

112.3. Настраиваемый ProgressBar

```

        android:layout_centerInParent="true"
        android:indeterminate="false"
        android:max="100"
        android:progress="0"
        android:progressDrawable="@drawable/custom_progressbar_drawable"
        android:secondaryProgress="0" />

    <TextView
        android:id="@+id/txtProgress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/progressBar"
        android:layout_centerInParent="true"
        android:textAppearance="?android:attr/textAppearanceSmall" />
</RelativeLayout>
```

custom_progressbar_drawable.xml:

```
<?xml version="1.0" encoding="utf-8"?>
```

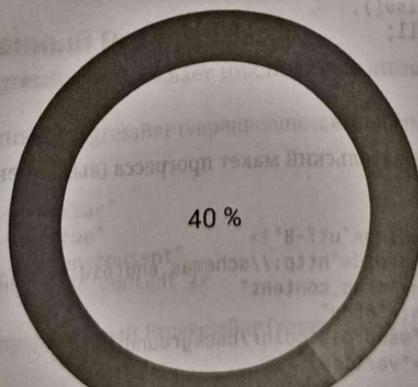
```
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="-90"
    android:pivotX="50%"
    android:pivotY="50%"
    android:toDegrees="270" >
```

CustomProgress

```

    <shape
        android:shape="ring"
        android:useLevel="false" >
        <gradient
            android:centerY="0.5"
            android:endColor="#FA5858"
            android:startColor="#0099CC"
            android:type="sweep"
            android:useLevel="false" />
    </shape>
</rotate>
```

Скриншот:



112.4. Создание пользовательского диалогового окна выполнения

Создав класс Custom Progress Dialog, можно использовать диалог для отображения в экземпляре UI, не пересоздавая его.

Сначала создайте пользовательский класс диалога прогресса (выполнения).

CustomProgress.java

```
public class CustomProgress {
    public static CustomProgress customProgress = null;
    private Dialog mDialog;

    public static CustomProgress getInstance() {
        if (customProgress == null) {
            customProgress = new CustomProgress();
        }
        return customProgress;
    }

    public void showProgress(Context context, String message, boolean cancelable) {
        mDialog = new Dialog(context);
        mDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        mDialog.setContentView(R.layout.progress_bar_dialog);
        mProgressBar = (ProgressBar) mDialog.findViewById(R.id.progress_bar);
        // mProgressBar.getIndeterminateDrawable().setColorFilter(context.
getResources()
        // .getColor(R.color.material_blue_gray_500), PorterDuff.Mode.SRC_IN);
        TextView progressText = (TextView) mDialog.findViewById(R.id.progress_text);
        progressText.setText(" " + message);
        progressText.setVisibility(View.VISIBLE);
        mProgressBar.setVisibility(View.VISIBLE);
        // вы можете изменить или добавить эту строку в соответствии с потребностями
        mProgressBar.setIndeterminate(true);
        mDialog.setCancelable(cancelable);
        mDialog.setCanceledOnTouchOutside(cancelable);
        mDialog.show();
    }

    public void hideProgress() {
        if (mDialog != null) {
            mDialog.dismiss();
            mDialog = null;
        }
    }
}
```

Теперь создадим пользовательский макет прогресса (выполнения):

progress_bar_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="65dp"
    android:background="@android:color/background_dark"
    android:orientation="vertical">

    <TextView
        android:id="@+id/progress_text"
```

```

    android:layout_width="wrap_content"
    android:layout_height="40dp"
    android:layout_above="@+id/progress_bar"
    android:layout_marginLeft="10dp"
    android:layout_marginStart="10dp"
    android:background="@android:color/transparent"
    android:gravity="center_vertical"
    android:text=""
    android:textColor="@android:color/white"
    android:textSize="16sp"
    android:visibility="gone" />

<-- СТИЛЬ может быть изменен на любой вид ProgressBar -->

<ProgressBar
    android:id="@+id/progress_bar"
    style="@android:style/Widget.DeviceDefault.ProgressBar.Horizontal"
    android:layout_width="match_parent"
    android:layout_height="30dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_gravity="center"
    android:background="@color/cardview_dark_background"
    android:maxHeight="20dp"
    android:minHeight="20dp" />

```

</RelativeLayout>

Вот и все. Теперь вызов диалога в коде:

```

CustomProgress customProgress = CustomProgress.getInstance();

// теперь у вас есть экземпляр CustomProgress
// для отображения ProgressBar

customProgress.showProgress(#Context, getString(#StringId), #boolean);

// для скрытия ProgressBar

customProgress.hideProgress();

```

112.5. Неопределенный ProgressBar

Неопределенный ProgressBar показывает циклическую анимацию без индикации прогресса.

Базовый неопределенный ProgressBar («вращающееся колесо»):

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Горизонтальный неопределенный ProgressBar (также известен как «плоский», flat bar):

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Другие встроенные стили ProgressBar:

```
style="@android:style/Widget.ProgressBar.Small"
style="@android:style/Widget.ProgressBar.Large"
style="@android:style/Widget.ProgressBar.Inverse"
style="@android:style/Widget.ProgressBar.Small.Inverse"
style="@android:style/Widget.ProgressBar.Large.Inverse"
```

Чтобы использовать неопределенный ProgressBar в активности:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

112.6. Определенный ProgressBar

Такой индикатор ProgressBar показывает текущее продвижение к определенному максимальному значению.

Горизонтальный определенный ProgressBar:

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Вертикальный определенный ProgressBar:

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="10dp"
    android:layout_height="match_parent"
    android:progressDrawable="@drawable/progress_vertical"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res/drawable/progress_vertical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/background">
        <shape>
            <corners android:radius="3dp"/>
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
    <item android:id="@+id/secondaryProgress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_light"/>
            </shape>
        </clip>
    </item>
</layer-list>
```

```

        </shape>
    </clip>
</item>
<item android:id="@+id/progressBar"
      <clip android:clipOrientation="vertical" android:gravity="bottom">
        <shape>
          <corners android:radius="3dp"/>
          <solid android:color="@android:color/holo_blue_dark"/>
        </shape>
      </clip>
</item>
</layer-list>

```

Кольцевой определенный ProgressBar:

```

<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:progressDrawable="@drawable/progress_ring"
    style="@android:style/Widget.ProgressBar.Horizontal"/>

```

res/drawable/progress_ring.xml

```

<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/secondaryProgress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#0000FF"/>
        </shape>
    </item>

    <item android:id="@+id/progress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#FFFFFF"/>
        </shape>
    </item>
</layer-list>

```

Для использования определенного ProgressBar в активности:

```

ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);

```

Глава 113. Пользовательские шрифты

113.1. Пользовательский шрифт в тексте объекта Canvas (холста)

Отрисовка текста на холсте (Canvas) своим шрифтом из активов (assets):

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Ваш текст здесь", x, y, textPaint);
```

113.2. Новые возможности для работы со шрифтами

Функция под названием *Fonts in XML* позволяет использовать шрифты в качестве ресурсов. Нет необходимости упаковывать шрифты как активы (assets), теперь они компилируются в R-файл и автоматически доступны в системе как ресурс.

Для того чтобы добавить новый шрифт, необходимо выполнить следующие действия:

- Создайте новый каталог ресурсов: res/font.
- Добавьте в эту папку файлы шрифтов. Например, добавив myfont.ttf, вы сможете использовать этот шрифт через R.font.myfont.

Вы также можете создать собственное **семейство шрифтов**, добавив в каталог res/font следующий XML-файл:

```
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
    <font
        android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/lobster_regular" />
    <font
        android:fontStyle="italic"
        android:fontWeight="400"
        android:font="@font/lobster_italic" />
</font-family>
```

Вы можете одинаково использовать как файл шрифта, так и файл семейства шрифтов:

- В XML-файле с помощью атрибута android:fontFamily.

Например так:

```
<TextView
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:fontFamily="@font/myfont"/>
```

...или вот так:

```
<style name="customfontstyle" parent="@android:style/TextAppearance.Small">
    <item name="android:fontFamily">@font/myfont</item>
</style>
```

- В своем коде, используя следующие строки:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

113.3. Пользовательский шрифт для всей активности

```
public class ReplaceFont {
    public static void changeDefaultFont(Context context, String oldFont, String assetsFont) {
        Typeface typeface = Typeface.createFromAsset(context.getAssets(), assetsFont);
        replaceFont(oldFont, typeface);
    }

    private static void replaceFont(String oldFont, Typeface typeface) {
        try {
            Field myField = Typeface.class.getDeclaredField(oldFont);
            myField.setAccessible(true);
            myField.set(null, typeface);
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
}
```

Затем в вашей активности в методе onCreate():

```
// Поместите шрифт в папку assets...
ReplaceFont.changeDefaultFont(getApplicationContext(), "DEFAULT", "LinLibertine.ttf");
```

113.4. Использование пользовательского шрифта в приложении

1. Перейдите в папку проекта.
2. Затем app -> src -> main.
3. Создайте папку 'assets -> fonts' в основной папке.
4. Поместите файл 'fontfile.ttf' в папку fonts.

113.5. Инициализация шрифта

```
private Typeface myFont;

// Хорошей практикой может быть вызов этого в onCreate() пользовательского класса
// приложения и передача 'this' в качестве Context. Ваш шрифт можно
// будет использовать до тех пор, пока открыто ваше приложение
public void initFont(Context context) {
    myFont = Typeface.createFromAsset(context.getAssets(), "fonts/Roboto-Light.ttf");
}
```

113.6. Использование пользовательского шрифта в TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

113.7. Применение шрифта к TextView с помощью xml (без Java-кода)

TextViewPlus.java:

```
public class TextViewPlus extends TextView {
```

```

private static final String TAG = "TextView";

public TextViewPlus(Context context) {
    super(context);
}

public TextViewPlus(Context context, AttributeSet attrs) {
    super(context, attrs);
    setCustomFont(context, attrs);
}

public TextViewPlus(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    setCustomFont(context, attrs);
}

private void setCustomFont(Context ctx, AttributeSet attrs) {
    TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.TextViewPlus);
    String customFont = a.getString(R.styleable.TextViewPlus_customFont);
    setCustomFont(ctx, customFont);
    a.recycle();
}

public boolean setCustomFont(Context ctx, String asset) {
    Typeface typeface = null;
    try {
        typeface = Typeface.createFromAsset(ctx.getAssets(), asset);
    } catch (Exception e) {
        Log.e(TAG, "Unable to load typeface: "+e.getMessage());
        return false;
    }
    setTypeface(typeface);
    return true;
}
}

```

attrs.xml (где разместить res/values):

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>

```

Способ применения:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:foo="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <com.mypackage.TextViewPlus
        android:id="@+id/textViewPlus1"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:text="@string/showingOffTheNewTypeface" />

```

```

    foo:customFont="my_font_name_regular.otf">
</com.mypackage.TextViewPlus>
</LinearLayout>

```

113.8. Эффективная загрузка шрифтов

Загрузка пользовательских шрифтов может привести к ухудшению производительности. Поэтому рекомендуется использовать этот небольшой помощник, который сохраняет и загружает уже используемые шрифты в таблицу Hashtable:

```

public class TypefaceUtils {

private static final Hashtable<String, Typeface> sTypeFaces = new Hashtable<>();

/** "шрифт" + "\u207a\ufe0f" */
Получить шрифт по имени файла из основной директории активов
*
@param context
@param fileName имя файла шрифта в основном каталоге актива
@return
*/
public static Typeface getTypeFace(final Context context, final String fileName) {
    Typeface tempTypeface = sTypeFaces.get(fileName);

    if (tempTypeface == null) {
        tempTypeface = Typeface.createFromAsset(context.getAssets(), fileName);
        sTypeFaces.put(fileName, tempTypeface);
    }
    return tempTypeface;
}
}

Использование:
Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);

```

Глава 114. Получение имен системных шрифтов и их использование

В следующих примерах показано, как получить имена системных шрифтов, хранящихся в каталоге /system/fonts/, и как использовать системный шрифт для установки шрифтовой гарнитуры элемента TextView.

114.1. Получение имен системных шрифтов

```

ArrayList<String> fontNames = new ArrayList<String>();
File temp = new File("/system/fonts/");
String fontSuffix = ".ttf";

```

```

for(File font : temp.listFiles()) {
    String fontName = font.getName();
    if(fontName.endsWith(fontSuffix)) {
        fontNames.add(fontName.subSequence(0, fontName.lastIndexOf(fontSuffix)).toString());
    }
}

```

114.2. Применение системного шрифта к TextView

В следующем коде необходимо заменить `fontsname` на имя шрифта, который вы хотите использовать:

```

TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));

```

Глава 115. Преобразование текста в речь (TTS)

115.1. Основы преобразования текста в речь

layout_text_to_speech.xml

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Введите текст здесь!"
        android:id="@+id/textToSpeak"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/textToSpeak"
        android:id="@+id/btnSpeak"/>

</RelativeLayout>

```

AndroidTextToSpeechActivity.java

```

public class AndroidTextToSpeechActivity extends Activity implements
    TextToSpeech.OnInitListener {

    EditText textToSpeak = null;
    Button btnSpeak = null;
    TextToSpeech tts;

```

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    textToSpeak = findViewById(R.id.textToSpeak);
    btnSpeak = findViewById(R.id.btnSpeak);
    btnSpeak.setEnabled(false);
    tts = new TextToSpeech(this, this);
    btnSpeak.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            speakOut();
        }
    });
}

@Override
public void onDestroy() {
    // Не забудьте выключить tts!
    if (tts != null) {
        tts.stop();
        tts.shutdown();
    }
    super.onDestroy();
}

@Override
public void OnInit(int status) {
    if (status == TextToSpeech.SUCCESS) {
        int result = tts.setLanguage(Locale.US);

        if (result == TextToSpeech.LANG_MISSING_DATA
            || result == TextToSpeech.LANG_NOT_SUPPORTED) {
            Log.e("TTS", "Этот язык не поддерживается");
        } else {
            btnSpeak.setEnabled(true);
            speakOut();
        }
    } else {
        Log.e("TTS", "Initialization Failed!");
    }
}

private void speakOut() {
    String text = textToSpeak.getText().toString();
    if(text == null || text.isEmpty())
        return;

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        String utteranceId=this.hashCode() + "";
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, utteranceId);
    } else {
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
    }
}
}

```

Язык, на котором будет звучать речь, может быть задан путем указания Locale методу setLanguage():

```
tts.setLanguage(Locale.CHINESE); // Китайский язык
```

Количество поддерживаемых языков варьируется в зависимости от версии Android. Проверить, поддерживается ли тот или иной язык, можно с помощью метода `isLanguageAvailable()`:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

Уровень высоты тона речи может быть установлен с помощью метода `setPitch()`. По умолчанию значение высоты тона равно 1.0. Используйте значения меньше 1.0 для уменьшения уровня высоты тона или значения больше 1.0 для увеличения уровня высоты тона:

```
tts.setPitch(0.6);
```

Скорость речи можно установить с помощью метода `setSpeechRate()`. По умолчанию она равна 1.0. Скорость можно увеличить вдвое, установив значение 2.0, или уменьшить вдвое, установив значение 0.5:

```
tts.setSpeechRate(2.0);
```

115.2. Реализация TTS в API-интерфейсах

В случае «холодной наблюдаемой реализации» (Cold observable implementation) получаем `true`, когда TTS-движок заканчивает генерировать речь; а начинает он генерировать речь только если есть подписка. Обратите внимание, что начиная с уровня API 21 есть разные способы работы с речью:

```
public class RxTextToSpeech {
    @Nullable RxTTSObservableOnSubscribe audio;
    WeakReference<Context> contextRef;

    public RxTextToSpeech(Context context) {
        this.contextRef = new WeakReference<>(context);
    }

    public void requestTTS(FragmentActivity activity, int requestCode) {
        Intent checkTTSIntent = new Intent();
        checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
        activity.startActivityForResult(checkTTSIntent, requestCode);
    }

    public void cancelCurrent() {
        if (audio != null) {
            audio.dispose();
            audio = null;
        }
    }

    public Observable<Boolean> speak(String textToRead) {
        audio = new RxTTSObservableOnSubscribe(contextRef.get(), textToRead, Locale.GERMANY);
        return Observable.create(audio);
    }

    public static class RxTTSObservableOnSubscribe extends UtteranceProgressListener
        implements ObservableOnSubscribe<Boolean>, Disposable, Cancellable, TextToSpeech.OnInitListener {
```

```
volatile boolean disposed;
ObservableEmitter<Boolean> emitter;
TextToSpeech textToSpeech;
String text = "";
Locale selectedLocale;
Context context;

public RxTTSObserverableOnSubscribe(Context context, String text, Locale locale) {
    this.selectedLocale = locale;
    this.context = context;
    this.text = text;
}

@Override public void subscribe(ObservableEmitter<Boolean> e) throws Exception {
    this.emitter = e;
    if (context == null) {
        this.emitter.onError(new Throwable("nullable context, cannot execute "
+ text));
    } else {
        this.textToSpeech = new TextToSpeech(context, this);
    }
}

@Override @DebugLog public void dispose() {
    if (textToSpeech != null) {
        textToSpeech.setOnUtteranceProgressListener(null);
        textToSpeech.stop();
        textToSpeech.shutdown();
        textToSpeech = null;
    }
    disposed = true;
}

@Override public boolean isDisposed() {
    return disposed;
}

@Override public void cancel() throws Exception {
    dispose();
}

@Override public void onInit(int status) {
    int languageCode = textToSpeech.setLanguage(selectedLocale);

    if (languageCode == android.speech.tts.TextToSpeech.LANG_COUNTRY_AVAILABLE)
    {
        textToSpeech.setPitch(1);
        textToSpeech.setSpeechRate(1.0f);
        textToSpeech.setOnUtteranceProgressListener(this);
        performSpeak();
    } else {
        emitter.onError(new Throwable("language " + selectedLocale.getCountry()
+ " is not supported"));
    }
}

@Override public void onStart(String utteranceId) {
```

```

    //no-op
}

@Override public void onDone(String utteranceId) {
    this.emitter.onNext(true);
    this.emitter.onComplete();
}

@Override public void onError(String utteranceId) {
    this.emitter.onError(new Throwable("error TTS " + utteranceId));
}

void performSpeak() {

    if (isAtLeastApiLevel(21)) {
        speakWithNewApi();
    } else {
        speakWithOldApi();
    }
}

@RequiresApi(api = 21) void speakWithNewApi() {
    Bundle params = new Bundle();
    params.putString(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, "");
    textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, params, uniqueId());
}

void speakWithOldApi() {
    HashMap<String, String> map = new HashMap<>();
    map.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, uniqueId());
    textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, map);
}

private String uniqueId() {
    return UUID.randomUUID().toString();
}
}

public static boolean isAtLeastApiLevel(int apiLevel) {
    return Build.VERSION.SDK_INT >= apiLevel;
}
}

```

Глава 116. Спиннер

116.1. Пример спиннера

Spinner – это тип выпадающего ввода (dropdown input).

В первую очередь внесите в макет код:

<Spinner

 android:id="@+id/spinner" <!-- идентификатор для обращения к спиннеру из JAVA-->
 android:layout_width="match_parent"

```
    android:layout_height="wrap_content">
</Spinner>
```

Теперь следует заполнить значения в спиннере. Для этого существует два основных способа.

1. Из самого XML создайте array.xml в каталоге values внутри каталога res. Создание этого массива:

```
<string-array name="defaultValue">
    <item>--Выберите район города--</item>
    <item>--Выберите район города--</item>
    <item>--Выберите район города--</item>
</string-array>
```

Теперь добавьте в XML спиннера следующую строку:

```
    android:entries="@array/defaultValue"
```

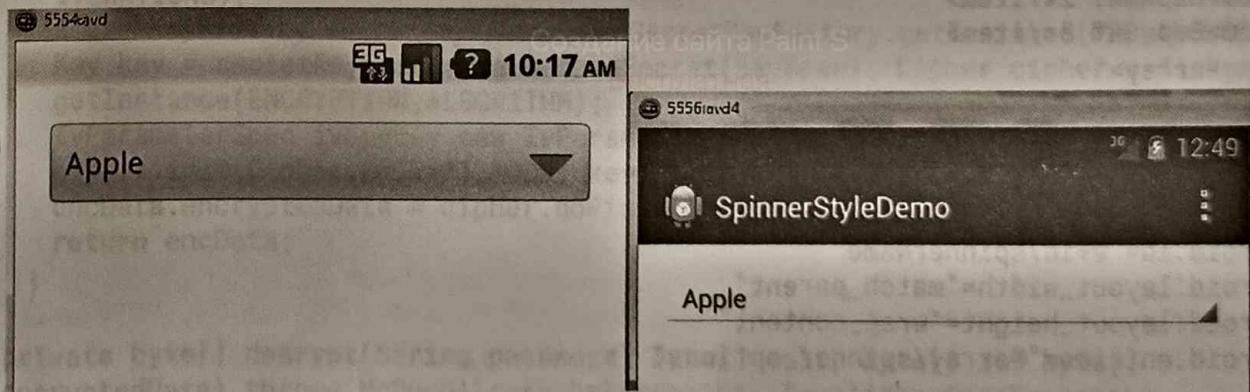
2. Вы также можете добавить значения с помощью JAVA, в активности или во фрагменте добавьте:

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Теперь создадим ArrayList из строк (Strings):

```
ArrayList<String> area = new ArrayList<>();
//добавим значения в область arrayList
cityArea.setAdapter(new ArrayAdapter<String>(context
        , android.R.layout.simple_list_item_1, area));
```

Это будет выглядеть следующим образом:



Стиль отображения будет зависеть от целевой версии приложения (targetSdkVersion). Спиннер может быть легко настроен пользователем с помощью xml, например:

```
    android:background="@drawable/spinner_background"
    android:layout_margin="16dp"
    android:padding="16dp"
```

Можно создать пользовательский фон в XML и использовать его. Позицию и другие сведения о выбранном элементе в спиннере легко получить при помощи кода:

```
cityArea.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```

@Override
public void onItemSelected(AdapterView<?> parent, View view, int position,
    long id) {
    areaNo = position;
}

@Override
public void onNothingSelected(AdapterView<?> parent) {

}

});

```

Изменение цвета текста выделенного элемента в спиннере можно осуществить двумя способами – в XML:

```
<item android:state_activated="true" android:color="@color/red"/>
```

Это изменит цвет выбранного элемента во всплывающем окне. Из JAVA можно изменить цвет текста так (в `setOnItemSelectedListener(...)`):

```

@Override
public void onItemSelected(AdapterView<?> parent, View view, int position,
    long id) { ((TextView) parent.getChildAt(0)).setTextColor(0x000000);
// аналогично изменяем цвет фона `background color` и т.д.
}

```

116.2. Добавление спиннера в активность

В файле `/res/values/strings.xml`:

```

<string-array name="spinner_options">
    <item>Вариант 1</item>
    <item>Вариант 2</item>
    <item>Вариант 3</item>
</string-array>

```

В макете XML:

```

<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />

```

В активности:

```

Spinner spinnerName = (Spinner) findViewById(R.id.spinnerName);
spinnerName.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long
        id) {
        String chosenOption = (String) parent.getItemAtPosition(position);
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});

```

Глава 117. Шифрование и расшифровка данных

117.1. Безопасное AES-шифрование данных с использованием пароля

В следующем примере заданный блок данных шифруется с помощью симметричного алгоритма блочного шифрования AES (https://en.wikipedia.org/wiki/Advanced_Encryption_Standard). Ключ шифрования получен безопасным способом (случайная «соль», 1000 раундов SHA-256). Для шифрования используется AES в режиме CBC со случайным IV.

Обратите внимание, что данные, хранящиеся в классе EncryptedData (salt, iv и encryptedData), могут быть сведены в один байтовый массив. Затем эти данные можно сохранить или передать получателю.

```

private static final int SALT_BYTES = 8;
private static final int PBK_ITERATIONS = 1000;
private static final String ENCRYPTION_ALGORITHM = "AES/CBC/PKCS5Padding";
private static final String PBE_ALGORITHM = "PBEwithSHA256and128BITAES-CBC-BC";

private EncryptedData encrypt(String password, byte[] data) throws
NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeySpecException,
InvalidKeyException, BadPaddingException, IllegalBlockSizeException,
InvalidAlgorithmParameterException {
    EncryptedData encData = new EncryptedData();
    SecureRandom rnd = new SecureRandom();
    encData.salt = new byte[SALT_BYTES];
    encData.iv = new byte[16]; // Размер блока AES
    rnd.nextBytes(encData.salt); rnd.nextBytes(encData.iv);

    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), encData.salt, PBK_
ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec); Cipher cipher = Cipher.
getInstance(ENCRYPTION_ALGORITHM);
    IvParameterSpec ivSpec = new IvParameterSpec(encData.iv);
    cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    encData.encryptedData = cipher.doFinal(data);
    return encData;
}

private byte[] decrypt(String password, byte[] salt, byte[] iv, byte[]
encryptedData) throws NoSuchAlgorithmException, InvalidKeySpecException,
NoSuchPaddingException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException, InvalidAlgorithmParameterException {
    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, PBK_
ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
    return cipher.doFinal(encryptedData);
}

private static class EncryptedData {
    public byte[] salt;
}

```

```

public byte[] iv;
public byte[] encryptedData;
}

```

Следующий пример кода показывает, как тестировать шифрование и расшифровку:

```

try {
    String password = "test12345";
    byte[] data = "plaintext11223344556677889900".getBytes("UTF-8");
    EncryptedData encData = encrypt(password, data);
    byte[] decryptedData = decrypt(password, encData.salt, encData.iv, encData.
        encryptedData);
    String decDataAsString = new String(decryptedData, "UTF-8");
    Toast.makeText(this, decDataAsString, Toast.LENGTH_LONG).show();
} catch (Exception e) {
    e.printStackTrace();
}

```

Глава 118. OkHttp

118.1. Пример базового использования

Хорошей практикой является обворачивание OkHttp в класс HttpClient, в этом классе находятся методы для каждого из основных HTTP-глаголов – post, get, put и delete. (Удобно включать туда интерфейс с целью сохранить для него реализацию, чтобы при необходимости можно было легко перейти на другую реализацию):

```

public class HttpClient implements HttpClientInterface{

private static final String TAG = OkHttpClient.class.getSimpleName();
public static final MediaType JSON
    = MediaType.parse("application/json; charset=utf-8");
OkHttpClient httpClient = new OkHttpClient();

@Override
public String post(String url, String json) throws IOException {
    Log.i(TAG, "Отправка почтового запроса с body:\n" + json + "\n на URL: " + url);

    RequestBody body = RequestBody.create(JSON, json);
    Request request = new Request.Builder()
        .url(url)
        .post(body)
        .build();
    Response response = httpClient.newCall(request).execute();
    return response.body().string();
}

```

Синтаксис для методов put, get и delete одинаков, за исключением одного слова, к примеру (.put(body)), поэтому публиковать и этот код нецелесообразно. Использование довольно простое, достаточно вызвать соответствующий метод по некоторому url с некоторой полезной нагрузкой в виде json, и метод вернет в качестве результата строку, которую можно будет впоследствии использовать и разобрать. Предположим, что ответ будет в виде json, из него мы можем легко создать JSONObject:

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// продолжаем разбирать ответ в соответствии с его структурой
```

118.2. Настройка OkHttp

С помощью Maven:

```
<dependency>
<groupId>com.squareup.okhttp3</groupId>
<artifactId>okhttp</artifactId>
<version>3.6.0</version>
</dependency>
```

...или Gradle:

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

118.3. Перехватчик протоколирования

Перехватчики (Interceptors) используются для перехвата вызовов OkHttp. Причинами перехвата могут быть мониторинг, перезапись и повторные вызовы. Он может использоваться как для исходящего запроса, так и для входящего ответа:

```
class LoggingInterceptor implements Interceptor {
    @Override public Response intercept(Interceptor.Chain chain) throws IOException {
        Request request = chain.request();

        long t1 = System.nanoTime();
        logger.info(String.format("Отправка запроса %s на %s%n%s",
                request.url(), chain.connection(), request.headers()));

        Response response = chain.proceed(request);

        long t2 = System.nanoTime();
        logger.info(String.format("Получен ответ за %s в %.1fms%n%s",
                request().url(), (t2 - t1) / 1e6d, response.headers()));

        return response;
    }
}
```

118.4. Синхронный вызов Get

```
private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Неожиданный код " +
            response);

    Headers responseHeaders = response.headers();
    System.out.println(response.body().string());
}
```

118.5. Асинхронный вызов Get

```

private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    client.newCall(request).enqueue(new Callback() {
        @Override public void onFailure(Call call, IOException e) {
            e.printStackTrace();
        }

        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if (!response.isSuccessful()) throw new IOException("Неожиданный код " +
                response);

            Headers responseHeaders = response.headers();
            System.out.println(response.body().string());
        }
    });
}

```

118.6. Параметры формы отправки сообщения

```

private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
   RequestBody formBody = new FormBody.Builder()
        .add("search", "Jurassic Park")
        .build();
    Request request = new Request.Builder()
        .url("https://en.wikipedia.org/w/index.php")
        .post(formBody)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Неожиданный код " +
        response);

    System.out.println(response.body().string());
}

```

118.7. Отправка многосоставного запроса

```

private static final String IMGUR_CLIENT_ID = "...";
private static final MediaType MEDIA_TYPE_PNG = MediaType.parse("image/png");

private final OkHttpClient client = new OkHttpClient();

public void run() throws Exception {
    // Используйте API для загрузки изображений imgur, как описано на сайте
    // https://api.imgur.com/endpoints/image.
    RequestBody requestBody = new MultipartBody.Builder()

```

```

.setBoundary("boundary")
.setCharset("UTF-8")
.setContentType(MultipartBody.FORM)
.addFormDataPart("title", "Square Logo")
.addFormDataPart("image", "logo-square.png",
    RequestBody.create(MEDIA_TYPE_PNG, new File("website/static/logo-square.png")))
.build();

Request request = new Request.Builder()
    .header("Authorization", "Client-ID " + IMGUR_CLIENT_ID)
    .url("https://api.imgur.com/3/image")
    .post(requestBody)
    .build();

Response response = client.newCall(request).execute();
if (!response.isSuccessful()) throw new IOException("Неожиданный код " +
    response);

System.out.println(response.body().string());
}

```

118.8. Перезапись ответов

```

private static final Interceptor REWRITE_CACHE_CONTROL_INTERCEPTOR = new
Interceptor() {
    @Override public Response intercept(Interceptor.
    Chain) throws IOException { Response
originalResponse = chain.proceed(chain.
request()); return originalResponse.newBuilder()
    .header("Cache-Control", "max-age=60")
    .build();
}
};

```

Глава 119. Обработка глубоких ссылок

Атрибут	Подробности
scheme	Часть схемы в URI (чувствительна к регистру). Примеры: http, https, ftp
host	Хостовая часть URI (с учетом регистра). Примеры: google.com, example.org
port	Портовая часть URI. Примеры: 80, 443
path	Часть пути в URI. Должна начинаться с /. Примеры: /, /about
pathPrefix	Префикс для части пути в URI. Примеры: /item, /article
pathPattern	Шаблон для поиска в части пути URI. Примеры: /item/.*, /article/[0-9]*.
imeType	Тип mime, с которым должно быть соответствие. Примеры: image/jpeg, audio/*

Глубокие ссылки – это URL-адреса, которые ведут пользователей непосредственно к определенному контенту в приложении. Вы можете настроить глубокие ссылки, добавив фильтры намерений и извлекая данные из входящих намерений, чтобы направить пользователей на нужный экран в приложении.

119.1. Получение параметров запроса

```

public class MainActivity extends Activity {

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

        Intent intent = getIntent();
        Uri data = intent.getData();

        if (data != null) {
            String param1 = data.getQueryParameter("param1");
            String param2 = data.getQueryParameter("param2");
        }
    }
}

```

В этом случае, если пользователь перейдет по ссылке на сайт <http://www.example.com/map?param1=FOO¶m2=BAR>, то параметр param1 здесь будет иметь значение “FOO”, а параметр param2 – значение “BAR”.

119.2. Простая глубокая ссылка

AndroidManifest.xml:

```

<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />
    </intent-filter>
</activity>

```

При этом в качестве глубокой ссылки для запуска MainActivity будет принята любая ссылка, начинающаяся с <http://www.example.com>.

119.3. Несколько путей в одном домене

AndroidManifest.xml:

```

<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />
    </intent-filter>
</activity>

```

```

<data android:path="/" />
<data android:path="/about" />
<data android:path="/map" />

</intent-filter>

</activity>

```

Это приведет к запуску вашей MainActivity, когда пользователь нажмет на любую из этих ссылок:

- <http://www.example.com/>
- <http://www.example.com/about>
- <http://www.example.com/map>

119.4. Несколько доменов и несколько путей

AndroidManifest.xml:

```

<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:scheme="http"
              android:host="www.example2.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>

```

Это приведет к запуску вашей MainActivity, когда пользователь щелкнет на любой из этих ссылок:

- <http://www.example.com/>
- <http://www.example2.com/>
- <http://www.example.com/map>
- <http://www.example2.com/map>

119.5. http и https для одного и того же домена

AndroidManifest.xml:

```

<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http" />
        <data android:scheme="https" />
    </intent-filter>

```

```

<data android:host="www.example.com" />
<data android:path="/" />
<data android:path="/map" />

</intent-filter>
</activity>

```

Это приведет к запуску вашей `MainActivity`, когда пользователь щелкнет на любой из этих ссылок:

- <http://www.example.com/>
- <https://www.example.com/>
- <http://www.example.com/map>
- <https://www.example.com/map>

119.6. Использование pathPrefix

`AndroidManifest.xml`:

```

<activity android:name="com.example.MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com"
              android:path="/item" />
    </intent-filter>
</activity>

```

Это приведет к запуску вашей `MainActivity` при нажатии пользователем любой ссылки, начинающейся с `http://www.example.com/item`, например:

- <https://www.example.com/item>
- <http://www.example.com/item/1234>
- <https://www.example.com/item/xyz/details>

Глава 120. Создание отчетов о сбоях

120.1. Fabric – Crashlytics

Fabric – это модульная мобильная платформа, предоставляющая полезные наборы, которые можно использовать для создания приложений. *Crashlytics* – это инструмент отчетности о сбоях и проблемах, предоставляемый *Fabric*, который позволяет детально отслеживать и контролировать работу приложений.

Как настроить Fabric-Crashlytics

Шаг 1: Измените свой build.gradle:

Добавьте репозиторий плагина и плагин gradle:

```
buildscript {
    repositories {
        maven { url 'https://maven.fabric.io/public' }
    }

    dependencies {
        // Плагин Fabric Gradle использует открытую (open ended) версию
        // для быстрого реагирования на обновление инструментария для Android
        classpath 'io.fabric.tools:gradle:1.+'
    }
}
```

Примените плагин:

```
apply plugin: 'com.android.application'
// Поместить плагин Fabric после плагина Android
apply plugin: 'io.fabric'
```

Добавьте репозиторий Fabric:

```
repositories {
    maven { url 'https://maven.fabric.io/public' }
}
```

Добавьте комплект Crashlytics Kit:

```
dependencies {
    compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
        transitive = true;
    }
}
```

Шаг 2: Добавьте свой API-ключ и разрешение INTERNET в AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
<application
    ...
    <meta-data
        android:name="io.fabric.ApiKey"
        android:value="25eeeca3bb31cd41577e097cabd1ab9eee9da151d"
    />
</application>
<uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

Шаг 3: Инициируйте Kit во время выполнения в коде, например:

```
public class MainActivity extends ActionBarActivity {

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

        //Инициализация Kit
        Fabric.with(this, new Crashlytics());

        setContentView(R.layout.activity_main);
    }
}
```

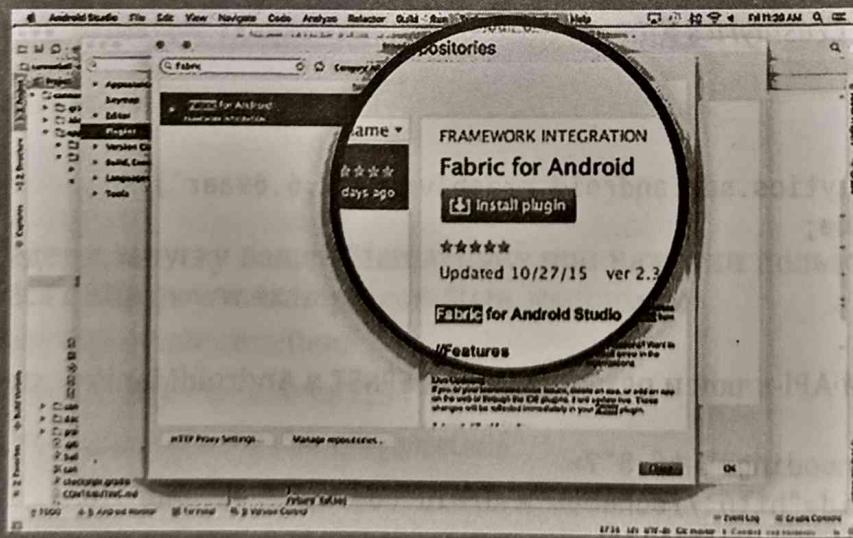
Шаг 4: Сборка проекта. Для сборки и запуска:



Использование плагина Fabric IDE

Установить комплекты (Kits) можно с помощью плагина Fabric IDE для Android Studio или IntelliJ по ссылке: <https://fabric.io/downloads/android>.

Android Studio / IntelliJ



Search for 'Fabric for Android' and install the plugin.

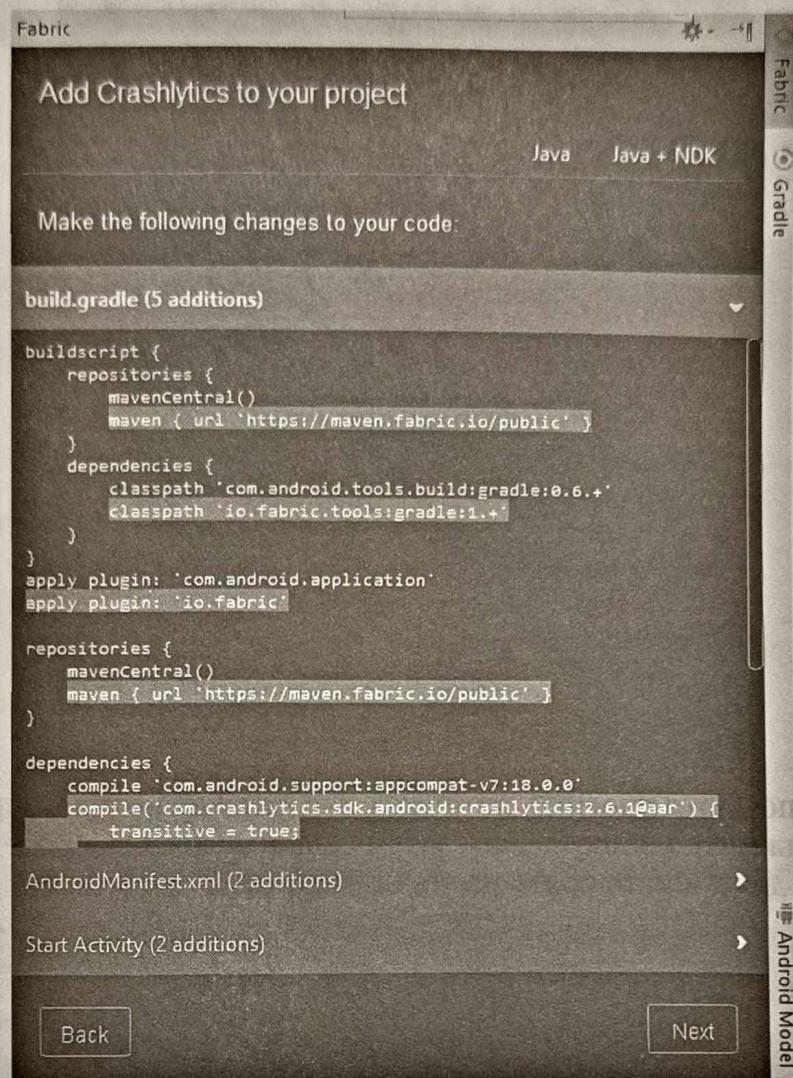
После установки **перезапустите** Android Studio и войдите под своей учетной записью (клавиатурное сочетание CTRL + L).



Затем на экране появятся проекты, которые у вас есть / проект, который вы открыли, выберите нужный и нажмите кнопку «next», затем снова «next». Выберите комплект, который вы хотите добавить, для данного примера это Crashlytics:

The screenshot shows the "All Kits" page on the Fabric website. The title "All Kits" is at the top. Below it, there is a list of integration options, each with an icon and a brief description. The options are: Crashlytics (Lightweight & Powerful crash reporting. Now with Answers, analytics you don't need to analyze.), Answers (Finally, mobile app analytics you don't need to analyze.), Twitter (Integrate Log in with Twitter, embed Tweets, and leverage the Twitter API.), Digits (Add phone number signup and login to your app with just a few lines of code.), MoPub (Monetize your app with the world's most powerful ads platform.), and Optimizely (Drive in-app engagement with fast, powerful mobile A/B testing and staged feature rollouts.). At the bottom left is a "Back" button, and on the right side, there are vertical tabs for "Fabric", "Gradle", and "Android Model".

Затем нажмите «Install». В этот раз вам не нужно ничего добавлять вручную, как в случае с вышеупомянутым gradle-плагином, вся работа будет выполнена автоматически.



Готово!

120.2. Перехват сбоев с помощью Sherlock

Полезный для разработчика инструмент Sherlock (<https://github.com/ajitsing/Sherlock>) фиксирует все сбои и сообщает о них в виде уведомления. При нажатии на уведомление открывается активность со всеми подробностями о произошедшем сбое, а также информацией об устройстве и приложении.

Как интегрировать Sherlock в ваше приложение?

Для этого достаточно добавить Sherlock в качестве gradle-зависимости в свой проект:

```

dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}

```

После синхронизации Android Studio инициализируйте Sherlock в классе Application:

```

package com.singhajit.login;

import android.app.Application;

import com.singhajit.sherlock.core.Sherlock;

```

```
public class SampleApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        Sherlock.init(this);
    }
}
```

Это все, что вам нужно сделать. Кроме того, Sherlock умеет делать гораздо больше, чем просто сообщать о сбоях. Для ознакомления со всеми его возможностями прочтите эту статью: <http://www.singhajit.com/integrating-sherlock-with-android-apps-to-get-crash-reports/>.

120.3. Принудительный краш-тест при помощи Fabric

Добавим кнопку, нажав на которую можно вызвать сбой. Вставьте этот код в макет в том месте, где должна появиться кнопка.

```
<Button
    android:layout_height="wrap_content" android:layout_width="wrap_content"
    android:text="Force Crash!" android:onClick="forceCrash"

    android:layout_centerHorizontal="true" />
```

Выдать исключение RuntimeException

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

Запустите приложение и нажмите кнопку «New», чтобы вызвать сбой. Через минуту или две вы сможете увидеть сбой на приборной панели Crashlytics, а также получите сообщение на e-mail.

120.4. Отчетность о сбоях с помощью ACRA

Шаг 1. Добавьте зависимость последней версии ACRA (см. <https://github.com/ACRA/acra/wiki/ChangeLog>) AAR в build.gradle.

Шаг 2. В класс приложения (класс, который расширяет Application; если его нет, то создайте его) добавьте аннотацию @ReportsCrashes и переопределите метод attachBaseContext().

Шаг 3. Инициализируйте класс ACRA в классе приложения.

```
@ReportsCrashes(
    formUri = "Your choice of backend",
    reportType = REPORT_TYPES(JSON/FORM),
    httpMethod = HTTP_METHOD(POST/PUT),
    formUriBasicAuthLogin = "AUTH_USERNAME",
    formUriBasicAuthPassword = "AUTH_PASSWORD",
    customReportContent = {
        ReportField.USER_APP_START_DATE,
        ReportField.USER_CRASH_DATE,
        ReportField.APP_VERSION_CODE,
        ReportField.APP_VERSION_NAME,
        ReportField.ANDROID_VERSION,
        ReportField.DEVICE_ID,
        ReportField.BUILD,
        ReportField.BRAND,
        ReportField.DEVICE_FEATURES,
```

```

ReportField.PACKAGE_NAME,
ReportField.REPORT_ID,
ReportField.STACK_TRACE,
},
mode = NOTIFICATION_TYPE(TOAST, DIALOG, NOTIFICATION)
resToastText = R.string.crash_text_toast)

public class MyApplication extends Application {
@Override
protected void attachBaseContext(Context base) {
super.attachBaseContext(base);
// Инициализация ACRA
ACRA.init(this);
}
}
}

```

...где AUTH_USERNAME и AUTH_PASSWORD – учетные данные нужных вам бэкендов (см. дополнительно <https://github.com/ACRA/acralyzer/wiki/setup>).

Шаг 4. Определите класс Application в файле AndroidManifest.xml:

```

<application
    android:name=".MyApplication">
    <service></service>
    <activity></activity>
    <receiver></receiver>
</application>

```

Шаг 5. Убедитесь, что у вас есть разрешение на получение отчета от давшего сбой приложения через Интернет:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

В случае если необходимо отправить «тихий» отчет в бэкенд, то для этого достаточно воспользоваться приведенным ниже методом:

```
ACRA.getErrorReporter().handleSilentException(e);
```

Глава 121. Проверка подключения к Интернету

Параметр

Подробности

Context

Ссылка на контекст активности

Этот метод используется для проверки наличия или отсутствия подключения к Wi-Fi.

121.1. Проверка наличия у устройства подключения к Интернету

Добавьте необходимые сетевые разрешения в файл манифеста приложения:

```

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />

```

```
/**  
Если сетевое подключение доступно, возвращается true  
*  
@param context - текущий контекст  
@return boolean true, если сетевое соединение доступно  
*/  
public static boolean isNetworkAvailable(Context context) {  
    ConnectivityManager connectivity = (ConnectivityManager) context  
        .getSystemService(Context.CONNECTIVITY_SERVICE);  
    if (connectivity == null) {  
        Log.d("NetworkCheck", "isNetworkAvailable: No");  
        return false;  
    }  
  
    // получить информацию о сети для всех интерфейсов передачи данных (например,  
    // Wi-Fi, 3G, LTE и т.д.)  
    NetworkInfo[] info = connectivity.getAllNetworkInfo();  
  
    // убедитесь, что существует хотя бы один интерфейс для тестирования  
    if (info != null) {  
        // итерация по интерфейсам  
        for (int i = 0; i < info.length; i++) {  
            // проверить данный интерфейс на наличие подключенного состояния  
            if (info[i].getState() == NetworkInfo.State.CONNECTED) {  
                Log.d("NetworkCheck", "isNetworkAvailable: Yes");  
                return true;  
            }  
        }  
    }  
    return false;  
}
```

121.2. Проверка качества сигнала в сети

Точное определение уровня сигнала в децибелах

```
ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.  
    CONNECTIVITY_SERVICE);  
    NetworkInfo Info = cm.getActiveNetworkInfo();  
    if (Info == null || !Info.isConnectedOrConnecting()) {  
        Log.i(TAG, "Нет соединения");  
    } else {  
        int netType = Info.getType();  
        int netSubtype = Info.getSubtype();  
  
        if (netType == ConnectivityManager.TYPE_WIFI) {  
            Log.i(TAG, "Wifi соединение");  
            WifiManager wifiManager = (WifiManager) getApplication().  
                getSystemService(Context.WIFI_SERVICE);  
            List<ScanResult> scanResult = wifiManager.getScanResults();  
            for (int i = 0; i < scanResult.size(); i++) {  
                Log.d("scanResult", "Скорость wifi "+scanResult.get(i).level);  
                //Уровень сигнала в dB  
            }  
  
            // Необходимо определить качество сигнала Wi-Fi  
        } else if (netType == ConnectivityManager.TYPE_MOBILE) {  
            Log.i(TAG, "GPRS/3G соединение");  
            // Необходимо провести различие между 3G/GPRS  
        }  
    }
```

Для проверки типа сети используйте этот класс.

```
public class Connectivity {  
    public static final int NETWORK_TYPE_EHRPD = 14; // Уровень 11  
    public static final int NETWORK_TYPE_EVDO_B = 12; // Уровень 9  
    public static final int NETWORK_TYPE_HSPAP = 15; // Уровень 13  
    public static final int NETWORK_TYPE_IDEN = 11; // Уровень 8  
    public static final int NETWORK_TYPE_LTE = 13; // Уровень 11  
  
    /**  
     * Проверка наличия подключения  
     *  
     * @param context  
     * @return  
     */  
    public static boolean isConnected(Context context) {  
        ConnectivityManager cm = (ConnectivityManager) context  
            .getSystemService(Context.CONNECTIVITY_SERVICE);  
        NetworkInfo info = cm.getActiveNetworkInfo();  
        return (info != null && info.isConnected());  
    }  
  
    /**  
     * Проверка наличия быстрого соединения  
     *  
     * @param context  
     * @return  
     */  
    public static String isConnectedFast(Context context) {  
        ConnectivityManager cm = (ConnectivityManager) context  
            .getSystemService(Context.CONNECTIVITY_SERVICE);  
        NetworkInfo info = cm.getActiveNetworkInfo();  
  
        if ((info != null && info.isConnected())) {  
            return Connectivity.isConnectionFast(info.getType(), info.getSubtype());  
        } else  
            return "Нет доступа к сети";  
    }  
  
    /**  
     * Проверка скорости соединения  
     *  
     * @param type  
     * @param subType  
     * @return  
     */  
    public static String isConnectionFast(int type, int subType) {  
        if (type == ConnectivityManager.TYPE_WIFI) {  
            System.out.println("Соединение по WIFI");  
            return "Соединение по WIFI";  
        } else if (type == ConnectivityManager.TYPE_MOBILE) {  
            switch (subType) {  
                case TelephonyManager.NETWORK_TYPE_1xRTT:  
                    return "Тип сети 1xRTT"; // ~ 50-100 kbps  
                case TelephonyManager.NETWORK_TYPE_CDMA:  
                    return "Тип сети CDMA (3G) Скорость: 2 Mbps"; // ~ 14-64 kbps  
                case TelephonyManager.NETWORK_TYPE_EDGE:
```

```
    return "Тип сети EDGE (2.75G) Скорость: 100-120 Kbps"; // ~ 50-100 kbps
case TelephonyManager.NETWORK_TYPE_EVDO_0:
    return "Тип сети EVDO_0"; // ~ 400-1000 kbps
case TelephonyManager.NETWORK_TYPE_EVDO_A:
    return "Тип сети EVDO_A"; // ~ 600-1400 kbps
case TelephonyManager.NETWORK_TYPE_GPRS:
    return "Тип сети GPRS (2.5G) Скорость: 40-50 Kbps"; // ~ 100 kbps
case TelephonyManager.NETWORK_TYPE_HSDPA:
    return "Тип сети HSDPA (4G) Скорость: 2-14 Mbps"; // ~ 2-14 Mbps
case TelephonyManager.NETWORK_TYPE_HSPA:
    return "Тип сети HSPA (4G) Скорость: 0.7-1.7 Mbps"; // ~ 700-1700 kbps
case TelephonyManager.NETWORK_TYPE_HSUPA:
    return "Тип сети HSUPA (3G) Скорость: 1-23 Mbps"; // ~ 1-23 Mbps
case TelephonyManager.NETWORK_TYPE_UMTS:
    return "Тип сети UMTS (3G) Скорость: 0.4-7 Mbps"; // ~ 400-7000 kbps
case Connectivity.NETWORK_TYPE_EHRPD:
    return "Тип сети EHRPD"; // ~ 1-2 Mbps
case Connectivity.NETWORK_TYPE_EVDO_B:
    return "Тип сети EVDO_B"; // ~ 5 Mbps
case Connectivity.NETWORK_TYPE_HSPAP:
    return "Тип сети HSPA+ (4G) Скорость: 10-20 Mbps"; // ~ 10-20 Mbps
case Connectivity.NETWORK_TYPE_IDEN:
    return "Тип сети IDEN"; // ~25 kbps
case Connectivity.NETWORK_TYPE_LTE:
    return "Тип сети LTE (4G) Скорость: 10+ Mbps"; // ~ 10+ Mbps
    // Неизвестный
case TelephonyManager.NETWORK_TYPE_UNKNOWN:
    return "Неизвестный тип сети";
default:
    return "";
}
else {
    return "";
}
```

Благодарности

Создание сайта Paint S

Большое спасибо всем участникам сообщества Stack Overflow Documentation, которые помогли предоставить читателю эти примеры. Присылайте материалы для публикации или обновления примеров на web@petercv.com.

- A.A. Глава 7
a.ch. Глава 8
Abdallah Alaraby Глава 41
Abdullah Глава 97
abhi Главы 41 и 47
Abhishek Jain Главы 1, 16, 40 и 47
abhishesh Глава 18
Abilash Глава 16
Ab_ Глава 62
adalPaRi Главы 62 и 96
Adam Ratzman Глава 78
adao7000 Главы 45 и 78
Adarsh Ashok Глава 10
Adhikari Bishwash Глава 33
Adil Saiyad Главы 63 и 114
Adnan Главы 51 и 96
Adrián Pérez Глава 13
AesSedai101 Глава 8
Ahmad Aghazadeh Главы 40, 44, 47, 53, 57, 84 и 115
ahmadalibaloch Глава 41
Ajit Singh Глава 120
Akash Patel Главы 16 и 37
Ala Eddine Jebali Глава 1
alanv Глава 83
Aleks G Глава 74
Aleksandar Stefanović Главы 1, 8, 37, 43 и 83
Alex Глава 59
Alex Bonel Глава 16
Alex Chengalan Главы 37 и 82
Alex Ershov Глава 77
Alex Sullivan Глава 95
alexey polusov Глава 69
Ali Sherafat Глава 21
Amit Глава 42
Amod Gokhale Глава 24
Anand Singh Глава 61
anatoli Глава 16
Anax Глава 146
Anderson K Глава 96
AndiGeeky Глава 121
Andrei T Глава 95
Andrew Brooke Главы 1, 41, 50 и 78
Andrew Fernandes Глава 41
AndroidMechanic Главы 3, 40, 41, 47 и 61
AndroidRuntimeException Главы 41, 47, 61, 76, 93 и 97
AndyRoid Глава 96
Anggrayudi H Глава 40
Anirudh Sharma Главы 16, 37, 41 и 47
Anish Mittal Глава 42
Anita Kunjir Глава 32
Ankit Sharma Глава 47
Ankur Aggarwal Глава 98
Anonsage Глава 116
anoo_radha Глава 34
antonio Главы 40, 41, 56 и 61
anupam_kamble Глава 72
AnV Глава 50
Apoorv Parmar Глава 1
appersiano Глава 80
aquib23 Глава 96
Arpit Gandhi Глава 89
Arth Tilva Глава 52
Aryan Najafi Глава 82
Ashish Ranjan Глава 40
astuter Главы 92 и 97
athor Глава 109
Atif Farrukh Глава 75
Aurasphere Главы 8 и 107
auval Главы 1, 2, 41, 42 и 47
Avinash R Глава 41
Ахе Глава 41
BadCash Главы 45 и 82
BalararamNayak Глава 19
Barend Главы 17 и 28
Bartek Lipinski Главы 8, 16, 37, 41, 47 и 81
Beena Глава 4

Благодарности

- Ben Глава 88
- Ben P. Глава 42
- Bhargavi Yamanuri Глава 91
- biddulph.r Глава 38
- Blackbelt Главы 2 и 40
- BlitzKraig Глава 43
- Blundering Philosopher Глава 81
- bpoiss Глава 41
- Braj Bhushan Singh Глава 18
- Brenden Kromhout Глава 47
- bricklore Глава 69
- BrickTop Глава 10
- Bryan Главы 9, 16 и 18
- Bryan Bryce Глава 39
- Buddy Главы 40 и 57
- Burhanuddin Rashid Глава 98
- busradeniz Глава 58
- Cabezas Глава 107
- Caique Oliveira Глава 39
- Carl Poole Глава 65
- Carlos Глава 14
- Carlos Borau Глава 43
- carvaq Глава 53
- CaseyB Главы 96 и 104
- Cassio Landim Глава 96
- cdeange Главы 47, 88, 105 и 106
- Charu Главы 1, 2, 8, 37, 41, 76, 78, 81, 100 и 120
- Chip Главы 21, 23 и 91
- Chirag SolankI Глава 16
- Chol Глава 61
- Code.IT Главы 34 и 41
- Cold Fire Глава 41
- cricket_007 Главы 42 и 47
- dakshbhattacharya Глава 37
- Dalija Prasnikar Главы 38 и 41
- Damian Kozlak Главы 40 и 42
- Dan Главы 14, 96 и 108
- Dan Hulme Главы 37 и 40
- Daniel Käfer Глава 41
- Daniel Nugent Главы 3, 8, 9, 12, 13, 16, 24, 38, 40, 41, 42, 45, 61, 62, 69, 70, 72, 74, 82, 88, 92, 96, 100, 109 и 110
- DanielDiSu Главы 21 и 41
- Daniele Segato Глава 1
- David Argyle Thacker Глава 39
- David Medenjak Главы 17 и 107
- davidgiga1993 Глава 27
- dev.mi Глава 37
- devnull69 Глава 70
- Dhaval Solanki Глава 96
- Dinesh Choudhary Главы 73 и 106
- Disk Crashер Глава 86
- Dmide Глава 25
- Don Chakkappan Глава 77
- Doron Behar Глава 1
- Doron Yakovlev Глава 55
- Douglas Drumond Глава 7
- Duan Bressan Глава 74
- Dus Глава 69
- Eixx Глава 81
- Ekin Глава 120
- EKN Глава 41
- Endzeit Глава 98
- EpicPandaForce Главы 107 и 108
- Er. Kaushik Kajavadara Глава 14
- Erik Minarini Главы 1, 41, 42 и 57
- Fabio Глава 47
- Felix Edelmann Глава 19
- Flayn Глава 59
- Floern Главы 8, 34, 38, 41, 47 и 71
- Florent Spahiу Главы 9 и 47
- Franck Dernoncourt Глава 41
- FromTheSeventhSky Глава 18
- fuwaneko Глава 18
- fyfyone Google Глава 44
- g4s8 Главы 24, 28, 34, 41, 42 и 72
- gaara87 Главы 4 и 39
- Gabriele Mariotti Главы 1, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 19, 26, 37, 40, 47, 50, 61, 66, 76, 78, 83, 88, 92, 100, 112 и 113
- Gaket Глава 1
- Gal Yedidovich Глава 37
- Gaurav Jindal Глава 8
- gbansal Главы 40 и 69
- Geert Глава 97
- GensaGames Глава 16
- Greg T Главы 40, 67, 70 и 96
- Guilherme Torres Castro Глава 31

- Guillaume Imbert Глава 39
Guillermo García Глава 37
gus27 Глава 110
H. Pauwelyn Глава 109
h22 Глава 48
Hamed Momeni Глава 104
hankide Главы 1 и 47
Harish Gyanani Главы 1, 5, 28, 38, 41 и 96
Harsh Pandey Глава 76
Harsh Sharma Глава 66
Hasif Seyd Глава 71
HDehghani Глава 34
hello_world Глава 84
herrmartell Глава 98
Hi Fm Frogatto Главы 41, 42 и 108
Hiren Patel Главы 2, 4, 25, 34, 42, 53, 81, 82, 112 и 113
honk Главы 4, 11, 18, 33, 37, 38, 44, 57, 58, 63, 74, 75, 86, 87, 91, 98, 97, 107, 111, 113, 114 и 117
Hussein El Feky Глава 99
Ichigo Kurosaki Главы 37, 62 и 121
Ichthyocentaur Главы 1, 47, 72 и 92
Ilya Krol Глава 61
Iman Hamidi Глава 91
Imdad Глава 77
IncredApp Глава 40
inetphantom Глава 1
insomniac Глава 41
Inzimam Tariq IT Глава 2
iravul Глава 22
Irfan Raza Глава 41
Ironman Главы 16, 41 и 96
Ishan Fernando Глава 79
Ishita Sinha Главы 37, 60 и 83
Iulian Popescu Глава 39
Ivan Wooll Глава 41
j2ko Глава 24
Jacob Глава 16
James_Parsons Глава 43
Jaseem Abbas Глава 57
Jason Bourne Глава 120
Jason Robinson Глава 83
jasonlam604 Глава 24
Jaymes Bearden Глава 62
Jean Vitor Глава 41
Jeeter Глава 1
jgm Главы 16, 42 и 47
Jinesh Francis Глава 76
JJ86 Глава 62
jlynch630 Глава 37
Joel Gritter Глава 28
John Snow Глава 46
johnrao07 Главы 34 и 85
Jon Adams Главы 2, 5, 28, 46, 56 и 98
JonasCz Главы 24, 40 и 45
Joost Verbraeken Глава 49
Jordan Главы 40 и 115
Jordi Castilla Глава 43
Joscandreu Глава 72
Joshua Глава 16
k3b Глава 47
kalan Глава 31
kann Глава 41
Karan Nagpal Глава 41
Karan Razdan Глава 58
KATHYxx Глава 96
Kayvan N Главы 16, 40, 41, 57 и 66
KDeogharkar Главы 38 и 74
Kedar Tendolkar Глава 15
Kevin DiTraglia Глава 16
kId Главы 14 и 96
Kiran Benny Joseph Глава 1
Kirill Kulakov Глава 72
kit Глава 113
Kling Klang Глава 38
krunal patel Глава 64
KuroObi Глава 86
Lazy Ninja Глава 72
Leo.Han Глава 72
Lewis McGeary Главы 8 и 61
Long Ranger Глава 39
LordSidious Главы 40 и 78
Lucas Paolillo Главы 45 и 61
Lukas Главы 78 и 115
M D P Глава 62
Madhukar Hebbar Глава 120
Mahmoud Ibrahim Глава 82
Marco Marchiori Главы 40 и 84

- Mark Ormesher Глава 41
Mark Yisri Глава 1
marshmallow Глава 76
Matas Vaitkevicius Глава 1
MathaN Главы 1, 16, 37 и 41
mattfred Глава 107
Mauker Глава 41
Max Главы 37, 41, 61 и 97
Max McKinney Глава 40
mayojava Глава 46
Medusalix Глава 72
Menasheh Главы 38 и 46
mhenryk Глава 61
Michael Allan Глава 1
Michael Spitsin Главы 43 и 45
Michele Глава 4
MidasLefko Главы 19 и 82
MiguelHincapieC Глава 65
Mike Глава 88
Mike Scamell Главы 71 и 76
Milad Nouri Главы 92 и 96
miss C Глава 76
mklimek Глава 16
mmBs Главы 37 и 97
Mochamad Taufik Hidayat Главы 62 и 76
mpkuth Главы 31, 37, 78 и 112
Mr.7 Главы 7 и 8
MrSalmon Глава 78
mrtuovinen Глава 96
mshukla Глава 47
Muhammad Umair Shafique Глава 85
Muhammad Younas Глава 20
Muhammed Refaat Главы 41, 43 и 71
Mukesh Kumar Swami Глава 98
Murali Глава 58
Muthukrishnan Rajendran Главы 17 и 98
Myon Глава 56
N Глава 103
NJ Главы 2, 41 и 47
Nambi Глава 55
Narayan Acharya Глава 42
NashHorn Глава 96
Nepster Глава 8
nibarius Глава 115
Nick Глава 1
Nick Cardoso Главы 38, 41, 43 и 95
Nickan В Глава 50
Nikita Kurtin Глава 37
niknetniko Глава 41
noob Глава 118
noongiya95 Глава 37
Nougat Lover Главы 14, 45, 97 и 113
null pointer Глава 108
Olu Глава 57
Omar Al Halabi Глава 96
once2go Глава 92
Onik Глава 59
Onur Глава 31
orelzion Глава 73
oshurmamadov Главы 37 и 92
Pablo Baxter Главы 16, 35, 82 и 98
Pankaj Kumar Глава 26
Paresh Mayani Глава 97
Parsania Hardik Глава 42
Patrick Dattilio Главы 8 и 76
Paul Lammertsma Глава 41
Pavel Durov Глава 56
Pavel Strelchenko Глава 47
Pavneet_Singh Главы 1, 41 и 47
Pawel Cala Глава 40
Peter Taylor Глава 115
Phan Van Linh Главы 2 и 8
Phil Глава 72
PhilLab Глава 90
Pinaki Acharya Глава 96
piotrek1543 Глава 69
Piyush Главы 76 и 83
Pongpat Глава 88
Prakash Bala Глава 24
pRaNaY Главы 92 и 120
Pratik Butani Главы 9, 21 и 26
Priyank Patel Главы 6 и 26
Pro Mode Главы 1 и 99
PSN Глава 1
R. Zagórski Главы 2, 38, 47, 56, 65, 73, 83, 84 и 88
rajan ks Глава 40
Rajesh Главы 8, 37, 41, 62, 72 и 93
Ramzy Hassan Глава 61

- Rasoul Miri Глава 76
 Ravi Rupareliya Главы 39 и 62
 rciovati Главы 41 и 47
 Reaz Murshed Главы 16, 41, 47 и 56
 RediOne1 Главы 13, 41, 53, 62 и 80
 Redman Глава 84
 rekire Главы 8, 15, 41, 44 и 47
 Revanth Gopi Глава 47
 Ricardo Vieira Глава 43
 ridsatrio Главы 37 и 97
 Robert Глава 117
 Rohan Arora Глава 96
 Rohit Arya Главы 40, 61, 78 и 118
 Rosário Pereira Fernandes Глава 76
 Rubin Nellikunnathu Главы 44 и 98
 Rupali Глава 69
 russjr08 Глава 41
 russt Глава 1
 S.D. Глава 90
 S.R Главы 14 и 68
 Saeed Глава 7
 Sagar Chavada Главы 16 и 37
 Sam Judd Глава 61
 sameera lakshitha Глава 98
 Sammy T Глава 78
 SANAT Глава 15
 Sanket Berde Главы 16 и 92
 Sanoop Главы 37 и 112
 Sasank Sunkavalli Глава 16
 Sashabraya Глава 22
 saul Глава 1
 saurav Глава 78
 Segun Famisa Глава 39
 Sevle Глава 15
 Shantanu Paul Глава 74
 Shinil M S Главы 40 и 92
 Shirane85 Глава 88
 ShivBuuya Главы 40 и 62
 shtolik Главы 24 и 112
 Siddharth Venu Глава 1
 Simon Главы 41 и 111
 Simon Schubert Глава 113
 Simone Carletti Глава 74
 Simplans Главы 1 и 41
 SimplyProgrammer Глава 37
 Sir SC Главы 8 и 97
 Smit.Satodia Глава 18
 Sneh Pandya Главы 1, 9, 14, 16, 37, 61 и 96
 Sohail Zahid Глава 4
 SoroushA Глава 41
 spaceplane Глава 41
 Stanojkovic Глава 113
 Stephane Mathis Глава 41
 Steve.P Глава 33
 still_learning Глава 59
 stkent Главы 29
 sud007 Главы 14, 15, 25, 26 и 37
 Sujith Niraikulathan Главы 2, 4, 28 и 113
 sukumar Глава 61
 Sup Глава 21
 Suragch Глава 28
 Suresh Kumar Глава 67
 Sweeper Глава 71
 Täg Глава 59
 tainy Глава 41
 Talha Mir Глава 8
 TameHog Глава 121
 Tanis.7x Главы 38, 39 и 119
 TDG Глава 111
 theFunkyEngineer Глава 41
 thetonrifles Глава 16
 thiagolr Глава 59
 Thomas Easo Глава 64
 ThomasThiebaud Главы 2, 8, 38, 41, 44 и 47
 Tim Kranen Глава 8
 Tot Zam Глава 16
 tpk Глава 98
 TR4Android Главы 28, 69 и 70
 Tudor Luca Глава 30
 tynn Главы 103 и 109
 ubuntudroid Глава 8
 Ufkoku Глава 92
 Uriel Carrillo Глава 81
 Uttam Panchasara Главы 50 и 87
 V Глава 56
 V. Kalyuzhnyu Глава 16
 Vicky Глава 78
 Vinícius Barros Глава 45

- Vinay Глава 41
vipsy Глава 40
Vishal Puri Глава 36
VISHWANATH N P Глава 98
Vishwesh Jainkuniya Глава 116
Vivek Mishra Главы 1, 5, 38 и 41
Vlonjat Gashi Главы 39, 47 и 83
Volodymyr Buberenko Главы 40 и 97
vrbsm Глава 92
Vucko Главы 78 и 118
W0rmH0le Глава 72
WarrenFaith Глава 14
webo80 Главы 43 и 54
weston Глава 69
Willie Chalmers III Главы 12 и 37
Xaver Kapeller Глава 37
Xavier Глава 38
Yasin Kaçmaz Главы 17 и 53
Yassie Глава 47
yennsarах Глава 39
Yojimbo Глава 62
younes zeboudj Глава 41
yuku Глава 47
Yury Fedorov Главы 1, 8, 9, 14, 28, 40, 41 и 47
Yvette Colomb Главы 57 и 101
Zeeshan Shabbir Глава 34
Zerntrino Глава 105
ZeroOne Главы 13 и 61
Zilk Глава 16
Zoe Главы 1, 3, 62, 94 и 102

Серия «Программирование от экспертов»

*Справочное издание
Анықтамалық басылым*

ANDROID. ПОЛНОЕ РУКОВОДСТВО ПО РАЗРАБОТКЕ ПРИЛОЖЕНИЙ ОТ СООБЩЕСТВА STACK OVERFLOW

Оформление обложки С. А. Арутюнян
Ответственный за выпуск И. В. Резько

Подписано в печать 26.01.2024 г.

Изготовлено в 2024 г.

Формат 70x100¹/₁₆. Бумага офсетная. Печать офсетная. Гарнитура Noto Serif SemiCondensed.
Усл. печ. л. 51,6. Тираж 2000 экз. Заказ № 673.

Общероссийский классификатор продукции ОК-034-2014 (КПЕС 2008);
58.11.1 — книги, брошюры печатные

Произведено в Российской Федерации

Изготовитель: ООО «Издательство АСТ»

129085, Российская Федерация, г. Москва, Звездный бульвар, дом 21,
строение 1, комната 705, пом. I, этаж 7.

Адрес места осуществления деятельности по изготовлению продукции:
123112, Российская Федерация, г. Москва, Пресненская набережная, д. 6, стр. 2,
Деловой комплекс «Империя», 14, 15 этаж.

Наш электронный адрес: ask@ast.ru

Наш сайт: www.ast.ru Интернет-магазин: www.book24.ru

Өндіруші: «Издательство АСТ» ЖШҚ

129085, Ресей Федерациясы, Звёздный бульвары, 21-үй, 1-құрылыс, 705-бөлме, I үй-жай, 7-қабат.
Өнім өндіру қызметін жүзеге асыру мекенжайы:

123112, Ресей Федерациясы, Мәскеу, Пресненская жағ., 6-үй, 2-құр.,
«Империя» іскерлік кешені, 14, 15-қабат

Біздің электрондық мекенжайымыз: www.ast.ru E-mail: ask@ast.ru

Интернет-магазин: www.book24.kz

Интернет-дүкен: www.book24.kz

Импортер в Республику Казахстан и Представитель по приему претензий
в Республике Казахстан — ТОО РДЦ Алматы, г. Алматы.

Қазақстан Республикасына импорттаушы және Қазақстан Республикасында
наразылықтарды қабылдау бойынша өкіл — «РДЦ-Алматы» ЖШС,

Алматы қ., Домбровский көш., 3«а», Б литері офис 1.

Тел.: 8(727) 2 51 59 90,91, факс: 8 (727) 251 59 92 ішкі 107;

E-mail: RDC-Almaty@eksмо.kz, www.book24.kz

Таяар белгісі: «АСТ» Өндірілген жылы: 2024

Өнімнің жарамдылық мерзімі шектелмеген

Ресей Федерациясында өндірілген

Отпечатано с готовых файлов заказчика

в АО «Первая Образцовая типография»,

филиал «УЛЬЯНОВСКИЙ ДОМ ПЕЧАТИ»

432980, Россия, г. Ульяновск, ул. Гончарова, 14

Современные мобильные устройства стали неотъемлемой частью нашей жизни, и, как следствие, программное обеспечение их работы превратилось в одну из самых актуальных и востребованных сфер деятельности для программистов. Большинство смартфонов по всему миру работают на операционной системе Android, на сегодняшний день удерживающей примерно 72% рынка мобильных операционных систем. Это неудивительно, учитывая широкий спектр устройств, работающих под ее управлением, и неоспоримые преимущества, в первую очередь открытость системы, означающую, что разработчики могут вносить свои изменения и настраивать операционную систему под свои нужды, создавая инновационные приложения на базе Android.

В данном руководстве по интегрированной среде разработки Android Studio для работы с платформой Android читатели найдут мощный инструментарий для профессионального создания интерфейсов, обработки событий, взаимодействия с активностями, службами, провайдерами контента и широковещательными приемниками, примеры использования API распознавания отпечатка пальца, определения местоположения и активности пользователя, узнают о возможностях стандартных и сторонних библиотек, а также получат советы по сборке собственных приложений от специалистов и экспертов сообщества Stack Overflow, куда входят самые авторитетные профессиональные разработчики программного обеспечения со всего мира.