

SQLMonitor: SQL Logging innerhalb Omnis

Demonstration von SQL-Logging, SQL-Breakpoints,
Durchsuchen und Filtern des SQL-Logs anhand eines
mitgelieferten Tools

Autor

Udo Sonnabend
usonnabend@wigasoft.ch

Programming Omnis seit 1986
Start mit Omnis 3 auf Macintosh Plus

Company

Wigasoft AG

CH-9014 St.Gallen Schweiz

0041 (0) 71 274 51 31

www.wigasoft.ch

- Eine der führenden Software Firmen für Healthcare im Spital und Heimen.
- Software läuft auf Windows Plattformen
- Studio 10.22
- Fat Client Software
- Framework ist ein modifiziertes Masterstudio Framework von Marc Smit.

Zusammenfassung

Anhand einer Beispiels Library SQLMonitor.lbs wird gezeigt, wie sich SQL-Queries direkt in Omnis aufzeichnen lassen. Es beinhaltet einen Monitor mit dem das Log inspiziert, gefiltert und sich Breakpoints setzen lassen.

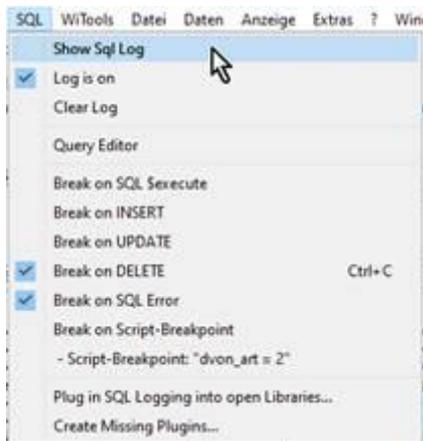
Die Library ist beiliegend und soll als Vorlage und Ideengeber gelten und kann auch beliebig abgeändert werden.

Test-Limits:

- Tool über 15 Jahre im Einsatz, kaum geändert
- Von Omnis Studio Version 5 bis 10
- Nur auf Fat-Client
- Nur Windows, rudimentär auf Mac
- Nur auf MSQL-Server, etwas SQLite
- Keine RPCs
- Keine Cursors
- Workers nur rudimentär abgebildet

Funktionsweise

Library in den Startup Folder von Omnis legen → Menu



- Monitorfenster in den Vordergrund holen
- Log an/ausschalten
- Aktuelles Log löschen

- Bei jedem SQL-Befehl anhalten
- Setzen von „groben“ Breakpoints
- Setzen von Script Breakpoints, regex-Suche

- Anpassen eines bestehenden Programms

Monitor

The screenshot displays the SQL Monitor application interface. At the top, there are settings for logging and breakpoints. The 'Log to File' level is set to 0. The 'Regex Break Script' is set to 'dVon_Art = 2'. The 'Break on' options include SQL Execute, UPDATE, DELETE, INSERT, SQL Error, and Script-Breakpoint.

The main log area shows a table of operations with columns: #, Operation, Call from, Sql, #CT, \$fetc..., Affected, Error, and ErrorText. The log shows several SQL queries being executed, including a complex query with multiple joins and a DECLARE statement.

Below the log, the 'SQL' section shows the full text of the selected query, including comments and SQL code. The 'Stack' section shows the call stack with columns: Class, Object, Method, Line, and Code. The stack includes methods like 'CurrentRecordChanged', 'PostNotification', 'ReleaseNotifications', and 'Sevent'.

#	Operation	Call from	Sql	#CT	\$fetc...	Affected	Error	ErrorText
420	SELECT / \$exe...	W\CareDocFallInfo.WdFallInfoMerkerSubwi...	SELECT count(*)FROM dat_VERORDNUNG INNER JOIN k...	0	0	-1	0	
421	SELECT / \$fetch	W\CareDocFallInfo.WdFallInfoMerkerSubwi...	SELECT count(*)FROM dat_VERORDNUNG INNER JOIN k...	0	10	-1	0	
424	SELECT / \$exe...	MoPatch.MoDatabase TableSuperclass.//\$.s...	SELECT CAST(qry_VERORDNUNG.qVON_Verordnung_ID as va...	0	0	-1		
425	SELECT / \$exe...	MoPatch.MoDatabase TableSuperclass.//\$.s...	SELECT CAST(kp_MEDIKAMENT.IMED_Medikament_ID as varc...	0	0	0		
435	SELECT / \$exe...	W\CareDocFallInfo.WdFallInfoMerkerSubwi...	DECLARE @Rea varchar(max)DECLARE @PatV varchar(max)...	0	0	-1	0	
436	SELECT / \$fetch	W\CareDocFallInfo.WdFallInfoMerkerSubwi...	DECLARE @Rea varchar(max)DECLARE @PatV varchar(max)...	0	1	-1	0	

Class	Object	Method	Line	Code
WdFallInfoMerkerDelegate		\$currentRecordChanged	2	Do MoContext.\$notificationCenter().\$postNotification(MoMasterDetailClasses. ^
MoNotificationCenter		\$postNotification	23	Do recipients.\$sendAll([\$ref.cSelectorPath](object))
MoNotificationCenter		\$releaseNotifications	57	Do !notificationCenterRef.\$postNotification(notification,,,,,kTrue)
WdFallFilterPane	MdDroplist	\$sevent	46	Do MoContext.\$notificationCenter().\$releaseNotifications

Bereiche:

- Anzeige Steuerung
- Breakpoints setzen
- Log mit Filter
- SQL Query
- Error Text
- Aufruf-Stack

Anzeigensteuerung

Bereiche des Monitor-Fensters lassen sich ein und ausblenden. Die Teilbereiche sind in der Grösse veränderbar.

- Logliste ist immer sichtbar
 - kann redrawed werden
 - kann gelöscht werden
 - kann als Datei speichern und zurück laden
- SQL-Text Anzeige mit markiertem Filter
 - Markierung lässt sich ein/ausschalten
 - Query lässt sich ggf. in separatem Editor ändern/testen
- Error: Anzeige mit Fehlercodes und Fehlertext
- Stack: Gesamter Stack zum Zeitpunkt des Queries

Geloggte Daten

API: Daten die die Methode \$addToSQLLog() aufzeichnet

- SQL-Skript, mit Ersetzung der Bindvariablen durch ihre Werte, wenn möglich
- Methoden-Stack, zum Zeitpunkt des Log-Aufrufs
- #CT: Die Dauer für die Durchführung des einzelnen Query, Fetch
- Anzahl betroffene Records
- Errors: Fehler und Fehlertext bei der Durchführung
- Call From: Leserliche Bezeichnung woher die Anfrage an die DB kam (Lib.Class/Method).

Das Log/Filter

- Filterbar:
 - mittels Regex-String (blau)
 - oder nach speziellen Kriterien, Doppelte, Langsame, usw.
- Spalten sind sortierbar.
- Das Log lässt sich speichern und zurückladen für spätere Vergleiche

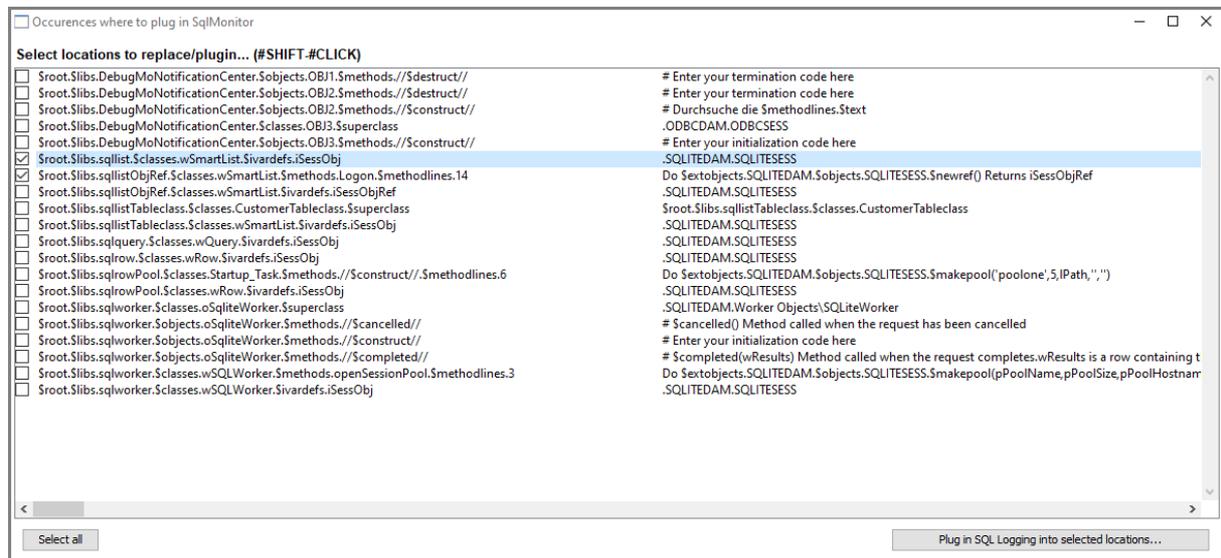
Breakpoints

Die Ausführung des Programms kann gestoppt werden:

- Anhalten des Programms bei jeder SQL-Operation
- bei speziellen Befehlen, zB. INSERT, UPDATE usw.
- wenn SQL-Fehler auftreten
- der SQL-Text bestimmte Key-Wörter (in Rot) enthält, die über einen Regex String definiert werden.

Plug in SQL Logging into open Libraries...

Das Tool lässt sich auf eigene Gefahr hin in geöffnete Libraries an gewünschten Orten einpluggen.



Im Fenster oben sind einige IDE- SQL-Samples von Omnis gezeigt. Orte an denen das Logging eingebaut werden soll, können selektiert werden. Die ersetzten Orte werden im Findlog von Omnis dargestellt. Findlog bitte gleich anschliessend wieder schliessen, siehe dort.

Schlussbemerkung

- Dieses Tool soll die Möglichkeit *skizzieren* wie so ein Log gebaut werden könnte.
- Für die aktuelle Situation müssen ggf. sehr grosse Anpassungen gemacht werden
- oder nur das Maintaining des Logs via Monitor verwenden.
- Das Ein-Pluggen der Klassen für das Logging ist auf eigene Gefahr hin.
- Um einfache Listen zu loggen verwendet die Demo auch unsupported Features von Omnis.