Hibernate Reactive Session/EntityManager is closed #23269

Closed gwenneg opened this issue on Jan 28 · 22 comments · Fixed by #23397

Labels	area/hibernate-reactive area/persistence area/resteasy-reactive kind/bug
Milestone	2.7.1.Final

gwenneg commented on Jan 28 • edited Member

Describe the bug

This may look similar to #22433 (which was fixed in `2.6.3.Final`) but this time the reproducer includes a resteasy-reactive request filter (see `RequestFilter` in the reproducer).

Here's the exception which is thrown randomly when the app is processing concurrent requests:

```
2022-01-28 15:11:21,411 ERROR [org.hib.rea.errors] (vert.x-eventloop-thread-4) HR000057: Failed to
execute statement [$1select fruit0_.id as id1_0_, fruit0_.name as name2_0_ from known_fruits
fruit0_ order by fruit0_.name]: $2could not execute query:
java.util.concurrent.CompletionException: java.lang.IllegalStateException: Session/EntityManager
is closed
        at
java.base/java.util.concurrent.CompletableFuture.encodeThrowable(CompletableFuture.java:314)
        at
java.base/java.util.concurrent.CompletableFuture.completeThrowable(CompletableFuture.java:319)
        at
java.base/java.util.concurrent.CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1081)
        at
java.base/java.util.concurrent.CompletableFuture.postComplete(CompletableFuture.java:506)
        at java.base/java.util.concurrent.CompletableFuture.complete(CompletableFuture.java:2073)
        at io.vertx.core.Future.lambda$toCompletionStage$2(Future.java:360)
        at io.vertx.core.impl.future.FutureImpl$3.onSuccess(FutureImpl.java:141)
        at io.vertx.core.impl.future.FutureBase.emitSuccess(FutureBase.java:60)
        at io.vertx.core.impl.future.FutureImpl.tryComplete(FutureImpl.java:211)
        at io.vertx.core.impl.future.PromiseImpl.tryComplete(PromiseImpl.java:23)
        at io.vertx.sqlclient.impl.QueryResultBuilder.tryComplete(QueryResultBuilder.java:102)
        at io.vertx.sqlclient.impl.QueryResultBuilder.tryComplete(QueryResultBuilder.java:35)
        at io.vertx.core.Promise.complete(Promise.java:66)
        at io.vertx.core.Promise.handle(Promise.java:51)
        at io.vertx.core.Promise.handle(Promise.java:29)
        at io.vertx.core.impl.future.FutureImpl$3.onSuccess(FutureImpl.java:141)
        at io.vertx.core.impl.future.FutureBase.lambda$emitSuccess$0(FutureBase.java:54)
        at
io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutor.java:164)
        at
io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:469)
        at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:503)
        at
io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:986)
        at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
        at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
        at java.base/java.lang.Thread.run(Thread.java:829)
Caused by: java.lang.IllegalStateException: Session/EntityManager is closed
        at
org.hibernate.internal.AbstractSharedSessionContract.checkOpen(AbstractSharedSessionContract.java:407

        at
org.hibernate.engine.spi.SharedSessionContractImplementor.checkOpen(SharedSessionContractImplementor.

        at
org.hibernate.reactive.session.impl.ReactiveSessionImpl.checkOpen(ReactiveSessionImpl.java:1558)
        at
org.hibernate.internal.AbstractSharedSessionContract.checkOpenOrWaitingForAutoClose(AbstractSharedSes

        at org.hibernate.internal.SessionImpl.getEntityUsingInterceptor(SessionImpl.java:603)
        at org.hibernate.loader.Loader.getRow(Loader.java:1610)
        at org.hibernate.loader.Loader.getRowFromResultSet(Loader.java:748)
        at org.hibernate.loader.Loader.getRowsFromResultSet(Loader.java:1047)
        at
org.hibernate.reactive.loader.hql.impl.ReactiveQueryLoader.getRowsFromResultSet(ReactiveQueryLoader.j

        at
org.hibernate.reactive.loader.ReactiveLoaderBasedResultSetProcessor.reactiveExtractResults(ReactiveLo

        at
org.hibernate.reactive.loader.hql.impl.ReactiveQueryLoader$1.reactiveExtractResults(ReactiveQueryLoad

        at
org.hibernate.reactive.loader.ReactiveLoader.reactiveProcessResultSet(ReactiveLoader.java:145)
        at
org.hibernate.reactive.loader.ReactiveLoader.lambda$doReactiveQueryAndInitializeNonLazyCollections$0(

        at
java.base/java.util.concurrent.CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1072)
        ... 21 more


2022-01-28 15:11:21,413 ERROR [io.qua.ver.htt.run.QuarkusErrorHandler] (vert.x-eventloop-thread-4)
HTTP Request to /fruits failed, error id: 51fc210d-cc94-487b-bfb9-5c5b593badfb-17:
java.lang.IllegalStateException: Session/EntityManager is closed
        at
org.hibernate.internal.AbstractSharedSessionContract.checkOpen(AbstractSharedSessionContract.java:407

        at
org.hibernate.engine.spi.SharedSessionContractImplementor.checkOpen(SharedSessionContractImplementor.
        at
org.hibernate.reactive.session.impl.ReactiveSessionImpl.checkOpen(ReactiveSessionImpl.java:1558)
        at
org.hibernate.internal.AbstractSharedSessionContract.checkOpenOrWaitingForAutoClose(AbstractSharedSes

        at org.hibernate.internal.SessionImpl.getEntityUsingInterceptor(SessionImpl.java:603)
        at org.hibernate.loader.Loader.getRow(Loader.java:1610)
        at org.hibernate.loader.Loader.getRowFromResultSet(Loader.java:748)
        at org.hibernate.loader.Loader.getRowsFromResultSet(Loader.java:1047)
        at
org.hibernate.reactive.loader.hql.impl.ReactiveQueryLoader.getRowsFromResultSet(ReactiveQueryLoader.j

        at
org.hibernate.reactive.loader.ReactiveLoaderBasedResultSetProcessor.reactiveExtractResults(ReactiveLo

        at
org.hibernate.reactive.loader.hql.impl.ReactiveQueryLoader$1.reactiveExtractResults(ReactiveQueryLoad

        at
org.hibernate.reactive.loader.ReactiveLoader.reactiveProcessResultSet(ReactiveLoader.java:145)
        at
org.hibernate.reactive.loader.ReactiveLoader.lambda$doReactiveQueryAndInitializeNonLazyCollections$0(

        at
java.base/java.util.concurrent.CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1072)
        at
java.base/java.util.concurrent.CompletableFuture.postComplete(CompletableFuture.java:506)
        at java.base/java.util.concurrent.CompletableFuture.complete(CompletableFuture.java:2073)
        at io.vertx.core.Future.lambda$toCompletionStage$2(Future.java:360)
        at io.vertx.core.impl.future.FutureImpl$3.onSuccess(FutureImpl.java:141)
        at io.vertx.core.impl.future.FutureBase.emitSuccess(FutureBase.java:60)
        at io.vertx.core.impl.future.FutureImpl.tryComplete(FutureImpl.java:211)
        at io.vertx.core.impl.future.PromiseImpl.tryComplete(PromiseImpl.java:23)
        at io.vertx.sqlclient.impl.QueryResultBuilder.tryComplete(QueryResultBuilder.java:102)
        at io.vertx.sqlclient.impl.QueryResultBuilder.tryComplete(QueryResultBuilder.java:35)
        at io.vertx.core.Promise.complete(Promise.java:66)
        at io.vertx.core.Promise.handle(Promise.java:51)
        at io.vertx.core.Promise.handle(Promise.java:29)
        at io.vertx.core.impl.future.FutureImpl$3.onSuccess(FutureImpl.java:141)
        at io.vertx.core.impl.future.FutureBase.lambda$emitSuccess$0(FutureBase.java:54)
        at
io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutor.java:164)
        at
io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:469)
        at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:503)
        at
io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:986)
        at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
        at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
        at java.base/java.lang.Thread.run(Thread.java:829)


2022-01-28 15:11:21,414 ERROR [org.jbo.res.rea.com.cor.AbstractResteasyReactiveContext] (vert.x-
eventloop-thread-4) Request failed: java.lang.IllegalStateException: Session/EntityManager is
closed
```

```
        at 
org.hibernate.internal.AbstractSharedSessionContract.checkOpen(AbstractSharedSessionContract.java:407

        at 
org.hibernate.engine.spi.SharedSessionContractImplementor.checkOpen(SharedSessionContractImplementor.

        at 
org.hibernate.reactive.session.impl.ReactiveSessionImpl.checkOpen(ReactiveSessionImpl.java:1558)
        at 
org.hibernate.internal.AbstractSharedSessionContract.checkOpenOrWaitingForAutoClose(AbstractSharedSes

        at org.hibernate.internal.SessionImpl.getEntityUsingInterceptor(SessionImpl.java:603)
        at org.hibernate.loader.Loader.getRow(Loader.java:1610)
        at org.hibernate.loader.Loader.getRowFromResultSet(Loader.java:748)
        at org.hibernate.loader.Loader.getRowsFromResultSet(Loader.java:1047)
        at 
org.hibernate.reactive.loader.hql.impl.ReactiveQueryLoader.getRowsFromResultSet(ReactiveQueryLoader.j

        at 
org.hibernate.reactive.loader.ReactiveLoaderBasedResultSetProcessor.reactiveExtractResults(ReactiveLo

        at 
org.hibernate.reactive.loader.hql.impl.ReactiveQueryLoader$1.reactiveExtractResults(ReactiveQueryLoad

        at 
org.hibernate.reactive.loader.ReactiveLoader.reactiveProcessResultSet(ReactiveLoader.java:145)
        at 
org.hibernate.reactive.loader.ReactiveLoader.lambda$doReactiveQueryAndInitializeNonLazyCollections$0(

        at 
java.base/java.util.concurrent.CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1072)
        at 
java.base/java.util.concurrent.CompletableFuture.postComplete(CompletableFuture.java:506)
        at java.base/java.util.concurrent.CompletableFuture.complete(CompletableFuture.java:2073)
        at io.vertx.core.Future.lambda$toCompletionStage$2(Future.java:360)
        at io.vertx.core.impl.future.FutureImpl$3.onSuccess(FutureImpl.java:141)
        at io.vertx.core.impl.future.FutureBase.emitSuccess(FutureBase.java:60)
        at io.vertx.core.impl.future.FutureImpl.tryComplete(FutureImpl.java:211)
        at io.vertx.core.impl.future.PromiseImpl.tryComplete(PromiseImpl.java:23)
        at io.vertx.sqlclient.impl.QueryResultBuilder.tryComplete(QueryResultBuilder.java:102)
        at io.vertx.sqlclient.impl.QueryResultBuilder.tryComplete(QueryResultBuilder.java:35)
        at io.vertx.core.Promise.complete(Promise.java:66)
        at io.vertx.core.Promise.handle(Promise.java:51)
        at io.vertx.core.Promise.handle(Promise.java:29)
        at io.vertx.core.impl.future.FutureImpl$3.onSuccess(FutureImpl.java:141)
        at io.vertx.core.impl.future.FutureBase.lambda$emitSuccess$0(FutureBase.java:54)
        at 
io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutor.java:164)
        at 
io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:469)
        at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:503)
        at 
io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:986)
        at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
        at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
        at java.base/java.lang.Thread.run(Thread.java:829)
```

Expected behavior

No response

Actual behavior

No response

How to Reproduce?

Reproducer: https://github.com/gwenneg/quarkus-hibernate-reactive-server-request-filter

Steps to reproduce the behavior:

- Start the app with `./mvnw clean quarkus:dev`
- Run jmeter.test.plan.zip with JMeter
- The exception should show up in the log eventually

⚠️ If you change the Quarkus version to `2.3.1.Final`, the exception should never be thrown. As a consequence, this looks like a regression to us but I'll wait for a confirmation on that specific point before tagging the issue as a regression.

Output of `uname -a` or `ver`

Linux glepage.remote.csb 4.18.0-348.2.1.el8_5.x86_64 #1 SMP Mon Nov 8 13:30:15 EST 2021 x86_64 x86_64 x86_64 GNU/Linux

Output of `java -version`

openjdk version "11.0.13" 2021-10-19 LTS

GraalVM version (if different from Java)

No response

Quarkus version or git rev

2.6.3.Final and 2.7.0.Final

Build tool (ie. output of `mvnw --version` or `gradlew --version`)

No response

Additional information

No response

gwenneg added the kind/bug label on Jan 28

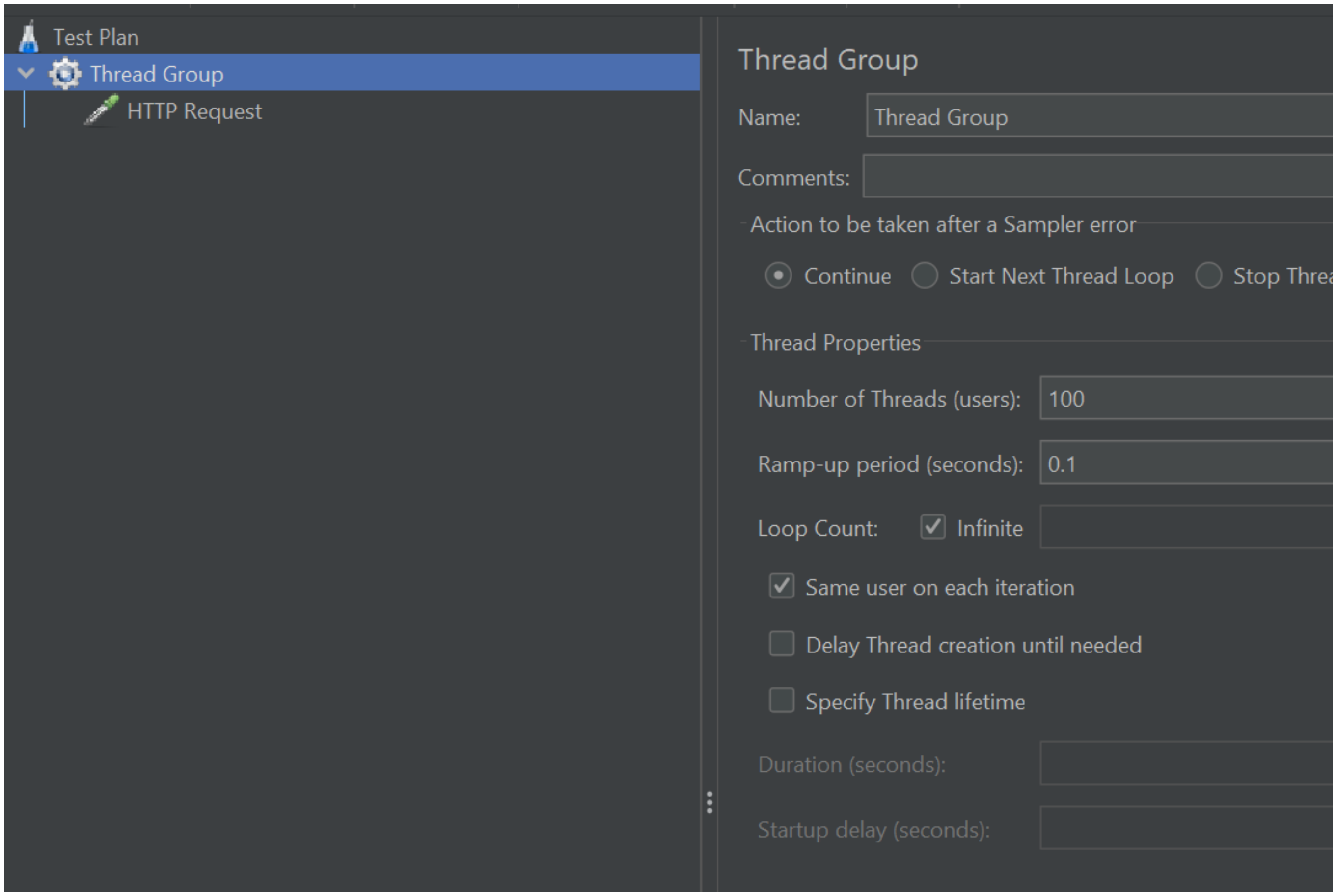
quarkus-bot bot added area/hibernate-reactive area/persistence labels on Jan 28

quarkus-bot bot commented on Jan 28

/cc **@DavideD**, **@Sanne**, **@gavinking**

gwenneg commented on Jan 28 Member Author

In case you have an issue with the enclosed zip file, here's the JMeter configuration I used:



Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Threa

Thread Properties

Number of Threads (users): 100

Ramp-up period (seconds): 0.1

Loop Count: ✓ Infinite

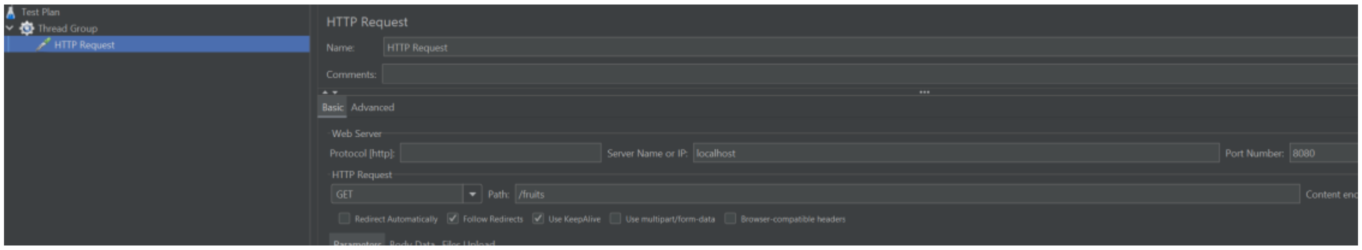
✓ Same user on each iteration

Delay Thread creation until needed

Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):



HTTP Request

Name: HTTP Request

Comments:

Web Server

Protocol [http]: Server Name or IP: localhost Port Number: 8080

HTTP Request

GET Path: /fruits Content enc

[ ] Redirect Automatically [✓] Follow Redirects [✓] Use KeepAlive [ ] Use multipart/form-data [ ] Browser-compatible headers

gwenneg commented on Jan 28 Member Author

@geoand I'm not sure whether this bug is caused by hibernate-reactive or resteasy-reactive as it's only happening when a request filter is involved, so I'm tagging you as well 😄

geoand commented on Jan 28 Contributor

I don't really see how it could be caused by RESTEasy Reactive, but I'll try and take a look next week

👍 1

gwenneg mentioned this issue on Jan 28

Downgrade Quarkus to 2.3.1.Final RedHatInsights/notifications-backend#947

Merged

geoand commented on Jan 31 Contributor

I don't see anything wrong from a RESTEasy Reactive perspective.

Just for completeness sake I'll mention that the filter **does** run a Vert.x event-loop thread as expected.

gwenneg commented on Jan 31 Member Author

Thanks for checking!

👍 1

DavideD self-assigned this on Feb 1

Sanne commented on Feb 1 Member

I've had a debugging session with **@DavideD**, we figured that the problem is caused by the fact that the `Mutiny.Session` used in the request filter and the one used in the resource are actually the same, as the `withSession` method will reuse a session instance from the vert.x context.

So the two sessions are the same instance, which is totally OK and expected from the point of view of Hibernate Reactive and vert.x.

However, the method annotated with `@ServerRequestFilter` (`Uni<Response> filter(ContainerRequestContext requestContext)`) is returning an Uni which is being run *interleaved* with the `Uni` being returned by the method annotated with `@Get` , which is illegal and results in state-changing operations being applied in the wrong order, ultimately leading to an attempt to use possibly closed Session instance.

I'm not sure why this happens but it seems to relate with the fact that `@ServerRequestFilter` is being processed as a blocking "around invoke" interceptor, instead of chaining the Uni operations. This seems intentional if I read the javadoc of this method but I don't really understand why a Uni should be returned for a blocking operation (?) and how I would be supposed to return a properly non-blocking operation up to the chain.

@geoand / **@FroMage** ? It seems a bit ironical that RestEasy Reactive doesn't support a reactively implemented filter?

I don't see what we can do in HR to improve on this, other than possibly be even more strict with checks and aggressively throw an exception, as I'm worried that this only gets identified as a problem under load.

@gwenneg an easy workaround would be to not use `withSession` :/

geoand commented on Feb 1 Contributor

> However, the method annotated with @ServerRequestFilter (Uni filter(ContainerRequestContext requestContext)) is returning an Uni which is being run interleaved with the Uni being returned by the method annotated with **@get**

This should not be the case... When a `Uni` is being returned, then the request pipeline is suspended and only resumed when the result of the `Uni` comes back.

> **@geoand** / **@FroMage** ? It seems a bit ironical that RestEasy Reactive doesn't support a reactively implemented filter?

Not sure what you mean here. As I explained above, the reactive return type of the filter should result in suspending request (I will make sure there isn't some kind of bug here).

DavideD removed their assignment on Feb 1

Sanne commented on Feb 1 Member

> > However, the method annotated with @ServerRequestFilter (Uni filter(ContainerRequestContext requestContext)) is returning an Uni which is being run interleaved with the Uni being returned by the method annotated with **@get**
>
> This should not be the case... When a `Uni` is being returned, then the request pipeline is suspended and only resumed when the result of the `Uni` comes back.

It's clearly being interleaved in this case.

> > **@geoand** / **@FroMage** ? It seems a bit ironical that RestEasy Reactive doesn't support a reactively implemented filter?
>
> Not sure what you mean here. As I explained above, the reactive return type of the filter should result in suspending request (I will make sure there isn't some kind of bug here).

I'm referring to the points described on:

[quarkus/independent-projects/resteasy-reactive/server/runtime/src/main/java/org/jboss/resteasy/reactive/server/ServerRequestFilter.java]
Lines 53 to 74 in `65f0a3d`

```
     * The return type of the method must be either be of type {@code void}, {@code Respon
     * {@code Optional<Response>}, {@code Optional<RestResponse>},
     * {@code Uni<Void>}, {@code Uni<Response>} or
     * {@code Uni<RestResponse>}.
     * <ul>
     * <li>{@code void} should be used when filtering does not need to perform any blockin
     * processing.
     * <li>{@code Response} or {@code RestResponse} should be used when filtering does not
     * and the filter cannot
     * abort
```

Most likely I'm just confused, but it seems to suggest that Uni responses should be used for blocking operations; it doesn't suggest what one is supposed to do in this case while having a "fully reactive" chain.

geoand commented on Feb 1 Contributor

> It's clearly being interleaved in this case.

I'll take a closer look.

> Most likely I'm just confused, but it seems to suggest that Uni responses should be used for blocking operations; it doesn't suggest what one is supposed to do in this case while having a "fully reactive" chain.

I agree it's not phrased well. I'll fix it.

Sanne commented on Feb 1 Member

thanks **@geoand** !

Sanne added the area/resteasy-reactive label on Feb 1

geoand commented on Feb 1 Contributor

👍

geoand commented on Feb 1 • edited Contributor

I am going to stand by my position that there is nothing wrong in the filter handling in RESTEasy Reactive.

I verified this by taking the reproducer and changing it in the following way:

```
    @ServerRequestFilter
    public Uni<Response> filter(ContainerRequestContext requestContext) {
        return sf.withSession(s -> {
            requestContext.getHeaders().add("test", "true");
            return s.createNativeQuery("SELECT 1")
                    .getSingleResult()
                    .replaceWith(() -> null);
                }

        );
    }
```

Note that inside the `Uni` I am adding a test HTTP header.

Then I made changed the JAX-RS method to:

```
    @GET
    public Uni<List<Fruit>> get(@HeaderParam("test") String testHeader) {
        if (!"true".equals(testHeader)) {
            throw new IllegalStateException("RR is severily misbehaving");
        }
        return sf.withTransaction((s,t) -> {
            return s.createNamedQuery("Fruits.findAll", Fruit.class)
                    .getResultList();
                }

        );
    }
```

Now when I hit this with a lot traffic, I do get

```
2022-02-01 19:40:27,131 ERROR [org.hib.rea.errors] (vert.x-eventloop-thread-44) HR000057: Failed
to execute statement [$1select fruit0_.id as id1_0_, fruit0_.name as name2_0_ from known_fruits
fruit0_ order by fruit0_.name]: $2could not execute query:
java.util.concurrent.CompletionException: java.lang.IllegalStateException: Session/EntityManager
is closed
```

but nowhere do I see `RR is severily misbehaving` that should be thrown if the execution of the filter and resource method is interleaved.

What's more, I did the same experiment taking HR out of the picture entirely, and just used the test HTTP header I mentioned above, through a lot more load at it and it never once failed.

The Javadoc issue mentioned should be fixed in #23347

Sanne commented on Feb 1 Member

I don't think you can test it in this way: the HR issue is only manifesting sporadically and with high load, and it's helped by the fact that the *span* of events being covered is quite long: we do need to send two operations to the RDBMS, and then wait for results for these.

But thanks for the idea, I will play with some variations of that. Worst case I will learn how wrong I am :)

geoand commented on Feb 2 Contributor

I'm very interested in what you find!

Let me know so I can dig into this more if needed

Sanne commented on Feb 2 Member

Complex discussion followed up:

- https://quarkusio.zulipchat.com/#narrow/stream/187038-dev/topic/Vertx.20Local.20context

In short, I believe RestEasy Reactive should isolate the vertx context, but others aren't convinced of this; it might take some time.

In that case **@gwenneg** you won't be able to use context-bound objects, which are implicitly relied on if you use `withSession` or `withTransaction` ; as alternative you'll have to open sessions and transactions explicitly (e.g. `openSession()`), and `close()` them explicitly.

👍 1

Sanne commented on Feb 2 Member

Also: I've adapted the reproducer to verify our requirements without using Hibernate Reactive:

- https://github.com/Sanne/quarkus-resteasy-reactive-vertx-context

It also fails only under load, but it might be a little simpler to understand why.

geoand commented on Feb 2 • edited Contributor

> It also fails only under load, but it might be a little simpler to understand why.

The reproducer is excellent in showing what the thorny issue is.

TL;DR: Hibernate Reactive expects `Vertx.currentContext().getLocal()` and `Vertx.currentContext().putLocal()` to work on a per-request basis. This does seem reasonable, but it appears like Hibernate Reactive was the only system that has this expectation.

geoand mentioned this issue on Feb 3

Use the correct context for RESTEasy Reactive #23397

Merged

Sanne commented on Feb 3 Member

@gwenneg this was fixed by #23397

Sanne closed this as completed on Feb 3

gsmet added this to the **2.7.1.Final** milestone on Feb 3

gwenneg commented on Feb 3 Member Author

Thanks!

👍 1

mklueh commented on Feb 8 Contributor

> Complex discussion followed up:
>
> - https://quarkusio.zulipchat.com/#narrow/stream/187038-dev/topic/Vertx.20Local.20context
>
> In short, I believe RestEasy Reactive should isolate the vertx context, but others aren't convinced of this; it might take some time.
>
> In that case **@gwenneg** you won't be able to use context-bound objects, which are implicitly relied on if you use `withSession` or `withTransaction` ; as alternative you'll have to open sessions and transactions explicitly (e.g. `openSession()`), and `close()` them explicitly.

Is this now not required anymore, as this issue is closed?

DavideD commented on Oct 13 Contributor

> Is this now not required anymore, as this issue is closed?

A bit late but that's correct, it should work as expected now

👍 1

Assignees

No one assigned

Labels

area/hibernate-reactive area/persistence area/resteasy-reactive kind/bug

Projects

None yet

Milestone

2.7.1.Final

Development

Successfully merging a pull request may close this issue.

Use the correct context for RESTEasy Reactive
stuartwdouglas/quarkus

6 participants