z2-Environment - Bug #2136

Z2-DataSourceWorkResource throws NPE on "isClosed" when connection is already closed

31.01.2022 12:40 - Udo Offermann

Status:	New	**Start date:**	31.01.2022
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	z2-base	**Estimated time:**	0.00 hour
Target version:			
origin:			

Description

History

#1 - 31.01.2022 12:56 - Udo Offermann

- Subject changed from com.zfabrik.impl.db.data.DataSourceWorkResource#DataSourceWorkResource throws NPE on "isClosed" when connextion is already closed to Z2-DataSourceWorkResource throws NPE on "isClosed" when connection is already closed

#2 - 31.01.2022 13:07 - Udo Offermann

Während der Ausführung des MTS Demo-Szenarios tritt folgender Fehler reproduzierbar auf:

```
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]: java.lang.NullPointerException
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at sun.reflect.GeneratedMethodAccessor65.invoke(Unknown Source
)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingM
ethodAccessorImpl.java:43)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at java.lang.reflect.Method.invoke(Method.java:498)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.impl.db.data.DataSourceWorkResource.lambda$0(Da
taSourceWorkResource.java:44)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.sun.proxy.$Proxy162.isClosed(Unknown Source)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.resource.jdbc.internal.LogicalConnectionManag
edImpl.releaseConnection(LogicalConnectionManagedImpl.java:214)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.resource.jdbc.internal.LogicalConnectionManag
edImpl.afterTransaction(LogicalConnectionManagedImpl.java:175)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.engine.jdbc.internal.JdbcCoordinatorImpl.afte
rTransaction(JdbcCoordinatorImpl.java:275)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.engine.jdbc.internal.JdbcCoordinatorImpl.afte
rTransactionCompletion(JdbcCoordinatorImpl.java:454)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.resource.transaction.backend.jta.internal.Jta
TransactionCoordinatorImpl.afterCompletion(JtaTransactionCoordinatorImpl.java:384)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.resource.transaction.backend.jta.internal.syn
chronization.SynchronizationCallbackCoordinatorNonTrackingImpl.doAfterCompletion(SynchronizationCallbackCoordi
natorNonTrackingImpl.java:60)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.resource.transaction.backend.jta.internal.syn
chronization.SynchronizationCallbackCoordinatorTrackingImpl.afterCompletion(SynchronizationCallbackCoordinator
TrackingImpl.java:72)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.hibernate.resource.transaction.backend.jta.internal.syn
chronization.RegisteredSynchronization.afterCompletion(RegisteredSynchronization.java:44)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.impl.tx.TransactionImpl$Resource.afterCompletio
n(TransactionImpl.java:124)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit._afterCompletion(WorkUnit.java:43
4)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit.lambda$_afterCompletion$3(WorkUni
t.java:429)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit.visitResources(WorkUnit.java:486)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit._afterCompletion(WorkUnit.java:42
9)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit.commit(WorkUnit.java:224)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.impl.tx.TransactionImpl.commit(TransactionImpl.
java:229)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.impl.tx.UserTransactionImpl.commit(UserTransact
ionImpl.java:96)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.springframework.transaction.jta.JtaTransactionManager.d
oCommit(JtaTransactionManager.java:1035)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.springframework.transaction.support.AbstractPlatformTra
```

```
nsactionManager.processCommit(AbstractPlatformTransactionManager.java:743)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.springframework.transaction.support.AbstractPlatformTra
nsactionManager.commit(AbstractPlatformTransactionManager.java:711)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.springframework.transaction.interceptor.TransactionAspe
ctSupport.commitTransactionAfterReturning(TransactionAspectSupport.java:654)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.springframework.transaction.interceptor.TransactionAspe
ctSupport.invokeWithinTransaction(TransactionAspectSupport.java:407)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.springframework.transaction.aspectj.AbstractTransaction
Aspect.ajc$around$org_springframework_transaction_aspectj_AbstractTransactionAspect$1$2a73e96c(AbstractTransac
tionAspect.aj:71)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.mts.workq.impl.service.TaskExecutionServiceImpl.p
rogressTaskTX(TaskExecutionServiceImpl.java:249)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.mts.workq.impl.service.TaskExecutionServiceImpl.p
rogressTask(TaskExecutionServiceImpl.java:174)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.mts.workq.impl.job.TaskJobImpl.execute(TaskJobImp
l.java:38)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.tt.scheduler.JobDelegate.lambda$1(JobDelegate.jav
a:145)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.util.threading.ThreadUtil.lambda$cleanContextEx
ceptionExecute$0(ThreadUtil.java:55)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at java.security.AccessController.doPrivileged(Native Method)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.util.threading.ThreadUtil.cleanContextException
Execute(ThreadUtil.java:55)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.tt.scheduler.JobDelegate.lambda$0(JobDelegate.jav
a:140)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit.lambda$work$1(WorkUnit.java:396)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit.supply(WorkUnit.java:368)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.work.WorkUnit.work(WorkUnit.java:396)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.impl.work.ThreadPoolImpl.doIt(ThreadPoolImpl.ja
va:273)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.zfabrik.impl.work.ThreadPoolImpl.executeAs(ThreadPoolIm
pl.java:359)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.tt.scheduler.JobDelegate.executeWrappedJob(JobDel
egate.java:123)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.tt.scheduler.JobDelegate.execute(JobDelegate.java
:55)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at com.gi_de.tt.jobs.ComponentJob.execute(ComponentJob.java:34
)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.quartz.core.JobRunShell.run(JobRunShell.java:216)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:  at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleTh
readPool.java:549)
01/31 11:01:42 [24]...ent/jobWorker@0.2 [800]:
0
```

Das Problem ist, dass com.zfabrik.impl.db.data.DataSourceWorkResource#DataSourceWorkResource#close() den Member conn=null setzt (im Finally-Block), danach aber noch Aufrufe - wie z.B. isClosed() auf der Connection per Reflection zulässt bzw. selbst aufruft. dadurch wird die NPE im Connection-Wrapper ausgelöst, der als Proxy im Konstruktor aufgebaut wird:

```
        this.wrapper = (Connection) Proxy.newProxyInstance(
            this.getClass().getClassLoader(),
            new Class<?>[]{Connection.class},
            (Object proxy, Method method, Object[] args) -> {
                if (!"close".equals(method.getName())) {
NPE --->            return method.invoke(DataSourceWorkResource.this.conn, args);
                } else {
                    // if anybody set the connection to auto commit,
                    // we set it back to no-autocommit
                    if (conn.getAutoCommit()) {
                        LOG.warning("Setting pool connection back to non-autocommit");
                        conn.setAutoCommit(false);
                    }
                }
                return null;
            }
        );
```

Lösung

Der Proxy muss für method== isClosed true zurückliefern, wenn this.conn==null ist.
An dieser Stelle sollte auch für den Fall method close auf this.connnull getestet werden.

#3 - 31.01.2022 14:08 - Udo Offermann

- Category set to z2-base

In MTS wurde das Problem durch folgende Änderung im Konstruktor von com.zfabrik.impl.db.data.DataSourceWorkResource gelöst:

```
   public DataSourceWorkResource(DataSourceResource dataSourceResource, final Connection connection) throws
Exception {
        this.dataSourceResource = dataSourceResource;
        this.conn = connection;


// Delegate the calls to the wrapped connection except the close method which will be handled separately.
        this.wrapper = (Connection) Proxy.newProxyInstance(
                this.getClass().getClassLoader(),
                new Class<?>[]{Connection.class},
                (Object proxy, Method method, Object[] args) -> {
                    switch (method.getName()) {
                        case "isClosed":
                            // delegate "isClosed", unless we are already closed
                            return this.conn == null || this.conn.isClosed();

                        case "close":
                            if (this.conn != null) {
                                // if anybody set the connection to auto commit,
                                // we set it back to no-autocommit
                                if (this.conn.getAutoCommit()) {
                                    LOG.warning("Setting pool connection back to non-autocommit");
                                    this.conn.setAutoCommit(false);
                                }
                            }
                            break;

                        default:
                            if (this.conn != null) {
                                // delegate methods
                                return method.invoke(DataSourceWorkResource.this.conn, args);
                            } else {

// This is a preferred behavior based on the java.sql.Connection documentation.
                                throw new SQLException("Connection already closed");
                            }
                    }
                    return null;
                }
        );
    }
```