

Разработка приложений для ОС Android. Файл манифеста AndroidManifest.xml

Файл манифеста AndroidManifest.xml предоставляет системе основную информацию о программе. Каждое приложение должно иметь свой файл AndroidManifest.xml. Редактировать файл манифеста можно вручную, изменяя XML-код или через визуальный редактор Manifest Editor, который позволяет осуществлять визуальное и текстовое редактирование файла манифеста приложения.

Разработка приложений для ОС Android.

Файл манифеста AndroidManifest.xml

Назначение файла:

- описывает компоненты приложения – Activities, Services, Broadcast receivers и Content providers;
- содержит список необходимых разрешений для обращения к защищенным частям API и взаимодействия с другими приложениями;
- объявляет разрешения, которые сторонние приложения обязаны иметь для взаимодействия с компонентами данного приложения;
- объявляет минимальный уровень API Android, необходимый для работы приложения;
- перечисляет связанные библиотеки.

Разработка приложений для ОС Android.

Файл манифеста AndroidManifest.xml

Корневым элементом манифеста является `<manifest>`.

Элемент `<application>` является основным элементом манифеста и содержит множество дочерних элементов, определяющих структуру и работу приложения.

Порядок расположения элементов, находящихся на одном уровне, произвольный. Все значения устанавливаются через атрибуты элементов.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="startandroid.ru.second">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Second"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Second">

        <activity
            android:name=".MainActivity"
            android:label="Second"
            android:theme="@style/Theme.Second.NoActionBar">

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

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

</manifest>
```

Файл манифеста AndroidManifest.xml

По умолчанию создается элемент

<manifest> с атрибутами:

- xmlns:android определяет пространство имен Android.
- package определяет уникальное имя пакета приложения.

Элемент <application> - один из основных, содержащий описание компонентов приложения. Содержит дочерние элементы (<activity>, <service>, <receiver>, <provider> и другие), которые объявляют каждый из компонентов, входящих в состав приложения. В манифесте может быть только один элемент <application>.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="startandroid.ru.second">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Second"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Second">

        <activity
            android:name=".MainActivity"
            android:label="Second"
            android:theme="@style/Theme.Second.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Файл манифеста AndroidManifest.xml

Атрибут `<permission>` объявляет разрешение, которое используется для ограничения доступа к определенным компонентам или функциональности данного приложения. В этой секции описываются права, которые должны запросить другие приложения для получения доступа к приложению.

Приложение может также защитить свои собственные компоненты (Activities, Services, Broadcast receivers и Content providers) разрешениями. Оно может использовать любое из системных разрешений, определенных Android или объявленных другими приложениями, а также может определить свои собственные разрешения.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="com.test.someapplication">
4
5    <uses-permission android:name="android.permission.INTERNET" />
6    <uses-permission android:name="android.permission.READ_CONTACTS" />
7    <uses-permission android:name="android.permission.BLUETOOTH" />
8    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
9    <uses-permission android:name="android.permission.CAMERA" />
10   <uses-permission android:name="android.permission.SEND_SMS" />
11
12   <application
13     ...
14   ></application>
15
16 </manifest>
```

Файл манифеста AndroidManifest.xml

- <uses-permission> запрашивает разрешения, которые приложению должны быть предоставлены системой для его нормального функционирования. Разрешения предоставляются во время установки приложения, а не во время его работы.

Наиболее распространённые разрешения:

- INTERNET – доступ к интернету
- READ_CONTACTS – чтение (но не запись) данных из адресной книги пользователя
- WRITE_CONTACTS – запись (но не чтение) данных в адресную книгу пользователя
- RECEIVE_SMS – обработка входящих SMS
- ACCESS_FINE_LOCATION – точное определение местонахождения при помощи GPS

Файл манифеста AndroidManifest.xml

- `<uses-sdk>` позволяет объявлять совместимость приложения с указанной версией (или более новыми версиями API) платформы Android. Уровень API, объявленный приложением, сравнивается с уровнем API системы мобильного устройства, на который устанавливается данное приложение.

Атрибуты:

`android:minSdkVersion` определяет минимальный уровень API, требуемый для работы приложения. Система Android будет препятствовать тому, чтобы пользователь установил приложение, если уровень API системы будет ниже, чем значение, определенное в этом атрибуте.

`android:maxSdkVersion` позволяет определить самую позднюю версию, которую готова поддерживать программа.

`targetSdkVersion` позволяет указать платформу, для которой разрабатывалось и тестировалось приложение.

Файл манифеста AndroidManifest.xml

- `<uses-configuration>` указывает требуемую для приложения аппаратную и программную конфигурацию мобильного устройства.

Спецификация используется, чтобы избежать инсталляции приложения на устройствах, которые не поддерживают требуемую конфигурацию. Если приложение может работать с различными конфигурациями устройства, необходимо включить в манифест отдельные элементы `<uses-configuration>` для каждой конфигурации.

Файл манифеста AndroidManifest.xml

- `<uses-feature>` объявляет определенную функциональность, требующуюся для работы приложения. Таким образом, приложение не будет установлено на устройствах, которые не имеют требуемую функциональность. Например, приложение могло бы определить, что оно требует камеры с автофокусом. Если устройство не имеет встроенную камеру с автофокусом, приложение не будет установлено.

Возможные атрибуты:

`android.hardware.camera` – требуется аппаратная камера.

`android.hardware.camera.autofocus` – требуется камера с автоматической фокусировкой.

- `<supports-screens>` определяет разрешение экрана, требуемое для функционирования приложения. По умолчанию современное приложение с уровнем API 4 или выше поддерживает все размеры экрана и должно игнорировать этот элемент.

Ресурсы

- В Android принято хранить такие объекты, как изображения, строковые константы, цвета, анимацию, стили и тому подобное, за пределами исходного кода. Система поддерживает хранение ресурсов во внешних файлах. Внешние ресурсы легче поддерживать, обновлять и редактировать.
- В основном, ресурсы хранятся в виде XML-файлов в каталоге `res` с подкаталогами `values`, `drawable-ldpi`, `drawable-mdpi`, `drawable-hdpi`, `layout`. Но также бывают еще два типа ресурсов: `raw` и `assets`.
- Для удобства система создает идентификаторы ресурсов и использует их в файле `R.java` (класс `R`, который содержит ссылки на все ресурсы проекта), что позволяет ссылаться на ресурсы внутри кода программы. Статический класс `R` генерируется на основе заданных ресурсов и создается во время компиляции проекта. Так как файл `R` генерируется автоматически, то нет смысла его редактировать вручную, потому что все изменения будут утеряны при повторной генерации.
- В общем виде ресурсы представляют собой файл (например, изображение) или значение (например, заголовок программы), связанные с создаваемым приложением. Удобство использования ресурсов заключается в том, что их можно изменять без повторной компиляции или новой разработки приложения.
- Самыми распространенными ресурсами являются, пожалуй, строки (`string`), цвета (`color`) и графические рисунки (`bitmap`).

Ресурсы

Основные ресурсы Android-приложения:

- Цвета /res/colors/ - идентификатор цвета, указывающий на цветовой код. Строки /res/strings/ Строковые ресурсы. В их число также входят строки в формате java и html.
- Меню /res/menus/ - меню в приложении можно задать как XML-ресурсы.
- Параметры /res/values/ - представляет собой параметры или размеры различных элементов.
- Изображения /res/drawable/ - ресурсы-изображения. Поддерживает форматы JPG, GIF, PNG (самый предпочтительный) и другие. Каждое изображение является отдельным файлом. Система также поддерживает stretchable images, в которых можно менять масштаб отдельных элементов, а другие элементы оставлять без изменений.
- Отрисовываемые цвета /res/values/ или /res/drawable/ - представляет цветные прямоугольники, которые используются в качестве фона основных отрисовываемых объектов, например точечных рисунков.
- Анимация /res/anim/ - Android может выполнить простую анимацию на графике или на серии графических изображений.
- Произвольные XML-файлы /res/xml/ - в Android в качестве ресурсов могут использоваться произвольные XML-файлы.
- Произвольные необработанные ресурсы /res/raw/ - любые некомпиллированные двоичные или текстовые файлы, например, видео.

Ресурсы

Помимо изображений в каталоге `res/drawable` могут храниться ресурсы простых геометрических фигур. Вот лишь некоторые из возможных атрибутов:

- `android:shape` задает тип фигуры: `rectangle` (прямоугольник), `oval` (овал), `line` (линия), `ring` (окружность);
- `<corners>` создает закругленные углы для прямоугольника;
- `<gradient>` задает градиентную заливку для фигуры; в Android можно создавать три типа градиентов: `Linear` (линейный), `Radial` (радиальный) и `Sweep` (разверточный);
- `<size>` задает размеры фигуры;
- `<solid>` задает сплошной цвет для фигуры.

Анимация в Android бывает двух видов:

- `Frame Animation` – кадровая анимация, традиционная анимация при помощи быстрой смены последовательных изображений, как на киноплёнке.
- `Tween Animation` – анимация преобразований может выполняться в виде ряда простых преобразований: изменение позиции (класс `TranslateAnimation`), размера (`ScaleAnimation`), угла вращения (`RotateAnimation`) и уровня прозрачности (`AlphaAnimation`). Команды анимации определяют преобразования, которые необходимо произвести над объектом. Преобразования могут быть последовательными или одновременными. Последовательность команд анимации определяется в XML-файле (предпочтительно) или в программном коде.

Ресурсы

В Android имеется еще один каталог, в котором могут храниться файлы, предназначенные для включения в пакет – `/assets`. Это не ресурсы, а просто необработанные файлы. Этот каталог находится на том же уровне, что и `/res`. Для файлов, располагающихся в `/assets`, в `R.java` не генерируются идентификаторы ресурсов. Для их считывания необходимо указать путь к файлу.

Путь к файлу является относительным и начинается с `/assets`. Этот каталог, в отличие от подкаталога `res/`, позволяет задавать произвольную глубину подкаталогов и произвольные имена файлов.

Разметка

В Android-приложениях, пользовательский интерфейс построен на View и ViewGroup объектах. Класс ViewGroup является основой для подкласса Layout (разметка).

Разметка (также используются термины компоновка или макет) хранится в виде XML-файла в папке /res/layout. Это сделано для того, чтобы отделить код от дизайна, как это принято во многих технологиях (HTML и CSS, Visual Studio и Expression Blend). Кроме основной компоновки для всего экрана, существуют дочерние компоновки для группы элементов. По сути, компоновка – это некий визуальный шаблон для пользовательского интерфейса приложения, который позволяет управлять элементами, их свойствами и расположением. В своей практике вам придется познакомиться со всеми способами размещения.

Создавая пользовательский интерфейс в XML-файле, можно отделить дизайн приложения от программного кода. Можно изменять пользовательский интерфейс в файле разметки без необходимости изменения программного кода. Например, можно создавать XML-разметки для различных ориентаций экрана мобильного устройства (portrait, landscape), размеров экрана и языков интерфейса. Впрочем, элементы интерфейса можно создавать и программно, когда это необходимо.

Разметка

Каждый файл разметки должен содержать только один корневой элемент компоновки, который должен быть объектом View или ViewGroup. Внутри корневого элемента можно добавлять дополнительные объекты разметки или дочерние элементы интерфейса, чтобы постепенно формировать иерархию элементов, которую определяет создаваемая разметка.

Существует несколько стандартных типов разметок:

- `FrameLayout` является самым простым типом разметки. Обычно это пустое пространство на экране, которое можно заполнить только дочерним объектом View или ViewGroup. Все дочерние элементы `FrameLayout` прикрепляются к верхнему левому углу экрана. В разметке `FrameLayout` нельзя определить различное местоположение для дочернего объекта View. Последующие дочерние объекты View будут просто рисоваться поверх предыдущих представлений, частично или полностью затеняя их, если находящийся сверху объект непрозрачен
- `LinearLayout` выравнивает все дочерние объекты в одном направлении – вертикально или горизонтально. Направление задается при помощи атрибута ориентации `android:orientation`. Все дочерние элементы помещаются в стек один за другим, так что вертикальный список представлений будет иметь только один дочерний элемент в строке независимо от того, насколько широким он является. Горизонтальное расположение списка будет размещать элементы в одну строку с высотой, равной высоте самого высокого дочернего элемента списка.

Разметка

- `TableLayout` позиционирует свои дочерние элементы в строки и столбцы. `TableLayout` не отображает линии обрамления для рядов, столбцов или ячеек. `TableLayout` может иметь ряды с разным количеством ячеек. При формировании разметки таблицы некоторые ячейки при необходимости можно оставлять пустыми. `TableLayout` удобно использовать, например, при создании логических игр типа Судоку, Крестики-Нолики и тому подобных.
- `RelativeLayout` позволяет дочерним элементам определять свою позицию относительно родительского представления или относительно соседних дочерних элементов.

Все описываемые разметки являются подклассами `ViewGroup` и наследуют свойства, определенные в классе `View`.

Разметки ведут себя как элементы управления, и их можно группировать. Расположение элементов управления может быть вложенным. Например, можно использовать `RelativeLayout` в `LinearLayout` и так далее. Однако, слишком большая вложенность элементов управления вызывает проблемы с производительностью.