**Integrációs tesztelés**

**(adatbázis réteg bevonásával már integrációs a teszt; SpringBootban sok lehetőség hogy rétegenként vagy egybe; csak repository teszteléshez: @DataJpaTest- és injektálások, H2vel dolgozik; minden teszt saját tranziban és a végén rollbackel; teszt mit ellenőríz: jók-e az annotációk, metódusnevek, saját jpql-ek; teljes alkalmazás teszteléséhez: @SpringBootTest**

A tesztbe adatbázis réteg bevonása: már nem unit, hanem integrációs teszt

**SpringBoot sok lehetőséget biztosít, hogy rétegenként, vagy akár egybe teszteljünk**

ha **csak repository réteget akarunk tesztelni**: akkor a tesztünkön **@DataJpaTest annotáció: csak jPA rep réteget fogja tesztelni**, csak azt indítja el; **ilyenkor H2vel dolgozik;** **saját Repository, Datasource, JdbcTemplate, entityManager injektálása**

**minden teszt metódus saját tranzakcióban indul el, végén rollbackkel,** ezáltal nem marad bent adat az adatbázis végén a test lefutása után

JpaRepository tesztelése: azt teszteljük vele, hogy **jól helyeztük-e el az entitáson az annotációkat, jól írtuk-e meg a metódusneveket**, ahol az implementációt a jpa spring data generálja**, jól írtuk-e meg a jpql lekérdezéseket**?

**DataJPaTest példa**

**minden teszt eset után rollback-el, így pucolja le magát!**

nem csak azért integrációs, mert adatbázis használ, hanem azért is, mert két repository metódust is meghív



**@SpringBootTest használata**

**a teljes alkalmazás teszteléséhez**; összes réteget(servicet és controllert is) elindítja

**érdemes** ilyenkor **már valódi adatbázissal** (amit productionben is használ fogunk) dolgozni

fontos foglalkozunk ilyenkor azzal is, hogy **létrejöjjön az adatbázis séma is,** és **a teszteset töltse fel adatokkal az adatbázist**

**Tesztek H2 adatbázisban**

Ha felül akarjuk írni az **adatbázis elérést a teszteseket futása során,** akkor az **application.propertiesben adhatjuk meg az adatbázis elérését;** ekkor nem a main ágon lévő applicaiton.properties kapcsolók fognak érvényesülni, hanem a teszt ágon lévők; **példa h2-s elérésre:**



**Séma inicializáció**

Döntenünk kell arról is, hogy mi hozza létre a táblákat?

Elvégezheti maga a JPA implementáció is; hibernate alapeset: create-drop

**de érdemes sémaincializációs eszközt használni** (Flyway, Liquibase)

JPA, tulajdonsága, hogyha a classpathon van schema.sql file, azt lefuttatja: ezzel fel lehet kezdeti adatokkal tölteni az adatbázist

**Adatfeltöltés**

**classpathon lévő data.sql állomány alapján vagy tesztosztályon @Sql annotációval, vagy programozott módon: @BeforeEach és metódussal vagy akár REST apin keresztül, vagy JdbcTemplate**

*Hogyan lehet a kezdeti adatokat feltölteni?*

**ha lefuttatunk egy tesztesetet, ami a teljes adatbázist meghajtja,** **szükségünk lehet kezdeti adatokra**, amin a teszteset dolgozni fog

erre egyik megoldás: **classpathon data.sql állomány**, amit lefuttat, amikor a teszteset elindul

sql utikat lefuttathatunk úgy, is, hogy a **teszt osztályon elhelyezünk egy @Sql annotációt;** **ilyenkor minden metódus előtt ezt az sqlt lefuttatja**

ha **programozott módon** akarjuk a feltöltést:

* BeforeEach, AfterEach-es **metódusokban**, akár publikus APin, **rest webszolgáltatásokon keresztül** az adatok inicializálására; de **beinjektálhatjuk az alkalmazás egy komponensét is**, es direktbe, a megfelelő rétegeket kikerülve tudunk az adatbázisba adatokat juttatni; de pl. egy JdbcTemplate injektálásával közvetlenül hozzácsatlakozhatunk az adatbázishoz is

**Tesztek egymásra hatása**

**a tesztesetek közötti állapotátmenetet kezelése; rendrakás eset előtt vagy után? teljes séma törlésével újra létrehozásával (kevés munka, de lassú), vagy külsőleg teljes adatbázis import, vagy a teszteset mindig külön adatkörrel dolgozzon (gyors de sok munka)**

Legnagyobb fejtörést az adatbázisműveletekkel kapcsolatos integrációs teszteknél az okozza, hogy két **teszt metódus futása között van állapotátmenet;** egyik metódus betesz valamit az adatbázisba, azt a másik látni fogja; nem függetlenek; hogyan kell ezt kezelni?

*Dilemmák:*

**a teszteset maga előtt vagy után rakjon rendet?** az utánt hajlamosak vagyunk elfelejteni

Dönthetünk úgy is, hogy **mindig a teljes sémát/adatbázist letöröljük és újra létrehozzuk, tiszta de lassú megoldás**

**Létezik külsőleg teljes adatbázis import és megfelelő állapotba hozás is**

Vagy figyelünk arra, hogy csak az adott teszteset által használt táblákat töröljük/kreáljuk újra, de ez macerásabb

**leggyorsabb: nem inicializáljuk újra mindig az adatokat, hanem a tesztesetek mindig külön adatkörrel dolgozzon**: létrehozza a tesztadatot, amit utána használ és assertel; körülményes sok munkával jár

**mindig újrainicializálás a legkevesebb munka**, viszont lassú futás

**Integrációs tesztelés – gyakorlat - H2**

A repositorym egy interface, amire nem írunk tesztet; azt teszteljük, hogy az entitásomra jól raktuk-e rá az annotációkat

1. pom.xml: h2 függőség tes scopeal(a classpathon csak a tesztesetek futtatásakor lesz rajta)
2. EmployeesRepositoryTest? osztály létrehozása

@DataJpaTest annotáció az osztályra és @Autowired employeerepository

1. void testpersist(); new Employee(), save() majd findAll()

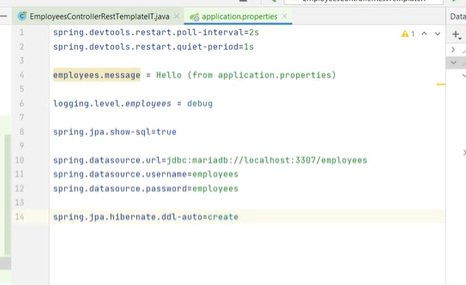
assertThat nevekre és containsexacly

futtatás; elindít egy H2et, legyárt egy repositoryt, ment, listáz, assert; log: starting embedded database és sql műveletek

**Integrációs tesztelés – gyakorlat – MariaDb**

**Teljes adatbázis meghajtásával teszt alatt ahhoz az adatbázishoz nem érdemes csatlakozni, amit az alkalmazás is használ,** mert a drop-and create hatására elvesznek az adatok; csatlakozzunk egy újhoz = indítsunk és konfiguráljunk egy újat és a tesztek ehhez csatlakozzanak

1. **applicaiton properties átmásolása teszt ágon a reosurceba és értékek átírása;** port alapértelmezett 3306osról 3307esre és csak create legyen, hogy a tesztesetek lefutása után tudjuk ellenőrizni, hogy mi van az adatbázisban



1. új adatbázis elindítása dockerben: a teszt adatbázis a sajátgépem 3307eseén legyen elérhető:



1. MariaDb kapcsolat felvétele (IDEAban Database-el, figyelni a portra:3307)
2. EmployeesControllerRestTemplateIt osztály tesztjeinek elindítása

sikeres lefutás esetén mert repeated(2) a test két alkalmazottat kell látni; az ideas konzolban select \* from employees lekérdezés

1. **a tesztmetódusban a deleteAllEmployees() eltávolítása, atriként injektált service eltávolítása; az osztályra SQL annotáció (statements =”delete from employees)**

**újrafuttatás**

**Kérdések**

*Adatbázisréteg bevonásával milyen tesztjeink leszek?*

*Milyen lehetőségek vannak a springbootban a repository réteg tesztelésére?*

*@DataJpaTest használata esetén mit használ adatbázisnak a Spring? Milyne konfigok kellenek hozzá? Hogyan működik ezt a tesztosztály és mit tesztelünk vele?*

*@SpringBootTest esetén milyen adatbázist használunk? Mivel fontos még ilyenkor foglalkoznunk?*

*Hol tudjuk felülírni az adatbázis elérést a tesztek futása során?*

*Hogyan történhet a séma inicializáció?*

*Hogyan lehet a kezdeti adatokat feltölteni?*

*Milyne dilemmákba ütközhetünk a esztek egymásr ahatásávla kapcsolatban?*

*Milyen módjai vannak az adatbzási megfelelő állapotba hozásának? Mik az előnyők, hátrányok?*