Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С.Галущака»

|  |  |  |
| --- | --- | --- |
| Председатель цикловой комиссии по специальности УГС 09.00.00Информатика и вычислительная техника  \_\_\_\_\_О.О.Чекушкина  Протокол № \_\_ от  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г | Председатель цикловой комиссии по специальности УГС 09.00.00Информатика и вычислительная техника  \_\_\_\_\_О.О.Чекушкина  Протокол № \_\_ от  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г | Председатель цикловой комиссии по специальности УГС 09.00.00Информатика и вычислительная техника  \_\_\_\_\_О.О.Чекушкина  Протокол № \_\_ от  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г |

**Изменение элементов дизайна**

Методические указания к практическому занятию 8

Междисциплинарный курс: МДК.01.03Разработка мобильных приложений

Специальность: 09.02.07 Информационные системы и программирование

Разработал:

И.С. Климова

2021

1. **Цели**
   1. В ходе выполнения работы студенты осваивают:
      1. Общие компетенции, включающие в себя способность:

ОК 9 Использовать информационные технологии в профессиональной деятельности

* + 1. Профессиональные компетенции:

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ

* 1. В результате выполнения студенты:
     1. Усваивают знания:
* основные этапы разработки программного обеспечения;
* основные принципы технологии структурного и объектно-ориентированного программирования;
* основные принципы отладки и тестирования программных продуктов;
  + 1. Осваивают умения:
* осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
* создавать программу по разработанному алгоритму как отдельный модуль;
* выполнять отладку и тестирование программы на уровне модуля.

1. **Оборудование**

* компьютеры;
* среда разработки Android Studio.

1. **Форма организации** – фронтальная
2. **Инструктаж**
   1. Работа состоит из заданий, предусматривающих освоение приёмов работы всреде разработки AndroidStudio.
   2. При выполнении работы следует пользоваться методическими указаниями для каждого задания.
   3. Отчет оформляется во время проведения практического занятия в программе Microsoft Word добавляется в git-репозиторий рядом с проектом AndroidStudio.
   4. Выполнение работы 90 минут
3. **Порядок выполнения**
   1. Ознакомиться с теоретическим материалом.
   2. Ознакомиться с методическими рекомендациями.
   3. Запустить программу Android Studio.
   4. Найти местонахождения проекта на вашем ПК.
4. **Методические рекомендации**

Краткие теоретические сведения содержатся в приложении А к методическим указаниям, если в этом есть необходимость.

1. **Форма отчета**
   1. Титульный лист.
   2. Цель практической работы.
   3. Выполнить и описать ход работы по п.п. 5.
   4. Вывод о проделанной работе.
2. **Критерии оценки**
   1. При контроле и оценке результатов выполнения задания учитывается:

* полное выполнения практического задания;
* отсутствие ошибок в коде;
* соотношение полученного результата примеру итоговой работы.
  1. В основу оценки выполненных заданий положен принцип:

«Отлично» −выполнен полный объем заданий в соответствии с п.8.1;

«Хорошо» −выполнен полный объем заданий в соответствии с п.8.1, но допущены ошибки в одном из заданий;

«Удовлетворительно» −выполнен полный объем заданий в соответствии с п.8.1, но допущеныошибки в двух заданиях;

«Неудовлетворительно» −выполненные задания не соответствуют п.8.1, студентом не реализованы цели данной работы

1. **Содержание задания**
2. Практическим путём научиться размещать элементы и менять их свойства.
3. Разработать визуальную часть приложения в соответствии с макетом.
4. Реализовать взаимодействие с управляющими элементами на экране.

**Варианты**

1. <https://www.figma.com/file/jgKtacRQ8fXvVOCdUoMO7v/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-8?node-id=0%3A1>
2. <https://www.figma.com/file/xBma7jKl21QT1DmofFndrP/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-11>
3. <https://www.figma.com/file/P2DMvZplQi18bPqnT7dQV5/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-14>
4. <https://www.figma.com/file/BJ2AWwFdivME5nsAgBQa6C/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-9?node-id=0%3A1>
5. <https://www.figma.com/file/ncHPUJEmv38en4sZ4ZOOOC/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-10?node-id=0%3A1>
6. <https://www.figma.com/file/P2DMvZplQi18bPqnT7dQV5/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-14?node-id=0%3A1>
7. <https://www.figma.com/file/4Y2I0f0isi5c88RKS2vb5y/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-12?node-id=0%3A1>
8. <https://www.figma.com/file/w89B4V99v8D4Qjl2sYq0Mq/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-13?node-id=0%3A1>
9. <https://www.figma.com/file/jgKtacRQ8fXvVOCdUoMO7v/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-8?node-id=0%3A1>

**Приложение А**

Как правило, для определения визуального интерфейса в проектах под Android используются специальные файлы xml. Эти файлы являются ресурсами разметки и хранят определение визуального интерфейса в виде кода XML. Подобный подход напоминает создание веб-сайтов, когда интерфейс определяется в файлах html, а логика приложения - в коде javascript.

Объявление пользовательского интерфейса в файлах XML позволяет отделить интерфейс приложения от кода. Что означает, что мы можем изменять определение интерфейса без изменения кода java. Например, в приложении могут быть определены разметки в файлах XML для различных ориентаций монитора, различных размеров устройств, различных языков и т.д. Кроме того, объявление разметки в XML позволяет легче визуализировать структуру интерфейса и облегчает отладку.

Файлы разметки графического интерфейса располагаются в проекте в каталоге **res/layout**. По умолчанию при создании проекта с пустой activity уже есть один файл ресурсов разметки **activity\_main.xml**, который может выглядеть примерно так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?xmlversion="1.0"encoding="utf-8"?>  <androidx.constraintlayout.widget.ConstraintLayoutxmlns:android="<http://schemas.android.com/apk/res/android>"      xmlns:app="<http://schemas.android.com/apk/res-auto>"      xmlns:tools="<http://schemas.android.com/tools>"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      tools:context=".MainActivity">      <TextView          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Hello World!"          app:layout\_constraintBottom\_toBottomOf="parent"          app:layout\_constraintLeft\_toLeftOf="parent"          app:layout\_constraintRight\_toRightOf="parent"          app:layout\_constraintTop\_toTopOf="parent"/>  </androidx.constraintlayout.widget.ConstraintLayout> |

В файле определяются все графические элементы и их атрибуты, которые составляют интерфейс. При создании разметки в XML следует соблюдать некоторые правила: каждый файл разметки должен содержать один корневой элемент, который должен представлять объект View или ViewGroup.

В данном случае корневым элементом является элемент ConstraintLayout, который содержит элемент TextView.

Как правило, корневой элемент содержит определение используемых пространств имен XML. Например, в коде по умолчанию в ConstraintLayout мы можем увидеть такие атрибуты:

|  |  |
| --- | --- |
| 1  2  3 | xmlns:android="<http://schemas.android.com/apk/res/android>"  xmlns:app="<http://schemas.android.com/apk/res-auto>"  xmlns:tools="<http://schemas.android.com/tools>" |

Каждое пространство имен задается следующим образом: xmlns:префикс="название\_ресурса". Например, в

|  |  |
| --- | --- |
|  | xmlns:android="<http://schemas.android.com/apk/res/android>" |

Названиересурса (или URI - Uniform Resource Indicator) - "http://schemas.android.com/apk/res/android". И этот ресурс сопоставляется с префиксом android (xmlns:android). То есть через префикс мы сможем ссылаться на функциональность этого пространства имен.

Каждое пространство имен определяет некоторую функциональность, которая используется в приложении, например, предоставляют теги и атрибуты, которые необходимые для построения приложения.

* xmlns:android="http://schemas.android.com/apk/res/android": содержит основные атрибуты, которые предоставляются платформой Android, применяются в элементах управления и определяют их визуальные свойства (например, размер, позиционирование). Например, в коде ConstraintLayout используется следующий атрибут из пространства имен "http://schemas.android.com/apk/res/android":

|  |  |
| --- | --- |
|  | android:layout\_width="match\_parent" |

* xmlns:app="http://schemas.android.com/apk/res-auto": содержит атрибуты, которые определены в рамках приложения. Например, в коде TextView:

|  |  |
| --- | --- |
|  | app:layout\_constraintBottom\_toBottomOf="parent" |

* xmlns:tools="http://schemas.android.com/tools": применяется для работы с режиме дизайнера в AndroidStudio

Это наиболее распространенные пространства имен. И обычно каждый корневой элемент (не обязательно только ConstraintLayout) их содержит. Однако, если вы не планируете пользоваться графическим дизайнером в AndroidStudio и хотите работать целиком в коде xml, то соответственно смысла в пространстве имен "http://schemas.android.com/tools" нет, и его можно убрать.

При компиляции каждый XML-файл разметки компилируется в ресурс View. Загрузка ресурса разметки осуществляется в методе Activity.onCreate. Чтобы установить разметку для текущего объекта activity, надо в метод setContentView() в качестве параметра передать ссылку на ресурс разметки.

|  |  |
| --- | --- |
|  | setContentView(R.layout.activity\_main); |

Для получения ссылки на ресурс в коде java необходимо использовать выражение R.layout.[название\_ресурса]. Название ресурса layout будет совпадать с именем файла, поэтому чтобы использовать файл activity\_main.xml в качестве источника визуального интерфейса, можно определить следующий код в классе MainActivity:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | packagecom.example.viewapp;  importandroidx.appcompat.app.AppCompatActivity;  importandroid.os.Bundle;  publicclassMainActivity extendsAppCompatActivity {      @Override      protectedvoidonCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);            // загрузкаинтерфейсаизфайла activity\_main.xml          setContentView(R.layout.activity\_main);      }  } |

### Добавление файла layout

Но у нас может быть и несколько различных ресурсов layout. Как правило, каждый отдельный класс Activity использует свой файл layout. Либо для одного класса Activity может использоваться сразу несколько различных файлов layout.

К примеру, добавим в проект новый файл разметки интерфейса. Для этого нажмем на папку **res/layout** правой кнопкой мыши и в появившемся меню выберем пункт **New ->LayoutResource File**:

После этого в специальном окошке будет предложено указать имя и корневой элемент для файла layout:

В качестве названия укажем **second\_layout**. Все остальные настройки оставим по умолчанию:

* в поле **Rootelement** указывается корневой элемент. Поумолчаниюэто **androidx.constraintlayout.widget.ConstraintLayout**.
* поле **Sourceset** указывает, куда помещать новый файл. По умолчанию это **main** - область проекта, с которой мы собственно работаем при разаботке приложения.
* поле **Directorymain** указывает папку в рамках каталога, выбранного в предыдущей опции, в который собственно помещается новый файл. По умолчанию для файлов с разметкой интерфейса это **layout**.

После этого в папку **res/layout** будет добавлен новый файл **second\_layout.xml**, с которым мы можем работать точно также, как и с activity\_main.xml. В частности, откроем файл second\_layout.xml и изменим его содержимое следующим образом:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <?xmlversion="1.0"encoding="utf-8"?>  <androidx.constraintlayout.widget.ConstraintLayout      xmlns:android="<http://schemas.android.com/apk/res/android>"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent">      <TextView          android:id="@+id/header"          android:text="Welcome to Android"          android:textSize="26sp"          android:layout\_width="match\_parent"          android:layout\_height="match\_parent"/>  </androidx.constraintlayout.widget.ConstraintLayout> |

Здесь определено текстовое поле TextView, которое имеет следующие атрибуты:

* android:id - идентификатор элемента, через который мы сможем ссылаться на него в коде. В записи android:id="@+id/header" символ @ указывает XML-парсеру использовать оставшуюся часть строки атрибута как идентификатор. А знак + означает, что если для элемента не определен id со значением header, то его следует определить.
* android:text - текст элемента - на экран будет выводиться строка "WelcometoAndroid".
* android:textSize - высота шрифта (здесь 26 единиц)
* android:layout\_width - ширинаэлемента. Значение "match\_parent" указывает, что элемент будет растягиваться по всей ширине контейнера ConstraintLayout
* android:layout\_height - высотаэлемента. Значение "match\_parent" указывает, что элемент будет растягиваться по всей высоте контейнера ConstraintLayout

Применим этот файл в качестве определения графического интерфейса в классе MainActivity:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | packagecom.example.viewapp;  importandroidx.appcompat.app.AppCompatActivity;  importandroid.os.Bundle;  publicclassMainActivity extendsAppCompatActivity {      @Override      protectedvoidonCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.second\_layout);      }  } |

Файл интерфейса называется second\_layout.xml, поэтому по умолчанию для него будет создаваться ресурс R.layout.second\_layout. Соответственно, чтобы его использовать, мы передаем его в метода setContentView. В итоге мы увидим на экране следующее:

### Получение и управлене визуальными элементами в коде

Выше определенный элемент TextView имеет один очень важный атрибут - id или идентификатор элемента. Этот идентификатор позволяет обращаться к элементу, который определен в файле xml, из кода Java. Например, перейдем к классу MainActivity и изменим его код:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | packagecom.example.viewapp;    importandroidx.appcompat.app.AppCompatActivity;  importandroid.os.Bundle;  importandroid.widget.TextView;    publicclassMainActivity extendsAppCompatActivity {        @Override      protectedvoidonCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          // устанавливаемвкачествеинтерфейсафайл second\_layout.xml          setContentView(R.layout.second\_layout);            // получаемэлемент textView          TextView textView = findViewById(R.id.header);          // переустанавливаемунеготекст          textView.setText("Hello from Java!");      }  } |

С помощью метода setContentView() устанавливается разметка из файла *second\_layout.xml*.

Другой важный момент, который стоит отметить - получение визуального элемента TextView. Так как в его коде мы определили атрибут android:id, то через этот id мы можем его получить.

Для получения элементов по id класс Activity имеет метод **findViewById()**. В этот метод передается идентификатор ресурса в виде **R.id.[идентификатор\_элемента]**. Этот метод возвращает объект View - объект базового класса для всех элементов, поэтому результат метода еще необходимо привести к типу TextView.

Далее мы можем что-то сделать с этим элементом, в данном случае изменяем его текст.

Причем что важно, получение элемента происходит после того, как в методе setContentView была установлена разметка, в которой этот визуальный элемент был определен.

И если мы запустим проект, то увидим, что TextView выводит новый текст:

Здесь можно посмотреть учебные материалы по данной практической:

<https://metanit.com/java/android/4.1.php>