# JAVA gyakorlati beadandó feladat

Szász András Gábor

Neptun kód: ZYKV5N

A feladatot egyedül készítettem el. A munkafolyamat során az elkészült munkarészeket a GITHUB -ra két profilon keresztül töltöttem fel, SzaszAndrasGabor és SzaszZYKV5N felhasználói fiókokat használva.

A feladat a következő linken keresztül érhető el:

<https://github.com/SzaszAndrasGabor/JavaGyak/commits/master/>

## adatbázis készítése

A feladathoz kezdetben egy feladat elnevezésű MySQL adatbázist hoztam létre, amely két táblát tartalmaz:

users tábla a regisztrált felhasználók számára, illetve messages tábla a bejelentkező és nem bejelentkező látogatók üzeneteinek eltárolására.A képen szöveg, szoftver, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Automatikusan generált leírás

A lekérdezéses feladathoz szükséges táblák később lettek létrehozva. Ezeket a táblákat szöveges állományokból kellett létrehozni.

A lekérdezés feladat megírása előtt a szöveges fájlban megkapott adatokat be kellett importálni az adatbázisba. Ehhez létre kellett hozni az új táblákat is. A feladatkiírásban a következő relációs adatbázist kellett megvalósítani:

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Automatikusan generált leírás

A szöveges fájlok nem tartalmaztak azonosító mezőt.

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

A létező oszlopokba beimportáltam az adatokt, az első sor kihagyásával.

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

Az importálást txt fájlból nem, csak .csv fájlok segítségével tudtam lefuttatni, amihez Excel programot használtam fel. Importálás után mentés másként, CSV formátumbanA képen szöveg, képernyőkép, szám, szoftver látható

Automatikusan generált leírás

A munkalap beolvasva:

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

Az **az** mező hozzáadása után előállt egy azonosítóval ellátott munkalap tábla.

A képen szöveg, képernyőkép, szoftver, sor látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

A hely tábla beolvasása:

A képen szöveg, képernyőkép, Betűtípus, multimédia látható

Automatikusan generált leírás

CSV előállítás:

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

A képen szöveg, Betűtípus, szám, képernyőkép látható

Automatikusan generált leírás

Az elkészült hely tábla:

A képen szöveg, elektronika, képernyőkép, szoftver látható

Automatikusan generált leírás

A szerelo táblánál ugyanezeket a lépéseket követtem.

## Adatbázis csatlakoztatása

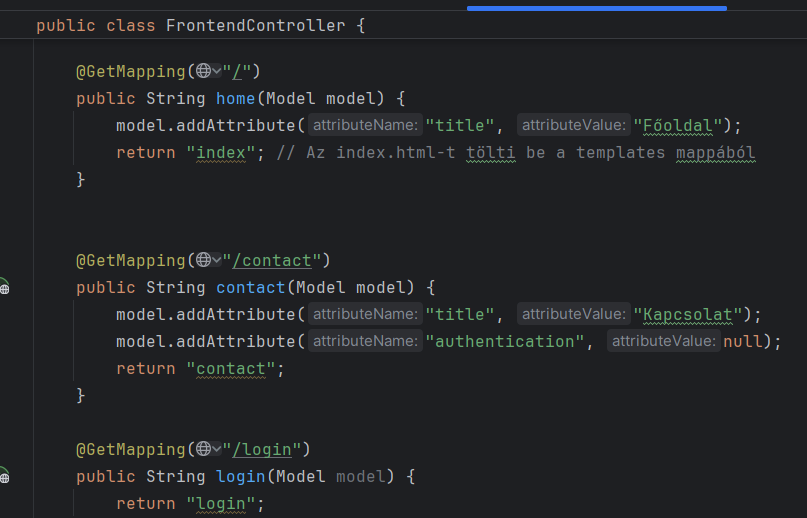
A beadandó feladathoz Springboot projektet hoztam létre.

A feladatban felhasznált MYSQL adatbázis eléréséhez az application.properties fájlban be kell állítani az adatbázis elérhetőségét: A képen szöveg, képernyőkép, szoftver, multimédia látható

Automatikusan generált leírás

A projekt strukturában a bejelentkezést kezelő fájlok részére egy login nevű csomagot, az üzenetek feltöltésére illetve azok lekérdezésére egy contact és egy messages, csomagot hoztam létre.

A JavaGyakApplication mellett egy Controller fájl és a belépést menedzselő SecurityConfig.java fájl lett elhelyezve. Előbbi az egyes weboldalak betöltési útvonalait, az utóbbi a SpringBootSecurity könyvtárak felhasználásával az egyes trtalmak elérhetőségét szabályozza a különböző bejelentkezési-jogosultsági szintek figyelembe vételével.



## Package és Entity osztály létrehozása

A login csomagban elhelyezett User.java Entity osztály kapcsolódik a user táblához.

package com.example.javagyak.login;  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "users")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
 @Column(nullable = false, unique = true)  
 private String username; // Ez az egyedi felhasználónév  
 @Column(nullable = false, unique = true)  
 private String email; // Egyedi e-mail cím, ha valaha kell  
 @Column(nullable = false)  
 private String password; // Jelszó  
 @Transient  
 private String confirmpassword; // Jelszó megerősítés, nem kerül adatbázisba  
 private String role; // Felhasználói szerep  
  
 // Getterek és setterek

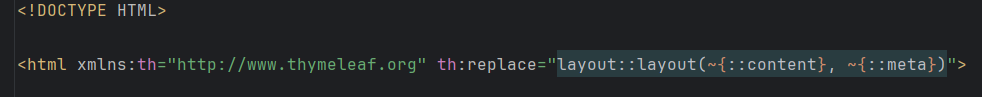
A tábla rekordjainak kezelésére User Repository néven lett létrehozva egy interfész:

package com.example.javagyak.login;  
  
import java.util.Optional;  
  
import org.springframework.data.repository.CrudRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface UserRepository extends CrudRepository<User, Integer> {  
  
 Optional<User> findByUsername(String username);  
}

## Index oldal létrehozás

A megjelenítendő tartalmakat egy ingyenes reszponzív websablon felhasználásával készítettem el, a megjelenésért felelős css elemek és képek a resources/static könyvtárban kerültek elhelyezésre. Az index.html fájlt a változó dinamikus tartalmak megjelenítése miatt át kellett alakítani, ezért létrehoztam egy egységes megjelenítést biztosító keretfájlt layout.html néven, amibe a hivatkozott oldalak tartalme be lett ágyazva.

Ennek megvalósításához a layout.html fájlban a következő beállítások szükségesek:

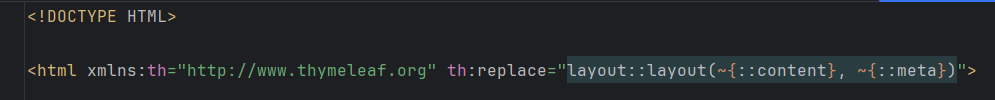


Majd a beágyazandó tartalmat <div></div> elempár közé az alábbiak szerint tudjuk behívni:

A képen képernyőkép, szöveg, Betűtípus látható

Automatikusan generált leírás

Ahhoz, hogy ez működni tudjon a hivatkozást a dinamikus tartalmakat tartalmazó html fájlokban a következő kiegészítéseket kell elvégezni:



Itt megadjuk, hogy az adott fájl a layout nevű sablonfájltól örökli a tartalmakat, amelynek van egy content nevű fragmentje, aminek a helyén kiegészítésre kerül a dinamikus tartalom. A fájlban a beágyazásra szánt részletnél a szintén egy <div></div> tagek közé zárt tartalmat jelölünk meg a következő módon:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A layout.html fájlban a fejlécben szereplő adatok örökölhetőségéhez a következő beállítást elvégezve a meta elemek egyszerűen átörökíthetőek:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

## Bejelentkezés megvalósítása

A bejelentkezéshez az adatbázis user nevű táblája lett felhasználva, amelyben az eltárolt adatok a következőképpen néznek ki:

A képen szöveg, szoftver, szám, Weblap látható

Automatikusan generált leírás

A felhasználó azonosítója, felhasználóneve, emailcíme mellett rögzítésre kerül egy titkosított jelszó, valamint egy megtekintési jogosultságot azonosító mező is.

A jelszó a regisztráció folyamatában kerül legenerálásra az alábbi kódrészlet alapján:

user.setPassword(passwordEncoder.encode(password));, amely kódrészlet a UserService osztályban kapott helyet.

A regisztráció a rgister.html-ben megadott adatok alapján kerül feldolgozásra a UserService.java fájlban, amit űrlap kitöltése után a gombra kattintva a FrontendControllerben POST módszerrel hívunk meg:

A képen szöveg, Betűtípus, szoftver, képernyőkép látható

Automatikusan generált leírás

Az adatbázisba történő eltárolás a következők szerint megy végbe:

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

Ehhez létre kellett hozni egy repository-t:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A bejelentkezéshez a következő kód nyújt segítséget, amit a login.html mezőiben kitöltött felhasználónév-jelszó páros alapján végez:

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

A feldolgozáshoz a SpringBootSecurity beállításait használjuk, ami nagyban megkönnyíti a felhasználó azonosítását, látogatható oldalak kijelölését, illetve a sikeres belépés után kijelölhető, hogy melyik oldalra térjünk vissza. Ezeket a beállításokat a SecurityConfig.java fájl tartalmazza.

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

A belépést követően a követhező képernyőkét tárul elénk: A képen szöveg, képernyőkép, Weblap, Webhely látható

Automatikusan generált leírás

A layout.html-ben meghatározott fejléc tartalmazza a bejelentkezés során lekérdezett felhasználói adatokat és jogosultságot, amit a menü kialakításakor figyelembe vesz.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Az adatbázisból lekért Admin jogosultság esetén megjelenik a Lekérdezés menüpont, ami a három mintatábla felhasználásával készült lekérdezést tartalmazza.

## Lekérdezés 3táblából

A három tábla adatait felhasználó lekérdezés oldal csak Admin jogosultságú felhasználók részére elérhető, amit a layout.txt-ben is meghatároztunk. A minta adatbázis adatainak lekérdezéséhez létrehoztam a tábláknak megfeleltethető osztályokat, Szerelo, Munkalap és Hely néven, melyeket a lekérdezés feladat részére létrehozott package-ben tároltam el.

A lekérdezést végül ebben az esetben közvetlen SQL lekérdezéssel oldottam meg, amelyhez létrehoztam egy lekerdez.html nézetet, egy Tablagenerator.java controllert, végül a FrontendControlleren belül paraméterként átadtam a meghívandó fájlnak a string változóként legenerált html táblázatot. A lekerdez.html fájlban ez a változó a következőképpen lett beszúrva:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A táblázat előállítása a TablaGenerator fájlban:

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

A paraméter átadása a FrontendControllernél:

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A lekérdezés eredménye:

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

## Üzenet küldése

Az üzenetek küldése az adatmázis message tábláján keresztül valósul meg, a küldött üzenetek tartalma, ideje, küldője mentésre kerül.

Ehhez Létre kell hozni egy Entity osztályt, és a kapcsolódő repository interfészt. Ezek külön, a message és a contact csomagban lettek elhelyezve.

A Kapcsolat gombra kattintva metöltődik a contact.html, majd a benne lévő adatokat elküldve a KuldController hívódik meg. Itt ellenőrzésre kerül az üzenet hossza, a bejelentkezett felhasználó adatai, a felhasználó neve, majd a KuldService használatával mentésre kerül az üzenet. Az adatbáziskapcsolathoz a KuldRepositoryn keresztül kapcsolódik a rendszer. Azonosítatlan felhasználó esetén NULL érték kerül az adatbázis user\_id mezőjébe.



## Üzenetek kiolvasása

Az elmentett üzenetek a messages csomagban található entitásosztály és MessageRepository segítségével és a MessageController által tölti be a FrontendController által meghívott messages.html fájlba a már eddig elmentett üzeneteket. A Null értékű user\_id esetén a Név oszlopba a „Vendég”, azonosított emailcím esetén az elmentett user\_id által kikeresett felhasználónév kerül betöltésre a users tábla username mezője alapján. A különböző táblákból lekérdezett adatok legyűjtéséhez az MsgDTO osztályt használjuk.

A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás

## RESTful feladatok

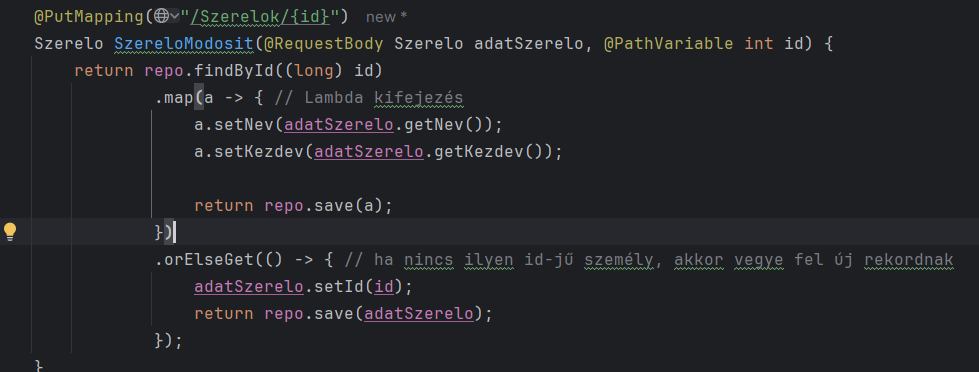
A Restful lekérdezésekhez egy restful nevű csomag került kialakításra. A szerelo tábla adatainak lekéréséhez a Szerelo osztály lett létrehozva, ez korábban a lekerdezes csomagban lett elhelyezve.

A feladatok megoldásához a SzereloRepository interfész lett kialakítva, a lekérdezéseket a SzereloController végzi.

A parancssori tesztelés az alábbi eredményekkel zárult:

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, képernyő, szoftver látható

Automatikusan generált leírás