

qflog - Das Handbuch

Version 0.95.1

Gregor Schmid
Quality First Software

Copyright © 2000 *Quality First Software*, Gregor Schmid

10. Juli 2000

Copyright

The contents of this manual are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this manual except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>.

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is qfs.de code.

The Initial Developer of the Original Code is Gregor Schmid. Portions created by Gregor Schmid are Copyright (C) 1999 Quality First Software, Gregor Schmid. All Rights Reserved.

Contributor(s):

Vorwort

Im Rahmen der Entwicklung unserer Produkte wurde bei *Quality First Software*¹ von Anfang an auf Logging als Werkzeug gesetzt, sowohl bei der Entwicklung selbst, als auch bei der späteren Überwachung der Programme. Das Problem dabei ist, dass der Wert von Logmeldungen indirekt proportional zu ihrer Anzahl ist, da wichtige Informationen in der Masse untergehen. Als Antwort darauf haben wir den Logserver *qflog* entwickelt, mit dessen mehrstufigen Filtern, der Sortierung und der Suchfunktion wir in der Lage sind, den Überblick auch über große Mengen von Logausgaben zu behalten. Parallel dazu wurde der Mechanismus zur Erzeugung der Meldungen immer weiter verbessert, so dass inzwischen eine fein abgestimmte Kontrolle über das Logging eines Programms zur Laufzeit möglich ist.

Als Dank für die Leistungen, die die Gemeinschaft der Entwickler freier Software vollbracht hat und ohne die es *Quality First Software* nie gegeben hätte, haben wir uns entschieden, sowohl *qflog*, als auch unsere Java Bibliothek *qflib*² unter eine Open Source Lizenz zu stellen und für die Allgemeinheit frei verfügbar zu machen.

Wir hoffen, dass Ihnen *qflog* ebenso gute Dienste leistet wie uns und dass dieses Handbuch Ihnen alle Informationen gibt, die Sie für seinen Einsatz benötigen. Für Probleme, Fragen und Anregungen haben wir unter qflog@qfs.de stets ein offenes Ohr.

Gregor Schmid³, München den 28.4.2000

¹<http://www.qfs.de>

²<http://www.qfs.de/de/projects/qflib/index.html>

³email: gs@qfs.de

Inhaltsverzeichnis

1	Einführung	6
1.1	Funktionalität	6
1.2	Anwendung	7
2	Installation	8
2.1	Voraussetzungen	8
2.2	Installation	8
2.3	RMI-Registry	9
3	Aufruf	10
3.1	Aufruf Syntax	10
3.2	Optionen	10
4	Bedienung des Hauptfensters	13
4.1	Laden und Speichern von Logdateien	13
4.2	Öffnen des Logfensters	14
4.3	Löschen von Clients	14
4.4	Einstellungen	14
4.4.1	Allgemeine Einstellungen	14
5	Bedienung der Logfenster	16
5.1	Allgemeines	16
5.1.1	Aufbau des Logfensters	16
5.1.2	Speichern von Logdateien	18
5.1.3	Laden und Speichern der Konfiguration	18

5.2	Die Tabelle	18
5.2.1	Kopieren von Logmeldungen	19
5.2.2	Löschen von Einträgen	20
5.2.3	Zusatzfilter in der Tabelle	20
5.2.4	Markierungen setzen und anspringen	21
5.2.5	Suchen	21
5.2.6	Einstellungen	22
5.3	Die Detailansicht	22
5.4	Der Filterbaum	23
5.4.1	Die Struktur des Baums	23
5.4.2	Festlegen der Filterlevel	25
5.4.3	Anzeigen der gesetzten Level	25
5.4.4	Zusätzliche Filterfunktion	25
5.5	Die Loglevel im Client	26
5.5.1	Die Struktur des Baums	26
5.5.2	Festlegen der Level	28
5.5.3	Anzeigen der gesetzten Level	28
5.5.4	Einstellungen	28
6	Ein Anwendungsbeispiel	29
6.1	Aufruf	29
6.2	Optionen	29
6.3	Beispiele	31
A	qflog und Applets	32
A.1	Internet Explorer	32
A.2	Netscape	33
A.3	Plugin	33
A.4	Beispiel Applet	33

Abbildungsverzeichnis

4.1	Das Hauptfenster	14
5.1	Ein Logfenster	17
5.2	Die Tabelle mit den Logmeldungen	19
5.3	Die Detailansicht	22
5.4	Der Filterbaum	23
5.5	Der Levelbaum	27

Kapitel 1

Einführung

Der Logserver *qflog* ist ein wichtiges Hilfsmittel, um den Überblick und die Kontrolle über die Logausgaben eines Programms zu erhalten. In diesem Handbuch ist ausführlich beschrieben, welche Funktionen er bietet und wie man ihn installiert, aufruft und bedient.

1.1 Funktionalität

qflog ist speziell auf die Zusammenarbeit mit dem Package `de.qfs.lib.log` der *qf-lib* zugeschnitten, kann aber auch Logdateien von anderen Programmen (auch nicht in Java geschriebenen) anzeigen, sofern sie das passende Format haben oder ein Importfilter existiert. Im Moment wird nur das *qflib* Format unterstützt, weitere Formate sind relativ einfach zu implementieren.

Das Herzstück von *qflog* ist das Logfenster, das die beiden wesentlichen Funktionen von *qflog* beinhaltet:

- Die Anzeige der Logmeldungen mit Sortierung, mehreren Filtern, Suchfunktionen und Markierungen.
- Die Steuerung der Parameter eines Programms, die für die Erzeugung von Logmeldungen ausschlaggebend sind.

Die Bedienung des Logfensters ist in Kapitel 4.4.1 beschrieben.

1.2 Anwendung

Es gibt drei grundsätzliche Wege auf denen die Logausgaben eines Programms in ein Logfenster gelangen können:

- Das Programm speichert seine Ausgaben in einer Logdatei, die mit *qflog* geöffnet wird.
- Zwischen *qflog* und dem Programm wird eine Client/Server-Verbindung aufgebaut.
- Das Logfenster wird direkt in das Programm integriert.

Die Steuerung der Erzeugung von Meldungen ist natürlich nur in den letzten beiden Fällen möglich. Beim betrachten einer Logdatei ist das Programm, welche die Meldungen erzeugt hat, möglicherweise bereits beendet.

Im Moment ist die Client/Server-Kommunikation nur über das *Java* eigene RMI Protoll möglich. Bei Bedarf könnten auch CORBA oder direkte Socket-Kommunikation implementiert werden. Damit wäre *qflog* auch zur Kommunikation mit nicht-*Java* Clients geeignet.

Kapitel 2

Installation

2.1 Voraussetzungen

qflog kann sowohl mit einem *JDK* ab Version 1.1.6 und *Swing* 1.1, als auch mit jedem *JDK* ab Version 1.2 verwendet werden.

Da das Programm ausgiebigen Gebrauch von den mit 1.2 eingeführten Collection Klassen macht, benötigen Sie für die *JDK* 1.1 Variante zusätzlich das Collections Paket für *JDK* 1.1, welches SUN hier¹ zur Verfügung stellt.

qflog baut auf unserer frei verfügbaren Java Bibliothek *qflib*² auf. Auf unserer Download³ Seite finden sie stets die aktuelle Version von *qflog* und der *qflib*.

2.2 Installation

Nachdem Sie sich die aktuelle Version von *qflog* und der *qflib* besorgt haben, entpacken Sie beide Archive an einem beliebigen Ort. Bitte lesen Sie die Dokumentation der *qflib* für Hinweise zu ihrer Installation. Normalerweise reichen folgende Aktionen aus:

JDK 1.1

Stellen Sie sicher, dass sich die Archive `qflog_11.jar` und `qflib_11.jar` im CLASSPATH befinden, ebenso wie *Swing* (`swingall.jar`) und die Collection Klassen (`collections-1.1.jar`).

JDK 1.2

Nehmen Sie entweder die Archive `qflog_12.jar` und `qflib_12.jar` in den

¹http://java.sun.com/beans/infobus/#DOWNLOAD_COLLECTIONS

²<http://www.qfs.de/de/projects/qflib/index.html>

³<http://www.qfs.de/de/download.html>

CLASSPATH auf, oder stellen Sie sie direkt in das `jre/lib/ext` Verzeichnis innerhalb Ihrer *JDK/JRE* Installation.

In dem `bin` Verzeichnis der *qflog* Distribution finden Sie die Startskripte `qflog` für Unix und `qflog.bat` für Windows. Der Einfachheit halber sollten Sie dafür sorgen, dass sich das richtige Skript für Ihr System in Ihrem `PATH` befindet.

Um seine Einstellungen speichern zu können, benötigt *qflog* Schreibzugriff auf ein Verzeichnis. Sofern Sie beim Aufruf nichts anderes angeben (siehe Abschnitt 3.1), schreibt *qflog* seine Daten in das Verzeichnis `.qflog` in Ihrem Homeverzeichnis (Unix) bzw. unter `Eigene Dateien` (Windows).

2.3 RMI-Registry

Wenn Sie die Client/Server Funktionalität von *qflog* einsetzen, ist es ratsam, eine RMI-Registry im Hintergrund laufen zu haben. Diese kann permanent mit dem Programm `rmiregistry` verfügbar gemacht werden, das jedem *JDK* oder *JRE* beigelegt ist.

Sowohl *qflog* als auch seine Clients können bei Bedarf eine solche Registry selbst erzeugen. Dies hat jedoch den Nachteil, dass andere Programme, die sich in dieser Registry registrieren, nach Beendigung von *qflog*, bzw. des Clients, nicht mehr erreichbar sind, da ihre Registry nicht mehr verfügbar ist.

Um die RMI-Kommunikation so einfach wie möglich zu halten, sind alle beteiligten Klassen (die sogenannten *Stubs* und *Skeletons*) bereits in der *qflib* enthalten, so dass kein Webserver für den Betrieb nötig ist. Als Konsequenz davon muß allerdings die *qflib* beim Aufruf von `rmiregistry` im `CLASSPATH` bzw. im `jre/lib/ext` Verzeichnis ihres *JDK* enthalten sein.

Kapitel 3

Aufruf

Beim Start führt *qflog* die folgenden Aktionen durch, die sich über Optionen , die auf der Kommandozeile übergeben werden, steuern oder unterdrücken lassen:

- Wenn auf dem Rechner keine RMI-Registry läuft, erzeugt *qflog* selbst eine Registry. Näheres dazu siehe Abschnitt 2.2 und Abschnitt 3.1.
- Anschließend registriert sich *qflog* selbst in der RMI-Registry als Logserver.
- Dann wird die Registry auf diesem oder anderen Rechnern nach bereits registrierten Clients abgesucht, die auf einen Kontakt mit dem Logserver warten.
- Zuletzt werden die gewünschten Logdateien eingelesen.

3.1 Aufruf Syntax

Die Syntax zum Aufruf von *qflog* lautet wie folgt:

```
java [java-Optionen...] de.qfs.apps.logserver.Start  
[qflog-Optionen...] [Datei...]
```

oder bei Verwendung des *qflog* bzw. *qflog.bat* Skripts kurz

```
qflog [qflog-Optionen...] [Datei...]
```

Die angegebenen Dateien werden geladen und im Hauptfenster angezeigt.

3.2 Optionen

Die folgenden Optionen werden von *qflog* verstanden:

-configfile <Datei>

Gibt die Datei mit den Konfigurationsinformationen an. Darin speichert *qflog* sowohl die Positionen und Größen der Fenster, als auch die Filterinformationen für die verschiedenen Clients. Standardwert ist `/.qflog/config` für Unix bzw. `Eigene Dateien\.qflog\config` für Windows. Das Verzeichnis, in dem sich die Konfigurationsdatei befindet, wird auch als Ausgangspunkt für das Speichern und Laden der Konfiguration eines Logfensters verwendet (siehe Abschnitt 5.1.2).

-nocreateregistry

Diese Option verhindert, dass *qflog* selbst eine RMI-Registry einrichtet, falls noch keine vorhanden ist. Wird diese Option angegeben und keine RMI-Registry gefunden, kann *qflog* nicht als Logserver fungieren.

-noquery

Unterdrückt die Suche nach wartenden Clients, egal ob als Standardverhalten oder über die `-query` Option.

-noserver

Startet *qflog* im reinen Anzeigemodus, ohne dass eine Registrierung in der RMI-Registry stattfindet. Die Optionen `-port` und `-servername` werden dann ignoriert. Die Option `-query` ist hiervon nicht betroffen.

-optionfile <Datei>

Mit dieser Option kann eine Datei oder URL angegeben werden, von der weitere Optionen gelesen werden. Die Datei muß im üblichen Format für Propertydateien vorliegen, d.h. Zeilen der Form `name = value` enthalten. Optionen auf der Kommandozeile haben grundsätzlich Vorrang vor Optionen, die über `-optionfile` eingelesen werden.

-port <Portnummer>

Gibt die Portnummer für die RMI-Registry an, falls *qflog* selbst eine Registry einrichtet. Standardwert ist 1099, der "Well known port" der RMI-Registry.

-query <[Hostname][:Portnummer]>

Ohne Angabe von `-query` oder `-noquery` fragt *qflog* die lokale Registry unter dem standard Registryport 1099 nach bereits gestarteten Clients für den Logserver ab. Die `-query` Option kann mehrfach mit optionalem Parameter angegeben werden. Kein Parameter entspricht der Abfrage der lokalen Registry, andernfalls muß der Parameter die Form `Hostname` oder `Hostname:Portnummer` haben.

-servername <Name>

Der Name unter dem sich der LogServer von *qflog* in der RMI-Registry einträgt. Standardwert ist `qflog`.

-serverhost <Hostname>

Unter dieser Option kann der Name des Rechners angegeben werden, auf dem

qflog gestartet wird. Dies ist normalerweise nicht nötig und sollte nur verwendet werden, wenn es zu Problemen beim Verbinden mit Clients auf anderen Rechnern kommt.

-version

Wird die `-version` Option angegeben, gibt *qflog* nach dem Start lediglich seine Versionsnummer aus und beendet sich dann.

Kapitel 4

Bedienung des Hauptfensters

Nach dem Start von *qflog* erscheint zunächst das Hauptfenster (Abbildung 4.1), welches eine Liste der geladenen Logdateien, sowie der verbundenen Clients anzeigt. Zusätzlich zum Namen der Clients wird noch eine laufende Nummer vergeben, um verschiedene Clients gleichen Namens unterscheiden zu können. Daneben werden noch die Anzahl der Logmeldungen, die Uhrzeit der Verbindungsaufnahme bzw. des Ladens der Datei, sowie der Status des Clients angezeigt (geladene Datei, verbundener Client oder abgemeldeter Client).

Vom Hauptfenster aus können die Logfenster für die Clients geöffnet, neue Logdateien geladen, alte Clients gelöscht sowie einige Einstellungen verändert werden.

Beim Beenden von *qflog* werden die bestehenden Verbindungen zu den Clients getrennt und alle aktuellen Einstellungen, auch die für die Logfenster, in der Konfigurationsdatei gespeichert (zum Festlegen der Konfigurationsdatei siehe Abschnitt 3.1).

4.1 Laden und Speichern von Logdateien

Mittels des Datei→Öffnen... Menüs, läßt sich eine Logdatei mit dem üblichen Dateiauswahldialog laden. Wenn die Datei von *qflog* geschrieben wurde, bleibt der Name des Clients, der sie erzeugt hat erhalten. Andernfalls wird der Clientname "unbekannt" vergeben.

Zum Speichern der Logmeldungen des selektierten Clients dient das Datei→Speichern unter... Menü. Über den Dateiauswahldialog wird die Datei ausgewählt, in der alle Logmeldungen des Clients gespeichert werden. Soll nur ein Teil der Meldungen geschrieben werden, kann dies aus dem Logfenster heraus veranlaßt werden (siehe Abschnitt 5.1.1).

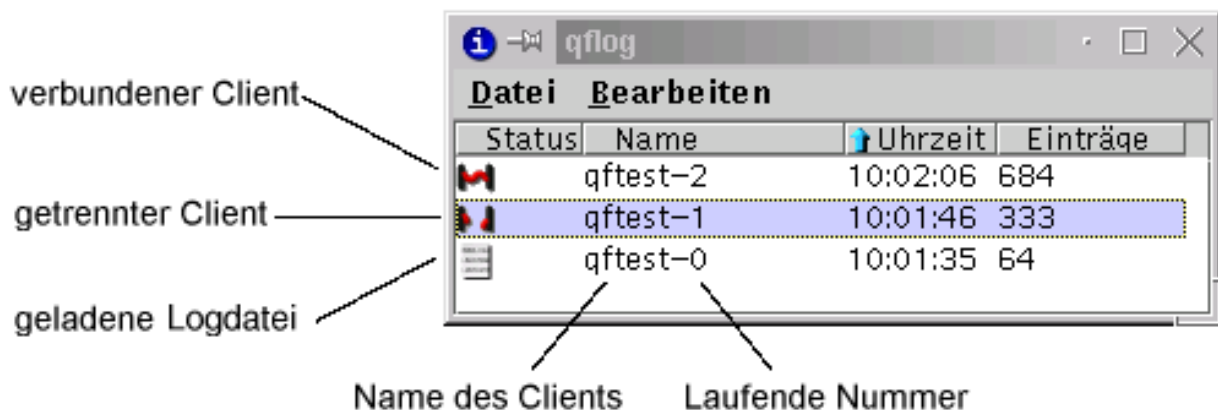


Abbildung 4.1: Das Hauptfenster

4.2 Öffnen des Logfensters

Das Logfenster des selektierten Clients kann sowohl durch drücken der **[Return]** Taste, als auch über das **[Bearbeiten→Anzeigen...]** Menü geöffnet werden.

4.3 Löschen von Clients

Während echte Clients automatisch in Abhängigkeit von den Einstellungen gelöscht werden, müssen geladene Logdateien bei Bedarf von Hand entfernt werden. Dies ist mittels der **[Delete]** Taste, oder des **[Bearbeiten→Löschen]** Menüs möglich.

4.4 Einstellungen

Der Dialog für die Einstellungen ist über das **[Bearbeiten→Einstellungen]** Menü zugänglich.

4.4.1 Allgemeine Einstellungen

Die folgenden Einstellungen dienen der Beschränkung der Anzahl von Clients, um den Speicherverbrauch in gewissen Grenzen zu halten. Alle Werte beziehen sich nur auf verbundene oder getrennte Clients, nicht aber auf geladene Logdateien. Wird einer der

Werte überschritten, so werden Clients gelöscht, wobei getrennte Clients vor verbundenen Clients, sowie die ältesten zuerst entfernt werden.

Maximale Anzahl Clients

Insgesamt erlaubte Zahl von verbundenen/getrennten Clients.

Maximale Anzahl Clients pro Name

Begrenzt die Zahl von Clients, die unter dem selben Namen angemeldet sind, unabhängig davon ob sie noch verbunden oder schon getrennt sind.

Maximale Anzahl nicht verbundener Clients

Maximalwert für getrennte Clients, unabhängig von ihrem Namen.

Maximale Anzahl nicht verbundener Clients pro Name

Das schärfste Kriterium. Es begrenzt die Zahl der Clients, die unter dem selben Namen verbunden waren und inzwischen getrennt sind.

Kapitel 5

Bedienung der Logfenster

5.1 Allgemeines

5.1.1 Aufbau des Logfensters

Ein Logfenster (Abbildung 5.1) besteht aus bis zu fünf Komponenten, von denen vier ein- oder ausgeblendet werden können.

In der Tabelle werden die Meldungen dargestellt. Zusätzlich zeigt die Detailansicht die zuletzt selektierte Meldung ausführlich an. Die Auswahl der Meldungen kann über die Filterkomponente eingestellt werden.

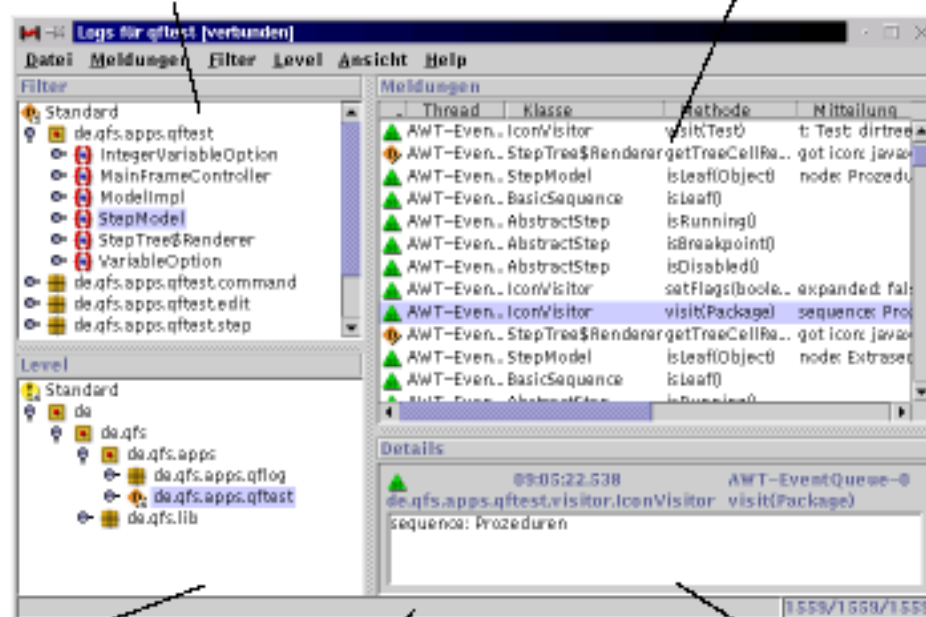
Die Loglevel Komponente ist nur dann vorhanden, wenn es sich um ein Logfenster für einen Client, oder ein integriertes Logfenster handelt. Sie macht für eine geladene Logdatei keinen Sinn, bleibt aber erhalten, wenn die Verbindung mit einem Client getrennt wird, da die Information noch von Interesse sein kann.

Zur Bedienung der drei Hauptkomponenten *Meldungen*, *Filter* und *Level* dienen die Menüs **Meldungen**, **Filter** und **Level**. Jede dieser Komponenten hat ein Kontextmenü, das genau dem zugehörigen Menü in der Menüleiste entspricht. Im folgenden wird daher auf die Kontextmenüs nicht mehr eingegangen.

Zur Statuszeile gibt es nicht viel zu sagen. Die drei Zahlen in der rechten Ecke bedeuten von links nach rechts:

- Die Anzahl der aktuell in der Tabelle angezeigten Meldungen.
- Die Gesamtzahl der im Moment im Logfenster verfügbaren, d.h. der angezeigten und der durch Filter ausgeblendeten Meldungen.
- Die Gesamtzahl aller Meldungen dieses Fensters, inklusive der bereits gelöschten.

Filter für die vorhandenen Meldungen Tabelle mit den Logmeldungen



Loglevel im Client

Statuszeile

Detailansicht einer Meldung

Abbildung 5.1: Ein Logfenster

Über das **Ansicht** Menü können alle Komponenten außer der Tabelle mit den Meldungen ein- oder ausgeblendet werden.

5.1.2 Speichern von Logdateien

Im **Datei** Menü befinden sich zwei Einträge zum speichern der Logmeldungen eines Logfensters. In beiden Fällen wird die zu schreibende Datei über einen Dateiauswahldialog festgelegt. Wie die Namen schon sagen, werden über **Alle Meldungen speichern unter...** alle Meldungen gespeichert, auch die, die im Moment nicht sichtbar sind. Die Reihenfolge der Logmeldungen entspricht dabei der Reihenfolge in der sie erzeugt wurden. Wird dagegen über **Sichtbare Meldungen speichern unter...** gespeichert, werden nur die Meldungen geschrieben, die gerade angezeigt werden, und zwar in der Reihenfolge der Anzeige in der Tabelle.

5.1.3 Laden und Speichern der Konfiguration

Sämtliche Einstellungen für das Logfenster und seine Komponenten werden beim Beenden von *qflog* in der Konfigurationsdatei gespeichert (vergleiche Kapitel 3.2). Dabei wird für jeden Clientnamen ein eigener Satz von Daten erzeugt. Man benötigt jedoch meist mehrere verschiedene Sätze von Filtereinstellungen für ein Programm. Über die Einträge **Konfiguration speichern unter...** und **Konfiguration laden...** im **Datei** Menü können gezielt Einstellungen in verschiedene Konfigurationsdateien geschrieben und von dort wieder geladen werden.

5.2 Die Tabelle

Der Hauptbestandteil des Logfensters ist die Tabelle mit den Meldungen (Abbildung 5.2). Sie ist immer vorhanden und kann nicht ausgeblendet werden.

Die Spalten der Tabelle entsprechen den Elementen einer Logmeldung:

Level

Der Level der Meldung wird mittels eines Icons dargestellt. Die Zuordnung der Icons zu den einzelnen Levels lässt sich am einfachsten anhand des **Filter** oder des **Level** Menüs erkennen.

Zeit

Die Zeit zu der die Meldung generiert wurde. Sie wird auf die Millisekunde genau angegeben.

Sortierung Filter Suchmuster

Meldungen Suche: abstract

...	Zeit	Thread	Klasse	Methode	Mitteilung
	09:45:58.389	AWT-Even...	AbstractStep	isRunning()	
	09:45:58.393	AWT-Even...	AbstractStep	isBreakpoint()	
	09:45:58.398	AWT-Even...	AbstractStep	isDisabled()	
	09:45:58.403	AWT-Even...	IconVisitor	setFlags(boole...	expanded: false, s...
	09:45:58.408	AWT-Even...	IconVisitor	visit(Package)	sequence: Prozedu...
	09:45:58.413	AWT-Even...	StepTree\$Render...	getTreeCellRe...	got icon: javax.swi...

Level der Meldung

Abbildung 5.2: Die Tabelle mit den Logmeldungen

Thread

Der Name des Threads, von dem aus die Meldung erzeugt wurde.

Klasse

Die Klasse des Objekts oder der `static` Methode, die die Meldung abgeschickt haben.

Methode

Die Methode, von der die Meldung stammt.

Mitteilung

Der eigentliche Inhalt der Meldung.

Die Reihenfolge und Breite der Spalten kann beliebig verändert werden. Diese Einstellungen werden für jeden Client getrennt in der Konfigurationsdatei gespeichert.

Die Sortierreihenfolge der Tabelleneinträge wird durch einen kleinen blauen Pfeil angezeigt. Ein einfacher Mausklick in einen Spaltenkopf legt die Sortierung anhand dieser Spalte fest, ein weiterer Click kehrt die Richtung der Sortierung um. Der selbe Effekt wird durch Eingabe von **[Ctrl-S]** für die selektierte Spalte erreicht, oder mittels des Menüeintrags **Spalte sortieren**.

5.2.1 Kopieren von Logmeldungen

Über die Menüeinträge **Markierte Meldungen kopieren**, **Sichtbare Meldungen kopieren** und **Alle Meldungen kopieren** lassen sich Logmeldungen in das System-Clipboard und damit in andere Programme

übernehmen. Vorsicht ist geboten, damit nicht zu viele Einträge kopiert werden, was manche Betriebssysteme überfordern könnte.

Das Format der kopierten Meldungen ist das selbe wie das, unter dem sie gespeichert werden.

Leider ist das System-Clipboard in manchen Versionen des *JDK* nicht richtig implementiert, so dass diese Funktionalität dort nicht zur Verfügung steht. Dies gilt unter anderem für die Linux Versionen.

5.2.2 Löschen von Einträgen

Um sich besser zurecht zu finden kann es sinnvoll sein, Teile der Logmeldungen zu löschen oder sogar alle Einträge zu entfernen, um die neu von einem verbundenen Client hereinkommenden Meldungen hervorzuheben. Die Menüeinträge **Alle Meldungen löschen**, **Unsichtbare Meldungen löschen** und **Sichtbare Meldungen löschen** entfernen entweder alle, die gerade weggefilterten, oder die gerade sichtbaren Einträge.

5.2.3 Zusatzfilter in der Tabelle

Die Auswahl der Einträge, die in der Tabelle sichtbar sind, wird hauptsächlich über die Filterkomponente gesteuert. Ein zusätzlicher Filtermechanismus innerhalb der Tabelle erlaubt es, diese Auswahl kurzfristig noch weiter einzuschränken.

Dieser Zusatzfilter kann für die Spalten *Level*, *Thread*, *Klasse* und *Methode* aktiviert werden. Er schränkt die angezeigten Einträge auf solche ein, die in dieser Spalte den selben Wert haben, wie der Eintrag, der zum Zeitpunkt der Aktivierung selektiert war. Zum Beispiel bewirkt eine Aktivierung des Zusatzfilters für die *Level* Spalte, bei einem selektierten Eintrag mit Level `ERR`, dass nur noch Einträge dieses Levels angezeigt werden.

Die Zusatzfilter für verschiedene Spalten lassen sich kombinieren. So lässt sich zum Beispiel die Ansicht auf Einträge einschränken, die zum Thread `AWT-EventQueue-0` gehören und gleichzeitig von der Klasse `de.qfs.lib.gui.SwingUtil` stammen.

Die Zusatzfilter können über den Menüeintrag **Spaltenfilter ein-/ausschalten** sowie über die Tastenkombination **Ctrl-F** aktiviert bzw. deaktiviert werden. Mittels **Ctrl-K** oder dem Menüeintrag **Alle Spaltenfilter ausschalten** lassen sich die Zusatzfilter für alle Spalten auf einmal deaktivieren.

5.2.4 Markierungen setzen und anspringen

Um wichtige Einträge einfach wiederzufinden, lassen sich bis zu 10 verschiedene Markierungen setzen und von jedem Punkt aus direkt wieder anspringen. In der aktuellen Version gibt es leider kein visuelles Feedback darüber, welche Markierungen wo gesetzt sind.

Das Setzen einer Markierung erfolgt über das **Marke setzen** Untermenü, oder über die Tastenkombinationen **Alt-0** bis **Alt-9**.

Angesprungen wird eine Marke analog über das **Zu Marke springen** Untermenü bzw. die Tastenkombinationen **Ctrl-0** bis **Ctrl-9**.

Aufgrund der Filtereinstellungen kann es sein, dass ein Eintrag, der zu einer Marke gehört, nicht sichtbar ist. In diesem Fall wird zu dem nächsten sichtbaren Eintrag entsprechend der aktuellen Sortierung gesprungen.

Wird eine markierte Meldung gelöscht, entweder explizit oder bei Überschreiten der zulässigen Anzahl von Einträgen, wird die Markierung nicht auf die nächstgelegene Meldung übertragen, sondern gelöscht, um Verwirrung zu vermeiden.

5.2.5 Suchen

Eine der wichtigsten Hilfen um sich in einer großen Menge von Logmeldungen zurechtzufinden ist die Suchfunktion. Sie ist quasi ständig aktiv und wird durch Eingabe von Text über die Tastatur ausgelöst.

Die Suche erfolgt grundsätzlich in der aktuellen Spalte. Im Fall der *Level* Spalte kann dabei natürlich nur nach einem Level gesucht werden, was aber sehr nützlich ist, um zum Beispiel den nächsten Fehler zu finden. Hier wird die Suche mittels der Tasten **0** bis **9** gesteuert, wobei die **1** dem Level `ERR`, die **9** dem Level `DBG` und die **0** dem Level `DBGDETAIL` entsprechen.

In den anderen Spalten kann nach beliebigem Text gesucht werden. Groß- und Kleinschreibung werden dabei grundsätzlich ignoriert. Das aktuelle Suchmuster wird in der Überschrift der Tabellenkomponente angezeigt. Mittels der **Backspace** Taste lassen sich die letzten Zeichen des Musters löschen, die **Escape** Taste leert es komplett.

Für alle Spalten sind zusätzlich einige weitere Tastenkombinationen relevant. Die Richtung der Suche lässt sich mittels **Ctrl-R** umkehren. **F2** übernimmt den Wert der aktuellen Zelle in das Suchmuster und **F3** wiederholt die Suche.

Die Suche springt jeweils zum nächsten Eintrag (abhängig von der Suchrichtung), der in der aktuellen Spalte das Suchmuster enthält (egal ob am Anfang oder in der Mitte der Spalte). Wird das Ende der Tabelle erreicht, wird die Suche automatisch am anderen Ende der Tabelle fortgesetzt. Dieser Umbruch wird neben dem Suchmuster in der

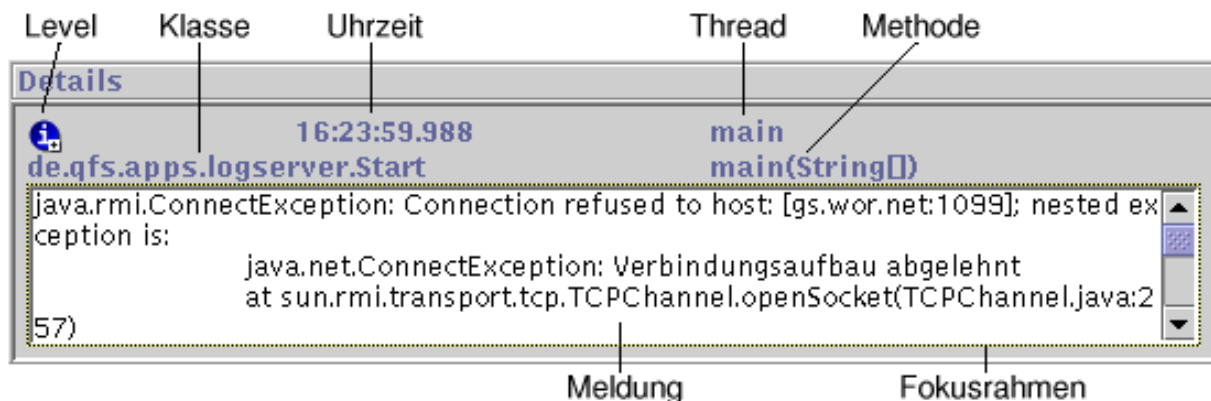


Abbildung 5.3: Die Detailansicht

Überschrift angezeigt. Dort wird auch ein Fehlschlagen der Suche gemeldet.

5.2.6 Einstellungen

Es gibt nur zwei Einstellungen, die für die Tabelle relevant sind, nämlich ob Linien angezeigt werden sollen und wie viele Meldungen in der Tabelle Platz finden. Wird diese Zahl überschritten, werden die ältesten Meldungen gelöscht. Ein Wert von 0 bedeutet keine Beschränkung.

Der Dialog für diese Einstellungen ist über den **Einstellungen...** Menüeintrag zugänglich.

5.3 Die Detailansicht

In der Detailansicht (Abbildung 5.3) wird die zuletzt selektierte Meldung dargestellt. Der Level wird durch sein Icon repräsentiert, Thread, Uhrzeit, Klasse und Methode werden direkt angezeigt. Die möglicherweise mehrzeilige Meldung erscheint in einem größeren Textfeld. Diesem kann durch einen Mausklick oder die **Tab** Taste der Fokus gegeben werden, so dass Scrollen über die Tastatur möglich wird.

Über den Menüeintrag **Ansicht→Details anzeigen** kann die Detailansicht ein- oder ausgeblendet werden.

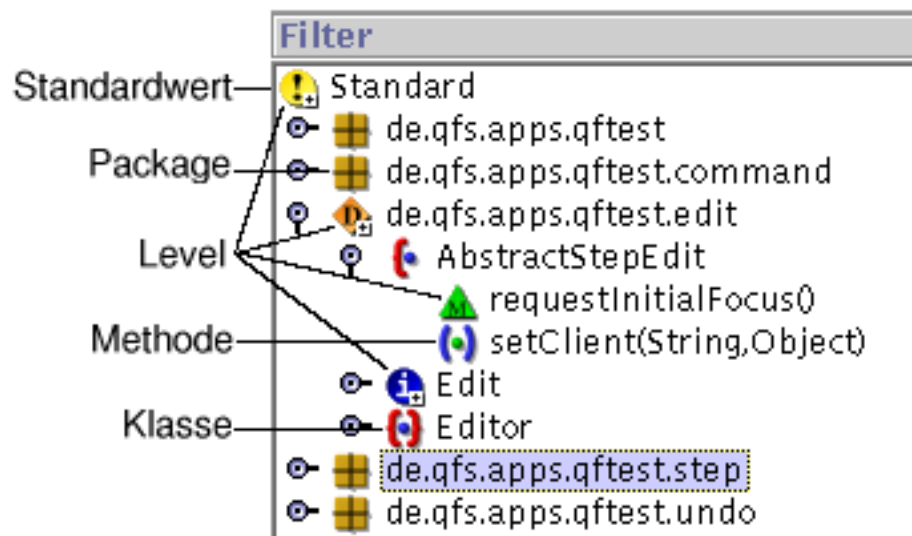


Abbildung 5.4: Der Filterbaum

5.4 Der Filterbaum

Das wichtigste Kontrollinstrument, um der Flut an Logmeldungen Herr zu werden, ist der Filterbaum (Abbildung 5.4). Mit seiner Hilfe kann auf Package-, Klassen- oder Methodenebene festgelegt werden, bis zu welchem Level Meldungen von diesem Package, bzw. dieser Klasse/Methode angezeigt werden.

Im Filterbaum ist Mehrfachselektion möglich. Alle Aktionen, egal ob über die Tastatur, das **Filter** Menü oder das Kontextmenü, beziehen sich auf alle selektierten Knoten.

Wenn die Filtereinstellungen geändert werden, führt dies zu einer Neuanzeige der Meldungen in der Tabelle. Dabei wird versucht, die Selektion beizubehalten, soweit die selektierten Einträge bei der neuen Filtereinstellung sichtbar sind. Darüber hinaus wird für den zuletzt selektierten Eintrag, d.h. die Tabellenzeile in der sich der Fokusrahmen befand, der nächste sichtbare Eintrag bezüglich der aktuellen Sortierreihenfolge gesucht, selektiert und in den sichtbaren Ausschnitt der Tabelle gescrollt.

5.4.1 Die Struktur des Baums

In der Struktur des Baums sind die Klassen und Methoden aller Meldungen dieses Logfensters angeordnet. Die Methoden werden unterhalb ihrer Klasse gruppiert, während die Klassen wiederum in ihrem Package zusammengefaßt sind. Die Packages stehen direkt unterhalb des Wurzelknotens. Die Verzeichnishierarchie der Packages wird in diesem Baum im Gegensatz zu den Loglevels des Clients nicht abgebildet, d.h. die

Packages `de.qfs.lib.config` und `de.qfs.lib.gui` stehen nicht in einem Ast namens `de.qfs.lib` sondern direkt unter der Wurzel.

Jedem Knoten im Baum kann ein Loglevel zugeordnet werden, bis zu dem zugehörige Meldungen in der Tabelle angezeigt werden sollen. Die Entscheidung, ob eine Meldung angezeigt wird oder nicht, wird wie folgt getroffen: Zunächst wird der Level der im Knoten der zugehörigen Methode gesetzt ist mit dem Level der Meldung verglichen. Ist der Level der Meldung höher, d.h. weniger wichtig, als der Level im Knoten, wird die Meldung nicht angezeigt, andernfalls schon.

Ist der Level im Knoten nicht gesetzt, wird der Level der übergeordneten Klasse verwendet. Ist dieser auch nicht gesetzt, der Level des Packages bzw. der Standardwert im Wurzelknoten, falls das Package ebenfalls keinen Wert hat.

Die Einstellung in Abbildung 5.4 hätte also folgende Wirkung:

- Der Standardwert ist `WRNDETAIL`. All Meldungen für die nichts näheres festgelegt ist, werden bis zu diesem Level wiedergegeben.
- Meldungen von Klassen, die zum Package `de.qfs.apps.qftest.edit` gehören, werden bis zum Level `DBGDETAIL` angezeigt, dem höchsten Level.
- Ausgenommen davon sind alle Meldungen, die von der Klasse `de.qfs.apps.qftest.edit.Edit` ausgehen, für welche der Level `MSGDETAIL` gesetzt ist.
- Ebenso ausgenommen sind alle Meldungen der Methode `requestInitialFocus()` in der Klasse `de.qfs.apps.qftest.edit.AbstractStepEdit`, da für diese Methode der Level `MTD` gilt.

Durch diese hierarchische Anordnung der Level, lassen sich gezielt die Meldungen anzeigen, die für das aktuelle Problem von Bedeutung sind. Stellt sich bei genauerem hinsehen heraus, dass man zusätzliche Information braucht, so läßt sie sich entsprechend hinzunehmen. Dabei muß natürlich bedacht werden, dass nur Meldungen angezeigt werden können, die auch tatsächlich generiert worden sind. Die Steuerung der Erzeugung von Meldungen ist dem Filterbaum sehr ähnlich und wird in Abschnitt 5.4.4 beschrieben.

Erfahrungsgemäß gibt es bei der Softwareentwicklung besonders heikle Bereiche, in denen immer wieder Fehler zu suchen sind. Die Einstellungen des Filterbaums können zusammen mit den anderen Einstellungen des Logfensters in verschiedenen Konfigurationsdateien - auch für den selben Client - gespeichert werden. Damit ist ein einfacher Wechsel zwischen verschiedenen Problemzonen möglich.

5.4.2 Festlegen der Filterlevel

Die Filterlevel können am einfachsten über die Tasten [0] bis [9] gesetzt werden. Wie bei der Suche in der Tabelle entspricht die [1] dem Level `ERR`, die [9] dem Level `DBG` und die [0] dem Level `DBGDETAIL`.

Für jeden Level gibt es außerdem einen eigenen Menüeintrag im **Filter** Menü und im Kontextmenü des Baums.

Das Löschen von Filterleveln ist in zwei Varianten möglich. Die einfache Variante, erreichbar über die **Delete** Taste bzw. den **Level entfernen** Menüeintrag, löscht nur die Level der selektierten Knoten des Baums. Die rekursive Variante entfernt zusätzlich auch die Level in allen Unterknoten der selektierten Knoten. Sie ist über den **Level rekursiv entfernen** Menüeintrag oder die Tastenkombination **Ctrl-Delete** zugänglich.

5.4.3 Anzeigen der gesetzten Level

Eine nützliche Funktion verbirgt sich hinter dem **Gesetzte Level anzeigen** Menüeintrag. Hiermit werden die Äste des Baums so ein- oder ausgeklappt, dass alle explizit gesetzten Level sichtbar sind.

In dieser Ansicht hat man den besten Überblick über die Filtereinstellungen, daher ist dies auch die initiale Anzeige nach dem Öffnen des Logfensters.

5.4.4 Zusätzliche Filterfunktion

Ähnlich zu den Zusatzfiltern in der Tabelle (siehe Abschnitt 5.2.2) gibt es im Filterbaum eine weitere Filterfunktion, jedoch mit etwas anderem Charakter als in der Tabelle.

Diese Zusatzfunktion lässt sich als ganzes über den **Extra Filter ein-/ausschalten** Menüeintrag aktivieren und deaktivieren. Den selben Effekt hat die Tastenkombination **Ctrl-F**.

Die Knoten des Baums lassen sich über die Tastenkombination **Ctrl-A** oder den Menüeintrag **Extra Filter hinzufügen** als extra Filter markieren. Ist die Zusatzfunktion der extra Filter aktiviert, so werden als extra Filter markierte Knoten mit roter Schrift dargestellt.

Diese Markierung lässt sich mittels **Ctrl-R** für die selektierten Knoten oder mittels **Ctrl-K** für alle extra Filter entfernen. Diese Funktionen sind über die Menüeinträge **Extra Filter entfernen** und **Alle extra Filter entfernen** zugänglich.

Die Aktivierung der extra Filter bewirkt, dass über die normale Filterwirkung des Baums

hinaus, nur noch Einträge angezeigt werden, die zu einem als extra Filter markierten Baumknoten gehören. Dabei wird wieder die Hierarchie der Baumstruktur berücksichtigt, d.h. ein als extra Filter markierter Knoten für eine Klasse aktiviert gleichzeitig die Darstellung aller Einträge für alle Methoden dieser Klasse, immer vorausgesetzt, dass diese Einträge bezüglich der normalen Filterwirkung des Baumes sichtbar sind. Ebenso bewirkt die Aktivierung eines Package-Knotens die Hinzunahme aller Klassen dieses Packages.

Die Funktionalität der extra Filter ist als einfache Möglichkeit gedacht, kurzfristig die Filterwirkung zu verschärfen, ohne die Level im Baum anfassen zu müssen. So kann über die Aktivierung zweier Methoden als extra Filter gut die Interaktion zwischen diesen beiden Methoden studiert werden, um sie nach der Deaktivierung der extra Filter Funktionalität wieder im Gesamtkontext zu betrachten.

Ein weiteres Anwendungsbeispiel ist die Suche nach einem Aufruf einer Methode mit besonderen Werten, die aus irgendwelchen Gründen nicht über die Suchfunktion der Tabelle (siehe Abschnitt 5.2.4) gefunden werden können. Aktivieren sie einfach die extra Filter und markieren sie nur den Knoten dieser Methode. Nun sind nur noch Logmeldungen, die von dieser Methode stammen in der Tabelle sichtbar und die gesuchte Meldung sollte einfach visuell zu ermitteln sein. Selektieren sie diesen Eintrag oder setzen sie eine Marke (siehe Abschnitt 5.2.3) und deaktivieren sie dann die extra Filter.

Aufgrund des kurzfristigen Charakters der extra Filter, werden diese *nicht* wie die Levels des Filterbaums in den Konfigurationsdateien gespeichert.

5.5 Die Loglevel im Client

Diese Komponente ist nur dann im Logfenster vorhanden, wenn das Fenster zu einem Client gehört. Sie gibt Zugang zu den Einstellungen des Clients, die für die Generierung der Logmeldungen verantwortlich sind. Die Bedienung dieser Komponente ist der des Filterbaums (siehe Abschnitt 5.3) sehr ähnlich.

5.5.1 Die Struktur des Baums

Wie in Abbildung 5.5 zu sehen, ist die Struktur des Levelbaums der des Filterbaums (siehe Abschnitt 5.4) bis auf zwei Unterschiede sehr ähnlich. Erstens endet die Hierarchie bereits auf Klassenebene, d.h. es gibt keine Knoten, die Methoden repräsentieren. Zweitens sind die Packages hierarchisch nach ihrer Struktur und nicht flach unter der Wurzel angeordnet, d.h. das Package `de.qfs.apps.qftest` ist vier Ebenen tief unter der Wurzel und den Knoten `de`, `de.qfs` und `de.qfs.apps` angeordnet.

Beide Unterschiede resultieren daraus, dass der Levelbaum genau die Struktur der Level der `Logger` Objekte im Client wiedergibt.

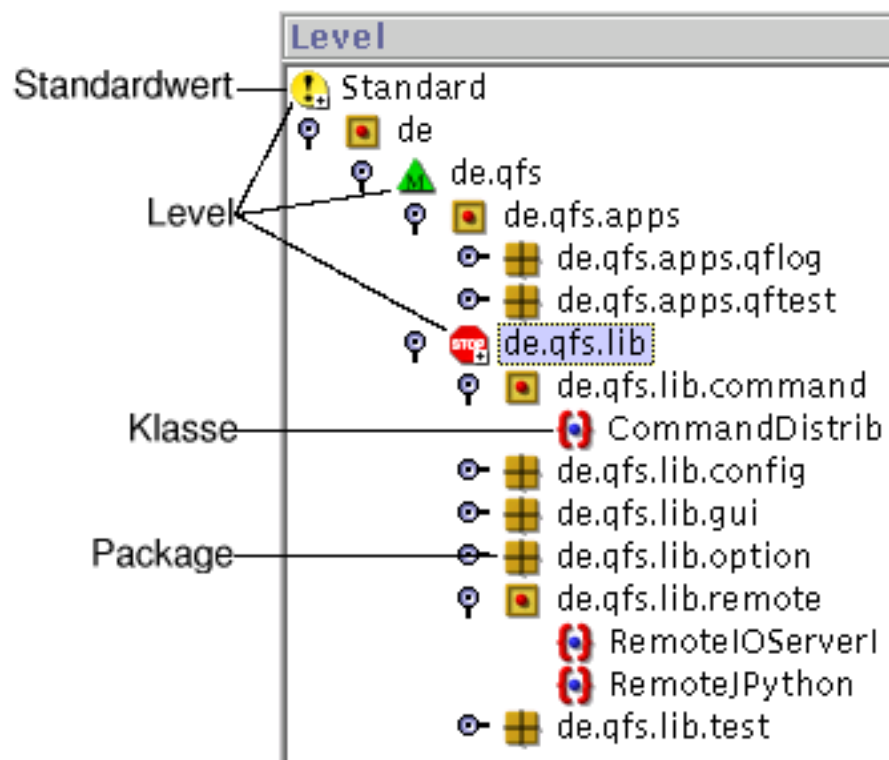


Abbildung 5.5: Der Levelbaum

5.5.2 Festlegen der Level

Die Level der Klassen und Packages im Levelbaum werden genauso gesetzt, wie die des Filterbaums (siehe Abschnitt 5.4.1), d.h. über die Tasten [0] bis [9], das [Level] Menü oder das Kontextmenü.

Ebenfalls analog funktioniert das Löschen der Level mittels [Delete], [Ctrl-Delete] oder die Menüs.

5.5.3 Anzeigen der gesetzten Level

Auch den Überblick über die gesetzten Level kann sich genau wie im Filterbaum über den Menüeintrag [Level→Gesetzte Level anzeigen] verschaffen (siehe Abschnitt 5.4.2).

5.5.4 Einstellungen

Eine ganze Anzahl von Einstellungen kann in dem Dialog verändert werden, der über den [Einstellungen...] Menüeintrag geöffnet wird. Bis auf die letzte Option entsprechen die Werte den Parametern des Logging Systems im Client. Bitte lesen Sie zum Verständnis dieser Werte die Dokumentation der Klasse `de.qfs.lib.log.Log`.

Der Schalter namens "Einstellungen des Clients beim Verbinden überschreiben" bestimmt das Verhalten von *qflog*, wenn ein sich ein neuer Client unter dem selben Namen anmeldet. Ist der Schalter aus, werden die Einstellungen für die Optionen und die Level vom Client übernommen. Andernfalls werden die Werte im Client mit der aktuellen Konfiguration überschrieben. Diese Option wird noch etwas sinnvoller, wenn auch die Möglichkeit vorhanden ist, die Werte im Client beim beenden der Verbindung wiederherzustellen.

Kapitel 6

Ein Anwendungsbeispiel

Die *qflog* Distribution enthält ein kleines Testprogramm, mit dessen Hilfe sich der Einsatz von *qflog* gut demonstrieren läßt. Es generiert zufällige Logmeldungen in verschiedenen Threads und kann diese wahlweise in eine Datei schreiben, ein eigenes Logfenster hochziehen, oder mit *qflog* über RMI kommunizieren, wobei sich diese Anwendungen kombinieren lassen.

6.1 Aufruf

Der Testclient kann direkt mittels

```
java [java-optionen...] de.qfs.apps.qflog.TestClient  
[optionen...]
```

gestartet werden, oder mit Hilfe der Startskripte im `bin` Verzeichnis der Distribution:

```
testclient [optionen...]
```

Nach dem Start erscheint lediglich ein triviales Fenster mit einem Button zum Beenden des Clients und er beginnt in Abhängigkeit von den angegebenen Optionen mit dem Loggen.

6.2 Optionen

Der Testclient interpretiert folgende Optionen:

-allownonlocal

Erlaubt den Zugriff für einen *qflog* Server von anderen Rechnern als dem local-host.

-clientname <Name>

Legt den Namen fest unter dem sich der Testclient beim *qflog* Server anmeldet. Standardwert ist "testclient".

-configfile <Datei>

Gibt eine Datei für die Konfigurationsinformationen an.

-createregistry

Erlaubt das Erzeugen einer RMI Registry, wenn beim Start des Clients noch keine vorhanden ist.

-internal

Erzeugt ein internes Logfenster, das beim Start des Testclients geöffnet wird.

-logfile <Datei>

Bewirkt die Ausgabe von Logmeldungen in eine Datei.

-logserver <Servername>

Weist den Testclient an, Verbindung mit einem Logserver aufzunehmen. Der korrekte Servername ist "qflog", sofern *qflog* beim Start nicht mit der `-servername` Option ein anderer Name zugewiesen wurde.

-numloggers <Anzahl>

Legt die Anzahl der Threads fest, die Logmeldungen erzeugen. Standardwert ist 3. Der erste Thread erzeugt Meldungen im Sekundenabstand, der zweite alle 2 Sekunden, der dritte alle 3 Sekunden usw.

-outputlevel <Level>

Legt den Level fest, bis zu dem Meldungen auf `System.err` ausgegeben werden. 0 unterdrückt die Ausgabe, 10 erlaubt alle Meldungen. Standardwert ist 2.

-port <Portnummer>

Legt die Portnummer der Registry fest, in der sich der Client zum Warten auf einen Logserver registrieren soll.

-waitforserver

Weist den Client an, sich in der RMI Registry einzutragen, so dass ein später gestarteter Logserver mit ihm Verbindung aufnehmen kann.

Dazu kommen noch Optionen der Form `-log-<Name> <Level>`, die die Level der Logger für die Ausgaben festlegen. Näheres dazu finden Sie in den Beispielen in der *qflib* Dokumentation.

6.3 Beispiele

Probieren Sie doch einfach die folgenden Beispiele aus, um die Anwendung von *qflog* kennzulernen:

- `testclient -internal` demonstriert, wie das Logfenster für den Testclient aussieht. Variieren Sie die Einstellungen in den Filter- und Levelbäumen und beobachten Sie das Ergebnis.
- Starten Sie zunächst *qflog* und anschliessend `testclient -logserver qflog`. Im *qflog* Hauptfenster sollte jetzt ein Eintrag namens "testclient" erscheinen. Öffnen Sie das zugehörige Logfenster, es verhält sich genauso wie das interne.
- Sie können beide Beispiele kombinieren, indem Sie mittels `testclient -internal -logserver qflog` sowohl das interne Logfenster öffnen, als auch die Verbindung mit dem Logserver aufnehmen. Beachten Sie, wie sich Änderungen im Levelbaum sofort auf das jeweils andere Logfenster auswirken.
- Starten Sie zunächst den Testclient mit `testclient -waitforserver -createregistry` und anschliessend *qflog*. Auch in diesem Fall sollte der "testclient" Eintrag im Hauptfenster von *qflog* erscheinen.

Anhang A

qflog und Applets

Die Kommunikation zwischen einem Applet und *qflog* ist aus mehreren Gründen problematisch. Zum einen fehlt in den meisten Versionen des Internet Explorer die RMI Unterstützung, zum anderen fallen RMI Callbacks unter die Sicherheitseinschränkungen, die Applets von der Sandbox auferlegt werden.

In jedem Fall kann aufgrund der Sandbox nur eine Verbindung zu *qflog* aufgenommen werden, wenn *qflog* auf dem Rechner läuft, von dessen Webserver das Applet geladen wurde. Ein entsprechend signiertes Applet sollte in der Lage sein, diese Beschränkung zu umgehen.

Dennoch lässt sich auch für Applets bereits jetzt einiger Nutzen aus dem Logging mittels *qflog* ziehen. In Zukunft wird es durch die Implementierung anderer Protokolle als RMI hoffentlich noch einfacher werden.

A.1 Internet Explorer

Das Hauptproblem für das Logging aus Applets im Internet Explorer ist die fehlende RMI Unterstützung in älteren Versionen (vermutlich vor 5.0). Allerdings lässt sich diese Unterstützung nachrüsten, was zumindest für den Entwicklungsrechner kein Problem darstellt. Microsoft bietet hierzu ein `jar` Archiv unter <http://www.microsoft.com/Java/resource/misc.htm> an. Mit diesem Archiv funktionieren dann sogar RMI Callbacks, so dass auch das Einstellen der Level zur Laufzeit des Applets unterstützt wird.

A.2 Netscape

Alle Netscape Versionen ab ca. 4.02 unterstützen *JDK 1.1* vollständig, allerdings nur in der Version 1.1.4. Damit können Applets zwar Verbindung zu *qflog* aufnehmen, allerdings keine Callbacks implementieren. Das bedeutet, dass keine Kontrolle zur Laufzeit über die Loglevel im Applet besteht. Theoretisch sollte sich diese Beschränkung mit einem signierten Applet beseitigen lassen, was wir aber noch nicht testen konnten.

A.3 Plugin

Wir hatten noch keine Gelegenheit, das Verhalten von Applets im *Java Plugin* von *SUN* zu testen und würden uns über Feedback dazu freuen, egal ob zur Plugin Version 1.1 oder 1.2.

A.4 Beispiel Applet

Ein Applet lässt sich so schreiben, dass es bei der Auslieferung lediglich das `de.qfs.lib.log` Package der *qflib* benötigt, während der Entwicklung aber die vollen Möglichkeiten ausnutzt. Ein Beispiel hierzu finden sie in den *qflib* Beispielen¹.

¹<http://www.qfs.de/de/projects/qflib/examples.html>