FICHE SAVOIRS
L'API REST

Contenu

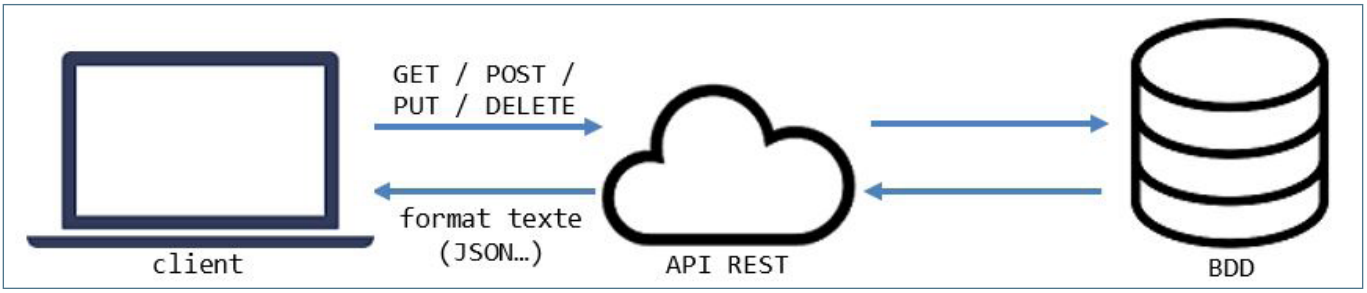
1. Principe 1
2. Standards et bonnes pratiques 1
3. Utilisation d'une API REST 3
4. Création d'une API REST 3

Pour rappel API signifie Application Programming Interface. Une API rend disponible des données et des fonctionnalités à d'autres applications (clientes).

Contrairement à SOAP qui est un protocole, REST (REpresentational State Transfert) est un style d'architecture. REST a été créé pour pallier certains problèmes de SOAP : lenteur et manque de flexibilité.

1. Principe

Une API REST va répondre à un besoin d'un client. Il représente un intermédiaire (interface) entre le client est les données à manipuler.



L'API REST reçoit les requêtes HTTP du client dans différents modes, exploite une BDD (ou autre) pour répondre à la requête, renvoie au client une réponse dans un format léger (texte), généralement le format JSON.

2. Standards et bonnes pratiques

2A. Les 6 règles du standard

Le standard RESTFul respecte 6 principes. On parle d'API RESTFul si les 6 principes sont respectés, d'API REST si certains principes sont respectés.

1) **Séparation du client et du serveur** : les 2 peuvent être modifiés sans incidence (à condition que l'interface, donc les conditions d'appels, ne change pas).

2) **Sans état** (stateless) : pas de conservation de session entre 2 requêtes. Lors d'un nouvel appel au serveur (à l'API), celui-ci n'a pas conservé les appels précédents.

3) **Uniformité de l'interface** : les actions possibles sont clairement définies et chaque réponse se suffit à elle-même.

4) **Mise en cache possible des réponses** pour éviter la surcharge.

5) **Architecture en couches** : le client se connecte à l'API REST uniquement et c'est l'API qui gère ensuite la communication avec d'autres serveurs (de données...).

6) (optionnel) Possibilité de retourner du code exécutable (javascript, applet...) : simplification du code client.

2B. Les bonnes pratiques

Un ensemble de bonnes pratiques est donné en ce qui concerne la construction d'une API REST. Ces bonnes pratiques découlent généralement des principes précédents. En voici quelques-unes :

1) Pour déterminer le type d'opération à réaliser, **utiliser les verbes HTTP** :
   - GET : (select) lecture d'une information
   - POST : (insert) insertion d'une information
   - PUT : (update) modification d'une information
   - DELETE : (delete) suppression d'une information

Remarque

Jusqu'à maintenant, vous ne connaissiez probablement que GET et POST à travers les formulaires HTML, en faisant le choix de l'une ou l'autre méthode de transfert, juste par rapport à la volonté d'avoir les informations cachées (POST) ou visibles dans l'url (GET).
Ici, on utilise plus de verbes HTTP et chacun dans un but précis.
Attention, ce n'est pas une obligation : juste une bonne pratique.

---

2) **L'URL doit représenter la requête.**

   Écriture conseillée :

   `adrapi/commande/valeur…`
   `plutôt que :`
   `adrapi/commande?variable=valeur`

   'commande' représente le point d'accès (endpoint). En réalité, le 'endpoint» représente l'accès, donc l'URL complète.

   Pour cette syntaxe, il faut utiliser la réécriture d'URL.

3) **Le retour est sous un format JSON** ou autre format texte (léger), avec classiquement une structure :

   "success" contenant "code" et "message" (ou les 2 informations séparées)
   "result" contenant l'information retournée (si nécessaire)

4) **Au niveau du retour, utiliser les codes standards**. Exemples :

   200 : OK
   201 : Created
   400 : Bad Request
   404 : Not Found
   ...

Remarque

Les codes sont classés en groupe :
1xx : information
2xx : succès
3xx : redirection
4xx : erreur client
5xx : erreur serveur
Vous pouvez retrouver les codes ici :
https://www.restapitutorial.com/httpstatuscodes.html

---

3. Utilisation d'une API REST

Chaque API a ses propres conventions (mode opératoire pour l'appeler, l'utiliser) généralement présentées dans une documentation.

3A. Où trouver une API REST ?

Il faut connaître son adresse sur internet (ou dans un réseau privé) et surtout l'adresse de sa documentation.

3B. Quelle technologie cliente utiliser ?

Toutes les technologies sont possibles.

3C. Comment l'invoquer ?

Son invocation est nettement plus facile que pour un web service SOAP : il n'y a pas d'intermédiaire (fichier WSDL). Il suffit d'accéder à l'adresse de l'API, via HTTP, en utilisant la bonne syntaxe donnée dans la documentation.

3D. Comment l'exploiter ?

Le fait d'invoquer l'API permet d'obtenir un résultat, généralement sous format JSON, qu'il suffit de décoder et exploiter. Là aussi, c'est la documentation de l'API qui va donner les éléments de syntaxe.

4. Création d'une API REST

4A. Quelle technologie serveur utiliser ?

Créer une API REST revient à créer une application qui accepte les requêtes HTTP. C'est possible dans n'importe quel langage.

4B. Comment la déployer ?

Il suffit de mettre en ligne l'API et sa documentation pour expliquer son utilisation.

CNED, BP 60200, 86980 Futuroscope Chasseneuil Cedex, France

87D22TEWB1621