



# Atelier SOA -TP08

## Tester un service web REST avec ARC

### Objectifs

Tester un Web Service REST avec ARC (Advanced REST Client)

#### 1. Créer un service Web REST

- Définir un path avec une méthode GET
- Définir un path avec une méthode POST

#### 2. Installer l'outil « Advanced REST Client » de Google Chrome

- Installer ARC
- Lancer une requête http avec ARC avec la méthode « GET »
- Configurer ARC pour envoyer une requête avec la méthode « POST »

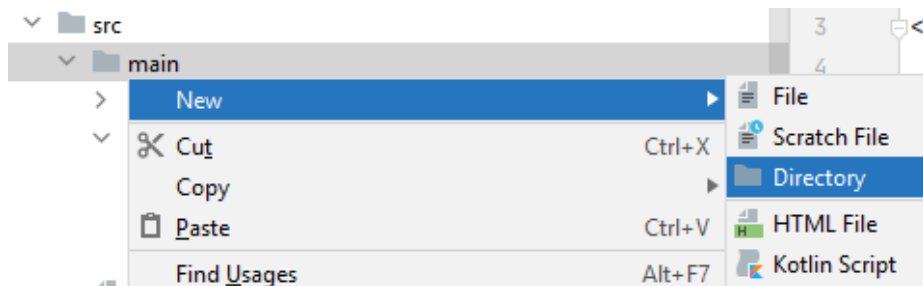
### A. Créer un projet maven de type web

1. Créer un projet **MAVEN** (**maven-archetype-webapp**) dans **IntelliJ IDEA** nommé **jax-rs-arc** (sous le dossier **workspace**) ayant les caractéristiques suivantes :
  - groupId : **org.soa.tp8**
  - artifactId : **jax-rs-arc**
  - version : **1.0-SNAPSHOT**
2. Configurer ce projet pour supporter le développement des services web REST et être déployé avec le plugin «**tomcat7**». Spécifier «**/rest\_arc**» comme nom de contexte de l'application web à déployer par le serveur « **tomcat7** » sur le port «**9999**» (voir atelier\_07).
3. Configurer l'application web pour définir une servlet JERSEY nommée «**servletREST**» qui référence les services web REST du

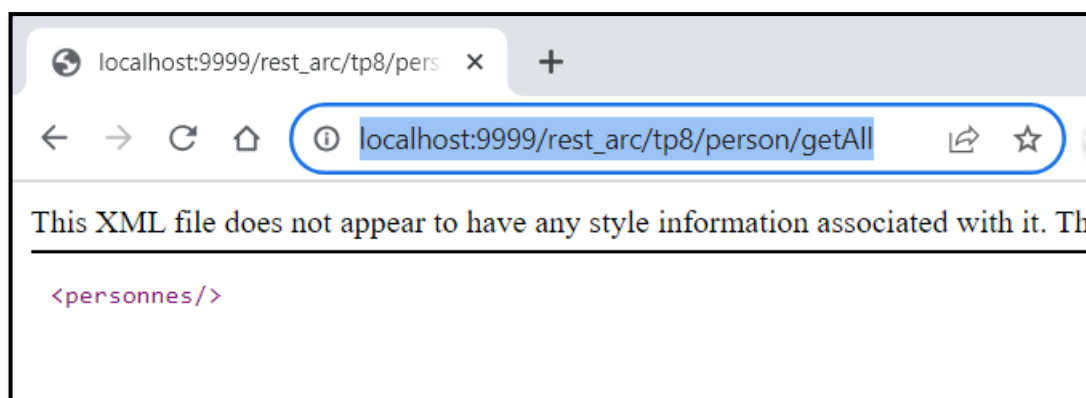
package nommé «**ws.rest.tp8**». La servlet est associée à l'url ayant le motif «**/tp8/\***». (voir atelier\_07)

## B. Développer et déployer le service web

4. Créer un dossier « **java** » sous le dossier « **main** » :



5. Créer un package « **ws.rest.tp8** » sous le dossier « **java** ».
6. Créer le modèle de l'application (classe « **Personne** » dont le code est donné en pièce jointe).
7. La réponse du serveur est représentée dans une classe « **Reponse** » dont le code est donné en pièce jointe.
8. Le service web est déclarée dans une interface « **PersonneService** » et son implémentation est définie dans « **PersonneServiceImpl** » (Le code de l'interface et de son implémentation est donné en pièce jointe).
9. Exécuter le plugin « **tomcat7** » pour déployer le service web.
10. Lancer la requête http qui permet d'afficher toutes les personnes existantes. Pour le moment, aucune personne n'est existante :



11. Lancer la requête http qui permet d'afficher l'objet « **Personne** » de test :



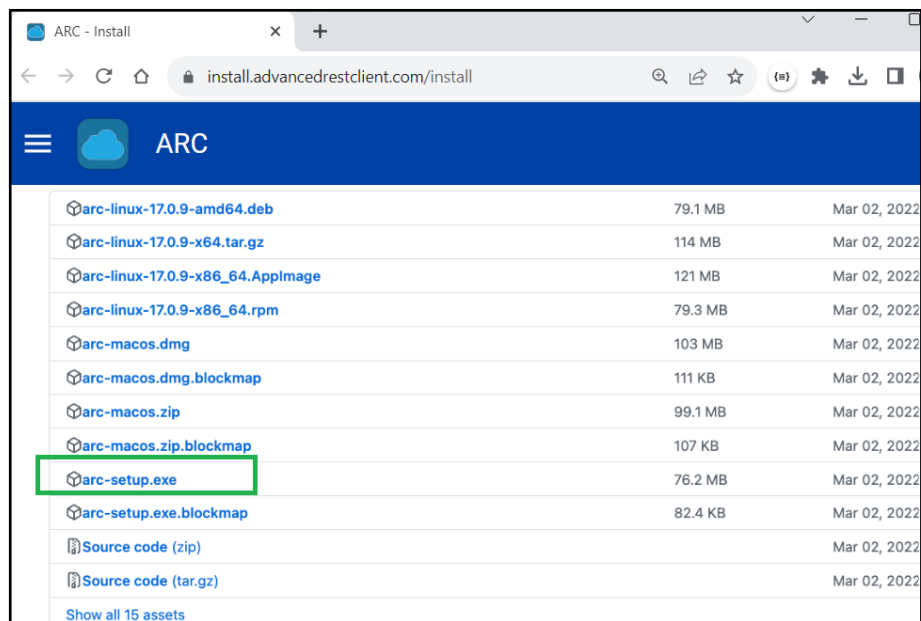
### C. Utiliser l'outil ARC pour tester un service web REST

Nous utilisons l'outil **RESTClient** (version Desktop) pour permettre de personnaliser les requêtes envoyées à un service RESTful. Il aide les programmeurs à développer l'application de test RESTful Service pour leurs services.

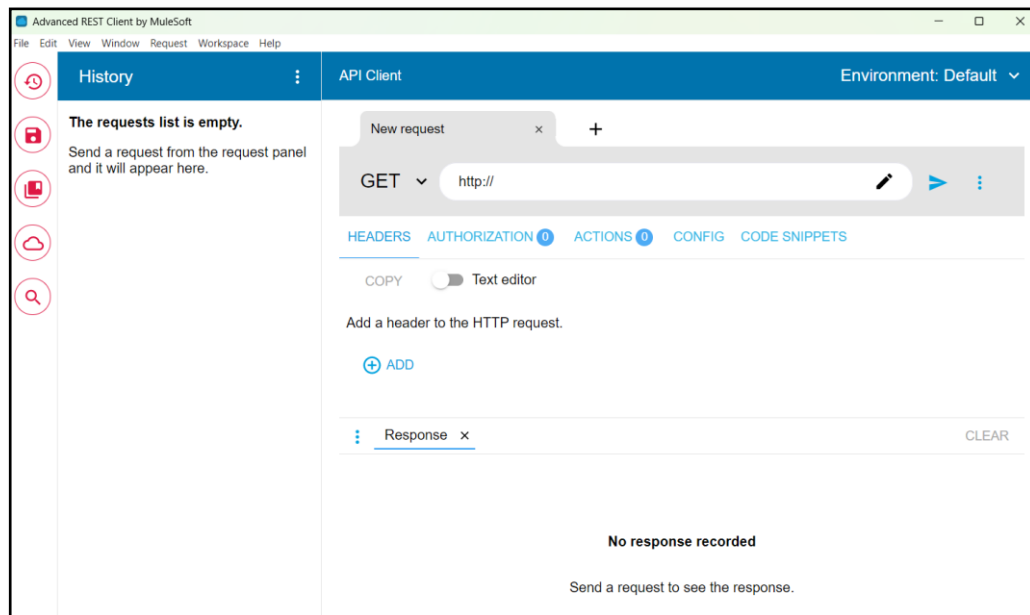
12. Accéder à l'adresse suivante :

<https://github.com/advanced-rest-client/arc-electron/releases>

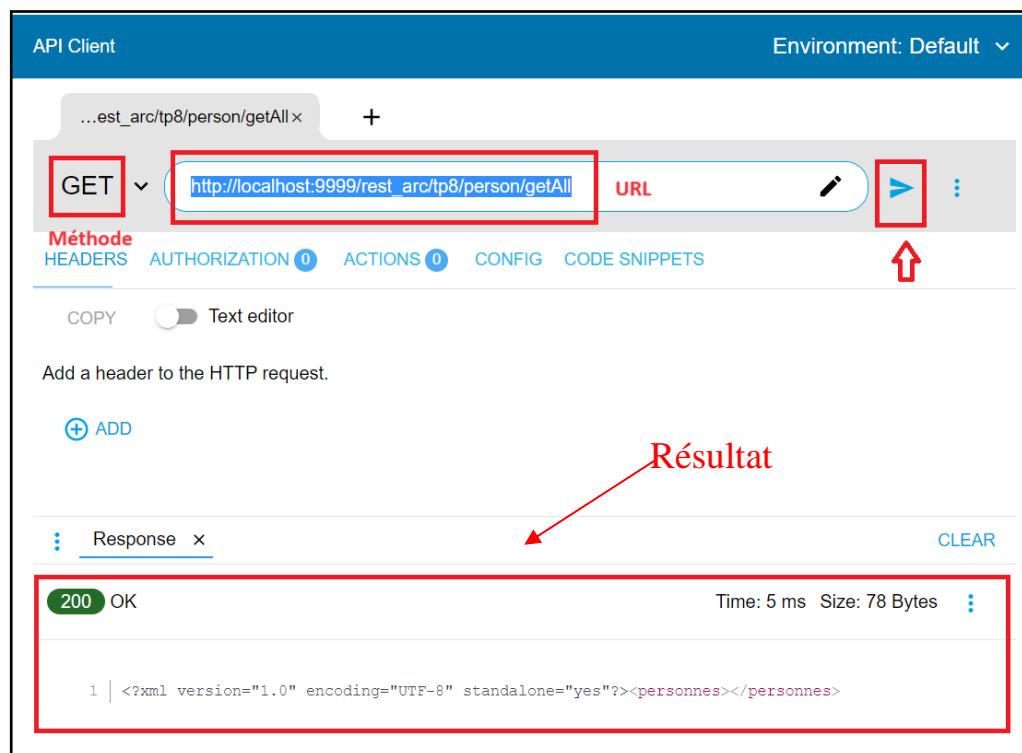
13. Choisir la version Windows :



14. Télécharger l'outil ARC et l'installer pour avoir cette page d'accueil :



15. Path « /getALL »






16. Path « /add » (ajouter une Personne (id= 1 , nom=Ayyadi , age=27))

API Client Environment: Default ▾

---

.../rest\_arc/tp8/person/add x +

**POST** ▾ http://localhost:9999/rest\_arc/tp8/person/add   

HEADERS **BODY** AUTHORIZATION 0 ACTIONS 0 CONFIG CODE SNIPPETS

Raw input ▾ XML ▾

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <person>
3   <age>27</age>
4   <id>1</id>
5   <nom>Ayyadi</nom>
6 </person>

```

⋮ Response x CLEAR

**200** OK Time: 34 ms Size: 146 Bytes ⋮

```

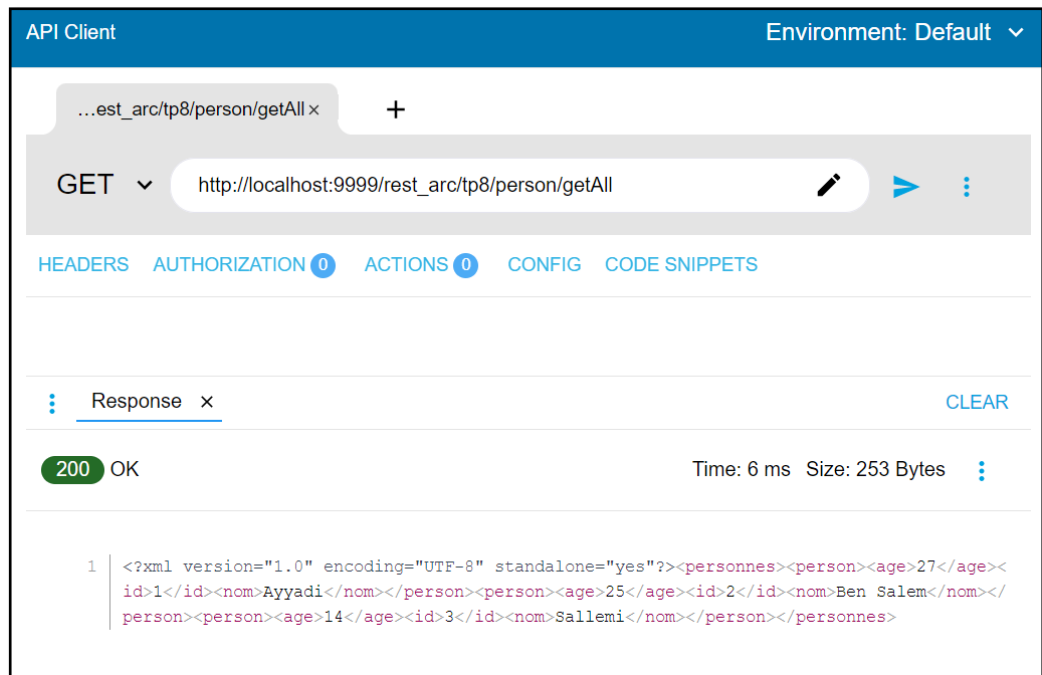
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?><reponse><message>
  Personne créée avec succès...</message><status>true</status></reponse>

```

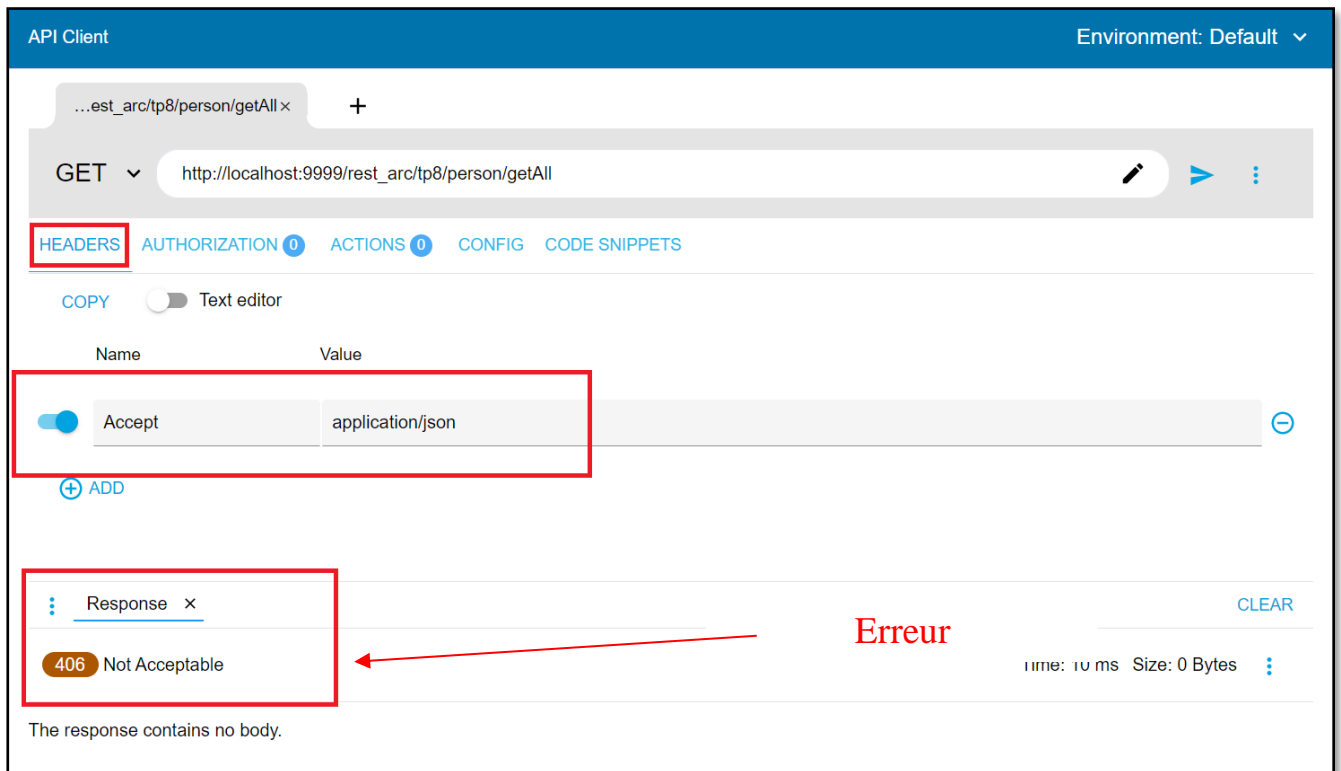
17. Réappeler le path « **add** » pour ajouter deux personnes :

- Personne (2 , « Ben Salem » , 25)
- Personne (3 , « Sallemi » , 14)

18. Réappeler le path « **getAll** » pour afficher toutes les personnes:



19. Ajouter un paramètre de type « **Headers** » nommé « **Accept** » ayant la valeur « **application/json** » (format non fourni selon la définition du service web) et relancer la requête. Remarquer l'affichage du code d'erreur suivant :



20. Pour remédier à ce problème, ajouter dans la définition du service web la possibilité de produire le format « **json** » comme suit :

```
@Produces ( {MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})  
public class PersonneServiceImpl implements PersonneService {
```

Ainsi les méthodes du service web produisent les deux formats (XML et JSON) et c'est au client de spécifier le format à accepter.

- Enregistre et republie le service web puis re-ajoute les trois personnes (questions 16 et 17)
- Relance l'URL « **getAll** » pour afficher la liste des personnes au format JSON :

The screenshot shows the API Client interface with the following details:

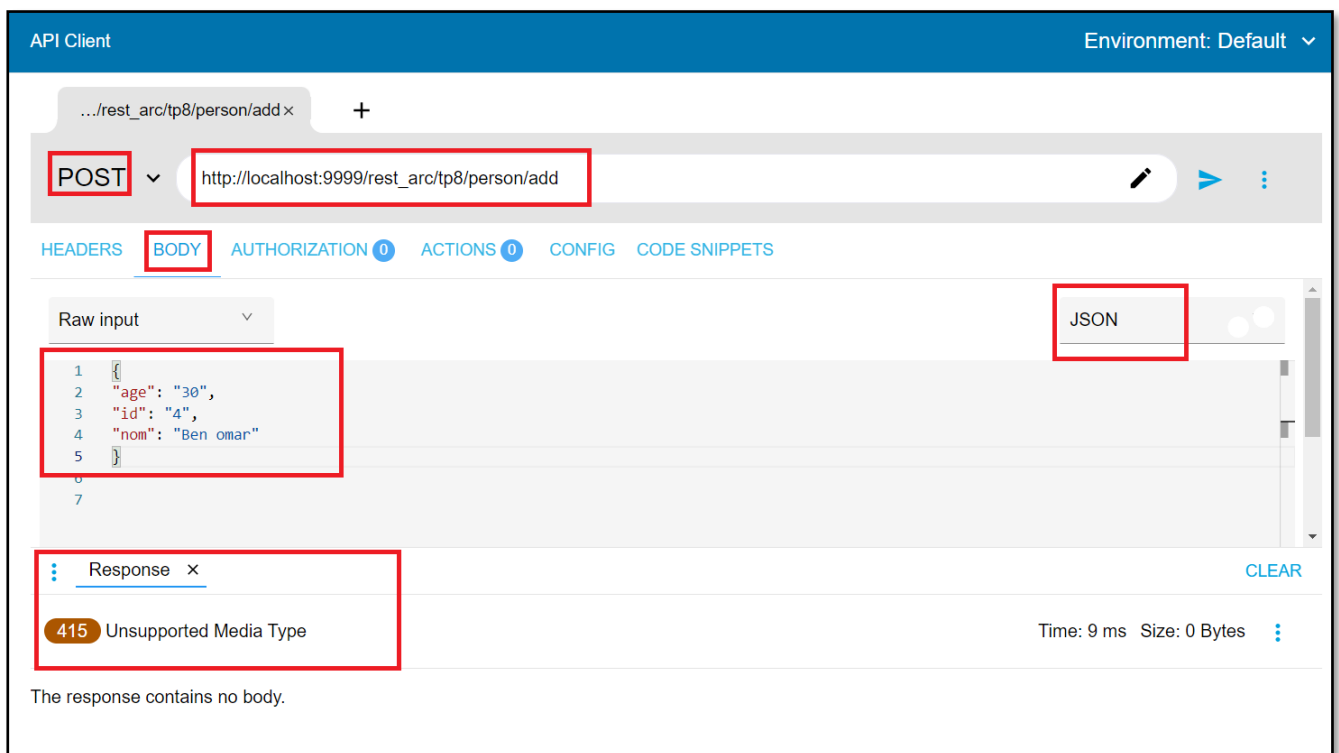
- Environment:** Default
- Request:** GET `http://localhost:9999/rest_arc/tp8/person/getAll`
- Headers:** Accept: `application/json`
- Response:** 200 OK, Time: 12 ms, Size: 127 bytes
- Response Body (JSON):**

```
{  
  "person": [  
    {  
      "age": "27",  
      "id": "1",  
      "nom": "Ayyadi"  
    },  
    {  
      "age": "25",  
      "id": "2",  
      "nom": "Ben Salem"  
    },  
    {  
      "age": "14",  
      "id": "3",  
      "nom": "Sallemmi"  
    }  
  ]  
}
```

21. On veut maintenant ajouter une personne (4, « Ben Omar », 30) au format « **json** » :

```
{  
  
  "age": "30",  
  
  "id": "4",  
  
  "nom": "Ben omar"  
}
```

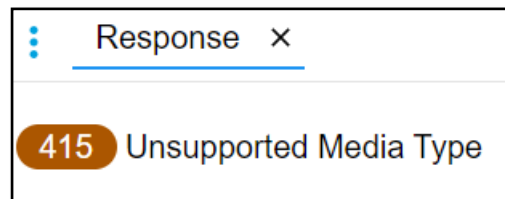
Il est nécessaire de préciser le type d'envoi de la requête (càd spécifier la valeur « **application/json** » au « **content type** ». Pour saisir une personne au format **json**, passer à l'onglet « BODY » et choisir le format JSON:



22. Mais, ceci reste insuffisant pour réussir à ajouter cette nouvelle personne : L'envoi de la requête génère une erreur qui



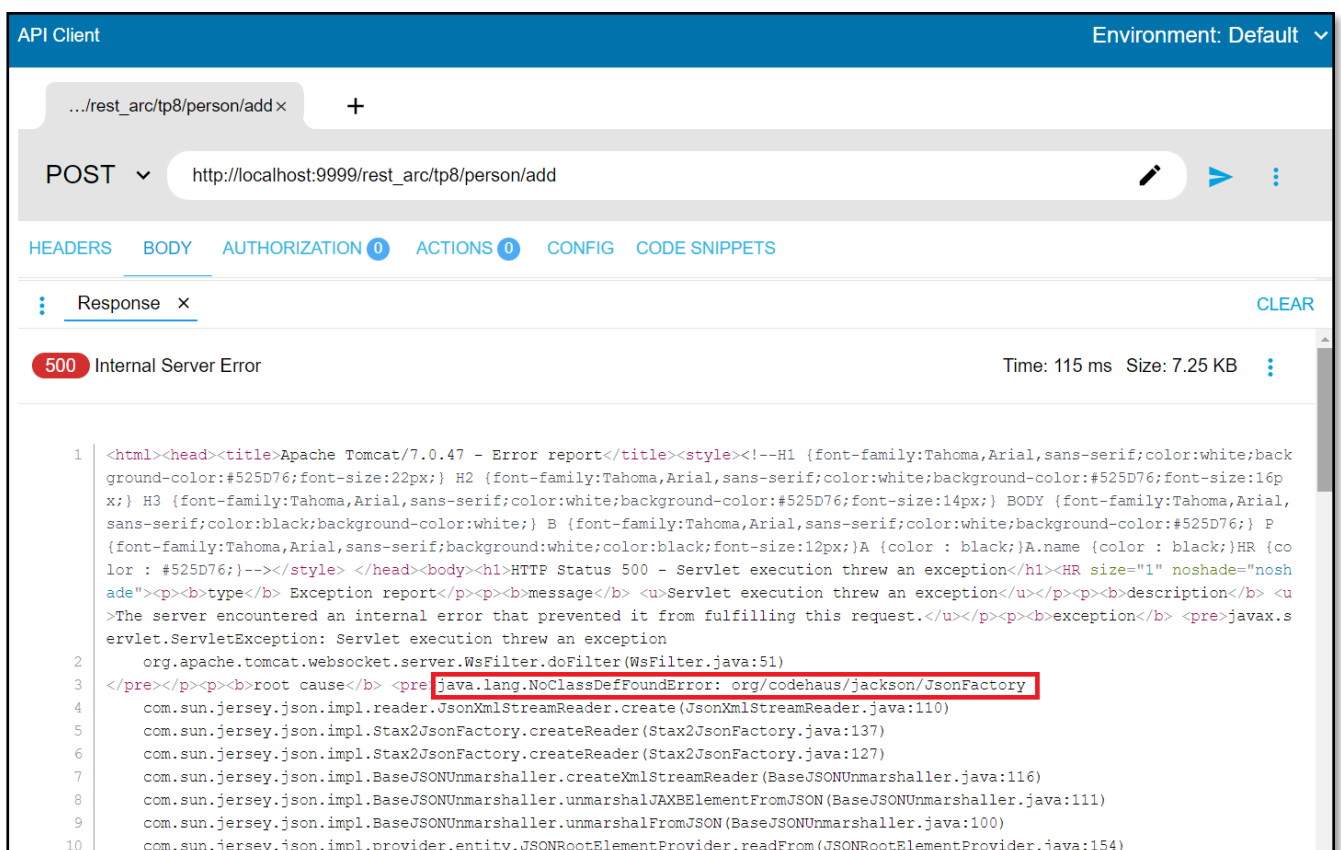
indique que le format « json » n'est pas supporté par le service web en consommation :



23. Il est nécessaire d'ajouter le format « json » dans la consommation de la requête http client par le service web :

```
@Consumes ( {MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
```

24. Après enregistrement et republication du service web, l'envoi de la requête génère une autre erreur :



25. Ceci nécessite l'ajout des deux dépendances « jackson » suivantes dans le fichier « pom.xml » :

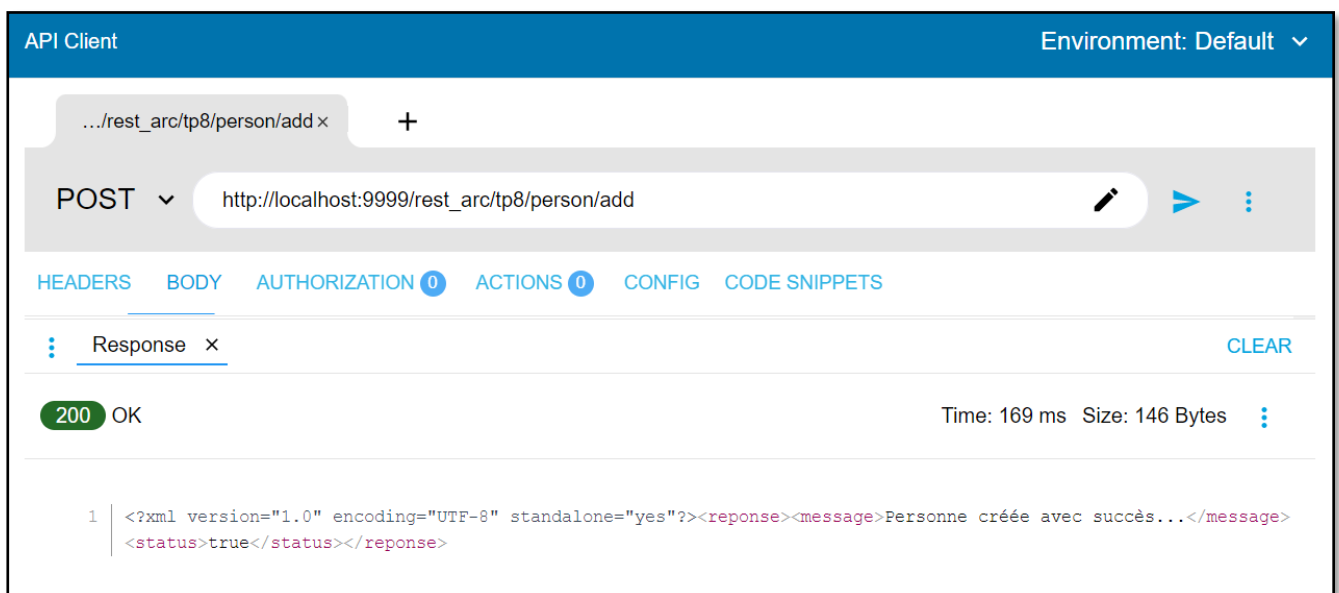
```

<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.6.3</version>
</dependency>

<dependency>
  <groupId>org.codehaus.jackson</groupId>
  <artifactId>jackson-mapper-asl</artifactId>
  <version>1.9.13</version>
</dependency>

```

26. Ré exécuter le serveur « **tomcat7** » et renvoyer la requête d'jour de la nouvelle personne au format « **json** » :



27. Path « **/ {id} /get** »

- Lancer une requête pour récupérer la personne ayant l'id « 1 ».

28. Path « **/ {id} /delete** »

- Lancer une requête pour supprimer la personne ayant l'id « 1 »
- Lancer une requête pour supprimer la personne ayant l'id « 5 ».

29. Path « **/update** »

- Définir une nouvelle méthode « **modifierPersonne** » associée à un path « **/update** » avec la méthode « **PUT** » pour mettre à jour une personne passée en paramètre. Tester l'appel à cette fonctionnalité.

### 30. Path « **/delete** »

- Définir une nouvelle méthode « **supprimerPersonne** » associée à un path « **/delete** » avec la méthode « **DELETE** » pour supprimer une personne passée en paramètre. Tester l'appel à cette fonctionnalité.