

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 tâche spécifique.

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

```
1  projectId = 72
2  stat={}
3
4  trackerIsart = shtrtk("https://tracker.isartintra.com", "q.masingarbe", "██████")
5
6  assets = trackerIsart.getAssetsSimple(projectId)
7
8  for asset in assets:
9      assetId = assets[asset]
10     coms = trackerIsart.getComSimple(assetId)
11     for com in coms:
12         if com[0] in stat:
13             stat[com[0]]+=1
14         else:
15             stat[com[0]]=1
16
17  print stat
```

Librairie et version

SHTTRK est une librairie pour Python 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** (coments ou com).

1. Connexion

La première chose à faire est de créer un objet de type *shtrk* qui gardera les informations de connexion et permettra d'utiliser les fonction. Cela permet également de se connecter à plusieurs utilisateurs ou à plusieurs tracker dans le même script.

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

Retourne un objet de connexion shtrk 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 *shtrkObject* représente la variable où a été stocké l'objet de connexion.

2.1 Projet

```
shtrkObject.getProjects()
```

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

2.2 Asset

```
shtrkObject.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, listeDesTags, dateDeCreation, ...).

```
shtrkObject.getAssetsSimple(int projectId)
```

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

```
shtrkObject.getAssetsInBin(int projectId)
```

Retourne un dictionnaire des assets dans la poubelle du projet (tag poubelle). Le nom des assets en *key* et l'id des assets en *value*.

```
shtrkObject.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.

```
shtrkObject.deleteAsset(int assetId)
```

Supprime un asset **définitivement** en fonction de son id.

```
shtrkObject.getAssetInfo(int assetId)
```

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

```
shtrkObject.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 du plan, il est affiché en haut des assets sur le site.

2.3 Commentaires

2.3.1 Général

```
shtrkObject.getCom(int assetId)
```

Retourne une liste des commentaires de l'asset. Chaque commentaire est représenté par un dictionnaire contenant ?????.

```
shtrkObject.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 ?????.

```
shtrkObject.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 le script.

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

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

```
shtrkObject.deleteCom(int comId)
```

Supprime **définitivement** le commentaire indiqué.

2.3.2 Like

```
shtrkObject.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.

```
shtrkObject.dislikeCom(int comId)
```

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

2.3.3 Fichiers attachés

```
shtrkObject.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:/shtrk/test.jpg).

```
shtrkObject.deleteFile(int assetId, int comId, int fileId)
```

Supprime le fichier indiqué dans le commentaire indiqué. ???????

2.4 Tags

```
shtrkObject.getAllTags(int projectId)
```

Retourne ???? de tous les tags disponible dans le projet et qui peuvent être appliqués aux assets.

```
shtrkObject.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*.

```
shtrkObject.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.

```
shtrkObject.deleteTag(int tagId)
```

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

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

Ajoute le tag indiqué sur l'asset indiqué.

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

Enlève le tag indiqué de l'asset indiqué.

```
shtrkObject.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.

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 exécuté 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.

La fonction postCom() permet de poster un commentaire au nom de n'importe quel utilisateur.

La fonction updateCom() permet de modifier le commentaire de n'importe quel utilisateur.

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

```
shtrkObject.postEncode('/json/send-notification.php',{'id_item' : int assetId, 'id_comment' : int comId})
```

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

4. Help (et disclaimer)

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.