

Android Project Localization

HUHG 2009-4-26

本文主要介绍 Android 项目的本地化方法和过程.

Build Android Project

在完成一个应用开发后, 首先要将该项目导出成一个安装文件, 也就是.apk 文件, 这个时候导出的文件是没有经过签名的, 涉及签名的部分可以查看其他相关资料, 这里只对构建项目的过程进行分析.

如下图建立了一个项目, 完成开发后, 利用 Export the unsigned apk 可以导出该应用的未签名安装文件.

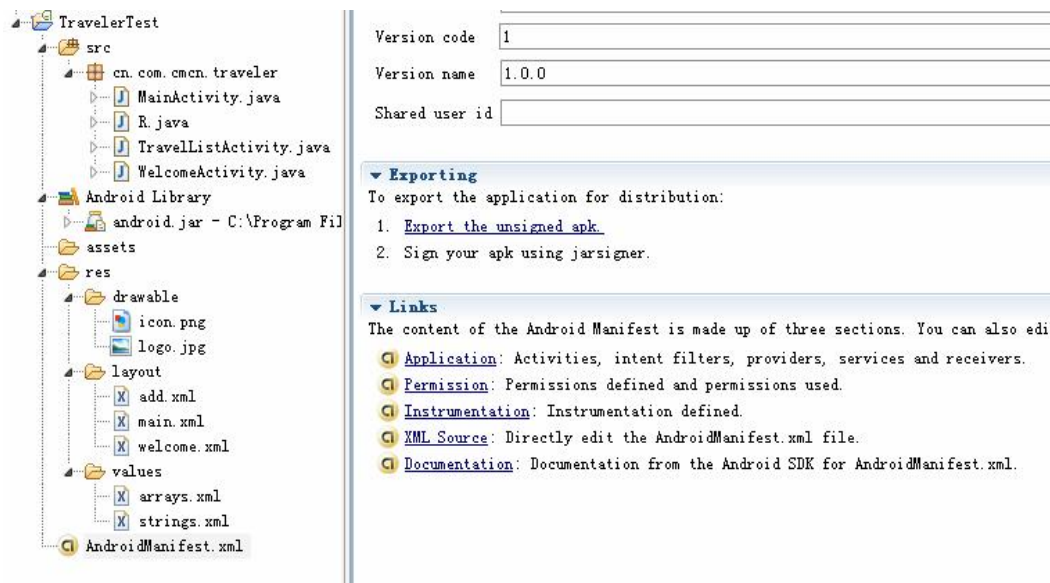


图 1 项目信息

导出未签名的安装文件经解压缩后, 可以包含文件信息如图 2 所示.



图 2 解压缩文件信息

注意上述文件中的所有.xml 文件均已经经过编译, 成为二进制文件, 因此如果我们直接打开的话, 看到的已经和应用的原始文件不同.

应用构建分析

由于在开发工具中直接导出了该安装文件, 我们看不到背后的处理过程, 因此只能通过其他途径去了解项目的构建过程.



图 3 包含文件信息

图 3 显示了一个经过签名的应用安装文件所包含的文件信息,可以看到我们导出的未经签名的应用包含了相同的文件信息,只是缺少了签名文件.并且可以看到签名信息是保存在 META-INF 文件夹之中的.

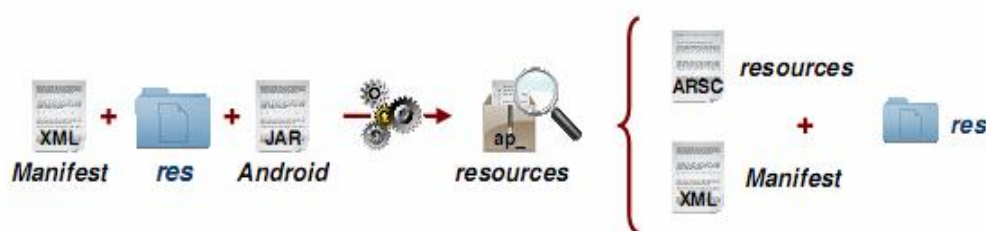


图 4 resources.arsc 构建过程

图 4 显示了文件 resources.arsc 的构建过程,可以看到 resources.arsc 文件的主要来源为 AndroidManifest.xml和 res 资源文件夹下的内容,还包含了应用用到的.jar 文件信息.

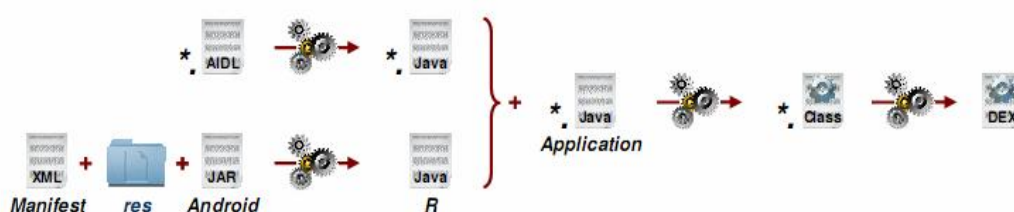


图 5 classes.dex 构建过程

图 5 显示了文件 classes.dex 的构建过程,可以看到 classes.dex 包含了所有的类文件信息,其中包括了系统自动产生的 R.java 的资源类信息.

Localize Android Project

应用本地化就是将原来只支持一种语言的应用本地化之后,可以支持多种语言,这样应用就可以在不同的语言地区得以应用.

通过学习与研究,应用本地化主要有以下两种方式:

1. 在原来的资源文件上进行修改

通过用特殊编辑器打开原来的资源文件,将所有字符信息整理出来,然后通过判断识别那些信息为应用中的字符信息,判断之后对相应字符翻译并替换,所有字符替换成功后保存为新的资源文件,然后替换掉原来的资源文件.

2. 通过应用还原生成新的资源文件

通过上面的构建分析,可以反向思考,如果能通过给定的.apk 应用文件,将相关文件还原,然后生成新的资源文件,就可以完成应用的本地化处理.这样做的好处就是本地化过程中知道了变量名称等信息,可以直接对某个变量单独进行替换,不会出现第一种方式中的由于混淆等情况出现的错误本地化.

2.1 分析资源文件信息

下面通过 AAPT 分析应用的资源文件信息,如图 6 所示.

资源文件信息分析如下:

首先为校验信息,检验文本是否存在错误.

其次为包的组信息,该应用包含一个包文件信息,随后列出了包文件的具体信息.

最后为资源文件中的所有资源信息,包括以下几种类型:

Type 0 该应用不含该类型内容,通过对比,该类型对应为应用中用到的 R.attr 信息.

Type 1 显示的信息为资源文件中 res/drawable 中的所有文件信息,对应为应用中用到的 R.drawable 信息.

Type 2 显示的信息为资源文件中 res/layout 中的所有文件信息,对应为应用中设

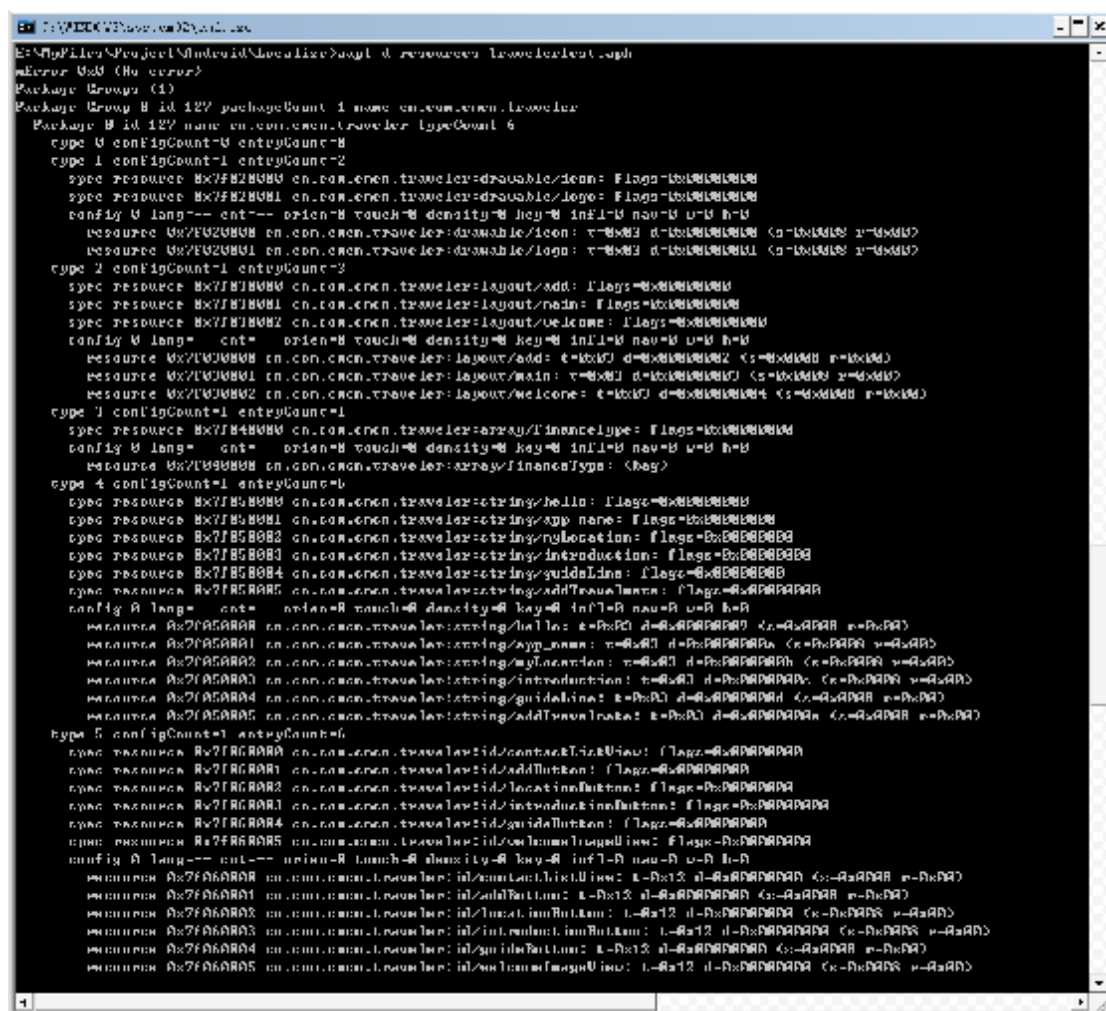


图 6 资源文件分析

置当前显示窗口的 R.layout 信息。

Type 3 显示的信息为资源文件中 res/values/arrays.xml 中的文件信息,对应为应用中寻找字符串数组时使用的 R.array 信息。

Type 4 显示的信息为资源文件中 `res/values/strings.xml` 中的文件信息，对应为 `res/layout` 中用到的字符信息，如 `@string/stringid` 方式。

Type 5 显示的信息为资源文件中 `res/layout` 中的所有组件索引信息，如按钮，列表等的索引，对应为应用中寻找组件时使用的 `R.id` 信息。

从以上分析可以得出，资源文件 `resources.arsc` 包含的就是 `AndroidManifest.xml` 和 `res` 文件夹下的所有文件信息，这样就可以通过还原上述文件信息，然后再根据还原的文件进行本地化，这样不但可以汉化的比较轻松简单，还可以了解原应用的组织结构，并且类的结构信息也可以还原出来。

2.2 还原资源文件过程

在开发应用时,所有的资源文件在增加,修改,删除等操作时都会自动产生一个索引信息,该信息保存了类文件 R.java 中,这样我们就能通过分析 R.java 文件来获得我们需要的信息。

2.2.1 查看应用配置文件

首先来看如何还原 AndroidManifest.xml文件,找到该文件,然后利用 AXMLPrinter2.jar 分析后得出以下信息:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0.0"
    package="cn.com.cmcn.traveler"
>
    <application
        android:label="@7F050001"
        android:icon="@7F020000"
    >
        <activity
            android:label="@7F050001"
            android:name=".MainActivity"
        >
            <intent-filter>
                <action
                    android:name="android.intent.action.MAIN"
                >
                </action>
                <category
                    android:name="android.intent.category.LAUNCHER"
                >
                </category>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```

        </activity>
        <uses-library
            android:name="com.google.android.maps"
        >
    </uses-library>
    //...省略信息
</application>
<uses-permission
    android:name="android.permission.INTERNET"
>
</uses-permission>
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION"
>
</uses-permission>
<uses-permission
    android:name="android.permission.READ_CONTACTS"
>
</uses-permission>
</manifest>

```

通过上述文件信息可以看到,原始的配置信息已经完好的还原出来,其中关于资源文件中内容的信息应该用标识信息表示了出来。这样如果我可以再通过其他文件内容,将该标识信息的原来内容找出来并替换掉,则将得到完整的应用配置文件信息。

2.2.2 查看资源类文件

所有的资源文件在 R.java 中建立了资源信息,该类里为每一种资源类型定义了不同的标识信息,以供我们在程序里使用资源文件。因此,我们要想办法把该文件信息得到,看该文件能给我们提示什么信息。

在.apk 文件中包含了这样一个文件, classes.dex 文件,这个文件是所有类文件的信息,而 R.java 只是其中的一个资源类,因此我们需要对 classes.dex 文件进行分析。我们通过 ddx1.4.jar 分析该文件,并生成所有类文件的机器码信息。

```

E:\MyFiles\Project\Android\Localize\TravelerTest>java -jar ddx1.4.jar -d
E:\MyFiles\Project\Android\Localize\TravelerTest classes.dex
Processing cn/com/cmcn/traveler/WelcomeActivity
Processing cn/com/cmcn/traveler/TravelListActivity$1
Processing cn/com/cmcn/traveler/TravelListActivity
Processing cn/com/cmcn/traveler/MainActivity$1
Processing cn/com/cmcn/traveler/MainActivity$3
Processing cn/com/cmcn/traveler/MainActivity$2
Processing cn/com/cmcn/traveler/R$array
Processing cn/com/cmcn/traveler/MainActivity
Processing cn/com/cmcn/traveler/R$drawable

```

```
Processing cn/com/cmcn/traveler/R$attr
Processing cn/com/cmcn/traveler/R$layout
Processing cn/com/cmcn/traveler/R$id
Processing cn/com/cmcn/traveler/R
Processing cn/com/cmcn/traveler/R$string
```

在上面看到的处理信息中,我们看到 TravelerTest 中的 classes.dex 已经被反编译成为类文件的相应信息,其中如果源码在命名类文件名称时采用了比较好的方法,可以看到包含了三个 Activity 类,分别为 WelcomeActivity, TravelListActivity 和 MainActivity 类,还有最重要的资源类 R.java 文件信息.

下面我们只需要分析 R.java 中的一个文件信息,如 R\$string.ddx 文件内容如下:

```
.class public final cn/com/cmcn/traveler/R$string
.super java/lang/Object
.source R.java

.field public static final addTravelmate I = 2131034117    ; 0x7f050005
.field public static final app_name I = 2131034113    ; 0x7f050001
.field public static final guideLine I = 2131034116    ; 0x7f050004
.field public static final hello I = 2131034112    ; 0x7f050000
.field public static final introduction I = 2131034115    ; 0x7f050003
.field public static final myLocation I = 2131034114    ; 0x7f050002

.method public <init>()V
.line 33
    invoke-direct {v0}, java/lang/Object/<init> ; <init>()V
    return-void
.end method
```

我们对上述内容信息进行分析,可以看到:

首先显示了类的文件信息,该类为 R 类的静态最终类 string 类.

其次显示了该类的父类为 java/lang/Object,程序表达则是 R extends Object,而默认可以不写

接着显示了来源信息为 R.java 类文件.

然后是我们最为关注的字段信息,可以从前面的修饰符看到,每个字段都是声明为公共静态最终的字段,即 public static final. 从提示的字段信息可以看到,变量名称为 addTravelmate 的字段的标识信息为 0x7f050005,这样我们可以根据字段对应信息建立对应关系,然后可以利用该对应关系进行内容替换.

最后是类文件的方法信息,这个我们可以不考虑.

由于 AndroidManifest.xml 还用到了 res/drawable 资源文件,因此也列出该文件对应的信息.

```
.class public final cn/com/cmcn/traveler/R$drawable
```

```
.super java/lang/Object
.source R.java

.field public static final icon I = 2130837504; 0x7f020000
.field public static final logo I = 2130837505; 0x7f020001

.method public <init>()V
.line 16
    invoke-direct {v0}, java/lang/Object/<init> ; <init>()V
    return-void
.end method
```

2.2.3 还原应用配置文件

从上述两个文件中我们可以找到标识分别为@7F050001, @7F020000 资源文件分别对应着字段 app_name, icon. 因此我们可以得到原配置文件信息.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0.0"
    package="cn.com.cmcn.traveler"
    >
    <application
        android:label="@string/app_name"
        android:icon="@drawable/icon"
        >
        <activity
            android:label="@string/app_name "
            android:name=".MainActivity"
            >
            <intent-filter
                >
                <action
                    android:name="android.intent.action.MAIN"
                    >
                </action>
                <category
                    android:name="android.intent.category.LAUNCHER"
                    >
                </category>
            </intent-filter>
        </activity>
```

```
<uses-library
    android:name="com.google.android.maps"
>
</uses-library>
//...省略信息
</application>
<uses-permission
    android:name="android.permission.INTERNET"
>
</uses-permission>
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION"
>
</uses-permission>
<uses-permission
    android:name="android.permission.READ_CONTACTS"
>
</uses-permission>
</manifest>
```

2.2.4 生成新的资源文件

将所有资源文件还原后,就可以通过 Eclipse 生成新的资源文件,然后替换原文件即可.

至此,第二种方式就可以将所有资源文件还原,然后可以重新生成本地化的资源文件,直接替换原有资源文件即可完成本地化过程.上述操作过程虽然不如直接替换来的直接,但可以通过编写某工具程序自动完成少数操作,然后只需要列出需要本地化的字符变量进行翻译即可.

作者基本信息:

姓 名: 胡怀国

出生日期: 1985 年 10 月

联系电话: 13685760951

电子邮件: tf1028@gmail.com