EJB

ENTERPRISE JAVA BEANS

Det finns tre typer av bönor och en böna kan vara antingen eller.

De viktigaste fördelarna med EJB:er är att de gör det enklare att utveckla stora enterprise lösningar.

De kan bara köras på en riktigt *applikations server,*
som tillhandahåller servicear på system nivå. Transaktioner,
Loggning, Lastbalansering osv.

EJB containern tar han om hela livscyceln, så utvecklarna behöver inte ta hand om skapandet eller tömningen av objekten.

Session Bean

Session Beans bevarar datan för en specifik entitet under en *single session*. Den är antingen *@Stateful* eller *@Stateless*. Skillnaden är att Stateless inte behåller sitt minne efter att man stängt av programmet om samma användare sen loggar på igen.

Stateful gör då exakt motsatsen. Loggar samma användare på igen, öppnas just denna session, och minnet återskapas. Detta är väldigt andvändbart i Shoppingcarts.

Debatten går hög om Stateful egentligen borde finnas, men än så länge finns den. Dock är *@Stateless* nästan det enda som man träffar på i arbetslivet.

Entity Bean

@Entity beans tar hand om databas lagret. Det är hur databasens tabell kommer se ut, hur eventuella ID ska skapas. För Java utvecklare är detta en POJO, oftas bara en massa Getters och Setters, samt Konstruktorer.

Message Driven Bean

Message Driven Beans används enbart av JMS.

För att en MSB skall fungera krävs det en tabell i en databas, en kö som är registrerad i servern. Därefter måste en böna skapas med annoteringen *@MessageDriven*. Därefter låter man klassen använda sig av CDI och skapar en sändare som skickar ut meddelandet på kön.

I lyssnaren skapas en motagare med *@MessageDriven* som skickar vidare sitt meddelande till DAO.

JNDI

Java Naming and Directory Interface.
Ett gäng av API och Service interface. För EJB finns två typer:

- Binding
- Lookup

Session Beans är bundna i JNDI på två sätt:

local remote

Exception Handling

När ett fel inträffar i applikationen, så hanterar EJB containern detta. Den gör *inte* en Rollback, om det *inte* är specificerat i *@EJBContext.setRollBackOnly()*.

När ett fel inträffar på System nivå, gör EJB containern en Rollback och städar upp hela uppgiften som smällde. Den skriver in hela felet i en wrapper och kastar in i applikationen för att skicka vidare till klienten.

System felen kan ske när som, men i regel när en fråga misslyckas, eller man inte har öppnat sin EntityManager i tid för en fråga.

Istället för att fånga felen med *(Exception)* bör man fånga dem med *throw (EjbException) new EJBException(e).initCause(e);*

På klient sidan läser man av dem med *exception.getCause();*

Istället för att skicka ett Java genererat fel, bör man istället returnera en status kod, och logga felet i en seperat databastabell eller JEE applikation.