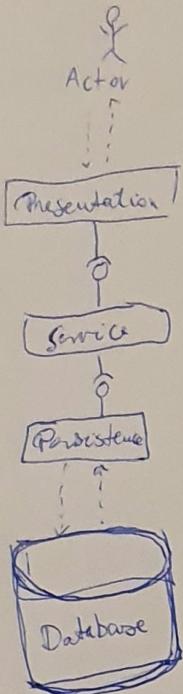


## 1) Spring framework szer modellje

- modul megnevezés + funkciók

CA'SPA'R ANDREA  
WERNIC

- Mi az IoC és DI. IoC szerepe összehasonlítva ha engedélyezés nélkül minden adatot a rendszertől való kiszolgáláshoz vezetével. DI implementálásának előnyeit.



Presentation: Feladata a felhasználói interfejsz vezérlése, a szolgáltatótól származó adatok megjelenítése, valamint a felhasználók interakcióit továbbítása a szolgáltatóhoz. A "nyers" adatok validálása is feldöntés. (Helyesírás az adatok nem tartalmazza a hibásnak adott.)

Service: A szolgáltatás (service) részleg végezi a műveletet, íht le a rendszertől implementálásra az üzleti logika. A persistence résztől kapott adatokkal dolgozik, feldolgozza, transzformálja, majd továbbítja a presentation részleg felé.

Persistence: Az alkalmazás persistenciájának felelős, ez a rész kommunikál az adatbázissal valamint az állománydal. Primitív tránszakciók műveletei vanak az adatbázisban, a fülötte lévő résznek kell választania.

## IoC: Inversion of Control (megfordítása a kontrollnak)

Olyan folyamat, amelyben egy objektum meghatározza függőségeit a többi, hogy lehessen kiadni azokat.

Olyan programozási technika, amelynek lényege, hogy a hívás nem a mezőkön keresztül. (Hollywood elvnek is szintén hívni a "Ne hív minden, csak mi hívunk téged!" miatt.) Emellett ezzel speciális eset az, amikor nem az az osztály hívja ki a példányt, hanem hívását kaphja meg.

## DI: Dependency Injection (függőség befelekezelése)

Egy objektum ezzel meghatározta függőségeit elérheti.

Ez keresési minden, teljesen kieppen az IoC ezzel speciális esete. Lényege, hogy az objektum mindenről hagyja meg azt a másik objektum példányt, amivel előbbi dolgozni akar.

Tehát mindenekkel való (delektál) lehetséget jelent. → Általában deklarált típusú objektumra van szüksége, de nem állíthat be (előzők nélkül).

Hosszú időn át a hívás belövével lezár, és ezt biztosítja azt, hogy minden az első hívás belövével lezár, és rendelkezésre álljan. Példával a használó el.

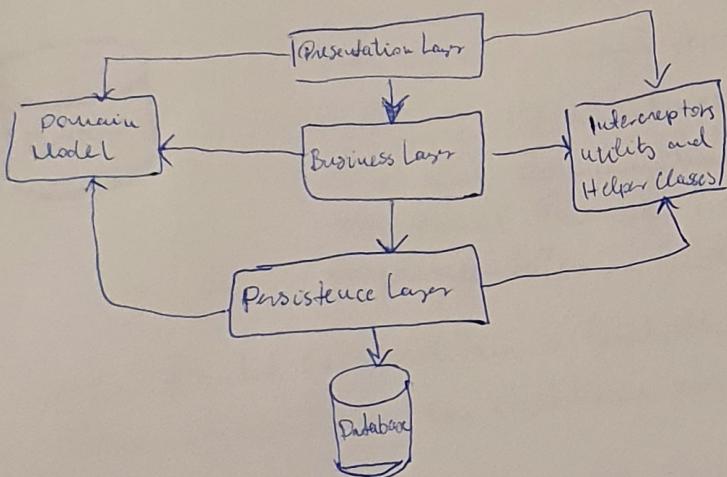
4) JPA, heretrendezés, osztályok, funkciók

GÁSPÁR ANDRÉA  
WERNER  
2021.01.28.

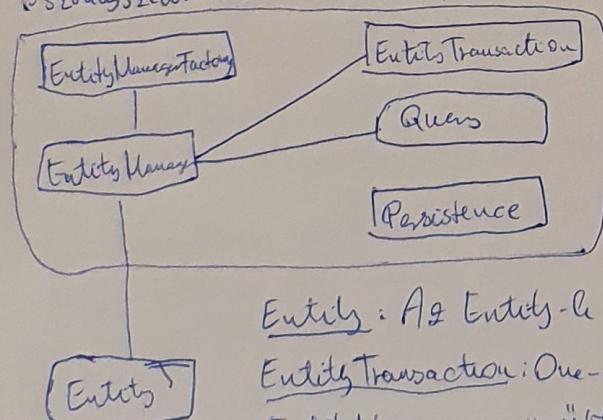
Egyesítsük a tableikhez kapcsolódó leírásokat a műveletekkel.

Külön (JPA) repositoryt hozhatunk létre a tableknak, előre definiált interfésszel, mely beszúrja például a GundRepository (REST frissítések). Ezután a frissítést automatikusan beszüllíthetjük a memóriahez, vagy tárolásra készíthetjük. Azt (elnevezés) konvencióit betartva, használunk SQL leírásokat.

- count() - entitások száma
- delete(T entity) - paraméterként kapott entity törlése
- deleteAll() - összes entitás törlése
- deleteByID(ID id) - paraméterként kapott id-jú entitás törlése
- findAll() - összes entitás lekérdezése
- findById(ID id) - paraméterként kapott id-jú entitás megtalálása, felidelezése
- save(S entity) - entitás mentése.



Osztályzott architektúra, class relationship



Entity Manager Factory: Az EntityManager factory osztálya. Létrehozza és menedzseli az EntityManager példányokat.

EntityManager: Interface, amely menedzseli a persistens műveleteket, az objektumokon. Nagyon hasonlít, mint egy factory (Proxy) a Query példányok.

Entity: Az Entity-le persistens objektumok, reldáblán valahol tárolva a DB-be.

Entity Transaction: One-to-One lehetsége van az EntityManager-rel. minden EntityManager műveletet az EntityTransaction által van végezhetve.

Persistence: Statisztus metódusokat tartalmaz az EntityManagerFactory példányhoz.

Query: Ez az interface a JPA-implementációk által a JPA megalártól a megfelelő relációs objektumok megtalálásához.