Cross Building for Android

UNDER CONSTRUCTION - these don't work

How to build Open JavaFX for Android.

Here's a short recipe for baking JavaFX for Android dalvik. We will need just a few ingredients but each one requires special care. So let's get down to the business.

Sources

The first ingredient is an open JavaFX repository. This should be piece of cake. As always there's a catch. You probably know that dalvik is jdk6 compatible and also that certain APIs are missing compared to the OpenJDK. Fortunately there is a repository which is a backport of OpenJFX to jdk7 and going from jdk7 to jdk6 is possible. The first thing to do is to clone or download the repository from https://bitbucket.org/narya/jfx78.

The Dalvik vm is missing some APIs which lead to a build failures. To get them, use another compatibility repository which is available on GitHub https://github.com/robovm/robovm-jfx78-compat. Download the zip and unzip sources into jfx78/modules/base.
We need also a javafx binary stubs. Use jfxrt.jar from jdk8.
The last thing to download are freetype sources from http://freetype.org. These will be necessary for native font rendering.

Toolchain setup

I have to point out that these instructions were tested only on linux. I suppose they will work with minimal changes also on Mac OS. I also presume that you were able to build open JavaFX. That means all tools like ant, gradle, gcc and jdk8 have been installed and are working all right. In addition to this you will need to download and install jdk7,Android SDK and Android NDK for native code compilation. Installing all of them will take some time. Don't forget to put them in your path.

```
export ANDROID_SDK=/opt/android-sdk-linux
export ANDROID_NDK=/opt/android-ndk-r9b
export JAVA_HOME=/opt/jdk1.7.0
export PATH=$JAVA_HOME/bin:$ANDROID_SDK/tools:$ANDROID_SDK/platform-tools:$ANDROID_NDK
```

Freetype

Unzip freetype release sources first. We will have to cross compile them for arm. Firstly we will create a standalone toolchain for cross compiling installed in ~/work/ndk-standalone-19.

```
$ANDROID_NDK/build/tools/make-standalone-toolchain.sh  --platform=android-19 --install-dir=~/work/ndk-standalone-19
```

After the standalone toolchain has been created cross compile freetype with following script:

```
export TOOLCHAIN=~/work/freetype/ndk-standalone-19
export PATH=$TOOLCHAIN/bin:$PATH
export FREETYPE=`pwd`
./configure --host=arm-linux-androideabi --prefix=$FREETYPE/install --without-png --without-zlib --enable-shared
sed -i 's/\-version\-info \$(version_info)/-avoid-version/' builds/unix/unix-cc.mk
make
make install
```

It will compile and install freetype library into $FREETYPE/install. We will link to this install dir later on. It would be possible also to link openjfx font support dynamically against skia library available on Android which already contains freetype. It creates smaller result but can have compatibility problems.

Patching

Download patches javafx-android-compat.patch + android-tools.patch and patch jfx78 repository. I recommend to have look at patches. First one android-compat.patch updates openjfx build script, removes dependency on SharedSecret classes and updates LensLogger to remove dependency on jdk specific PlatformLogger. Second one android-tools.patch creates helper script in android-tools. The script helps to setup javaFX Android projects.

Building

Now is time to try the build. Run following script:

```
JAVA_HOME=/opt/jdk1.7.0
JDK_HOME=/opt/jdk1.7.0
ANDROID_SDK=/opt/android-sdk-linux
ANDROID_NDK=/opt/android-ndk-r9b
PATH=$JAVA_HOME/bin:$ANDROID_SDK/tools:$ANDROID_SDK/platform-tools:$ANDROID_NDK:$PATH
gradle -PDEBUG -PDALVIK_VM=true -PBINARY_STUB=~/work/binary_stub/linux/rt/lib/ext/jfxrt.jar \
-PFREETYPE_DIR=~/work/freetype/install -PCOMPILE_TARGETS=android
```

If everything went all right the output is in build/android-sdk blocked URL

Create first JavaFX Android project

Use gradle script int android-tools. The script sets the project structure for you. Following command creates Android HelloWorld project which links to a freshly built javafx runtime and to a HelloWorld application.

- NAME is a name of Android project.
- DIR where to create our first project.
- PACKAGE is package name required by Android. It has nothing to do with a packaging of javafx application.
- JFX_SDK points to our recently built runtime.
- JFX_APP points to dist directory of javafx application. (where all application jars sit)
- JFX_MAIN is fully qualified name of a main class.

```
gradle -PDEBUG -PDIR=/home/user/work -PNAME=HelloWorld -PPACKAGE=com.helloworld \
-PJFX_SDK=/home/user/work/jfx78/build/android-sdk -PJFX_APP=/home/user/NetBeansProjects/HelloWorld/dist \
-PJFX_MAIN=com.helloworld.HelloWorld createProject
```

Now cd to the created project and use it like any other android project. ant clean, debug, uninstall, installd will work. I haven't tried it from any IDE Eclipse nor Netbeans.