Comprendre le fichier AndroidManifest.xml

Si vous vous lancez dans le développement Android, l'un des premiers fichiers que vous rencontrerez est le `AndroidManifest.xml`. C'est comme le plan de votre application —il indique au système Android tout ce qu'il doit savoir sur votre application avant même qu'elle ne s'exécute. Aujourd'hui, nous allons disséquer un fichier manifeste d'exemple d'une application appelée "Flower"(nom du package : `com.lzw.flower`) et explorer ses composants clés, concepts et motifs.

Qu'est-ce que le `AndroidManifest.xml` ? Le fichier `AndroidManifest.xml` est un fichier de configuration requis pour chaque application Android. Il se trouve dans le répertoire racine de votre projet et déclare des informations essentielles comme le nom du package de l'application, les permissions, les composants (par exemple, les activités), et les fonctionnalités matérielles/logicielles dont elle a besoin. Pensez-y comme à la carte d'identité de l'application que le système d'exploitation Android lit.

Passons en revue l'exemple étape par étape.

La Structure du Manifest

Voici le manifeste que nous allons utiliser (légèrement simplifié pour la lisibilité) :

```
<?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>
```

Maintenant, décomposons-le en ses sections principales et expliquons les concepts qui se cachent derrière.

---

1. L'élément racine `<manifest>`

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

- `xmlns:android` : Cela définit l'espace de noms XML pour les attributs spécifiques à Android. C'est une balise standard que vous verrez dans chaque manifeste.
- `package` : C'est l'identifiant unique de votre application (par exemple, `com.lzw.flower`). C'est également le namespace par défaut pour vos classes Java/Kotlin.
- `android:versionCode` : Un entier interne (ici, `8`) utilisé pour suivre les versions. Il s'incrémente avec chaque mise à jour.
- `android:versionName` : Une chaîne de version lisible par l'homme (ici, `1.5.2`) affichée aux utilisateurs.

Concept : La balise `<manifest>` définit l'identité et la version de l'application, assurant que le système sait quelle application il traite et comment gérer les mises à jour.

---

2. La version SDK avec `<uses-sdk>`

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

- `android:minSdkVersion` : Spécifie le niveau minimum de l'API Android que l'application prend en charge. L'API 14 correspond à Android 4.0 (Ice Cream Sandwich).

Concept : Cela assure la compatibilité. Les appareils exécutant des versions d'Android inférieures à 4.0 ne peuvent pas installer cette application. Il n'y a pas de `targetSdkVersion` ou `maxSdkVersion` ici, mais ils pourraient être ajoutés pour affiner davantage la compatibilité.

---

3. Les permissions avec `<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" />
```

Cette application demande plusieurs permissions : - `CAMERA` : Pour accéder à l'appareil photo du dispositif. - `WRITE_EXTERNAL_STORAGE` : Pour enregistrer des fichiers (par exemple, des photos) dans le stockage externe. - `INTERNET` : Pour l'accès réseau. - `ACCESS_NETWORK_STATE` : Pour vérifier la connectivité réseau. - `READ_PHONE_STATE` : Pour accéder aux informations du dispositif (par exemple, IMEI). - `ACCESS_WIFI_STATE` : Pour vérifier l'état Wi-Fi.

Concept : Android utilise un système de permissions pour protéger la vie privée et la sécurité des utilisateurs. Ces déclarations indiquent au système (et à l'utilisateur) quelles fonctionnalités sensibles l'application nécessite. Après Android 6.0 (API 23), les permissions dangereuses (comme `CAMERA`) nécessitent également des demandes en temps réel dans le code de l'application.

---

4. Les fonctionnalités avec `<uses-feature>`

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

- `android.hardware.camera` : Déclare que l'application nécessite un appareil photo.
- `android.hardware.camera.autofocus` : Spécifie que l'appareil photo doit prendre en charge l'autofocus.

Concept : Contrairement aux permissions, les balises `<uses-feature>` filtrent l'application sur le Google Play Store. Si un dispositif manque d'un appareil photo ou d'autofocus, l'application ne sera même pas affichée comme installable, sauf si elles sont marquées comme optionnelles avec `android:required="false"`.

---

5. L'élément `<application>`

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

- `android:label` : Le nom de l'application, tiré d'une ressource de chaîne (`@string/app_name`).
- `android:icon` : L'icône de l'application, faisant référence à une ressource graphique (`@drawable/icon128`).
- `android:name` : Une classe Application personnalisée (`.base.App`), qui étend la classe `Application` d'Android pour la logique globale de l'application.
- `android:theme` : Le thème visuel par défaut pour l'application (`@style/AppTheme`).

Concept : La balise `<application>` définit les paramètres globaux de l'application. Les ressources comme `@string` et `@drawable` sont stockées dans les dossiers `res/`, favorisant la réutilisabilité et la localisation.

---

6. Les activités avec `<activity>`

Le manifeste liste plusieurs activités, qui sont les écrans d'interface utilisateur de l'application :

Exemple 1 : Écran de démarrage (Activité de lancement)

```
<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` : Le nom de la classe (`.base.SplashActivity`).
- `intent-filter` : Marque ceci comme le point d'entrée de l'application (`MAIN` action + `LAUNCHER` category), donc il apparaît dans le lanceur d'applications du dispositif.
- `android:theme` : Un thème en plein écran sans barre d'action.

Motif : L'activité de lancement est un point de départ courant, souvent un écran de démarrage ou un écran d'accueil.

Exemple 2 : Activité de l'appareil photo

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

- `android:screenOrientation` : Force le mode paysage.
- `.deprecated` : Suggère que cette activité pourrait être obsolète mais toujours incluse.

Motif : Les activités imposent souvent une orientation pour des cas d'utilisation spécifiques (par exemple, les applications d'appareil photo fonctionnent mieux en mode paysage).

Autres activités Le manifeste liste plus d'activités comme `DrawActivity`, `ResultActivity`, `PhotoActivity`, etc., avec des motifs similaires : - La plupart sont en mode paysage, suggérant une application axée sur le visuel ou les médias. - Certaines remplacent le thème par défaut de l'application (par exemple, `Theme.Holo.Light`).

Concept : Les activités sont les blocs de construction de l'interface utilisateur d'une application Android. Chaque balise `<activity>` enregistre un écran avec le système.

Motifs clés dans ce manifeste

1. **Conception axée sur les médias** : Les permissions et fonctionnalités pour l'appareil photo, le stockage et l'autofocus suggèrent une application photo ou de dessin (peut-être pour identifier des fleurs, étant donné le nom du package `com.lzw.flower`).
2. **Contrôle de l'orientation** : Une utilisation intensive de `android:screenOrientation="landscape"` suggère une concentration sur des tâches visuelles.
3. **Activités modulaires** : Plusieurs activités (`CameraActivity`, `DrawActivity`, `ResultActivity`) indiquent un flux de travail en plusieurs étapes.
4. **Utilisation des ressources** : Les références à `@string`, `@drawable` et `@style` montrent une structure propre et maintenable.

Conclusion

Le `AndroidManifest.xml` est plus qu'un simple fichier de configuration —c'est une fenêtre sur le but et le comportement d'une application. Dans ce cas, "Flower"semble être une application multimédia avec des fonctionnalités d'appareil photo, des fonctionnalités de dessin et des capacités réseau, probablement pour télécharger ou traiter des images. En comprenant ses composants —permissions, fonctionnalités et activités —vous pouvez voir comment les applications Android sont structurées et comment concevoir les vôtres.

Vous voulez construire quelque chose de similaire ? Commencez avec un but clair (comme l'identification des fleurs), définissez vos permissions et fonctionnalités, et planifiez vos activités. Le manifeste les réunira toutes !