilename="c:\log.txt"/>
  </system.diagnostics>
</configuration>
```

屬性	說明
assertuienabled	在 Debug.Assert()評估爲 false 時是否要顯示訊息方塊 (在 ASP.NET 中會失效)
logfilename	指定在 Debug.Assert()評估爲 false 時, 輸出訊息的記錄檔名稱, 預設是使用 DefaultTraceListener 來作爲輸出器, 需要是絕對路徑。

當 Debug.Assert()中評估的條件爲 false 時, 若在 Web.config 中有設定輸出記錄檔, 則 Debug.Assert()會將資料寫入記錄檔中, 包含了程式堆疊的訊息, 以及指定的訊息和細目訊息 (由 Debug.Assert()的第二個和第三個參數指定) 。

函式宣告：

```
Debug.Assert(condition);
Debug.Assert(condition, msg);
Debug.Assert(condition, msg, detailmsg);
```

參數	說明
condition	條件, 若爲 false 時才會輸出訊息。
msg	訊息。
detailmsg	更詳細的訊息。

如果想要將 Debug.Assert()的輸出, 轉向到特定的輸出器 (Listener) 時, 可以透過設定 Web.config 的方式, 加入想要的 Listener 即可, 目前 ASP.NET 支援幾種Listener：

DefaultTraceListener	預設的輸出器。
TextWriterTraceListener	以 TextWriter 爲主的輸出器, 可以指向特定的檔案。
EventLogTraceListener	以 Event Log 爲主的輸出器, 可以將訊息寫入事件記錄器中。
DelimitedTraceListener	輸出到文字檔, 但內容是以 Delimited 屬性所指定的分隔符號來分隔。
XmlWriterTraceListener	輸出到 XML 資料檔。

```
<system.diagnostics>
  <trace autoflush="true" indentsize="1" useGlobalLock="true">
    <listeners>
      <!-- 設定記錄的輸出器 -->
      <add name="WebPageTraceListener"
        type="System.Web.WebPageTraceListener, System.Web,
        Version=2.0.3600.0, Culture=neutral,
        PublicKeyToken=b03f5f7f11d50a3a"/>
      <add name="TextWriterListener"
        type="System.Diagnostics.TextWriterTraceListener, System,
        Version=2.0.3600.0, Culture=neutral,
        PublicKeyToken=b77a5c561934e089"
        initializeData="d:\log.txt" />
    </listeners>
  </trace>
  <assert assertuienabled="false" />
</system.diagnostics>
```

解決方案

可應用 System.Diagnostics 的 Debug 類別, 來做程式的訊息輸出與檢查工作。

Debug.Write() 等函式會將訊息輸出到除錯器的輸出視窗, 若有設定輸出時, 則也會一併寫入到輸出檔案。

若想要做檢查, 則 Debug.Assert()較適合, 它會輸出到輸出檔案中, 但不會輸出到除錯器的輸出視窗中。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-536: TS: Microsoft .NET Framework 2.0 Application Development Foundation 的下列主題：

- 在.NET Framework 應用程式中嵌入組態、診斷、管理與安裝功能。
  - 使用 System.Diagnostics 命名空間, 對.NET Framework 應用程式執行追蹤與除錯。
    - Debug 與 Debugger 類別。
    - XmlWriterTraceListener類別, DelimitedListTraceListener類別與EventlogTraceListener類別。

相關問題

- Q76：如何在程式執行時紀錄特定資訊以協助除錯？

Q76 如何在程式執行時紀錄特定資訊以協助除錯？

適用範圍：V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我的應用程式已經完成上線了, 並且是用 Release 模式來建置, 所以程式中都沒有除錯碼, 如果我想要在 Release 模式時輸出一些特定的資料來做除錯用, 要如何做？

問題說明

如果應用程式在除錯階段測試時, 大多可以很快的將問題找出來, 但如果是在發行 (Release) 階段時, 因爲除錯符號都被移除了, 所以無法很快的知道發生問題的程式在哪裡, 此時就需要依賴一些輸出的訊息來知道出問題的程式碼在哪個位置, 這樣可以縮短檢查的時間, 加快問題修補的速度。

在「如何讓除錯器輸出自訂的訊息」問答中, 筆者已說明了 Debug 類別, 它雖然好用, 但它只能在除錯模式中使用, 因此若想要在發行模式中提供與 Debug 模式相同的功能時, 就需要它的姐妹類別 Trace 了。

Trace 的使用法和 Debug 幾乎是一模一樣, 在 Debug 類別中的方法, 在 Trace 中都可以使用, 而且在發行階段的程式碼中, Trace 仍然有效, 若要在 ASP.NET 程式碼中使用 Trace, 請用 System.Diagnostics.Trace, 以免和 ASP.NET 另外提供的 Trace 名稱衝突。

Trace 函式宣告如下：

```
Trace.Write(msg, category);
Trace.WriteIf(condition, msg, category);
Trace.WriteLine(msg, category);
Trace.WriteLineIf(condition, msg, category);
```

參數	說明
msg	訊息內容，可以是字串或物件，如果是給物件的話，則會輸出物件的字串輸出結果。
category	訊息的分類，可有可無。
condition	輸出的條件，只要這個條件是 true，就會輸出指定的訊息。

範例：

```
Trace.Write("輸出訊息");
Trace.Write("輸出訊息", "輸出訊息分類");
Trace.WriteLine("輸出訊息");
Trace.WriteLine("輸出訊息", "輸出訊息分類");
Trace.WriteLineIf((myObject.Value == 1), "輸出訊息");
Trace.WriteLineIf((myObject.Value == 1), "輸出訊息", "輸出訊息分類");
```

如同 Debug.Assert()一樣, Trace 也有個 Assert()方法, 使用方式和 Debug.Assert()相同, 也具有評估爲 false 時即輸出訊息的能力。

除了 System.Diagnostics 本身的 Trace 功能外, 在 ASP.NET 中還另外提供了一種 Trace, 它是以 TraceContext 類別爲主的 Trace 機能, 會直接收集網頁中的要求與回應, 並且會收集事件執行的資料, 以及 HTTP 標頭的所有資訊, 不過這類型的 Trace 功能是需要在 Web.config 中啓用的。

```
<trace
  enabled="true|false" // 設定是否在 ASP.NET 中啓用 Trace 功能
  localOnly="true|false" // 設定是否只能在本機上看到
  pageOutput="true|false" // 設定是否要在網頁中輸出 Trace 資訊
  requestLimit="integer"  // 限制 Trace 的大小
  mostRecent="true|false" // 設定是否只顯示最近的 Trace 資訊
```

```
  // 設定是否要和 System.Diagnostics 的 Trace 整合
  writeToDiagnosticsTrace="true|false"
  traceMode="SortByTime|SortByCategory" // 設定輸出的排序方法
/>
```

TraceContext 的啓用方式除了 Web.config 以外, 如果只是想要針對單一頁面做 Trace, 可以只在想要做 Trace 的網頁的 Page 宣告中, 設定 Trace="true", 以啓用 Trace 的功能, 而如果是在 Web.config 中設定啓用, 而且設定 pageOutput="false"的話, 則可以透過根目錄中的 trace.axd 來瀏覽所有在網站中產生的 Trace 的資訊, trace.axd 是一個 HTTP Handler, 由 ASP.NET 內部維護的資料。

TraceContext 提供了 Warn() 及 Write() 兩個方法, 作用相同, 但唯一的差別是, TraceContext.Warn()輸出的是警告訊息, 在 Trace 記錄中是用紅色字體, 而 TraceContext.Write()輸出的只是一般性的資料, 字體是用黑色的, 如此可以很便利的分辨出哪些訊息是重要的, 哪些是一般性的訊息。

```
Trace.Warn("警告訊息");
Trace.Write("一般訊息");
```

如果想要將 TraceContext 的資訊, 輸出到輸出器時, 可以將 ASP.NET 的 Trace 功能和 System.Diagnostics 的 Trace 功能整合在一起, 只要在 Web.config 中加入 writeToDiagnosticsTrace="true"即可 (這個屬性是.NET 2.0 才新增的, .NET 1.x 沒有) , TraceContext.Warn()以及 TraceContext.Write()輸出的訊息也會一併寫入, 但如此一來, Trace 輸出的就只會有事件執行的資訊, 而不會有像 HTTP Header, 工作階段與控制項樹狀結構等較完整的資訊。

解決方案

在發行的程式碼中, 可以使用 Trace 功能來輸出特定的資訊, 也可以和 System.Diagnostics 的 Trace 輸出功能做整合, 並輸出到記錄檔中, 以備後續的除錯處理。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-536: TS: Microsoft .NET Framework 2.0 Application Development Foundation 的下列主題：

- 在.NET Framework 應用程式中嵌入組態、診斷、管理與安裝功能。
  - 使用 System.Diagnostics 命名空間, 對.NET Framework 應用程式執行追蹤與除錯。
    - Trace 類別。

亦可用來準備 Exam 70-528: TS: Microsoft .NET Framework 2.0 Web Client Development 的下列主題：

- 追蹤, 設定與部署應用程式。
  - Web 應用程式的最佳化與故障排除。
    - 使用 ASP.NET tracing 的功能進行故障排除。

相關問題

- Q75：如何讓除錯器輸出自訂的訊息？

Q77 如何收集能夠重現錯誤環境的資料？

適用範圍：V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我的應用程式已經完成上線了, 並且是用 Release 模式來建置, 但應用程式中有時候會三不五時的出現狀況, 請問有什麼樣的方法, 能夠有效的收集到能重現問題的資料？

問題說明

應用程式維護 (Maintenance) 與支援 (Support) 是每個開發人員在完成階段任務後, 都會經歷的步驟之一, 除了要保持應用程式的穩定與可用之外, 對於資料的處理以及正確性, 也都要嚴格把關 (如果有 DBA, 那麼這件事就是 DBA 的工作) , 以確保應用程式不會有 GIGO (Garbage-In, Garbage-Out) 的狀況, 也確保數據的正確的可信度。

只要有寫過程式的讀者應該都知道, 如果程式中有問題 (行話稱爲 Bug) 存在時, 想要修補 Bug 的唯一方法, 就是重現 (Reproduce) 這個 Bug 的環境, 如此才能夠用除錯器來檢查程式可能發生的問題, 進而修復 Bug。

但應用程式有時候就是因爲不明的原因而出現問題, 通常使用者只會狂 call 負責處理的人員而已, 也就是開發人員, 有時候在和使用者確定問題時, 會抓不到問題的核心, 因此, 如能確保取得較多的資訊, 事情處理起來會快一些。

通常在一個程式問題發生時, 使用者看到的訊息、使用者做了什麼事 (操作步驟與輸入資料) 以及當下的系統負載狀況等, 都會影響到問題的判斷與處理, 因此能夠問多一些資訊的話, 就盡量的問, 由使用者獲得的資訊, 可以用來判斷問題的位置 (某個功能) , 而且在某些情況下, 還可以判斷是否爲使用者操作上的問題。

接著, 檢視系統中有沒有一些資訊, 例如事件檢視器、效能檢視器、工作管理員圖表, 以及事先安排好的 Trace 資訊等等, 通常由系統記錄的資訊, 會比較詳細, 而且在重現問題時也會比較能夠下手, 像是 CPU/RAM 或 I/O 的問題, 都可以在這一層找出來, 此時就可以進一步檢查是哪些程式造成過量的 I/O。如果應用程式有資料庫的話, 可以利用 SQL Profiler 來檢查是否有 SQL 指令的耗時問題。

如果問題是發生在營運 (也就是一般作業) 期間, 那麼就可以利用模擬方法或是到資料庫中取出一段資料, 來做測試, 如此可以找出在營運期間會發生的問題, 有了資料後, 就可以配合除錯器來找出問題的發生點在哪裡, 並且修補問題。

任何的系統問題都會有其癥狀, 只要掌握住癥狀, 並且善加利用工具, 即可快速的找出問題的發生點加以修復。

解決方案

筆者依自身經驗, 提供一些常見的應用程式可能發生問題的癥狀與找問題的方法, 列爲一張表, 供讀者參考。

問題	癥狀	可能原因	線索
操作期間，程式發生錯誤問題	使用者看到錯誤訊息，或是錯誤重導向的網頁。	程式碼本身有 Bug, 或使用者操作錯誤。	使用者的回報，包括操作的流程、輸入的資料以及錯誤訊息等。
應用程式效能問題	處理速度變慢，或者回應速度變慢。	負載過重，系統資源不足，或者有程式佔用大量系統資源，亦有可能是程式中有問題。	利用效能監視器，網路監視器與工作管理員等工具判斷是硬體問題或軟體（程式）問題。若無法由軟體解決，就要由硬體下手。

問題	癥狀	可能原因	線索
資料庫效能問題	資料庫查詢變慢，容易 Timeout。	資料庫負載過重，或是資料庫伺服器資源太少，或查詢設計不良。	使用 SQL Server Management Studio, SQL Profiler 或效能監視器，對資料庫做監控，找出可能的原因，例如索引或查詢設計的問題。
安全性問題	由稽核資料或記錄中發現有不明的帳戶活動。	資料未更新，或是程式中有漏洞。	使用 SQL Profiler, 或者在程式中安插 Trace 功能執行進一步稽核，並檢查帳戶與權限設定。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-547: PRO: Designing and Developing Web Applications by using Microsoft .NET Framework 的下列主題：

- 應用程式的測試與穩定化
  - 解決 Bug
    - 調查已回報的 Bug。
    - 重現 Bug。
    - 評估 Bug 的影響程度, 以及修復 Bug 的時程與成本。
    - 修復 Bug。

相關問題

- Q80：如何監測 ASP.NET 應用程式的效能？