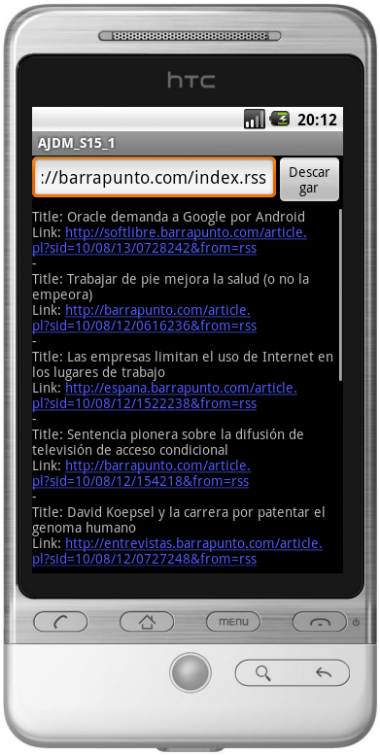
Servicios de red - Ejercicios

Índice

1. Cargar y trocear XML

En este ejercicio vamos a hacer un sencillo lector de RSS. En las plantillas de la sesión tenemos el proyecto AJDM_S15_1 que viene con la interfaz de usuario perparada. Se trata de un `EditText` donde se introducirá la URL, y cuando se pulse el botón o bien se pulse la tecla enter habrá que descargar el XML desde la URL indicada, trocearlo, y mostrar los resultados en el `TextView` que ocupa el resto de la pantalla.

- Añade al `AndroidManifest.xml` los permisos para utilizar Internet.
- Completa el método `accionDescargar()` que se encargará de recoger la URL del `EditText` y de introducir los resultados en el `TextView`.
- Necesitarás buscar tags `item`, y dentro de cada uno de ellos habrá tags `title` y `link` cuyo texto tendrás que añadir al resultado. Si no buscas las aperturas y cierres de `item`, también se mostraría el título de la propia página, que en realidad no es ninguna noticia que queramos mostrar.
- Opcional: Muestra también la fecha de cada noticia



Sencillo lector de RSS

2. Operaciones largas de caraga de datos

Si la página web tardara en cargarse, el programa del ejercicio anterior se quedaría congelado durante el tiempo de la descarga. Si éste fuera superior a 2 segundos, podríamos obtener un mensaje de que la aplicación no responde y si deseamos cerrarla. Este tipo de experiencia de usuario es totalmente indeseable

La actual versión del emulador (Android SDK r06, 2.2 platform) parece tener un bug que no nos permite simular conexiones lentas. Debería funcionar así:

```
telnet localhost 5554
network delay gprs
OK
network speed gsm
OK
```

Vamos a intentar cargar los RSS de un sitio lento, o bien creernos los efectos indeseables de las cargas lentas.

La solución a las operaciones largas es la creacion de hilos o `Thread`. El problema de los hilos es que los componentes gráficos sólo deben ser modificados desde su propio hilo, y no desde otro. Por tanto hay que separar el código de la tarea que va en el segundo hilo, y de la modificación de los componentes, que debe ejecutarse en el hilo principal.

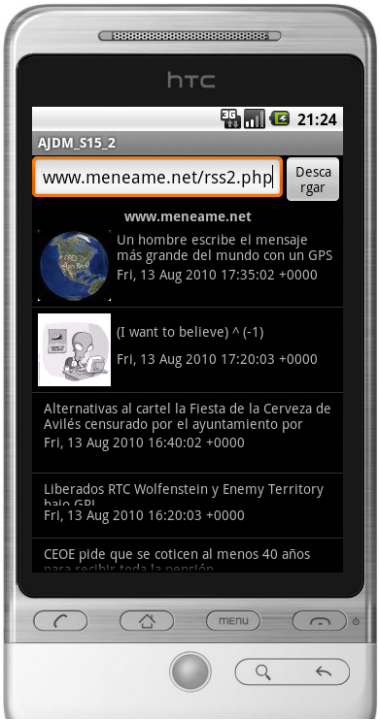
A partir de Android 1.5 se introdujo una nueva clase que nos facilita la programación de tareas en segundo plano: `AsyncTask`.

- Vamos a modificar el código del ejercicio anterior, introduciendo la carga y parseo de XML en el método `doInBackground(...)` de una nueva clase que extienda a `AsyncTask`.
- Tendremos que separar la actualización de componentes y pasarla a los métodos `onPostExecute(...)` y `onProgressUpdate(...)`. Una forma sería ir cargando el resultado en un campo String que declaramos en la clase, para finalmente actualizar el resultado en el campo de texto. También se puede ir actualizando conforme se va cargando.
- Añadir un campo de texto en la parte superior de la pantalla. En él se indicará el progreso de la descarga, por tanto será actualizado por el método `onProgressUpdate(...)`. Este método deberá ser llamado varias veces a lo largo del progreso de la carga, y lo tenemos que hacer explícitamente desde el método `doInBackground(...)`, con una llamada a `publishProgress("Cargando item número "+n)`.
- Finalmente para iniciar la tarea hay que crear un objeto de la nueva clase que hemos creado, y ejecutar su método `execute(...)`, pasándole como parámetro la dirección de la RSS. Si hemos hecho los anteriores puntos, veremos el progreso de la carga en el nuevo campo de texto que hemos añadido.

3. Lector de RSS - versión gráfica

En este ejercicio vamos a ampliar el código del anterior. Vamos a organizar las noticias en una lista de objetos `Noticia`. Después vamos a crear un adaptador para que las noticias puedan ser cargadas en un `ListView`. Aprovecharemos para añadir una imagen si la hay.

Utilizad el proyecto AJDM_S15_3 que hay en las plantillas de la sesión.



Lector de RSS con imágenes

- Crea una clase `Noticia` con los siguientes campos:

```
        private String titulo;
        private String descripcion;
        private String link;
        private String fecha;
        private String linkImagen;
        private Drawable imagen;
```

  Genera getters y setters para todos ellos, y un constructor que los inicialice a "" y a null la imagen. Añade también un método `public void loadImagen(String url)`. Escribe en él el código que carga en el campo `imagen` un `BitmapDrawable` cargado desde la URL indicada. (Se puede utilizar un `InptuStream` que se obtenga a partir de un objeto de tipo `URL`).
- Copia en el método `doInBackground(...)` el código que parsea el XML y rellena la noticia con los campos de Título, Fecha, Descripción, Link, LinkImagen. Hay que tener en cuenta que puede haber diferentes formatos de RSS. Para simplificar el

problema vamos a intentar que funcione al menos para los dos siguientes ejemplos:

- http://barrapunto.com/index.rss

```
<item rdf:about="http://softlibre.barrapunto.com/article.pl?sid=
                                  10/08/13/0728242&amp;from=rss">
<title>Oracle demanda a Google por Android</title>
<link>http://softlibre.barrapunto.com/article.pl?sid=
                               10/08/13/0728242&amp;from=rss</link>
<description>Oracle ha presentado una demanda contra Google
                                 asegurando que ..."</description>
<dc:creator>mig21</dc:creator>
<dc:date>2010-08-13T07:30:00+00:00</dc:date>
<dc:subject>Oracle</dc:subject>
<slash:department>patent-IP-trolling</slash:department>
<slash:section>softlibre</slash:section>
<slash:comments>85</slash:comments>
<slash:hit_parade>85,83,41,29,11,7,3</slash:hit_parade>
</item>
```

- http://www.meneame.net/rss2.php

```
<item>
<meneame:link_id>1025318</meneame:link_id>
<meneame:user>Whitefox</meneame:user>
<meneame:votes>147</meneame:votes>
<meneame:negatives>0</meneame:negatives>
<meneame:karma>616</meneame:karma>
<meneame:comments>25</meneame:comments>
<meneame:url>http://techcrunch.com/2010/08/13/android-oracle-java-lawsuit/
</meneame:url>
<title>Google responde a la denuncia sobre Android presentada por
Oracle
[ENG]</title>
<link>http://m.menea.me/lz52</link>
<comments>http://m.menea.me/lz52</comments>
<pubDate>Sat, 14 Aug 2010 09:40:02 +0000</pubDate>
<dc:creator>Whitefox</dc:creator>
<guid>http://m.menea.me/lz52</guid>
<description>&lt;![CDATA[ ... ]&gt;</description>
<media:thumbnail
  url="http://aws.mnmstatic.net/cache/a5/26/thumb-1025318.jpg"
  width='75' height='49' />
<wfw:commentRss>http://www.meneame.net/comments_rss2.php?id=1025318
</wfw:commentRss>
</item>
```

Recuerda que cada vez que se cierre un `item` en el XML, habrá que añadir el nuevo objeto `noticia` al ArrayList `noticias`.

- Asignar un `OnItemClickListener` al `ListView` para que al hacer click sobre una noticia, se muestre un `Toast` con la información de la noticia. Opcionalmente se puede lanzar un navegador del sistema que nos lleve a la URL de la noticia.