Actor

The Whole GPars Team <gpars-developers@googlegroups.com>

Version 2.0, 2017-02-01

Table of Contents

To read this topic in the PDF format, please click here.

Actor

*The actor support in **GPars** was inspired by the Actors library in Scala but has subsequently gone beyond that.*

Actors allow for a messaging-based concurrency model, built from independent active objects that exchange messages and have no mutable shared state. Actors can help solve or avoid issues like deadlocks, livelocks or starvation, so typical for shared memory. A nice wrap-up of the key concepts behind actors was written recently by Ruben Vermeersch

Actors Inside

Actors can share a relatively small thread pool. This can go as far as having many concurrent actors that share a single pooled thread. They avoid the threading limitations of the JVM.

Actor code is processed in chunks separated by quiet periods of waiting for new events (messages). This can be naturally modeled through *continuations*.

As JVM doesn't support continuations directly, they have to be simulated in the actors frameworks, which has slight impact on organization of the actors' code. However, the benefits in most cases outweigh the difficulties.

Actor sample

```
import groovyx.gpars.actor.Actor
import groovyx.gpars.actor.DefaultActor

class GameMaster extends DefaultActor {
    int secretNum

    void afterStart() {
        secretNum = new Random().nextInt(10)
    }

    void act() {
        loop {
            react { int num ->
                if (num > secretNum)
                    reply 'too large'
                else if (num < secretNum)
                    reply 'too small'
                else {
                    reply 'you win'
                    terminate()
                }
            }
        }
    }
}

class Player extends DefaultActor {
    String name
    Actor server
    int myNum

    void act() {
        loop {
            myNum = new Random().nextInt(10)
            server.send myNum
            react {
```

```
                switch (it) {
                    case 'too large':
                        println "$name: $myNum was too large"
                        break
                    case 'too small':
                        println "$name: $myNum was too small"
                        break
                    case 'you win':
                        println "$name: I won $myNum"; terminate()
                }
            }
        }
    }
}

def master = new GameMaster().start()
def player = new Player(name: 'Player', server: master).start()

[master, player]*.join()
```

example by *Jordi Campos i Miralles, Departament de Matematica Aplicada i Analisi, MAiA Facultat de Matematiques, Universitat de Barcelona*

Usage of Actors

For more details on Actors visit the Actors section of our Reference Book.

Please also see the numerous Actor Demos.