Module java.base
Package java.util.concurrent

Class StructuredTaskScope.ShutdownOnFailure

java.lang.Object
  java.util.concurrent.StructuredTaskScope[PREVIEW]<Object>
    java.util.concurrent.StructuredTaskScope.ShutdownOnFailure

All Implemented Interfaces:

AutoCloseable

Enclosing class:

StructuredTaskScope[PREVIEW]<T>

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

> **`ShutdownOnFailure` is a preview API of the Java platform.**
> *Programs can only use `ShutdownOnFailure` when preview features are enabled.*
> *Preview features may be removed in a future release, or upgraded to permanent features of the Java platform.*

A `StructuredTaskScope` that captures the exception of the first subtask to fail[PREVIEW]. Once captured, it shuts down[PREVIEW] the task scope to interrupt unfinished threads and wakeup the task scope 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.

API Note:

This class implements a policy to shut down the task scope when a subtask fails. There shouldn't be any need to directly shut down the task scope with the `shutdown`[PREVIEW] method.

Since:

21

Nested Class Summary

Nested classes/interfaces declared in class java.util.concurrent.StructuredTaskScope[PREVIEW]

StructuredTaskScope.ShutdownOnFailure[PREVIEW], StructuredTaskScope.ShutdownOnSuccess[PREVIEW]<T>, StructuredTaskScope.Subtask[PREVIEW]<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 of the first subtask that failed[PREVIEW].
StructuredTaskScope.ShutdownOnFail	join()	Wait for all subtasks started in this task scope to complete or for a subtask to fail[PREVIEW].

StructuredTaskScope.ShutdownOnFail	joinUntil(Instant deadline)	Wait for all subtasks started in this task scope to complete or for a subtask to fail[PREVIEW], up to the given deadline.
void	throwIfFailed()	Throws if a subtask failed[PREVIEW].
<X extends Throwable> void	throwIfFailed(Function<Throwable,? extends X> esf)	Throws the exception produced by the given exception supplying function if a subtask failed[PREVIEW].

Methods declared in class java.util.concurrent.StructuredTaskScope[PREVIEW]

close, ensureOwnerAndJoined, fork, handleComplete, isShutdown, 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 subtasks are forked[PREVIEW]. The task scope is owned by the current thread.

Construction captures the current thread's scoped value[PREVIEW] bindings for inheritance by threads started in the task scope. The Tree Structure section in the class description details how parent-child relations are established implicitly for the purpose of inheritance of scoped value bindings.

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.

Implementation Requirements:

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

join

```
public StructuredTaskScope.ShutdownOnFailure[PREVIEW] join()
                                             throws InterruptedException
```

Wait for all subtasks started in this task scope to complete or for a subtask to fail[PREVIEW].

This method waits for all subtasks by waiting for all threads started[PREVIEW] in this task scope to finish execution. It stops waiting when all threads finish, a subtask fails, or the current thread is interrupted. It also stops waiting if the `shutdown`[PREVIEW] method is invoked directly to shut down this task scope.

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

Overrides:

`join` in class `StructuredTaskScope`[PREVIEW]`<Object>`

Returns:

this task scope

Throws:

`IllegalStateException` - if this task scope is closed

WrongThreadException - if the current thread is not the task scope owner

InterruptedException - if interrupted while waiting

joinUntil

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

Wait for all subtasks started in this task scope to complete or for a subtask to fail[PREVIEW], up to the given deadline.

This method waits for all subtasks by waiting for all threads started[PREVIEW] in this task scope to finish execution. It stops waiting when all threads finish, a subtask fails, the deadline is reached, or the current thread is interrupted. It also stops waiting if the shutdown[PREVIEW] method is invoked directly to shut down this task scope.

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

Overrides:

joinUntil in class StructuredTaskScope[PREVIEW]<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 task scope owner

InterruptedException - if interrupted while waiting

TimeoutException - if the deadline is reached while waiting

exception

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

Returns the exception of the first subtask that failed[PREVIEW]. If no subtasks failed then an empty `Optional` is returned.

Returns:

the exception for the first subtask to fail or an empty optional if no subtasks failed

Throws:

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

`IllegalStateException` - if the task scope owner did not join after forking

throwIfFailed

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

Throws if a subtask failed[PREVIEW]. If any subtask failed with an exception then `ExecutionException` is thrown with the exception of the first subtask to fail as the cause. This method does nothing if no subtasks failed.

Throws:

`ExecutionException` - if a subtask failed

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

`IllegalStateException` - if the task scope owner did not join after forking

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 failed[PREVIEW]. If any subtask failed with an exception then the function is invoked with the exception of the first subtask to fail. The exception returned by the function is thrown. This method does nothing if no subtasks failed.

Type Parameters:

X - type of the exception to be thrown

Parameters:

`esf` - the exception supplying function

Throws:

X - produced by the exception supplying function

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

`IllegalStateException` - if the task scope owner did not join after forking
