**Reset API**

"Les API REST fournissent un accès programmatique pour lire et écrire des données. Lisez le profil de l'auteur et les données, et plus encore. L'API REST identifie les applications et les utilisateurs utilisant OAuth; les réponses sont disponibles dans JSON. "

|  |  |
| --- | --- |
| Title | Nom\_Api |
| Url | Pour les Url fixes: /user ou /photo  Pour les Url Contenant des paramètres : /user/ :id ou /photo/ :id |
| les méthodes(Entête) | Get, Post,Put/Patch/,Delete |
| Url Params | Si des paramètres d'URL existent, spécifiez-les en fonction du nom mentionné dans la section URL. Séparer en option et requis. |
| Data Params | Si vous faites une demande de publication, à quoi devrait ressembler la charge utile du corps? C'est un bon moment pour documenter vos différentes contraintes de données aussi. |
| Success response | Quel devrait être le code de statut en cas de succès et y a-t-il des données renvoyées? Ceci est utile lorsque les gens ont besoin de savoir ce que leurs rappels devraient attendre! |
| Error response | La plupart des points de terminaison auront plusieurs façons d'échouer. De l'accès non autorisé, aux paramètres erronés, etc. Tous ceux-ci devraient être énumérés ici. Cela peut sembler répétitif, mais cela aide à éviter que des hypothèses ne soient faites là où elles ne devraient pas l'être. |
| Sample call | Juste un exemple d'appel à votre point de terminaison dans un format exécutable (appel $ .ajax ou une demande de curl) - cela rend la vie plus facile et plus prévisible. |

Exemple

|  |  |
| --- | --- |
| Url | /user/ :id |
| Méthode | Get |
| Url params | Required id=[integer] |
| Data params | None |
| Success reponse | Code :200  Contient :{id : 12 ,name : ‘exemple’} |
| Error reponse | Code : 404 notfound  Contient :{error ,’user doesn’t exist’} |
| Sample call | $.ajax  ({  [url: ’/users/id](url:'/users/id)’,  datatype:’json’,  type:’Get’,  success :function(r){  console.long(r) ;  }  }) |
|  |  |

Remarque: essayez d'utiliser des verbes qui correspondent à la fois au type de requête (extraction vs modification) et à la pluralité (une par multiple).

**Url formalise**

|  |  |  |
| --- | --- | --- |
| Url | Méthode | Action |
| /resources | Get | Liste des resources |
| /resources/ :id | Get | Récupère la Resource :id |
| /resources/ : | Post | Ajouter une resource |
| /resources/ :id | Delete | Supprimer la resource :id |
| /resources/ :id | Put/Patch | Modifier la resource :id |

**Les Méthodes**

|  |  |
| --- | --- |
| Methode | Action |
| Get | Les méthodes utilisées get récupération des informations |
| Post | ajouté des informations |
| Delete | supprimé des informations |
| Put/Patch | modification des données |

**Réponse serveur**

La reponse de serveur doit etre en json ou xml

|  |  |
| --- | --- |
| Code XML | Code json |
| <country>  <city>  <temp>33</temp>  <humidity>40</humidity>  </city>  </country> | {  ‘Country‘:  {  ‘City’ :{  ‘Temp’ :33,  ‘Humidity’ :40  }  }  } |

**Exemple**

Voici une requête GET lancée à l’API Instagram pour obtenir les photos d’un certain lieu à partir de sa longitude et sa latitude : dans l’exemple, c’est à New York (40.7127° N, 74.0059° W) :

**GET/v1/locations/search?access\_token=ACCESS\_TOKEN&lat=40.7127&lng=74.0059**

La réponse d’Instagram, sous forme de fichier JSON :

|  |
| --- |
| HTTP/1.1 200 OK  {   "meta": {     "code": 200   },   "data": [     {       "latitude": 40.714198749,       "id": "93496093",       "longitude": 74.006001183,       "name": "John's Pizzeria 278 Bleecker St NY, NY"     },     {       "latitude": 40.7142,       "id": "46371155",       "longitude": 74.0064,       "name": "Thunderpocalypse 2012"     },     {       "latitude": 40.714201754,       "id": "35932492",       "longitude": 74.006397137,       "name": "Avenue of the Americas, New York"     },     {       "latitude": 40.71296389,       "id": "1023103828",       "longitude": 74.00388611,       "name": "Manhattan Municipal Building"     },     {       "latitude": 40.71322,       "id": "92582758",       "longitude": 74.003963,       "name": "Sleepers Filming Location"     },     {       "latitude": 40.716833,       "id": "97921846",       "longitude": 74.005833,       "name": "Atera"     }   ]  **}** |

**Structure d’un fichier JSON**

Les éléments de JSON sont:

1. Un objet: contient d'autres objets ou des variables.
2. Une variable scalaire: Number, String, Boolean.
3. Un tableau.
4. Les valeurs litérales: *null*, *true*, *false*, chaîne de caractères, et les valeurs numériques.

**L’objet**

Il contient un membre ou listes des membres chaque membre étant de la forme :

|  |
| --- |
| ‘nom’ : ‘valeur’ |

La syntaxe de l’objet et

{membre1, membre2, …}

**Le Tableau :** contient un ou plusieurs valeurs séparées

|  |
| --- |
| [valeur, valeur, …] |

**Les valeurs :** une valeur peut être un objet, un tableau, un litéral (chaine, true, nomber, false, null, …)

Exemples de fichier Json

|  |
| --- |
| {  "menu": "Fichier",  "commandes": [  {"titre": "Nouveau",  "action":"CreateDoc"},  {"titre": "Ouvrir",  "action": "OpenDoc"},  {"titre": "Fermer",  "action": "CloseDoc"}]  } |

L’équivalence en XML

|  |
| --- |
| <?xml version="1.0" ?>  <racine>  <menu>Fichier</menu>  <commandes>  <item>  <titre>Nouveau</titre>  <action>CreateDoc</action>  </item>  <item>  <titre>Ouvrir</titre>  <action>OpenDoc</action>  </item>  <item>  <titre>Fermer</titre>  <action>CloseDoc</action>  </item>  </commandes>  </racine> |