

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

Обратите внимание, что важно проверять наличие разрешений в любой активности, требующей разрешений, так как разрешения могут быть отозваны во время работы приложения в фоновом режиме, что приведет к его аварийному завершению.

```
final private int REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS = 124;
```

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

if (Build.VERSION.SDK_INT >= 23) {
    checkMultiplePermissions();
}
```

Нам нужно запросить разрешение только для одной из этих групп, а все остальные разрешения от этой группы будут предоставлены, если только пользователь не отменит их:

```
private void checkMultiplePermissions() {

    if (Build.VERSION.SDK_INT >= 23) {
        List<String> permissionsNeeded = new ArrayList<String>();
        List<String> permissionsList = new ArrayList<String>();

        if (!addPermission(permissionsList, android.Manifest.permission.ACCESS_FINE_LOCATION)) {
            permissionsNeeded.add("GPS");
        }

        if (!addPermission(permissionsList, android.Manifest.permission.READ_EXTERNAL_STORAGE)) {
            permissionsNeeded.add("Read Storage");
        }

        if (permissionsList.size() > 0) {
            requestPermissions(permissionsList.toArray(new String[permissionsList.size()]), REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS);
            return;
        }
    }
}

private boolean addPermission(List<String> permissionsList, String permission) {
    if (Build.VERSION.SDK_INT >= 23) {
        if (checkSelfPermission(permission) != PackageManager.PERMISSION_GRANTED) {
            permissionsList.add(permission);

            // Проверка варианта обоснования
            if (!shouldShowRequestPermissionRationale(permission))
                return false;
        }
    }
    return true;
}
```

Здесь рассматривается, даны ли пользователем разрешения. Если разрешения не даны, приложение будет закрыто:

```

@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {
    switch (requestCode) {
        case REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS: {
            Map<String, Integer> perms = new HashMap<String, Integer>();
            // Initial
            perms.put(android.Manifest.permission.ACCESS_FINE_LOCATION, PackageManager.
PERMISSION_GRANTED);
            perms.put(android.Manifest.permission.READ_EXTERNAL_STORAGE,
PackageManager.PERMISSION_GRANTED);

            // Заполнить результатами
            for (int i = 0; i < permissions.length; i++) perms.put(permissions[i],
grantResults[i]);
            if (perms.get(android.Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED
                && perms.get(android.Manifest.permission.READ_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED) {
                // Все разрешения даны
                return;
            } else {
                // В разрешениях отказано
                if (Build.VERSION.SDK_INT >= 23) {
                    Toast.makeText(
                        getApplicationContext(),
                        "Приложение не может работать без Location and Storage " +
                        "Permissions.\nПерезапустите или дайте разрешение" +
                        " в настройках приложения",
                        Toast.LENGTH_LONG).show();
                    finish();
                }
            }
        }
        break;
    default:
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}

```

Дополнительная информация: <https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en>.

## 57.3. Использование PermissionUtil

PermissionUtil (см. <https://github.com/kayvannj/PermissionUtil>) – это простой и удобный способ запроса разрешений в контексте. Вы можете легко указать, что должно произойти в случае, если все запрошенные разрешения были предоставлены (onAllGranted()), любой запрос был отклонен (onAnyDenied()) или в случае, если требуется рациональное разрешение (onRational()).

В любом месте вашего AppCompatActivity или фрагмента, где вы хотите запросить разрешение пользователя:

```
mRequestObject = PermissionUtil.with(this).request(Manifest.permission.WRITE_
EXTERNAL_STORAGE).onAllGranted(
```

```

new Func() {
    @Override protected void call() {
        // С разрешениями
    }
}).onAnyDenied(
new Func() {
    @Override protected void call() {
        // Без разрешений
    }
}).ask(REQUEST_CODE_STORAGE);
}

```

И добавьте это в onRequestPermissionsResult:

```

if(mRequestObject!=null){
    mRequestObject.onRequestPermissionsResult(requestCode, permissions,
    grantResults);
}

```

Добавьте требуемое разрешение также в файл AndroidManifest.xml:

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

## 57.4. Включение всего кода, связанного с разрешениями, в абстрактный базовый класс

Включим весь код, связанный с разрешениями, в абстрактный базовый класс и расширим активность этого базового класса для достижения более чистого и полезного кода:

```

public abstract class BaseActivity extends AppCompatActivity {
    private Map<Integer, PermissionCallback> permissionCallbackMap = new
    HashMap<>();

    @Override
    protected void onStart() {
        super.onStart();
        ...
    }

    public void onRequestPermissionsResult(int requestCode) {
        super.onRequestPermissionsResult(requestCode);
        // Сделать что-нибудь
    }

    @Override
    public void setContentView(int layoutResId) {
        super.setContentView(layoutResId);
        bindViews();
    }

    @Override
    public void onRequestPermissionsResult(
        int requestCode, @NonNull String[] permissions, @NonNull int[]
        grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        PermissionCallback callback = permissionCallbackMap.get(requestCode);

        if (callback == null) return;

        // Проверяем, был ли отклонен запрос на разрешение.
        if (grantResults.length < 0 && permissions.length > 0) {
            callback.onPermissionDenied(permissions);
            return;
        }
    }
}

```

```

List<String> grantedPermissions = new ArrayList<>();
List<String> blockedPermissions = new ArrayList<>();
List<String> deniedPermissions = new ArrayList<>();
int index = 0;

for (String permission : permissions) {
    List<String> permissionList = grantResults[index] == PackageManager.
    PERMISSION_GRANTED
        ? grantedPermissions
        : ! ActivityCompat.shouldShowRequestPermissionRationale(this,
            permission)
        ? blockedPermissions
        : deniedPermissions;
    permissionList.add(permission);
    index++;
}

if (grantedPermissions.size() > 0) {
    callback.onPermissionGranted(
        grantedPermissions.toArray(new String[grantedPermissions.size()]));
}
if (deniedPermissions.size() > 0) {
    callback.onPermissionDenied(
        deniedPermissions.toArray(new String[deniedPermissions.size()]));
}
if (blockedPermissions.size() > 0) {
    callback.onPermissionBlocked(
        blockedPermissions.toArray(new String[blockedPermissions.size()]));
}

permissionCallbackMap.remove(requestCode);
}

/**
 * Проверка, выдано ли разрешение.
 *
 * @param permission разрешение
 * @return
 */
public boolean hasPermission(String permission) {
    return ContextCompat.checkSelfPermission(this, permission) == PackageManager.
    PERMISSION_GRANTED;
}

/**
 * Запрос разрешения и получение результата в обратном вызове.
 *
 * @param permissions
 * @param callback
 */
public void requestPermission(String [] permissions, @NotNull PermissionCallback
callback) {
    int requestCode = permissionCallbackMap.size() + 1;
    permissionCallbackMap.put(requestCode, callback);
    ActivityCompat.requestPermissions(this, permissions, requestCode);
}

```

```
/**  
 * Запросить разрешение и получить результат в обратном вызове.  
 *  
 * @param разрешение  
 * @param callback  
 */  
public void requestPermission(String permission, @NonNull PermissionCallback  
callback) {  
    int requestCode = permissionCallbackMap.size() + 1;  
    permissionCallbackMap.put(requestCode, callback);  
    ActivityCompat.requestPermissions(this, new String[] { permission },  
        requestCode);  
}  
  
}  
  
Пример использования в активности  
Активность должна расширять абстрактный базовый класс, определенный выше, следующим образом:  
  
private void requestLocationAfterPermissionCheck(){  
    if (hasPermission(Manifest.permission.ACCESS_FINE_LOCATION)) {  
        requestLocation();  
        return;  
    }  
  
    // Вызов метода базового класса  
    requestPermission(Manifest.permission.ACCESS_FINE_LOCATION, new  
    PermissionCallback() {  
        @Override  
        public void onPermissionGranted(String[] grantedPermissions) {  
            requestLocation();  
        }  
  
        @Override  
        public void onPermissionDenied(String[] deniedPermissions) {  
            // Сделать что-нибудь  
        }  
  
        @Override  
        public void onPermissionBlocked(String[] blockedPermissions) {  
            // Сделать что-нибудь  
        }  
    });  
}
```

## 57.5. Обеспечение прав доступа в широковещательных передачах и URI

При отправке намерения зарегистрированному широковещательному приемнику можно выполнить проверку разрешений. Отправляемые разрешения перепроверяются с разрешениями, зарегистрированными для данного тега. Они ограничивают круг отправителей широковещательных сообщений связанному приемнику.

Для отправки широковещательного запроса с разрешениями укажите разрешение в виде строки в вызове `Context.sendBroadcast(Intent intent, String permission)`, но не забывайте, что приложение получателя **ДОЛЖНО** иметь это разрешение, чтобы получить вашу широковещательную передачу. Приемник должен быть установлен первым перед отправителем.

Сигнатура метода имеет вид:

```
void sendBroadcast (Intent intent, String receiverPermission)
```

```
//для примера отправки широковещательной передачи на приемник Bcastreceiver  
Intent broadcast = new Intent(this, Bcastreceiver.class);  
sendBroadcast(broadcast, "org.quadcore.mypermission");
```

...и вы можете указать в манифесте, что отправитель широковещательной рассылки обязан включить запрашиваемое разрешение, передаваемое через sendBroadcast:

```
<!-- Ваше специальное разрешение -->  
<permission android:name="org.quadcore.mypermission"  
    android:label="my_permission"  
    android:protectionLevel="dangerous"></permission>
```

Также объягите разрешение в манифесте приложения, которое должно получать эту трансляцию:

```
<!-- Используем разрешение ! -->  
<uses-permission android:name="org.quadcore.mypermission"/>  
<!-- вместе с приемником -->  
<receiver android:name="Bcastreceiver" android:exported="true"/>
```

**Примечание:** разрешение может потребоваться как приемнику, так и передатчику, и в этом случае для доставки намерения к соответствующей цели должны пройти обе проверки. Приложение, определяющее разрешение, должно быть установлено первым.

Полная документация о разрешениях: <https://developer.android.com/guide/topics/security/permissions.html>.

## Глава 58. API Places

### 58.1. Получение текущего местоположения с помощью API Places

С помощью Google Places API (<https://developers.google.com/android/reference/com/google/android/gms/location/places/Places>) можно получить информацию о текущем местоположении и локальных местах пользователя.

Для получения информации о местном бизнесе или других местах необходимо сначала вызвать метод PlaceDetectionApi.getCurrentPlace(). Этот метод возвращает объект PlaceLikelihoodBuffer, содержащий список объектов PlaceLikelihood. Затем можно получить объект Place (см. <https://developers.google.com/android/reference/com/google/android/gms/location/places/Place>), вызвав метод PlaceLikelihood.getPlace() (см. [https://developers.google.com/android/reference/com/google/android/gms/location/places/PlaceLikelihood.html#getPlace\(\)](https://developers.google.com/android/reference/com/google/android/gms/location/places/PlaceLikelihood.html#getPlace())).

**Важно:** для того чтобы приложение могло получать доступ к точной информации о местоположении, необходимо запросить и получить разрешение ACCESS\_FINE\_LOCATION:

```
private static final int PERMISSION_REQUEST_TO_ACCESS_LOCATION = 1;

private TextView txtLocation;
private GoogleApiClient googleApiClient;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_location);

txtLocation = (TextView) this.findViewById(R.id.txtLocation);
googleApiClient = new GoogleApiClient.Builder(this)
    .addApi(Places.GEO_DATA_API)
    .addApi(Places.PLACE_DETECTION_API)
    .enableAutoManage(this, this)
    .build();

getCurrentLocation();
}

private void getCurrentLocation() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        Log.e(LOG_TAG, "Разрешение не дано");

        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISSION_REQUEST_TO_ACCESS_LOCATION);
        return;
    }

    Log.i(LOG_TAG, "Разрешение выдано");

    PendingResult<PlaceLikelihoodBuffer> result = Places.PlaceDetectionApi.
getPlace(googleApiClient, null);
    result.setResultCallback(new ResultCallback<PlaceLikelihoodBuffer>() {
        @Override
        public void onResult(PlaceLikelihoodBuffer likelyPlaces) {
            Log.i(LOG_TAG, String.format("Result received : %d ", likelyPlaces.
                getCount()));
            StringBuilder stringBuilder = new StringBuilder();

            for (PlaceLikelihood placeLikelihood : likelyPlaces) {
                stringBuilder.append(String.format("Place : '%s' %n",
                    placeLikelihood.getPlace().getName()));
            }
            likelyPlaces.release();
            txtLocation.setText(stringBuilder.toString());
        }
    });
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults) {
    switch (requestCode) {
        case PERMISSION_REQUEST_TO_ACCESS_LOCATION: {
            // Если запрос отменен, то массивы результатов будут пусты.
            if (grantResults.length > 0 && grantResults[0] == PackageManager.
                PERMISSION_GRANTED) {
                getCurrentLocation();
            } else {
                // Разрешения не даны
                // Отключить функциональность, зависящую от данного разрешения.
            }
        }
    }
    return;
}
```

```

    // Добавьте дополнительные строки 'case' для проверки других разрешений,
    которые может запросить данное приложение.
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    Log.e(LOG_TAG, "Не удалось установить соединение с GoogleApiClient: " +
        connectionResult.getErrorMessage());
}

```

## 58.2. Интеграция автозаполнения мест

Функция автозаполнения в Google Places API для Android предоставляет пользователю подсказки мест. Когда пользователь набирает текст в строке поиска, автозаполнение показывает места в соответствии с его запросами.

```

AutoCompleteActivity.java

private TextView txtSelectedPlaceName;

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

    txtSelectedPlaceName = (TextView) this.findViewById(R.id.txtSelectedPlaceName);

    PlaceAutocompleteFragment autocompleteFragment =
        (PlaceAutocompleteFragment) getFragmentManager().findFragmentById(R.id.fragment_autocomplete);

    autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(Place place) {
            Log.i(LOG_TAG, "Place: " + place.getName());
            txtSelectedPlaceName.setText(String.format("Выбранные места : %s - %s",
                place.getName(), place.getAddress()));
        }
    });

    @Override
    public void onError(Status status) {
        Log.i(LOG_TAG, "Произошла ошибка: " + status);
        Toast.makeText.AutoCompleteActivity.this, "Место не может быть
        выбрано!!!", Toast.LENGTH_SHORT).show();
    }
});

```

activity\_autocomplete.xml

```

<fragment
    android:id="@+id/fragment_autocomplete"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:name="com.google.android.gms.location.places.ui.PlaceAutocompleteFragment"
/>

```

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/txtSelectedPlaceName"
    android:layout_margin="20dp"
    android:padding="15dp"
    android:hint="@string/txt_select_place_hint"
    android:textSize="@dimen/place_autocomplete_prediction_primary_text"/>
    android:id="@+id/txtSelectedPlaceName" android:layout_margin="20dp"
    android:padding="15dp" android:hint="@string/txt_select_place_hint"
    android:textSize="@dimen/place_autocomplete_prediction_primary_text"/>
```

### 58.3. Пример использования Place Picker

Place Picker – это очень простой виджет пользовательского интерфейса, предоставляемый API Places. Он предоставляет встроенную карту, текущее местоположение, близлежащие места, возможности поиска и автозаполнения.

Приведем пример использования UI-виджета Place Picker:

```
private static int PLACE_PICKER_REQUEST = 1;

private TextView txtPlaceName;

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

    txtPlaceName = (TextView) this.findViewById(R.id.txtPlaceName);
    Button btnSelectPlace = (Button) this.findViewById(R.id.btnSelectPlace);
    btnSelectPlace.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            openPlacePickerView();
        }
    });
}

private void openPlacePickerView(){
    PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();
    try {
        startActivityForResult(builder.build(this), PLACE_PICKER_REQUEST);
    } catch (GooglePlayServicesRepairableException e) {
        e.printStackTrace();
    } catch (GooglePlayServicesNotAvailableException e) {
        e.printStackTrace();
    }
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_PICKER_REQUEST) {
        if (resultCode == RESULT_OK) {
            Place place = PlacePicker.getPlace(this, data);
            Log.i(LOG_TAG, String.format("Place Name : %s", place.getName()));
            Log.i(LOG_TAG, String.format("Place Address : %s", place.getAddress()));
            Log.i(LOG_TAG, String.format("Place Id : %s", place.getId()));
        }
    }
}
```

```
        txtPlaceName.setText(String.format("Place : %s - %s" , place.getName() ,  
place.getAddress()));  
    }  
}  
}
```

#### 58.4. Настройка фильтров типов мест для PlaceAutocomplete

В некоторых случаях у нас может появиться стремление сузить результаты, показываемые автозаполнителем PlaceAutocomplete, до определенной страны или, возможно, показать только определенные регионы. Этого можно добиться, установив фильтр AutocompleteFilter для намерения. Например, если поставлена задача искать только места типа REGION и только принадлежащие Индии, нужно сделать следующее:

## MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    private static final int PLACE_AUTOCOMPLETE_REQUEST_CODE = 1;
    private TextView selectedPlace;

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

        selectedPlace = (TextView) findViewById(R.id.selected_place);
        try {
            AutocompleteFilter typeFilter = new AutocompleteFilter.Builder()
                .setTypeFilter(AutocompleteFilter.TYPE_FILTER_REGIONS)
                .setCountry("IN")
                .build();

            Intent intent =
                new PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_FULLSCREEN)
                    .setFilter(typeFilter)
                    .build(this);
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_REQUEST_CODE);
        } catch (GooglePlayServicesRepairableException
                 | GooglePlayServicesNotAvailableException e) {
            e.printStackTrace();
        }
    }

    protected void onActivityResult(int requestCode,
                                    int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == PLACE_AUTOCOMPLETE_REQUEST_CODE && resultCode == Activity.RESULT_OK) {
            final Place place = PlacePicker.getPlace(this, data);
            selectedPlace.setText(place.getName().toString().toUpperCase());
        } else {
            Toast.makeText(MainActivity.this, "Не удалось получить местоположение.", Toast.LENGTH_SHORT).show();
        }
    }
}
```

**activity\_main.xml**

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/selected_place"/>

</LinearLayout>
```

Автоматически запустится PlaceAutocomplete, и вы сможете выбрать из результатов место, которое будет иметь тип REGION и принадлежать только указанной стране. Запуск на мерения также может быть осуществлен нажатием одной кнопки.

## 58.5. Добавление более одной активности Google Autocomplete

```
public static final int PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE=1;
public static final int PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE=2;

fromPlaceEdit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            // Выполняем действия _FROM_PLACE_
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_FROM_
PLACE_REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException e) {
            // TODO: Обработка ошибки.
        } catch (GooglePlayServicesNotAvailableException e) {
            // TODO: Обработка ошибки.
        }
    }
});
toPlaceEdit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            // Выполняем действия _TO_PLACE_
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_TO_PLACE_
REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException e) {
            // TODO: Обработка ошибки.
        } catch (GooglePlayServicesNotAvailableException e) {
            // TODO: Обработка ошибки.
        }
    }
});
```

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // ok, действия _FROM_PLACE_
        } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
            // Обработка ошибки
        } else if (resultCode == RESULT_CANCELED) {
            // Пользователь отменил операцию.
        }
    } else if (requestCode == PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // ok, действия _TO_PLACE_
        } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
            // Обработка ошибки
        } else if (resultCode == RESULT_CANCELED) {
            // Пользователь отменил операцию.
        }
    }
}

```

## Глава 59. Android NDK

### 59.1. Вход в систему NDK

Сначала убедитесь, что в файле `Android.mk` установлена ссылка на библиотеку протоколирования:

```
LOCAL_LDLIBS := -llog
```

Затем используйте один из следующих вызовов `android_log_print()`:

```
#include <android/log.h>
#define TAG "MY LOG"

__android_log_print(ANDROID_LOG_VERBOSE, TAG, "Значение 1 + 1 = %d", 1 + 1)
__android_log_print(ANDROID_LOG_WARN, TAG, "Значение 1 + 1 = %d", 1 + 1)
__android_log_print(ANDROID_LOG_DEBUG, TAG, "Значение 1 + 1 = %d", 1 + 1)
__android_log_print(ANDROID_LOG_INFO, TAG, "Значение 1 + 1 = %d", 1 + 1)
__android_log_print(ANDROID_LOG_ERROR, TAG, "Значение 1 + 1 = %d", 1 + 1)
```

...или используйте их в более удобном виде, определив соответствующие макросы:

```
#define LOGV(...) __android_log_print(ANDROID_LOG_VERBOSE, TAG, VA_ARGS)
#define LOGW(...) __android_log_print(ANDROID_LOG_WARN, TAG, VA_ARGS)
#define LOGD(...) __android_log_print(ANDROID_LOG_DEBUG, TAG, VA_ARGS)
#define LOGI(...) __android_log_print(ANDROID_LOG_INFO, TAG, VA_ARGS)
#define LOGE(...) __android_log_print(ANDROID_LOG_ERROR, TAG, VA_ARGS)
```

Пример:

```
int x = 42;
LOGD("The value of x is %d", x);
```

## 59.2. Создание нативных исполняемых файлов (native executables) для Android

project/jni/main.c:

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
    printf("Hello world!\n");
    return 0;
}
```

project/jni/Android.mk:

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE      := hello_world
LOCAL_SRC_FILES   := main.c
include $(BUILD_EXECUTABLE)
```

project/jni/Application.mk:

```
APP_ABI := all
APP_PLATFORM := android-21
```

## 59.3. Как очистить сборку

Если необходимо очистить сборку, выполните:

ndk-build clean

## 59.4. Как использовать makefile, отличающийся от Android.mk

ndk-build NDK\_PROJECT\_PATH=PROJECT\_PATH APP\_BUILD\_SCRIPT=MyAndroid.mk

## 61.2. Добавление Glide в проект

# Глава 60. Тема DayNight

Тема DayNight дает приложению возможность переключать цветовые схемы в зависимости от времени суток и последнего известного местоположения устройства.

Добавьте в файл styles.xml следующее:

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
    <!-- Настроить тему можно здесь. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Ниже перечислены темы, которые можно расширять, чтобы добавить возможность переключения тем "день-ночь":

- "Theme.AppCompat.DayNight"

- "Theme.AppCompat.DayNight.NoActionBar"
- "Theme.AppCompat.DayNight.DarkActionBar"

Помимо `colorPrimary`, `colorPrimaryDark` и `colorAccent`, можно добавить любые другие цвета, которые необходимо переключать, например, `textColorPrimary` или `textColorSecondary`. В этот стиль можно добавить и пользовательские цвета приложения.

Для того чтобы переключение тем работало, необходимо определить стандартный файл `colors.xml` в каталоге `res/values` и еще один `colors.xml` в каталоге `res/values-night` и соответствующим образом определить дневные/ночные цвета.

Чтобы переключить тему, вызовите из своего Java-кода метод `AppCompatActivity.setDefaultNightMode(int)`. (При этом цветовая схема будет изменена для всего приложения, а не только для какого-либо одного действия или фрагмента). Например:

```
AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT_NO);
```

Вы можете выбрать одну из трех возможностей:

- `AppCompatActivity.MODE_NIGHT_NO`: задает тему по умолчанию для вашего приложения и принимает цвета, определенные в каталоге `res/values`. Для этой темы рекомендуется использовать светлые цвета.
- `AppCompatActivity.MODE_NIGHT_YES`: устанавливает ночную тему для вашего приложения и принимает цвета, определенные в каталоге `res/values-night`. Для этой темы рекомендуется использовать темные цвета.
- `AppCompatActivity.MODE_NIGHT_AUTO`: это автоматическое переключение цветов приложения в зависимости от времени суток и цветов, заданных в директориях `values` и `values-night`.

Также можно получить текущее состояние ночного режима с помощью метода `getDefaultNightMode()`. Например:

```
int modeType = AppCompatActivity.getDefaultNightMode();
```

Однако следует иметь в виду, что переключение темы не сохранится, если закрыть приложение и открыть его снова. В этом случае тема снова переключится на `AppCompatActivity.MODE_NIGHT_AUTO`, которая является значением по умолчанию. Если вы хотите, чтобы переключение темы сохранилось, обязательно сохраните значение в общих настройках и загружайте его при каждом открытии приложения.

## Глава 61. Библиотека Glide

Официальная документация Glide:  
Для Glide v4 см. <http://bumptech.github.io/glide/>.

### 61.1. Загрузка изображения

#### ImageView

Загрузка изображения из указанного URL, Uri, идентификатора ресурса или любой другой модели в `ImageView`:

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";
Glide.with(context)
```

```
.load(yourUrl)
.into(imageView);
```

Для Uri: замените yourUrl на свой Uri (content://media/external/images/1). Для drawable-ресурсов замените yourUrl на идентификатор ресурса (R.drawable.image).

### RecyclerView и ListView

В ListView или RecyclerView можно использовать точно такие же строки:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    Glide.with(context)
        .load(currentUrl)
        .into(myViewHolder.imageView);
}
```

Если вы не хотите запускать загрузку в onBindViewHolder, то перед изменением ImageView убедитесь, что вы очистили при помощи clear() все ImageView, которыми может управлять Glide:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    if (TextUtils.isEmpty(currentUrl)) {
        Glide.clear(viewHolder.imageView);
        // Теперь, когда представление очищено, можно устанавливать собственный ресурс
        viewHolder.imageView.setImageResource(R.drawable.missing_image);
    } else {
        Glide.with(context)
            .load(currentUrl)
            .into(myViewHolder.imageView);
    }
}
```

## 61.2. Добавление Glide в проект

С помощью Gradle:

```
repositories {
    mavenCentral() // jcenter() тоже работает, поскольку берется из Maven Central
}

dependencies {
    compile 'com.github.bumptech.glide:glide:4.0.0'
    compile 'com.android.support:support-v4:25.3.1'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.0.0'
}
```

С помощью Maven:

```
<dependency>
<groupId>com.github.bumptech.glide</groupId>
<artifactId>glide</artifactId>
<version>4.0.0</version>
```

```

</dependency>
<dependency>
    <groupId>com.google.android</groupId>
    <artifactId>support-v4</artifactId>
    <version>r7</version>
</dependency>
<dependency>
    <groupId>com.github.bumptech.glide</groupId>
    <artifactId>compiler</artifactId>
    <version>4.0.0</version>
    <optional>true</optional>
</dependency>

```

В зависимости от конфигурации и особенностей использования ProGuard (DexGuard), возможно, потребуется включить в файл proguard.cfg следующие строки (см. дополнительную информацию в <https://github.com/bumptech/glide/wiki/Configuration#keeping-a-glidemodule>):

```

-keep public class * implements com.bumptech.glide.module.GlideModule
-keep public class * extends com.bumptech.glide.AppGlideModule
-keep public enum com.bumptech.glide.load.resource.bitmap.ImageHeaderParser$** {
    **[] $VALUES;
    public *;
}

# for DexGuard only
-keepresourceelements manifest/application/meta-data@value=GlideModule

```

### 61.3. Преобразование круга в Glide (загрузка изображения в круговой ImageView)

Создадим круглое изображение с помощью Glide:

```

public class CircleTransform extends BitmapTransformation {

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

    @Override protected Bitmap transform(BitmapPool pool, Bitmap toTransform, int
outWidth, int outHeight) {
        return circleCrop(pool, toTransform);
    }

    private static Bitmap circleCrop(BitmapPool pool, Bitmap source) {
        if (source == null) return null;

        int size = Math.min(source.getWidth(), source.getHeight());
        int x = (source.getWidth() - size) / 2;
        int y = (source.getHeight() - size) / 2;

        Bitmap squared = Bitmap.createBitmap(source, x, y, size, size);

        Bitmap result = pool.get(size, size, Bitmap.Config.ARGB_8888);
        if (result == null) {
            result = Bitmap.createBitmap(size, size, Bitmap.Config.ARGB_8888);
        }

        Canvas canvas = new Canvas(result);
        Paint paint = new Paint();

```

## 61.4. Преобразования по умолчанию

```

paint.setShader(new BitmapShader(squared, BitmapShader.TileMode.CLAMP,
BitmapShader.TileMode.CLAMP));
paint.setAntiAlias(true);
float r = size / 2f;
canvas.drawCircle(r, r, r, paint);
return result;
}

@Override public String getId() {
    return getClass().getName();
}
}

```

### Использование:

```
Glide.with(context)
.load(yourimageUrl)
.transform(new CircleTransform(context))
.into(userImageView);
```

## 61.4. Преобразования по умолчанию

Glide включает в себя два стандартных преобразования – fit center и center crop.

### Fit center:

```
Glide.with(context)
.load(yourUrl)
.fitCenter()
.into(yourView);
```

Fit center выполняет то же преобразование, что и ScaleType.FIT\_CENTER в Android.

### Center crop:

```
Glide.with(context)
.load(yourUrl)
.centerCrop()
.into(yourView);
```

Эта обрезка выполняет такое же преобразование, как и ScaleType.CENTER\_CROP в Android. Дополнительную информацию можно найти в <https://github.com/bumptech/glide/wiki/Transformations#default-transformations>.

## 61.5. Изображение с закругленными углами с помощью пользовательской Glide Target

Сначала сделайте класс утилиты или используйте этот метод в необходимом классе:

```

public class UIUtils {
public static BitmapImageViewTarget getRoundedImageTarget(@NonNull final Context
context, @NonNull
final ImageView imageView, final float radius) {
    return new BitmapImageViewTarget(imageView) {
        @Override
        protected void setResource (final Bitmap resource) {
            RoundedBitmapDrawable circularBitmapDrawable =
                RoundedBitmapDrawableFactory.create(context,
getResources(), resource);

```

```
        circularBitmapDrawable.setCornerRadius(radius);
        imageView.setImageDrawable(circularBitmapDrawable);
    }
};
```

## Загрузка изображения:

```
Glide.with(context)
    .load(imageUrl)
    .asBitmap()
    .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Поскольку вы используете функцию `asBitmap()`, анимация будет удалена. На этом месте можно использовать собственную анимацию, используя метод `animate()`.

Пример с аналогичным плавным переходом к стандартной анимации Glide.

```
Glide.with(context)
    .load(imageUrl)
    .asBitmap()
    .animate(R.anim.abc_fade_in)
    .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Обратите внимание, что данная анимация является частным ресурсом библиотеки поддержки – использовать ее не рекомендуется, так как она может измениться или даже быть удалена. Для использования RoundedBitmapDrawableFactory необходимо также иметь библиотеку поддержки.

## 61.6. Обработка заполнителей и ошибок

Если необходимо добавить отрисовываемый (drawable) объект, который будет отображаться во время загрузки, можно добавить заполнитель (placeholder):

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
    .into(imageView);
```

Если вы хотите, чтобы при неудачной загрузке по какой-либо причине отображался drawable-объект, используйте:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

Если вы хотите, чтобы drawable-объект отображался при предоставлении нулевой модели (URL, Uri, путь к файлу и т. д.), то используйте:

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

## 61.7. Предварительная загрузка изображений

Для предварительной загрузки удаленных изображений и обеспечения однократной загрузки изображения:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Затем:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // Здесь тоже работает все
    .into(imageView);
```

Для предварительной загрузки локальных изображения и обеспечения наличия преобразованной копии в дисковом кэше (и, возможно, в кэше памяти):

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Или любое другое преобразование по вашему желанию
    .preload(200, 200); // Или любую ширину и высоту по вашему желанию
```

Затем:

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Необходимо использовать то же преобразование, что и выше
    .override(200, 200) // Вы должны использовать те же ширину и высоту, что и выше
    .into(imageView);
```

## 61.8. Обработка ошибок при загрузке изображений Glide

```
Glide
    .with(context)
    .load(currentUrl)
    .into(new BitmapImageViewTarget(profilePicture) {
        @Override
        protected void setResource(Bitmap resource) {
            RoundedBitmapDrawable circularBitmapDrawable = RoundedBitmapDrawableFactory.create(context.getResources(), resource);
            circularBitmapDrawable.setCornerRadius(radius);
            imageView.setImageDrawable(circularBitmapDrawable);
        }
        @Override
        public void onLoadFailed(@NonNull Exception e, Drawable errorDrawable) {
            super.onLoadFailed(e, SET_YOUR_DEFAULT_IMAGE);
            Log.e(TAG, e.getMessage(), e);
        }
    });
});
```

На месте SET\_YOUR\_DEFAULT\_IMAGE можно установить любой drawable-ресурс по умолчанию (будет показано в случае неудачной загрузки изображения).

## 61.9. Загрузка изображения в круговой ImageView без пользовательских преобразований

Создайте пользовательскую цель BitmapImageViewTarget для загрузки изображения:

```
public class CircularBitmapImageViewTarget extends BitmapImageViewTarget
```

```
{
    private Context context;
    private ImageView imageView;

    public CircularBitmapImageViewTarget(Context context, ImageView imageView)
    {
        super(imageView);
        this.context = context;
        this.imageView = imageView;
    }

    @Override
    protected void setResource(Bitmap resource)
    {
        RoundedBitmapDrawable bitmapDrawable = RoundedBitmapDrawableFactory.
            create(context.getResources(), resource);
        bitmapDrawable.setCircular(true);
        imageView.setImageDrawable(bitmapDrawable);
    }
}
```

Использование:

```
Glide
    .with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));
```

## Глава 62. Диалог

`show();`

Показывает диалог

`setContentView(R.layout.yourlayout);` Устанавливает ContentView диалога на ваш

пользовательский макет

`dismiss()`

Закрывает диалог

### 62.1. Добавление Material Design AlertDialog в приложение с помощью Appcompat

`AlertDialog` – это подкласс `Dialog`, который может отображать одну, две или три кнопки. Если вы хотите отображать в этом диалоговом окне только строку, используйте метод `setMessage()`.

`AlertDialog` из пакета `android.app` отображается по-разному на разных версиях ОС Android. Библиотека Android V7 Appcompat предоставляет реализацию `AlertDialog`, которая будет отображаться с использованием материального оформления (Material Design) на всех поддерживаемых версиях ОС Android, как показано ниже:

Are you sure?

You'll lose all photos and media!

CANCEL ERASE

Сначала необходимо добавить библиотеку V7 Appcompat в проект. Это можно сделать в файле build.gradle на уровне приложения:

```
dependencies {  
    compile 'com.android.support:appcompat-v7:24.2.1'  
    //.....  
}
```

Убедитесь, что импортирован правильный класс:

```
import android.support.v7.app.AlertDialog;
```

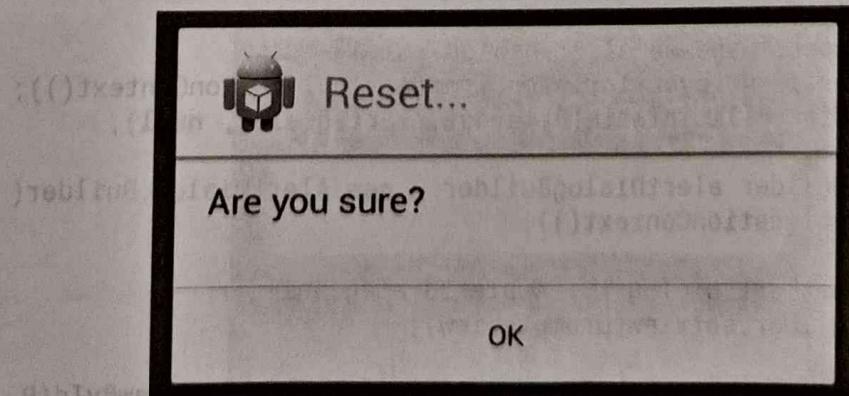
Затем создайте AlertDialog следующим образом:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
builder.setTitle("Вы уверены?");  
builder.setMessage("Вы потеряете все фотографии и медиафайлы!");  
builder.setPositiveButton("ERASE", null);  
builder.setNegativeButton("CANCEL", null);  
builder.show();
```

## 62.2. Базовый диалог оповещения

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);  
//Установить название  
builder.setTitle("Reset...")  
//Установить сообщение  
.setMessage("Вы уверены?")  
//Установка значка диалога  
.setIcon(drawable)  
//Установить кнопку положительного ответа, в данном случае OK, которая  
завершит диалог и выполнит все действия в методе onClick  
.setPositiveButton(android.R.string.ok, new DialogInterface.  
OnClickListener() {  
    @Override  
    public void onClick(DialogInterface dialogInterface, int i) {  
        // Сброс  
    }  
});  
AlertDialog dialog = builder.create();  
// Теперь к нему можно обратиться в любой момент:  
dialog.show();  
//Для того чтобы можно было показать диалог.
```

Теперь код позволит добиться этого:



(Источник изображения: *WikiHow*)

## 62.3. ListView в диалоге оповещения

Мы всегда можем использовать ListView или RecyclerView для выбора из списка элементов, но если у нас есть небольшое количество вариантов и мы хотим, чтобы среди них пользователь выбрал один, мы можем использовать AlertDialog.Builder.setAdapter:

```
private void showDialog()
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Выберите любой элемент");

    final List<String> lables = new ArrayList<>();
    lables.add("элемент 1");
    lables.add("элемент 2");
    lables.add("элемент 3");
    lables.add("элемент 4");

    ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, lables);
    builder.setAdapter(dataAdapter, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(MainActivity.this, "Вы выбрали " +
lables.get(which), Toast.LENGTH_LONG).show();
        }
    });
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

Возможно, если нам не нужен какой-то конкретный ListView, то можно воспользоваться базовым способом:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Выберите элемент")
.setItems(R.array.your_array, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        // Аргумент 'which' содержит индексную позицию выбранного элемента
        Log.v(TAG, "Выбранный элемент на позиции " + which);
    }
});
builder.create().show();
```

## 62.4. Пользовательский диалог оповещения с EditText

```
void alertDialogDemo() {
    // получить представление alert_dialog.xml
    LayoutInflator li = LayoutInflator.from(getApplicationContext());
    View promptsView = li.inflate(R.layout.alert_dialog, null);

    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
        getApplicationContext());

    // установить alert_dialog.xml в alertDialog builder
    alertDialogBuilder.setView(promptsView);

    final EditText userInput = (EditText) promptsView.findViewById(R.
id.etUserInput);
```

```

// установить диалоговое сообщение
AlertDialogBuilder
    .setCancelable(false)
    .setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            // получить пользовательский ввод и установить его
            // в результат
            // редактировать текст
            Toast.makeText(getApplicationContext(), "Введено: "
                +userInput.getText().toString(), Toast.LENGTH_LONG).show();
        }
    })
    .setNegativeButton("Cancel",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }
        });
}

// создать диалог оповещения
AlertDialog alertDialog = alertDialogBuilder.create();

// показать его
alertDialog.show();
}

```

Xml-файл: res/layout/alert\_dialog.xml

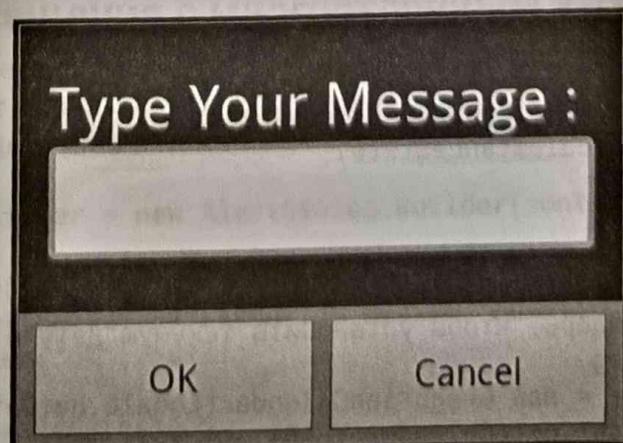
```

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Введите ваше сообщение : "
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/etUserInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <requestFocus />

</EditText>

```



## 62.5. DatePickerDialog

`DatePickerDialog` – это самый простой способ использования `DatePicker`, поскольку диалог можно отображать в любом месте приложения. С виджетом `DatePicker` вам не нужно реализовывать собственный макет.

Как показать диалог:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year,
month, day);
datePickerDialog.show();
```

Вы можете получить виджет `DatePicker` из диалога выше, чтобы получить доступ к большему количеству функций и, например, установить минимальную дату в миллисекундах:

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

## 62.6. DatePicker

`DatePicker` позволяет пользователю выбирать дату. При создании нового экземпляра `DatePicker` мы можем установить начальную дату. Если мы не зададим ее, то по умолчанию будет установлена текущая дата.

Мы можем показать пользователю `DatePicker`, используя `DatePickerDialog` или создав собственный макет с виджетом `DatePicker`.

Также мы можем ограничить диапазон дат, которые может выбрать пользователь.

Задавая минимальную дату в миллисекундах

```
// В данном случае пользователь может выбрать дату только из будущего
```

```
datePicker.setMinDate(System.currentTimeMillis());
```

Задавая максимальную дату в миллисекундах

```
// В данном случае пользователь может выбрать только дату, предшествующую следующей неделе.
```

```
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

Для получения информации о том, какая дата была выбрана пользователем, нам необходимо использовать слушатель (`Listener`).

Если мы используем `DatePickerDialog`, то можно установить `OnDateSetListener` в конструкторе при создании нового экземпляра `DatePickerDialog`:

### Пример использования DatePickerDialog

```
public class SampleActivity extends AppCompatActivity implements DatePickerDialog.OnDateSetListener {

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

    private void showDatePicker() {
        //Нам нужен календарь, чтобы установить текущую дату в качестве начальной в DatePickerDialog.
        Calendar calendar = new GregorianCalendar(Locale.getDefault());
        int year = calendar.get(Calendar.YEAR);
```

```

    int month = calendar.get(Calendar.MONTH);
    int day = calendar.get(Calendar.DAY_OF_MONTH);

    DatePickerDialog datePickerDialog = new DatePickerDialog(this, this, year,
        month, day); datePickerDialog.show();
}

@Override
public void onDateSet(DatePicker datePicker, int year, int month, int day) {
}
}

```

В противном случае, если мы создаем собственный макет с виджетом `DatePicker`, нам также необходимо создать собственный слушатель, как это было показано в другом примере.

## 62.7. Диалог оповещения

```

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder( MainActivity.
this);

alertDialogBuilder.setTitle("Заголовок диалога");
alertDialogBuilder
    .setMessage("Сообщения диалога")
    .setCancelable(true)
    .setPositiveButton("Да",
        new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int arg1) {
                // Обработка кнопки положительного ответа
            }
        })
    .setNegativeButton("Нет",
        new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int arg1) {
                // Обработка кнопки отрицательного ответа
                dialog.cancel();
            }
        });
}

AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show();

```

## 62.8. Диалог оповещения с многострочным заголовком

Метод `setCustomTitle()` в `AlertDialog.Builder` позволяет указать произвольное представление, которое будет использоваться для заголовка диалога. Обычно этот метод используется для создания диалога оповещения с длинным заголовком.

```

AlertDialog.Builder builder = new AlertDialog.Builder(context, Theme_Material_
Light_Dialog);
builder.setCustomTitle(inflater.inflate(context, R.layout.my_dialog_title, null))
    .setView(inflater.inflate(context, R.layout.my_dialog, null))
    .setPositiveButton("OK", null);

Dialog dialog = builder.create();
dialog.show();

```

**my\_dialog\_title.xml:**

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

    <TextView
        style="@android:style/TextAppearance.Small"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Curabitur tincidunt condimentum tristique. Vestibulum ante ante, pretium
        porttitor iaculis vitae, congue ut sem. Curabitur ac feugiat ligula. Nulla
        tincidunt est eu sapien iaculis rhoncus. Mauris eu risus sed justo pharetra
        semper faucibus vel velit."
        android:textStyle="bold"/>

</LinearLayout>
```

**my\_dialog.xml:**

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:scrollbars="vertical">

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world!"/>

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!"/>

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!"/>

        <TextView
            style="@android:style/TextAppearance.Small"
```

```

    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="10dp"
    android:text="Hello world again! "/>

</LinearLayout>
<ScrollView>

```

**62.9. Выбор даты в диалоговом фрагменте**

**Лекция 62**

**Задача 1.** Создать диалоговый фрагмент, позволяющий выбирать дату.

**Решение.** Для этого в XML-файле диалога нужно определить элемент `<DatePicker>`. Внутри него можно указать различные атрибуты, определяющие внешний вид и поведение этого элемента. В частности, можно указать, что пользователь не может видеть календарь, а видимость календаря определяется атрибутом `calendarViewShown`.

Hello world!

Hello world again!

Hello world again!

Hello world again!

OK

## 62.9. Выбор даты в диалоговом фрагменте

xml диалога:

```

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

    <DatePicker
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:id="@+id/datePicker" android:layout_gravity="center_horizontal"
        android:calendarViewShown="false"/>

    <Button
        android:layout_width="match_parent" android:layout_height="wrap_content"
        android:text="ACCEPT" android:id="@+id/buttonAccept" />

</LinearLayout>

```

Диалоговый класс:

```

public class ChooseDate extends DialogFragment implements View.OnClickListener {
    private DatePicker datePicker;

```

```
private Button acceptButton;

private boolean isDateSetted = false; private int year;
private int month;
private int day;

private DateListener listener;

public interface DateListener {
    onDateSelected(int year, int month, int day);
}

public ChooseDate() {}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.dialog_year_picker, container);

    getDialog().setTitle(getResources().getString("TITLE"));

    datePicker = (DatePicker) rootView.findViewById(R.id.datePicker);
    acceptButton = (Button) rootView.findViewById(R.id.buttonAccept);
    acceptButton.setOnClickListener(this);

    if (isDateSetted) {
        datePicker.updateDate(year, month, day);
    }

    return rootView;
}

@Override
public void onClick(View v) {
    switch(v.getId()){
        case R.id.buttonAccept:
            int year = datePicker.getYear();
            int month = datePicker.getMonth() + 1; // месяцы начинаются с 0
            int day = datePicker.getDayOfMonth();

            listener.onDateSelected(year, month, day);
            break;
    }
    this.dismiss();
}

@Override
public void onAttach(Context context) {
    super.onAttach(context);
    listener = (DateListener) context;
}

public void setDate(int year, int month, int day) {

    this.year = year;
    this.month = month;
    this.day = day;
    this.isDateSetted = true;
}
```

Активность, вызывающая диалог:

```
public class MainActivity extends AppCompatActivity implements ChooseDate.
DateListener{

    private int year;
    private int month;
    private int day;

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

        private void showAlertDialog();
    }

    private void showAlertDialog(){
        ChooseDate pickDialog = new ChooseDate();
        // Мы можем установить дату
        // pickDialog.setDate(23, 10, 2016);
        pickDialog.show(getFragmentManager(), "");
    }

    @Override
    onDateSelected(int year, int month, int day){
        this.day = day;
        this.month = month;
        this.year = year;
    }
}
```



## 62.10. Полнозаданный пользовательский диалог без фона и заголовка

Добавьте свой собственный стиль в styles.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.
Fullscreen">
    </style>
</resources>
```

Создайте свой собственный макет диалога: fullscreen.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

Затем в java-файле вы можете использовать его для Activity, Dialog и т.д.

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

public class FullscreenActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
//Вы можете не задавать никакого содержания для активности.
Dialog mDialog = new Dialog(this, R.style.AppBaseTheme);
mDialog.setContentView(R.layout.fullscreen);
mDialog.show();
}
```

## Глава 63. Расширение возможностей диалоговых окон оповещения

Эта тема посвящена расширению AlertDialog дополнительными возможностями.

### 63.1. Диалог оповещения, содержащий ссылку, по которой можно перейти

Для отображения диалога оповещения, содержащего ссылку, которая открывается по щелчку на ней, можно использовать следующий код:

```
AlertDialog.Builder builder1 = new AlertDialog.Builder(youractivity.this);

builder1.setMessage(Html.fromHtml("ваше сообщение,<a href=\"http://www.google.com\">link</a>"));

builder1.setCancelable(false);
builder1.setPositiveButton("ok", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
    }
});

AlertDialog Alert1 = builder1.create();
Alert1 .show();
((TextView)Alert1.findViewById(android.R.id.message)).setMovementMethod(LinkMovementMethod.getInstance());
```

## Глава 64. Анимированное диалоговое окно AlertDialog

Анимированный диалог оповещения может отображаться с некоторыми анимационными эффектами. Вы можете получить некоторые анимации для диалогового окна, такие как Fadein, Slideleft, Slidetop, SlideBottom, Slideright, Fall, Newsgaper, Fliph, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill, чтобы сделать ваше приложение более привлекательным.

## 64.1. Код для анимированного диалога

animated\_android\_dialog\_box.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#1184be"
        android:onClick="animatedDialog1"
        android:text="Animated Fall Dialog"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="#1184be"
        android:onClick="animatedDialog2"
        android:text="Animated Material Flip Dialog"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#1184be"
        android:onClick="animatedDialog3"
        android:text="Animated Material Shake Dialog"
        android:textColor="#fff" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="#1184be"
        android:onClick="animatedDialog4"
        android:text="Animated Slide Top Dialog"
        android:textColor="#fff" />
```

AnimatedAndroidDialogExample.java:

```
public class AnimatedAndroidDialogExample extends AppCompatActivity {

    NiftyDialogBuilder materialDesignAnimatedDialog;

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

        materialDesignAnimatedDialog = NiftyDialogBuilder.getInstance(this);
    }
```

```

public void animatedDialog1(View view) {
    materialDesignAnimatedDialog
        .withTitle("Заголовок диалогового окна анимированного падения")
        .withMessage("Добавьте сюда ваше диалоговое сообщение. Место описания
анимированного диалога.")
        .withDialogColor("#FFFFFF")
        .withButton1Text("OK")
        .withButton2Text("Cancel")
        .withDuration(700)
        .withEffect(Effectstype.Fall)
        .show();
}

public void animatedDialog2(View view) {
    materialDesignAnimatedDialog
        .withTitle("Анимированный заголовок перелистываемого диалога")
        .withMessage("Добавьте сюда ваше диалоговое сообщение. Место описания
анимированного диалога.")
        .withDialogColor("#1c90ec")
        .withButton1Text("OK")
        .withButton2Text("Cancel")
        .withDuration(700)
        .withEffect(Effectstype.Flip)
        .show();
}

public void animatedDialog3(View view) {
    materialDesignAnimatedDialog
        .withTitle("Заголовок диалогового окна анимированного встряхивания")
        .withMessage("Добавьте сюда ваше диалоговое сообщение. Место описания
анимированного диалога.")
        .withDialogColor("#1c90ec")
        .withButton1Text("OK")
        .withButton2Text("Cancel")
        .withDuration(700)
        .withEffect(Effectstype.Shake)
        .show();
}

public void animatedDialog4(View view) {
    materialDesignAnimatedDialog
        .withTitle("Заголовок диалогового окна анимированного слайд-топа")
        .withMessage("Добавьте сюда ваше диалоговое сообщение. Место описания
анимированного диалога.")
        .withDialogColor("#1c90ec")
        .withButton1Text("OK")
        .withButton2Text("Cancel")
        .withDuration(700)
        .withEffect(Effectstype.Slidetop)
        .show();
}

```

Добавьте следующие строки в файл build.gradle для включения NifyBuilder(CustomView):

```

dependencies {
    compile 'com.nineoldandroids:library:2.4.0'
}

```

```
compile 'com.github.sd6352051.niftydialogeffects:niftydialogeffects:1.0.0@aar'
```

Дополнительная информация: <https://github.com/sd6352051/NiftyDialogEffects>.

## Глава 65. GreenDAO

GreenDAO – это библиотека объектно-реляционного отображения, помогающая разработчикам использовать базы данных SQLite для постоянного локального хранения данных.

### 65.1. Вспомогательные методы для запросов SELECT, INSERT, DELETE, UPDATE

В данном примере показан вспомогательный класс, содержащий методы, полезные при выполнении запросов к данным. Каждый метод здесь использует Java Generic, чтобы быть очень гибким:

```
public <T> List<T> selectElements(AbstractDao<T, ?> dao) {
    if (dao == null) {
        return null;
    }
    QueryBuilder<T> qb = dao.queryBuilder();
    return qb.list();
}

public <T> void insertElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.insertOrReplaceInTx(items);
}

public <T> T insertElement(AbstractDao<T, ?> absDao, T item) {
    if (item == null || absDao == null) {
        return null;
    }
    absDao.insertOrReplaceInTx(item);
    return item;
}

public <T> void updateElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.updateInTx(items);
}

public <T> T selectElementByCondition(AbstractDao<T, ?> absDao,
                                         WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
}
```

```

QueryBuilder<T> qb = absDao.queryBuilder();
for (WhereCondition condition : conditions) {
    qb = qb.where(condition);
}
List<T> items = qb.list();
return items != null && items.size() > 0 ? items.get(0) : null;
}

public <T> List<T> selectElementsByCondition(AbstractDao<T, ?> absDao,
                                              WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSort(AbstractDao<T, ?> absDao,
                                                      Property sortProperty,
                                                      String sortStrategy,
                                                      WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderCustom(sortProperty, sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSortWithNullHandling(AbstractDao<T,
?> absDao,
                                                       Property sortProperty,
                                                       boolean handleNulls,
                                                       String sortStrategy,
                                                       WhereCondition... conditions)
{
    if (!handleNulls) {
        return selectElementsByConditionAndSort(absDao, sortProperty, sortStrategy,
conditions);
    }
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderRaw("(CASE WHEN " + "T." + sortProperty.columnName + " IS NULL then 1
ELSE 0 END)," + "T." + sortProperty.columnName + " " + sortStrategy);
}

```

```
List<T> items = qb.list();
return items != null ? items : null;
}

public <T, V extends Class> List<T> selectByJoin(AbstractDao<T, ?> absDao,
                                                V className,
                                                Property property, WhereCondition
whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    return qb.list();
}

public <T> void deleteElementsByCondition(AbstractDao<T, ?> absDao,
                                            WhereCondition... conditions) {
    if (absDao == null) {
        return;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> list = qb.list();
    absDao.deleteInTx(list);
}

public <T> T deleteElement(DaoSession session, AbstractDao<T, ?> absDao, T object)
{
    if (absDao == null) {
        return null;
    }
    absDao.delete(object);
    session.clear();
    return object;
}

public <T, V extends Class> void deleteByJoin(AbstractDao<T, ?> absDao,
                                                V className,
                                                Property property, WhereCondition whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    qb.buildDelete().executeDeleteWithoutDetachingEntities();
}

public <T> void deleteAllFromTable(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return;
    }
    absDao.deleteAll();
}

public <T> long countElements(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return 0;
    }
    return absDao.count();
}
```

## 65.2. Создание сущности в GreenDAO 3.X с составным первичным ключом

При создании модели для таблицы, имеющей составной первичный ключ, требуется дополнительная работа над объектом, чтобы модель Entity соблюдала эти ограничения.

Следующий пример SQL-таблицы и Entity демонстрирует структуру для хранения отзыва, оставленного покупателем о товаре в интернет-магазине. В этом примере мы хотим, чтобы столбцы `customer_id` и `item_id` были составным первичным ключом, позволяющим иметь только один отзыв для конкретного покупателя и товара.

**Таблица SQL:**

```
CREATE TABLE review (
    customer_id STRING NOT NULL,
    item_id STRING NOT NULL,
    star_rating INTEGER NOT NULL,
    content STRING,
    PRIMARY KEY (customer_id, item_id)
);
```

Обычно мы используем аннотации `@Id` и `@Unique` над соответствующими полями в классе сущности Entity, однако для составного первичного ключа мы делаем следующее.

1. Добавим аннотацию `@Index` внутрь аннотации `@Entity` на уровне класса. Свойство `value` содержит разделенный запятыми список полей, составляющих ключ. Для обеспечения уникальности ключа используйте свойство `unique`.

2. GreenDAO требует, чтобы каждая сущность (Entity) имела в качестве первичного ключа `long` или объект `Long`. Нам по-прежнему необходимо добавить его в класс Entity, однако нам не нужно использовать его или беспокоиться о том, что он повлияет на нашу реализацию. В приведенном ниже примере он называется `localID`.

**Entity:**

```
@Entity(indexes = { @Index(value = "customer_id,item_id", unique = true) })
public class Review {

    @Id(autoincrement = true)
    private Long localID;

    private String customer_id;
    private String item_id;

    @NotNull
    private Integer star_rating;

    private String content;

    public Review() {}
}
```

## 65.3. Начало работы с GreenDao v3.X

После добавления зависимости от библиотеки GreenDao и плагина Gradle нам необходимо создать объект сущности (Entity).

**Entity**

Entity (сущность) – это объект Plain Old Java Object (POJO), который моделирует некоторые данные в базе данных. GreenDao использует этот класс для создания таблицы в базе данных

SQLite и автоматически генерирует вспомогательные классы, которые мы можем использовать для доступа и хранения данных без необходимости написания SQL-запросов.

```
@Entity
public class Users {
    @Id(autoincrement = true)
    private Long id;

    private String firstname;
    private String lastname;

    @Unique
    private String email;

    // Здесь геттеры и сеттеры для полей...
}
```

### Однократная настройка GreenDao

При каждом запуске приложения GreenDao необходимо инициализировать. GreenDao предлагает хранить этот код в классе Application или в каком-либо другом месте, где он будет выполняться только один раз.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase",
        null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

### Классы-помощники GreenDao

После создания объекта сущности GreenDao автоматически создает вспомогательные классы, используемые для взаимодействия с базой данных. Они называются аналогично имени созданного объекта сущности, после которого следует Dao, и извлекаются из объекта daoSession:

```
UsersDao usersDao = daoSession.getUsersDao();
```

Теперь многие типичные действия с базой данных могут быть выполнены с помощью этого Dao объекта с объектом Entity.

#### Запрос

```
String email = "jdoe@example.com";
String firstname = "John";

// Однопользовательский запрос, в котором email совпадает с "jdoe@example.com"
Users user = userDao.queryBuilder()
        .where(UsersDao.Properties.Email.eq(email)).build().unique();

// Запрос к нескольким пользователям, в котором firstname = "John"
List<Users> user = userDao.queryBuilder()
        .where(UsersDao.Properties.Firstname.eq(firstname)).build().list();
```

#### Вставка

```
Users newUser = new User("John", "Doe", "jdoe@example.com");
usersDao.insert(newUser);
```

## Обновление

```
// Модификация ранее полученного объекта пользователя и обновление
user.setLastname("Dole");
usersDao.update(user);
```

## Удаление

```
// Удаление ранее полученного объекта пользователя
usersDao.delete(user);
```

# Глава 66. Атрибуты инструментов

Атрибуты Designtime Layout Attributes используются при отрисовке макета в Android Studio, но не оказывают влияния на время выполнения. В общем случае можно использовать любой атрибут фреймворка Android, просто используя пространство имен `tools:`, а не `android:` namespace для предварительного просмотра макета. Вы можете добавить как атрибут `android:` namespace (который используется во время выполнения), так и соответствующий атрибут `tools:` (который переопределяет атрибут во время выполнения только в предварительном просмотре макета). Просто определите пространство имен инструментов, как описано в разделе замечаний. Пример для атрибута `text`:

```
<EditText
    tools:text="Мой текст"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

...или атрибут `visibility`, чтобы снять настройку вида для предварительного просмотра:

```
<LinearLayout
    android:id="@+id/l11"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:visibility="gone" />
```

...или атрибут `context`, чтобы связать макет с активностью или фрагментом:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity" >
```

...или атрибут `showIn` для отображения включенного предварительного просмотра макета в другом макете:

```
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/text"
    tools:showIn="@layout/activity_main" />
```

# Глава 67. Форматирование строк

## 67.1. Форматирование строкового ресурса

Вы можете добавлять подстановочные знаки в строковые ресурсы и заполнять их во время выполнения программы.

1. Редактирование файла strings.xml:

```
<string name="my_string">Это %1$s</string>
```

2. Форматирование строки по мере необходимости:

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

## 67.2. Форматирование типов данных в String и обратно

### Типы данных для форматирования

Такие типы данных, как int, float, double, long, boolean, могут быть приведены к строковому типу данных с помощью функции `String.valueOf()`:

```
String.valueOf(1); //Вывод -> "1"
String.valueOf(1.0); //Вывод -> "1.0"
String.valueOf(1.2345); //Вывод -> "1.2345"
String.valueOf(true); //Вывод -> "true"
```

Обратное действие, форматирование строки в другой тип данных:

```
Integer.parseInt("1"); //Вывод -> 1
Float.parseFloat("1.2"); //Вывод -> 1.2
Boolean.parseBoolean("true"); //Вывод -> true
```

## 67.3. Форматирование временной метки в строку

Полное описание шаблонов приведено в справочнике `SimpleDateFormat`: <https://developer.android.com/reference/java/text/SimpleDateFormat.html>.

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

# Глава 68. SpannableString

## 68.1. Добавление стилей в TextView

В следующем примере мы создаем `Activity` для отображения одного `TextView`.

В качестве содержимого этого `TextView` будет использоваться `SpannableString`, которая будет иллюстрировать некоторые из доступных стилей. Вот какие возможности планируется реализовать для текста:

- Сделать крупнее
- Полужирное начертание (Bold)

- Подчеркивание (Underline)
- Курсив (Italicize)
- Перечеркивание (Strike-through)
- Цветной (Colored)
- Выделенный цветом (Highlighted)
- Показать в виде надстрочного знака (Show as superscript)
- Показать в виде подстрочного знака (Show as subscript)
- Показать в виде ссылки (Show as a link)
- Сделать кликабельным (Make clickable)

```

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

    SpannableString styledString
        = new SpannableString("Large\n\n" // индекс 0 - 5
            + "Bold\n\n" // индекс 7 - 11
            + "Underlined\n\n" // индекс 13 - 23
            + "Italic\n\n" // индекс 25 - 31
            + "Strikethrough\n\n" // индекс 33 - 46
            + "Colored\n\n" // индекс 48 - 55
            + "Highlighted\n\n" // индекс 57 - 68
            + "K Superscript\n\n" // индекс "Superscript" 72 - 83
            + "K Subscript\n\n" // индекс "Subscript" 87 - 96
            + "Url\n\n" // индекс 98 - 101
            + "Clickable\n\n"); // индекс 103 - 112

    // сделать текст в два раза больше
    styledString.setSpan(new RelativeSizeSpan(2f), 0, 5, 0);

    // сделать текст жирным
    styledString.setSpan(new StyleSpan(Typeface.BOLD), 7, 11, 0);

    // подчеркнуть текст
    styledString.setSpan(new UnderlineSpan(), 13, 23, 0);

    // сделать текст курсивным
    styledString.setSpan(new StyleSpan(Typeface.ITALIC), 25, 31, 0);

    styledString.setSpan(new StrikethroughSpan(), 33, 46, 0);

    // изменить цвет текста
    styledString.setSpan(new ForegroundColorSpan(Color.GREEN), 48, 55, 0);

    // выделение текста
    styledString.setSpan(new BackgroundColorSpan(Color.CYAN), 57, 68, 0);

    // надстрочный индекс
    styledString.setSpan(new SuperscriptSpan(), 72, 83, 0);
    // сделать надстрочный текст меньше
    styledString.setSpan(new RelativeSizeSpan(0.5f), 72, 83, 0);

    // подстрочный индекс
    styledString.setSpan(new SubscriptSpan(), 87, 96, 0);
    // сделать подстрочный текст меньше
    styledString.setSpan(new RelativeSizeSpan(0.5f), 87, 96, 0);
}

```

```

// url
styledString.setSpan(new URLSpan("http://www.google.com"), 98, 101, 0);

// кликабельный текст
ClickableSpan clickableSpan = new ClickableSpan() {

    @Override
    public void onClick(View widget) {
        // Выводим на экран тост. Здесь можно сделать все что угодно.
        Toast.makeText(SpanExample.this, "Clicked", Toast.LENGTH_SHORT).show();
    }
};

styledString.setSpan(clickableSpan, 103, 112, 0);

// Передаем строку с примененными стилями в TextView
TextView textView = new TextView(this);

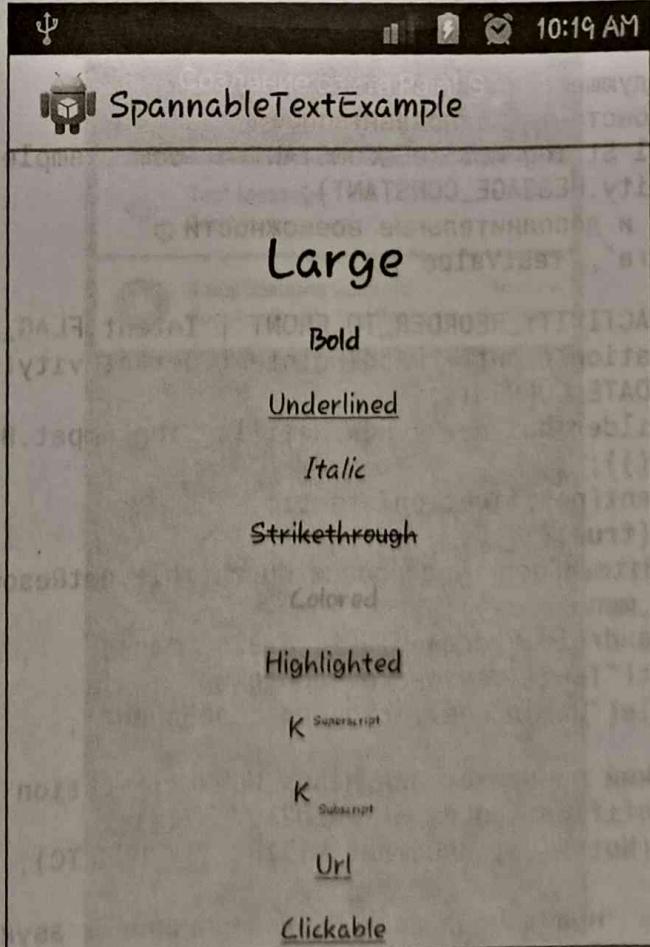
// Этот шаг обязателен для url и clickable.
textView.setMovementMethod(LinkMovementMethod.getInstance());

// придание аккуратности
textView.setGravity(Gravity.CENTER);
textView.setBackgroundColor(Color.WHITE);

textView.setText(styledString);
setContentView(textView);
}

```

Результат будет выглядеть следующим образом:



## 68.2. Много строк, использование разных цветов

Метод: `setSpanColor`

```
public Spanned setSpanColor(String string, int color){
    SpannableStringBuilder builder = new SpannableStringBuilder();
    SpannableString ss = new SpannableString(string);
    ss.setSpan(new ForegroundColorSpan(color), 0, string.length(), 0);
    builder.append(ss);
    return ss;
}
```

Использование:

```
String a = getString(R.string.string1);
String b = getString(R.string.string2);

Spanned color1 = setSpanColor(a, Color.CYAN);
Spanned color2 = setSpanColor(b, Color.RED);
Spanned mixedColor = TextUtils.concat(color1, "1", color2);
// Теперь мы используем `mixedColor`.
```

# Глава 69. Уведомления

## 69.1. Уведомление в режиме Heads Up

В главе описано, как сделать Heads Up Notification.

```
// При нажатии на уведомление откроется MainActivity
Intent i = new Intent(this, MainActivity.class);

// действие для последующего использования
// определяется как константа приложения:
// public static final String MESSAGE_CONSTANT = "com.example.myapp.notification";
i.setAction(MainActivity.MESSAGE_CONSTANT);
// можно использовать и дополнительные возможности
i.putExtra("some_extra", "testValue");

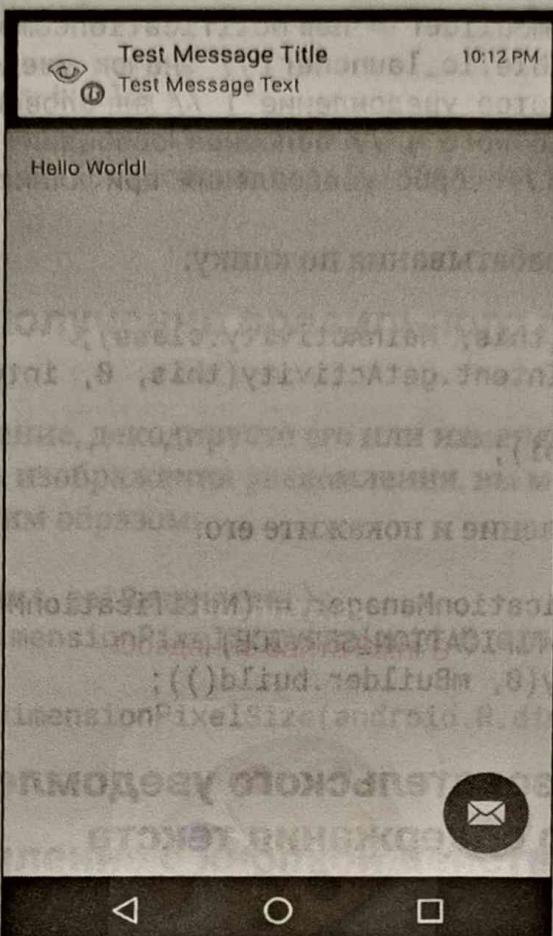
setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_SINGLE_TOP);
PendingIntent notificationIntent = PendingIntent.getActivity(this, 999, i,
PendingIntent.FLAG_UPDATE_CURRENT);
NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
getApplicationContext());
builder.setContentIntent(notificationIntent);
builder.setAutoCancel(true);
builder.setLargeIcon(BitmapFactory.decodeResource(this.getResources(),
android.R.drawable.ic_menu_view));
builder.setSmallIcon(android.R.drawable.ic_dialog_map);
builder.setContentText("Текст тестового сообщения");
builder.setContentTitle("Заголовок тестового сообщения");

// устанавливаем высокий приоритет для Heads Up Notification
builder.setPriority(NotificationCompat.PRIORITY_HIGH);
builder.setVisibility(NotificationCompat.VISIBILITY_PUBLIC);

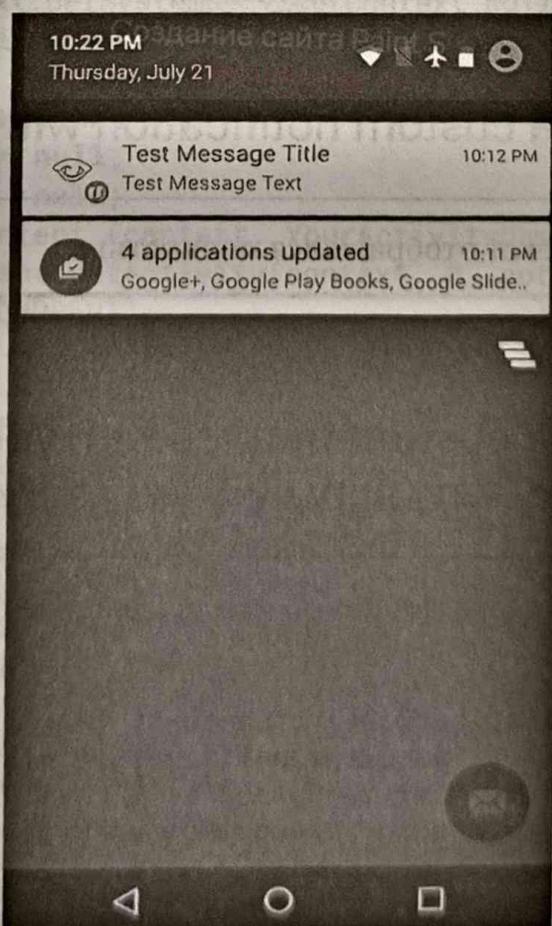
// Не будет показывать "Heads Up", если не воспроизводит звук
if (Build.VERSION.SDK_INT >= 21) builder.setVibrate(new long[0]);
```

```
NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);  
mNotificationManager.notify(999, builder.build());
```

Вот как это выглядит:



На всех версиях Android уведомление (notification) отображается в ящике уведомлений:



## 69.2. Создание простого уведомления

В этом примере показано, как создать простое уведомление, запускающее приложение при клике по нему пользователем.

Укажите содержание уведомления:

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
    .setSmallIcon(R.drawable.ic_launcher) // значок уведомления
    .setContentTitle("Простое уведомление") // заголовок
    .setContentText("Hello word") // основное сообщение
    .setAutoCancel(true); // сброс уведомления при клике по нему
```

Создайте намерение для срабатывания по клику:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

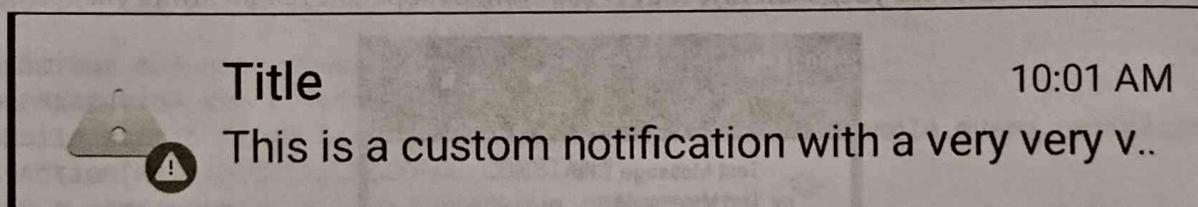
Наконец, создайте уведомление и покажите его:

```
NotificationManager mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

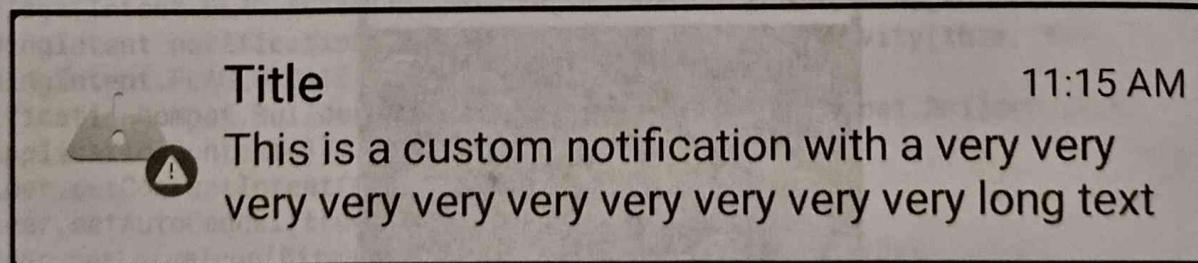
## 69.3. Установка пользовательского уведомления – отображение полного содержания текста

Если вы хотите, чтобы в контексте отображался длинный текст, необходимо задать пользовательское содержимое.

Например, у вас есть следующее:



Но вы хотите, чтобы ваш текст отображался полностью:



Все, что вам нужно сделать, это **добавить стиль** к вашему содержимому, как показано ниже:

```
private void generateNotification(Context context) {
    String message = "This is a custom notification with a very very very very very very very very very long text";
    Bitmap largeIcon = BitmapFactory.decodeResource(getResources(),
        android.R.drawable.ic_dialog_alert);

    NotificationCompat.Builder builder = new NotificationCompat.Builder(context);
```

```
builder.setContentTitle("Заголовок").setContentText(message)
    .setSmallIcon(android.R.drawable.ic_dialog_alert)
    .setLargeIcon(largeIcon)
    .setAutoCancel(true)
    .setWhen(System.currentTimeMillis())
    .setStyle(new NotificationCompat.BigTextStyle().bigText(message));

Notification = builder.build();
NotificationManagerCompat notificationManager = NotificationManagerCompat.
from(context); notificationManager.notify(101, notification);
}
```

## 69.4. Динамическое получение правильного пиксельного размера для большого значка

Если вы создаете изображение, декодируете его или изменяете его размер, чтобы оно поместилось в большую область изображения уведомления, вы можете получить правильные размеры в пикселях следующим образом:

```
Resources resources = context.getResources();
int width = resources.getDimensionPixelSize(android.R.dimen.notification_large_
icon_width);
int height = resources.getDimensionPixelSize(android.R.dimen.notification_large_
icon_height);
```

## 69.5. Текущее уведомление с кнопкой действия

```
// Отмена старого уведомление с тем же id
NotificationManager notificationMgr =
(NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationMgr.cancel(CALL_NOTIFY_ID); // любое постоянное значение

// Создание отложенного намерения
Intent notificationIntent = null;
PendingIntent contentIntent = null;
notificationIntent = new Intent(context, YourActivityName);
contentIntent = PendingIntent.getActivity(context, 0, notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

// Построитель уведомлений
builder = new NotificationCompat.Builder(context);
builder.setContentText("Текущее уведомление...");
builder.setContentTitle("образец текущего уведомления");
builder.setSmallIcon(R.drawable.notification_icon);
builder.setUsesChronometer(true);
builder.setDefaults(Notification.DEFAULT_LIGHTS);
builder.setContentIntent(contentIntent);
builder.setOngoing(true);

// Добавить кнопку действия в уведомление
Intent intent = new Intent("action.name");
PendingIntent pIntent = PendingIntent.getBroadcast(context, 1, intent, 0);
builder.addAction(R.drawable.action_button_icon, "Название кнопки
действия", pIntent);
```

```
// Уведомление с помощью notificationMgr
Notification finalNotification = builder.build();
notificationMgr.notify(CALL_NOTIFY_ID, finalNotification);
```

Зарегистрируйте широковещательный приемник (broadcast receiver) для того же действия, чтобы обрабатывать событие нажатия кнопки действия.

## 69.6. Установка различных приоритетов в уведомлении

```
NotificationCompat.Builder mBuilder =
    (NotificationCompat.Builder) new NotificationCompat.Builder(context)
        .setSmallIcon(R.drawable.some_small_icon)
        .setContentTitle("Заголовок")
        .setContentText("Это тестовое уведомление с MAX приоритетом")
        .setPriority(Notification.PRIORITY_MAX);
```

Если уведомление содержит изображение и необходимо автоматически разворачивать его при получении уведомления, используйте значение "PRIORITY\_MAX", другие уровни приоритета можно использовать в соответствии с требованиями.

Информация о различных уровнях приоритетности:

**PRIORITY\_MAX** – Используется для критических и срочных уведомлений, предупреждающих пользователя о состоянии, которое критично по времени или должно быть решено, прежде чем он сможет продолжить выполнение конкретной задачи.

**PRIORITY\_HIGH** – Используется в первую очередь для важных сообщений, например, сообщений или событий чата с особенно интересным для пользователя содержанием. Уведомления с высоким приоритетом запускают отображение уведомлений в верхней части экрана.

**PRIORITY\_DEFAULT** – Используется для всех уведомлений, которые не попадают ни в один из других описанных здесь приоритетов.

**PRIORITY\_LOW** – Используется для уведомлений, о которых вы хотите проинформировать пользователя, но которые не являются срочными. Уведомления с низким приоритетом обычно отображаются в нижней части списка, что делает их хорошим выбором для таких вещей, как открытые или ненаправленные социальные обновления: пользователь попросил уведомить его о них, но эти уведомления никогда не должны превалировать над срочными или прямыми сообщениями.

**PRIORITY\_MIN** – Используется для контекстной или фоновой информации, например, информации о погоде или местоположении. Уведомления с минимальным приоритетом не отображаются в строке состояния. Пользователь обнаруживает их, разворачивая тень уведомлений (notification shade).

Подробнее на эту тему см.: [http://developer.android.com/design/patterns/notifications.html#correctly\\_set\\_and\\_manage\\_notification\\_priority](http://developer.android.com/design/patterns/notifications.html#correctly_set_and_manage_notification_priority).

## 69.7. Установка пользовательского значка уведомления с помощью библиотеки Picasso

```
PendingIntent pendingIntent = PendingIntent.getActivity(context, uniqueIntentId,
    intent, PendingIntent.FLAG_CANCEL_CURRENT);

final RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
    R.layout.remote_view_notification);
remoteViews.setImageResource(R.id.remoteview_notification_icon, R.mipmap.
    ic_navigation_favorites);

Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_
    NOTIFICATION);
```

```

NotificationCompat.Builder notificationBuilder =
    new NotificationCompat.Builder(context)
        .setSmallIcon(R.mipmap.ic_navigation_favorites) //просто фиктивный
        .setContent(remoteViews) // здесь мы применяем наше представление
        .setAutoCancel(true)
        .setContentIntent(pendingIntent)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);

final Notification notification = notificationBuilder.build();

if (android.os.Build.VERSION.SDK_INT >= 16) {
    notification.bigContentView = remoteViews;
}

NotificationManager notificationManager =
    (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(uniqueIntentId, notification);

//не забудьте включить picasso в файл build.gradle
Picasso.with(context)
    .load(avatar)
    .into(remoteViews, R.id.remoteview_notification_icon, uniqueIntentId,
    notification);

```

Затем определите макет в папке layouts:

```

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

    <ImageView
        android:id="@+id/remoteview_notification_icon"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_marginRight="2dp"
        android:layout_weight="0"
        android:scaleType="centerCrop" />
</LinearLayout>

```

## 69.8. Планирование уведомлений

Иногда требуется вывести уведомление в определенное время, что, к сожалению, не является тривиальной задачей в системе Android, поскольку для уведомлений не существует метода `setTime()` или аналогичного ему. В данном примере описаны действия, необходимые для планирования уведомлений с помощью `AlarmManager`.

1. Добавьте приемник `BroadcastReceiver`, который прослушивает намерения, передаваемые менеджером Android `AlarmManager`.

Создайте свое уведомление на основе дополнительных данных, предоставленных намерением:

```

public class NotificationReceiver extends BroadcastReceiver {
    @Override

```

```

public void onReceive(Context context, Intent intent) {
    // Построение уведомления на основе намерения
    Notification notification = new NotificationCompat.Builder(context)
        .setSmallIcon(R.drawable.ic_notification_small_icon)
        .setContentTitle(intent.getStringExtra("title", ""))
        .setContentText(intent.getStringExtra("text", ""))
        .build();
    // Показать уведомление
    NotificationManager manager = (NotificationManager) context.
        getSystemService(Context.NOTIFICATION_SERVICE);
    manager.notify(42, notification);
}
}

```

**2. Зарегистрируйте приемник BroadcastReceiver в файле AndroidManifest.xml (в противном случае приемник не будет получать никаких намерений (Intents) из AlarmManager):**

```

<receiver
    android:name=".NotificationReceiver"
    android:enabled="true" />

```

**3. Запланируйте уведомление**, передав в AlarmManager системы отложенное намерение (PendingIntent) для вашего BroadcastReceiver с необходимыми расширениями Intent. По наступлении заданного времени ваш широковещательный приемник получит намерение и выведет уведомление на экран. Планирование оповещения осуществляется следующим методом:

```

public static void scheduleNotification(Context context, long time, String title,
String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);
    // Запланируем уведомление
    AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_
    SERVICE);
    manager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, time, pending);
}

```

Обратите внимание, что приведенное выше значение 42 должно быть уникальным для каждого запланированного уведомления, иначе PendingIntents будут заменять друг друга, что приведет к нежелательным последствиям!

**4. Отмените уведомление** путем перестройки связанного с ним PendingIntent и его отмены в системе AlarmManager. Используйте следующий метод:

```

public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);
    // Отмена уведомления
    AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_
    SERVICE);
    manager.cancel(pending);
}

```

Обратите внимание, что число 42 должно совпадать с числом из шага 3!

# Глава 70. AlarmManager

## 70.1. Как отменить сигнал Alarm

Если вы хотите отменить сигнал Alarm, но у вас нет ссылки на исходное отложенное намерение (PendingIntent), использованное для установки сигнала, вам необходимо воссоздать PendingIntent в том виде, в котором оно было создано изначально.

Согласно документации (см. [https://developer.android.com/reference/android/content/Intent.html#filterEquals\(android.content.Intent\)](https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))):

Для менеджера AlarmManager намерения считаются равными, если их действия, данные, тип, класс и категории одинаковы. При этом не сравниваются никакие дополнительные данные, включенные в намерения.

Обычно код запроса для каждого Alarm-сигнала задается как константа:

```
public static final int requestCode = 9999;
```

Итак, для простого сигнала Alarm настройка будет следующей:

```
Intent intent = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, targetTimeInMillis, pendingIntent);
```

Вот как можно создать новую ссылку на PendingIntent, которую можно использовать для отмены сигнала тревоги с помощью новой ссылки AlarmManager:

```
Intent intent = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_NO_CREATE);
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
if(pendingIntent != null) {
    alarmManager.cancel(pendingIntent);
}
```

## 70.2. Создание точных Alarm-сигналов на всех версиях Android

Со временем, когда в систему Android вносились все новые и новые оптимизации работы с батареей, методы AlarmManager также претерпели значительные изменения (для более «щадящего» определения времени). Однако для некоторых приложений все же требуется максимально точное время срабатывания на всех версиях Android. Следующий помощник использует для планирования отложенного намерения PendingIntent наиболее точный метод, доступный на всех платформах:

```
public static void setExactAndAllowWhileIdle(AlarmManager alarmManager, int type,
long triggerAtMillis, PendingIntent operation) {
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M){
        alarmManager.setExactAndAllowWhileIdle(type, triggerAtMillis, operation);
    } else if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT){
        alarmManager.setExact(type, triggerAtMillis, operation);
    } else {
        alarmManager.set(type, triggerAtMillis, operation);
    }
}
```

## 70.3. На API23+ режим Doze вмешивается в работу AlarmManager

Начиная с API23 появился режим Doze, который вмешивается в работу AlarmManager. Он использует определенные окна обслуживания для обработки сигналов тревоги, поэтому

даже при использовании функции `setExactAndAllowWhileIdle()` нельзя быть уверенным, что сигнал тревоги сработает в нужный момент времени.

Вы можете отключить это поведение для своего приложения в настройках телефона (*Настройки > Общие > Батарея и энергосбережение >忽енгрировать оптимизацию* (*Settings > General > Battery & power saving > Battery usage > Ignore optimizations*) или аналогичное).

Вы можете проверить эту настройку внутри своего приложения:

```
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
if (pm.isIgnoringBatteryOptimizations(packageName)) {
    // Ваше приложение игнорирует оптимизацию работы батареи Doze
}
```

...и в конечном итоге отобразить соответствующий диалог настроек:

```
Intent intent = new Intent();
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
intent.setData(Uri.parse("package:" + packageName));
startActivity(intent);
```

## 70.4. Как отложить запуск намерения

1. Создадим приемник. Этот класс будет получать намерение и обрабатывать его по своему усмотрению:

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Обработка намерений
        int requestCode = intent.getIntExtra("requestCode");
        ...
    }
}
```

2. Передадим намерение в `AlarmManager`. В данном примере намерение будет отправлено на `AlarmReceiver` через 1 минуту:

```
final int requestCode = 1337;
AlarmManager am = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, requestCode,
    intent, PendingIntent.FLAG_UPDATE_CURRENT);
am.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + 60000, pendingIntent);
```

# Глава 71. Обработчик

## 71.1. HandlerThreads и взаимодействие между потоками

Поскольку обработчики (`Handlers`) используются для отправки сообщений (`Messages`) и `Runnable`-объектов в очередь сообщений потока можно легко реализовать взаимодействие между несколькими потоками на основе событий. Каждый поток, имеющий `Looper`, способен принимать и обрабатывать сообщения. `HandlerThread` – это поток, реализующий подобный `Looper`, например, основной поток (`UI Thread`) реализует функции `HandlerThread`.