## Architektúra és integrációs tesztelés

## Architektúra

## (3 réhegű webes alkalmazás, perzisztens réteg, komponensek = DAO, feladatuk =CRUD műveletek; kiszervezés DAO osztályba; Egy DAO-egy entitás

## Architektúra: DAO osztály, EntityManagerFactory attribútum konstruktor DI-vel, metódusokban EntityManager létrehozása EntityFactoryból(=.createEntityManager() -> létrejön egy persistence context = munkamenet, tranzakció indítása begin()-el, majd persist(), commit() és close())

## *A JPA hogyan illeszkedik bele egy alkalmazás architektúrájába?*

alkalmazások manapság többrétegű webes alkalmazások; 3 réteg:

**legalsó – perzisztenciáért felelős**

középső az üzleti logikáért felelős

felső a felhasználói felület réteg, amire még esetleg jön egy webszolgáltatás réteg is

**Az adatbázissal a kapcsolatot tartó rétegben elhelyezkedő komponenseket JEE tervezési mintha alapján DAO-nak nevezzük; ezek végzik az adatbázis műveleteket, tipikusan CRUD műveleteket.**

**DAO**

Adatbázisműveletek kiszervezési külön osztályba

**tipikusan egy DAO egy entitással foglalkozik**

**DAO osztály EntityManagerFactory atrival,** amit konstrujtor injectionnel, kívülről kap

**konkrét DAO metódusban: EntityManager létrehozása a factoryból**; ezáltal **létrejön egy munkamenet, egy persistence context**, amiben majd dolgozni fogunk

**Elindítjuk a tranzakciót beginnel, majd persist, commit és végül close**



**Integrációs tesztelés**

**Teszt osztály, benne DAO példány, @Before és DAO init EntityManagerFactory példánnyal(Persistence.createEnttyManagerFactory(”pu”); konrét teszt metódus: új entitás memóriába, lementés, betöltés find()-al, majd assert**

szükségünk lesz egy **DAO példányra**; ezt a **teszt osztályban atriként** deklaráljuk

**@Before-**al minden teszteset előtt **inicializáljuk** az atrit

a teszt **metódusban** entitás **létrehozása**, **save** majd betöltés **find()**-al és **assertequals**



## Architektúra és integrációs tesztelés – gyakorlat

## *DAO réteg felépítése*

## előzmények:meglévő Employee osztály, meglévő persistence.xml, flyway migrációs utasítás a db\_migrationben; pom xml függőségek:hibernate, flyway, jdbc driver és jUNit

## 

1. *EmployeeDAo osztály létrehozása;*

EntityManagerFactory + DI

*void save(Employee)*; Entitymanager, begin, persist, commit, close

*Employee findById(Long id)*; Entitymanager, find

1. *EmployeeDaoTest*
   1. *EmployeeDao atri, @Before init()* -> DataSource, Flyway; clean és migrate,

EntityManagerFactory létrehozás, majd EmployeeDao incializálás

* 1. *void testSaveThenFind()* -> new Employee, dao.save(xx), getId, another =

findById(), majd assert a névre

**flyway használata esetén felesleges megadni az xmlben a drop and create propertyt, ezért kikommenteltük**(mert a flyway fogja létrehozni és eldobni a táblákat)

* 1. *List<Emp> listAll()- DAO osztályba* először új metódus -> Entitymanager,

createquery()select e from Employee e order by e.name)y, getResultList(), close és return

erre *test: testSaveThenListAll();* 2 entitás lementése majd lekérdezése névre, streammel

* 1. *void changeName(id, name) DAOba*; Entytymanager, begin, .find, .setname(); **(nincs külön lementés, jpa automaitkusan rögzíti, milyen változások történnek az entitáson, commitnál összegzi a változásokat és updatet ad ki),** végül commit és close
  2. erre test: *testChangeName();* save, getid, changeName, find, assert
  3. *void delete(id) – DAOba*; entityManager, begin, .**getReference(Employee.class, id)**, .remove(employee), .commit és close;
  4. majd *test erre*, save, delete, mad a listAllra adjon emptyt

**Commit()-nál kerülnek végrehajtásra az összegyűjtött műveletek!**

**Kérdések**

*Hogy épül fel egy 3 rétegű webes alkalmazás?*

*Hogy hívjuk az adatbázissal kapcsolatot tartó réteg komponenseit JEE tervezési minta szerint?*

*Hogy épül fel egy ilyen osztály?*

*Integrációs tesztnél hogy épül fle a teszt osztály?*