## CONTENTS

1 Basic Abstractions
   1.1 Executors ......................................................... 3
   1.2 Queues ............................................................. 3
   1.3 Work ............................................................... 3

2 Library Reference
   2.1 Executors ......................................................... 5
   2.2 Queues ............................................................. 6
   2.3 Work ............................................................... 9

API Index ............................................................. 11

Index ................................................................. 13
This library provides various concurrency utilities for use with Dylan programs.
The abstractions in this library are somewhat inspired by javax.concurrent.

1.1 Executors

Executors perform work that is requested from them asynchronously. Currently, all executors use their own private threads.

See: <executor>, <fixed-thread-executor>, <thread-executor>, and <single-thread-executor>.

1.2 Queues

Queues are job-streams that can have items enqueued and subsequently dequeued. These form the synchronization mechanism for thread executors.

See: <queue>, <locked-queue>.

1.3 Work

Work objects represent something to be done.

See: <work>, <locked-work>. 
2.1 Executors

<executor> Abstract Class
Superclasses <object>
Init-Keywords
  • name –
Operations
  • executor-name
  • executor-request

<thread-executor> Abstract Class
Superclasses <executor>
Init-Keywords
  • queue –
Operations
  • executor-shutdown

<fixed-thread-executor> Class
Superclasses <thread-executor>
Init-Keywords
  • thread-count –

<single-thread-executor> Class
Superclasses <thread-executor>

executor-name Generic function
Signature executor-name (executor) => (name)
Parameters
  • executor – An instance of <executor>.
Values
  • name – An instance of <string>.
executor-request Generic function
Request that this executor do some work.

Signature executor-request (executor work) => ()

Parameters
• executor – An instance of <executor>.
• work – An instance of <object>.

evaluator-request(<function>) Method
A convenience method that converts the given function into a <work> object. The function must not have any required arguments.

Signature executor-request (executor function) => ()

Parameters
• executor – An instance of <executor>.
• work – An instance of <function>.

evaluator-request(<work>) Method
Signature executor-request (executor work) => ()

Parameters
• executor – An instance of <executor>.
• work – An instance of <work>.

evaluator-shutdown Generic function

Signature evaluator-shutdown (executor #key join? drain?) => ()

Parameters
• executor – An instance of <thread-executor>.
• join? (#key) – An instance of <boolean>.
• drain? (#key) – An instance of <boolean>.

2.2 Queues
<queue> Abstract Class

Superclasses <object>
Init-Keywords
• name –

Discussion This is a base class for specific implementations that modify queueing behaviour.

Operations
• dequeue
• enqueue
• queue-name

<locked-queue> Class
Locked multi-reader multi-writer queue
Superclasses  <queue>

Discussion

Locked multi-reader multi-writer queue

A notification is used for synchronization. The associated lock is used for all queue state.

Locked queues can be  STOPPED  so that no further work will be accepted and processing will end once all previously submitted work has been finished.

After stopping, all further enqueue operations will signal  <queue-stopped>.

Dequeue operations will continue until the queue has been drained, whereupon they will also be signalled.

Locked queues can be  INTERRUPTED  so that no further work will be accepted or begun. Work that has already been started will continue.

Interrupting implies stopping, so enqueue operations will be signalled  <queue-stopped>. Dequeue operations will signal  <queue-interrupt>.

Operations

•  interrupt-queue
•  stop-queue

dequeue  Generic function
Dequeue the next available item from the queue.

Signature  dequeue (queue) => (object)

Parameters

•  queue – An instance of  <queue>.

Values

•  object – An instance of  <object>.

Discussion

Dequeue the next available item from the queue.

May signal  <queue-interrupt>  or  <queue-stopped>  when the queue has reached the respective state.

enqueue  Generic function
Enqueue a work item onto the queue.

Signature  enqueue (queue object) => ()

Parameters

•  queue – An instance of  <queue>.
•  object – An instance of  <object>.

Discussion

Enqueue a work item onto the queue.

May signal  <queue-stopped>  when the queue no longer accepts work.

queue-name  Generic function
Returns the name of the queue.
Signature queue-name (queue) => (name?)

Parameters

• queue – An instance of <queue>.

Values

• name? – An instance of false-or(<string>).

**interrupt-queue** Generic function
Interrupts the queue, abandoning submitted work.

Signature interrupt-queue (queue) => ()

Parameters

• queue – An instance of <locked-queue>.

Discussion
Interrupts the queue, abandoning submitted work.

Submitters will be signalled <queue-stopped> in enqueue if they try to submit further work.

Receivers will be signalled <queue-interrupt> at the first dequeue operation they perform.

**stop-queue** Generic function
Stops the queue so that submitted work can still continue.

Signature stop-queue (queue) => ()

Parameters

• queue – An instance of <locked-queue>.

Discussion
Stops the queue so that submitted work can still continue.

Submitters will be signalled <queue-stopped> in enqueue if they try to submit further work.

Receivers will be signalled <queue-stopped> in dequeue once the queue has been drained.

**<queue-condition>** Abstract Class
Conditions related to <locked-queue> operations.

Superclasses <condition>

Init-Keywords

• queue –

• thread –

**<queue-interrupt>** Class
Signalled when the queue has been interrupted.

Superclasses <queue-condition>

**<queue-stopped>** Class
Signalled when the queue has been stopped.

Superclasses <queue-condition>
queue-condition-queue Generic function

Signature queue-condition-queue (condition) => (queue)
Parameters
  • condition – An instance of <queue-condition>.
Values
  • queue – An instance of <queue>.

queue-condition-thread Generic function

Signature queue-condition-thread (condition) => (thread)
Parameters
  • condition – An instance of <queue-condition>.
Values
  • thread – An instance of <thread>.

2.3 Work

<work> Class

Superclasses <object>
Init-Keywords
  • function – A function to perform some work. The function must not have any required arguments.
Operations
  • work-finished?
  • work-perform
  • work-started?
  • work-thread

<locked-work> Class

Superclasses <work>
Operations
  • work-wait

work-finished? Generic function

Signature work-finished? (work) => (finished?)
Parameters
  • work – An instance of <work>.
Values
  • finished? – An instance of <boolean>.
work-perform Generic function
Signature  
work-perform (work) => ()
Parameters
• work – An instance of <work>.

work-started? Generic function
Signature  
work-started? (work) => (started?)
Parameters
• work – An instance of <work>.
Values
• started? – An instance of <boolean>.

work-thread Generic function
Return the thread on which the work was executed.
Signature  
work-thread (work) => (thread)
Parameters
• work – An instance of <work>.
Values
• thread – An instance of <thread>.

work-wait Generic function
Wait for a work item to reach the given state. Valid states are $work-started and $work-finished.
Signature  
work-wait (work state) => ()
Parameters
• work – An instance of <locked-work>.
• state – An instance of <work-state>. One of $work-started or $work-finished.

$work-started Constant
Used with work-wait to indicate that you want to wait until work has started executing.
Type <work-state>
See also: $work-finished

$work-finished Constant
Used with work-wait to indicate that you want to wait until work has finished executing.
Type <work-state>
See also: $work-finished
API INDEX

D
dequeue (generic function), 7

E
<executor> (class), 5
enqueue (generic function), 7
extor-name (generic function), 5
concurrency:concurrency:executor-request (generic function), 5
concurrency:concurrency:executor-request(<function>)
(method), 6
concurrency:concurrency:executor-request(<work>)
(method), 6
executor-shutdown (generic function), 6

F
<fixed-thread-executor> (class), 5

I
interrupt-queue (generic function), 8

L
<locked-queue> (class), 6
<locked-work> (class), 9

Q
<queue-condition> (class), 8
<queue-interrupt> (class), 8
<queue-stopped> (class), 8
<queue> (class), 6
queue-condition-queue (generic function), 8
queue-condition-thread (generic function), 9
queue-name (generic function), 7

S
<single-thread-executor> (class), 5
stop-queue (generic function), 8

T
<thread-executor> (class), 5

W
$work-finished (constant), 10
$work-started (constant), 10
<work> (class), 9
work-finished? (generic function), 9
work-perform (generic function), 9
work-started? (generic function), 10
work-thread (generic function), 10
work-wait (generic function), 10
Symbols

$work-finished, 10
$work-started, 10
<executor>, 5
<fixed-thread-executor>, 5
<locked-queue>, 6
<locked-work>, 9
<queue-condition>, 8
<queue-interrupt>, 8
<queue-stopped>, 8
<queue>, 6
<single-thread-executor>, 5
<thread-executor>, 5
<work>, 9

D
dequeue, 7

E
enqueue, 7
executor-name, 5
executor-request, 5
    executor-request(<function>), 6
    executor-request(<work>), 6
executor-shutdown, 6

I
interrupt-queue, 8

Q
queue-condition-queue, 8
queue-condition-thread, 9
queue-name, 7

S
stop-queue, 8

W
work-finished?, 9
work-perform, 9
work-started?, 10