Programmation répartie. Module 4102C TP 5 : l'aboutissement de deux années de programmation objet : discord, la météo et les chats. Année universitaire 2018-2019

Samuel Delepoulle et Frank Vandewiele

15 mars 2020

1 Introduction

On peut définir un service web (ou web service) comme « un protocole d'interface informatique de la famille des technologies web permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.» [1].

Concrètement, une application va proposer des services (traitements, accès à des données) et ces services seront utilisables en utilisant les protocoles du web (par exemple HTTP).

Il existe plusieurs technologies dont le but est d'implémenter les services web :

REST (representational state transfer) dans ce cas, on utilise les protocoles web existants pour la communication entre les serveurs web et leurs clients : un service est identifié par une URI, accessible via HTTP. Une représentation de la ressource est renvoyée au client (fichier HTML, XML, CSV, JSON, ...).

les services WS-* dans ce cas, des protocoles particuliers sont utilisés pour la communication entre les serveurs et les consommateurs en particulier, on peut utiliser :

1. `https://fr.wikipedia.org/wiki/Service_web`

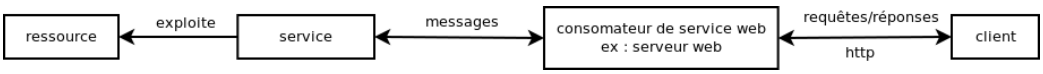

FIGURE 1 – principe de fonctionnement d'un service web.

— SOAP (Simple Object Access Protocol)
— WSDL (Web Service Description Language)

Dans ce TP nous allons utiliser des web services REST pour programmer des bots discord.

2 Accès via un wrapper

Pour accéder à l'API REST de discord, nous allons utiliser JDA qui est un Wrapper Java (`https://github.com/DV8FromTheWorld/JDA`).

Voici les étapes à réaliser pour obtenir un bot minimal :

1. Téléchargez la dernière version du jar
   `https://github.com/DV8FromTheWorld/JDA/releases`
   prendre la version du jar "withDependencies".
2. suivez les étape détaillées dans le wiki de JDA : `https://github.com/DV8FromTheWorld/JDA/wiki/3%29-Getting-Started`
3. Compilez la classe `Bot`. Pour cela :

   en ligne de commande ajoutez le chemin de du fichier jar :
   `javac -cp .:JDA-4.1.1_101-withDependencies.jar MainRunner`
   (ou redéfinir la variable d'environnement `CLASSPATH`) ;

   depuis un IDE (Eclipse, Netbean, ...) ajoutez le fichier jar à votre projet.

   Vérifiez que le programme s'exécute :
   `java -cp .:JDA-4.1.1_101-withDependencies.jar Bot`
4. Le programme du bot est fonctionnel mais il faut maintenant créer un compte pour votre bot, pour cela suivez la procédure précisée dans le wiki. Vous donnerez à votre bot le nom "bot_VotreNom" et vous l'inscirez au serveur `M4102C` `https://discord.gg/2q5uTWD` (inscrivez vous au serveur pour y avoir les droits d'administration).

5. Exécutez votre bot pour l'activer :
   `java -cp .:JDA-4.1.1_101-withDependencies.jar MainRunner XXXXX`
   avec `XXXXX` votre token.
6. Définissez un "BOT_PREFIX = "/nn" avec `nn` un nombre qui vous sera donné par l'enseignant. Votre bot ne devra répondre que lorsque le le message commence par ce préfixe.
7. implémentez la commande `/nn ping`, votre bot devra répondre pong (référez vous à la documentation en ligne de JDA pour comprendre comment les évenements sont gérés).
8. implémentez la commande `/nn dice max` qui donnera le résultat d'un lancé de dé de `max` faces (sans précision, le dé aura 6 faces).

3 Bot indispensable : générateur de mèmes avec des chats

Pour commencer, vous allez intégrer le service `CATAAS` : Cat as a service. Ce service qui expose une API REST permet d'obtenir des images de chat (aléatoires, suivant des tags et avec message personnalisé).

Travail à réaliser : alimentez votre bot avec l'API CATAAS.

4 Bot météo : traiter une réponse JSON

De nombreux autres services sont utilisables via des web services. Nous allons utiliser les services de `https://openweathermap.org` pour afficher des informations relatives à la météo. Pour cela, vous devez avoir un compte sur ce service (le compte gratuit autorise au maximum 60 requêtes par minutes ce qui suffira pour notre bot.). Une fois enregistré vous pouvez récupérer votre ID.

Voici comment lire les informations depuis le service web. Le format de retour est un fichier JSON. Idéalement on devrait utiliser une bibliothèque pour le parcourir mais pour commencer nous allons traiter la réponse comme une chaîne de caractères (que vous devrez analyser) :

```
static String getMeteo(){

    String result = "";
```

```
    try {

        // création de l'URL
        String APIkey = "XXXXX";     // XXXXX = votre id
        String serv = "http://api.openweathermap.org/data/2.5/
    weather";
        String param = "q=Calais,fr&appid=";
        URL url = new URL(serv+"?"+param+APIkey);

        // préparation de la connexion
        HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");

        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Failed : HTTP error code
    : "
                    + conn.getResponseCode());
        }

        BufferedReader br = new BufferedReader(new
    InputStreamReader(
            (conn.getInputStream())));

        result = br.lines().collect(Collectors.joining());
        conn.disconnect();

      } catch (MalformedURLException e) {

        e.printStackTrace();

      } catch (IOException e) {

        e.printStackTrace();

      }

    return result;
}
```

FIGURE 2 – résultat de la commande /nn meteo.

5 Travail à réaliser

1. Afficher la sortie du fichier JSON lorsqu'on demande la météo au bot (`/nn meteo`).
2. Documentez vous sur l'API json-simple : `https://code.google.com/archive/p/json-simple/`.
3. utilisez cette API pour extraire les informations à afficher