Module java.base
Package java.util.concurrent

Class StructuredTaskScope.ShutdownOnSuccess<T>

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

Type Parameters:

T - the result type

All Implemented Interfaces:

AutoCloseable

Enclosing class:

StructuredTaskScope[PREVIEW]<T>

```
public static final class StructuredTaskScope.ShutdownOnSuccess<T>
extends StructuredTaskScope[PREVIEW]<T>
```

> **ShutdownOnSuccess is a preview API of the Java platform.**
> *Programs can only use ShutdownOnSuccess 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 result of the first subtask to complete successfully[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 result of any subtask will do ("invoke any") and where 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 completes successfully. 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
`ShutdownOnSuccess()`	Constructs a new unnamed `ShutdownOnSuccess` that creates virtual threads.
`ShutdownOnSuccess(String name, ThreadFactory factory)`	Constructs a new `ShutdownOnSuccess` with the given name and thread factory.

Method Summary

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

Modifier and Type	Method	Description
`StructuredTaskScope.ShutdownOnSucc <T>`	`join()`	Wait for a subtask started in this task scope to complete successfully[PREVIEW] or all subtasks to complete.

StructuredTaskScope.ShutdownOnSucc<T>	joinUntil(Instant deadline)	Wait for a subtask started in this task scope to complete successfully[PREVIEW] or all subtasks to complete, up to the given deadline.
T	result()	Returns the result of the first subtask that completed successfully[PREVIEW].
<X extends Throwable> T	result(Function<Throwable,? extends X> esf)	Returns the result of the first subtask that completed successfully[PREVIEW], otherwise throws an exception produced by the given exception supplying function.

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

ShutdownOnSuccess

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

Constructs a new `ShutdownOnSuccess` 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

ShutdownOnSuccess

```
public ShutdownOnSuccess()
```

Constructs a new unnamed `ShutdownOnSuccess` 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.ShutdownOnSuccess[PREVIEW]<T> join()
                                                    throws InterruptedException
```

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

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 completes successfully, 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]<T>

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.ShutdownOnSuccess[PREVIEW]<T> joinUntil(Instant deadline)
                                                    throws InterruptedException,
                                                           TimeoutException
```

Wait for a subtask started in this task scope to complete successfully[PREVIEW] or all subtasks to complete, 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 completes successfully, 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]<T>

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

result

```
public T result()
         throws ExecutionException
```

Returns the result of the first subtask that completed successfully[PREVIEW].

When no subtask completed successfully, but a subtask failed[PREVIEW] then `ExecutionException` is thrown with the subtask's exception as the cause.

Returns:

the result of the first subtask that completed successfully[PREVIEW]

Throws:

`ExecutionException` - if no subtasks completed successfully but at least one subtask failed

`IllegalStateException` - if no subtasks completed or the task scope owner did not join after forking

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

result

```
public <X extends Throwable> T result(Function<Throwable,? extends X> esf)
                              throws X
```

Returns the result of the first subtask that completed successfully[PREVIEW], otherwise throws an exception produced by the given exception supplying function.

When no subtask completed successfully, but a subtask failed[PREVIEW], then the exception supplying function is invoked with subtask's exception.

Type Parameters:

`X` - type of the exception to be thrown

Parameters:

`esf` - the exception supplying function

Returns:

the result of the first subtask that completed with a result

Throws:

X - if no subtasks completed successfully but at least one subtask failed

`IllegalStateException` - if no subtasks completed or the task scope owner did not join after forking

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

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, 2024, 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. Modify Cookie Preferences. Modify Ad Choices.