Лекция 7

Ресурсы приложения

1. Ресурсы приложения – это разные статические файлы, значения, с которыми вам придется работать очень часто.
2. К ним относятся объекты как картинки, строки, цвета, анимации, стили и т.п.
3. И такие вещи в Андроид нужно держать за пределами Java-кода программы.
4. Все ресурсы хранятся в папке res/ проекта.
5. У папки res/ есть определенная структура. В нем могут находиться только папки:
    1. layout – файлы интерфейса
    2. values – разные значения: строки, размеры, цвета
    3. menu – описания меню
    4. drawable – изображения
    5. anim – описания анимаций
    6. raw – необработанные любые файлы
    7. xml – любые другие xml-файлы
    8. mipmap – иконки приложения

Класс R

1. Система автоматически создает идентификаторы ресурсов и использует их в файле R.java
2. С помощью этого класс R, в коде программы мы можем ссылаться на ресурсы приложения:
    1. R.layout.main
    2. R.drawable.image1
    3. R.string.app_name
    4. И т.д.

Строковые ресурсы

1. Чтобы каждый раз не писать один и тот же текст, его нужно сохранить в ресурсах.
2. Строковые ресурсы обычно создаются в файле res/values/string.xml
3. Но их можно создавать в любом другом файле в папке res/values/
4. Общая структура файлов со значениями(values) такая:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        …. Тут разные значения
</resources>
```

5. Пример строкового ресурса:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name”>Сырное приложение</string>
   <string name=”login">Войти</string>
</resources>
```

Строковые ресурсы

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name”>Сырное приложение</string>
   <string name="cheese_name">Passendale</string>
</resources>
```

1. **Для получения этой строки из кода, используем R.string.название_ресурса, например:**

```
cheeseNameText.setText(R.string.cheese_name);
```

2. **После этого в** cheeseNameText **появится текст «Passendale»**
3. **Чтобы получить саму строку используем метод getSting():**

```
String cheese = getString(R.string.cheese_name);
```

4. **После этого в переменной cheese будет храниться текст “Passendale”**
5. **Чтобы использовать текстовый ресурс в layout файлах, используем выражение** @string/название_ресурса:

```
<TextView
   android:id="@+id/cheese_name_text"
   android:text="@string/cheese_name"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content" />
```

Массивы строк в ресурсах

1. Также в ресурсах можно хранить массивы строк. Они так же должны находиться в файле ресурсов в папке res/values/:

```
<string-array name="names">
    <item>Chechil</item>
    <item>Mozarella</item>
    <item>Passendale</item>
</string-array>
```

2. Программно получаем их так:

```
String[] names = getResources().getStringArray(R.array.names);
```

3. И наш файл res/values/string.xml будет примерно таким:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
    <string name="app_name">Cheesesquare</string>
    <string name="cheese_name">Passendale</string>
    <string-array name="names">
        <item>Chechil</item>
        <item>Mozarella</item>
        <item>Passendale</item>
    </string-array>
</resources>
```

Цвета в ресурсах

1. **Цвета хранятся также как строки, но обычно создаются в файле res/values/color.xml.**
2. **Цвета задаются в шестнадцатеричном формате #RRGGBB:**
3. **Или в формате #AARRGGBB где AA – это прозрачность.**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="grey">#5b5b5b</color>
    <color name="transparent_red">#88ff0000</color>
</resources>
```

4. **Цвета в layout файлах можно использовать как фоновый цвет и как цвет текста через @color/:**

```
<Button
    android:background="@color/transparent_red"
    android:textColor="@color/grey"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Цвета в ресурсах

1. **Для доступа к цветам в ресурсах используем выражение R.color.название_цвета.**
2. **Для получения значения этого цвета вызываем метод: getResources().getColor():**

```
int grey_color = getResources().getColor(R.color.grey);
```

3. **Меняем цвет кнопки:**

```
changeButton.setTextColor(getResources().getColor(R.color.grey));
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="grey">#5b5b5b</color>
    <color name="transparent_red">#88ff0000</color>
</resources>
```

Размеры в ресурсах

1. **Размеры также задаются как строки и цвета, обычно в файле res/values/dimens.xml:**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="detail_backdrop_height">256dp</dimen>
    <dimen name="card_margin">16dp</dimen>
    <dimen name="fab_margin">16dp</dimen>
    <dimen name="list_item_avatar_size">40dp</dimen>
</resources>
```

2. **Размеры можно использовать в layout файлах, в любом месте где можно использовать числа или размеры, используя выражение @dimen/**

```
<TextView
    android:id="@+id/cheese_name_text"
    android:text="@string/cheese_name»
    android:layout_margin=“@diman/fab_margin”
    android:layout_width="@dimen/list_item_avatar_size"
    android:layout_height="@dimen/detail_backdrop_height" />
```

Изображения в ресурсах

1. Изображения в ресурсах можно представить двумя способами:
   1. В графическом файле, типа png, jpg, bmp
   2. Описать с помощью xml файла
2. Чтобы использовать обычные картинки, их нужно скопировать в папку res/drawable/
3. К изображениям в layout файлах нужно обращаться через @drawable/ :

```
<Button
   android:id="@+id/another_cheese"
   android:background="@drawable/cheese_5"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content" />
<ImageView
   android:src="@drawable/ic_forum"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />
```

Изображения в ресурсах

1. **Программно изображения получаем так:**

```
Drawable image =  getResources().getDrawable(R.drawable.cheese_1);
```

2. **Потом можем установить как фон:**

```
cheeseNameText.setBackgroundDrawable(image);
```

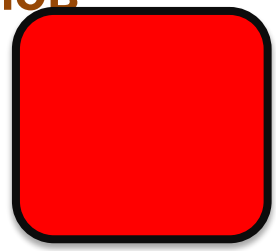
3. **Либо сразу установить по id ресурса:**

```
cheeseNameText.setBackgroundResource(R.drawable.cheese_1);
imageView.setImageResource(R.drawable.cheese_3);
```

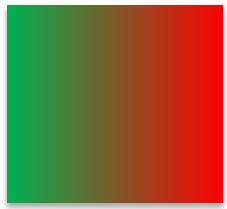
Shape

1. Изображения разных фигур можно самому рисовать используя ограниченный набор инструментов.
2. Для этого нужно создать xml-файл в папке res/drawable/ с корневым элементов <shape>. В shape нужно добавить дочерние элементы чтобы определить свойства фигуры:
    1. <solid> - цвет фона заполняется одним указанным цветом
    2. <stroke> - границы фигуры: ширина и цвет
    3. <size> - можно задать размеры фигуры
    4. <corners> - задание округления углов
    5. <gradient> - градиентовый фон

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <corners android:radius="6dp"/>
   <solid android:color="#FF0000"/>
   <stroke android:color="#000" android:width="1dp"/>
   <size android:width="50dp" android:height="50dp"/>
</shape>
```



```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <corners android:radius="6dp"/>
   <solid android:color="#FF0000"/>
   <gradient android:startColor="#00FF00" android:endColor="#FF0000"/>
</shape>
```



Ресурсы стилей

1. **Вы наверное заметили что в layout-файлах приходится для каждого элемента указывать очень много параметров и в некоторых местах повторять одно и тоже.**
2. **Чтобы этого избежать, такие параметры можно определить в стиле, и в элементах вместо всех этих параметров указывать только стиль.**

```
<android.support.design.widget.CollapsingToolbarLayout
    android:id="@+id/collapsing_toolbar"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    app:contentScrim="?attr/colorPrimary"
    app:expandedTitleMarginEnd="64dp"
    app:expandedTitleMarginStart="48dp"
    app:layout_scrollFlags="scroll|exitUntilCollapsed">

    <ImageView
        android:id="@+id/backdrop"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        android:scaleType="centerCrop"
        app:layout_collapseMode="parallax" />
```

Ресурсы стилей

1. **Стили создаются в папке res/values/styles/ обычно. Пример:**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="Widget.CardContent" parent="android:Widget">
      <item name="android:paddingLeft">16dp</item>
      <item name="android:paddingRight">16dp</item>
      <item name="android:paddingTop">24dp</item>
      <item name="android:paddingBottom">24dp</item>
      <item name="android:orientation">vertical</item>
   </style>
</resources>
```

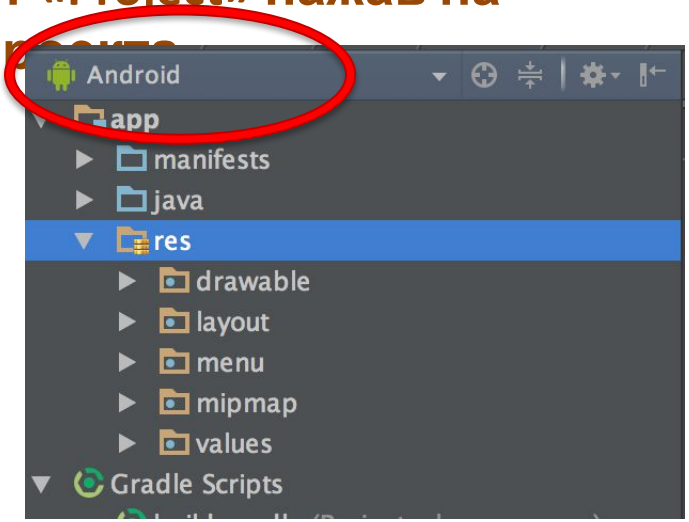
2. **И после этого мы можем применить стиль к View элементу:**

```
<Button style="@style/Widget.CardContent"/>
```

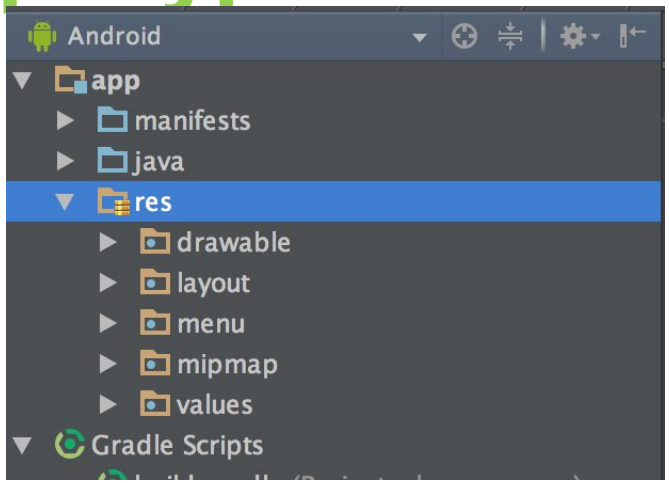
3. **И нам не придется в layout – файле указывать все его параметры.**
4. **Стили могут быть унаследованы от других стилей. Для указания родителя используется аргумент parent.**

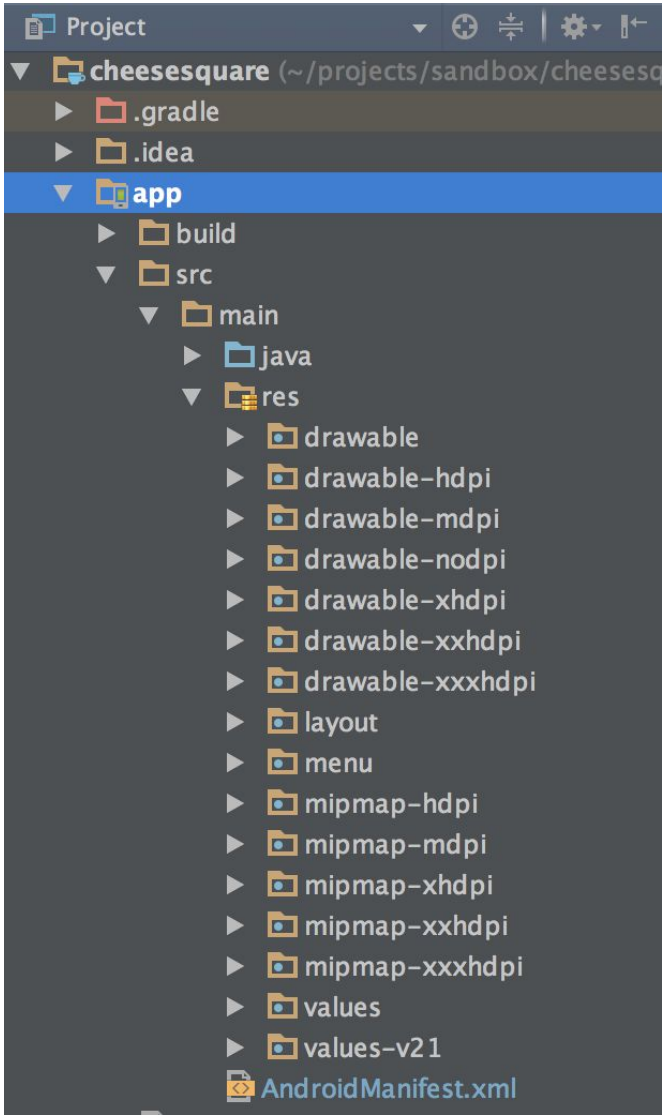
Динамический выбор ресурсов

1. Использование ресурсов позволяет динамически выбрать нужный ресурс в программе в зависимости от текущего языка и аппаратных конфигураций.
2. Для этого нужно задать определенную структуру каталогов в ресурсах.
3. Альтернативные значение задаются с использованием дефиса в названиях папок.
4. Так как стандартный вид Android Studio не позволяет увидеть эту структуру, нужно переключить дерево файлов проекта в режим Project.
5. Для этого нужно выбрать пункт «Project» нажав на заголовок окошка с файлами пр[illegible]



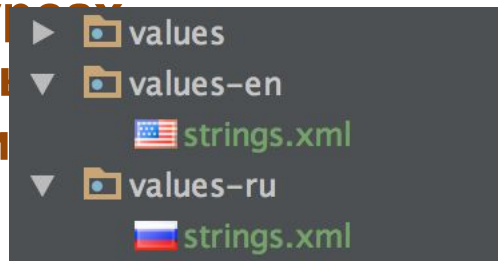
Динамический выбор ресурсов





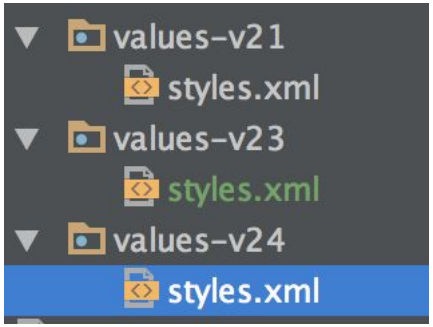
Динамический выбор ресурсов

1. Используя суффикс с кодом языка, можно задать разные значения в ресурсах, для разных языков.
2. Например в разных строковых ресурсах можно хранить тексты на разных яз
3. После этого по одному и тоже имени ресурса, у нас будет разный текст в зависимости от текущего языка устройства.



4. Так делаются локализация приложений. Т.е. Поддержка языков.

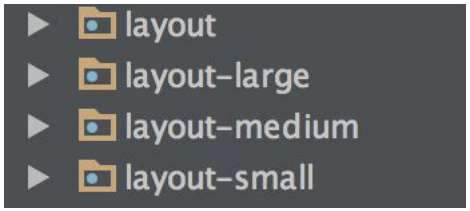
5. Можно также используя суффикс с версией Андроида хранить разные значения для разных версия андроида, чтобы один и тот же элемент по разному отображался на разных версиях Андроида



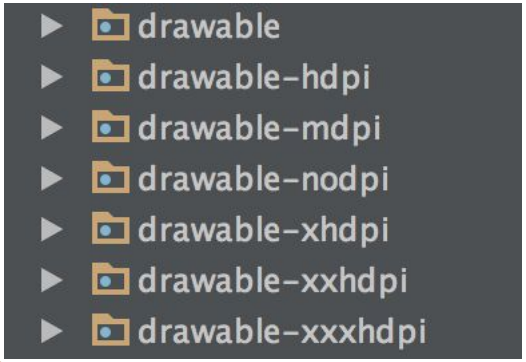
Динамический выбор ресурсов

1. Можно задавать разные значения в зависимости от ориентация экрана, используя суффиксы –land, -port.

2. Под разные размеры экранов
   -large – большие
   -medium – средние
   -small – маленькие



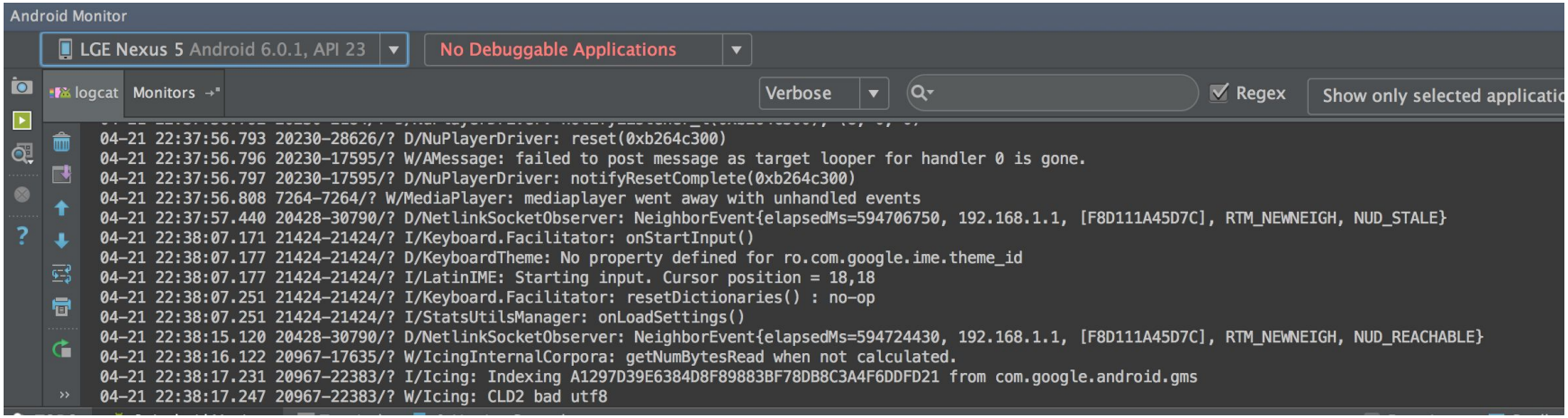
3. Под разные плотности экранов:
   -ldpi, mdpi, hdpi, xhdpi, xxhdpi



4. И еще много других вариантов использования суффиксов для динамического выбора ресурсов для разных конфигураций

Логи приложения

1. Во время выполнения мы можем видеть разные отладочные сообщения или другую полезную информацию о состоянии программы, об ошибках и тп.
2. Это делается с помощью логирования.
3. Чтобы увидеть лог-сообщения, нужно открыть окно Android Monitor и вкладку Logcat.
4. Каждое сообщение показывается со временем его создания.



Логи приложения

1. Чтобы в приложение писать в лог используется стандартный класс Log.
2. У класса Log есть несколько похожих методов:
   1. Log.d() – печатает просто любой текст
   2. Log.e() – печатает текст ошибки, отображается красным
   3. Log.i() – печатает просто текст, информационное
3. Все три метода принимает два параметра, первое это любой тэг, второй – сам текст сообщения. Например:

```
Log.i("API", "API удачно инициализирован!");
Log.e("API", "Не удалось подключиться!");
```

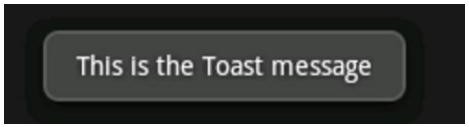
4. Важно знать, что логи предназначены только для разработчиков, потому что пользователь их не видит.
5. У объектов Exception есть метод printStackTrace(), который печатает весь стек вызовов. Здесь можно увидеть какие методы были вызваны, когда

Показ сообщения пользователю

1. Во время выполнения программы, часто приходится сообщать пользователю некоторые сообщения об ошибке или о каком нибудь другом событии.
2. Для этого можно использовать всплывающие сообщения класса Toast:

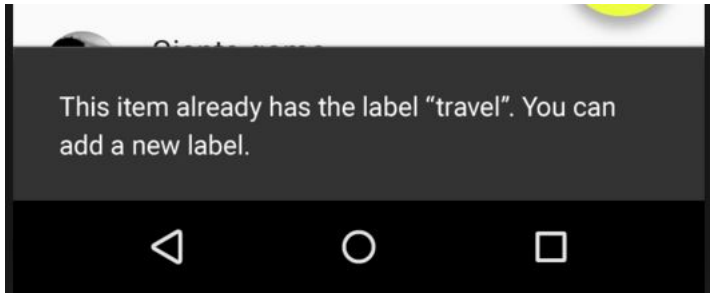
```
Toast.makeText(this, "Нажата кнопка ОК", Toast.LENGTH_LONG).show();
```

```
Toast.makeText(this, "Ошибка", Toast.LENGTH_SHORT).show();
```



3. Также можно использовать SnackBar из Material дизайна из библиотеки Support:

```
SnackBar.Builder snackBar = new SnackBar.Builder(SplashScreen.this);
snackBar.withMessage(getString(R.string.internet_connection_error))
        .withActionMessage(getString(R.string.retry))
        .withTextColorId(R.color.gs_orange_color)
        .withDuration(SnackBar.PERMANENT_SNACK)
        .withOnClickListener(new SnackBar.OnMessageClickListener() {
            @Override
            public void onMessageClick(Parcelable token) {
                SplashScreen.this.snackBar.clear();
                startPreparingActions();
            }
        });
SplashScreen.this.snackBar = snackBar.show();
```



Варианты лаб работы 2

1. Простой список контактов
2. Часы с таймером и секундомером
3. Игра «Крестики и нолики»
4. Игра «Пятнашки»
5. Игра «Сапер»
6. Журнал событий со стастистикой
7. Записная книжка
8. Словарь
9. Тестировщик
10. Справочник лекарственных средств(брать с сайта)
11. Любой другой проект

Спасибо!