Log4cxx 的使用说明

2007-08-08 17:45

摘要

Log4cxx 是开放源代码项目 Apache Logging Service 的子项目之一，用于为 C++程序提供日志功能，以便开发者对目标程序进行调试和审计。本文对 log4cxx 的使用及配置进行介绍，并给出一个可以快速开始的实例。最后，针对日志服务给出一些实践方面的建议。

1. 介绍

Log4cxx 是开放源代码项目 Apache Logging Service 的子项目之一，是 Java 社区著名的 log4j 的 c++移植版，用于为 C++程序提供日志功能，以便开发者对目标程序进行调试和审计。

有关 log4cxx 的更多信息可以从 Apache Loggin Service 的网站 http://logging.apache.org 获得。

当前的稳定版本为 0.9.7，本文内容及示例代码都是基于此版本。此外，示例代码的编译环境为 Windows 环境中的 Microsoft Visual C++ .Net 2003。

本文的示例代码可以在此下载，其中也包含了预编译好的 log4cxx 的库文件。

2. 集成 log4cxx 到 IDE

要使用 log4cxx，首先需要将其集成到你的项目开发环境中。以下针对 Windows 环境中的 Microsoft Visual C++ .Net 2003 进行说明，其他环境的配置信息请参考官方文档。

要让 log4cxx 为你工作，通常情况下需要如下几个步骤：

- 获取软件包：得到 log4xx 的源代码；
- 编译：构建库文件；
- 项目环境设置：加入 log4cxx 支持。

2.1 获取软件包

请从官方网站获得合适的版本。也可以从下面这个链接中直接获取（直接的链接地址可能不会永远有效）：

http://mirror.vmmatrix.net/apache/logging/log4cxx/log4cxx-0.9.7.tar.gz

下载完成后解压缩到合适的目录中，因为我们在下一步中需要使用软件包，包括编译和复制必要的文件。

2.2 编译

原始发行包中不含编译后的代码，这个工作需要我们自己来做。打开你的 IDE，并加载以下目录中的工程：

- Msvc\static：该工程产生 log4cxx 的静态链接库（lib4cxx.lib 和 lib4cxxs.lib）；
- Msvc\dll：该工程产生 log4cxx 的动态链接库（lib4cxx.dll）。

通常情况下，工程都可以顺利编译通过。查看输出目录，把这些生成的库文件找出来，以便在下一步骤中使用。

2.3 项目环境设置

请先在 IDE 中打开一个需要加入日志功能的工程，或者出于实验目的，新建一个工程，以便对其进行设置。

首先需要设置 log4cxx 的 include 文件。这些文件位于 log4cxx 软件包的 include\log4cxx 目录内。请查看你的 VC++ IDE 中“工具->选项->项目->VC++目录->包含文件”所列出的内容，以便确定你以何种方式加入这些 include 文件：

- 将 include\log4cxx 直接拷贝到已定义的包含文件目录中。如果将 log4cxx 看作是一项系统服务的话，这样做是胡合乎情理的，因为你可以采用标准库的方式使用它，例如：#include <log4cxx/logger.h>
- 增加一个包含路径，以指向位于 IDE 外部的某一文件目录。可以简单的指向在 2.1 中解压缩后形成的 log4cxx 软件包目录。

下一步需要对 2.2 节产生的 log4cxx 库进行设置。这取决于你使用该库的方式：静态链接或者动态链接。

l 静态链接情况下需要做如下工作：为预编译器定义 LOG4CXX_STATIC 宏，设置位置为“项目->属性->配置属性->C/C++->预处理器->预处理器定义”；为链接器指定依赖的库 lib4cxxs.lib 和 Ws2_32.lib，设置位置为“项目->属性->配置属性->链接器->输入->附加依赖项”。

l 动态链接情况下只需要为链接器指定依赖的库 lib4cxxs.lib 即可，设置方式同上。

3. 示例代码

本节展示了一个最简单的 log4cxx 示例，以便你可以快速的了解它。

该示例在功能上创建了一个日志服务，该日志可通过配置文件进行必要控制，并可以同时向文件和控制台输出信息。

在实现上，我们采用了一个简单的控制台程序，并使用动态链接库的方式使用 log4cxx。

要实现这个目标，请按如下步骤进行：

1）创建一个名为 logdemo 的空白 win32 控制台工程，并按照 2.3 节所述内容对其进行设置。注意，这里我们使用动态连接口的方式。

2）在 logdemo.cpp 中加入实现日志功能的代码。完成后的代码清单如下：

```
#include "stdafx.h"
#include <log4cxx/logger.h>
#include <log4cxx/propertyconfigurator.h>

using namespace log4cxx;

int _tmain(int argc, _TCHAR* argv[])
{
	//加载 log4cxx 的配置文件，这里使用了属性文件
	PropertyConfigurator::configure("log4cxx.properties");

	//获得一个 Logger，这里使用了 RootLogger
	LoggerPtr rootLogger = Logger::getRootLogger();

	//发出 INFO 级别的输出请求
	LOG4CXX_INFO(rootLogger, _T("它的确工作了"));
//rootLogger->info(_T("它的确工作了")); //与上面那句话功能相当

return 0;
}
```

以 Debug 方式编译工程，调试程序直到成功为止。

3）新建一个文本文件，命名为 log4cxx.properties，并键入如下内容：

```
# 设置 root logger 为 DEBUG 级别，使用了 ca 和 fa 两个 Appender
log4j.rootLogger=DEBUG, ca, fa
```

4）复制 log4cxx.dll 到输出目录。在动态链接方式下，应用程序需要能够找到这个库文件。

5）运行生成的 logdemo.exe 文件，查看一下运行结果，看看我们工作有没有取得成效。如果一切顺利，无论是在控制台还是在输出文件中，都应该能看到类似下面那样的输出内容：
2006-06-02 16:09:50,609 [2528] INFO root - 它的确工作了
4. 体系结构
4.1 核心类
Log4cxx 有三个关键组件，它们是 loggers, appenders 和 layouts。
Logger 是 log4cxx 的核心类，只要执行日志操作；looger 有层次结构，最顶层为 RootLogger；logger 是有级别的。每个 logger 可以附加多个 Appender。Appender 代表了日志输出的目标，如输出到文件、控制台等等。对于每一种 appender，都可以通过 layout 进行格式设置。
这三类组件用示意图表示如下（不代表类关系）：
（TODO：在此对三种组件分别进行说明）
4.2 配置类
此外在使用中还会用到的类有 BasicConfigurator、PropertyConfigurator 和 DOMConfigurator 等，用于对 log4cxx 进行配置。其中：
BasicConfigurator 提供了一种简单配置，包括使用 ConsoleAppder 作为 root appender 和 PatternLayout 作为缺省布局。
PropertyConfigurator 使用 properties 文件作为配置方式。
DOMConfigurator 则使用 properties 文件作为配置方式。
（TODO：在此对配置内容进行说明）
5. 实践指导
在项目中是否使用日志，以及如何使用日志，对开发者来说都是一个需要做出的技术选择，这通常会牵扯到系统的性能，使用日志的目的等问题。我们使用日志的方式，有些是这个行业积累了多年的经验，有些则纯粹关乎个人的喜好。
1）何时使用日志
通常情况下，日志的作用在于调试和审计，如果你的项目对此有特殊需求，即可考虑使用日志。
对于调试，通常用于 IDE 调试器无法达到的地方。一些常见的场景包括：
分布式组件的调试。在服务器端的组件，需要通过客户端的调用来验证其工作是否正确，此时利用日志的输出作为辅助工具对错误进行诊断。
链接库调试。在无法跟踪进外部库中的情况下，这种方法非常有效。
生产环境下的调试。生产环境通常是指产品在客户处处于正式运行的状态，在出现问题时，开发者常常不在现场，借助日志的输出进行错误判断就是一个非常有效的手段。
对于审计应用，则需要视特定的情况而定，程序级的记录能力，无疑可以作为业务级审计手段的有效补充。
无论是在哪种场景下，log4cxx 都是可以胜任工作的，这取决于它的灵活的配置能力及多种类型的输出方式。
2）性能问题
关闭日志，通过配置文件设置日志的关闭和打开
使用宏代替 logger 的输出命令
选择性输出日志。建立 logger 的层次结构，根据级别选择性输出
输出目标。尽可能减少输出目标
选择合适的输出格式。使用 SimpleLayout 将达到与 std::cout 相当的速度。
3）其它
使用类的全限定名对 logger 命名

6. 结论

Log4cxx 具有的一些显著特性使得 C++者可以将其放入自己的工具箱中，这些特性包括灵活的配置能力，多种输出手段，丰富的格式控制，出色的性能。如果在你的开发中需要借助于日志进行调试和审计，你也许需要 log4cxx。最后，重要的一点是，如你所见，log4cxx 的使用是如此的简单。