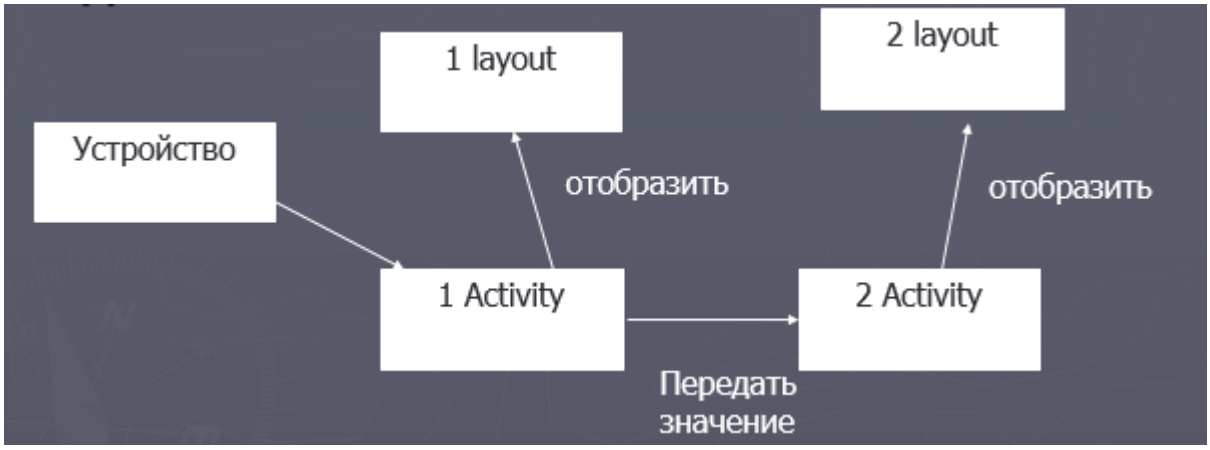
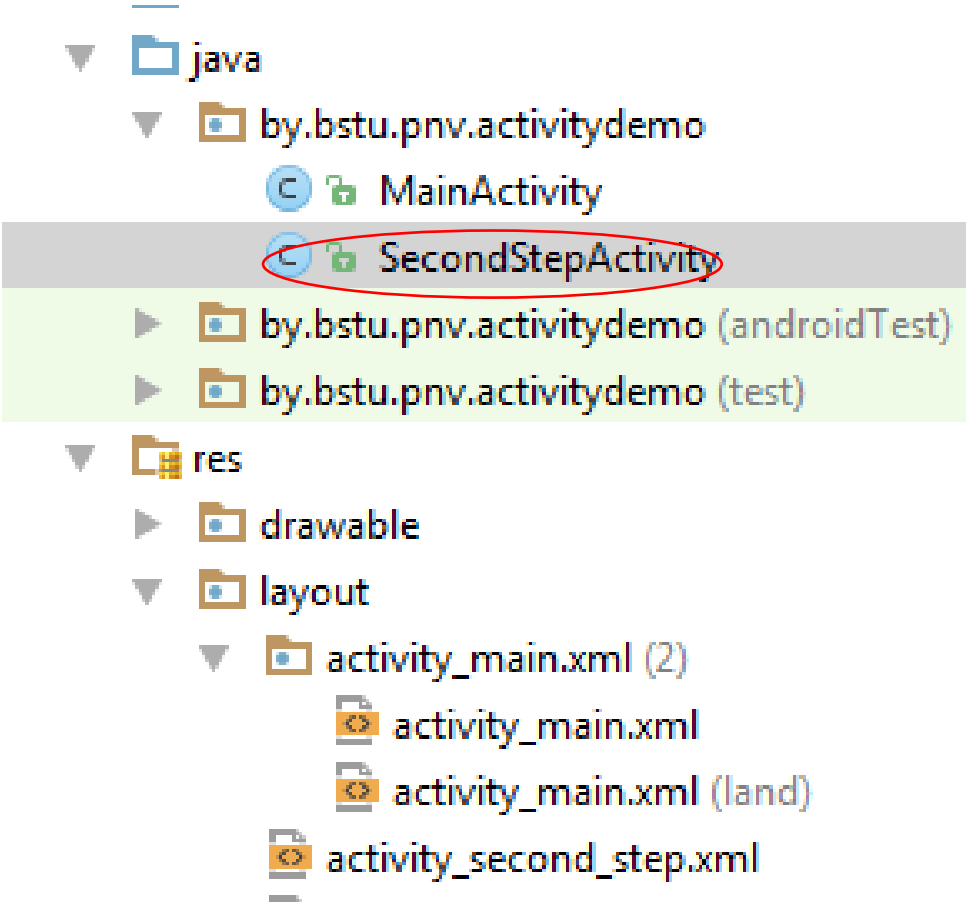
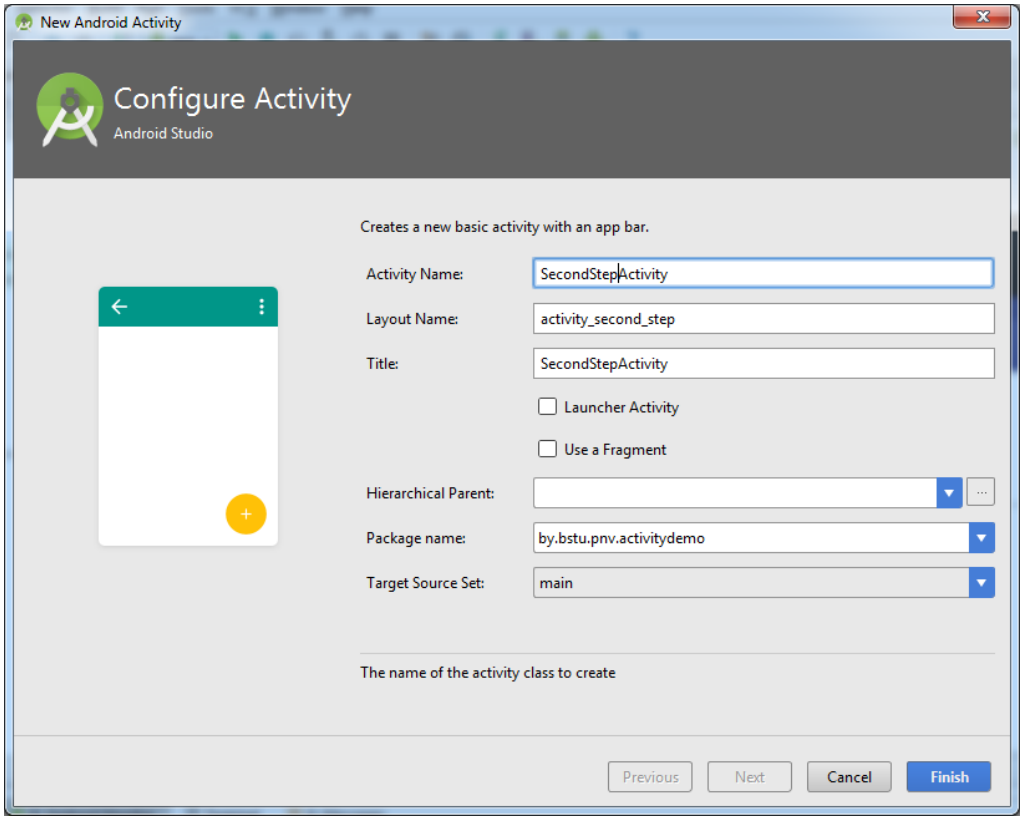
Лекция №5 Понятие Intent, взаимодействие Activity

Пусть необходимо написать приложение с двумя activity с передачей значение между activity.



Добавим активность File → New → Activity





Если мы откроем файл манифеста *AndroidManifest.xml*, то можем найти там следующие строки:

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action
android:name="android.intent.action.MAIN" />

        <category
android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity
    android:name=".SecondStepActivity"
    android:label="@string/title_activity_second_step"
    android:theme="@style/AppTheme.NoActionBar">
```

активность является главной точкой входа

имя класса активности

активность может использоваться для запуска приложения в среде

Вторая активность

метка активности

Фильтр определяется в элементе **intent-filter**. В элементе action значение "android.intent.action.MAIN" представляет главную точку входа в приложение.

Для SecondActivity просто указано, что она в проекте, и никаких intent-фильтров для нее не задано.

При создании нового приложения с помощью инструментов Android SDK в заготовке activity, создаваемой автоматически, имеется фильтр намерений, который объявляет activity. Эта activity реагирует на выполнение «основного» действия, и ее следует поместить в категории средства запуска.

Если приложение планируется создать самодостаточным и запретить другим приложениям активировать его activity, то других фильтров намерений создавать не нужно. В этом случае только в одной activity должно иметься «основное» действие, и ее следует поместить в категорию средства запуска, как в примере выше.

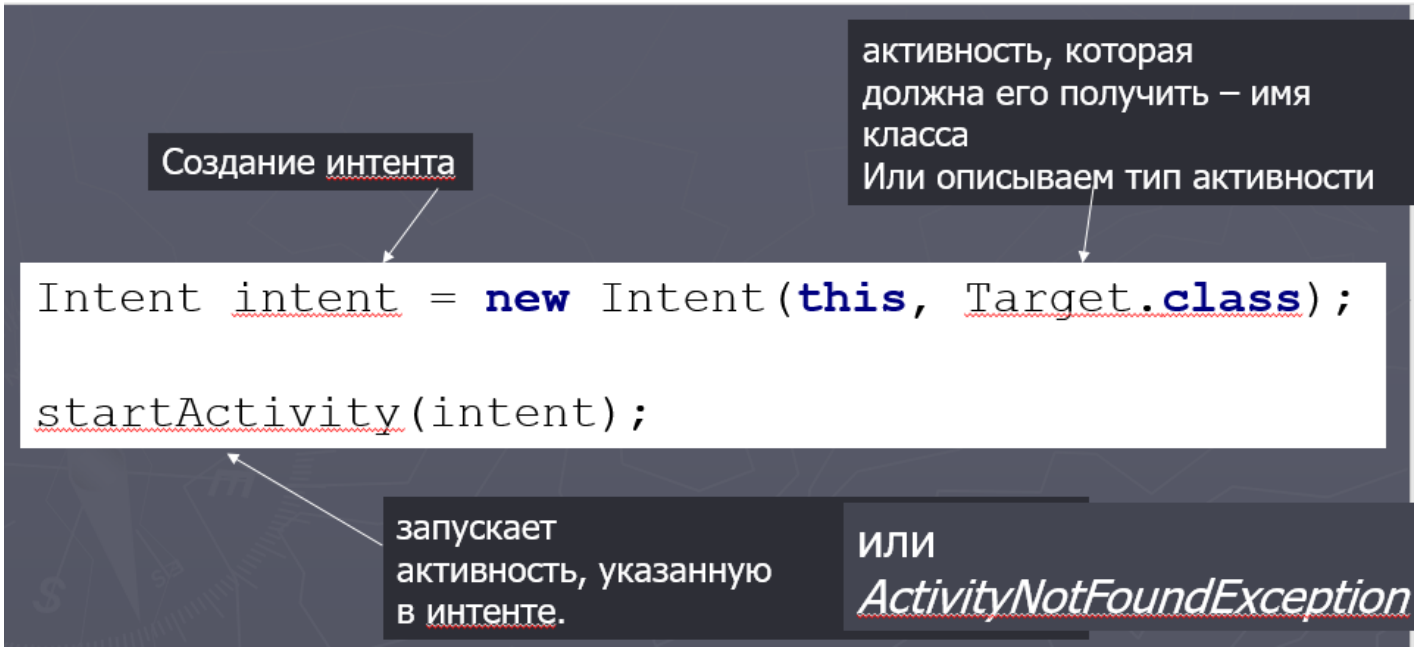
Однако, если вам необходимо, чтобы activity реагировала на неявные намерения (Intent), получаемые от других приложений (а также из вашего приложения), для activity необходимо определить дополнительные фильтры намерений. Для каждого типа намерения, на который необходимо реагировать, необходимо указать объект intent-filter;, включающий элемент action; и необязательный элемент category или data (или оба этих элемента). Эти элементы определяют тип intent, на который может реагировать ваша операция.

Intent

Для взаимодействия между различными объектами activity ключевым классом является **android.content.Intent**. Он представляет собой задачу, которую надо выполнить приложению.

Для запуска другой activity достаточно вызвать метод **startActivity()**, передав в него объект Intent, который описывает запускаемую операцию. В намерении указывается либо точная activity для запуска, либо описывается тип activity, которую вы хотите выполнить (после чего система выбирает для вас подходящую операцию, которая может даже находиться в другом приложении). Intent также может содержать небольшой объем данных, которые будут использоваться запущенной activity.

При работе с собственным приложением зачастую требуется лишь запустить нужную activity. Для этого необходимо создать intent, который явно определяет требуемую activity с помощью имени класса. Ниже представлен пример запуска одной activity другой activity.



При создании Intent мы использовали конструктор
Intent (Context packageContext, Class cls) с двумя параметрами.

Первый параметр – это Context. Activity является подклассом Context, поэтому мы можем использовать ее – this. Вкратце, Context – это объект, который предоставляет доступ к базовым функциям приложения таким как: доступ к ресурсам, к файловой системе, вызов Activiy и т.д.

Второй параметр – имя класса. При создании записи Activity в манифест-файле мы указываем имя класса. И теперь если мы укажем тот же класс в Intent – то система, просмотрев манифест-файл обнаружит соответствие и покажет соответствующий Activity.

Передача и получение значений

Также при переходе от одной activity к другой мы можем передать различную информацию с помощью метода putExtra().

```
intent.putExtra("имя", значение);
```

Фактически мы передаем словарь, который состоит из пар ключ-значение. Вызов intent.putExtra("Product", "Планшет") добавляет в этот словарь пару с ключом "Product" и значением "Планшет". Причем можно передавать не только строковые значения, но и другие, например, числовые, логические.

В коде java, применяя метод **getIntent()**, мы можем получить объект Intent, а с помощью его метода **getExtras()** - те данные, которые ранее были переданы с объектом Intent.

Для получения данных мы вызываем соответствующий метод: например, для получения строковых данных - метод **getStringExtra()**, для получения данных типа float (если бы такие были) - метод **getFloatExtra()**, а в качестве параметра используется ключ.

```
Intent intent = getIntent();
String string = intent.getStringExtra("имя");
int intNum = intent.getIntExtra("имя", 0);
```

Запуск Activity для получения результата

В некоторых случаях после запуска activity может потребоваться получить результат. Для этого вызовите метод startActivityForResult() (вместо startActivity()). Чтобы получить результат после выполнения последующей activity, реализуйте метод обратного вызова onActivityResult(). По завершении последующей activity она возвращает результат в объекте **Intent** в вызванный метод **onActivityResult()**.

К примеру, пользователю потребуется выбрать один из контактов, чтобы ваша операция могла выполнить некоторые действия с информацией об этом контакте. Ниже представлен пример создания такого намерения и обработки результата.

```
private void pickContact() {

    Intent intent = new Intent(Intent.ACTION_PICK,
                                    ContactsContract.Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (resultCode == Activity.RESULT_OK && requestCode ==
                                                    PICK_CONTACT_REQUEST) {


        }
    }
}
```

В этом примере демонстрируется базовая логика, которой следует руководствоваться для обработки результата выполнения activity. Первое условие проверяет, успешен ли запрос, и если он успешен, то результат для resultCode будет RESULT_OK; также проверяется, известен ли запрос, для которого получен этот результат, и в этом случае requestCode соответствует второму параметру, отправленному в метод startActivityForResult(). Здесь код обрабатывает результат выполнения операции путем запроса данных, возвращенных в Intent (параметр data).

С некоторого времени данный способ считается устаревшим и рекомендуется использовать *registerForActivityResult* для регистрации обратного вызова результата. registerForActivityResult () принимает *ActivityResultContract* и *ActivityResultCallback* и возвращает *ActivityResultLauncher*, который будете использовать для запуска другого действия.

ActivityResultContract определяет тип ввода, необходимый для получения результата, вместе с типом вывода результата.

https://developer.android.com/training/basics/intents/result#java

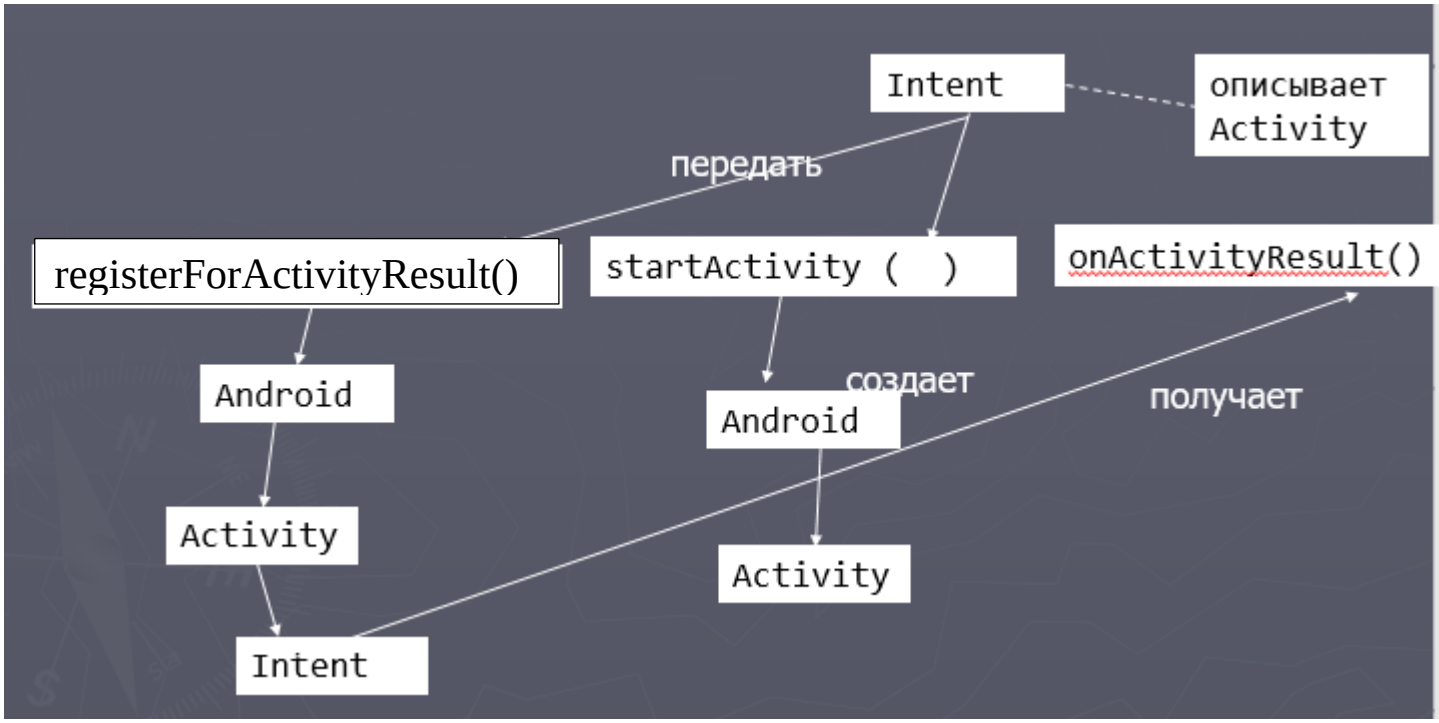
```
// в  onCreat до отображения активности
ActivityResultLauncher<Intent> someActivityResultLauncher =
registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if (result.getResultCode() == Activity.RESULT_OK) {
                    // There are no request codes
                    Intent data = result.getData();
                    doSomeOperations();
                }
            }
        });

public void openSomeActivityForResult() {
    Intent intent = new Intent(this, SomeActivity.class);
    someActivityResultLauncher.launch(intent);
}
```

Использование Intent

Однако вызовом другой activity и передача ей значений функциональность Intent не ограничивается.

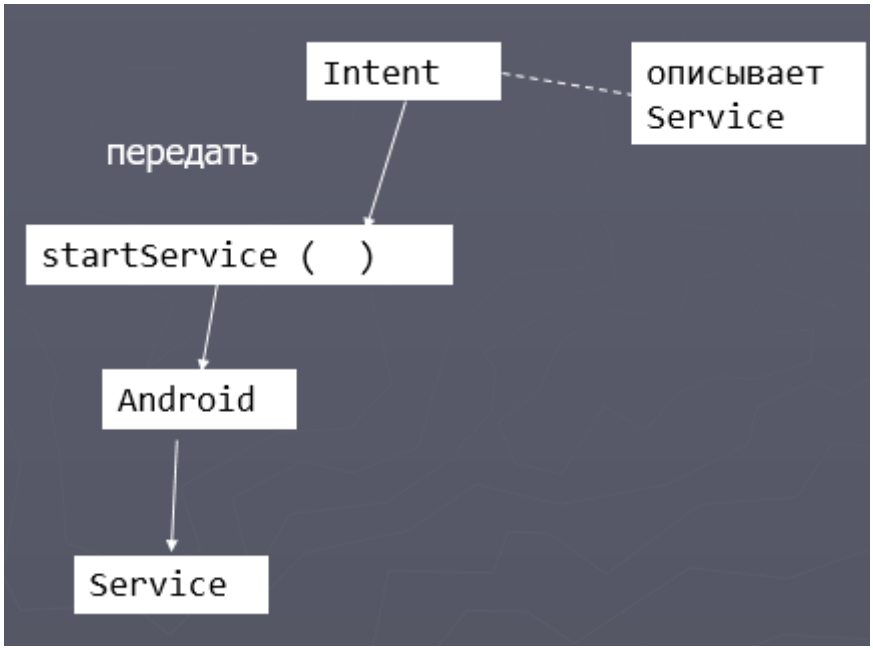
Intent представляет собой объект обмена сообщениями, с помощью которого можно запросить выполнение действия у компонента другого приложения. Несмотря на то, что объекты Intent упрощают обмен данными между компонентами по нескольким аспектам, в основном они используются в трех ситуациях:

1) Для запуска activity: см. пример выше.

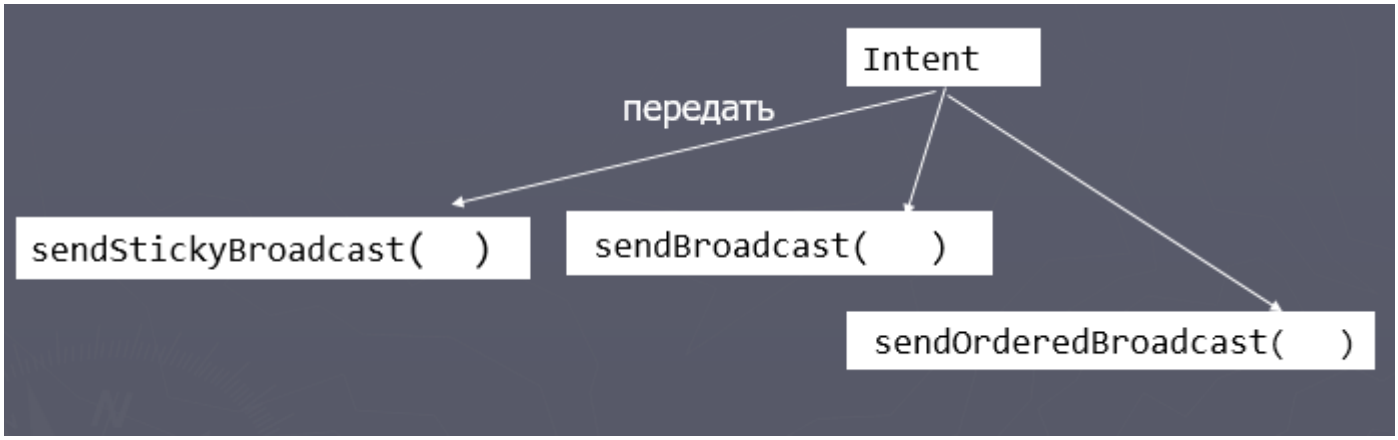


2) Для запуска службы:Service является компонентом, который выполняет действия в фоновом режиме без пользовательского интерфейса. Службу можно запустить для выполнения однократного действия (например, чтобы загрузить файл), передав объектIntent методу startService(). Объект Intent описывает службу, которую требуется запустить, а также содержит все остальные необходимые данные.

Если служба сконструирована с интерфейсом клиент-сервер, к ней можно установить привязку из другого компонента, передав объектIntent методу bindService().



3) **Для рассылки широковещательных сообщений:** Широковещательное сообщение — это сообщение, которое может принять любое приложение. Для выдачи широковещательных сообщений другим приложениям необходимо передать объект **Intent** методу **sendBroadcast**(), **sendOrderedBroadcast**() или **sendStickyBroadcast**().



Типы объектов Intent

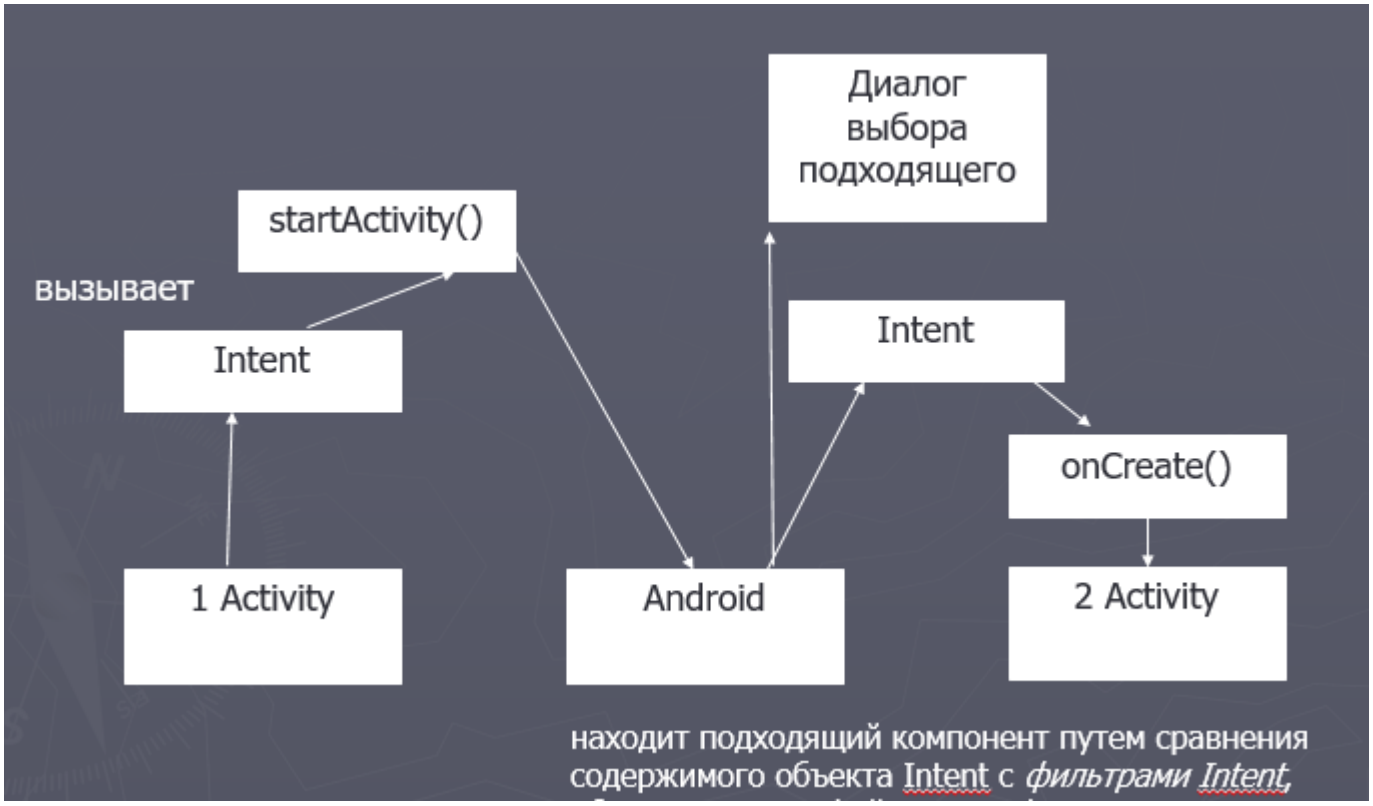
Есть два типа объектов Intent:

Явные объекты Intent указывают компонент, который требуется запустить, по имени (полное имя класса). Явные объекты Intent обычно используются для запуска компонента из вашего собственного приложения, поскольку вам известно имя класса activity или службы, которую необходимо запустить. Например , можно запустить новую activity в ответ на действие пользователя или запустить службу, чтобы загрузить файл в фоновом режиме.

```
Intent intent = new Intent(this, Target.class);
startActivity(intent);
```

Неявные объекты Intent не содержат имени конкретного компонента. Вместо этого они в целом объявляют действие, которое требуется выполнить, что дает возможность компоненту из другого приложения обработать этот запрос. Например, если требуется показать пользователю место на карте, то с помощью неявного объекта Intent можно запросить, чтобы это сделало другое приложение, в котором такая возможность предусмотрена.

Когда создан неявный объект Intent, система Android находит подходящий компонент путем сравнения содержимого объекта Intent с *фильтрами Intent*, объявленными в **файлах манифеста** других приложений, имеющихся на устройстве. Если объект Intent совпадает с фильтром Intent, система запускает этот компонент и передает ему объект Intent. Если подходящими оказываются несколько фильтров Intent, система выводит диалоговое окно, где пользователь может выбрать приложение для выполнения данного действия.



Фильтр Intent представляет собой выражение в файле манифеста приложения, указывающее типы объектов Intent, которые мог бы принимать компонент. Например, объявив фильтр Intent для activity, вы даете другим приложениям возможность напрямую запускать вашу активность с помощью некоторого объекта Intent. Точно так же, если вы *не* объявите какие-либо фильтры Intent для операции, то ее можно будет запустить только с помощью явного объекта Intent.

В целях обеспечения безопасности приложения всегда используйте явный объект Intent при запуске **Service** и не объявляйте фильтры Intent для своих служб. Запуск служб с помощью неявных объектов Intent является рискованным с точки зрения безопасности, поскольку нельзя быть на абсолютно уверенным, какая служба отреагирует на такой объект Intent, а пользователь не может видеть, какая служба запускается. Начиная с Android 5.0 (уровень API 21) система вызывает исключение при вызове метода bindService() с помощью неявного объекта Intent.

Задание неявного интента

Однако в приложении также может потребоваться выполнить некоторое действие, например, отправить письмо, текстовое сообщение или обновить статус, используя данные из activity. В этом случае в приложении могут отсутствовать такие действия, поэтому можете воспользоваться activity из других приложений, имеющихся на устройстве, которые выполнят требуемые действия. Как раз в этом случае Intent особенно полезны — можно создать intent, который описывает необходимое действие, после чего система запускает его из другого приложения.

Например, если пользователю требуется предоставить возможность отправить электронное письмо, можно создать следующее намерение:

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// проверка возможности выполнения
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
```

Дополнительный компонент EXTRA_, добавленный в intent, представляет собой текст письма. Когда почтовая программа реагирует на intent, она считывает дополнительно добавленную строку и помещает ее в поле текста письма. При этом запускается activity почтовой программы, а после того, как пользователь завершит требуемые действия, возобновляется ваша activity.

Создание объекта Intent

Объект Intent содержит информацию, на основании которой система Android определяет, какой компонент требуется запустить (например, точное имя компонента или категорию компонентов, которые должны получить этот объект Intent), а также сведения, которые необходимы компоненту-получателю, чтобы надлежащим образом выполнить действие (а именно — выполняемое действие и данные, с которыми его требуется выполнить).

Основные сведения, содержащиеся в объекте Intent:

1)Имя компонента

Имя компонента, который требуется запустить. Эта информация является необязательной, но именно она и делает объект Intent **явным**. Ее наличие означает, что объект Intent следует доставить только компоненту приложения, определенному по имени. При отсутствии имени компонента объект Intent является **неявным**, а система определяет, какой компонент получит этот объект Intent по другим сведениям, которые в нем содержатся (например, по действию, данным и категории). Поэтому, если вам требуется запустить определенный компонент из своего приложения, следует указать его имя.

Его можно задать через:

- объект ComponentName (by.bstu.ExampleActivity)
- setComponent(), setClass(), setClassName()
- или конструктор Intent

2) Действие

Строка, определяющая стандартное действие, которое требуется выполнить (например, *view* (просмотр) или *pick*(выбор)). Действие в значительной степени определяет, каким образом структурирована остальная часть объекта Intent,—в частности, что именно содержится в разделе данных и дополнительных данных.

Для использования объектами Intent в пределах своего приложения (либо для использования другими приложениями, чтобы вызывать компоненты из вашего приложения) можно указать собственные действия. Обычно же следует использовать константы действий, определенные классом Intent или другими классами платформы. Вот несколько стандартных действий для запуска операции:

ACTION_VIEW

Используйте это действие когда имеется определенная информация, которую активность может показать пользователю, например, фотография в приложении галереи или адрес для просмотра в картографическом приложении.

ACTION_SEND

Его еще называют объектом Intent "share" (намерение предоставить общий доступ). Это действие следует использовать при наличии определенных данных, доступ к которым пользователь может предоставить через другое приложение, например приложение для работы с электронной почтой или социальными сетями.

Другие константы, определяющие стандартные действия, см. в справочнике по классу Intent .

Итак: Действие - строка, определяющая стандартное действие, которое требуется выполнить.

- Можно определять свои действия (константа с префиксом пакета)
- Существуют стандартные константы действий
- Задается **setAction**() или конструктором Intent

3)Данные

Данные – это URI (объект Uri), ссылающийся на данные, с которыми будет выполняться действие и/или тип MIME этих данных. Тип передаваемых данных обычно определяется действием объекта Intent. Например, если действием является **ACTION_EDIT**, в данных должен содержаться URI документа, который требуется отредактировать. При создании объекта Intent, помимо URI, зачастую бывает важно указать тип данных (их тип MIME). Например, активность, которая может выводить на экран изображения, скорее всего, не сможет воспроизвести аудиофайл, даже если и у тех, и у других данных будут одинаковые форматы URI. Поэтому указание типа MIME данных помогает системе Android найти наиболее подходящий компонент для получения объекта Intent.

Чтобы задать только URI данных, вызовите **setData()**. Чтобы задать только тип MIME, вызовите **setType()**. При необходимости оба этих параметра можно в явном виде задать с помощью **setDataAndType()**.

Если требуется задать и URI, и тип MIME, **не** вызывайте **setData()** и **setType()**, поскольку каждый из этих методов аннулирует результат выполнения другого. Чтобы задать URI и тип MIME всегда используйте метод **setDataAndType()**.

4)Категория

Строка, содержащая прочие сведения о том, каким компонентом должна выполняться обработка этого объекта Intent. В объект Intent можно поместить любое количество описаний категорий, однако большинству объектов Intent категория не требуется. Вот некоторые стандартные категории:

CATEGORY_BROWSABLE Целевая activity позволяет запускать себя веб-браузером для отображения данных, указанных по ссылке — например, изображения или сообщения электронной почты.

CATEGORY_LAUNCHER Эта activity является начальной для задачи, она указана в средстве запуска приложений системы.Полный список категорий см. в описании класса **Intent** .

Указать категорию можно с помощью **addCategory().**

Приведенные выше свойства (имя компонента, действие, данные и категория) представляют собой характеристики, определяющие объект **Intent**. На основании этих свойств система Android может решить, какой компонент следует запустить.

5) Дополнительные данные

Это пары "ключ-значение", содержащие прочую информацию, которая необходима для выполнения запрошенного действия. Точно так же, как некоторые действия используют определенные виды URI данных, некоторые действия используют определенные дополнительные данные. Добавлять дополнительные данные можно с помощью различных методов **putExtra**(), каждый из которых принимает два параметра: имя и значение ключа. Также можно создать объект **Bundle** со всеми дополнительными данными, затем вставить объект **Bundle** в объект **Intent** с помощью метода **putExtras**().

Например, при создании объекта Intent для отправки сообщения электронной почты с методом **ACTION_SEND** можно указать получателя с помощью ключа **EXTRA_EMAIL**, а тему сообщения — с помощью ключа **EXTRA_SUBJECT**.

Класс **Intent** указывает много констант EXTRA_* для стандартных типов данных.

- EXTRA_REPLACING
- EXTRA_SHORTCUT_ICON
- EXTRA_SHORTCUT_ICON_RESOURCE
- EXTRA_SHORTCUT_INTENT
- EXTRA_STREAM
- EXTRA_SHORTCUT_NAME
- EXTRA_SUBJECT
- EXTRA_TEMPLATE

- EXTRA_TEXT
- EXTRA_TITLE
- EXTRA_UIDs

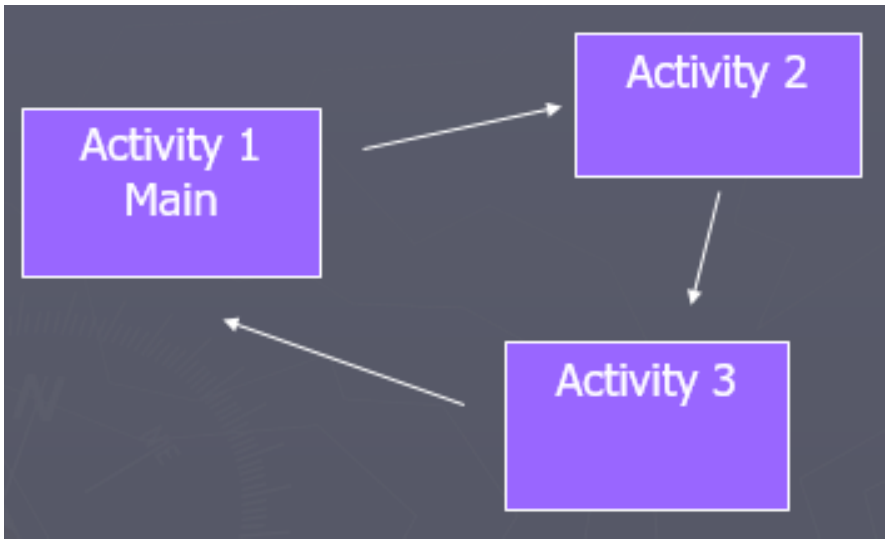
Если вам требуется объявить собственные дополнительные ключи (для объектов Intent, которые принимает ваше приложение), обязательно указывайте в качестве префикса имя пакета своего приложения. Например:

```
static final String EXTRA_BSTU = "by.bstu.EXTRA_BSTU";
```

6) Флаги

Флаги, определенные в классе **Intent**, которые действуют как метаданные для объекта Intent. Флаги должны указывать системе Android, каким образом следует запускать активность (например, к какой задаче должна принадлежать activity) и как с ней обращаться после запуска (например, будет ли она указана в списке последних).Подробные сведения см. в документе, посвященном методу setFlags().

Теперь рассмотрим некоторые особенности взаимодействия между activity в одном приложении. Допустим, у нас есть три activity: MainActivity, SecondActivity и ThirdActivity.



С помощью Intent, например, по нажатию кнопки MainActivity запускает Activity2. На Activity2 тоже есть кнопка, которая запускает Activity3. На Activity3 также есть кнопка, которая возвращается к первой activity - MainActivity:

Если мы последовательно запустим все activity: из главной MainActivity запустим Activity2, из Activity2 – Activity3. Если после этого из Activity3 мы захотим обратиться к MainActivity1, то метод startActivity() запустит новый объект MainActivity1 (а не вернется к уже существующему), и стек уже будет выглядеть следующим образом:

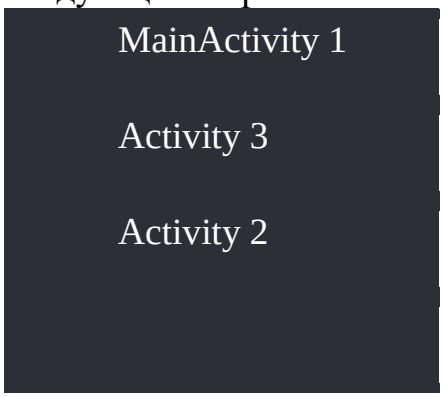
У нас будут две независимые копии MainActivity1. Такое положение нежелательно, если мы просто хотим перейти к существующей.

Если мы нажмем на кнопку Back (Назад), то мы сможем перейти к предыдущей activity в стеке.

Чтобы выйти из этой ситуации, мы можем использовать флаг **Intent.FLAG_ACTIVITY_REORDER_TO_FRONT**:

```
Intent intent = new Intent(this, MainActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
startActivity(intent);
```

В этом случае после перехода из Activity 3 к MainActivity стек будет выглядеть следующим образом:



Если же нам просто надо перейти из Activity3 к MainActivity, как если бы мы перешли назад с помощью кнопки Back, то мы можем использовать флаги **Intent.FLAG_ACTIVITY_CLEAR_TOP** и **Intent.FLAG_ACTIVITY_SINGLE_TOP**:

```
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
```

В этом случае после перехода из Activity3 к MainActivity стек будет полностью очищен, и там останется одна MainActivity.

Пример неявного объекта Intent

Неявный объект Intent указывает действие, которым может быть вызвано любое имеющееся на устройстве приложение, способное выполнить это действие. Неявные объекты Intent используются, когда ваше приложение не может выполнить то или иное действие, а другие приложения, скорее всего, могут и вы хотите, чтобы пользователь имел возможность выбрать, какое приложение для этого использовать.

Например, если у вас есть контент и вы хотите, чтобы пользователь поделился им с другими людьми, создайте объект Intent с действием **ACTION_SEND** и добавьте дополнительные данные, указывающие на контент, общий доступ к которому следует предоставить. Когда с помощью этого объекта Intent вы вызываете **startActivity**(), пользователь сможет выбрать приложение, посредством которого к контенту будет предоставлен общий доступ.

Возможна ситуация, когда на устройстве пользователя не будет *никакого* приложения, которое может откликнуться на неявный объект Intent, отправленный вами методу **startActivity**(). В этом случае вызов закончится неудачей, а работа приложения аварийно завершится. Чтобы проверить, будет получен ли операцией объект Intent, вызовите метод **resolveActivity**() для своего объекта **Intent**. Если результатом будет значение, отличное от null, значит, имеется хотя бы одно приложение, которое способно откликнуться на объект Intent и можно вызывать **startActivity**(). Если же результатом будет значение null, объект Intent не следует использовать и по возможности следует отключить функцию, которая выдает этот объект Intent.

```
Button sbut = (Button) findViewById(R.id.bSend);

        sbut.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View view) {
                Intent sendIntent = new Intent();
                sendIntent.setAction(Intent.ACTION_SEND);
                sendIntent.putExtra(Intent.EXTRA_TEXT, "Send something here");
                sendIntent.setType("text/plain");

      Intent chooser = Intent.createChooser(sendIntent, "Choose the activity");


                if (sendIntent.resolveActivity(getPackageManager()) != null) {
                    startActivity(chooser);
                }
            }
        });
```

При вызове метода **startActivity**() система анализирует все установленные приложения, чтобы определить, какие из них могут откликнуться на объект Intent этого вида. Если имеется только одно подходящее приложение, оно будет сразу же открыто и получит данный объект Intent. Если объект Intent принимают несколько

activity, система отображает диалоговое окно, в котором пользователь может выбрать приложение для выполнения данного действия.

При наличии нескольких приложений, откликающихся на ваш неявный объект Intent, пользователь может выбрать требуемое приложение и указать, что оно будет по умолчанию выполнять это действие. Это удобно в случае действия, для выполнения которого пользователь обычно хочет всегда использовать одно и то же приложение, например, при открытии веб-страницы (пользователи обычно используют один и тот же браузер).

Настройка фильтров для intent

Чтобы указать, какие неявные объекты Intent может принимать ваше приложение, объявите один или несколько фильтров Intent для каждого компонента приложения с помощью элемента **intent-filter**; в файле манифеста.

Каждый фильтр Intent указывает тип объектов Intent, которые принимает компонент на основании действия, данных и категории, заданных в объекте Intent. Система передаст неявный объект Intent вашему приложению, только если он может пройти через один из ваших фильтров Intent.

Явный объект Intent всегда доставляется его целевому компоненту, без учета любых фильтров Intent, объявленных компонентом.

Компонент приложения должен объявлять отдельные фильтры для каждой уникальной работы, которую он может выполнить. Например, у activity из приложения для работы с галереей изображений может быть два фильтра: один фильтр для просмотра изображения, и второй для его редактирования. Когда активность запускается, она анализирует объект **Intent** и выбирает режим своей работы на основании информации, приведенной в **Intent** (например, показывать элементы управления редактора или нет).

```
<intent-filter>
    <action>
    </action>
    <data>
    </data>
    <category>
    </category>
</intent-filter>
```

Каждый фильтр Intent определяется элементом **intent-filter** в файле манифеста приложения, указанном в объявлении соответствующего компонента приложения. Внутри элемента **intent-filter** , можно указать тип объектов Intent, которые будут приниматься, с помощью одного или нескольких из следующих трех элементов:

action

Объявляет принимаемое действие, заданное в объекте Intent, в атрибуте name. Значение должно быть текстовой строкой действия, а не константой класса.

data

Объявляет тип принимаемых данных, для чего используется один или несколько атрибутов, указывающих различные составные части URI данных (scheme, host, port, path и т. д.) и тип MIME

category;

Объявляет принимаемую категорию, заданную в объекте Intent, в атрибуте name. Значение должно быть текстовой строкой действия, а не константой класса.

Дополнения и флаги не играют никакой роли в принятии решения, какой компонент получает намерение.

Можно создавать фильтры, в которых будет несколько экземпляров **action, data, category.** В этом случае просто нужно убедиться в том, что компонент может справиться с любыми сочетаниями этих элементов фильтра.

Неявный объект Intent проверяется фильтром путем сравнения объекта Intent с каждым из этих трех элементов. Чтобы объект Intent был доставлен компоненту, он должен пройти все три теста. Если он не будет соответствовать хотя бы одному из них, система Android не доставит этот объект Intent компоненту. Однако, поскольку у компонента может быть несколько фильтров Intent, объект Intent, который не проходит через один из фильтров компонента, может пройти через другой фильтр.

Пример:

```
<activity android:name="SActivity">
   <intent-filter>
      <action android:name="android.intent.action.SEND"/>
      <category android:name="android.intent.category.DEFAULT"/>
      <data android:mimeType="text/plain"/>
   </intent-filter>
</activity>
```

Здесь объявлена Activity с фильтром Intent, определяющим получение объекта Intent ACTION_SEND, когда данные относятся к типу text.

Разрешение объектов Intent

Когда система получает неявный объект Intent для запуска activity, она выполняет поиск наиболее подходящей операции путем сравнения объекта Intent с фильтрами Intent по трем критериям:

- действие объекта Intent;
- данные объекта Intent (тип URI и данных);
- категория объекта Intent.

Тестирование действия

Для указания принимаемых действий объекта Intent фильтр Intent может объявлять любое (*в том числе нулевое*) число элементов **action**

```
<intent-filter>
   <action android:name="android.intent.action.EDIT" />
   <action android:name="android.intent.action.VIEW" />
   ...
</intent-filter>
```

Чтобы пройти через этот фильтр, действие, указанное в объекте Intent, должно соответствовать **одному или нескольким** действиям, перечисленным в фильтре.

Если в фильтре не перечислены какие-либо действия, объекту Intent будет нечему соответствовать, поэтому все объекты Intent не пройдут этот тест. Однако, если в объекте Intent не указано действие, он пройдет тест (если в фильтре содержится хотя бы одно действие).

Константы действия

- ACTION_ANSWER — Открывает активность, которая связана с входящими звонками. Это действие обрабатывается стандартным экраном для приема звонков;
- ACTION_CALL — инициализирует обращение по телефону;
- ACTION_DELETE — Запускает активность, с помощью которой можно удалить данные, указанные в пути URI внутри намерения;
- ACTION_EDIT — Отображает данные для редактирования пользователем;
- ACTION_INSERT — Открывает активность для вставки в Курсор (Cursor) нового элемента, указанного с помощью пути URI. Дочерняя активность, вызванная с этим действием, должна вернуть URI, ссылающийся на вставленный элемент;
- ACTION_HEADSET_PLUG - Подключение наушников;
- ACTION_MAIN — Запускается как начальная активность задания;
- ACTION_PICK - Загружает дочернюю Активность, позволяющую выбрать элемент из источника данных, указанный с помощью пути URI. При закрытии должен возвращаться URI, ссылающийся на выбранный элемент. Активность, которая будет запущена, зависит от типа выбранных данных, например при передаче пути content://contacts/people вызовется системный список контактов;
- ACTION_SEARCH — Запускает активность для выполнения поиска. Поисковый запрос хранится в виде строки в дополнительном параметре намерения по ключу SearchManager.QUERY;
- ACTION_SEND — Загружает экран для отправки данных, указанных в намерении. Контакт-получатель должен быть выбран с помощью полученной

активности. Используйте метод setType, чтобы указать тип MIME для передаваемых данных. Эти данные должны храниться в параметре намерения *extras* с ключами EXTRA_TEXT или EXTRA_STREAM, в зависимости от типа. В случае с электронной почтой стандартное приложение в Android также принимает дополнительные параметры по ключам EXTRA_EMAIL, EXTRA_CC, EXTRA_BCC и EXTRA_SUBJECT. Используйте действие ACTION_SEND только в тех случаях, когда данные нужно передать удаленному адресату (а не другой программе на том же устройстве);

- ACTION_SENDTO — Открывает активность для отправки сообщений контакту, указанному в пути URI, который передаётся через намерение;
- ACTION_SYNC — Синхронизирует данные сервера с данными мобильного устройства;
- ACTION_TIMEZONE_CHANGED - Смена часового пояса;
- ACTION_VIEW — Наиболее распространенное общее действие. Для данных, передаваемых с помощью пути URI в намерении, ищется наиболее подходящий способ вывода. Выбор приложения зависит от схемы (протокола) данных. Стандартные адреса http: будут открываться в браузере, адреса tel: — в приложении для дозвона, geo: — в программе Google Maps, а данные о контакте — отображатся в приложении для управления контактной информацией;
- ACTION_WEB_SEARCH — Открывает активность, которая ведет поиск в интернете, основываясь на тексте, переданном с помощью пути URI (как правило, при этом запускается браузер);

Тестирование категории

Для указания принимаемых категорий объекта Intent фильтр Intent может объявлять любое (в том числе нулевое) число элементов **category**.

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

Чтобы объект Intent прошел тестирование категории, **все категории**, приведенные в объекте Intent, должны соответствовать категории из фильтра. Обратное не требуется — фильтр Intent может объявлять и другие категории, которых нет в объекте Intent, объект Intent при этом все равно пройдет тест. Поэтому объект Intent без категорий всегда пройдет этот тест, независимо от того, какие категории объявлены в фильтре.

Система Android автоматически применяет категорию CATEGORY_DEFAULT ко всем неявным объектам Intent, которые передаются в startActivity() и registerForActivityResult(). Поэтому, если вы хотите,

чтобы ваша операция принимала неявные объекты Intent, в ее фильтрах Intent должна быть указана категория для "android.intent.category.DEFAULT"

Вы можете задать собственные категории или же брать стандартные значения, предоставляемые системой:

ALTERNATIVE - Наличие данной категории говорит о том, что действие должно быть доступно в качестве альтернативного тому, которое выполняется по умолчанию для элемента этого типа данных. Например, если действие по умолчанию для контакта — просмотр, то в качестве альтернативы его также можно редактировать

SELECTED_ALTERNATIVE - То же самое, что и ALTERNATIVE, но вместо одиночного действия с использованием утверждения намерения, которое описано выше, применяется в тех случаях, когда нужен список различных возможностей. Одной из функций фильтра намерений может стать динамическое заполнение контекстного меню с помощью действий.

BROWSABLE - Говорит о том, что действие доступно из браузера. Когда намерение срабатывает в браузере, оно всегда содержит данную категорию. Если вы хотите, чтобы приложение реагировало на действия, инициированные браузером (такие как перехват ссылок на конкретный сайт), то должны добавить в его манифест категорию BROWSABLE.

DEFAULT - Установите эту категорию, чтобы сделать компонент обработчиком по умолчанию для действия, выполняемого с указанным типом данных внутри Фильтра намерений. Это необходимо и для Активностей, которые запускаются с помощью явных Намерений

GADGET - Наличие этой категории указывает на то, что данная активность может запускаться внутри другой активности.

HOME - Устанавливая эту категорию и не указывая при этом действия, вы создаете альтернативу для стандартного домашнего экрана.

LAUNCHER - Используя эту категорию, вы помещаете Активность в окно для запуска приложений.

Тестирование данных

Для указания принимаемых данных объекта Intent фильтр Intent может объявлять любое (в том числе нулевое) число элементов **data**. Например:

```
<intent-filter>
   <data android:mimeType="video/mpeg" android:scheme="http" ... />
   <data android:mimeType="audio/mpeg" android:scheme="http" ... />
   ...
</intent-filter>
```

Каждый элемент **<data>** может конкретизировать структуру URI и тип данных (тип мультимедиа MIME). Имеются отдельные атрибуты
— scheme, host, port и path — для каждой составной части URI:

scheme://host:port/path

Например:
content://by.example.project:200/folder/subfolder/etc

схема - content, узел — by.example.project, порт — 200, путь — folder/subfolder/etc.

Каждый из этих атрибутов является необязательным, однако имеются линейные зависимости:

- Если схема не указана, узел игнорируется.
- Если узел не указан, порт игнорируется.
- Если не указана ни схема, ни узел, путь игнорируется.

Когда URI, указанный в объекте Intent, сравнивается с URI из фильтра, сравнение выполняется только с теми составными частями URI, которые приведены в фильтре.

При выполнении тестирования данных сравнивается и URI, и тип MIME, указанные в объекте Intent, с URI и типом MIME из фильтра. Действуют следующие правила:

1)Объект Intent, который не содержит ни URI, ни тип MIME, пройдет этот тест, только если в фильтре не указано никаких URI или типов MIME.

2) Объект Intent, в котором имеется URI, но отсутствует тип MIME (ни явный, ни тот, который можно вывести из URI), пройдет этот тест, только если URI соответствует формату URI из фильтра , а в фильтре также не указан тип MIME.

3) Объект Intent, в котором имеется тип MIME, но отсутствует URI, пройдет этот тест, только если в фильтре указан тот же тип MIME и не указан формат URI.

4) Объект Intent, в котором имеется и URI, и тип MIME (явный или тот, который можно вывести из URI), пройдет только часть этого теста, проверяющую тип MIME, в том случае, если этот тип совпадает с типом, приведенным в фильтре. Он пройдет часть этого теста, которая проверяет URI, либо если его URI совпадает с URI из фильтра, либо если этот объект содержит URI content: или file:, а в фильтре URI не указан. Другими словами, предполагается, что компонент поддерживает данные content: и file:, если в его фильтре указан *только* тип MIME.

Это последнее правило (правило (4)) отражает ожидание того, что компоненты будут в состоянии получать локальные данные из файла или от поставщика контента. Поэтому их фильтры могут содержать только тип данных, а явно указывать схемы content: и file: не требуется.

Другой стандартной конфигурацией являются фильтры со схемой и типом данных. Например, элемент сообщает системе Android, что компонент может получать видеоданные из сети для выполнения действия:

```
<intent-filter>
   <data android:scheme="http" android:type="video/*" />
   ...
</intent-filter>
```

Принцип работы фильтров намерений

При использовании метода **startActivity()** передаваемое неявное намерение, как правило, доходит лишь до одной активности. Если для выполнения заданного действия с указанными данными годятся сразу несколько активностей, пользователю предоставляется список выбора. Процесс, когда решается, какую активность лучше запустить, называется Утверждением намерений. Его цель — найти наиболее подходящий фильтр намерений. В целом весь алгоритм работает следующим образом.

1. Android собирает список всех доступных Фильтров намерений из установленных пакетов.
2. Фильтры, которые не соответствуют действию или категории Намерения, удаляются из списка. Совпадение происходит только в том случае, если Фильтр намерений содержит указанное действие (или если действие для него вовсе не задано). Совпадения не произойдет, только если ни одно из действий Фильтра намерений не будет эквивалентно тому, которое задано в Намерении. Для категорий процесс соответствия более строгий. Фильтр намерений должен включать в себя все категории, заданные в полученном Намерении. Фильтр, для которого категории не указаны, может соответствовать только таким же Намерениям (нет категорий).
3. Наконец, каждая часть пути URI из Намерения сравнивается с тегом data Фильтра намерений. Если в Фильтре указаны схема (протокол), сервер/принадлежность, путь или тип MIME, все эти значения проверяются на соответствие пути URI из Намерения. При любом несовпадении Фильтр будет удален из списка. Если в Фильтре намерений не указано ни одного параметра data, его действие будет распространяться на любые данные.
    - MIME — тип данных, который должен совпасть. При сравнении типов данных вы можете использовать маски, чтобы охватывать все подтипы (например, cats/*). Если в Фильтре намерения указан тип данных, он должен совпасть с тем, который значится в намерении, при отсутствии тега data подойдет любой тип.
    - Схема — это протокольная часть пути URI, например http:, mailto: или tel:.
    - Имя сервера (или принадлежность данных) — часть URI между схемой и самим путем (например, www.google.com). Чтобы совпало имя сервера, схема Фильтра намерений также должна подойти.
    - После имени сервера идет путь к данным (например, /ig). Путь пройдет проверку только после схемы и имени сервера, содержащихся в теге.
4. Когда вы неявным образом запускаете Активность и вышеописанный процесс возвращает более одного совпадения, пользователю выводится список со всеми вариантами

Общие Intents

Alarm Clock

```
Intent intent = new Intent(AlarmClock.ACTION_SET_ALARM)
        .putExtra(AlarmClock.EXTRA_MESSAGE, message)
        .putExtra(AlarmClock.EXTRA_HOUR, hour)
        .putExtra(AlarmClock.EXTRA_MINUTES, minutes);
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

Фильтр:

```
<intent-filter>
    <action android:name="android.intent.action.SET_ALARM" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Календарь – создать событие

```
Intent intent = new Intent(Intent.ACTION_INSERT)
            .setData(CalendarContract.Events.CONTENT_URI)
            .putExtra(CalendarContract.Events.TITLE, title)
            .putExtra(CalendarContract.Events.EVENT_LOCATION,
location)
            .putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
begin)
            .putExtra(CalendarContract.EXTRA_EVENT_END_TIME, end);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
```

Фильтр:

```
<intent-filter>
    <action android:name="android.intent.action.INSERT" />
    <data android:mimeType="vnd.android.cursor.dir/event" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Открытие камеры в режиме изображения

```
Intent intent = new
Intent(MediaStore.INTENT_ACTION_STILL_IMAGE_CAMERA);
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivityForResult(intent);
}
```

Фильтр:

```
<intent-filter>
    <action android:name="android.media.action.STILL_IMAGE_CAMERA"
/>
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Посылка Email с вложениями

```
public void composeEmail(String[] addresses, String subject, Uri
attachment) {
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("*/*");
    intent.putExtra(Intent.EXTRA_EMAIL, addresses);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    intent.putExtra(Intent.EXTRA_STREAM, attachment);

 if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

ACTION_SENDTO (for no attachment)
ACTION_SEND (for one attachment)
ACTION_SEND_MULTIPLE (for multiple attachments)

Фильр

```
<intent-filter>
    <action android:name="android.intent.action.SEND" />
    <data android:type="*/*" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Излечение специфического файла из файловой системы

```
static final int REQUEST_IMAGE_GET = 1;

public void selectImage() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_IMAGE_GET);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
```

```
<intent-filter>
    <action android:name="android.intent.action.GET_CONTENT" />
    <data android:type="image/*" />
    <category android:name="android.intent.category.OPENABLE" />
</intent-filter>
```

`OPENABLE` - Доступны из контент- провайдера

Указание геолокации

```
public void showMap(Uri geoLocation) {
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(geoLocation);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

фильтр

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <data android:scheme="geo" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

Набор номера и звонок

```
public void dialPhoneNumber(String phoneNumber) {
    Intent intent = new Intent(Intent.ACTION_DIAL);
    intent.setData(Uri.parse("tel:" + phoneNumber));
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
```

- ACTION_DIAL - Opens the dialer or phone app.
- ACTION_CALL - Places a phone call (requires the CALL_PHONE permission)

Загрузка web URL

```
public void openWebPage(String url) {
    Uri webpage = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }

}
```

Фильтр

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />

    <data android:scheme="http" android:host="www.belstu.by" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
</intent-filter>
```