aapt 改造——自定义 packageId

目录

一、前言

Android 插件化后，每个插件 apk 中都有资源，因为 Android 中的资源都是以 packageId + typeId + entryId 组成，packageId 又是默认的 0x7f，所以，如果插件中有重名的资源时，那么在使用的时候就会互相覆盖，这是有问题的。

其中的一个解决思路是，在 Android 的 aapt 工具进行资源打包的时候，可以自定义 packageId，每个插件一个 packageId，这样的话就不会冲突了。

二、思路

加入一个参数“--apk-module”，来读取外部传入的 packageId。
根据 Bundle 把 packageId 传入，并进行传递。
设置 packageId 时判断 Bundle 中是否有 packageId，有的话覆盖。

三、流程

1、下载 Android 源码

这里示例的版本是 android-5.1.1_r38，aapt 所在的包路径是“frameworks/base/tools/aapt”。

2、修改源码

（1）、修改 **Main.cpp**，增加参数“**--apk-module**”：

```
else if(strcmp(cp, "-apk-module") == 0){
        argc--;
        argv++;
        if (!argc) {
            fprintf(stderr, "ERROR: No argument supplied for '--apk-module'
option\n");
            wantUsage = true;
            goto bail;
        }
        bundle.setApkModule(argv[0]);
}
```

（2）、在 **Bundle.h** 中加入 **setApkModule** 和 **getApkModule** 方法，还有对应的成员变量：

```
android::String8 mApkModule;
const android::String8& getApkModule() const {return mApkModule;}
void setApkModule(const char* str) { mApkModule=str;}
```

（3）、在 **ResourceTable.cpp** 的构造函数中，增加如下代码：

```
if(!bundle->getApkModule().isEmpty()){
   android::String8 apkmoduleVal=bundle->getApkModule();
   packageId=apkStringToInt(apkmoduleVal);
}
```

（4）、在 **ResourceTable.cpp** 中增加如下两个方法：

```
ssize_t ResourceTable::apkStringToInt(const String8& s){
        size_t i = 0;
        ssize_t val = 0;
        size_t len=s.length();
        if (s[i] < '0' || s[i] > '9') {
                return -1;
        }

        // Decimal or hex?
        if (s[i] == '0' && s[i+1] == 'x') {
                i += 2;
                bool error = false;
                while (i < len && !error) {
                        val = (val*16) + apkgetHex(s[i], &error);
                        i++;
                }
                if (error) {
                        return -1;
                }
        } else {
                while (i < len) {
                        if (s[i] < '0' || s[i] > '9') {
                                return false;
                        }
                        val = (val*10) + s[i]-'0';
                        i++;
                }
        }
```

```
    if (i == len) {
        return val;
    }
    return -1;
}

uint32_t ResourceTable::apkgetHex(char c, bool* outError){
    if (c >= '0' && c <= '9') {
        return c - '0';
    } else if (c >= 'a' && c <= 'f') {
        return c - 'a' + 0xa;
    } else if (c >= 'A' && c <= 'F') {
        return c - 'A' + 0xa;
    }
    *outError = true;
    return 0;
}
```

（5）、在 **ResourceTable.h** 中声明增加的这两个方法：

```
private:
    ssize_t apkStringToInt(const String8& s);
   uint32_t apkgetHex(char c, bool* outError);
```

（6）、在“**frameworks/base/include/androidfw**”包下，新建“**Helper.h**”：

```
#ifndef __Helper_h
#define __Helper_h

#include <stdio.h>
#include <stdlib.h>

using namespace std;

class Helper
{
        size_t packageId;
public:
        static Helper* getInstance()
        {
                static Helper instance;
                return &instance;
        }
        size_t getPackageId();
```

```
        void setPackageId(size_t _packageId);
protected:
        struct Object_Creator
        {
                Object_Creator()
                {
                        Helper::getInstance();
                }
        };
        static Object_Creator _object_creator;
        Helper() {
                packageId=0x7f;
        }
        ~Helper() {
        }
};
#endif
```

（7）、在“**frameworks/base/libs/androidfw**”包下，新建“**Helper.cpp**”：

```
#include <androidfw/Helper.h>

using namespace std;

Helper::Object_Creator Helper::_object_creator;

size_t Helper::getPackageId(){
        return packageId;
}
void Helper::setPackageId(size_t    _packageId){
        packageId=_packageId;
}
```

（8）、在“**frameworks/base/libs/androidfw**”包下的“**Android.mk**”文件中增加如下内容：

```
commonSources := \
    原来的内容   \
    Helper.cpp
```

（9）、在 **ResourceTable.cpp** 中引入新加的“**Helper**”，并调用 **Helper** 中的方法：

```
#include <androidfw/Helper.h>
```

在刚才新增取“**--apk-module**”值的后面增加调用 **Helper** 的方法：

```
if(!bundle->getApkModule().isEmpty()){
    android::String8 apkmoduleVal=bundle->getApkModule();
    packageId=apkStringToInt(apkmoduleVal);
}
//set package id
Helper::getInstance()->setPackageId(packageId);
```

（10）、在“**frameworks/base/libs/androidfw/ResourceType.cpp**”中引入 **Helper**，并增加如下代码：

```
#include <androidfw/Helper.h>
```

在 **bool ResTable::stringToValue** 方法中增加一个判断条件：

```
else if (packageId== Helper::getInstance()->getPackageId() ||packageId ==
APP_PACKAGE_ID || packageId == SYS_PACKAGE_ID) {
      // We accept packageId's generated as 0x01 in order to support
      // building the android system resources
      outValue->data = rid;
      return true;
}
```

四、编译

编译流程和编译 Android 源码流程相似，只不过最后执行的是：

```
make aapt
```