Projet site de timbre V2

Thomas Aucoin-Lo
e2395387

Notes

2 niveaux d'accès sont possibles, soit Root et utilisateur normal. Ils ont chacuns leurs propres pages, il est donc nécéssaire de se connecter en tant que root et ensuite en tant qu'utilisateur normal pour voir tout le projet.

voir **niveaux d'accès pour plus de détails*

Liens

webdev: https://e2395387.webdev.cmaisonneuve.qc.ca/PW2-TP2/

git-hub: https://github.com/thomasIRA/PW2-TP2.git

Résumé

Un site où les utilisateurs peuvent se connecter et ajouter des timbres categorisés, les échanger (à venir), et autres fonctionnalités

Connexions

voir **niveaux d'accès pour plus de détails*

Sur webdev:

root**:**
user email: root@root.com
password: rootroot

utilisateur normal:
user email: bill@bill.com
password: billbill
**ou créez votre propre utilisateur*

Sur votre serveur:

root:
créer un utilisateur ayant comme username: **root**

utilisateur normal:
créez un utilisateur

Sur le diagramme entité relationnel

**voir diagram.png ou presentation.pdf*

Le projet consiste de 5 tables, dont une où la clé est composée.
la table aspect et user donne leur clés à la table stamp.
la table stamp_category partage la clé composé de ces deux tables.

Sur la base de données

**voir diagram.png ou presentation.pdf*
**voir stampDB.sql*

Les tables de la DB sont relativement lousses où la majorité des clés étrangères sont non-obligatoires.
Des champs ont aussi été altérés par rapport au TP1.

Sur les niveaux d'accès

root:
- accès à la page *panel*
- peut modifier, créer et supprimer toutes les tables
attention: Supprimer une entrée donnant des clés étrangères obligatoires supprimera aussi les entrées
correspondantes.

utilisateur normal:
- accès à la page *profile*
- peut modifier, créer et supprimer les tables suivantes où le **id** visé correspont au sien:
    - user
    - stamp

commentaires:
les niveaux d'accès sont non protégées et à titre démonstratif seulement.

Sur l'architecture du projet

Le projet actuel est construit suivant un modèle MVC et emploi l'API externe TWIG pour la gestion des rendus *(sauf exception)*

Sur le script php

Le script permet de mettre en place la majorité des fonctionnalités du projet.
**voir models et controllers*

Sur la classe CRUD

r***eadStampCat():*** permet de lire une/des entrées d'une table à deux clés composées.

deleteStampCat(): permet de supprimer une/des entrées de la table Stamp_Category en spécifiant une ou deux valeurs cibles.

readWhere(): permet de lire une/des entrées d'une table en spécifiant la cible et la valeur.

Sur les controllers

Les controllers partagent plusieurs méthodes et utilisent des concepts vus en classes. Parcontre, certaines méthodes implementent plusieurs fonctions afin de concorder à la DB.

I.e: Supprimer un utilisateur comporte 3 étapes:
- trouver le *user* et le supprimer
- trouver ses *stamps* et les supprimer
- trouver leurs *stamp_categories* et les supprimer

Commentaires
Seules les méthodes et fonctions complexes ou non-vues sont explicitement commentées dans le script.

Sur la validation et la gestion d'erreurs

Une validation et gestion d'erreur et est en place sur les niveaux d'accès et sur l'url. Une petite validation est aussi en place quant à certaines données formulaires coté client.

commentaires:
Le projet actuel ne gère pas les erreurs liés aux index des données envoyées vers le serveur.
I.e: Champs DB unique, etc.

Le Diagramme d'entité relationnel