ISEN OUEST

TP N°3 - Communications Web *REST*

Thibault Napoléon
thibault.napoleon@isen-ouest.yncrea.fr

Notions abordées :

- ✓ Architecture *REST*
- ✓ Méthode *GET*
- ✓ Méthode *POST*
- ✓ Méthode *PUT*
- ✓ Méthode *DELETE*

Préambule

Le but de ce TP est de comprendre l'architecture *REST* à travers un exemple simple et concret en *PHP*. Afin de ne pas perdre de temps sur l'aspect visuel, mais aussi pour respecter les standards actuels d'affichage sur téléphone et tablette (*Reponsive Design*), nous utiliserons le framework *CSS* nommé « `Bootstrap` » : https://getbootstrap.com/docs/

Nous utiliserons aussi par moment la bibliothèque d'icônes « `Font Awesome` » pour plus de lisibilité : https://fontawesome.com/

Le but du TP est d'écrire le code *PHP* permettant de créer / accéder / mettre à jour / supprimer des *Tweets* stockés dans une base de données. Pour illustrer ce mécanisme, nous utiliserons un code *JavaScript* fourni permettant de requêter le serveur pour effectuer ces différentes opérations, voir figure 1.

FIGURE 1 – *Capture d'écran de la gestion des Tweets en JavaScript.*

Afin de gérer les *Tweets*, cinq requêtes, suivant le formalisme *REST*, sont effectuées par le code *JavaScript* :

`GET` `php/request.php/tweets/` Récupération des tweets

`GET` `php/request.php/tweets/?login=...` Récupération des tweets d'un user

POST	`php/request.php/tweets/ login=...&text=...`	Ajout d'un tweet
PUT	`php/request.php/tweets/i login=...&text=...`	Modification du tweet
DELETE	`php/request.php/tweets/i?login=...`	Suppression du tweet

Afin d'afficher correctement les *Tweets*, le code *JavaScript* attend une chaîne *JSON* dont le format est le suivant :

```
[{"id":"1","text":"Un premier tweet !!","login":"cir2"},
  {"id":"2","text":"Un second tweet !!","login":"cir3"}]
```

Pour démarrer, la structure du TP vous est donnée dans les ressources du TP qui sont à télécharger sur l'ENT.

1 Initialisation de la base de données *MySQL*

Avant toute chose, il est nécessaire de créer et d'initialiser la base de données des *Tweets*. Pour cela connectez-vous à l'interface *phpmyadmin* : http://localhost/phpmyadmin
Chargez le contenu du script *SQL* « sql.sql » disponible dans les ressources du TP. Les modèles conceptuel et physique de données sont fournis dans la figure 2.

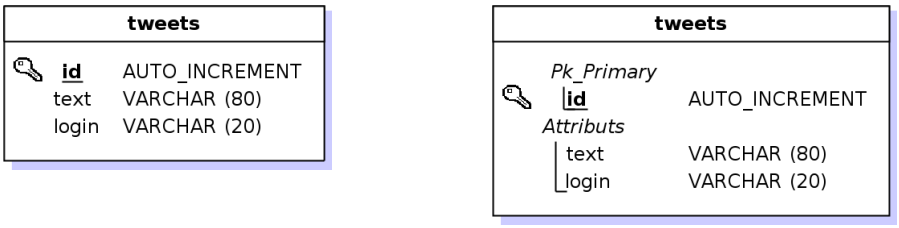

FIGURE 2 – *Modèles conceptuel (à gauche) et physique (à droite) de données de la BDD mise en place.*

2 Traitement des requêtes en *PHP*

Afin de répondre correctement aux requêtes du client, il est nécessaire d'écrire le code de traitement côté serveur en *PHP*. Pour cela, créez un script « request.php » dans le dossier « php ». Dans celui-ci, vous allez ajouter le code nécessaire à l'extraction des informations et aux traitements des quatre « verbes » *REST*. Pour l'interfaçage avec la base de données, créez un script nommé « database.php » dans le dossier « php ».

Récupération des *Tweets*

Avant toute chose, ajoutez la fonction de connexion à la base de données `dbConnect` (voir les TP précédents) dans le script « database.php ». Ensuite, afin de récupérer les différents *Tweets* présents dans la base de données, écrivez la fonction suivante :

```
dbRequestTweets($db, $login = '')
```

Cette fonction doit permettre d'interroger la base de données, d'identifiant `db`, et de renvoyer la liste de tous les *Tweets* si le paramètre `login` est absent ou la liste des *Tweets* d'un utilisateur en particulier sinon.

Pour pouvoir répondre aux requêtes de la partie client concernant la récupération des *Tweets*, ajoutez dans un script « request.php » les instructions permettant d'établir la connexion avec la base de données et d'extraire les informations disponibles dans l'*URI*.

> **Ressource dans l'*URI*** : Afin de récupérer la ressource spécifiée par le client lors de sa requête, *PHP* définit une variable serveur nommée `PATH_INFO`. Ainsi, pour récupérer la ressource *HTTP*, on utilise :
>
> ```
> $request = $_SERVER['PATH_INFO'];
> ```
>
> Pour extraire les différentes parties de la ressource, séparées par des `/`, on peut utiliser :
>
> ```
> $request = explode('/', $request);
> ```

Ensuite, appelez la fonction `dbRequestTweets` en prenant soin devérifier la présence d'un login ou non dans la requête client pour renvoyer la bonne liste de *Tweets*.

Ajout d'un *Tweet*

En premier lieu, afin d'offrir la possibilité d'ajout d'un *Tweet* dans la base de données, écrivez la fonction suivante :

```
dbAddTweet($db, $login, $text)
```

Où `db` est le lien *PDO* de la base de données, `login` est son propriétaire et `text` son contenu.

Ensuite, pour pouvoir répondre à la requête de la partie client concernant l'ajout d'un *Tweet*, ajoutez dans le script « request.php » un ensemble d'instructions permettant de vérifier la méthode *HTTP* utilisée par le client pour différencier un *GET* d'un *POST*. Enfin, appelez la fonction `dbAddTweet` lors d'un *POST* en vérifiant bien la présence des différents champs demandés lors de l'ajout.

> **Méthode *HTTP*** : Afin de récupérer la méthode *HTTP* utilisée par le client lors de sa requête, *PHP* définit une variable serveur nommée `REQUEST_METHOD`. Ainsi, pour récupérer la méthode *HTTP*, on utilise :
>
> ```
> $requestMethod = $_SERVER['REQUEST_METHOD'];
> ```

Modification d'un *Tweet*

Tout d'abord, afin d'offrir la possibilité de modifier un *Tweet* dans la base de données, écrivez la fonction suivante :

```
dbModifyTweet($db, $id, $login, $text)
```

Où `db` est le lien *PDO* de la base de données, `id` est l'identifiant du *Tweet* à modifier, `login` son propriétaire et `text` son nouveau contenu.

Ensuite, pour pouvoir répondre à la requête de la partie client concernant la modification d'un *Tweet*, modifiez la partie « aiguillage » des requêtes dans le script « request.php » pour traiter le cas d'un *PUT*. Enfin, appelez la fonction `dbModifyTweet` lors d'un *PUT* en vérifiant bien la présence des différents champs demandés pour la modification.

Suppression d'un *Tweet*

Afin d'offrir la possibilité de supprimer un *Tweet* dans la base de données, écrivez la fonction suivante :

```
dbDeleteTweet($db, $id, $login)
```

Où `db` est le lien *PDO* de la base de données, `id` est l'identifiant du *Tweet* à supprimer et `login` son propriétaire.

Ensuite, pour pouvoir répondre à la requête de la partie client concernant la suppression d'un *Tweet*, modifiez la partie « aiguillage » des requêtes dans le script « request.php » pour traiter le cas d'un *DELETE*. Enfin, appelez la fonction `dbDeleteTweet` lors d'un *DELETE* en vérifiant bien la présence des différents champs demandés pour la suppression.