AAPT e aapt2 -- ID de recurso fixo e tag pública

2021-07-20 13:39:11 / POR STVEN_ KING

Prefácio

O artigo inteiro é sobre os pontos `tinker` de `TinkerResourceIdTask` conhecimento no .

1. `aapt` e `aapt2` A diferença de Ambiente de execução e resultados de execução ;
2. `id` Fixação de recursos ;
3. Conduzir `PUBLIC` A etiqueta ;

`aapt` O ambiente operacional é `gradle:2.2.0` e `gradle-wrapper:3.4.1`

`aapt2` O ambiente operacional é `gradle:3.3.2` e `gradle-wrapper:5.6.2`

android-aapt-sample O projeto é um exemplo do meu próprio experimento. Sim `aapt` e `aapt2` Dois ramos, Correspondendo à sua implementação respectivamente.

Resumo da AAPT

a partir do `Android Studio 3.0` início, `google` o padrão é em `aapt2` Como compilador para compilação de recursos, `aapt2` Aparência, Fornece suporte para compilação incremental de recursos. Claro, haverá alguns problemas no processo de usar , Podemos passar em **gradle.properties** Configuração do meio **android.enableAapt2=false** Para fechar `aapt2` .

`Android` Nascido para ser compatível com uma ampla gama de dispositivos diferentes para fazer muito trabalho, como tamanho da tela, internacionalização, teclado, densidade de pixels e assim por diante, podemos fazer compatibilidade para uma variedade de cenários específicos usando recursos específicos sem alterar uma linha de código , Suponha que adaptemos diferentes recursos para diferentes cenários , Como podemos aplicar esses recursos rapidamente `Android` Para nós `R` Esta classe , Especifica o índice de um recurso `id` , Então precisamos apenas informar ao sistema que em diferentes cenários de negócios , Basta usar o recursos correspondentes, Quanto ao arquivo específico no recurso especificado, é decidido pelo sistema de acordo com a configuração do desenvolvedor.

Neste caso , Suponha que demos valor `id` sim `x` , Agora quando o negócio precisa usar este recurso , O estado do telefone é `y` valor , Com (`x,y`), Em uma tabela, você pode localizar rapidamente o caminho específico do arquivo de recurso . Este relógio é `resources.arsc` , é de `aapt` compilado .

Na verdade, recursos binários Como a imagem Não há necessidade de compilar, é apenas esse ato de “compilar”, é gerar `resources.arsc` e correto a `xml` operação binária do arquivo, `resources.arsc` é o relógio acima, `xml` a binarização de é para melhor desempenho na leitura do sistema . `AssetManager` Depois que chamamos `R` dependente `id` Quando, Você encontrará o arquivo correspondente nesta tabela, Leia.

`Gradle` No processo de compilação de recursos, é o que chamamos de comando aapt2 , os parâmetros também são descritos neste documento, está apenas ocultando os detalhes da chamada dos desenvolvedores.

`aapt2` Existem principalmente dois passos, um passo é `compile` , um passo é `link` .

Crie um projeto vazio Apenas dois `xml` , Nomeadamente `AndroidManifest.xml` e `activity_main.xml` .

Compilar

```
mkdir compiled
aapt2 compile src/main/res/layout/activity_main.xml -o compiled/
Copy code
```

fique `compiled` na pasta , Gerado `layout_activity_main.xml.flat` Este arquivo , É `aapt2` Peculiar , `aapt` Não, (`aapt` A cópia é o arquivo fonte), `aapt2` Pode ser usado para compilação incremental . Se tivermos muitos documentos, você precisa chamar... Por sua vez `compile` Talento, Na verdade, também pode ser usado aqui `-dir` Parâmetros, Só que este parâmetro não tem efeito de compilação incremental. em outras palavras, ao passar o diretório inteiro, mesmo que apenas um recurso tenha sido alterado, `AAPT2` ele também recompilará todos os arquivos do diretório.

Link

`link` Taxa de carga de trabalho de `compile` Um pouco mais , A entrada aqui é múltipla `flat` O arquivo de e `AndroidManifest.xml` , Recursos externos , A saída é somente recurso `apk` e `R.java` . A ordem é a seguinte:

```
aapt2 link -o out.apk \
-I $ANDROID_HOME/platforms/android-28/android.jar \
compiled/layout_activity_main.xml.flat \
--java src/main/java \
--manifest src/main/AndroidManifest.xml
 Copy code
```

- A segunda linha `-I` sim `import` Recursos externos , Aqui está principalmente `android` Algumas propriedades definidas sob o namespace , Geralmente usamos `@android:xxx` É tudo neste `jar` Inside , Na verdade, também podemos fornecer nossos próprios recursos para outros vincularem ;
- A terceira linha é o `flat` arquivo de entrada , se houver mais de um , fica logo atrás ;
- A quarta linha é `R.java` Diretório gerado;
- A quinta linha especifica `AndroidManifest.xml` ;

`Link` Quando terminar, irá gerar `out.apk` e `R.java` , `out.apk` Contém um `resources.arsc` arquivo . Somente com arquivo de recurso pode-se usar sufixo `.ap_` .

Veja os recursos compilados

Além de usar `Android Studio` Para visualizar o `resources.arsc` , também pode ser usado diretamente `aapt2 dump apk` Informações para visualizar o status e o recurso relacionado `ID`

A saída é a seguinte:

```
Binary APK
Package name=com.geminiwen.hello id=7f
  type layout id=01 entryCount=1
    resource 0x7f010000 layout/activity_main
      () (file) res/layout/activity_main.xml type=XML
 Copy code
```

Você pode ver `layout/activity_main` Correspondente `ID` sim `0x7f010000` .

Compartilhamento de recursos

`android.jar` É apenas uma pilha para compilar , Quando é realmente implementado , `Android OS` Fornece uma biblioteca de tempo de execução (`framework.jar`). `android.jar` É muito parecido com um `apk` , é só que existe `class` file , então existe um `AndroidManifest.xml` e `resources.arsc` . Isso significa que também podemos usá-lo `aapt2 dump` , execute o seguinte comando:

```
aapt2 dump $ANDROID_HOME/platforms/android-28/android.jar > test.out
 Copy code
```

Você obtém um monte de saída como esta:

```
resource 0x010a0000 anim/fade_in PUBLIC
      () (file) res/anim/fade_in.xml type=XML
    resource 0x010a0001 anim/fade_out PUBLIC
      () (file) res/anim/fade_out.xml type=XML
    resource 0x010a0002 anim/slide_in_left PUBLIC
      () (file) res/anim/slide_in_left.xml type=XML
    resource 0x010a0003 anim/slide_out_right PUBLIC
      () (file) res/anim/slide_out_right.xml type=XML
 Copy code
```

É um pouco mais `PUBLIC` Campo de , Um `apk` Os recursos no arquivo , Se estiver marcado com isso , Pode ser usado por outros `apk` Os referenciados , A forma de citar é `@ Package name : type / name` , por exemplo `@android:color/red` .

Se quisermos fornecer nossos recursos , Bem, antes de tudo, estabeleça uma base para

quanto a `AAPT2` Como gerar `PUBLIC`, Se você estiver interessado, leia este artigo .

resumo de ids.xml

`ids.xml` Forneça recursos exclusivos para recursos relacionados ao aplicativo `id`. `id` Trata-se de obter `xml` os parâmetros necessários para objetos em , ou seja, `Object = findViewById(R.id.id_name);` Medium `id_name`.

Esses valores podem ser usados no código `android.R.id` Quote to . Se em `ids.xml` It define **ID** , está em `layout` Pode ser definido da seguinte `@id/price_edit` forma , caso contrário `@+id/price_edit`.

> vantagem

1. É fácil nomear, podemos nomear alguns controles específicos primeiro, referência direta ao usar `id` isso fará, uma etapa de nomenclatura é omitida.
2. Otimize a eficiência da compilação:
   - add to `id` I'll be in `R.java` No meio de ;
   - Use `ids.xml` o gerenciamento unificado, compile uma vez e use muitas vezes. Mas use `"@+id/btn_next"` Na forma de , Toda vez que um arquivo for salvo `(Ctrl+s`) `after R.java` Será testado novamente , Se o `id` Então não gerar , Se não existir, você precisa adicionar o `id`. Portanto, a eficiência de compilação é reduzida.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="forecast_list" type="id"/>
<!--    <item name="app_name" type="string" />-->
</resources>
 Copy code
```

Algumas pessoas podem estar curiosas que há uma linha de código anotado nele , Abra o comentário e você verá que o compilador reportará um erro

```
Execution failed for task ':app:mergeDebugResources'.
> [string/app_name] /Users/tanzx/AndroidStudioProjects/AaptDemo/app/src/main/res/values/string
 Copy code
```

porque `app_name` os recursos para já estão `value` declarados em .

resumo public.xml

Instruções oficiais Site oficial Selecione os recursos que deseja tornar públicos .

> Tradução original Todos os recursos da biblioteca são públicos por padrão . Para tornar todos os recursos implicitamente privados, você deve definir pelo menos um atributo específico como público. Os recursos incluem... Do seu projeto `res/` Todos os arquivos no diretório, como imagens. Para evitar que os usuários da biblioteca acessem recursos apenas para uso interno, você deve usar esse mecanismo automático de identificação privada declarando um ou mais recursos públicos. talvez , Você também pode adicionar uma `<public />` Tag vazia torna todos os recursos privados , Esta tag não torna nenhum recurso público , Vai levar tudo Todos os recursos Todos privados .
>
> Tornando as propriedades implicitamente privadas, você pode não apenas impedir que os usuários da biblioteca obtenham conselhos de conclusão de código dos recursos internos da biblioteca, mas também renomear ou remover recursos privados, sem destruir o lado cliente da biblioteca. O sistema filtra recursos privados do preenchimento de código, além disso, um aviso Lint Um aviso é emitido quando você tenta fazer referência a recursos privados.
>
> Ao construir a biblioteca, Android Gradle O plug-in obtém a definição de recurso

Os resultados medidos estão apenas incompletos, está vermelho. Se realizado `lint` Check , Não há avisos para compilar ~

Agora a maior parte da explicação é arquivo **RES/value/public.xml** Usado para colocar recursos fixos `ID` Atribuído a `Android` recursos .

stackoverfloew: Qual é o uso do arquivo res/values/public.xml no Android?

`public.xml` O conteúdo do documento:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <public name="forecast_list" id="0x7f040001" type="id" />
    <public name="app_name" id="0x7f070002" type="string" />
    <public name="string3" id="0x7f070003" type="string" />
</resources>
 Copy code
```

Correção de ID de recursos

recursos id A fixação de é extremamente importante em reparo a quente e plug-in. Em reparo a quente , estrutura `patch` quando , Precisa manter os recursos do `patch` pacote `id` E recursos do pacote de referência `id` Acordo ; No plug-in , se o plug-in precisar fazer referência aos recursos do host , você precisará transferir os recursos do host `id` Fix , portanto , recursos `id` É especialmente importante ser corrigido nesses dois cenários .

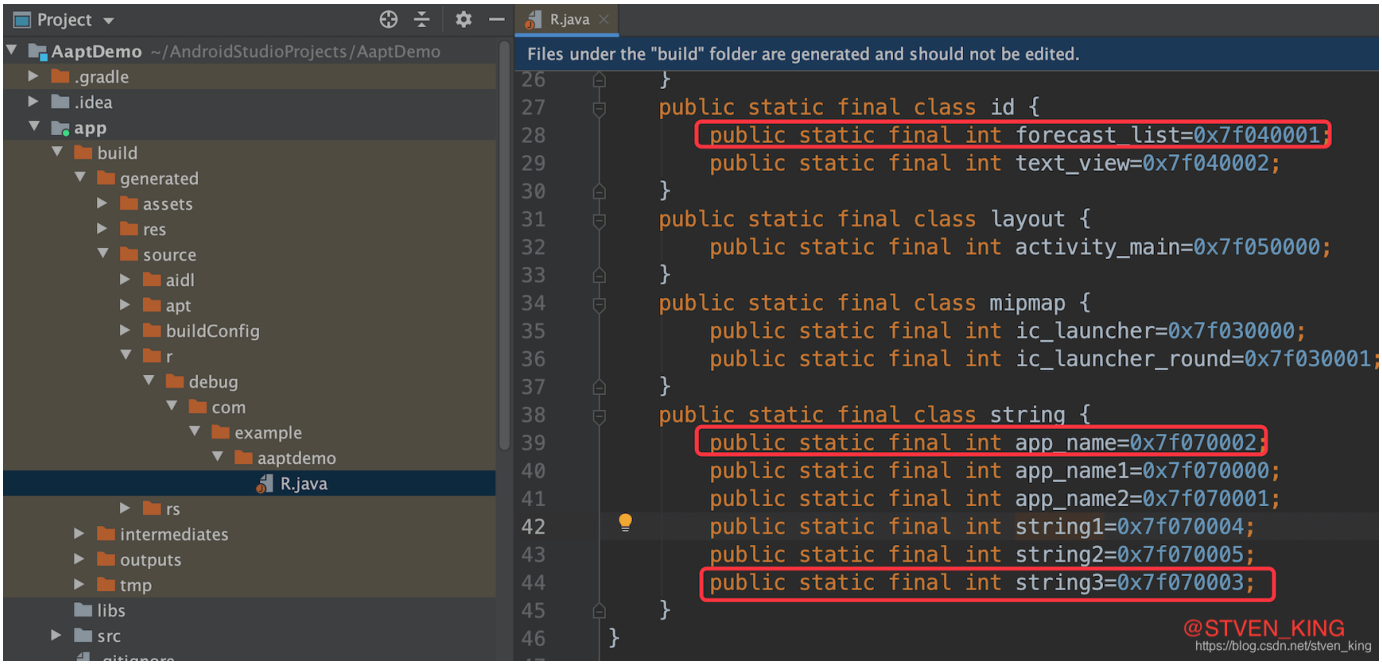
`aapt and aapt2` Como realizar o gerenciamento de recursos, respectivamente `id` Fixação.

`aapt id` Fixação de Conduta

> Configuração do ambiente do projeto PS Faça reclamações sobre isso aapt Foi aapt2 Em vez de ,aapt Há pouca informação sobre isso, É muito difícil construir o ambiente ~
>
> `com.android.tools.build:gradle:2.2.0`
>
> `distributionUrl=https\://services.gradle.org/distributions/gradle-3.4.1-all.zip`
>
> `compileSdkVersion 24`
>
> `buildToolsVersion '24.0.0'`

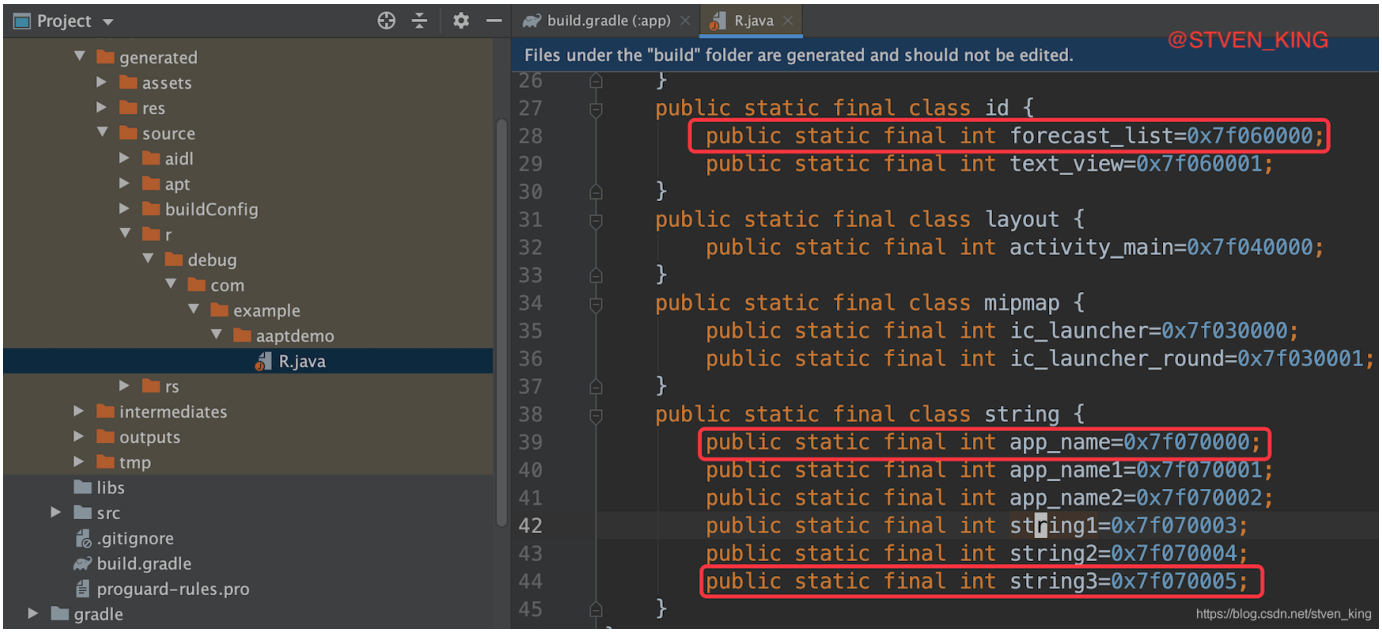
Primeiro em `value` Sob o arquivo, siga o acima `ids.xml` e `public.xml` E o nome do arquivo, Gere o arquivo correspondente.

> Resultados diretos da compilação



Através da compilação direta do `R file` conteúdo de , você pode ver os recursos que queremos configurar. `id` Não gerou como esperávamos.

```
afterEvaluate {
    for (variant in android.applicationVariants) {
        def scope = variant.getVariantData().getScope()
        String mergeTaskName = scope.getMergeResourcesTask().name
        def mergeTask = tasks.getByName(mergeTaskName)
        mergeTask.doLast {
            copy {
                int i=0
                from(android.sourceSets.main.res.srcDirs) {
                    include 'values/public.xml'
                    rename 'public.xml', (i++ == 0? "public.xml": "public_${i}.xml")
                }
                into(mergeTask.outputDir)
            }
        }
    }
}
 Copy code
```



Desta vez podemos ver os recursos `id` gerados diretamente de acordo com nossas necessidades.

> Por que é que ?

1. `android gradle` unidade de plug-in `1.3` A versão a seguir pode colocá-lo diretamente `public.xml` no código-fonte `res` O diretório está envolvido na compilação;

2. `android gradle` unidade de plug-in `1.3+` A versão está em execução `mergeResource` A

índice. É por isso que funciona, porque `aapt` é suporte em si mesmo `public.xml`, é apenas `gradle` o plug-in que está pré-processando recursos `(merge)` quando o `public.xml` faz o filtro.

`aapt2 id` Fixação de Conduta

ficar `aapt2` compilar Compilar o arquivo de recursos em formato binário após , Descobrir `merge` Todos os recursos foram pré-compilados , `flat` Arquivo produzido , Desta vez será `public.xml` Se o arquivo for copiado para este diretório, ocorrerá um erro de compilação .

Mas na fase `aapt2` Of **link In , vamos ver as opções de Link** relevantes

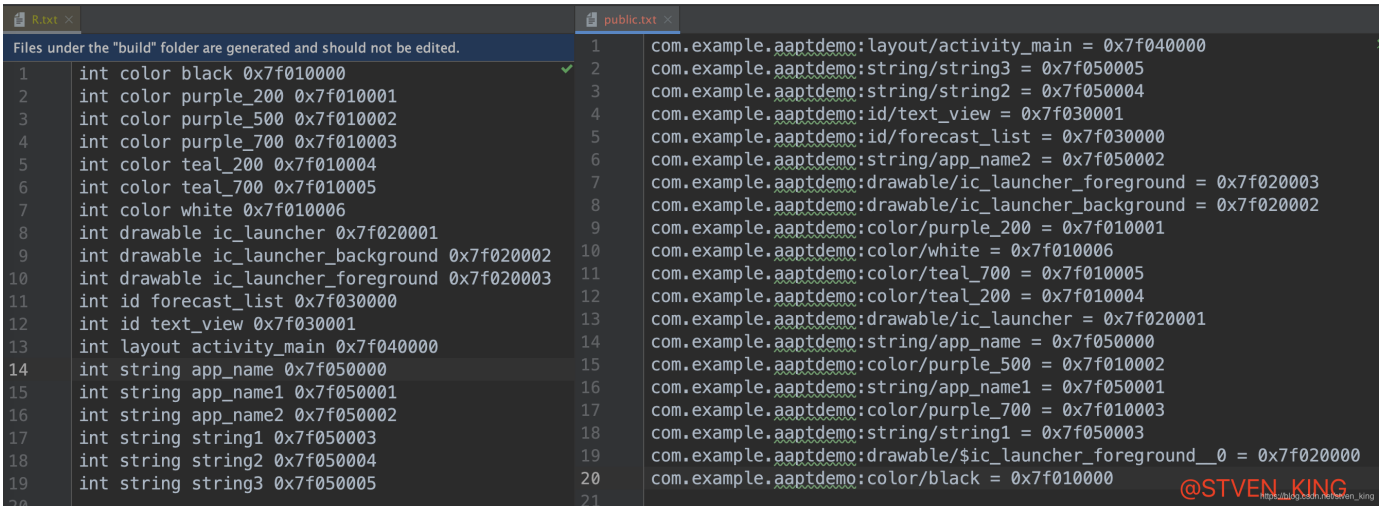
Opções	explique
`--emit-ids path`	Gere um arquivo no caminho fornecido, Este arquivo contém o nome do tipo de recurso e seu ID Uma lista de mapeamentos. É adequado para `--stable-ids` usá-lo com .
`--stable-ids outputfilename.ext`	Use pass `--emit-ids` Generated files , This file contains the name of the resource type and the ID A list of . This option allows the assigned ID Stay stable , Even if you delete a resource or add a new resource when linking .

Find out `--emit-ids` and `--stable-ids` Command collocation can achieve `id` Fixation .

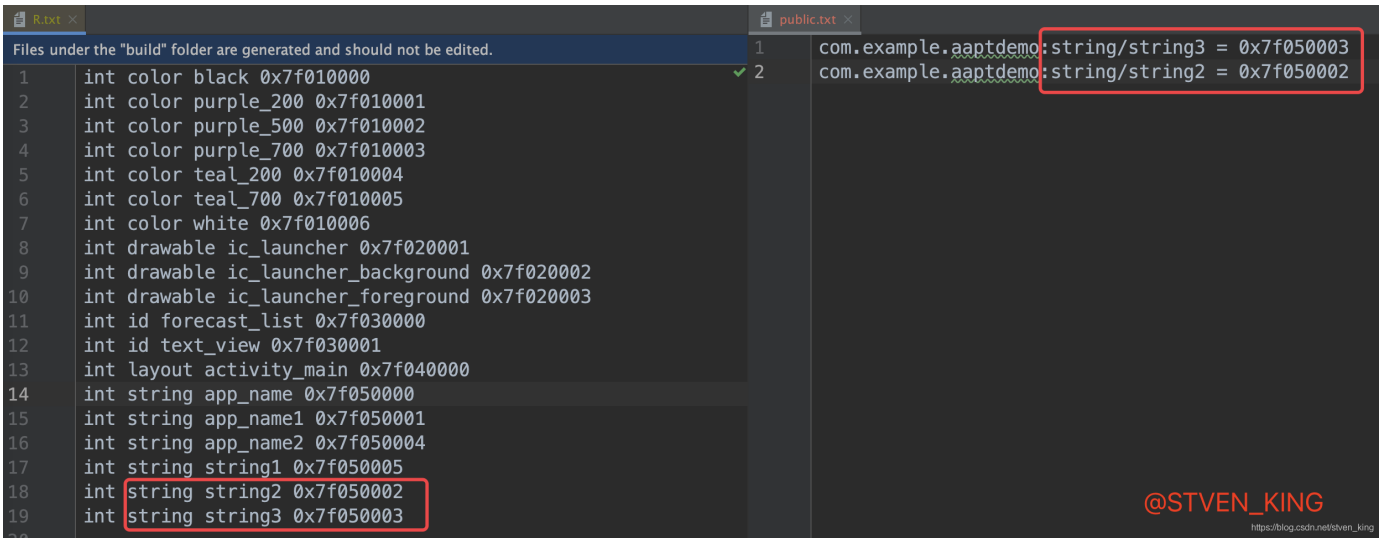
```
android {
  aaptOptions {
        File publicTxtFile = project.rootProject.file('public.txt')
        //public File exists , The application of , If it doesn't exist, it generates
        if (publicTxtFile.exists()) {
            project.logger.error "${publicTxtFile} exists, apply it."
            //aapt2 add to --stable-ids Parameter application
            aaptOptions.additionalParameters("--stable-ids", "${publicTxtFile}")
        } else {
            project.logger.error "${publicTxtFile} not exists, generate it."
            //aapt2 add to --emit-ids Parameter generation
            aaptOptions.additionalParameters("--emit-ids", "${publicTxtFile}")
        }
    }
}
 Copy code
```

2. then `public.txt` It's about `id` Change to what you want to fix `id` ;
3. Compile again , adopt `--stable-ids` And the root directory `public.txt` Carry out resources `id` Fixation ;

> `--emit-ids` Compilation result



> modify `public.txt` The contents of the file are compiled again



R.txt turn public.txt

The intermediate product that we normally package is `build/intermediates/symbols/debug/R.txt` , It needs to be translated into `public.txt` .

`R.txt` Format `int type name id` perhaps `int[] styleable name {id,id,xxxx}`

`public.txt` Format `applicationId:type/name = id`

So it needs to be filtered out in the conversion process `R.txt` In the document `styleable` type .

```
android {
    aaptOptions {
        File rFile = project.rootProject.file('R.txt')
        List<String> sortedLines = new ArrayList<>()
        //  Read line by line
        rFile.eachLine {line ->
            //rLines.add(line)
            String[] test = line.split(" ")
            String type = test[1]
            String name = test[2]
            String idValue = test[3]
            if ("styleable" != type) {
                sortedLines.add("${applicationId}:${type}/${name} = ${idValue}")
            }
        }
        Collections.sort(sortedLines)
        File publicTxtFile = project.rootProject.file('public.txt')
        if (!publicTxtFile.exists()) {
            publicTxtFile.createNewFile()
            sortedLines?.each {
                publicTxtFile.append("${it}\n")
            }
        }
    }
```

PUBLIC Mark

stay `AAPT summary` In this part, we talked about if a `apk` The resources in the file , If you add `PUBLIC` Marked words , Can be used by other `apk` The referenced , The way to quote is `@Package name : type / name` , for example `@android:color/red` .

Read the above 《 `aapt` Conduct `id` Fixation 》 To 《 `aapt2` Conduct `id` Fixation 》 These two parts , We know `aapt` and `aapt2` Conduct `id` Fixed methods are different .

In fact, if we use `aapt2 dump build/intermediates/res/resources-debug.ap_` Command to view information about the generated resources .

`aapt` adopt `public.xml` Conduct `id` The fixed resource information is `PUBLIC` Mark

```
type id id=4 entryCount=2
  spec resource 0x7f040001 com.example.aaptdemo:id/forecast_list PUBLIC
    () (id)
  spec resource 0x7f040002 com.example.aaptdemo:id/text_view
    () (id)

type layout id=3 entryCount=1
  spec resource 0x7f030000 com.example.aaptdemo:layout/activity_main
    () (file) res/layout/activity_main.xml

type string id=7 entryCount=6
  spec resource 0x7f070000 com.example.aaptdemo:string/app_name1
    () (string) "AaptDemo1"
  spec resource 0x7f070001 com.example.aaptdemo:string/app_name2
    () (string) "AaptDemo2"
  spec resource 0x7f070002 com.example.aaptdemo:string/app_name PUBLIC
    () (string) "AaptDemo"
  spec resource 0x7f070003 com.example.aaptdemo:string/string3 PUBLIC
    () (string) "String 3"
  spec resource 0x7f070004 com.example.aaptdemo:string/string1
    () (string) "String 1"
  spec resource 0x7f070005 com.example.aaptdemo:string/string2
    () (string) "String 2"
```


2. Use the above `aapt2` Conduct `id` The fixed way is not as shown in the figure below `PUBLIC` Of the tag .

The reason is still `aapt` and `aapt2` It's caused by the difference between , `aapt2` Of `public.txt` It's not equal to `aapt` Of `public.xml` , stay `aapt2` If you want to add `PUBLIC` Mark , In fact, we have to find another way .

Think back

> review

1. `aapt` Carry out resources `id` Fixed and `PUBLIC` Price , Yes, it will `public.xml` Copied to the `${mergeResourceTask.outputDir}` ;
2. `aapt2` Compared with `aapt` , Do the optimization of incremental compilation . `AAPT2` The file is parsed and a file with the extension `.flat` The intermediate binary of .

> reflection

Can you use `aapt2` I will `public.xml` Compiled into `public.arsc.flat`, And image `aapt` Copy it to `${mergeResourceTask.outputDir}`;

Hands-on practice

```
android {
    // take public.txt Turn into public.xml, Also on public.xml Conduct aapt2 Copy the results
  // Most of the code below is copy since tinker Source code
  applicationVariants.all { def variant ->
      def mergeResourceTask = project.tasks.findByName("merge${variant.getName().capitalize()}
      if (mergeResourceTask) {
          mergeResourceTask.doLast {
              // Target conversion file , Be careful public.xml The superior directory must ca
              File publicXmlFile = new File(project.buildDir, "intermediates/res/public/${vari
              // transformation public.txt File for publicXml file , Last parameter true Ident
              convertPublicTxtToPublicXml(project.rootProject.file('public.txt'), publicXmlFi
              def variantData = variant.getMetaClass().getProperty(variant, 'variantData')
              def variantScope = variantData.getScope()
              def globalScope = variantScope.getGlobalScope()
              def androidBuilder = globalScope.getAndroidBuilder()
              def targetInfo = androidBuilder.getTargetInfo()
              def mBuildToolInfo = targetInfo.getBuildTools()
              Map<BuildToolInfo.PathId, String> mPaths = mBuildToolInfo.getMetaClass().getProp
                // adopt aapt2 compile Commands are generated by themselves public.arsc.flat A
              project.exec(new Action<ExecSpec>() {
                  @Override
                  void execute(ExecSpec execSpec) {
                      execSpec.executable "${mPaths.get(BuildToolInfo.PathId.AAPT2)}"
                      execSpec.args("compile")
                      execSpec.args("--legacy")
                      execSpec.args("-o")
                      execSpec.args("${mergeResourceTask.outputDir}")
                      execSpec.args("${publicXmlFile}")
                  }
              })
          }
      }
  }
}
 Copy code
```

take `public.txt` File to `public.xml` file .

- `public.txt` in `styleable` Type resources , `public.xml` Does not exist in the , So if you encounter `styleable` type , Need to ignore ;
- `vector` Vector resources if there are internal resources , It also needs to be ignored , stay `aapt2` in , It's named after `$` start , And then the primary resource name , Follow closely `__` Digital incremental index , These resources can't be referenced from outside , Just fix it `id` , No need to add `PUBLIC` Mark , also `$` The sign is in `public.xml` It's illegal , So just ignore it ;
- because `aapt2` There are resources `id` It's fixed in a way that , Therefore, it can be directly discarded in the conversion process `id` , Just a simple statement PS Through here `withId` Whether the parameter control needs to be fixed `id` ;
- `aapt2` Compilation of `public.xml` The superior directory of the file must be `values` Folder , Otherwise, the compilation process will report an illegal path ;

```
/**
 *  transformation publicTxt by publicXml
 * copy tinker:com.tencent.tinker.build.gradle.task.TinkerResourceIdTask#convertPublicTxtToPub
 */
@SuppressWarnings("GrMethodMayBeStatic")
void convertPublicTxtToPublicXml(File publicTxtFile, File publicXmlFile, boolean withId) {
    if (publicTxtFile == null || publicXmlFile == null || !publicTxtFile.exists() || !publicTx
        throw new GradleException("publicTxtFile ${publicTxtFile} is not exist or not a file")
    }

    GFileUtils.deleteQuietly(publicXmlFile)
    GFileUtils.mkdirs(publicXmlFile.getParentFile())
    GFileUtils.touch(publicXmlFile)

    project.logger.info "convert publicTxtFile ${publicTxtFile} to publicXmlFile ${publicXmlFi

    publicXmlFile.append("<!-- AUTO-GENERATED FILE.  DO NOT MODIFY -->")
    publicXmlFile.append("\n")
    publicXmlFile.append("<resources>")
    publicXmlFile.append("\n")
    Pattern linePattern = Pattern.compile(".*?:(.*?)/(.*?)\\s+=\\s+(.*?)")

    publicTxtFile.eachLine {def line ->
        Matcher matcher = linePattern.matcher(line)
        if (matcher.matches() && matcher.groupCount() == 3) {
            String resType = matcher.group(1)
            String resName = matcher.group(2)
            if (resName.startsWith('$')) {
                project.logger.info "ignore to public res ${resName} because it's a nested res
            } else if (resType.equalsIgnoreCase("styleable")) {
                project.logger.info "ignore to public res ${resName} because it's a styleable
            } else {
                if (withId) {
                    publicXmlFile.append("\t<public type=\"${resType}\" name=\"${resName}\" id
                } else {
                    publicXmlFile.append("\t<public type=\"${resType}\" name=\"${resName}\" />
                }

            }
        }
    }
    publicXmlFile.append("</resources>")
}
 Copy code
```

The above process of thinking and practicing , We not only solved `aapt2` Conduct `PUBLIC` The question of marking , And found a new `aapt2` Conduct `id` Fixed method .

Possible misstatements

The solution is to modify `gradle` Version is `gradle:3.3.2` and `gradle-wrapper:5.6.2` , After all `tinker` It doesn't support the latest version of `gradle` .

Reference resources

Github:tinker

android public.xml usage

Android-Gradle note

aapt2 Adaptive resources id Fix

This is the end of the article , If you need to communicate with others, you can leave a message ~ ~

[Stven_ King]
https://cdmana.com/2021/07/20210718112558372F.html

Tag RECURSO FIXO AAPT AAPT2 AAPT

Oracle 12C RAC modificando a configuração de varredura

Oracle modificar fuso horário

Estudo aprofundado sobre a relação entre modificação de hora do sistema do servidor Or

O primeiro marcador da Oracle ASM Translation Series: Basics ASM Au, extensões, espelhamento e grupos de falhas

Todo programador deve aprender o conhecimento Maven

sem dormir

Jia Ling se tornou a diretora feminina de maior bilheteria do mundo, e o filme emocionou i números audiências!

Mulheres com mais de 50 anos usam calças menos compridas no verão e aprendem com a mãe Miki. É elegante e moderno

nginx

SpringBoot-Elasticsearch

SpringBoot-Kotlin

docker-compose

docker

docker Springboot

Linux 777

SpringSecurity —

SpringSecurity

SpringSecurity

SpringSecurity

maven

Git

SpringSecurity

SpringSecurity

SpringSecurity

Aprendizagem Nginx

SpringBoot-Elasticsearch

Entrada do poço do docker

Docker implantando projeto springboot

5 VueUse

Recuperação após operação incorreta do Linux 777

Introdução à segurança da primavera (1)

Introdução à segurança de molas (3)

Introdução à segurança de molas (2)

Introdução à segurança da mola (4)

Implantar servidor privado Maven

Entrada do poço do Git

Entrada do poço de segurança da mola (III)

Entrada do poço de segurança da mola (II)

Entrada do poço de segurança da mola (V)

5 funções da biblioteca vueuse que podem acelerar o desenvolvimento

Linux

Resumo do Linux

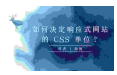
Guia definitivo para otimização de desempenho de front-end do site

Tutorial para iniciantes do estúdio de desenvolvimento Android

Compor | para entender o modificador mágico

Análise do princípio do binder para engenheiros de aplicativos Android

Série de plug-in gradle (I) - configure seu próprio plug-in gradle



CSS

Github1.3 JAX TensorFlow ،PyTorch

GitHub 13000 stars, Jax em rápido desenvolvimento, comparado com tensorflow e pytorch

[algoritmo diário] revisão de algoritmo I

(java

Yu Zheng: deve haver algumas pessoas que não gostam da primavera das man sões de jade, como aqueles com baixa escolaridade e aqueles que gostam de se r exigentes

Fila para aprendizado de algoritmo de estrutura de dados (implementação Java de simul ação de matriz)


chxpostbox@gmail.com
Copyright © 2020 AAPT e aapt2 -- ID de recurso fixo e tag pública Todos os direitos reservados.