# Spring Transaction Management

# Transaction management is something cross- cutting funtionality need to separate from buisness logic so used declative transaction management using spring aop concept.This is also one of the advantages over progamttic approach for transaction management.

ACID:

Atomicity : all or nothing rollback can be case.

Consistency: data shoud be consistent[after all credit/debit operation overall data to be consistent]

Isolation: transaction should not interleaved with each other.

Durablity: one all commit happen result should be stoder permenatntly nad next time on querying data its should give updated date.

Programmatic way: As we did in jdbc and hibernate

Declarative way: Spring AOP based:

Transaction attribute:

At a high level, **Spring creates proxies for all the classes annotated with @Transactional** – either on the class or on any of the methods. The proxy allows the framework to inject transactional logic before and after the method being invoked – mainly for **starting and committing the transaction**.

Propogation

Impact :

Isolation

Impact :

Timeout

Impact :

*ReadOnly*

This just serves as a hint for the actual transaction subsystem; it will not necessarily cause failure of write access attempts. A transaction manager which cannot interpret the read-only hint will not throw an exception when asked for a read-only transaction

*Ex:* **@Transactional( propagation = Propagation.SUPPORTS,readOnly = true )**

HibernateTransactionManager

JPATransactionManager

<bean id="txManager" class="org.springframework.orm.jpa.JpaTransactionManager">

   <property name="entityManagerFactory" ref="myEmf" />

</bean>

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

Query: How we are able to use hibernate specific fetature using EntityManagerFactory?