02

Основные компоненты Android

Nikolay Moskvin
moskvin@sibext.com

19 октября 2013 г.

Содержание

MVC

sibext company

В Android нет четкого разделения и некоторые компоненты могут выполнять сразу несколько ролей. Нужно избегать смешивания этого, чтобы код можно было переиспользовать

- Model (Данные)
- View (Видно конечному пользователю)
- Controller (Процессы, которые не видно конечному пользователю)

Observer

sibext company

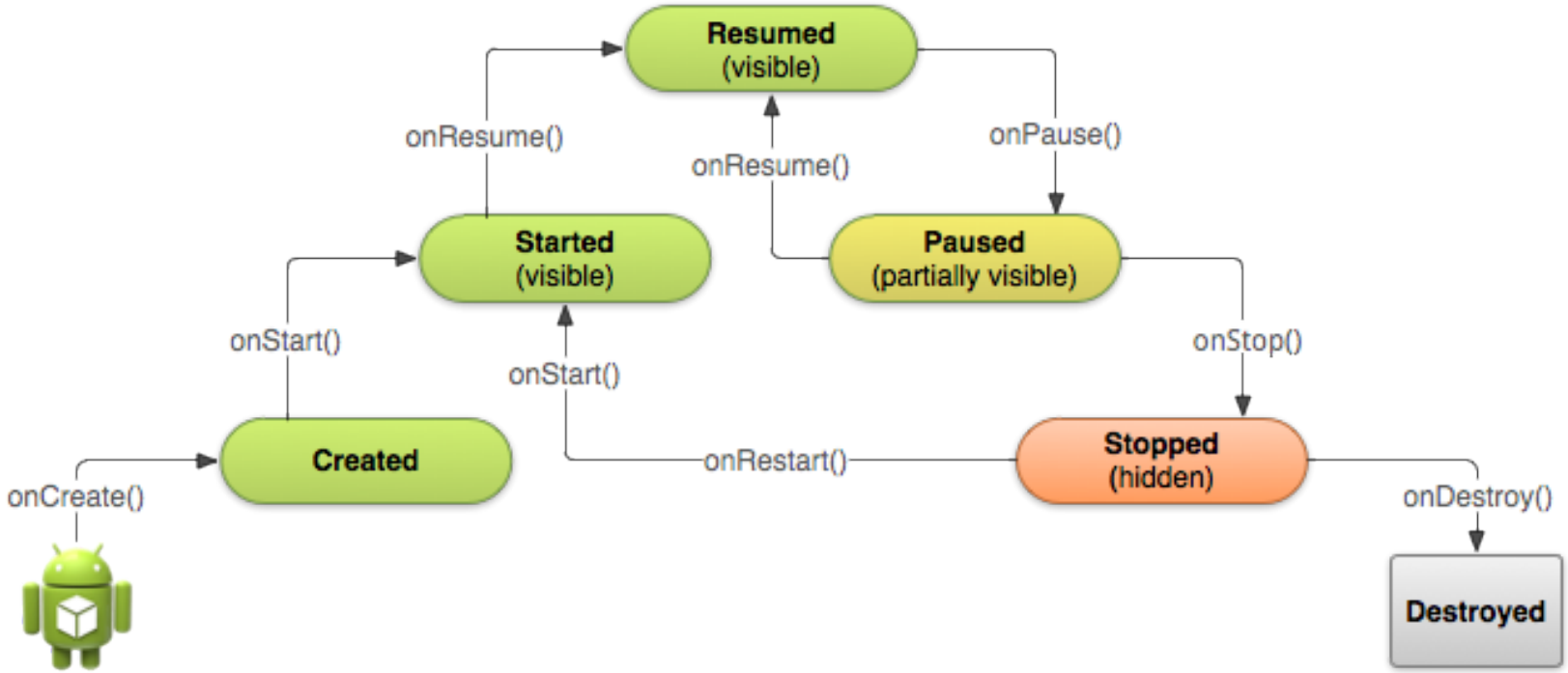
Помогает разграничить зоны ответственности между классами.

```
public class ExampleView extends View {
    public interface OnExampleListener {
        void onExample(View root, String example);
    }
    private OnExampleListener l;

    public ExampleView(Context context) {
        super(context);
        setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if (l != null) {
                    l.onExample(v, "Example!");
                }
            }
        });
    }
    public void setOnExampleListener(OnExampleListener l) {
        this.l = l;
    }
}
```

Activity

- Пирамида жизненного цикла



Activity

sibext company

- Главный метод

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

- Доступ к элементам UI

```
View view = findViewById(R.id.progress);
ViewGroup viewGroup = (ViewGroup)findViewById(R.id.root);
```

- Верстка

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/root" ... >
 <ProgressBar android:id="@+id/progress" ... />
</RelativeLayout>
```

Основные методы Activity

- **onCreate** – создание, инициализация
- **onPause, onStop** – сохранение важной информации
- **finish** – завершение Activity
- **setContentView, findViewById** – связь с интерфейсом и получение конкретного View
- **startActivity, startActivityForResult** – запуск другой Activity
- **setResult** – передаем результат обработки Activity
- **onActivityResult** – обработка результатов работы другой Activity

Activity

sibext company

- Точка входа в приложение

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ...
  package="com.sibext.skeleton"
  android:versionCode="1"
  android:versionName="1.0">
  ...
  <application android:label="@string/app_name" android:icon="@drawable/icon">
     ...
    <activity android:name=".MainActivity"
        android:label="@string/app_name"
        android:icon="@drawable/icon">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    ...
    <activity android:name=".act.SecondActivity"
    ...
    <activity android:name="com.example.Activity"
    ...

  </application>
...
</manifest>
```

Intent

Связующий элемент многих компонентов Android приложений (activities, services и broadcast receivers) Включает в себя:

- **Action** – общее описание действия
- **Data** – основные данные для обработки
- **Category** – дополнительная информация о действии
- **Extras и Flags** – дополнительные данные

Guideline: `http://developer.android.com/guide/components/intents-filters.html`

Запуск Activity

sibext company

- Явный

```
startActivity(new Intent(this, SecondActivity.class));
```

- Неявный
  - Запуск браузера

```
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse("http://developer.android.com"));
startActivity(i);
```

  - Отсылка письма

```
Intent i = new Intent(Intent.ACTION_SEND);
i.setType("text/plain");
i.putExtra(Intent.EXTRA_EMAIL,
    new String[]{"moskvin@sibext.com"});
i.putExtra(Intent.EXTRA_SUBJECT,
    "subject␣of␣email");
i.putExtra(Intent.EXTRA_TEXT,
    "body␣of␣email");
startActivity(Intent.createChooser(i, "Sending"));
```

Опрос результата

sibext company

- Запускаем с кодом

```
private static final int REQUEST_CODE = 404;
startActivityForResult(intent, REQUEST_CODE);
```

- Activity может отдать данные

```
setResult(RESULT_OK,
  new Intent().putExtra("DATA", "Hello World"));
```

- Ждем результата

```
@Override
protected void onActivityResult(int requestCode,
  int resultCode, Intent data) {
  switch (requestCode) {
  case REQUEST_CODE:
    if (resultCode == RESULT_OK) {
      Log.d(TAG, "" + data.getStringExtra("DATA"));
      // do something
    }
    break;
  }
}
```

Inflater

Объект класса View может быть динамически надут из xml-разметки с помощью LayoutInflater

- Надуваем

```
View.inflate(this, R.layout.my_view, null);
```

- XML ресурс

```
<ImageView
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/my_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:textColor="@color/text_gray"
    android:paddingTop="6dp"
    android:paddingBottom="6dp"
    android:textStyle="bold"
    android:scaleType="fitEnd"
    android:src="@drawable/button_focus"/>
```

Ресурсы Android

sibext company

Каждый конкретный тип ресурсов должен быть помещен в соответствующую подпапку **res**

animator/	описание анимаций
color/	со state-list для цвета
drawable/	картинки или state-list для картинок
layout/	разметка интерфейса пользователя
menu/	описание меню (элементов actionbar)
raw/	произвольные файлы
values/	определяются простые типы
xml/	произвольные xml файлы

Ресурсы Android

На основе ресурсов автоматически создается класс **R**, который можно использовать для доступа к ресурсам в коде:

- Выставляем фон для текущего экрана из drawable ресурса

```
getWindow()
  .setBackgroundDrawableResource(R.drawable.bg_image);
```

- Текст для TextView из string ресурса

```
TextView msgTextView =
  (TextView) findViewById(R.id.msg);
msgTextView.setText(R.string.hello_message);
getWindow()
  .setBackgroundDrawableResource(R.drawable.bg_image);
```

Доступ к ресурсам внутри ресурсов

sibext company

- Синтаксис

```
@[<package_name>:]<resource_type>/<resource_name>
```

- Доступ до цветов и строк

```
<EditText ...
    android:textColor="@color/red"
    android:text="@string/hello" />

...

<TextView ...
    android:textColor="@android:color/secondary_text_dark"
    android:text="@string/hello" />
```

Атрибуты и Стили

sibext company

- Стили должны лежать в **values** и назначаются с помощью @style/<название стиля>

```
<style name="NormalTextView">
    <item name="android:textColor">@color/black</item>
    <item name="android:textSize">15sp</item>
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:gravity">left</item>
</style>
...
<RelativeLayout @style="NormalTextView"/>
```

- Атрибуты подаются напрямую

```
<RelativeLayout
    android:id="@+id/right_place"
    android:layout_width="0dip"
    android:layout_height="wrap_content"
    android:layout_weight="0.40"
    android:visibility="gone"/>
```

View

Визуальный компонент, предоставляющий определенную функциональность для пользователя

- Android Framework содержит большое количество самых разнообразных View
- Все View выделены в пакет android.widget

View

- AnalogClock
- Button
- Checkbox
- Chronometer
- DatePicker
- EditText
- ImageView
- ImageButton

View

sibext company

- MapView
- Progressbar
- RatingBar
- RadioButton
- SeekBar
- TextView
- TimePicker
- WebView
- VideoView
- ...

Layouts или ViewGroup

sibext company

Используя различные комбинации стандартных Layouts, можно описать практически любой интерфейс

- FrameLayout
- LinearLayout
- TableLayout
- RelativeLayout
- TabLayout
- GridLayout

Скролл

Стандартные Layouts не предоставляют возможность листать элементы, которые не вошли на экран

- За это отвечают ViewGroups: ScrollView и HorizontalScrollView
- ScrollViews поддерживают только один вложенный элемент
- Для реализации скроллинга в двух направлениях HorizontalScrollView вкладывается в ScrollView

В следующей лекции

sibext company

- Рассмотрим понятие Adapter
- Recycle для элементов экрана
- Продолжим изучать основные компоненты