DCMTK - Feature #14

Vereinheitlichung der Log-Ausgabe

2008-11-14 00:00 - Jörg Riesmeier

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Target version:			
Module:	alle	**Compiler:**	
Operating System:			

Description

aus einer E-Mail-Kommunikation zwischen JR und ME Ende 2006

> ich nehme den heutigen Anruf von Herrn Hailer aus Straubing mal zum Anlaß, das Thema Log-Dateien im DCMTK aufzugreifen.

Das Thema ist ja nicht wirklich neu - darüber haben wir vor Jahren schon diskutiert. Die bestehende Klasse OFLogFile ist ja auch ein erster Versuch in diese Richtung, wird allerdings meines Wissens nach nirgendwo im Toolkit wirklich verwendet.

Ich denke auch, dass es sehr wünschenswert wäre, wenn man quer über alle Tools eine systematische Logging-Möglichkeit hätte, mit einem einheitlichen Logformat für die Fehleranalyse. Allerdings wird man verschiedene Arten von Logging unterscheiden wollen und müssen:

- Logging der Netzwerkkommunikation: Hier möchte man im Prinzip ein detailliertes Log aller eingehenden und ausgehenden DIMSE-Nachrichten haben, so wie es etwa der DCMPRINT-Printserver im Modus --dump produziert - das hat sich bei der Fehlersuche schon sehr häufig exzellent bewährt.

- Logging von internen Abläufen in den Tools. Es gibt ja in vielen der Teilbibliotheken irgendwelche "Debug"-Mechanismen, die mehr oder weniger hilfreiche Ausgaben i.d. Regel auf der Konsole ausspucken. Die Ausgaben von dcmdata z.B. finde ich durchgängig eher hinderlich als hilfreich :-)
  Aber grundsätzlich wäre es bei der Fehlersuche schon gut, zu wissen, was im Inneren eines Tools abläuft. Beispiele dafür könnten sein: * Call-Traces (welche Methoden werden wann aufgerufen) * Ein Log aller Fehlercodes (OFCondition-Codes), die in der Anwendung generiert werden, inklusive der Stellen, an denen sie generiert werden (der Printserver hat einen ziemlich primitiven Mechanismus, der so etwas ähnliches realisiert. In tcpsrv.log werden sogenannte "Assertions" geschrieben, immer dann, wenn irgendeine im Sourcecode definierte Bedingung nicht erfüllt ist und etwas "ungewöhnliches" passiert. Die Logausgabe enthält dann Dateiname und Zeilennummer im Quelltext, was es recht leicht macht, die entsprechende Stelle aufzuspüren (Beispiel: "In tcpsrv.cc Zeile 1234 wurde der Fehlercode "cannot process" zurückgeliefert, weil BitsAllocated > 16 war" * Etwas Ähnliches gibt es ja im Parser von dcmsr, wo der Parser auf Wunsch sehr detaillierte Warnungen und Fehlermeldungen "ausspucken" kann.

Fazit bis hierhin: Logging muss immer auf einen bestimmten Zweck bezogen sein; es hat wenig Sinn, von Logging als abstraktem "Feature" zu sprechen. Was wollen wir loggen, und für welchen Zweck?

> Ein erster Schritt wäre die Einführung eines einheitlichen Logging-Mechanismus. Folgende Stichworte fallen mir dazu auf die Schnelle ein:
> - flexibles Loggen in Datei, auf Console und/oder in Datenbank (?)

Ein weiterer Kandidat wäre Loggen via Syslog, gerade für Prozesse, die lange als Serverprozesse laufen. Es ist natürlich recht einfach, eine abstrakte "Logging Facility" zu definieren, die ihre Ausgabe "irgendwo" hinschreibt, und davon Klassen abzuleiten, die dann die verschiedenen Ausgabeziele realisieren. Im übrigen halte ich das eher für ein unwichtiges

Detail.

> - bei Dateien: Ablage in einem bestimmten Verzeichnis, Dateiname enthält z.B. Time Stamp, Calling AE Title, Kürzel des Programms

Hier gehst Du implizit vom Loggen der Netzwerkkommunikation aus.
Ich weiss nicht, ob es den Aufwand wirklich wert ist, so komplexe Dateinamen zu generieren, die dann spezifisch für Netzwerke sind und sich erst dann generieren lassen, wenn ein A-ASSOCIATE-Paket erfolgreich empfangen oder versendet wurde - was macht man denn mit fehlgeschlagenen Verbindungsversuchen?

> - Einheitliche Struktur der Einträge mit Time Stamp, AE Title, Art des Eintrags ... (zumindest für den "Kopf" eines Log-Eintrags)

> - evtl. Unterteilung in Kurzdarstellung (eine Zeile pro Eintrag) und Detailausgabe (wie bisher), eine solche Unterteilung in zwei Dateien habe ich bei einigen Firmen auf dem IHE Connect-a-thon so gesehen

Halte ich nicht für sinnvoll, es sei denn, man benötigt die Kurzdarstellung, um die "Kennnummer" einer Verbindung zu finden, die man dann im Logfile nachschlägt (bei Fuji gibt es so etwas).

> - ggf. Filtermöglichkeit nach Detaillierungsgrad der Meldung (Info, Error, Warning, Debug)

Frage: Filtern //vor// der Ausgabe ins Log oder Filtern in einem Visualisierungtool? Wollen wir ein solches bauen?
Ich denke hier an nichts sonderlich komplexes, vielleicht aber an ein Skript (Tcl/Tk o.ä.), das ein Log mit gewissem Syntaxhighlighting darstellt und verschiedene Filter anwenden kann.

Möchte man die Ausgabe vielleicht in XML machen, um besser Filtern/Konvertieren zu können?

> Sicherlich muß man nicht alle o.g. und weitere Features, die einem noch so einfallen, sofort umsetzen, aber einige sind sicherlich mit vertretbarem Aufwand umsetzbar und wie gesagt m.E. sehr hilfreich. Vielleicht kann man ja auch eine existierende Bibliothek für diese Zwecke nutzen (z.B. log4cpp).

Ich habe mir log4cpp mal angesehen, hat mir in Sachen Features und Ausgabeformat gar nicht gefallen. Aber meine Meinung muss ja nicht repräsentativ sein.

Das Entscheidende ist meiner Meinung nach, * Anwendungszenarien für das Logging zu finden und dafür passende Log-Inhalte zu definieren (Netzwerkmitschnitt ist ein offensichtliches, aber es gibt sicher weitere) * Ein Logging dann auf der niedrigsten möglichen Ebene zu implementieren - ein Logging der Netzwerkkommunikation gehört etwa in dcmnet realisiert, so dass jede Anwendung, die DICOM-Netzwerkverbindungen nutzt, automatisch auch ein entsprechendes Log aktivieren kann.

=== Nachtrag JR ===

Einige der zahlreichen verfügbaren Logger-Klassen für C++:

- log4cpp: http://log4cpp.sourceforge.net/
- log4cxx: http://logging.apache.org/log4cxx/
- log4cplus: http://log4cplus.sourceforge.net/

History

#1 - 2014-09-22 16:53 - Michael Onken

- Description updated

- Private changed from No to Yes

#2 - 2015-01-15 13:40 - Michael Onken

- Private changed from Yes to No