Verstehen der AndroidManifest.xml-Datei

Wenn du dich in die Android-Entwicklung einarbeitest, ist eine der ersten Dateien, die du begegnen wirst, die `AndroidManifest.xml`. Sie ist wie der Bauplan deiner App –sie teilt dem Android-System alles mit, was es über deine Anwendung wissen muss, bevor sie überhaupt läuft. Heute werden wir eine Beispielmanifestdatei einer App namens “Flower”(Paketname: `com.lzw.flower`) zerlegen und ihre wichtigsten Komponenten, Konzepte und Muster erkunden.

Was ist die AndroidManifest.xml? Die `AndroidManifest.xml`-Datei ist eine erforderliche Konfigurationsdatei für jede Android-App. Sie befindet sich im Stammverzeichnis deines Projekts und gibt wesentliche Informationen wie den Paketnamen der App, Berechtigungen, Komponenten (z. B. Aktivitäten) und Hardware-/Software-Funktionen an, die sie benötigt. Denke daran als die Ausweiskarte der App, die das Android-Betriebssystem liest.

Lass uns die Beispieldatei Schritt für Schritt durchgehen.

Die Struktur des Manifests

Hier ist das Manifest, mit dem wir arbeiten (etwas vereinfacht für die Lesbarkeit):

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lzw.flower"
    android:versionCode="8"
    android:versionName="1.5.2">

    <uses-sdk android:minSdkVersion="14" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application
```

```
        android:label="@string/app_name"
        android:icon="@drawable/icon128"
        android:name=".base.App"
        android:theme="@style/AppTheme">

        <activity android:name=".deprecated.CameraActivity" android:screenOrientation="landscape" />
        <activity android:name=".base.SplashActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".draw.DrawActivity" android:screenOrientation="landscape" />
        <activity android:name=".result.ResultActivity" android:screenOrientation="landscape" />
        <activity android:name=".material.MaterialActivity" android:screenOrientation="landscape" />
        <activity android:name=".activity.PhotoActivity" android:screenOrientation="landscape" />
        <activity android:name=".activity.LoginActivity" android:screenOrientation="portrait" />
    </application>
</manifest>
```

Jetzt zerlegen wir es in seine Kernabschnitte und erklären die dahinterstehenden Konzepte.

---

1. Das Stamm-`<manifest>`-Element

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lzw.flower"
    android:versionCode="8"
    android:versionName="1.5.2">
```

- **`xmlns:android`**: Dies definiert den XML-Namensraum für Android-spezifische Attribute. Es ist eine Standardvorlage, die du in jedem Manifest sehen wirst.
- **`package`**: Dies ist der eindeutige Bezeichner für deine App (z. B. `com.lzw.flower`). Es ist auch der Standardnamensraum für deine Java/Kotlin-Klassen.
- **`android:versionCode`**: Eine interne ganze Zahl (hier `8`), die verwendet wird, um Versionen zu verfolgen. Sie wird mit jedem Update erhöht.
- **`android:versionName`**: Eine für Menschen lesbare Versionszeichenfolge (hier `1.5.2`), die den Benutzern angezeigt wird.

Konzept: Das `<manifest>`-Tag setzt die Identität und Versionierung der App fest, sodass das System weiß, mit welcher App es es zu tun hat und wie es Updates verarbeiten soll.

---

2. SDK-Version mit `<uses-sdk>`

```
<uses-sdk android:minSdkVersion="14" />
```

- `android:minSdkVersion`: Gibt die minimale Android-API-Ebene an, die die App unterstützt. API 14 entspricht Android 4.0 (Ice Cream Sandwich).

Konzept: Dies stellt die Kompatibilität sicher. Geräte, die Android-Versionen unter 4.0 ausführen, können diese App nicht installieren. Es gibt keine `targetSdkVersion` oder `maxSdkVersion` hier, aber sie könnten hinzugefügt werden, um die Kompatibilität weiter zu verfeinern.

---

3. Berechtigungen mit `<uses-permission>`

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Diese App fordert mehrere Berechtigungen an: - `CAMERA`: Zum Zugriff auf die Gerätekamera. - `WRITE_EXTERNAL_STORAGE`: Zum Speichern von Dateien (z. B. Fotos) auf externem Speicher. - `INTERNET`: Für Netzwerkzugriff. - `ACCESS_NETWORK_STATE`: Um die Netzwerkverbindung zu überprüfen. - `READ_PHONE_STATE`: Zum Zugriff auf Geräteinformationen (z. B. IMEI). - `ACCESS_WIFI_STATE`: Um den Wi-Fi-Status zu überprüfen.

Konzept: Android verwendet ein Berechtigungssystem, um die Privatsphäre und Sicherheit der Benutzer zu schützen. Diese Deklarationen teilen dem System (und dem Benutzer) mit, welche sensiblen Funktionen die App benötigt. Ab Android 6.0 (API 23) erfordern gefährliche Berechtigungen (wie `CAMERA`) auch Laufzeitanfragen im App-Code.

---

4. Funktionen mit `<uses-feature>`

```
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

- `android.hardware.camera`: Gibt an, dass die App eine Kamera benötigt.

- `android.hardware.camera.autofocus`: Gibt an, dass die Kamera Autofokus unterstützen muss.

Konzept: Im Gegensatz zu Berechtigungen filtern `<uses-feature>`-Tags die App im Google Play Store. Wenn ein Gerät keine Kamera oder keinen Autofokus hat, wird die App nicht einmal als installierbar angezeigt, es sei denn, diese sind als optional mit `android:required="false"` gekennzeichnet.

---

5. Das `<application>`-Element

```
<application
    android:label="@string/app_name"
    android:icon="@drawable/icon128"
    android:name=".base.App"
    android:theme="@style/AppTheme">
```

- `android:label`: Der Name der App, der aus einer Zeichenkettenressource (`@string/app_name`) entnommen wird.
- `android:icon`: Das Symbol der App, das auf eine Zeichnungsressource (`@drawable/icon128`) verweist.
- `android:name`: Eine benutzerdefinierte Application-Klasse (`.base.App`), die die Android-`Application`-Klasse für appweite Logik erweitert.
- `android:theme`: Das Standardvisuelle Thema für die App (`@style/AppTheme`).

Konzept: Das `<application>`-Tag definiert appweite Einstellungen. Ressourcen wie `@string` und `@drawable` werden in `res/`-Ordnern gespeichert, was Wiederverwendbarkeit und Lokalisierung fördert.

---

6. Aktivitäten mit `<activity>`

Das Manifest listet mehrere Aktivitäten auf, die die Bildschirme der App sind:

Beispiel 1: Splash-Bildschirm (Starter-Aktivität)

```
<activity
    android:name=".base.SplashActivity"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar.Fullscreen">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

- `android:name`: Der Klassenname (`.base.SplashActivity`).
- `intent-filter`: Markiert dies als den Einstiegspunkt der App (`MAIN`-Aktion + `LAUNCHER`-Kategorie), sodass es im App-Launcher des Geräts erscheint.
- `android:theme`: Ein Vollbildthema ohne Aktionsleiste.

Muster: Die Starter-Aktivität ist ein häufiger Ausgangspunkt, oft ein Splash-Bildschirm oder ein Startbildschirm.

Beispiel 2: Kamera-Aktivität

```
<activity
    android:name=".deprecated.CameraActivity"
    android:screenOrientation="landscape">
```

- `android:screenOrientation`: Erzwingt Querformat.
- `.deprecated`: Deutet darauf hin, dass diese Aktivität möglicherweise veraltet ist, aber immer noch enthalten ist.

Muster: Aktivitäten legen oft die Ausrichtung für spezifische Anwendungsfälle fest (z. B. Kamera-Apps funktionieren besser im Querformat).

Weitere Aktivitäten Das Manifest listet weitere Aktivitäten wie `DrawActivity`, `ResultActivity`, `PhotoActivity` usw. mit ähnlichen Mustern auf: - Die meisten sind im Querformat, was auf eine visuelle oder medienorientierte App hinweist. - Einige überschreiben das Standardthema der App (z. B. `Theme.Holo.Light`).

Konzept: Aktivitäten sind die Bausteine der Benutzeroberfläche einer Android-App. Jedes `<activity>`-Tag registriert einen Bildschirm beim System.

Wichtige Muster in diesem Manifest

1. **Medienzentrierte Gestaltung**: Berechtigungen und Funktionen für Kamera, Speicher und Autofokus deuten auf eine Foto- oder Zeichnungs-App hin (vielleicht zur Blumenidentifikation, gegeben den Paketnamen `com.lzw.flower`).
2. **Steuerung der Ausrichtung**: Der häufige Gebrauch von `android:screenOrientation="landscape"` deutet auf eine Fokussierung auf visuelle Aufgaben hin.
3. **Modulare Aktivitäten**: Mehrere Aktivitäten (`CameraActivity`, `DrawActivity`, `ResultActivity`) deuten auf einen mehrstufigen Arbeitsablauf hin.

4. **Ressourcennutzung**: Verweise auf `@string`, `@drawable` und `@style` zeigen eine saubere, wartbare Struktur.

Fazit

Die `AndroidManifest.xml` ist mehr als nur eine Konfigurationsdatei –sie ist ein Fenster in den Zweck und das Verhalten einer App. In diesem Fall scheint "Flower"eine Medien-App mit Kamerafunktionen, Zeichnungsfunktionen und Netzwerkfähigkeiten zu sein, möglicherweise zum Hochladen oder Verarbeiten von Bildern. Durch das Verständnis ihrer Komponenten –Berechtigungen, Funktionen und Aktivitäten –kannst du sehen, wie Android-Apps strukturiert sind und wie du deine eigene gestalten kannst.

Möchtest du etwas Ähnliches bauen? Beginne mit einem klaren Zweck (wie Blumenidentifikation), definiere deine Berechtigungen und Funktionen und plane deine Aktivitäten. Das Manifest wird alles zusammenfügen!