Travaux dirigés n°7

JSON et APIs

Dans ce TD, nous allons découvrir un nouveau format de données nous permettant de communiquer avec un serveur. Puis, dans un second temps, nous utiliserons ce format pour récupérer des données depuis un site tiers par le biais de son API.

Avant-propos

Dans les TDs précédents, nous avons vu comment récupérer les données d'un fichier texte en Javascript et les traiter simplement (c'est-à-dire juste les afficher dans une page web). Nous avons également vu qu'il existant une multitude d'extensions au framework jQuery et qu'on pouvait trouver un module pour presque toutes les fonctionnalités qu'on voudrait développer et qui ne seraient pas forcément prévues nativement dans jQuery : valider un formulaire, en faciliter la saisie, etc. Aujourd'hui, nous allons voir qu'on peut récupérer des données un peu plus compliquées qu'un simple contenu textuel.

La majorité des sites (communautaires, en général) à fort trafic possèdent leur propre **API** (**A**pplication **P**rogramming **I**nterface). Mais une API, qu'est-ce-que c'est ? C'est un moyen mis à la disposition de développeurs te communiquer avec la base de données du site et d'en récupérer des informations. Par exemple, l'API d'Allociné permet de récupérer toutes les informations de n'importe quel film, l'API de VieDeMerde permet de récupérer des VDM, etc.

Dans ce TD, nous verrons :

- Quel format est utilisé pour communiquer avec ces APIs.
- Nous étudierons l'API en particulier, celle du service de microblogging Twitter.

1. JSON

JSON est un format de données textuel permettant d'architecturer des données. En gros, un objet JSON comprend une liste de plusieurs nœuds (ensemble «clé», «valeur») qui peuvent s'imbriquer ou non.

Voici un exemple d'objet au format JSON :

```
{
    "movie": {
      "title": "Avatar",
      "year": "2009",
      "casting": {
            "actors": {
                  {"name": "Sam Worthington", "character": "Jake Sully"},
                  {"name": "Zoe Saldana", "character": "Neytiri"},
                  {"name": "Sigourney Weaver", "character": "Dr. Augustine"}
            }
      }
    }
}
```

Comme vous le constatez, les données sont organisées par ensemble de paires comme pourrait l'être un tableau associatif. L'ensemble ci-dessous est une adaptation de comment seraient présentées les mêmes données au format XML (que nous avons déjà vu en créant une page XHTML) :

```
<movie>
      <title="Avatar" />
      <year="2009" />
      <casting>
            <actors>
                  <actor name="Sam Worthington" character="Jake Sully" />
                  <actor name="Zoe Saldana" character="Neytiri" />
                  <actor name="Sigourney Weaver" character="Dr. Augustine" />
            </actors>
      </casting>
</movie>
```

Ne paniquez pas, vous n'aurez pas à créer de fichier JSON. En tous cas, pas à la main. Vous l'aurez compris, il s'agit-là du format principal utilisé par les APIs les plus connues. Dont celle de Twitter et Facebook.

1.1 Facebook Graph API

La "**Graph API**" de Facebook permet d'accéder et d'interagir avec les objets et connexions.

- Objets : personnes, photos, événements, pages, groupes, messages, vidéos…
- Connexion : amitié, contenu partagé, "like"

L'API nous permettra d'accéder à ces différents objets pour ensuite utiliser les connexions et avoir accès à d'autres objets. Pour cela, Facebook propose un Web Service. Les requêtes se feront à l'aide d'une url et les réponses seront retournées au format JSON.

Par soucis de simplicité, Facebook ne propose qu'un seul point d'entrée :

```
http://graph.facebook.com
```

Le parcours du graphe se fait à partir de l'URL :

```
http://graph.facebook.com/identifiant
```

en précisant l'identifiant de l'objet visé.

Le résultat est obtenu au format JSON. JQuery permet de traiter les objets JSON facilement, grâce à la méthode **getJSON()**. Celle-ci ressemble beaucoup à la méthode Ajax vue précédemment :

```
$.getJSON(requestURL, function(data) {
    // Les données récoltées sont présents dans la variable data
});
```

La méthode **each()** permet de récupérer toutes les valeurs des attributs d'un objet JSON :

```
$.each( ObjetJSON , function(attribut, valeur) {
    …
});
```

Pour accéder à un sous objet au sein d'un nœud JSON : obj1.obj2

1.2 Exercice

- Dans votre navigateur, tapez l'URL suivant pour récupérer les informations publiques de la page youTube sur Facebook. Étudiez le contenu renvoyé:

  http://graph.facebook.com/youtube

- Utilisez la méthode getJSON() pour afficher les informations suivantes dans une liste sur votre page HTML :
  - nom de la page
  - date de création de la page
  - la description de la page
  - nombre de "like"s

2. Twitter et son API

2.1 Twitter, c'est quoi ?

Twitter est un réseau social, plus particulièrement un service de microblogging, qui permet aux utilisateurs de diffuser des messages courts (« tweets ») aux membres de son réseau. Le réseau d'un utilisateur est caractérisé par le nombre de personnes abonné à ses publications (« followers ») et le nombre de personnes auxquelles l'utilisateur est lui-même abonné (« followings »).

Dans ce TD, nous allons créer un widget pour récupérer les derniers tweets et les afficher aux côtés d'informations sur le compte de votre promotion, dont voici les identifiants :

- **Login :** imac2015_web
- **Mot de passe :** valentin

Le profil de cet utilisateur est par ailleurs visible à l'adresse https://twitter.com/imac2015_web.

2.2 L'API v1.1

Comme bon nombre de réseaux sociaux, **Twitter possède sa propre API**. Cette dernière permet d'effectuer des actions de toutes sortes sur un compte twitter donné, comme par exemple :

- Publier des messages.
- Récupérer le nombre de followings / followers.
- Récupérer les tweets postés.

Pour qu'un utilisateur puisse faire des requêtes, Twitter exige qu'il soit clairement identifié, dans le but de protéger la confidentialité des utilisateurs.

Si vous ne vous identifiez pas auprès de twitter, vos requêtes vont retourner le message d'erreur suivant:

```
<error code="215">Bad Authentication data</error>
```

Twitter utilise le protocole **oAuth** pour l'identification. Ce protocole accepte 4 paramètres, dont les valeurs sont uniques et qui doivent rester secrètes : Consumer key, consumer secret, Access token et Access token secret. Ces paramètres sont générés lors de la création d'une application sur le site des développeurs de Twitter. Une fois authentifié, le développeur peut envoyer des requêtes php. L'utilisation de php au lieu de Javascript (Ajax par exemple) permet de garder les valeurs des paramètres secrètes.

Dans ce TP, pour bien comprendre l'API Twitter sans écrire de code php, on va utiliser une console interactive. Elle est utile pour tester des requêtes et fournir un déboggage facile de votre code. Nous avons choisi la console **apigee** afin d'accéder aux données de l'api Twitter.

- Depuis le champs **Authentication**, choisissez l'option **oAuth 1**. Pour autoriser l'accès, veuillez vous identifier via votre compte Twitter.
- Dans le champs **Request URL**, gardez la methode **GET** et tapez :

```
https://api.twitter.com/1.1/statuses/user_timeline.json?count=[le nombre tweets à récupérer]&screen_name=[le nom du compte]
```

  - récupérer 10 tweets du compte imac2015_web
- Copiez le tableau des tweets renvoyé depuis la fenêtre **Reponse**, et stockez le dans un fichier *.txt* dans un dossier *ajax/*.

Vous pouvez constater que le résultat obtenu est en fait un objet au format JSON. Nous allons le parser (parcourir) afin d'en récupérer les informations qui nous intéressent. Pour résumer, voici l'organisation du JSON pour un tweet :

```
{                                   : Correspond au debut du tweet
      user                          : L'auteur du tweet
            screen_name             : Son nom
            profile_image_url       : Sa photo
            friends_count           : Nombre de followings
            text                    : Le contenu du tweet
}                                   : Correspond à la fin du tweet
```

La méthode getJSON() ne fonctionnera pas pour votre fichier .txt stocké en local. Vous allez donc devoir utiliser la méthode **ajax()** qui n'a maintenant plus aucun secret pour vous. Voici la méthode qui permet de récupérer un objet JSON exploitable :

```
$.ajax({
      url: 'ajax/twitter.txt',
      success: function(data) {
            var json = eval(data);
            /* Le traitement de votre JSON */
      }
});
```

La variable **json** contient donc l'objet attendu (parsé grâce à la fonction passe-partout **eval()**).

L'idée est maintenant de parcourir la variable *json* (grâce par exemple à la méthode **each**) et d'en retirer les informations suivant l'organisation décrite ci-dessus. A chaque itération sur data, vous aurez accès à un tweet !

3. A vous de jouer

A l'aide de toutes les informations données plus haut, c'est à vous de jouer... Vous devrez réaliser un widget regroupant tous les tweets du compte imac2015. Par exemple :

4. Quelques pistes

- Vous allez créer une div vide, dans laquelle vous allez injecter du code HTML (listes, div, tout ce que vous jugerez nécessaire) généré au fur et à mesure du parcours de l'objet JSON.

5. Aller plus loin

- Utilisez des effets jQuery pour animer votre widget.
- Créez un timer pour rafraîchir automatiquement votre widget toutes les minutes.
- Affichez la date de façon relative (« 1 day ago », ...).
- Détectez les liens vers d'autres utilisateurs (@) (et rediriger les).
- Faites de même pour les hashtags (#).