SHTTRK documentation

SHTTRK est une librairie python qui permet de communiquer facilement avec shotTracker.

Cela permet entre autre d'intégrer shottracker à un pipeline existant (shotgun, royalrender, deadline...) ou d'automatiser une tache spécifique.

Exemple de code basique : Compter le nombre de commentaire par personne dans un projet

```
projectId = 72
stat={}

trackerIsart = shttrk("https://tracker.isartintra.com", "q.masingarbe", "[redacted]")

assets = trackerIsart.getAssetsSimple(projectId)

for asset in assets:
    assetId = assets[asset]
    coms = trackerIsart.getComSimple(assetId)
    for com in coms:
        if com[0] in stat:
            stat[com[0]]+=1
        else:
            stat[com[0]]=1

print stat
```

Librairie et version

SHTTRK est une librairie pour Pyhton 2.7

Les librairies *requests*, *re* & *HTMLParser* sont utilisées.

Lexique spécifique shotTracker :

Un shotTracker contient plusieurs **projets.**

Chaque projet contient plusieurs **assets** (S150P195, EPEE, PLANING...). Les assets ont des informations (description, durée, miniature...).

Chaque asset peut être précisé par des **tags** venant d'une liste globale de tags.

Chaque asset contient des **commentaires** (comments ou com).

Tous ces objets possède un id unique, utilisé par la librairie (projectId, assetId, comId, tagId).

1. Connexion

La première chose à faire est de créer un objet de type *shttrk* qui gardera les informations de connexion et permettra d'utiliser les fonction.

```
variable = shttrk(string url, string user, string password)
```

Retourne un objet de connexion shttrk contenant toutes les fonctions. L'url est celle de l'accueil ou de la page de connexion du shotTracker.

2. Fonctions

A partir de maintenant *shttrkObject* représente la variable où a été stocké l'objet de connexion.

2.1 Projet

```
shttrkObject.getProjects()
```

Retourne un dictionnaire des projets. Le nom des projet en *key* et l'id des projet en *value*.

2.2 Asset

```
shttrkObject.getAssets(int projectId)
```

Retourne une liste des assets du projet. Chaque asset de la liste est un dictionnaire contenant les informations de l'asset (name, id, tags, creationdate, ...).

```
shttrkObject.getAssetsSimple(int projectId)
```

Retourne un dictionnaire des assets du projet. Le nom des assets en *key* et l'id des assets en *value*.

```
shttrkObject.getAssetsInBin(int projectId)
```

Retourne un dictionnaire (similaire à getAssets()). Uniquement pour les assets avec le tag Corbeille.

```
shttrkObject.createAsset(int projectId, string name, string description = '', int length = 0)
```

Crée un asset dans le projet avec le nom indiqué. Les paramètres de descriptions et length ne sont pas obligatoires mais peuvent être renseignés.

```
shttrkObject.deleteAsset(int assetId)
```

Supprime un asset **définitivement** d'après son id.

```
shttrkObject.getAssetInfo(int assetId)
```

Retourne un dictionnaire contenant les infos d'un asset (description, follower, id, length, name, storage, tags, vignette, vignette_bg).

```
shttrkObject.updateInfo(int assetId, string name = None, string description = None, int length = None, string storage = None)
```

Change les infos d'un asset. Les paramètres *name*, *description*, *length* & *storage* ne sont pas obligatoires mais peuvent être renseignés. *Storage* correspond au chemin de stockage des fichiers de l'asset, il est affiché en haut des assets sur l'interface web.

2.3 Commentaires

2.3.1 Général

```
shttrkObject.getCom(int assetId)
```

Retourne une liste des commentaires de l'asset. Chaque commentaire est représenté par un dictionnaire contenant les informations du commentaire (utilisateur, like, date, ...).

```
shttrkObject.getComSimple(int assetId)
```

Retourne une liste des commentaires de l'asset. Chaque commentaire est représenté par une liste contenant le nom de la personne, son commentaire et l'id du commentaire.

```
shttrkObject.postCom(int assetId, string text, int user = 1)
```

Poste un commentaire contenant le *text* dans l'asset indiqué. Le commentaire sera posté au nom de l'utilisateur dont l'id est indiqué par *user* (par défaut le premier utilisateur de shotTracker). L'idéal serait de créer un utilisateur pour les script. Si l'utilisateur indiqué n'existe pas, le script s'executera sans erreur mais ne postera pas le commentaire.

```
shttrkObject.updateCom(int comId, string newText)
```

Remplace le texte du commentaire indiqué par le *newText*.

```
shttrkObject.deleteCom(int comId)
```

Supprime **définitivement** le commentaire indiqué.

2.3.2 Like

```
shttrkObject.likeCom(int comId)
```

Ajoute un +1 sur le commentaire indiqué. Le +1 sera au nom de l'utilisateur indiqué à la connexion, il ne peut pas liker ses propre commentaires.

```
shttrkObject.dislikeCom(int comId)
```

Enlève un +1 s'il y en a un sur le commentaire indiqué.

2.3.3 Fichiers attachés

```
shttrkObject.postFile(int comId, string pathToFile)
```

Attache un fichier (dont le chemin sur le disque dur est *pathToFile*) au commentaire indiqué. Le chemin du fichier ne doit pas contenir de \ et mener directement au fichier (ex : D:/shttrk/test.jpg).

```
shttrkObject.deleteFile(int assetId, int comId, string fileName)
```

Supprime le fichier indiqué dans le commentaire indiqué. *fileName* doit-être le nom du fichier (ex : tournette.mp4), si plusieurs fichiers ont le même nom, ils seront renommé par shotTracker (ex : screenshot.jpg, screenshot_0.jpg, screenshot_1.jpg).

2.4 Tags

```
shttrkObject.getAllTags(int projectId)
```

Retourne une liste de tous les tags disponible dans le projet et qui peuvent être appliqués aux assets. Chaque tag est représenté par un dictionnaire contenant les information de ce tag (id, name, type...).

Il existe 3 type de tags :

- Type 100 => simple tag de rangement (seul le nom)
- Type 1 => tag de progression (nom + barre d'avancement)
- Type 2 => tag d'état (nom + liste déroulante des états)

```
shttrkObject.getAllTagsSimple(int projectId)
```

Retourne un dictionnaire des tags disponibles dans le projet. Le nom des tags en *key* et l'id des tags en *value*.

```
shttrkObject.createTag(int projectId, string name)
```

Crée un nouveau tag (au nom indiqué par *name*) disponible dans le projet indiqué. Par défaut le tag créé est un simple tag de rangement (juste un nom, pas d'avancement ni d'état). Pour créer des tags complexes utiliser l'interface web. Pour créer un tag disponible dans tous les projets d'un shotTracker, utiliser l'interface web avec un compte administrateur.

```
shttrkObject.deleteTag(int tagId)
```

Supprime **définitivement** un tag sur tout le projet (y compris sur tous les assests).

```
shttrkObject.addTag(int assetId, int tagId)
```

Ajoute le tag indiqué sur l'asset indiqué.

```
shttrkObject.removeTag(int assetId, int tagId)
```

Enlève le tag indiqué de l'asset indiqué.

```
shttrkObject.updateTag(int assetId, int tagId, int state)
```

Change l'avancement ou l'état sur le tag indiqué dans l'asset indiqué. *State* indique l'état dans lequel on veut mettre le tag, c'est l'index d'une liste (ex : 0 = 0%, 1 = 25%, 2 = 50%,... ou 0 = TODO, 1 = WIP, 2 = CBB, ...)

3. Exploit

La fonction postCom() contourne l'interdiction de poster des balises HTML en commentaire. Il est ainsi possible de poster un commentaire avec une balise <script> pointant vers un fichier javascript attaché au commentaire. Ce fichier sera alors executé à chaque chargement de la page web shotTracker tant qu'il est dans le premier commentaire d'un asset. Une utilisation basique de ce système permet d'overrider la couleur du shotTracker sur un projet.

Les fonction postCom(), updateCom(), & deleteCom() peuvent agir au nom de n'importe quel utilisateur.

La librairie python coutourne l'envoi de notification par mail. Pour envoyer une notification depuis python utiliser :

```
shttrkObject.sendNotification(int assetId, int comId)
```

Un mail shotTracker sera envoyé pour le commentaire indiqué aux utilisateurs qui suivent l'asset indiqué.

4. Help (et disclaimer)

Les dernières versions de la librairie et de la documentation : https://github.com/qmasingarbe/shttrk

Une question, un bug ? q.masingarbe@gmail.com

La librairie peut être très puissante mais également dangereuse si mal utilisé. Il est conseillé de tester ses script sur un projet test séparé des projets de production. Les fonctions de suppression d'asset, de tag, de fichiers et de commentaires sont direct et supprime définitivement les données.