Architecture Orientée Services
Services REST*

Licence Professionnelle IDSE
2015-2016

IUT de Nice

Simon Urli
urli@i3s.unice.fr

Ce qu'on cherche à obtenir

Mais comment faire ?

1. Objets
2. Composants
3. Services

Objets

- Encapsulation de méthodes et d'attributs au sein d'un élément de programmation unitaire

- Nécessite d'être compilé ou interprété
- Réalisé pour être utilisé en interaction avec d'autres objets

➡ Assemblage à grain très fin

Composants

- Différentes notions de composants : bibliothèques logicielles, paquets, plugin, etc
- Un composant est considéré comme une boîte noire utilisable immédiatement à travers ses interfaces
- Un composant est **inclus** dans le logiciel

➡ Assemblage à gros grain

Services

- Un service est un composant **externe** au logiciel
- Dédié à réaliser une fonctionnalité spécifique
- Utilisable également en boîte noire à travers des interfaces
- Nécessite de spécifier la **sérialisation** des informations

➡ Assemblage à gros grain et distribué

Architecture Orientée Service

- Comment faire passer les messages ?
- Comment retrouver un service et connaître ses interfaces ?
- Comment manipuler les données récupérées ?

Faire passer les messages

- Services = systèmes distribués
- Moyen de communication : le réseau
- Plusieurs protocoles : RPC, JRMP (pour RMI), IIOP (pour CORBA), etc

  ➡ Au dessus de TCP/IP

- Ou le Web ! D'où les WebServices (SOAP et REST)

  ➡ Au dessus de HTTP

Retrouver un service et décrire les interfaces

- UDDI : Annuaire de service

➡ Centralisé, interrogeable pour obtenir les informations d'un service

- WSDL : Fichier xml contenant toutes les informations associées à un services web

➡ Pour chaque service, recommandé par le W3C, la norme avec SOAP

Manipuler les données

- WSDL : Information sur les types de données
- Sérialisation en XML ou JSON

Services Web

- SOAP :
  - Standardisé par l'OMG
  - Utilisé dans les gros systèmes
  - Assez lourd à déployer
- REST* :
  - Pas de standard mais des pratiques
  - Utilisé (plus ou moins bien) dans de nombreuses API ouvertes sur le web (Twitter, FlickR, Facebook, Instagram, ...)
  - Léger à déployer

API RESTful

- Client-Serveur

GET statuses/retweets/:id

Returns a collection of the 100 most recent retweets of the tweet specified by the `id` parameter.

Resource URL

`https://api.twitter.com/1.1/statuses/retweets/:id.json`

Resource Information

Response formats	JSON
Requires authentication?	Yes
Rate limited?	Yes
Requests / 15-min window (user auth)	15
Requests / 15-min window (app auth)	60

API RESTful

- Client-Serveur
- Pas d'état

GET statuses/user_timeline

Returns a collection of the most recent Tweets posted by the user indicated by the `screen_name` or `user_id` parameters.

User timelines belonging to protected users may only be requested when the authenticated user either "owns" the timeline or is an approved follower of the owner.

The timeline returned is the equivalent of the one seen when you view a user's profile on twitter.com.

API RESTful

- Client-Serveur
- Pas d'état
- Mise en cache

Rate Limits: Chart

Title	Resource family	Requests / 15-min window (user auth)	Requests / 15-min window (app auth)
GET application/rate_limit_status	application	180	180
GET favorites/list	favorites	15	15
GET followers/ids	followers	15	15

API RESTful

- Client-Serveur
- Pas d'état
- Mise en cache
- Orienté ressources

GET statuses/retweets/:id

Returns a collection of the 100 most recent retweets of the tweet specified by the `id` parameter.

Resource URL

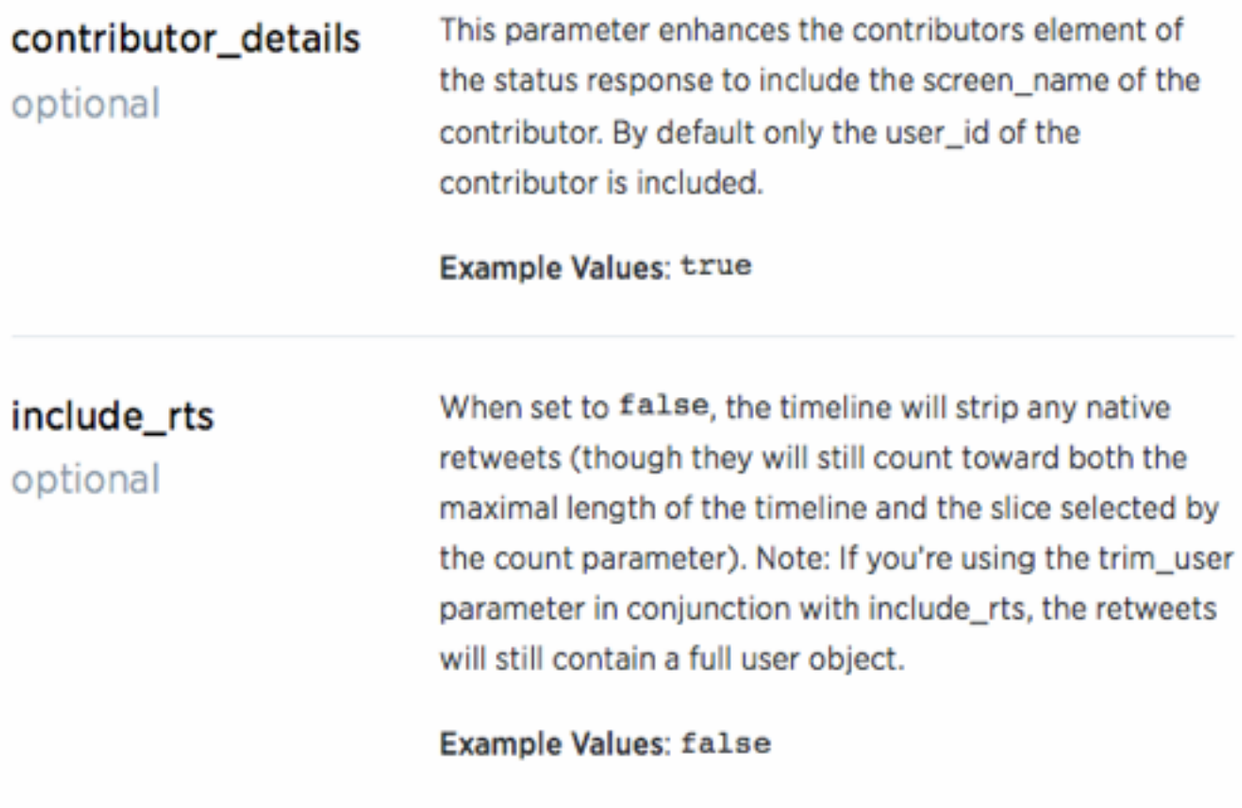
`https://api.twitter.com/1.1/statuses/retweets/:id.json`

Resource Information

Response formats	JSON
Requires authentication?	Yes
Rate limited?	Yes
Requests / 15-min window (user auth)	15
Requests / 15-min window (app auth)	60

API RESTful

- Client-Serveur
- Pas d'état
- Mise en cache
- Orienté ressources
- Ressources organisées en couches



contributor_details
optional

This parameter enhances the contributors element of the status response to include the screen_name of the contributor. By default only the user_id of the contributor is included.

Example Values: `true`

include_rts
optional

When set to `false`, the timeline will strip any native retweets (though they will still count toward both the maximal length of the timeline and the slice selected by the count parameter). Note: If you're using the trim_user parameter in conjunction with include_rts, the retweets will still contain a full user object.

Example Values: `false`

API REST**like**

- Client-Serveur
- Pas d'état
- ~~Mise en cache~~
- Orienté ressources
- ~~Ressources organisées en couches~~

API **Web**

- Client-Serveur
- ~~**Pas d'état**~~
- ~~Mise en cache~~
- Orienté ressources
- ~~Ressources organisées en couches~~

REST et les ressources

- Utilisation des verbes HTTP pour faire du CRUD :
  - GET : **R**ead
  - POST : **C**reate (**U**pdate)
  - PUT : **U**pdate
  - DELETE : **D**elete
- Identification des ressources via l'URL :
  - GET api.com/book/:id
  - GET api.com/book
  - POST api.com/book
  - DELETE api.com/book/:id

Créer un service Web Java utilisant les verbes HTTP

- Utilisation de l'implémentation JERSEY :
  - https://jersey.java.net/
- Exploitation d'annotations @GET, @POST, etc
- Spécifier un package war dans le fichier maven
- Utilisation possible de plugin maven pour lancer un serveur d'application
- Déploiement sur un serveur d'application Tomcat ou Jetty
- Voir exemple de code sur la page du module
- **Attention aux problèmes de sécurité en cas de déploiement distant !!! Accès CROSS DOMAIN : voir les solutions CORS.**