Log4Net使用详解（续）

标签： log4net layout thread exception string newline

2010-11-23 10:35 28903人阅读 评论(29) 收藏 举报

分类： C#基础（110） asp.net（103）

说明自从上次在2008年在博客上发表过有关log4net的用法介绍文章之后（网址：http://blog.csdn.net/zhoufoxcn/archive/2008/03/26/2220533.aspx），有不少朋友在博文下留言询问一些细节，现在就一些比较普遍的问题做一些稍微深入的解答，希望大家满意。

首先说明一点的是，log4net解决的问题是提供一个记录日志的框架，它提供了向多种目标写入的实现，比如利用log4net可以方便地将日志信息记录到文件、控制台、Windows事件日志和数据库（包括MS SQL Server, Access, Oracle9i,Oracle8i,DB2,SQLite）中，一般来说我们只需要提供一个描述性的字符串，然后log4net就会自动提供有关运行时的一些信息。

Log4Net的版本仍是1.2.10（2008年我写博文的时候也是这个版本），有.NET1.0和.NET1.1和.NET2.0版本，如果有正在使用高于.NET2.0开发的也不用担心，可以直接引用这个类库，像在.NET2.0中开发一样，它的网址是：http://logging.apache.org/log4net/

关于在**Web**中支持的问题

在我们开发项目时都会使用到config文件，可以在config文件中配置log4net。这一点Web项目和WinForm项目都是一样的。需要注意的是，因为在Web项目中一般以较低权限的角色来运行Web项目的，所以在使用文件型日志时要注意不要放在本项目根文件夹之外。

在**config**文件中的配置

要使用log4net，首先要在config文件的<configSections>节点中增加配置（如果没有这个节点请手动增加），如下：

```xhtml
[xhtml]
<configSections>
    <section name="log4net" type="System.Configuration.IgnoreSectionHandler"/>
</configSections>
```

除此之外，还要在顶级节点<configuration>下增加<log4net>子节点。在<log4net>节点下就可以增加<appender>子节点，每个<appender>子节点代表一种记录日志的方式（仅在这里配置了不代表启用了）。

具体说来有如下Appender:

AdoNetAppender：利用ADO.NET记录到数据库的日志。

AnsiColorTerminalAppender：在ANSI 窗口终端写下高亮度的日志事件。

AspNetTraceAppender：能用asp.net中Trace的方式查看记录的日志。

BufferingForwardingAppender：在输出到子Appenders之前先缓存日志事件。

ConsoleAppender：将日志输出到控制台。

EventLogAppender：将日志写到Windows Event Log.

FileAppender：将日志写到文件中。

LocalSyslogAppender：将日志写到local syslog service (仅用于UNIX环境下).

MemoryAppender：将日志存到内存缓冲区。

NetSendAppender：将日志输出到Windows Messenger service.这些日志信息将在用户终端的对话框中显示。

RemoteSyslogAppender：通过UDP网络协议将日志写到Remote syslog service。

RemotingAppender：通过.NET Remoting将日志写到远程接收端。

RollingFileAppender：将日志以回滚文件的形式写到文件中。

SmtpAppender：将日志写到邮件中。

TraceAppender：将日志写到.NET trace 系统。

UdpAppender：将日志connectionless UDP datagrams的形式送到远程宿主或以UdpClient的形式广播。

关于使用**log4net**中可能会使用到的一些参数

%m(message):输出的日志消息，如ILog.Debug(…)输出的一条消息

%n(new line):换行

%d(datetime):输出当前语句运行的时刻

%r(run time):输出程序从运行到执行到当前语句时消耗的毫秒数

%t(thread id):当前语句所在的线程ID

%p(priority): 日志的当前优先级别，即DEBUG、INFO、WARN…等

%c(class):当前日志对象的名称，例如：

%f(file):输出语句所在的文件名。

%l(line)：输出语句所在的行号。

%数字：表示该项的最小长度，如果不够，则用空格填充，如“%-5level”表示level的最小宽度是5个字符，如果实际长度不够5个字符则以空格填充。

下面以一个实际的例子来说明问题，比如在配置中有“%date [%thread] (%file:%line) %-5level %logger [%property{NDC}] - %message%newline”，那么实际的日志中会是如下格式：

“记录时间：2010-11-17 16:16:36,561 线程ID:[9] 日志级别：文件：所在行ERROR 出错类：Log4NetDemo.Program property:[(null)] - 错误描述：error System.Exception: 在这里发生了一个异常,Error Number:2036084948”

关于对数据库的支持

前面已经说过，log4net是支持包括MS SQL Server, Access, Oracle9i,Oracle8i,DB2,SQLite在内的数据库的，如果是文件型数据库（如Access或SQLite）的话就需要指定数据库文件的位置（在Web中最好指定在有读写权限的文件夹下，并且实现创建好表），如果是网络数据库就需要指定正确的数据库连接字符串。

比如要记录到Oracle数据库中，在配置文件中可以增加一个< appender>节点，配置如下：

```
[xhtml]
```

```
<appender name="AdoNetAppender_Oracle" type="log4net.Appender.AdoNetAppender">
      <connectionType value="System.Data.OracleClient.OracleConnection, System.Data.Oracle
      <connectionString value="data source=[mydatabase];User ID=[user];Password=
[password]" />
      <commandText value="INSERT INTO Log (Datetime,Thread,Log_Level,Logger,Message) VALUE
      <bufferSize value="128" />
      <parameter>
        <parameterName value=":log_date" />
        <dbType value="DateTime" />
        <layout type="log4net.Layout.RawTimeStampLayout" />
      </parameter>
      <parameter>
        <parameterName value=":thread" />
        <dbType value="String" />
        <size value="255" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%thread" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value=":log_level" />
        <dbType value="String" />
        <size value="50" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%level" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value=":logger" />
        <dbType value="String" />
        <size value="255" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%logger" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value=":message" />
        <dbType value="String" />
        <size value="4000" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%message" />
        </layout>
      </parameter>
    </appender>
```

当然从上面的配置中的SQL语句中可以看得出这个表的参数，日志表的创建语句如下：

```
[c-sharp]
create table log (
```

```
    Datetime timestamp(3),
    Thread varchar2(255),
    Log_Level varchar2(255),
    Logger varchar2(255),
    Message varchar2(4000)
    );
```

在本例中周公采用了将日志记录到SQLite这个单机数据库的方式，并且还将记录记录日志时的文件名和行号，创建SQLite的SQL语句如下：

```
[c-sharp]
CREATE TABLE Log (
    LogId        INTEGER PRIMARY KEY,
    Date        DATETIME NOT NULL,
    Level        VARCHAR(50) NOT NULL,
    Logger        VARCHAR(255) NOT NULL,
    Source        VARCHAR(255) NOT NULL,
    Message        TEXT DEFAULT NULL
    );
```

增加的< appender>节点配置如下：

```
[c-sharp]
<appender name="AdoNetAppender_SQLite" type="log4net.Appender.AdoNetAppender">
  <bufferSize value="100" />
  <connectionType value="System.Data.SQLite.SQLiteConnection, System.Data.SQLite, Version=
  <!--SQLite连接字符串-->
  <connectionString value="Data Source=c://log4net.db;Version=3" />
  <commandText value="INSERT INTO Log (Date, Level, Logger,Source, Message) VALUES (@Date,
  <parameter>
    <parameterName value="@Date" />
    <dbType value="DateTime" />
    <layout type="log4net.Layout.RawTimeStampLayout" />
  </parameter>
  <parameter>
    <parameterName value="@Level" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%level" />
    </layout>
  </parameter>
  <parameter>
    <parameterName value="@Logger" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%logger" />
    </layout>
  </parameter>
```

```
  <parameter>
    <parameterName value="@Source" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%file:%line" />
    </layout>
  </parameter>
  <parameter>
    <parameterName value="@Message" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%message" />
    </layout>
  </parameter>
</appender>
```

从上面的配置中可以看出插入数据的SQL语句及参数信息，下面的<parameter>节点就是指定插入的数据，比如我们指定SQL语句中的"@Source"参数来源于log4net中的"%file:%line"参数，也就是这两个参数用“:”用连接起来。

控制日志文件大小的问题

对于一个长时间使用并且有大量业务日志的系统来说，如果使用FileAppender将日志一直记录到一个文件中会引起性能低下的问题，我曾见过有个系统的日志文件达到了800多M，最后系统无法及时响应了，在这种情况下可考虑使用RollingFileAppender循环记录日志，一种是指定文件的最大长度，如果超过了就重新生成一个文件，如下面的配置：

```
[xhtml]
<appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
  <file value="RollingFileAppender_log.txt" />
  <appendToFile value="true" />
  <rollingStyle value="Size" />
  <maxSizeRollBackups value="10" />
  <maximumFileSize value="100KB" />
  <staticLogFileName value="true" />
  <layout type="log4net.Layout.PatternLayout">
    <conversionPattern value="%date [%thread] (%file:%line) %-5level %logger [%property{ND
  </layout>
</appender>
```

在上面的配置中，每个日志文件最大100KB，最大日志文件个数是10生成的日志文件名会是RollingFileAppender_log.txt.1, RollingFileAppender_log.txt.2 ... RollingFileAppender_log.txt.10，如果记录的日志超过10个，会从RollingFileAppender_log.txt.1开始覆盖。

还有一种方式就是按照日期记录日志，它的配置如下：

```xhtml
<appender name="RollingLogFileAppender_DateFormat" type="log4net.Appender.RollingFileAppen
  <file value="RollingLogFileAppender_DateFormat_log.txt" />
  <appendToFile value="true" />
  <rollingStyle value="Date" />
  <!--<datePattern value="yyyyMMdd-HHmm" />-->
  <datePattern value="yyyyMMdd" />
  <layout type="log4net.Layout.PatternLayout">
    <conversionPattern value="%date [%thread]
(%file:%line) %-5level %logger [%property{NDC}] - %message%newline" />
  </layout>
</appender>
```

这样一来，每天的日志都写入到一个文件中，当天的日志文件名为“RollingLogFileAppender_DateFormat_log.txt”，非当天的日志都会带上当天的日期，如“RollingLogFileAppender_DateFormat_log.txt20101117”表示2010年11月17日的日志，这样就可以很方便地区分每天的日志了，将来查找起来也相当方便。

在配置中启用和关闭日志

在config文件中可以很方便地关闭和启用日志，就是在<root>进行配置，如下就是一个例子：

```xhtml
<root>
  <!--文件形式记录日志-->
  <appender-ref ref="LogFileAppender" />
  <!--控制台控制显示日志-->
  <appender-ref ref="ConsoleAppender" />
  <!--Windows事件日志-->
  <!--<appender-ref ref="EventLogAppender" />-->
  <!--SQLite事件日志-->
  <appender-ref ref="AdoNetAppender_SQLite" />
  <!--RollingFileAppender事件日志-->
  <appender-ref ref="RollingFileAppender" />
  <!--RollingFileAppender事件日志，每天一个日志-->
  <appender-ref ref="RollingLogFileAppender_DateFormat" />
  <!-- 如果不启用相应的日志记录，可以通过这种方式注释掉
  <appender-ref ref="AdoNetAppender_Access" />
  -->
</root>
```

在上面的例子中可以看出，如果想增加日志输出目的地，增加<appender-ref>节点就是了，注意后面的ref是在config中配置的appender name，如果想要取消，删除或者注释掉这行就可以了。

Log4Net的使用

首先要添加config文件，在类库项目、命令行程序及WinForm中添加的是app.config，在WebForm中添加的是web.config。

下面是一个在WinForm项目中的使用Log4Net的例子：

```c-sharp
using System.Collections.Generic;
using System.Text;
using log4net;
using System.Reflection;

//注意下面的语句一定要加上，指定log4net使用.config文件来读取配置信息
//如果是WinForm（假定程序为MyDemo.exe，则需要一个MyDemo.exe.config文件）
//如果是WebForm，则从web.config中读取相关信息
[assembly: log4net.Config.XmlConfigurator(Watch = true)]
namespace Log4NetDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            Random random = new Random();
            for (int i = 0; i < 1; i++)
            {
                //创建日志记录组件实例
                //ILog log = log4net.LogManager.GetLogger(MethodBase.GetCurrentMethod().De

                ILog log=log4net.LogManager.GetLogger(typeof(Program));
                //记录错误日志
                //log.Error("error", new Exception("在这里发生了一个异
常,Error Number:"+random.Next()));
                //记录严重错误
                //log.Fatal("fatal", new Exception("在发生了一个致命错误，
Exception Id: "+random.Next()));
                //记录一般信息
                //log.Info("提示：系统正在运行");
                //记录调试信息
                //log.Debug("调试信息：debug");
                //记录警告信息
                //log.Warn("警告：warn");
            }
            Console.WriteLine("日志记录完毕。");
            Console.Read();

        }
    }
}
```

在WebForm中也可以使用Log4net，下面是一个在ASP.NET中使用Log4Net的例子：

```c-sharp
```

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using log4net;
using System.Reflection;

//如果是web项目，需要在global.asax的Application_Start方法中配置web.config的引用,log4net.Config.XmlConfigurator.Configure();
[assembly: log4net.Config.XmlConfigurator(Watch = true)]
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            Random random = new Random();
            for (int i = 0; i < 1; i++)
            {
                //创建日志记录组件实例
                ILog log = log4net.LogManager.GetLogger(MethodBase.GetCurrentMethod().Decl

                //ILog log = log4net.LogManager.GetLogger(typeof(Program));
                //记录错误日志
                //log.Error("error", new Exception("在这里发生了一个异常,Error Number:"+random.Next()));
                //记录严重错误
                //log.Fatal("fatal", new Exception("在发生了一个致命错误，Exception Id: "+random.Next()));
                //记录一般信息
                //log.Info("提示：系统正在运行");
                //记录调试信息
                //log.Debug("调试信息：debug");
                //记录警告信息
                log.Warn("警告：warn");
                Response.Write("日志记录完毕。");
            }
        }
    }
}
```

可以看出它们的代码基本没有区别。

下面是一个在WinForm下的config文件的完整配置，该配置文件所使用的代码就是在上面所使用到的代码，使用LogFileAppender、ConsoleAppender、EventLogAppender、AdoNetAppender_SQLite、RollingFileAppender、RollingLogFileAppender_DateFormat方式记录日志都在本地通过测试。完整的config文件代码如下：

[xhtml]

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="log4net" type="System.Configuration.IgnoreSectionHandler"/>
  </configSections>
  <appSettings>
    <!-- To enable internal log4net logging specify the
             following appSettings key -->
     <add key="log4net.Internal.Debug" value="true"/>
  </appSettings>
  <log4net>
    <!--定义输出到文件中-->
    <appender name="LogFileAppender" type="log4net.Appender.FileAppender">
      <!--每条日志末尾的文字说明-->
      <footer value="by 周公" />
      <!--定义文件存放位置-->
      <file value="LogFileAppender_log.txt" />
      <appendToFile value="true" />
      <datePattern value="yyyyMMdd-HH:mm:ss" />
      <layout type="log4net.Layout.PatternLayout">
        <!--每条日志末尾的文字说明-->
        <footer value="by 周公" />
        <!--输出格式-->
        <!--样例：2010-11-
17 15:50:23,443 [9] (D:/CSProjects/Log4NetDemo/Log4NetDemo/Program.cs:27) FATAL Log4NetDem
System.Exception: 在发生了一个致命错误，Exception Id：548828745-->
        <conversionPattern value="记录时间：%date 线程ID:[%thread] 日志级别：文件：所在
行%-5level 出错类：%logger property:[%property{NDC}] - 错误描述：%message%newline" />
      </layout>
    </appender>
    <!--定义输出到控制台命令行中-->
    <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date [%thread] (%file:%line) %-5level %logger [%property
      </layout>
    </appender>
    <!--使用Rolling方式记录日志
    每个日志文件最大100KB，生成的日志文件名会是log.txt.1,log.txt.2 ...log.txt.10
    如果记录的日志超过10个，会从log.txt.1开始覆盖
    -->
<appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
  <file value="RollingFileAppender_log.txt" />
  <appendToFile value="true" />
  <rollingStyle value="Size" />
  <maxSizeRollBackups value="10" />
  <maximumFileSize value="100KB" />
  <staticLogFileName value="true" />
  <layout type="log4net.Layout.PatternLayout">
    <conversionPattern value="%date [%thread] (%file:%line) %-5level %logger [%property{ND
  </layout>
```

```
</appender>
    <!--使用Rolling方式记录日志
    按照日来记录日志
    -->
<appender name="RollingLogFileAppender_DateFormat" type="log4net.Appender.RollingFileAppen
  <file value="RollingLogFileAppender_DateFormat_log.txt" />
  <appendToFile value="true" />
  <rollingStyle value="Date" />
  <!--<datePattern value="yyyyMMdd-HHmm" />-->
  <datePattern value="yyyyMMdd" />
  <layout type="log4net.Layout.PatternLayout">
    <conversionPattern value="%date [%thread]
(%file:%line) %-5level %logger [%property{NDC}] - %message%newline" />
  </layout>
</appender>

    <!--记录到SQLite这样的单机数据库中去
    创SQLite表的SQL语句:
    CREATE TABLE Log (
    LogId        INTEGER PRIMARY KEY,
    Date        DATETIME NOT NULL,
    Level        VARCHAR(50) NOT NULL,
    Logger        VARCHAR(255) NOT NULL,
    Source        VARCHAR(255) NOT NULL,
    Message        TEXT DEFAULT NULL
    );
    -->
<appender name="AdoNetAppender_SQLite" type="log4net.Appender.AdoNetAppender">
  <bufferSize value="100" />
  <connectionType value="System.Data.SQLite.SQLiteConnection, System.Data.SQLite, Version=
  <!--SQLite连接字符串-->
  <connectionString value="Data Source=c://log4net.db;Version=3;" />
  <commandText value="INSERT INTO Log (Date, Level, Logger,Source, Message) VALUES (@Date,
  <parameter>
    <parameterName value="@Date" />
    <dbType value="DateTime" />
    <layout type="log4net.Layout.RawTimeStampLayout" />
  </parameter>
  <parameter>
    <parameterName value="@Level" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%level" />
    </layout>
  </parameter>
  <parameter>
    <parameterName value="@Logger" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%logger" />
    </layout>
```

```
  </parameter>
  <parameter>
    <parameterName value="@Source" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%file:%line" />
    </layout>
  </parameter>
  <parameter>
    <parameterName value="@Message" />
    <dbType value="String" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%message" />
    </layout>
  </parameter>
</appender>
    <!--定义输出到SQL Server数据库中-->
    <!--
    在SQL Server中创建表的SQL语句
    CREATE TABLE [dbo].[Log] (
    [Id] [int] IDENTITY (1, 1) NOT NULL,
    [Date] [datetime] NOT NULL,
    [Thread] [varchar] (255) NOT NULL,
    [Level] [varchar] (50) NOT NULL,
    [Logger] [varchar] (255) NOT NULL,
    [Message] [varchar] (4000) NOT NULL,
    [Exception] [varchar] (2000) NULL
);
    -->
    <appender name="AdoNetAppender_SQLServer" type="log4net.Appender.AdoNetAppender">
      <bufferSize value="100" />
      <connectionType value="System.Data.SqlClient.SqlConnection, System.Data" />
      <connectionString value="data source=[database server];initial catalog=
[database name];integrated security=false;persist security info=True;User ID=
[user];Password=[password]" />
      <commandText value="INSERT INTO Log ([Date],[Thread],[Level],[Logger],[Message],
[Exception]) VALUES (@log_date, @thread, @log_level, @logger, @message, @exception)" />
      <parameter>
        <parameterName value="@log_date" />
        <dbType value="DateTime" />
        <layout type="log4net.Layout.RawTimeStampLayout" />
      </parameter>
      <parameter>
        <parameterName value="@thread" />
        <dbType value="String" />
        <size value="255" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%thread" />
        </layout>
      </parameter>
      <parameter>
```

```
147.         <parameterName value="@log_level" />
148.         <dbType value="String" />
149.         <size value="50" />
150.         <layout type="log4net.Layout.PatternLayout">
151.           <conversionPattern value="%level" />
152.         </layout>
153.       </parameter>
154.       <parameter>
155.         <parameterName value="@logger" />
156.         <dbType value="String" />
157.         <size value="255" />
158.         <layout type="log4net.Layout.PatternLayout">
159.           <conversionPattern value="%logger" />
160.         </layout>
161.       </parameter>
162.       <parameter>
163.         <parameterName value="@message" />
164.         <dbType value="String" />
165.         <size value="4000" />
166.         <layout type="log4net.Layout.PatternLayout">
167.           <conversionPattern value="%message" />
168.         </layout>
169.       </parameter>
170.       <parameter>
171.         <parameterName value="@exception" />
172.         <dbType value="String" />
173.         <size value="2000" />
174.         <layout type="log4net.Layout.ExceptionLayout" />
175.       </parameter>
176.     </appender>
177.     <!--定义输出到Oracle9i中-->
178.     <!--
179.     在Oracle9i中创建表的SQL语句
180.     create table log (
181.    Datetime timestamp(3),
182.    Thread varchar2(255),
183.    Log_Level varchar2(255),
184.    Logger varchar2(255),
185.    Message varchar2(4000)
186.    );
187.      -->
188.     <appender name="AdoNetAppender_Oracle" type="log4net.Appender.AdoNetAppender">
189.       <connectionType value="System.Data.OracleClient.OracleConnection, System.Data.Oracle
190.       <connectionString value="data source=[mydatabase];User ID=[user];Password=
[password]" />
191.       <commandText value="INSERT INTO Log (Datetime,Thread,Log_Level,Logger,Message) VALUE
192.       <bufferSize value="128" />
193.       <parameter>
194.         <parameterName value=":log_date" />
195.         <dbType value="DateTime" />
196.         <layout type="log4net.Layout.RawTimeStampLayout" />
```

```
      </parameter>
      <parameter>
        <parameterName value=":thread" />
        <dbType value="String" />
        <size value="255" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%thread" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value=":log_level" />
        <dbType value="String" />
        <size value="50" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%level" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value=":logger" />
        <dbType value="String" />
        <size value="255" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%logger" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value=":message" />
        <dbType value="String" />
        <size value="4000" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%message" />
        </layout>
      </parameter>
    </appender>
    <!--定义输出到IBM DB2中-->
    <!--
    在DB2中创建表的SQL语句
    CREATE TABLE "myschema.LOG" (
    "ID" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (
        START WITH +1
        INCREMENT BY +1
        MINVALUE +1
        MAXVALUE +2147483647
        NO CYCLE
        NO CACHE
        NO ORDER
    ),
    "DATE" TIMESTAMP NOT NULL,
    "THREAD" VARCHAR(255) NOT NULL,
    "LEVEL" VARCHAR(500) NOT NULL,
    "LOGGER" VARCHAR(255) NOT NULL,
```

```
    "MESSAGE" VARCHAR(4000) NOT NULL,
    "EXCEPTION" VARCHAR(2000)
)
IN "LRGTABLES";
    -->
    <appender name="AdoNetAppender_DB2" type="log4net.Appender.AdoNetAppender">
      <bufferSize value="100" />
      <connectionType value="IBM.Data.DB2.DB2Connection,IBM.Data.DB2" />
      <connectionString value="server=192.168.0.0;database=dbuser;user Id=username;passwor
      <commandText value="INSERT INTO myschema.Log (Date,Thread,Level,Logger,Message,Excep
      <parameter>
        <parameterName value="@log_date" />
        <dbType value="DateTime" />
        <layout type="log4net.Layout.RawTimeStampLayout" />
      </parameter>
      <parameter>
        <parameterName value="@thread" />
        <dbType value="String" />
        <size value="255" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%thread" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@log_level" />
        <dbType value="String" />
        <size value="500" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%level" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@logger" />
        <dbType value="String" />
        <size value="255" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%logger" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@message" />
        <dbType value="String" />
        <size value="4000" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%message" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@exception" />
        <dbType value="String" />
        <size value="2000" />
```

```
        <layout type="log4net.Layout.ExceptionLayout" />
      </parameter>
    </appender>
    <!--定义输出到windows事件中-->
    <appender name="EventLogAppender" type="log4net.Appender.EventLogAppender">
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date [%thread] %-5level %logger [%property{NDC}] - %mes
      </layout>
    </appender>
    <!--定义输出到数据库中，这里举例输出到Access数据库中，数据库为C盘的log4net.mdb
    创建Access表的SQL语句：
    -->
    <appender name="AdoNetAppender_Access" type="log4net.Appender.AdoNetAppender">
      <connectionString value="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:/log4net.mdb
      <commandText value="INSERT INTO LogDetails ([LogDate],[Thread],[Level],[Logger],
[Message]) VALUES (@logDate, @thread, @logLevel, @logger,@message)" />
      <!--定义各个参数-->
      <parameter>
        <parameterName value="@logDate" />
        <dbType value="String" />
        <size value="240" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%date" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@thread" />
        <dbType value="String" />
        <size value="240" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%thread" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@logLevel" />
        <dbType value="String" />
        <size value="240" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%level" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@logger" />
        <dbType value="String" />
        <size value="240" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%logger" />
        </layout>
      </parameter>
      <parameter>
        <parameterName value="@message" />
```

```
        <dbType value="String" />
        <size value="240" />
        <layout type="log4net.Layout.PatternLayout">
          <conversionPattern value="%message" />
        </layout>
      </parameter>
    </appender>
    <!--定义日志的输出媒介，下面定义日志以四种方式输出。也可以下面的按照一种类型或其他类型输出。-
->
<root>
  <!--文件形式记录日志-->
  <appender-ref ref="LogFileAppender" />
  <!--控制台控制显示日志-->
  <appender-ref ref="ConsoleAppender" />
  <!--Windows事件日志-->
  <!--<appender-ref ref="EventLogAppender" />-->
  <!--SQLite日志-->
  <!--<appender-ref ref="AdoNetAppender_SQLite" />-->
  <!--RollingFileAppender事件日志-->
  <appender-ref ref="RollingFileAppender" />
  <!--RollingFileAppender事件日志，每天一个日志-->
  <appender-ref ref="RollingLogFileAppender_DateFormat" />
  <!-- 如果不启用相应的日志记录，可以通过这种方式注释掉
  <appender-ref ref="AdoNetAppender_Access" />
  -->
</root>

  </log4net>
  <!--<system.diagnostics>
    <trace autoflush="true">
      <listeners>
        <add
            name="textWriterTraceListener"
            type="System.Diagnostics.TextWriterTraceListener"
            initializeData="D:/CSProjects/Log4NetDemo/Log4NetDemo/bin/log4net.txt" />
      </listeners>
    </trace>
  </system.diagnostics>-->

</configuration>
```

总结：

本篇主要是补充在上一篇关于Log4Net中未尽之处，并集中详尽回答了一些朋友在该篇博文下的提问，如有未尽之处，请在本篇下留言。如有初学者碰巧路过不知config文件为何物，请看《asp.net夜话之十一：web.config详解》，网址是：http://blog.csdn.net/zhoufoxcn/archive/2008/11/10/3265141.aspx。