

SOA – Services web REST

Comprendre le style d'architecture : REST

Mickaël BARON – 2010 (Rev. Février 2019)

<mailto:baron.mickael@gmail.com> ou <mailto:baron@ensma.fr>



@mickaelbaron



mickael-baron.fr

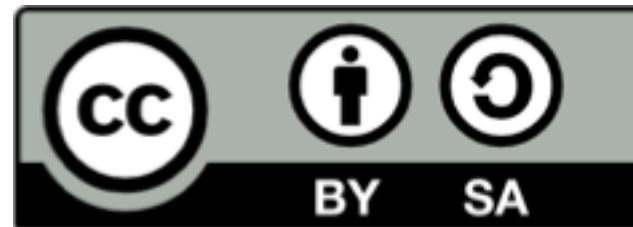
Licence

Creative Commons

Contrat Paternité

Partage des Conditions Initiales à l'Identique

2.0 France



<http://creativecommons.org/licenses/by-sa/2.0/fr>

À propos de l'auteur ...



➤ Mickaël BARON

➤ Ingénieur de Recherche au LIAS

- <https://www.lias-lab.fr> 
- Équipe : Ingénierie des Données et des Modèles
- Responsable des plateformes logicielles, « coach » technique



@mickaelbaron



mickael-baron.fr

➤ Responsable Rubriques Java de Developpez.com

- Communauté Francophone dédiée au développement informatique
- <https://java.developpez.com>
- 4 millions de visiteurs uniques et 12 millions de pages vues par mois
- 750 00 membres, 2 000 forums et jusqu'à 5 000 messages par jour



Plan du cours

- L'utilisation du Web aujourd'hui ...
- Protocole HTTP, un rappel (requête et réponse)
- C'est quoi REST ?
 - Ressources
 - Verbes
 - Représentations
- Exemple
- Services web REST Versus Etendus
- Décrire son API
- Outils

Déroulement du cours

➤ Pédagogie du cours

- Des bulles d'aide tout au long du cours
- Survol des principaux concepts en évitant une présentation exhaustive

➤ Logiciels utilisés

- Navigateur Web, cURL, Postman



➤ Pré-requis

- Ingénierie des données
- Schema XML



Ressources : liens sur le Web

➤ Billets issus de Blog

- wintermuteblog.blogspot.com/2010/01/wadl-toolbox.html
- bitworking.org/news/193/Do-we-need-WADL
- www.pompage.net/pompe/comment-j-ai-explique-rest-a-ma-femme
- www.biologeek.com/rest,traduction,web-semantique/pour-ne-plus-etre-en-rest-comprendre-cette-architecture

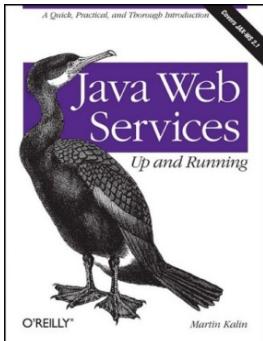
➤ Articles

- www.ibm.com/developerworks/webservices/library/ws-restvsoap
- fr.wikipedia.org/wiki/Representational_State_Transfer
- www.infoq.com/articles/rest-introduction
- restcookbook.com/HTTP%20Methods/idempotency/

➤ Présentations

- www.slideshare.net/gouthamrv/restful-services-2477903
- www.parleys.com/#id=306&st=5&sl=14





Ressources : bibliothèque

➤ RESTful Web Services

- Auteur : Leonard Richardson & Sam Ruby
- Éditeur : Oreilly
- Edition : Dec. 2008 - 448 pages - ISBN : 0596529260

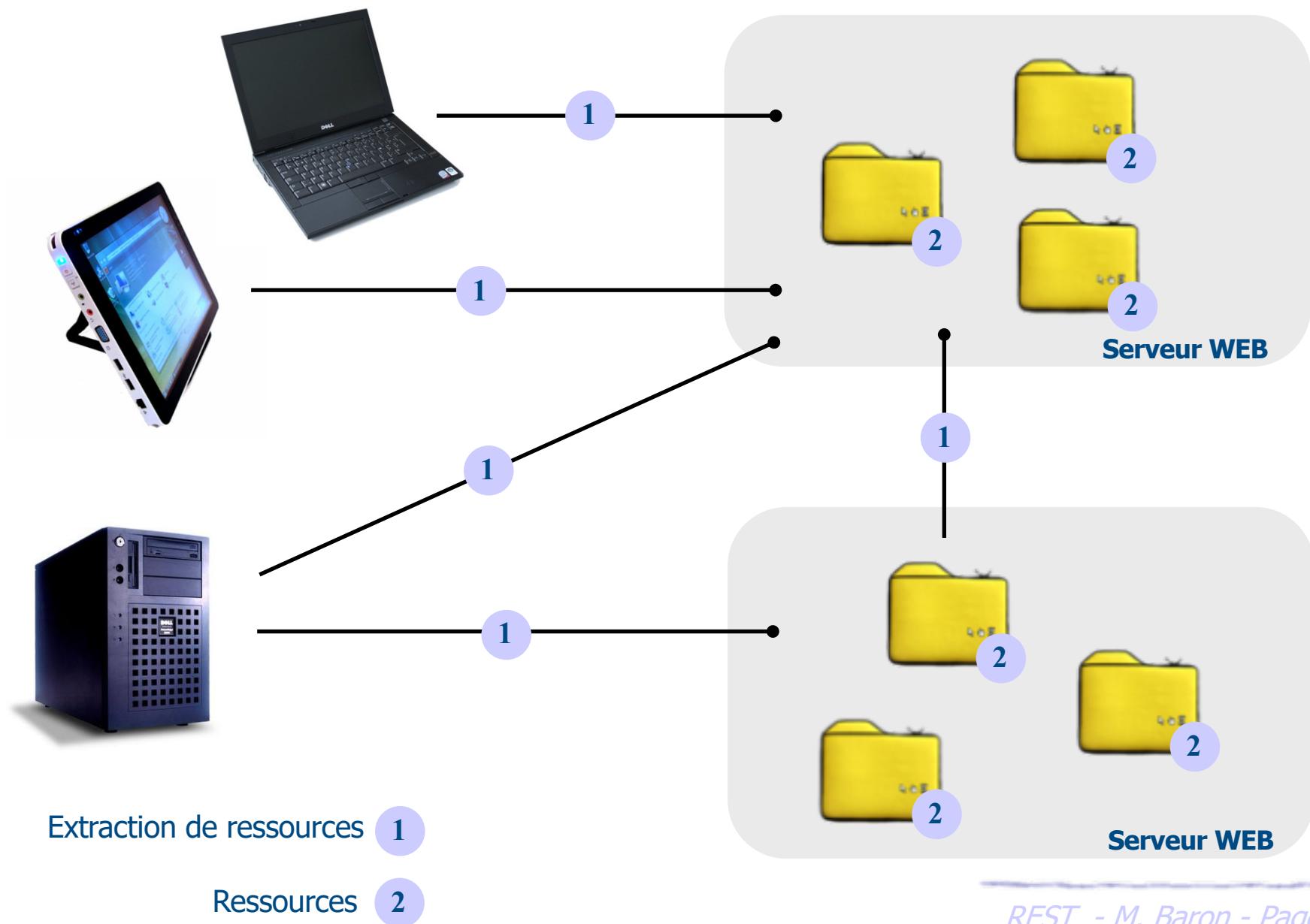
➤ Java Web Services : Up and Running

- Auteur : Martin Kalin
- Éditeur : Oreilly
- Edition : Février 2009 - 316 pages - ISBN : 059652112X

➤ RESTful .NET

- Auteur : Jon Flanders
- Éditeur : Oreilly
- Edition : Nov. 2008 - 320 pages - ISBN : 0596519206

L'utilisation du Web aujourd'hui ...



L'utilisation du Web aujourd'hui ...

► Les ressources sont récupérées au travers les URLs

The screenshot shows a Mozilla Firefox window displaying a product page for an HP ProLiant ML150 G6 server on LDLC.com. The URL in the address bar, <http://www.ldlc.com/fiche/PB00096757.html>, is highlighted with a red box. The page content includes the server's specifications (Intel Xeon E5502, 1 Go RAM, 500 Go storage), pricing information (899,95 € TTC), and various service options like Chronopost delivery and payment methods.

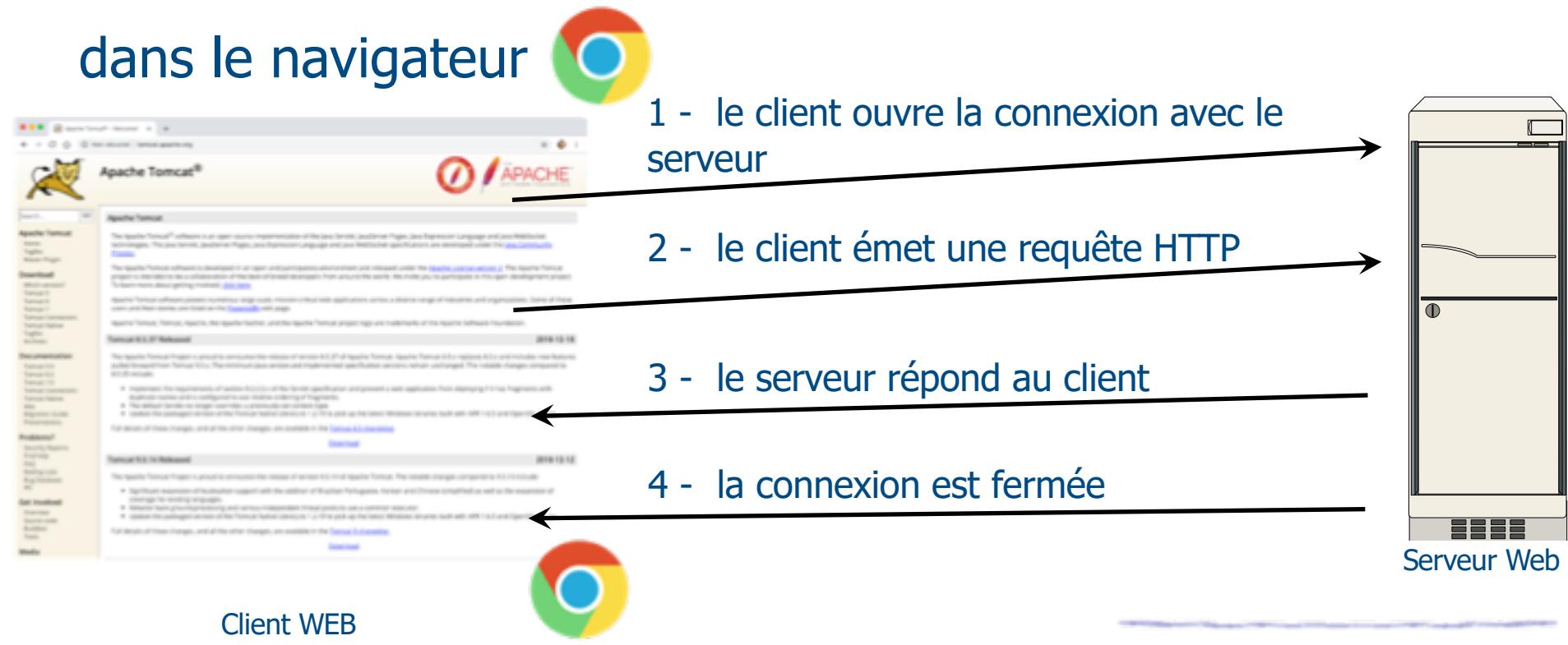
Une ressource (serveur) est identifiée par une URL

Plan du cours

- L'utilisation du Web aujourd'hui ...
- **Protocole HTTP, un rappel**
- C'est quoi REST ?
 - Ressources
 - Verbes
 - Représentations
- Exemple
- Services web REST Versus Etendus
- Décrire son API
- Outils

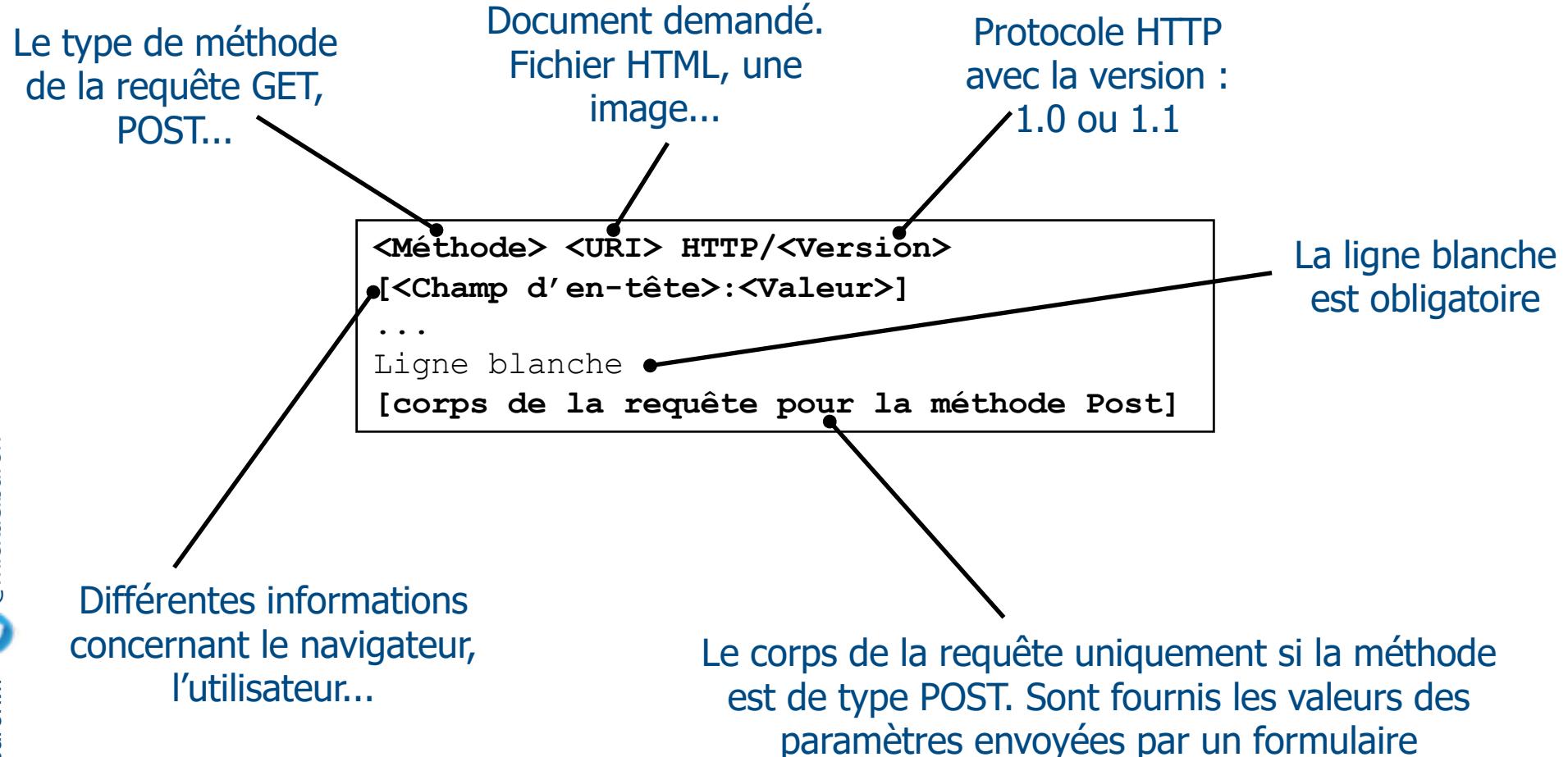
Protocole HTTP : généralités

- Hyper Text Transfer Protocol v1.1
- Protocole Client/Serveur sans état
 - Impossibilité de conserver des informations issu du client
- La conversation HTTP est initialisée lorsque l'URL est saisie dans le navigateur



Protocole HTTP : requête

➤ Requête envoyée par le client (navigateur) au serveur WWW



Protocole HTTP : en-têtes de requête

- Correspond aux formats de documents et aux paramètres pour le serveur
 - *Accept* = types MIME acceptés par le client (text/html, text/plain...)
 - *Content-Type* = type MIME envoyé par le client
 - *Accept-Encoding* = codage acceptées (compress, x-gzip, x-zip...)
 - *Accept-Charset* = jeu de caractères préféré du client
 - *Accept-Language* = liste de langues (fr, en, de...)
 - *Authorization* = type d'autorisation
 - BASIC nom:mot de passe (en base64)
 - Transmis en clair, facile à déchiffrer
 - *Cookie* = cookie retourné
 - *From* = adresse email de l'utilisateur

Protocole HTTP : type de méthodes

- Lorsqu'un client se connecte à un serveur et envoie une requête, cette requête peut-être de plusieurs types appelés **méthodes**
- Requête de type GET
 - Pour extraire des informations (document, graphique...)
 - Intègre les données de formatage à l'URL (chaîne d'interrogation)
 - *www.exemple.com/hello?key1=titi&key2=raoul&...*
- Requête de type POST
 - Pour poster des informations secrètes, des données graphiques...
 - Transmis dans le corps de la requête

```
<Méthode> <URI> HTTP/<Version>
[<Champ d'en-tête>:<Valeur>]
...
Ligne blanche
[corps de la requête pour la méthode Post]
```



Protocole HTTP : réponse

- Réponse envoyée par le serveur WWW au client (navigateur)

Protocole HTTP
avec la version :
1.0 ou 1.1

Statuts des réponses
HTTP. Liées à une erreur
ou à une réussite : 200

Donne des
informations sur
le statut : OK

```
HTTP/<Version><Status><Commentaire Status>
Content-Type:<Type MIME du contenu>
[<Champ d'en-tête>:<Valeur>]
...
Ligne blanche
Document
```

La ligne blanche
est obligatoire

Type de contenu qui sera
retourné : text/html,
text/plain,
application/octet-stream

Différentes informations
concernant le serveur...

Le document peut contenir
du texte non formaté, du
code HTML...

Protocole HTTP : en-têtes de réponse

- Correspond aux informations concernant le serveur WWW
 - *Accept-Ranges* = accepte ou refus d'une requête par intervalle
 - *Content-Type* = type MIME envoyé par le serveur
 - *Age* = ancienneté du document en secondes
 - *Server* = information concernant le serveur qui retourne la réponse
 - *WWW-Authenticate* = système d'authentification. Utiliser en couple avec l'en-tête réponse *Authorization*
 - *ETag* = ...
 - *Location* = ...

Protocole HTTP : statuts des réponses

- Réponse du serveur au client <*Status*><*Commentaire*>
 - *100-199 : Informationnel*
 - 100 : Continue (le client peut envoyer la suite de la requête)
 - *200-299 : Succès de la requête client*
 - 200 : OK, 204 : No Content (pas de nouveau corps de réponse)
 - *300-399 : Re-direction de la requête client*
 - 301 : Redirection, 302 : Moved Temporarily
 - *400-499 : Erreur client*
 - 401 : Unauthorized, 404 : Not Found (ressource non trouvée)
 - *500-599 : Erreur serveur*
 - 503 : Service Unavailable (serveur est indisponible)

Plan du cours

- L'utilisation du Web aujourd'hui ...
- Protocole HTTP, un rappel
- **C'est quoi REST ?**
 - Ressources
 - Verbes
 - Représentations
- Exemple
- Services web REST Versus Etendus
- Décrire son API
- Outils

C'est quoi REST ?

- REST est l'acronyme de REpresentational State Transfert
- Principes définis dans la thèse de Roy FIELDING en 2000
 - Principaux auteurs de la spécification HTTP
 - Membre fondateur de la fondation Apache
 - Développeur du serveur Web Apache
- REST est un style d'architecture inspiré de l'architecture du Web
- **REST** est
 - un style d'architecture
 - une approche pour construire une application
- **REST** n'est pas
 - un format
 - un protocole
 - un standard

C'est quoi REST ?

- Les services web REST sont utilisés pour développer des architectures orientées ressources
- Différentes nominations disponibles dans la littérature
 - Architectures Orientées Données (**DOA**)
 - Architectures Orientées Ressources (**ROA**)
- Les applications qui respectent les architectures orientées ressources sont respectivement nommées **RESTful**
- Dans la suite du cours nous utiliserons indifféremment la nomination REST et RESTful



C'est quoi REST ? Les fournisseurs

- Principaux acteurs qui fournissent des services web REST



C'est quoi REST ? Les fournisseurs



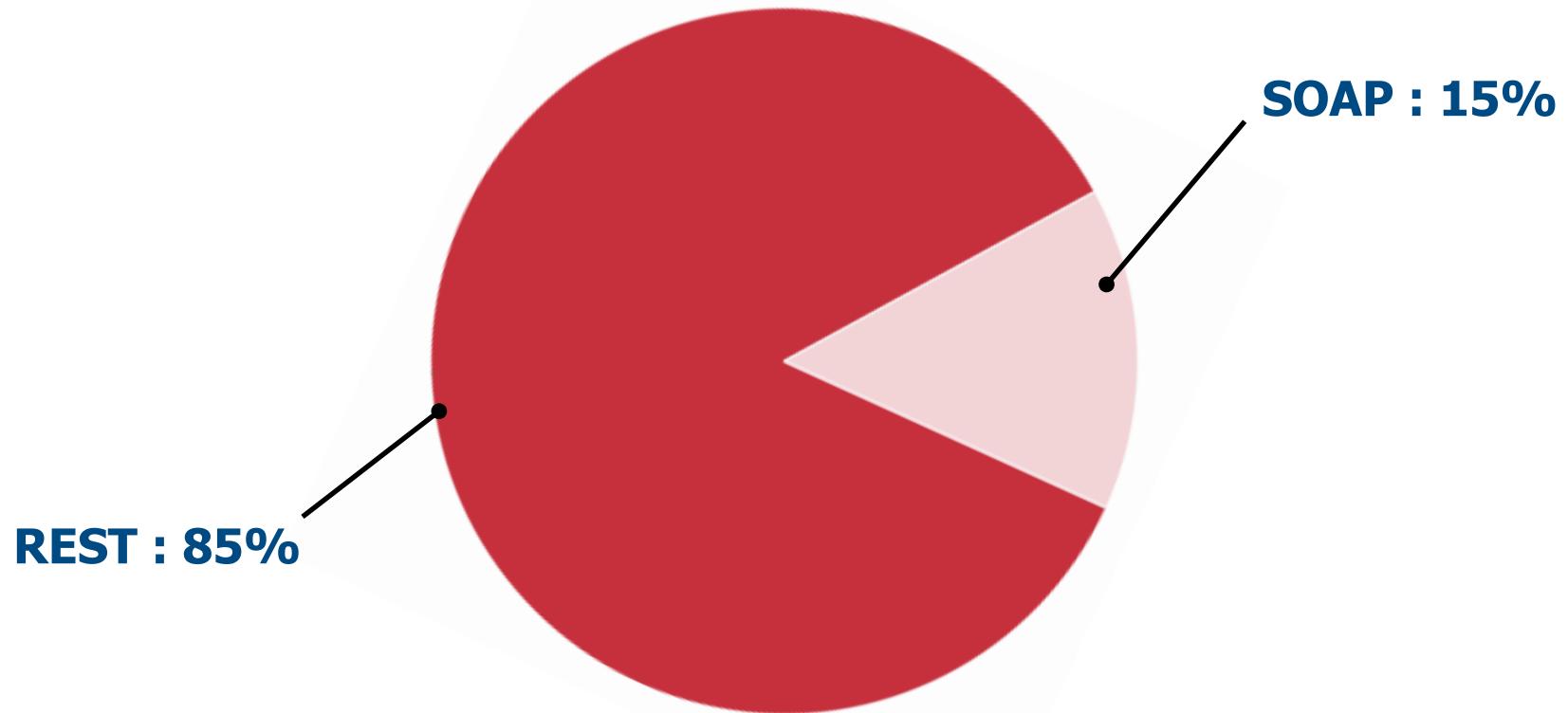
➤ Où trouver des services web REST

- www.programmableweb.com : annuaire des API du moment
- github.com/toddmotto/public-apis : un dépôt d'API publiques
- ovrstat.com : API pour accéder aux statistiques Overwatch
- openweathermap.org/api : API pour la météo
- open-notify.org : API de la Nasa (exemple : position de l'ISS)
- pokeapi.co : tout connaître du monde Pokémons
- icanhazdadjoke.com/api : API pour des blagues (EN)
- api.chucknorris.io : API sur les blagues de Chuck Norris (EN)
- developers.giphy.com : API pour les gifs animés
- deckofcardsapi.com : API pour gérer un jeu de cartes
- swapi.co : tout connaître du monde Star Wars
- httpbin.org : un service de test



C'est quoi REST ? Les fournisseurs

- Statistiques de l'utilisation de services web REST et SOAP chez AMAZON
- www.oreillynet.com/pub/wlg/3005



C'est quoi REST ? : caractéristiques

- Les services web REST sont sans états (Stateless)
 - Chaque requête envoyée vers le serveur doit contenir toutes les informations à leur traitement
 - Minimisation des ressources systèmes, pas de session ni d'état
- Les services web REST fournissent une interface uniforme basée sur les méthodes HTTP
 - GET, POST, PUT et DELETE
- Les architectures orientées REST sont construites à partir de ressources qui sont uniquement identifiées par des URI

C'est quoi REST ? : caractéristiques

- Dans une architecture orientée REST, les ressources sont manipulées à travers des formats de représentations
 - Une ressource liée à un *Bon de Commande* est représentée par un document XML
 - La création d'un *Bon de Commande* est réalisée par la combinaison d'une méthode HTTP Post et d'un document XML
- Dans une architecture orientée REST, la communication est obtenue par le transfert de la représentation des ressources
- L'état est maintenue par la représentation d'une ressource
- Par conséquent, le client est responsable de l'état de la ressource

C'est quoi REST ? : requête REST

➤ Ressources (Identifiant)

- Identifié par une URI
- Exemple : *http://localhost:8080/libraryrestwebservice/api/books*

➤ Méthodes (Verbes) pour manipuler l'identifiant

- Méthodes HTTP : GET, POST, PUT and DELETE

➤ Représentation donne une vue sur l'état

- Informations transférées entre le client et le serveur
- Exemples : XML, JSON...

Ressource et URI

- Une ressource est quelque chose qui est identifiable dans un système
 - Personne, Agenda, Collection, Document, Image, Carte...
- Une URI (Uniform Resource Identifier) identifie une ressource de manière unique sur le système
- Une ressource peut avoir plusieurs URI et la représentation de la ressource peut évoluer avec le temps
- Exemple

`http://localhost:8080/books/adventures/harrypotter/2`

Ressource de type
collection

Identifier primaire
de la ressource

Ressource et URI

➤ Exemples d'URI

Deux URI différentes pour
une même ressource

/books/adventures/harrypotter/2

/books/adventures/harrypotter/the_prisoner_of_azkaban

Ressource = 2^{ème} livre de Harry Potter

Ressource = « The Prisoner of Azkaban »

/books/adventures/harrypotter

Ressource = tous les livres d'Harry Potter

/books/adventures

Ressource = tous les livres d'aventure

Ressource et URI : les bonnes pratiques

- Privilégier l'usage des **noms** et non des verbes (différence avec l'approche opération de SOAP)
/books/adventures
- Utiliser le pluriel pour exprimer une collection et le singulier pour la récupération d'une seule instance
/users
/users/12ab
- Eviter l'usage de majuscule car certains serveurs ignorent les différentes types de cases
/books/adventures/harrypotter
- Si plusieurs versions de l'API commencer par le numéro de version
/v3/books/adventures/harrypotter

Méthode

- Une ressource quelconque peut subir quatre opérations de base désignées par **CRUD**
 - Create (Créer)
 - Retrieve (Lire)
 - Update (mettre à jour)
 - Delete (Supprimer)
- REST s'appuie sur le protocole HTTP pour exprimer les opérations via les méthodes HTTP
 - Create par la méthode **POST**
 - Retrieve par la méthode **GET**
 - Update par la méthode **PUT**
 - Delete par la méthode **DELETE**
- Possibilité d'exprimer des opérations supplémentaires via d'autres méthodes HTTP (**HEAD**, **OPTIONS**)

Méthode : GET

- Méthode **GET** fournit la représentation de la ressource
 - Idempotente => le résultat d'une requête (à succès) est indépendant du nombre de fois où elle a été exécutée
 - « Safe » => ne modifie pas la ressource

Action : récupérer

GET /books/adventures/harrypotter/2



Client

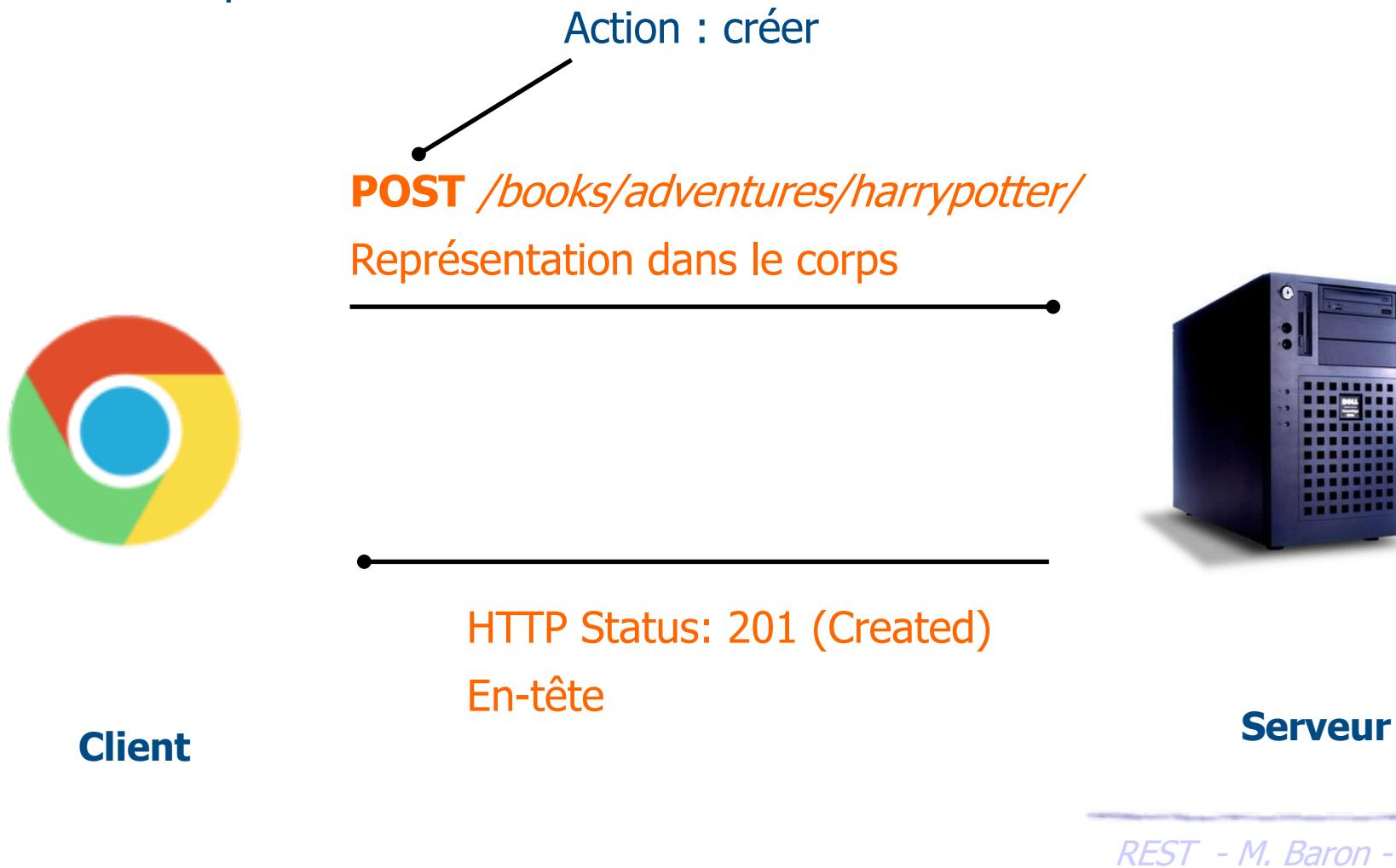


Serveur

HTTP Status: 200 (OK)
En-tête + Représentation

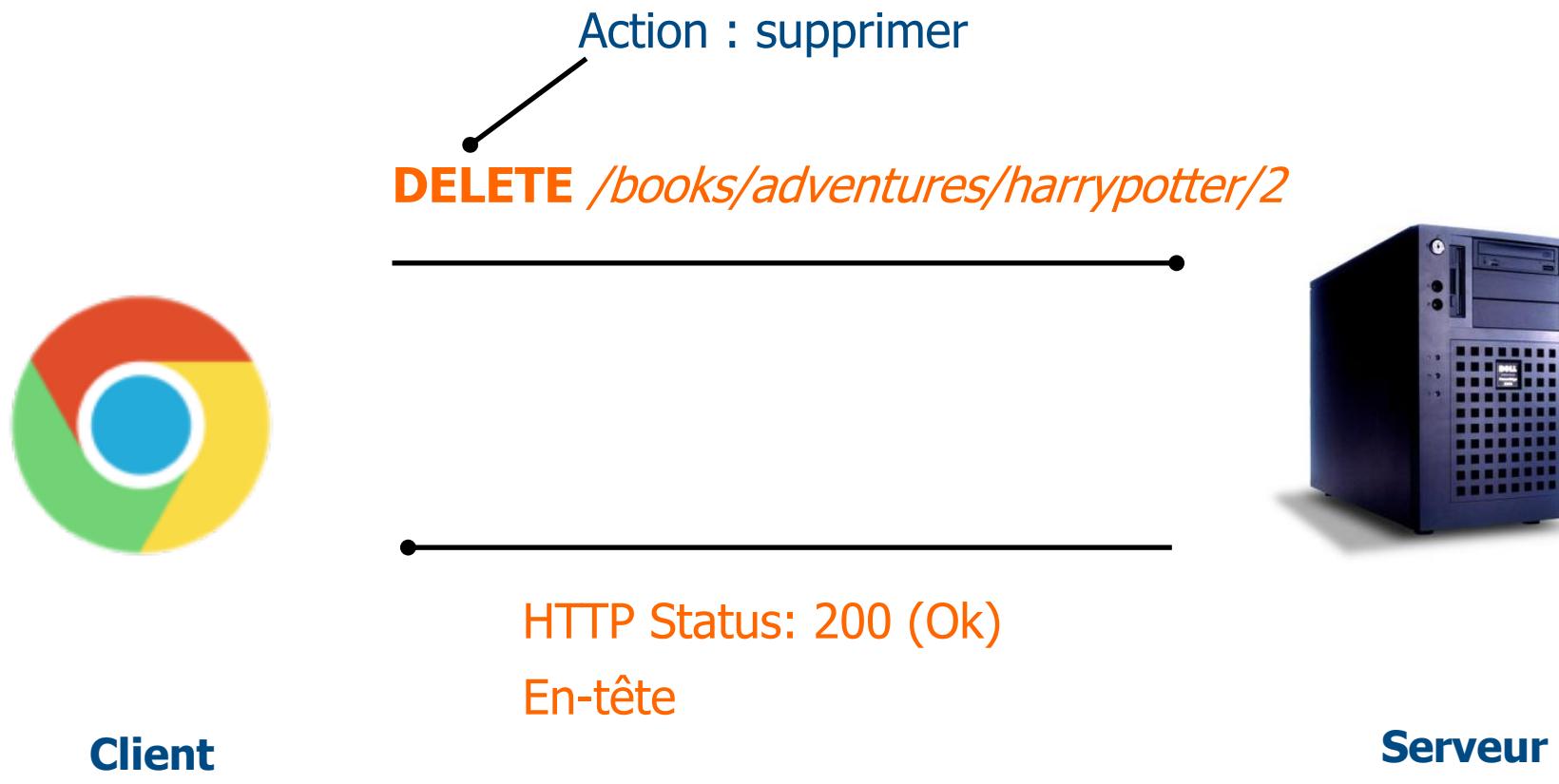
Méthode : POST

- Méthode **POST** crée une ressource
 - Non idempotente (plusieurs créations de la même ressource)
 - N'est pas « Safe »



Méthode : DELETE

- Méthode **DELETE** supprime une ressource
 - Idempotente => si la ressource à supprimer n'existe pas retourner code **200** et non 204 ou 404
 - N'est pas « Safe »



Méthode : PUT

- Méthode **PUT** met à jour une ressource
 - Idempotente => pour s'assurer que la mise à jour s'est effectuée même en cas de panne
 - N'est pas « Safe »



@mickaelbaron



mickael-baron.fr

Client

Action : mise à jour
PUT /books/adventures/harrypotter/2
Représentation dans le corps



HTTP Status: 200 (Ok)

En-tête

Serveur



Méthode : les bonnes pratiques

- Utiliser les verbes appropriés pour les actions souhaitées

GET = Récupérer

UPDATE = Mettre à jour

POST = Créer

DELETE = Supprimer

- S'assurer qu'une méthode « Safe » ou sécurisée ne modifie pas une ressource

GET est « Safe »

POST, PUT et DELETE ne sont pas « Safe »

- S'assurer qu'une méthode « Idempotente » puisse être appelée autant de fois avec toujours le même résultat

GET, PUT et DELETE sont idempotentes

POST n'est pas idempotente

Représentation

- Fournir les données suivant une représentation pour
 - le client (GET)
 - pour le serveur (PUT et POST)
- Données retournées sous différents formats
 - XML
 - JSON
 - (X)HTML
 - CSV
 - ...
- Le format d'entrée (POST) et le format de sortie (GET) d'un service web d'une ressource peuvent être différents

Représentation

➤ Exemples : format JSON et XML

GET <https://www.googleapis.com/urlshortener/v1/url?shortUrl=http://goo.gl/fbsS>

```
{  
  "kind": "urlshortener#url",  
  "id": "http://goo.gl/fbsS",  
  "longUrl": "http://www.google.com/",  
  "status": "OK"  
}
```

Représentation des données en JSON

GET <http://localhost:8080/librarycontentrestwebservices/contentbooks/string>

```
<?xml version="1.0"?>  
<details>  
  Ce livre est une introduction sur la vie  
</details>
```

Représentation des données en XML

Plan du cours

- L'utilisation du Web aujourd'hui ...
- Protocole HTTP, un rappel
- C'est quoi REST ?
 - Ressources
 - Verbes
 - Représentations
- **Exemple**
- Services web REST Versus Etendus
- Décrire son API
- Outils

Exemple : Message HelloWorld

- Service qui sera mis en œuvre dans le tutoriel sur le développement de microservices (dernière partie)
- Principales fonctionnalités de l'API
 - Créer des messages « HelloWorld »
 - Récupérer la liste des messages « HelloWorld »
 - Format JSON pour la création et la récupération des messages
 - Utilisation de l'outil **Postman** pour la manipulation de l'API

Exemple : Message HelloWorld

➤ Création de message « HelloWorld » avec Postman

POST *https://localhost:8080/helloworld*

Content-Type: application/json

{"message": "Mon HelloWorld"}

The screenshot shows the Postman application interface. At the top, there's a toolbar with 'New', 'Import', 'Runner', 'Builder' (which is selected), 'Team Library', and other options. Below the toolbar, the URL 'http://localhost:8080/' is entered, and the method 'POST' is selected. The 'Body' tab is active, showing the JSON payload: {"message": "Mon HelloWorld"}. The 'Send' button is highlighted with a blue border. Below the request area, the 'Headers' tab is selected, showing the response headers: Status: 201 Created, Time: 879 ms, Size: 393 B. The headers listed include Access-Control-Allow-Credentials, Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin, Access-Control-Max-Age, Content-Length, Date, Server, and X-Powered-By.

1 - Requête HTTP
de type POST
avec contenu de type JSON

2 – Ressource créée 201 et
informations placées dans
l'en-tête de la réponse



Exemple : Message HelloWorld

- Récupération des message « HelloWorld » avec Postman

GET *http://localhost:8080/helloworld*

The screenshot shows the Postman interface with a successful GET request to `http://localhost:8080/helloworld`. The response body is displayed in JSON format:

```
[{"id": 3, "message": "Mon HelloWorld", "startDate": "Wed Jan 02 19:23:11 UTC 2019"}, {"id": 2, "message": "Mon HelloWorld", "startDate": "Wed Jan 02 19:23:18 UTC 2019"}, {"id": 1, "message": "Mon HelloWorld", "startDate": "Wed Jan 02 19:13:00 UTC 2019"}]
```

Contenu du corps de la réponse qui correspond à la ressource nouvellement créée au format JSON

Plan du cours

- L'utilisation du Web aujourd'hui ...
- Protocole HTTP, un rappel
- C'est quoi REST ?
 - Ressources
 - Verbes
 - Représentations
- Exemple
- **Services web REST Versus Etendus**
- Décrire son API
- Outils

Service web Etendus VERSUS REST

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:hel="http://helloworldwebservice.lisi.ensma.fr/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <hel:GetOrderDetail>  
            <value>14546-xx-45</value>  
        </hel:GetOrderDetail>  
    </soapenv:Body>  
</soapenv:Envelope>
```



Client



Serveur

SOAP

REST

```
GET http://localhost:8080/order?ordernum=14546-xx-45
```



Client



Serveur

Service web Etendus VERSUS REST

- Les services web étendus (SOAP) et les services web REST diffèrent par le fait que
 - Services web étendus reposent sur des standards
 - REST est un style d'architecture
- Services web étendus (SOAP)
 - **Avantages**
 - Standardisé
 - Interopérabilité
 - Sécurité (WS-Security)
 - Outilé
 - **Inconvénients**
 - Performances (enveloppe SOAP supplémentaire)
 - Complexité, lourdeur
 - Cible l'appel de service

Service web Etendus VERSUS REST

➤ Services web **REST**

➤ Avantages

- Simplicité de mise en œuvre
- Lisibilité par l'humain
- Evolutivité
- Repose sur les principes du Web
- Représentations multiples

➤ Inconvénients

- Sécurité restreinte par l'emploi des méthodes HTTP
- Cible uniquement l'appel de ressource

Plan du cours

- L'utilisation du Web aujourd'hui ...
- Protocole HTTP, un rappel
- C'est quoi REST ?
 - Ressources
 - Verbes
 - Représentations
- Exemple
- Services web REST Versus Etendus
- **Décrire son API**
- Outils

Décrire son API

- Pas vraiment de solution officielle pour décrire son API
- À l'opposé pour les services web étendus une seule solution se dégagée => WSDL
- Dans le monde des services web REST plusieurs solutions
 - OpenAPI (anciennement Swagger) => www.openapis.org
 - RAML => raml.org
 - API BluePrint => apiblueprint.org
 - WADL => www.w3.org/Submission/wadl/
- Fonctionnalités proposées
 - Décrire un service web REST quelque soit son implémentation
 - Générer un squelette pour faciliter l'implémentation
 - Fournir des outils pour faciliter les tests
- **Sans vouloir choisir**, OpenAPI s'est rapidement imposée par son aspect communautaire

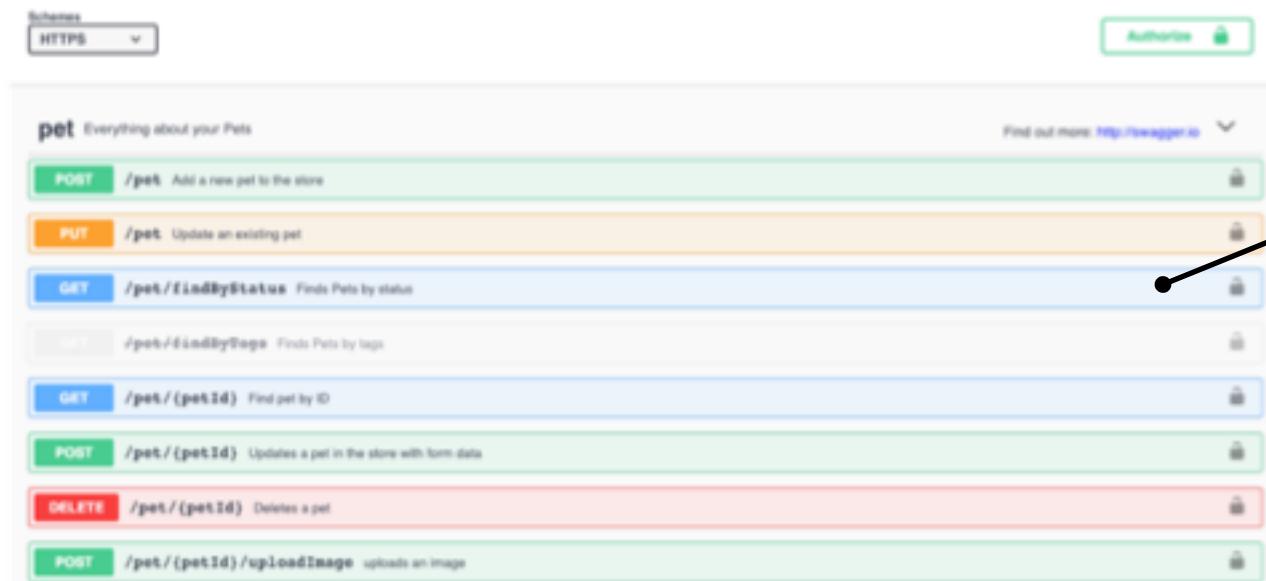
Décrire son API

➤ Swagger UI : outil pour manipuler un contrat OpenAPI



The screenshot shows a web browser window titled "Swagger UI". The address bar contains the URL <https://petstore.swagger.io/v2/swagger.json>. Below the address bar, there's a navigation bar with a "Swagger" logo and an "Explore" button. The main content area is titled "Swagger Petstore" and displays a sample server information page. It includes links for "Terms of service", "Contact the developer", "Apache 2.0", and "Find out more about Swagger". A note at the top states: "This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on try.samsara.io. For this sample, you can use the api key `special-key` to test the authorization filters."

Le contrat est décrit dans un format JSON



The screenshot shows the "pet" resource page in the Swagger UI. At the top left, there's a dropdown menu set to "HTTPS". On the right side, there's an "Authorize" button. The main content lists various operations for managing pets:

- pet** Everything about your Pets
- POST /pet**: Add a new pet to the store
- PUT /pet**: Update an existing pet
- GET /pet/findByStatus**: Finds Pets by status
Sub-operations:
 - /pet/findByStatus
 - /pet/findByTags
- GET /pet/{petId}**: Find pet by ID
- POST /pet/{petId}**: Updates a pet in the store with form data
- DELETE /pet/{petId}**: Deletes a pet
- POST /pet/{petId}/uploadImage**: uploads an image

Visualisation de l'ensemble des ressources et des moyens d'y accéder (Verbes)

Décrire son API

➤ Swagger UI : outil pour manipuler un contrat OpenAPI

The screenshot shows the Swagger UI interface for a Petstore API. At the top, there's a navigation bar with tabs for 'pet' and 'Find out more: <http://swagger.io>'. Below this, there are three main API operations listed:

- POST /pet**: Add a new pet to the store.
- PUT /pet**: Update an existing pet.
- GET /pet/findByStatus**: Finds Pets by status.

For the GET operation, there's a note: "Multiple status values can be provided with comma separated strings". Below the operations, there's a section for "Parameters" with a "Try it out" button. The "Responses" section shows two entries: a successful 200 response labeled "successful operation" and an error 400 response labeled "invalid status value". At the bottom, there are links to other API endpoints: "/pet/findByTags" and "/pet/{petId} Find pet by ID".

Possibilité d'invoquer le service pour tester

Description des paramètres et des retours possibles



Décrire son API : l'occasion manquée de WADL

- WADL (Web Application Description Language) est un langage de description XML de services de type REST
- WADL était une spécification W3C initiée par SUN
 - www.w3.org/Submission/wadl/
- Description des services par éléments de type
 - *Ressource, Méthode, Paramètre, Requête, Réponse*
- L'objectif était de pouvoir générer automatiquement les APIs clientes d'accès aux services REST
- Remarques
 - Peu d'outils exploitent la description WADL <https://javaee.github.io/wadl/>
 - Apparu bien plus tard

Décrire son API : l'occasion manquée de WADL

➤ Exemple : afficher le WADL de services REST

```
<application>
<doc jersey:generatedBy="Jersey: 1.4 09/11/2010 10:30 PM"/>
<resources base="http://localhost:8088/librarycontentrestwebservice/">
    <resource path="/contentbooks">
        <resource path="uribuilder2">
            <method name="POST" id="createURIBooks">
                <request>
                    <representation mediaType="application/xml"/>
                </request>
                <response>
                    <representation mediaType="*/*"/>
                </response>
            </method>
        </resource>
        <resource path="uribuilder1">
            <method name="POST" id="createBooksFromURI">
                <request>
                    <representation mediaType="application/xml"/>
                </request>
                <response>
                    <representation mediaType="*/*"/>
                </response>
            </method>
        </resource>
        ...
    </resource>
</resources>
</application>
```

Outils et bibliothèques

- Des outils pour appeler des services REST
 - En ligne de commande : cURL et HTTPie
 - Plugins Navigateur : RESTClient, Poster, RestEasy
 - SOAPUI : www.soapui.org
 - Postman : www.getpostman.com
 - Wuzz (~ Postman) : github.com/asciimoo/wuzz
 - SwaggerUI : swagger.io/tools/swagger-ui
- Des plateformes pour développer (serveur) et appeler (client) des services REST
 - JAX-RS (Jersey)  pour la plateforme Java
 - .NET 
 - PHP 
 - Python... 

Outils et bibliothèques

- Exemple : affichage sympathique avec curl -4 wttr.in/City

```
1. bash
LIAS-Baron:~ baronm$ curl -4 wttr.in/Poitiers
Weather for City: Poitiers, France

Sunny
0 - 4 °C
↑ 17 km/h
6 km
0.0 mm

Morning Noon Fri 19. Feb Evening Night
Cloudy
-2 - 2 °C
↑ 14 - 26 km/h
10 km
0.0 mm | 0%
Sunny
2 - 6 °C
10 km
0.0 mm | 0%
Clear
0 - 4 °C
10 km
0.1 mm | 4%
Clear
-1 - 3 °C
10 km
0.0 mm | 0%

Morning Noon Sat 20. Feb Evening Night
Light drizzle
4 - 7 °C
↑ 26 - 39 km/h
2 km
0.3 mm | 48%
Light drizzle
7 - 10 °C
↑ 24 - 33 km/h
2 km
0.2 mm | 86%
Cloudy
8 - 10 °C
10 km
0.1 mm | 56%
Overcast
7 - 10 °C
10 km
0.1 mm | 86%

Morning Noon Sun 21. Feb Evening Night
Light drizzle
7 - 10 °C
↑ 24 - 32 km/h
2 km
0.2 mm | 56%
Overcast
10 - 13 °C
↑ 26 - 30 km/h
10 km
0.1 mm | 9%
Partly Cloudy
9 - 11 °C
10 km
0.0 mm | 36%
Clear
7 - 9 °C
10 km
0.0 mm | 6%
```

Follow [@igor_chubin](#) for wttr.in updates
LIAS-Baron:~ baronm\$



Outils et bibliothèques : cURL (Cheat Sheet)

- *-H* ou *--header* : pour passer un paramètre en en-tête
 - -H "Content-Type: application/json"
 - -H "Accept: application/json"
- *-d* ou *--data* : pour envoyer du contenu dans le payload
 - --data '{"id":"TR123"}'
 - --data "@myfile.json"
- *-i* : ajoute dans la réponse l'en-tête
- *-I* : montre uniquement l'en-tête de la réponse
- *-v* ou *--verbose* : cURL en mode « blabla »
- *-X* : pour préciser le verbe POST, GET, PUT ou DELETE
- Copie depuis Chrome ou Firefox
 - Developer Tools
 - Onglet Network