

SQL Logging

SQL Logging

Ab 4.19.0 und 4.18.2: Im Webclient kann der Super-User das Logging während des Betriebs an- und ausschalten, rechts unter dem Zahnrad-Dropdown Menüpunkt "Server Info". Log level für den gewünschten Logger auf "DEBUG" setzen:

The screenshot shows the Nuclos web client interface. At the top, there is a navigation bar with a search term "Search term" and a user profile "nuclos". Below the navigation bar, the "Server-Log" section is active. It shows "SQL logging" and a dropdown menu with options: "User management", "System parameters", and "Server info". The "Log level" is set to "DEBUG". There is a "Text filter" input field and a checked "Auto-scrolling" checkbox. The log content shows a single entry with a timestamp "18.07.2017 10:53:58" and a message: "UPDATE V594_AUFTRAG SET DATACHANGED = ?, STRCHANGED = ?, INTVERSION = ?, INTanzahl = ?, STRbemerkung = ?, STRUID_STRDOKUMENT = null, STRemail = ?, INTID_STRKATEGORIE = null, STRlink = ?, STRname = ?, STRnogroup = ?, STRnotiz = ?, c_passwort = ?, DBLpreis = ?, INTID_STRRECHNUNG = ?, DATwann = null, BLNwichtig = ?, STRNUCLOSSYSTEMID = ?, STRUID_NUCLOSPROCESS = null, STRNUCLOSSORIGIN = null, BLANUCLOSDELETED = ?, STRUID_NUCLOSSTATE = ? WHERE INTID = ? = [2017-07-18 10:53:58.091, nuclos, 27, 78, Ungültigxx, test@nuclos.de, www.google.de, Mein Auftrag, NoGroup2, Enter some text with mutiple lines, abc, 5.9, 40005674, false, AU1707#00029, false, OHibg9LOjrFishgHIWMo, 40005672]=(2 ms)". At the bottom, there is a "Clear" button and a "Message limit" set to "10000". A link "Download the complete log file" is also present.

SQL Timer per REST einschalten:

```
curl --cookie-jar cookies.txt http://localhost:8080/nuclos-war/rest -X POST -H "Content-Type: application/json" -d '{"username":"'nuclos'", "password":'""'}';
curl --cookie cookies.txt http://localhost:8080/nuclos-war/rest/maintenance/logging/SQLTimer/level -X PUT -H "Content-Type: application/json" -d '{"level":"'debug'"}';
```

bzw. reines SQL Logging vor der Ausführung ohne Timer mit "SQLLogger" statt "SQLTimer" bzw. mit "SQLUpdate" für das ausschließliche Loggen von "UPDATE/INSERT/DELETE"-Statements. Zum Abschalten des Logging den Level wieder von "debug" auf "info" setzen.

Standard SQL Logging

Um alle SQL Statements zu loggen: So sieht die Konfiguration der log4j2.xml aus, wenn einfache SQLLogging (**vor** dem Ausführen des SQL Statements) eingeschalten werden soll:

log4j2 - Ausgabe des SQL-Logging im Server-Log

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d %p [%c] - %m%n"/>
    </Console>
    <RollingFile name="Logfile" fileName="<Nuclos-Log-Verzeichnis>/server.log"
      filePattern="<Nuclos-Log-Verzeichnis>/server-%i.log" append="true">
      <PatternLayout pattern="%d %p [%c] - %m%n"/>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Logger name="org.apache.log4j.xml" level="info"/>
    <Logger name="SQLLogger" level="debug"/>
    <Root level="info">
      <AppenderRef ref="Console"/>
      <AppenderRef ref="Logfile"/>
    </Root>
  </Loggers>
</Configuration>
```

Die Datei log4j2.xml ist im "conf" Verzeichnis des Servers zu finden, zusammen mit den Dateien jdbc.properties und server.properties. Alternativ (oder additional) kann noch "SQLTiming" angeschaltet werden:

```
<Logger name="org.apache.log4j.xml" level="info"/>
<Logger name="SQLTimer" level="debug"/>
```

Damit wird das SQL **nach** der Ausführung des Statements auf der Datenbank geloggt und die Ausführungszeit in Millisekunden mit angegeben. Hier ist eine typische Ausgabe des SQLTimers:

```
2018-01-04 18:10:54,844 DEBUG [SQLTimer] - SELECT COUNT (t.INTID) FROM D2SC_FH_LASTPOSITION t WHERE 1=1
= []=(2 ms)
```

SQL StackTrace Logging

Es kann zusätzlich noch der StackTrace, sowohl vom Server, als auch vom Client, bis zur Ausführung des SQL-Statements geloggt werden. Dafür kann der Textfilter unter "Server Info" im Webclient verwendet werden (siehe oben): Außerdem kann in der Klasse "DataSourceExecuter" ein String-Suchbegriff angegeben werden, entweder über einen Debugger durch setzen des Strings in folgender Methode

```
public static String getDebugSQL() {
    return "SELECT COUNT";
}
```

oder durch setzen aus dem Programm mit der statischen Methoden DataSourceExecuter.setDebugSQL(String debugSQL)

```
DataSourceExecuter.setDebugSQL("SELECT COUNT");
```

Dann werden für alle SQL Statements die "SELECT COUNT" beinhaltet zwei erweiterte Stacktraces ausgegeben:

```
2018-01-04 18:10:54,843 DEBUG [ServerStackTrace] - Server Stack Trace:
org.nuclos.server.dblayer.impl.DataSourceExecutor.logStackTrace(DataSourceExecutor.java:179)
org.nuclos.server.dblayer.impl.DataSourceExecutor.beforeExecution(DataSourceExecutor.java:139)
org.nuclos.server.dblayer.impl.DataSourceExecutor.execute(DataSourceExecutor.java:230)
org.nuclos.server.dblayer.impl.standard.StandardSqlDBAccess.executeQueryImpl(StandardSqlDBAccess.java:295)
org.nuclos.server.dblayer.impl.standard.StandardSqlDBAccess.executeQuery(StandardSqlDBAccess.java:274)
org.nuclos.server.dblayer.impl.standard.StandardSqlDBAccess.executeQuery(StandardSqlDBAccess.java:262)
org.nuclos.server.dblayer.impl.standard.StandardSqlDBAccess.executeQuerySingleResult(StandardSqlDBAccess.java:400)
org.nuclos.server.dal.processor.jdbc.impl.EntityObjectProcessor.countExact(EntityObjectProcessor.java:647)
org.nuclos.server.dal.processor.jdbc.impl.EntityObjectProcessor.count(EntityObjectProcessor.java:625)
org.nuclos.server.common.ejb3.EntityObjectFacadeBean.countEntityObjectRows_aroundBody14(EntityObjectFacadeBean.java:297)
org.nuclos.server.common.ejb3.EntityObjectFacadeBean$AjcClosure15.run(EntityObjectFacadeBean.java:1)
org.nuclos.server.common.ejb3.EntityObjectFacadeBean.countEntityObjectRows(EntityObjectFacadeBean.java:293)
org.nuclos.server.common.NuclosRemoteInvocationExecutor.invoke(NuclosRemoteInvocationExecutor.java:207)
org.nuclos.server.common.NuclosHttpInvokerServiceExporter.invokeAndCreateResult
(NuclosHttpInvokerServiceExporter.java:13)

2018-01-04 18:10:54,843 DEBUG [ServerStackTrace] - Client Stack Trace:
org.nuclos.client.remote.http.NuclosHttpInvokerProxyFactoryBean.executeRequest
(NuclosHttpInvokerProxyFactoryBean.java:93)
org.nuclos.common.cache.AllEntityObjectsCache.refill(AllEntityObjectsCache.java:125)
org.nuclos.common.cache.AllEntityObjectsCache.register(AllEntityObjectsCache.java:118)
org.nuclos.common.cache.AllEntityObjectsCache.register(AllEntityObjectsCache.java:105)
de.bmw.fdm.common.fh.gaal.LastfallCommonCache.init(LastfallCommonCache.java:76)
org.nuclos.client.main.Startup$1.run(Startup.java:388)
```

Hinweis: Der Client Stack Trace ist nur erhältlich, wenn der Server im Entwicklungsmodus (-Dfunctionblock.dev=true) gestartet worden ist.

SQL Logging in separates Log-File

Hier steht, wie man [SQL Logging in separates Log-File](#) konfiguriert.

SQL Logging für alte Nuclos-Versionen (bis 4.5)

Hier sind die Logging-Einstellung für Nuclos bis 4.5 zu finden: [Altes SQL Logging \(Bis Nuclos 4.5\)](#)