

HTTP

Introduction

HTTP (*Hypertext Transfer Protocol*) est un protocole de communication client-serveur développé pour le *World Wide Web*. HTTPS est la variante du HTTP sécurisée par l'usage des protocoles SSL ou TLS.

HTTP est un protocole de la couche application qui utilise le protocole TCP comme couche de transport. Un serveur HTTP utilise alors par défaut le port 80 (443 pour HTTPS).

Il est parfois nécessaire d'effectuer des requêtes HTTP vers un serveur distant.

AndroidManifest.xml

Afin d'effectuer des opérations réseau depuis l'application, il faut doit inclure les autorisations suivantes dans le fichier `AndroidManifest.xml` :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplicationhttp">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

WebView

Dans la plupart des cas, il est suffisant d'utiliser un navigateur Web standard, comme Chrome, pour fournir le contenu d'une page Web à l'utilisateur.

Sinon, les objets `WebView` permettent d'afficher un contenu Web dans le *layout* d'une activité. Evidemment, un `WebView` ne dispose pas des fonctionnalités des navigateurs classiques.

Pour en savoir plus : lire la [documentation sur le contenu Web](#).

Un objet de type `WebView` qui, comme son nom l'indique, est un object graphique fait pour visualiser une

page web. Cependant, les liens web appellent tout de même le navigateur, ce qui est assez gênant.

Afin de ne pas lancer le navigateur par défaut, il faut surcharger le client web afin de recoder la méthode agissant lorsqu'un lien est cliqué. La classe est très simple :

```
private class MyWebViewClient extends WebViewClient
{
    Context context;

    public MyWebViewClient(Context c)
    {
        context = c;
    }

    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url)
    {
        view.loadUrl(url);

        //Log.d("MyWebViewClient", "Chargement " + url);
        Toast.makeText(getApplicationContext(), "Chargement " + url, Toast.LENGTH_SHORT).show();

        return true;
    }
}
```

Du côté de l'activité, il faut remplacer le comportement du client web par défaut.

```
public class MainActivity extends Activity
{
    WebView wv = null;
    final private String url = "http://tvaira.free.fr/index.html";

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        //requestWindowFeature(Window.FEATURE_NO_TITLE); // enlève la barre de menu
        setContentView(R.layout.activity_main);

        wv = (WebView)findViewById(R.id.webView1);
        wv.setWebViewClient(new MyWebViewClient(this));
        wv.getSettings().setJavaScriptEnabled(true);
        wv.loadUrl(url);
    }

    private class MyWebViewClient extends WebViewClient
    {
        // ...
    }
}
```

On définit le *layout* `activity_main.xml` pour notre `WebView` :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent">
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"/>
</RelativeLayout>
```

Voir aussi : [Application hybride](#)

Requête HTTP

Android fournit le client `HttpURLConnection` pour effectuer des requêtes HTTP. Il s'appuie sur la classe abstraite `URLConnection`.

Le principe d'utilisation d'un `HttpURLConnection` est assez simple en soi :

```

URL url = new URL("http://tvaira.free.fr/index.html");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
try
{
    InputStream in = new BufferedInputStream(urlConnection.getInputStream());
    BufferedReader reader = new BufferedReader(new InputStreamReader(in));
    StringBuilder sb = new StringBuilder();
    String ligne;
    while((ligne = reader.readLine()) != null)
    {
        sb.append(ligne);
    }
    Log.v("HttpURLConnection", sb.toString());
}
finally
{
    urlConnection.disconnect();
}
```

Par contre, Android oblige à effectuer les opérations réseau sur un *thread* autre que le *thread* UI principal sinon une exception `NetworkOnMainThreadException` sera levée.

Voir aussi : [Tâches d'arrière-plan \[PDF\]](#)

Pour effectuer la requête et la traiter, on va donc utiliser une `AsyncTask`. L'activité comprend seulement une zone de texte `TextView` et un bouton pour envoyer la requête :

```

public class MainActivity extends AppCompatActivity
{
    private final String TAG = "HttpJSON";
    private Button boutonEnvoyer;
    private TextView texte;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
```

```

setContentView(R.layout.activity_main);

texte = (TextView) findViewById(R.id.texte);
texte.setText("");
boutonEnvoyer = (Button)findViewById(R.id.boutonEnvoyer);
boutonEnvoyer.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v)
    {
        AsyncTaskHTTP task = new AsyncTaskHTTP(MainActivity.this);
        task.execute("http://tvaira.free.fr/index.html");
    }
});
}

public class AsyncTaskHTTP extends AsyncTask<String, Void, String>
{
    private AppCompatActivity myActivity;

    public AsyncTaskHTTP(AppCompatActivity mainActivity)
    {
        myActivity = mainActivity;
    }

    @Override
    protected String doInBackground(String... strings)
    {
        URL url = null;
        HttpURLConnection urlConnection = null;
        String data = null;

        try
        {
            Log.d("AsyncTaskHTTP", "url := " + strings[0]);
            url = new URL(strings[0]);
            urlConnection = (HttpURLConnection) url.openConnection();

            int responseCode = urlConnection.getResponseCode();
            Log.d("AsyncTaskHTTP", "responseCode := " + responseCode);
            if (responseCode != HttpURLConnection.HTTP_OK)
            {
                return data;
            }

            InputStream in = new BufferedInputStream(urlConnection.getInputStream());
            BufferedReader reader = new BufferedReader(new InputStreamReader(in));
            StringBuilder sb = new StringBuilder();
            String ligne;
            while((ligne = reader.readLine()) != null)
            {
                sb.append(ligne);
            }
            data = sb.toString();
        }
        catch (MalformedURLException e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {

```

```

        e.printStackTrace();
    }
    finally
    {
        if (urlConnection != null)
            urlConnection.disconnect();
    }

    return data;
}

@Override
protected void onPostExecute(String s)
{
    if(s == null)
        return;

    texte.setText(s);
}
}
}

```

Il est aussi possible d'intégrer des métadonnées dans la partie *header* de la requête. Exemple en HTTPS :

```

@Override
protected String doInBackground(String... strings)
{
    URL url = null;
    HttpsURLConnection urlConnection = null;
    String data = null;

    try
    {
        Log.d("AsyncTaskHTTP", "url := " + strings[0]);
        url = new URL(strings[0]);
        urlConnection = (HttpsURLConnection) url.openConnection();

        urlConnection.setRequestProperty("Accept", "application/json");
        urlConnection.setRequestProperty("Authorization", "xxxxxxxx");

        int responseCode = urlConnection.getResponseCode();
        Log.d("AsyncTaskHTTP", "responseCode := " + responseCode);
        if (responseCode != HttpURLConnection.HTTP_OK)
        {
            return data;
        }

        InputStream in = new BufferedInputStream(urlConnection.getInputStream());
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        StringBuilder sb = new StringBuilder();
        String ligne;
        while((ligne = reader.readLine()) != null)
        {
            sb.append(ligne);
        }
        data = sb.toString();
    }
    catch (MalformedURLException e)

```

```

    {
        e.printStackTrace();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
    finally
    {
        if (urlConnection != null)
            urlConnection.disconnect();
    }

    return data;
}

```

Avant d'envoyer une requête HTTP, il est possible de vérifier si une connexion au réseau est possible :

```

private boolean estConnecteReseau()
{
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();
    if (networkInfo == null || !networkInfo.isConnected() || (networkInfo.getType() != ConnectivityManager.TYPE_WIFI && networkInfo.getType() != ConnectivityManager.TYPE_MOBILE))
    {
        return false;
    }
    return true;
}

```

Autres

Il existe d'autres bibliothèques pour effectuer des requêtes HTTP (plus simplement) :

- [OkHttp](#) développée par Square
- [Volley](#) développée par Android
- [Google HTTP Client](#) développée par Google

Pour illustrer l'utilisation des différentes bibliothèques, on va effectuer une requête vers un serveur openweathermap.org pour obtenir la météo d'une ville. Le résultat obtenu est au format JSON :

```
{
    "coord": {"lon": 4.83, "lat": 44.08},
    "weather": [{"id": 500, "main": "Rain", "description": "légère pluie", "icon": "10d"}],
    "base": "stations",
    "main": {"temp": 11.66, "feels_like": 5.58, "temp_min": 11, "temp_max": 13, "pressure": 1011, "humidity": 81},
    "visibility": 10000,
    "wind": {"speed": 8.2, "deg": 130},
    "rain": {"1h": 0.25},
    "clouds": {"all": 90},
    "dt": 1582969672,
    "sys": {"type": 1, "id": 6516, "country": "FR", "sunrise": 1582957146, "sunset": 1582997254},
    "timezone": 3600,
    "id": 3035679,
}
```

```
        "name":"Avignon",
        "cod":200
    }
```

Pour les tests, on va juste extraire la température ("temp") et le *timestamp* ("dt").

OkHttp

Voir : [OkHttp](#)

Dans `app/build.gradle` :

```
...
dependencies {
    ...
    implementation 'com.squareup.okhttp3:okhttp:3.4.1'
    ...
}
```

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener, Callback
{
    private final String TAG = "OkHttp";
    private Button boutonEnvoyer;
    private TextView texte;
    private final String APPID = "xxxxxxxxxx";
    private final OkHttpClient client = new OkHttpClient();

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        texte = (TextView) findViewById(R.id.texte);
        texte.setText("");
        boutonEnvoyer = (Button)findViewById(R.id.boutonEnvoyer);
        boutonEnvoyer.setOnClickListener(this);
    }

    @Override
    public void onClick(View view)
    {
        Log.d("OkHttp", "onClick");
        Request request = new Request.Builder().url("http://api.openweathermap.org/data/2.5/weather?q=Avignon,FR&units=metric&lang=fr&appid=" + APPID).build();
        client.newCall(request).enqueue(this);
    }

    @Override
    public void onFailure(Call call, IOException e)
    {
        runOnUiThread(new Runnable()
        {
            @Override
            public void run()
            {

```

```

        texte.setText("Erreur !");
    }
});

}

@Override
public void onResponse(Call call, Response response) throws IOException
{
    Log.d("OkHttp", "onResponse := " + response.message());
    if (!response.isSuccessful())
    {
        throw new IOException(response.toString());
    }

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

    runOnUiThread(new Runnable()
    {
        @Override
        public void run()
        {
            JSONObject json = null;
            try
            {
                json = new JSONObject(body);
                long timestamp = json.getLong("dt");
                Log.d("OkHttp", "timestamp := " + timestamp);
                Date date = new Date(timestamp*1000);
                String strDate = new java.text.SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(date);
                JSONObject payloadFields = json.getJSONObject("main");
                double temperature = payloadFields.getDouble("temp");
                Log.d("OkHttp", "Température := " + temperature + " °C");
                texte.setText(strDate + "\nTempérature : " + temperature + " °C");
            }
            catch (JSONException e)
            {
                e.printStackTrace();
            }
        }
    });
}
}

```

Il est aussi possible d'intégrer des métadonnées dans la partie *header* de la requête :

```

String url = "";
Request request = new Request.Builder().url(url)
    .addHeader("Accept", "application/json")
    .addHeader("Authorization", "xxxxx")
    .build();

```

Volley

Voir : [Volley](#)

Dans `app/build.gradle` :

```
...
dependencies {
    ...
    implementation 'com.android.volley:volley:1.1.0'
    ...
}
```

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener, Response.Listener<String>, Response.ErrorListener
{
    private final String TAG = "Volley";
    private Button boutonEnvoyer;
    private TextView texte;
    private final String APPID = "xxxxxxxxxxxx";

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        texte = (TextView) findViewById(R.id.texte);
        texte.setText("");
        boutonEnvoyer = (Button) findViewById(R.id.boutonEnvoyer);
        boutonEnvoyer.setOnClickListener(this);
    }

    @Override
    public void onClick(View view)
    {
        RequestQueue queue = Volley.newRequestQueue(this);
        StringRequest request = new StringRequest(Request.Method.GET, "http://api.openweathermap.org/data/2.5/weather?q=Avignon,FR&units=metric&lang=fr&appid=" + APPID, this, this);
        queue.add(request);
    }

    @Override
    public void onErrorResponse(VolleyError error)
    {
        texte.setText("Erreur !");
    }

    @Override
    public void onResponse(String response)
    {
        JSONObject json = null;
        try
        {
            json = new JSONObject(response);
            long timestamp = json.getLong("dt");
            Log.d("Volley", "timestamp := " + timestamp);
            Date date = new Date(timestamp*1000);
            String strDate = new java.text.SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(date);
            JSONObject payloadFields = json.getJSONObject("main");
            double temperature = payloadFields.getDouble("temp");
            Log.d("Volley", "Température := " + temperature + " °C");
            texte.setText(strDate + "\nTempérature : " + temperature + " °C");
        }
    }
}
```

```

        }
    catch (JSONException e)
    {
        e.printStackTrace();
    }
}
}

```

On peut directement recevoir un `JSONObject` en modifiant le code comme ceci :

```

public class MainActivity extends AppCompatActivity implements View.OnClickListener, Response.Listener<JSONObject>, Response.ErrorListener
{
    ...
    @Override
    public void onClick(View view)
    {
        RequestQueue queue = Volley.newRequestQueue(this);
        JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, "http://api.openweathermap.org/data/2.5/weather?q=Avignon,FR&units=metric&lang=fr&appid=" + APPID, null, this, this);
        queue.add(request);
    }

    @Override
    public void onResponse(JSONObject response)
    {
        try
        {
            long timestamp = response.getLong("dt");
            Log.d("Volley", "timestamp := " + timestamp);
            Date date = new Date(timestamp*1000);
            String strDate = new java.text.SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(date);
            JSONObject payloadFields = response.getJSONObject("main");
            double temperature = payloadFields.getDouble("temp");
            Log.d("Volley", "Température := " + temperature + " °C");
            texte.setText(strDate + "\nTempérature : " + temperature + " °C");
        }
        catch (JSONException e)
        {
            e.printStackTrace();
        }
    }
}

```

Il est aussi possible d'intégrer des métadonnées dans la partie *header* de la requête :

```

String url = "";
JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, null, this, this)
{
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError
    {
        Map<String, String> params = new HashMap<String, String>();
        params.put("Accept", "application/json");
        params.put("Authorization", "xxxxxx");
        return params;
    }
}

```

```
    }  
};
```

Google HTTP Client

Voir : [Google HTTP Client](#)

Dans `app/build.gradle` :

```
...  
  
android {  
    ...  
    packagingOptions {  
        exclude 'META-INF/DEPENDENCIES'  
    }  
}  
  
dependencies {  
    ...  
    implementation 'com.google.api-client:google-api-client:1.30.9'  
    ...  
}
```

```
public class MainActivity extends AppCompatActivity  
{  
    private final String TAG = "HttpJSON";  
    private Button boutonEnvoyer;  
    private TextView texte;  
    private final String APPID = "xxxxxxxxxxxx";  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState)  
    {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        texte = (TextView) findViewById(R.id.texte);  
        texte.setText("");  
        boutonEnvoyer = (Button) findViewById(R.id.boutonEnvoyer);  
        boutonEnvoyer.setOnClickListener(new View.OnClickListener()  
        {  
            public void onClick(View v)  
            {  
                AsyncMeteoJSON task = new AsyncMeteoJSON(MainActivity.this);  
                task.execute("http://api.openweathermap.org/data/2.5/weather");  
            }  
        });  
    }  
  
    public class AsyncMeteoJSON extends AsyncTask<String, Void, JSONObject>  
    {  
        private AppCompatActivity myActivity;  
  
        public AsyncMeteoJSON(AppCompatActivity mainActivity)  
        {  
            myActivity = mainActivity;
```

```

    }

    @Override
    protected JSONObject doInBackground(String... strings)
    {
        JSONObject json = null;
        try
        {
            HttpTransport httpTransport = new NetHttpTransport();
            HttpRequestFactory requestFactory = httpTransport.createRequestFactory();
            GenericUrl url = new GenericUrl(strings[0]);
            url.put("q", "Avignon,FR");
            url.put("units", "metric");
            url.put("lang", "fr");
            url.put("appid", APPID);
            HttpRequest request = null;
            request = requestFactory.buildGetRequest(url);
            HttpResponse httpResponse = request.execute();
            Log.d("AsyncMeteoJSON", "httpResponse := " + httpResponse.getStatusCode() + " " +
            httpResponse.getStatusMessage());
            if(httpResponse.getStatusCode() == 200)
            {
                try
                {
                    json = new JSONObject(httpResponse.parseAsString());
                }
                catch (JSONException e)
                {
                    e.printStackTrace();
                }
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }

            return json;
        }

        @Override
        protected void onPostExecute(JSONObject s)
        {
            if(s == null)
                return;

            JSONObject payloadFields = null;
            try
            {
                long timestamp = s.getLong("dt");
                Log.d("AsyncMeteoJSON", "timestamp := " + timestamp);
                Date date = new Date(timestamp*1000);
                String strDate = new java.text.SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(date);
                payloadFields = s.getJSONObject("main");
                double temperature = payloadFields.getDouble("temp");
                Log.d("AsyncMeteoJSON", "Température := " + temperature + " °C");
                texte.setText(strDate + "\nTempérature : " + temperature + " °C");
            }
            catch (JSONException e)

```

```
        {
            e.printStackTrace();
        }
    }
}
```

HttpURLConnection

```
public class MainActivity extends AppCompatActivity
{
    private final String TAG = "MeteoJSON";
    private Button boutonEnvoyer;
    private TextView texte;
    private final String APPID = "xxxxxxxxxxxxxxxxxxxxxx";

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        texte = (TextView) findViewById(R.id.texte);
        texte.setText("");
        boutonEnvoyer = (Button) findViewById(R.id.boutonEnvoyer);
        boutonEnvoyer.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
                AsyncMeteoJSON task = new AsyncMeteoJSON(MainActivity.this);
                task.execute("http://api.openweathermap.org/data/2.5/weather?q=Avignon,FR&units=metric&lang=fr&appid=" + APPID);
            }
        });
    }

    public class AsyncMeteoJSON extends AsyncTask<String, Void, JSONObject>
    {
        private AppCompatActivity myActivity;

        public AsyncMeteoJSON(AppCompatActivity mainActivity)
        {
            myActivity = mainActivity;
        }

        @Override
        protected JSONObject doInBackground(String... strings)
        {
            URL url = null;
            HttpURLConnection urlConnection = null;
            String dataJSON = null;

            try
            {
                Log.d("AsyncMeteoJSON", "url := " + strings[0]);
                url = new URL(strings[0]);
                urlConnection = (HttpURLConnection) url.openConnection();
                InputStream in = new BufferedInputStream(urlConnection.getInputStream());
            
```

```

        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        StringBuilder sb = new StringBuilder();
        String ligne;
        while((ligne = reader.readLine()) != null)
        {
            sb.append(ligne);
        }
        dataJSON = sb.toString();
    }
    catch (MalformedURLException e)
    {
        e.printStackTrace();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
    finally
    {
        if (urlConnection != null)
            urlConnection.disconnect();
    }

    JSONObject json = null;
    try
    {
        json = new JSONObject(dataJSON);
    }
    catch (JSONException e)
    {
        e.printStackTrace();
    }

    return json;
}

@Override
protected void onPostExecute(JSONObject s)
{
    if(s == null)
        return;

    JSONObject payloadFields = null;
    try
    {
        long timestamp = s.getLong("dt");
        Log.d("AsyncMeteoJSON", "timestamp := " + timestamp);
        Date date = new Date(timestamp*1000);
        String strDate = new java.text.SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(date);
        payloadFields = s.getJSONObject("main");
        double temperature = payloadFields.getDouble("temp");
        Log.d("AsyncMeteoJSON", "Température := " + temperature + " °C");
        texte.setText(strDate + "\nTempérature : " + temperature + " °C");
    }
    catch (JSONException e)
    {
        e.printStackTrace();
    }
}

```

```
    }
```

© Thierry Vaira <tvaira@free.fr>