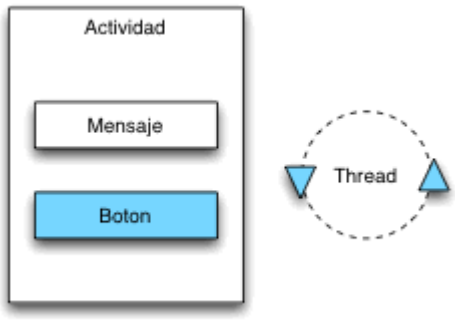
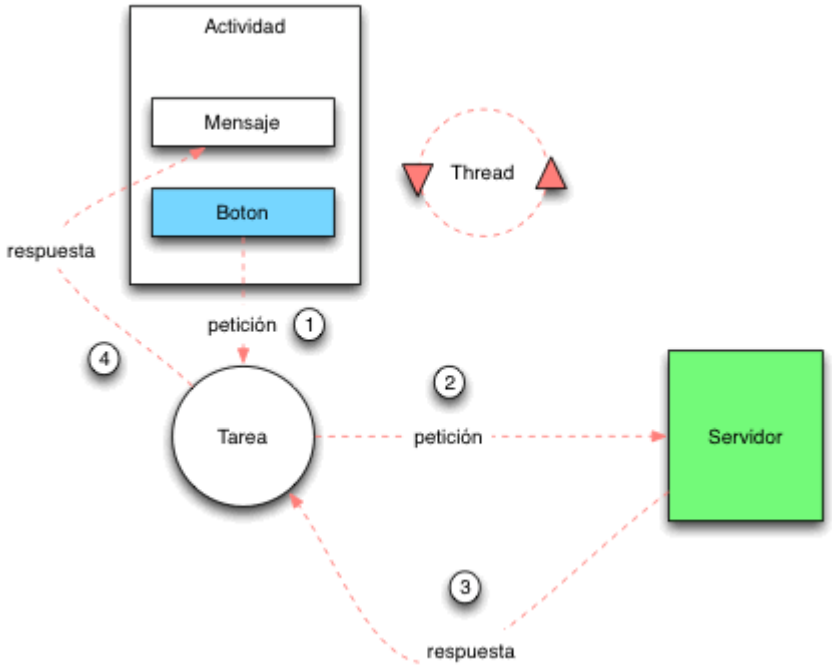
Una de las tareas mas habituales que tenemos que hacer cuando programamos aplicaciones Android es acceder a información ubicada de forma remota. Esta operación se suele realizar a traves de una petición Android REST. Al principio todo parece muy sencillo sin embargo cuando nos ponemos a programarlo las cosas se pueden complicar un poco ¿Porque? Vamos a intentar explicarlo. Supongamos que tenemos una actividad que contiene un boton que va a realizar una petición remota.



Recordemos que una Actividad dispone de un Thread (UI Thread) que se encarga de su correcto funcionamiento y de responder a las peticiones del Usuario . Si nosotros solicitamos que el botón realice una tarea que implique una petición remota nos podemos encontrar con la siguiente situación.



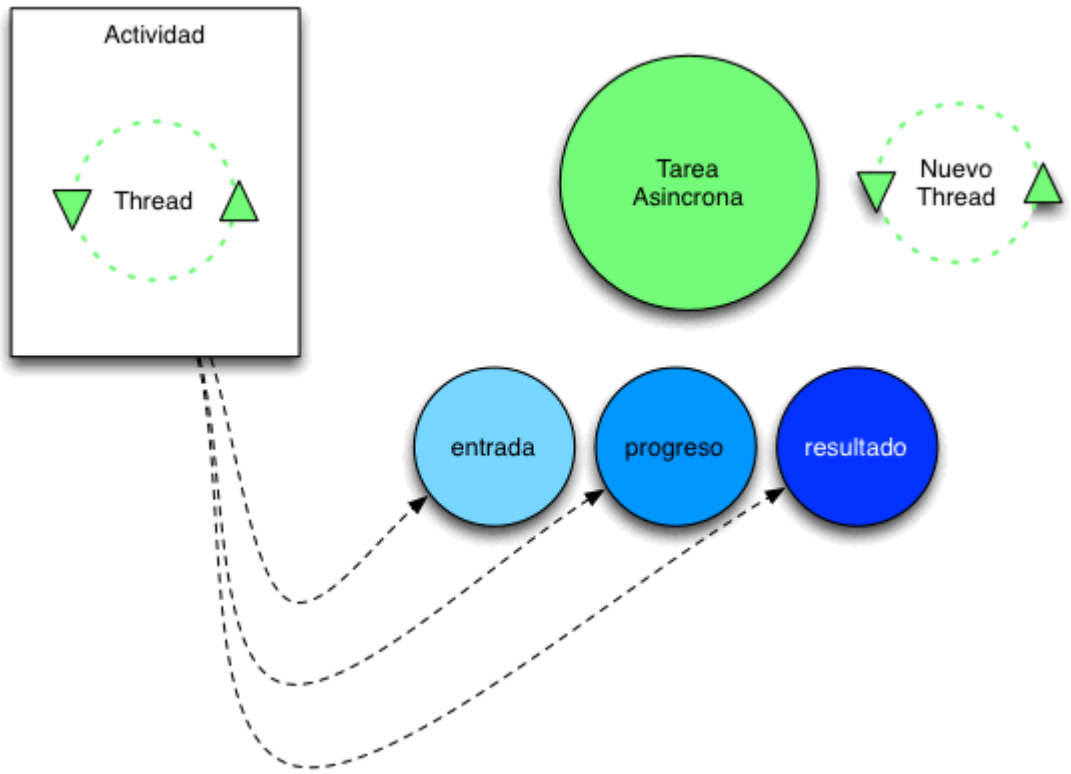
En este caso pulsamos un boton y realizamos la petición que remota tardará unos segundos en ejecutarse .Así pues el Thread que se esta encargando de ejecutarla (Thread UI) quedará bloqueado a la espera de una respuesta . Esto nos generará un efecto colateral . El interface de usuario dejará de responder ya que su Thread esta ocupado .Por lo tanto la aplicación quedará bloqueada para el usuario .¿Como podemos solventar esto?

Android REST y AsyncTask

Para solventar estos problemas Android dispone de una clase especial de tipo Genérico que se encarga de realizar peticiones Asincronas en otro Thread y que se denomina AsyncTask. Simplemente deberemos extender de ella y utilizarla. Ahora bien cuando vamos a realizar esta operación nos encontramos con lo siguiente

```
private class AccesoRemoto extends AsyncTask&amp;amp;lt;String,
String, String&amp;amp;gt; {
..
}
```

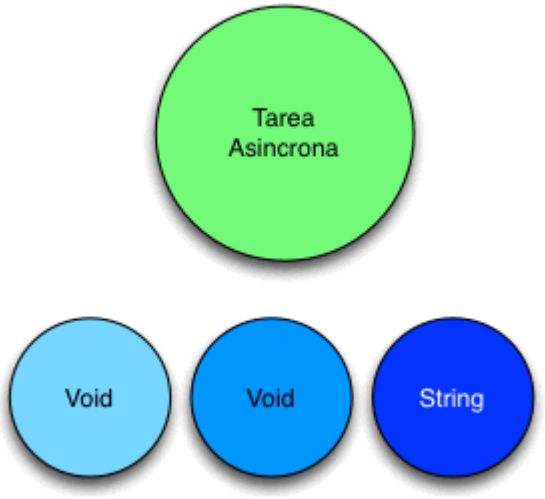
¿Que significan los distintos parámetros de tipo Genérico que la clase obliga a utilizar? . Parece difícil de entender sin embargo las cosas son mas sencillas de lo que parecen. Cuando una actividad crea una tarea asincrona necesita comunicarse con ella ,ahora bien ¿En que situaciones?. Pues parece lógico que para pasarla información , ver su progreso y obtener un resultado final .



Así pues los parámetros genéricos que recibe la tarea Asincrona son simplemente los tipos de datos a gestionar en cada una de las situaciones . Por ejemplo en nuestro caso queremos realizar una petición Http a un servicio REST . ¿Queremos pasar algún dato? . La respuesta es NO . Por lo tanto el primer parametro será de tipo Void. ¿Queremos obtener información sobre el progreso de la tarea? .La respuesta es NO por lo tanto el segundo parámetro será Void también .¿Queremos recibir algún resultado? La respuesta es Si, una cadena con la respuesta del Servicio ,por lo tanto el tercer parámetro será de tipo String.

```
private class AccesoRemoto extends AsyncTask&amp;amp;lt;Void, Void,
```

```
String&amp;amp;gt; {
..
}
```



Una vez aclarados los parámetros .Vamos a construir nuestra TareaAsincrona dentro de una Actividad y rellenar su método doInBackground que será el encargado de realizar una petición Http para mas adelante recibir los resultados.

```
package com.arquitecturajava;

import java.io.BufferedReader;
import java.io.InputStreamReader;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
```

```
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

import com.example.android020accesoremoto.R;

public class MainActivity extends Activity implements OnClickListener
{

Button boton;

@Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 boton = (Button) findViewById(R.id.button1);
 boton.setOnClickListener(this);
 }

@Override
 public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is
present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }
```

```
@Override
 public void onClick(View arg0) {

AccesoRemoto acceso= new AccesoRemoto();

 acceso.execute();
 }

private class AccesoRemoto extends AsyncTask&amp;amp;lt;Void, Void,
String&amp;amp;gt; {

protected String doInBackground(Void... argumentos) {

 StringBuffer bufferCadena = new StringBuffer("");

 try {
 HttpClient cliente = new DefaultHttpClient();
 HttpGet peticion = new HttpGet(
 "http://10.0.2.2:8080/Web001/ServletHola");
// ejecuta una petición get
 HttpResponse respuesta = cliente.execute(peticion);

//lee el resultado
BufferedReader entrada = new BufferedReader(new InputStreamReader(
 respuesta.getEntity().getContent()));

String separador = "";
 String NL = System.getProperty("line.separator");
//almacena el resultado en bufferCadena

 while ((separador = entrada.readLine()) != null) {
```

```
 bufferCadena.append(separador + NL);
 }
 entrada.close();
 } catch (Exception e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }

 return bufferCadena.toString();

 }

protected void onPostExecute(String mensaje) {

 Toast.makeText(MainActivity.this, mensaje,
Toast.LENGTH_SHORT).show();

 }

 }
```

Aunque el código parece un poco complejo de entender simplemente se realiza una petición get a “http://10.0.2.2:8080/Web001/ServletHola” que nos devuelve una cadena. Una vez tenemos la cadena se ejecuta el método onPostExecute que la recibe como parametro para mostrarla en un mensaje Toast. Recordemos que tendremos que haber habilitado los permisos de red en el fichero de manifiesto.

```
&amp;amp;lt;uses-permission android:name="android.permission.INTERNET"
```

```
/&amp;amp;gt;
&amp;amp;lt;uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" /&amp;amp;gt;
```