КАЛУЖСКИЙ ФИЛИАЛ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Э. БАУМАНА (национальный исследовательский университет)»

Факультет «Информатика и управление"

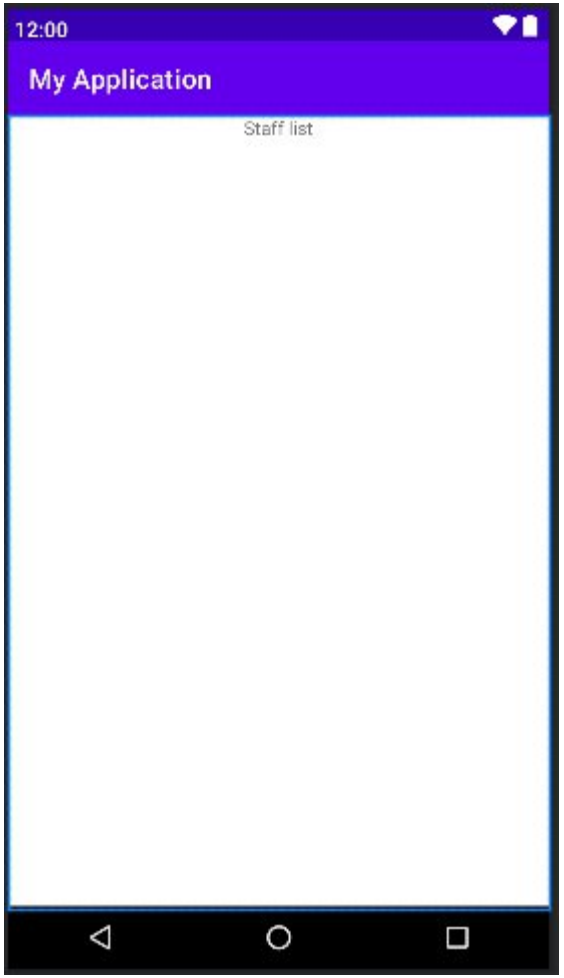
Кафедра "Программное обеспечение ЭВМ, информационные технологии"

Создание и использование списков в android-приложениях

Калуга

LayoutInflater – это класс, который из содержимого layout-файла создает View-элемент. Метод который это делает называется inflate. Используя этот класс можно создать простейший список

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Staff list"
        android:layout_gravity="center_horizontal">
    </TextView>
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/scroll">
        <LinearLayout
            android:id="@+id/linLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">
        </LinearLayout>
    </ScrollView>
</LinearLayout>
```



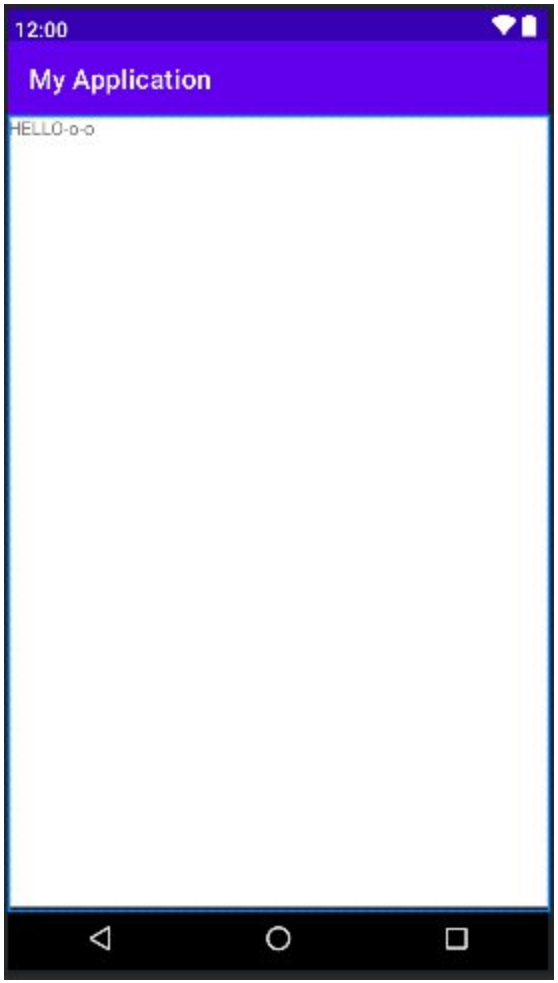
item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="10dp">
    <TextView
        android:id="@+id/tvName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:layout_gravity="top|center_horizontal"
        android:textSize="24sp">
    </TextView>
    <TextView
        android:id="@+id/tvPosition"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:layout_gravity="bottom|left"
        android:layout_marginLeft="5dp">
    </TextView>
    <TextView
        android:id="@+id/tvSalary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:layout_gravity="bottom|right"
        android:layout_marginRight="5dp">
    </TextView>
</FrameLayout>
```

```
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    colors[0] = Color.parseColor("#559966CC");
    colors[1] = Color.parseColor("#55336699");

    LinearLayout linLayout = (LinearLayout) findViewById(R.id.linLayout);

    LayoutInflater ltInflater = getLayoutInflater();

    for (int i = 0; i < name.length; i++) {
      Log.d("myLogs", "i = " + i);
      View item = ltInflater.inflate(R.layout.item, linLayout, false);
      TextView tvName = (TextView) item.findViewById(R.id.tvName);
      tvName.setText(name[i]);
      TextView tvPosition = (TextView) item.findViewById(R.id.tvPosition);
      tvPosition.setText("Должность: " + position[i]);
      TextView tvSalary = (TextView) item.findViewById(R.id.tvSalary);
      tvSalary.setText("Оклад: " + String.valueOf(salary[i]));
      item.getLayoutParams().width = LayoutParams.MATCH_PARENT;
      item.setBackgroundColor(colors[i % 2]);
      linLayout.addView(item);
    }
  }
}
```

При создании **ListView** создавать пункты за нас будет **адаптер**. Адаптеру нужны от нас **данные** и **layout-ресурс** пункта списка. Далее мы присваиваем **адаптер** списку **ListView**. Список при построении запрашивает у адаптера пункты, адаптер их создает (используя данные и layout) и возвращает списку. В итоге мы видим готовый список.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello">
    </TextView>
    <ListView
        android:id="@+id/lvMain"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </ListView>
</LinearLayout>
```



```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  // находим список
  ListView lvMain = (ListView) findViewById(R.id.lvMain);

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

  // присваиваем адаптер списку
  lvMain.setAdapter(adapter);

}
```

Адаптер для ExpandableListView

```
groupData = new ArrayList<Map<String, String>>();
for (String group : groups) {
  // заполняем список атрибутов для каждой группы
  m = new HashMap<String, String>();
  m.put("groupName", group); // имя компании
  groupData.add(m);
}
// список атрибутов групп для чтения
String groupFrom[] = new String[] {"groupName"};
// список ID view, в которые будет помещены
атрибуты групп
int groupTo[] = new int[] {android.R.id.text1};
// создаем коллекцию для коллекций элементов
childData = new ArrayList<ArrayList<Map<String,
String>>>();
// создаем коллекцию элементов для первой группы
childDataItem = new ArrayList<Map<String, String>>();
// заполняем список атрибутов для каждого
элемента
for (String phone : phonesHTC) {
   m = new HashMap<String, String>();
   m.put("phoneName", phone); // название телефона
   childDataItem.add(m);
}
// добавляем в коллекцию коллекций
childData.add(childDataItem);
// создаем коллекцию элементов для второй группы
childDataItem = new ArrayList<Map<String, String>>();
for (String phone : phonesSams) {
   m = new HashMap<String, String>();
   m.put("phoneName", phone);
   childDataItem.add(m);
}
childData.add(childDataItem);
// список атрибутов элементов для чтения
String childFrom[] = new String[] {"phoneName"};
// список ID view, в которые будет помещены атрибуты
int childTo[] = new int[] {android.R.id.text1};
SimpleExpandableListAdapter adapter = new
  SimpleExpandableListAdapter(
          this,
          groupData,
          android.R.layout.simple_expandable_list_item_1,
          groupFrom,
          groupTo,
          childData,
          android.R.layout.simple_list_item_1,
          childFrom,
          childTo);
elvMain = (ExpandableListView) findViewById(R.id.elvMain);
elvMain.setAdapter(adapter);
```

Интерфейс Adapter. Описывает базовые методы, которые должны содержать адаптеры: getCount, getItem, getView и пр.

Интерфейс ListAdapter. Этот интерфейс должен быть реализован адаптером, который будет использован в ListView (метод setAdapter). Содержит описание методов для работы с разделителями(separator) списка.

Интерфейс SpinnerAdapter. Адаптеры, реализующие этот интерфейс, используются для построения Spinner(выпадающий список или drop-down). Содержит метод getDropDownView, который возвращает элемент выпадающего списка.

Интерфейс WrapperListAdapter. Адаптеры, наследующие этот интерфейс используются для работы с вложенными адаптерами. Содержит метод getWrappedAdapter, который позволяет вытащить из основного адаптера вложенный.

Класс HeaderViewListAdapter. Готовый адаптер для работы с Header и Footer. Внутри себя содержит еще один адаптер (ListAdapter), который можно достать с помощью выше рассмотренного метода getWrappedAdapter из интерфейса WrapperListAdapter.

Абстрактный класс BaseAdapter. Содержит немного своих методов и реализует методы интерфейсов, которые наследует, но не все. Своим наследникам оставляет на обязательную реализацию методы: getView, getItemId, getItem, getCount из ListAdapter.

Класс ArrayAdapter<T>. Готовый адаптер, который мы уже использовали. Принимает на вход список или массив объектов, перебирает его и вставляет строковое значение в указанный TextView. Кроме наследуемых методов содержит методы по работе с коллекцией данных – add, insert, remove, sort, clear и метод setDropDownViewResource для задания layout-ресурса для отображения пунктов выпадающего списка.

Класс SimpleAdapter. Также готовый к использованию адаптер. Принимает на вход список Map-объектов, где каждый Map-объект – это список атрибутов. Кроме этого на вход принимает два массива – from[] и to[]. В to указываем id экранных элементов, а в from имена(key) из объектов Map, значения которых будут вставлены в соответствующие (из from) экранные элементы.

Абстрактный класс CursorAdapter. Реализует абстрактные методы класса BaseAdapter, содержит свои методы по работе с курсором и оставляет наследникам методы по созданию и наполнению View: newView, bindView.

Абстрактный класс ResourceCursorAdapter. Содержит методы по настройке используемых адаптером layout-ресурсов. Реализует метод newView из CursorAdapter.

Класс SimpleCursorAdapter. Готовый адаптер, похож, на SimpleAdapter. Только использует не набор объектов Map, а Cursor, т.е. набор строк с полями. Соответственно в массив from[] вы заносите наименования полей, значения которых хотите вытащить в соответствующие View из массива to.