

ОСНОВНЫЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

1) TextView

Для простого вывода текста на экран предназначен элемент **TextView**. Он просто отображает текст без возможности его редактирования. Некоторые его основные атрибуты:

- `android:text`: устанавливает текст элемента
- `android:textSize`: устанавливает высоту текста, в качестве единиц измерения для указания высоты используются `sp`
- `android:background`: задает фоновый цвет элемента в виде цвета в шестнадцатиричной записи или в виде цветового ресурса
- `android:textColor`: задает цвет текста
- `android:textAllCaps`: при значении `true` делает все символы в тексте заглавными
- `android:textDirection`: устанавливает направление текста. По умолчанию используется направление слева направо, но с помощью значения `rtl` можно установить направление справа налево
- `android:textAlignment`: задает выравнивание текста. Может принимать следующие значения:
 - ✓ `center`: выравнивание по центру
 - ✓ `textStart`: по левому краю
 - ✓ `textEnd`: по правому краю
 - ✓ `viewStart`: при направлении текста слева направо выравнивание по левому краю, при направлении справа налево - по правому
 - ✓ `viewEnd`: при направлении текста слева направо выравнивание по правому краю, при направлении справа налево - по левому
- `android:fontFamily`: устанавливает тип шрифта. Может принимать следующие значения:
 - ✓ `monospace`
 - ✓ `serif`
 - ✓ `serif-monospace`
 - ✓ `sans-serif`
 - ✓ `sans-serif-condensed`
 - ✓ `sans-serif-smallcaps`
 - ✓ `sans-serif-light`
 - ✓ `casual`
 - ✓ `cursive`
 - ✓ `cursive`

2) EditText

Элемент **EditText** является подклассом класса **TextView**. Он также представляет текстовое поле, но теперь уже с возможностью ввода и редактирования текста. Таким образом, в **EditText** мы можем использовать все те же возможности, что и в **TextView**.

Из тех атрибутов, что не рассматривались в теме про **TextView**, следует отметить атрибут `android:hint`. Он позволяет задать текст, который будет отображаться в качестве подсказки, если элемент **EditText** пуст. Кроме того, мы можем использовать атрибут `android:inputType`, который позволяет задать клавиатуру для ввода. В частности, среди его значений можно выделить следующие:

- `text`: обычная клавиатура для ввода однострочного текста
- `textMultiLine`: многострочное текстовое поле
- `textEmailAddress`: обычная клавиатура, на которой присутствует символ @, ориентирована на ввод email
- `textUri`: обычная клавиатура, на которой присутствует символ /, ориентирована на ввод интернет-адресов
- `textPassword`: клавиатура для ввода пароля
- `textCapWords`: при вводе первый введенный символ слова представляет заглавную букву, остальные - строчные
- `number`: числовая клавиатура
- `phone`: клавиатура в стиле обычного телефона
- `date`: клавиатура для ввода даты
- `time`: клавиатура для ввода времени
- `datetime`: клавиатура для ввода даты и времени

3) Button

Одним из часто используемых элементов являются кнопки, которые представлены классом `android.widget.Button`. Ключевой особенностью кнопок является возможность взаимодействия с пользователем через нажатия.

Некоторые ключевые атрибуты, которые можно задать у кнопок:

- `text`: задает текст на кнопке
- `textColor`: задает цвет текста на кнопке
- `background`: задает фоновый цвет кнопки
- `textAllCaps`: при значении `true` устанавливает текст в верхнем регистре. По умолчанию как раз и применяется значение `true`
- `onClick`: задает обработчик нажатия кнопки

4) Checkbox

Элементы `Checkbox` представляют собой флажки, которые могут находиться в отмеченном и неотмеченном состоянии. Флажки позволяют производить множественный выбор из нескольких значений.

Атрибут `android:onClick`, как и в случае с простыми кнопками, позволяет задать обработчик нажатия на флажок.

5) RadioButton

Схожую с флажками функциональность предоставляют переключатели, которые представлены классом `RadioButton`. Но в отличие от флажков одновременно в группе переключателей мы можем выбрать только один переключатель.

Чтобы создать список переключателей для выбора, вначале надо создать объект `RadioGroup`, который будет включать в себя все переключатели

6) DatePicker

DatePicker представляет элемент для выбора даты. Среди его атрибутов можно отметить следующие:

- android:calendarTextColor: цвет текста календаря
- android:calendarViewShown: указывает, будет ли отображаться вид календаря
- android:datePickerMode: устанавливает режим выбора даты
- android:dayOfWeekBackground: устанавливает фоновый цвет панели выбора дня недели
- android:endYear: устанавливает последний отображаемый год
- android:firstDayOfWeek: устанавливает первый день недели
- android:headerBackground: устанавливает фоновый цвет для панели выбранной даты
- android:maxDate: устанавливает максимальную отображаемую дату в формате mm/dd/yyyy
- android:minDate: устанавливает минимальную отображаемую дату в формате mm/dd/yyyy
- android:spinnersShown: указывает, будет ли отображаться спиннер в виджете
- android:startYear: устанавливает начальный отображаемый год
- android:yearListSelectorColor: устанавливает цвет для поля выбора года

Среди методов DatePicker можно отметить следующие:

- int getDayOfMonth(): возвращает номер выбранного дня
- int getMonth(): возвращает номер выбранного месяца (от 0 до 11)
- int getYear(): возвращает номер выбранного года
- void init(int year, int monthOfYear, int dayOfMonth, DatePicker.OnDateChangedListener onDateChangedListener): устанавливает начальную дату. Последний параметр устанавливает слушатель изменения выбранной даты
- void setOnDateChangedListener(DatePicker.OnDateChangedListener onDateChangedListener): устанавливает слушатель изменения выбранной даты
- void setFirstDayOfWeek(int firstDayOfWeek): устанавливает первый день недели
- void updateDate(int year, int month, int dayOfMonth): программно обновляет выбранную дату

7) TimePicker

TimePicker представляет виджет для выбора времени, который может отображать время либо в 24-часовом, либо в 12-часовом формате.

Среди атрибутов TimePicker следует выделить timePickerMode, который позволяет режим отображения и может принимать одно из двух значений: clock (отображение в виде часов) и spinner (отображение в виде спиннера).

Среди методов TimePicker можно отметить следующие:

- int getHour(): возвращает час (в 24-часовом формате)
- int getMinute(): возвращает минуты
- boolean is24HourView(): возвращает true, если используется 24-часовой формат
- void setHour(int hour): устанавливает час для TimePicker
- void setIs24HourView(Boolean is24HourView): устанавливает 24-часовой формат
- void setMinute(int minute): устанавливает минуты

- `void setOnTimeChangeListener(TimePicker.OnTimeChangeListener onTimeChangeListener):` устанавливает слушатель изменения времени в `TimePicker` в виде объекта `TimePicker.OnTimeChangeListener`

8) SeekBar

Элемент `SeekBar` выполняет роль ползунка, то есть шкалу делений, на которой мы можем менять текущую отметку.

Среди его атрибутов можно отметить следующие:

- `android:max`: устанавливает максимальное значение
- `android:min`: устанавливает минимальное значение
- `android:progress`: устанавливает текущее значение, которое находится в диапазоне между минимальным и максимальным

Для управления `SeekBar` определяет ряд методов, из которых выделим следующие:

- `void setProgress(int progress)`: устанавливает текущее значение ползунка
- `void setMin(int min)`: устанавливает минимальное значение
- `void setMax(int max)`: устанавливает максимальное значение
- `void incrementProgressBy(int diff)`: увеличивает текущее значение на `diff`
- `int getMax()`: возвращает максимальное значение
- `int getMin()`: возвращает минимальное значение
- `int getProgress()`: возвращает текущее значение
- `void setOnSeekBarChangeListener(SeekBar.OnSeekBarChangeListener l)`: устанавливает слушателя изменения значения в `SeekBar`