

Agents

# Table of Contents

Concepts .....	1
Agent .....	1
Usage .....	2
Agent implements a Clojure-like agent concept .....	2

# Concepts

## Agent

In the Clojure programming language you can find a concept of Agents, which essentially behave like actors accepting code (functions) as messages. After reception the received function is run against the internal state of the Agent and the return value of the function is considered to be the new internal state of the Agent. Essentially, agents safe-guard mutable values by allowing only a single **agent-managed thread** to make modifications to them. The mutable values are **not directly accessible** from outside, but instead **requests have to be sent to the agent** and the agent guarantees to process the requests sequentially on behalf of the callers. Agents guarantee sequential execution of all requests and so consistency of the values.

# Usage

## Agent implements a Clojure-like agent concept

```
import groovyx.gpars.agent.Agent

def jugMembers = new Agent<List>(['Me']) // Add Me.
jugMembers.send {it.add 'James'} // Add James.
final Thread t1 = Thread.start{
    jugMembers {it.add 'Jo'} // Add Jo --- using the implicit call() method to send the
    function.
}
final Thread t2 = Thread.start{
    jugMembers << {it.add 'Dave'} // Add Dave.
    jugMembers << {it.add 'Alice'} // Add Alice.
}
[t1, t2]*.join()
println jugMembers.val
jugMembers.valAsync {println "Current members: $it"}
System.in.read()
jugMembers.stop()
```