Programación multimedia y dispositivos móviles

4 Layouts

IES Nervión
Miguel A. Casado Alías

LinearLayout

- Sus elementos hijos son alineados uno tras otro, bien formando una columna o una fila
- orientation: horizontal (fila) o vertical (columna)
- El tamaño de los elementos internos al LinearLayout se define mediante layout_width y layout_height
  - Podemos dar un tamaño en px o dip
  - wrap_content: El tamaño será el necesario para albergar el contenido deseado
  - match_parent (fill_parent en API<8) El widget rellena todo el tamaño hasta llegar a su contenedor padre
- layout_weight indica qué proporción del espacio libre disponible debería ir a cada widget

LinearLayout (II)

- p.ej si layout_weight es 2 para un widget y 1 para otros dos, el primero tomará la mitad (2/4) del espacio libre y cada uno de los otros dos tomarán la cuarta (1/4) parte cada uno
- Podemos usar layout_weight en combinación con layout_width: 0 dip para así asignar todo el espacio, ya que todo el espacio está libre
- layout_gravity controla la alineación
- layout_margin controla los márgenes. Se puede usar un margen para cada lado: layout_marginTop, layout_marginRight, etc...
- Ver ejemplos Containers/Linear y Containers/LinearPercent

RelativeLayout

- Los elementos se posicionan de forma relativa con respecto a otros elementos o su contenedor
- Las relaciones relativas al contenedor toman como valor un booleano (true o false) y son:
  - layout_alignParentTop: La parte de arriba del widget se alinea con la parte de arriba de su contenedor
  - layout_alignParentBottom, layout_alignParentLeft y layout_alignParentRight se comportan de forma análoga a la anterior propiedad pero en los lados inferior, izquierdo y derecho
  - layout_centerHorizontal, layout_centerVertical y layout_centerInParent centran el widget en referencia a su contenedor. La primera propiedad lo centra horizontalmente sólo, la segunda verticalmente y la tercera en ambos ejes

RelativeLayout: posición respecto a otros widgets

- Toman como valor el “id” del widget con respecto al que queremos posicionar el widget actual
  - layout_above, layout_below, layout_toLeftOf y layout_toRightOf posicionan el widget encima, debajo, a la izquierda o a la derecha del widget referenciado respectivamente
  - layout_alignTop alinea el borde superior del widget con el borde superior del widget referenciado
  - layout_alignBottom, layout_alignLeft y layout_alignRight son análogas a la anterior pero respecto a los demás bordes
  - layout_alignBaseline alinea las “líneas base” (línea imaginaria sobre la que el texto va escrito) de ambos widgets
- Ejemplos: Containers/Relative y Containers/RelativeOverlap

TableLayout

- Es similar a las tablas XHTML
- TableRow es el contenedor para albergar las filas
- layout_span indica el número de columnas a través de las que se “expande” el widget (similar a colspan en XHTML)
- Las columnas se enumeran empezando en 0
- Los widgets por defecto son posicionados en la primera columna disponible, pero este comportamiento se puede alterar con layout_column (indica la columna en la que queremos que se sitúe el widget)
- Se pueden poner widgets entre las filas
- stretchColumns, shrinkColumns y collapseColumns sirven para ensanchar, estrechar y hacer invisibles las columnas
- Ejemplo: Containers/Table

ScrollView y HorizontalScrollView

- Sirven para permitir desplazar el contenido del layout
- ScrollView permite desplazamiento vertical
- HorizontalScrollView fue introducido en Android 1.5 y permite desplazamiento horizontal
- No podemos permitir desplazamiento bidireccional
- Ejemplo: Containers/Scroll