

```

String url;
public PintrestItem(String url, String name){
    this.url=url;
    this.name=name;
}
public String getUrl() {
    return url;
}
public String getName(){
    return name;
}
String name;
}

```

### 3. Создайте файл компоновки для размещения элементов RecyclerView:

```

<ImageView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:adjustViewBounds="true"
    android:scaleType="centerCrop"
    android:id="@+id/imageView"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:id="@+id/name"
    android:layout_gravity="center"
    android:textColor="@android:color/white"/>

```

### 4. Создайте класс адаптера для RecyclerView:

```

public class PintrestAdapter extends RecyclerView.Adapter<PintrestAdapter.PintrestViewHolder>{
    private ArrayList<PintrestItem> images; Picasso picasso;
    КОНТЕКСТНЫЙ КОНТЕКСТ;
    public PintrestAdapter(ArrayList<PintrestItem> images, Context context){
        this.images=images;
        picasso=Picasso.with(context);
        this.context=context;
    }

    @Override
    public PintrestViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view=LayoutInflater.from(parent.getContext()).inflate(R.layout.pintrest_
        layout_item,parent,false);
        return new PintrestViewHolder(view);
    }

    @Override
    public void onBindViewHolder(PintrestViewHolder holder, int position) {
        picasso.load(images.get(position).getUrl()).into(holder.imageView);
        holder.tv.setText(images.get(position).getName());
    }

    @Override
    public int getItemCount() {

```

```

        return images.size();
    }

    public class PintrestViewHolder extends RecyclerView.ViewHolder{
        ImageView imageView;
        TextView tv;
        public PintrestViewHolder(View itemView) {
            super(itemView);
            imageView=(ImageView)itemView.findViewById(R.id.imageView);
            tv=(TextView)itemView.findViewById(R.id.name);
        }
    }
}

```

5. Инстанцируйте RecyclerView в своей активности или фрагменте:

```

RecyclerView recyclerView = (RecyclerView)findViewById(R.id.recyclerView);
// Создаем экземпляр StaggeredGridLayoutManager с двумя строками (количеством
диапазонов) и задаем ориентацию
StaggeredGridLayoutManager layoutManager=new new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);
// Создание фиктивных данных и добавление их в список List<PintrestItem>
List<PintrestItem>items=new ArrayList<PintrestItem>
items.add(new PintrestItem("url изображения, которое вы хотите показать",
"imagename"));
items.add(new PintrestItem("url изображения, которое вы хотите показать",
"imagename"));
items.add(new PintrestItem("url изображения, которое вы хотите показать",
"imagename"));
recyclerView.setAdapter(new PintrestAdapter(items,getContext()));

```

Не забудьте добавить зависимость Picasso в файл build.gradle:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

## Глава 20. Пагинация в RecyclerView

Пагинация является общей проблемой для многих мобильных приложений, которым приходится работать со списками данных. В настоящее время большинство мобильных приложений переходит на модель “бесконечной страницы”, когда при прокрутке автоматически загружается новый контент. CWAC Endless Adapter позволяет очень просто использовать этот паттерн в Android-приложениях.

### 20.1. MainActivity.java

```

import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;

```

```
import android.util.Log;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";
    private Toolbar toolbar;

    private TextView tvEmptyView;
    private RecyclerView mRecycler View;
    private DataAdapter mAdapter;
    private LinearLayoutManager mLayoutManager;
    private int mStart=0,mEnd=20;
    private List<Student> studentList;
    private List<Student> mTempCheck;
    public static int pageNumber;
    public int total_size=0;

    protected Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pageNumber = 1;
        toolbar = (Toolbar) findViewById(R.id.toolbar);
        tvEmptyView = (TextView) findViewById(R.id.empty_view);
        mRecycler View = (RecyclerView) findViewById(R.id.my_recycler_view);
        studentList = new ArrayList<>();
        mTempCheck=new ArrayList<>();
        handler = new Handler();
        if (toolbar != null) {
            setSupportActionBar(toolbar);
            getSupportActionBar().setTitle("Android Students");
        }
        mRecycler View.setHasFixedSize(true);
        mLayoutManager = new LinearLayoutManager(this);
        mRecycler View.setLayoutManager(mLayoutManager);
        mAdapter = new DataAdapter(studentList, mRecycler View);
        mRecycler View.setAdapter(mAdapter);
        GetGroupData(" " + mStart, " " + mEnd);
        mAdapter.setOnLoadMoreListener(new OnLoadMoreListener() {
            @Override
```

```

        public void onLoadMore() {
            if( mTempCheck.size() > 0 ) {
                studentList.add(null);
                mAdapter.notifyItemInserted(studentList.size() - 1);
                int start = pageNumber * 20;
                start = start + 1;
                ++ pageNumber;
                mTempCheck.clear();
                GetData(" " + start, " " + mEnd);
            }
        }
    });
}

public void GetData(final String LimitStart, final String LimitEnd) {
    Map<String, String> params = new HashMap<>();
    params.put("LimitStart", LimitStart);
    params.put("Limit", LimitEnd);
    Custom_Volly_Request jsonObjReq = new Custom_Volly_Request(Request.Method.
    POST, "Your php file link", params,
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                Log.d("ResponseSuccess", response.toString());
                // обработка данных от сервопривода
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                VolleyLog.d("ResponseErrorVolly: " + error.getMessage());
            }
        });
}

// загрузка исходных данных
private void loadData(int start,int end,boolean notifyadapter) {
    for (int i = start; i <= end; i++) {
        studentList.add(new Student("Student " + i, "androidstudent" + i + "@
        gmail.com"));
        if(notifyadapter)
            mAdapter.notifyItemInserted(studentList.size());
    }
}
}

```

**OnLoadMoreListener.java**

```
public interface OnLoadMoreListener { void onLoadMore(); }
```

**DataAdapter.java**

```

import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;

```

```
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class DataAdapter extends RecyclerView.Adapter {
    private final int VIEW_ITEM = 1;
    private final int VIEW_PROG = 0;

    private List<Student> studentList;

    // Минимальное количество элементов, которое должно находиться ниже текущей
    // позиции прокрутки перед загрузкой.
    private int visibleThreshold = 5;
    private int lastVisibleItem, totalItemCount;
    private boolean loading;
    private OnLoadMoreListener onLoadMoreListener;

    public DataAdapter(List<Student> students, RecyclerView recyclerView) {
        studentList = students;
        if (recyclerView.getLayoutManager() instanceof LinearLayoutManager) {
            final LinearLayoutManager linearLayoutManager = (LinearLayoutManager)
                recyclerView.getLayoutManager();
            recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
                @Override
                public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
                    super.onScrolled(recyclerView, dx, dy);
                    totalItemCount = linearLayoutManager.getItemCount();
                    lastVisibleItem = linearLayoutManager.
                        findLastVisibleItemPosition();
                    if (!loading && totalItemCount <= (lastVisibleItem +
                        visibleThreshold)) {
                        if (onLoadMoreListener != null) {
                            onLoadMoreListener.onLoadMore();
                        }
                        loading = true;
                    }
                }
            });
        }
    }

    @Override
    public int getItemViewType(int position) {
        return studentList.get(position) != null ? VIEW_ITEM : VIEW_PROG;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        RecyclerView.ViewHolder vh;
        if (viewType == VIEW_ITEM) {
            View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_
                row, parent, false);
            vh = new StudentViewHolder(v);
        } else {
            vh = new ProgressViewHolder(v);
        }
        return vh;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        if (holder instanceof StudentViewHolder) {
            StudentViewHolder vh = (StudentViewHolder) holder;
            vh.name.setText(studentList.get(position).name);
            vh.progress.setProgress((int) (studentList.get(position).progress * 100));
        }
    }

    @Override
    public int getItemCount() {
        return studentList.size();
    }
}
```

```

    } else {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.
        progress_item, parent, false);
        vh = new ProgressViewHolder(v);
    }
    return vh;
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    if (holder instanceof StudentViewHolder) {
        Student singleStudent=studentList.get(position);
        ((StudentViewHolder) holder).tvName.setText(singleStudent.getName());
        ((StudentViewHolder) holder).tvEmailId.setText(singleStudent.getEmailId());
        ((StudentViewHolder) holder).student= singleStudent;
    } else {
        ((ProgressViewHolder) holder).progressBar.setIndeterminate(true);
    }
}

public void setLoaded(boolean state) {
    loading = state;
}

@Override
public int getItemCount() {
    return studentList.size();
}

public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener) {
    this.onLoadMoreListener = onLoadMoreListener;
}

// 
public static class StudentViewHolder extends RecyclerView.ViewHolder {
    public TextView tvName;

    public TextView tvEmailId;

    public Student student;

    public StudentViewHolder(View v) {
        super(v);
        tvName = (TextView) v.findViewById(R.id.tvName);
        tvEmailId = (TextView) v.findViewById(R.id.tvEmailId);
    }
}

public static class ProgressViewHolder extends RecyclerView.ViewHolder {
    public ProgressBar progressBar;

    public ProgressViewHolder(View v) {
        super(v);
        progressBar = (ProgressBar) v.findViewById(R.id.progressBar1);
    }
}
}

```

# Глава 21. ImageView

Параметр	Описание
<code>resId</code>	имя вашего файла изображения в папке <code>res</code> (обычно в папке <code>drawable</code> )

ImageView (`android.widget.ImageView`) – это представление для отображения и манипулирования ресурсами изображений, такими как Drawables и Bitmaps. К изображению могут быть применены некоторые эффекты, рассмотренные в этой теме. Источник изображения может быть задан в XML-файле (в папке `layout`) или программно при помощи Java-кода.

## 21.1. Установка оттенка

Устанавливает цвет тонирования для изображения. По умолчанию оттенок будет смешиваться в режиме SRC\_ATOP. Задайте оттенок с помощью XML-атрибута:

```
android:tint="#009c38"
```

**Примечание:** значение цвета должно быть в виде "#rgb", "#argb", "#rrggb" или "#aargbb".

Установка оттенка программно:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

можно очистить этот цветовой фильтр:

```
imgExample.clearColorFilter();
```

## 21.2. Установка альфа-канала

Альфа-канал используется для задания непрозрачности изображения. Задать его можно с помощью атрибута XML:

```
android:alpha="0.5"
```

**Примечание:** принимает значение с плавающей точкой от 0 (прозрачное изображение) до 1 (полностью видимое).

Установка альфа-канала программно:

```
imgExample.setAlpha(0.5f);
```



## 21.3. Установка типа масштабирования

Управляет изменением размера или перемещением изображения в соответствии с размерами ImageView.

XML-атрибут:

```
android:scaleType="..."
```

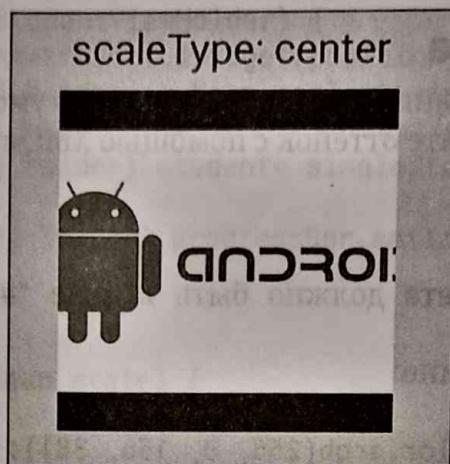
Проиллюстрируем различные типы масштабов на примере квадратного ImageView, который имеет черный фон, а мы хотим отобразить в ImageView прямоугольный drawable-объект на белом фоне.

```
<ImageView  
    android:id="@+id/imgExample"
```

```
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:background="#000"
    android:src="@drawable/android2"
    android:scaleType="..."/>
```

`scaleType` может принимать одно из следующих значений:

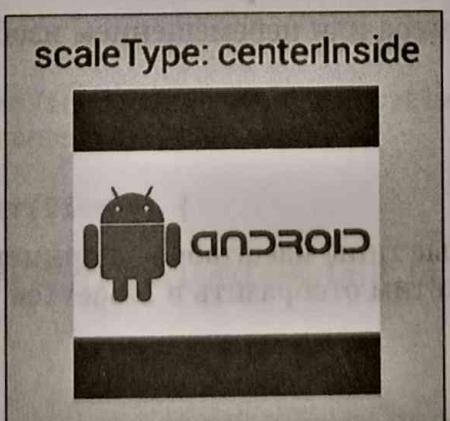
1. `center`: выравнивает изображение по центру, но не выполняет масштабирования.



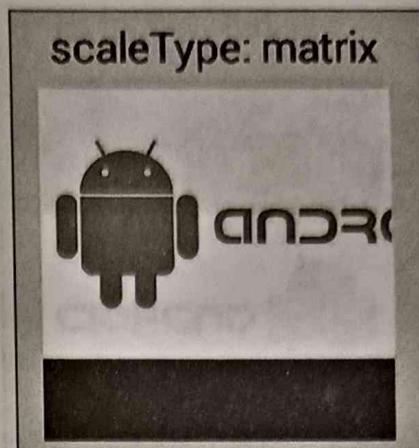
2. `centerCrop`: равномерно масштабирует изображение (сохраняя его соотношение сторон), так что оба размера (ширина и высота) изображения будут равны или больше соответствующего размера представления (за вычетом подкладок). После этого изображение центрируется в представлении.



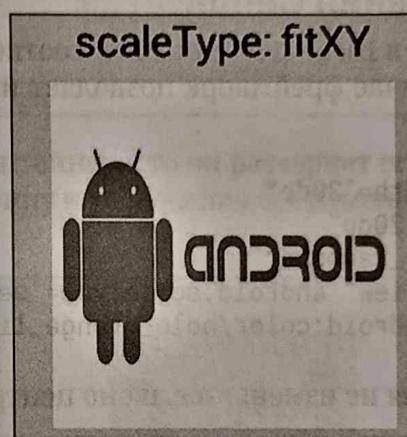
3. `centerInside`: равномерно масштабирует изображение (сохраняя его соотношение сторон) таким образом, чтобы оба размера (ширина и высота) изображения были равны или меньше соответствующих размеров представления (минус подкладки). После этого изображение центрируется в представлении.



4. matrix: масштабирование с использованием матрицы изображения при отрисовке.



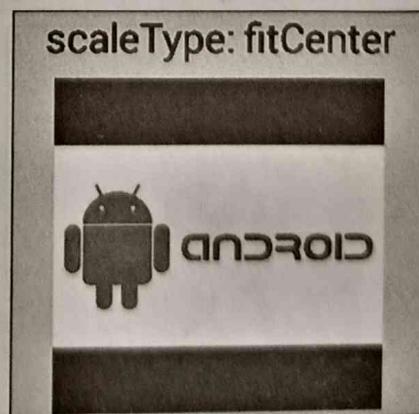
5. fitXY: масштабирование изображения с помощью функции FILL.



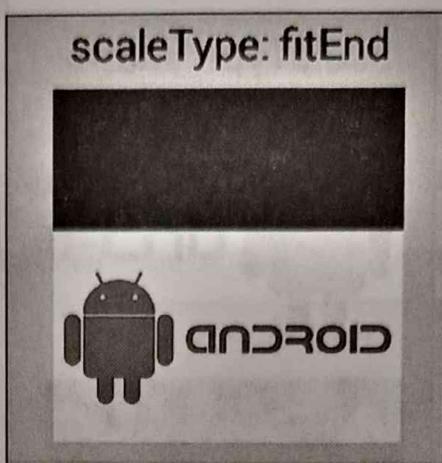
6. fitStart: масштабирование изображения с помощью функции START.



7. fitCenter: масштабирование изображения с помощью функции CENTER.



8. fitEnd: масштабирование изображения с помощью END.

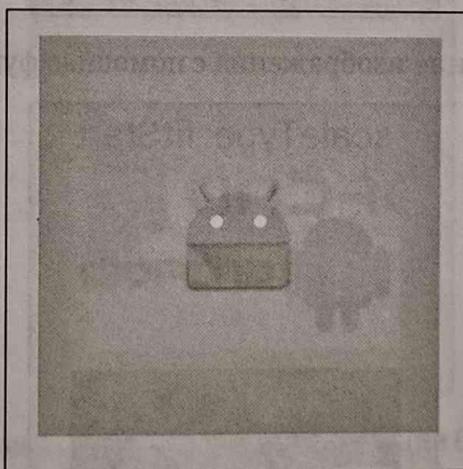


## 21.4. Тип масштабирования Center

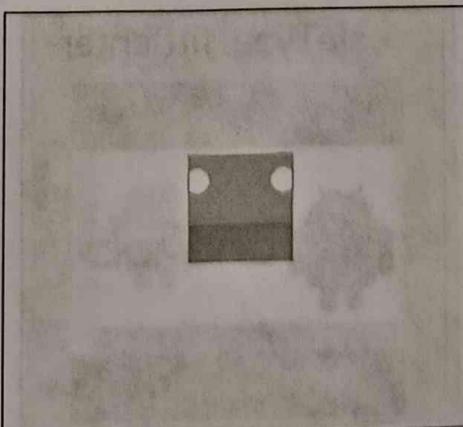
Изображение, содержащееся в ImageView, может не соответствовать точному размеру, заданному контейнеру. В этом случае фреймворк позволяет изменять размеры изображения различными способами.

```
<ImageView android:layout_width="20dp"
           android:layout_height="20dp"
           android:src="@mipmap/ic_launcher"
           android:id="@+id/imageView" android:scaleType="center"
           android:background="@android:color/holo_orange_light"/>
```

При этом размер изображения не изменяется, и оно центрируется внутри контейнера.



В случае если ImageView меньше изображения, размер изображения не будет изменен, и вы сможете увидеть только его часть:

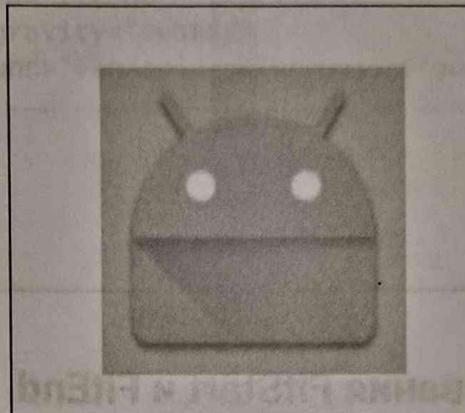


## 21.5. Тип масштабирования CenterCrop

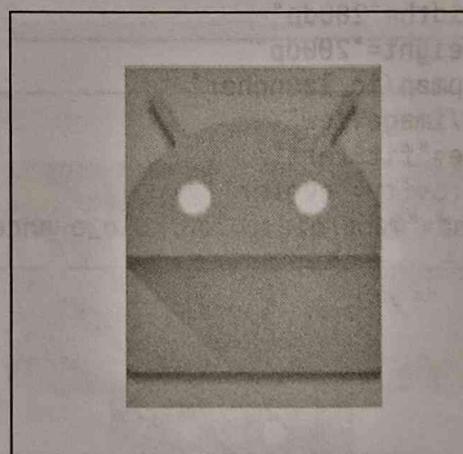
Равномерно масштабирует изображение (сохраняет соотношение сторон изображения) таким образом, чтобы оба размера (ширина и высота) изображения были равны или больше соответствующих размеров представления (за вычетом подгонки подкладками (padding)).

Официальная документация: <https://developer.android.com/reference/android/widget/ImageView.ScaleType.html>.

Когда изображение соответствует пропорциям контейнера:

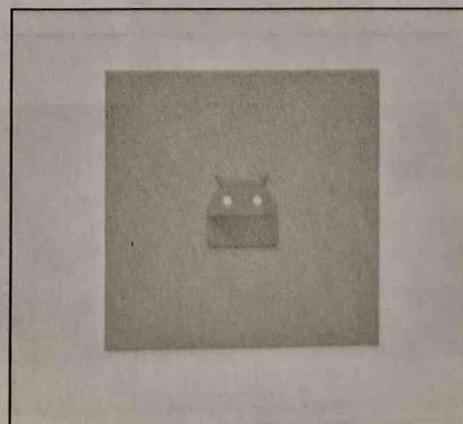


Если изображение шире контейнера, то он развернет его до большего размера (в данном случае высоты) и изменит ширину изображения без изменения его пропорций, что приведет к его обрезке.

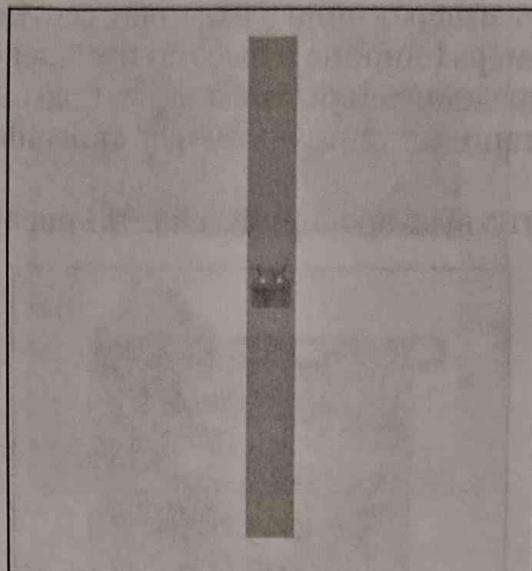


## 21.6. Тип масштабирования CenterInside

Равномерно масштабирует изображение (сохраняет соотношение сторон изображения) таким образом, чтобы оба размера (ширина и высота) изображения были равны или меньше соответствующих размеров представления (за вычетом подгонки подкладками). Центрирует изображение и изменяет его размер на меньший; если оба размера контейнера больше, то он будет действовать так же, как и центрирование типа Center.



Но если один из размеров мал, то будет соответствие этому размеру.

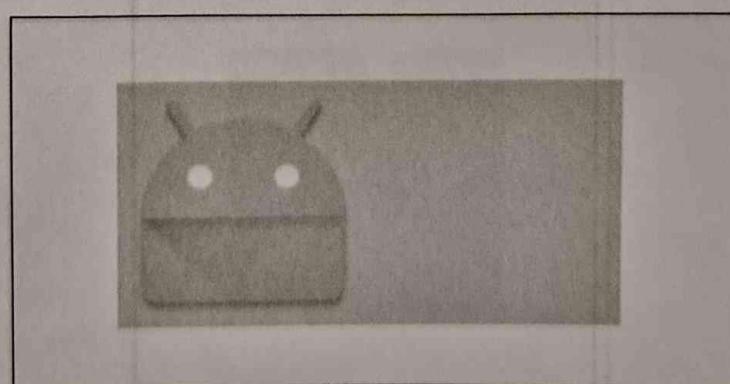
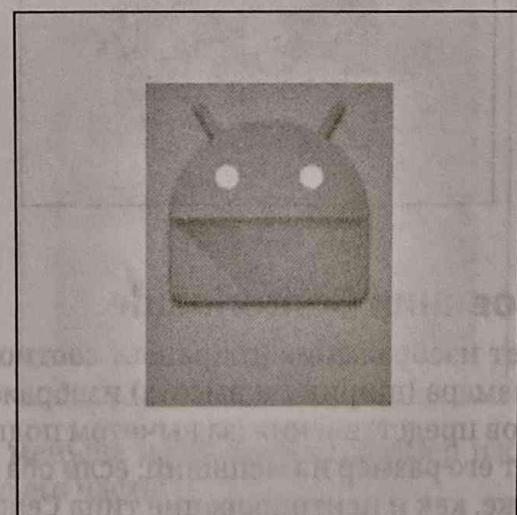


## 21.7. Тип масштабирования FitStart и FitEnd

### FitStart

Разрешает подгонку под меньший из размеров контейнера и выравнивает по началу.

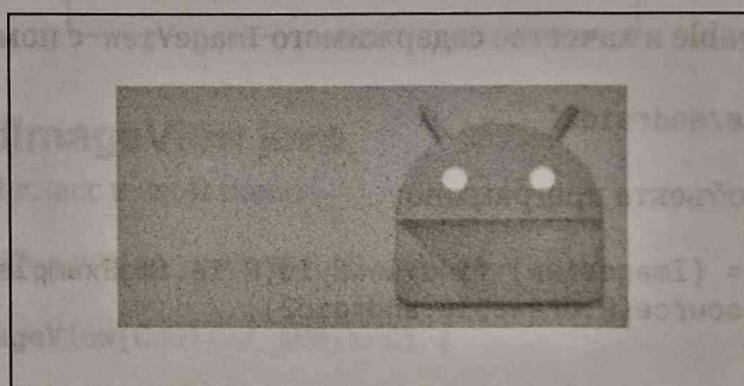
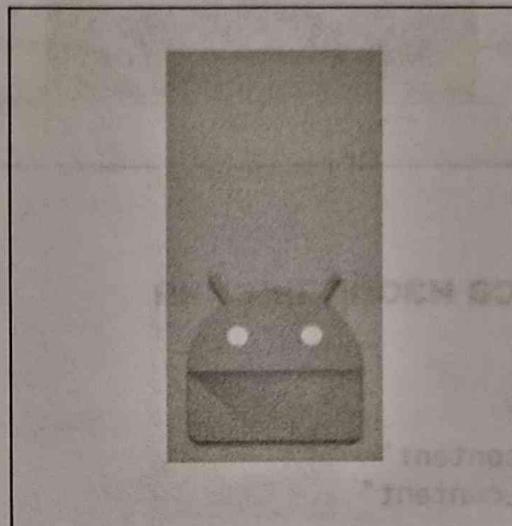
```
<ImageView android:layout_width="200dp"
           android:layout_height="200dp"
           android:src="@mipmap/ic_launcher"
           android:id="@+id/imageView"
           android:scaleType="fitStart"
           android:layout_gravity="center"
           android:background="@android:color/holo_orange_light"/>
```



### FitEnd

Этот тип масштабирования разрешает подгонку под наименьший размер контейнера и выравнивает по его концу.

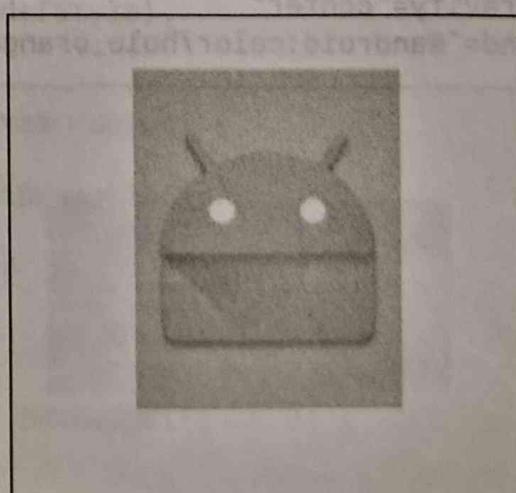
```
<ImageView android:layout_width="200dp"
    android:layout_height="100dp"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:scaleType="fitEnd"
    android:layout_gravity="center"
    android:background="@android:color/holo_orange_light"/>
```



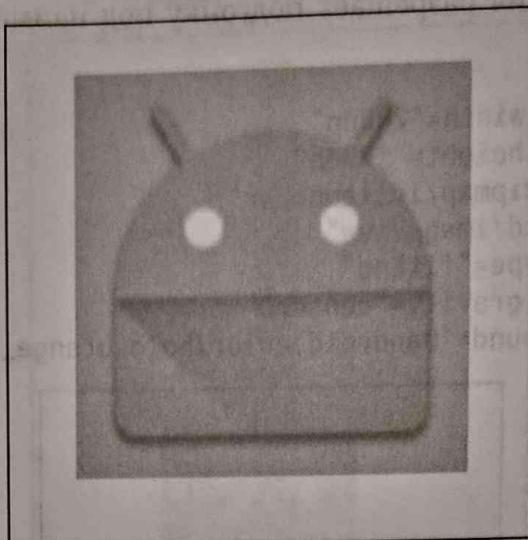
## 21.8. Тип масштабирования FitCenter

В этом случае изображение расширяется, пытаясь соответствовать контейнеру, и выравнивается по центру, подгоняясь под меньший размер.

Случай, если больше высота (подгонка к ширине):



Однаковые ширина и высота.



## 21.9. Установка ресурса изображения

```
<ImageView
    android:id="@+id/imgExample"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    ...
/>
```

Устанавливаем drawable в качестве содержимого ImageView с помощью атрибута XML:

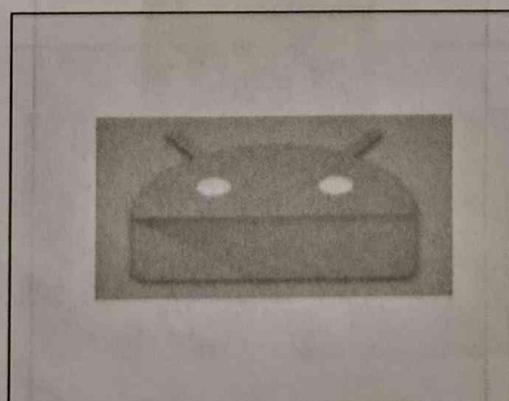
```
    android:src="@drawable/android2"
```

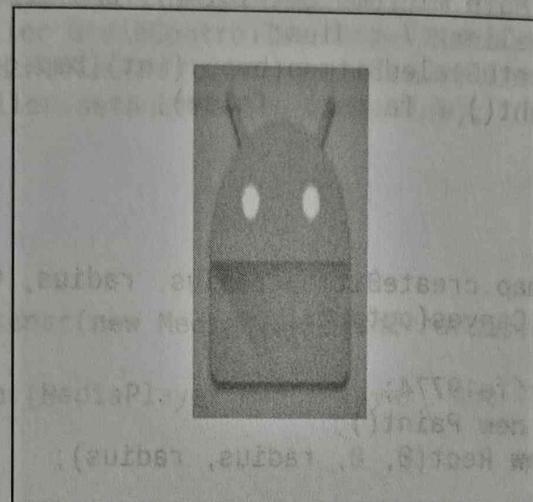
Установка drawable-объекта программно:

```
ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
imgExample.setImageResource(R.drawable.android2);
```

## 21.10. Тип масштабирования FitXY

```
<ImageView android:layout_width="100dp"
          android:layout_height="200dp"
          android:src="@mipmap/ic_launcher"
          android:id="@+id/imageView"
          android:scaleType="fitXY"
          android:layout_gravity="center"
          android:background="@android:color/holo_orange_light"/>
```





## 21.11. MLRoundedImageView.java

Добавьте следующий класс в свой пакет:

```
public class MLRoundedImageView extends ImageView {  
  
    public MLRoundedImageView(Context context) {  
        super(context);  
    }  
  
    public MLRoundedImageView(Context context, AttributeSet attrs) {  
        super(context, attrs);  
    }  
  
    public MLRoundedImageView(Context context, AttributeSet attrs, int defStyle) {  
        super(context, attrs, defStyle);  
    }  
  
    @Override  
    protected void onDraw(Canvas canvas) {  
  
        Drawable drawable = getDrawable();  
  
        if (drawable == null) {  
            return;  
        }  
  
        if (getWidth() == 0 || getHeight() == 0) {  
            return;  
        }  
    }  
}
```

```

Bitmap b = ((BitmapDrawable) drawable).getBitmap(); Bitmap bitmap =
b.copy(Bitmap.Config.ARGB_8888, true);

int w = getWidth(), h = getHeight();

Bitmap roundBitmap = getCroppedBitmap(bitmap, w);
canvas.drawBitmap(roundBitmap, 0, 0, null);

}

public static Bitmap getCroppedBitmap(Bitmap bmp, int radius) {
    Bitmap sbmp;

    if (bmp.getWidth() != radius || bmp.getHeight() != radius) {
        float smallest = Math.min(bmp.getWidth(), bmp.getHeight());
        float factor = smallest / radius;
        sbmp = Bitmap.createScaledBitmap(bmp, (int)(bmp.getWidth() / factor),
            (int)(bmp.getHeight() / factor), false);
    } else {
        sbmp = bmp;
    }

    Bitmap output = Bitmap.createBitmap(radius, radius, Config.ARGB_8888);
    Canvas canvas = new Canvas(output);

    final int color = 0xffa19774;
    final Paint paint = new Paint();
    final Rect rect = new Rect(0, 0, radius, radius);

    paint.setAntiAlias(true);
    paint.setFilterBitmap(true);
    paint.setDither(true);
    canvas.drawARGB(0, 0, 0, 0);
    paint.setColor(Color.parseColor("#BAB399"));
    canvas.drawCircle(radius / 2 + 0.7f,
        radius / 2 + 0.7f, radius / 2 + 0.1f, paint);
    paint.setXfermode(new PorterDuffXfermode_Mode.SRC_IN));
    canvas.drawBitmap(sbmp, rect, rect, paint);

    return output;
}
}

```

Используйте этот класс в XML с именем пакета вместо ImageView.

```

<com.androidbutts.example.MLRoundedImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/ic_launcher" />

```

## Глава 22. VideoView

### 22.1. Воспроизведение видео с URL-адреса с помощью VideoView

```

videoView.setVideoURI(Uri.parse("http://example.com/examplevideo.mp4"));
videoView.requestFocus();

```

```

videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
    }
});

videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mediaPlayer) {
        videoView.start();
        mediaPlayer.setOnVideoSizeChangedListener(new MediaPlayer.
            OnVideoSizeChangedListener() {
                @Override
                public void onVideoSizeChanged(MediaPlayer mp, int width, int height){
                    MediaController mediaController = new MediaController(ActivityName.this);
                    videoView.setMediaController(mediaController);
                    mediaController.setAnchorView(videoView);
                }
            });
    }
});

videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
    @Override
    public boolean onError(MediaPlayer mediaPlayer, int i, int i1) {
        return false;
    }
});

```

## 22.2. Создание VideoView

Найдите VideoView в активности и добавьте в него видео:

```

VideoView videoView = (VideoView).findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);

```

Начните воспроизведение видео:

```
videoView.start();
```

Определите VideoView в файле XML Layout:

```

<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />

```

# Глава 23. Оптимизация VideoView

Воспроизведение видео с помощью VideoView, который расширяет SurfaceView, внутри строки ListView, на первый взгляд, работает, пока пользователь не попытается прокрутить список. Как только список начинает прокручиваться, видео становится черным (ино-

гда отображается белым). Оно продолжает воспроизводиться в фоновом режиме, но его уже не видно, поскольку остальная часть видео отображается как черный прямоугольник. С помощью оптимизированного VideoView видеоролики будут воспроизводиться при прокрутке в ListView так же, как во всех распространенных социальных сетях.

## 23.1. Оптимизация VideoView в ListView

Это пользовательский VideoView, который необходимо иметь в своем пакете.

Пользовательский макет VideoView:

```
<your.packagename.VideoView
    android:id="@+id/video_view"
    android:layout_width="300dp"
    android:layout_height="300dp" />
```

Код для пользовательского оптимизированного VideoView:

```
package your.package.com.whateveritis;

import android.content.Context;
import android.content.Intent;
import android.graphics.SurfaceTexture;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnErrorListener;
import android.media.MediaPlayer.OnInfoListener;
import android.net.Uri;
import android.util.AttributeSet;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.widget.MediaController;
import android.widget.MediaController.MediaPlayerControl;

import java.io.IOException;

/**
 * VideoView используется для воспроизведения видео,
 * как и {@link android.widget.VideoView VideoView}. Мы определяем пользовательское
 * представление custom view, поскольку не смогли использовать
 * {@link android.widget.VideoView VideoView} в ListView.
 * VideoView внутри ScrollView не прокручиваются должным образом. Даже если
 * вы используете обходной путь для установки цвета фона, MediaController
 * не прокручивается вместе с VideoView. Кроме того, при прокрутке видео
 * выглядит ужасно при использовании обходного пути, много мерцаний.
 *
 * @author leo
 */
public class VideoView extends TextureView implements MediaPlayerControl {
    private static final String TAG = "tag";

    // все возможные внутренние состояния
    private static final int STATE_ERROR = -1;
```

```
private static final int STATE_IDLE = 0;
private static final int STATE_PREPARING = 1;
private static final int STATE_PREPARED = 2;
private static final int STATE_PLAYING = 3;
private static final int STATE_PAUSED = 4;
private static final int STATE_PLAYBACK_COMPLETED = 5;

// currentState - текущее состояние объекта VideoView.
// targetState - состояние, которого намерен достичь вызывающий метод.
// Например, независимо от текущего состояния объекта VideoView,
// вызов функции pause() предназначен для перевода объекта в целевое состояние
// STATE_PAUSED.
private int mCurrentState = STATE_IDLE;
private int mTargetState = STATE_IDLE;

// То, что нам нужно для воспроизведения и показа видео
private MediaPlayer mMediaPlayer;
private int mVideoWidth;
private int mVideoHeight;
private int mSurfaceWidth;
private int mSurfaceHeight;
private SurfaceTexture mSurfaceTexture;
private Surface mSurface;
private MediaController mMediaController;
private MediaPlayer.OnCompletionListener mOnCompletionListener;
private MediaPlayer.OnPreparedListener mOnPreparedListener;

private MediaPlayer.OnErrorListener mOnErrorListener;
private MediaPlayer.OnInfoListener mOnInfoListener;

private int mSeekWhenPrepared;// запись позиции поиска
// при подготовке
private int mCurrentBufferPercentage;
private int mAudioSession;
private Uri mUri;

private Context mContext;

public VideoView(final Context context) {
    super(context);
    mContext = context;
    initVideoView();
}

public VideoView(final Context context, final AttributeSet attrs) {
    super(context, attrs);
    mContext = context;
    initVideoView();
}

public VideoView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    mContext = context;
    initVideoView();
}

public void initVideoView() {
    mVideoHeight = 0;
    mVideoWidth = 0;
```

```

    setFocusable(false);
    setSurfaceTextureListener(mSurfaceTextureListener);
}

public int resolveAdjustedSize(int desiredSize, int measureSpec) {
    int result = desiredSize;
    int specMode = MeasureSpec.getMode(measureSpec);
    int specSize = MeasureSpec.getSize(measureSpec);

    switch (specMode) {
        case MeasureSpec.UNSPECIFIED:
            /*
             * Можно установить настолько большой размер, насколько мы захотим.
             * Но не превышать наложенный максимальный размер.
             */
            result = desiredSize;
            break;

        case MeasureSpec.AT_MOST:
            /*
             * Можно установить настолько большой размер, насколько мы захотим,
             * вплоть до specSize. Не следует превышать specSize и наложенный
             * максимальный размер.
             */
            result = Math.min(desiredSize, specSize);
            break;

        case MeasureSpec.EXACTLY:
            // В этом случае нет выбора, придется делать то, что нам указывают.
            result = specSize;
            break;
    }
    return result;
}

public void setVideoPath(String path) {
    Log.d(TAG, "Setting video path to: " + path);
    setVideoURI(Uri.parse(path));
}

public void setVideoURI(Uri _videoURI) {
    mUri = _videoURI;
    mSeekWhenPrepared = 0;
    requestLayout();
    invalidate();
    openVideo();
}

public Uri getUri() {
    return mUri;
}

public void setSurfaceTexture(SurfaceTexture _surfaceTexture) {
    mSurfaceTexture = _surfaceTexture;
}

public void openVideo() {
    if ((mUri == null) || (mSurfaceTexture == null)) {
}

```

```
Log.d(TAG, "Невозможно открыть видео, uri или текстура поверхности равны null.");
return;
}
// Нужно сообщить службе воспроизведения музыки о необходимости сделать паузу
// TODO: эти константы должны быть опубликованы где-нибудь
// во фреймворке.
Intent i = new Intent("com.android.music.musicservicecommand");
i.putExtra("command", "pause");
mContext.sendBroadcast(i);
release(false);
try {
    mSurface = new Surface(mSurfaceTexture);
    mMediaPlayer = new MediaPlayer();
    if (mAudioSession != 0) {
        mMediaPlayer.setAudioSessionId(mAudioSession);
    } else {
        mAudioSession = mMediaPlayer.getAudioSessionId();
    }

    mMediaPlayer.setOnBufferingUpdateListener(mBufferingUpdateListener);
    mMediaPlayer.setOnCompletionListener(mCompleteListener);
    mMediaPlayer.setOnPreparedListener(mPreparedListener);
    mMediaPlayer.setOnErrorListener(mErrorListener);
    mMediaPlayer.setOnInfoListener(mOnInfoListener);
    mMediaPlayer.setOnVideoSizeChangedListener(mVideoSizeChangedListener);

    mMediaPlayer.setSurface(mSurface);
    mCurrentBufferPercentage = 0;
    mMediaPlayer.setDataSource(mContext, mUri);

    mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
    mMediaPlayer.setScreenOnWhilePlaying(true);

    mMediaPlayer.prepareAsync();
    mcurrentState = STATE_PREPARING;
} catch (IllegalStateException e) {
    mcurrentState = STATE_ERROR;
    mTargetState = STATE_ERROR;
    String msg = (e.getMessage() == null) ? "" : e.getMessage();
    Log.i("", msg); // TODO автоматически сгенерированный блок catch
} catch (IOException e) {
    mcurrentState = STATE_ERROR;
    mTargetState = STATE_ERROR;
    String msg = (e.getMessage() == null) ? "" : e.getMessage();
    Log.i("", msg); // TODO автоматически сгенерированный блок catch
}
}

public void stopPlayback() {
    if (mMediaPlayer != null) {
        mMediaPlayer.stop();
        mMediaPlayer.release();
        mMediaPlayer = null;
        if (null != mMediaControllerListener) {
            mMediaControllerListener.onStop();
        }
    }
}

public void setMediaController(MediaController controller) {
    if (mMediaController != null) {
```

```

        mMediaController.hide();
    }
    mMediaController = controller;
    attachMediaController();
}

private void attachMediaController() {
    if (mMediaPlayer != null && mMediaController != null) {
        mMediaController.setMediaPlayer(this);
        View anchorView = this.getParent() instanceof View ? (View) this.
            getParent() : this;
        mMediaController.setAnchorView(anchorView);
        mMediaController.setEnabled(isInPlaybackState());
    }
}

private void release(boolean cleartargetstate) {
    Log.d(TAG, "Освобождение медиаплеера.");
    if (mMediaPlayer != null) {
        mMediaPlayer.reset();
        mMediaPlayer.release();
        mMediaPlayer = null;
        mCurrentState = STATE_IDLE;
        if (cleartargetstate) {
            mTargetState = STATE_IDLE;
        }
    } else {
        Log.d(TAG, "Значение null, медиаплеер не освобожден.");
    }
}

@Override
protected void onMeasure(final int widthMeasureSpec, final int
heightMeasureSpec) {
    // Будет изменен размер представления, если были найдены размеры видео.
    // Размеры видео находятся после вызова функции onPrepared MediaPlayer
    int width = getDefaultSize(mVideoWidth, widthMeasureSpec);
    int height = getDefaultSize(mVideoHeight, heightMeasureSpec);
    if ((mVideoWidth > 0) && (mVideoHeight > 0)) {
        if ((mVideoWidth * height) > (width * mVideoHeight)) {
            Log.d(TAG, "Видео слишком высокое, измените размер.");
            height = (width * mVideoHeight) / mVideoWidth;
        } else if ((mVideoWidth * height) < (width * mVideoHeight)) {
            Log.d(TAG, "Видео слишком широкое, измените размер.");
            width = (height * mVideoWidth) / mVideoHeight;
        } else {
            Log.d(TAG, "Верное соотношение размеров.");
        }
    }
    setMeasuredDimension(width, height);
}

@Override
public boolean onTouchEvent(MotionEvent ev) {
    if (isInPlaybackState() && mMediaController != null) {
        toggleMediaControlsVisibility();
    }
    return false;
}

```

```
@Override
public boolean onTrackballEvent(MotionEvent ev) {
    if (isInPlaybackState() && mMediaController != null) {
        toggleMediaControlsVisibility();
    }
    return false;
}
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    boolean isKeyCodeSupported = keyCode != KeyEvent.KEYCODE_BACK && keyCode != KeyEvent.KEYCODE_VOLUME_UP && keyCode != KeyEvent.KEYCODE_VOLUME_DOWN
        && keyCode != KeyEvent.KEYCODE_VOLUME_MUTE && keyCode != KeyEvent.KEYCODE_MENU && keyCode != KeyEvent.KEYCODE_CALL
        && keyCode != KeyEvent.KEYCODE_ENDCALL;
    if (isInPlaybackState() && isKeyCodeSupported && mMediaController != null) {
        if (keyCode == KeyEvent.KEYCODE_HEADSETHOOK || keyCode == KeyEvent.KEYCODE_MEDIA_PLAY_PAUSE) {
            if (mMediaPlayer.isPlaying()) {
                pause();
                mMediaController.show();
            } else {
                start();
                mMediaController.hide();
            }
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_MEDIA_PLAY) {
            if (!mMediaPlayer.isPlaying()) {
                start();
                mMediaController.hide();
            }
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_MEDIA_STOP || keyCode == KeyEvent.KEYCODE_MEDIA_PAUSE) {
            if (mMediaPlayer.isPlaying()) {
                pause();
                mMediaController.show();
            }
            return true;
        } else {
            toggleMediaControlsVisibility();
        }
    }
    return super.onKeyDown(keyCode, event);
}
private void toggleMediaControlsVisibility() {
    if (mMediaController.isShowing()) {
        mMediaController.hide();
    } else {
        mMediaController.show();
    }
}

public void start() {
    // Потенциально эта функция может быть вызвана в нескольких точках,
    // для срабатывания нужны все условия:
    // 1. Установленное URI видео
```

```

// 2. Поверхность (surface) становится доступной
// 3. Указание от активности
if (isInPlaybackState()) {
    mMediaPlayer.start();
    mCurrentState = STATE_PLAYING;
    if (null != mMediaControllListener) {
        mMediaControllListener.onStart();
    }
} else {
    Log.d(TAG, "Не удалось запустить. Текущее состояние " + mCurrentState);
}
mTargetState = STATE_PLAYING;
}

public void pause() {
    if (isInPlaybackState()) {
        if (mMediaPlayer.isPlaying()) {
            mMediaPlayer.pause();
            mCurrentState = STATE_PAUSED;
            if (null != mMediaControllListener) {
                mMediaControllListener.onPause();
            }
        }
    }
    mTargetState = STATE_PAUSED;
}

public void suspend() {
    release(false);
}

public void resume() {
    openVideo();
}

@Override
public int getDuration() {
    if (isInPlaybackState()) {
        return mMediaPlayer.getDuration();
    }
    return -1;
}

@Override
public int getCurrentPosition() {
    if (isInPlaybackState()) {
        return mMediaPlayer.getCurrentPosition();
    }
    return 0;
}

@Override
public void seekTo(int msec) {
    if (isInPlaybackState()) {
        mMediaPlayer.seekTo(msec);
        mSeekWhenPrepared = 0;
    } else {
        mSeekWhenPrepared = msec;
    }
}

```

```
    }

}

@Override
public boolean isPlaying() {
    return isInPlaybackState() && mMediaPlayer.isPlaying();
}

@Override
public int getBufferPercentage() {
    if (mMediaPlayer != null) {
        return mCurrentBufferPercentage;
    }
    return 0;
}

private boolean isInPlaybackState() {
    return ((mMediaPlayer != null) && (mCurrentState != STATE_ERROR) &&
        (mCurrentState != STATE_IDLE) && (mCurrentState != STATE_PREPARING));
}

@Override
public boolean canPause() {
    return false;
}

@Override
public boolean canSeekBackward() {
    return false;
}

@Override
public boolean canSeekForward() {
    return false;
}

@Override
public int getAudioSessionId() {
    if (mAudioSession == 0) {
        MediaPlayer foo = new MediaPlayer();
        mAudioSession = foo.getAudioSessionId();
        foo.release();
    }
    return mAudioSession;
}

// Слушатели
private MediaPlayer.OnBufferingUpdateListener mBufferingUpdateListener = new
MediaPlayer.OnBufferingUpdateListener() {
    @Override
    public void onBufferingUpdate(final MediaPlayer mp, final int percent) {
        mCurrentBufferPercentage = percent;
    }
};

private MediaPlayer.OnCompletionListener mCompleteListener = new MediaPlayer.
OnCompletionListener() {
    @Override
    public void onCompletion(final MediaPlayer mp) {
```

```

mCurrentState = STATE_PLAYBACK_COMPLETED;
mTargetState = STATE_PLAYBACK_COMPLETED;
mSurface.release();

if (mMediaController != null) {
    mMediaController.hide();
}

if (mOnCompletionListener != null) {
    mOnCompletionListener.onCompletion(mp);
}

if (mMediaControllListener != null) {
    mMediaControllListener.onComplete();
}
}

private MediaPlayer.OnPreparedListener mPreparedListener = new MediaPlayer.
OnPreparedListener() {
    @Override
    public void onPrepared(final MediaPlayer mp) {
        mCurrentState = STATE_PREPARED;

        mMediaController = new MediaController(getContext());
        if (mOnPreparedListener != null) {
            mOnPreparedListener.onPrepared(mMediaPlayer);
        }
        if (mMediaController != null) {
            mMediaController.setEnabled(true);
            //mMMediaController.setAnchorView(getRootView());
        }

        mVideoWidth = mp.getVideoWidth();
        mVideoHeight = mp.getVideoHeight();

        int seekToPosition = mSeekWhenPrepared; // mSeekWhenPrepared может быть
        // изменено после вызова seekTo()
        if (seekToPosition != 0) {
            seekTo(seekToPosition);
        }

        requestLayout();
        invalidate();
        if ((mVideoWidth != 0) && (mVideoHeight != 0)) {
            if (mTargetState == STATE_PLAYING) {
                mMediaPlayer.start();
                if (null != mMediaControllListener) {
                    mMediaControllListener.onStart();
                }
            }
        } else {
            if (mTargetState == STATE_PLAYING) {
                mMediaPlayer.start();
                if (null != mMediaControllListener) {
                    mMediaControllListener.onStart();
                }
            }
        }
    }
}

```



```
        }
    }
};

private MediaPlayer.OnVideoSizeChangedListener mVideoSizeChangedListener = new
MediaPlayer.OnVideoSizeChangedListener() {
    @Override
    public void onVideoSizeChanged(final MediaPlayer mp, final int width, final
int height) {
    mVideoWidth = mp.getVideoWidth();
    mVideoHeight = mp.getVideoHeight();
    if (mVideoWidth != 0 && mVideoHeight != 0) {
        requestLayout();
    }
}
};

private MediaPlayer.OnErrorListener mErrorListener = new MediaPlayer.
OnErrorListener() {
    @Override
    public boolean onError(final MediaPlayer mp, final int what, final int extra) {
        Log.d(TAG, "Error: " + what + "," + extra);
        mCurrentState = STATE_ERROR;
        mTargetState = STATE_ERROR;

        if (mMediaController != null) {
            mMediaController.hide();
        }

        /* Если был предоставлен обработчик ошибок, используйте его и завершите
работу. */
        if (mOnErrorListener != null) {
            if (mOnErrorListener.onError(mp, what, extra)) {
                return true;
            }
        }
    }
};

public void onStop() {
    /*
     * В противном случае следует вывести диалог ошибки, чтобы пользователь
     * знал - что-то прошло неправильно. Попытайтесь вызвать диалог только в том
     * случае, если мы прикреплены к окну. Если мы уходим из окна
     * больше нет, не стоит показывать пользователю ошибку.
     */
    if (getWindowToken() != null) {

//        new AlertDialog.Builder(mContext).setMessage("Error: " + what + "," + extra).
//setPositiveButton("OK", new DialogInterface.OnClickListener() {
//            @Override
//            public void onClick(DialogInterface dialog, int whichButton) {
//                /*
//                 * Если мы попали сюда, то слушателя onError нет, поэтому
//                 * нужно хотя бы сообщить им, что видео закончилось.
//                */
//                if (mOnCompletionListener != null) {
//                    mOnCompletionListener.onCompletion(mp);
//                }
//            }
//        }).setCancelable(false).show();
    }
}
```

```

        return true;
    }

};

SurfaceTextureListener mSurfaceTextureListener = new SurfaceTextureListener() {
    @Override
    public void onSurfaceTextureAvailable(final SurfaceTexture surface, final int width, final int height) {
        Log.d(TAG, "onSurfaceTextureAvailable.");
        mSurfaceTexture = surface;
        openVideo();
    }

    @Override
    public void onSurfaceTextureSizeChanged(final SurfaceTexture surface, final int width, final int height) {
        Log.d(TAG, "onSurfaceTextureSizeChanged: " + width + '/' + height);
        mSurfaceWidth = width;
        mSurfaceHeight = height;
        boolean isValidState = (mTargetState == STATE_PLAYING);
        boolean hasValidSize = (mVideoWidth == width & mVideoHeight == height);
        if (mMediaPlayer != null && isValidState && hasValidSize) {
            if (mSeekWhenPrepared != 0) {
                seekTo(mSeekWhenPrepared);
            }
            start();
        }
    }

    @Override
    public boolean onSurfaceTextureDestroyed(final SurfaceTexture surface) {
        mSurface = null;
        if (mMediaController != null)
            mMediaController.hide();
        release(true);
        return true;
    }

    @Override
    public void onSurfaceTextureUpdated(final SurfaceTexture surface) {
    }
};

/***
 * Зарегистрируйте обратный вызов, который будет вызываться при загрузке
 * и готовности медиафайла.
 */
public void setOnPreparedListener(MediaPlayer.OnPreparedListener l) {
    mOnPreparedListener = l;
}

/***
 * Зарегистрируйте обратный вызов, который будет вызываться при достижении
 */

```

```
* завершения медиафайла во время воспроизведения.  
*  
* @param l Обратный вызов, который будет запущен  
*/  
public void setOnCompletionListener(OnCompletionListener l) {  
    mOnCompletionListener = l;  
}  
/**  
 * Зарегистрируйте обратный вызов, который будет вызываться при возникновении  
 * ошибки во время воспроизведения или настройки. Если слушатель не указан  
 * или если слушатель вернул false, VideoView сообщит пользователю  
 * о возникших ошибках.  
*  
* @param l Обратный вызов, который будет запущен  
*/  
public void setOnErrorListener(OnErrorListener l) {  
    mOnErrorListener = l;  
}  
/**  
 * Зарегистрируйте обратный вызов, который будет вызываться при наступлении  
 * информационного события во время воспроизведения или настройки.  
*  
* @param l Обратный вызов, который будет запущен  
*/  
public void setOnInfoListener(OnInfoListener l) {  
    mOnInfoListener = l;  
}  
  
public static interface MediaControllerListener {  
    public void onStart();  
  
    public void onPause();  
  
    public void onStop();  
  
    public void onComplete();  
}  
  
MediaControllerListener mMediaControllerListener;  
  
public void setMediaControllerListener(MediaControllerListener mediaControllerListener) {  
    mMediaControllerListener = mediaControllerListener;  
}  
  
@Override  
public void setVisibility(int visibility) {  
    System.out.println("setVisibility: " + visibility);  
    super.setVisibility(visibility);  
}  
}
```

Основа для этого материала была взята с github-репозитория <https://github.com/aizhang/Android-VideoView/blob/master/src/com/ai/videoview/widget/VideoView.java>; в нем были некоторые недостатки, которые удалось исправить в вышеприведенном коде.

# Глава 24. WebView

WebView – это представление, отображающее веб-страницы внутри вашего приложения. С его помощью вы можете добавить свой собственный URL.

## 24.1. Поиск и устранение неисправностей WebView с помощью вывода консольных сообщений или удаленной отладки

### Вывод консольных сообщений webView в logcat

Для обработки консольных сообщений с веб-страницы можно переопределить `onConsoleMessage` в клиенте `WebChromeClient`:

```
final class ChromeClient extends WebChromeClient {
    @Override
    public boolean onConsoleMessage(ConsoleMessage msg) {
        Log.d(
            "WebView",
            String.format("%s %s:%d", msg.message(), msg.lineNumber(), msg.sourceId())
        );
        return true;
    }
}
```

...и установить его в своей активности или фрагменте:

```
webView.setWebChromeClient(new ChromeClient());
```

Итак, данный пример страницы:

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

будет записывать журнал (лог) ‘test message’ в logcat:

WebView: test message sample.html:4

Chrome-клиентом также поддерживаются `console.info()`, `console.warn()` и `console.error()`.

### Удаленная отладка Android-устройств с помощью Chrome

Вы можете удаленно выполнять отладку приложений, основанных на `WebView`, с рабочего стола Chrome.

### Включите отладку по USB на устройстве Android

На устройстве Android откройте “Настройки”, найдите раздел “Параметры разработчика” (Developer options) и включите отладку по USB (USB debugging).

### Подключение и обнаружение устройства Android

Откройте в Chrome следующую страницу: `chrome://inspect/#devices`.

В диалоговом окне Inspect Devices выберите устройство и нажмите кнопку **Inspect**. Откроется программа Chrome DevTools, подробное руководство и описание которой при не-

обходимости можно найти на сайте <https://developers.google.com/web/tools/chrome-devtools/remote-debugging/>.

## 24.2. Передача данных из JavaScript в Java (Android)

### Android-активность

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        WebView webView = new WebView(this);
        setContentView(webView);

        /*
         * Обратите внимание на метку Android, она используется в JavaScript-версии
         * Разумеется, это можно изменить.
         */
        webView.addJavascriptInterface(new JavascriptHandler(), "Android");

        webView.loadUrl("http://example.com");
    }
}
```

### Обработчик Java Javascript

```
import android.webkit.JavascriptInterface;

public class JavascriptHandler {

    /**
     * Ключевым моментом здесь является аннотация @JavascriptInterface
     */
    @JavascriptInterface
    public void jsCallback() {
        // Код, который делает что-нибудь
    }

    @JavascriptInterface
    public void jsCallbackTwo(String dummyData) {
        // Код, который делает что-нибудь
    }
}
```

### Веб-страница, вызов Javascript

```
<script>
...
Android.jsCallback();

```

```
...
Android.jsCallback('hello test');
...
</script>
```

### Дополнительный совет

При передаче сложной структуры данных возможным решением является использование JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

На стороне Android используйте ваш парсер JSON, например JSONObject.

## 24.3. Связь между Java и JavaScript

### Базовый пример:

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {

    private Webview webView;

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

        webView = new WebView(this);
        webView.getSettings().setJavaScriptEnabled(true);

        setContentView(webView);

        webView.loadUrl("http://example.com");

        /*
         * Вызов функции JavaScript
         */
        webView.loadUrl("javascript:testJsFunction('Hello World!')");
    }

    /**
     * Вызов функции JavaScript
     */
    public void DoSomething() {
        this.webView.loadUrl("javascript:testAnotherFunction('Hello World Again!')");
    }
}
```

## 24.4. Пример работы с набирателем номера

Если веб-страница содержит номер телефона, то можно позвонить с помощью телефонного набирателя номера. Данный код проверяет наличие url, который начинается с «tel:».

затем пытается открыть набиратель номера и предоставить возможность позвонить на нажатый номер телефона:

```
public boolean shouldOverrideUrlLoading(WebView view, String url) {
    if (url.startsWith("tel:")) {
        Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse(url));
        startActivity(intent);
    } else if(url.startsWith("http:") || url.startsWith("https:")) {
        view.loadUrl(url);
    }
    return true;
}
```

## 24.5. Открытие локального файла и создание динамического содержимого в WebView

Layout.xml:

```
<WebView
    android:id="@+id/WebViewToDisplay"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:fadeScrollbars="false" />
```

Загрузка данных в WebViewToDisplay:

```
WebView webViewDisplay;
StringBuffer LoadWEb1;

webViewDisplay = (WebView) findViewById(R.id.WebViewToDisplay);
LoadWEb1 = new StringBuffer();
LoadWEb1.append("<html><body><h1>Мой первый заголовок</h1><p>Мой первый абзац.</p>");
//Пример кода для чтения параметров во время выполнения
String strName = "Test Paragraph";
LoadWEb1.append("<br/><p>" + strName + "</p>");
String result = LoadWEb1.append("</body></html>").toString(); WebSettings
webSettings = webViewDisplay.getSettings();
webSettings.setJavaScriptEnabled(true);
webViewDisplay.getSettings().setBuiltInZoomControls(true);
if (android.os.Build.VERSION.SDK_INT >= 11){
    webViewDisplay.setLayerType(View.LAYER_TYPE_SOFTWARE, null);
    webViewDisplay.getSettings().setDisplayZoomControls(false);
}

webViewDisplay.loadDataWithBaseURL(null, result, "text/html", "utf-8", null);
//для загрузки локального файла непосредственно из папки assets используйте
//следующий код: webViewDisplay.loadUrl("file:///android_asset/aboutapp.html");
```

## 24.6. Диалоговые окна оповещения JavaScript в WebView

По умолчанию WebView не реализует JavaScript-диалоги оповещения, т.е. `alert()` ничего не сделает. Для того чтобы сделать это, необходимо сначала разрешить JavaScript (что очевидно), а затем установить `WebChromeClient` для обработки запросов на диалоги оповещения со страницы:

```
webView.setWebChromeClient(new WebChromeClient() {
```

```
//Другие методы для вашего WebChromeClient здесь, если необходимо...
@Override
public boolean onJsAlert(WebView view, String url, String message, JsResult
result) {
    return super.onJsAlert(view, url, message, result);
}
});
```

Здесь мы переопределяем `onJsAlert`, а затем обращаемся к суперреализации (`super implementation`), которая выдает нам стандартный диалог Android. Вы также можете использовать сообщение и URL самостоятельно, например, если хотите создать диалог в собственном стиле или вести журнал (лог) диалогов.

## Глава 25. SearchView

### 25.1. Установка темы для SearchView

Чтобы применить тему для `SearchView`, извлеченного как `app:actionViewClass` из `menu.xml`, нужно понимать, что она полностью зависит от стиля, примененного к нижележащей панели инструментов (`Toolbar`). Для создания темы панели инструментов выполните следующие действия.

Создайте стиль в файле `styles.xml`:

```
<style name="ActionBarThemeOverlay">
    <item name="android:textColorPrimary">@color/prim_color</item>
    <item name="colorControlNormal">@color/normal_color</item>
    <item name="colorControlHighlight">@color/high_color</item>
    <item name="android:textColorHint">@color/hint_color</item>
</style>
```

Примените стиль к панели инструментов:

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    app:theme="@style/ActionBarThemeOverlay"
    app:popupTheme="@style/ActionBarThemeOverlay"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="@color/colorPrimary"
    android:title="@string/title"
    tools:targetApi="m" />
```

Это позволяет придать нужный цвет всем видам, соответствующим панели инструментов (кнопке “Назад”, значкам меню и `SearchView`).

### 25.2. SearchView в панели инструментов с использованием Fragment

`menu.xml (res -> menu)`

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```

    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".HomeActivity">

```

```

<item
    android:id="@+id/action_search"
    android:icon="@android:drawable/ic_menu_search"
    android:title="Search"
    app:actionViewClass="android.support.v7.widget.SearchView"
    app:showAsAction="always" />

```

```

</menu>

```

### MainFragment.java

```

public class MainFragment extends Fragment {

```

```

    private SearchView searchView = null;
    private SearchView.OnQueryTextListener queryTextListener;
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
        savedInstanceState) {
        return inflater.inflate(R.layout.fragment_main, container, false);
    }

```

```

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
    }

```

```

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu, menu);
        MenuItem searchItem = menu.findItem(R.id.action_search);
        SearchManager searchManager = (SearchManager)
            getActivity().getSystemService(Context.SEARCH_SERVICE);
        if (searchItem != null) {
            searchView = (SearchView) searchItem.getActionView();
        }
        if (searchView != null) {

```

```

            searchView.setSearchableInfo(searchManager.getSearchableInfo(getActivity().
                getComponentName()));

```

```

            queryTextListener = new SearchView.OnQueryTextListener() {
                @Override
                public boolean onQueryTextChange(String newText) {
                    Log.i("onQueryTextChange", newText);
                    return true;
                }
                @Override
                public boolean onQueryTextSubmit(String query) {
                    Log.i("onQueryTextSubmit", query);
                    return true;
                }
            };

```

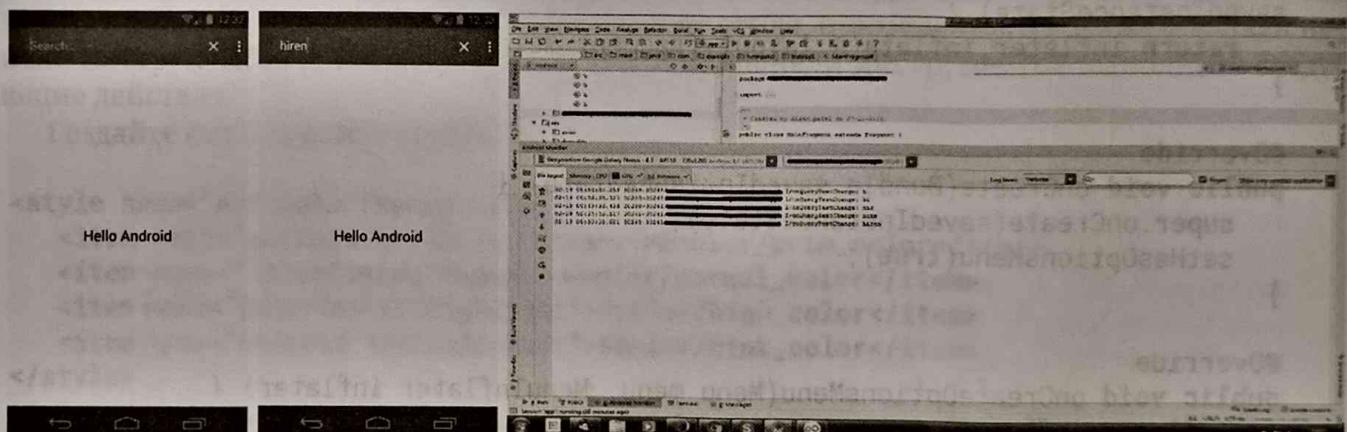
```

        }
    }
    searchView.setOnQueryTextListener(queryTextListener);
}
super.onCreateOptionsMenu(menu, inflater);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_search:
            // Здесь не реализовано
            return false;
        default:
            break;
    }
    searchView.setOnQueryTextListener(queryTextListener);
    return super.onOptionsItemSelected(item);
}
}

```

### Ссылочный скриншот:



## 25.3. Appcompat SearchView с наблюдателем RxBindings

### build.gradle:

```

dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}

```

### menu/menu.xml:

```

<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="Search"
          android:icon="@android:drawable/ic_menu_search"
          app:actionViewClass="android.support.v7.widget.SearchView"
          app:showAsAction="always"/>
</menu>

```

**MainActivity.java:**

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem);

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // здесь ваша подсказка

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // дополнительно: установить количество букв, после которого начнется поиск,
    // равным 1 (по умолчанию 2)
    try {
        int autoCompleteTextViewID = getResources().getIdentifier("android:id/search_src_text", null, null);
        AutoCompleteTextView searchAutoCompleteTextView = (AutoCompleteTextView)
            searchView.findViewById(autoCompleteTextViewID);
        searchAutoCompleteTextView.setThreshold(1);
    } catch (Exception e) {
        Log.e(TAG, "не удалось установить порог букв поискового представления");
    }

    searchView.setOnSearchClickListener(v -> {
        // необязательные действия для расширения вида поиска
    });
    searchView.setOnCloseListener(() -> {
        // необязательные действия для закрытия вида поиска
        return false;
    });

    RxSearchView.queryTextChanges(searchView)
        .doOnEach(notification -> {
            CharSequence query = (CharSequence) notification.getValue();
            searchAdapter.filter(query);
        })
        .debounce(300, TimeUnit.MILLISECONDS) // для пропуска промежуточных букв
        .flatMap(query -> MyWebService.search(query)) // делаем запрос на поиск
        .retry(3)
        .subscribe(results -> {
            searchAdapter.populateAdapter(results);
        });

    // дополнительно: сворачивание SearchView при закрытии
    searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
        if (!queryTextFocused) {
            collapseSearchView();
        }
    });
}
```

**SearchAdapter.java**

```

public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
                c.addRow(new Object[]{i});
            }
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ViewHolder holder = (ViewHolder) view.getTag();
        int position = cursor.getPosition();
        if (position < items.size()) {
            SearchResult result = items.get(position);
            // привяжите здесь ваше представление
        }
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        LayoutInflator inflater = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View v = inflater.inflate(R.layout.search_list_item, parent, false);
        ViewHolder holder = new ViewHolder(v);

        v.setTag(holder);
        return v;
    }

    private static class ViewHolder {
        public final TextView text;
        public ViewHolder(View v) {
            this.text = (TextView) v.findViewById(R.id.text);
        }
    }
}

```

# Глава 26. BottomNavigationView

BottomNavigationView уже давно присутствует в рекомендациях по материальному дизайну (материальному оформлению) (см. <https://material.io/guidelines/components/bottom-navigation.html>), но его реализацию в приложениях не назовешь простой задачей. В некоторых приложениях есть собственные решения, в то время как другие полагаются на сторонние библиотеки с открытым исходным кодом. Теперь в библиотеке поддержки дизайна появилась эта нижняя панель навигации, давайте рассмотрим, как ее использовать.

## 26.1. Базовая реализация

Для добавления BottomNavigationView выполните следующие действия.

1. Добавьте в build.gradle зависимость:

```
compile 'com.android.support:design:25.1.0'
```

2. Добавьте BottomNavigationView в свой макет:

```
<android.support.design.widget.BottomNavigationView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:menu="@menu/bottom_navigation_menu" />
```

3. Создайте меню для заполнения представления:

```
<!-- res/menu/bottom_navigation_menu.xml -->

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/my_action1"
        android:enabled="true"
        android:icon="@drawable/my_drawable"
        android:title="@string/text"
        app:showAsAction="ifRoom" />
    ...
</menu>
```

4. Прикрепите слушателя для событий щелчка (клика):

```
//Получить представление
BottomNavigationView bottomNavigationView = (BottomNavigationView) findViewById(R.
    id.bottom_navigation);
//Присоединение слушателя
bottomNavigationView.setOnNavigationItemSelectedListener(
    new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            switch (item.getItemId()) {
                case R.id.my_action1:
                    //Выполнить что-либо...
                    break;
            }
        }
    }
);
```

```

    }
    return true; //возвращение false отключает анимацию панели навигации
}
);

```

Демо-код можно найти на сайте <https://github.com/1priyank1/BottomNavigation-Demo>.

## 26.2. Настройка BottomNavigationView

В этом примере рассказывается, как добавить селектор для BottomNavigationView, чтобы вы могли указать пользовательский интерфейс для иконок и текстов.

Создайте отрисовываемый файл bottom\_navigation\_view\_selector.xml в виде

```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/bottom_nv_menu_selected" android:state_checked="true" />
    <item android:color="@color/bottom_nv_menu_default" />
</selector>

```

...и используйте следующие атрибуты в BottomNavigationView в файле макета:

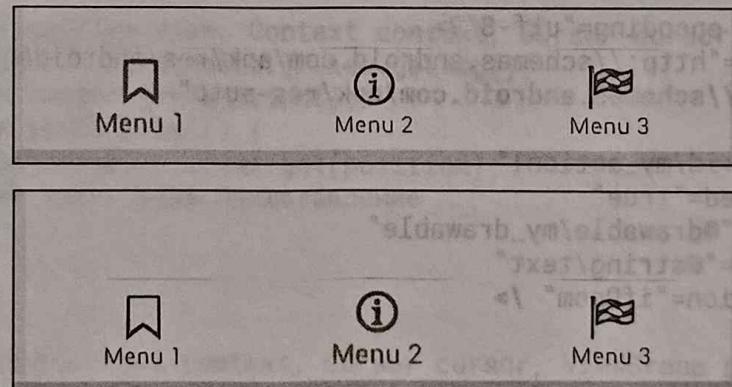
```

app:itemIconTint="@drawable/bottom_navigation_view_selector"
app:itemTextColor="@drawable/bottom_navigation_view_selector"

```

В приведенном выше примере использован один и тот же селектор bottom\_navigation\_view\_selector для app:itemIconTint и app:itemTextColor, чтобы сохранить цвета текста и иконок одинаковыми. Но если в вашем дизайне цвет текста и иконки отличается, вы можете определить два разных селектора и использовать их.

На выходе получаем:



## 26.3. Обработка состояний Enabled / Disabled

Создадим селектор для включения/выключения пункта меню:

### selector.xml

```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/white" android:state_enabled="true" />
    <item android:color="@color/colorPrimaryDark" android:state_enabled="false" />
</selector>

```

### design.xml

```

<android.support.design.widget.BottomNavigationView
    android:id="@+id/bottom_navigation"

```

```

    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:itemBackground="@color/colorPrimary"
    app:itemIconTint="@drawable/nav_item_color_state"
    app:itemTextColor="@drawable/nav_item_color_state"
    app:menu="@menu/bottom_navigation_main" />

```

## 26.4. Разрешение более чем трех меню

Данный пример является лишь обходным решением, поскольку в настоящее время не существует способа отключить поведение, известное как ShiftMode. Создайте функцию следующего вида:

```

public static void disableMenuShiftMode(BottomNavigationView view) {
    BottomNavigationMenuView menuView = (BottomNavigationMenuView) view.
        getChildAt(0);
    try {
        Field shiftingMode = menuView.getClass().getDeclaredField("mShiftingMode");
        shiftingMode.setAccessible(true);
        shiftingMode.setBoolean(menuView, false);
        shiftingMode.setAccessible(false);
        for (int i = 0; i < menuView.getChildCount(); i++) {
            BottomNavigationItemView item = (BottomNavigationItemView) menuView.
                getChildAt(i);
            // без проверки RestrictedApi
            item.setShiftingMode(false);
            // еще раз устанавливаем проверяемое значение, поэтому представление
            // будет обновлено
            // без проверки RestrictedApi
            item.setChecked(item.getItemData().isChecked());
        }
    } catch (NoSuchFieldException e) {
        Log.e("BNVHelper", "Unable to get shift mode field", e);
    } catch (IllegalAccessException e) {
        Log.e("BNVHelper", "Unable to change value of shift mode", e);
    }
}

```

Это отключает функцию сдвига меню (Shifting behaviour), если в нем более трех пунктов.

### Использование:

```
BottomNavigationView navView = (BottomNavigationView) findViewById(R.id.bottom_
navigation_bar); disableMenuShiftMode(navView);
```

**Проблема с Proguard:** Добавьте следующую строку в конфигурационный файл proguard, иначе это не будет работать.

```
-keepclassmembers class android.support.design.internal.BottomNavigationMenuView {
    boolean mShiftingMode;
}
```

В качестве альтернативы можно создать класс и обращаться к этому методу из класса. Об этом см. здесь: <http://stackoverflow.com/a/40189977/1149398>.

# Глава 27. Отрисовка на объекте холста Canvas с использованием SurfaceView

## 27.1. SurfaceView с потоком отрисовки

Этот пример описывает создание SurfaceView с выделенным потоком отрисовки. Эта реализация также обрабатывает такие нестандартные ситуации, как проблемы, связанные с производительностью, а также запуск/остановку потока для экономии процессорного времени:

```

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
/**
 * Определяет пользовательский класс SurfaceView, который обрабатывает поток отрисовки.
 */
public class BaseSurface extends SurfaceView implements SurfaceHolder.Callback,
View.OnTouchListener, Runnable
{
    /**
     * Рамка поверхности
     */
    private SurfaceHolder holder;

    /**
     * Поток отрисовки
     */
    private Thread drawThread;

    /**
     * True, когда поверхность готова к отрисовке
     */
    private boolean surfaceReady = false;

    /**
     * Флаг потока отрисовки
     */

    private boolean drawingActive = false;

    /**
     * Paint отрисовывает прямоугольник образца
     */
    private Paint samplePaint = new Paint();

    /**
     * Время на кадр для 60 FPS
     */
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);
}

```

```
private static final String LOGTAG = "surface";
```

```
public BaseSurface(Context context, AttributeSet attrs)
```

```
{
```

```
    super(context, attrs);
```

```
    SurfaceHolder holder = getHolder();
```

```
    holder.addCallback(this);
```

```
    setOnTouchListener(this);
```

```
    // красный
```

```
    samplePaint.setColor(0xffff0000);
```

```
    // сглаживание краев
```

```
    samplePaint.setAntiAlias(true);
```

```
}
```

```
@Override
```

```
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
```

```
{
```

```
    if (width == 0 || height == 0)
```

```
    {
```

```
        return;
```

```
    }
```

```
    // изменить размер пользовательского интерфейса
```

```
}
```

```
@Override
```

```
public void surfaceCreated(SurfaceHolder holder)
```

```
{
```

```
    this.holder = holder;
```

```
    if (drawThread != null)
```

```
    {
```

```
        Log.d(LOGTAG, "draw thread still active...");
```

```
        drawingActive = false;
```

```
        try
```

```
        {
```

```
            drawThread.join();
```

```
        } catch (InterruptedException e)
```

```
        { // ничего не делать
```

```
        }
```

```
    }
```

```
    surfaceReady = true;
```

```
    startDrawThread();
```

```
    Log.d(LOGTAG, "Created");
```

```
}
```

```
@Override
```

```
public void surfaceDestroyed(SurfaceHolder holder)
```

```
{
```

```
    // Поверхность больше не используется - остановите поток отрисовки
```

```
    stopDrawThread();
```

```
    // и освободите поверхность
```

```
    holder.getSurface().release();
```

```
    this.holder = null;
```

```
    surfaceReady = false;
```

```

        Log.d(LOGTAG, "Destroyed");
    }

@Override
public boolean onTouch(View v, MotionEvent event)
{
    // Обработка событий касания
    return true;
}

/**
 * Остановка потока отрисовки
 */
public void stopDrawThread()
{
    if (drawThread == null)
    {
        Log.d(LOGTAG, "DrawThread имеет значение null");
        return;
    }
    drawingActive = false;
    while (true)
    {
        try
        {
            Log.d(LOGTAG, "Запрос последнего кадра");
            drawThread.join(5000);
            break;
        } catch (Exception e)
        {
            Log.e(LOGTAG, "Не удалось соединиться с потоком отрисовки");
        }
    }
    drawThread = null;
}

/**
 * Создает новый поток отрисовки и запускает его
 */
public void startDrawThread()
{
    if (surfaceReady && drawThread == null)
    {
        drawThread = new Thread(this, "Поток отрисовки");
        drawingActive = true;
        drawThread.start();
    }
}

@Override
public void run()
{
    Log.d(LOGTAG, "Поток отрисовки запущен");
    long frameStartTime;
    long frameTime;

    /*
     * Для надежной работы на Nexus 7 ставим задержку ~500 мс на начало потока
     * отрисовки
    */
}

```

```
/*
 * if (android.os.Build.BRAND.equalsIgnoreCase("google") && android.
 * os.Build.MANUFACTURER.equalsIgnoreCase("asus") && android.os.Build.MODEL.
 * equalsIgnoreCase("Nexus 7"))
 {
    Log.w(LOGTAG, "Sleep 500ms (Device: Asus Nexus 7)");
    try
    {
        Thread.sleep(500);
    } catch (InterruptedException ignored)
    {}
}
try
{
    while (drawingActive)
    {
        if (holder == null)
        {
            return;
        }
        frameStartTime = System.nanoTime();
        Canvas canvas = holder.lockCanvas();
        if (canvas != null)
        {
            // очистить экран с помощью черного цвета
            canvas.drawRGB(255, 0, 0, 0, 0);
            try
            {
                // Ваша отрисовка здесь
                canvas.drawRect(0, 0, getWidth() / 2, getHeight() / 2,
samplePaint);
            } finally
            {
                holder.unlockCanvasAndPost(canvas);
            }
        }
        // вычислить время, необходимое для отрисовки кадра, в мс
        frameTime = (System.nanoTime() - frameStartTime) / 1000000;

        if (frameTime < MAX_FRAME_TIME) // быстрее максимального fps - ограничить
        {
            try
            {
                Thread.sleep(MAX_FRAME_TIME - frameTime);
            } catch (InterruptedException e)
            {
                // игнорировать
            }
        }
    } catch (Exception e)
    {
        Log.w(LOGTAG, "Иключение при блокировке/разблокировке");
    }
    Log.d(LOGTAG, "Draw thread finished");
}
```

```

    }
}
```

Этот макет содержит только пользовательское представление SurfaceView и максимизирует его под размер экрана.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sample.devcore.org.surfaceviewsample.MainActivity">
    <sample.devcore.org.surfaceviewsample.BaseSurface
        android:id="@+id/baseSurface"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

Активность, использующая SurfaceView, отвечает за запуск и остановку потока отрисовки. Такой подход позволяет экономить заряд батареи, поскольку отрисовка прекращается, как только активность переходит в фоновый режим.

```

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{
    /**
     * Объект поверхности
     */
    private BaseSurface surface;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        surface = (BaseSurface) findViewById(R.id.baseSurface);
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        // запуск отрисовки
        surface.startDrawThread();
    }

    @Override
    protected void onPause()
    {
        // остановить отрисовку для экономии процессорного времени
        surface.stopDrawThread();
        super.onPause();
    }
}
```

# Глава 28. Создание пользовательских представлений

## 28.1. Основы создания пользовательских представлений

Если вам необходимо полностью настроить представление, то вам придется подклассифицировать View (суперкласс всех представлений Android) и предоставить свои собственные методы определения размеров (onMeasure(...)) и отрисовки (onDraw(...)):

1. **Создайте скелет пользовательского представления:** этот процесс практически одинаков для каждого пользовательского представления. Здесь мы создадим скелет для пользовательского представления, которое может отрисовывать смайлик, под названием SmileyView:

```
public class SmileyView extends View {
    private Paint mCirclePaint;
    private Paint mEyeAndMouthPaint;

    private float mCenterX;
    private float mCenterY;
    private float mRadius;
    private RectF mArcBounds = new RectF();

    public SmileyView(Context context) {
        this(context, null, 0);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        initPaints();
    }

    private void initPaints() {/* ... */}

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {/* ... */}

    @Override
    protected void onDraw(Canvas canvas) {/* ... */}
}
```

2. **Инициализация объектов Paint:** эти объекты – кисти вашего виртуального холста, определяющие способ визуализации геометрических объектов (например, цвет, стиль заливки и обводки и т. д.). Здесь мы создаем два таких объекта, один желтый с заливкой для круга и один черный обводочный для глаз и рта:

```
private void initPaints() {
    mCirclePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mCirclePaint.setStyle(Paint.Style.FILL);
    mCirclePaint.setColor(Color.YELLOW);
    mEyeAndMouthPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mEyeAndMouthPaint.setStyle(Paint.Style.STROKE);
    mEyeAndMouthPaint.setStrokeWidth(16 * getResources().getDisplayMetrics().density);
    mEyeAndMouthPaint.setStrokeCap(Paint.Cap.ROUND);
```

```
mEyeAndMouthPaint.setColor(Color.BLACK);
}
```

**3. Реализуйте собственный метод `onMeasure(...)`:** он необходим для того, чтобы родительские макеты (например `FrameLayout`) могли правильно выровнять ваше пользовательское представление. Он предоставляет набор `measureSpecs`, который можно использовать для определения высоты и ширины представления. Здесь мы создаем квадрат, убедившись, что высота и ширина одинаковы:

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int w = MeasureSpec.getSize(widthMeasureSpec);
    int h = MeasureSpec.getSize(heightMeasureSpec);

    int size = Math.min(w, h);
    setMeasuredDimension(size, size);
}
```

Обратите внимание, что `onMeasure(...)` должен содержать хотя бы один вызов `setMeasuredDimension(...)`, иначе ваше пользовательское представление завершится с ошибкой `IllegalStateException`.

**4. Реализуйте собственный метод `onSizeChanged(...)`:** он позволяет перехватить текущие высоту и ширину пользовательского представления, чтобы правильно настроить код отображения. Здесь мы просто вычисляем центр и радиус:

```
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    mCenterX = w / 2f;
    mCenterY = h / 2f;
    mRadius = Math.min(w, h) / 2f;
}
```

**5. Реализуйте собственный метод `onDraw(...)`:** в этом методе осуществляется собственно отрисовка представления. Он предоставляет объект «холста» (`Canvas`), на котором можно отрисовывать (все доступные для этого методы см. в официальной документации: <https://developer.android.com/reference/android/graphics/Canvas.html>).

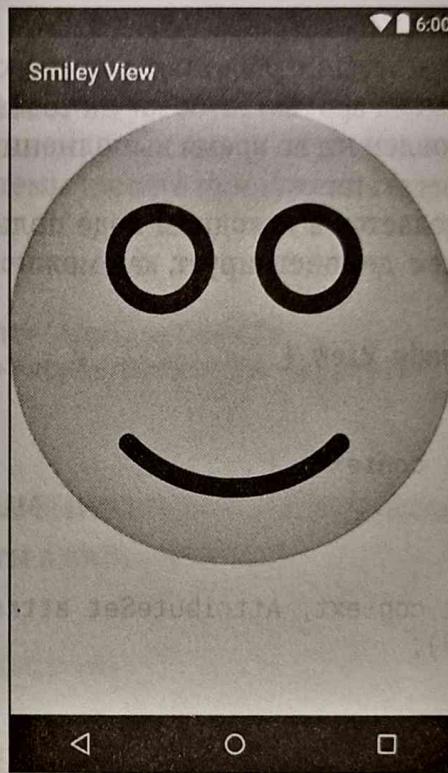
```
@Override
protected void onDraw(Canvas canvas) {
    // нарисовать лицо
    canvas.drawCircle(mCenterX, mCenterY, mRadius, mCirclePaint);
    // нарисовать глаза
    float eyeRadius = mRadius / 5f;
    float eyeOffsetX = mRadius / 3f;
    float eyeOffsetY = mRadius / 3f;
    canvas.drawCircle(mCenterX - eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
        mEyeAndMouthPaint);
    canvas.drawCircle(mCenterX + eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
        mEyeAndMouthPaint);
    // нарисовать рот
    float mouthInset = mRadius / 3f;
    mArcBounds.set(mouthInset, mouthInset, mRadius * 2 - mouthInset, mRadius * 2 -
        mouthInset);
    canvas.drawArc(mArcBounds, 45f, 90f, false, mEyeAndMouthPaint);
}
```

**5. Добавьте пользовательское представление в макет:** теперь пользовательское представление может быть включено в любые файлы макетов, которые у вас есть. Здесь мы просто обернули его внутри `FrameLayout`:

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <com.example.app.SmileyView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

Обратите внимание, что после завершения работы над кодом представления рекомендуется собрать проект. Без сборки вы не сможете увидеть вид на экране предварительного просмотра в Android Studio.

После того как все будет готово, при запуске активности, содержащей указанный выше макет, вы должны увидеть следующее окно:



## 28.2. Добавление атрибутов к представлениям

Пользовательские представления также могут принимать пользовательские атрибуты, которые могут быть использованы в файлах ресурсов компоновки Android. Чтобы добавить атрибуты к пользовательскому представлению, необходимо выполнить следующие действия.

1. Определите имя и тип атрибутов: это делается в файле `res/values/attrs.xml` (при необходимости создайте его). В следующем файле определены атрибут `color` для цвета лица нашего смайлика и атрибут `enum` для выражения смайлика:

```
<resources>
    <declare-styleable name="SmileyView">
        <attr name="smileyColor" format="color" />
        <attr name="smileyExpression" format="enum">
            <enum name="happy" value="0"/>
            <enum name="sad" value="1"/>
        </attr>
    </declare-styleable>
    <!-- Атрибуты для других представлений -->
</resources>
```

**2. Использование атрибутов внутри макета:** это можно сделать внутри всех файлов макета, использующих ваше пользовательское представление. Следующий файл макета создает экран с желтым счастливым смайлом:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <com.example.app.SmileyView
        android:layout_height="56dp"
        android:layout_width="56dp"
        app:smileyColor="#ffff00"
        app:smileyExpression="happy" />
</FrameLayout>
```

**Совет:** пользовательские атрибуты не работают с префиксом tools: в Android Studio 2.1 и старше. В данном примере замена app:smileyColor на tools:smileyColor приведет к тому, что smileyColor не будет установлен ни во время выполнения программы, ни во время проектирования.

**3. Чтение атрибутов:** это делается в исходном коде пользовательского представления. Следующий фрагмент SmileyView демонстрирует, как можно извлечь атрибуты:

```
public class SmileyView extends View {
    // ...

    public SmileyView(Context context) {
        this(context, null);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
            defStyleAttr, 0);
        mFaceColor = a.getColor(R.styleable.SmileyView_smileyColor, Color.TRANSPARENT);
        mFaceExpression = a.getInteger(R.styleable.SmileyView_smileyExpression,
            Expression.HAPPY);
        // Важно: всегда перерабатывать TypedArray
        a.recycle();

        // initPaints(); ...
    }
}
```

**4. (Необязательно) Добавление стиля по умолчанию:** для этого нужно добавить стиль со значениями по умолчанию и загрузить его в пользовательское представление. Следующий стиль смайлика по умолчанию представляет собой счастливый желтый смайлик:

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
```