**Deklaratív tranzakciókezelés**

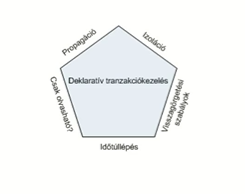
**(programozott(=mi írjuk a begint, commitot és rollbacket)- és deklaratív konfigurálás(=mi konfigurálunk annotációkkal a keretrendszer végrehajt); tranzakciókezelési szempontok: propaggáció: ha metódus metódust hív, az adatbázisműveleteik egy tranziban vagy külön legyenek? izoláció, visszagörgetés, timeout; propagáció default:bean1hez generált proxy(@Transactional alapján), aminek feladata az indítás és lezárás, a bean2 művelete bekapcsolódik a a bean1 tranzijába meghíváskor, visszatéréskor a bean1 proxyja tudja, hogy majd neki kell lezárni; required propagációs tulajdonság: ha van tranzi, csatlakozik, ha nincs, létrehoz egyet; további propagációs tulajdonságok; izolációs problémák és szintek(dirty-, non-repetable-, phantom read; serializable izolációs szint mindháromra megoldás: egymás után futtatja a tranzikat; minél magasabb izoláció annál több mem és lassabb; visszagörgetés: kivétel alapján a konténer dönt a rollbackről; timeout megadása tranzinál: kivétel dobás lesz; csak olvasható tranzakciók(=nem megy le érte az adatbázisba, hanem memben tárolja) cashelési céllal)**

hogyan lehet a **tranzakciókezelést konfigurálni?**

2 mód:

* *programozott mód;*. mi írjuk a begint, commitot és rollbacket
* *deklaratív mód*: mi csak konfiguráljuk, és az alkalmazásserver fogja kiadni a megfelelő metódushívásokat

**Tranzakciókezelés**

****

*Propagáció*

**Ha különböző metódusok hívják egymást** és mindegyik valamilyen adatbázis műveletet végez, akkor ezek **egy tranzakcióba kerüljenek vagy külön tranzakciókba**?

*Izoláció*

Párhuzamos használat esetén izolációs problémák lehetnek; ezek megszűntetése

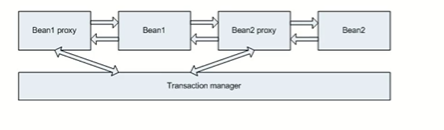
*Visszagörgetési szabályok*

**deklaratívnál mi csak jelezzük a hogyant, de nem mi görgetjük vissza;** tipikusan valamilyen exceptionnel tudjuk megadni

*Időtúllépés, timeout*

+lehet indítani ún. csak olvasható tranzakciókat

**Propagáció**



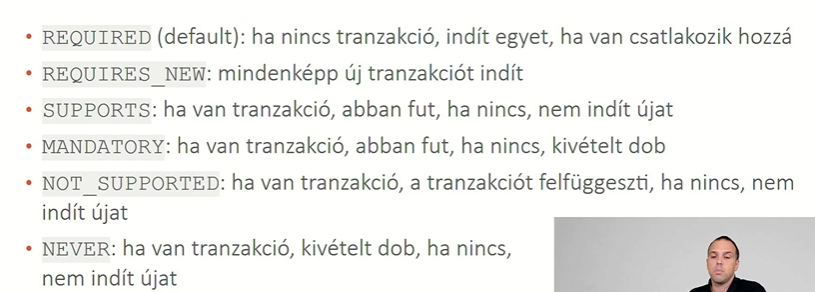
Bean1ben és Bean2ben is vannak műveletek; egy tranziba vagy külön? **alapesetben egyik bean adatbázisművelete által elindított tranzakcióba kerüljön a meghívott másik bean művelete is, csatlakozzon oda be;** begin comit és rollback úgy jön létre, hogy a bean példányhoz a keretrendszer generál egy proxyt, amelyen keresztül történik a hívás és feladata ennek a tranzakció indítás és lezárás**; bean1 proxija csatlakozik a transactionmanagerhez és elindítja a proxyt**, majd **továbbítja a hívást a beannek** (onnan tudja magáról, hogy tranzakciót kell neki indítani, hogy szerepel rajta a **@Transactional annotáció**); majd a bean2 proxy megvizsgálja, hogy van-e **tranzakció, ha van**, **csatlakozik hozzá;** **visszatéréskor a bean1 proxyja tudni fogja, hogy neki is kell lezárnia és döntenie kell a commitról vagy a rollbackről**

* ez az alapértelmezett működés = **required propagációs tulajdonság;** **egy adott beannek egy metódusa csatlakozik a tranzakcióhoz, ha már megvan nyitva egy; ha nem volt, ő maga indít el egyet**

**@Transactional akkor kell a metódusra, ha olyan metódust hív, aminek végrehajtásához is kell tranzakció és így a műveletek egy tranziba kerülnek; osztályra tenni jó gyakorlat?**

**További propagációs tulajdonságok**

ezeket lehet még megadni **attribútum értéknek a @Transactional annotációnak**

****

utolsó 4 csak assertre, hogy ellenőrizzük, ki hívta meg a metódust

**Izoláció**

*Izolációs problémák*

dirty read

ha fut egy tranzi annak belső **állapotát egy másik tranzi tudja olvasni**; ilyennel nem szoktunk találkozni**, adatbázisok alapból kivédik**

non-repeatable read

ha kiolvasunk egy rekordot a **tranzakció elején és a végén**, **nem ugyanazt kapjuk** (pl. ha lassú a futás?)

phantom read

előző speciális esete; **nem egy rekord hanem pl egy select eredménye lesz más számosságú a tranzi elején és végén**; első és második olvasás **között lefutott egy gyors tranzakció comittal** ami updatelt valamit

*Izolációs szintek*

ezek megfelelő állításával lehet az előbbi problémákat kiküszöbölni

*read uncommited* – legalacsonyabb szint

*read commited* – megoldja a dirty read problémát

*repeatable read* – megoldja a non repetable readet is

***serializable***– egymás után futtatja le a tranzikat, és ezért nem jelentkeznek izolációs problémák, **mindháromra megoldás**

állítgatni ezeket nem nagyon kell, **alapértelmezett beállításokkal** az alkalmazásunk működni fog

**de minél magasabb az izolációs szint, annál lassabb lesz a működés**, az adatbázisnak annál több adminisztrációs feladatot kell végeznie és memet használnia

**Visszagörgetési szabályok**

mikor van rollback?

**kivételek alapján történik**; spring és ee-be is

egyik esetben kivételdobásra is mehessen a commit, többinél mindenképp legyen rollback

**nem mi adjuk ki a commitot és rollbacket, hanem a kivétel alapján a konténer fog dönteni**

**Timeout**

Ezt is meg lehet adni tranzakciónál

tranzakcióban egy **timeoutnál** a JPA implementáció **kivételt** fog dobni

**Csak olvasható tranzakció**

Spring esetén **casheléseket, gyorsító folyamatokat** tud a keretrendszer a háttérben elvégezni a **memóriában bizonyos entitások tárolásával;** **nem megy le az adatbázisba, hanem kiolvassa a memből;** ha ilyen egy tranzi, akkor ez jelzi, hogy erre van lehetőség

**Deklaratív tranzakciókezelés - gyakorlat**

**@EnableTransactionManagement** main()t tartalmazó osztályra

Dao-ban (=@Repositorys osztály) a metóduson @Transactional

1. Springes project (jpa-transaction); maines osztályra @EnableTransactionManagement
2. Entitás létrehozása @Entityvel, Lombokkal
3. Repository/DAO osztály: private entitymanager atri @Persistencecontext-el, metódusra @Transactional, ami módosít és kell neki tranzi

**Kérdések**

*Hogyan lehet a tranzakciókezelést konfigurálni?*

*Milyen szabályai, tulajdonságai vannak a tranzakcióknak?*

*Metódusokon átívelő tranzakciónál hogy érjük el, hogy a műveletek egy tranzakcióba kerüljenek?*

*Mi az a required propagációs tulajdonság és Spring beaneknél hogy működik?*

*Milyen további propagációs tulajdonságokat ismersz?*

*Milyen izolációs problémákat ismersz?*

*Az előbbi problémákra mely izolációs szintek nyújtanak megoldás?*

*Milyen előnyőkkel és hátrányokkal jár az izolációs szint növelése?*

*Mi alapján történik a rollback? Spring esetén hogy kerül meghívásra a commit() és a rollback()?*

*Megadhatunk-e timeoutot trakzakcióknál? Ha igen, mi fog történni?*

*Mik azok a csak olvasható tranzakciók?*