INF5190 - API Web

Jean-Philippe Caissy

25 septembre 2019

API Web

API Web : Une interface de programmation exposé à travers une application web

L'interface peut être exposé publiquement et accessible à tous, ou être derrière un système d'authentification

API Web

RPC

RPC : *Remote Procedural Call*
Un appel procédural distant est fait lorsqu'un programme informatique appelle une méthode en dehors de son espace d'adressage :

- Différent processus, sur le même système
- Système différent à travers le réseau

Un RPC décrit une abstraction où le client n'est pas au courant de l'appel distant. C'est comme si c'était local.

API Web

RPC

1. Le client appelle une méthode faisant partie d'une interface
2. Les données en arguments de la méthode sont sérialisés
3. Le client envoie la requête au service distant
4. Le serveur reçoit la requête et dé-sérialise les paramètres
5. Le serveur exécute la méthode
6. Le serveur sérialise les paramètres de retour
7. Le serveur envoie la réponse au client
8. Le client reçoit la réponse de la fonction

API Web

RPC

RPC est un protocole de type requête-réponse.
Le client et le serveur doivent s'entendre sur :

- Format de sérialisation
- Protocole de communication
- Mécanisme d'authentification (si besoin)

API Web

RPC

SOAP

- Utilise XML pour sérialiser les données
- Mécanisme d'authentification : HTTP
- Standardisé par le W3C
- Composé d'une enveloppe, d'entêtes et d'un corps de requête

API Web

RPC

SOAP

Avantages :

- Standardisé
- Rigide
  - Vérification des types
  - Validation XML
- Découverte des services (WSDL)

Désavantages :

- Standard très lourd et complexe
- Difficile à implémenter, besoin de plusieurs composantes
- Les requêtes et réponses sont difficile à lire pour un humain

API Web

REST

REST : *Representational state transfer*

Patron architecturale utilisant les URI comme ressources et les méthodes HTTP comme actions

1. Architecture client-serveur
2. Protocole sans état
3. Requêtes et réponses peuvent être mise dans une cache
4. Système de couches
5. Possibilité au serveur de transmettre du code à exécuter (optionnel)
6. Interface uniformisée

API Web

REST

Architecture client-serveur

- Séparation clair des responsabilités :
  - L'interface client n'est pas au courant de la persistance des données
- Permet une meilleur portabilité à travers plusieurs systèmes (e.g. : application mobile, site Web, etc)

API Web

REST

Protocole sans état

- La communication client-serveur ne partage aucun état entre les connexions d'un client
- Chaque requête d'un client doit contenir toutes les informations pour traiter la requête
- La session serveur peut être transféré au client

API Web

REST

Cache

- Des mises en cache intermédiaire sont possible (Proxy)
- Les requêtes et réponses doivent explicitement indiquer les possibilités de caching de la éponse
- Une bonne solution de mise en cache permet d'éliminer des requêtes et d'améliorer la performance des clients

API Web

REST

Système de couches

- Un client ne sait pas s'il est connecté à un serveur unique, ou a un proxy, ou un balanceur de charge
- Permet au système d'évoluer avec l'ajout de plus de serveurs sans changer le client
- Le système de couche permet de séparation la responsabilité de plusieurs composantes :
  - e.g.: authentification

API Web

REST

Transmission de code au client

- Un serveur peut étendre les fonctionnalités d'un client en retournant du code à exécuter
  - e.g. : Javascript à faire exécuter par le client

API Web

REST

Interface uniformisée

- Identification des ressources dans la requête (e.g.: URI)
- Mécanisme de manipulation (e.g.: méthode HTTP: GET, POST, etc)
- Type de contenu clair : (e.g.: `Content-Type` pour représenter du JSON, XML, etc)

API Web

REST

Une application Web qui répond aux contraintes du patron REST sont des API *RESTful*

- Une URI de base, e.g.: `http://example.com/produits/`
- Méthode HTTP standard (GET, POST, PUT, DELETE)
- Un type de média qui définit le changement d'état d'une ressource

API Web

REST

Exemples à ne pas faire

- `GET http://api.example.com/services/?id=123&op=fetch`
- `GET http://api.example.com/services/?operation=new_product`
- `GET http://api.example.com/delete_product/123`
- `GET http://api.example.com/blog_post_comments/123`
- `GET http://api.example.com/products/123/update`
- `PUT http://api.example.com/products/123/update`

API Web

REST

Exemple d'une API RESTful

- `GET http://api.example.com/products/123`
- `POST http://api.example.com/products/`
- `DELETE http://api.example.com/products/123`
- `DELETE http://api.example.com/products/123`

API Web

REST

URI

Respecter l'hiérarchie des ressources.

- `GET http://api.example.com/blog/123/comments`
- `GET http://api.example.com/blog/123/comments/11`
- `POST http://api.example.com/blog/123/comments`
- `POST http://api.example.com/products/123/images`
- `PATCH http://api.example.com/products/123/images/333`

API Web

REST

Méthodes HTTP

Utiliser les méthodes HTTP standards pour indiquer l'opération sur la ressource

- GET : récupérer
- DELETE : supprimer
- POST : nouveau
- PATCH/PUT : modifier

API Web

REST

Codes de retour HTTP

Méthode	Erreurs
`GET`	200 (OK), 404 (Not Found)
`POST`	201 (Created), 404 (Not Found), 400 (Bad Request), 422 (Unprocessable Entity)
`PUT`	200 (OK), 204 (No Content), 400 (Bad Request), 422 (Unprocessable Entity)
`DELETE`	200 (OK), 404 (Not Found)

API Web

Documentation

- Une documentation précise, claire et avec des exemples est primordial
- Malheureusement peu standardisé

API Web

REST

Exercice

Créer une API RESTful pour gérer l'inscription des étudiants à des cours

Composantes

- Étudiant
  - Code MS (identifiant unique)
- Cours
  - Sigle du cours (identifiant unique)
  - Session (Automne, Hiver, Été)

Liens

- Stripe : exemple d'API REST très bien documentée
- What is REST?
  - Part 1
  - Part 2