JAVAX-SPR - Alkalmazásfejlesztés Spring keretrendszerrel tanfolyam

Inversion of Control és Dependency injection

Feladatként egy kedvenc helyeket nyilvántartó alkalmazást kell fejleszteni. Egy kedvenc helyet a *Location* osztály reprezentál. Rendelkezik egy azonosítóval, névvel és két koordinátával (rendre *Long id*, *String name*, *double lat*, *double lon*).

Hozz létre egy új projektet *Locations* néven.

Létre kell hozni egy *LocationsService* és *LocationsDao* osztályt. A *LocationsDao* osztály a *Location* példányokat egy belső listában tárolja.

A *LocationsService* metódusai:

- *List<Location> listLocations()* - összes kedvenc hely listázása
- *void createLocation(String name, double lat, double lon)* - kedvenc hely létrehozása
- *getLocationById(long id)* - kedvenc hely keresése id alapján
- *void updateLocation(long id, String name, double lat, double lon)* - kedvenc hely módosítása id alapján
- *deleteLocation(long id)* - kedvenc hely törlése

A *LocationDao* metódusai rendre:

- *List<Location> findAll()* - összes kedvenc hely listázása
- *void save(String name, double lat, double lon)* - kedvenc hely létrehozása
- *findById(long id)* - kedvenc hely keresése id alapján
- *void update(long id, String name, double lat, double lon)* - kedvenc hely módosítása id alapján
- *delete(long id)* - kedvenc hely törlése

A *LocationService* delegálja a kéréseket a *LocationDao* osztálynak. Legyen konstruktor injection.

Írj egy *LocationMain* osztályt, mely példányosít egy *ApplicationContext* példány, lekéri a *LocationService* beant, majd meghívja rajta a fenti metódusokat.

Unit és integrációs tesztelés Spring környezetben

Írj egy unit tesztet a *LocationService listLocations()* metódusára!

Írj egy integrációs tesztet, mely az összes metódust teszteli. Írj egy *deleteAll* metódust a *LocationsDao*-ba, és hívd meg minden teszt metódus előtt. Ez kitörli a lista tartalmát.

Beanek személyre szabása

Hozz létre egy *Location* beant az Application Contextben, melynek típusa scope-ja prototype. Legyen a neve *Choose name*, legyen a koordinátája *47,50, 19,05*. Legyen a neve *templateLocation*

A *LocationsService*-ben hozz létre egy *createLocationTemplate()* metódust, mely létrehoz mindig egy új példányt az előbbi prototype beanből. Ehhez az *ApplicationContext*-et kell a service-be injektálni, majd a *getBean* metódusát hívni.

Ellenőrizd integrációs tesztesetben, hogy két egymás után létrehozott példány tényleg nem ugyanaz!

Konfiguráció XML-lel és annotációval

Módosítsd úgy az alkalmazást, hogy a *LocationsService* és *LocationsDao* osztályokon sterotype annotáció legyen, valamint component scannel legyenek felolvasva!

Injektálás konfigurálása

Emeld ki a *LocationDao* metódusait egy külön interfészbe, neve legyen *LocationDao*. Ennek legyen az eredeti osztály az implementációja, neve legyen *ListLocationDao*. Írj egy *DummyLocationDao* osztályt is, mely implementálja a *LocationDao* interfészt. Milyen hibaüzenet jön indításkor? Milyen annotációval kell ellátni a *ListLocationDao* osztályt, hogy mindig az kerüljön injektálásra?

Eseménykezelés

Amennyiben módosítás történik, a *LocationsService* dobjon egy *LocationHasChangedEvent* objektumot, melyben legyen benne egy régi értékeket tartalmazó objektum, valamint egy új értékeket tartalmazó objektum.

Legyen egy *NameChangeListener* osztály, mely iratkozzon fel erre az eseményre, és amennyiben a név vátozott, egy listában tárolja el ezeket a változásokat *régi név -> új név* formátumban. Ha csak a koordináták változnak, ne történjen semmi.

Írj rá tesztesetet. Ehhez injektálni kell a *NameChangeListener* beant, valamint legyen egy *deleteAll()* metódusa, mely a listát üríti, valamint egy *List<String> getChanges()* metódusa, mellyel a módosítások lekérdezhetőek.

Konfigurációs állományok

A *templateLocation* bean *name*, *lat*, *lon* attribútumainak értékét (, ami most a kódban beégetve *Choose name*, stb.) töltsd fel egy *application.properties* állományból, ami az értékeket a következő kulcsokkal tartalmazza: *template.location.name*, *template.location.lat*, *template.location.lon*.

Profile használata

Jelenleg a *DummyLocationsDao* és a *ListLocationsDao LocationsDao* implementáció létezik az application contextben. Módosítsd úgy az alkalmazást, hogy a *DummyLocationsDao* csak a *dummy* profile esetén, a *ListLocationsDao* pedig a *normal* profile esetén legyen aktív. A teszteket módosítsd, hogy a *normal* profile-lal fussanak. A main-t is módosítsd, hogy *normal* profile-lal fusson. Írj egy integrációs tesztet a *dummy* profile-lal is.

Conditional beans

Jelenleg a *DummyLocationsDao* és a *ListLocationsDao LocationsDao* implementáció létezik az application contextben. Különböző profile-ok esetén kerülnek aktiválásra. Módosítsd úgy az alkalmazást, hogy a *DummyLocationsDao* csak akkor legyen érvényben, ha a *mode=dummy* környezeti változó deklarálva van, ellenkező esetben mindig a *ListLocationsDao* legyen aktív.

Naplózás

A *LocationsService*-ben minden módosító műveletnél legyen SLF4J naplózás *debug* szinten, és ez jelenjen is meg a konzolon a *LocationsMain* és a tesztesetek futtatásakor.

Aspektusorientált programozás

Írj egy advice-t, mely azt számolja, hogy milyen kezdőbetűvel kezdődő kedvenc hely hányszor került létrehozásra. Ezt tárolja egy *Map*-ben, a kulcs a betű (mindig kisbetűsítve), az érték pedig egy számláló. A *createLocation()* hívásokat kell elkapni, és nézni a paramétereket.

Írj rá egy tesztesetet. Ehhez az advice is legyen egy *@Component*, legyen getter a *Map*-hez, és injektálni kell a tesztesetbe.

Spring Framework repository réteg

Hozz létre egy *DataSource* beant, mely a saját számítógépre telepített MySQL adatbázishoz kapcsolódik. A kapcsolódási paramétereket az *application.properties* állományból olvassa be.

Séma inicializálás Flyway eszközzel

Hozz létre egy *Flyway* beant, mely inicializálja az adatbázist! A következő SQL utasítást adja ki migrációs szkriptben:

```
create table locations(id int auto_increment primary key, name varchar(255)
, lat double, lon double);

insert into locations(name, lat, lon) values ('Budapest', 47.4979, 19.0402)
;
```

Spring JdbcTemplate

Implementáld a *LocationsDao* interfészt *JdbcLocationsDao* osztállyal, a *JdbcTemplate* használatával!

Az eddigi *LocationsDao* implementációkon tedd megjegyzésbe a *@Repository* annotációkat, és akkor nem fognak beanként megjelenni.

Implementálj integrációs teszteket!

JPA használata Spring Frameworkkel

Implementáld a *LocationsDao* interfészt *JpaLocationsDao* osztállyal, a *JdbcTemplate* használatával!

Futtasd ugyanazokat az integrációs teszteket, melyeket a *JdbcLocationsDao* esetén implementáltál!

Spring Data JPA

A *JpaLocationsDao* osztályon lévő *@Repository* annotációt tedd megjegyzésbe! Hozz létre egy *LocationsRepository* interfészt (*extends JpaRepository*), és a *LocationsService* hívja ezt a repository-t.

Deklaratív tranzakciókezelés

Implementálj adatbázisba audit naplózást!

Írj egy *AuditLog* osztályt, melynek van egy *Long id* és egy *String message* attribútuma! Írj egy *AuditLoggerRepository* interfészt (*extends JpaRepository*)! Írj hozzá egy *AuditLoggerService* service-t! A *LocationsService* írási műveletek esetén hívja meg az *AuditLoggerService saveAuditLog()* metódusát! A *saveAuditLog* metódus indítson saját tranzakciót! Legyen lekérdezési lehetőség is a *listAuditLogs()* metódussal! (Ne felejtsd el létrehozni a táblát is Flyway migrációs szkripttel!)

Írj egy integrációs tesztet, amennyiben ellenőrzöd, hogy mentés esetén jelenik meg audit log.

Bevezetés a Spring MVC használatába

Hozz létre egy új projektet *locationsweb* néven. Készíts egy oldalt, mely kiírja a következő tartalmat:

```
<html><body><h1>Locations</h1></body></html>
```

Thymeleaf view

Az előző szöveget ne Java String objektum alapján adja vissza, hanem egy Thymeleaf template alapján. Az oldalon jelenjen meg az aktuális idő is.

A projektbe hozz létre egy *Location* osztályt (előző projektből átemelhető).

Az oldalon jelenjen meg egy táblázat, ami kiír pár kedvenc helyet!

Erőforrások kezelése

Az alkalmazásban helyezd el a Bootstrap CSS állományát! Kösd is be a fejlécbe!

Thymeleaf oldalstruktúra

Hozz létre egy oldaltöredéket, mely a fejlécet tartalmazza!

Controllerek használata

Hozz létre egy részletek oldalt, mely elérhető a `/Location/{id}` címen! Jelenítse meg a kedvenc hely adatait!

Hozzáadás és szerkesztés (@ModelAttribute használata)

A főoldalon hozz létre egy űrlapot, amin új kedvenc helyet lehet felvenni. Csak két beviteli mező legyen, az egyik első a név, a második a koordináták. A koordinátákat a következő formátumban kell megadni: `49.12, 13.45`. Az űrlap mögött egy `LocationForm` álljon (két `String` attribútummal)! A controller a `LocationForm` alapján hozzon létre egy `Location` példányt, amit aztán adjon tovább a service-nek, ami egy belső listába elmenti! A koordinátákat `String` műveletekkel kell feldolgozni.

Tesztelés

Hozz létre egy unit tesztet a controllerre és egy integrációs tesztet a mentésre és betöltésre!

Handlerek (opcionális)

Hozz létre egy handlert, mely az vizsgálja, hogy a böngésző milyen nyelvre van állítva! Ezt kiírja a naplóba. Ehhez a `Accept-Language` headert kell vizsgálni.

I18N

A felületen megjelenő összes feliratot properties állományból tölts be!

Opcionális feladat: Tedd lehetővé a nyelvváltást!

Validáció

Ellenőrizd, hogy a név nem-e üres, valamint azt, hogy a koordináták a megfelelő formátumban vannak-e beírva. Erre több lehetőséged is van:

- `@Pattern` annotáció használatával reguláris kifejezést használhatsz
- Saját Bean Validation annotációt implementálsz
- A controllerben implementálod az ellenőrzést, és `BindingResult`-nak meghívod az `addError(ObjectError)` metódusát, és explicit módon töltöd fel hibával.

Fájlkezelés (opcionális)

A kedvenc helyhez lehessen képet is feltölteni! Egészítsd ki az űrlapot úgy, hogy fájlt is lehessen megadni! Egészítsd ki a `Location` osztályt egy `byte[]` típusú mezővel!

Témák használata (opcionális)

Tedd lehetővé, hogy az alkalmazásban témát lehessen váltani!

Hiba és kivételkezelés

Konfigurálj egy 404-es és 500-as oldalt, hogy az alkalmazás designjába illeszkedjen!

Backend integráció

Az előző projektből emeld át a Spring Data JPA megvalósítást, és integráld a webes projektbe!

Ha megvalósítottad a képfeltöltést, azt is mentsd le adatbázisba. Ehhez blobot kell használnod, az annotáció a JPA entitás megfelelő attribútumán (`Location.image`) `@Lob`.

RESTful webszolgáltatások

Implementáld a CRUD műveleteket a `Location` entitásra! Legyen RESTful művelet kedvenc helyek listázására, egyedi azonosító alapján lekérésre, létrehozásra, módosításra és törlésre!