

# Desarrollo de Aplicaciones para Android

## Sesión 7: Servicios de red



# Puntos a tratar

- Conexiones HTTP
- Parsing de XML
- Cargar imágenes de red
- Estado de la red
- Operaciones lentas



# Permisos para usar la red

- En AndroidManifest.xml, fuera del tag application:

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



# Conexiones HTTP

```
TextView textView = (TextView)findViewById(R.id.TextView01);
textView.setText("Conexión http.\n\n");
try {
    textView.setText("Cabeceras www.ua.es:\n");

    URL url = new URL("http://www.ua.es");
    URLConnection http = (URLConnection)url.openConnection();

    textView.append(" longitud = "+http.getContentLength()+"\n");
    textView.append(" encoding = "+http.getContentEncoding()+"\n");
    textView.append(" tipo = "+http.getContentType()+"\n");
    textView.append(" response code = "+http.getResponseCode()+"\n");
    textView.append(" response message = "+http.getResponseMessage()
        +"\n");

    textView.append(" content = "+http.getContent()+"\n");
} catch (MalformedURLException e) {
} catch (IOException e) {
}
```



# Parsing de XML

- Trocear el XML en tags, atributos, contenido por medio de librerías. En Android tenemos:
  - `SAXParser` : requiere implementar manejadores que reaccionan a eventos al encontrar etiquetas o atributos.
  - `XmlPullParser` : itera sobre el árbol conforme el código lo va pidiendo.



# XmlPullParser

```
XmlPullParserFactory parserCreator = XmlPullParserFactory.newInstance();
XmlPullParser parser = parserCreator.newPullParser();
parser.setInput(url.openStream(), null);
int parserEvent = parser.getEventType();
while (parserEvent != XmlPullParser.END_DOCUMENT) {

    switch (parserEvent) {
        case XmlPullParser.START_DOCUMENT:
            break;
        case XmlPullParser.END_DOCUMENT:
            break;
        case XmlPullParser.START_TAG:
            break;
        case XmlPullParser.END_TAG:
            break;
    }
    parserEvent = parser.next();
}
```



# XmlPullParser

```
case XmlPullParser.START_TAG:
    String tag = parser.getName();
    if (tag.equalsIgnoreCase("title")) {
        Log.i("XML", "El titulo es: " + parser.nextText());
    } else if (tag.equalsIgnoreCase("meta")) {
        String name = parser.getAttributeValue(
            null, "name");
        if (name.equalsIgnoreCase("description")) {
            Log.i("XML", "La descripción es: " +
                parser.getAttributeValue(
                    null, "content"));
        }
    }
    break;
```

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">
<head>
<title>Universidad de Alicante</title>
<meta name="Description" content="Informacion Universidad Alicante. Estudios,
    masteres, diplomaturas, ingenierias, facultades, escuelas" />
<meta http-equiv="pragma" content="no-cache" />
...
```



# Cargar imágenes desde la red

```
<ImageView
    android:id="@+id/ImageView01"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:layout_marginRight="10dip"
    android:src="@drawable/icon" />
```

```
ImageView imageView = (ImageView)convertView.findViewById(R.id.FilaImagen);
try{
    InputStream is= new URL("http://www.ua.es/css/imagenes/logoua.gif")
                        .openStream();
    Drawable imagen = new BitmapDrawable(BitmapFactory.decodeStream(is));
    imageView.setImageDrawable(imagen);
}catch(MalformedURLException e1){
}catch(IOException e2){
}
```





# Estado de la red

- Comprobar el estado

```
ConnectivityManager cm = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo wifi = cm.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
NetworkInfo mobile = cm.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
boolean hayWifi = wifi.isAvailable();
boolean hayMobile = mobile.isAvailable();
boolean noHay = (!hayWifi && !hayMobile); //iiiiinnteerneeeeeeeer!!
```

- Cambiar red preferente

```
cm.setNetworkPreference(NetworkPreference.PREFER_WIFI);
```

- Forzar cambio de estado

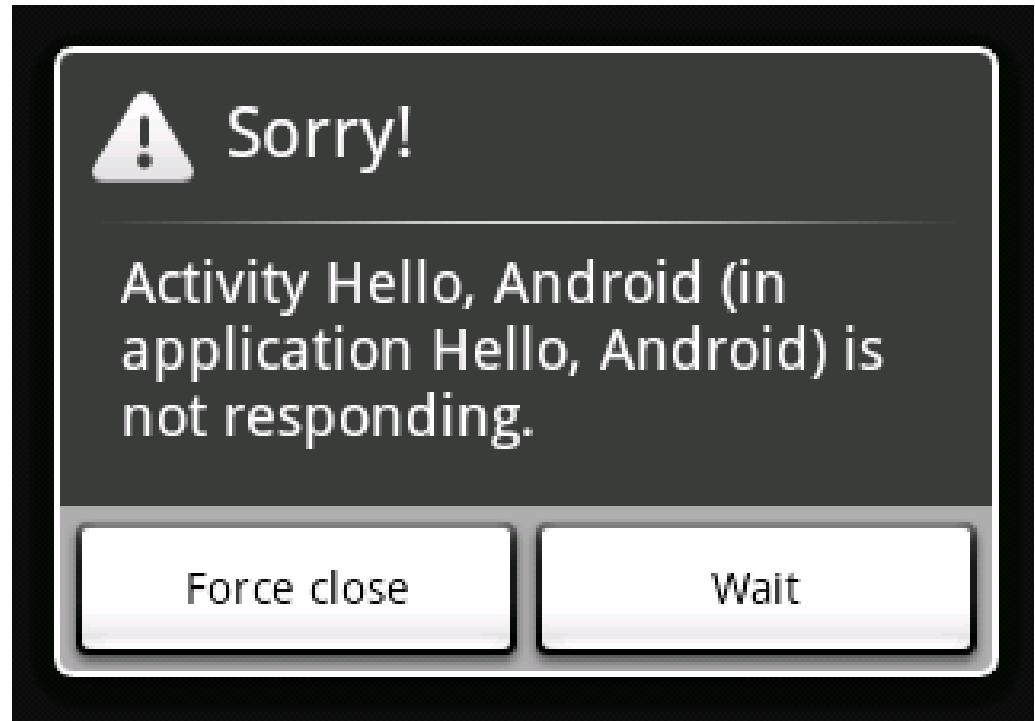
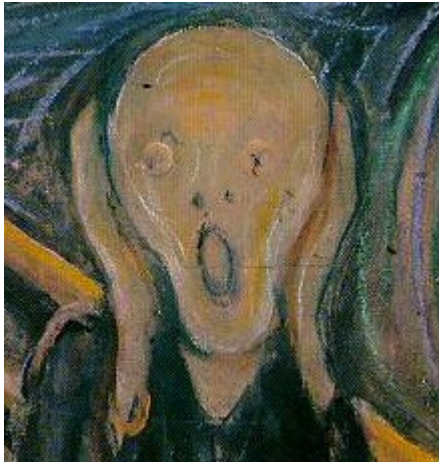
```
cm.setRadio(NetworkType.MOBILE, false);
cm.setRadio(NetworkType.WIFI, true);
```



# Operaciones lentas

- En dispositivos móviles no se puede asumir que ninguna operación de red vaya a ser rápida, ni aunque sea descargar un byte.
- Por tanto hay que tratar las operaciones de red como operaciones lentas.
- En android las operaciones lentas se realizan en un hilo aparte, de lo contrario obtendremos la pantalla ARN, Application not responding.
  - El hilo de la actividad se encarga de mantener operativa la interfaz de usuario. Si la interfaz deja de ser operativa durante 2 segundos, Android sugiere terminar la aplicación con la ARN Screen.

# ANR Screen



- El usuario normalmente no esperará.
- A la segunda vez (con suerte no antes) desinstalará nuestra aplicación.
- Sería raro que volviera a instalar una versión posterior.



# Threads

- Crear otro hilo de ejecución es fácil, pero no nos permite acceder a componentes gráficos: modelo de hilo único para la interfaz gráfica

```
ImageView imageView = (ImageView)findViewById(R.id.ImageView01);
new Thread(new Runnable() {
    public void run() {
        Drawable imagen = cargarLaImagen("http://...");
        //Desde aquí NO debo acceder a imageView
    }
}).start();
```



# Enviar datos a la interfaz

- Una solución es utilizar el método `View.post (Runnable)`

```
ImageView imageView = (ImageView)findViewById(R.id.ImageView01);
new Thread(new Runnable() {
    public void run() {
        Drawable imagen = cargarLaImagen("http://...");
        imageView.post(new Runnable() {
            public void run() {
                imageView.setDrawable(imagen);
            }
        });
    }
}).start();
```



# AsyncTask

- Clase diseñada para facilitar el trabajo con hilos e interfaz gráfica
- Facilita la separación entre
  - Tarea secundaria
  - Interfaz gráfica
- Permite solicitar refresco del progreso desde la tarea secundaria
- El refresco, tras ser solicitado, se ejecutará (en algún momento) en el hilo principal



# AsyncTask

- Crear una clase que herede de `AsyncTask`
- Sobrecargar como mínimo el método `doInBackground (Params . . . )`
- Ejecutar una instancia de la clase con el método `.execute ( . . . )`
- Otros métodos que podemos sobrecargar
  - `onCancelled ()`
  - `onPostExecute (Result)`
  - `onPreExecute ()`
  - `onProgressUpdate (Progress . . . )`
  - `publishProgress (Progress . . . )`



# AsyncTask

- La ejecutamos con `.execute(...)`
- Se invoca `onPreExecute()` en el hilo de la interfaz.
- Se invoca `doInBackground(Params...)` en el segundo hilo
  - Puede realizar llamadas a `publishProgress(Progress...)`
  - Los valores se publicarán en el hilo principal con `publishProgress(Progress...)`
- Se invoca `onPostExecute(Result)` en el hilo de la interfaz.





# AsyncTask

- Restricciones de uso
  - La instancia de la AsyncTask debe ser creada en el hilo de la interfaz.
  - El método `execute(Params...)` debe ser invocado en el hilo de la interfaz.
  - No se deben llamar `onPreExecute()`, `onPostExecute()`, `doInBackground(Params...)` y `onProgressUpdate(Progress...)` manualmente.
  - La AsyncTask sólo se puede ejecutar una vez.



# AsyncTask

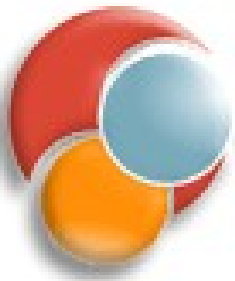
- TextView con el progreso y varios ImageView en los que queremos cargar imágenes con nuestra BajarImagenesTask.

```
TextView textView;  
ImageView[] imageView;  
public void bajarImagenes(){  
    textView = (TextView)findViewById(R.id.TextView01);  
    imageView[0] = (ImageView)findViewById(R.id.ImageView01);  
    imageView[1] = (ImageView)findViewById(R.id.ImageView02);  
    imageView[2] = (ImageView)findViewById(R.id.ImageView03);  
    imageView[3] = (ImageView)findViewById(R.id.ImageView04);  
  
    new BajarImagenesTask().execute(  
        "http://a.com/1.png",  
        "http://a.com/2.png",  
        "http://a.com/3.png",  
        "http://a.com/4.png");  
}
```



# AsyncTask

```
private class BajarImagenesTask extends AsyncTask<String, Integer, List<Drawable>> {
    @Override
    protected List<Drawable> doInBackground(String... urls) {
        ArrayList<Drawable> imagenes = new ArrayList<Drawable>();
        for(int i=1;i<urls.length; i++){
            cargarLaImagen(urls[i]);
            publishProgress(i);
        }
        return imagenes;
    }
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        textView.setText("Cargando imagenes...");
    }
    @Override
    protected void onProgressUpdate(String... values) {
        textView.setText(values[0] + " imagenes cargadas...");
    }
    @Override
    protected void onPostExecute(List<Drawable> result) {
        for(int i=0; i<result.length; i++){
            imageView[i].setDrawable(result.getItemAt(i));
        }
        textView.setText("Descarga finalizada");
    }
    @Override
    protected void onCancelled() {
        textView.setText("Cancelada la descarga");
    }
}
```



# ¿Preguntas...?