COMPRENDRE L'ARCHITECTURE DES WEB SERVICES REST

Amosse EDOUARD, Doctorant

Evaluation

- Présence, Participation➔ 10%
- Mini Projet ➔ 50%
- Examen Théorique ➔ 40%

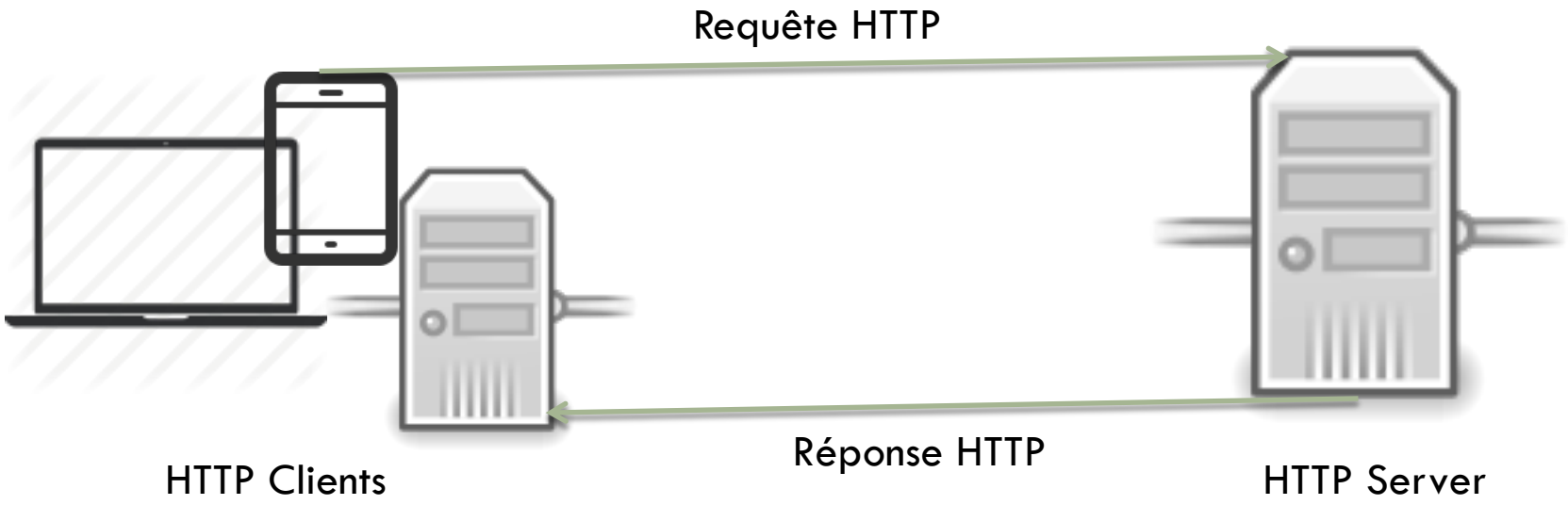
Objectifs du cours

- SOAP
  - Initiation au protocole et les Web services SOAP
  - Comprendre les enveloppes SOAP
  - Créer et tester des services SOAP
- REST
  - Comprendre l'architecture des applications compatibles REST
  - Exposer des services REST
  - Consommer des services REST dans de applications Web

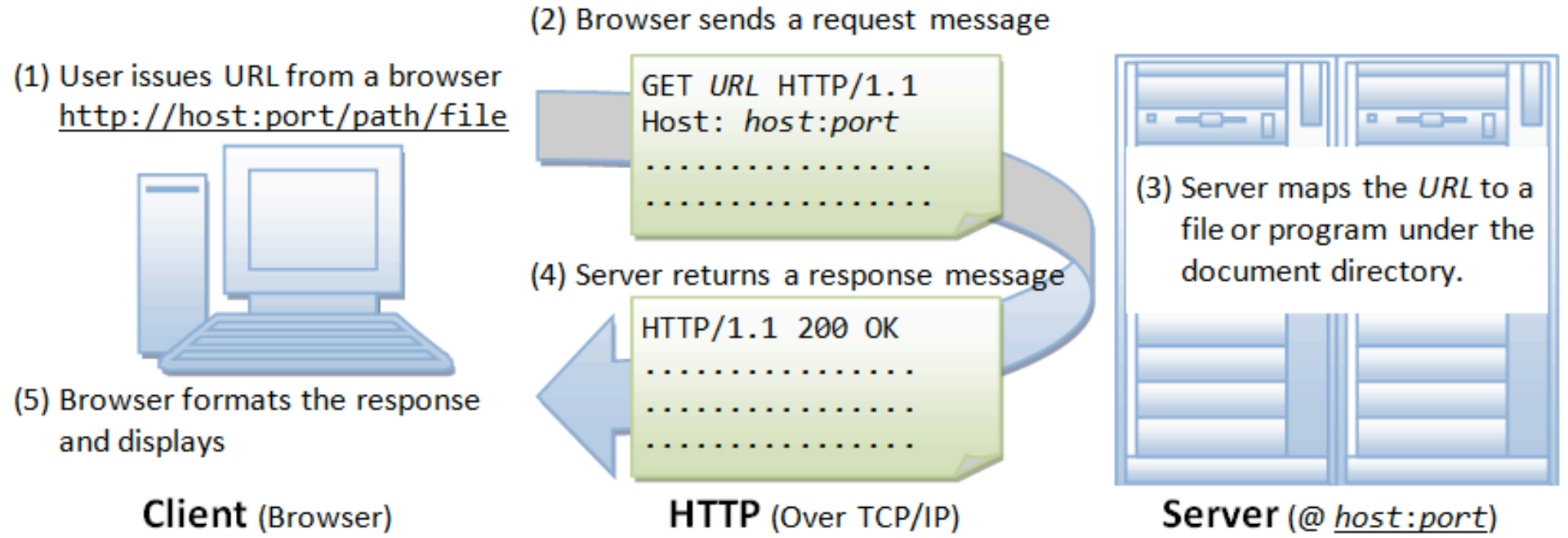
Rappel → Protocol HTTP

Le Protocole HTTP

- HyperText Transfer Protocol
- Protocole d'échanges d'information sur le web
- Basé sur TCP/IP



Enchainement Client Serveur



URL

- Unique Resource Location
- Identifie les ressources de manière unique sur le Web
- 4 parties
  - Protocole (http, ftp, mail, …)
  - Host (google.com)
  - Port (8080, 80)
  - Path (Chemin vers la ressource sur le serveur)

Requêtes HTTP

- Permet à un client d'envoyer des messages à un serveur
- Format d'un message HTTP
  - Request Message Header
    - Request Line
    - Request Headers [Optional]
  - Request Message Body

Request Message Header

- Request Line

```
POST /bibliotheque/faces/views/categorie/Create.xhtml HTTP/1.1
```

- Request Headers

```
Host: localhost:8080
Connection: keep-alive
Content-Length: 176
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_0) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/39.0.2171.65 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: http://localhost:8080/bibliotheque/faces/views/categorie/List.xhtml
Accept-Encoding: gzip, deflate
Accept-Language: fr,fr-FR;q=0.8,en;q=0.6
Cookie: JSESSIONID=d64a9484e61761662575b5d14af1
```

Request Message Body

- Contient les données supplémentaires envoyées au serveur

```
j_idt13:nom:Miage
j_idt13:description:NTDP
```

Réponse HTTP

- Réponse du serveur au client

- Format d'une réponse HTTP
  - Response Message Header
    - Response Line
    - Response Headers
  - Response Message [Optional]

Response Message Header

- Response Line

```
HTTP/1.1 200 OK
```

- Response Headers

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition  4.0  Java/Oracle Corporation/1.8)
Server: GlassFish Server Open Source Edition  4.0
Content-Type: text/html;charset=UTF-8
Date: Sun, 23 Nov 2014 16:05:39 GMT
Content-Length: 2274
```

Response Message Body

- Response Body

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"><html xmlns="http://www.w3.org/1999/xhtml"><head><link
type="text/css" rel="stylesheet" href="/bibliotheque/faces/javax.faces.resource/theme.css?ln=primefaces-aristo"
/><link type="text/css" rel="stylesheet" href="/bibliotheque/faces/javax.faces.resource/css/jsfcrud.css" />
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>Create New Categorie</title></head><body>
        <h1>Create New Categorie
        </h1>
        <p><div id="messagePanel"><table><tr style="color: green"><td>Categorie was successfully created. </td>
</tr></table></div>
</html>
</html>
```

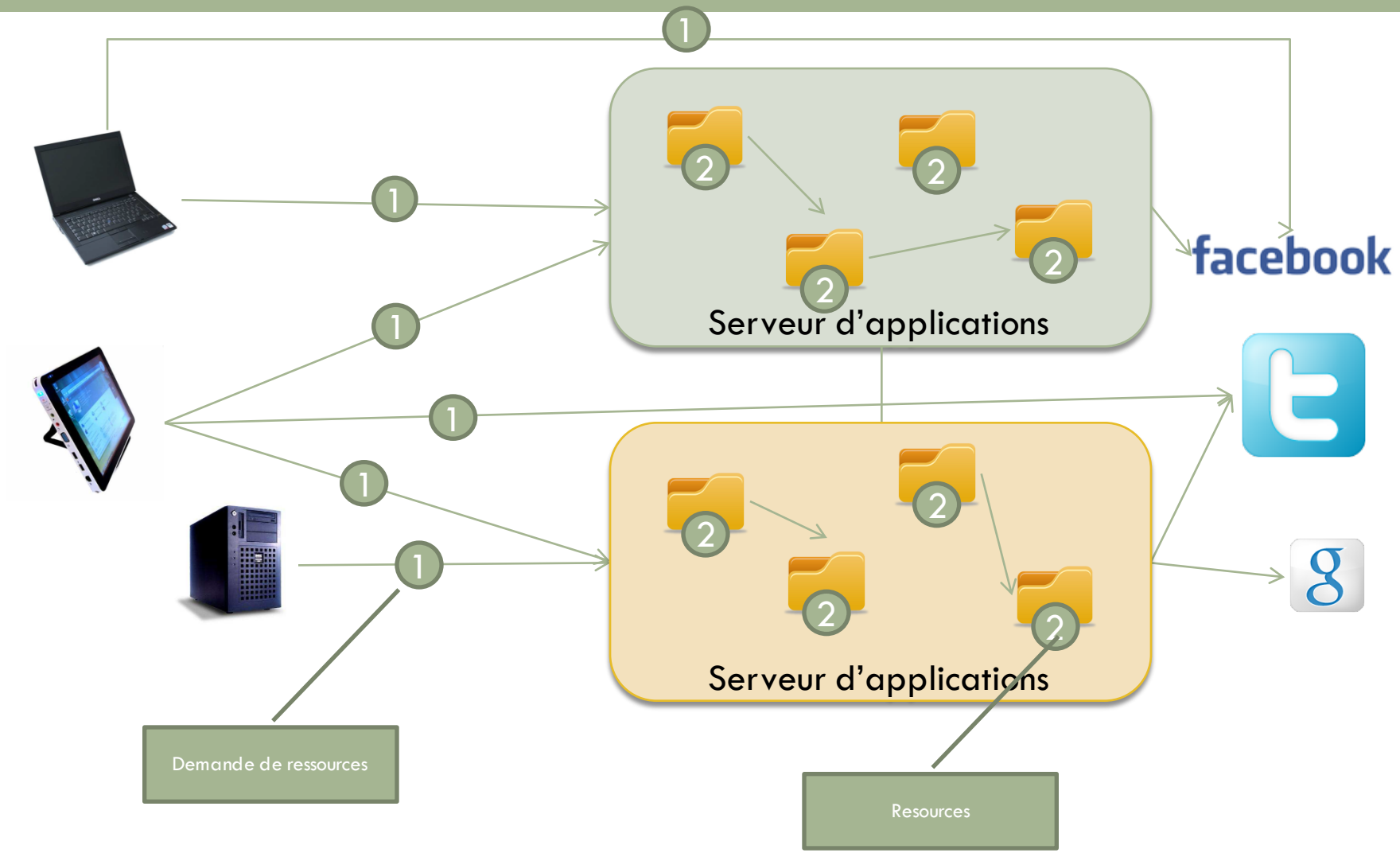
Méthodes HTTP

- HTTP définit un ensemble de méthode permet de caractériser les requêtes
  - GET : Récupérer des ressources à un serveur
  - POST : Envoyer des données à un serveur
  - PUT : Modifier des données
  - DELETE : Suppression de données
  - OPTIONS : Demander la liste des méthodes supportées par un serveur
  - Autres : HEAD, TRACE, CONNECT

Partie 1

Introduction aux Web Services

Utilisation du web aujourd'hui (1)



Web Services (Définition)

- Services informatiques de la famille des technologies web permettant la communication entre des applications **hétérogènes** dans des environnements distribués *(Wikipédia)*.
- Ils ont été proposé à la base comme solution d'intégrations de différents logiciels développés par des entreprises (ERP, SCM, CRM) leur permettant de communiquer entre eux.
- Basés sur XML (déscription et échange) et utilisant en général les protocoles du web comme canal de communication;

Types de Services Webs

- Deux principaux types
  - SOAP
  - REST

Web Services SOAP

- **S**imple **O**bject **A**ccess **P**rotocol

- Protocole d'échanges d'informations dans un environnement distribué basé sur XML
  - Interopérabilité entre applications d'une même entreprise (Intranet)
  - Interopérabilité inter entreprises entre applications et services web

- Similaire au protocole RCP,

Web Services SOAP

- SOAP peut être utilisé de concert avec plusieurs autres protocoles : HTTP, SMTP, POP

→ HTTP est le plus utilisé

Web Services SOAP

	RMI	DCOM	CORBA	SOAP
Défini par	SUN	Microsoft	OMG	W3C
Plate-forme	Multi	Win32	Multi	Multi
Langage de Développement	Java	C++, VB, VJ	Multi	Multi
Langage de définition	Java	ODL	IDL	WSDL
Transport	TCP, HTTP, IIOP	IP/IPX	GIOP, IIOP	HTTP, HTTPR, SMTP
Transaction	Non	Oui	Oui	Oui
Sécurité	SSL, JAAS	?	SSL	SSL

Web Services SOAP

- SOAP est principalement composé de trois parties:
  - Les enveloppes SOAP (ou Message)
  - Les règles d'encodages
  - La représentation RPC

Messages SOAP

- L'Enveloppe SOAP → Obligatoire
  - Une en-tête (Header) → Optionnel
  - Le corps (Body) → Obligatoire

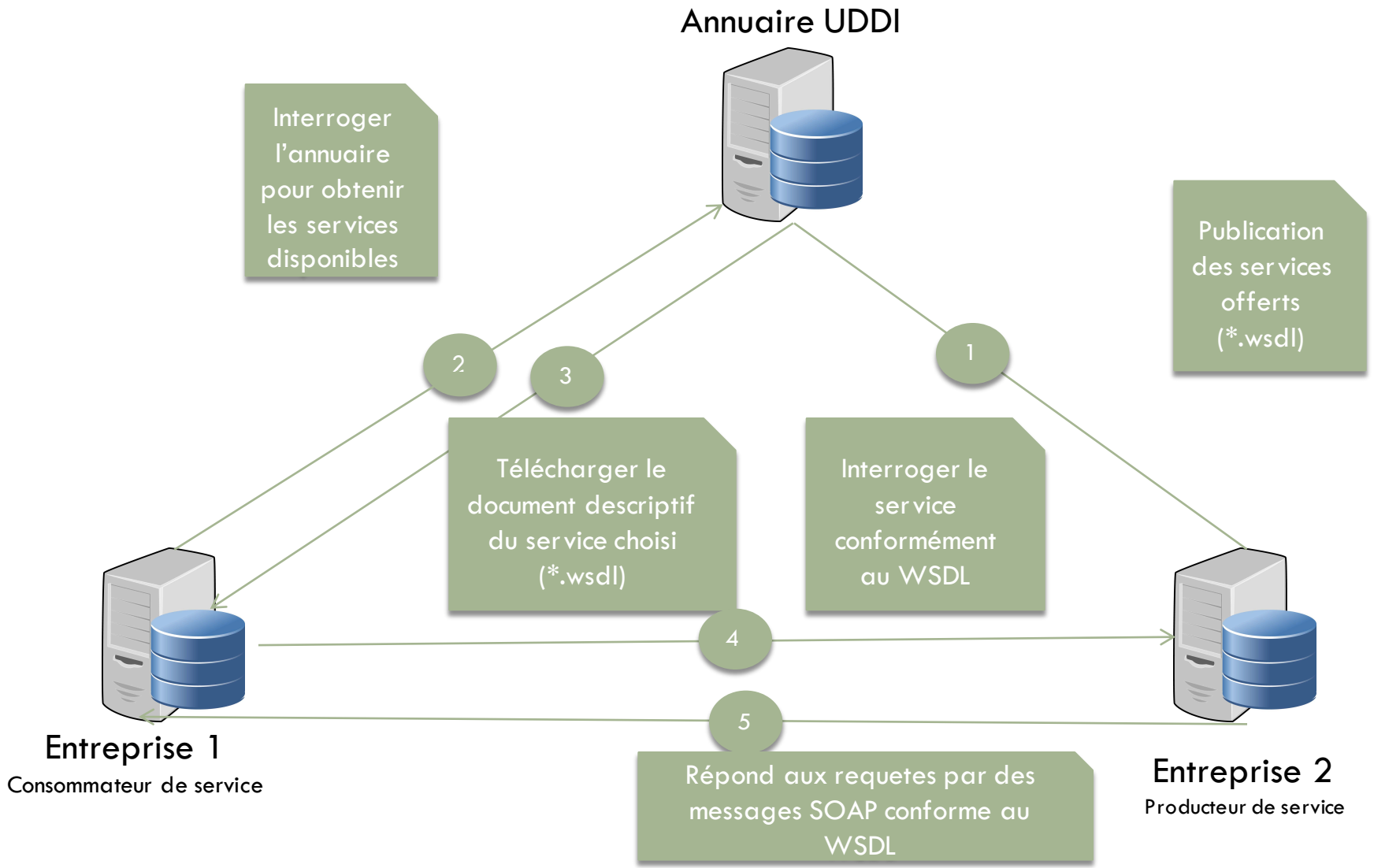
Messages SOAP

- Les messages SOAP sont utilisés pour envoyer (requête) et recevoir (réponse) des informations d'un consommateur vers un producteur
- Un message SOAP peut être transmis à plusieurs récepteurs intermédiaires avant d'être reçu par le récepteur final (➔ chaîne de responsabilité)
- Le format SOAP peut contenir des messages spécifiques correspondant à des erreurs identifiées par le récepteur
- Un message SOAP est véhiculé en utilisant un protocole de transport (HTTP, SMTP, …)

SOAP : WSDL

- Web Service Description Language
- Fichier au format XML
- Décrit les actions exposées par le web service

SOAP – Exemples d'utilisation



SOAP : Enveloppe

- Constitue la racine d'un message SOAP
- Identifié par la balise <namespace:Envelop>
- La balise doit être **obligatoirement** associé à un espace de noms [spec W3C]
- SOAP définit deux espaces de noms
  - Enveloppe SOAP : http://schemas.xmlsoap.org/soap/envelope/
  - Serialization SOAP: http://schemas.xmlsoap.org/soap/encoding/
- Requête et Réponse ont la même structure

SOAP : En-tête

- Balise optionnelle identifié par <namespace:Header>
- Quand il est présent, il doit être avant le Body
- Utilisé pour transmettre des informations supplémentaires entre le consommateur et le fournisseur du service
- Usages possibles
  - Informations d'authentification
  - Contexte d'une transaction
  - Transiter des informations intermédiaires

SOAP : Corps

- Identifié par la balise <namespace:Body>
- Contient la réponse à l'appel d'une action du service
  - Une erreur <namespace:Fault>
  - Réponse de l'action
- L'encodage est des informations est précisé par les bindings du WSDL

SOAP : Requête

- Appeler les opérations d'un web service SOAP

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:sayHelloToTheWorld
xmlns:ns2="http://soap.bibliotheque.android.mbds.fds.edu.ht/"/>
    </S:Body>
</S:Envelope>
```

Appel à la méthode sayHelloToTheWorld sans paramètre

SOAP : Réponse

- Réponse du service à l'appel de la méthode

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:sayHelloToTheWorldResponse
xmlns:ns2="http://soap.bibliotheque.android.mbds.fds.edu.ht/">
            <return>Hello World</return>
        </ns2:sayHelloToTheWorldResponse>
    </S:Body>
</S:Envelope>
```

Réponse du web service à l'appel de la méthode sayHelloToTheWorld

SOAP : Requête

- Appeler les opérations d'un web service SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:sayHelloTo
xmlns:ns2="http://soap.bibliotheque.android.mbds.fds.edu.ht/">
            <name>Miage NTDP</name>
        </ns2:sayHelloTo>
    </S:Body>
</S:Envelope>
```

Appel à la méthode sayHelloTo du service avec une valeur en paramètre

SOAP : Réponse

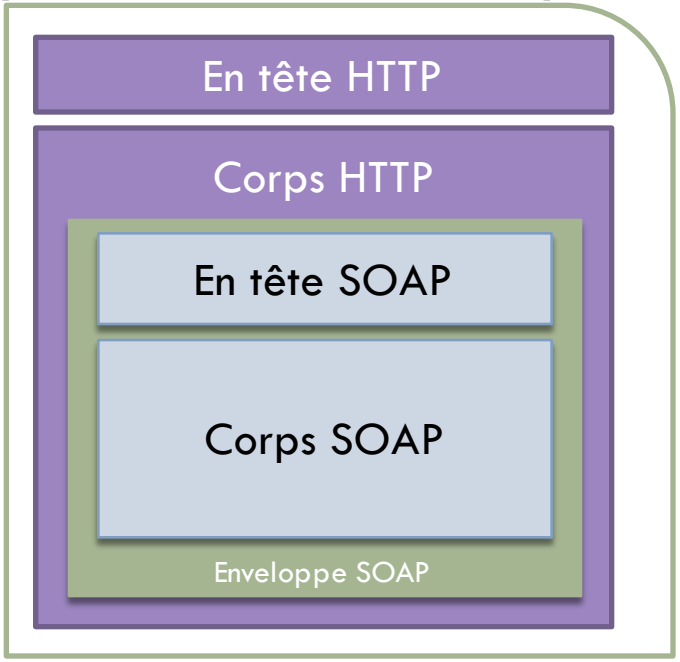
- Réponse du service à l'appel de la méthode

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:sayHelloToResponse
xmlns:ns2="http://soap.bibliotheque.android.mbds.fds.edu.ht/">
            <return>Hello Miage NTDP !</return>
        </ns2:sayHelloToResponse>
    </S:Body>
</S:Envelope>
```

Réponse du web service à l'appel de la méthode sayHelloTo

SOAP : Transport HTTP

- Structure d'une requête HTTP
  - En-tête (http header)
  - Corps (http body)
- Les messages SOAP sont encapsulés dans le corp de la requête HTTP



SOAP : Prise en main

- Créer des services web SOAP en JAVA jax-ws et Netbeans

Services Web RESTFul

Web Service REST

Définition

- Acronyme de **RE**presentational **S**tate **T**ransfert défini dans la thèse de Roy Fielding en 2000.
- REST n'est pas un protocole ou un format, contrairement à SOAP, HTTP ou RCP, mais un style d'architecture inspiré de l'architecture du web fortement basé sur le protocole HTTP
- Il n'est pas dépendant uniquement du web et peut utiliser d'autre protocoles que HTTP

Web Service REST

Ce qu'il est :

- Un système d'architecture
- Une approche pour construire une application

Ce qu'il n'est pas

- Un protocole
- Un format
- Un standard

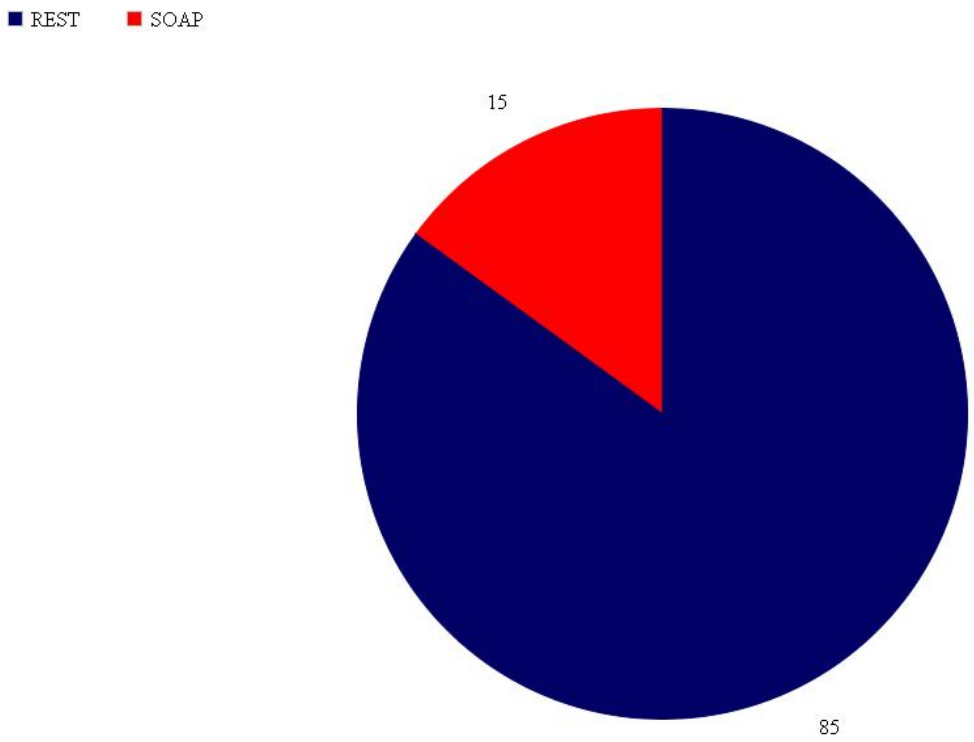
REST → utilisation

- Utiliser dans le développement des applications orientés ressources (ROA) ou orientées données (DOA)
- Les applications respectant l'architecture REST sont dites RESTful

REST → Fournisseurs

REST → Statistics

Statistique d'utilisation des services web REST et SOAP chez AMAZON



REST → Caractéristiques

- Les services REST sont sans états (Stateless)
  - Chaque requête envoyée au serveur doit contenir toutes les informations relatives à son état et est traitée indépendamment de toutes autres requêtes
  - Minimisation des ressources systèmes (pas de gestion de session, ni d'état)
- Interface uniforme basée sur les méthodes HTTP (GET, POST, PUT, DELETE)
- Les architectures RESTful sont construites à partir de ressources uniquement identifiées par des URI(s)

Requêtes REST

- Ressources
  - Identifiée par une URI (http://unice.fr/cursus/master/miage)
- Méthodes (verbes) permettant de manipuler les ressources (identifiants)
  - Méthodes HTTP : GET, POST, PUT, DELETE
- Représentation : Vue sur l'état de la ressource
  - Format d'échanges entre le client et le serveur (XML, JSON, text/plain,…)

Ressources

- Une ressource est un objet identifiable sur le système
- → Livre, Catégorie, Client, Prêt

Une ressources n'est pas forcément un objet matérialisé (Prêt, Consultation, Facture…)

- Une ressource est identifiée par une URI : Une URI identifie uniquement une ressource sur le système

http://ntdp.miage.fr/bookstore/books/1

Clef primaire de la ressource dans la BDD

Méthodes (Verbes)

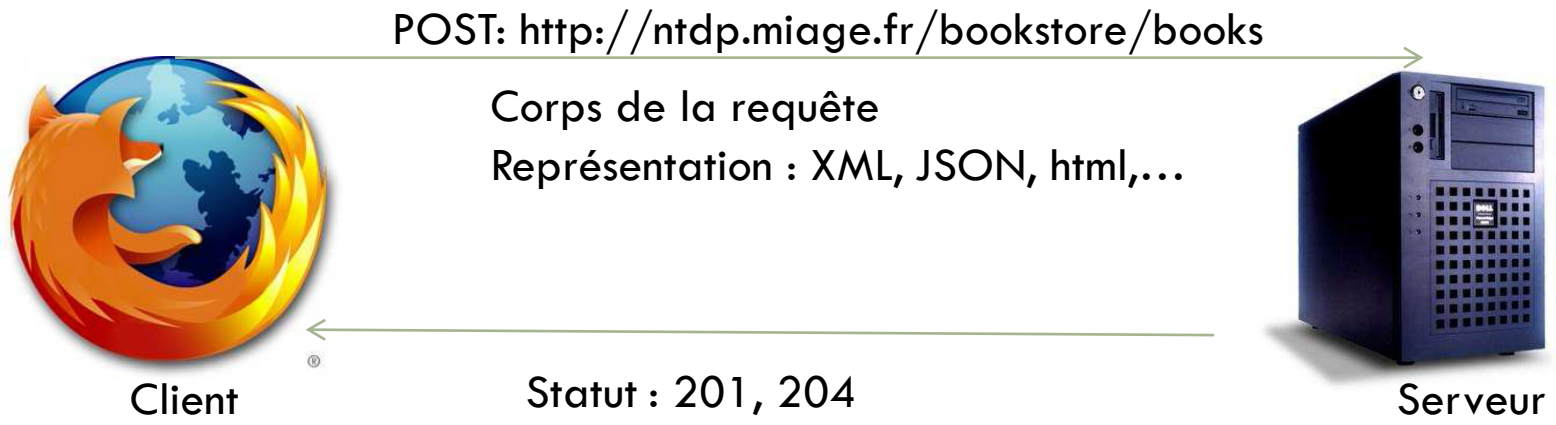
- Une ressource peut subir quatre opérations de bases CRUD correspondant aux quatre principaux types de requêtes HTTP (GET, PUT, POST, DELETE)
- REST s'appuie sur le protocole HTTP pour effectuer ces opérations sur les objets
  - CREATE → POST
  - RETRIEVE → GET
  - UPDATE → PUT
  - DELETE → DELETE

Méthode GET

- La méthode GET renvoie une représentation de la ressource tel qu'elle est sur le système

Méthode POST

- La méthode POST crée une nouvelle ressource sur le système

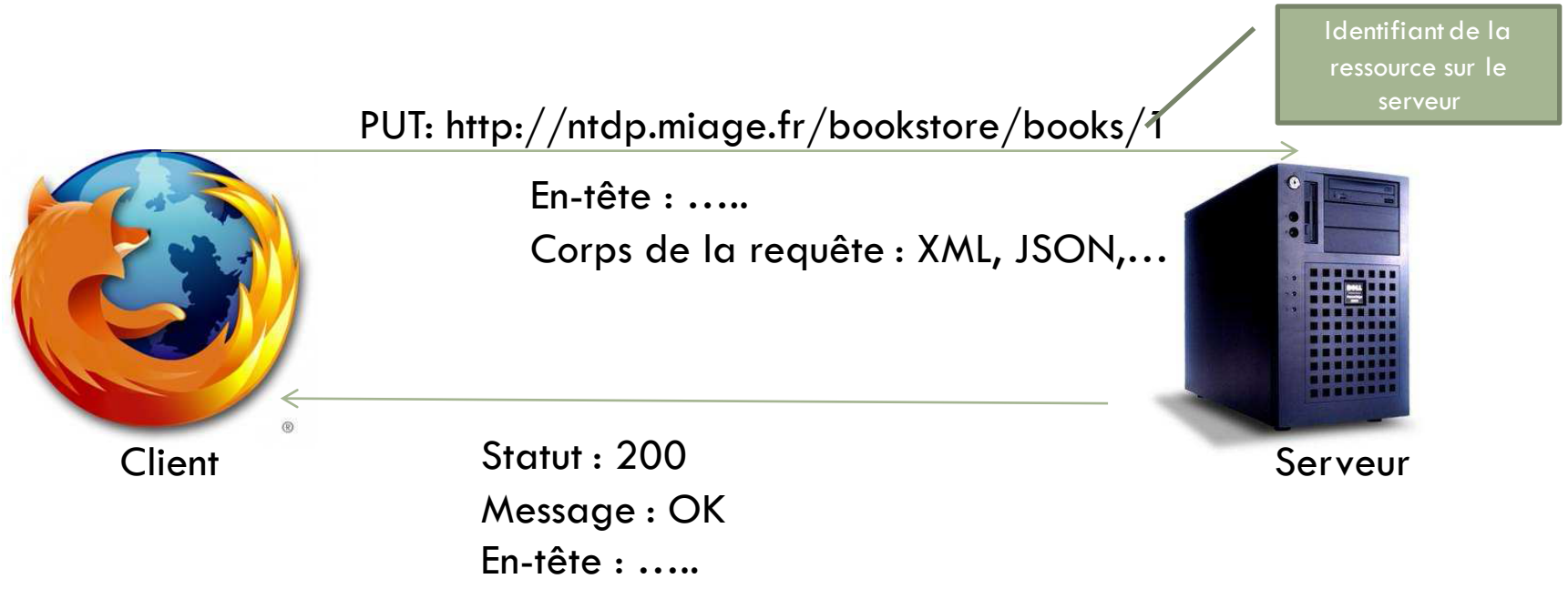


Méthode DELETE

- Supprime la ressource identifiée par l'URI sur le serveur

Méthode PUT

- Mise à jour de la ressource sur le système



Reflexions

- Que se passe t il
  - si on fait de la lecture avec un POST ?
  - Si on fait une mise à jour avec un DELETE ?
  - Si on fait une suppression avec un PUT ?

- ➔ REST ne l'interdit pas
- ➔ Mais si vous le faites, votre application ne respecte pas les exigences REST et donc n'est pas RESTFul

Représentation

Une représentation désigne les données échangées entre le client et le serveur pour une ressource:

- HTTP GET ➔ Le serveur renvoie au client l'état de la ressource
- PUT, POST ➔ Le client envoie l'état d'une ressource au serveur

Peut être sous différent format :

- JSON
- XML
- XHTML
- CSV
- Text/plain
- …..

WADL

- Web Application Description Language
- Standard du W3C
- Permet de décrire les éléments des services
  - Resource, Méthode, Paramètre, Réponse
- Permet d'interagir de manière dynamique avec les applications REST

→ Moins exploité que le WSDL pour les Services SOAP

WADL

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<application xmlns="http://wadl.dev.java.net/2009/02">
   <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 2.0 2013-05-03 14:50:15"/>
   <grammars/>
  ▼<resources base="http://localhost:8080/Bibliotheque/webresources/">
    ▼<resource path="category">
      ▼<method id="test" name="GET">
        ▼<response>
            <representation mediaType="application/xml"/>
            <representation mediaType="application/json"/>
          </response>
        </method>
      ▼<method id="apply" name="OPTIONS">
        ▼<request>
            <representation mediaType="*/*"/>
          </request>
        ▼<response>
            <representation mediaType="application/vnd.sun.wadl+xml"/>
          </response>
        </method>
      ▼<method id="apply" name="OPTIONS">
        ▼<request>
            <representation mediaType="*/*"/>
          </request>
        ▼<response>
            <representation mediaType="text/plain"/>
          </response>
        </method>
      ▼<method id="apply" name="OPTIONS">
        ▼<request>
            <representation mediaType="*/*"/>
          </request>
        ▼<response>
            <representation mediaType="*/*"/>
          </response>
        </method>
```

```
▼<resource path="application.wadl">
  ▼<method id="getWadl" name="GET">
    ▼<response>
        <representation mediaType="application/vnd.sun.wadl+xml"/>
        <representation mediaType="application/xml"/>
      </response>
    </method>
  ▼<method id="apply" name="OPTIONS">
    ▼<request>
        <representation mediaType="*/*"/>
      </request>
    ▼<response>
        <representation mediaType="text/plain"/>
      </response>
    </method>
  ▼<method id="apply" name="OPTIONS">
    ▼<request>
        <representation mediaType="*/*"/>
      </request>
    ▼<response>
        <representation mediaType="*/*"/>
      </response>
    </method>
```

Rappel → JSON

JSON

JSON « **J**ava**S**cript **O**bect **N**otation » est un format d'échange de données, facile à lire par un humain et interpréter par une machine.

Basé sur JavaScript, il est complètement indépendant des langages de programmation mais utilise des conventions qui sont communes à toutes les langages de programmation (C, C++, Perl, Python, Java, C#, VB, JavaScript,….)

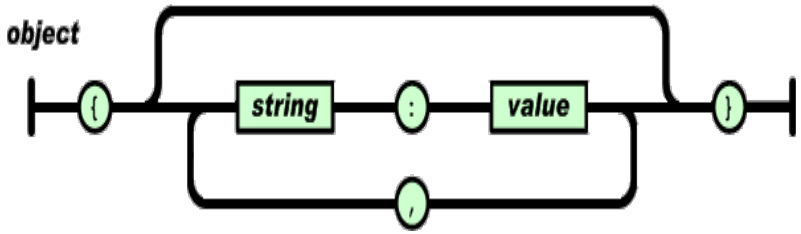
Deux structures :

- Une collection de clefs/valeurs → Object
- Une collection ordonnée d'objets → Array

JSON

Objet

Commence par un « { » et se termine par « } » et composé d'une liste non ordonnée de paire clefs/valeurs. Une clef est suivie de « : » et les paires clef/valeur sont séparés par « , »

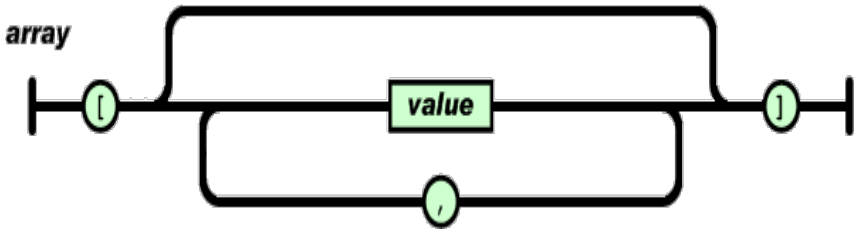



```
{ "id": 51,
"nom": "Mathematiques 1", "resume":
"Resume of math ", "isbn": "123654",
"categorie":
     {
     "id": 2, "nom": "Mathematiques",
     "description": "Description of
     mathematiques "
     },
"quantite": 42,
"photo": ""
}
```

JSON

ARRAY

Liste ordonnée d'objets commençant par « [« et se terminant par «] », les objets sont séparés l'un de l'autre par « , ».

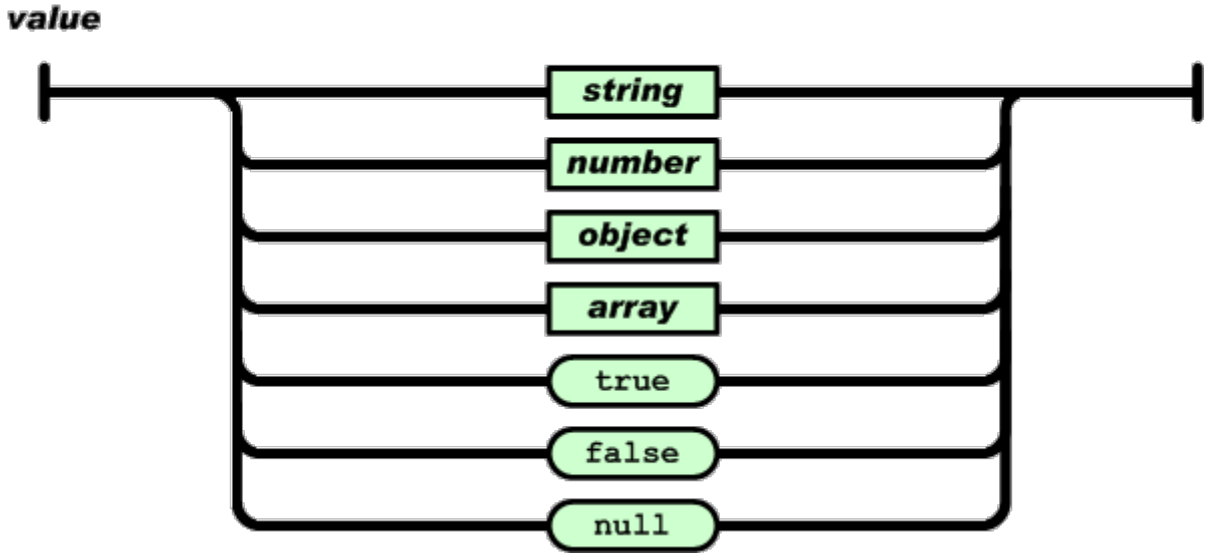



```
[
{ "id": 51,
"nom": "Mathematiques 1",
"resume": "Resume of math ",
"isbn": "123654",
"quantite": 42,
"photo": ""
},
{ "id": 102,
"nom": "Mathematiques 1",
"resume": "Resume of math ",
"isbn": "12365444455",
"quantite": 42,
"photo": ""
}
]
```

JSON

Value

Un objet peut être soit un string entre « ""» ou un nombre (entier, décimal) ou un boolean (true, false) ou null ou un objet.



Services Web étendus VS REST

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body> <ns2:hello
xmlns:ns2="http://services.bibliotheque.ntdp.miage.unice.fr/">
<name>Miage NTDP</name>
</ns2:hello>
</S:Body>
</S:Envelope>
```

Client

SOAP

Serveur

Client

http://localhost:8080/Bibliotheque/webresources/category/Miage%20NTDP

REST

Serveur

Services Web étendus VS REST

SOAP

- Avantages
  - Standardisé
  - Interopérabilité
  - Sécurité (WS-Security)
- Inconvénients
  - Performances (enveloppe SOAP supplémentaire)
  - Complexité, lourdeur
  - Cible l'appel de service

Services Web étendus VS REST

REST

- Avantages
  - Simplicité de mise en œuvre
  - Lisibilité par un humain
  - Evolutivité
  - Repose sur les principes du web
  - Représentations multiples (XML, JSON,…)
- Inconvénients
  - Sécurité restreinte par l'emploi des méthodes HTTP
  - Cible l'appel de ressources

WADL

- Web Application Definition Language est un langage de description des services REST au format XML. Il est une spécification di W3C initié par SUN (www.w.org/Submission/wadl)
- Il décrit les éléments à partir de leur type (Ressources, Verbes, Paramètre, type de requête, Réponse)
- Il fournit les informations descriptives d'un service permettant de construire des applications clientes exploitant les services REST.

Partie 2

Développer des Web Services REST avec JAVA

Session 1

JAX-RS

- Acronyme de Java API for RestFul Web Services
- Version courante 2.0 décrite par JSR 339
- Depuis la version 1.1, il fait partie intégrante de la spécification Java EE 6
- Décrit la mise en œuvre des services REST web coté serveur
- Son architecture se repose sur l'utilisation des classes et des annotations pour développer les services web

JAX-RS ➔ Implémentation

- JAX-RS est une spécification et autour de cette spécification sont développés plusieurs implémentations
  - JERSEY : implémentation de référence fournie par Oracle (http://jersey.java.net)
  - CXF : Fournie par Apache (http://cfx.apache.org)
  - RESTEasy : fournie par JBOSS
  - RESTLET : L'un des premiers framework implémentant REST pour Java

JERSEY

- Version actuelle 2.3.1 implémentant les spécifications de JAX-RS 2.0
- Intégré dans Glassfish et l'implémentation Java EE (6,7)
- Supportés dans Netbeans

JAX-RS : Développement

- Basé sur POJO (Plain Old Java Object) en utilisant des annotations spécifiques JAX-RS
- Pas de modifications dans les fichiers de configuration
- Le service est déployé dans une application web
- Pas de possibilité de développer le service à partir d'un WADL contrairement à SOAP
- Approche Bottom/Up
  - Développer et annoter les classes
  - Le WADL est automatiquement généré par l'API

Annotation JAX-RS

La spécification JAX-RS dispose d'un ensemble d'annotation permettant d'exposer une classe java dans un services web :

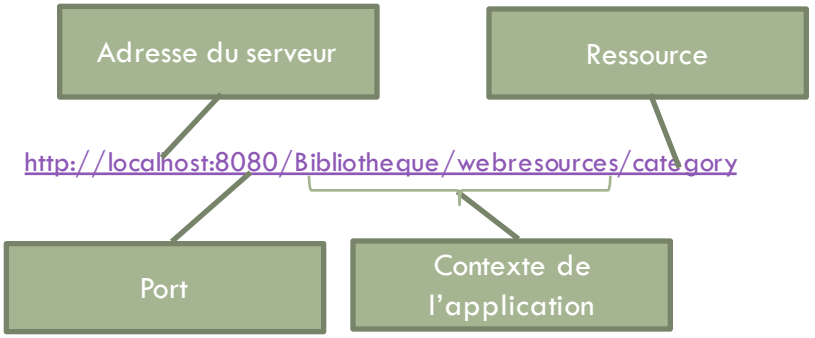
- @Path
- @GET, @POST, @PUT, @DELETE
- @Produces, @Consumes
- @PathParam

Modéliser les URIs

URIs sont déterminés par l'annotation @Path

- Permet d'exposer une classe dans le WS
- Définit la racine des ressources (Root Racine Ressources)
- Sa valeur correspond à l'URI relative de la ressource

```
@Path("category")
public class CategoryService  {
……
}
```



URIs des méthodes

- @Path peut être utilisée pour annoter des méthodes d'une classe
- L'URI résultante est la concaténation entre le valeur de @Path de la classe et celle de la méthode

```
@Path("category")
public class CategoryFacade {
@GET
@Produces({MediaType.APPLICATION_XML,
MediaType.APPLICATION_JSON})
@Path("test")
public String hello()
{
return "Hello World!";
}
..
}
```

http://localhost:8080/Bibliotheque/webresources/category/test

URIs dynamiques

- La valeur définie dans l'annotation @Path n'est forcément un constante, elle peut être variable.
- Possibilité de définir des expressions plus complexes, appelées Template Parameters
- Les contenus complexes sont délimités par « {} »
- Possibilité de mixer dans la valeur @Path des expressions régulières

```
@GET
@Consumes ({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
@Produces ({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
@Path( "hello/{nom}")
public String hello (@PathParam("nom") String nom){
return "Hello " + nom;
}
```

http://localhost:8080/Bibliotheque/webresources/category/hello/Miage

URIs dynamiques

```
   @GET
   @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    @Path("coucou/{nom}/{prenom}")
    public String hello(@PathParam("nom") String nom,
@PathParam("prenom") String prenom) {
        return "Hello " + nom + " " + prenom;
    }
```

GET http://localhost:8080/Bibliotheque/webresources/category/coucou/Miage/NTDP

```
  @GET
    @Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    @Path("{id}")
    public Categorie find (@PathParam("id") Long id) {
        return super.find(id);

    }
```

GET http://localhost:8080/Bibliotheque/webresources/category/1

@GET, @POST, @PUT, @DELETE

- Permettent de mapper une méthode à un type de requête HTTP
- Ne sont utilisables que sur des méthodes
- Plusieurs méthodes peuvent avoir le même chemin, le mapping uri/méthode est fait automatiquement par JAX-RS en fonction du type de la requête

http://localhost:8080/Bibliotheque/webresources/category/test

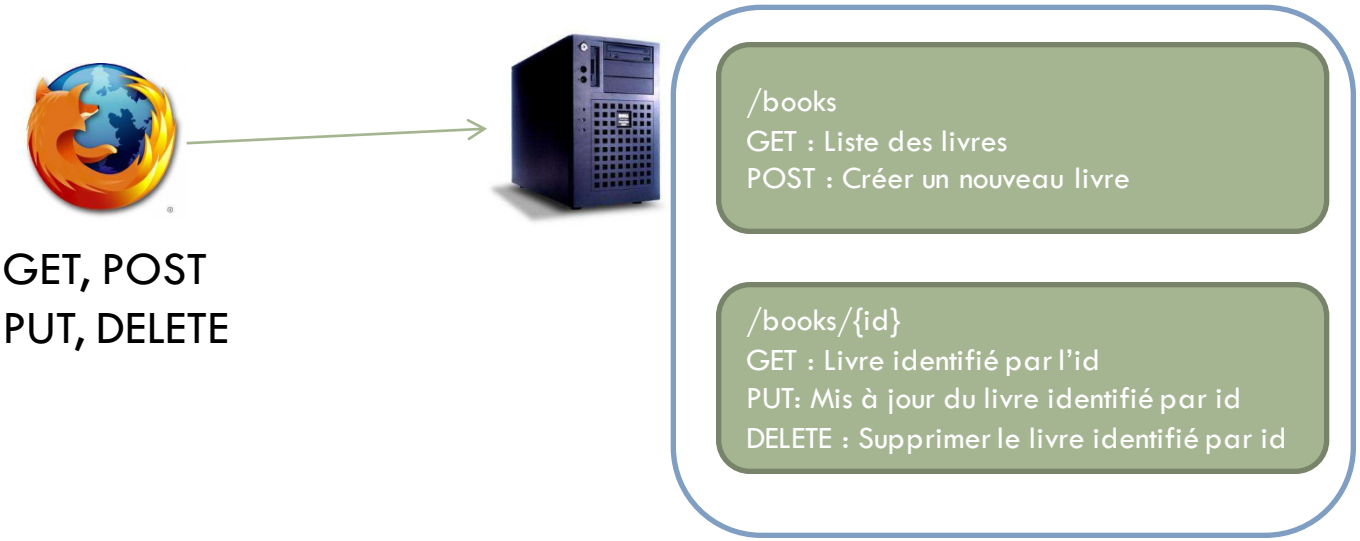
```
@GET
    @Produces({MediaType.APPLICATION_JSON,
MediaType.APPLICATION_XML})
    @Path("hello/{nom}/{prenom}")
    public String hello(@PathParam("nom") S
nom, @PathParam("prenom") String prenom) {
        return "GET " + nom + " " + prenom
    }
```

```
 @POST
    @Produces({MediaType.APPLICATION_JSON,
MediaType.APPLICATION_XML})
    @Path("hello/{nom}/{prenom}")
    public String helloPost(@PathParam("nom")
String nom, @PathParam("prenom") String prenom) {
        return "POST " + nom + " " + prenom;
    }
```

[GET/POST] http://localhost:8080/Bibliotheque/webresources/hello/Miage/NTDP

@GET, @POST, @PUT, @DELETE

- Les opérations CRUD sur les ressources sont réalisées au travers des méthodes de la requête HTTP



Outils de test

- Il existe de nombreux outils en ligne permettant de tester les services Web REST

- Certains sont disponibles sous forme d'extansion que vous pouvez installer dans les navigateurs
  - RestConsole
  - PostMan

Travaux Dirigés!