Ficheros en red

paco@portadaalta.es

Índice

Índice

WWW (World Wide Web)

- Aplicación más utilizada en Internet:
  - Simplifica el acceso a todo tipo de información
  - Interface unificado y fácil de usar

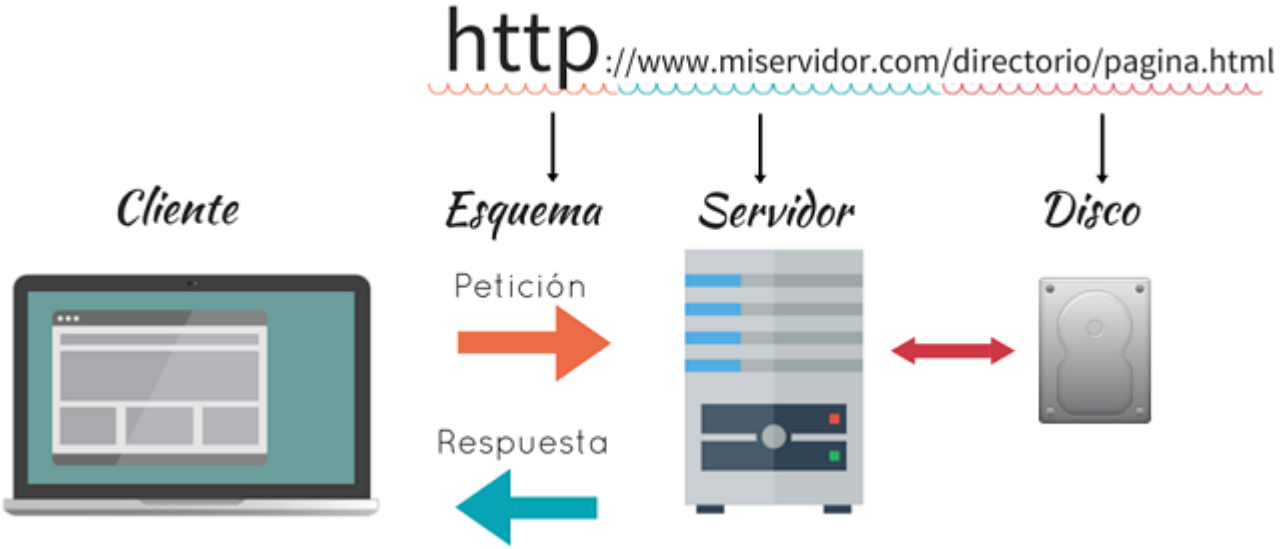
- WWW ofrece un sistema de acceso a información distribuida en miles de servidores distribuidos por toda Internet. Se puede navegar entre los diferentes documentos utilizando un sencillo sistema de hipervínculos.

- Utiliza una arquitectura cliente/servidor:
  - Navegador (Firefox, Chrome, . . .)
  - Servidor web (Apache, Nginx, IIS, . . .)

HTTP (HyperText Transfer Protocol)

- Protocolo utilizado para el intercambio de información entre el cliente y el servidor en la aplicación WWW

- Características:
  - El servidor se encuentra a la espera de conexiones de clientes a través de un puerto conocido (el 80 en HTTP)
  - Tras conectarse, el navegador solicita alguna petición al servidor. Esto lo realiza transmitiendo comandos codificados en ASCII



HTTP (HyperText Transfer Protocol)

A continuación describimos los pasos habituales que se siguen en una interacción del protocolo:

1) El usuario quiere acceder a la página http://www.upm.es/institucional/Estudiantes ,para lo cual pincha en un enlace de un documento HTML o introduciendo directamente en el campo Dirección del navegador Web.
2) El navegador averigua la dirección IP de www.upm.es.
3) El navegador establece una conexión con el puerto 80 de esta IP.
4) El navegador envía por esta conexión:

Cliente: GET institucional/Estudiantes HTTP/1.0
User-Agent: Mozilla Firefox v28.0
Host: www.upm.es
Accept: text/html, image/gif, image/jpeg

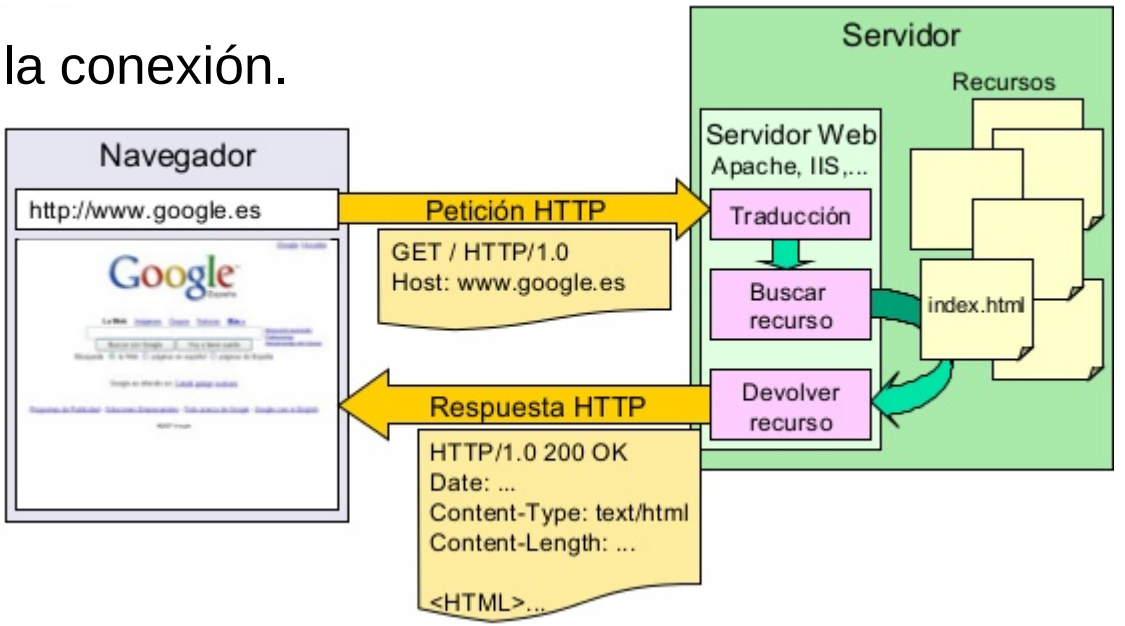
HTTP (HyperText Transfer Protocol)

5) El servidor devuelve la página a través de la conexión:

Servidor: HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Last-Modified: Mon, 22 Feb 2010 15:49:37 GMT
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>Página de ... </TITLE>
...
</HTML>

6) El servidor cierra la conexión.



Instalación del servidor web Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation:

"The Apache Software Foundation provides support for the Apache community of open-source software projects, which provide software products for the public good."

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado: netcraft.com

Instalación del servidor web Apache2:

```
sudo apt-get install apache2
```

Apache usa varios ficheros de configuración situados en "*/etc/apache2*".

Carpeta de archivos del servidor web (en */etc/apache2/sites-available/000-default.com*):

```
DocumentRoot /var/www/html
```

Códigos de error HTTP

Error 200: Ok.

Error 301: Moved Permanently.

Error 307: Moved Temporarily.

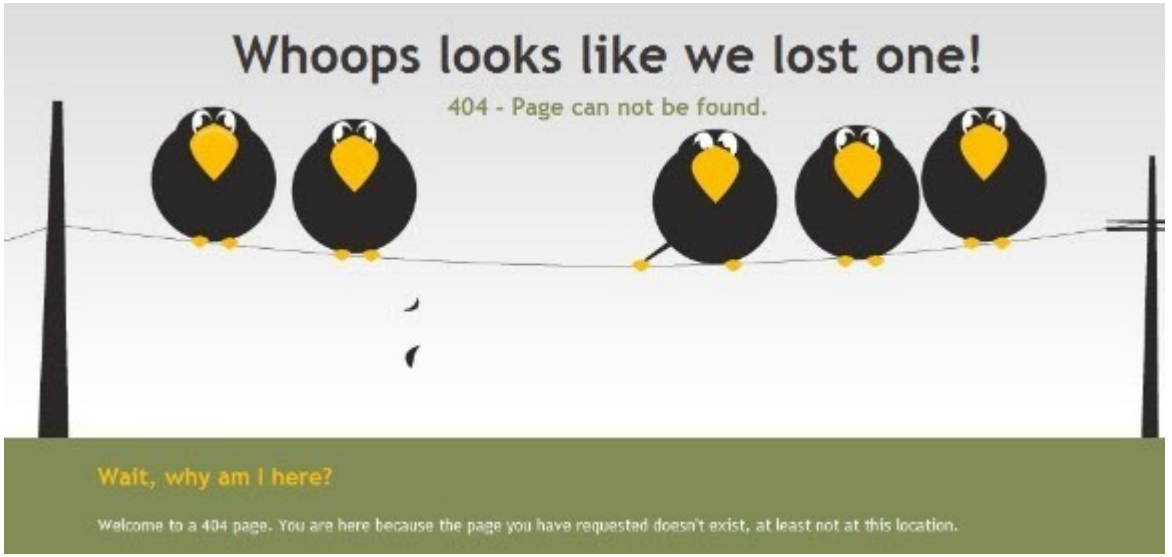
Error 400: Bad request

Error 401: Unauthorized

Error 403: Forbidden.

Error 404: Not found.

Error 500: Internal Server Error.



Utilizando HTTP desde Android

Android dispone de herramientas para utilizar el protocolo HTTP.

Para este propósito tenemos dos alternativas principales desde Android usando las bibliotecas:

- java.net.*
- org.apache.http.* (y apache.org)

El acceso a Internet desde la aplicación requiere establecer el permiso en el manifiesto

android.permission.INTERNET

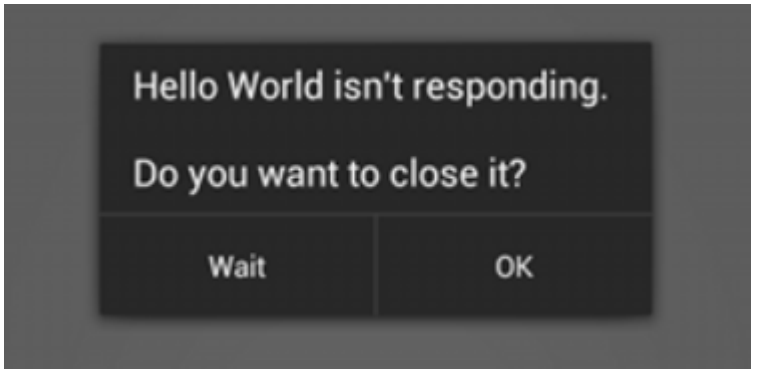
En caso que verifiquemos la conexión de nuestro sistema deberemos añadir además el permiso

android.permission.ACCESS_NETWORK_STATE.

Conexiones HTTP

El acceso a la web desde el hilo principal no es una buena práctica y desde la versión 3 de Android obtendremos un error, a menos que especifiquemos este comando en el método onCreate de la actividad principal:

```
StrictMode.setThreadPolicy(
    new  StrictMode.ThreadPolicy.Builder().permitNetwork().build());
```



HttpURLConnection

Esta es la clase estándar proporcionada por Java:

HttpURLConnection Class Overview:

```
URL url = new URL("http://www.android.com/");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
try {
        InputStream in = new BufferedInputStream(urlConnection.getInputStream());
        readStream(in);
} finally {
            urlConnection.disconnect();
}
```

HttpURLConnection

```
public static Resultado conectarJava(String texto) {
    URL url;
    HttpURLConnection urlConnection = null;
    int respuesta;
    Resultado resultado = new Resultado();

    try {
        url = new URL(texto);
        urlConnection = (HttpURLConnection) url.openConnection();
        respuesta = urlConnection.getResponseCode();
        if (respuesta == HttpURLConnection.HTTP_OK){
            resultado.setCodigo(true);
            resultado.setContenido(leer(urlConnection.getInputStream()));
        }
        else {
            resultado.setCodigo(false);
            resultado.setMensaje("Error en el acceso a la web: " + String.valueOf(respuesta));
        }
    } catch (IOException e) {
        resultado.setCodigo(false);
        resultado.setMensaje("Excepción: " + e.getMessage());
    } finally {
        try {
            if (urlConnection != null)
                urlConnection.disconnect();
        } catch (Exception e) {
            resultado.setCodigo(false);
            resultado.setMensaje("Excepción: " + e.getMessage());
        }
        return resultado;
    }
```

HttpURLConnection

```
private static String leer(InputStream entrada) throws IOException{
    BufferedReader in;
    String linea;
    StringBuilder miCadena = new StringBuilder();

    in = new BufferedReader(new InputStreamReader(entrada), 32000);
    while ((linea = in.readLine()) != null)
        miCadena.append(linea);
        //miCadena.append(linea).append('\n');
    in.close();

    return miCadena.toString();
}
```

Apache HTTP Client

A partir de la versión 6 de Android se ha quitado el soporte para Apache HTTP Client:

Apache HTTP Client Removal

```
android {
    useLibrary 'org.apache.http.legacy'
}
```

Uso de HTPP Client para Android:

Android Port

Apache HttpClient packages for Android maintained by Marek Sebera when targeting Android API 23 and newer:

```
dependencies {
    compile group: 'cz.msebera.android' , name: 'httpclient', version: '4.4.1.1'
}
```

Apache HTTP Client

Añadir a Gradle estas dependencias:

```
android {
    useLibrary 'org.apache.http.legacy'

}
dependencies {
    compile group: 'cz.msebera.android' , name: 'httpclient', version: '4.4.1.2'
}
```

Apache HTTP Client

El proyecto Apache HttpComponents™ es responsable de crear y mantener un conjunto de herramientas de componentes de bajo nivel en Java centrados en HTTP y protocolos asociados.

```
private String ConectarApache (String url){
	StringBuffer pagina = new StringBuffer();
	String line = "";

	try {
		HttpClient client = new DefaultHttpClient();
		HttpGet request = new HttpGet(url);
		HttpResponse response = client.execute(request);
		BufferedReader rd = new BufferedReader(
			new InputStreamReader(response.getEntity().getContent()));

		while ((line = rd.readLine()) != null)
			pagina.append(line);
	} catch (Exception e) {
		e.printStackTrace();
	}
	return pagina.toString();
}
```

Apache HTTP Client

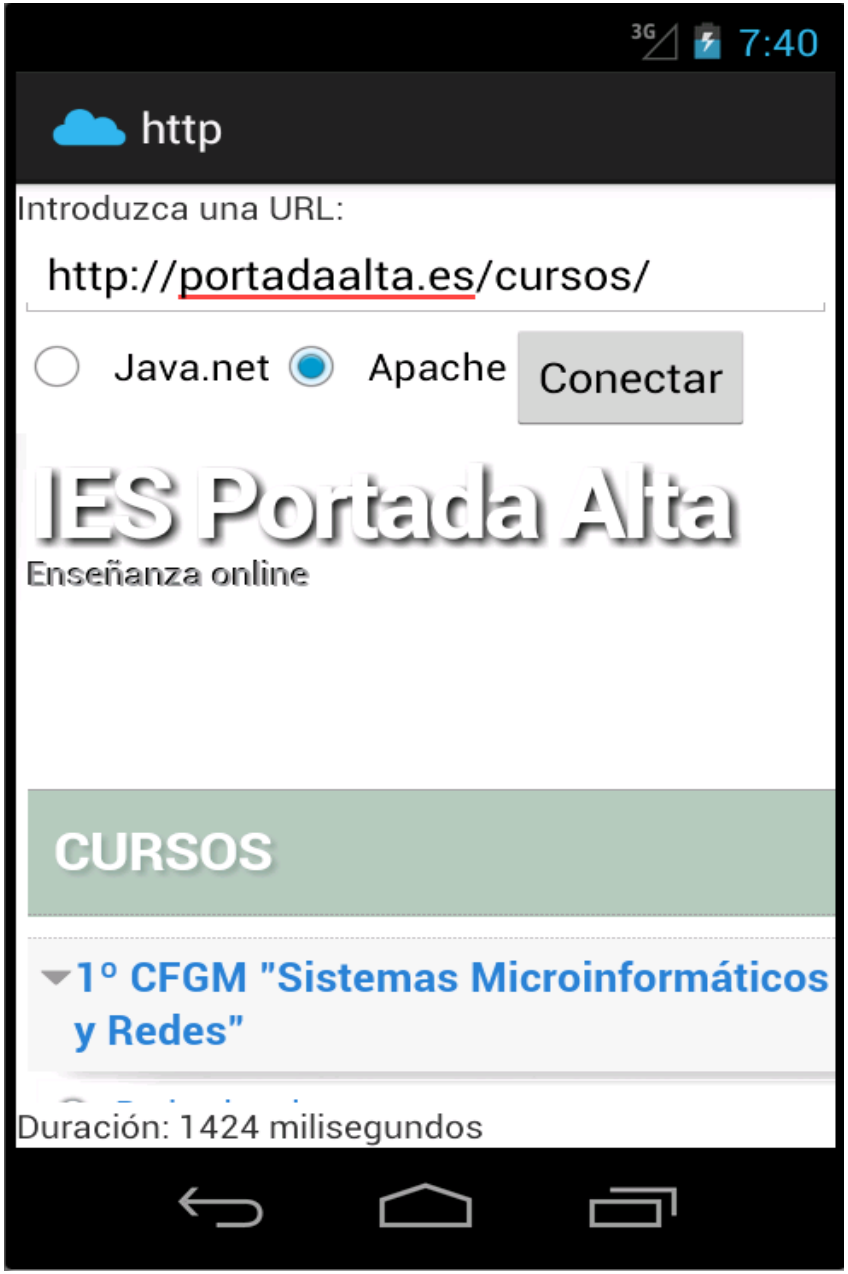
```
public static Resultado conectarApache(String texto) {
    CloseableHttpClient cliente = null;
    HttpPost peticion;
    HttpResponse respuesta;
    int valor;
    Resultado resultado = new Resultado();

    try {
        //cliente = new DefaultHttpClient();
        cliente = HttpClientBuilder.create().build();
        peticion = new HttpPost(texto);
        respuesta = cliente.execute(peticion);
        valor = respuesta.getStatusLine().getStatusCode();
        if (valor == HttpURLConnection.HTTP_OK) {
            resultado.setCodigo(true);
            resultado.setContenido(leer(respuesta.getEntity().getContent()));
        }
        else {
            resultado.setCodigo(false);
            resultado.setMensaje("Error en el acceso a la web: " + String.valueOf(valor));
        }
        cliente.close();
    } catch (IOException e) {
        resultado.setCodigo(false);
        resultado.setMensaje("Excepción: " + e.getMessage());
        if (cliente != null)
            try {
                cliente.close();
            } catch (IOException excep) {
                resultado.setCodigo(false);
                resultado.setMensaje("Excepción: " + excep.getMessage());
            }
    }

    return resultado;
}
```

Ejercicio de HTTP (I)

Crear una aplicación que pida una dirección web y muestre en pantalla la página indicada en esa dirección.

Se podrá elegir entre java.net y apache.org para realizar la conexión con el servidor web.

También se mostrará el tiempo que tarda en cargarse la página web.



Ejercicio de HTTP (I)

```
public class ConexionHTTP extends AppCompatActivity implements View.OnClickListener{
    EditText direccion;
    RadioButton radioJava, radioApache;
    Button conectar;
    WebView web;
    TextView tiempo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_conexion_aahc_okhttp);
        iniciar();
    }
    private void iniciar() {
        direccion = (EditText) findViewById(R.id.direccion);
        radioJava = (RadioButton) findViewById(R.id.radioJava);
        radioApache = (RadioButton) findViewById(R.id.radioApache);
        conectar = (Button) findViewById(R.id.conectar);
        conectar.setOnClickListener(this);
        web = (WebView) findViewById(R.id.web);
        tiempo = (TextView) findViewById(R.id.tiempo);
        StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder().permitNetwork().build());
    }
    @Override
    public void onClick(View v) {
        String texto = direccion.getText().toString();
        long inicio, fin;
        Resultado resultado;
        if (v == conectar) {
            inicio = System.currentTimeMillis();



        }
    }
}
```

No olvidar poner el permiso de Internet en el manifiesto:
<uses-permission android:name="android.permission.INTERNET"/>

Tareas asíncronas

En Android es muy frecuente lanzar nuevos hilos. Tendremos que hacerlo siempre que exista la posibilidad de que una tarea pueda bloquear el hilo del interfaz de usuario. Esto suela ocurrir en cálculos complejos o en accesos a la red. Es necesario crear nuevos hilos.

Hay una clase creada en Android que nos ayudará a resolver este tipo de problemas de forma sencilla, la clase AsyncTask.

Una tarea asíncrona (async task) se define por una acción que se ejecuta en un hilo secundario y cuyo resultado queremos que se publique en el hilo del interfaz de usuario.

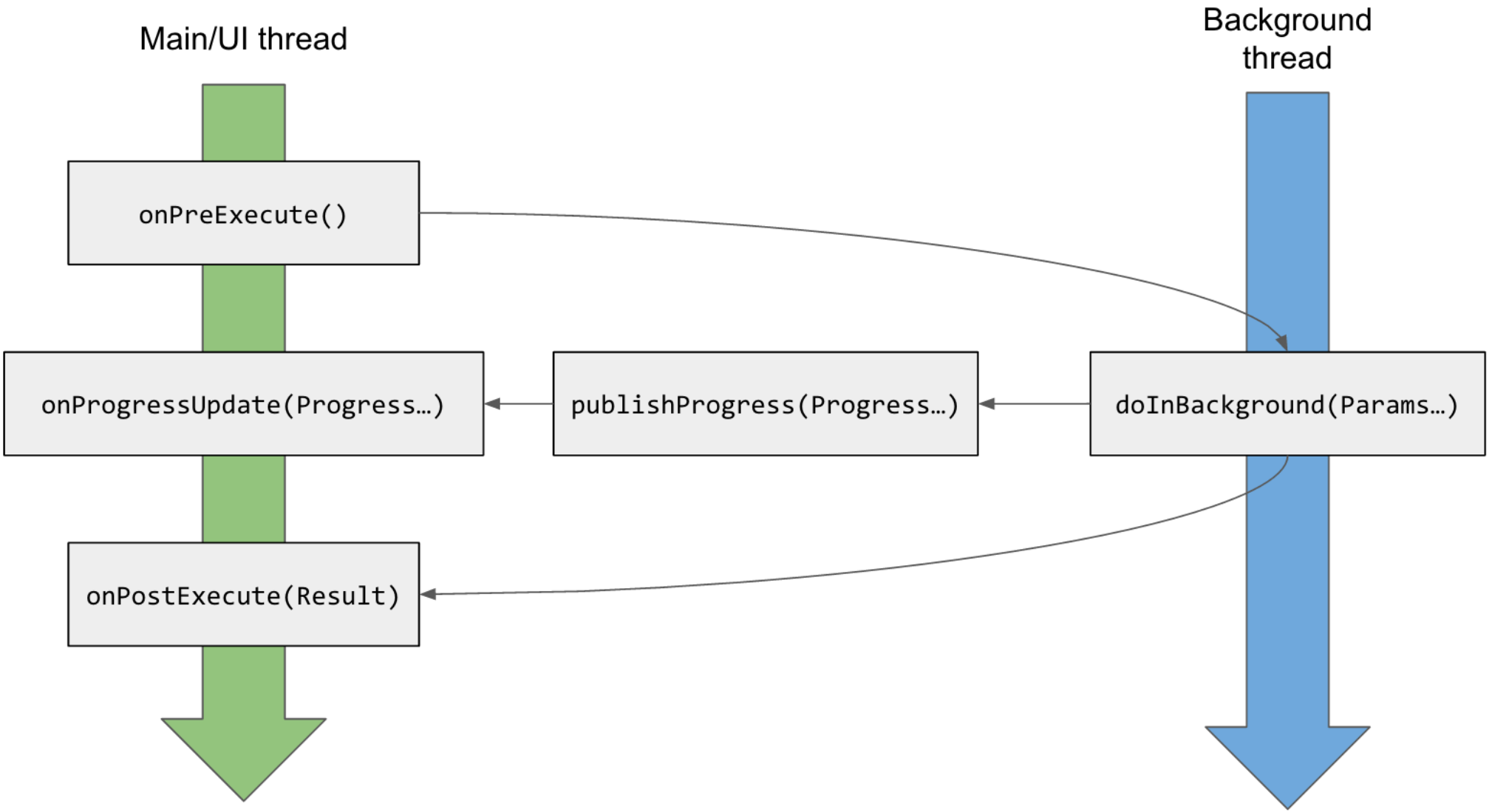
Tareas asíncronas

Para crear una nueva tarea asíncrona se puede usar el siguiente esquema:

```
public class MiTarea extends AsyncTask<Parametros, Progreso, Resultado> {

    protected void onPreExecute() {
        . . .
    }
    protected Resultado doInBackground(Parametros... param) {
        . . .
    }
    protected void onProgressUpdate(Progreso... prog) {
        . . .
    }
    protected void onPostExecute(Resultado result) {
        . . .
    }
}
```

donde Parametros, Progreso y Resultado han de ser reemplazados por nombres de clases según los tipos de datos con los que trabaje la tarea.

Tareas asíncronas



Tareas asíncronas

Los cuatro métodos que podemos sobreescribir corresponden a los cuatro pasos que seguirá AsyncTask para ejecutar la tarea:

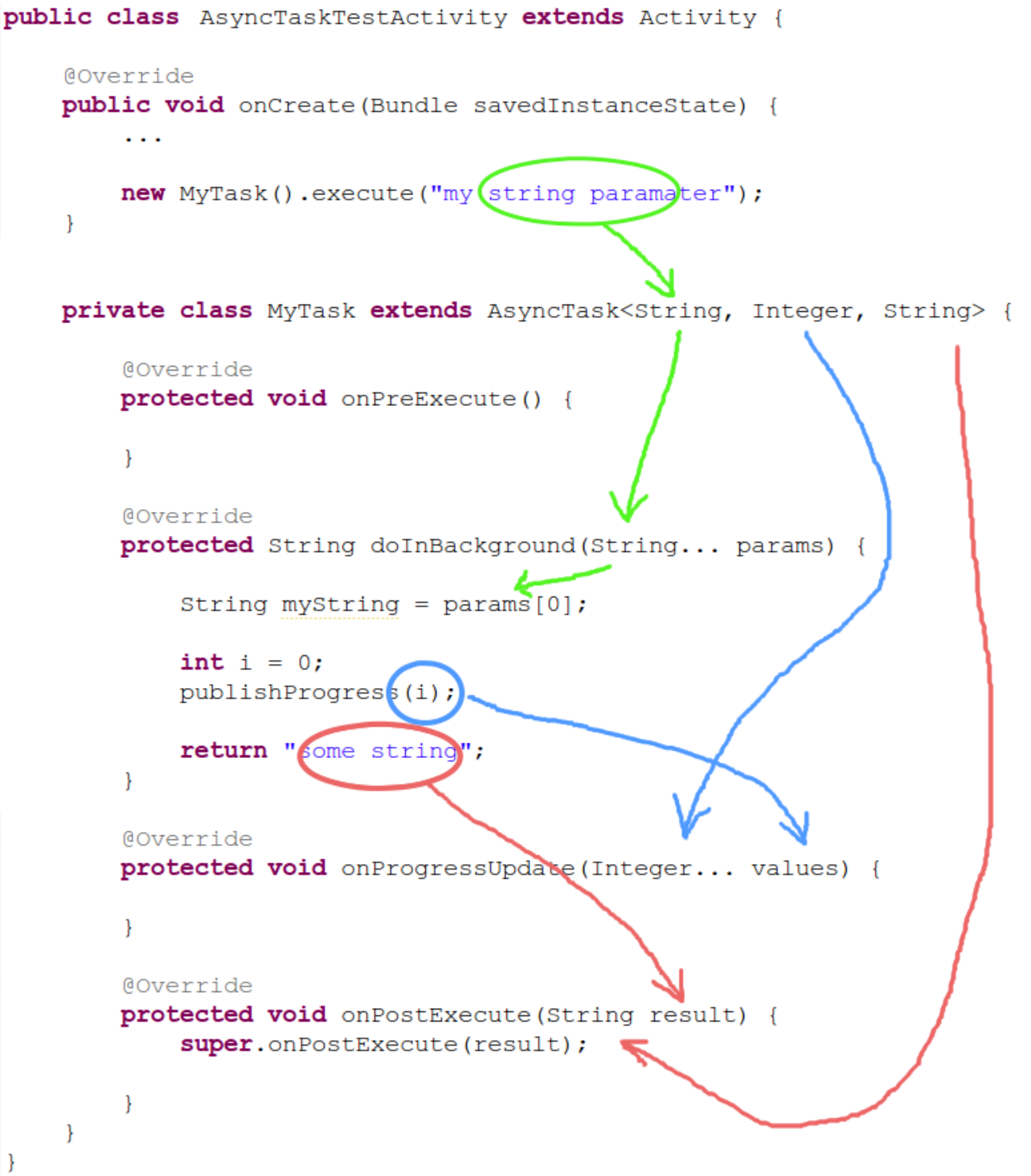
onPreExecute(): En este método tenemos que realizar los trabajos previos a la ejcución de la tarea. Se utiliza normalmente para configurar la tarea y para mostrar en la interfaz de usuario que empieza la tarea.

doInBackground(Parametros...): Es llamado cuando termina onPreExecute(). Es la parte más importante donde tenemos que realizar la tarea propiamente dicha. Es el único método de los cuatro que no se ejecuta en el hilo del interfaz de usuario. Lo va a hacer en un hilo nuevo creado para este propósito.

onProgressUpdate(Progress...): Este método se utiliza para mostrar el progreso de la tarea al usuario. Se ejecuta en el hilo interfaz de usuario, por lo que podremos interactuar con las vistas. El progreso de una determinada tarea ha de ser controlado por el programador llamando al método publishProgress(Progress ...) desde doInBackground().

onPostExecute(Result): Este método se usa para mostrar en el interfaz de usuario el resultado de la tarea. El parámetro de entrada (de la clase Result) corresponde con el objeto devuelto por el método doInBackground().

Tareas asíncronas



```
public class AsyncTaskTestActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...

        new MyTask().execute("my string paramater");
    }


    private class MyTask extends AsyncTask<String, Integer, String> {

        @Override
        protected void onPreExecute() {

        }

        @Override
        protected String doInBackground(String... params) {

            String myString = params[0];

            int i = 0;
            publishProgress(i);

            return "some string";
        }

        @Override
        protected void onProgressUpdate(Integer... values) {

        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);

        }
    }
}
```

Tareas asíncronas

Una vez definida la clase descendiente de AsyncTask podremos arrancar una tarea de la siguiente forma:

```
MiTarea tarea = new MiTarea();

tarea.execute(p1, p2, p3);
```

Donde p1, p2, p3 ha de ser una lista de objetos de la clase Parametros, pudiendo introducirse un número variable de parámetros.

Hay que resaltar que execute() es un método asíncrono. Esto significa que, tras ser llamado, se pondrá en marcha la tarea en otro hilo, pero en paralelo se continuarán ejecutando las instrucciones que hayas escrito a continuación de execute. El nombre de AsyncTask se ha puesto precisamente por este comportamiento.

Tareas asíncronas

Reglas

Se deben seguir varias reglas para que la clase de la tarea asíncrona funcione correctamente:

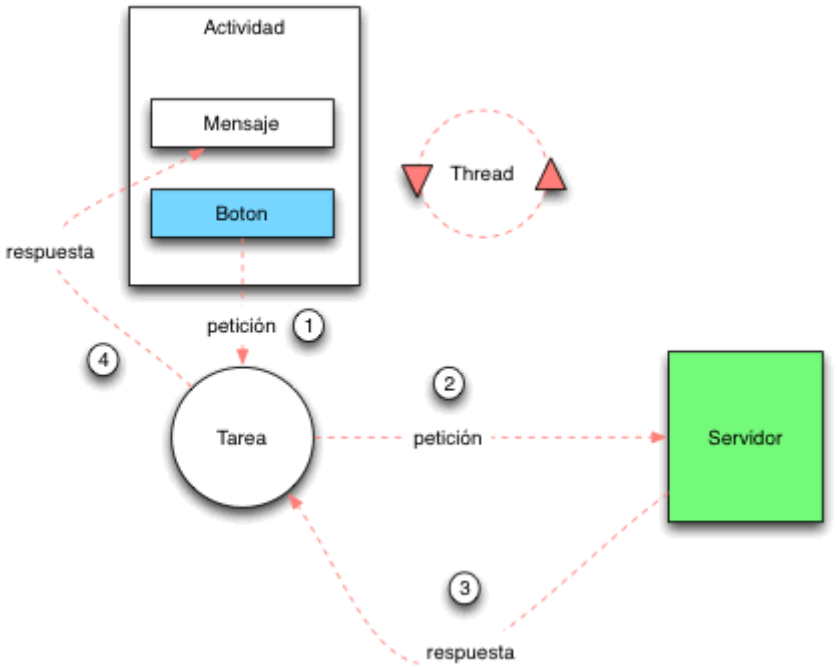
- La clase basada en AsyncTask se debe definir en el hilo principal (UI).
- La instancia de la tarea asíncrona debe ser creada en el hilo principal.
- El método execute(Params . . .) debe ser ejecutado en el hilo principal.
- No hay que llamar a los métodos onPreExecute(), onPostExecute(Result), doInBackground(Params...), onProgressUpdate(Progress...) manualmente.
- La tarea puede ejecutarse una sola vez (se lanza una excepción si se intenta ejecutar una segunda vez).

Tareas asíncronas

```
public class TareaAsincrona extends AsyncTask<String, Void, String> {
	private ProgressDialog progreso;

	protected void onPreExecute() {
		progreso = new ProgressDialog(MainActivity.this);
		progreso.setProgressStyle(ProgressDialog.STYLE_SPINNER);
		progreso.setMessage("Conectando . . .");
		progreso.setCancelable(false);
		progreso.show();
	}
	protected String doInBackground(String... cadena) {
		try {
			// operaciones en el hilo secundario
			. . .
		} catch (Exception e) {
			Log.e("HTTP", e.getMessage(), e);
			resultado = null;
			cancel(true);
		}
		return resultado;
	}
	protected void onPostExecute(String result) {
		progreso.dismiss();
		// mostrar el resultado
	}
	protected void onCancelled() {
		progreso.dismiss();
		// mostrar cancelación
	}
}
```

Tareas asíncronas

Comunicación con un servidor usando una tarea asíncrona:



Cancelación de una tarea asíncrona

Una tarea puede ser cancelada en cualquier momento llamando al método cancel(boolean). Esto causará que las siguientes llamadas a isCancelled() devolverán true.

Después de la llamada a este método, se ejecutará onCancelled(Object), en lugar de onPostExecute(Object), una vez que termine doInBackground(Object[]).

Para asegurarse que una tarea termina tan rápido como sea posible, siempre deberíamos comprobar periódicamente (en un bucle, por ejemplo) en doInBackground(Object[]) el valor devuelto por isCancelled().

Tareas asíncronas

Cancelación de una tarea asíncrona

```
public class TareaAsincrona extends AsyncTask<String, Integer, Resultado > {
	private ProgressDialog progreso;
	private Context context;

	public TareaAsincrona(Context context){
		this.context = context;
	}

	protected void onPreExecute() {
		progreso = new ProgressDialog(context);
		progreso.setProgressStyle(ProgressDialog.STYLE_SPINNER);
		progreso.setMessage("Conectando . . .");
		progreso.setCancelable(true);
		progreso.setOnCancelListener(new DialogInterface.OnCancelListener(){
			public void onCancel(DialogInterface dialog){
				TareaAsincrona.this.cancel(true);
			}
		});
		progreso.show();
	}
```

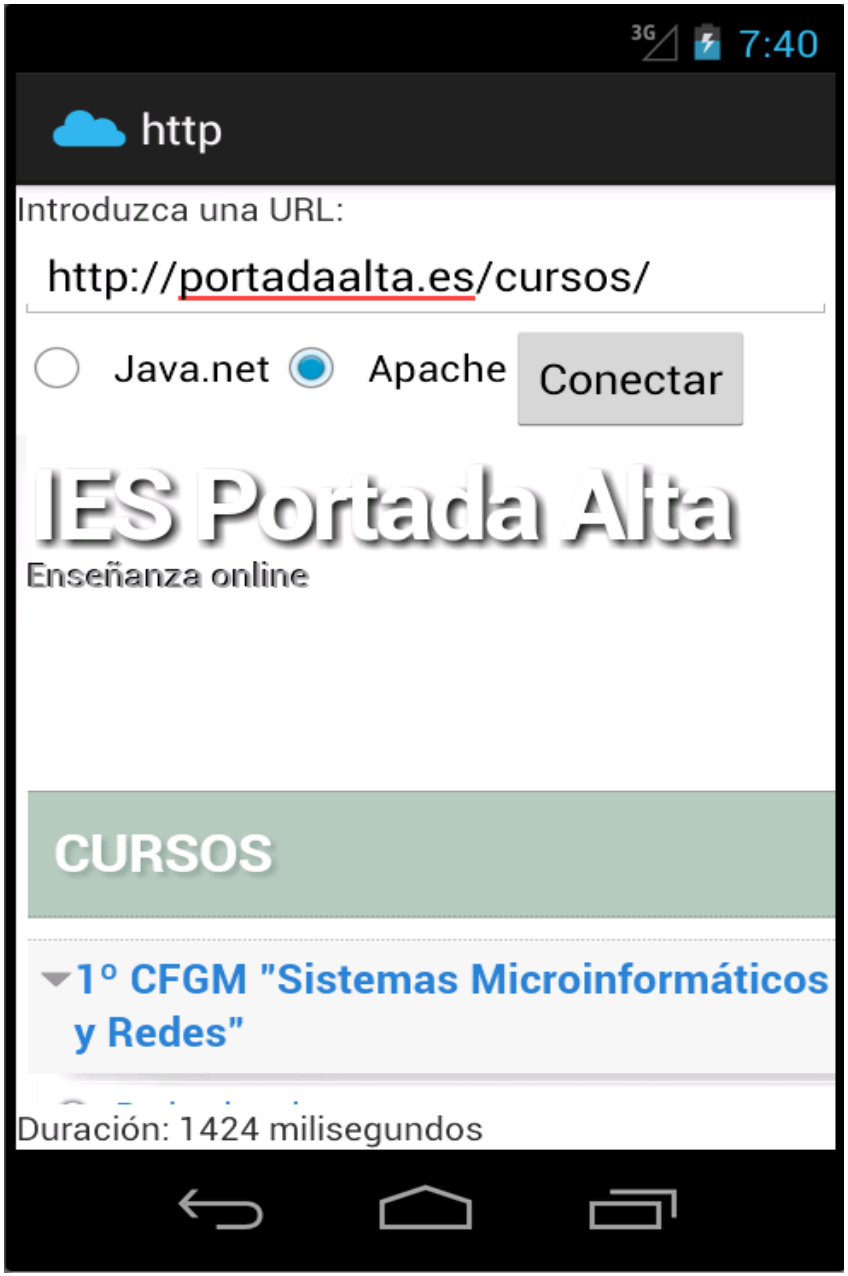
Tareas asíncronas

```
        protected Resultado doInBackground(String... cadena) {
                Resultado resultado;
                int i = 1;

                try {
                        // operaciones en el hilo secundario
                        publishProgress(i++);

                        . . .

                } catch (Exception e) {
                        . . .
                }
                return resultado;
        }
        protected void onProgressUpdate(Integer... progress) {
                progreso.setMessage("Conectando " + Integer.toString(progress[0]));
        }
        protected void onPostExecute(Resultado result) {
                progreso.dismiss();
                // mostrar el resultado
                . . .
        }
        protected void onCancelled() {
                progreso.dismiss();
                // mostrar cancelación
                . . .
        }
}
```

Ejercicio de HTTP (II)

Crear una aplicación que pida una dirección web y muestre en pantalla la página indicada en esa dirección.

Se usará una tarea asíncrona para realizar la conexión con el servidor web.



Android Asynchronous Http Client

A Callback-Based Http Client Library for Android

Overview

Download
version 1.4.9 (latest)

or fork me on github

An asynchronous callback-based Http client for Android built on top of Apache's HttpClient libraries. All requests are made outside of your app's main UI thread, but any callback logic will be executed on the same thread as the callback was created using Android's Handler message passing. You can also use it in Service or background thread, library will automatically recognize in which context is ran.

Features

- Using upstream HttpClient of version 4.3.6 instead of Android provided DefaultHttpClient
- Compatible with Android **API 23** and higher
- Make **asynchronous** HTTP requests, handle responses in **anonymous callbacks**
- HTTP requests happen **outside the UI thread**

Documentación: Package com.loopj.android.http

Android Asynchronous Http Client

Installation & Basic Usage

Add maven dependency using Gradle buildscript in format

```
dependencies {
        compile group: 'cz.msebera.android' , name: 'httpclient', version: '4.4.1.1'
        compile 'com.loopj.android:android-async-http:1.4.9'
}
```

Import the http package:

```
        import cz.msebera.android.httpclient.Header;
        import com.loopj.android.http.*;
```

Android Asynchronous Http Client

Create a new AsyncHttpClient instance and make a request with TextHttpResponseHandler():

```
AsyncHttpClient client = new AsyncHttpClient();
client.get("http://www.google.com", new TextHttpResponseHandler() {
        @Override
        public void onStart() {
        // called before request is started
        }
       @Override
       public void onSuccess(int statusCode, Header[] headers, String response) {
               // called when response HTTP status is "200 OK"
       }
       @Override
       public void onFailure(int statusCode, Header[] headers, String response, Throwable t) {
               // called when response HTTP status is "4XX" (eg. 401, 403, 404)
       }
   }
);
```

Android Asynchronous Http Client

Recommended Usage: Make a Static Http Client

In this example, we'll make a http client class with static accessors to make it easy to communicate with Twitter's API.

```
import com.loopj.android.http.*;

public class TwitterRestClient {
  private static final String BASE_URL = "https://api.twitter.com/1/";
  private static AsyncHttpClient client = new AsyncHttpClient();
  public static void get(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {
      client.get(getAbsoluteUrl(url), params, responseHandler);
  }
  public static void post(String url, RequestParams params, AsyncHttpResponseHandler responseHandler)
{
      client.post(getAbsoluteUrl(url), params, responseHandler);
  }
  private static String getAbsoluteUrl(String relativeUrl) {
      return BASE_URL + relativeUrl;
  }
}
```

Android Asynchronous Http Client

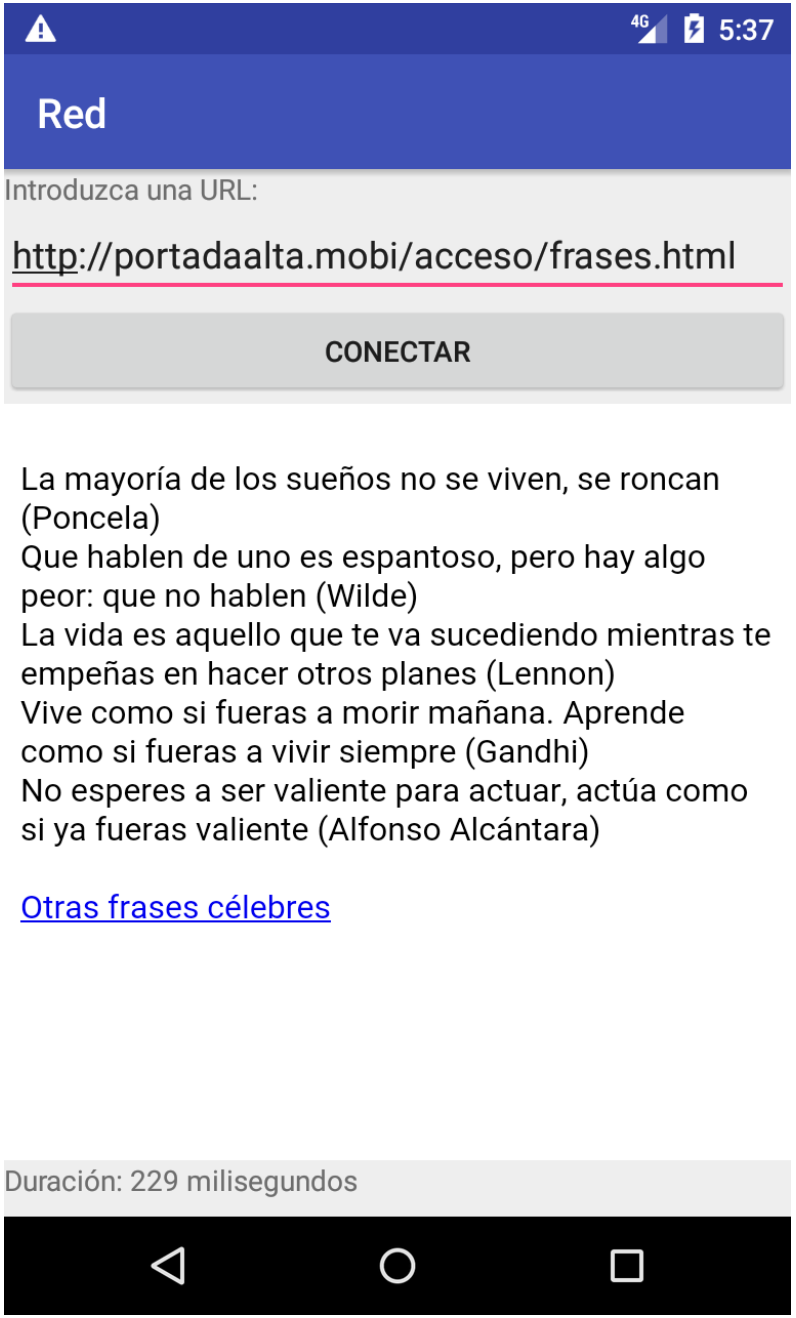
This then makes it very easy to work with the Twitter API in your code:

```
import org.json.*;
import com.loopj.android.http.*;

class TwitterRestClientUsage {
    public void getPublicTimeline() throws JSONException {
        TwitterRestClient.get("statuses/public_timeline.json", null, new JsonHttpResponseHandler() {
            @Override
            public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
                // If the response is JSONObject instead of expected JSONArray
            }

            @Override
            public void onSuccess(int statusCode, Header[] headers, JSONArray timeline) {
                // Pull out the first event on the public timeline
                JSONObject firstEvent = timeline.get(0);
                String tweetText = firstEvent.getString("text");
                // Do something with the response
                System.out.println(tweetText);
            }
        });
    }
}
```

Ejercicio de HTTP (III)

Crear una aplicación que pida una dirección web y muestre en pantalla la página indicada en esa dirección.

Se utilizará Android Asynchronous Http Client



Ejercicio de HTTP (III)

```
public class RestClient {
    private static final String BASE_URL = "";
    private static final int MAX_TIMEOUT = 2000;
    private static final int RETRIES = 3;
    private static final int TIMEOUT_BETWEEN_RETRIES = 5000;

    private static AsyncHttpClient client = new AsyncHttpClient(true, 80, 443);

    public static void get(String url, AsyncHttpResponseHandler responseHandler) {
        client.setTimeout(MAX_TIMEOUT);
        client.setMaxRetriesAndTimeout(RETRIES, TIMEOUT_BETWEEN_RETRIES);
        client.get(getAbsoluteUrl(url), responseHandler);
    }
    public static void get(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {
        client.setTimeout(MAX_TIMEOUT);
        client.setMaxRetriesAndTimeout(RETRIES, TIMEOUT_BETWEEN_RETRIES);
        client.get(getAbsoluteUrl(url), params, responseHandler);
    }
    public static void post(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {
        client.setTimeout(MAX_TIMEOUT);
        client.setMaxRetriesAndTimeout(RETRIES, TIMEOUT_BETWEEN_RETRIES);
        client.post(getAbsoluteUrl(url), params, responseHandler);
    }
    private static String getAbsoluteUrl(String relativeUrl) {
        return BASE_URL + relativeUrl;
    }
    public static void cancelRequests(Context c, boolean flag) {
        client.cancelRequests(c, flag);
    }
}
```

Ejercicio de HTTP (III)

```
private void AAHC() {
    final String texto = direccion.getText().toString();
    final long inicio;
    final long[] fin = new long[1];
    final ProgressDialog progreso = new ProgressDialog(ConexionAsincrona.this);

    inicio = System.currentTimeMillis();
    RestClient.get(texto, new TextHttpResponseHandler() {
        @Override
        public void onStart() {
            // called before request is started
            progreso.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            progreso.setMessage("Conectando . . .");
            //progreso.setCancelable(false);
            progreso.setOnCancelListener(new DialogInterface.OnCancelListener(){
                public void onCancel(DialogInterface dialog){
                    RestClient.cancelRequests(getApplicationContext(), true);
                }
            });
            progreso.show();
        }
```

Ejercicio de HTTP (III)

```
        @Override
        public void onSuccess(int statusCode, Header[] headers, String response) {
            // called when response HTTP status is "200 OK"
            fin[0] = System.currentTimeMillis();
            progreso.dismiss();



        }

        @Override
        public void onFailure(int statusCode, Header[] headers, String response, Throwable t) {
            // called when response HTTP status is "4XX" (eg. 401, 403, 404)
            fin[0] = System.currentTimeMillis();
            progreso.dismiss();



        }
    });
}
```

Volley

Transmitting Network Data Using Volley

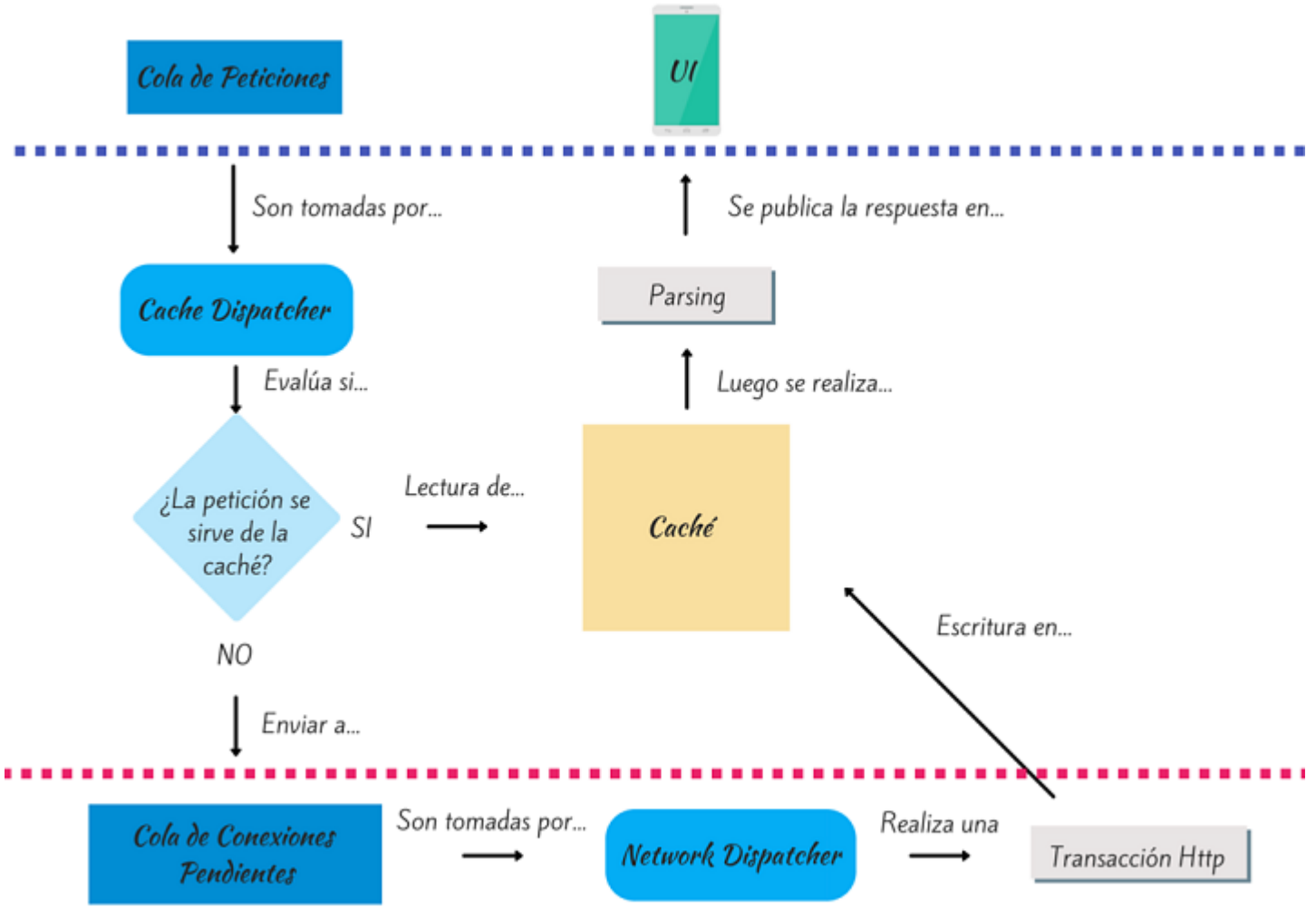
Volley offers the following benefits:

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching with standard HTTP cache coherence.

. . .

```
dependencies {
   ...
   compile 'com.android.volley:volley:1.0.0'
}
```

Volley


Cola de Peticiones
UI
Son tomadas por...
Se publica la respuesta en...
Cache Dispatcher
Parsing
Evalúa si...
Luego se realiza...
¿La petición se sirve de la caché?
SI
Lectura de...
Caché
Escritura en...
NO
Enviar a...
Cola de Conexiones Pendientes
Son tomadas por...
Network Dispatcher
Realiza una
Transacción Http

Volley

Use newRequestQueue

Volley provides a convenience method Volley.newRequestQueue that sets up a RequestQueue for you, using default values, and starts the queue. For example:

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...

// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        // Display the first 500 characters of the response string.
        mTextView.setText("Response is: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        mTextView.setText("That didn't work!");
    }
});
// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Volley

Use newRequestQueue

Handling Error Codes

```
@Override
public void onErrorResponse(VolleyError error) {

        String message = "";
        if (error instanceof TimeoutError || error instanceof NoConnectionError) {
                message = "Timeout Error "  + error.getMessage();
        } else if (error instanceof AuthFailureError) {
                message = "AuthFailure Error "  + error.getMessage();
        } else if (error instanceof ServerError) {
                message = "Server Error "  + error.getMessage();
        } else if (error instanceof NetworkError) {
                message = "Network Error "  + error.getMessage();
        } else if (error instanceof ParseError) {
                message = "Parse Error "  + error.getMessage();
        }
        Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
}
```

Volley

Use newRequestQueue

Handling Error Codes

```
@Override
public void onErrorResponse(VolleyError error) {
    String mensaje = "Error";
    if (error instanceof TimeoutError || error instanceof NoConnectionError)
        mensaje = "Timeout Error: "  + error.getMessage();
    else {
        NetworkResponse errorResponse = error.networkResponse;
        if (errorResponse != null && errorResponse.data != null)
            try {
                mensaje = "Error: " + errorResponse.statusCode + " " + "\n" + new
                     String(errorResponse.data, "UTF-8");
                Log.e("Error", mensaje);
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
    }
    Toast.makeText(getApplicationContext(), mensaje, Toast.LENGTH_SHORT).show();
}
```

Volley

Use newRequestQueue

Retry Policy

```
stringRequest.setRetryPolicy(new DefaultRetryPolicy(3000, 1, 1));
// Add the request to the RequestQueue.
mRequestQueue.add(stringRequest);
```

Volley

Cancel a Request

To cancel a request, call cancel() on your Request object.

Here is an example that uses a string value for the tag:

1. Define your tag and add it to your requests.

```
public static final String TAG = "MyTag";
StringRequest stringRequest; // Assume this exists.
RequestQueue mRequestQueue;  // Assume this exists.

// Set the tag on the request.
stringRequest.setTag(TAG);

// Add the request to the RequestQueue.
mRequestQueue.add(stringRequest);
```

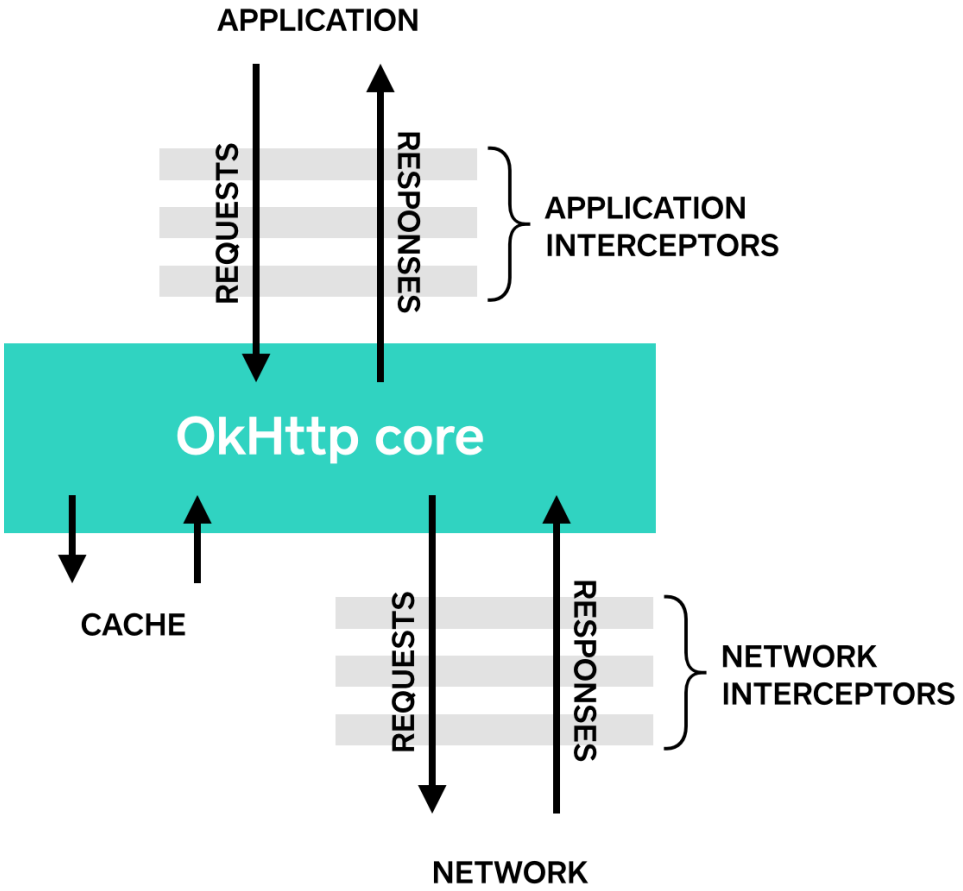
2. In your activity's onStop() method, cancel all requests that have this tag.

```
@Override
protected void onStop () {
    super.onStop();
    if (mRequestQueue != null) {
        mRequestQueue.cancelAll(TAG);
    }
});
```

Volley

Usar OkHttp como cliente HTTP

```
OkHttpClient myOkHttpClient = new OkHttpClient();

OkHttp3Stack myOkHttp3Stack = new OkHttp3Stack(myOkHttpClient);
mRequestQueue = Volley.newRequestQueue(this, myOkHttp3Stack);
```



Volley

Usar OkHttp como cliente HTTP

Descargar OkHttp3Stack y añadirlo en una nueva clase

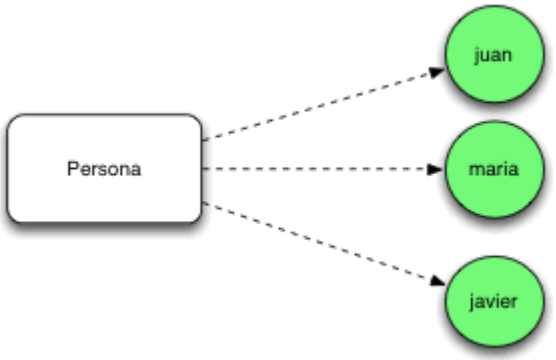
Configurar las dependencias:

```
android {
    compileSdkVersion 23
    buildToolsVersion "24.0.3"
    useLibrary 'org.apache.http.legacy'
    . . .

dependencies {
    compile 'com.squareup.okhttp3:okhttp:3.9.0'
    . . .
```

Volley

Patrón Singleton

Este patrón de diseño se encarga de que una clase determinada pueda tener únicamente un único objeto. Normalmente se pueden instanciar todos los objetos que se necesiten de una clase.



Sin embargo una clase que siga el patrón Singleton tiene la peculiaridad de que solo puede instanciar un único objeto.

Este tipo de clases son habituales en temas como configurar parámetros generales de la aplicación, ya que una vez instanciado el objeto, los valores se mantienen y son compartidos por toda la aplicación.

Volley

Use a Singleton Pattern

```
public class MySingleton {
    private static MySingleton mInstance;
    private RequestQueue mRequestQueue;
    private ImageLoader mImageLoader;
    private static Context mCtx;

    private MySingleton(Context context) {
        mCtx = context;
        mRequestQueue = getRequestQueue();
    }

    public static synchronized MySingleton getInstance(Context context) {
        if (mInstance == null) {
            mInstance = new MySingleton(context);
        }
        return mInstance;
    }

    public RequestQueue getRequestQueue() {
        if (mRequestQueue == null) {
            // getApplicationContext() is key, it keeps you from leaking the
            // Activity or BroadcastReceiver if someone passes one in.
            mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext());
        }
        return mRequestQueue;
    }

}
```

Volley

Use a Singleton Pattern

Modificación para usar OkHttp:

```
public RequestQueue getRequestQueue() {
    if (mRequestQueue == null) {
        // getApplicationContext() is key, it keeps you from leaking the
        // Activity or BroadcastReceiver if someone passes one in.
        //mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext());
        mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext(),
            new OkHttp3Stack(new OkHttpClient()));
    }
    return mRequestQueue;
}
```

Volley

```
public class MySingleton {
    private static MySingleton mInstance;
    private RequestQueue mRequestQueue;
    private static Context mCtx;

    private MySingleton(Context context) {
        mCtx = context;
        mRequestQueue = getRequestQueue();
    }

    public static synchronized MySingleton getInstance(Context context) {
        if (mInstance == null) {
            mInstance = new MySingleton(context);
        }
        return mInstance;
    }

    public RequestQueue getRequestQueue() {
        if (mRequestQueue == null) {
            // getApplicationContext() is key, it keeps you from leaking the
            // Activity or BroadcastReceiver if someone passes one in.
            //mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext());
            mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext(), new
OkHttp3Stack(new OkHttpClient()));
        }
        return mRequestQueue;
    }
}
```

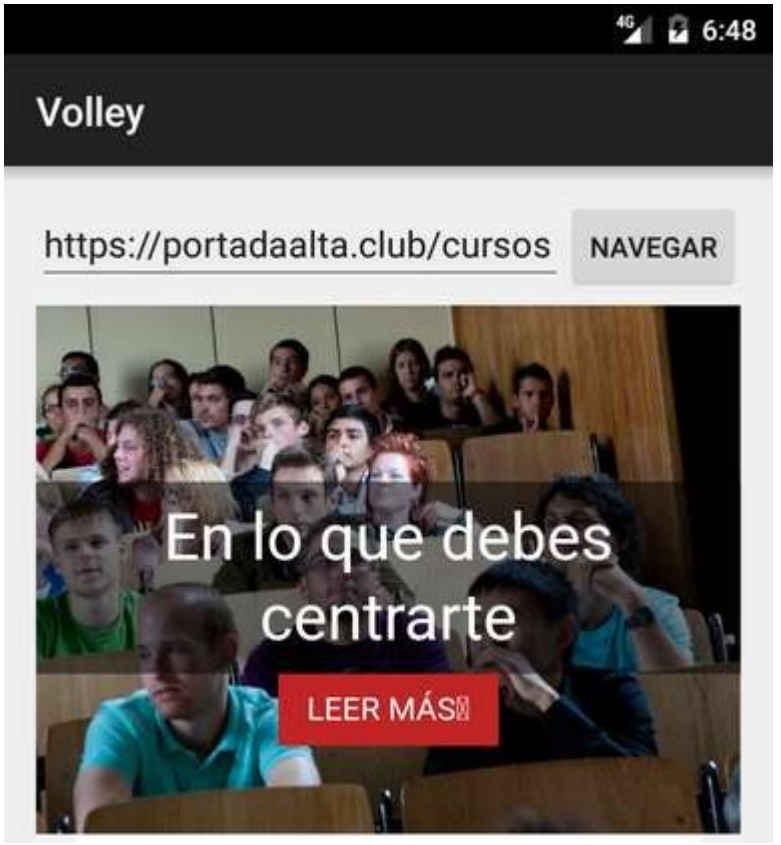
Volley

```
RequestQueue mRequestQueue;


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



     mRequestQueue = MySingleton.getInstance(this.getApplicationContext()).getRequestQueue();
     }
```

Ejercicio de HTTP (IV)

Crear una aplicación que pida una dirección web y muestre en pantalla la página indicada en esa dirección.

Se usará Volley para la comunicación con la red.



Ejercicio de HTTP (IV)

```
public class ConexionVolley extends AppCompatActivity implements View.OnClickListener {
    public static final String TAG = "MyTag";
    EditText mEditText;
    Button mButton;
    WebView mWebView;
    TextView mTextView;
    RequestQueue mRequestQueue;
    long inicio, fin;

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

        mEditText = (EditText) findViewById(R.id.editText);
        mButton = (Button) findViewById(R.id.button);
        mButton.setOnClickListener(this);
        mWebView = (WebView) findViewById(R.id.webView);
        mTextView = (TextView) findViewById(R.id.textView);
    }

    @Override
    public void onClick(View view) {
        String url;

        if (view == mButton) {
            url = mEditText.getText().toString();
            inicio = System.currentTimeMillis();
            makeRequest(url);
        }
    }
```

Ejercicio de HTTP (IV)

```
public void makeRequest(String url) {
    final String enlace = url;
    // Instantiate the RequestQueue.
    mRequestQueue = Volley.newRequestQueue(this);
    // Request a string response from the provided URL.
    StringRequest stringRequest = new StringRequest(Request.Method.GET,  url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {


                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {


                }
            });
    // Set the tag on the request.
    stringRequest.setTag(TAG);
    // Set retry policy
    stringRequest.setRetryPolicy(new DefaultRetryPolicy(3000, 1, 1));
    // Add the request to the RequestQueue.
    mRequestQueue.add(stringRequest);
}
@Override
protected void onStop() {
    super.onStop();
    if (mRequestQueue != null) {
        mRequestQueue.cancelAll(TAG);
    }
}

}
```

Ejercicio de HTTP (IV)

Uso del patrón Singleton

Añadir la clase OkHttp3Stack

Añadir la clase MySingleton

Modificar la creación de la cola en Volley:

```
RequestQueue mRequestQueue;
mRequestQueue = MySingleton.getInstance(this.getApplicationContext()).getRequestQueue();

@Override
protected void onCreate(Bundle savedInstanceState) {




    mRequestQueue = MySingleton.getInstance(this.getApplicationContext()).getRequestQueue();
}

public void makeRequest(String url) {
     // Instantiate the RequestQueue.
     //RequestQueue mRequestQueue = Volley.newRequestQueue(this);



}
@Override
protected void onStop() {

}
```

Descarga de imágenes

Mostrando imágenes remotas (The "Hard" Way)

Se descarga la imagen de la red, se convierten los bytes a un bitmap y se inserta en una imagen (imageView).

```
// 1. Declare a URL Connection
URL url = new URL("https://i.imgur.com/tGbaZCY.jpg");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
// 2. Open InputStream to connection
conn.connect();
InputStream in = conn.getInputStream();
// 3. Download and decode the bitmap using BitmapFactory
Bitmap bitmap = BitmapFactory.decodeStream(in);
in.close();
// 4. Insert into an ImageView
ImageView imageView = (ImageView) findViewById(R.id.imageView);
imageView.setImageBitmap(bitmap);
```

Descarga de imágenes

Mostrando imágenes remotas (The "Hard" Way)

Uso de una tarea asíncrona:

```
public class MainActivity extends AppCompatActivity {
    private ImageView ivBasicImage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ivBasicImage = (ImageView) findViewById(R.id.ivBasicImage);
        String url = "https://i.imgur.com/tGbaZCY.jpg";
        // Download image from URL and display within ImageView
        new ImageDownloadTask(ivBasicImage).execute(url);
    }

    // Defines the background task to download and then load the image within the ImageView
    private class ImageDownloadTask extends AsyncTask<String, Void, Bitmap> {
        ImageView imageView;

        public ImageDownloadTask(ImageView imageView) {
            this.imageView = imageView;
        }
```

Descarga de imágenes

```
    protected Bitmap doInBackground(String... addresses) {
       Bitmap bitmap = null;
       InputStream in;
       try {
          // 1. Declare a URL Connection
          URL url = new URL(addresses[0]);
          HttpURLConnection conn = (HttpURLConnection) url.openConnection();
          // 2. Open InputStream to connection
          conn.connect();
          in = conn.getInputStream();
          // 3. Download and decode the bitmap using BitmapFactory
          bitmap = BitmapFactory.decodeStream(in);
       } catch (IOException e) {
          e.printStackTrace();
       } finally {
         if(in != null)
           in.close();
       }
       return bitmap;
    }
    // Fires after the task is completed, displaying the bitmap into the ImageView
    @Override
    protected void onPostExecute(Bitmap result) {
       // Set bitmap image for the result
       imageView.setImageBitmap(result);
    }
  }
}
```

Descarga de imágenes

Mostrando imágenes remotas (The "Easy" Way)

Mostrar imágenes es mucho más fácil usando una biblioteca externa como Picasso, de Square, que descargará y almacenará en caché imágenes remotas.

```
String imageUri = "https://i.imgur.com/tGbaZCY.jpg";
ImageView ivBasicImage = (ImageView) findViewById(R.id.ivBasicImage);
Picasso.with(context).load(imageUri).into(ivBasicImage);
```

Descarga de imágenes

Picasso

Download v2.5.2

A powerful **image downloading** and **caching** library for Android

Introduction

Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!

```
Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);
```

Many common pitfalls of image loading on Android are handled automatically by Picasso:

- Handling `ImageView` recycling and download cancelation in an adapter.
- Complex image transformations with minimal memory use.
- Automatic memory and disk caching.

Descarga de imágenes

Uso:

Fichero AndroidManifest.xml:

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

Dependencia (añadir en app/build.gradle):

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

Descarga de imágenes

Image Transformations

Transform images to better fit into layouts and to reduce memory size.

```
Picasso.with(context)
  .load(url)
  .resize(50, 50)
  .rotate(90)
  .into(imageView)
```

Descarga de imágenes

Place Holders (Imágenes pre-descarga y post-error)

Picasso soporta poder mostrar ambos tipos de imágenes: las que se muestran antes de que se descargue la imagen real o la que se mostraría en caso de no poder descargarse.

```
Picasso.with(context)
    .load(url)
    .placeholder(R.drawable.user_placeholder)
    .error(R.drawable.user_placeholder_error)
    .into(imageView);
```

Una petición se realizará tres veces antes de mostrar la imagen post-error.

Descarga de imágenes

Carga de recursos

Recursos, assets, archivos, etc son soportados como fuentes de imágenes.

```
Picasso.with(context).load(R.drawable.landing_screen).into(imageView1);

Picasso.with(context).load("file:///android_asset/DvpvklR.png").into(imageView2);

Picasso.with(context).load(new File(...)).into(imageView3);
```

Descarga de imágenes

Picasso 2 OkHttp 3 Downloader

Usage

Create an OkHttp3Downloader instance wrapping your OkHttpClient or Call.Factory and pass it to downloader.

```
OkHttpClient client = // ...
Picasso picasso = new Picasso.Builder(context)
    .downloader(new OkHttp3Downloader(client))
    .build()
```

You can also use the the other constructors for a default OkHttpClient instance.

Download

Gradle:

```
compile 'com.jakewharton.picasso:picasso2-okhttp3-downloader:1.1.0'
```

Descarga de imágenes

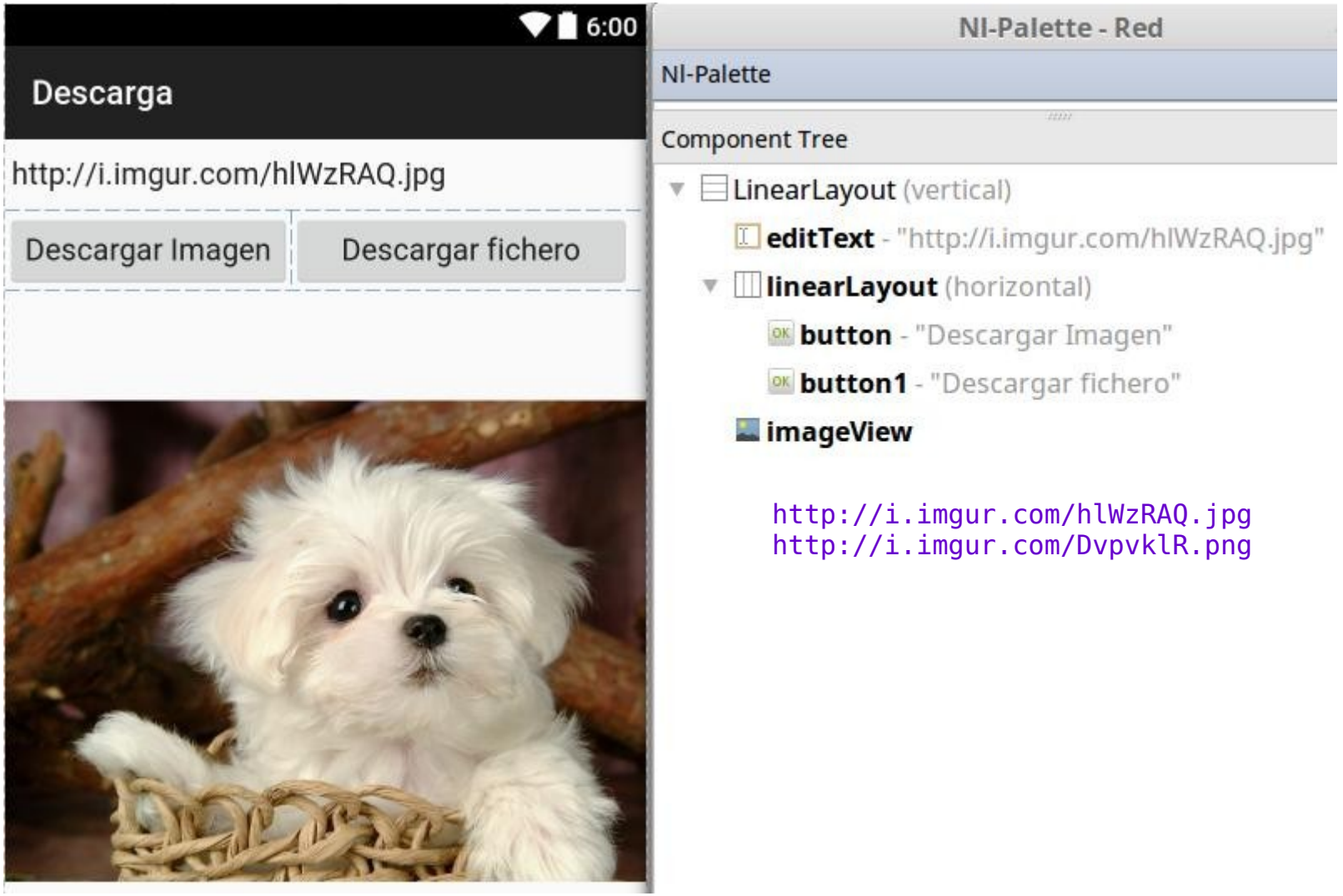
Migrate from Picasso 2.5.2 to Picasso 3.0.0

.with Without Context

Remove OkHttp 3 Hacks

Callback Interface Changes

Ejercicio: Descarga de imágenes




```
http://i.imgur.com/hlWzRAQ.jpg
http://i.imgur.com/DvpvklR.png
```

Ejercicio: Descarga de imágenes

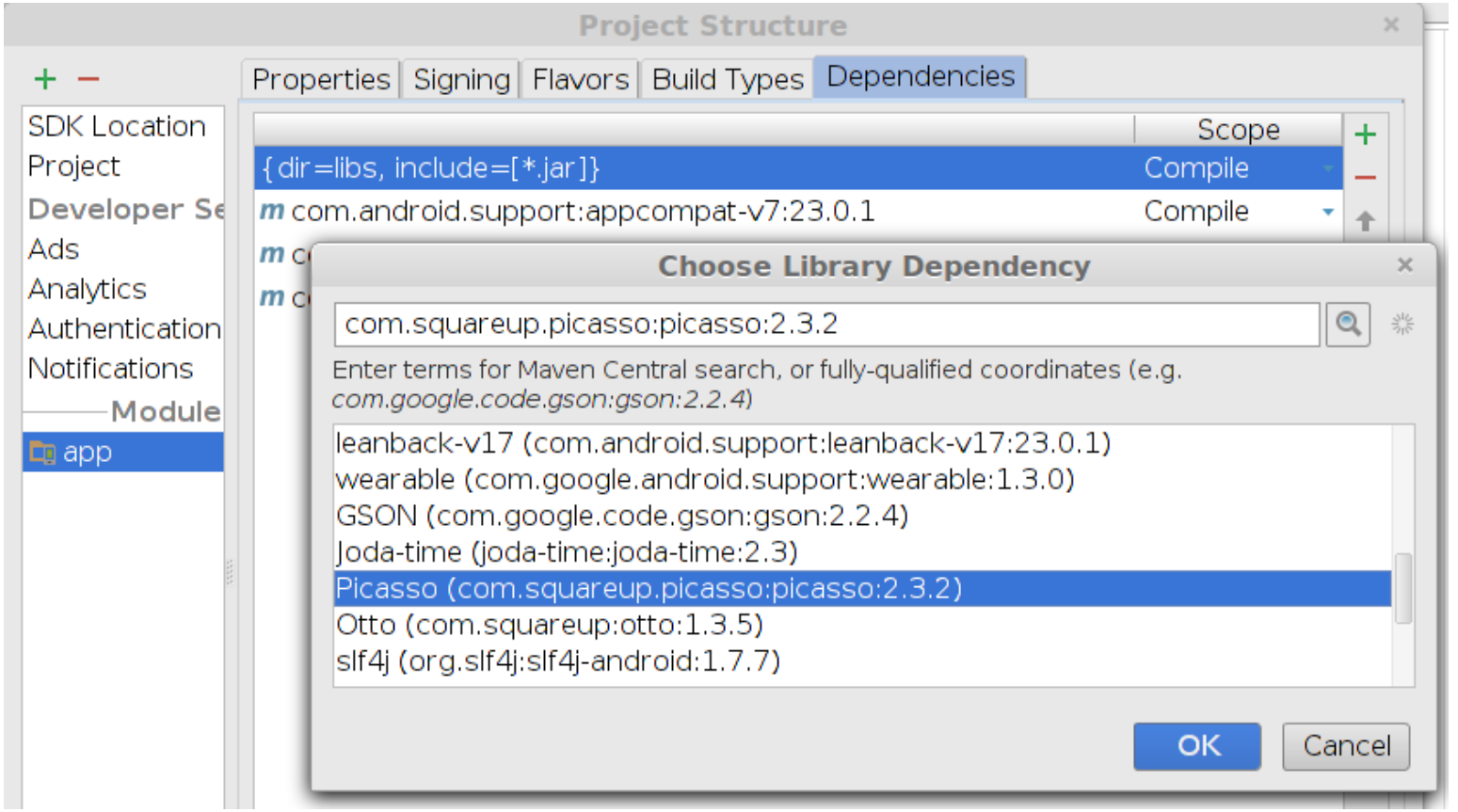
Instalar dependencias:

Añadir al fichero build.gradle (del módulo) la línea:

compile 'com.squareup.picasso:picasso:2.5.2'

Otra forma:

File /Project Structure . . .



Permisos:

No olvidar añadir el permiso de Internet en el manifiesto:

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

Ejercicio: Descarga de imágenes

```
public class Descarga extends AppCompatActivity implements OnClickListener {
    EditText texto;
    Button botonImagen;
    ImageView imagen;

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

        //http://i.imgur.com/hlWzRAQ.jpg
        texto = (EditText) findViewById(R.id.editText);
        botonImagen = (Button) findViewById(R.id.button);
        botonImagen.setOnClickListener(this);
        imagen = (ImageView) findViewById(R.id.imageView);
    }

    @Override
    public void onClick(View v) {
        String url = texto.getText().toString();

        if (v == botonImagen)
               descargaImagen(url);
    }

    private void descargaImagen(String url) {
        //utilizar OkHttp3



    }

}
```

Descargar ficheros con Android Asynchronous Http Client

Downloading Binary Data with FileAsyncHttpResponseHandler

The FileAsyncHttpResponseHandler class can be used to fetch binary data such as images and other files. For example:

```
AsyncHttpClient client = new AsyncHttpClient();
client.get("https://example.com/file.png", new FileAsyncHttpResponseHandler(/* Context */ this) {
        @Override
       public void onStart() {
        // Start

        }
        @Override
       public void onFailure(int statusCode, Header[] headers, Throwable throwable, File file) {
        // Failure

        }
        @Override
        public void onSuccess(int statusCode, Header[] headers, File response) {
        // Do something with the file `response`

        }
});
```

Descargar ficheros con Android Asynchronous Http Client

Downloading Binary Data with FileAsyncHttpResponseHandler

FileAsyncHttpResponseHandler

public FileAsyncHttpResponseHandler(java.io.File file)

Obtains new FileAsyncHttpResponseHandler and stores response in passed file

Parameters:
file - File to store response within, must not be null

FileAsyncHttpResponseHandler

public FileAsyncHttpResponseHandler(java.io.File file, boolean append)

Obtains new FileAsyncHttpResponseHandler and stores response in passed file

Parameters:
file - File to store response within, must not be null
append - whether data should be appended to existing file

See the FileAsyncHttpResponseHandler Javadoc for more information.

No olvidar poner el permiso de escritura en el manifiesto:
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

Ejercicio: Descarga de ficheros

Descargar un fichero de texto de un servidor web, guardarlo en la tarjeta de memoria y mostrar una notificación indicando si la descarga se ha realizado correctamente o no.

Ejercicio: Descarga de ficheros

```
public class Descarga extends AppCompatActivity implements OnClickListener {
    EditText texto;
    Button botonImagen, botonFichero;
    ImageView imagen;

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

        //http://i.imgur.com/hlWzRAQ.jpg
        texto = (EditText) findViewById(R.id.editText);
        botonImagen = (Button) findViewById(R.id.button);
        botonImagen.setOnClickListener(this);
        botonFichero = (Button) findViewById(R.id.button1);
        botonFichero.setOnClickListener(this);
        imagen = (ImageView) findViewById(R.id.imageView);
    }

    @Override
    public void onClick(View v) {
        String url = texto.getText().toString();
        if (v == botonImagen)
            descargaImagen(url);
        else if (v == botonFichero)
            descargaFichero(url);
    }

    private void descargaImagen(String url) {

    }
    private void descargaFichero(String url) {

    }
}
```

Subir ficheros con Android Asynchronous Http Client

Adding GET/POST Parameters with RequestParams

The RequestParams class is used to add optional GET or POST parameters to your requests. Create empty RequestParams and immediately add some parameters:

```
RequestParams params = new RequestParams();
params.put("key", "value");
params.put("more", "data");
```

Subir ficheros con Android Asynchronous Http Client

Uploading Files with RequestParams

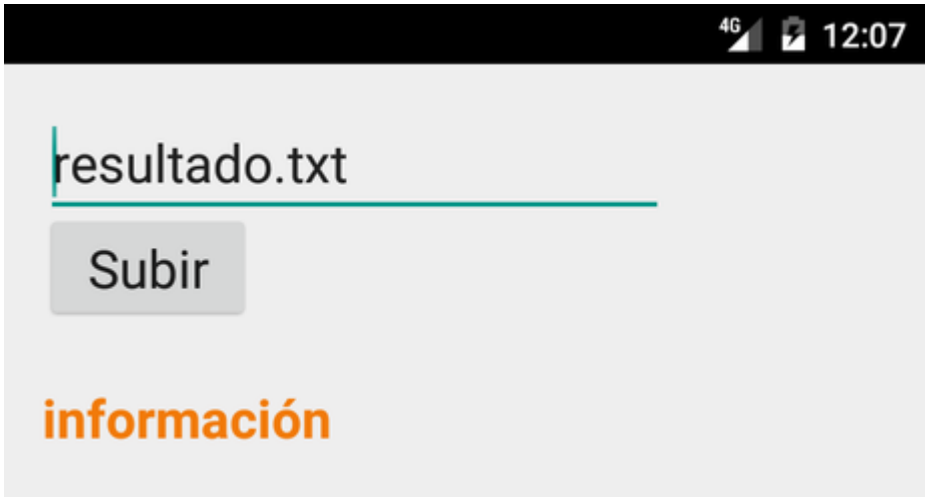
Add a File object to the RequestParams to upload:

```
File myFile = new File("/path/to/file.png");
RequestParams params = new RequestParams();
try {
        params.put("profile_picture", myFile);
} catch (FileNotFoundException e) {}

AsyncHttpClient client = new AsyncHttpClient();
client.post("https://myendpoint.com", params, responseHandler);
```

See the RequestParams Javadoc for more information.

Ejercicio: Subir un fichero

Se subirá un archivo en la tarjeta de memoria a una carpeta en el servidor web situado en el equipo local o en Internet.



Ejercicio: Subir un fichero

Servidor:

a) Instalar el soporte para php en el servidor web:

```
sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

b) Fichero **subir.html**:

```
<!DOCTYPE html>
<html>
<body>
<form action="upload.php" method="post" enctype="multipart/form-data">
    Select a file to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <br>
    <input type="submit" value="Upload file" name="submit">
</form>
</body>
</html>
```

Ejercicio: Subir un fichero

c) Fichero **upload.php** en el servidor web:

```
<?php
//create uploads and put permissions
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$FileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
// Check file size bigger than 100 kb
else if ($_FILES["fileToUpload"]["size"] > 100000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Allow certain file formats: JPG, PNG  or TXT
else if($FileType != "jpg" && $FileType != "png" && $FileType != "txt" ) {
        echo "Sorry, only JPG, PNG or TXT files are allowed.";
        $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "\nYour file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

Ayuda : Subida de ficheros

Ejercicio: Subir un fichero

d) Fichero **subidaFichero.java:**

```
public final static String WEB = "http://192.168.1.200/acceso/upload.php";

private void subida() {
    String fichero = texto.getText().toString();
    final ProgressDialog progreso = new ProgressDialog(SubidaFicheros.this);
    File myFile;
    Boolean existe = true;

    myFile = new File(Environment.getExternalStorageDirectory(), fichero);
    //File myFile = new File("/path/to/file.png");

    RequestParams params = new RequestParams();
    try {
        params.put("fileToUpload", myFile);
    } catch (FileNotFoundException e) {
        existe = false;
        informacion.setText("Error en el fichero: " + e.getMessage());
        //Toast.makeText(this, "Error en el fichero: " + e.getMessage(), Toast.LENGTH_SHORT).show();
    }
    if (existe)
        RestClient.post(WEB, params, new TextHttpResponseHandler() {
            @Override
            public void onStart() {
                // called before request is started
                progreso.setProgressStyle(ProgressDialog.STYLE_SPINNER);
                progreso.setMessage("Conectando . . .");
                //progreso.setCancelable(false);
                progreso.setOnCancelListener(new DialogInterface.OnCancelListener(){
                    public void onCancel(DialogInterface dialog){
                        RestClient.cancelRequests(getApplicationContext(), true);
                    }
                });
                progreso.show();
            }
```

Ejercicio: Subir un fichero

```
        @Override
        public void onSuccess(int statusCode, Header[] headers, String response) {
            // called when response HTTP status is "200 OK"
            progreso.dismiss();

        }

        @Override
        public void onFailure(int statusCode, Header[] headers, String response, Throwable t) {
            // called when response HTTP status is "4XX" (eg. 401, 403, 404)
            progreso.dismiss();

        }
    });
}
```

Ejercicio: Subir un fichero

¿Cuál es el tamaño máximo del fichero que se puede subir?

¿Cómo se puede mejorar la seguridad?

¿Dudas?

¿Sugerencias?

paco@portadaalta.es