



WEBSERVICES – Architecture REST

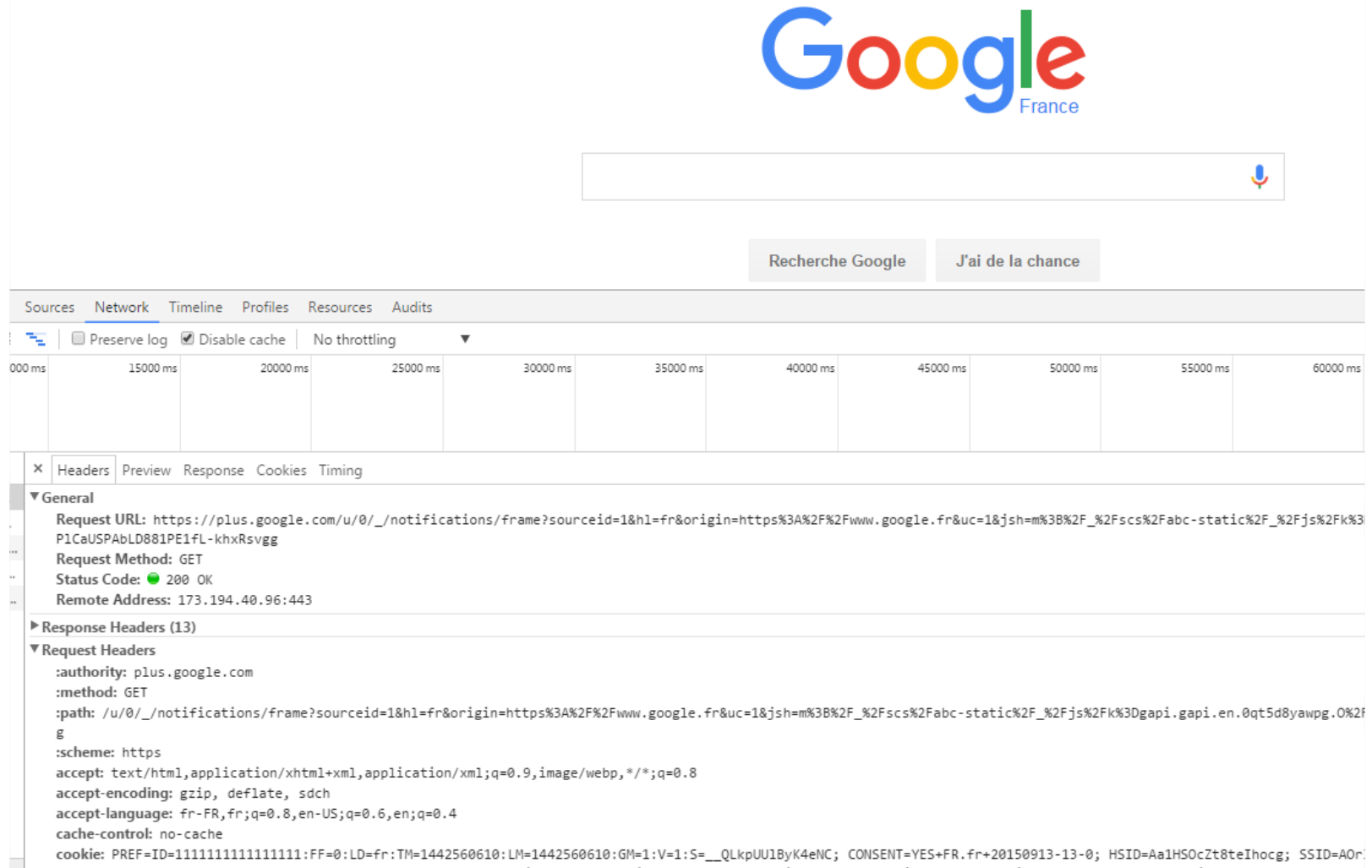
Max Devulder

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 ...

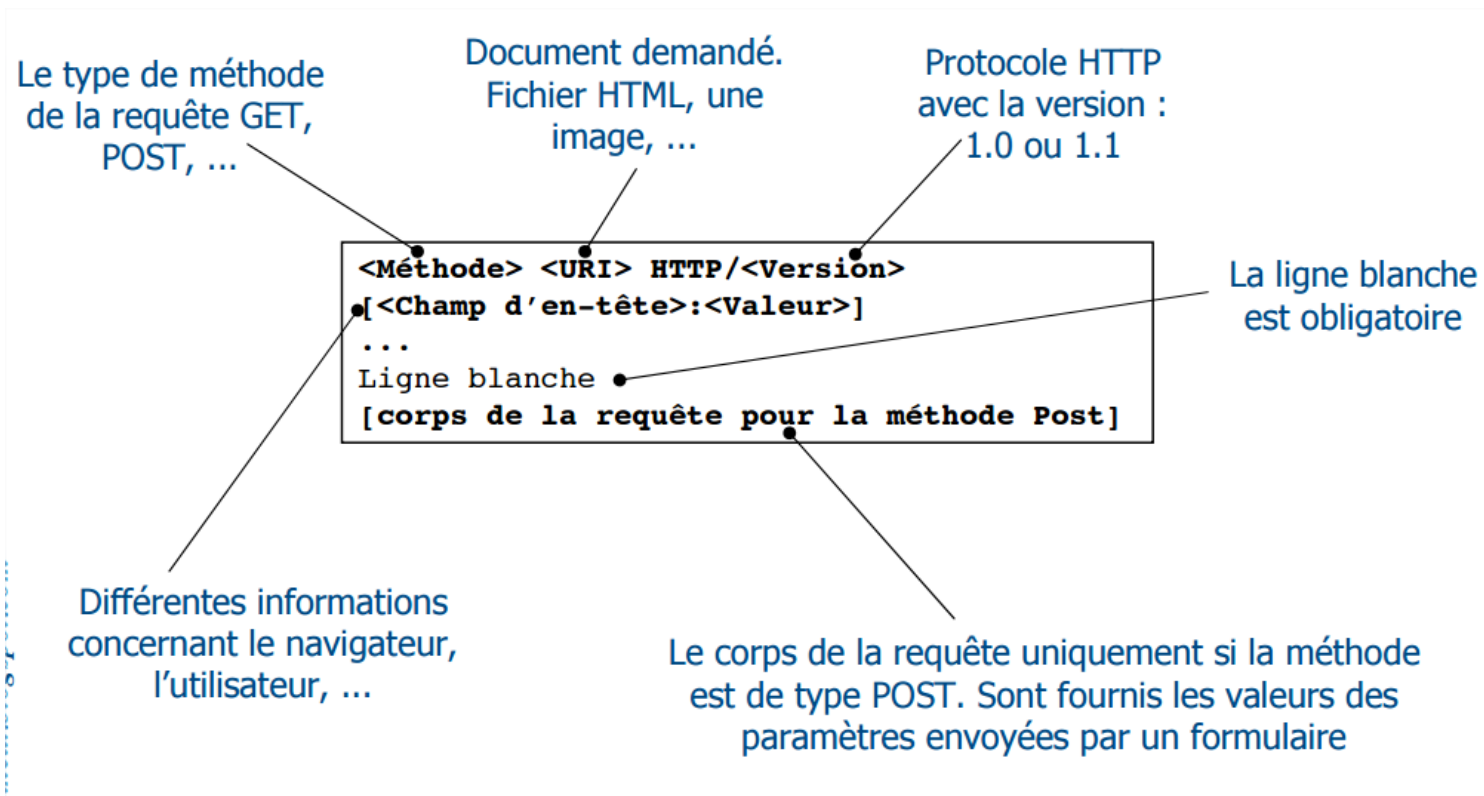
- HTTP est un protocole de communication pair à pair.



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=1111111111111111: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...`

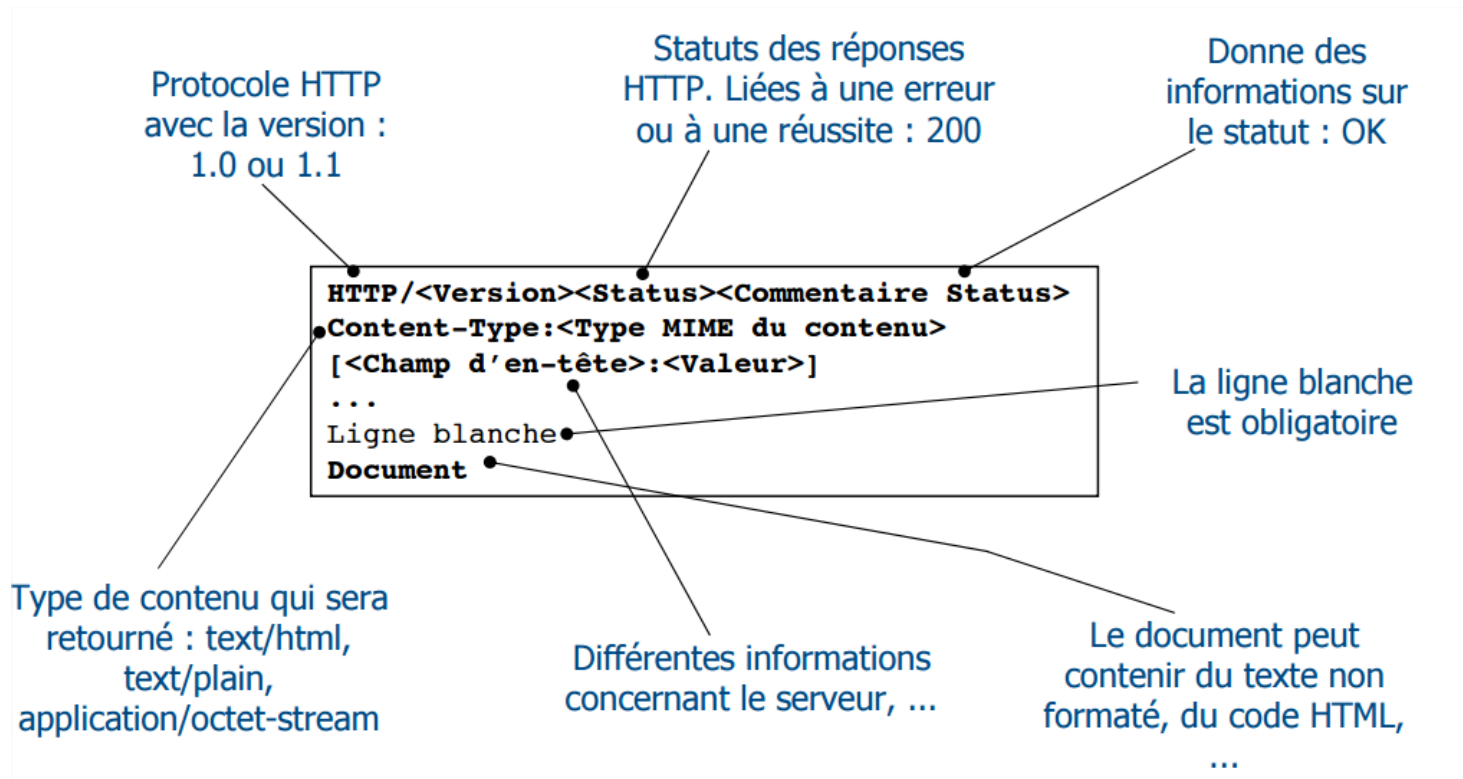
➤ Les requêtes : **Format**



➤ 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**



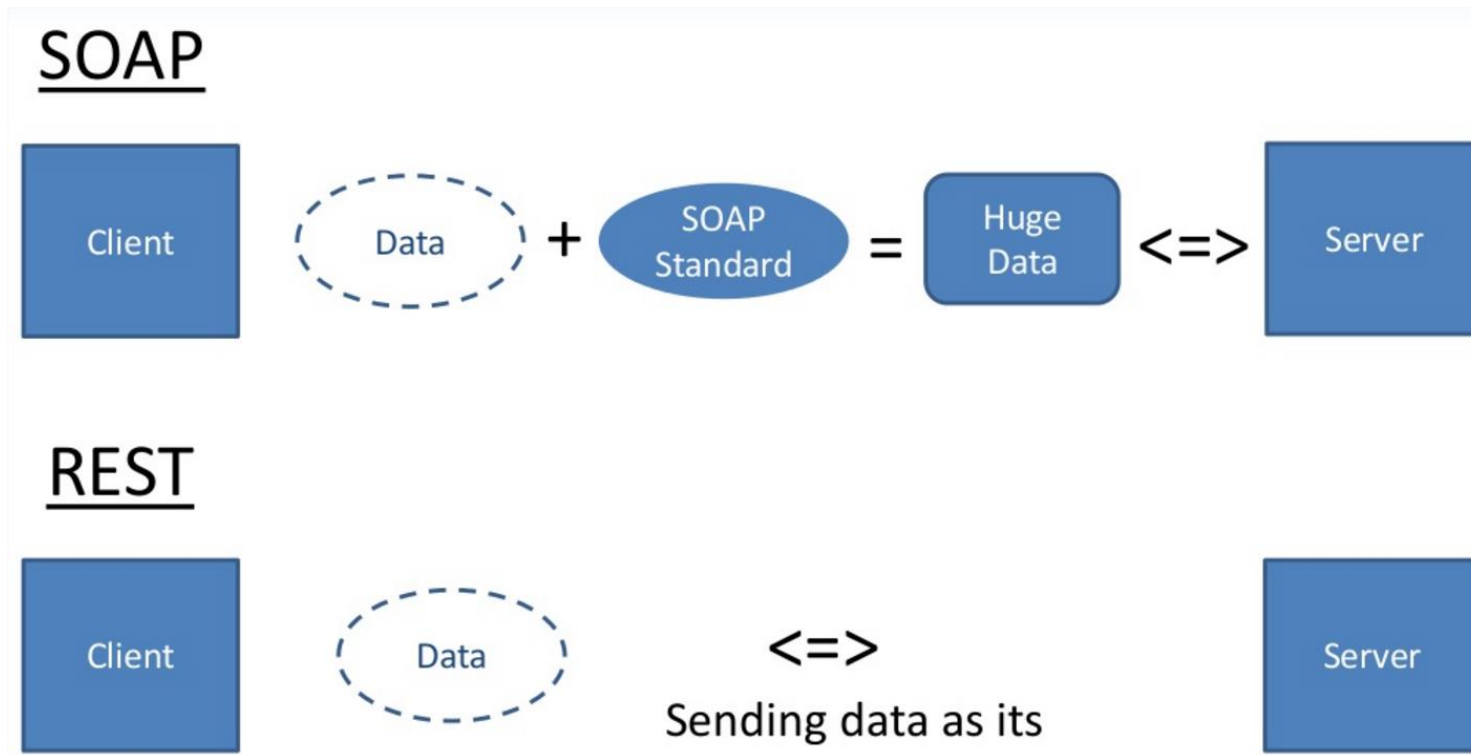
➤ Les responses: **Détail**

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



➤ 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

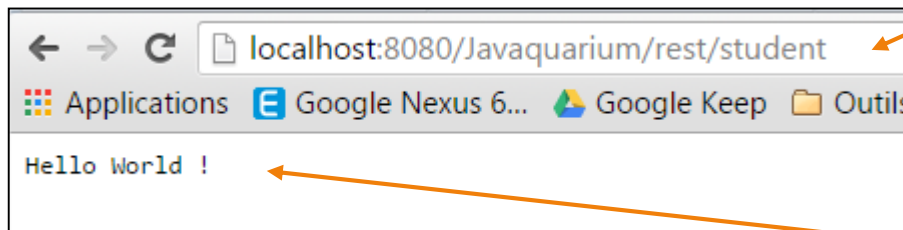
- JAX-RS = Standard (<https://jcp.org/en/jsr/detail?id=339>)
- Plusieurs implémentations :

Nom	Détail
Spring-web	Le plus populaire.
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**

```
@RestController
@RequestMapping(value = "/student")
public class StudentRestService {

    @RequestMapping(method = RequestMethod.GET)
    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.

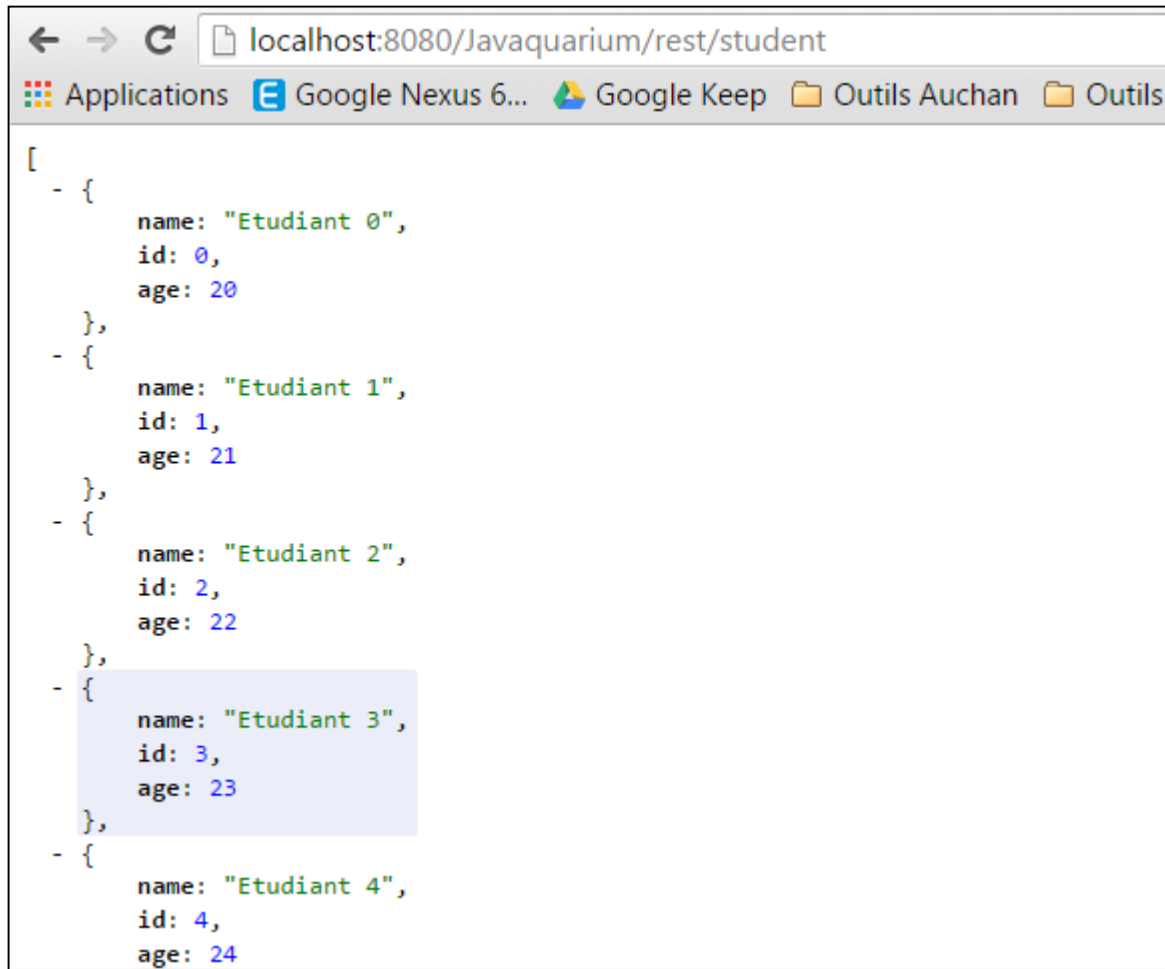
```
public class Student {  
    private int id;  
    private String name;  
    private Integer age;  
    // getter + setter  
}
```

Un bean classique

Requête HTTP GET

Type de retour complexe !

```
@RestController  
@RequestMapping(value = "/student")  
public class StudentRestService {  
  
    @RequestMapping(method = RequestMethod.GET)  
    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;  
    }  
}
```



A screenshot of a web browser window displaying a REST API response. The address bar shows the URL `localhost:8080/Javaaquarium/rest/student`. The browser's tab bar includes icons for Applications, Google Nexus 6..., Google Keep, Outils Auchan, and Outils. The main content area displays a JSON array of five student objects. The fourth object, representing 'Etudiant 3', is highlighted with a light blue background. The JSON structure is as follows:

```
[
  - {
    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 {

    @RequestMapping(method = RequestMethod.GET)
    public List<Student> getAll()

    @RequestMapping(value =("/{id}", method = RequestMethod.GET)
    public String getOne(@PathVariable Integer id)

    @RequestMapping(method = RequestMethod.POST)
    public Response create(@RequestBody Student student)

    @RequestMapping(method = RequestMethod.PUT)
    public Response update(@RequestBody Student student)

    @RequestMapping(value =("/{id}", method = RequestMethod.DELETE)
    public Response delete(@PathVariable Integer id)
```


- 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

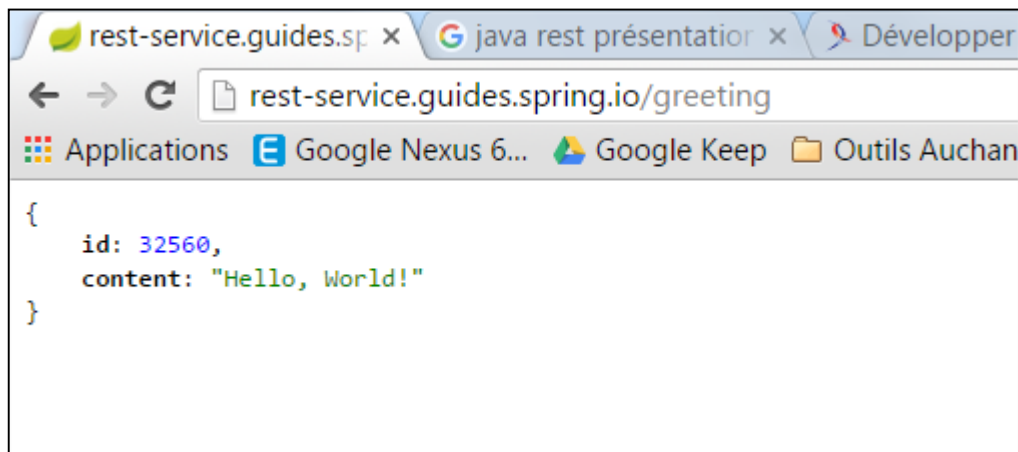


➤ **Concept à étudier :**

- > 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);
    });
});
```

➤ JSON = Indépendance totale Front / Back

Name	Type	Year released	GitHub stars	Maintained by
Angular	Framework	2010	58.4K	Google
React.js	Library	2013	95.8K	Facebook
Vue.js	Framework	2014	75K	Community
Ember.js	Framework	2011	19K	Community
Next.js	Framework	2016	25.3K	React (Facebook)

FRONTEND

JSON

SPRING

TOMCAT

JAVA

BACKEND

Spring DATA

➤ Une autre façon d'accéder aux données.

```
@Entity
@Table(name = "PRDVSPDA")
public class JourFerieDO {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Integer id;

    @Column(name = "SD_DATE")
    private Date date;

    @Column(name = "IS_ON")
    private Integer travaille;

    @Column(name = "SD_TYPE")
    private String type;
```

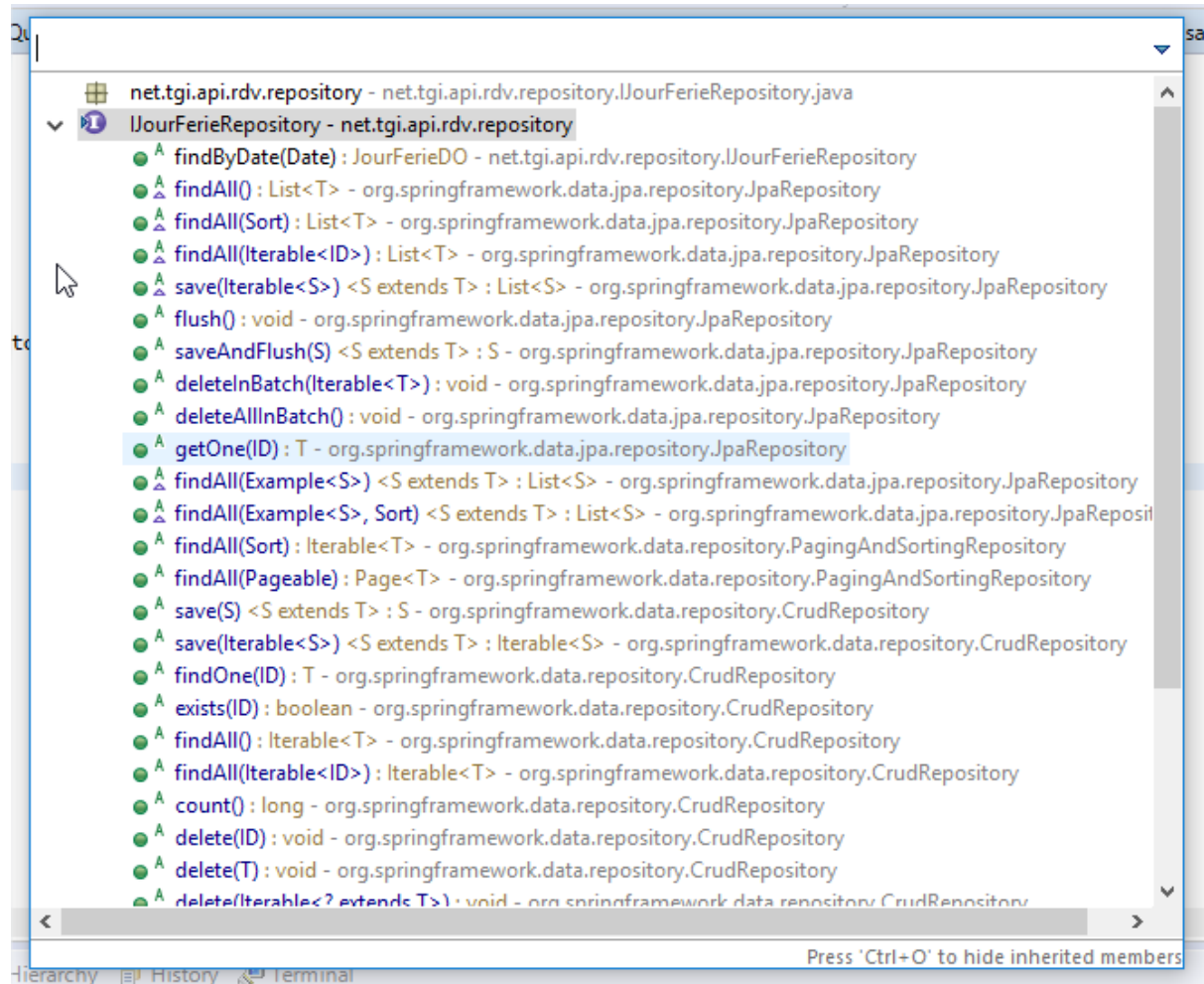
➤ Une autre façon d'accéder aux données.

```
@Transactional(propagation = Propagation.MANDATORY)
public interface IJourFerieRepository extends JpaRepository<JourFerieDO, Integer> {

    // Nothing here.

}
```


➤ Une interface vide ? Pas tout à fait...



➤ Du HQL pour les requêtes complexes avec @Query

```
15- /**
16  * Repository for {@link OperationNavireDO}
17  *
18  * @author MaxD
19  */
20 @Transactional(propagation = Propagation.MANDATORY)
21 public interface IOperationNavireRepository extends JpaRepository<OperationNavireDO, Long> {
22
23- /**
24  * @param escale
25  * @return
26  */
27 Set<OperationNavireDO> findDistinctByEscaleAndTagSuppressionOrderByTypeOperationAscPositionABordDesc(final Stri
28
29- @Query("select distinct openav from OperationNavireDO openav " +
30      "where openav.escale = :codeEscale " +
31      "and openav.tagSuppression = :tagSuppression " +
32      "and substring(openav.positionABord, 1,3) in :baies " +
33      "order by openav.typeOperation asc, openav.positionABord desc ")
34 Set<OperationNavireDO> findDistinctByEscaleAndTagSuppressionAndBaiesOrderByTypeOperationAscPositionABordDesc(@P
35
```