**🡪Spring Tx management:**

|-Ways of working with Spring Transactions

1. Programmatic Tx’s management mechanism

2. Declarative Tx management mechanism

3. Annotation Tx management mechanism

|-Purpose of EJB

|-Types of EJB

|-EJB is replaced by spring

In spring we will not use UserTransaction class or TrnsactionManager bcz these are coming from the JTA API so that it refers to the Impl classes of the Server vendors, rather spring provided API to work with Tx’s on top of the JTA. That means we will work with spring Tx classes and we will commit or rollback on spring classes. It will refers internally Jdbc, JTA, Hibernate, IBatis or anything.

**Ways of working with Spring Transactions:**

1. Programmatic Tx’s management mechanism

2. Declarative Tx management mechanism

3. Annotation Tx management mechanism

🡪In industry no one uses Programmatic Tx management mechanism, Most of the industry uses declarative mechanism and some sort of industry use Annotation mechanism.

🡪Spring is Frame work which will never replaces the EJB but most of the industry

Purpose of EJB:

EJB’s are meant for exposing the business logic over the n/w in distributed manner to other J2EE applications.

EJB is the acronym of Enterprise JavaBeans. EJB is the server side component that is used for constructing enterprise applications in a modular way. EJB is needed to incorporate the business logic of the enterprise application. EJB has a specification which forms part of Java Enterprise Edition specification. EJB are required to handle security, integrity, persistence, concurrency control and transaction processing in a typical way. Thus EJB allows the programmers to concentrate only on the functionality to be implemented. EJB is also required to support execution in an asynchronous way by using session beans and message driven beans.

🡪Actually spring not replaces the EJB rather

🡪Types of EJB:

🡪Majorly these are meant for distributing the business logic in distrusted manner over the network.

1. Session Bean

2. Entity Bean

3. Message Driven Bean (JMS API)

class OderProcessEjb {

public Status processOrder() {

try {

//Tx logic

} catch() {

}

}

}

🡪We need to write the business logic and Tx manager logic in the Ejb and this EJB will be used by the external world so that we need to place it in the Ejb container and J2EE Application server.

crete the ejb.jar

ejb-jar.xml

Write in this file info about the Ejb and url pattern

🡪If we wanted to get the Ejb then we need to connect to the JNDI-Registry so that another J2EE containers and in order to Tx the developer need no to write any code rather it will takes care by the underlying J2EE container if configure Ejb in the ejb-jar.xml which is called as Declarative Tx mechanism that is the reason huge adaption of EJB has happen bcz of the Tx management is easy.

But just bcz of Declarative Tx mechanism we need to adapt the EJB even though we wanted to expose it a external world in an distributed manner but later EJB 2.1 released where they made Declarative Tx management common bcz just for the sake Tx's peoples are using EJB even though they don't wanted expose EJB to external world so that EJB and Application server need communicate over the network or remotely even though they are part of same system that's where EJB developers realised 2-types EJB's

1. Local EJB's (Servlet will never call EJB's over the n/w rather servlet can call the Local EJB's directly)

2. Remote EJB's (Servlet can call the EJB's over the n/w using Remote EJB's)

🡪In order to avoid the problem of 2-containers For the sake of Declarative Tx's we need to use EJB-Container which is part of Application server that means we need must an Application sever but not a normal servlet container so that performance will be degraded so in order to avoid this need of Application server and we can any server web server without any application server done if we use spring which will improves the performance.

**EJB is replaced by spring:**

Why EJB is replaced by spring?

🡪 App servers written to support the EJB standard can, in theory, be ported from one compliant Java EE app server to another. But that means staying away from any and all vendor-specific extensions that lock you in to one vendor.

🡪Spring ports easily between app servers (e.g., WebLogic, Tomcat, JBOSS, etc.) because it doesn't depend on them.

🡪Spring encourages good OO design practices (e.g., interfaces, layers, separation of concerns) that benefit any problem they touch, even if you decide to switch to Guice or another DI framework.