Project Loom
Debugger Support

March 24, 2021

Background

- Java is made of Threads
    - Exceptions, Thread locals, Debugger, Profiler, …
- java.lang.Thread
    - Historically one implementation, a thin wrapper around an OS thread
    - Project Loom brings a second low cost implementation “virtual threads”
- No new programming model, no new concepts to learn
- Maintain invariant that Thread.currentThread() does not change in a thread of execution

Virtual threads

“carrier” OS threads managed by a scheduler

Mounted thread

```
Thread vthread = Thread.ofVirtual().start(() -> { .. });
```

Stack trace of virtual thread

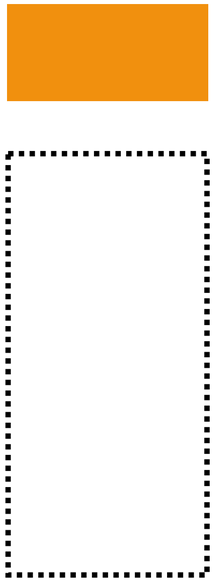
```
:
StackTest.lambda$main$2(StackTest.java:25)
java.base/java.lang.VirtualThread.run(VirtualThread.java:295)
java.base/java.lang.VirtualThread$VThreadContinuation.lambda$new$0(VirtualThread.java:171)
java.base/java.lang.Continuation.enter0(Continuation.java:372)
java.base/java.lang.Continuation.enter(Continuation.java:365)
```

Stack trace of carrier thread

```
java.base/java.lang.Continuation.run(Continuation.java:300)
java.base/java.lang.VirtualThread.runContinuation(VirtualThread.java:224)
java.base/java.util.concurrent.ForkJoinTask$RunnableExecuteAction.exec(ForkJoinTask.java:1395)
java.base/java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:373)
java.base/java.util.concurrent.ForkJoinPool$WorkQueue.topLevelExec(ForkJoinPool.java:1177)
java.base/java.util.concurrent.ForkJoinPool.scan(ForkJoinPool.java:1648)
java.base/java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:1615)
java.base/java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:165)
```

Umounted thread

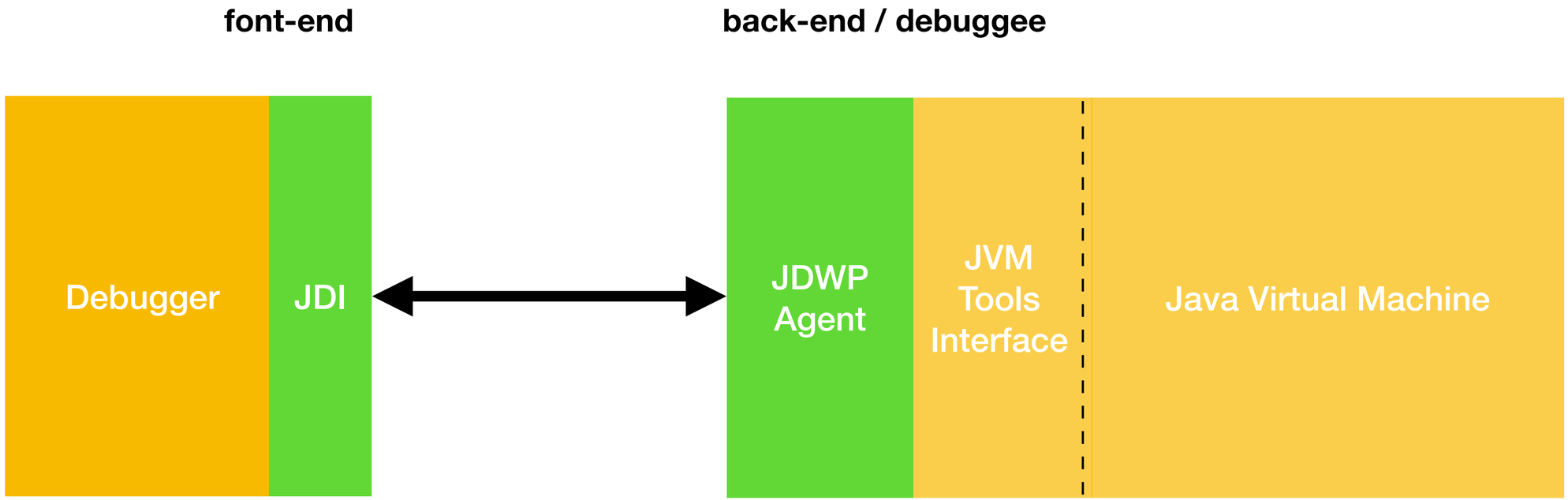
```
BlockingQueue<String> queue = ..

Thread vthread = Thread.ofVirtual().start(() -> {
    :
    String message = queue.take();
    :
});
```



Stack trace of virtual thread

```
java.base/java.lang.Continuation.yield(Continuation.java:402)
java.base/java.lang.VirtualThread.yieldContinuation(VirtualThread.java:367)
java.base/java.lang.VirtualThread.park(VirtualThread.java:531)
java.base/java.lang.System$2.parkVirtualThread(System.java:2346)
java.base/jdk.internal.misc.VirtualThreads.park(VirtualThreads.java:60)
java.base/java.util.concurrent.locks.LockSupport.park(LockSupport.java:369)
java.base/java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionNode.block(AbstractQueuedSynchronizer.java:506)
java.base/java.util.concurrent.ForkJoinPool.unmanagedBlock(ForkJoinPool.java:3469)
java.base/java.util.concurrent.ForkJoinPool.managedBlock(ForkJoinPool.java:3440)
java.base/java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:1623)
java.base/java.util.concurrent.ArrayBlockingQueue.take(ArrayBlockingQueue.java:420)
StackTest.lambda$main$2(StackTest.java:25)
java.base/java.lang.VirtualThread.run(VirtualThread.java:295)
java.base/java.lang.VirtualThread$VThreadContinuation.lambda$new$0(VirtualThread.java:171)
java.base/java.lang.Continuation.enter0(Continuation.java:372)
java.base/java.lang.Continuation.enter(Continuation.java:365)
```

Java Platform Debugger Architecture



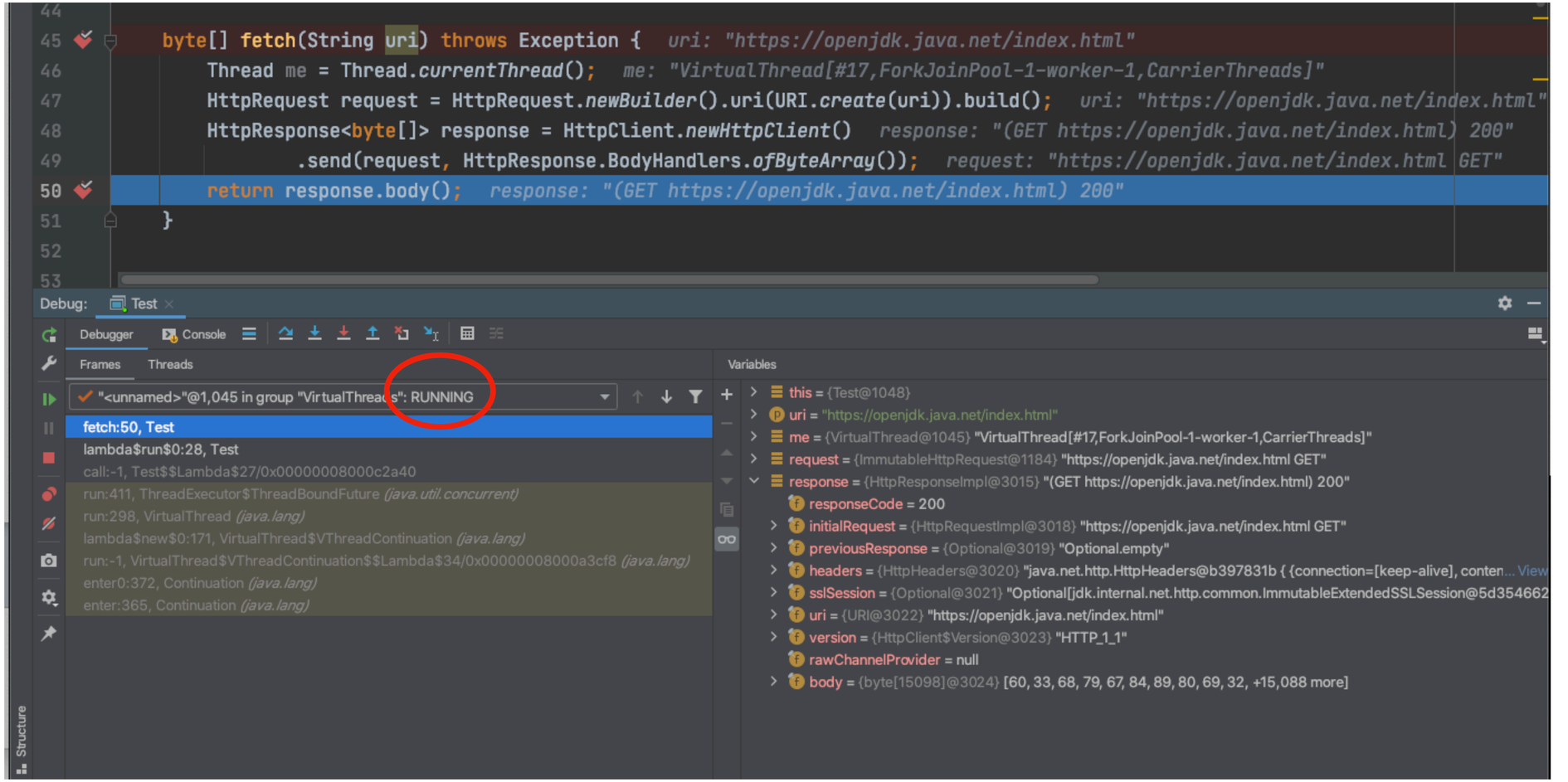
JVM TI

- Most JVM TI functions and events are working with virtual threads
- New JNI function IsVirtualThread
- New can_support_virtual_threads capability, can be added by “virtual thread aware” agents
- New functions SuspendAllVirtualThreads/ResumeAllVirtualThreads, working on new version that supports exclude list
- Virtual thread aware agents get fine control over thread start and thread end events
- GetAllThreads/GetAllStackTraces gets platform threads only
- GetThreadGroupChildren gets the platform threads only (virtual threads are not active members of thread group)
- Not currently working with virtual threads
    - StopThread, InterruptThread
    - SetLocalXXXX
    - PopFrame, ForceEarlyReturnXXX
    - GetObjectMonitorUsage (not used by debugger agent)
    - Get{Current}ThreadCpuTime (not used by debugger agent)

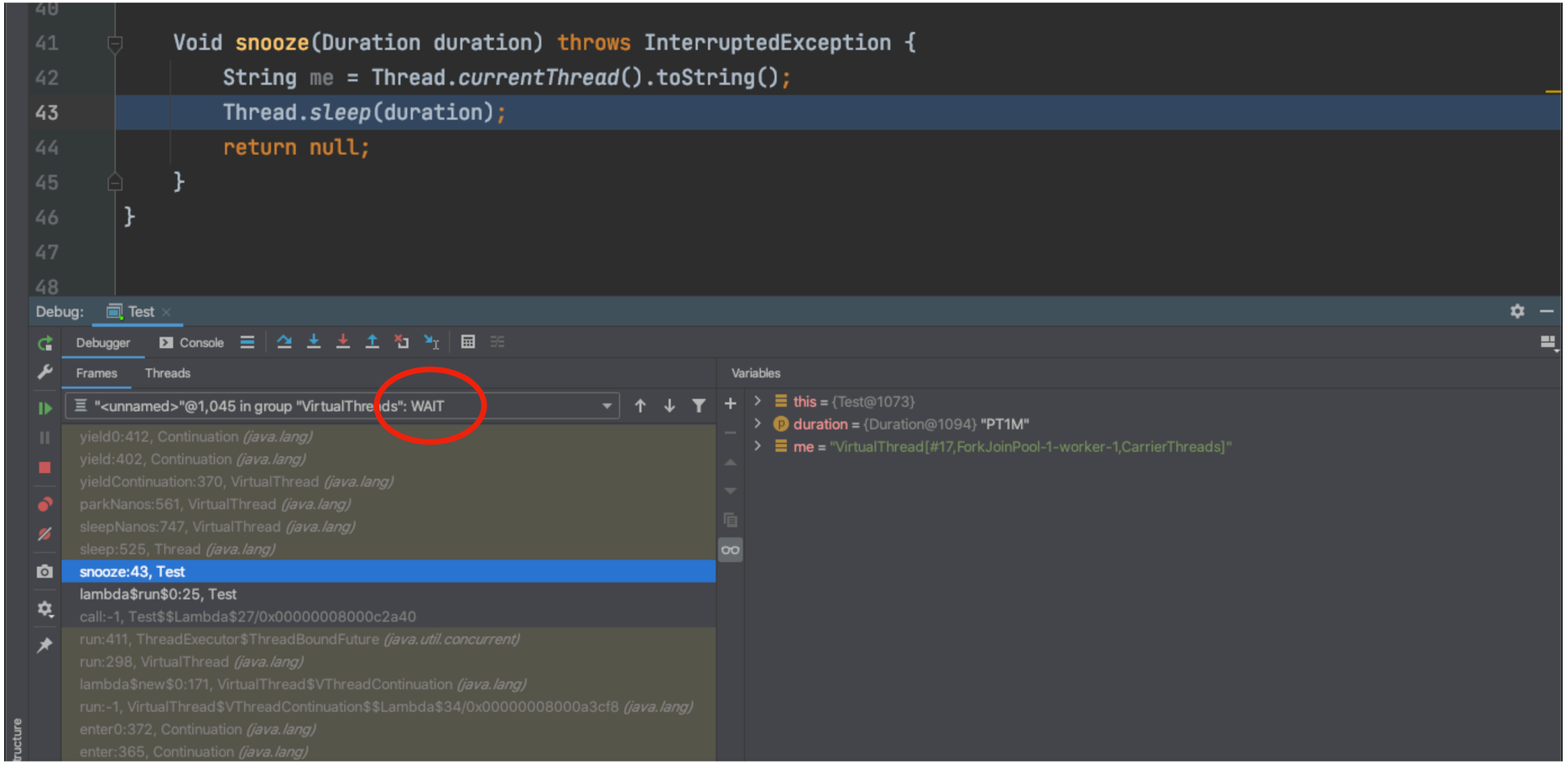
JDWP / JDI

- JDWP
    - With Loom EA builds, assume virtual threads supported if reply to Version command has jdwpMajor >= 17
    - VirtualMachine/AllThreads returns platform threads only
    - New ThreadReference/IsVirtual command
    - New PlatformThreadsOnly filter for EventRequest/Set command when requesting thread start/end events
    - JVM TI issues impact ThreadReference/{Stop,Interrupt,ForceEarlyReturn}, StackFrame/{PopFrames,SetValues}
- JDI
    - With Loom EA builds, assume virtual threads supported if VirtualMachine.version() has major version >= 17
    - VirtualMachine.allThreads() returns platform threads only
    - New method ThreadReference::isVirtual
    - New methods {ThreadStartRequest,ThreadDeathRequest}::addPlatformThreadsOnlyFilter
    - JVM TI issues impact ThreadReference.{stop,interrupt,popFrames,forceEarlyReturn}, StackFrame.setValue

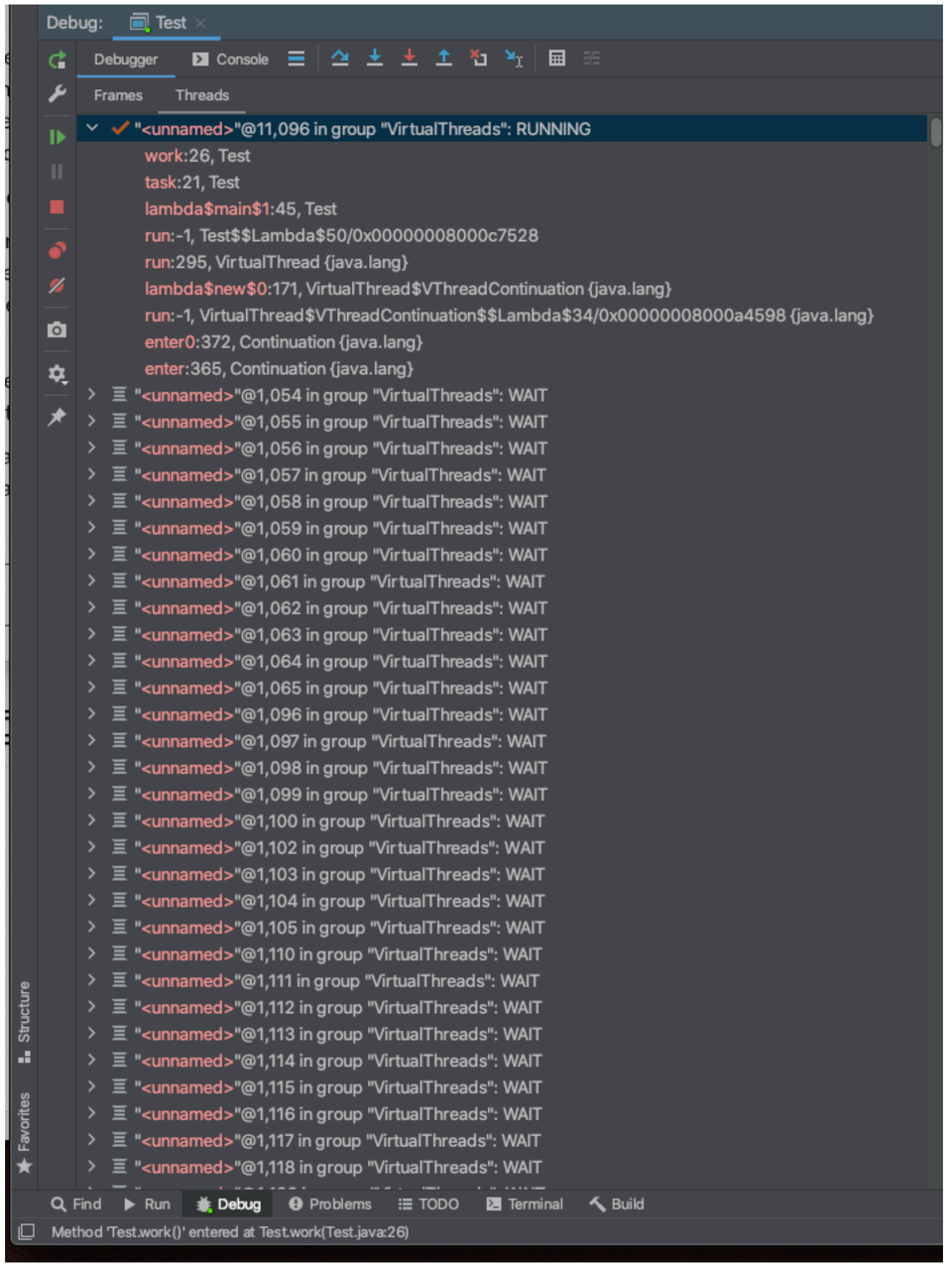
mounted


byte[] fetch(String uri) throws Exception { uri: "https://openjdk.java.net/index.html"
Thread me = Thread.currentThread(); me: "VirtualThread[#17,ForkJoinPool-1-worker-1,CarrierThreads]"
HttpRequest request = HttpRequest.newBuilder().uri(URI.create(uri)).build(); uri: "https://openjdk.java.net/index.html"
HttpResponse<byte[]> response = HttpClient.newHttpClient() response: "(GET https://openjdk.java.net/index.html) 200"
.send(request, HttpResponse.BodyHandlers.ofByteArray()); request: "https://openjdk.java.net/index.html GET"
return response.body(); response: "(GET https://openjdk.java.net/index.html) 200"
}
Debug:
Test
Debugger
Console
Frames
Threads
"<unnamed>"@1,045 in group "VirtualThreads": RUNNING
fetch:50, Test
lambdarun0:28, Test
call:-1, Test$$Lambda$27/0x00000008000c2a40
run:411, ThreadExecutor$ThreadBoundFuture (java.util.concurrent)
run:298, VirtualThread (java.lang)
lambdanew0:171, VirtualThread$VThreadContinuation (java.lang)
run:-1, VirtualThread$VThreadContinuation$$Lambda$34/0x00000008000a3cf8 (java.lang)
enter0:372, Continuation (java.lang)
enter:365, Continuation (java.lang)
Variables
this = {Test@1048}
uri = "https://openjdk.java.net/index.html"
me = {VirtualThread@1045} "VirtualThread[#17,ForkJoinPool-1-worker-1,CarrierThreads]"
request = {ImmutableHttpRequest@1184} "https://openjdk.java.net/index.html GET"
response = {HttpResponseImpl@3015} "(GET https://openjdk.java.net/index.html) 200"
responseCode = 200
initialRequest = {HttpRequestImpl@3018} "https://openjdk.java.net/index.html GET"
previousResponse = {Optional@3019} "Optional.empty"
headers = {HttpHeaders@3020} "java.net.http.HttpHeaders@b397831b { {connection=[keep-alive], conten... View
sslSession = {Optional@3021} "Optional[jdk.internal.net.http.common.ImmutableExtendedSSLSession@5d354662
uri = {URI@3022} "https://openjdk.java.net/index.html"
version = {HttpClient$Version@3023} "HTTP_1_1"
rawChannelProvider = null
body = {byte[15098]@3024} [60, 33, 68, 79, 67, 84, 89, 80, 69, 32, +15,088 more]
Structure

unmounted


41 Void snooze(Duration duration) throws InterruptedException {
42 String me = Thread.currentThread().toString();
43 Thread.sleep(duration);
44 return null;
45 }
46 }
47
48
Debug: Test
Debugger
Console
Frames
Threads
"<unnamed>"@1,045 in group "VirtualThreads": WAIT
yield0:412, Continuation (java.lang)
yield:402, Continuation (java.lang)
yieldContinuation:370, VirtualThread (java.lang)
parkNanos:561, VirtualThread (java.lang)
sleepNanos:747, VirtualThread (java.lang)
sleep:525, Thread (java.lang)
snooze:43, Test
lambdarun0:25, Test
call:-1, Test$$Lambda$27/0x00000008000c2a40
run:411, ThreadExecutor$ThreadBoundFuture (java.util.concurrent)
run:298, VirtualThread (java.lang)
lambdanew0:171, VirtualThread$VThreadContinuation (java.lang)
run:-1, VirtualThread$VThreadContinuation$$Lambda$34/0x00000008000a3cf8 (java.lang)
enter0:372, Continuation (java.lang)
enter:365, Continuation (java.lang)
Variables
this = {Test@1073}
duration = {Duration@1094} "PT1M"
me = "VirtualThread[#17,ForkJoinPool-1-worker-1,CarrierThreads]"

😞


Debug: Test
Debugger
Console
Frames
Threads
"<unnamed>"@11,096 in group "VirtualThreads": RUNNING
work:26, Test
task:21, Test
lambda$main$1:45, Test
run:-1, Test$$Lambda$50/0x00000008000c7528
run:295, VirtualThread {java.lang}
lambdanew0:171, VirtualThread$VThreadContinuation {java.lang}
run:-1, VirtualThread$VThreadContinuation$$Lambda$34/0x00000008000a4598 {java.lang}
enter0:372, Continuation {java.lang}
enter:365, Continuation {java.lang}
"<unnamed>"@1,054 in group "VirtualThreads": WAIT
"<unnamed>"@1,055 in group "VirtualThreads": WAIT
"<unnamed>"@1,056 in group "VirtualThreads": WAIT
"<unnamed>"@1,057 in group "VirtualThreads": WAIT
"<unnamed>"@1,058 in group "VirtualThreads": WAIT
"<unnamed>"@1,059 in group "VirtualThreads": WAIT
"<unnamed>"@1,060 in group "VirtualThreads": WAIT
"<unnamed>"@1,061 in group "VirtualThreads": WAIT
"<unnamed>"@1,062 in group "VirtualThreads": WAIT
"<unnamed>"@1,063 in group "VirtualThreads": WAIT
"<unnamed>"@1,064 in group "VirtualThreads": WAIT
"<unnamed>"@1,065 in group "VirtualThreads": WAIT
"<unnamed>"@1,096 in group "VirtualThreads": WAIT
"<unnamed>"@1,097 in group "VirtualThreads": WAIT
"<unnamed>"@1,098 in group "VirtualThreads": WAIT
"<unnamed>"@1,099 in group "VirtualThreads": WAIT
"<unnamed>"@1,100 in group "VirtualThreads": WAIT
"<unnamed>"@1,102 in group "VirtualThreads": WAIT
"<unnamed>"@1,103 in group "VirtualThreads": WAIT
"<unnamed>"@1,104 in group "VirtualThreads": WAIT
"<unnamed>"@1,105 in group "VirtualThreads": WAIT
"<unnamed>"@1,110 in group "VirtualThreads": WAIT
"<unnamed>"@1,111 in group "VirtualThreads": WAIT
"<unnamed>"@1,112 in group "VirtualThreads": WAIT
"<unnamed>"@1,113 in group "VirtualThreads": WAIT
"<unnamed>"@1,114 in group "VirtualThreads": WAIT
"<unnamed>"@1,115 in group "VirtualThreads": WAIT
"<unnamed>"@1,116 in group "VirtualThreads": WAIT
"<unnamed>"@1,117 in group "VirtualThreads": WAIT
"<unnamed>"@1,118 in group "VirtualThreads": WAIT
Structure
Favorites
Find
Run
Debug
Problems
TODO
Terminal
Build
Method 'Test.work()' entered at Test.work(Test.java:26)

Current status

- Current focus: Stability, Performance, API, Libraries, Diagnosability, …
- Debugger support
    - Remaining issues with suspend/resume support
    - Virtual thread “unaware” debuggers
        - JDWP options (enumeratevthreads, trackvthreads, ..)
    - Performance
    - A small number of missing JVM TI features
- Working towards “JEP: Virtual threads (Preview)”

Discussion topics

- Project Loom needs first class debugger support for virtual threads
- How will the debugger work if there are a lot of threads?
    - A drop down list of threads does not scale
- Will the debugger work when attaching to an existing target VM?
    - Assume no API support for finding all virtual threads
- Future work, structured serviceability and observability
    - Threads dump support already has early support for this

Links

- Early access builds: http://jdk.java.net/loom/
- Mailing list: loom-dev@openjdk.java.net
- JVMTI: https://download.java.net/java/early_access/loom/docs/specs/jvmti.html
- JDWP: https://download.java.net/java/early_access/loom/docs/specs/jdwp/jdwp-protocol.html
- JDI: https://download.java.net/java/early_access/loom/docs/api/jdk.jdi/module-summary.html