Diseño básico de Layouts y Eventos

Dra. Mª del Puerto Paule Ruiz

Diseño de layout

- Carpeta res/layout
  - Cada pantalla se corresponde con un fichero en esa carpeta
- Definimos los componentes que se muestran en la **Activity** y su posición
- Se diseña el layout utilizando una jerarquía de **Views y ViewGroups** (para algunos layouts)

Layouts

- Definen la posición de todos los elementos en pantalla y son los elementos raíz del fichero XML
  - **LinearLayout** : Alinea los elementos en una sola dirección, uno a continuación de otro (vertical u horizontal)
  - **RelativeLayout**: La posición de los elementos es relativa a los otros. Todos los elementos precisan un id.
  - **FrameLayout:** Almacena un elemento. Es el layout más simple
  - **TableLayout:** Muestra elementos en torno a filas y columnas
  - **GridLayout:** Desde API14. Organiza los elementos en torno a una rejilla rectangular
  - **CoordinatorLayout**: Desde API21. Coordinar las vistas que hay en él. Surge con Material Design
  - **Constrait_Layout**: Similar a Relative Layout, pero más flexible. Editor adaptado para el constraint_layout

Views

- Componentes básicos de la interfaz de usuario e interactúan con éste
- Definen un área rectangular en la pantalla y se encargan de dibujarse y manejar los eventos
- Es la clase base para crear los componentes UI interactivos (Buttons, TextFields, EditText, ….)

Carpeta res/values

- Almacena ficheros xml
- strings.xml
  - Almacena todos los strings de la aplicación
- Se puede añadir cualquier fichero de tipo xml que agrupe una característica común:
  - color.xml
  - dimens.xml
  - styles.xml

Referencia a los recursos

- ¿Cómo podemos acceder a los recursos?
  - @[package:]type/name
- Ejemplos:
  - android:text="@string/hello"
    - <string name="hello">Hola!</string>
  - android: background: "@color/red"
- Identificadores
  - "@+id/nombreRecurso"
    - android:id="@+id/nombreRecurso"

Algunos atributos

- android:id: Identificador del componente
- android:orientation: vertical u horizontal
- android: layout_width: ancho
- android: layout_height: alto
- Constantes:
  - match_parent: El elemento es tan grande como su padre
  - wrap_content: El elemento tendrá el tamaño suficiente como para albergar su contenido

Activity

- Se declara extendiendo de **Activity/ActionBarActivity/AppCompatActivity**
- ~~**ActionBarActivity**~~ significa deprecated. Se puede seguir usando, pero no da soporte para ella.
  - Sustituir AppCompatActivity (API 23)
- Crea una venta donde podemos dibujar nuestor UI
  - Dibujamos **Views y/o ViewsGroups**
  - Con el método **setContentView (View)** establecemos que View vamos a mostrar

Eventos en Android

- Un evento es un mensaje de software que indica que algo ha ocurrido:
  - Se ha pulsado un botón
  - Pulsación de una tecla
  - Se ha presionado un botón
  - ….

Eventos en Android

- En Android (dentro de una **Activity**) antes de interactuar con los objetos **Views**, debemos obtener una referencia a dichos elementos:
  - (View) findViewById(R.id.name):
- Ejemplo:

  XML:

  ```
  <Button android:id="@+id/ButtonAceptar"/>
  ```

  Código Java:

  ```
  mButton=(Button) findViewById(R.id.ButtonAceptar)
  ```

Eventos en Android

- Una vez obtenido un elemento de la clase **View**, podemos establecer diversos eventos para su gestión:
  - setOnClickListener: Asocia un listener al evento OnClick

```
mButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View v) {
        //Lo que sea al hace el click
}

});
```

Eventos en Android (cont.)

```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Aceptar"
        android:id="@+id/bAceptar"
        android:onClick="checkPassword" />
```

```
 public void checkPassword(View view){
    //código a incluir

}
```

Algunos eventos más…

- setOnLongClickListener
- setOnKeyListener
- setOnFocusChangeListener
- ….

Toast

- Widget para informar al usuario
  - Toast.makeTest (Context, text, duration).show()
- El método getString(R.string.nombre) nos permite, dentro del código, obtener el string “nombre” declarado en el fichero strings.xml
  - XML:
    - <string name=“seleccioneValor”> Seleccionar un valor!</string>
  - Código Java
    - getString (R.string.seleccioneValor);

Toast

- Por ejemplo, si creamos un Toast con el siguiente código
  - Toast.makeText (getApplicationContext(),
    getString(R.string.seleccionarValor),
    Toast.LENGTH_SHORT).show()

  Resultado: Sale un mensaje en la pantalla

Clase Log

- Nos permite realizar anotaciones en el código
  - Útil en caso de depuración
- Tipos de mensajes:
  - Debug:
    - Log.d(“DEBUG”, “Hola soy un debug”);
  - Error:
    - Log.e(“ERROR”, “Hola, soy un error”);
  - Info:
    - Log.i (“INFO”, “Hola, soy un info”);
  - Verbose:
    - Log.v(“VERBOSE”, “Hola, soy un verbose”);
  - Warn:
    - Log.w(“WARN”, “Hola, soy un warn!);

Ejercicio

- Verificar que la clave introducida por el usuario es correcta.
  - LinearLayout
  - RelativeLayout
  - ConstraintLayout

Ejemplo

- AndroidVersions