Programmation orientée objet

Introduction aux API REST

Qu’est-ce qu’une API?

- API : Application Programming Interface
- En termes généraux, une API est une interface de programmation applicative.
    - C’est en fait par une telle interface que les applications (Web ou autres) fournissent un accès à un sous-ensemble des fonctionnalités dont elles seules connaissent la logique interne. Elles sont généralement utilisées par d’autres applications qui veulent avoir accès à ces données / opérations privilégiées!
    - Exemples d’API :
        - API d’Adobe Acrobat (un programme « desktop » que l’on peut automatiser de façon à créer un PDF, par exemple)
        - API de Word ou Excel (pour créer des documents XLS ou DOC à l’aide de données dans une BD, par exemple)
    - Exemples d’API pour application Web (souvent payants) :
        - API de Google Maps
        - API de Facebook
        - API de Postes Canada
        - API d’Instagram
- Pour transmettre les données d’une application à l’autre, les API utilisent généralement le format JSON.
    - Voir exemple d’utilisation d’une API (OMDB)

JSON

- JSON : JavaScript Object Notation (json.org)
- Officiellement, JSON est le format standard pour représenter les objets dans JavaScript sous forme de chaine de caractères. Il est utilisé très souvent pour envoyer de l'information du serveur vers le client pour que de la nouvelle information soit affichée dans le fureteur.
    - Rappel : c'est le format utilisé pour déclarer des objets littéraux.

```
var professeur = {
"nom": "Harvey",
"prenom": "Guillaume",
"age": "36"
}
```

- JSON est maintenant utilisé indépendamment de JavaScript.
    - Plusieurs environnements permettent maintenant de lire (avec un *parser*) ou de générer du JSON.
- JSON peut exister sous forme d'objet ou sous forme de chaine de caractères, ce qui est très particulier.
    - Les langages (PHP et JS, entre autres) permettent généralement de passer d'une forme à l'autre.

Le format JSON

- Le format JSON est basé sur une combinaison de deux structures :
  - Des collections de paires noms-valeurs, en d'autres mots, un tableau associatif ou un dictionnaire.
    - Les paires sont séparées par des virgules, et le caractère séparant le nom de la valeur est le deux-points.
    - La collection, elle, est circonscrite par des accolades.
    - Ex : {"nom":"Harvey","prenom":"Guillaume"}
  - Des suites de valeurs ordonnées, en d'autres mots un tableau.
    - Les tableaux sont circonscrits par le crochet carré, et les éléments du tableau sont séparés par des virgules.
    - Ex : [1, 3, 6]
  - Les valeurs, elles, peuvent être d'une multitude de types :
    - string, nombre, objet, tableaux, true, false, null

Le format JSON (suite)

- Les deux types de structures peuvent être combinées pour former des tableaux d'éléments, ou encore un élément contenant des tableaux, et ainsi de suite…
- Cela permet à JSON d'être très flexible et de représenter une très grande quantité de structures différentes.
  - Voir exemple
  - Si vous désirez visualiser votre JSON, vous pouvez utiliser ceci :
    - http://jsonviewer.stack.hu/

Qu’est-ce qu’une API REST?

- REST signifie Representational State Transfer.
- Les API REST, ou services Webs RESTful, doivent respecter 5 contraintes pour être considérés comme RESTful.

1. Elle doit présenter une interface uniforme, c’est-à-dire que :
    - Chacune des ressources (un article, un film, un produit, etc…) doit être identifiée par un URI différent. Cet URI renvoie une représentation des données représentant la ressource, qui doit être du HTML, du XML ou, plus fréquemment, du JSON.
    - La manipulation des ressources se fait via cette URI. On supprimera, ou mettra à jour, via la même représentation de la ressource que celle qui est accédée pour lire la ressource.
2. Elle doit être sans état. Chaque requête est indépendante. Pas de variables sessions, donc.
3. Chaque réponse doit spécifier si elle peut être mise en cache.
4. Le client et le serveur sont entièrement séparés. Le client ne sait pas quelle est la source de données, le serveur ne sait pas ce que le client fera avec les infos et dans quelle interface elles s’afficheront.
5. Le client ne sait pas si le serveur possède les infos ou les obtient à partir d’un autre serveur, qui lui pourrait les obtenir d’un autre, etc.

API REST et méthodes HTTP

- L'interface uniforme décrite précédemment appelle à une utilisation des différentes méthodes HTTP pour déterminer les opérations que l'on veut effectuer sur une ressource.
  - En effet, comme une ressource est identifiée par un URI, ce même URI ne peut définir l'action que l'on veut effectuer!
- Un API REST typique fournit les mêmes méthodes qu'un CRUD typique. À chacune des actions du CRUD sont associées une des méthodes HTTP d'usage.
  - GET pour LIRE une ou des ressources.
  - POST pour CRÉER une ressource.
  - PUT pour MODIFIER une ressource.
  - DELETE pour SUPPRIMER une ressource.
- Dépendamment de l'action, une réponse différente sera envoyée et attendue. Dans le cas général, ces réponses seront en JSON, accompagnées d'un code de réponse HTTP approprié.
  - Regardons chacune de ces requêtes plus en détail.

Création avec POST

- La création s'effectue avec une requête de type POST vers la ressource dont on veut créer une instance.
    - Ex : monsite.com/articles
- Cette requête POST doit contenir, dans le corps de la requête, un objet JSON représentant toutes les caractéristiques de la ressource à créer.
- Ex :

```
{
  "article": {
    "titre": “Le canadien perd!”,
      "texte": “Le canadien perd en fusillade contre les Oilers…”,
    "idAuteur": 8
  }
}
```

- Il n'est pas facile d'accéder à du JSON dans les paramètres de la requête en PHP. Il est impossible de le faire via $_POST, par exemple, qui s'attend à être rempli par des variables très simples, via des couples noms-valeurs du style (key1=value1&key2=value2).
- On accède directement au corps de la requête en PHP via l'instruction suivante :
    - file_get_contents("php://input")
- Si la ressource a été créée, la réponse devrait contenir la ressource créée en JSON ainsi qu'un code de réponse HTTP 201 CREATED. Si la ressource existe déjà, le code de réponse devrait être 409 Conflict.

Lecture avec GET

- Lorsque l'on veut faire une lecture sur tous les items de la ressource que l'on veut accéder, la lecture s'effectue avec une requête de type GET vers la ressource que l'on veut obtenir.
    - Ex : monsite.com/articles
- La réponse devrait contenir tous les items de la ressource dans un tableau JSON ainsi qu'un code de réponse HTTP 200 OK.
- Ex :


```
{
  "articles": [{
    "titre": “Le canadien perd!”,
      "texte": “Le canadien perd en fusillade contre les Oilers…”,
    "idAuteur": 8
  },
{
    "titre": “Le canadien perd encore!”,
      "texte": “Le canadien perd 3-2 contre les Flames…”,
    "idAuteur": 9
  }]

}
```

Lecture avec GET (suite)

- Lorsque l'on veut faire une lecture sur un seul item de la ressource que l'on veut accéder, la lecture s'effectue avec une requête de type GET vers la ressource que l'on veut obtenir, suivi de l'ID de l'item,
    - Ex : monsite.com/articles/9
- La réponse devrait contenir tous les items de la ressource dans un tableau JSON ainsi qu'un code de réponse HTTP 200 OK, ou un code 404 NOT FOUND si l'item n'a pas été trouvé.
- Ex :

```
{
  "article":
     {
    "titre": “Le canadien perd encore!”,
       "texte": “Le canadien perd 3-2 contre les Flames…”,
    "idAuteur": 9
  }
}
```

- Dans ce cas particulier, il peut être utile de modifier le fichier .htaccess pour effectuer un routage via la ré-écriture d'URL.
    - La ré-écriture d'URL est un sujet passablement complexe, mais un exemple simple, qui suffira pour une interface REST de base, vous sera fourni.

Mise à jour avec PUT

- La mise à jour s'effectue avec une requête de type PUT vers la ressource dont on veut créer une instance.
  - Ex : monsite.com/articles
- Cette requête PUT doit contenir, dans le corps de la requête, un objet JSON représentant toutes les nouvelles caractéristiques de la ressource à modifier. La modification est toujours effectuée en fonction de la clé primaire.
- Si la ressource a été modifiée, la réponse devrait contenir un code de réponse HTTP 200 OK. Si elle n'a pas été trouvée, le code devrait être 404 NOT FOUND.

Suppression avec DELETE (suite)

- Lorsque l'on veut faire la suppression d'un seul item de la ressource que l'on veut accéder, la suppression s'effectue avec une requête de type DELETE vers la ressource que l'on veut supprimer, suivi de l'ID de l'item.
  - Ex : monsite.com/articles/9
- La réponse devrait retourner un code 204 NO CONTENT si l'item a été trouvé et supprimé, et un code 404 si l'item n'a pas été trouvé.