Ashkan Farivarmoheb

Transaction Processing Assignment

Advanced Software Architecture

Part I) Nested transactions

(App.java)

I created a resource that is prepared with @Transactional. This resource has implemented an interface that it also is prepared with @Transactional too. I use this resource and create it with Container.

```
Container<Counter> container = new Container<Counter>();

Counter counterImpl1 = container.create(new CounterImpl());
Counter counterImpl2 = container.create(new CounterImpl());
Counter counterImpl3 = container.create(new CounterImpl());
Counter counterImpl4 = container.create(new CounterImpl());
Counter counterImpl5 = container.create(new CounterImpl());
Counter counterImpl6 = container.create(new CounterImpl());
Counter counterImpl7 = container.create(new CounterImpl());
```

These counters will be used in t1 – t7 transactions. Transactions are AtmoicAction that are Narayana based. I created transactions as the assignment. And if the parent aborts, all its children will be aborted too. If parent commits, all its children will be committed too. I throw a RunTimeException in t4 transaction. Because the t4 is parent of t7, as a result the t4 and t7 will be aborted. Also values for counterImpl4 and counterImpl7 won't be changed.

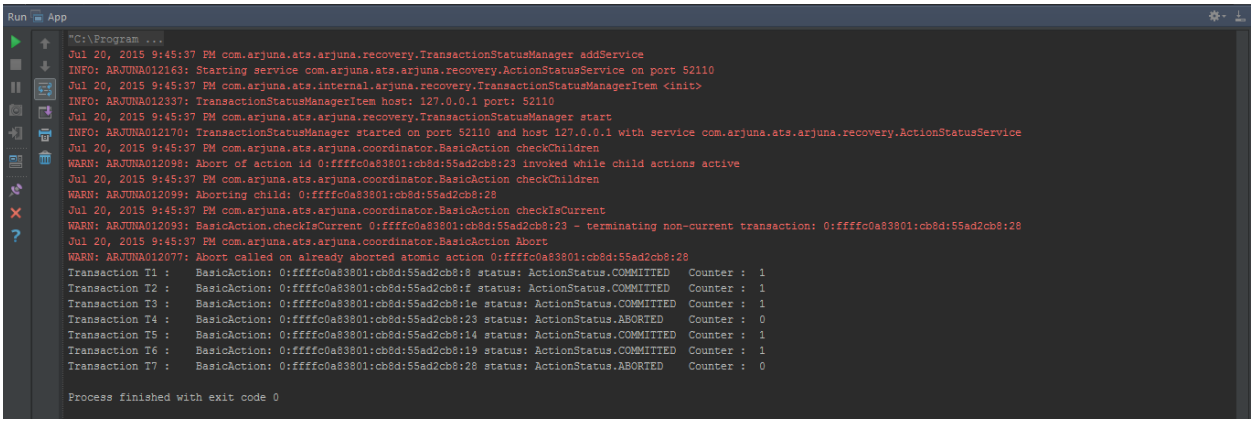
```
t4 = new AtomicAction();
t4.begin();
try {
    counterImpl4.increment();

    t7 = new AtomicAction();
    t7.begin();
    try {
        counterImpl7.increment();
    }catch (Exception e){
        abortRecursive(t7);
    }

    //int b = 1 / 0;
```

```
    if (true)
        throw new RuntimeException();

}catch (Exception e){
    abortRecursive(t4);
}
```



Part II) Compensation for Chained Transactions

(ChainWithCompensate.java)

Like the above I use a Counter resource that is prepared with @Transactional.

First I begin the t1 transaction, increment the counter by one and then commit it. For beginning the t2 transaction, I check the status of t1 is committed or not. If it equals with committed then I will begin the t2 transaction, increment by one. After that I throw a RunTimeException and the catch section will execute a compensate and decrement the counter by one. Because the t2 can't commit the counter will be one and in compensate section I decrement by one to reach initial state of t1 or compensate t1.