GLASSFISH 4.1

Code Inspection Document

Class: TopCoordinator

Methods: get_status, register_resource

version 1.1

Rota Diego, 841344, 10 hours

Montalto Simone, 841359, 10 hours

Politecnico di Milano, A.A. 2015-2016

Software Engineering 2 – Prof.ssa Mirandola Raffaela

Table of contents

1. Assigned Class and methods

There has been assigned the **TopCoordinator** class (location: appserver/transaction/jts/src/main/java/com/sun/jts/CosTransactions), and in the specific, these are the two methods:

- get_status();
- register_resource(Resource res).

2. Functional Role of the assigned methods

To understand the behaviour of the class and methods, we have follow this strategy:

- Read associated Javadoc;
- Search for objects type in the other classes of source code;
- Detailed reading of source code
- Check the online version of GlassFish manual and diagram, to better understand the relations between classes and packages.

The **TopCoordinator** class allows **Resources** to be registered in and recovered from a transaction. The class is thread-safety because in case of failure the information managed should be constructible again. We have to take into account two methods: *get_status()* and *register_resource(Resource res)*.

2.1 get_status()

The *get_status()* method convert the state value of a transaction. This method merges the possible states of a transaction (attribute *tranState.state,* with 15 possible values) into some more generic (9 possible values) with the following procedure:

- reads the **tranState** variable (public attribute of TopCoordinator class) that identifies the actual status of the transaction;
- using a **switch-case**, map more detailed value into a generic one;
- if tranState is equal to null an INVALID_TRANSACTION exception is thrown;
- if the status of transaction is not recognized, a generic StatusUnknown is assigned;
- The get_status() method return the local status of the target transaction (a variable of type Status).

2.2 Register_resource(Resource res)

The *register_resource(Resource res)* method manage the registration of a resource (passed as parameter) to a target transaction. Various checks are made to verify if a resource could be registered:

- First, the method check the status of the associated transaction: if is not *ACTIVE* it throws an Inactive exception (to avoid the useless occupation of a resource from an invalid transaction);
- If the associated transaction is checked as rollbackOnly (transaction not completed) a Transaction_Rolledback exception is throw;
- In GlassFish, the set of coordinators (of class Coordinator and its inherited classes) is managed like a tree structure: we have a root coordinator of type TopCoordinator (with a root flag set true) and multiple subcoordinators. Every coordinator must be registered to a superior one. To manage this structure, a Decorator pattern is used. A CoordinatorResourceImpl class is created, so we can add these information: globalTID, the coordinator for the transaction and a flag to indicate if this is a subtran. If the resource is not registered yet, the CoordinatorResourceImpl object is created and registered;
- A set of resources that is used in the transaction is created (called partecipants) to keep track of the resources registered in the transaction;
- A RecoveryCoordinatorImpl is an object used to allow the recovery of a resource from a previous failure. This object is created at the end of this method and is stored in a vector called recoveryCoordinatorList;
- The method return the RecoveryCoordinator object.

3. List of issues

We refer only on the class *TopCoordinator* for the methods and attributes name, in the specific only for the two methods *get_status* and *register_resource* (for the others rules).

In general, we have founded that the following rules (rules that are in the PDF file "*Assignment 3*") are not been satisfied:

RULES NUMBER	***RESULTS***
1	PROBLEMS
3	PROBLEMS
5	PROBLEMS
9	PROBLEMS
11	PROBLEMS
12	PROBLEMS
23	PROBLEMS
25	PROBLEMS

In the following table there are all the details about the problems found, with line number, the number of the rule and a comment.

RULE NUMBER	***LINE NUMBER***	***COMMENTS***
Naming Conventions		
1	816	Rename the method *get_status()* in *get_local_status()* because the method return the local status of the transaction
1	128	Rename *tranState* attribute in *transactionState*
1	1193	Rename the class *Inactive* in *InactiveException* because is an exception class
1	124	Rename *participants* in *partecipants_resources* because otherwise is not clear enough which type of participants is referred to
3	1200	Rename the class *TRANSACTION_ROLLEDBACK* in *TransactionRolledBack*, according to java style conventions
3	1269	Rename the class *INTERNAL* in *Internal*, according to java style conventions

5	947	Rename *is_same_transaction* in *isSameTransaction*
5	985	Rename *is_related_transaction* in *isRelatedTransaction*
5	1082	Rename *is_root_transaction* in *isRootTransaction*
5	1036	Rename *is_ancestor_transaction* in *isAncestorTransaction*
5	1069	Rename *is_descendant_transaction* in *isDescendantTransaction*
5	1100	Rename *is_top_level_transaction* in *isTopLevelTransaction*
5	1122	Rename *hash_transaction* in *hashTransaction*
5	1148	Rename *hash_top_level_tran* in *hashTopLevelTran*
5	1183	Rename *register_resource* in *registerResource*
5	1353	Rename *register_subtran_awake* in *registerSubtranAwake*
5	1372	Rename *rollback_only* in *rollbackOnly*
5	1399	Rename *get_transaction_name* in *getTransactionName*
5	1429	Rename *create_substransaction* in *createSubTransaction*
Indention		
9	1228 → 1237	In this block multiple TAB are found
9	1293 → 1297	In this block multiple TAB are found
Braces		
11	829	Curly braced is missed
11	831	Curly braced is missed
11	1229	Curly braced is missed
File Organization		
12	123	Blank line is missed between class declaration and attributes
12	822 → 875	After each break statement a blank line is needed
Java Source Files		
23	1517	Javadoc missed for method *getGlobalTid()*
23	1521	Javadoc missed for method *getParticipantsCount()*
Class and Interface Declarations		
25	142	Static attribute _*logger* must be before attributes declarations

25	1761	Static attribute *resultName* must be before attributes declarations
25	2764	Static attribute *emptyData* must be before attributes declarations