TP réalisé par : Tartara Guillaume
Réalisé le : 18/10/2107

TP Android : Traiter des données JSON

Résumé

Le but de ce TP est créer une application Android qui va utiliser des données JSON.

Nous allons dans un premier temps envoyer des données JSON que nous allons créer puis nous récupérerons une réponse en JSON que nous exploiterons.

Pour se faire, nous n'utiliserons pas d'appel à des web services pour éviter les problèmes internet. Nous utiliserons en permanence un manager qui simule le web service.

Pré- requis

- Savoir programmer une application Android qui utilisent plusieurs activités.
- Comprendre le JSON

Code source

Par défaut l'ensemble du projet est hébergé sur GitHub à l'adresse suivante :

https://github.com/tguillaume59/AND_MaJsonApp

Dans ce répertoire se trouve 2 dossiers contenant:

- Le code pour démarrer le TP
- Le code finale

Contexte

Dans ce TP, vous allez devenir un agent secret très connu en travaillant pour le MI6 pendant une demi heure. Vous allez devoir vous connecter pour choisir votre mission. Une fois ceci fait, vous allez choisir votre arme (et donc les équipements de celle-ci) ainsi que votre voiture. Lorsque vous êtes sur de vous, vous validez et la mission commence.

Votre mission si vous l'acceptez créer l'application en utilisant des données JSON pour créer la mission de James Bond.

Réalisation

Introduction

Les classes:

- Le préfixe "MJA devant chaque classe pour **MaJsonApp**. Ainsi lorsque l'on débogue avec les logs nous pouvons facilement identifier si le crash provient d'une de nos classes ou s'il s'agit d'une classe Android qui a générée le bug.
- Une classe qui définit une activité se termine par Activity : HBGTestActivity

Les variables :

- Pour un attribut de classe il y a un m devant: mAttribut.
- Pour un paramètre de méthode il y a un s devant: sParametre.
- Pour une variable temporaire dans une méthode il y a un t : tVariable.
- Pour une valeur de retour, il y a un r devant : rValeur.

Connexion

Dans un premier temps, nous allons créer une page de connexion. Sur celle-ci nous allons récupérer les données saisies par l'utilisateur dans les zones de saisies, créer un objet JSON contenant ces données et les envoyer à notre WS factice. Dans l'ensemble du TP, nous ne toucherons pas à la classe: "MJAWebServicesManager".

Les bons identifiants pour se connecter sont : login : "jamesbond" et mdp : "azerty".
Pour gagner du temps et ne pas retaper à chaque fois les aidentifiants, vous pouvez retirer le commentaire dans la méthode OnCreate() de la mainActivity.

MainActivity :: OnCreate()

```
//mLoginEditText.setText(getString(R.string.ws_manager_good_login));
//mMdpEditText.setText(getString(R.string.ws_manager_good_mdp));
```

Pour commencer nous allons récupérer les données saisies lorsque l'utilisateur clique sur le bouton de connexion.

MainActivity :: onClick()

```
public void onClick(View view) {
        switch (view.getId()){

            case R.id.activity_main_login_btn:
                String tLogin = mLoginEditText.getText().toString();
                String tMdp = mMdpEditText.getText().toString();
```

Une fois les infos récupérées, nous devons les envoyer au WS via du JSON. Nous utiliserons donc la méthode : createLoginJsonObject().

Vous pouvez retirez le commentaire ci dessous dans le onClick().

```
//JSONObject tJsonLogin = createLoginJsonObject(tLogin,tMdp);
```

Vous trouverez la méthode de création du Json ici. Cette méthode existe déjà pas besoin de la copier de nouveau. Dans celle-ci, nous créons un Objet JSON auquel nous ajouter chaque champs de connexion. Le premier paramètre de la méthode "put" est la clé de l'objet, pour éviter des erreurs de frappes : utiliser les clés dans le fichier String.xml.

```
private JSONObject createLoginJsonObject(String sLogin, String sMdp)
{
        JSONObject rJsonObject = new JSONObject();
        try{
            rJsonObject.put(getString(R.string.json_login),sLogin);
            rJsonObject.put(getString(R.string.json_mdp),sMdp);
        }catch (Exception e){
            Log.e(TAG, "createLoginJsonObject : erreur --> "
+e.getLocalizedMessage());
        }
        return rJsonObject;
    }
```

le JSON obtenu :

```
{"login":"jamesbond","password":"azerty"}
```

Notre objet est donc prêt à être envoyé au web service. Ce dernier nous renverra une réponse que nous traiterons via OnSuccessLoginResponse().

```
//étape 2
//on appelle le WS <-- fictif ici
//String tResponse = mWSManager.startConnexionService(tJsonLogin);
//OnSuccessLoginResponse(tResponse);
```

Si les identifiants sont corrects, le WS retourne un message positif sous cette forme:

```
{"code":200,"message":"SUCCESS","agent":"007"}
```

Pour traiter cette réponse, nous allons transformer le String reçu en un objet JSON et l'exploiter dans la méthode : private void OnSuccessLoginResponse()

```
private void OnSuccessLoginResponse(String sJsonReceive){
        try {
            JSONObject tJsonReceive = new JSONObject(sJsonReceive);
            String tMessage = tJsonReceive.getString("message");

if((this.getString(R.string.ws_manager_success)).equals(tMessage)){
                //SUCCESS donc on passe à la vue suivante
                String tAgent = tJsonReceive.getString("agent");
                Intent tIntent = new
Intent(this,MJAMissionActivity.class);
                tIntent.putExtra("BUNDLE_AGENT",tAgent);

                startActivity(tIntent);
            }else {
                //erreur donc on affiche le messagde d'erreur
                Toast.makeText(this, tMessage,
Toast.LENGTH_SHORT).show();
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
```

Dans cette méthode nous allons vérifier que la connexion a réussie. Si oui, nous lançons la vue suivante pour choisir la mission en lui donnant l'information de l'di Agent reçu via le JSON.

Listing des missions

Dans cette partie, nous allons occuper de lister l'ensemble des missions disponibles. Nous travaillerons donc dans l'activité : "MJAMissionActivity".

Nous allons prendre soin dans le OnCreate de récupérer l'id Agent via le code ci dessous :

```
Bundle tBundle = this.getIntent().getExtras();
if(tBundle != null){
     mAgentId = tBundle.getString("BUNDLE_AGENT");
}
```

Dans le onResume(), nous allons récupérer la liste des missions en appelant notre WS.

```
@Override
protected void onResume() {
     super.onResume();
     //getMissions(mAgentId);
}
```

Dans la méthode getMissions(), nous allons effectuer la requête au WS et ensuite la traiter avec le code ci-dessous.

```
private void getMissions(String sAgentId){
      String JsonMissions =
MJAWebServicesManager.getInstance(this).startGetMissionsService(sAge
ntId);
      //onReceiveResponse(tJsonMissions);
}
```

Il faudra donc retirer le commentaire dans le onResume et dans le traitement via la méthode onReceive().

Le WS nous retourne un JSON contenant le message et le code d'erreur ainsi qu'un tableau de missions sous la forme suivante :

```
{
   "message":"SUCCESS",
   "code":200,
   "agent":"007",
   "missions":[
      {
         "id":1,
         "nom":"Spectre",
         "date":2015,
         "ville":"Rome",
         "pays":"Italie"
      },
      {
         "id":2,
         "nom":"Skyfall",
         "date":2012,
         "ville":"Londres",
         "pays":"Angleterre"
      },
      {
         "id":3,
         "nom":"Quantum of Solace",
         "date":2008,
         "ville":"Bregenz",
         "pays":"Autriche"
      },
      {
         "id":4,
         "nom":"Casino Royale",
         "date":2006,
         "ville":"Nassau",
         "pays":"Bahmas"
      }
   ]
}
```

Dans la méthode void onReceiveResponse(), nous allons vérifier que nous avons bien un code erreur success.

Nous allons donc récupérer l'objet missions qui est un tableau de missions :

```
JSONArray tJsonArrayMission =
tJsonObjectReceive.getJSONArray(getString(R.string.json_mission_arra
y));
```

Puis nous allons parcourir ce tableau pour se faire une liste de missions.

```
for(int i = 0 ; i <  tJsonArrayMission.length() ; i++){
      //On récupere toutes les missions dans le json
      tObjectRead = tJsonArrayMission.getJSONObject(i);
      MJAMission tMission = new MJAMission(

tObjectRead.getInt(this.getString(R.string.json_mission_id)),

tObjectRead.getString(this.getString(R.string.json_mission_name)),

tObjectRead.getInt(this.getString(R.string.json_mission_date)),

tObjectRead.getString(this.getString(R.string.json_mission_town)),

tObjectRead.getString(this.getString(R.string.json_mission_country))

      );
     mListMissions.add(tMission);
}
```

Pour finir on donne cette liste à l'adapteur pour que les missions s'affichent correctement.

```
mArrayAdapter = new
MJAListMissionAdapter(this,R.layout.item_listview_mission,mListMissi
ons);
mListviewMission.setAdapter(mArrayAdapter);
```

Pour finir lorsque que l'on clique sur un item de la liste, on transmet les infos de la mission sélectionnée à la vue suivante : celle de la sélection de l'équipement.

```
@Override
public void onItemClick(AdapterView<?> adapterView, View view, int
i, long l) {
   Intent tIntent = new Intent(this, MJASelectEquipActivity.class);
   MJAMission tMission = mListMissions.get(i);
   tIntent.putExtra("MISSION_NAME", tMission.getNom());
   tIntent.putExtra("MISSION_ID",tMission.getId());
   tIntent.putExtra("AGENT_ID",mAgentId);
   startActivity(tIntent);
}
```

Valider le choix de l'équipement

Lorsque l'utilisateur a sélectionné son arme, et sa voiture il valide son choix. Nous allons créer un objet JSON plus complexe que les précédents. En effet pour valider la mission nous devons sauvegarder toutes les donnés stockée depuis le début de la navigation ainsi que les équipements sélectionnés.

- L'id de l'agent
- L'id de la mission
- Le nom de la mission
- La voiture
    - La marque
    - Le model
- L'arme
    - un tableau d'équipements

Le json à créer ressemblera donc a ceci:

```
{
   "mission_id":1,
   "mission_name":"Spectre",
   "car":{
      "car_model":"Bestia GTS",
      "car_brand":"Grotti"
   },
   "agent_id":"007",
   "weapon":{
      "gun_manufacter":"Walter",
      "gun_equipments":[
         {
            "mName":"lampe",
            "mType":"tactique"
         }
      ],
      "gun_model":"PP9"
   }
}
```

Nous allons donc utiliser cette fois ci la bibliothèque GSON de Google pour construire l'objet Json rapidement.

Pour ajouter GSON, dans build.gradle (module app), dans la section dépendance : rajouté la ligne suivante :

```
dependencies {
    //… … …

    //GSON
```

```
    compile 'com.google.code.gson:gson:2.8.2'

}
```

Puis synchroniser de nouveau le gradle. Voilà GSON a été ajouté.

On retourne dans l'activité MJASelectEquipActivity().

Dans le onClick, on créé le Json

```
case R.id.activity_equipment_start_mission_btn:
      //… …
      String tJson = createJsonDataEquipmentsSelected();
```

Pour créer un Objet Json via GSON, il nous faut une hasmap, et un GSON builder qui transformera la hashmap en JSON.

```
private String createJsonDataEquipmentsSelected(){
        Log.i(TAG, "createJsonDataEquipmentsSelected");
        final GsonBuilder builder = new GsonBuilder();
        final Gson gson = builder.create();

        //le json final
        HashMap<String, Object> tJsonMap = new HashMap<>();
```

Ensuite on ajoute les valeurs des bundles transmirent d'activité en activité.

```
//on ajoute les données de premier niveau
tJsonMap.put("agent_id",mAgentId);
tJsonMap.put("mission_id",mMissionId);
tJsonMap.put("mission_name",mMissionName);
```

Ensuite on ajoute l'arme choisie ainsi que les équipements

```
//on créé le niveau 2 : l'arme avec ses équipement
HashMap<String, Object> tJsonGun = new HashMap<>();

tJsonGun.put("gun_manufacter",mCurrentGunSelected.getManufacter());
        tJsonGun.put("gun_model",mCurrentGunSelected.getModel());

tJsonGun.put("gun_equipments",mCurrentGunSelected.getListEquipment()
);
//on ajoute le niveau 2 dans le niveau 1
tJsonMap.put("weapon",tJsonGun);
```

Et enfin on ajoute la voiture :

```
HashMap<String, Object> tJsonCar = new HashMap<>();
tJsonCar.put("car_brand",mCurrentCarSelected.getBrand());
tJsonCar.put("car_model",mCurrentCarSelected.getModel());
//on ajoute la voiture dans le premier niveau
tJsonMap.put("car",tJsonCar);
```

On transforme le tout en Json :

```
//on transforme la HashMap en un string JSON
String rJson = gson.toJson(tJsonMap);
```

Pour terminer le process: on envoie au web service le json, et si on a un success alors on affiche la vue "Mission Acceptée !".

```
//envoie au WS normalement
String tJsonResponse =
MJAWebServicesManager.getInstance(this).startMissionEquipmentSelecte
dService(tJson);

//si SUCCESS : mission validé donc on passe à la derniere vue
onSuccessSelectEquipementWS(tJsonResponse);
```

Conclusion

Au cours de ce TP, nous avons créé des objets JSON simples, nous avons lu un objet JSON sur plusieurs niveaux contenant un tableau d'objets JSON. Et enfin nous avons créé notre objet final qui contient plusieurs niveaux avec une librairie.

Pour aller plus loin

Il est possible de désiralisé directement un objet JSON en un objet java sans passé par des JSONObject :: getxxxx("ma_clé").

```
Gson gson = new Gson();
String jsonInString = "{\"userId\":\"1\",\"userName\":
\"Guillaume\"}";
User user= gson.fromJson(jsonInString, User.class);
```

https://stackoverflow.com/questions/38636254/how-to-convert-json-to-java-object-using-gson