ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Module :

Développement mobile native
Chapitre 7: Communication réseau sous Android

Pr. Zakia EL UAHHABI

Pourquoi une Base de Données Distante??

- Dans le développement des applications mobiles, nous aurons recours à une base de données distante lorsque le scenario nécessite que les données soient **stockées de manière centralisée**.

- Mettre en place une architecture **Client/Serveur** afin de centraliser les données et permettre à plusieurs clients de les récupérer et d'y contribuer.

Architecture Client/Serveur dans Android

Architecture Client/Serveur dans Android

Pour instaurer cette architecture, il faut passer par les étapes suivantes:

Partie Serveur:

1.Installation et démarrage du serveur.
2.Création d'une base de données et de tables MySQL.
3.Mettre en place un web service via un script PHP pour se connecter à la base de données MySQL.
4.Retourner des données au format JSON

Partie Android Client

1.Ajouter la permission de la connexion internet dans le Manifest.XML
2.Ecrire le code java qui communique avec le web service.

Partie serveur

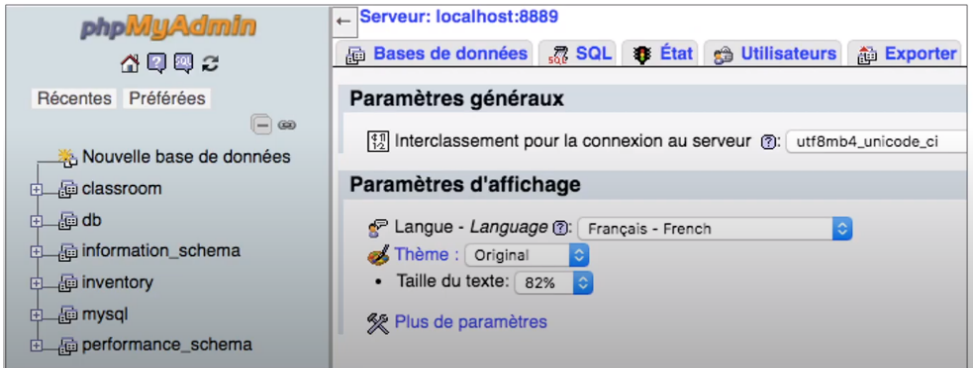
Etape 1: Installation et démarrage du serveur

- Le choix du serveur dépend du système d'exploitation utilisé.
- Il existe plusieurs logiciels qui font office de serveur Web et qui supportent automatiquement le langage PHP. Les plus connus sont: **WAMP** (Windows), **XAMP** (Windows, Linux et Mac OS), **MAMP** (Macintosh), **LAMP** (linux), **FAMP** (FreeBSD).

Partie serveur

Etape 2: Création d'une base de données et de tables MySQL

- Utiliser l'outil phpMyAdmin pour créer une base de données et une table.



Partie serveur

Etape 3: Création d'un web service en script php pour se connecter à la base de données

```php
<?php
  DEFINE('DB_USERNAME', 'root');
  DEFINE('DB_PASSWORD', 'root');
  DEFINE('DB_HOST', 'localhost');
  DEFINE('DB_DATABASE', 'db');
  $mysqli = new mysqli(DB_HOST, DB_USERNAME, DB_PASSWORD, DB_DATABASE);
  if (mysqli_connect_error())
  {
    die('Connect Error ('.mysqli_connect_errno().') '.mysqli_connect_error());
    echo "Connection failed";
  }
  else
    echo "Connected successfully.<br/><br/>";
    $mysqli->close();
  ?>
```

Partie serveur

Etape 3: Création d'un web service en script php pour se connecter à la base de données

Exécuter le web service dans le serveur:

- **Sur Windows:**
  Pour exécuter le web service par le serveur il faut le placer dans le dossier C:/wamp/www
- **Sur MacOS:**
  Pour exécuter le web service par le serveur il faut le placer dans le dossier C:applications/mamp/htdocs
- **sur Linux:**
  Pour exécuter le web service par le serveur il faut le placer dans le dossier /var/www/html/

Partie Android (Client)

Etape 1: Ajouter la permission de la connexion internet dans le manifest.xml

Il faut ajouter la permission d'utiliser l'internet dans le fichier manifest.xml

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

Etape 2: Ecrire le code java qui communique avec le web service

Partie Android (Client)

Pour établir des connexions à des serveurs distants sous Android, vous pouvez utiliser plusieurs méthodes:

- **Sockets TCP/IP :** on peut utiliser la classe Socket pour les connexions TCP/IP, si on veut établir une connexion directe avec un serveur distant sur un port spécifique.
- **WebSockets** : Si on a besoin d'une connexion bidirectionnelle persistante entre votre application Android et un serveur, on peut utiliser **WebSockets**. Il existe plusieurs bibliothèques WebSocket disponibles pour Android, comme **OkHttp WebSocket, AndroidAsync**, etc.

Partie Android (Client)

Pour établir des connexions à des serveurs distants sous Android, vous pouvez utiliser plusieurs méthodes:

- **API HttpURLConnection :** HttpURLConnection est une classe Java standard qui vous permet d'envoyer et de recevoir des données via HTTP ou HTTPS.
- **Bibliothèques de communication réseau :** il existe des bibliothèques tierces populaires qui simplifient la communication réseau, telles que Retrofit, Volley, OkHttp, etc.

Communication distante via HTTP?

- La communication distante via HTTP sous Android fait référence à l'échange de données entre une application Android et un serveur distant en utilisant le protocole HTTP (Hypertext Transfer Protocol).
- Mettre en palace une architecture **Client/Serveur** afin de centraliser les données et permettre a plusieurs clients de les récupérer et d'y contribuer.

Partie Android (Client)

HTTP(S) avec HttpURLConnection ?

Etape 2: Ecrire le code java qui communique avec le web service

Pour communiquer avec le web service qui joue le rôle d'intémédiaire entre l'application Android et la base de données distante, nous allons utiliser des classes qui hérite de la classe **AsyncTask**. car cette classe permettra de communiquer avec le serveur par un thread en tache de fond afin d'éviter le blocage de l'application durant le temps de l'envoi des requêtes et réception des réponses à destination et en provenance du serveur.

Qu'est ce qu'une AsyncTask?

Une tache asynchrone est une classe abstraite qui fonctionne sur un thread en arrière plan et dont le résultat est publie sur le ThreadUI. Une tache asynchrone est définie par 3 types génériques, appelés: le paramètre, la progression et le résultat ainsi que 4 étapes:

- OnPreExecute
- doInBackground
- onProgressUpdate
- onPostExecute

Idéalement, les AsyncTasks sont utilisés pour des utilisations courtes (quelques secondes au plus). Pour des opérations de longues périodes, il est fortement recommande d'utiliser les différentes API comme Executor, threPoolExecutor et FutureTask.

Qu'est ce qu'une AsyncTask?

Les types génériques d'AsyncTask:

- Params: le type des paramètres envoyés à la tache lors de l' exécution.
- Progrès: le type des unités d'étape publiées lors du calcul de fond.
- Résultat: le type de résultat du calcul en arrière plan.

Qu'est ce qu'une AsyncTask?

- Le cycle de vie de l'AsyncTask:
  - OnPreExecute(): appelé sur le thread d'interface avant que la tache soit exécuté.
  - doInBackground(Param ...): invoqué sur le thread en arrière plan immédiatement après OnPreExecute. Cette étape est utilisée pour effectuer le calcul de fond qui peut prendre un certain temps. Les paramètres de la tache asynchrone sont passes à cette étape. le résultat du calcul doit retourne par cette étape et sera envoyé à la dernière étape..

Qu'est ce qu'une AsyncTask?

- Le cycle de vie de l'AsyncTask:
  - onProgressUpdate(Progress...): cette méthode est utilisée pour afficher toute forme de progression dans l'interface utilisateur tandis que le calcul de fond est toujours en cours d'exécution. Par exemple, il peut être utilisé pour animer une barre de progression ou afficher des logs.
  - onPostRExecute(Résultat):appelé sur le thread UI après la fin du calcul. Le résultat est transmis à cette étape en tant que paramètre.

HTTP(S) avec HttpURLConnection ?

- Pour établir la communication distante via HTTP sous Android, on doit:
  1. Etablir la connexion : Android doit établir une connexion réseau avec le serveur distant en utilisant la méthode openConnection() de la classe URL pour créer une instance de HttpURLConnection

```
URL url = new URL("https://test.com/api");
HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
```

  2. Créer la requête HTTP : l'application Android crée une requête HTTP pour spécifier l'action (POST,GET,…) à effectuer sur le serveur distant .

```
urlConnection.setRequestMethod("GET");
urlConnection.setRequestProperty("Content-Type", "application/json");
```

HTTP(S) avec HttpURLConnection ?

- Pour établir la communication distante via HTTP sous Android, on doit:

3**. Traiter la requête côté serveur**: Le serveur distant reçoit la requête HTTP de l'application Android et la traite en conséquence.

4. Générer la réponse HTTP :Une fois que le serveur distant a traité la requête, il génère une réponse HTTP pour renvoyer à l'application Android.
urlConnection.getResponseCode(); // Pour obtenir le code de statut de la réponse
Recevoir et traiter la réponse : L'application Android reçoit la réponse HTTP du serveur distant et la traite en conséquence.

```
InputStream inputStream = urlConnection.getInputStream();
BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
String line;
StringBuilder response = new StringBuilder();
while ((line = reader.readLine()) != null) {
   response.append(line);}
```

HTTPS & HTTP?

- « Google » force l'utilisation du protocole sécurisé « https ». Si on utilise au contraire le protocole « http » pour afficher une page web, on obtient le message d'erreur :

- Une des solutions consiste à ajouter dans le fichier « AndroidManifest.xml » de l'application, la section «application», le passage suivant : android:usesCleartextTraffic="true"

Bibliothèques de réseau tierces

- Les bibliothèques offrent différentes fonctionnalités et abstractions pour répondre à différents besoins de communication réseau dans les applications Android.

- Le choix de la bibliothèque dépend souvent de la complexité de vos besoins, de votre préférence de style de codage et de l'intégration avec d'autres bibliothèques de votre projet.

- bibliothèques tierces les plus populaires pour la communication réseau sous Android : OkHttp, Retrofit, Volley

On utilise **Retrofit** pour communiquer entre le web service et la base de données distante.

Bibliothèques de réseau tierces

- Retrofit

  C'est une bibliothèque réseau simple utilisée pour les transactions réseau. En utilisant cette bibliothèque , nous pouvons capturer de manière transparente la réponse JSON a partir du service Web/de l'API Web. C'est une bibliothèque simple et rapide pour récupérer et télécharger les données (JSON ou toute autre donnée structurée) via un service Web base sur REST.

Retrofit

- **Ajouter Retrofit à votre projet:**

Il faut ajouter la bibliothèque **Retrofit** dans le fichier **build.gradle** dans la section des dépendances :

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
```

- **Créer un modèle de données :**

```
public class Technologie {
   private int id;
   private String nom;
....
}
```

Retrofit

- **Définir une interface de service :**

Créez une interface qui spécifie les appels d'API que vous souhaitez effectuer. Chaque méthode de cette interface représente une requête HTTP vers l'API.

```
public interface myapi {
@POST("insert.php")
   Call<Technologie> adtech(@Field("nom") String nom);
...
}
```

Retrofit

- **Créer une instance de l'interface de service :**

```
myapi api=retrofit.create(myapi.class);
```

Retrofit

- **Utiliser la méthode enqueue() :**

La méthode enqueue() est une fonctionnalité de Retrofit permettant d'effectuer des appels asynchrones aux API REST.

```
call.enqueue(new Callback<Technologie>() {
  @Override
  public void onResponse(Call<Technologie> call, Response<Technologie> response) {
    if (response.isSuccessful()) {
      // La réponse de l'API est réussie, et vous pouvez utiliser les données de la réponse ici
      technologie tech = response.body();
    } else {
      // La requête a échoué, et vous devez gérer les erreurs ici
      // Par exemple, vous pouvez utiliser response.errorBody() pour obtenir plus de détails sur l'erreur
    }}
  @Override
  public void onFailure(Call<User> call, Throwable t) {
    // La requête a échoué à cause d'une erreur de réseau ou d'une autre erreur
    // Vous devez gérer les erreurs ici }});
```