Spring
Transaction Management

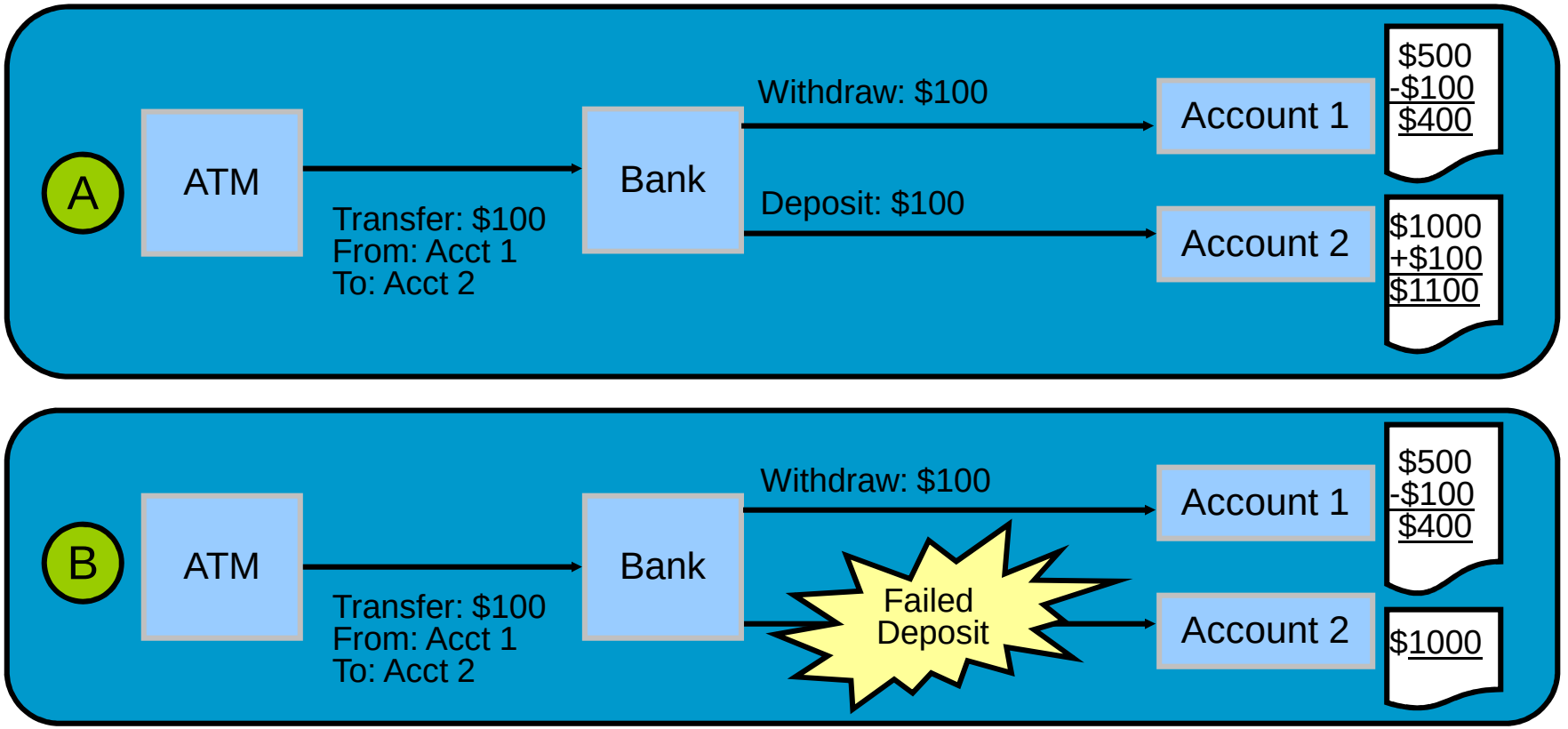
What Is a Transaction?

2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820707

- A transaction:
  - Is a single, logical unit of work or a set of tasks that are executed together
  - May access one or more shared resources (such as databases)
  - Must be atomic, consistent, isolated, and durable (ACID)

Example of a Transaction

- Successful transfer (A)
- Unsuccessful transfer (accounts are left in an inconsistent state) (B)

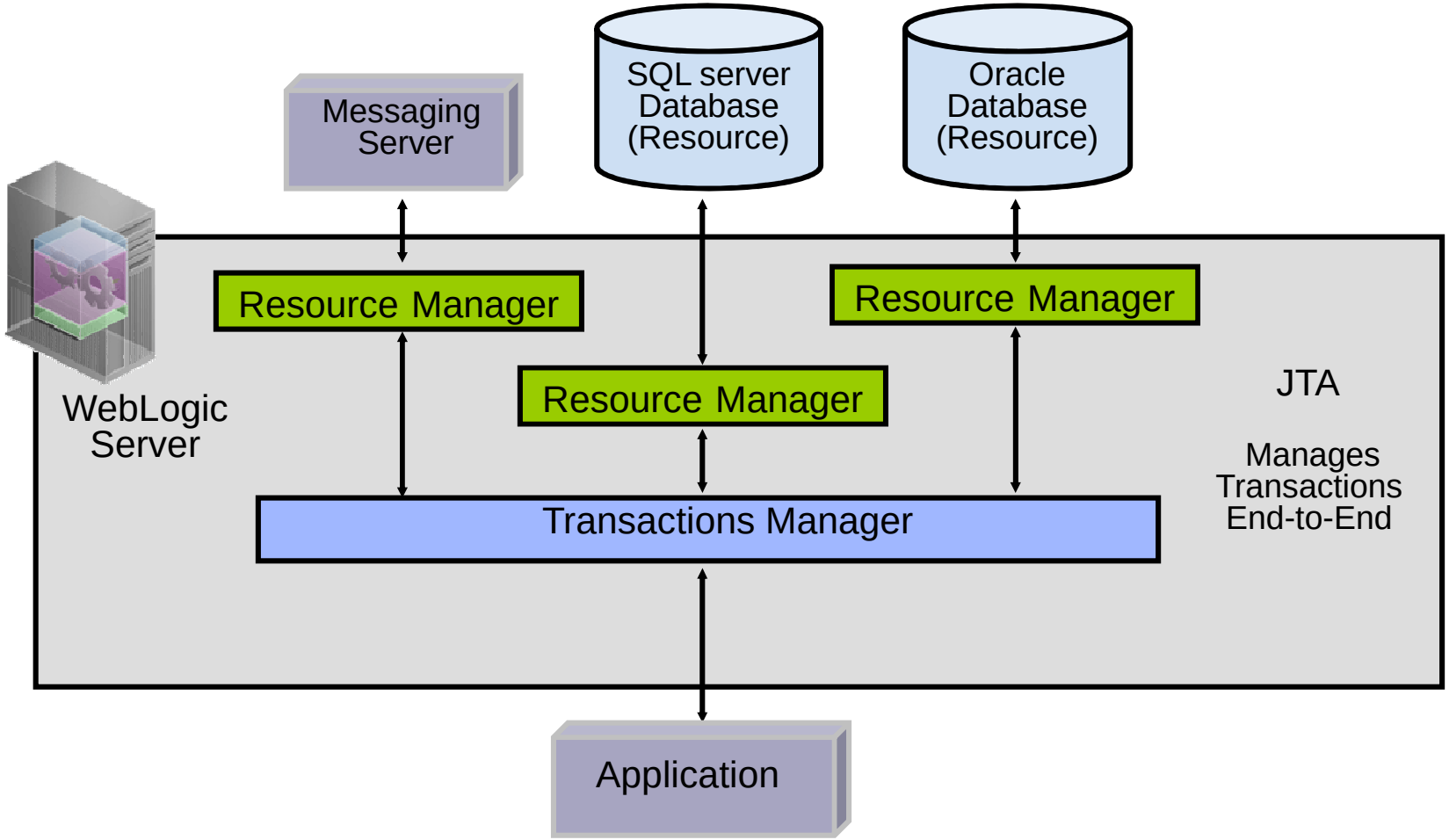


Types of Transactions

2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820707

- A *local transaction* deals with a single resource manager. It uses the non-Extended Architecture (non-XA) interface between WebLogic Server and resource managers.
- A *distributed transaction* coordinates or spans multiple resource managers.
- A *Global transaction* can deal with multiple resource managers. It uses the Extended Architecture (XA) interface between WebLogic Server and resource managers.
- You need to create non-XA or XA resources for local transactions. However, for global transactions, you need to create only XA resources.

Transaction Management

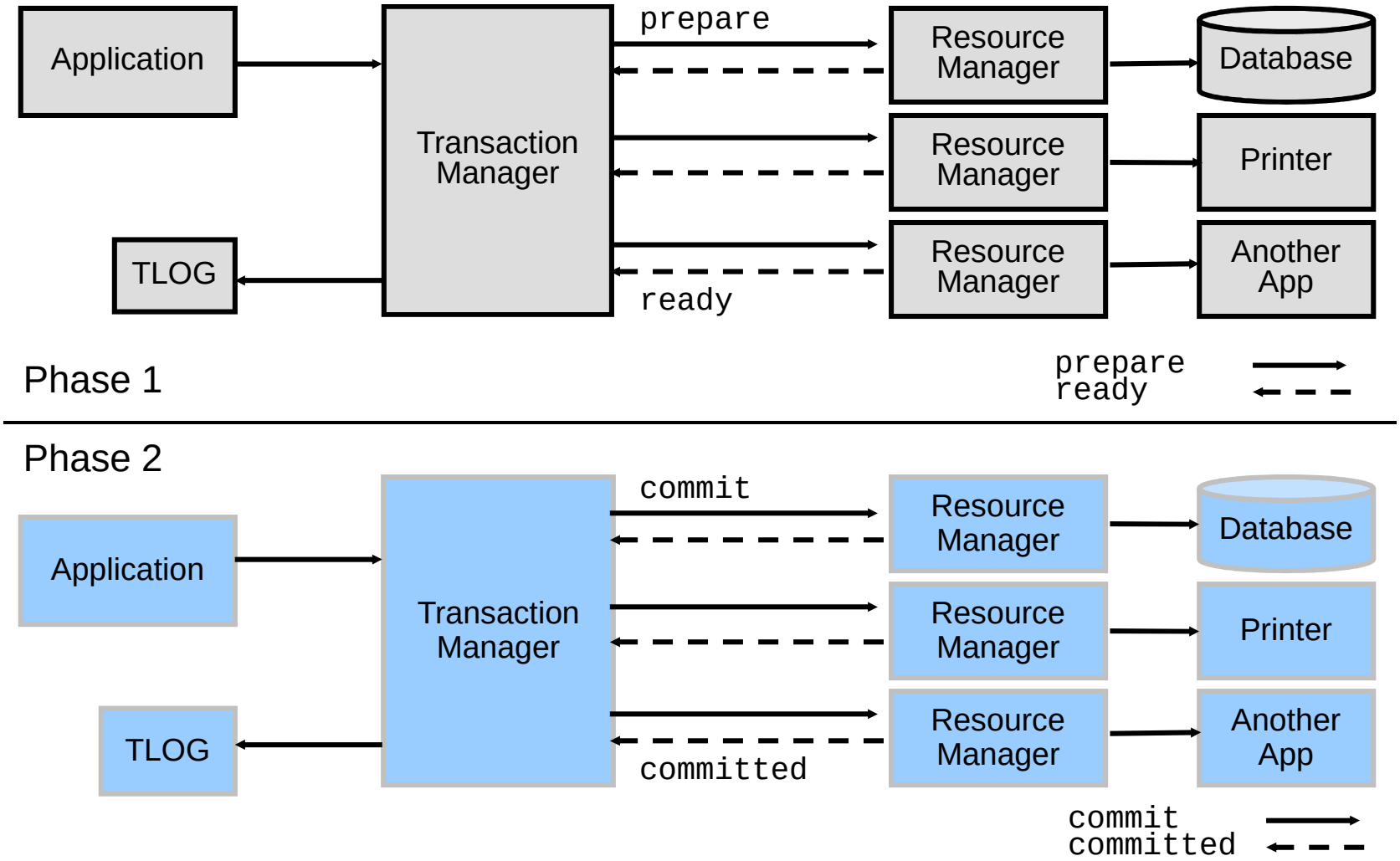


Two-Phase Commit Protocol

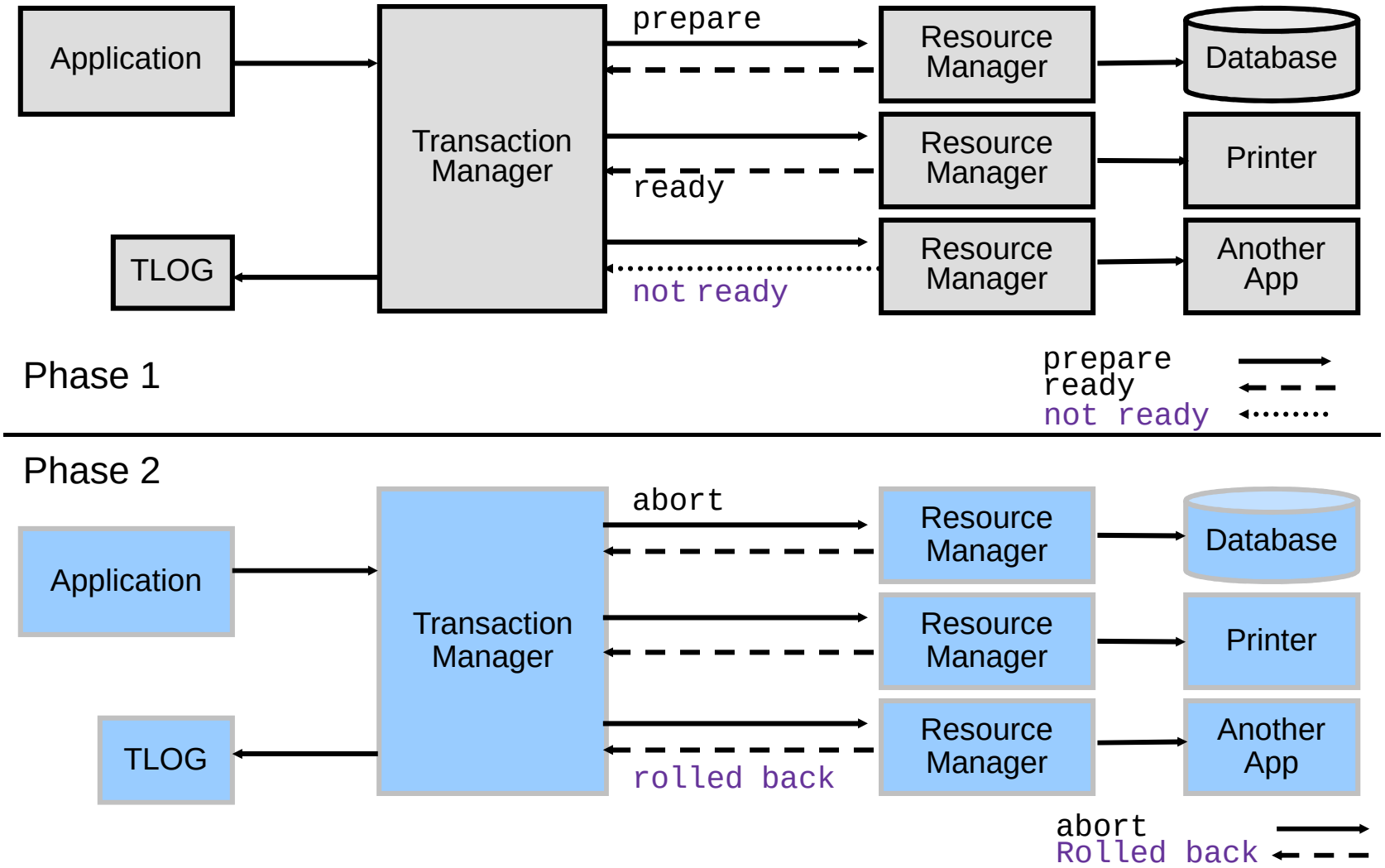
2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820706572666f726d616e63652e2044656c697665726564e2f4889676820707

- The Two-Phase Commit (2PC) protocol uses two steps to commit changes within a distributed transaction.
  - Phase 1 asks RMs to prepare to make the changes
  - Phase 2 asks RMs to commit and make the changes permanent, or to roll back the entire transaction
- A global transaction ID (XID) is used to track all changes associated with a distributed transaction.

Successful Two-Phase Commit



Unsuccessful Two-Phase Commit



Spring Transaction support

- Spring provides support for declarative transactions
- Delegates to a PlatformTransactionManager instance
  - DataSourceTransactionManager
  - HibernateTransactionManager
  - JdoTransactionManager
  - JtaTransactionManager

Programmatic Transaction

2f4889676820706572666f726d616e63652e2044656c6976657265642e2f4889676820706572666f726d616e63652e2044656c6976657265642e2f4889676820706572666f726d616e63652e2044656c6976657265642e2f4889676820706572666f726d616e63652e2044656c6976657265642e2f4889676820706572666f726d616e63652e2044656c6976657265642e2f4889676820707

- Transactions managed by the program
- Used only when partial code of the method is part of a transaction and to have full control through the code

- Resources used :
  - TransactionManager
  - TransactionStatus
  - TransactionDefinition

Programmatic Transaction example

```
PlatformTransactionManager transactionManager; // configure using datasource
TransactionDefinition def = new DefaultTransactionDefinition();
 TransactionStatus status = transactionManager.getTransaction(def);

try {
        jdbcTemplate.update( ……);
         transactionManager.commit(status);
     } catch (DataAccessException e) {
        System.out.println("Error in creating record, rolling back");
        transactionManager.rollback(status);
       }
```

Declarative Transaction

- Declarative transactions provide a very attractive alternative to the programmatic solutions
- Whole method participates in the transaction
- Use @Transactional Annotation to make the method transactional

- When the method executes successfully, transaction is automatically committed
- When the method throws any RunTimeException, transaction is rolledback

Declarative Transaction

- Define a transaction manager in XML file

```
<bean id="txManager“
   class="org.springframework.jdbc.datasource.DataSourceTransactionManager"/>
```

- Add this line to config

```
<tx:annotation-driven  transaction-manager="txManager"/>
```

- Add the @Transactional annotation to the method

```
@Transactional
public int insertMember(Member member) {
...
```

Transaction propogation

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- **REQUIRED:** the method must participate in a transaction. A new transaction will be started if one is not already active (default)
- **REQUIRES NEW:** A new transaction will always be started for this method. If there is an active transaction for the calling component, then that transaction is suspended
- **NOT SUPPORTED:** The method will not take part in any transactions. If there is one active in the calling component, then it is suspended while this method is processing. The suspended transaction is resumed once this method has completed.
- **SUPPORTS:** There is no requirement that this method should be executed in a transaction. If one is already started, then this method will take part in that transaction.
- **MANDATORY:** The calling component must already have an active transaction that this method will take part in.
- **NEVER:** This method is not participating in a transaction and it is also required that there is not an active transaction for the calling component.