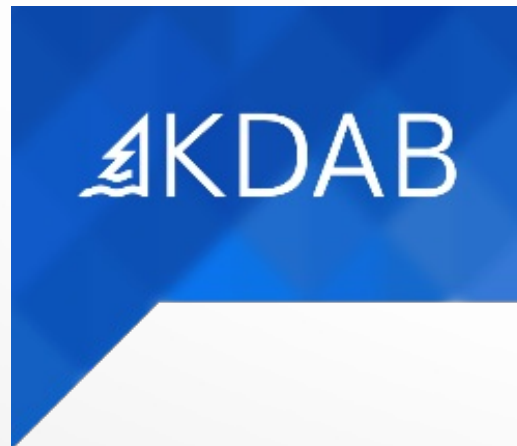


Testing & Profiling Qt on Android

BogDan Vatră, Programmer at KDAB



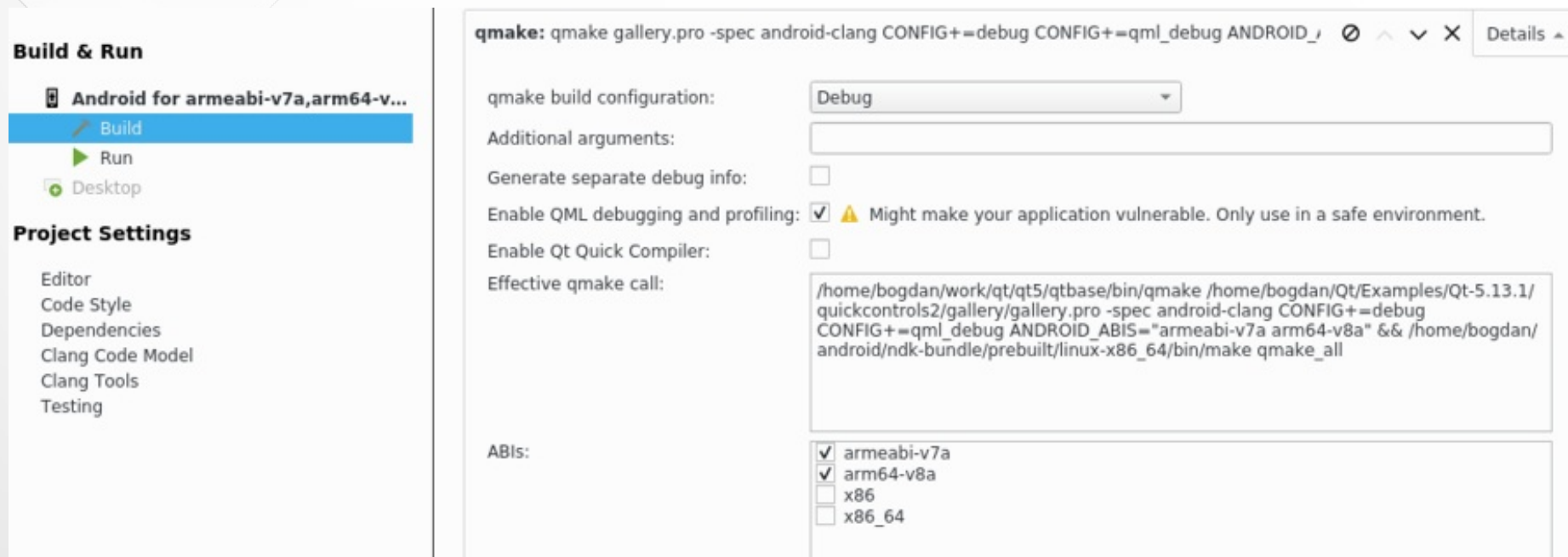
It's all about Qt 5.14!

KDAB

For all these features you'll need Qt 5.14 !

- **What's new on Qt 5.14 for Android ?**
- How to use Qt Test on Android
- How to use AVDs for running tests
- How to profile a Qt app on Android
- How to use address sanitizer on Android

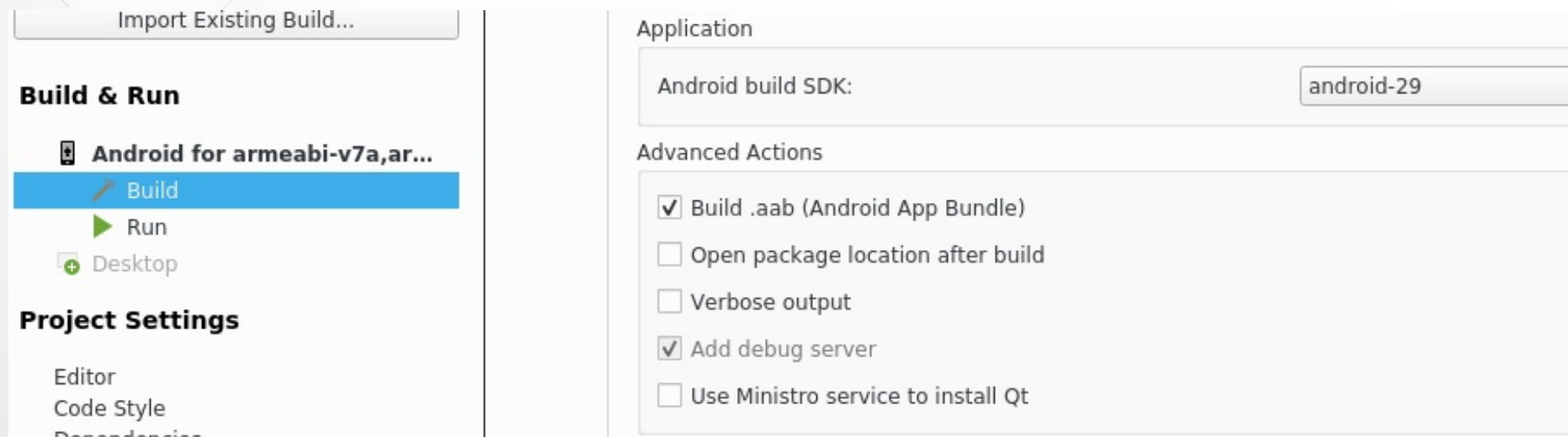
- the biggest feature added since I made the Qt on Android port
- by default all ABIs (armeabi-v7a, arm64-v8, x86 & x86_64) are built
- allows you to decide which ABI(s) you want to build for:



- from command line:

```
$ qmake ANDROID_ABI="armeabi-v7a arm64-v8a"
```

- Android App Bundles (aab) is the new (preferred) way to distribute your apps on Google Play
- generates and serves optimized APKs for each user's device configuration
- supports dynamic feature modules, via play core library



- from command line:

```
$ make aab
```

See <https://developer.android.com/guide/app-bundle> for more info

- Same as for **.aab** you can now create an **.apk** file from command line:

```
$ make apk
```

- reworked assets support: now it lists all the files **and folders**.
- load Qt plugins directly from android libs folder
- a few more, check <https://www.kdab.com/qt-for-android-better-than-ever-before/> for more info

- What's new on Qt 5.14 for Android ?
- **How to use Qt Test on Android**
- How to use AVDs for running tests
- How to profile a Qt app on Android
- How to use address sanitizer on Android

- how do we run tests on desktop ?

```
$ make check
```


How did we used to run tests on Android?

- usually we didn't
- there were (at least?) two scripts which used to help
- these scripts never worked properly ;(

- how should we run tests on Android ?

```
$ make check
```

This is how we are going to run tests on Android using Qt 5.14:

```
$ make check
```

Yes, now it's that easy!

- 25 minutes just tell us to run:

```
$ make check
```

There are a few Android specific things which you really need to know:

- say hello to **androidtestrunner**
- how to build & run your tests efficiently
- how to run your tests on a specific device/emulator
- how to pass arguments to **androidtestrunner**
- how to pass arguments to your test

androidtestrunner is a new tool added to Qt 5.14 which:

- creates the .apk (if it's not already created)
- installs the .apk
- runs the first Activity from **AndroidManifest.xml**
 - use **--activity** parameter to specify another one
- checks the test results
- if specified, pulls the tests output files in the build folder

Be aware: multiple instances will **wait for each other** to complete!

- build only the needed ABIs
- build your APKs in parallel !
- run the tests

```
1 $ qmake ANDROID_ABIS="x86_64"  
2 $ make -j$(nproc) apk  
3 $ make check
```

How to run your tests on a specific device

Use **ANDROID_DEVICE_SERIAL** environment variable

```
$ ANDROID_DEVICE_SERIAL="emulator-5554" make check
```


How to pass arguments to **androidtestrunner**

Use **TESTARGS** variable to pass params to **androidtestrunner**

```
$ make TESTARGS="--timeout 600" check
```

- use **TESTARGS** variable to pass params to **androidtestrunner**
- use **--** param to pass params to the test itself

```
$ make TESTARGS="-- -o out.xml,xml -o out.txt,txt -o -,tap -vs" check
```

- **--** following params will be passed directly to test app
- **-o out.xml,xml** stores the tests results in **out.xml**
- **-o out.txt,txt** stores the tests results in **out.txt**
- **-o -,tap** prints (to stdout) tap format
- **-vs** enables logging of every signal emission
- **out.xml** and **out.txt** are pulled by **androidtestrunner** to test build folder

- What's new on Qt 5.14 for Android ?
- How to use Qt Test on Android
- **How to use AVDs for running tests**
- How to profile a Qt app on Android
- How to use address sanitizer on Android

- use **sdkmanager** to install the emulator and a system image

```
$ .../tools/bin/sdkmanager "emulator" "system-images;android-29;default;x86_64"
```

- use **avdmanager** to create an AVD

```
$ .../tools/bin/avdmanager create avd -n test -k "system-images;android-29;default;x86_64"
```

- choose custom hardware profile
- choose **yes** for **hw.gpu.enabled**

- use **emulator** tool

```
$ .../tools/emulator -avd test
```

- pass **--no-window** param to **emulator** for docker hosts

- What's new on Qt 5.14 for Android ?
- How to use Qt Test on Android
- How to use AVDs for running tests
- **How to profile a Qt app on Android**
- How to use address sanitizer on Android

What we are going to cover:

- how to use NDK tools to do C/C++ profiling

What we are NOT going to cover:

- how to do QML profiling
- how to do Java profiling

- build & install the application using QtCreator
- use NDK tools to start profile recording
- start the application
- stop the application
- generate a report using the NDK tools

- build your Qt 5.14 app (in release mode) using QtCreator
 - pre Qt 5.14 strips the **.so** files in your android-build dir, which will cause the report step to not show you any symbols
- do NOT sign your application
 - signing the application will remove the debugable manifest flag.
- install the application on the target device

- go to **ndk_folder/simpleperf** folder
- run **./app_profiler.py** to start recording the profiling data. Pass at least:
 - **--app** parameter to specify the package name
 - **--lib** parameter to specify where is the build dir of your application
 - for more parameters please check https://android.googlesource.com/platform/system/extras/+master/simpleperf/doc/android_application_profiling.md

```
$ ./app_profiler.py --app org.qtproject.example.profile \  
--lib /home/bogdan/projects/build-profile-Release/android-build
```

Wait a bit until the profiler is ready for recording.

- start the application
- stop the application
- wait until **./app_profiler.py** pulls the recorded profile data

There are a couple of reporting tools in that folder, the most important ones are:

- **report.py** is a wrapper of the **perf report** command on the host

```
# Report call graph  
$ ./report.py -g
```

- **report_html.py** generates **report.html** file based on the profiling data

```
# Generate interactive chart statistics, sample table and flamegraphs, based on perf.data  
$ ./report_html.py
```

- **qtMainLoopThrea** is the thread that calls your **main** function
 - usually this is the thread that you're looking for
- **QtThread, QQmlThread** are Qt internal threads used by Qml Engine (Renderer)
- **RenderThread, Binder:XXXXXX** are Android internal threads
- **unnamed threads** usually are Android internal threads, but might be application's too
 - It's highly recommended to explicitly name all your application threads (use **QThread::setObjectName** before you start it)

- What's new on Qt 5.14 for Android ?
- How to use Qt Test on Android
- How to use AVDs for running tests
- How to profile a Qt app on Android
- **How to use address sanitizer on Android**

Same support as you find on GNU/Linux

- first class support for address sanitizer on Android in Qt 5.14

```
$ qmake CONFIG+=sanitizer CONFIG+=sanitize_address  
$ make apk
```

Yep, that's all you need to do!

Caveats:

- it seems it works only on **arm64-v8**
- it seems it works only on **Android 9+**
- it worked for me only on Google's Pixel 3

Thank you!



www.kdab.com

bogdan@kdab.com