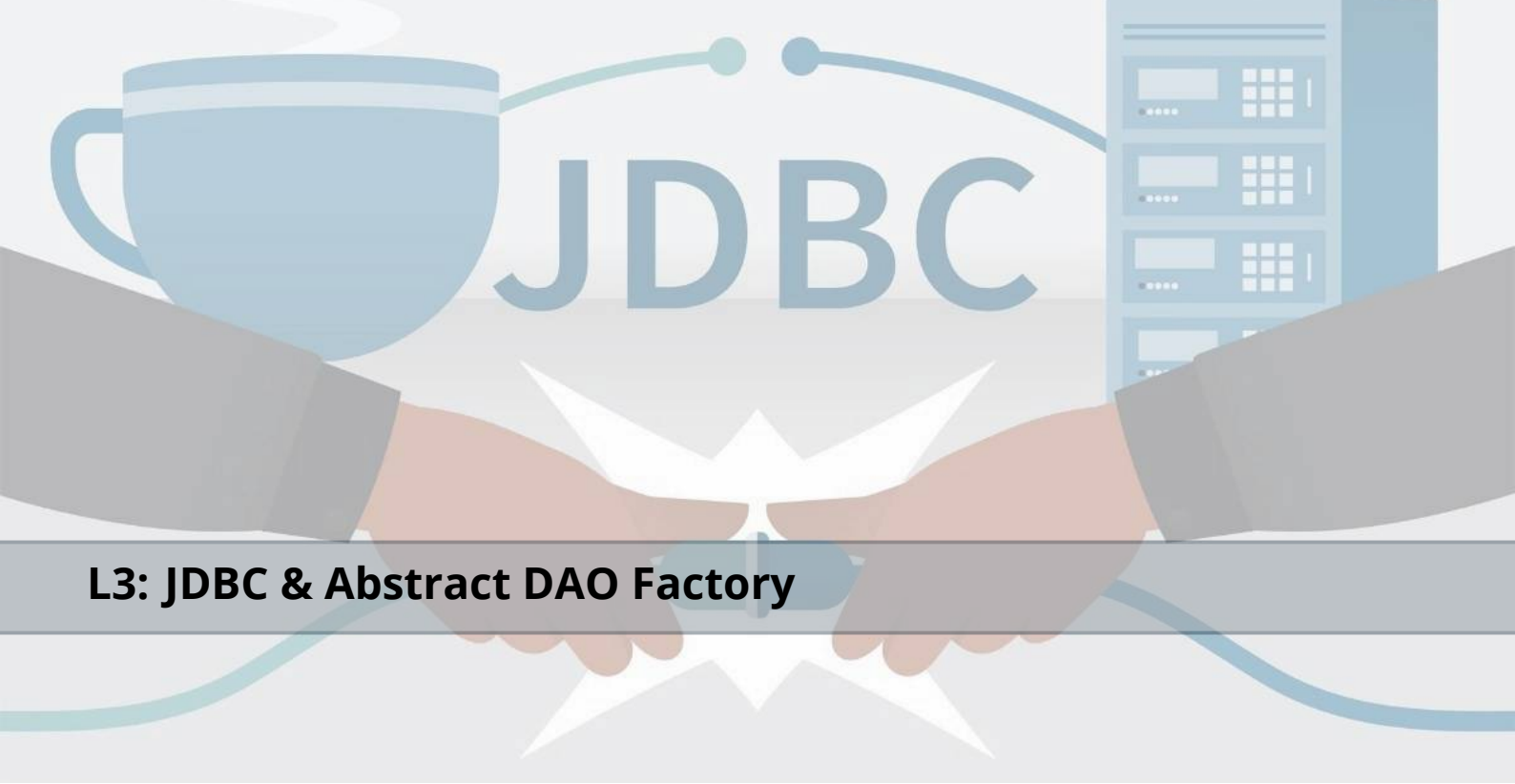

L3: JDBC & Abstract DAO Factory

Adatelérési réteg JDBC-vel & Abstract DAO Factory

- A jelen laborfeladatban egy konvencionális és kiterjeszthető adatelérési réteget építünk be a meglévő webalkalmazásba.
- A laborfeladatok továbbra is használják az első feladatban előírt projekttematikát (ld. `ubb-idde-lab1-build.pdf`).
- Továbbra is feltétel, hogy ne legyenek statikus kódelemző jelzések, illetve hogy helyesen fusson le a `check` taszk a folyamatos integrációban (ld. korábbi laborfeladatok leírása).
- Az alábbi leírások általánosan érvényesek mindenkire, de mindenki alkalmazza őket azokra az entitásokra/funkcionalitásokra, melyek már léteznek a korábbi laborokról. Ha valahol az általános leírás nem tisztán alkalmazható a meglévő projektre, kérdezzétek a laboránsokat tisztázásért.

Környezeti elemzés

Környezeti elemzés

- Végezzünk *környezeti elemzést* a megadott projekten, s vezessünk be (ha még nincs) **összesen legalább 2** entitást. A két entitás álljon valamilyen kapcsolatban egymással (egy-a-többhöz, több-a-többhöz, stb.).
- Kövessük a kurzuson bemutatott osztály- és csomagstruktúrát:
  - A modellosztályok kapjanak saját `model` alcsomagot a meglévő struktúrában (pl. `edu.bbte.abcd1234.backend.model`).
  - A modellosztályoknak legyen egy közös szülőosztályuk, ahol az elsődleges kulcsot tároljuk.
  - A modellosztályok kövessék a *Java Bean* specifikációt–ne tartalmazzanak üzleti logikát.
  - Használjunk *Lombok*ot vagy Java >=14 `record`okat a boilerplate kód automatikus generálásáért.

Adatelérési réteg

Adatelérési réteg JDBC-vel

Építsünk fel egy *adatelérési réteget*, mely JDBC segítségével végzi a CRUD műveleteket a fent készített 2 entitáson.

- Konfiguráljunk egy **relációs** adatbázist az alkalmazásunkból történő használatra (lehet bármely típusú AB, amelyhez van JDBC Driver támogatás, **de legyen különálló folyamat, s nem in-memory**).
- Kövessük a kurzuson bemutatott osztály- és csomagstruktúrát.
  - Az adatelérési osztályok kapjanak saját `dao` vagy `repo` alcsomagot a meglévő struktúrában (pl. `edu.bbte.abcd1234.backend.repo`).
  - Az adatelérési osztályok használjanak megfelelő szintű absztrakciót: legyen **külön** absztrakció a kezelt entitástípus, s az elérési mód fölött. Használjunk Reflection API-t, ha szükséges.
  - A következő CRUD műveletek legyenek implementálva mindkét entitásra: `findAll`, `findById`, `create`, `update`, `delete`, `findByXXX` (ahol az `XXX` reprezentál egy entitás-specifikus nem-ID adattagot).
  - Használjunk *naplózást* legalább az írási műveletek és a hibák esetén.
- A JDBC kapcsolathoz szükséges drivert `runtimeOnly` konfigurációjú Gradle függőség segítségével kössük be a projektbe.
- A JDBC kapcsolatok esetén használjuk a **connection pooling** mintát. Használhatunk saját implementációt, vagy az általános `HikariCP` külső csomagot.
- Implementáljuk az **Abstract DAO Factory** tervezési mintát a backenden. Ahol a frontenden használjuk a DAO osztály(oka)t, a factory-tól kérjünk példányt, ne példányosítsunk kézzel (a kliens ne is használja az implementáció osztályt, csak az interfészt).
- *Megjegyzés*: Az adatbázis futhat egy lokális környezetben, a Dockerizálása nem kötelező.

Konfiguráció

Konfiguráció

- Vezessünk be **általános konfigurációs mechanizmust** az alkalmazásunkba. A rendszer használjon egy gazdag formátumot (pl. JSON, XML, YAML), melyet olvassunk be egy (vagy több) általunk definiált POJO-ba (ajánlott a JackSON könyvtár használata) a classpath-ről.
- Használjuk ezen konfigurációs beant minden változtatható konfigurációra, pl. dönteni DAO osztályok között, JDBC kapcsolati beállításai (URL, driver-név, felhasználónév, jelszó), a connection pool mérete, stb.
- Adjuk meg a lehetőséget **profilonkénti** szétválasztásra. Támogassunk legalább 2 profilt, melyből az egyik a memóriában tárolt adatokat használja, míg a másik a JDBC-s implementációt. Profilok között lehessen váltani környezeti változó és/vagy JVM property szerint.

Feltöltés

- A feladatot töltsük fel a saját git tárolónkra, egy **dedikált ágra**. Az ág neve tartalmazza a laborfeladat sorszámát. További információk a `ubb-idde-lab0-setup.pdf` állományban.
- Hozzunk létre egy **merge request**et, amely tartalmazza a laborfeladat számát, állítsuk a merge requestet a jelen feladathoz tartozó **csoportos milestone-ra**, majd linkjét **adjuk le Canvasen**. Végső leadási időpontnak tekintjük az utolsó commit, az utolsó push, és a Canvasre való linkfeltöltés közül a **legkésőbbi** mozzanatot.