Java Transaction API

Giuseppe Sottile

Nell'ambito della programmazione web le transazioni occupano un ruolo chiave per il corretto mantenimento dei dati. Una transaction (o transazione) é essenzialmente un insieme di job o task che devono essere eseguiti necessariamente nella loro interezza, ovvero vige la regola del **tutto o niente**: o la transazione ha successo completamente *(in questo caso una particolare istruzione: commit, indica al gestore delle transazioni o al DBMS di rendere persistenti tutte le operazioni che compongono la transazione.)*; oppure se qualcosa va storto viene effettuata un'operazione di aborto **roll-back** in cui viene ripristinato lo stato in cui si trovava il sistema antecedente alla transazione e le modifiche non vengono rese persistenti.

ACID

Ogni sistema informativo o transazionale *(cioé orientato alle transazioni)* supporta 4 proprietá note con l'acronimo **ACID** che stá per **Atomicitá, Consistenza, Isolamento, Durevolezza**. Analizziamone le caratteristiche piú in dettaglio:

- ◇ Atomicitá: La transazione é indivisibile ed avrá successo solo se tutti i task vengono portati a termine.
- ◇ Consistenza: Il sistema deve mantenere uno stato di consistenza dei dati sia prima che dopo ogni transazione.
- ◇ Isolamento: Specie nei sistemi concorrenti in cui piú processi vengono eseguiti contemporaneamente deve essere garantito che le transazioni vengano isolate le une dalle altre e risultino appunto indipendenti e non interferiscano l'un l'altra. Per l'isolamento si distinguono 4 casi differenti:
  - Read Uncommitted: una transazione puó leggere i dati computati dalle altre transazioni, di conseguenza se una delle transazioni fallisce ci possono essere letture di dati errati. Non garantisce nulla di buono.
  - Read Committed: una transazione rende visibile gli aggiornamenti solo dopo aver fatto il commit. Risolve il problema delle letture sporche *dirty reads*.
  - Repeatable read: risolve il problema delle letture ripetute. Una transazione rende visibili gli aggiornamenti solo dopo aver fatto il commit,

in piú se un'altra transazione aveva iniziato a leggere i dati della prima, prima che questa avesse fatto il commit essa continuerá a vedere i dati precedenti, fino al termine del suo ciclo di vita.

- Serializable: É il vincolo piú forte. Simile al caso precedente ma bloccante nelle letture.

◇ DUREVOLEZZA: La caratteristica forse piú importante, la tolleranza ai guasti anche detta **fault tolerant**, in cui il sistema garantisce che si congeli lo stato dei dati a seguito di guasti.

JTA

In J2EE si possono effettuare 2 tipi di transazioni: Quelle gestite direttamente dal container dell'applicazione **CMT: Container-managed Transaction** oppure quelle applicative **BMT: Bean-managed Transaction**. Di default ogni sistema gestisce le transazioni secondo l'approccio CMT. Analizziamo piú in dettaglio come funziona la gestione standard con l'impiego del container.

CMT Transaction

Il modo piú semplice per implementare una transazione é quello di delegare tutto il lavoro sporco al container, utilizzando delle semplici annotazioni java nel file sorgente. Il container si occuperá di inizializzare la transazione di farne il commit o eventualmente il roll-back nel caso di errori. Diamo uno sguardo alle annotazioni da impiegare: Per dire che tipo di transazione si vuole impiegare (CMT oppure BMT) si usa **@TransactionManagement** come nell'esempio:

Listing 1: mybean.java

```
@TransactionManagement(TransactionManagementType.CONTAINER)
public class MyBean
{
   @Resource
   private SessionContext sessioncontext;

   public MyBean()
   {
   }
}
```

Nell'esempio la riga
`@TransactionManagement(TransactionManagementType.CONTAINER)` informa il sistema che si vuole far uso della gestione tramite container delle transazioni, alternativamente per specificare l'uso dei BMT dobbiamo scrivere:

Listing 2: mybean.java

```
@TransactionManagement(TransactionManagementType.BEAN)
public class MyBean
{
   @Resource
   private SessionContext sessioncontext;

   public MyBean()
   {
   }
}
```

In questo modo abbiamo attivato il commit ed il roll-back sul java bean. Il framework java capisce sulla base del corpo della classe quando termina il suo transaction-scope, cioé l'ambito in cui il gestore di transazioni deve agire. Quando viene invocato un metodo che fa parte dello scope-transacrtion il sistema inizializza la transazione e la termina (commit o rollback) dopo che il metodo ritorna al chiamante.

0.1 Gli attributi di tipo

Gli attributi di tipo specificano il modo in cui la transazione deve essere gestita e si distinguono in sei categorie:

- REQUIRED: richiede che un metodo sia sempre invocato con una transazione.
- REQUIRES_NEW: richiede che la transazione avvenga prima dell'invocazione di un metodo.
- SUPPORTS: il metodo eredita le proprietá transazionali del chiamante.
- MANDATORY: se il metodo é transazionale verrá eseguito all'interno della transazione esistente.
- NOT_SUPPORTED: evita l'invocazione di un metodo in un ambiente transazionale.
- NEVER: fa in modo che un metodo non venga mai invocato in un contesto transazionale.