Conexión a Internet (1)

Nuestras aplicaciones pueden conectarse a Internet y obtener un recurso, ya sea para mostrarlo dentro de la propia aplicación o para procesar el contenido de dicho recurso.

En cualquiera de los dos casos vamos a necesitar permiso para acceder a Internet, pero al tratarse de un permiso "normal", basta con solicitarlo en el Manifiesto:

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

63

Conexión a Internet (2)

- Mostrar una web

En primer lugar tendremos que agregar a la vista un componente *WebView*.

```
<WebView
    android:id="@+id/webPanel"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

Y después, bastará con obtener la referencia a dicho *WebView* y utilizar el método *loadUrl (String url)*.

```
WebView webPanel = (WebView)findViewById(R.id.webPanel);
webPanel.loadUrl("http://www.myweb.com");
```

64

Conexión a Internet (3)

- Obtener un recurso

Si en lugar de mostrar directamente el sitio web, lo queremos es obtener el contenido (html, txt, etc.) para procesarlo, el proceso es algo más complicado, ya que vamos a tener que almacenar dicha información en un String para lo que es recomendable utilizar un Buffer.

Además el tiempo de descarga puede ser elevado, así que para evitar el bloqueo de la aplicación mientras se realiza la descarga es conveniente realizar el proceso en segundo plano utilizando un servicio, un hilo o una tarea asíncrona.

65

Conexión a Internet (4)

En primer lugar añadimos al Manifiesto la solicitud de permiso para acceder a Internet.

Además, antes de realizar la conexión vamos a comprobar el estado de la red para lo que necesitamos permiso el correspondiente permiso:

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

Por otro lado, el tráfico de texto plano en HTTP está prohibido por defecto y tenemos que permitirlo añadiendo la siguiente propiedad al elemento "*application*" en el Manifiesto:

```
android:usesCleartextTraffic="true"
```

66

Conexión a Internet (5)

Las transferencias de datos con servidores de Internet se realizan mediante las clases:

URL, *URLConnection*, *HttpURLConnection* y *HttpsURLConnection*.

La definición de la URL depende del medio:

- **Archivos locales**:
  file:ruta_archivo
- **Servidor FTP**:
  ftp://usuario:passwd@servidor/ruta
- **Servidor HTTP/HTTPS**:
  http[s]://servidor/ruta:puerto

67

Conexión a Internet (6)

Conexión y transferencia de datos:

1. Creamos un objeto de tipo URL con la ruta del medio al que nos queremos conectar:

```
URL url = new URL("ftp://ftp.rediris.es/");
```

```
URL url = new URL("http://www.google.es/");
```

68

Conexión a Internet (7)

Conexión y transferencia de datos:

2. Creamos un objeto de la clase adecuada al tipo de conexión a establecer y abrir la conexión mediante el método *openConnection*:
   a. FTP: URLConnection

```
URLConnection con = (URLConnection)url.openConnection();
```

   b. HTTP: HttpURLConnection

```
HttpURLConnection con = (HttpURLConnection)url.openConnection();
```

   c. HTTPS: HttpsURLConnection

```
HttpsURLConnection con = (HttpsURLConnection)url.openConnection();
```

69

Conexión a Internet (8)

Conexión y transferencia de datos:

3. Establecemos los parámetros de la conexión:
   - *setReadTimeout*: tiempo de lectura (ms).
   - *setConnectTimeout*: tiempo de conexión (ms).
   - *setRequestMethod*: método GET o POST.
   - *setDoInput*: recepción de datos (*boolean*).
   - *setDoOutput*: envío de datos (*boolean*).

```
con.setReadTimeout(10000);
con.setConnectTimeout(15000);
con.setRequestMethod("GET");
con.setDoInput(true);
```

70

Conexión a Internet (9)

Conexión y transferencia de datos:

4. Descargamos los datos llamando al método *connect* sobre el objeto de la conexión, y los recogemos en un objeto de tipo *InputStream* mediante el método *getInputStream*:

```
con.connect();
int response = con.getResponseCode();
InputStream is = con.getInputStream();
```

71

Conexión a Internet (10)

Conexión y transferencia de datos:

5. Convertimos el *InputStream* obtenido a *String* utilizando un *ByteArrayOutputStream* intermedio que va a ir almacenando cada uno de los caracteres del *InputStream*.

   Al convertir el *ByteArrayOutputStream* a *String* mediante *toString* podemos establecer el *charset* adecuado.

```
ByteArrayOutputStream os = new ByteArrayOutputStream();
int i;
while ( (i=is.read()) != -1) {
    os.write(i);
}
String result = os.toString("iso-8859-1");
```

72

Conexión a Internet (11)

Conexión y transferencia de datos:

6. Por último, una vez que tenemos los datos guardados, debemos cerrar el *InputStream*:

```
if (is != null) {
    is.close();
}
```

73

Conexión a Internet (12)

Este proceso de descarga de un archivo desde Internet puede ser lento, por lo que es muy recomendable realizar la tarea en segundo plano para evitar la sensación al usuario de que la aplicación se ha quedado bloqueada.

Para ejecutar la tarea en segundo plano podemos utilizar varias alternativas:

- **Servicios (*Services*)**.
- **Hilos (*Threads*)**.
- **Tareas Asíncronas (*AsyncTask*)**.

74