Implementing a Custom StructuredTaskScope

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Understand Java’s structured concurrency model
- Recognize the classes used to program Java’s structure concurrency model
- Evaluate the design & performance of various Java concurrency models
- Know why & how to implement a custom StructuredTaskScope

```java
class ShutdownOnNonNullSuccess<T> extends StructuredTaskScope<T> {
    private volatile T mResult;
    protected void handleComplete(Future<T> future) {
        T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }

    public T result()
    { return mResult; }
}
```
Motivating the Need for a Custom Structured Task Scope
Motivating the Need for a Custom StructuredTaskScope

- ShutdownOnSuccess works if the first task to complete provides a valid value.

```java
try (var scope = new StructuredTaskScope
    .ShutdownOnSuccess
    <List<BigFraction>>()
    ) {
    scope.fork(() -> quickSort(list));
    scope.fork(() -> heapSort(list));
    scope.join();
    return scope.result();
}
```

The result of either quicksort() or heapsort() is valid.

See [howtodoinjava.com/java/multi-threading/structured-concurrency](https://howtodoinjava.com/java/multi-threading/structured-concurrency)
Motivating the Need for a Custom StructuredTaskScope

- However, it does not work if the first task to complete provides an invalid or “missing” value

```java
try (var scope = new StructuredTaskScope.ShutdownOnSuccess<Integer>()) {
    numbers
        .forEach(number -> scope.fork(() -> isPrime(number) == 0
                ? number
                : null));

    scope.join();

    return scope.result();
}
```

Any “null” results should not be considered valid

See howtodoinjava.com/java/multi-threading/structured-concurrency
It’s therefore necessary to create a custom Structured TaskScope subclass.

StructuredTaskScope is a framework that can be customized via inheritance.

See jdk/incubator/concurrent/StructuredTaskScope.html
Programming a Custom StructuredTaskScope
Programming with Java ShutdownOnSuccess

- **ShutdownOnNonNullSuccess** extends **StructuredTaskScope**

```java
public class ShutdownOnNonNullSuccess<T> extends StructuredTaskScope<T> {
    private volatile T mResult;
    protected void handleComplete(Future<T> future) {
        ...
        T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }

    public T result() {
        return mResult;
    }
    ...
}
```

*Captures the result of the first subtask to complete successfully (i.e., without returning a null) or returns null if no subtask completes successfully.*

See [Loom/ex5/src/main/java/utils/ShutdownOnNonNullSuccess.java](Loom/ex5/src/main/java/utils/ShutdownOnNonNullSuccess.java)
public class ShutdownOnNonNullSuccess<T> extends StructuredTaskScope<T> {
    private volatile T mResult;
    protected void handleComplete(Future<T> future) {
        ... 
        T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }
    public T result() {
        return mResult;
    }
} ...
Programming with Java ShutdownOnSuccess

- ShutdownOnNonNullSuccess extends StructuredTaskScope

```java
class ShutdownOnNonNullSuccess<T> extends StructuredTaskScope<T> {
    private volatile T mResult;
    protected void handleComplete(Future<T> future) {
        ... T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }

    public T result() {
        return mResult;
    }
    ...
}
```

See [jdk/incubator/concurrent/StructuredTaskScope.html#handleComplete](https://jdk/incubator/concurrent/StructuredTaskScope.html#handleComplete)
public class ShutdownOnNonNullSuccess\<T\> extends StructuredTaskScope\<T\> {
    private volatile T mResult;
    protected void handleComplete (Future\<T\> future) {
        ... 
        T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }
    
    public T result() {
        return mResult; }
    ... 
}

This parameter contains the result of the completed task

See jdk20/docs/api/java.base/java/util/concurrent/Future.html
• ShutdownOnNonNullSuccess extends StructuredTaskScope

```java
class ShutdownOnNonNullSuccess<T>
    extends StructuredTaskScope<T> {
    private volatile T mResult;
    protected void handleComplete(
        Future<T> future) {
        ... 
        T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }

    public T result()
    { return mResult; } ...
}
```

If the result is non-null then store it & shutdown the scope

See jdk.incubator.concurrent/jdk/incubator/concurrent/StructuredTaskScope.html#shutdown
Programming with Java ShutdownOnSuccess

- ShutdownOnNonNullSuccess extends StructuredTaskScope

```java
public class ShutdownOnNonNullSuccess<T> extends StructuredTaskScope<T> {
    private volatile T mResult;
    protected void handleComplete(Future<T> future) {
        ...
        T result = future.resultNow();
        if (result != null) {
            mResult = result; shutdown();
        }
    }

    public T result() {
        return mResult;
    }
    ...
}
```

Allow a caller to get the result after scope.join() returns
End of Implementing a Custom StructuredTaskScope