Создание пользовательского интерфейса

Отделение ресурсов от кода приложения

- При создании реальных приложений чаще всего ресурсы приложения отделяют от кода и хранят за пределами исходного кода. Такой подход позволяет создавать приложения, обеспечивающие вы сокое качество реализации пользовательского интерфейса (user interface, UI), независящее от усло вий работы программы. Наиболее важно то, что такая адаптация приложения к новым условиям не требует каких-либо изменений в коде приложения, необходимо только обеспечить доступ к требу емым ресурсам.

- Android поддерживает выделение во внешние файлы таких ресурсов, как изображения, анимация и визуальные стили. Внешние ресурсы легче поддерживать, обновлять и контролировать. Можно так же описывать альтернативные ресурсы для поддержки различного аппаратного обеспечения и ло кализации. Android динамически выбирает данные из дерева ресурсов, содержащего разные зна чения для разных аппаратных конфигураций, языков и регионов, что позволяет описывать уни кальные значения для конкретных языков, стран, экранов и клавиатур. При запуске приложения Android автоматически выбирает ресурс с соответствующими данными.

Ресурсы приложения хранятся в каталоге res/ внутри дерева проекта.

Каждый тип ресурсов представлен в виде подкаталога, содержащего соот ветствующие данные.

В каталоге res/ хранятся разметка (размещение элементов интерфейса) по умолчанию, значок приложения и определения строковых констант. При соз дании нового проекта Android автоматически добавляет в него каталог res.

Разные каталоги предусмотрены для главных типов ресурсов: простых зна чений, ресурсов Drawable (изображений), менеджеров компоновки (разметки), анимации, стилей, меню, настроек поиска, XML и «сырых» (необработанных) данных. При сборке приложения эти ресурсы скомпилируются и включатся в программный пакет.

Для доступа к ресурсам необходимо создать ссылку на идентификатор ре сурса. Для задания всех таких идентификаторов в проекте используется класс R. При сборке приложения генерируется файл для класса R, содержа щий ссылки на все ресурсы проекта, что позволяет ссылаться на ресурсы внутри кода программы и проверять синтаксис в процессе разработки.

Идентификатор ресурсов состоит из следующих компонентов:

1) тип ресурса (string, drawable, layout);

2) имя ресурса (имя файла без расширения либо значение атрибута XML android:name, если ресурс представляет собой простое значение, например строку).

Существует два способа доступа к ресурсу:

1) из кода с помощью статической целочисленной переменной из подкласса R, нап ример, R.string.hello (string — тип ресурса, hello — имя ресурса);

2) из XML с помощью особого синтаксиса XML, который также соответствует иденти фикатору ресурса, заданному в классе R, например, @string/hello (string — тип ресур са, hello — имя ресурса).

Имена файлов для ресурсов должны состоять исключительно из букв в нижнем ре гистре и чисел, а также символов «точка» и «подчеркивание».

Ресурсы изображений

Одним из наиболее распространенных источников ресурсов являются файлы изображений. Android поддерживает следующие форматы файлов: *.png* (предпочтителен), *.jpg* (приемлем), *.gif* (нежелателен). Для графических файлов в проекте уже по умолчанию создана папка **res/drawable**. При добавлении графических файлов в эту папку для каждого из них Android создает ресурс **Drawable**. После этого мы можем обратиться к ресурсу следующим образом в коде Java:

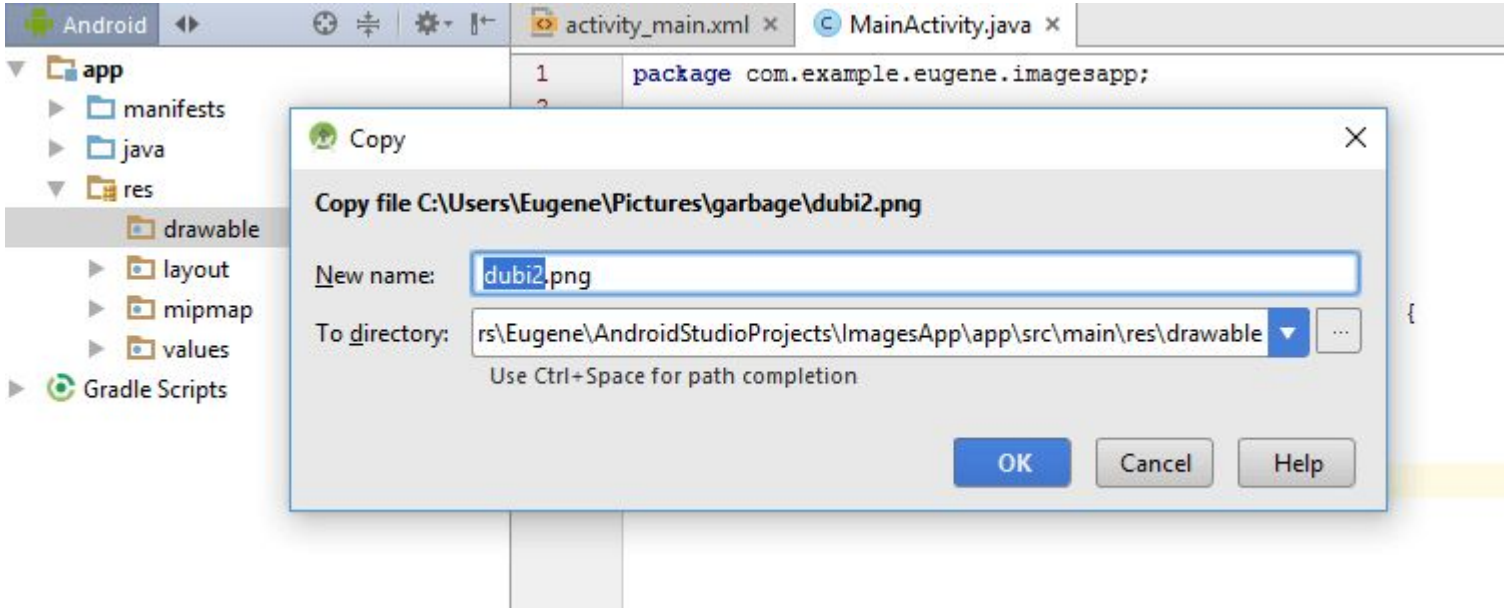
```
R.drawable.
имя_файла
```

Или в коде xml:

```
@[имя_пакета:]drawable/имя_файл
а
```

После создания проекта скопируем в проект в папку *res/drawable* какой-нибудь файл изображения. Здесь разу стоит учитывать, что файл изображения будет добавляться в приложение, тем самым увеличивая его размер. Кроме того, большие изображения отрицательно влияют на производительность. Поэтому следует использовать небольшие и оптимизированные (сжатые) графические файлы.

При копировании файла нам будет предложено установить для него новое имя.



Для работы с изображениями в Android можно использовать различные элементы, но непосредственно для вывода изображений предназначен **ImageView**. Поэтому изменим файл *activity_main.xml* следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk
/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/dubi2" />
</RelativeLayout>
```

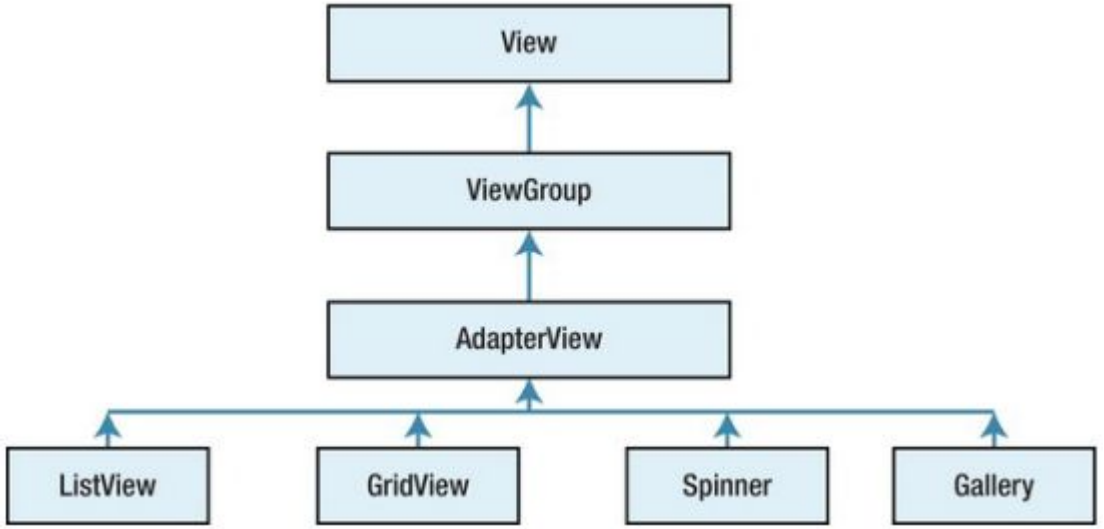
ImageView

Среди его функционала следует отметить возможность масштабирования изображения, которое задается с помощью перечисления . Это перечисление определяет следующие значения:

1. CENTER: изображение центрируется по центру без масштабирования
2. CENTER_CROP: изображение центрируется по центру и масштабируется с сохранением аспектного отношения между шириной и высотой. Если какая-то часть не помещается в пределы экрана, то она обрезается
3. CENTER_INSIDE: изображение центрируется по центру и масштабируется с сохранением аспектного отношения между шириной и высотой, но ширина и высота не могут быть больше ширины и высоты ImageView
4. FIT_CENTER: изображение масштабируется и центрируется
5. FIT_START: изображение масштабируется и устанавливается в начало элемента (вверх при портретной оринетации и влево - при альбомной)
6. FIT_END: изображение масштабируется и устанавливается в конец элемента (вниз при портретной оринетации и вправо - при альбомной)
7. FIT_XY: изображение масштабируется без сохранения аспектного отношения межуд шириной и высотой, заполняя все пространство ImageView
8. MATRIX: изображение масштабируется с применением матрицы изображения

Адаптеры и списки

Android представляет широкую палитру элементов,которые представляют списки. Все они является наследниками класса **android.widget.AdapterView**. Это такие виджеты как ListView, GridView, Spinner. Они могут выступать контейнерами для других элементов управления



При работе со списками мы имеем дело с тремя компонентами. Во-первых, это сам элементы списков (ListView, GridView), которые отображают данные. Во-вторых, это источник данных - массив, объект ArrayList, база данных и т.д., в котором находятся сами отображаемые данные. И в-третьих, это адаптеры - специальные компоненты, которые связывают источник данных с элементом списка.

ArrayAdapter

Класс ArrayAdapter представляет собой простейший адаптер, который связывает массив данных с набором элементов TextView, из которых, к примеру, может состоять ListView. То есть в данном случае источником данных выступает массив объектов. ArrayAdapter вызывает у каждого объекта метод toString() для приведения к строковому виду и полученную строку устанавливает в элемент TextView.

Для создания адаптера используем следующий конструктор ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, countries), где

this : текущий объект activity

android.R.layout.simple_list_item_1 : файл разметки списка, который фреймворк представляет по умолчанию. Он находится в папке Android SDK по пути platforms/[android-номер_версии]/data/res/layout. Если нас не удовлетворяет стандартная разметка списка, мы можем создать свою и потом в коде изменить id на id нужной нам разметки

countries : массив данных. Здесь необязательно указывать именно массив, это может быть список ArrayList<T>.

В конце неоходимо установить для ListView адаптер с помощью метода setAdapter().

```
public class MainActivity extends AppCompatActivity {

    // набор данных, которые свяжем со списком
    String[] countries = { "Бразилия", "Аргентина",
"Колумбия", "Чили", "Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // получаем элемент ListView
        ListView countriesList = (ListView)
findViewById(R.id.countriesList);

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, countries);

        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
    }
}
```

Выбор элемента в ListView

кроме простого вывода списка элементов ListView позволяет выбирать элемент и обрабатывать его выбор. Рассмотрим, как это сделать. Определим следующую разметку в файле *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/selection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="22sp" />
    <ListView
        android:id="@+id/countriesList"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Теперь свяжем список ListView с источником данных и закрепим за ним слушатель нажатия на элемент списка:

```
public class MainActivity extends AppCompatActivity {

    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили", "Уругвай"};
    private TextView selection;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // получаем элемент TextView
        selection = (TextView) findViewById(R.id.selection);
        // получаем элемент ListView
        ListView countriesList = (ListView) findViewById(R.id.countriesList);

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, countries);
        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
        // добвляем для списка слушатель
        countriesList.setOnItemClickListener(new OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position, long id)
            {
                // по позиции получаем выбранный элемент
                String selectedItem = countries[position];
                // установка текста элемента TextView
                selection.setText(selectedItem);
            }
        });
    }
}
```

Итак, метод setAdapter связывает элемент ListView с определенным адаптером. Далее для обработки выбора элемента списка устанавливается слушатель OnItemClickListener. Этот слушатель имеет один метод onItemClick, через параметры которого мы можем получить выделенный элемент и сопутствующие данные. Так, он принимает следующие параметры:

---

1. **parent : нажатый элемент AdapterView (в роли которого в данном случае выступает наш элемент ListView)**
2. **view : нажатый виджет внутри AdapterView**
3. **position : индекс нажатого виждета внутри AdapterView**
4. **id : идентификатор строки нажатого элемента**

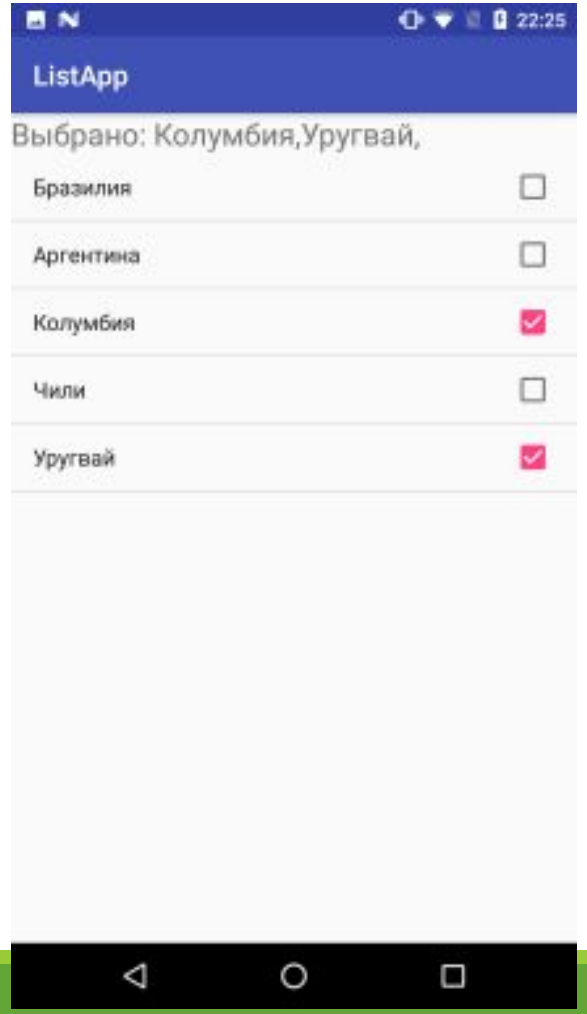
В итоге, получая индекс нажатого виджета, который соответствует индексу элемента в массиве строк, мы устанавливаем его текст в качестве текста элемента TextView (selection.setText(countries[position])).

Множественный выбор в списке

Иногда требуется выбрать не один элемент, как по умолчанию, а несколько. Для этого, во-первых, в разметке списка надо установить атрибут android:choiceMode="multipleChoice":

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/selection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="22sp" />
    <ListView
        android:choiceMode="multipleChoice"
        android:id="@+id/countriesList"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Теперь определим в коде MainActivity обработку выбора элементов списка:

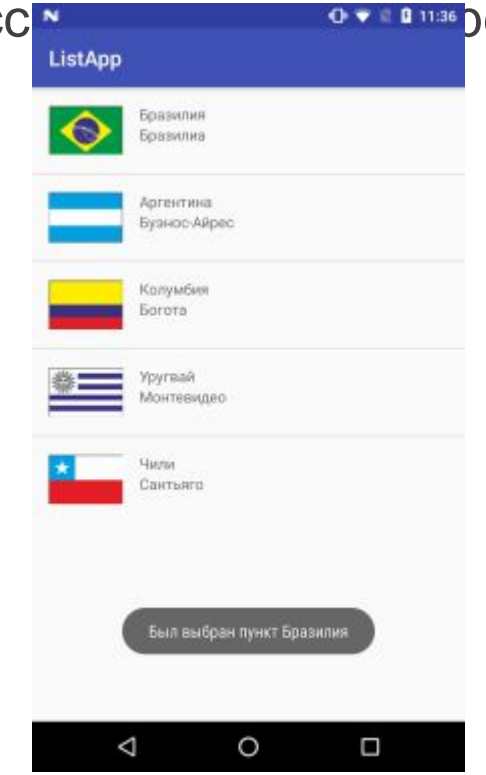



```
Public class MainActivity extends AppCompatActivity {

    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили", "Уругвай"};
    TextView selection;
    ListView countriesList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // получаем элемент TextView
        selection = (TextView) findViewById(R.id.selection);
        // получаем элемент ListView
        countriesList = (ListView) findViewById(R.id.countriesList);

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_multiple_choice, countries);
        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
        // добвляем для списка слушатель
        countriesList.setOnItemClickListener(new OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position, long id)
            {
                SparseBooleanArray sp=countriesList.getCheckedItemPositions();

                String selectedItems="";
                for(int i=0;i < countries.length;i++)
                {
                    if(sp.get(i))
                        selectedItems+=countries[i]+",";
                }
                // установка текста элемента TextView
                selection.setText("Выбрано: " + selectedItems);
            }
        });
    }
}
```

Расширение списков и создание адаптера

Традиционные списки ListView, использующие стандартные адаптеры ArrayAdapter, прекрасно работают с массивами строк. Однако чаще мы будем сталкиваться с более сложными по структуре списками, где один элемент представляет не одну строку, а несколько строк, картинок и других компонентов.

Расс ре.




```
public class State {

   private String name; // название
   private String capital;  // столица
   private int flagResource; // ресурс флага

   public State(String name, String capital, int flag){

      this.name=name;
      this.capital=capital;
      this.flagResource=flag;
   }

   public String getName() {
      return this.name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getCapital() {
      return this.capital;
   }

   public void setCapital(String capital) {
      this.capital = capital;
   }

   public int getFlagResource() {
      return this.flagResource;
   }

   public void setFlagResource(int flagResource) {
      this.flagResource = flagResource;
   }
}
```

добавим в папку *res/layout* новый файл **list_item.xml**, который будет представлять разметку однго элемента в списке:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="16dp">
    <ImageView
        android:id="@+id/flag"
        android:layout_marginRight="16dp"
        android:layout_marginEnd="16dp"
        android:layout_width="70dp"
        android:layout_height="50dp" />

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Название" />

        <TextView
            android:id="@+id/capital"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Столица" />

    </LinearLayout>
</LinearLayout>
```

добавим в каталог, где находятся классы MainActivity и State, новый класс, который назовем StateAdapter:

```
public class StateAdapter extends ArrayAdapter<State> {

    private LayoutInflater inflater;
    private int layout;
    private List<State> states;

    public StateAdapter(Context context, int resource, List<State> states) {
        super(context, resource, states);
        this.states = states;
        this.layout = resource;
        this.inflater = LayoutInflater.from(context);
    }
    public View getView(int position, View convertView, ViewGroup parent) {

        View view=inflater.inflate(this.layout, parent, false);

        ImageView flagView = (ImageView) view.findViewById(R.id.flag);
        TextView nameView = (TextView) view.findViewById(R.id.name);
        TextView capitalView = (TextView) view.findViewById(R.id.capital);

        State state = states.get(position);

        flagView.setImageResource(state.getFlagResource());
        nameView.setText(state.getName());
        capitalView.setText(state.getCapital());

        return view;
    }
}
```

Все взаимодействие со списком здесь будет идти через класс StateAdapter. В конструкторе StateAdapter нам надо передать в конструктор базового класса три параметра:

контекст, в котором используется класс. В его роли кк правило выступает класс Activity

ресурс разметки интерфейса, который будет использоваться для создания одного элемента в ListView

набор объектов, которые будут выводиться в ListView

В конструкторе StateAdapter мы получаем ресурс разметки и набор объекто и сохраняем их в отдельные переменные. Кроме того, для создания объекта View по полученному ресурсу разметки потребуется объект LayoutInflater, который также сохраняется в переменную.

В методе getView() устанавливается отображение элемента списка. Данный метод принимает три параметра:

position: передает позицию элемента внутри адаптера, для которого создается представление

convertView: старое представление элемента, которое при наличии используется ListView в целях оптимизации

parent: родительский компонент для представления элемента

в файле MainActivity соединим StateAdapter с ListView:

```
public class MainActivity extends AppCompatActivity {

    private List<State> states = new ArrayList();

    ListView countriesList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

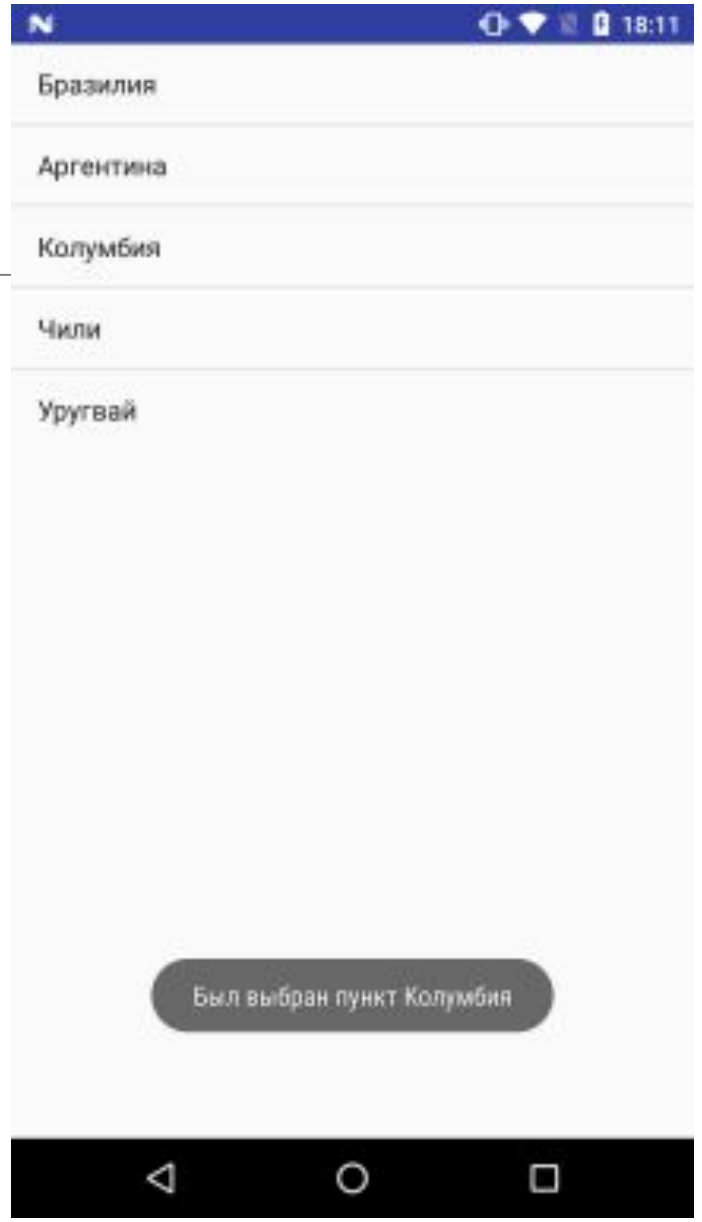
```
        // начальная инициализация списка
        setInitialData();
        // получаем элемент ListView
        countriesList = (ListView) findViewById(R.id.countriesList);
        // создаем адаптер
        StateAdapter stateAdapter = new StateAdapter(this, R.layout.list_item, states);
        // устанавливаем адаптер
        countriesList.setAdapter(stateAdapter);
        // слушатель выбора в списке
        AdapterView.OnItemClickListener itemListener = new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {

                // получаем выбранный пункт
                State selectedState = (State)parent.getItemAtPosition(position);
                Toast.makeText(getApplicationContext(), "Был выбран пункт " + selectedState.getName(),
                        Toast.LENGTH_SHORT).show();
            }
        };
        countriesList.setOnItemClickListener(itemListener);
    }
    private void setInitialData(){

        states.add(new State ("Бразилия", "Бразилиа", R.drawable.brazilia));
        states.add(new State ("Аргентина", "Буэнос-Айрес", R.drawable.argentina));
        states.add(new State ("Колумбия", "Богота", R.drawable.columbia));
        states.add(new State ("Уругвай", "Монтевидео", R.drawable.uruguai));
        states.add(new State ("Чили", "Сантьяго", R.drawable.chile));
    }
}
```

ListActivity

Для упрощения доступа к элементам списка используется класс **ListActivity**. ListActivity представляет собой класс, унаследованный от Activity и разработанный специально для работы со списками.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">

   <ListView android:id="@android:id/list"
      android:layout_height="match_parent"
      android:layout_width="match_parent" />
</LinearLayout>
```

```
Public class MainActivity extends ListActivity {

    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили", "Уругвай"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, countries);
        setListAdapter(adapter);

        AdapterView.OnItemClickListener itemListener = new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {

                Toast.makeText(getApplicationContext(), "Был выбран пункт " +
                        parent.getItemAtPosition(position).toString(),
Toast.LENGTH_SHORT).show();
            }
        };
        getListView().setOnItemClickListener(itemListener);
    }
}
```



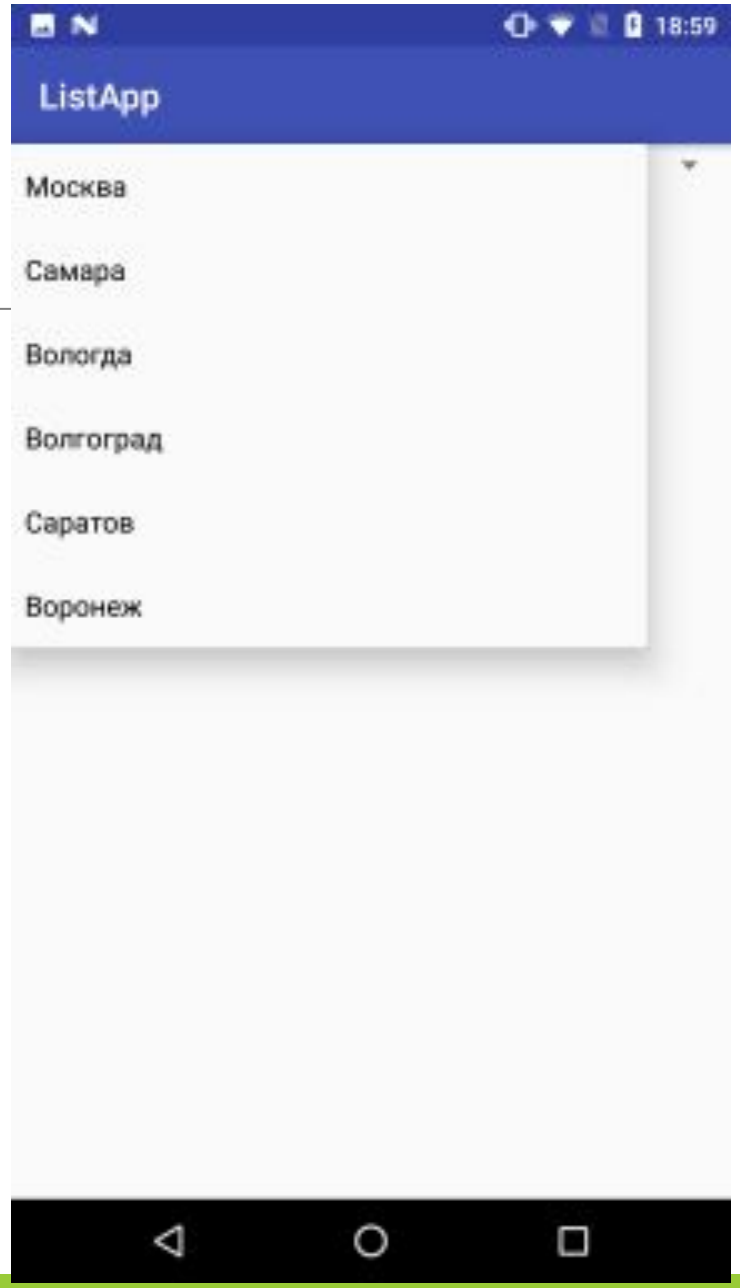
Выпадающий список Spinner

Spinner представляет собой выпадающий список. Определим в файле разметки activity_main.xml элемент Spinner:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Spinner
        android:id="@+id/cities"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

В качестве источника данных, как и для ListView, для Spinner может служить простой список или массив, соданный программно, либо ресурс string-array. Взаимодействие с источником данных также будет идти через адаптер

```
public class MainActivity extends AppCompatActivity {

    String[] cities = {"Москва", "Самара", "Вологда",
"Волгоград", "Саратов", "Воронеж"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Spinner spinner = (Spinner)
findViewById(R.id.cities);
        // Создаем адаптер ArrayAdapter с помощью массива
строк и стандартной разметки элемета spinner
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, cities);
        // Определяем разметку для использования при выборе
элемента
        adapter.setDropDownViewResource(android.R.layout.simp
le_spinner_dropdown_item);
        // Применяем адаптер к элементу spinner
        spinner.setAdapter(adapter);
    }
}
```



Обработка выбора элемента

Метод onItemSelected получает четыре параметра:

1. parent: объект Spinner, в котором произошло событие выбора элемента
2. view: объект View внутри Spinnera, который представляет выбранный элемент
3. position: индекс выбранного элемента в адаптере
4. id: идентификатор строки того элемента, который был выбран

Получив позицию выбранного элемента, мы можем найти его в списке:

```
String item = (String)parent.getItemAtPosition(position);
```

Виджет автодополнения AutoCompleteTextView

AutoCompleteTextView представляет элемент, созданный на основе класса EditText и обладающий возможностью автодополнения.

Во-первых, объявим в ресурсе разметке данный элемент:

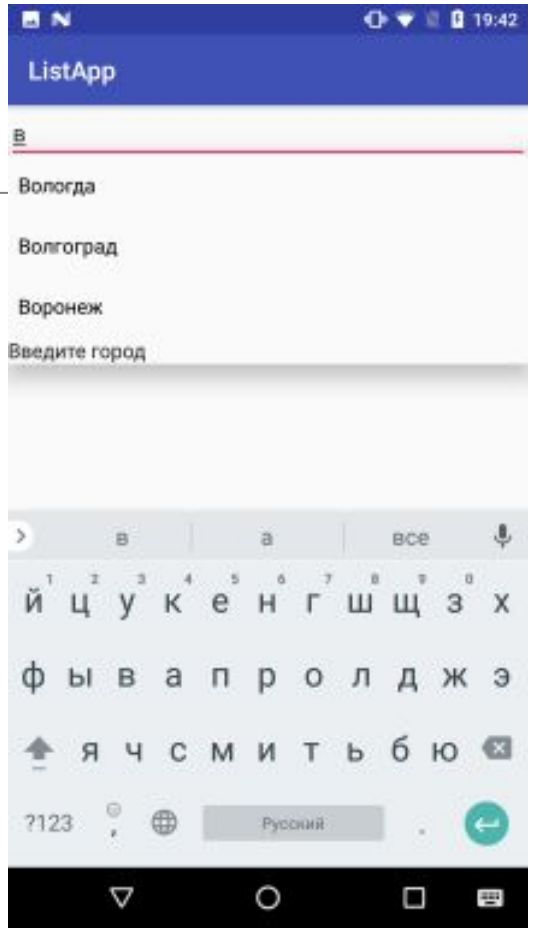
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <AutoCompleteTextView
        android:id="@+id/autocomplete"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionHint="Введите город"
        android:completionThreshold="1"/>

</LinearLayout>
```

Атрибут android:completionHint позволяет задать надпись, которая отображается внизу списка, а свойство android:completionThreshold устанавливает, какое количество символов надо ввести, чтобы начало работать автодополнение. То есть в данном случае уже после ввода одного символа должен появться список с подстановками.

Как и в случае с элементами ListView и Spinner, AutoCompleteTextView подключается к источнику данных через адаптер. Источником данных опять же может служить массив или список объектов, либо ресурс string-array.

Теперь подключим к виджету массив строк в классе MainActivity:

```
public class MainActivity extends AppCompatActivity {

    String[] cities = {"Москва", "Самара", "Вологда", "Волгоград", "Саратов", "Воронеж"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Получаем ссылку на элемент AutoCompleteTextView в разметке
        AutoCompleteTextView autoCompleteTextView = (AutoCompleteTextView)
findViewById(R.id.autocomplete);
        // Создаем адаптер для автозаполнения элемента AutoCompleteTextView
        ArrayAdapter<String> adapter =
                                new ArrayAdapter<String>(this,
R.layout.support_simple_spinner_dropdown_item, cities);
        autoCompleteTextView.setAdapter(adapter);
    }
}
```



*****MultiAutoCompleteTextView – Найти и рассмотреть самостоятельно.**
MultiAutoCompleteTextView позволяет использовать автодополнения не только для одной строки, но и для отдельных слов. Например, если вводится слово и после него ставится запятая, то автозаполнение все равно будет работать для вновь вводимых слов после запятой или другого разделителя

Меню

Меню в приложениях представляет класс **android.view.Menu**, и каждая activity ассоциируется с объектом этого типа. Объект android.view.Menu может включать различное количество элементов, а те в свою очередь могут хранить подэлементы.

Определение меню в xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/
android">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="1"
        android:title="Настройки" />
    <item
        android:id="@+id/save_settings"
        android:orderInCategory="3"
        android:title="Сохранить" />
    <item
        android:id="@+id/open_settings"
        android:orderInCategory="2"
        android:title="Открыть" />
</menu>
```

Тег <menu> является корневым узлом файла и определяет меню, состоящее из одного или нескольких элементов <item> и <group>.

Элемент <item> включает следующие атрибуты, которые определяют его внешний вид и поведение:

1. android:id: уникальный id элемента меню, который позволяет его опознать при выборе пользователем и найти через поиск ресурса по id
2. android:icon: ссылка на ресурс drawable, который задает изображение для элемента (android:icon="@drawable/ic_help")
3. android:title: ссылка на ресурс строки, содержащий заголовок элемента. По умолчанию имеет значение "Settings"
4. android:orderInCategory: порядок следования элемента в меню

Наполнение меню элементами

```
public class MainActivity extends AppCompatActivity
{

    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.main_menu,
menu);
        return true;
    }
}
```

Метод getMenuInflater получает объект MenuInflater и вызываем его метод inflate(). Этот метод в качестве первого параметра принимает ресурс, представляющий наше декларативное описание меню в xml, и наполняет им объект menu, переданный в качестве второго параметра.

Обработка нажатий в меню

Если мы нажмем на любой из пунктов меню, то ничего не произойдет. Чтобы привязать к меню действия, нам надо переопределить в классе activity **onOptionsItemSelected**.

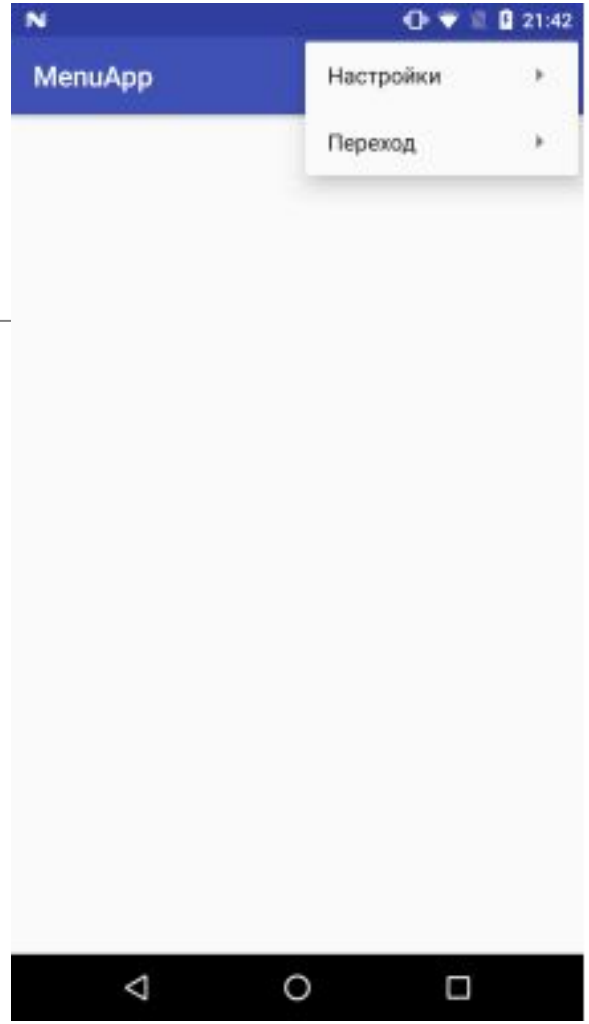
Чтобы понять, какой пункт меню выбран, вначале получаем его идентификатор int id = item.getItemId(). Затем пробегаемся в конструкции switch..case и выбираем нужный вариант и в зависимости от выбора производим определенные действия - в данном случае устанавливаем текст TextView.

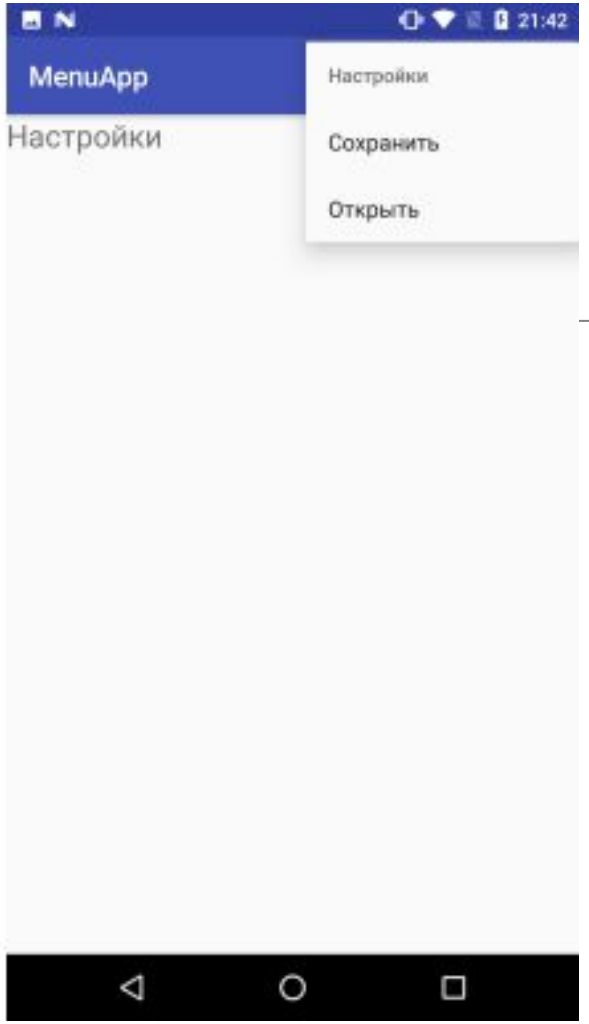
```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        TextView headerView = (TextView) findViewById(R.id.header);
        switch(id){
            case R.id.action_settings :
                headerView.setText("Настройки");
                return true;
            case R.id.open_settings:
                headerView.setText("Открыть");
                return true;
            case R.id.save_settings:
                headerView.setText("Сохранить");
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Создание подменю

Для создания подменю в файле разметки меню определим внутренний элемент menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_settings"
        android:title="Настройки">
        <menu>
            <item android:id="@+id/save_settings"
                android:title="Сохранить" />
            <item android:id="@+id/open_settings"
                android:title="Открыть" />
        </menu>

    </item>
    <item
        android:id="@+id/action_move"
        android:title="Переход">
        <menu>
            <item android:id="@+id/forward"
                android:title="Вперед" />
            <item android:id="@+id/back"
                android:title="Назад" />
        </menu>

    </item>
</menu>
```





**Изучить самостоятельно Группы в меню

Упражнения

1. Реализовать приложение использующее список и список с кнопками
2. Реализовать виджет автодополнения (для одной строки и для отдельных слов)
3. Реализовать меню в приложении