משתמשים בדאטהבייס אמיתי

עד עכשיו השתמשנו בדאטהבייס בזכרון של ספרינג, עכשיו אנחנו רוצים אחד אמיתי.

הויזרים שלנו עד עכשיו היו בתוך מבנה נתונים מסוג UserDetailsService, הטיפוס הקונקרטי שבו שמרנו את המידע היה InMemoryUserDetailsManager

```
@Override
@Bean
protected UserDetailsService userDetailsService() {
```

```
return new InMemoryUserDetailsManager(rootUser, studentUser, traineeAdmin);
```

עכשיו כדי לשנות את זה שזה לא יהיה דאטהבייס בזכרון קודם נראה איזה עוד מחלקות ממשות את UserDetailsService

```
CachingUserDetailsService (org.springframework.security.authentication)
InMemoryUserDetailsManager (org.springframework.security.provisioning)
JdbcDaoImpl (org.springframework.security.core.userdetails.jdbc)
JdbcUserDetailsManager (org.springframework.security.provisioning)
ReactiveUserDetailsServiceAdapter in WithUserDetailsSecurityContextFactory (o
UserDetailsManager (org.springframework.security.provisioning)
UserDetailsServiceDelegator in WebSecurityConfigurerAdapter (org.springframew
```

אז אפשר לראות למשל את JdbcDaoImpl, שזאת מחלקה כבר נותנת לנו הרבה שאילתות sql מכונוץ והרבה עזרה.

אבל אנחנו רוצים משלנו.

קודם נראה מה הטיפוס שאנחנו משתמשים בהם עכשיו צריכים להיות, אז הם מטיפוס של הממשק UserDetails ואלו המתודות שהוא מצהיר עליהן

```
public interface UserDetails extends Serializable {
    Collection<? extends GrantedAuthority> getAuthorities();

    String getPassword();

    String getUsername();

    boolean isAccountNonExpired();

    boolean isAccountNonLocked();

    boolean isCredentialsNonExpired();

    boolean isEnabled();
}
```

אז בואו ניצור מחלקה שתממש את המתודות האלו

```
public class ApplicationUser implements UserDetails {
    private final Set<? extends GrantedAuthority> grantedAuthorities;
    private final String password;
    private final String userName;
    private final boolean isAccountNonExpired;
    private final boolean isAccountNonLocked;
    private final boolean isCredentialsNonExpired;
    private final boolean isEnabled;

    public ApplicationUser(
                            String userName,
                            String password,
                            Set<? extends GrantedAuthority> grantedAuthorities,
                           boolean isAccountNonExpired,
                           boolean isAccountNonLocked,
                           boolean isCredentialsNonExpired,
                           boolean isEnabled) {
        this.grantedAuthorities = grantedAuthorities;
        this.password = password;
        this.userName = userName;
        this.isAccountNonExpired = isAccountNonExpired;
        this.isAccountNonLocked = isAccountNonLocked;
        this.isCredentialsNonExpired = isCredentialsNonExpired;
        this.isEnabled = isEnabled;
    }


    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return grantedAuthorities;
    }

    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public String getUsername() {
        return userName;
    }

    @Override
    public boolean isAccountNonExpired() {
        return isAccountNonExpired;
    }

    @Override
    public boolean isAccountNonLocked() {
        return isAccountNonLocked;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return isCredentialsNonExpired;
    }

    @Override
    public boolean isEnabled() {
        return isEnabled;
    }
}
```

ועכשיו בוא ניצור את service של המחלקה שלנו, הsrvice יהיה אחראי למשוך את המידע מהדאטהבייס

אז בואו ניצור מחלקה שתירש מUserDetailsService

```java
public interface UserDetailsService {
    UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;
}
```

אז זאת המחלקה שיצרנו

```java
@Service
public class ApplicationUserService implements UserDetailsService {

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return null;
    }
}
```

בואו ניצור repository שיוכל לדבר עם הדאטהבייס

```java
@Repository
public interface ApplicationUserDao {
    Optional<ApplicationUser> selectApplicationUserByUsername(String username);
}
```

ועכשיו ניצור מחלקה שתממש את האינטרפייס, בגדול היא תהיה מחלקה אמתית רק יהיה שם רשימה של כל מה היוזרים שהיו שם מקודם

אז היא תממש את האינטרפייס, יהיה לה את הpasswordEncoder כדי להצפין ססמאות, ויהיה שם את המתודה מהאינטרפייס שמוצאת את המשתמש לפי השם משתמש, ששם אנחנו לוקחים את הרשימה שאנחנו מייצרים בgetApplicationUsers ומוצאים את המשתמש הראשון עם השם הזה.

```java
@Repository("fake")
public class FakeApplicationServiceDao implements ApplicationUserDao{

    private final PasswordEncoder passwordEncoder;

    public FakeApplicationServiceDao(PasswordEncoder passwordEncoder) {
        this.passwordEncoder = passwordEncoder;
    }


    @Override
    public Optional<ApplicationUser> selectApplicationUserByUsername(String username) {
        return getApplicationUsers() List<ApplicationUser>
                .stream() Stream<ApplicationUser>
                .filter(applicationUser -> username.equals(applicationUser.getUsername()))
                .findFirst();
    }
```

המתודה הזאת מדמה הבאה של כל המשתמשים מהדאטהבייס, היא פשוט יוצר רשימה של יוזרים

```
private List<ApplicationUser> getApplicationUsers(){
    List<ApplicationUser> applicationUsers = Lists.newArrayList(
            new ApplicationUser(
                    userName: "admin",
                    passwordEncoder.encode( rawPassword: "password"),
                    ADMIN.getGrantedAuthorities(),
                    isAccountNonExpired: true,
                    isAccountNonLocked: true,
                    isCredentialsNonExpired: true,
                    isEnabled: true
            ),
            new ApplicationUser(
                    userName: "traineeAdmin",
                    passwordEncoder.encode( rawPassword: "password"),
                    ADMIN_TRAINEE.getGrantedAuthorities(),
                    isAccountNonExpired: true,
                    isAccountNonLocked: true,
                    isCredentialsNonExpired: true,
                    isEnabled: true
            ),
            new ApplicationUser(
                    userName: "student",
                    passwordEncoder.encode( rawPassword: "password"),
                    STUDENT.getGrantedAuthorities(),
                    isAccountNonExpired: true,
                    isAccountNonLocked: true,
                    isCredentialsNonExpired: true,
                    isEnabled: true
            )
    );
    return applicationUsers;
}
```

כדי להיות הכי ברורים (ואם יהיו לי עוד מימושים של ApplicationUserDao) אני מציין בדיוק איזה אחד אני רוצה לקחת

```
@Autowired
public ApplicationUserService(@Qualifier("fake")ApplicationUserDao applicationUserDao) {
    this.applicationUserDao = applicationUserDao;
}
```

עכשיו במחלקה של ApplicationSecurityConfig ונוסף את ה service שלנו

```
private final PasswordEncoder passwordEncoder;
private final ApplicationUserService applicationUserService;
@Autowired
public ApplicationSecurityConfig(PasswordEncoder passwordEncoder, ApplicationUserService applicationUserSer
    this.passwordEncoder = passwordEncoder;
    this.applicationUserService = applicationUserService;
}
```

נמחק את protected UserDetailsService userDetailsService() שהיה קודם ונשים את המתודה הזאת במקום

```
@Bean
public DaoAuthenticationProvider daoAuthenticationProvider(){
    DaoAuthenticationProvider provider = new DaoAuthenticationProvider();
    provider.setPasswordEncoder(passwordEncoder);
    provider.setUserDetailsService(applicationUserService);
    return provider;
}
```

ועכשיו נדרוס עוד מתודה configure שהפעם תקבל AuthenticationManagerBuilder, והיא תמשתמש בprovider שלנו

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.authenticationProvider(daoAuthenticationProvider());
}
```