Bases de données documentaires et distribuées, http://b3d.bdpedia.fr

Web, REST et CouchDB

Philippe Rigaux

le cnam

Ce qu'il faut savoir (pour nous)

Web = immense base de données documentaire ! Les notions essentielles.

- Le Web est constitué de **ressources**.
  Une ressource est une entité fournissant des services et communiquant par message.
- Les ressources sont identifiées et accessibles à des **URL**.
  `https://www.example.com:443/chemin/vers/doc?nom=b3d&type=json#fragment`
- Les messages sont codés selon un **protocole**, HTTP.
  Très utile (indispensable ?) : l'outil cURL pour parler le HTTP avec la ligne de commande.
- Un "message" est une enveloppe dont le contenu est un **document**.
  Souvent le document est en HTML (affichage). Mais cela peut être un **document structuré**.

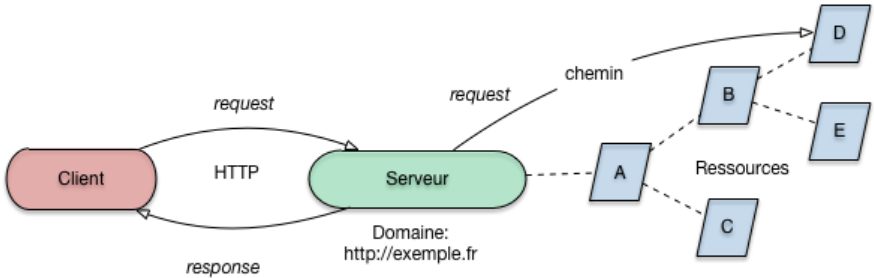
L'architecture REST

REST = une définition de services Web basée sur les standards du Web.

- on s'adresse à des ressources qui fournissent des services ;
- les messages se font en HTTP, et sont (le plus souvent) **structurés** en XML ou JSON ;
- les documents structurés fournis par une ressource sont (le plus souvent) produits **à la volée** (par **calcul** : distinction entre service et ressource statique).

Très répandu ! Excellent moyen de récupérer des informations directement exploitables par une machine (par comme HTML).

Ressources et opérations



Quatre opérations (celles de HTTP).

`GET` **lit** (la représentation d')une ressource.

`PUT` **crée** une nouvelle ressource (ou, plus flexible : la remplace).

`POST` envoie un message (**demande de service**) à une ressource.

`DELETE` **détruit** une ressource.

Un peu de rigueur

Qu'est-ce qu'un bon (un vrai) service REST ?

- Des ressources clairement organisées **et stables**.
- Des services bien conçus (simples, clairs, concis) **et stables**.
- Respect de la sémantique des 4 opérations. Notamment :

  `GET` : **lecture seule**, pas d'effet de bord (état des ressources inchangé).
  `PUT` : **création**, donc idempotent (plusieurs appels = même effet qu'un seul).
  `POST` : **envoi de message** : s'adresse à une ressource existante (modification possible de l'état des ressources).

Ce sont des principes : détails sujets à de longues discussions...

Où sont les services REST ?

Ils sont partout ! Beaucoup d'applications Web (Twitter, Facebook par exemple) ont une interface HTML **et** une interface REST.

Essayons : quel temps fait-il à Paris aujourd'hui ?

```
curl -X GET api.openweathermap.org/data/2.5/weather?q=Paris
```

Et à Londres ? Et ailleurs ?

Vous essaierez (au moins) les services de géolocalisation de Google.

Ou ceux de velib : https ://developer.jcdecaux.com/#/opendata/vls

De quoi se constituer au fil du temps une base de documents à analyser.

CouchDB, en bref

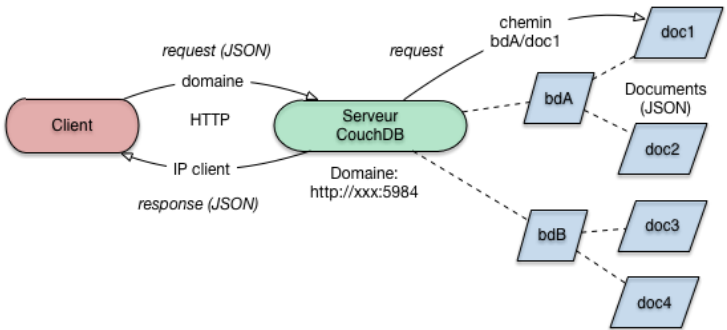
Un système typiquement NoSQL.

- gère des documents structurés (JSON) ;
- pas de schéma, pas de langage d'interrogation, entièrement **non** normalisé.
- échanges client/serveur basés sur REST ;
- passe en mode distribué facilement (à vérifier).
- applique quelques techniques courantes de calcul distribué (MapReduce).

Pas le système le plus répandu, mais facile à mettre en œuvre, intéressant à étudier.

Serveurs, bases, documents

Souvenez-vous : on s'adresse à des **ressources**



Démonstration (REST)

Vérifions que le serveur (ici, machine locale) est prêt à nous parler.

```
$ curl -X GET http://localhost:5984
{"couchdb":"Welcome","version":"1.0.1"}
```

Création d'une base de données = `PUT` d'une nouvelle ressource.

```
$ curl -X PUT http://localhost:5984/films
```

Récupérons la nouvelle ressource (sa représentation) avec `GET`.

Création d'une base de données = `PUT` d'une nouvelle ressource.

```
$ curl -X GET http://localhost:5984/films
```

Création d'un document = `PUT` d'une nouvelle ressource dans la base de données.

```
$ curl -X PUT http://localhost:5984/films/doc1
      -d '{"clef": "valeur"}'
{"ok":true,"id":"doc1","rev":"1-25eca"}
```

Notez le champ `rev`. Et bien sûr on peut lire la nouvelle ressource.

```
$ curl -X GET http://localhost:5984/films/doc1
```

Ajout de documents complets

Envoi de messages PUT (à la nouvelle ressource) ou POST (à la base de données).
On adresse un PUT à une **nouvelle** ressource (il faut donc produire l'URL).

```
curl -X PUT http://localhost:5984/films/us
         -d @movie_52.json -H "Content-Type: application/json"
```

On adresse un POST à la base de données.

```
$ curl -X POST  http://localhost:5984/films/ -d @movie_52.json \
      -H "Content-Type: application/json"
```

Réponse :

```
{
 "ok":true,
 "id":"movie:52",
 "rev":"1-68d58b7e3904f702a75e0538d1c3015d"
}
```

Mise à jour

CouchDB est un système **multiversions**

Mettre à jour un document = ajouter une version à une version existante ;

```
$ curl -X PUT http://localhost:5984/films/us -d @newDoc.json
        -H "Content-Type: image/jpg"
{"ok":true,"id":"tsn","rev":"2-26863"}
```

Détruire avec `DELETE` = ajouter une version marquée "détruite".

```
$ curl -X DELETE http://localhost:5984/films/us?rev=2-26863
{"ok":true,"id":"tsn","rev":"3-48e92b"}
```

Bilan

À retenir :

- REST = échange de documents sur le Web.
- Possibilité de constituer à peu de frais une base documentaire.
- Opérations de type "dictionnaire" : le minimum syndical des bases NoSQL.
- CouchDB : un système original concrétisant une vision de Web comme un serveur de documents structurés.

À vous de jouer : reproduire les commandes CouchDB, trouver et interroger des services REST en JSON ou XML.