Programación multimedia y dispositivos móviles

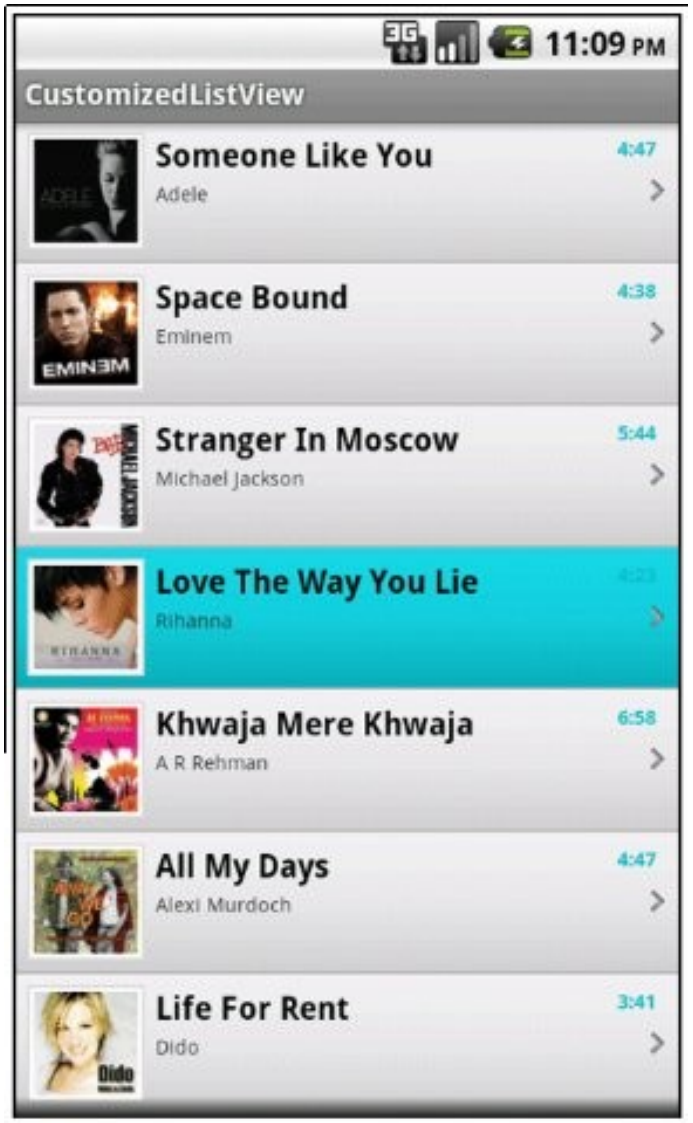
5 Widgets de selección

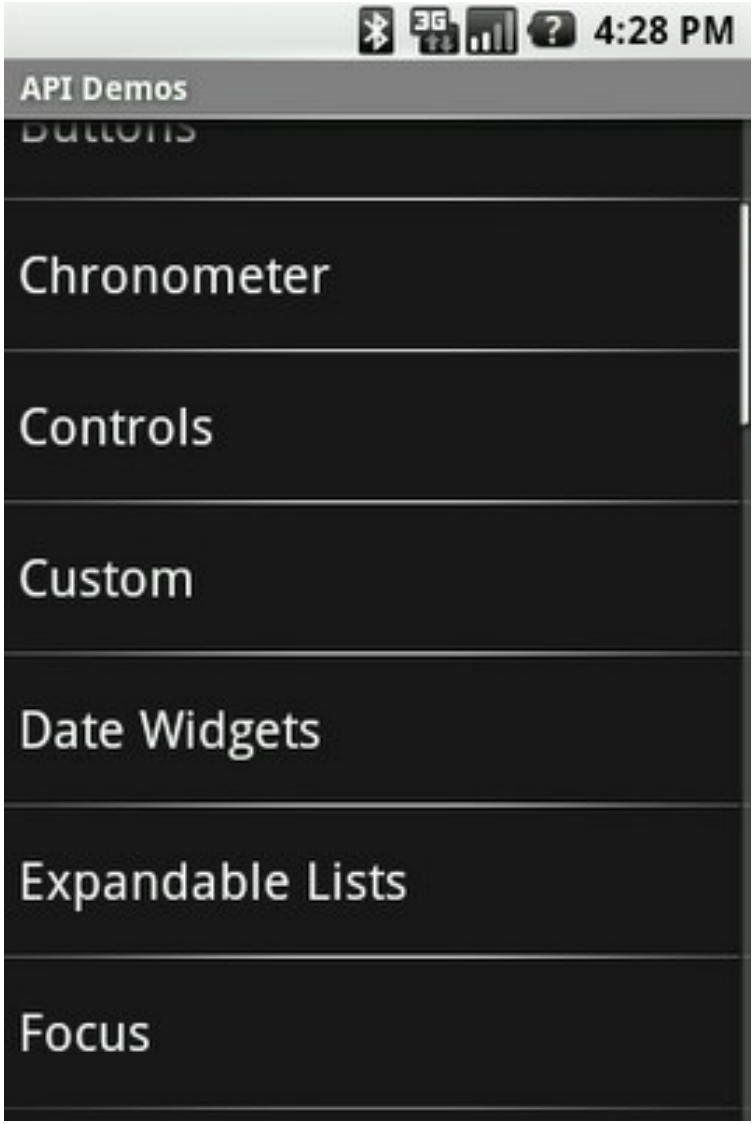
IES Nervión
Miguel A. Casado Alías

Adaptadores (Adapters)

- Los adaptadores son los encargados de proporcionar los datos a los widgets de selección y además convierten esos datos individuales en vistas (View) para ser mostradas dentro del widget de selección
- Ejemplo: ArrayAdapter
  - String[] ciudades={"Roma",”París”,”Londres”};
  - ArrayAdapter a = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, ciudades);
  - Los parámetros que se le pasan al constructor son el contexto (generalmente la clase actual), el ID de un layout o vista para “envolver” cada elemento de la lista (o de cualquier otro widget de selección) y el array con los datos a meter en el widget de selección
  - Por defecto, ArrayAdapter llama a toString() para los objetos del array, y mete esas cadenas en la vista que se le diga

Ejemplos de ListView





ListView

- Es la clásica Lista de Android
- Debemos incluir un **ListView** en nuestro layout
- Con el método **setAdapter** le asignaremos el adaptador a través del que le proporcionaremos los datos y la vista a usar para envolverlos
- Con el método **setOnItemClickListener** le indicaremos al sistema qué objeto será el que se encargue de gestionar los clicks en los elementos de la lista. Dicho objeto a su vez deberá implementar el método **OnItemClick**.
- De todas formas, cuando nuestra actividad sea básicamente una lista, podemos hacer que nuestra clase herede de **ListActivity** en lugar de hacerlo de **Activity**

ListView en una ListActivity

- Si queremos una lista a pantalla completa, ni siquiera debemos proporcionar un layout, ListActivity lo genera
- Si queremos personalizar nuestro layout, deberemos identificar el ListView como @**android:id/list** para que ListActivity sepa cuál es
- Con el método **setListAdapter** determinaremos el adaptador a usar para la lista
- El método **onListItemClick** es el encargado de gestionar lo que ocurre cuando un elemento es seleccionado
- Ver ejemplo: Selection/List

Modos de selección en ListView

- El método **setChoiceMode** aplicado sobre la ListView indicará si la lista será de selección múltiple (CHOICE_MODE_MULTIPLE) o de selección única (CHOICE_MODE_SINGLE)
- También se puede indicar esto mismo en el layout XML con la propiedad choiceMode
- Como layout para envolver los elementos, podemos usar **android.R.layout.simple_list_item_multiple_choice** o bien **android.R.layout.simple_list_item_single_choice** en lugar de android.R.layout.simple_list_item_1
- Métodos útiles: **getCheckedItemPositions** (para saber items marcados), **setItemChecked** (para marcar item)
- Ver ejemplo: Selection/Checklist

Personalizando ListView

- Podemos añadir un icono a los elementos de nuestra lista fácilmente, para ello deberemos:
  - Crear nuestro propio Layout para cada fila de la lista, por ejemplo un LinearLayout con un ImageView y un TextView
  - Al construir el ArrayAdapter habrá que pasarle un parámetro que indique cuál es el identificador del elemento que portará el texto, además de nuestro layout para las filas:
    - new ArrayAdapter<String> (this,R.layout.row, R.id.label,items)
  - Ver ejemplo FancyLists/Static

Personalizando ListView: añadiendo dinamismo

- ¿Y si quisiéramos poner un icono u otro en función de nuestras propias reglas?
  - Deberemos crear nuestro propio adaptador, por ejemplo a partir de ArrayAdapter (o BaseAdapter que es más genérico)
  - El método **getView** de nuestro adaptador será llamado por el AdapterView (por ejemplo un ListView o un Spinner) cuando necesite la vista asociada al dato que esté tratando
  - En el caso de ArrayAdapter, su método getView será llamado cada vez que se procese una casilla del array
  - En nuestro getView pondremos las reglas que se seguirán para rellenar los datos para la lista

Personalizando ListView: añadiendo dinamismo (II)

- Los parámetros de getView son:
  - position: posición del elemento en el conjunto de datos que se está procesando (la primera posición es 0)
  - convertView: antigua vista a reciclar, si es posible. En la siguiente diapositiva se amplía información.
  - parent: la vista padre a la que la vista que generará getView se podrá adosar como hija si se desea
- En nuestro caso, en función del dato tratado, le asignamos un icono u otro pero las posibilidades de personalización son prácticamente ilimitadas
- Ver ejemplo ListaDinamica
- Más info: getView en developer.android.com

getView: Reciclando vistas

- Android nos permite reutilizar objetos en las listas para ahorrar recursos (ciclos de CPU => batería) y mejorar el rendimiento
- Los elementos de la lista que desaparecen de la pantalla por efecto del “scroll” son “reciclados” por android => Los envía como parámetro (convertView) a getView en el instante de crear la nueva fila
- Si podemos reutilizar esa vista reciclada, nos ahorraremos tener que crear el objeto desde cero
- En listas heterogéneas, podría ocurrir que el objeto reciclado fuese de un tipo diferente al del objeto a usar en la nueva fila. Para asegurarnos de que Android siempre envíe para reciclar un objeto del tipo que necesitamos debemos implementar getViewTypeCount y getItemViewType

“Inflando” las filas de nuestras listas

- Los objetos **LayoutInflater** sirven para convertir una especificación de layout XML en el árbol de objetos de tipo View que dicho layout XML representa
- En el caso de las listas, usaremos LayoutInflater para construir nuestras filas cuando tengamos que personalizar un método getView
- El método inflate tiene estos parámetros:
  - resource : ID del layout a cargar / “inflar”
  - root : View a la que se acoplará el recurso como hijo siempre y cuando attachToRoot sea TRUE.
  - attachToRoot : ¿debe ser root el padre del recurso inflado? Si se le manda FALSE, root sólo se usará para proporcionar sus parámetros de layout (p.ej: orientation, etc...)
- Ver ejemplo ListaDinamicaInflater

El patrón ViewHolder

- Una operación que hacemos a menudo, sobre todo si las filas de nuestra lista son muy elaboradas es **findViewById**
- Esta operación puede ser muy “costosa”
- Para ahorrarnos muchas llamadas a findViewById:
  - Cuando creemos una fila le asociaremos mediante **setTag** un “contenedor” (ViewHolder)
  - Dicho contenedor llevará asociados como propiedades los elementos hijos de la fila
  - Cuando usemos una fila “reciclada” llamaremos al método **getTag** para recuperar el contenedor que lleve asociado, y así nos ahorraremos tener que llamar a findViewById nuevamente para obtener los elementos hijos de la fila
- Ver ejemplo ListaViewHolder

SimpleCursorAdapter

- Adaptador para cursor de BD
- Aunque las BDs en Android las veremos más adelante, no está de más echarle un vistazo al siguiente ejemplo para saber que tenemos la posibilidad de usar adaptadores para resultados de consultas a BDs
- Ver ejemplo: ListaMatrixCursor

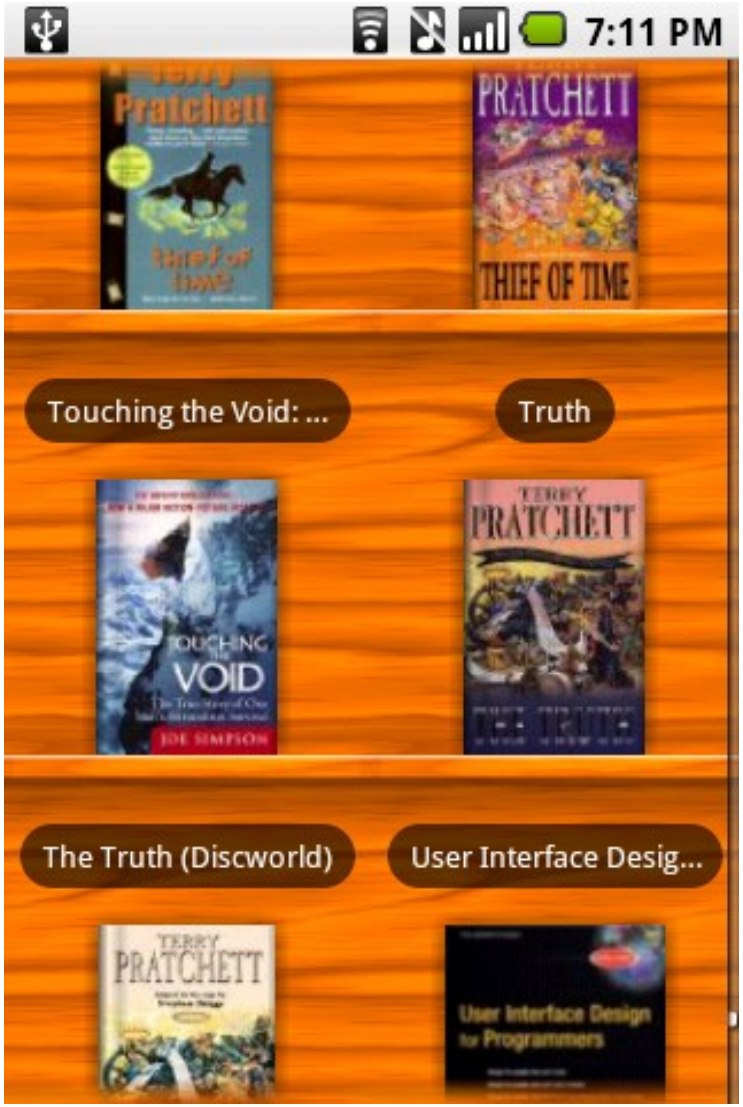
Spinner

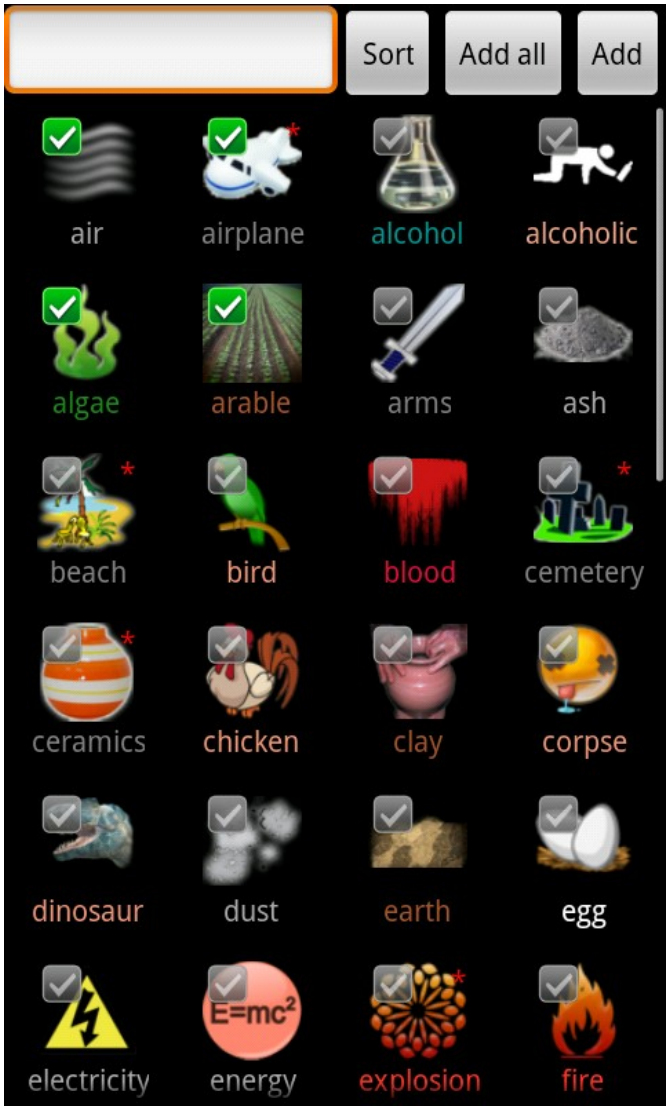
- **getDropDownView**: para la lista desplegable
- **setAdapter** se usará para asignarle el adaptador
- Con **setOnItemSelectedListener** indicamos el “listener”
- **setDropDownViewResource** es un método de ArrayAdapter con el que podemos facilitarle el ID del layout a usar para la lista desplegable
- **drawSelectorOnTop** indica si queremos que se dibuje el selector (“parpadeo” al pulsar) encima de la vista de la fila
- **onItemSelected** es el método que gestiona las acciones a tomar cuando un item es seleccionado
- **onNothingSelected** gestiona qué hacer si no hay seleccionado ningún item
- Ver ejemplo: Selection/Spinner

Spinner (II)

- Los Spinners no están pensados para tener filas heterogéneas
  - En las APIs >= 21 es **obligatorio** que el método getViewTypeCount de los adaptadores para Spinners devuelva 1
  - En las APIs más antiguas se permite que el método getViewTypeCount de los adaptadores para Spinners devuelva un entero mayor que uno, **PERO** a pesar de eso, **el sistema no lo tendrá en cuenta** a la hora de enviar filas para reciclar, por lo que puede mandar a getView una vista (convertView) que no sea del tipo que se requiere

Ejemplos de GridView





GridView

- Es una cuadrícula que contendrá los datos que deseemos
- Propiedades interesantes:
  - numColumns : número de columnas. Puede ser “auto_fit”
  - verticalSpacing y horizontalSpacing : espacio entre items
  - columnWidth : ancho de las columnas
  - stretchMode : indica qué se debe hacer con el espacio sobrante en el caso de que “numColumns” valga “auto_fit”. Si le damos como valor “spacingWidth” el espacio sobrante será para los interespaciados de items, si por el contrario le damos “columnWidth” será para las columnas
- Por lo demás, es parecido a los demás selectores, usaremos setAdapter , setOnItemSelectedListener , etc...
- Ver ejemplo Selection/Grid

AutocompleteTextView

- Mezcla entre EditText y Spinner
- A medida que el usuario teclea, las opciones posibles a seleccionar se van filtrando y le aparecen en forma de lista desplegable. El usuario puede introducir un valor no “ofertado” o elegir uno de los que se le ofrecen
- Hereda de TextView
- La propiedad **completionThreshold** indica el mínimo número de caracteres que el usuario debe teclear para que se le ofrezca la lista de posibles valores
- Usaremos setAdapter como de costumbre
- En vez de un listener implementaremos TextWatcher, para ser notificados cuando el texto del campo cambie
- Ver ejemplo: Selection/AutoComplete