Dataflow

Russell Winder

Version 1.0, 2015-10-01

Table of Contents

Concepts

Dataflow Concurrency

Dataflow Concurrency offers an alternative concurrency model, which is inherently safe and robust. It puts an emphasis on the data and their flow though your processes instead of the actual processes that manipulate the data. **Dataflow** algorithms relieve developers from dealing with live-locks, race-conditions and make dead-locks deterministic and thus 100% reproducible. If you don't get dead-locks in tests you won't get them in production.

Dataflow Variable

A single-assignment multi-read variable offering a thread-safe data-exchange among threads.

Dataflows Class

A virtual infinite map of **Dataflow Variables** with on-demand creation policy.

Dataflow Stream

A thread-safe unbound deterministic blocking stream with a **Dataflow Variable**-compatible interface.

Dataflow Queue

A thread-safe unbound blocking queue with a **Dataflow Variable**-compatible interface.

Dataflow Task

A lightweight thread of execution, which gets assigned a physical thread from a thread pool to execute the body of the task. Tasks should typically exchange data using **Dataflow Variables* and *Streams**.

Dataflow Operator

A cornerstone of a more thorough *dataflow concurrency algorithm*. Such algorithms typically define a number of operators and connect them with channels, represented by *Dataflow Streams, Queues* or *Variables*.

Each operator specifies its input and output channels to communicate with other operators. Repeatedly, whenever all input channels of a particular operator contain data, the operator's body is executed and the produced output is sent into the output channels.

Usage

Dataflow Variables

A Sample

```
import static groovyx.gpars.dataflow.Dataflow.task

final def x = new DataflowVariable()
final def y = new DataflowVariable()
final def z = new DataflowVariable()
task{
    z << x.val + y.val
    println "Result: ${z.val}"
}

task{
    x << 10
}

task{
    y << 5
}
```

Dataflows

A Sample

```
import static groovyx.gpars.dataflow.Dataflow.task
final def df = new Dataflows()

task{
    df.z = df.x + df.y
    println "Result: ${df.z}"
}

task{
    df.x = 10
}

task{
    df.y = 5
}
```

Dataflow Queues

A Sample

```
import static groovyx.gpars.dataflow.Dataflow.task

def words = ['Groovy', 'fantastic', 'concurrency', 'fun', 'enjoy', 'safe', 'GPars', '
data', 'flow']
final def buffer = new DataflowQueue()

task{
    for (word in words) {
        buffer << word.toUpperCase()  // Add to the buffer.
    }
}

task{
    while(true) println buffer.val  // Read from the buffer in a loop.
}
```

Bind Handlers

A Sample

```
def a = new DataflowVariable()
a >> {println "The variable has just been bound to $it"}

a.whenBound{println "Just to confirm that the variable has been really set to $it"}
```

Dataflow Operators

A Sample

```
operator(inputs: [a, b, c], outputs: [d]) {x, y, z ->
    ...
    bindOutput 0, x + y + z
}
```