Desmullier Gabriel

Rousseau Jules

Projet JEE 2021:

Site de téléchargement de fonds d'écrans

Concept :

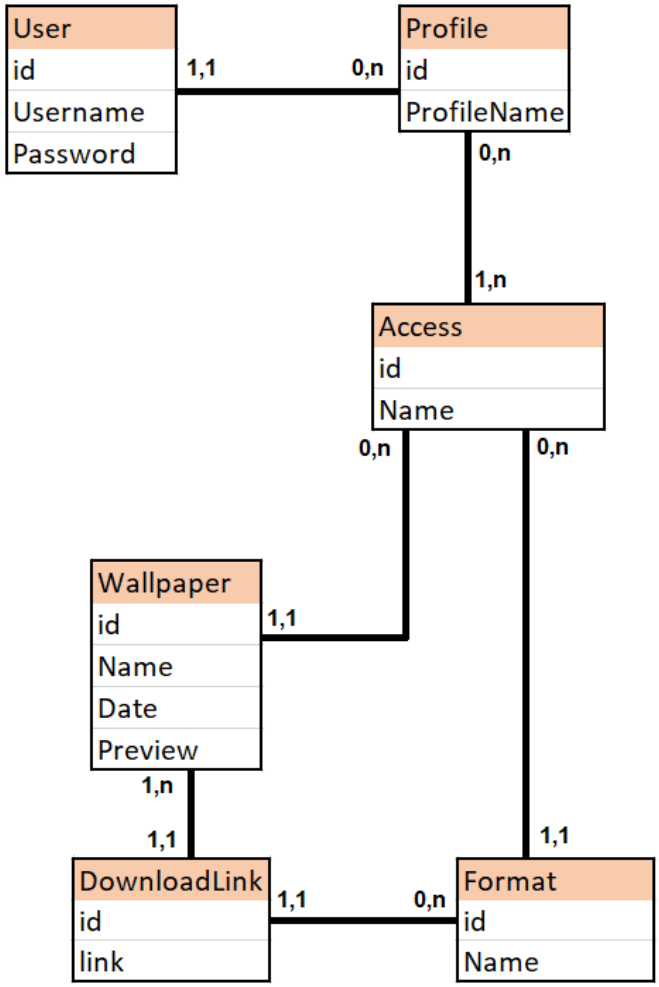
Nous avons créé une application web de téléchargement de fonds d'écrans pour ordinateur qui propose régulièrement des fonds d'écrans réalisés par des artistes ou tirés d'œuvres populaires (films, séries, bd, jeux vidéo…) de bonne qualité dans divers formats. Afin de continuer la collaboration pour offrir de nouveaux fonds d'écrans chaque semaine, nous devons faire payer nos utilisateurs. Nous avons établi trois types de comptes différents. Le compte free, gratuit, permettra aux utilisateurs de nous découvrir avant éventuellement de souscrire à un compte payant. Les utilisateurs free auront accès à environ 10% de nos fonds d'écrans et ne pourront télécharger ceux-ci qu'au format standard. Ensuite il existe deux types de compte payant : le compte basique qui permet de télécharger 50% des fonds d'écrans et le compte premium qui permet d'avoir accès à la totalité de notre répertoire. Les utilisateurs qui ont l'un de ces deux types de compte pourront télécharger leurs fonds dans tous les formats.

Technologies :

Pour réaliser notre application web, nous avons utilisé Spring MVC et nous nous sommes donc inspirés du Labwork 04. A cela nous avons ajouté un système de connexion et de session à l'aide de Spring Security. Enfin pour échanger avec la base de données, nous avons utilisé JPA.

La base de données :

Notre base de données est composée de 5 entités et suit le Modèle conceptuel de données suivant :



Un utilisateur possède un profil, profil qui sera soit free, soit basique, soit premium. Un fond d'écran aura plusieurs liens de téléchargement, chaque lien étant associé à un type de format. Afin d'autoriser l'accès d'un fond d'écran ou d'un format à un utilisateur en fonction de son profil, nous avons créé une entité Access. L'entité Access correspond à la liste des profils qui ont accès à la donnée.

L'ajout de nouveaux types de profils et la modification des accès peut grâce à ce schéma être réalisé très rapidement sans devoir faire énormément de changements.

Structure du projet

Notre projet se divise entre trois modules Maven. Le module Core contient toute la logique métier qui permet d'interagir avec la base de données. On y trouve les classes de configuration, les entités, les DAO et les services.

Le module Data contient une simple classe Application qui permet d'initialiser la base de données, c'est une alternative au fichier dump.

Le module web est le plus important car c'est lui qui gère notre application web. Il possède deux classes de config, l'une webConfig permet de configurer le modèle MVC, l'autre SecurityConfig sert à configurer la partie « sécurité » (avec Spring Security) et donc tout ce qui est authentification. Le module contient aussi un controller pour gérer l'ensemble des pages, deux classes UserDetailsImpl et UserDetailsServicesImpl servant à effectuer l'authentification de l'utilisateur, et une classe SpringSecurityInitalizer qui permet de générer les filtres pour chaque page. Enfin la classe Initializer permet comme son nom l'indique d'initializer l'ensemble de l'application en créant les liens entre les différents modules et packages.

Quelques détails :

Core : DAO : Les requêtes spécifiques

WallpaperDAO et DownloadLinkDAO sont pourvus de requêtes assez complexes.

Pour wallpaperDAO, il existe deux méthodes qui retournent des listes de wallpaper. La première sert à obtenir la liste des wallpaper qui ont un accès qui comporte le profil donné :

```
List<Wallpaper> findAllByAccessProfileListContaining(Profile profile);
```

La seconde fait l'inverse, c'est-à-dire qu'elle va chercher l'ensemble des wallpaper dont l'accès ne comporte pas le profil donné. Cette méthode est utilisée pour montrer à l'utilisateur, les wallpaper qu'il pourra télécharger s'il souscrit à un autre profil.

```
List<Wallpaper> findAllByAccessProfileListIsNotContaining(Profile profile);
```

La requête la plus complexe est celle qui se trouve dans DownloadLinkDAO. Cette méthode sera utilisée pour afficher les liens d'un fond d'écran. En premier lieu, elle cherche les liens avec l'id du wallpaper puis ensuite vérifie l'accès dans l'accès du Format qui est associé à ce lien.

```
List<DownloadLink>
findAllByWallpaperIdAndFormatAccessProfileListContaining(Long id,Profile
profile);
```

Data : Application : Les utilisateurs enregistrés

La classe application lorsqu'elle est lancée, va enregistrer des utilisateurs. Pour pouvoir utiliser l'application web, il faut s'authentifier avec l'un des comptes suivants :

Username	Password
HenriPotier	1234
LucasCielmarcheur	1234
IndianaJean	1234

Web : comment fonctionne le login ?

-La page du login n'a pas de controller pour l'afficher. La méthode addViewControllers de la classe WebConfig a été implémentée dans l'objectif d'afficher la page de login avec une url /login. Nous avons fait de même pour la page d'accueil.

-Nous avons ajouté une classe SecurityConfig, classe fille de WebSecurityConfigurerAdapter. Cette classe possède une instance UserDetailsService, une interface définie par Spring Security dont le rôle est de charger les détails de l'utilisateur. Nous avons dû bien sûr implémenter cette classe. Le Bean NoOpPasswordEncoder indique que les mots de passe ne sont pas cryptés : notre objectif n'étant pas de faire application web sécurisé mais une application fonctionnelle. La méthode configureGlobal définit l'accès à notre application. La seule ligne de cette méthode indique que l'authentification est basée sur l'implémentation de l'UserDetailService passé en paramètre.

-La classe UserDetailsServiceImpl est la classe qui implémente UserDetailsService. Elle implémente donc la méthode qui vérifie l'existence d'un utilisateur dans la base de données grâce à son username. Pour cela elle utilise l'UserService du module Core. Cette méthode renvoie une implémentation de l'interface UserDetails.

-La classe UserDetailsImpl implémente UserDetails et donc implémente toutes les méthodes qui retournent les informations sur l'utilisateur, c'est-à-dire sur l'instance user injecter dans cette classe. La méthode getAuthorities sert simplement à donner à l'utilisateur une clé de rôle. Cette clé de rôle permet à l'utilisateur d'avoir accès à certaines pages et de ne pas avoir accès à d'autres grâce à un système d'annotations. Dans notre projet, cela ne nous intéresse pas, tous les utilisateurs ont accès à toutes les pages. Tous les utilisateurs ont donc le rôle « USER » et aucune annotation @Secured(« USER ») n'est utilisée.

-La classe SpringSecurityInitializer qui implémente AbstractSecurityWebApplicationInitializer qui reste vide sert à générer les filtres pour chaque page de l'application en fonction de ce qui a été configuré et des annotations.

-Dans la classe Java Initializer a été ajouté en sortie de la méthode getRootConfigClasses() la classe SecurityConfig pour appliquer notre système de login.

Web : WallpaperControllerRécupérer le profil de l'utilisateur connecté

Pour afficher les fonds d'écrans correspondant au profil de l'utilisateur connecté, il faut en premier lieu récupérer l'UserDetailsImpl qui contient donc les informations de l'utilisateur connecté. Puis à partir de là, on peut récupérer l'entité utilisateur et son profil.