Introduction à JavaFX

A travers la création d'une application « TodoList »

Sommaire

Gestion de la session utilisateur avec un Singleton

Nous allons mettre en place un **Singleton** pour gérer la session utilisateur et permettre l'accès aux informations de l'utilisateur connecté dans toute l'application.

Pourquoi utiliser un Singleton ?

- **Stocke une seule instance de l'utilisateur connecté**, accessible depuis n'importe quelle classe.
- **Évite de répéter des requêtes en base** pour récupérer l'utilisateur.
- **Facilite la gestion des accès** en gardant en mémoire les informations de connexion.

Objectif : Stocker et récupérer l'utilisateur connecté

Fichiers concernés :

- src/main/java/session/SessionUtilisateur.java
- src/main/java/appli/accueil/LoginController.java
- src/main/java/appli/accueil/MainController.java (ou toute autre classe nécessitant l'utilisateur connecté)

Création de la classe SessionUtilisateur

Étape 1 : Créer un package session et ajouter SessionUtilisateur.java

Dans src/main/java, créez un package session, puis ajoutez-y la classe suivante :

```
public class SessionUtilisateur {
    private static SessionUtilisateur instance;
    private Utilisateur utilisateurConnecte;

    private SessionUtilisateur() { }

    public static SessionUtilisateur getInstance() {
        if (instance == null) {
            instance = new SessionUtilisateur();
        }
        return instance;
    }
public void sauvegardeSession(Utilisateur utilisateur) {
        if (this.utilisateurConnecte != null) {
            this.utilisateurConnecte = utilisateur;
        }
    }

    public Utilisateur getUtilisateur() {
        return utilisateurConnecte;
    }

    public void deconnecter() {
        utilisateurConnecte = null;
    }
}
```

Utilisation dans LoginController

Nous allons modifier handleLogin() pour stocker l'utilisateur connecté dans SessionUtilisateur.

Étape 2 : Modifier handleLogin()

Ajoutez l'importation dans LoginController.java :

- import session.SessionUtilisateur;

Modifiez handleLogin() pour stocker l'utilisateur après connexion :

```
    if (utilisateur != null && passwordEncoder.matches(password, utilisateur.getMdp())) {
        System.out.println("Connexion réussie pour : " + utilisateur.getNom());
        SessionUtilisateur.getInstance().sauvegardeSession(utilisateur);
        errorLabel.setVisible(false);
        // Redirection possible vers une autre page
    } else {
        System.out.println("Échec de la connexion. Email ou mot de passe incorrect.");
        errorLabel.setText("Email ou mot de passe incorrect.");
        errorLabel.setVisible(true);
    }
```

Récupération de l'utilisateur connecté

Dans n'importe quelle classe, on peut récupérer l'utilisateur connecté avec :

```
Utilisateur utilisateurActuel = SessionUtilisateur.getInstance().getUtilisateur();
if (utilisateurActuel != null) {
    System.out.println("Utilisateur connecté : " + utilisateurActuel.getNom());
}
```

Déconnexion de l'utilisateur

Ajoutez cette méthode dans un contrôleur pour gérer la déconnexion :

```
@FXML
protected void handleLogout() {
    SessionUtilisateur.getInstance().deconnecter();
    System.out.println("Utilisateur déconnecté.");
    // Redirection vers la page de connexion
}
```

Tâches à effectuer

- Créer la classe SessionUtilisateur.java en Singleton.
- Modifier handleLogin() pour stocker l'utilisateur connecté.
- Tester l'accès à SessionUtilisateur depuis une autre classe.
- Créer un bouton de déconnexion qui vide la session.

Prochaine étape : Utiliser la session pour gérer les permissions et les rôles !