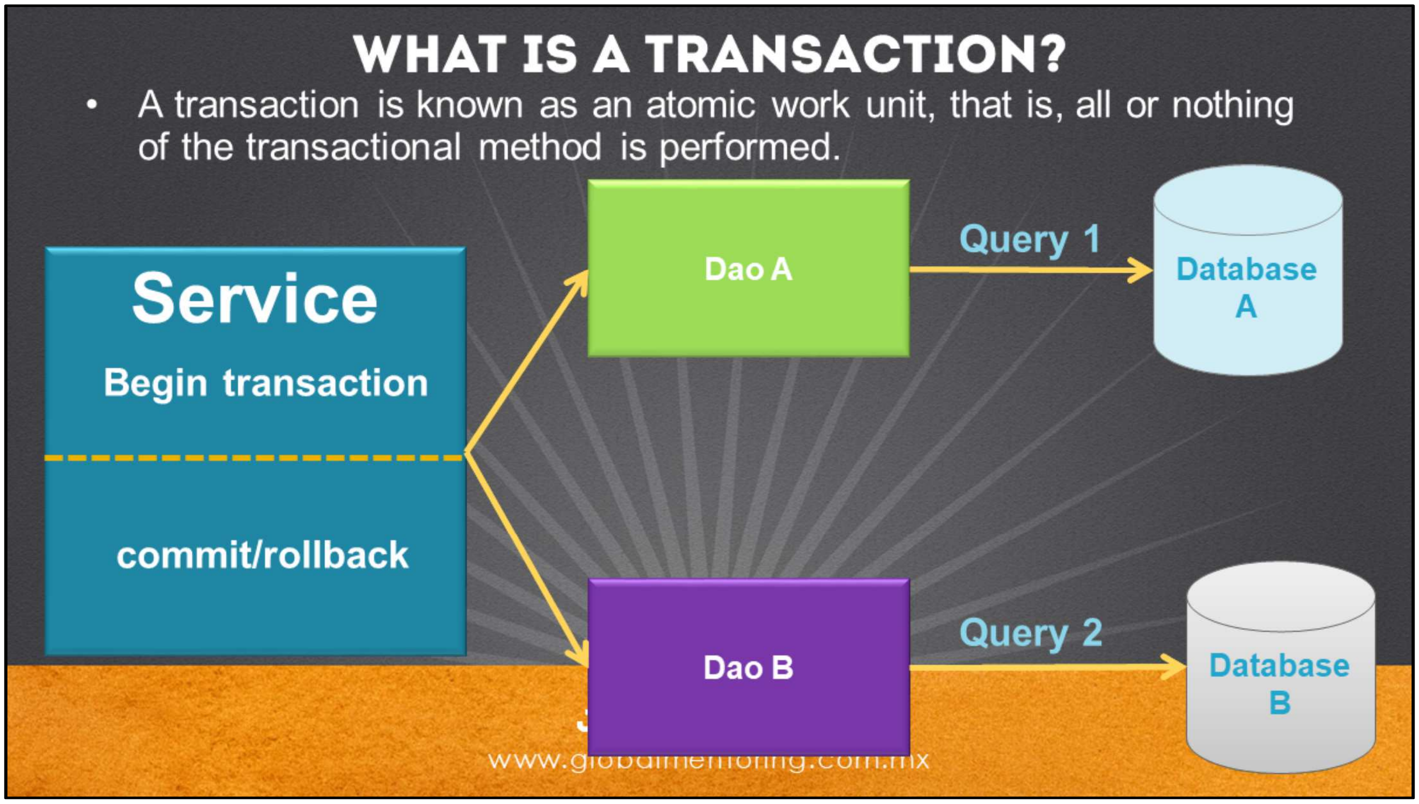
Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson.

We are going to study the topic of Transaction Management in Java EE.

Are you ready? Let's go!



WHAT IS A TRANSACTION?

- A transaction is known as an atomic work unit, that is, all or nothing of the transactional method is performed.

Transactional Management is one of the crucial issues in terms of requirements for business applications. This motivation arises because in any business system we are interested in maintaining the integrity of our information, with this in mind is that the issue of transactions arises.

In the figure we can see a service method that executes calls to more than one DAO, and in turn each DAO modifies the state of the database when writing and / or modifying its information.

The objective of a transaction is to execute all the lines of code of our method and finally save the information in a repository, for example in our case, a database. This is known as commit of our transaction.

If for some reason something were to fail in our Service method, the changes made to the database would be reversed. This is known as rollback.

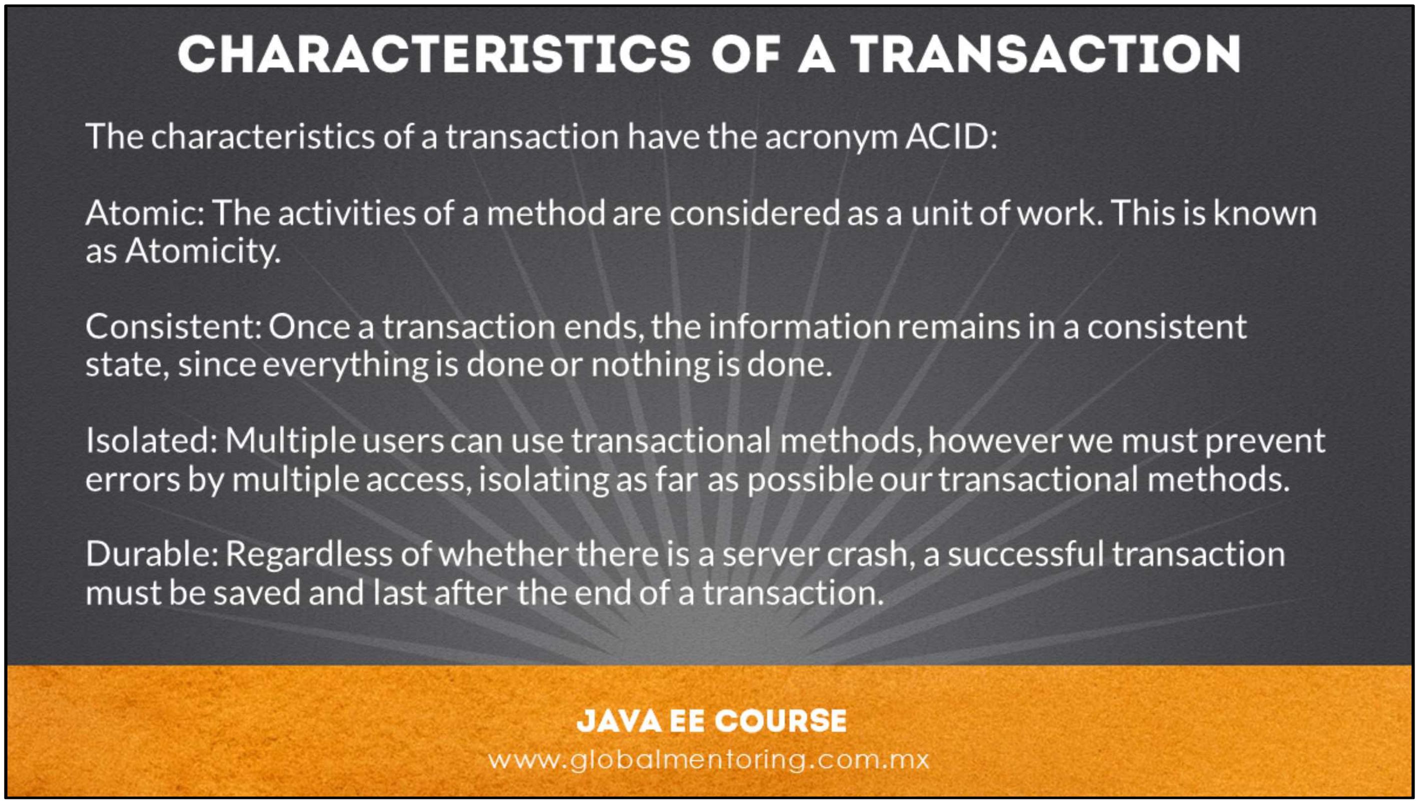
This allows our information, whether a single database or not, is complete, and there is no possibility of corrupted data due to errors or failures in the execution of our Java methods.

For example, imagine a system for selling airline tickets online. In this case, the necessary steps to reserve a ticket are:

1. Verify available tickets
2. Book a ticket
3. Make the payment
4. Receive the ticket information

If for any reason any of the steps fail, then the ticket should not be considered as a sold ticket, but as an available ticket. In case everything has been done successfully, then the ticket is no longer available and the state of the database is updated to reflect one less ticket for new users interested in purchasing a plane ticket.

So the methods that we make part of a transaction will be executed as a single action, which guarantees that they are executed completely, or if they fail, none of the information is persisted. The EJB when running in a Java Enterprise container, by default are transactional, therefore, any method of an EJB manages this concept automatically.



CHARACTERISTICS OF A TRANSACTION

The characteristics of a transaction have the acronym ACID:

Atomic: The activities of a method are considered as a unit of work. This is known as Atomicity.

Consistent: Once a transaction ends, the information remains in a consistent state, since everything is done or nothing is done.

Isolated: Multiple users can use transactional methods, however we must prevent errors by multiple access, isolating as far as possible our transactional methods.

Durable: Regardless of whether there is a server crash, a successful transaction must be saved and last after the end of a transaction.

JAVA EE COURSE
www.globalmentoring.com.mx

The characteristics of a transaction have the acronym **ACID**:

Atomicity: The activities of a method are considered as a unit of work. This is known as Atomicity. This concept ensures that all transactions in a transaction run all or nothing.

If all the instructions or lines of code of a transactional method are executed successfully, then at the end a commit is made, that is, saved from the information.

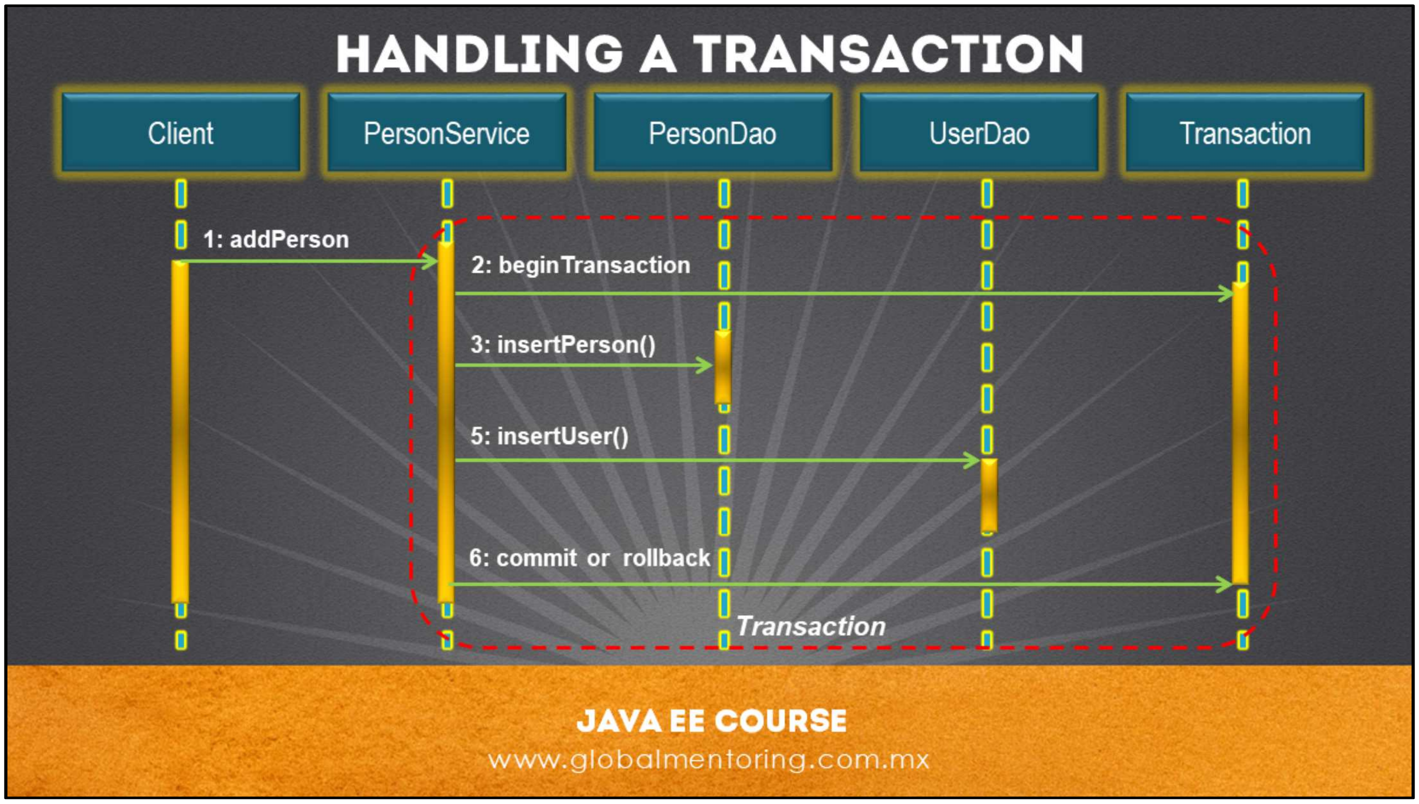
If any of the instructions fails, a rollback is performed, that is, none of the information is stored in the database or the repository where said information persists.

Consistent: Once a transaction ends (regardless of whether it has been successful or not) the information remains in a consistent state, since everything was done or nothing, and therefore the data should not be corrupt in any way.

Isolated: Multiple users can use transactional methods, without affecting the access of other users. However, we must prevent errors by multiple access, isolating as far as possible our transactional methods. Isolation normally involves blocking registers or database tables, this is known as locking.

Durable: Regardless of whether there is a server crash, a successful transaction must be saved and last after the end of a transaction.

In our example of buying tickets, a transaction can ensure atomicity because you can undo all changes made if any of the steps fails. The consistency of data would also be applied to ensure that nothing is left in a partial way, executing all or nothing. Isolation foresees that another user takes the reserved ticket while the transaction has not yet finished. And finally at the time of committing or rollback of the transaction the information is stored without inconsistencies, allowing it to be durable even after the completion of the transaction.

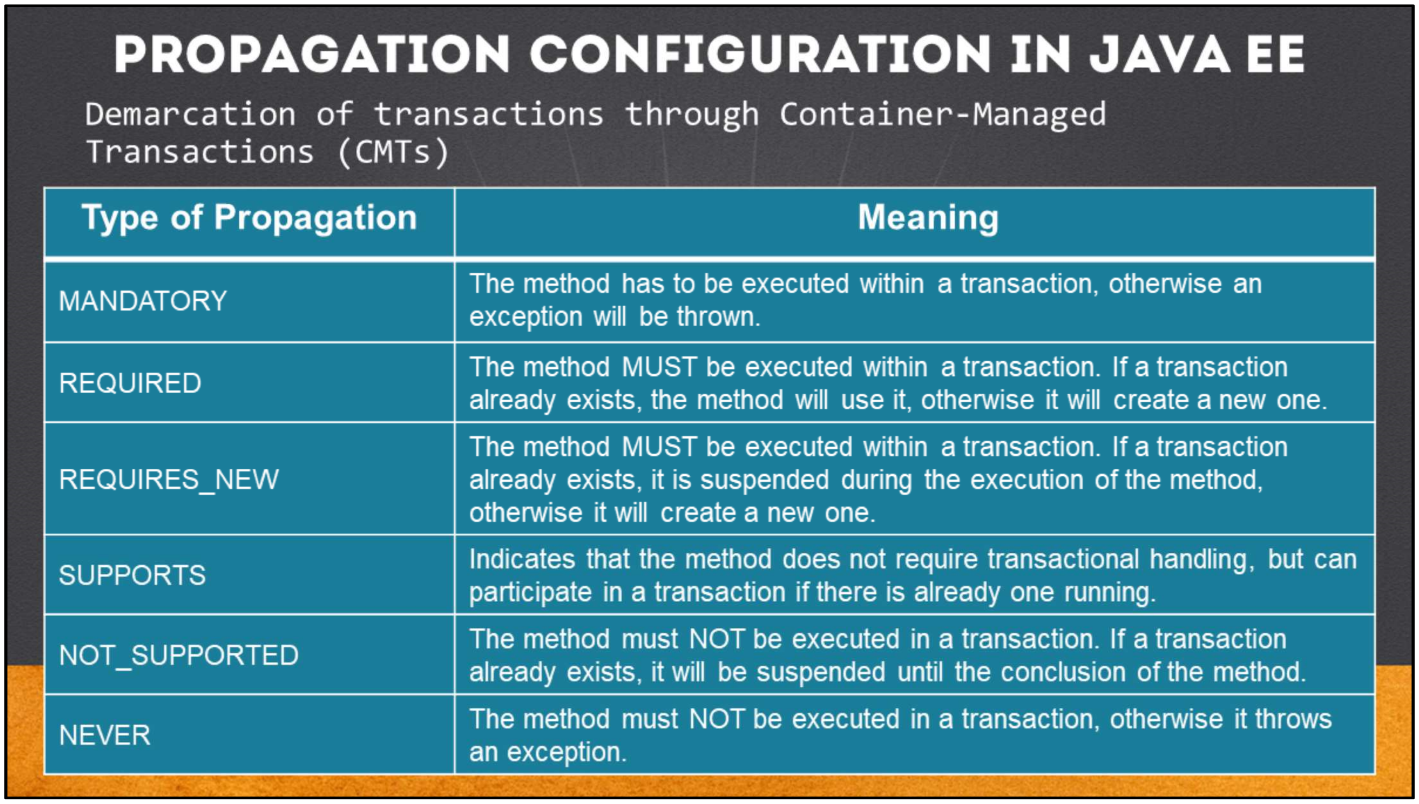


When creating business applications we must pay special attention to persistence. In addition, a business application, as we have studied, is divided into different layers, which have well-defined responsibilities.

In the figure we can see at what moment the Transaction object enters into participation. We observe the 3 layers of a business architecture (Client - Service - Data Access). In this architecture, the transaction begins at the service layer, which is in charge of establishing communication with the Data Layer (DAO).

The data layer is responsible for establishing communication with the database through the Entity Manager object. The transaction starts from the service layer, and propagates to the data layer. This is because the service layer can communicate with many DAO classes, and therefore the transaction ends until the business method has inserted, modified, deleted and selected all the required data from the database, applying the characteristics of the transaction management that we have studied.

The methods of the Service layer are those that contain much of the business logic of the application, and therefore it is at this level that we define transactional management, since if we apply it at the level of the data layer, the commit / rollback handling for each operation of a DAO, would affect the remaining operations that a business method has.



PROPAGATION CONFIGURATION IN JAVA EE

Demarcation of transactions through Container-Managed Transactions (CMTs)

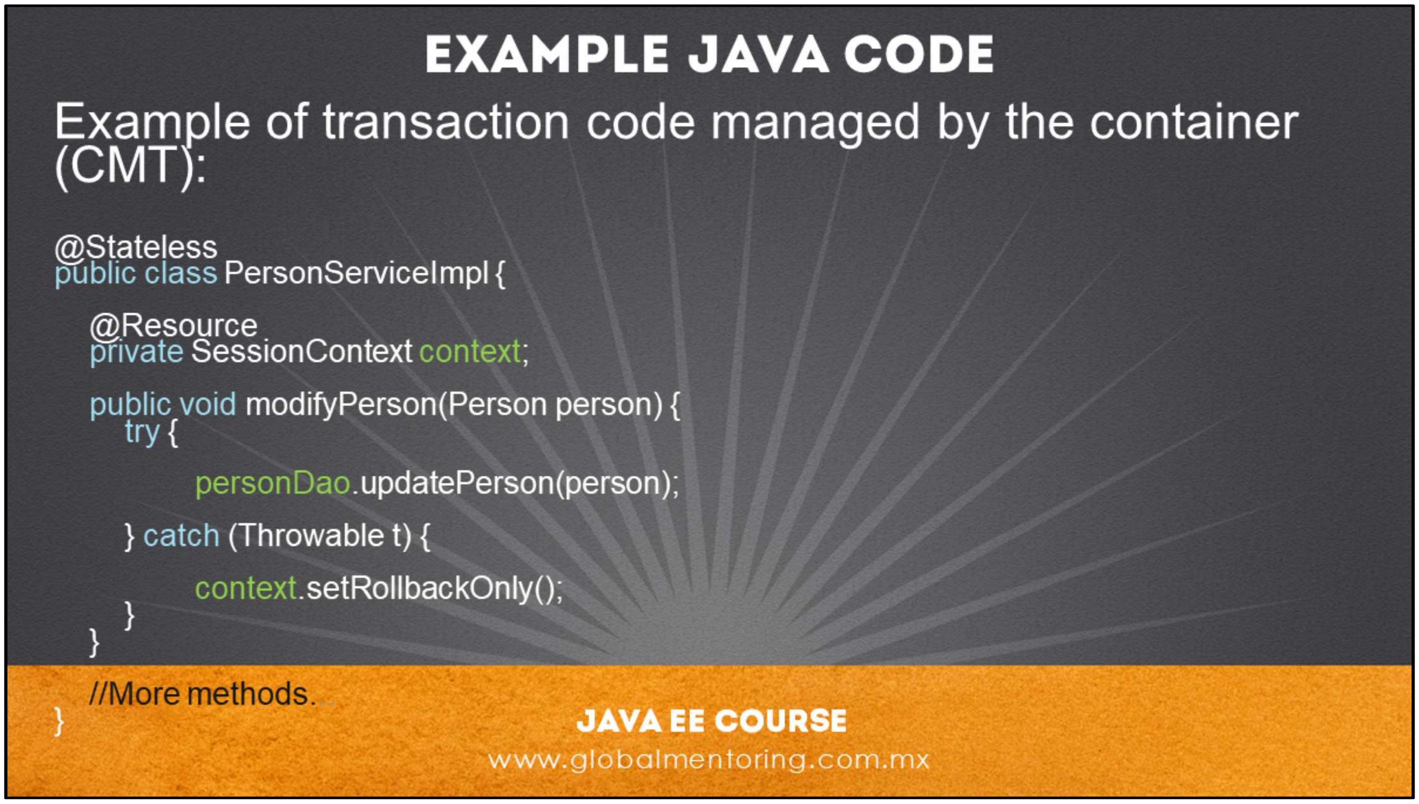
Type of Propagation	Meaning
MANDATORY	The method has to be executed within a transaction, otherwise an exception will be thrown.
REQUIRED	The method MUST be executed within a transaction. If a transaction already exists, the method will use it, otherwise it will create a new one.
REQUIRES_NEW	The method MUST be executed within a transaction. If a transaction already exists, it is suspended during the execution of the method, otherwise it will create a new one.
SUPPORTS	Indicates that the method does not require transactional handling, but can participate in a transaction if there is already one running.
NOT_SUPPORTED	The method must NOT be executed in a transaction. If a transaction already exists, it will be suspended until the conclusion of the method.
NEVER	The method must NOT be executed in a transaction, otherwise it throws an exception.

Propagation indicates how a method will behave before a transaction that has been previously initiated in another method, that is, how a transaction propagates between transactional methods. The default value of the propagation is REQUIRED. The types of propagation in a transaction are the following:

- MANDATORY: If there is no transaction, an exception is thrown.
- REQUIRED: This is the default behavior. If a transaction already exists, it spreads and uses it, otherwise it creates a new one.
- REQUIRES_NEW: Should only be used if the action on the database needs to be confirmed (commit) regardless of the result of the transaction in progress, for example, a log in an activity log, for each persistence of information, regardless of whether persists successfully or not.
- SUPPORTS: It does not need a transaction, however if there is one, it is used.
- NOT_SUPPORTED: Because it indicates that it does not support transactions, if there is any, it suspends it until the execution of the method marked with this attribute is completed.
- PROPAGATION_NEVER: Throws an exception if a transaction is running.

To indicate whether a method manages a method other than programming, the code is used before the method definition:

@TransactionAttribute (TransactionAttributeType.SUPPORTS)



EXAMPLE JAVA CODE

Example of transaction code managed by the container (CMT):

```
@Stateless
public class PersonServiceImpl {

  @Resource
  private SessionContext context;

  public void modifyPerson(Person person) {
    try {
          personDao.updatePerson(person);
    } catch (Throwable t) {
          context.setRollbackOnly();
    }
  }
  //More methods...
}
```

JAVA EE COURSE
www.globalmentoring.com.mx

As we have seen, the handling of transactions is carried out in the EJBs of the service or business layer. As we observed in the code shown, the EJB PersonaService contains service methods, which by default are transactional when executed in a business container.

The default behavior of an EJB method is of the REQUIRED type. However if we want to modify it we can use the code TransactionAttribute (TransactionAttributeType.SUPPORTS) either at the level of the class so that it affects all the methods, or at the level of the method that we want to modify.

Any code that expects to throw an exception, and if we want to apply rollback explicitly, we must use setRollbackOnly () of the SessionContext object, which we can inject directly using the @Resource annotation.

However, using the setRollbackOnly method will not cause the rollback of the transaction immediately, it is only an indication that as soon as the container finishes executing the method, the rollback must be performed.

Each JTA transaction is associated with the execution of a thread (Thread), so only one transaction can be executed at a time. In such a way that if a transaction is active, the user can NOT start another within the same thread (unless that transaction is suspended).

ONLINE COURSE

JAVA EE
JAKARTA EE

By: Eng. Ubaldo Acosta

JAVA EE COURSE
www.globalmentoring.com.mx