[**Android动态加载jar/dex**](http://www.cnblogs.com/over140/archive/2011/11/23/2259367.html)

**前言**

　　 在目前的软硬件环境下，Native App与Web App在用户体验上有着明显的优势，但在实际项目中有些会因为业务的频繁变更而频繁的升级客户端，造成较差的用户体验，而这也恰恰是Web App的优势。本文对网上Android动态加载jar的资料进行梳理和实践在这里与大家一起分享，试图改善频繁升级这一弊病。

**声明**

　　欢迎转载，但请保留文章原始出处:)

　　　　博客园：http://www.cnblogs.com

　　　　农民伯伯： http://over140.cnblogs.com

　　　　Android中文翻译组：<http://androidbox.sinaapp.com/>

**正文**

　　一、 基本概念和注意点

　　　　1.1　　首先需要了解一点：在Android中可以动态加载，但无法像Java中那样方便动态加载jar

　　　　　　原因：Android的虚拟机(Dalvik VM)是不认识Java打出jar的byte code，需要通过dx工具来优化转换成Dalvik byte code才行。这一点在咱们Android项目打包的apk中可以看出：引入其他Jar的内容都被打包进了classes.dex。

　　　　　　所以这条路不通，请大家注意。

　　　　1.2　　当前哪些API可用于动态加载

　　　　　　1.2.1　　DexClassLoader

　　　　　　　　这个可以加载jar/apk/dex，也可以从SD卡中加载，也是本文的重点。

　　　　　　1.2.3　　PathClassLoader

　　　　　　　　只能加载已经安装到Android系统中的apk文件。

　　二、 准备

　　　　本文主要参考"四、参考文章"中第一篇文章，补充细节和实践过程。

　　　　2.1　　下载开源项目

<http://code.google.com/p/goodev-demo>

　　　　　　将项目导入工程，工程报错的话应该是少了gen文件夹，手动添加即可。注意这个例子是从网上下载优化好的jar(已经优化成dex然后再打包成的jar)到本地文件系统，然后再从本地文件系统加载并调用的。本文则直接改成从SD卡加载。

　　三、实践

　　　　3.1　　编写接口和实现

　　　　　　3.1.1　　接口IDynamic

package com.dynamic;  
  
public interface IDynamic {  
public String helloWorld();  
}

　　　　　　3.1.2　　实现类DynamicTest

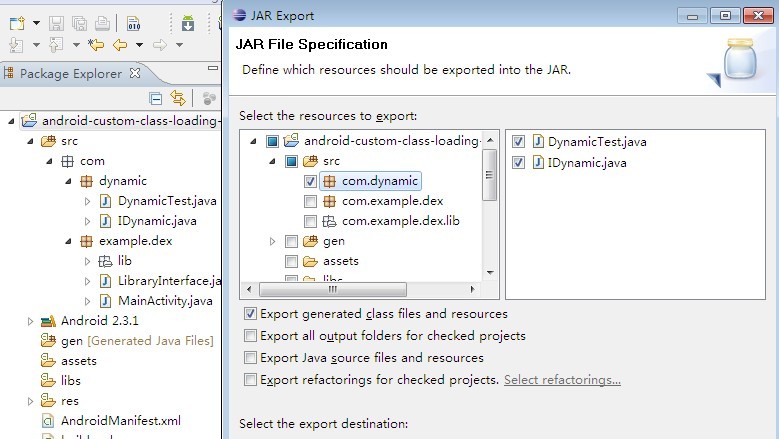
[复制代码](javascript:void(0);)

package com.dynamic;  
  
public class DynamicTest implements IDynamic {  
  
@Override  
public String helloWorld() {  
return "Hello World!";  
}  
}

[复制代码](javascript:void(0);)

　　　　3.2　　打包并转成dex

　　　　　　3.2.1　　选中工程，常规流程导出即可，如图：



　　　　　　注意：在实践中发现，自己新建一个Java工程然后导出jar是无法使用的，这一点大家可以根据文章一来了解相关原因，也是本文的重点之一。这里打包导出为dynamic.jar

　　　　　　（后期修复：打包请不要把接口文件打进来，参见文章末尾后续维护！）

　　　　　　3.2.2　　将打包好的jar拷贝到SDK安装目录android-sdk-windows\platform-tools下，DOS进入这个目录，执行命名：

dx --dex --output=test.jar dynamic.jar

　　　　3.3　　修改调用例子

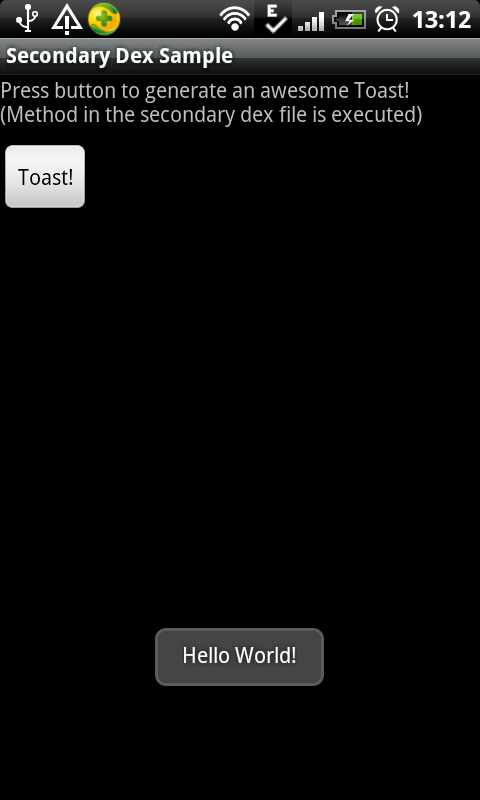
　　　　　　修改MainActivity，如下：

[复制代码](javascript:void(0);)

@Override  
public void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
setContentView(R.layout.main);  
mToastButton = (Button) findViewById(R.id.toast\_button);  
  
// Before the secondary dex file can be processed by the DexClassLoader,  
// it has to be first copied from asset resource to a storage location.  
// final File dexInternalStoragePath = new File(getDir("dex", Context.MODE\_PRIVATE),SECONDARY\_DEX\_NAME);  
// if (!dexInternalStoragePath.exists()) {  
// mProgressDialog = ProgressDialog.show(this,  
// getResources().getString(R.string.diag\_title),   
// getResources().getString(R.string.diag\_message), true, false);  
// // Perform the file copying in an AsyncTask.  
// // 从网络下载需要的dex文件  
// (new PrepareDexTask()).execute(dexInternalStoragePath);  
// } else {  
// mToastButton.setEnabled(true);  
// }  
  
mToastButton.setOnClickListener(new View.OnClickListener() {  
public void onClick(View view) {  
// Internal storage where the DexClassLoader writes the optimized dex file to.  
//final File optimizedDexOutputPath = getDir("outdex", Context.MODE\_PRIVATE);  
final File optimizedDexOutputPath = new File(Environment.getExternalStorageDirectory().toString()  
+ File.separator + "test.jar");  
// Initialize the class loader with the secondary dex file.  
// DexClassLoader cl = new DexClassLoader(dexInternalStoragePath.getAbsolutePath(),  
// optimizedDexOutputPath.getAbsolutePath(),  
// null,  
// getClassLoader());  
DexClassLoader cl = new DexClassLoader(optimizedDexOutputPath.getAbsolutePath(),  
Environment.getExternalStorageDirectory().toString(), null, getClassLoader());  
Class libProviderClazz = null;  
  
try {  
// Load the library class from the class loader.  
// 载入从网络上下载的类  
// libProviderClazz = cl.loadClass("com.example.dex.lib.LibraryProvider");  
libProviderClazz = cl.loadClass("com.dynamic.DynamicTest");  
  
// Cast the return object to the library interface so that the  
// caller can directly invoke methods in the interface.  
// Alternatively, the caller can invoke methods through reflection,  
// which is more verbose and slow.  
//LibraryInterface lib = (LibraryInterface) libProviderClazz.newInstance();  
IDynamic lib = (IDynamic)libProviderClazz.newInstance();  
  
// Display the toast!  
//lib.showAwesomeToast(view.getContext(), "hello 世界!");  
Toast.makeText(MainActivity.this, lib.helloWorld(), Toast.LENGTH\_SHORT).show();  
} catch (Exception exception) {  
// Handle exception gracefully here.  
exception.printStackTrace();  
}  
}  
});  
}

[复制代码](javascript:void(0);)

　　　　3.4　　执行结果



　　四、参考文章

[[推荐]在Android中动态载入自定义类](http://yunfeng.sinaapp.com/?p=87)

[Android app中加载jar插件](http://www.cnblogs.com/franksunny/archive/2011/09/27/2193085.html)

[关于Android的ClassLoader探索](http://blog.csdn.net/czh0766/article/details/6736826)

[Android App 如何动态加载类](http://blog.csdn.net/quaful/article/details/6096951)

　　五、补充

　　　　大家可以看看DexClassLoader的API文档，里面不提倡从SD卡加载，不安全。此外，我也正在组织翻译组尽快把这个命名空间下的几个类都翻译出来，以供大家参考。

　　　　工程下载：[这里](http://files.cnblogs.com/over140/2011/11/android-custom-class-loading-goodev-demo.zip)，Dex文件下载：[这里](http://files.cnblogs.com/over140/2011/11/dynamic.zip)。大家可以直接把Dex文件拷贝到SD卡，然后运行例子。

　　六、后期维护

　　　　6.1　　2011-12-1　　修复本文错误

　　　　　　感谢网友ppp250和liuzhaocn的反馈，基本按照评论2来修改：

　　　　　　6.1.1　　不需要在本工程里面导出jar，自己新建一个Java工程然后导出来也行。

　　　　　　6.1.2　　导出jar时不能带接口文件，否则会报以下错：

　　　　　　　　java.lang.IllegalAccessError: Class ref in pre-verified class resolved to unexpected implementation

　　　　　　6.1.3　　将jar优化时应该重新成jar(jar->dex->jar)，如果如下命令：

　　　　　　dx --dex --output=test.jar dynamic.jar

　　　　6.2　　2012-3-29　　本文升级版：

[Android应用开发提高系列（4）——Android动态加载（上）——加载未安装APK中的类](http://www.cnblogs.com/over140/archive/2012/03/29/2423116.html)

　　　　　　请大家参照最新的文章来做动态加载！

**结束**

　　除了翻译组的工作和自己本职的工作以外，很难抽时间出来分享一些开发心得，但正所谓挤挤总是有的，欢迎交流！

分类: [1、Android](http://www.cnblogs.com/over140/category/253648.html)

标签: [android DexClassLoader](http://www.cnblogs.com/over140/tag/android%20DexClassLoader/), [android 动态加载](http://www.cnblogs.com/over140/tag/android%20%E5%8A%A8%E6%80%81%E5%8A%A0%E8%BD%BD/), [android 加载jar](http://www.cnblogs.com/over140/tag/android%20%E5%8A%A0%E8%BD%BDjar/), [android load jar](http://www.cnblogs.com/over140/tag/android%20load%20jar/), [android dynamic jar](http://www.cnblogs.com/over140/tag/android%20dynamic%20jar/)

Button btn = (Button) findViewById(R.id.*btn*);

btn.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

Bundle paramBundle = **new** Bundle();

paramBundle.putBoolean("KEY\_START\_FROM\_OTHER\_ACTIVITY", **true**);

String dexpath = "/mnt/sdcard/TestB.apk";

String dexoutputpath = "/mnt/sdcard/";

LoadAPK(paramBundle, dexpath, dexoutputpath);

}

});

}

**public** **void** LoadAPK(Bundle paramBundle, String dexpath, String dexoutputpath) {

ClassLoader localClassLoader = ClassLoader.*getSystemClassLoader*();

DexClassLoader localDexClassLoader = **new** DexClassLoader(dexpath,

dexoutputpath, **null**, localClassLoader);

**try** {

PackageInfo plocalObject = getPackageManager()

.getPackageArchiveInfo(dexpath, 1);

**if** ((plocalObject.activities != **null**)

&& (plocalObject.activities.length > 0)) {

String activityname = plocalObject.activities[0].name;

Log.*d*(*TAG*, "activityname = " + activityname);

Class localClass = localDexClassLoader.loadClass(activityname);

Constructor localConstructor = localClass

.getConstructor(**new** Class[] {});

Object instance = localConstructor.newInstance(**new** Object[] {});

Log.*d*(*TAG*, "instance = " + instance);

Method localMethodSetActivity = localClass.getDeclaredMethod(

"setActivity", **new** Class[] { Activity.**class** });

localMethodSetActivity.setAccessible(**true**);

localMethodSetActivity.invoke(instance, **new** Object[] { **this** });

Method methodonCreate = localClass.getDeclaredMethod(

"onCreate", **new** Class[] { Bundle.**class** });

methodonCreate.setAccessible(**true**);

methodonCreate.invoke(instance, **new** Object[] { paramBundle });

}

**return**;

} **catch** (Exception ex) {

ex.printStackTrace();

}

**private** **static** **final** String *TAG* = "TestBActivity";

**private** Activity otherActivity;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**boolean** b = **false**;

**if** (savedInstanceState != **null**) {

b = savedInstanceState.getBoolean("KEY\_START\_FROM\_OTHER\_ACTIVITY", **false**);

**if** (b) {

**this**.otherActivity.setContentView(**new** TBSurfaceView(

**this**.otherActivity));

}

}

**if** (!b) {

**super**.onCreate(savedInstanceState);

// setContentView(R.layout.main);

setContentView(**new** TBSurfaceView(**this**));

}

}

**public** **void** setActivity(Activity paramActivity) {

Log.*d*(*TAG*, "setActivity..." + paramActivity);

**this**.otherActivity = paramActivity;

}

}