TP 5 Android Web Services

Télécharger PDF

[../tp4.pdf]

Objectifs du TP

Interrogation de services web REST à partir de votre application Android avec la bibliothèque Retrofit

Outils et Versions

- Android Studio [https://developer.android.com/studio/index.html#tos-header] Version: 3.0.1
- Android : Version 5.0.1 (API 21)
- Retroit : Version 2.0.0

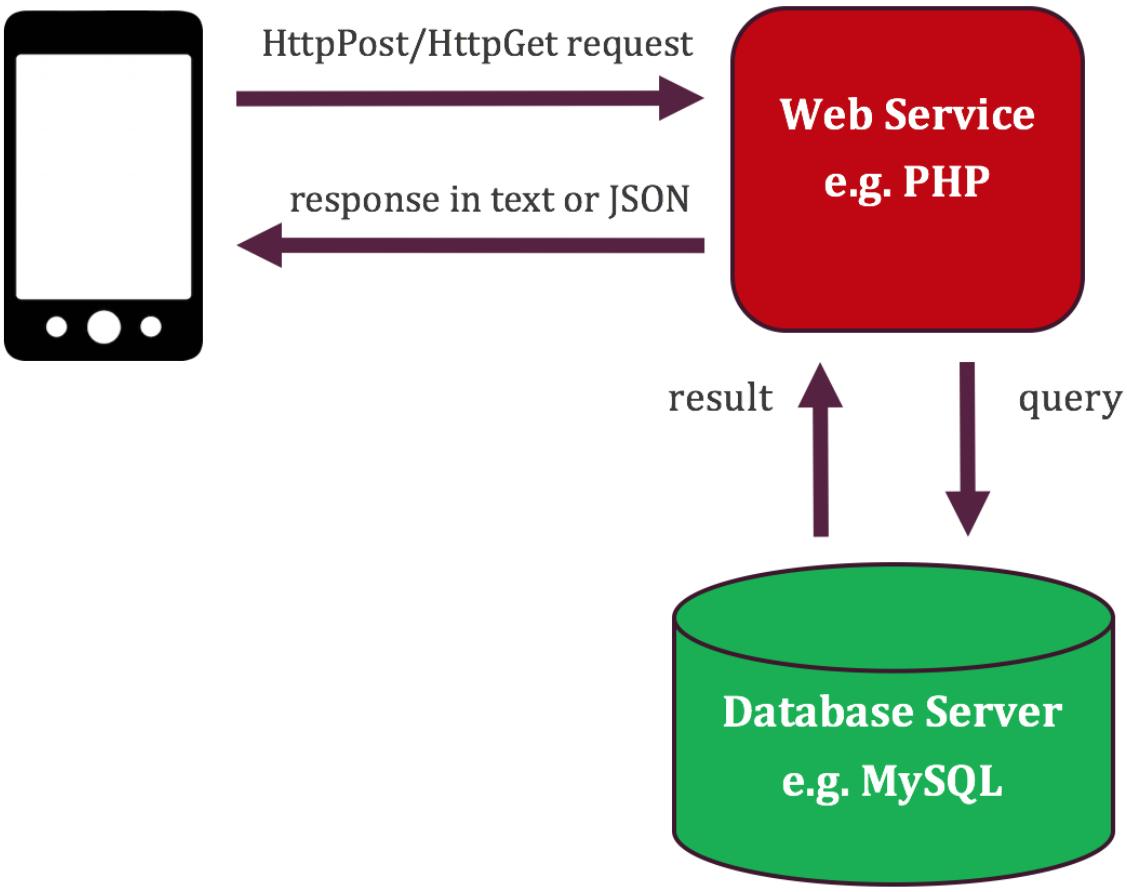
I. Les web services

Un service Web est une norme utilisée pour échanger des informations entre des applications ou des systèmes de type hétérogène. Les applications logicielles écrites dans divers langages de programmation et s'exécutant sur diverses plates-formes peuvent utiliser des services Web pour échanger des informations sur Internet à l'aide du protocole http.

1.Le besoin d'utiliser le service Web dans les applications Android

Les applications Web existantes ont également besoin de créer des applications mobiles pour montrer leur présence dans la plate-forme mobile. Exposer les fonctionnalités existantes des applications est un peu difficile car toutes les fonctionnalités doivent être réécrites dans les plates-formes respectives. Mais il peut être facilement réalisé avec beaucoup de facilité en créant un service Web et en exposant les fonctionnalités existantes en tant que méthodes Web aux plateformes mobiles.
Voici les quelques avantages de l'utilisation de Web Service sous Android:
- Rendre le client plus léger.
- Réutilisation des fonctionnalités existantes.
- Utilisation d'une base de données distante.

Voici une architecture simplifée d'une application Android qui utilise des web services :



II. Retrofit

Retrofit est un client REST pour Android et Java, permettant de grouper et organiser toutes vos APIs de manière simple et propre.
Les annotations sont utilisées pour contrôler les paramètres d'appel de l'API, définies dans des interfaces, que vous appelez à partir de classes.
Toutes les APIs http sont représentées sous forme d'interfaces, comme suit :

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

1. Déclaration d'APIs :

Chaque méthode doit avoir une annotation http qui fournit la méthode utilisée ainsi que l'URL. 5 annotations sont fournies : GET, POST, PUT, DELETE et HEAD. Voici un exemple d'appel :

```
@GET("users/list")
```

Il est possible de spécifier les paramètres de la requête dans l'URL :

```
@GET("users/list?sort=desc")
```

Pour un remplacement dynamique de blocs et de paramètres dans la méthode, utiliser : {}, puis référencer ce bloc avec l'annotation @ PATH, comme suit :

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId);
```

Il est également possible d'utiliser des paramètres de requêtes (query parameters)
comme suit :

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId, @Query("sort")
String sort);
```

L'URL résultante sera alors comme suit : **group/id_val/users ?sort=sort_val**

Pour insérer plusieurs paramètres de requête à la fois, utiliser une Map:

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId, @Query("sort")
String sort);
```

Consulter le site de Retrofit pour plus d'exemples d'utilisation de Retrofit.

2. Format de Données

Par défaut, Retrofit peut seulement dé-sérialiser les HTTP bodies en objets ResponseBody de la bibliothèque OkHttp, et n'accepte les requêtes que sous format
RequestBody pour le @Body.
Cependant, un ensemble de convertisseurs sont disponibles pour supporter d'autres
types, dont Gson. Gson est une librairie Java utilisée pour convertir les objets Java en
leur représentation JSON, et vice-versa.

Pour supporter Retrofit, il faut d'abord ajouter deux dépendances :

1. Librairie Retrofit
2. Gson

Activité 1
Créer un projet Android TP5. Ajouter les dépendances pour Retrofit et gson dans le fichier build.gradle (du Module), en insérant les lignes suivantes dans la partie dependencies:

```
compile 'com.squareup.retrofit2:retrofit:2.4.0'
compile 'com.google.code.gson:gson:2.6.2'
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

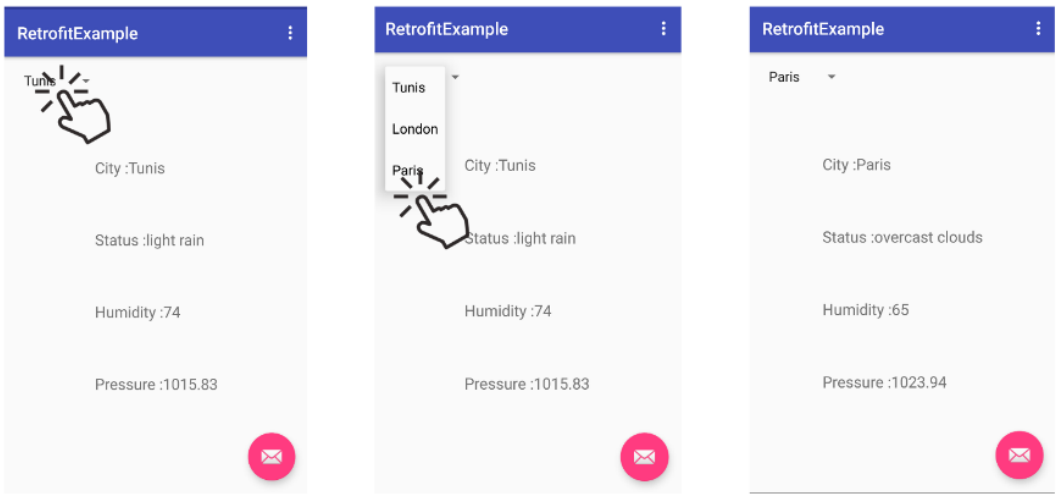
N'oubliez pas de synchroniser le projet pour que les dépendances soient prises en
compte !

III. Exercice : Accès à un Service Web

1. Objectif

Durée estimée de l'exercice : 2h

L'objectif de ce TP est de réaliser une application android qui accède à des données provenant d'un service web externe.



2. Génération des classes POJO à partir des données JSON

Le service web que nous appelons est un service public de consultation de météo.
Pour consulter la météo actuelle à Tunis, nous utilisons l'URI suivante :
http://api.openweathermap.org/data/2.5/weather?q=Tunis&APPID=17db59488cadcad345211c36304a9266
[http://api.openweathermap.org/data/2.5/weather?q=Tunis&APPID=17db59488cadcad345211c36304a9266]
La réponse obtenue ressemble à la suivante :

```
{
  - coord: {
        lon: 10.17,
        lat: 36.82
    },
  - weather: [
      - {
            id: 500,
            main: "Rain",
            description: "light rain",
            icon: "10n"
        }
    ],
    base: "cmc stations",
  - main: {
        temp: 286.629,
        pressure: 1015.83,
        humidity: 74,
        temp_min: 286.629,
        temp_max: 286.629,
        sea_level: 1031.53,
        grnd_level: 1015.83
    },
  - wind: {
        speed: 6.91,
        deg: 290.504
    },
  - rain: {
        3h: 0.395
    },
  - clouds: {
        all: 92
    },
    dt: 1462227497,
  - sys: {
        message: 0.0025,
        country: "TN",
        sunrise: 1462162975,
        sunset: 1462212597
    },
    id: 2464470,
    name: "Tunis",
    cod: 200
}
```

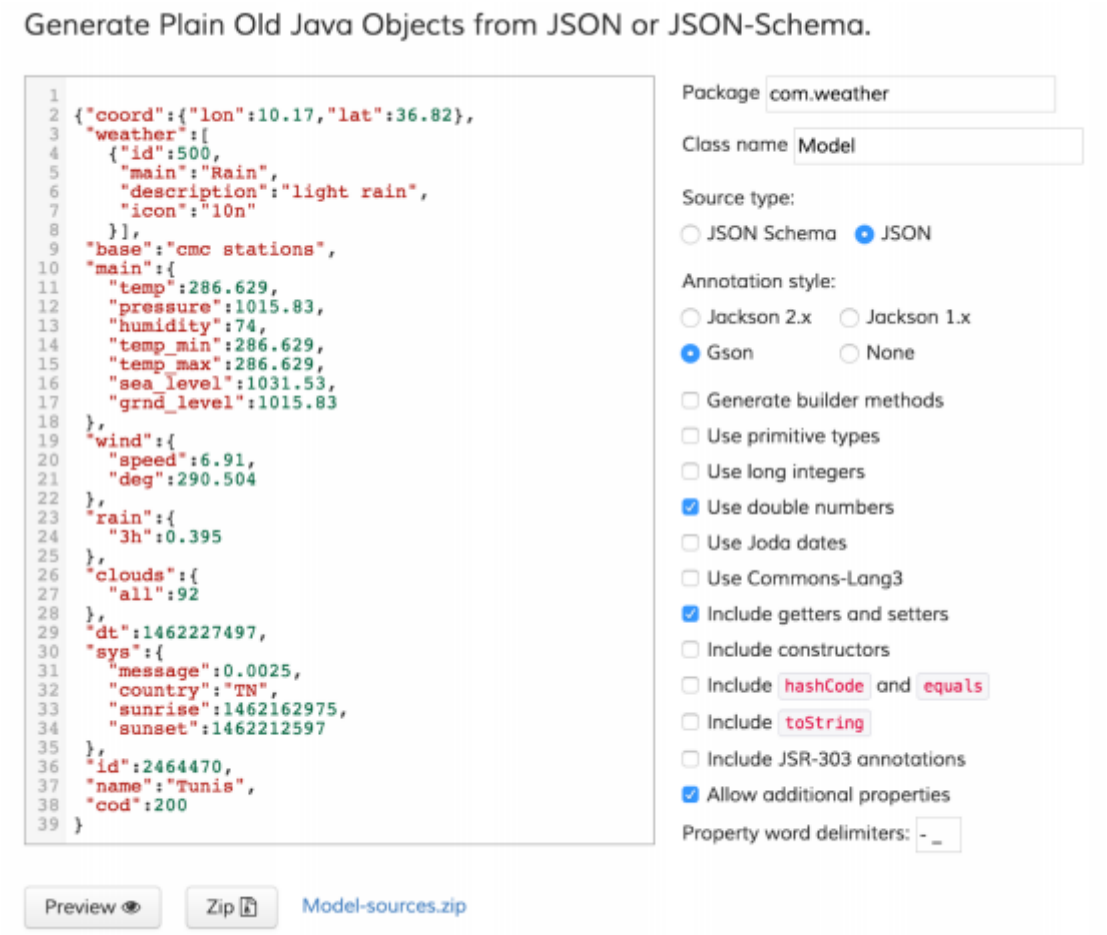

Pour obtenir un APPID valide, il suffit de s'inscrire sur le site http://openweathermap.org [http://openweathermap.org] , un APPID sera généré pour vous, vous le trouverez dans votre profil. Il suffira ensuite de remplacer le APPID de l'URI précédente par la votre. Une fois une réponse valide obtenue, sous la forme d'un document JSON comme le montre la figure précédente, il est possible de générer les classes POJO que vous allez utiliser pour manipuler les données du service web à votre guise.
Pour cela :

• Aller au site : http://www.jsonschema2pojo.org/ [http://www.jsonschema2pojo.org/]
• Coller le document JSON, que vous obtenez en tapant l'URI dans le navigateur, dans

la case de gauche

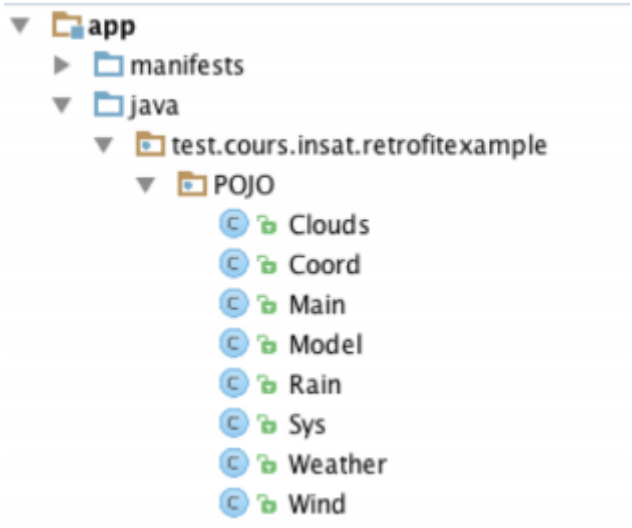
• Indiquer dans la colonne de droite le nom de package que vous choisissez (par exemple com.weather), le nom de la classe principale (par exemple Model), JSON

comme type source, et Gson comme style d'annotation, comme suit :



• Cliquer sur Zip pour télécharger le projet contenant les classes générées.

• Copier ensuite les classes téléchargées sous un nouveau package POJO de votre répertoire source :



3. Création de l'interface

Commencer par créer une interface comportant 4 champs de texte, intitulés respectivement txt_city, txt_status, txt_humidity et txt_press.
Ajouter également, dans le fichier Manifest, la permission d'accès à internet :

```
<uses-permission android:name="android.permission.INTERNET"/>
```

4. Interface RestInterface

• Créer une interface appelée RestInterface dans votre projet. Cette interface permettra de mapper une URI donnée à un appel de méthode, grâce aux annotations fournies par l'API Retrofit.
• Pour obtenir la météo de Tunis, le code de RestInterface devra être comme suit :

```
public interface RestInstance {


    @GET("weather?q=Tunis&APPID=17db59488cadcad345211c36304a9266")
    Call<Model> getWeatherReport() ;
}
```

Vous remarquerez ici que l'URI commence par weather. En effet, le path principal sera

indiqué dans la classe principale, lors de l'appel du service.

5. Retrofit Builder

Dans la même classe RestInterface, créer un Retrofit Builder pour construire la requête:

```
public  static Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(url).addConverterFactory(GsonConverterFactory.create
        .build() ;
```

L'URL utilisée ici est une chaîne statique représentant l'URL principale fixe du web service
http://api.openweathermap.org/data/2.5/
[http://api.openweathermap.org/data/2.5/]

6. Appel au service

Pour faire appel au service, aller dans la classe de votre activité principale (pour mon
cas : MainActivity.java) et suivre les étapes suivantes :
• Créer les TextViews appropriés.
• Créer un objet RestInterface:

```
RestInstance restInstance = RestInstance.retrofit.create(RestInstance
```

• Appeler le service:

```
Call<Model> call = restInstance.getWeatherReport();
```

• La fonction getWeatherReport, définie dans l'interface, est une fonction asynchrone,
elle doit donc implémenter deux fonctions de callback : en cas de succès et en cas
d'échec. Son appel sera comme suit :

```
call.enqueue(new Callback<Model>() {
                @Override
                public void onResponse(Call<Model> call, Response<Mode

                    city.setText("City: "+response.body().getName());
                    status.setText("Status: " +response.body().getWeat
                    humidity.setText("Humidity: "+ response.body().get
                    pressure.setText("Pressure: "+ response.body().get

                }

                @Override
                public void onFailure(Call<Model> call, Throwable t) {
                    Toast.makeText(MainActivity.this,t.getMessage(),To
                }
        });
```

Activité 2
Lancer le projet que vous avez créé. Bien vérifier que vous disposez d'une connexion internet, et que le APPID défini dans votre URI est valide (pour cela, tester l'URI donnée dans II.2 directement dans le navigateur).

7. Ajout de paramètres

Nous aimerions maintenant que l'appel au web service soit paramétré. C'est à dire que le nom de la ville dont on va afficher la météo sera passé en paramètre à l'appel de la fonction getWeatherReport, et non pas en dur directement dans RestInterface.

Activité 3
Ajouter un paramètre à la fonction getWeatherReport. Attention, la fonction de Callback doit toujours être le dernier paramètre défini !

8. Liste de Choix

Pour obtenir le résultat final, comme montré dans la partie II.1, il faut définir une liste déroulante (spinner), dans lequel on trouvera les noms des villes dont nous voulons afficher la météo.

Activité 4

Ajouter une liste déroulante à votre activité. Insérer les noms des villes de votre choix (pour moi, c'était Tunis, London et Paris). Modifier votre code de façon à ce que l'appel au service web change selon le nom de la ville sélectionnée.