Interopérabilité – Mini projet

Réaliser une application qui va communiquer avec votre chatbot Telegram d'assistance personnelle. L'application devra assurer les fonctionnalités décrites ci-dessous. Vous avez 1,5 séances pour réaliser ce mini projet, la 2ème moitié de la 2ème séance (1h30) sera consacrée à une mini présentation de votre application (10-15 min max). Avant de commencer récupérez le projet nommé « chatbot-api-ensim » sur UMTICE.

Votre application va exposer une API afin de communiquer avec votre chatbot Telegram, votre API exposera les fonctionnalités suivantes :

- Envoyer un message sur une conversation Telegram
- Envoyer des informations météo
- Envoyer une histoire drôle au hasard (parmi une liste 5 histoires par exemple)

Description de l'API exposée

1. Envoyer un message sur une conversation Telegram

Une opération « message » qui permet d'envoyer un message sur une conversation Telegram en précisant le « chatId » de la conversation (API SendMessage de Telegram) : https://core.telegram.org/bots/api#sendmessage). Dans l'application de départ qui vous ai fourni, les classes objets Telegram vous sont déjà fournis (par exemple la classe Message qui vous sera certainement très utile). Pour récupérer le chatId, envoyer un message à votre bot et voir l'opération « getUpdates » pour récupérer les derniers événements passés ou se connecter sur Telegram via votre navigateur web, l'id de la conversation apparait dans l'url.

2. Récupérer des informations météo

L'opération sur la ressource « weather » aura en paramètre d'entrée une ville (String) et renverra la météo du jour (soleil, pluie, nuageux, etc) et la température. Une option devra permettre de récupérer en plus de la météo du jour, la météo des deux prochains jours (voir « 5 Day / 3 Hour Forecast» pour l'API OpenWeatherMap (https://openweathermap.org/forecast5).

3. Envoyer une histoire drôle au hasard

L'opération sur la ressource « joke » renverra une blague au hasard parmi la liste disponible. Une « joke » aura comme atribut un id, un titre, un texte ainsi qu'une note sur 10.

Vous devez écrire l'OAS exposant ces 3 opérations et qui sera à présenter lors de la démo.

Intéraction avec le chatbot

Avant toute chose vous devez créer votre bot, suivre les étapes décrites ici : https://core.telegram.org/bots.

Maintenant que votre bot est créé et que votre API sait réaliser les fonctionnalités ci-dessus, allons un peu plus loin pour rendre notre chatbot « vivant ». Votre application devra faire parler votre chatbot en fonction du message qu'il aura reçu :

- Si on saisit le mot « meteo » dans votre conversation avec votre chatbot, votre application devra renvoyer la météo du jour
- Si on saisit le mot « blague » dans votre conversation avec votre chatbot, votre application devra renvoyer une blague au hasard

Comment faire ?

Votre API n'étant pas exposée sur internet, il n'est pas possible de mettre en place de Webhook pour réceptionner les événements qu'ils se passent sur votre chatbot, nous allons alors utiliser la technique du polling pour capter ce qu'il se passe sur votre chatbot.

L'API Telegram mets à disposition l'opération « getUpdates » qui va vous renvoyer tous les événements passés sur votre chatbot sur les dernières 24 heures, voir la doc ici : https://core.telegram.org/bots/api#getting-updates

Vous devez capter les événements de type « message » et réagir en fonction. Attention, au redémarrage de votre application, vous allez certainement reconsommer des événements que vous avez déjà traités, pour éviter ça il faut prévenir Telegram des événements que vous avez déjà consommés, voir le paramètre « offset » de l'opération « getUpdates » qui vous sera d'une grande aide.

Votre projet contient par défaut un grand nombre de classes représentant les objets « Telegram » qui vous feront gagner beaucoup de temps pour utiliser l'API Bot Telegram (voir principalement les classes Message, ApiResponseTelegram et ApiResponseUpdateTelegram).

Mise en place du polling

Votre application contient une classe nommée « ListenerUpdateTelegram », vous placerez ici votre code effectuant le polling. Pour répéter une tâche à une fréquence donnée, la classe « Timer » fera votre bonheur (voir ici : https://www.baeldung.com/java-timer-and-timertask).

Attention à bien mettre à jour l'offset pour ne pas consommer plusieurs fois le même événement et passer au suivant.

Pour aller plus loin (et obtenir une meilleure note !)

Vous trouverez ci-dessous une liste non exhaustive de fonctionnalité que vous pouvez ajouter à votre application :

- Répondre à un message particulier en le citant dans la conversation (cela peut être utile si vous envoyez plusieurs messages à votre chatbot et que le polling n'a pas eu le temps de les traiter au fur et à mesure), voir l'attribut « reply_to_message_id » de l'objet Message.
- Pouvoir manipuler l'API Joke depuis le chatbot
    - On doit pouvoir ajouter, consulter (avec un id ou un titre), modifier ou une supprimer une blague
- Demander au chatbot une blague nulle (note faible) ou une très bonne blague (très bonne note)
- Pouvoir demander depuis le chatbot la météo des prochains jours
- Auto générer vos classes depuis votre OAS avec un plugin maven
- Bien gérer les erreurs qui peuvent survenir dans votre application (par exemple, que se passe-t-il si on envoie un message sur un chatId qui n'existe pas, que renvoie votre API ?).
- Ajouter de la sécurité sur votre API avec un token d'accès (faire très simple)
- Pour aller encore plus loin : demander le synopsis d'un film et/ou d'autres infos comme la note du film en utilisant la célèbre API IMDB : https://imdb-api.com/)

Structure du projet

Vous démarrez avec projet à « moitié structuré », à vous de le développer de la plus belle des manières, voici quelques infos sur le projet :

- Package client :
    - Vous trouverez ici comme sur les autres TP un main pour tester des appels depuis votre application, théoriquement cette classe ne vous servira que de test. Il est plus facile de tester des bouts de codes isolés de cette manière plutôt que réexécuter tout votre code en déclenchant des appels via Postman.
- Package controller
    - Théoriquement vos controller (méthodes qui exposent vos opérations d'API) se trouveront ici
    - Vous trouverez le controller « MessageRestController » qui peut contenir les opérations concernant la ressource « Message » et notamment les appels à l'API Telegram (d'autres controller peuvent être nécessaires)
    - Vous trouverez également un exemple pour pouvoir récupérer des properties depuis le fichier « application.properties » (plus d'infos ici : https://www.baeldung.com/properties-with-spring).
- Package model

    - Un sous package « Telegram » qui contient les objets pour manipuler l'API Telegram. Elle n'est pas exhaustive mais devrait suffire pour les fonctions minimales.
    - Ce package contiendra certainement les autres modèles de classes pour vos différentes opérations (Question/Réponse) ainsi que les classes pour l'API OpenWeatherMap.
- Ressources
    - Vous trouverez ici le fichier « application.properties » qui contient déjà certaines propriétés pour les API OpenWeatherMap et Telegram. Rien ne vous oblige à l'utiliser, vous pouvez saisir les url en dur dans votre code mais cela le rendra plus élégant et plus structuré.

Critères de notation

La note finale prendra en compte les éléments suivants :

- Les fonctionnalités prises en charge par votre application et qui évidemment fonctionnent lors de la démo
- La lisibilité, la structuration et la qualité de votre code (nommage, structure, indentation, etc.)
- La qualité de votre présentation
- La qualité de votre OAS (respecte-t-il les bonnes pratiques REST ?)
- Les réponses apportées suites aux différentes questions qui pourraient survenir lors de votre démo
- Le nombre de fonctionnalités supplémentaires apportées et fonctionnelles

À la suite de votre démo, vous devrez **m'envoyer votre code source zipé** à l'adresse mail girault.johan@gmail.com (en me précisant vos noms/prénoms). Aucun retard ne sera accepté ! Votre zip ne devra **contenir que le dossier « src » et le fichier « pom.xml »**, rien de plus !