Module jdk.incubator.concurrent
Package jdk.incubator.concurrent

Class StructuredTaskScope.ShutdownOnFailure

java.lang.Object
  jdk.incubator.concurrent.StructuredTaskScope<Object>
    jdk.incubator.concurrent.StructuredTaskScope.ShutdownOnFailure

All Implemented Interfaces:

AutoCloseable

Enclosing class:

StructuredTaskScope<T>

```
public static final class StructuredTaskScope.ShutdownOnFailure
extends StructuredTaskScope<Object>
```

A StructuredTaskScope that captures the exception of the first subtask to complete abnormally. Once captured, it invokes the shutdown method to interrupt unfinished threads and wakeup the owner. The policy implemented by this class is intended for cases where the results for all subtasks are required ("invoke all"); if any subtask fails then the results of other unfinished subtasks are no longer needed.

Unless otherwise specified, passing a null argument to a method in this class will cause a NullPointerException to be thrown.

Since:
19

Nested Class Summary

Nested classes/interfaces declared in class jdk.incubator.concurrent.StructuredTaskScope

StructuredTaskScope.ShutdownOnFailure, StructuredTaskScope.ShutdownOnSuccess<T>

Constructor Summary

Constructors

Constructor	Description
ShutdownOnFailure()	Constructs a new unnamed ShutdownOnFailure that creates virtual threads.
ShutdownOnFailure(String name, ThreadFactory factory)	Constructs a new ShutdownOnFailure with the given name and thread factory.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
Optional<Throwable>	exception()	Returns the exception for the first subtask that completed with an exception.
protected void	handleComplete (Future<Object> future)	Shut down the given task scope when invoked for the first time with a Future for a task that completed abnormally (exception or cancelled).
StructuredTaskScope.ShutdownOnFa	join()	Wait for all threads to finish or the task scope to shut down.
StructuredTaskScope.ShutdownOnFa	joinUntil(Instant deadline)	Wait for all threads to finish or the task scope to shut down, up to the given deadline.
void	throwIfFailed()	Throws if a subtask completed abnormally.
<X extends Throwable> void	throwIfFailed(Function<Throwable,? extends X> esf)	Throws the exception produced by the given exception supplying function if a subtask completed abnormally.

Methods declared in class jdk.incubator.concurrent.StructuredTaskScope

close, fork, shutdown

Methods declared in class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

ShutdownOnFailure

```
public ShutdownOnFailure(String name,
                         ThreadFactory factory)
```

Constructs a new `ShutdownOnFailure` with the given name and thread factory. The task scope is optionally named for the purposes of monitoring and management. The thread factory is used to `create` threads when tasks are forked. The task scope is owned by the current thread.

Parameters:

`name` - the name of the task scope, can be null

`factory` - the thread factory

ShutdownOnFailure

```
public ShutdownOnFailure()
```

Constructs a new unnamed `ShutdownOnFailure` that creates virtual threads.

This constructor is equivalent to invoking the 2-arg constructor with a name of `null` and a thread factory that creates virtual threads.

Method Details

handleComplete

```
protected void handleComplete(Future<Object> future)
```

Shut down the given task scope when invoked for the first time with a `Future` for a task that completed abnormally (exception or cancelled).

Overrides:

`handleComplete` in class `StructuredTaskScope<Object>`

Parameters:

`future` - the completed task

See Also:

`StructuredTaskScope.shutdown()`, `Future.State.FAILED`, `Future.State.CANCELLED`

join

```
public StructuredTaskScope.ShutdownOnFailure join()
                                           throws InterruptedException
```

Wait for all threads to finish or the task scope to shut down. This method waits until all threads started in the task scope finish execution (of both task and `handleComplete` method), the `shutdown` method is invoked to shut down the task scope, or the current thread is interrupted.

This method may only be invoked by the task scope owner.

Overrides:

`join` in class `StructuredTaskScope<Object>`

Returns:

this task scope

Throws:

`IllegalStateException` - if this task scope is closed

`WrongThreadException` - if the current thread is not the owner

`InterruptedException` - if interrupted while waiting

joinUntil

```
public StructuredTaskScope.ShutdownOnFailure joinUntil(Instant deadline)
                                                throws InterruptedException,
                                                       TimeoutException
```

Wait for all threads to finish or the task scope to shut down, up to the given deadline. This method waits until all threads started in the task scope finish execution (of both task and `handleComplete` method), the `shutdown` method is invoked to shut down the task scope, the current thread is interrupted, or the deadline is reached.

This method may only be invoked by the task scope owner.

Overrides:

`joinUntil` in class `StructuredTaskScope<Object>`

Parameters:

`deadline` - the deadline

Returns:

this task scope

Throws:

`IllegalStateException` - if this task scope is closed

`WrongThreadException` - if the current thread is not the owner

`InterruptedException` - if interrupted while waiting

`TimeoutException` - if the deadline is reached while waiting

exception

```
public Optional<Throwable> exception()
```

Returns the exception for the first subtask that completed with an exception. If no subtask completed with an exception but cancelled subtasks were notified to the `handleComplete` method then a `CancellationException` is returned. If no subtasks completed abnormally then an empty `Optional` is returned.

API Note:

This method is intended to be invoked by the task scope owner after it has invoked `join` (or `joinUntil`). A future release may add enforcement to prevent the method being called by other threads or before joining.

Returns:

the exception for a subtask that completed abnormally or an empty optional if no subtasks completed abnormally

throwIfFailed

```
public void throwIfFailed()
                  throws ExecutionException
```

Throws if a subtask completed abnormally. If any subtask completed with an exception then `ExecutionException` is thrown with the exception of the first subtask to fail as the cause. If no subtask completed with an exception but cancelled subtasks were notified to the `handleComplete` method then `CancellationException` is thrown. This method does nothing if no subtasks completed abnormally.

API Note:

This method is intended to be invoked by the task scope owner after it has invoked `join` (or `joinUntil`). A future release may add enforcement to prevent the method being called by other threads or before joining.

Throws:

`ExecutionException` - if a subtask completed with an exception

`CancellationException` - if no subtasks completed with an exception but subtasks were cancelled

throwIfFailed

```
public <X extends Throwable> void throwIfFailed(Function<Throwable,? extends X> esf)
                                        throws X
```

Throws the exception produced by the given exception supplying function if a subtask completed abnormally. If any subtask completed with an exception then the function is invoked with the exception of the first subtask to fail. If no subtask completed with an exception but cancelled subtasks were notified to the `handleComplete` method then the function is called with a `CancellationException`. The exception returned by the function is thrown. This method does nothing if no subtasks completed abnormally.

API Note:

This method is intended to be invoked by the task scope owner after it has invoked `join` (or `joinUntil`). A future release may add enforcement to prevent the method being called by other threads or before joining.

Type Parameters:

X - type of the exception to be thrown

Parameters:

`esf` - the exception supplying function

Throws:

X - produced by the exception supplying function

Report a bug or suggest an enhancement
For further API reference and developer documentation see the Java SE Documentation, which contains more detailed, developer-targeted descriptions with conceptual overviews, definitions of terms, workarounds, and working code examples. Other versions.
Java is a trademark or registered trademark of Oracle and/or its affiliates in the US and other countries.
Copyright © 1993, 2022, Oracle and/or its affiliates, 500 Oracle Parkway, Redwood Shores, CA 94065 USA.
All rights reserved. Use is subject to license terms and the documentation redistribution policy.