TD 2 - Définition et Implémentation des Web Services

Ludovic Denoyer

6 février 2013

Nous allons dans ce TD spécifier les Web Services implémentés dans le cadre de notre site Web. Ces Web Services seront programmés en suivant les spécifications REST, et en retournant des résultats JSON. Le travail effectué en TD servira de base pour la programmation des Servlets. Ce travail de spécification doit être un **travail précis** et il sera important de conserver **une documentation des services implémentés** pour la suite du projet. Pour chaque service, il faudra spécifier :

Nom du web service	
URL du web service	
Description du service	
Paramètres en entrée	
Format de sortie	
Exemple de sortie	
Erreurs possibles	
Avancement du Service	
Classes JAVA en rapport avec le Web service	
Informations additionnelles	

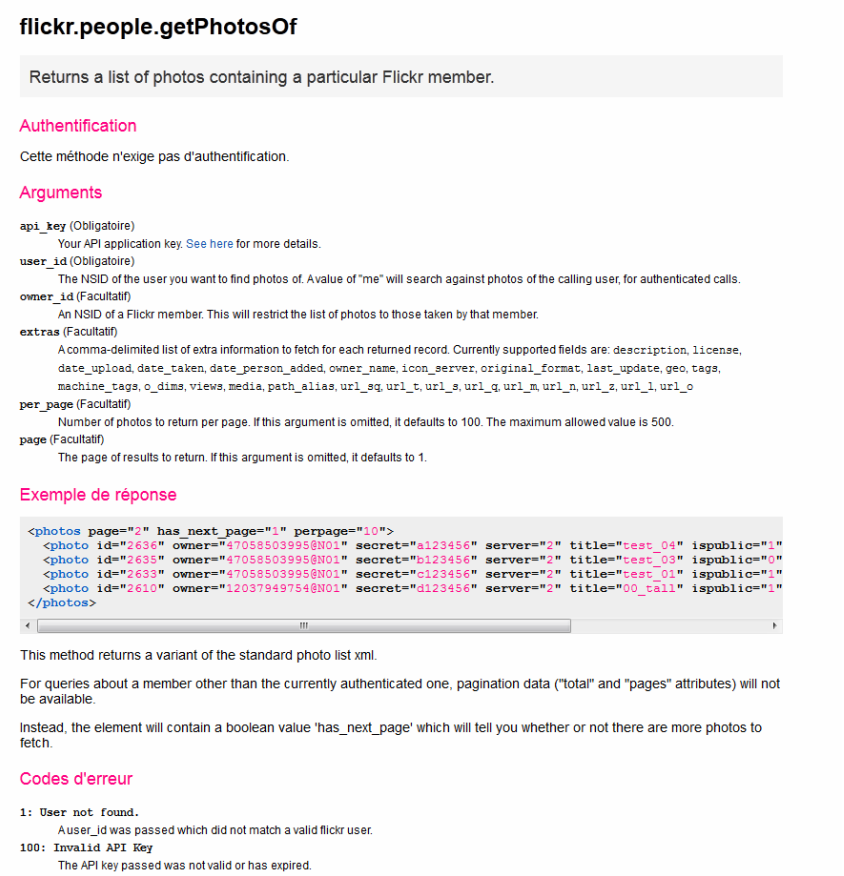
Le projet sera organisé de la manière suivante :

- le package **services** contiendra les différents services (sous forme de fonction **static**) du type : **public static JSONObject monservice(String arg1,String arg2)**.
- le package **services.servlets** contiendra les servlets correspondants aux services
- le package **services.test** contiendra les exécutables permettant de tester les services avant de les déployer sur TOMCAT
- le package **services** contiendra une classe **ServicesTools** contenant des fonctions utilitaires et particulièrement :
- le package **bd** contiendra les classes permettant l'éxécution/interrogation des bases de données.
  - une méthode **static JSONObject error(String message,int codeerreur)** permettant de générer un JSON en cas d'erreur du service
  - une méthode **static JSONObject ok()** permettant de générer un JSON correspondant à une bonne exécution d'un service

1 Rappel de Cours - sur feuille séparée, sans documents - 10 minutes

Question 1

Rappelez les principes fondamentaux d'une API REST



flickr.people.getPhotosOf

Returns a list of photos containing a particular Flickr member.

Authentification

Cette méthode n'exige pas d'authentification.

Arguments

api_key (Obligatoire)
Your API application key. See here for more details.
user_id (Obligatoire)
The NSID of the user you want to find photos of. A value of "me" will search against photos of the calling user, for authenticated calls.
owner_id (Facultatif)
An NSID of a Flickr member. This will restrict the list of photos to those taken by that member.
extras (Facultatif)
A comma-delimited list of extra information to fetch for each returned record. Currently supported fields are: description, license, date_upload, date_taken, date_person_added, owner_name, icon_server, original_format, last_update, geo, tags, machine_tags, o_dims, views, media, path_alias, url_sq, url_t, url_s, url_q, url_m, url_n, url_z, url_l, url_o
per_page (Facultatif)
Number of photos to return per page. If this argument is omitted, it defaults to 100. The maximum allowed value is 500.
page (Facultatif)
The page of results to return. If this argument is omitted, it defaults to 1.

Exemple de réponse

```
<photos page="2" has_next_page="1" perpage="10">
  <photo id="2636" owner="47058503995@N01" secret="a123456" server="2" title="test_04" ispublic="1"
  <photo id="2635" owner="47058503995@N01" secret="b123456" server="2" title="test_03" ispublic="0"
  <photo id="2633" owner="47058503995@N01" secret="c123456" server="2" title="test_01" ispublic="1"
  <photo id="2610" owner="12037949754@N01" secret="d123456" server="2" title="00_tall" ispublic="1"
</photos>
```

This method returns a variant of the standard photo list xml.

For queries about a member other than the currently authenticated one, pagination data ("total" and "pages" attributes) will not be available.

Instead, the element will contain a boolean value 'has_next_page' which will tell you whether or not there are more photos to fetch.

Codes d'erreur

1: User not found.
A user_id was passed which did not match a valid flickr user.
100: Invalid API Key
The API key passed was not valid or has expired.

Question 2

Donnez un exemple d'interrogation de l'API Flick de la figure 1. Donnez un exemple de réponse obtenue au format JSON.

2 Premier Service : Création d'utilisateurs

Nous allons considérer le système d'authentification suivant :

- Lors du *login*, une clef de 32 caractères sera générée automatiquement par le serveur
- la clef aura une durée de vie déterminée, et devra être passée comme paramètre des différents services Web
- Certaines clefs (clefs administrateur) auront une durée de vie illimitée

Question 1

Spécifiez le service permettant de créer un nouvel utilisateur. Donnez l'URL correspondant.

Question 2

Ecrire la servlet correspondante - en utilisant du pseudo-code pour les connections à la base de données.

3 Second Service : Login

Question 1

Spécifier le service *login*. Par simplicité, nous considérerons que le mot de passe est passé en clair par le protocole GET.

Question 2
Ecrire l'algo correspondant en pseudo-code très simple

Question 3
Quelles sont les fonctions dont vous allez avoir besoin ?

Question 4
Ecrire le service correspondant en JAVA

Question 5
Ecrire la servlet correspondante

4 Troisième Service : Logout

Question 1
Spécifier le service de logout. Donnez l'URL correspondant. Ecrire le code

5 Autres Services

Question 1
Faire la liste des services **de base** qui devront être implémentées dans votre projet. Les servies seront regroupés par familles :
- authentification
- messages
- amis
- recherche
- commentaires

Pour chaque service, vous spécifierez les entrées et sorties. La spécification complète et documentée des services sera à faire chez vous.