esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

JAX-RS

Module Services Web

A.U 2021-2022

United Nations Educational, Scientific and Cultural Organization
UNESCO Chair "Project-based learning" ESPRIT School of engineering, Tunisia
EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur
CONFÉRENCE DES GRANDES ÉCOLES
cdio CONCEIVE DESIGN IMPLEMENT OPERATE

Objectifs

- Comprendre le style d'architecture REST.
- Concevoir et consommer des services Web RESTful en utilisant l'API JAX-RS.
- Assurer l'interopérabilité entre un client et un serveur.
- Sécuriser des services Web RESTful.

Plan

- Présentation de REST
- Motivation pour REST
- Principes de REST
- Développement de services web REST en java

Présentation de REST 1/2

- REST est l'acronyme de **REpresentational State Transfert**

- Principe défini dans la thèse de Roy FIELDING en 2000

  -L'un des principaux auteurs de la spécification HTTP

  -Le développeur du serveur Web Apache

- REST est un **style d'architecture** inspiré de l'architecture du **Web** pour construire des services web

- Les applications respectant les architectures orientées ressources sont nommées **RESTful**

Un style d'architecture est un ensemble de contraintes qui permettent, lorsqu'elles sont appliquées aux composants d'une architecture, d'optimiser certains critères propres au cahier des charges du système à concevoir.

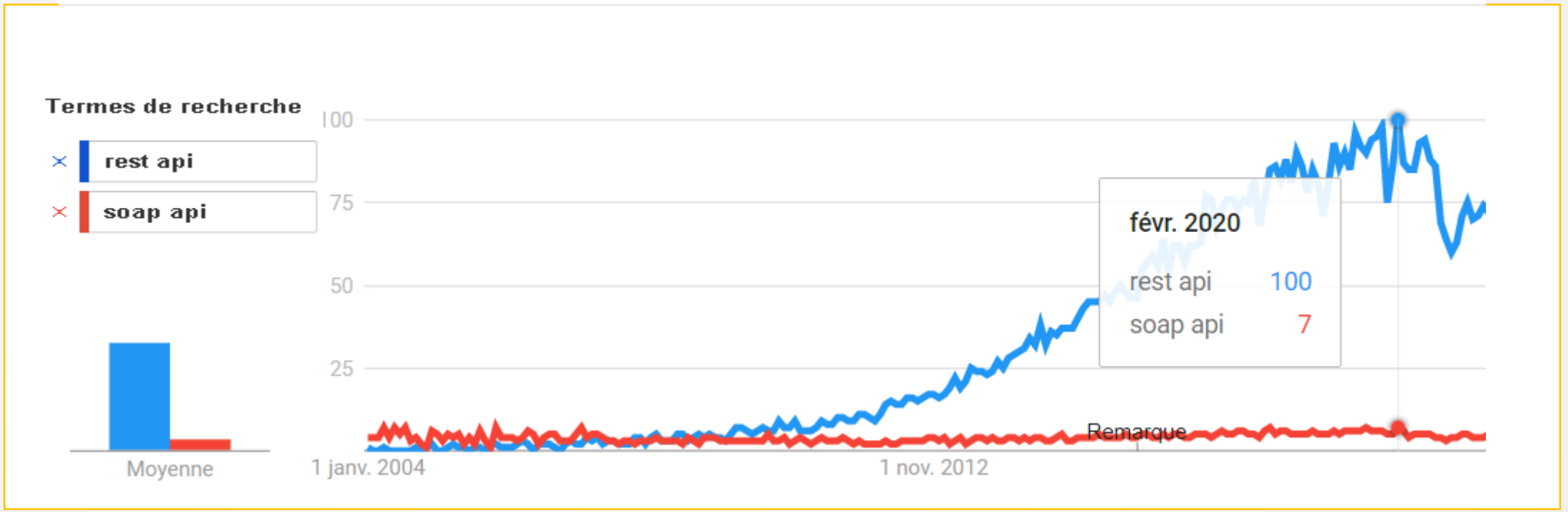
Présentation de REST 2/2

- REST est:
  - un style d'architecture non standardisé
  - une approche pour construire une application
- REST n'est pas:
  - un format
  - un protocole
  - un standard
- Bien que REST ne soit pas un standard, il utilise des standards:
  - HTTP
  - URL
  - XML/HTML

Motivation pour REST 1/2

- REST est une alternative à SOAP

- En 2006, Google a abandonné son API SOAP au profit d'une API simplifiée REST



Source: http://www.google.com/trends/explore?hl=fr#q=rest%20api%2Csoap%20api&cmpt=q

Motivation pour REST 2/2

- REST est léger et simple :
  - Les messages sont courts, faciles à décoder par le navigateur et par le serveur d'application.

- REST est auto-descriptif :

  vous pouvez naviguer à travers ses ressources comme vous le feriez avec une page Web. Il y a une URL intuitive unique pour chaque ressource. On peut facilement en déduire la structure des ressources sans avoir besoin de beaucoup de documentation.

- REST est stateless :
  - Consommation de mémoire inférieure

- REST peut être géré en cache
  - mise en cache possible donc meilleure montée en charge

Principes de REST 1/7

URI

http://weather.com/tunis

Identifie

Ressource

La météo de Tunis

Représente

Représentation

```
Metadata:
Content-type:
application/xhtml+xml
---------------------
Data:
<!DOCTYPE html PUBLIC "...
    "http://www.w3.org/...
<html xmlns="http://www...
<head>
<title>5 Day Forecaste for
Oaxaca</title>
...
</html>
```

Principes de REST 2/7

- Une ressource
- Un identifiant de ressource
- Une représentation de la ressource
- Interagir avec les ressources
  - Requêtes HTTP : GET, POST, PUT et DELETE

Principes de REST 3/7

- **Ressources (Identifiant)**
  - Identifié par une URI

    Exemple : http://localhost:8080/libraryrestwebservice/books

Méthodes (Verbes)

- Pour manipuler la ressource
- Méthodes HTTP : GET, POST, PUT and DELETE

- **Représentation**
  - Donne une vue sur l'état de la ressource
  - Informations transférées entre le client et le serveur

    Exemples : XML, Text, JSON, …

Principes de REST 4/7

❑ Méthodes

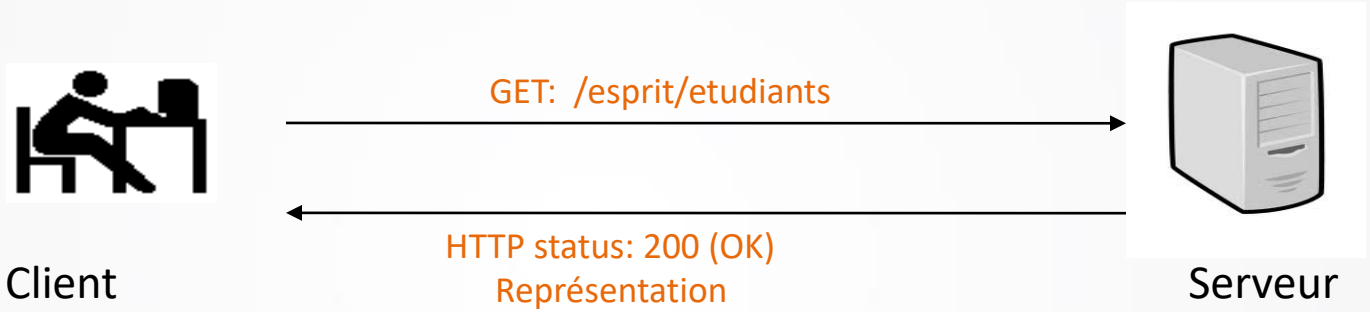
- Une ressource quelconque peut subir quatre **opérations** de base désignées par **CRUD**
  - **C**reate (Créer)
  - **R**etrieve (Lire)
  - **U**pdate (mettre à jour)
  - **D**elete (Supprimer)

- REST s'appuie sur le protocole **HTTP** pour exprimer les opérations via les méthodes HTTP
  - Create ⟷ **POST**
  - Retrieve ⟷ **GET**
  - Update ⟷ **PUT**
  - Delete ⟷ **DELETE**

Principes de REST 5/7

❑ Méthodes

- Méthode **GET** fournit la représentation de la ressource

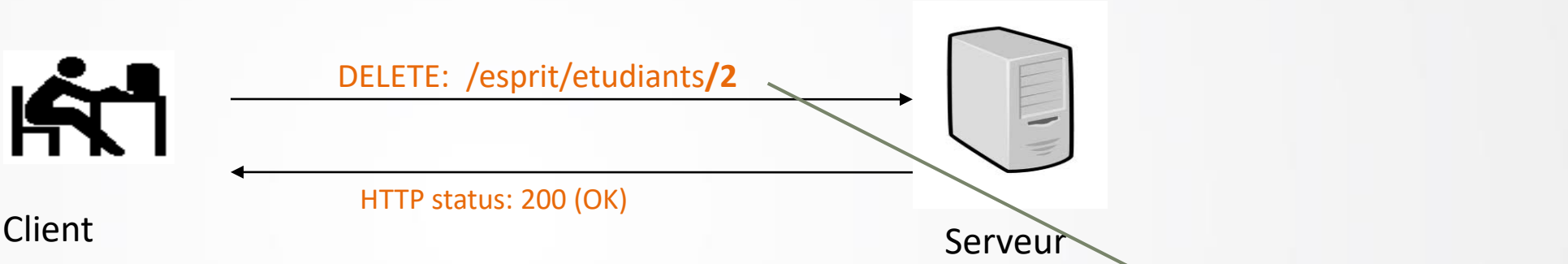


- Méthode **POST** crée une ressource

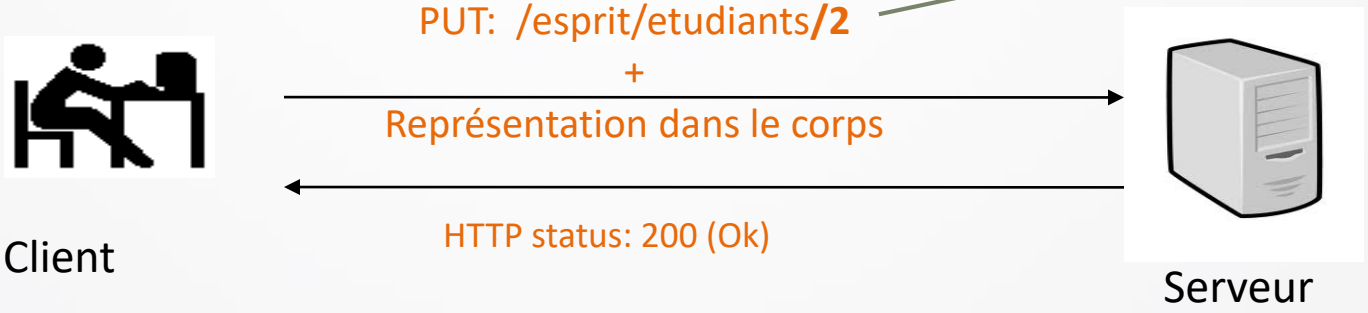
Principes de REST 6/7

❑ Méthodes

- Méthode **DELETE** supprime une ressource



- Méthode **PUT** met à jour une ressource



Principes de REST 7/7

❑ Représentation

Désigne les données échangées entre le client et le serveur pour une ressource:

- le client (GET): format de sortie
- le serveur (PUT et POST): format d'entrée

La représentation d'une ressource peut prendre différents formats:

- XML
- JSON
- Text, HTML
- …

Le format d'entrée (PUT et POST) et le format de sortie (GET) d'un service Web d'une ressource peuvent être différents.

WADL 1/2

- **W**eb **A**pplication **D**escription **L**anguage
- Un langage de description XML de services de type REST
- Une spécification W3C initiée par SUN
- l'objectif est de pouvoir générer automatiquement les APIs clientes d'accès aux services REST

Remarques

-Peu d'outils exploite la description WADL

-Apparu bien plus tard

WADL 2/2

Exemple

```
<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>
```

Services Web REST avec Java

- JAX-RS: **J**ava **A**PI for **R**ESTful Web **S**ervices
- Spécification décrivant la mise en œuvre et la consommation des services web REST
- JAX-RS est basé sur des annotations Java
- Pas de possibilité de développer le service à partir d'un WADL (contrairement à SOAP):
  - Seule **l'approche bottom-up** est possible:
    - Annoter une classe POJO
    - Compiler et déployer

Services Web REST avec Java

- Différentes implémentations de JAX-RS sont disponibles:
  - JERSEY (Oracle)
  - CXF (Apache)
  - RESTEasy (JBoss)
  - RESTlet

Annotations JAX-RS

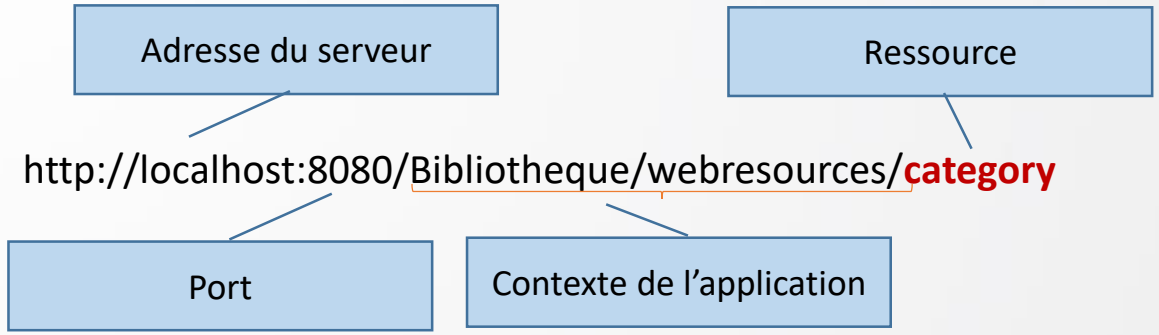
- La spécification JAX-RS dispose d'un ensemble d'annotation permettant **d'exposer** *une classe java sous forme d'un service web* :

@Path	Définit le chemin de la ressource. Cette annotation se place sur la classe et/ou sur la méthode implémentant le service.
@GET, @PUT, @POST, @DELETE	Définit l'action implémentée par le service
@Produces	Spécifie le type de la réponse du service
@Consumes	Spécifie le type accepté en entré du service

Annotation @PATH

- L'annotation @**PATH** permet de rendre une classe accessible par une requête HTTP
- Elle définit la racine des ressources (Root Racine Ressources)
- La valeur donnée correspond à l'URI relative de la ressource

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



Annotation @PATH

- L'annotation 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)

   @Path("helloTest")
   public String hello(){
      return "Hello World!";
   }
..
}
```

http://localhost:8080/Bibliotheque/webresources/**category**/**helloTest**

Annotation @GET, @POST, @PUT, @DELETE

- Permettent de mapper une méthode à un type de requête HTTP
- Ne sont utilisables que sur des méthodes
- Le nom de la méthode n'a pas d'importance, JAX-RS détermine la méthode à exécuter en fonction de la requête

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

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

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

Annotation @GET, @POST, @PUT, @DELETE

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

GET, POST
PUT, DELETE

/books
GET : Liste des livres
POST : Créer un nouveau livre

/books/{id}
GET : Livre identifié par l'id
PUT: Mis à jour du livre identifié par id
DELETE : Supprimer le livre identifié par id

Paramètres des requêtes

- JAX-RS fournit des mécanismes pour extraire des paramètres dans la requête
- Utilisés sur les paramètres des méthodes des ressources pour réaliser des injections de contenu
- Différentes annotations :
  - @PathParam : valeurs dans templates parameters
  - @QueryParam : valeurs des paramètres de la requête
  - @FormParam : Valeurs des paramètres de formulaire
  - @HeaderParam: Valeurs dans l'en tète de la requête
  - @DefaultValue: valeur définie par défaut pour les annotations ci-dessus lorsque la clé est introuvable.
  - @CookieParam : Valeurs des cookies

Annotation @PathParam

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

```
@GET
@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**

Annotation @QueryParam

- L'annotation @QueryParam permet de récupérer des paramètres passés à l'URI côté serveur

```
@GET
@Produces ( MediaType.APPLICATION_XML)

@Path("hello")

public String hello (@QueryParam("from") String exp, @QueryParam("to") String dest){
    return "Hello to" + dest + "from me: " + exp;
}
```

- Accéder à l'URL suivant:

http://localhost:8080/Bibliotheque/webresources/category/**hello?from=Sarra&to=Celine**

Mise en place du projet

- Pour indiquer le chemin de votre application, vous devez commencer par la création d'une sous-classe **javax.ws.rs.core.Application** (classe qui étend la classe Application).
- L'annotation @ApplicationPath va mapper toutes les ressources de votre application si elles existent
- Créer une classe de test, dans notre exemple la classe **CategoryFacade**

```
import javax.ws.rs.core.Application;
import javax.ws.rs.ApplicationPath;

@ApplicationPath("Webressource")
public class CategApplication extends Application {

}
```

```
@Path("category")
public class CategoryFacade {
 @GET
 @Produces(……)
 @Path("test")
 public String hello(){
    return "Hello World!";
 }
}
```

- Lancer l'application avec Tomcat et accéder à l'URL suivant :

http://localhost:8080/Bibliotheque/**Webressource**/category/test

En résumé

- REST est un style d'architecture
- REST est une alternative aux services web étendus (SOAP)
- REST se base sur le protocole HTTP
- JAX-RS est l'API java permettant de développer et consommer des services web REST