Domain Specific Debugging Tools

Volker Krause
vkrause@kde.org

KDAB

General Purpose Tools

- Generic debuggers
  - Instruction-level debuggers (gdb)
  - printf
- Similar for profilers
- No understanding of your frameworks

KDAB

Framework Complexity

- Increasing abstraction
  - Qt model/view
- Asynchronous API
  - KJob
- Distributed architecture
  - Akonadi/Nepomuk
- Runtime interpreted code / JIT compiler
  - QML/JavaScript

KDAB

Debugging JIT'ed QML in GDB?

- Requires understanding of framework internals
- Inefficient
- In some cases even impossible for mere mortals
- In no way specific to Qt/KDE, but we tend to build nice frameworks :)

KDAB

Adapting the Tools

- Make tools understand the frameworks
  - No longer general purpose
  - Requires specialized tools for each framework
- Removes the need of knowing internals
  - Makes usage of a framework much more efficient
  - Increasingly important feature for your framework

KDAB

What do we have already?

- Developer API
  - q/kDebug operator<< overloads
  - ModelTest
- Introspection Tools
  - QDBusViewer
  - Akonadi Console, Plasma Engine Explorer, NepSak
  - GammaRay

KDAB

Do we have more?

- Built-in debugging features
  - QT_FLUSH_PAINT, QDBUS_DEBUG, ...
  - D-Bus interfaces to dump internal state (eg. org.kde.dfaure.dump)
- Full-blown development environments
  - QML tooling in QtCreator

KDAB

Should I build my own?

- Struggling with complex control flow
- Repeatedly adding the same non-trivial debug output
- Complex internal structures that benefit from a specialized visualization
- Performance metrics lacking correlation to the actual cost cause

KDAB

But it pollutes my code!

- Having development code built in is perfectly fine!
- Disable by default
  - compile time switches
  - runtime switches
- Called on-demand
  - e.g. using D-Bus to trigger additional diagnostics
- Introspection hooks/callbacks
  - overwriting with library preloading
  - registration of callbacks for interesting events

KDAB

Building Stand-Alone Tools

- Using your framework's public API
- Using introspection hooks
  - GammaRay provides you easy access to those of Qt
- Binary instrumentation
  - Pin, Valgrind
- OS-level tracing tools
  - DTrace, SystemTap

KDAB

Qt Introspection Hooks

- Load additional code into any Qt application
- Watch interesting events such as QObject creation/destruction
- Runtime information using QMetaObject et al.
- Powerful, but:
  - platform specific
  - hooks triggered before virtual tables exist
  - multi-threading

KDAB

GammaRay

- Framework for building Qt introspection tools
- Hides the nasty details
- Startup or runtime injection on Linux/Mac/Win
- Plug-in based with very simple API
  - object creation/destruction notification
  - flat object model
  - hierarchical object model

KDAB

Let's try!

KDAB

Where do I get it?

- http://github.com/KDAB/GammaRay
- Free Software (GPL)
- Mailing-list: gammaray-interest@kdab.com

KDAB

Conclusion

- Efficiency of using a framework is important
- Increased complexity requires better tooling
- Tools will help both framework authors and users
  - Makes your framework more efficient to use
  - Time invested in tooling easily pays off
- Consider turning your repeatedly added debug output into something more reusable :-)

KDAB

Questions?

KDAB