**Cours : Acces à Internet en Android**

**Objectifs pédagogiques**

À la fin de ce module, l'étudiant saura :

* Effectuer des requêtes réseau HTTP avec OkHttpClient.
* Gérer les réponses (JSON, erreurs, timeout).
* Intégrer l’asynchronicité et la sécurité dans ses requêtes.
* Lire et écrire avec GET et POST.

**1. Introduction à OkHttp**

**OkHttp** est une bibliothèque Java/Android efficace, moderne et facile à utiliser pour effectuer des appels HTTP.  
Elle gère :

* Connexions HTTP/HTTPS
* Caching
* Redirections
* Gestions d'erreurs réseau
* Requêtes asynchrones

**2. Ajout de la dépendance dans Gradle**

Dans build.gradle (Module) :

implementation 'com.squareup.okhttp3:okhttp:4.12.0'

**3. Structure de base d’une requête GET**

OkHttpClient client = new OkHttpClient();

Request request = new

Request.Builder()

.url(<https://jsonplaceholder.typicode.com/posts/1>)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

Log.e("HTTP\_ERROR", "Erreur réseau : " + e.getMessage());

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

String jsonResponse = response.body().string();

Log.d("HTTP\_RESPONSE", jsonResponse);

} else {

Log.e("HTTP\_ERROR", "Code erreur : " + response.code());

}

}

});

✅ **Note :** .enqueue() permet de ne pas bloquer le thread principal (appel asynchrone).

**4. Effectuer une requête POST**

**Exemple : envoyer des données JSON**

OkHttpClient client = new OkHttpClient();

MediaType JSON = MediaType.get("application/json; charset=utf-8");

String json = "{\"title\":\"Cours OkHttp\",\"body\":\"Exemple POST\",\"userId\":1}";

RequestBody body = RequestBody.create(json, JSON);

Request request = new Request.Builder()

.url("https://jsonplaceholder.typicode.com/posts")

.post(body)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

Log.e("HTTP\_ERROR", "POST échoué : " + e.getMessage());

}

@Override

public void onResponse(Call call, Response response) throws IOException {

Log.d("HTTP\_RESPONSE", "POST réussi : " + response.body().string());

}

});

**5. Configurer les délais (timeout)**

OkHttpClient client = new OkHttpClient.Builder()

.connectTimeout(10, TimeUnit.SECONDS)

.writeTimeout(10, TimeUnit.SECONDS)

.readTimeout(30, TimeUnit.SECONDS)

.build();

**6. Exercice pratique simple**

Créer une application qui :

* Fait un appel GET vers https://jsonplaceholder.typicode.com/posts/1
* Affiche le titre dans une TextView
* Affiche un message d'erreur si pas de connexion

**7. Gestion de la permission Internet**

Dans le fichier AndroidManifest.xml :

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

**8. ✅ Bonnes pratiques**

* Toujours faire les appels réseau en **arrière-plan** (enqueue, jamais dans le thread principal).
* Utiliser Log.d() pour déboguer les réponses.
* Toujours vérifier response.isSuccessful() avant d’utiliser response.body().

**Résumé des classes principales**

| **Classe** | **Rôle** |
| --- | --- |
| OkHttpClient | Gère les connexions HTTP |
| Request | Contient l’URL et les entêtes |
| RequestBody | Corps de la requête (POST, PUT) |
| Response | Résultat du serveur |
| Callback | Réponse asynchrone (succès / échec) |

## ****Étapes et solution de l'exercice du point 6****

### 1. ****Structure du layout (activity\_main.xml)****

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<TextView

android:id="@+id/textViewTitle"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Chargement..."

android:textSize="18sp"

android:padding="8dp" />

</LinearLayout>

### 2. ****Code Java (MainActivity.java)****

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import android.widget.TextView;

import org.json.JSONObject;

import java.io.IOException;

import okhttp3.Call;

import okhttp3.Callback;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.Response;

public class MainActivity extends AppCompatActivity {

TextView textViewTitle;

OkHttpClient client;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

textViewTitle = findViewById(R.id.textViewTitle);

client = new OkHttpClient();

fetchPost();

}

private void fetchPost() {

String url = "https://jsonplaceholder.typicode.com/posts/1";

Request request = new Request.Builder()

.url(url)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

Log.e("HTTP\_ERROR", e.getMessage());

runOnUiThread(() -> textViewTitle.setText("Erreur réseau"));

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

final String responseBody = response.body().string();

try {

JSONObject json = new JSONObject(responseBody);

final String title = json.getString("title");

runOnUiThread(() -> textViewTitle.setText(title));

} catch (Exception e) {

runOnUiThread(() -> textViewTitle.setText("Erreur parsing JSON"));

}

} else {

runOnUiThread(() -> textViewTitle.setText("Réponse invalide"));

}

}

});

}

}

### ****Résultat attendu :****

L'application affiche dans la TextView :

sunt aut facere repellat provident occaecati excepturi optio reprehenderit

(le titre du post avec id=1).

### ****3. Code du serveur (****server.js****)****

const express = require('express');

const app = express();

const port = 3000;

// Middleware pour permettre le CORS (pour tests avec Android)

app.use((req, res, next) => {

res.setHeader("Access-Control-Allow-Origin", "\*");

next();

});

// Route GET /posts/1

app.get('/posts/1', (req, res) => {

res.json({

userId: 1,

id: 1,

title: "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",

body: "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\n" +

"reprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"

});

});

// Lancement du serveur

app.listen(port, () => {

console.log(`Serveur lancé sur http://localhost:${port}`);

});

### ****4. Test****

Démarrez le serveur :

node server.js

Puis testez avec votre navigateur ou Postman :  
<http://localhost:3000/posts/1>

### Android côté client

Remplacez l'URL dans votre code Android par :

String url = "http://10.0.2.2:3000/posts/1";

### 5. ****Code du serveur PHP (****server.php****)****

<?php

// Définir les en-têtes pour indiquer que la réponse est au format JSON

header('Content-Type: application/json');

// Route /posts/1

if ($\_SERVER['REQUEST\_METHOD'] == 'GET' && $\_SERVER['REQUEST\_URI'] == '/posts/1') {

// Réponse JSON avec les données du post

echo json\_encode([

"userId" => 1,

"id" => 1,

"title" => "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",

"body" => "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"

]);

} else {

// Si l'URL n'est pas /posts/1, renvoyer une erreur 404

http\_response\_code(404);

echo json\_encode(["error" => "Not Found"]);

}

?>

### 6. ****Configurer un serveur local (si nécessaire)****

Pour exécuter ce script PHP en local, vous pouvez utiliser un serveur intégré avec PHP :

php -S localhost:3000

Cela lancera un serveur sur http://localhost:3000.

### 7. ****Tester****

Accédez à l'URL suivante via votre navigateur ou Postman pour tester :

http://localhost:3000/posts/1

### 8. ****Android Client (Modifiez l'URL)****

Dans votre application Android, modifiez l'URL pour pointer vers votre serveur PHP :

String url = "http://10.0.2.2:3000/posts/1"