

Guide d'utilisation

Présentation

Dap est une application qui est divisée en deux :

- Un serveur permet d'accéder aux API de Google et Microsoft.
- Un client vous permet de visualiser les différentes informations.

Prérequis

Afin d'utiliser cette application, vous avez besoin d'un accès Internet et d'un navigateur Internet, d'un compte Google et Microsoft et d'avoir Java installé.

Configuration

Vous trouverez en fin de documentation des informations concernant la configuration.

Base de données

Afin d'utiliser l'application serveur, vous devez configurer une base de données. Nous vous recommandons une base de données MySQL.

Modifiez le fichier « application.properties » à votre convenance. Voici un exemple de configuration par défaut :




```
spring.datasource.url=jdbc:mysql://localhost:3306/dap?useSSL=false
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.jpa.database=MYSQL
spring.jpa.show-sql=true
spring.jpa.generate-ddl=true
```

La base de données **MySQL** doit être ouverte sur le port **3306**, avec une base nommée « **dap** ». Le nom d'utilisateur est « **root** » et il n'y a pas de mot de passe.

Vous pouvez charger un autre fichier de configuration *application.properties* en démarrant le serveur en ligne de commande et en ajoutant le paramètre « --spring.config.location=classpath:/another-location/application.properties ». Vous pouvez également mettre le fichier « application.properties » à côté du jar (./), ou bien dans un dossier « config » à côté du jar (./config/).

Configuration

Mettez les différents fichiers nécessaires pour le démarrage de l'application dans un répertoire « dap » situé dans votre répertoire d'utilisateur (cf image ci-contre) :

Ce PC > OS (C:) > Utilisateurs > adrij > dap				
Nom	Modifié le	Type	Taille	
 tokens	28/11/2018 01:59	Dossier de fichiers		
 auth.properties	28/11/2018 01:48	Fichier PROPERTIES	1 Ko	
 credentials.json	18/10/2018 09:36	JSON File	1 Ko	

Credential.json

Placez le fichier credential.json obtenu avec votre compte Google.

Auth.properties

Vous devez remplir ce fichier avec la configuration suivante :

appId=YourID

appPassword=YourPassword

redirectUrl=http://localhost:8080/authorize

Dossier tokens

Créez le répertoire « tokens ». Le fichier StoreCredential de Google sera conservé à l'intérieur.

Serveur – Démarrer le serveur

Démarrez le serveur par exemple avec Powershell en exécutant

« java -jar .\jalabert_adrien_dap_client.jar »

Par défaut, le serveur est démarré sur localhost sur le port 8080.



Veillez à ce qu'aucun autre programme utilise déjà le port 8080.

Serveur – Utilisation du serveur.

Enregistrer des comptes

Tout d'abord, vous devez ajouter un compte. Pour ce faire, accédez à la route [user/add/{NomDuCompte}](#)

Par exemple, pour créer un compte « user » : <http://localhost:8080/user/add/user>

Un message vous confirmera la création du compte, ainsi que le code HTTP **Code d'état : 201**.

Désormais, il s'agit d'ajouter des comptes Google et Microsoft. Pour ce faire, accédez aux routes suivantes :

`/account/add/google/{NomDuCompteGoogle}?userKey={NomDuCompte}`

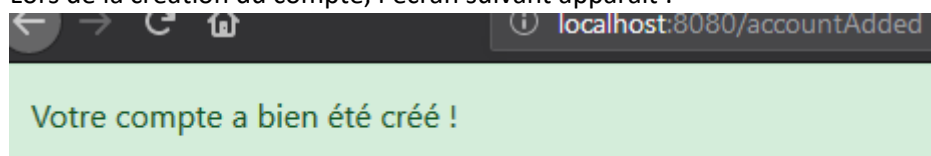
`/account/add/microsoft/{NomDuCompteMicrosoft}?userKey={NomDuCompte}`

Par exemple, pour ajouter un compte Microsoft nommé « ynov » et le lier au compte de l'application « user » précédemment créé :

<http://localhost:8080/account/add/microsoft/ynov?userKey=user>

Vous serez amené à vous authentifier que votre compte Google ou Microsoft. Veuillez accepter les permissions.

Lors de la création du compte, l'écran suivant apparaît :



Interface d'administration

Si vous souhaitez vérifier les comptes Google et Microsoft ajoutés à votre compte de l'application, accédez à la route : `/admin ?userKey={NomDuCompte}`

Par exemple pour accéder aux informations admin de notre compte créé précédemment :

<http://localhost:8080/admin?userKey=user>

Interface d'affichage des mails Microsoft

Si vous souhaitez consulter les mails de vos compte Microsoft liés à un compte de l'application, accédez à la route : `/microsoftMails?userKey={NomDuCompte}`

Par exemple : <http://localhost:8080/microsoftMails?userKey=user>

API

Voici les différentes routes disponibles de l'API :

- <http://localhost:8080/email/nbUnread?userKey=user&user=me> : permet de récupérer le nombre de mails non lus sur l'ensemble des comptes Google et Microsoft d'un compte de l'application.

- <http://localhost:8080/people/number?userKey=user> : permet de récupérer le nombre de contacts que vous possédez sur l'ensemble des comptes Google et Microsoft d'un compte de l'application.
- <http://localhost:8080/calendar/event/next?userKey=user> : permet de récupérer l'événement le plus proche sur l'ensemble des comptes Google et Microsoft d'un compte de l'application.

Client – Utilisation du client

Pendant que le serveur est actif, nous pouvons utiliser le client.

Démarrez le client par exemple avec Powershell en exécutant

« `java -jar .\jalabert_adrien_dap_server.jar` »

Écrivez « `java -jar .\dap-1.0.0-SNAPSHOT.jar help` » pour obtenir l'aide.

Enregistrer un nouvel utilisateur

Entrez `java -jar .\dap-1.0.0-SNAPSHOT.jar add adrien`

Afin de créer l'utilisateur « adrien ».

Ajouter un compte Google

Entrez

`java -jar .\client.jar addgoogle user googleaccountname`

Afin de créer un compte Google nommé « googleaccountname » lié à un compte utilisateur nommé « user ».

Le navigateur va s'ouvrir, vous serez invité à vous authentifier à votre compte google et à valider les permissions.

Ajouter un compte microsoft

Entrez

`java -jar .\client.jar addmicrosoft user microsoftaccountname`

Afin de créer un compte Google nommé « microsoftaccountname » lié à un compte utilisateur nommé « user ».

Le navigateur va s'ouvrir, vous serez invité à vous authentifier à votre compte google et à valider les permissions.

Voir le nombre de mails non lus

Entrez `java -jar .\dap-1.0.0-SNAPSHOT.jar unread adrien`

Vous verrez apparaître le nombre de mail :

```
Nombre de mails :  
43
```

Note : vous pouvez aussi préciser le user avec le service gmail (par défaut « me ») en écrivant :
« unread adrien me ».

Voir le nombre de contact que vous possédez

Entrez `java -jar .\dap-1.0.0-SNAPSHOT.jar contacts adrien`

Vous verrez apparaître votre nombre de contacts :

```
Nombre de contacts :  
1
```

Voir votre prochain événement

Entrez `java -jar .\dap-1.0.0-SNAPSHOT.jar event adrien`

Vous verrez apparaître votre prochain événement :

```
Sujet : Mamamia  
Date de début : Fri Oct 19 16:00:00 CEST 2018  
Date de fin : Fri Oct 19 17:00:00 CEST 2018  
Événement : peut-être  
Vous êtes l'organisateur : oui
```

Voici les différentes informations que vous retrouverez :

- Le nom de l'événement (sujet)
- La date de début
- La date de fin
- Si vous avez accepté, refusé l'événement. Ou si vous allez « peut-être » y aller.

Informations annexes

Si vous recevez cette erreur :

```
Erreur : 401
```

Cela signifie que le userKey que vous avez fourni est invalide. Essayez de créer un nouvel utilisateur.

Vous pouvez aussi changer l'adresse et le port du serveur auquel se connecter. Pour ce faire, précisez l'adresse en dernier argument.

Par exemple : « java -jar .\dapclient.jar event adrien http://localhost:8080 »

« java -jar .\dapclient.jar unread adrien http://localhost:8080 »

Informations annexes concernant la configuration

J'ai essayé de configurer la config externe. Je n'ai pas pu tester la configuration, donc je n'ai pas souhaité l'intégrer. Voici cependant la démarche que j'ai suivi :

```
Config.java  application.properties  AuthHelper.jav
1 spring.datasource.url=jdbc:mysql://localhost:3306/
2 spring.datasource.username=root
3 spring.datasource.password=
4 spring.datasource.driverClassName = com.mysql.jdbc
5 spring.jpa.database=MYSQL
6 spring.jpa.show-sql=true
7 spring.jpa.generate-ddl=true
8
9 config.defaultRootDir=C:\Users\adrij\dap\lol
10
```

Ajouter les propriétés dans le fichier application.properties

```
Config.java  application.pro
1 package fr.ynov.dap;
2
3 import java.io.File;
4
5 import org.springframework
6 import org.springframework
7
8 /**
9  * Config class.
10 *
11 */
12 @Configuration
13 public class Config {
14     /**
15     * Microsoft auth app
```

Dans la classe Config, ajouter l'annotation @Configuration.

Par défaut, Spring cherchera les propriétés dans application.properties

```

/**
 * default root directory.
 */
@Value("${config.defaultRootDir:none}")
private String defaultRootDir;
/**
 * @return the defaultRootDir
 */
public String getDefaultRootDir() {
    if (defaultRootDir.equals("none")) {
        return DEFAULT_ROOT_DIR;
    }
    return defaultRootDir;
}
/**
 * @param defRootDir the defaultRootDir to set
 */
private void setDefaultRootDir(final String defRootDir) {
    this.defaultRootDir = defRootDir;
}

```

Ajouter le @Value avec le nom de la propriété à chercher. « none » représente une valeur par défaut s'il n'est pas trouvé dans le application.properties. Dans le getter, nous vérifions si la valeur a été initialisé via application.properties ou bien si on doit charger la valeur par défaut.