CNT in JOTM

DUSART Doriane

June 10th 2002

Université de Valenciennes
LAMIH/ROI/SID
Le Mont Houy, 59313 Valenciennes Cedex 9
ddusart@meletu.univ-valenciennes.fr

10/06/02
Table of contents

- What exists
- CNT
- CNT in local and distributed
What exists

– Local transaction / Distributed transaction
– JOTM architecture
– JTA
– Local transaction
– JTS
– Distributed transaction
A remote client of an enterprise bean has the following traits:
Local transaction in J2EE

Source: http://java.sun.com/j2ee/tutorial/1_3-cs/doc/EJBConcepts6.html#62995

A local client has these characteristics:
JOTM

Client ➔ JTA ➔ JOTM

Bean

Begin, End Transaction

JTA

Local Transaction Manager

TM “JTS Like”
JTA Operation

JTA specification: ftp://ftp.java.sun.com/pub/jta/tui00/jta-spec1_0_1.pdf
JTA interfaces (1 /2)

- **javax.transaction** :
  - **Status** : status of the transaction.
  - **Synchronization** : synchronization allows transaction manager to notify application server, before the begin of transaction, and after this end.
  - **Transaction** : This interface allows to associate a transaction with a transactionnal object when a transaction is created.
  - **TransactionManager** : This interface allows to manage states of a transaction.
  - **UserTransaction** : This interface can be used by a custom program or an EJB bean.
JTA interfaces (2 /2)

- `javax.transaction.xa`:
  - **XID**: It's a mapping of the XID identification structure of a transaction according to X/Open. It's used by transaction and resource managers.
  - **XAResource**: This interface gives the contract between the transaction manager and resources manager in case of distributed transactions.
## JTA / JOTM

<table>
<thead>
<tr>
<th>Interfaces JTA</th>
<th>Implementation of JTA in JOTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransactionManager</td>
<td>Current</td>
</tr>
<tr>
<td>UserTransaction</td>
<td>Current</td>
</tr>
<tr>
<td>Transaction</td>
<td>TransactionImpl</td>
</tr>
<tr>
<td>Xid</td>
<td>XidImpl</td>
</tr>
</tbody>
</table>
Local transaction (1 /2)

UserTransaction

Current

Begin
Commit
Rollback

Ask to begin

Ask to commit

Ask to abort
Local transaction (2 /2)

Class **Current**: JTA TransactionManager

begin()

Class **TransactionImpl**: JTA Transaction

TransactionImpl(xid:Xid,timeout:int ,corba:boolean)

doAttach(status:int)
enlistResource(xares:XAResource)
makeSubCoord()

**Legend**: JTA implementation, JTS implementation, proprietary implementation

**TransactionContextRMI**

TransactionContextRMI(t:int,c:Coordinator,x: Xid)

**SubCoordinator**: Resource

SubCoordinator(tx:TransactionImpl,xid:Xid)
JTS API


Java Mapping of OTS 1.1 Interfaces Impl.

JTA Support

javax.transaction.xa.XAResource

Resource Adapter

javax.transaction.TransactionManager

Resource Manager

Application Server

javax.transaction.xa.XAResource

org.omg.CosTransactions.Resource

Java Applications

javax.transaction.TransactionManager

CNT in JOTM
JTS interface

- Mapping java of OTS Specification version 1.1
- Low level API
- A transaction manager (JTS compliant) gives the following services:
  - Possibility of controlling the range and the duration of a transaction.
  - Several objects allowing to do a work like part of a simple transaction.
  - Gives the possibility of associating a total transaction with a work carried out transactional resources.
  - Coordinates the achievement of a total transaction through several resource manager.
  - Supports the synchronization of the transactions.
  - Gives the possibility to interoperate with other transaction managers using standard CORBA ORB/TS.
# JTS / JOTM

<table>
<thead>
<tr>
<th>Interface JTS</th>
<th>Implementation of JTS in JOTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>ControlImpl</td>
</tr>
<tr>
<td>Coordinator</td>
<td>ControlImpl</td>
</tr>
<tr>
<td>RecoveryCoordinator</td>
<td>ControlImpl</td>
</tr>
<tr>
<td>Terminator</td>
<td>ControlImpl</td>
</tr>
</tbody>
</table>
Distributed transaction (1 /2)

**LEGENDE :**
- JTA implementation
- JTS implementation
- proprietary implementation

**JTA**
- Begin
- Commit
- Rollback

**“JTS Like”**
- Begin
- Commit
- Rollback

- Ask to begin
- Ask to commit
- Ask to abort
Distributed transaction (2/2)

### Classes and Methods

**Class **TransactionImpl: JTA Transaction
- `TransactionImpl(pctx:TransactionContext)`
- `makeSubCoord()`

**Class **Current: JTA TransactionManager
- `setPropagationContext(pctx:TransactionContext, isReply:boolean)`
- `getJTM()`

**SubCoordinator: **Resource
- `SubCoordinator(tx:TransactionImpl, xid:Xid)`

**Skeleton**
- `inRequest(call:java.rmi.server.RemoteCall)`
- `outReply(call:java.rmi.server.RemoteCall)`

**TransactionFactoryImpl**
- `create(timeout:int)`

**ControlImpl:** JTS
- `ControlImpl(timeout:int, x: Xid, supco:Object)`

**Legend:**
- JTA implementation
- JTS implementation
- Proprietary implementation

10/06/02 CNT in JOTM
CNT

- Description
- CNT for distributed transactions
- CNT for local transactions
- Problem
- Proposal
CNT
Description

Cf cours de Didier Donsez
CNT

Advantages (1 /2)
CNT
Advantages (2/2)

Begin transaction

Find Stocks + vehicle + update database

Error during update database

Begin transaction

Find Stocks + vehicle + update database

Error during update database

CNT in JOTM
CNT for distributed transaction (1/4)

- Pepita project
- Allows to create CNT in a distributed environment
- New interface and update of interfaces and classes
CNT for distributed transaction (2/4)

Begin transaction

Class `Current` : JTA TransactionManager

```
begin()
```

LEGEND:

- JTA implementation
- JTS implementation
- proprietary implementation

Class `ControllImpl` : JTS Control

```
ControllImpl(int timeout, Xid x, Object supco)
create_subtransaction()
```

Class `XidImpl` : JTA Xid

```
XidImpl(String serverName, int ipAddr)
```
CNT for distributed transaction (3 /4)
Relation parent-children

ControlImpl : Ctrl1

Create_subtransaction()

Xid X2

ControlImpl: Ctrl2
Ctrl2.isdescendant = true

Ctrl2.parent = Ctrl1

Ctrl1.child = Ctrl2
Ctrl1.subTransactions = Ctrl2

LEGENDE :
- JTA implementation
- JTS implementation
- proprietary implementation

CNT in JOTM
CNT for distributed transaction (4 /4)

End of transaction

**Class Current**: JTA TransactionManager
- `commit()`
- `rollback()`

**Class ControlImpl**: JTS Control
- `commit()`
- `do_commit(heuristic)`
- `rollback()`

**Resource Commit**: Commit
- `Commit()`
- `Rollback()`

**Subcoordinator**: Resource Commit
- `Commit()`
- `Rollback()`

**XAResource**: JTA
- `Commit()`
- `Rollback()`

**LEGENDE**:
- JTA implementation
- JTS implementation
- proprietary implementation
CNT for local transaction

**JOTM uses a special transaction manager for local transactions**

**Keep coherence of JOTM's architecture**

**Needed to integrate CNT in local**

- For example, if an update of 1,000,000,000 of products in a database have a problem in the 999,999 product, we can keep the modifications of the 999,998 first products, and repeat only the subtransaction 999,999.
Problem

• Commit in local transaction:
  - Need of relation between parents and children.
  - Need to have a commit and rollback for local subtransaction.
  - Transaction manager for local transactions

• 2 solutions:
  1. Use only one transaction manager for local and distributed transactions
     - JTS via JTA
  2. Extends the transaction manager for local transaction to the CNT
     - A JTS light with relations parents-children and commit-rollback
Proposal (1 /2)

This proposal is an extension of the local transaction manager of JOTM => Extension of JTA implementation in JOTM

- **CoordinatorLocal**: Give identifier of top level transaction and the recovery coordinator

- **TerminatorLocal**: idem OTS

- **TransactionControlLocal**: idem OTS

- **TransactionLocal**: Relations between parents and children

- **SubtransactionAwareResourceLocal**: idem OTS
Proposal (2/2)

Access to a remote Resource

LEGENDE:
TI : TransactionImpl
CI : ControlImpl
CNT in distributed transaction now
Begin Transaction

Class **Current** : JTA TransactionManager
- `begin()`

Class **TransactionImpl** : JTA Transaction, TransactionControlLocal
- `TransactionImpl(xid:Xid,timeout:int,corba:boolean)`
- `createSubTransaction()`

Class **ControlImpl** : JTS Control
- `ControlImpl(int timeout, Xid x, Object supco)`
- `create_subtransaction()`

Class **XidImpl** : JTA Xid
- `XidImpl(String serverName, int ipAddr)`

**LEGENDE :**
- JTA implementation
- JTS implementation
- proprietary implementation
CNT for distributed transaction now
End of transaction

Class **Current**: JTA TransactionManager
- **commit()**
- **rollback()**

**TransactionImpl**: JTA Transaction
- **Commit()**
- **Rollback()**

**ControlImpl**: JTS Control
- **commit()**
- **do_commit(heuristic)**
- **rollback()**

**XAResource**: JTA
- **Commit()**
- **Rollback()**

**Subcoordinator**: Resource
- **Commit()**
- **Rollback()**

**Resource**
- **Commit()**
- **Rollback()**

**LEGENDE**:
- **JTA implementation**
- **JTS implementation**
- **proprietary implementation**
CNT in local transaction

Begin Transaction

Class Current: JTA TransactionManager

begin()

Class TransactionImpl: JTA Transaction

TransactionImpl(xid:Xid,timeout:int,corba:boolean)

class TransactionImpl(xid:Xid,timeout:int,corba:boolean)

dispatchStatus(int)

enlistResource(xares:XAResource)

makeSubCoord()

SubCoordinator: Resource

SubCoordinator(tx:TransactionImpl.xid:Xid)

TransactionContextRMI

TransactionContextRMI(t:int,e:Coordinator,x:Xid)

LEGENDE:

JTA implementation
JTS implementation
proprietary implementation
CNT for local transaction
End of transaction

Class **Current** : JTA TransactionManager
- commit()
- rollback()

**TransactionImpl** : JTA Transaction
- Commit()
- Rollback()

**TerminatorLocal**
- Commit()
- Rollback()

**LEGENDE** :
- JTA implementation
- JTS implementation
- new classes
Conclusion

- What exists in JOTM
- The 2 transactions managers
- CNT for distributed transaction
- Problem to keep the coherence of architecture
- Need to choose an architecture to include CNT
- Include CNT for local and distributed transactions
- Include the ONT in JOTM