

Groupe Quddus Artishokasi

# Rapport de projet Web Service

Politiciens et réseaux sociaux

Courtin Florent & Fleutry Eva  
14/04/2023

## Table des matières

Présentation du projet .....	2
Services web.....	2
Client .....	2
Service et client SOAP .....	3
Service .....	3
Client .....	3
Service et client REST .....	4
Service .....	4
Client .....	5

## Présentation du projet

Dans le cadre de l'unité d'enseignement Web Service, nous réalisons un projet afin de mettre en pratique nos connaissances. Notre sujet avait pour objectif d'ajouter et récupérer des informations concernant des politiciens et leur parti politique.

Nous avons plusieurs critères à respecter :

### Services web

- Créer un service SOAP et un service REST avec pour chaque au moins deux méthodes
- Possibilité de créer une base de données pour récupérer les informations

### Client

- Appeler toutes les méthodes du service créé avec un seul client
- Faire appel à une API pour un des services (REST ou SOAP)

Pour la réalisation de ce projet, nous avons utilisé les frameworks JAX-WS, JAX-RS et Apache CXF et le serveur d'application Apache Tomcat 9.

## Service et client SOAP

### Service

Pour créer le service soap, nous avons utilisé la méthode Bottom-up pour coder le service en Java et ensuite générer le fichier .wsdl. Le package contient donc trois classes Java : la classe Politician pour avoir un objet Politician contenant toutes les informations sur un politique, l'interface PoliticianService, et la classe PoliticianServiceImpl qui implémente PoliticianService, et qui définit toutes les méthodes. Pour stocker les différentes entrées, nous avons utilisé un tableau static de Politician .


De cette manière nous avons créé plusieurs opérations :

- addPolitician : qui permet d'ajouter un politicien dans le tableau à partir d'un objet Politician
- deletePolitician : qui permet de supprimer un politicien du tableau à partir d'un id
- getPolitician : qui récupère un politicien du tableau à partir de son id s'il existe
- getAllPoliticians : qui permet d'avoir la liste de tous les politiciens présents dans le tableau
- getPoliticianParti : pour récupérer le parti politique d'un politicien à partir de son id

Une fois que le service est créé, un fichier wsdl est généré, on a pu donc l'utiliser pour créer le client.

### Client

Grâce à l'IDE Eclipse utilisé pour ce projet, le client PoliticianServiceImpl\_PoliticianServiceImplPort\_Client a pu être généré depuis le fichier .wsdl ce qui a donc permis de tester toutes les méthodes depuis le service ss de type PoliticianServiceImplService et le port récupéré depuis le service ss.



```
<terminated> PoliticianServiceImplPort_Client (4) [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (14 avr. 2023, 17:31:47 - 17:31:49) [pid: 8448]
avr. 14, 2023 5:31:48 PM org.apache.cxf.wsdl.service.factory.ReflectionServiceFactoryBean buildServiceFromWSDL
INFOS: Creating Service (http://www.wsPolitician.com)PoliticianServiceImplService from WSDL: file:C:/Users/evafi/Politicians-soap-ws-fonctionnel/Politicians-so
Invoking addPolitician...
addPolitician.result=false
Beressi has been added succesfully
Invoking addPolitician...
addPolitician.result=false
Bazin has been added succesfully
Invoking deletePolitician...
deletePolitician.result=false
The politician doesn't exist or couldn't have been deleted
Invoking getAllPoliticians...
getAllPoliticians.result=[com.wspolitician.client.Politician@24d09c1, com.wspolitician.client.Politician@54c62d71, com.wspolitician.client.Politician@65045a87]
Invoking getPoliticianParti...
getPoliticianParti.result=4
Invoking getPolitician...
getPolitician.result=com.wspolitician.client.Politician@4b45dcb8
```

Figure 1 Appel de toutes les méthodes du service par le client

## Service et client REST

### Service

Pour implémenter le service REST, nous nous sommes basé sur deux classes de données : Politician et PoliticalParty. Ces classes de données et leurs attributs sont directement inspirés des informations que nous avons pu trouver dans l'API REST que nous utilisons : NosDéputés.fr (documentation de l'API disponible ici : <https://github.com/regardscitoyens/nosdeputes.fr/blob/master/doc/api.md>). Cette API fournit des informations sur les partis politiques en mandat actuellement à l'Assemblée Nationale française ainsi que les députés les représentant.

À l'adresse <https://www.nosdeputes.fr/organismes/groupe/xml>, l'API donne de nombreuses informations sur les partis politiques dont nous avons retenu les plus pertinentes : l'identifiant, le nom et la couleur politique.

À l'adresse <http://www.nosdeputes.fr/<id>/xml> (avec <id> le prénom et nom du député, séparé par des tirets, sans caractères spéciaux), nous avons retenu les informations suivantes : l'identifiant, le prénom, le nom, le sexe, le nom et le numéro du département où il est élu, l'identifiant de parti politique, le métier et l'identifiant Twitter.

Avec ces informations, nous avons conçu une base de données dans l'objectif de l'utiliser dans notre service REST et de conserver les informations saisies, modifiées ou supprimées même lors de l'interruption du service. Pour interagir avec une base de données MySQL en Java, nous avons utilisé la librairie mysql-connector-java (<https://mvnrepository.com/artifact/mysql/mysql-connector-java>).

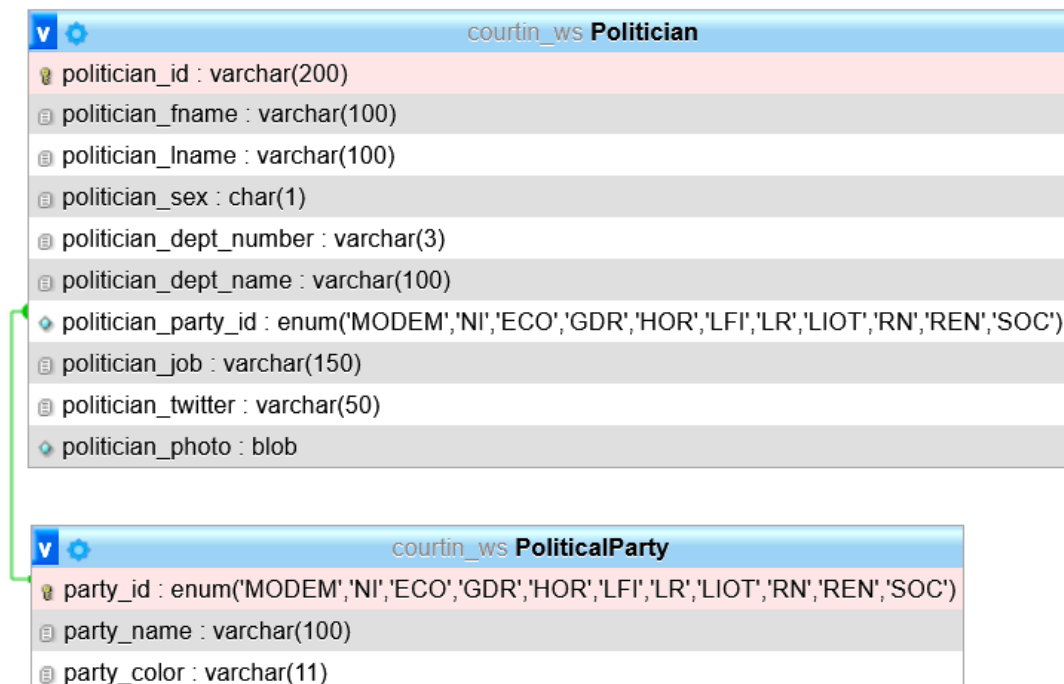


Figure 2: Conception de la base de données

Les classes de données ainsi implémentées, nous avons créé un paquetage `ws.rest.politicians.resouce` constitué de la classe `PoliticalPartyResource` et de la classe `PoliticianResource`. La classe `PoliticalPartyResource` implémente les quatre méthodes suivantes :

- `addPoliticalParty` (accessible depuis `/api/parties`) ajoute un parti politique dans le service à partir de son identifiant, à condition que celui-ci existe bien parmi les onze partis en mandat actuellement à l'Assemblée Nationale.
- `getPoliticalParty` (accessible depuis `/api/parties/{id}`) permet de récupérer les informations d'un parti politique grâce à son identifiant.
- `getAllPoliticalParties` (accessible depuis `/api/parties`) récupère les informations de l'ensemble des partis politiques du service.
- `addInfoPoliticalParty` (accessible depuis `/api/parties/{id}`) permet de mettre à jour les informations du parti politique dans le service grâce aux informations à disposition dans l'API `NosDéputés`.

La classe `PoliticianResource` est constituée des quatre méthodes suivantes :

- `addPolitician` (accessible depuis `/api/politicians`) ajoute un politicien dans le service et implémente tous les attributs disponibles via l'API `NosDéputés`.
- `getPolitician` (accessible depuis `/api/politicians/{id}`) permet de récupérer toutes les informations d'un politicien à partir de son identifiant.
- `getAllPoliticians` (accessible depuis `/api/politicians`) permet de récupérer tous les informations de tous les politiciens qui ont été référencés dans le service. Il est également possible de filtrer les résultats en indiquant un paramètre optionnel « `party_id` » dans l'URI qui récupère ainsi tous les politiciens qui sont issus du même groupe politique (par exemple `/api/politicians?party_id=LR` récupérera tous les députés du groupe « Les Républicains »).
- `deletePolitician` (accessible depuis `/api/politicians/{id}`) supprime un politicien du service.

## Client

Dans un nouveau projet Java classique nommé `ws.rest.client`, nous avons simplement implémenter des fonctions utilisables pour un client en ligne de commande depuis Eclipse. De la même manière, nous avons donc créé quatre méthodes en lien avec les parties politique et quatre méthodes en lien avec les politiciens. Ces huit méthodes se basent donc sur l'URI du service suivant : <http://localhost:8080/ws.rest.politicians/api/> et construisent ainsi les URI complètes énoncées dans la section précédente. Les méthodes de type GET retournent des objets `Politician` ou `PoliticalParty`, ou des listes de ces objets, tandis que les autres méthodes retournent des objets `Response`. Les tests effectués dans le client sont implémentés de tel sorte à connaître le résultat de la requête selon son code (200 : OK, 201 : donnée insérée, 404 : ressource introuvable).