



WEBSERVICES – Architecture REST

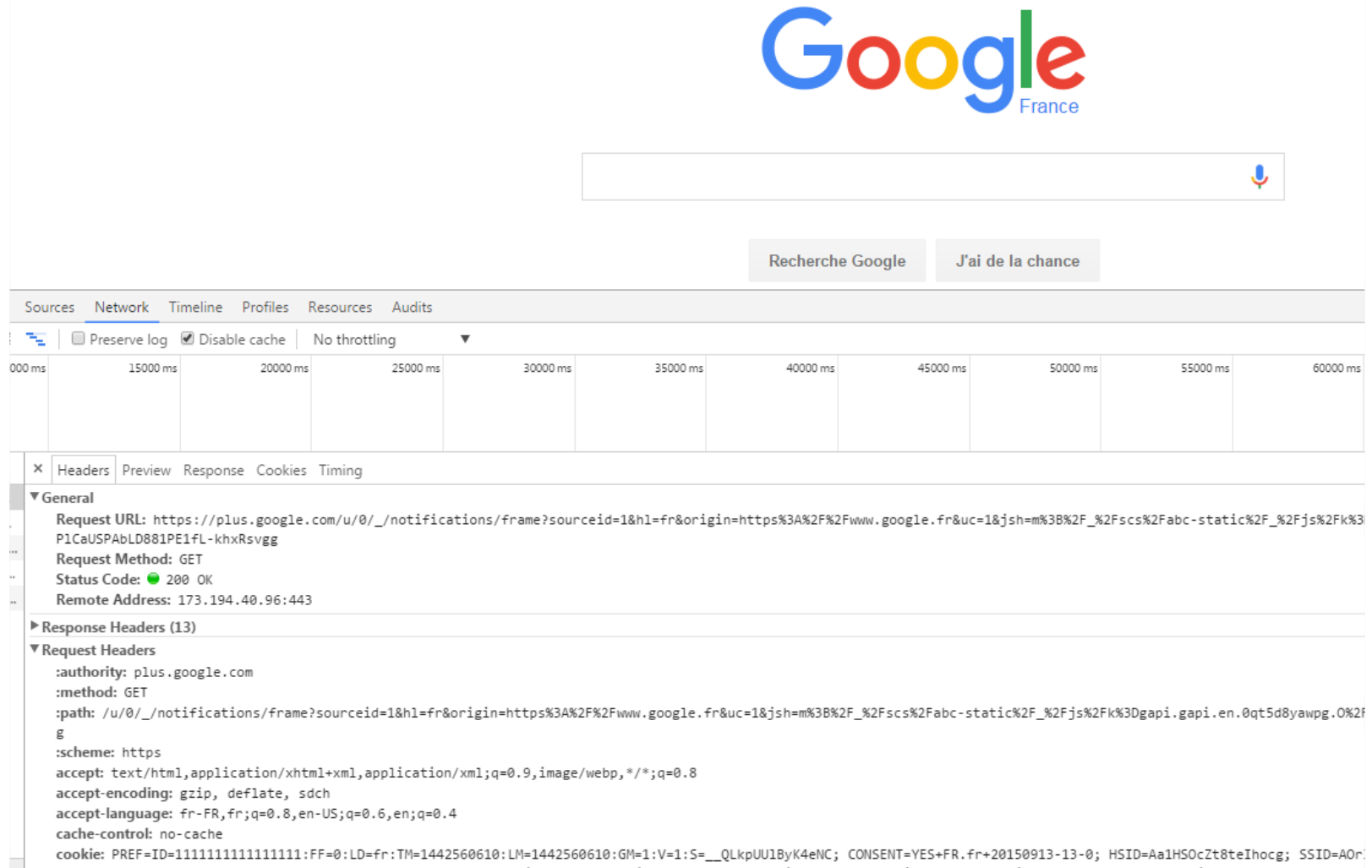
Max Devulder

Sommaire

- I. Introduction**
- II. Le protocole HTTP
- III. REST vs SOAP
- IV. Implémentation REST en java
- V. Consommation par un client : JQuery

- REST = **RE**presentational **S**tate **T**ransfer
- **Pattern d'architecture** pour développer des web services sous forme de **ressources**.
- Les WS Rest communiquent au travers du **protocole HTTP** :
 - Methodes : GET, POST, PUT, DELETE ...
 - Syntaxe : Path, Parameters
 - Médias : XML, JSON, HTML, PLAIN TEXT ...
 - HTTP response : 404, 200, 503 ...

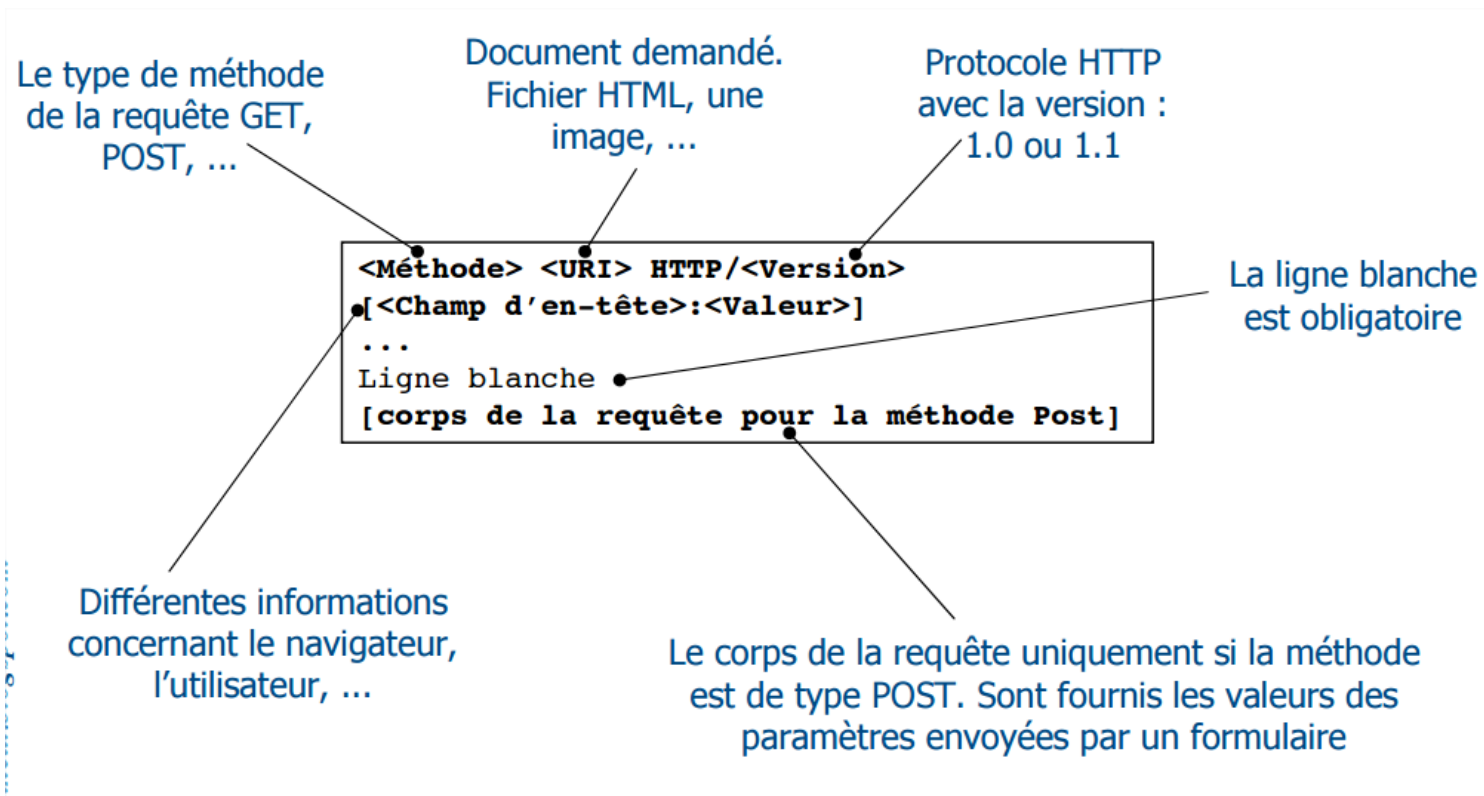
- HTTP est un protocole de communication pair à pair.



The screenshot shows the Google France homepage. Below the search bar, the Chrome DevTools Network tab is open, displaying a request to a notification frame. The request details are as follows:

- General**
 - Request URL: `https://plus.google.com/u/0/_/notifications/frame?sourceid=1&hl=fr&origin=https%3A%2F%2Fwww.google.fr&uc=1&jsh=m%3B%2F_%2Fscs%2Fabc-static%2F_%2Fjs%2Fk%3P1CaUSPAbLD881PE1fL-khxRsvgg`
 - Request Method: GET
 - Status Code: 200 OK
 - Remote Address: 173.194.40.96:443
- Response Headers (13)**
- Request Headers**
 - `:authority: plus.google.com`
 - `:method: GET`
 - `:path: /u/0/_/notifications/frame?sourceid=1&hl=fr&origin=https%3A%2F%2Fwww.google.fr&uc=1&jsh=m%3B%2F_%2Fscs%2Fabc-static%2F_%2Fjs%2Fk%3Dgapi.gapi.en.0qt5d8yawpg.0%2Fg`
 - `:scheme: https`
 - `accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8`
 - `accept-encoding: gzip, deflate, sdch`
 - `accept-language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4`
 - `cache-control: no-cache`
 - `cookie: PREF=ID=11111111111111111111:FF=0:LD=fr:TM=1442560610:LM=1442560610:GM=1:V=1:S=__QLkpUU1ByK4eINC; CONSENT=YES+FR.fr+20150913-13-0; HSID=Aa1HS0cZt8teIhocg; SSID=AOr...`

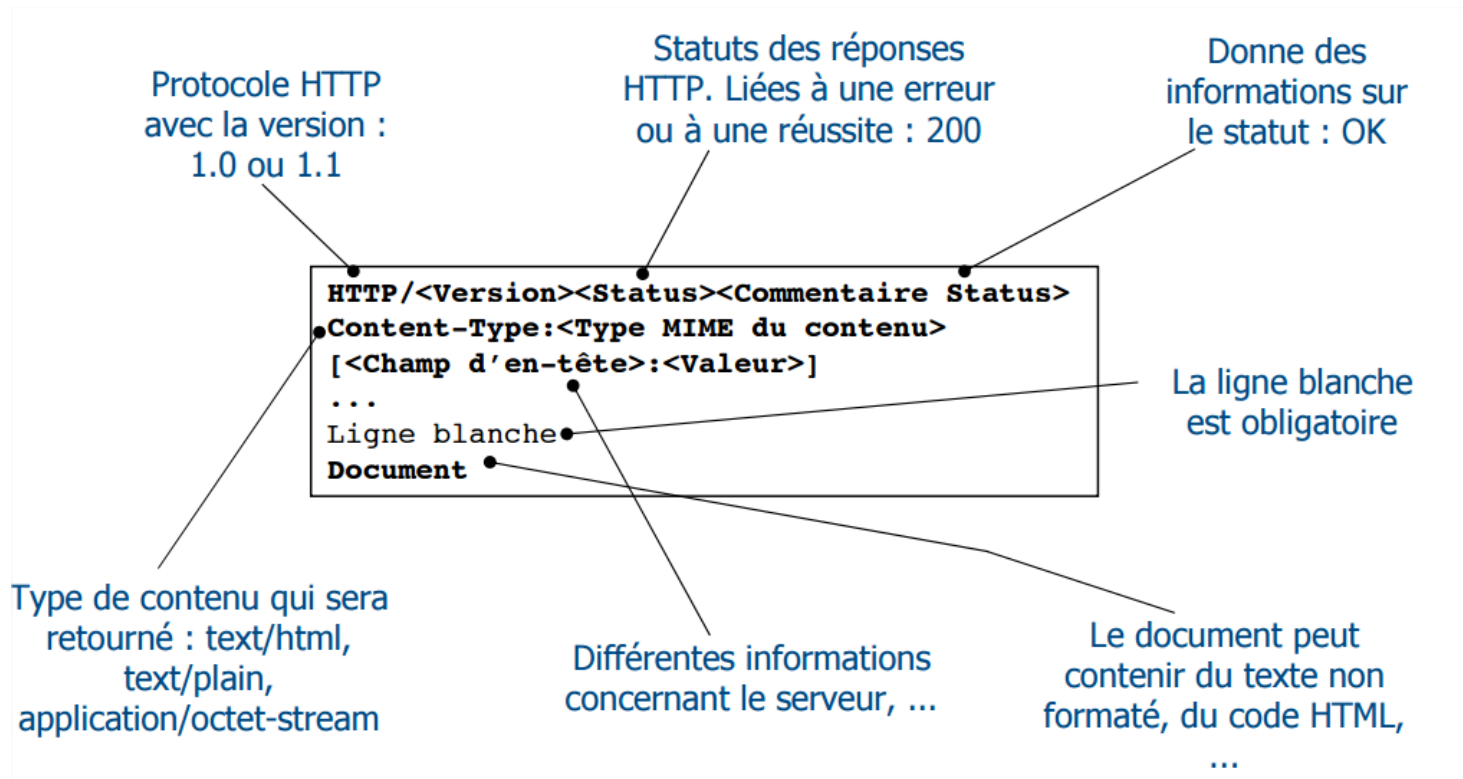
➤ Les requêtes : **Format**



➤ Les requêtes : **Détail**

Requête	Détail
Accept	Type MIME accepté par le client (text/html, application/json ..)
Accept-encoding	Encodage accepté (compress, gzip, x-zip...)
Accept-charset	Jeu de caractère du navigateur (dos, unix ...)
Accept-language	Langue de votre navigateur
Cookie	
From	@mail de l'utilisateur

➤ Les réponses: **Format**



➤ Les responses: **Détail**

Requête	Détail
Age	Ancienneté du document en secondes
Server	Diverses informations sur le serveur
WWW-Authenticate	Système d'authentification
Accept-language	Langue de votre navigateur
Location	...

➤ Les responses: **Status** (https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP)

Code	Détail
200	Succès de la requête
301, 302	Redirection, respectivement permanente et temporaire
403	Accès refusé
404	Page non trouvée
500 et 503	Erreur serveur

➤ REST vs SOAP

SOAP



REST



➤ REST vs SOAP

	SOAP	REST
Implémentation	JAX-WS standard	JAX-RS standard (intégré dans Java >= 1.6)
Interface	Nécessite un WDSL pour exposer ses services	Pas d'interface, seulement des méthodes HTTP
Media	XML	XML & JSON
Procotol	SOAP protocol	HTTP protocol
Human readable	Non	Oui

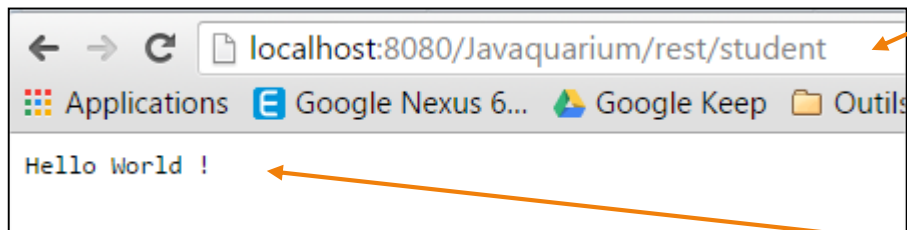
- JAX-RS = Standard (<https://jcp.org/en/jsr/detail?id=339>)
- Plusieurs implémentations :

Nom	Détail
JERSEY	Implémentation de référence fournir par Oracle
CXF	Apache
RESTEasy	JBoss
RESTlet	L'un des premiers framework indépendant
WINK	Apache

- Bon ok, et techniquement ? Très simple !
- Basé sur 2 concepts : Les **beans** et les **annotations**

```
@Path("/student")
public class StudentRestService {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayHello() {
        return "Hello World !";
    }
}
```



Requête HTTP GET

Type de retour directement interprétable par le navigateur

- Jersey (rappe = l'implémentation Oracle de JAX-RS) convertie pour vous les ressources en média.

```
public class Student {  
    private int id;  
    private String name;  
    private Integer age;  
    // getter + setter  
}
```

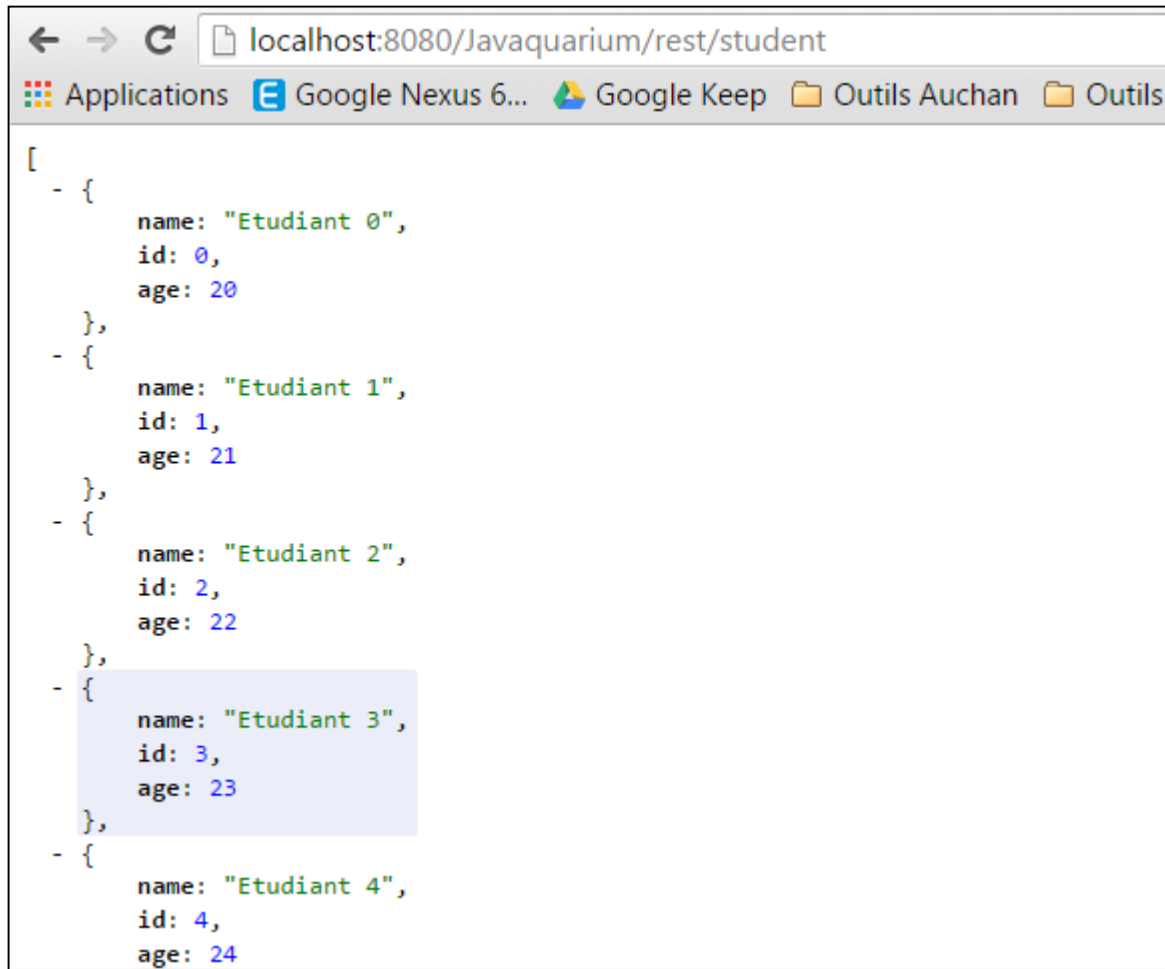
Un bean classique

Requête HTTP GET

Média = JSON

Type de retour complexe !

```
@Path("/student")  
public class StudentRestService {  
  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public List<Student> getAll() {  
        final List<Student> students;  
        students = new ArrayList<Student>(10);  
        for (int i = 0; i < 10; i++) {  
            students.add(new Student(i, "Etudiant " + i, 20 + i));  
        }  
        return students;  
    }  
}
```



```
[
  - {
    name: "Etudiant 0",
    id: 0,
    age: 20
  },
  - {
    name: "Etudiant 1",
    id: 1,
    age: 21
  },
  - {
    name: "Etudiant 2",
    id: 2,
    age: 22
  },
  - {
    name: "Etudiant 3",
    id: 3,
    age: 23
  },
  - {
    name: "Etudiant 4",
    id: 4,
    age: 24
  }
]
```

- Base de l'architecture REST = créer **une API** de type **CRUD** !
- Facilement compréhensible par l'humain, pas d'explication à donner.

```
@Path("/student")
public class StudentRestService {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public List<Student> getAll()

    @GET
    @Path("/{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public String getOne(@PathParam("id") Integer id)

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    public Response create(Student student)

    @PUT
    @Consumes(MediaType.APPLICATION_JSON)
    public Response update(Student student)

    @DELETE
    @Consumes(MediaType.APPLICATION_JSON)
    public Response delete(Student student)
```


- Base de l'architecture REST = créer **une API** de type **CRUD** !
- Facilement compréhensible par l'humain, pas d'explication à donner.

URL	Verbe	Action
http://.../rest/student	GET	Retourne la liste des étudiants
http://.../rest/student/1	GET	Retourne le 1 ^{er} étudiant
http://.../rest/student/fr/1	GET	Retourne le 1 ^{er} étudiant français
http://.../rest/student/	POST	Créer un nouvel étudiant
http://.../rest/student/	PUT	Mise à jour d'un étudiant
http://.../rest/student/	DELETE	Supprime un étudiant

➤ @PATH

Où ? Sur la classe ou une méthode

Pourquoi ? Donne une URI relative à un contexte



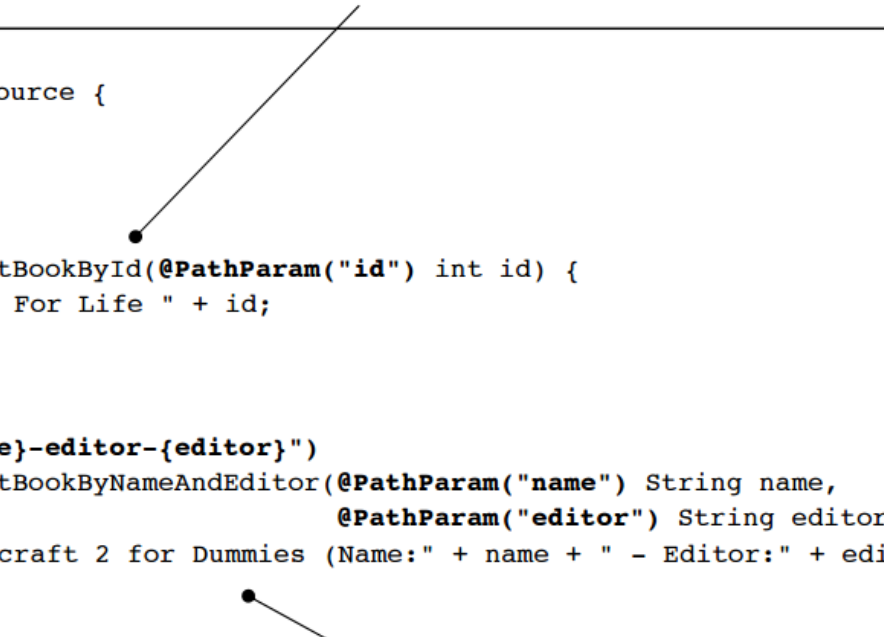
➤ @PATH

Permet également le passage de paramètres

```
/books/123
@Path("/books")
public class BookResource {
    ...

    @GET
    @Path("{id}")
    public String getBookById(@PathParam("id") int id) {
        return "Java For Life " + id;
    }

    @GET
    @Path("name-{name}-editor-{editor}")
    public String getBookByNameAndEditor(@PathParam("name") String name,
                                         @PathParam("editor") String editor) {
        return "Starcraft 2 for Dummies (Name:" + name + " - Editor:" + editor + ")";
    }
}
```



➤ Les autres :

Annotation	Détail
@PathParam	Mapper les valeurs de l'URI sur les paramètres de méthode
@QueryParam	Extraire les paramètres de requête
@FormParam	Extraire les paramètres du form
@HeaderParam	Extraire les paramètres de l'entête
@CookieParam	Extraire les paramètres des cookies
@Context	Extraire les informations lié au ressource de contexte.

➤ Les autres :

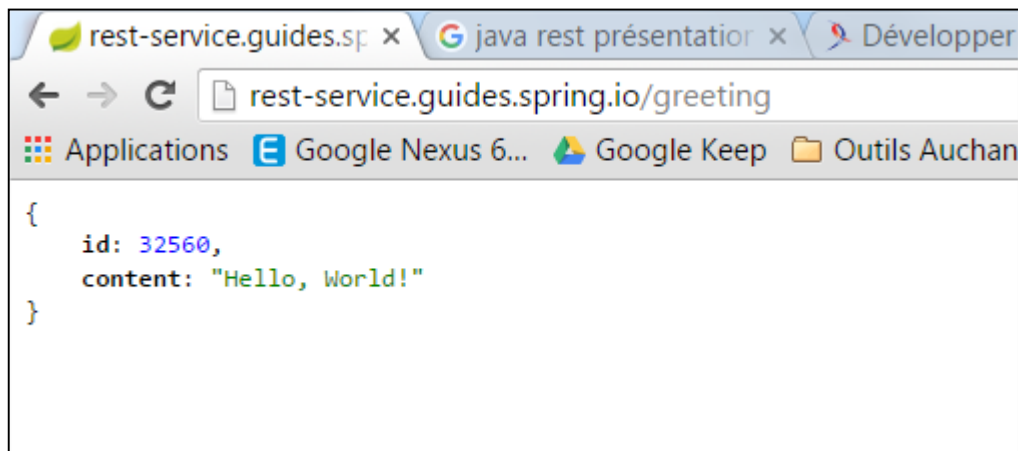
Annotation	Détail
@Consumes	Type MIME d'entrée (request)
@Produces	Type MIME de sortie (response)

➤ **Concept à étudier :**

- > Gestion des exceptions
- > Type mime exotique ? Fichier, vidéo ...
- > Mise en place de la sécurité ?

➤ Exemple sur le site de SPRING : <https://spring.io/guides/gs/consuming-rest-jquery/>

Web service d'exemple : <http://rest-service.guides.spring.io/greeting>



```
$(document).ready(function() {  
    $.ajax({  
        url: "http://rest-service.guides.spring.io/greeting"  
    }).then(function(data) {  
        $('#greeting-id').append(data.id);  
        $('#greeting-content').append(data.content);  
    });  
});
```