**Мобильная разработка**

**Занятие 15. Контекст. Намерения**

**Контекст**

Конте́кст (от лат. contextus — «соединение», «связь») — законченный отрывок, общий смысл которого позволяет уточнить значение входящих в него отдельных слов, предложений, и т. п. «Говорить, опираясь на контекст» — значит придерживаться установившегося в разговоре уровня абстракции и использовать понятия заданного в нём семантического поля.

Рассмотрим пример — автобусный рейс, на который проданы билеты. У каждого пассажира свой билет, таким образом само понятие билета может существовать в контексте пассажира Иванова Коли или в контексте Петрова Васи. Мы можем говорить о билетах на автобусный рейс в целом — что они все проданы или проданы наполовину. Но когда мы коснемся какого-то конкретного билета, то про него мы уже сможем говорить в контексте какого-то конкретного человека, который этот билет купил. У Васи билет лежит в кармане и он помятый. А у Коли билет лежит в кошельке и непомят.

В программировании понятие контекст имеет те же свойства и семантическую нагрузку - задавать элементу переменные окружения, которые обуславливают его работу. Например, у нас есть компонент Button, который существует сам по себе только в коллекции компонент. Как только мы хотим создать какую-то конкретную кнопку, она будет существовать в контексте какого-то конкретного интерфейса конктретной активности. Если мы поместим кнопку на другую активность, то это будет уже совсем другая кнопка, существующая уже в другом контексте.

Context присутствует практически повсюду в Android приложении и является самой важной его частью, поэтому необходимо понимать, как правильно его использовать.

Контекст (Context) – это базовый абстрактный класс, реализация которого обеспечивается системой Android. Этот класс имеет методы для доступа к специфичным для конкретного приложения ресурсам и классам и служит для выполнения операций на уровне приложения, таких, как запуск активностей, отправка широковещательных сообщений, получение намерений и прочее. От класса Context наследуются такие крупные и важные классы, как Application, Activity и Service, поэтому все его методы доступны из этих классов, что является отличным примером применения наследования в Android-разработке.

Объект Context часто применяется в программировании под Android. Неправильное использование Context может легко привести к утечкам памяти в Android приложении.

Существует несколько разных типов контекста, поэтому давайте разберёмся, что каждый из них представляет из себя, как и когда их правильно использовать.

Методы, с помощью которых вы можете получить контекст:

* getApplicationContext (получить ссылку на объект приложения)
* getContext (внутри активности или сервиса получить ссылку на этот объект)
* getBaseContext (получить ссылку на базовый контекст)
* this (то же, что и getContext)

### Контекст приложения

Это singleton-экземпляр (единственный на всё приложение), и к нему можно получить доступ через функцию getApplicationContext(). Этот контекст привязан к жизненному циклу приложения. Контекст приложения может использоваться там, где вам нужен контекст, жизненный цикл которого не связан с текущим контекстом или когда вам нужно передать контекст за пределы Activity.

Например, если вам нужно создать какой-то объект для вашего приложения который должен «жить» на протяжении жизненного цикла приложения, не зависимо от отображаемых активностей, то вы должны этот объект связывать с контекстов всего приложения.

Таким образом, getApplicationContext() нужно использовать тогда, когда известно, что вам нужен контекст для чего-то, что может жить дольше, чем любой другой контекст, который есть в вашем распоряжении.

### Контекст Activity

Этот контекст доступен в Activity и привязан к её жизненному циклу. Получить его можно с помощью оператора getContext(). Контекст Activity следует использовать, когда вы передаете контекст в рамках Activity или вам нужен контекст, жизненный цикл которого привязан к текущему контексту.

Контекст другого контекста

В Java существует класс ContextWrapper, который позволяет на основе какого-то контекста создать еще один и переопределить в этот новом контексте какой-то метод. Так вот, если вам нужен доступ к первоначальному контексту из нового переопределенного контекста,то нужно использовать оператор getBaseContext ().

Важно понимать, что время жизни разных контекстов будет разное. Ссылка на переданный объект будет работать, пока будет жить этот объект. Поэтому в качестве контекста важно передать такую ссылку, которая будет рабочей на всём протяжении работы вызываемого метода.

Например, если вызвать сообщение с помощью Toast, используя разные context, то:

Сообщение умрёт вместе с активностью:

Toast.makeText(this, «Text», Toast.LENGTH\_SHORT).show();

Сообщение умрёт вместе с приложением:

Toast.makeText(getApplicationContext(), «Text «, Toast.LENGTH\_SHORT).show();

### Когда нельзя использовать getApplicationContext()?

* Это не полноценный контекст, поддерживающий всё, что может Activity. Некоторые вещи, которые вы попытаетесь сделать с этим контекстом, потерпят неудачу, в основном связанные с графическим интерфейсом.
* Могут появляться утечки памяти, если контекст из getApplicationContext() удерживается на каком-то объекте, который вы не очищаете впоследствии. Если же где-то удерживается контекст Activity, то как только Activity уничтожается сборщиком мусора, всё остальное тоже уничтожается. Объект Application остается на всю жизнь вашего процесса.

### Намерения

Люди в большинстве своем не блуждают по миру бесцельно: большинство из того, что они делают – от просмотра телевизора до написания приложений – имеет за собой какую-то цель или намерение.

Android работает практически также. Прежде чем заствить приложение выполнить какое-либо действие, ему необходимо опеределить цель или намерение этого действия.

Намерения (Intent) – это абстрактная концепция работы или функциональности, которая может быть выполнена вашим приложением в будущем. Иными словами, это что-то, что ваше приложение должно сделать. В основном намерения состоят из следующих элементов:

* Действия (Actions) – это то, чего намерению необходимо достичь, например, набрать номер, открыть ссылку, передать данные. Действие является просто строковой константой.
* Данные (Data) – это ресурсы, которыми оперирует намерение. Тип требуемых данных для намерения изменяется в зависимости от действия. Вы ведь не хотите, чтобы ваше намерения для набора номера пыталось получить номер из изображения.

Эта возможность объединять действия и данные позволяет Android точно знать, что намерение собирается делать и с чем оно должно работать.

Intent представляет собой объект описания операции, которую необходимо выполнить через систему Android. Т.е. необходимо сначала описать некоторую операцию в виде объекта Intent, после чего отправить её на выполнение в систему вызовом одного из методов активности Activity. Намерения могут применяться для:

1. Старт операции

В android-приложении один экран представлен компонентом Activity, описанном в файле манифеста. Если необходимо в приложении иметь несколько экранов (страниц/форм), то все они должны быть зарегистрированы в манифесте приложения. Переход от одной активности к другой выполняется с помощью объекта Intent, который передается методу startActivity(). Объект Intent наряду с описанием операции может дополнительно включать все необходимые данные для выполнения операции.

2. Запуск сервиса

Android-приложение может выполнять действия в фоновом режиме без пользовательского интерфейса с помощью определенного сервиса, в качестве которого используется компонент Service. Сервис можно стартовать для выполнения какого-либо действия, например, сбора сообщений от мессенжера. Для старта сервиса необходимо вызвать метод активности startService() и передать ему объект Intent.

3. Рассылка широковещательных сообщений

Широковещательное сообщение может принимать любое приложение android. Система генерирует различные широковещательные сообщения о системных событиях, например, о необходимости включения режима геолокации.

## Типы объектов Intent

Существует два типа объектов Intent : явные и неявные.

### Явные объекты Intent

Явные объекты Intent , как правило, используются для запуска компонента из собственного приложения, где известно наименование запускаемых классов и сервисов. Например, у вас есть две активности и вы из первой активности при нажатии кнопки запускаете вторую активность. Обе активности описаны в вашем проекте, поэтому для запуска второй активности вы в намерение передадите просто имя второй активности.

При использовании явного Intent указываются:

- контекст, предоставляющий нужную информацию об окружающей среде приложения;

- класс целевого компонента (будь то сервис, активность), который нужно будет запустить.

Существует три варианта работы с явными намерениями.

1 Переход из одного Activity в другой Activity.

2 Открытие и передача каких-либо данных в открываемую Activity.

3 Открытие и обратный возврат какого-то результата при закрытии открываемого Activity.

Рассмотрим каждый из них на примере. В проекте создадим две активности с полем и кнопкой. При нажатии на кнопку первой активности запускается вторая и в поле второй активности отображается строка, которая записана в поле первой активности. И аналогично при нажатии на кнопку во второй активности происходит заполнение текстового поля первой активности.

Переход из одного Activity в другой Activity.

Для этого нам нужно создать намерение, указать ему контекст для запуска и передать имя запускаемой активности.

*Intent intent = new Intent(getApplicationContext(), SecondActivity.class);*

*startActivity(intent);*

Стоит обратить внимание на то, как в системе определяется, что за компонент ему передан в качестве цели. В Android путеводителем для системы является файл AndroidManifest. При создании класса новой Activity в манифест-файле указывается имя класса. Если указать тот же класс в Intent, то система, просмотрев манифест-файл, обнаружит соответствие и покажет соответствующий Activity. Но если такого компонента не окажется в манифест-файле, или тип компонента не будет соответствовать совершаемому действию, система не сможет проделать нужные действия. Поэтому важно правильно создавать активность, чтобы информация о ней появилась в файле файл AndroidManifest.

Как это работает: система Android запускает соответствующую классу, указанному в интенте, активность B, вызвав ее метод onCreate() и передав ей объект Intent. Активность А при этом перешла в остановленное состояние (paused), но не уничтожена, то есть остается в памяти, а верней, в стеке активностей. Активность B находится в состоянии Activity Running. Если же активность B завершит работу, то активность, сохраненная на верху стека активностей, то есть активность A, будет переведена в рабочее состояние.

Открытие и передача каких-либо данных в открываемую Activity.

Как было сказано выше, намерение представляет собой объект обмена сообщениями между операциями (активностями, сервисами) Android. Поэтому логично, что в Intent можно вложить или извлечь данные при помощи методов putExtra / getExtra. Таким образом, Intent можно представить посылкой, которую Android доставляет от абонента A к абоненту B. Метод putextra получает в качестве параметров ключ, по которому значение можно извлечь, и само значение. Ключ имеет строковый тип, а значение может иметь либо примитивный тип, либо строковый тип, либо являться массивом, либо объектный тип, реализующий интерфейс Parcelable или Serializable.

*Intent intent = new Intent(this, SecondActivity.class);*

*String eText = "information to send";*

*i.putExtra("MyData", eText);*

*startActivity(intent);*

В запущенной активности данные извлекаются, здесь ключ для извлечения — строка «MyData»

String str = getIntent().getStringExtra("et");

Также через намерения можно передавать и полноценные объекты собственных классов. Однако для такой возможности класс этого объекта должен реализовать интерфейс Serializable или Parcelable. Что это такое и зачем оно нужно, мы рассмотрим попозже.

Открытие и обратный возврат какого-то результата при закрытии открываемого Activity.

Последний рассматриваемый аспект взаимодействия активностей при помощи намерения — возврат результатов из вызванной активности. В случае, когда ожидается возврат данных из запускаемой активности, необходимо использовать другой метод вызова активности. Это метод startActivityForResult. Первым аргументом ему передают подготовленный объект намерения, а вторым целочисленное значение requestCode. Это значение используют для идентификации операции, чтобы понять, на какую именно вызванную операцию был получен ответ в вызывающей активности.

*Intent intent = new Intent new Intent(this, SecondActivity.class);*

*startActivityForResult(intent, 1);*

По правилам хорошего кода, переменную requestCode не стоит передавать в виде какого-то непонятного числа. Правильнее завести глобальные константы для каждой активности и вместо неинформативной цифры указывать именно константу.

*startActivityForResult(intent,* *REQEST\_SECOND\_ACTIVITY);*

После выполнения нужных действий и задач вызванная активность завершает свою работу и возвращает результат рабботы с числовом виде или намерение с данными работы.

Добавим на вторую активность один чекбокс и еще одну обычную кнопку. Чекбокс назовем «Мой выбор», первую кнопку обозначим «Подтвердить», вторую добавленую кнопку «Отмена».

На первую кнопку напишем обработчик нажатия, который возвращает текст с выбранного чекбокса:

*String str = "";  
if(chck.isChecked())  
 str = (String) chck.getText();  
Intent i = new Intent();  
i.putExtra("ReturnData", str);  
setResult(RESULT\_OK, i);  
finish();*

При нажатии на вторую кнопку возвращается строка с информацией об отмене операции.

В данном случае объект Intenet создается для передачи результата обратно в родительскую Activity. Опять же своего рода посылка, только для передачи данных обратно. Методом setResult в качестве результата передаются код результата и намерение. Код результата нужен для обработки возвращаемых данных в первой активности.

Для того чтобы получить обратную передачу интента, в первом классе необходимо переопределить метод onActivityResult, который будет вызван, когда завершится выполнение вызванной активности (B). В этот же метод будут переданы requestCode (ранее переданный из A в B), resultCode (код возврата) и data, передаваемый обратно Intent:

*@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data ) {  
 if (requestCode == REQEST\_SECOND\_ACTIVITY) {  
 TextView tv = findViewById(R.id.textView);  
 switch (resultCode) {  
 case RESULT\_OK:  
 tv.setText(data.getStringExtra("ReturnData"));  
 break;  
 case RESULT\_CANCELED:  
 tv.setText(data.getStringExtra("ReturnData"));  
 break;  
 } } }*