STM (Software Transactional Memory)

Russell Winder

Version 1.0, 2015-10-01

Table of Contents

Concepts

Software Transactional Memory

Software Transactional Memory or **STM**, gives developers transactional semantics for accessing in-memory data. When multiple threads share data in memory, by marking blocks of code as transactional (atomic) the developer delegates the responsibility for data consistency to the Stm engine. **GPars** leverages the Multiverse STM engine.

Atomic Closures

GPars allows developers to structure their concurrent code into atomic blocks (closures), which are then performed as single units, preserving the transactional ACI (*Atomicity, Consistency, Isolation*) attributes.

Usage

Running a Piece of Code Atomically

An Atomic Sample

```
import groovyx.gpars.stm.GParsStm
import org.multiverse.api.references.TxnInteger

import static org.multiverse.api.StmUtils.newTxnInteger

public class Account {
    private final TxnInteger amount = newTxnInteger(0);

    public void transfer(final int a) {
        GParsStm.atomic {
            amount.increment(a);
        }
    }

    public int getCurrentAmount() {
        GParsStm.atomicWithInt {
            amount.get();
        }
    }
}
```

Customizing the Transactional Properties

A Sample

```
import groovyx.gpars.stm.GParsStm
import org.multiverse.api.AtomicBlock
import org.multiverse.api.PropagationLevel

final TxnExecutor block = GParsStm.createTxnExecutor(maxRetries: 3000, familyName:
'Custom', PropagationLevel: PropagationLevel.Requires, interruptible: false)

assert GParsStm.atomicWithBoolean(block) {
    true
}
```