4. ПОНЯТИЕ INTENT. ВЗАИМОДЕЙСТВИЕ ACTIVITY

4.1. Добавление *Activity*

Как правило, приложение состоит из нескольких активностей, которые связаны (иногда слабо) друг с другом. Обычно одна из *Activity* в приложении обозначается как «основная» (отображается при первом запуске приложения). В свою очередь, каждая *Activity* может запустить другую *Activity*. При этом предыдущая останавливается и система сохраняет ее в стеке (называется *BackStack*).

Когда одна активность запускает другую, в жизненных циклах обеих из них происходит переход из одного состояния в другое. Первая активность приостанавливается и завершается (однако она не будет остановлена, если по-прежнему видима на фоне), а вторая активность создается. В случае если эти операции обмениваются данными, сохраненными на диске или в другом месте, первая *Activity* не останавливается полностью до тех пор, пока не будет создана вторая *Activity*. Наоборот, процесс запуска второй накладывается на процесс остановки первой. Порядок обратных вызовов жизненного цикла четко определен, в частности, когда в одном и том же процессе находятся две *Activity* и одна из них запускает другую.

Добавим к проекту новую *Activity* (рис. 4.1, рис. 4.2).

Если открыть файл манифеста *AndroidManifest.xml*, то можем найти там определение второй активности:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="by.bstu.patsei.myapplication">

    <application
        android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"          android:theme="@style/AppTheme">
        <activity android:name=".SecondActivity"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```
        </activity>
    </application>

</manifest>
```

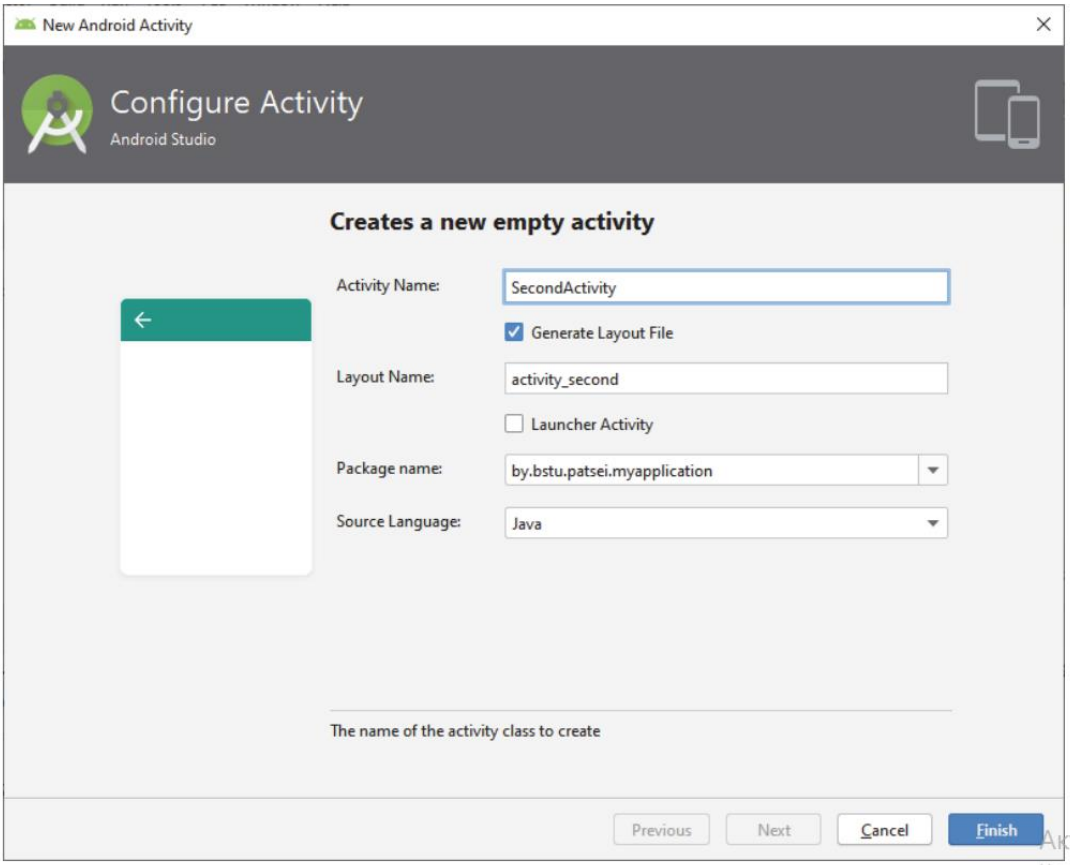

Рис. 4.1. Окно создания новой *Activity*

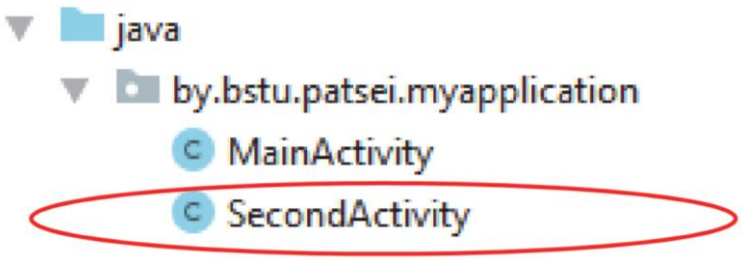

Рис. 4.2 Проект с двумя Activity

В <*action*> значение *android.intent.action.MAIN* представляет главную точку входа в приложение. Для *SecondActivity* просто указано, что она используется в проекте.

Если приложение будет самодостаточным и запрещено другим приложениям активировать его, то других фильтров намерений (intentfilter) создавать не нужно. Только в одной активности должно иметься «основное» действие, и ее следует поместить в категорию средств запуска, как в данном примере.

4.2. Понятие Intent

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

Для запуска другой активности достаточно вызвать метод *startActivity*(), передав в него объект *Intent*, который ее описывает. В *Intent* либо указывается точная активность для запуска, либо описывается тип *Activity*, которую надо выполнить (после чего ОС выбирает подходящую активность, которая может находиться в другом приложении). *Intent* может содержать небольшой объем данных, которые будут использоваться запущенной *Activity*.

При работе с собственным приложением требуется лишь запустить нужную *Activity*. Для этого необходимо создать *Intent*, который явно определяет *Activity* с помощью имени класса. Например:

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

При создании *Intent* используется конструктор с двумя параметрами. Первый параметр – это *Context*. Здесь *Activity* является подклассом *Context* (объект, который предоставляет доступ к базовым функциям приложения, таким как доступ к ресурсам и файловой системе, вызов *Activity* и т. д.). Второй параметр – имя класса.

4.2.1. Передача и получение значений из Activity

При переходе от одной активности к другой можно передавать различную информацию с помощью метода *putExtra*():

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

Фактически это словарь, который состоит из пар «ключ – значение». Можно передавать не только строковые значения, но и другие, например числовые, логические. Вызывая метод *getIntent*(), можно получить объект *Intent*, а с помощью его метода *getExtras*() – те данные, которые ранее были переданы с объектом *Intent*. Для получения строковых данных используется метод *getStringExtra*(), для получения данных типа *float* – *getFloatExtra*(), а в качестве параметра используется ключ:

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

4.2.2. Запуск Activity для получения результата

После запуска активности может потребоваться получить результат. Для этого вызывается метод *startActivityForResult*() вместо *startActivity*(). Чтобы получить результат после выполнения последующей активности, необходимо реализовать метод обратного вызова *onActivityResult*(). По завершении она возвращает результат в объекте *Intent* в вызванный метод *onActivityResult*().

4.2.3. Использование Intent

Intent представляет собой объект обмена сообщениями, с помощью которого можно запросить выполнение действия у компонента другого приложения. *Intent* используется в трех ситуациях:

1) для запуска *Activity*, как было показано выше;
2) для запуска *Service* (службы). Объект *Intent* описывает службу, которую требуется запустить, а также содержит все остальные необходимые данные;
3) для рассылки широковещательных сообщений. Для выдачи широковещательных сообщений другим приложениям необходимо передать объект *Intent* методу *sendBroadcast*(), *sendOrderedBroadcast*() или *sendStickyBroadcast*().

4.3. Типы объектов Intent

Есть два типа объектов Intent: явные и неявные.

Явный объект *Intent* указывает компонент, который требуется запустить, по имени (полное имя класса). Явные объекты *Intent* обычно используются для запуска компонента из своего приложения, поскольку известно имя класса *Activity* или службы, которую необходимо запустить.

Неявный объект *Intent* не содержит имени конкретного компонента. Вместо этого он в целом объявляет действие, которое требуется выполнить, и компонент из другого приложения обработает 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. 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, при объявлении фильтра *Intent* для активности другим приложениям дается возможность напрямую запускать *Activity* с помощью некоторого объекта *Intent*. Точно так же, если не объявить какие-либо фильтры *Intent* для активности, ее можно будет запустить только с помощью явного объекта *Intent*.

4.3.1. Задание неявного объекта Intent

В приложении может потребоваться выполнить некоторое действие, например отправить письмо, текстовое сообщение или обновить статус, используя данные из текущей *Activity*. В этом случае можно воспользоваться *Activity* из других приложений, имеющихся на устройстве.

Например, если пользователю требуется предоставить возможность отправить электронное письмо, можно создать следующий *Intent*:

```
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 активность почтовой программы, а после того, как пользователь завершит требуемые действия, возобновляется текущая активность.

4.3.2. Определение объекта Intent

Рассмотрим сведения, которые содержит *Intent*.

Имя компонента, который требуется запустить. Эта информация является необязательной, но именно она и делает объект *Intent* ***явным***. При отсутствии имени компонента объект *Intent* является ***неявным***. Имя можно задать через объект *ComponentName*(), *setComponent*(), *setClass*(), *setClassName*() или конструктор *Intent.*

Действие – это строка, определяющая стандартное действие, которое требуется выполнить (*view*, *pick*, ...). Действие в значительной степени определяет, каким образом будет структурирована остальная часть объекта *Intent*. Для использования *Intent* в пределах своего приложения можно указать собственные действия. Обычно следует использовать константы

действий, определенные классом *Intent* или другими классами платформы (например, *ACTION_VIEW*, *ACTION_SEND* и др.). Задается методом *setAction*() или конструктором *Intent*.

Данные – это объект URI, ссылающийся на данные, с которыми будет выполняться действие и (или) тип MIME этих данных. Тип данных определяется действием объекта *Intent*. Например, если действием является *ACTION_EDIT*, в данных должен содержаться URI документа, который требуется отредактировать. Чтобы задать только URI данных, вызывается *setData*(). Чтобы задать только тип MIME – *setType*(). При необходимости определить оба этих параметра – *setDataAndType*().

Категория – строка, содержащая прочие сведения о том, каким компонентом должна выполняться обработка этого объекта *Intent*. В объект *Intent* можно поместить любое количество описаний категорий, но большинству объектов *Intent* категория не требуется. Например, стандартные категории: *CATEGORY_BROWSABLE* (целевая активность позволяет запускать себя веб-браузером для отображения данных, указанных по ссылке); *CATEGORY_LAUNCHER* (активность является начальной для задачи). Указать категорию можно с помощью *addCategory*().

Дополнительные данные – это пары «ключ – значение», содержащие прочую информацию. Добавлять дополнительные данные можно с помощью методов *putExtra*(), каждый из которых принимает два параметра: имя и значение ключа. Можно создать объект *Bundle* со всеми дополнительными данными, затем вставить объект *Bundle* в объект *Intent* с помощью метода *putExtras*().

Класс *Intent* указывает много констант *EXTRA_** для стандартных типов данных. Если требуется объявить собственные дополнительные ключи, то в качестве префикса указывается имя пакета приложения.

Флаги, определенные в классе *Intent*, действуют как метаданные. Флаги должны указывать ОС, каким образом следует запускать активность (например, к какой задаче должна принадлежать активность и как с ней обращаться после запуска).

Возможна ситуация, когда на устройстве пользователя не будет никакого приложения, которое может откликнуться на неявный объект *Intent*. В этом случае вызов закончится неудачей, а работа приложения аварийно завершится. Чтобы проверить, будет ли получен объект *Intent*, необходимо вызвать метод *resolveActivity*() для объекта *Intent*. Если результатом будет значение, отличное от *null*, значит, имеется хотя бы одно приложение, которое способно откликнуться на объект *Intent*, поэтому можно вызывать *startActivity*().

Рассмотрим пример определения неявного объекта *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);
                }
            }
});
```

4.4. Настройка фильтров для объекта Intent

Чтобы указать, какие неявные объекты *Intent* может принимат приложение, следует объявить один или несколько фильтров *Intent* для каждого компонента приложения с помощью элемента *intent-filter* в файле манифеста. Система передаст неявный объект *Intent* приложению, только если он может пройти через один из фильтров.

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

Компонент приложения должен объявлять отдельные фильтры для каждой уникальной работы, которую он может выполнить. 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* с помощью одного или нескольких элементов: *action* – объявляет принимаемое действие, заданное в объекте *Intent*, в атрибу- те *name*; *data* – объявляет тип принимаемых данных; *category* – объявляет принимаемую категорию, заданную в объекте *Intent*, в атрибуте *name*:

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

        <data>
        </data>

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

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.

Чтобы объект *Intent* был доставлен компоненту, он должен пройти все три теста. Если он не будет соответствовать хотя бы одному из них, ОС не доставит этот объект *Intent* компоненту. Поскольку у компонента может быть несколько фильтров, объект *Intent*, который не проходит ни через один из них, может пройти через другой фильтр.

4.5. Разрешение объектов Intent

Когда система получает неявный объект *Intent* для запуска активности, она выполняет поиск путем сравнения объекта *Intent* с фильтрами по трем критериям: действие объекта *Intent*; данные объекта (структура URI и тип данных); категория объекта.

4.5.1. Тестирование действия

Для указания принимаемых действий объекта *Intent* фильтр 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 действия, объекту *Intent* будет нечему соответствовать, поэтому все объекты *Intent* не пройдут этот тест. Если в объекте *Intent* не указано действие, он пройдет тест.

Действия задаются константами действий:

– *ACTION_ANSWER* – открывает активность, которая связана с входящими звонками;

–*ACTION_CALL* – инициализирует обращение по телефону;

–*ACTION_DELETE* – запускает активность, с помощью которой можно удалить данные, указанные в пути URI внутри *Intent*;

– *ACTION_EDIT* – отображает данные для редактирования пользователем;

–*ACTION_INSERT* – открывает активность для вставки в *Cursor* нового элемента, указанного с помощью пути URI;

–*ACTION_HEADSET_PLUG* – подключение наушников;

–*ACTION_MAIN* – запускается как начальная активность задания;

– *ACTION_PICK* – загружает дочернюю *Activity*, позволяющую выбрать элемент из источника данных, указанный с помощью URI;

– *ACTION_SEARCH* – запускает активность для выполнения поиска;

– *ACTION_SEND* – загружает экран для отправки данных, указанных в намерении;

– *ACTION_SENDTO* – открывает активность для отправки сообщений контакту, указанному в пути URI, который передается через *Intent*;

– *ACTION_SYNC* – синхронизирует данные сервера с данными мобильного устройства;

– *ACTION_VIEW* – наиболее распространенное общее действие.

4.5.2. Тестирование категории

Для указания принимаемых категорий 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*, при этом он все равно пройдет тест. Поэтому объект *Intent* без категорий всегда пройдет этот тест, независимо от того, какие из них объявлены в фильтре.

Система Android автоматически применяет категорию *CATEGORY_DEFAULT* ко всем неявным объектам *Intent*, которые передаются в *startActivity*() и *startActivityForResult*(). Поэтому, если нужно, чтобы *Activity* принимала неявные объекты *Intent*, в ее фильтрах должна быть указана категория для *android.intent.category.DEFAULT*.

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

– *ALTERNATIVE* – 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 применяется в тех случаях, когда нужен список различных возможностей;

– *BROWSABLE* – действие доступно из браузера;

– *DEFAULT* – делает компонент обработчиком по умолчанию для действия, выполняемого с указанным типом данных внутри фильтра; –

GADGET – активность может запускаться внутри другой активности; – *LAUNCHER* – помещает *Activity* в окно для запуска приложений.

4.5.3. Тестирование данных

Для указания принимаемых данных объекта *Intent* фильтр 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. Имеются отдельные атрибуты – *scheme, host, port* и *path* – для каждой составной части URI: *scheme://host:port/path*.

Каждый из этих атрибутов является необязательным, но есть линейные зависимости: если схема не указана, узел игнорируется; если узел не указан, порт игнорируется; если не указана ни схема, ни узел, путь игнорируется.

4.6. Принцип работы фильтров

В целом весь алгоритм работает следующим образом. Android собирает список всех доступных фильтров из установленных пакетов.

Фильтры, которые не соответствуют действию или категории *Intent*, удаляются из списка. Совпадение происходит только в том случае, если фильтр содержит указанное действие (или если действие для него не задано).

Для категорий процесс соответствия более строгий. Фильтр должен включать в себя все категории, заданные в полученном *Intent*. Фильтр, для которого категории не указаны, может соответствовать только таким же объектом *Intent* (нет категорий).

Каждая часть пути URI из *Intent* сравнивается с тегом *data*. Если в фильтре указаны схема (протокол), сервер/принадлежность, путь или тип MIME, все эти значения проверяются на соответствие пути URI из *Intent*. При любом несовпадении фильтр будет удален из списка. Если в фильтре не указано ни одного параметра *data*, его действие будет распространяться на любые данные.

MIME – тип данных, который должен совпасть. При сравнении типов данных можно использовать маски, чтобы охватывать все подтипы. Если в фильтре указан тип данных, он должен совпасть с тем, который значится в *Intent*, 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.