UE AD FIL A1 - API REST

2022-2023

Hélène Coullon

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Table of Contents

Avant de démarrer

Rappels sur HTTP

HyperText Transfer Protocol

Développé par Tim Berners-Lee au CERN en 1989 (Suisse) avec d'autres concepts qui ont servi de base à la création du World Wide Web (HTML et URI)

- Protocole de communication réseau
- Couche applicative du modèle OSI
- Protocole de type client/serveur requête/réponse

Les méthodes de HTTP

Une requête HTTP est constituée des éléments suivants:

- la ligne de requête : `METHOD URL PROTOCOL-VERSION`
- des éléments d'en-tête optionnels
- un corps de message optionnel

Les méthodes les plus utilisées sont les suivantes :

- GET : obtenir une ressource située à l'URL spécifiée
- POST : envoi de données à l'URL spécifiée
- PUT : envoi de données à l'URL spécifiée (typiquement pour une mise à jour)
- DELETE : suppression de la ressource à l'URL spécifiée

Les versions de HTTP (1/2)

- 1991 HTTP/0.9
  - permet de récupérer le contenu (GET) d'une page HTML uniquement
  - la connexion TCP/IP est toujours fermée aprèsl'envoi de la réponse
- 1996 HTTP/1.0
  - permet de récupérer (GET) tout type de fichiers
  - la connexion TCP/IP est toujours fermée aprèsl'envoi de la réponse
- 1997 HTTP/1.1
  - conservation de la connexion
  - pipelining (plusieurs requêtes avant la réponse)
  - permet d'envoyer des informations au serveur (POST/PUT)
  - TRACE (suivi du chemin entre le client et le serveur)
  - caching (conservation du contenu en mémoire tampon)
  - ...

Les versions de HTTP (2/2)

- 2015 HTTP/2.0
  - données binaires au lieu de texte
  - requêtes parallèles côté client et serveur
  - compression des en-têtes de messages
  - PUSH (envoi en cash client de données à l'avance)
- 2020 HTTP/3.0 (draft)
  - UDP au lieu de TCP

Formats JSON et YAML

- Formats standardisés de données textuelles
- Représentation structurée de l'information textuelle
- Facilement lisible par un être humain
- Formats sérialisables
  - communication sur le réseau
- Formats comparables à XML mais plus légers

Formats JSON et YAML

JSON

- JavaScript Object Notation
- Exemples d'utilisation :
  - Requêtes et réponses HTTP
  - Bases de données NoSQL

YAML

- Yet Another Markup Language - (version 1.0)
- YAML Ain't Markup Language - (version 1.1)
- Exemples d'utilisation :
  - Fichiers configuration
  - Spécifications

Exemples JSON et YAML

Exemples Json et Yaml

API REST

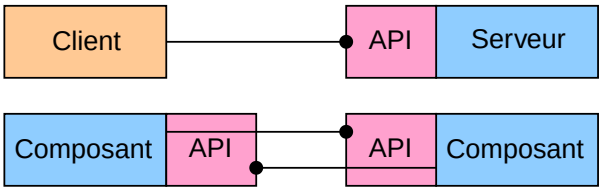
Faire communiquer des composants

- Coder à la main l'utilisation de protocoles réseau ?
  - TCP/IP socket, websocket, HTTP etc.
  - Non ! Besoin d'abstraction !
- Comment connaître les interactions possibles avec les autres composants ?
  - Besoin d'une API !

Utilisation de bibliothèques et de frameworks pour régler ces questions !

API

Application Programming Interface



Accord ou contrat entre un client et un serveur sur les interactions possibles avec le serveur

REST

REpresentational State Transfer (Roy Fielding 2000)

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Tirer profit de la flexibilité du web pour le design d'applications et de systèmes distribués.

- Penser son application/système comme une page web.
- Les ressources sont demandées au serveur sous la forme d'une requête HTTP
- Une ressource correspond à une URL
- REST utilise les méthodes standardisées de HTTP/1.1

Ex : `GET http::://adress/resource HTTP/1.1`

Découverte de l'API REST

On peut évidemment documenter son API (c'est ce qu'on va voir ensuite)

→ MAIS comment rendre possible la découverte automatique de l'API sans intervention humaine ?

C'est le niveau ultime recherché avec REST et qui n'est pas vraiment résolu

Article pour aller plus loin :
https://www.martinfowler.com/articles/richardsonMaturityModel.html

Tutoriel

Installation des environnements

Tutoriel Flask

Documenter son API REST

Pourquoi documenter son API ?

- Source de référence précise sur les intéractions possibles avec le serveur
  - cahier des charges, spécification
- Outil et guide pour les utilisateurs de l'API
- Meilleure adoption de l'API par les utilisateurs

Pourquoi suivre un standard ?

- Niveau d'information adequat et standardisés
- Compréhensible des humains connaissant le standard
- Plus facilement compréhensible pour des outils automatiques

Exemples de standards

- OpenAPI (anciennement Swagger)
  - https://swagger.io/docs/specification/about/
- RAML
  - https://raml.org/

Tutoriel OpenAPI

On passe au TP !

TP !