IUT de Paris - Rives de Seine
Université Paris Cité

R4.Real.11 – Développement pour applications mobiles

– TP 5 –
Connectivité mobile et utilisation des services Web

Pr Chaouche A.-C.
ac.chaouche@gmail.com

Résumé

Prérequis

- Maîtrise de la programmation Android
- Maîtrise des langages PHP et SQL

SQL

Objectifs du TP

- Elaborer une communication avec un serveur Web sous Android
- Consommer des services obtenus depuis le serveur

Utilisation des services Web sous Android

Comment ?

- PHP récupère les données à partir d'une BD MySQL, qui seront par la suite **encodées au format JSON** et envoyées au client Android
- L'App Android **récupère** les données, les **parse** et les **exploite** dans l'App

TP5a : Accéder au serveur via le navigateur

Lancer dans le navigateur :

- **http://[server]/powerhome_server/getHabitats.php**

- Sous **Chrome**, **l'extension** JSONView permet de mieux afficher le format JSON

Utilisation des services Web sous Android

Côté serveur

1. Préparation du schéma de la BD
2. Lancer les serveurs Apache et MySQL (XAMPP)
3. Créer une BD via un client MySQL
4. Se connecter à la BD via un script PHP
5. Retourner des données au format JSON
6. Authentification des clients

Côté client

1. **Préparer le projet Android**
2. **Importer la bibliothèque externe ION**
3. **Récupérer les données distantes sous Android**
4. **Parser les données JSON pour constituer les objets Java**
5. **Envoyer des données au serveur distant**

Client-1. Préparer le projet Android

- **Ajouter la permission :**

/manifests/AndroidManifest.xml

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

- **Préparer l'URL : http://[server]/powerhome_server/getHabitats.php**

/java/MainActivity.java

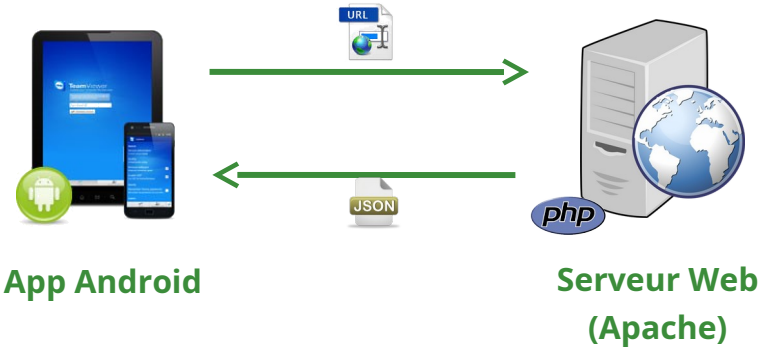
```
String urlString = "http://[server]/powerhome_server/getHabitats.php";
...
```

Client-2. Importer la bibliothèque externe ION (1/3)

Deux modes :

1. **En mode synchrone**
(l'IHM est bloquée en attendant la réponse)

2. **En mode asynchrone**
(la requête est exécutée en arrière plan)



Deux méthodes en **mode asynchrone** :

Android SDK : `android.os.AsyncTask`

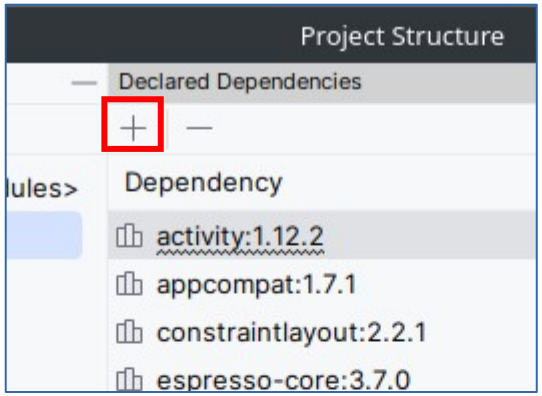
- **Doc :** https://developer.android.com/reference/android/os/AsyncTask.html

Bibliothèque externe : `com.koushikdutta.ion`

- **Doc :** https://github.com/koush/ion

Client-2. Importer la bibliothèque externe ION (2/3)

Bibliothèque : `com.koushikdutta.ion (v3.1.0)`



Méthode 1 :

- Project Structure > Depenendencies > Modules > app

Méthode 2 :

- Dans un script Gradle

/Gradle Scripts/build.gradle (Module: app)

```
...
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation libs.ion
}
```

Client-2. Importer la bibliothèque externe ION (3/3)

Configuration de la bibliothèque : `libs.ion`

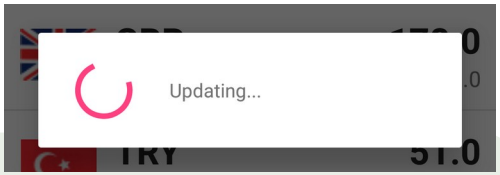
/Gradle Scripts/libs.versions.toml (Version Catalog)

```
[versions]
...
ion = "3.1.0"

[libraries]
...
ion = { group = "com.koushikdutta.ion", name = "ion",
        version.ref = "ion" }
...
```

Utilisation des services Web sous Android

Client-3.a. Afficher un `ProgressDialog`



/java/MainActivity.java

```
ProgressDialog pDialog;

public void getRemoteHabitats() {
    pDialog = new ProgressDialog(this);
    pDialog.setMessage("Getting list of habitats...");
    pDialog.setIndeterminate(true);
    pDialog.setCancelable(false);
    pDialog.show();
    ...
}
```

Client-3.b. Récupérer un String

/java/MainActivity.java

```
public void getRemoteHabitats() {
    ...
    String urlString = "http://[server]/powerhome_server/getHabitats.php";
    Ion.with(this)
        .load(urlString)
        .asString()
        .setCallback(new FutureCallback<String>() {
            @Override
            public void onCompleted(Exception e, String result) {
                pDialog.dismiss();
                if(result == null)
                    Log.d(TAG, "No response from the server!!!");
                else {
                    // Traitement de result
                }
            }
        });
}
```

Utilisation des services Web sous Android

Client-3.c. Récupérer la réponse HTTP complète

/java/MainActivity.java

```
public void getRemoteHabitats() {
    ...
    Ion.with(this)
      .load(urlString)
      .asString()
      .withResponse()
      .setCallback(new FutureCallback<Response<String>>() {
         @Override
         public void onCompleted(Exception e,Response<String> response){
            ...
            if(e == null && response != null){
               Log.d(TAG, "Http code: " + response.getHeaders().code());
               Log.d(TAG, "Result: " + response.getResult());
            }
            ...
        }
     });
}
```

Client-4.a. Préparer le modèle de données

/java/entities/User.java

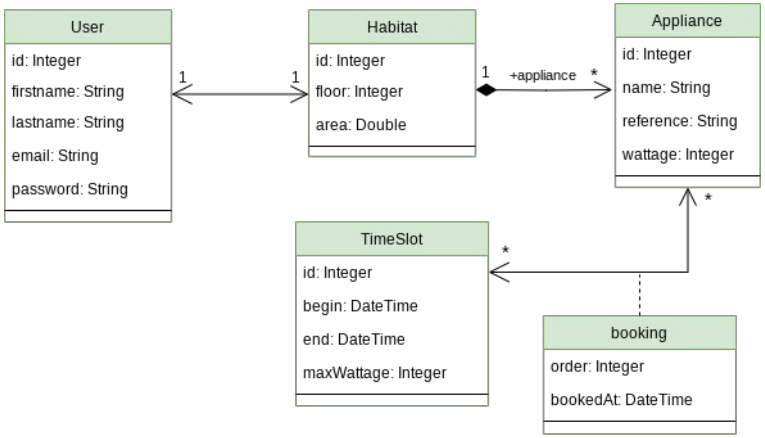
```
public class User {
   int id; String firstname, ...
   Habitat habitat;
}
```

/java/entities/Habitat.java

```
public class Habitat {
   int id; ...
   User resident;
   List<Appliance> appliances;
}
```

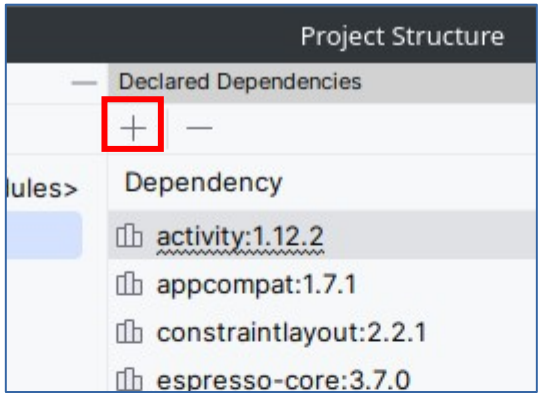
/java/entities/Appliance.java

```
public class Appliance {
   int id; ...
}
```



Client-4.b. Importer la bibliothèque Gson (1/2)

Bibliothèque : com.google.code.gson (v2.13.1)



Méthode 1 :

- Project Structure > Modules > app
  > Onglet Depenendencies

Méthode 2 :

- Dans un script Gradle

/Gradle Scripts/build.gradle (Module: app)

```
...
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation libs.gson
}
```

Client-4.b. Importer la bibliothèque Gson (2/2)

Configuration de la bibliothèque : `libs.gson`

/Gradle Scripts/libs.versions.toml (Version Catalog)

```
[versions]
...
gson = "2.13.1"

[libraries]
...
gson = { group = "com.google.code.gson", name = "gson",
         version.ref = "gson" }
...
```

Client-4.c. Parser la réponse Json

/java/entities/Habitat.java

```
public static Habitat getFromJson(String json){
    Gson gson = new Gson();
    Habitat obj = gson.fromJson(json, Habitat.class);
    return obj;
}

public static List<Habitat> getListFromJson(String json){
    Gson gson = new Gson();
    Type type = new TypeToken<List<Habitat>>(){}.getType();
    List<Habitat> list = gson.fromJson(json, type);
    return list;
}
```

TP5b : Récupérer les données distantes

1. **Préparer l'URL de la requête**

```
String urlString = "[server]/powerhome_server/getHabitats.php";
```

1. **Récupérer la liste des habitats depuis le serveur (en utilisant `Ion`)**

```
Ion.with(this).load(urlString) ... .setCallback(...);
```

1. **Parser la liste des habitats (en utilisant `Gson`)**

```
Habitat.getFromJson(json);
```

Client-5. Envoyer des données au serveur

```
urlString = "http://[server]/powserhome_server/addHabitat.php?...";
```

/java/MainActivity.java

```
...
String urlString =
          "http://[server]/powserhome_server/addHabitat.php?...";
Ion.with(this)
   .load(urlString)
   .asString()
   .withResponse()
   .setCallback(new FutureCallback<Response<String>>() {
       @Override
       public void onCompleted(Exception e, Response<String> response) {
         ...
       }
   });
...
```

TP5c : Envoyer des données au serveur

1. **Préparer l'URL de la requête**

```
String urlString = "http://[server]/powerhone_server/" +
                   "addHabitat.php?floor=X&...";
```

1. **Envoyer la requête au serveur (en utilisant `Ion`)**

```
Ion.with(this).load(urlString) ... .setCallback(...);
```

1. **Vérifier le bon traitement de la requête**

```
...
if(response.getHeaders().code() == 200) { ... }
...
```

TP5d : Autorisation basée sur les jetons

Côté client

Authentification (Login)

```
String urlString = "http://[server]/powerhone_server/" +
                   "login.php?email=X&password=Y";
```

Récupération de la liste des habitats (getHabitats)

```
String urlString = "http://[server]/powerhone_server/" +
                   "getHabitats.php?token=X";
```

Ajout d'un nouvel habitat (addHabitat)

```
String urlString = "http://[server]/powerhone_server/" +
                   "addHabitat.php?token=X&floor=Y&...";
```