The Design and Implementation of KJSEmbed

Richard Moore, rich@kde.org

September 24th 2006, aKademy

Introduction

- KJSEmbed is a light-weight wrapper around KJS
- Available for KDE 3 for a long time
- Some parts are moving into kdelibs for KDE 4
- Allows applications to easily embed KJS
- Allows creation of simple applications
- Not a full binding to the KDE APIs
- Aim is to be simple and lightweight rather than complete

A Brief History of Bindings

- Early binding systems were manual (XSUB for perl)
- Tools like SWIG were created that automated this
- These tools were good at binding C
- These first generation tools had little or no support for C++
- Can't override virtual methods

A Brief History of Bindings 2

- A second generation of bindings tools designed for C++
- SIP (python)
- Smoke (language independent)
- Designed for C++ (and Qt) from ground up
- Create a subclass of each wrapped class
- Support for reimplementing virtual methods
- Support for signals and slots
- Support for inheritance

KJSEmbed/Qt3

- Falls somewhere between generations
- Intended only for scripting applications
- Not a full binding for C++ libraries
- Relies heavily on Qt features, not a general C++ solution
- Does not create a subclass of wrapped widgets

KJSEmbed/Qt3 Features

- Full access to all properties defined by any QObject
- Access enums defined by QObjects
- Call any slot
- Access slot return values (yes, slots can have return values)
- Connect signals to javascript methods
- Reimplement event handlers from javascript
- Understands inheritance so you can access bindings of the baseclass
- Support for loading UI files from designer
- Support for loading KParts

KJSEmbed/Qt3 Features

- All of the above you get free with any QObject, no binding effort required
- If you use Q_PROPERTY, Q_ENUMS and slots this is all free
- Also supports custom bindings for methods that are not slots
- Provides custom bindings for common non-QObjects
- Tool to generate custom bindings
- Many example scripts from a basic calculator to a basic browser
- Lots of reference documentation

KJSEmbed/Qt3 Implementation

- Transparently converts basic types between javascript and C++
- Holds pointers as void *
- Tracks the type of the objects itself
- Casts back to the correct type on demand
- This leads to problems with multiple inheritance
- There are also some problems ensuring things are cleanly deleted
- Poor use of JS prototypes leading to unnecessary overhead

KJSEmbed/Qt3 Event Handling

- Despite lacking support for virtuals you can reimplement event handlers
- Implemented using Qts event filters
- Detects when you assign to an event handler and installs the filter
- Uses a bitset to determine which events to filter
- Filtered events are passed to the javascript handler function
- Surprisingly fast – custom paint events work fine
- Lets us handle the most common case of virtual methods in Qt

KJSEmbed Qt 3 -> Qt4

- Attempt to keep the good parts of Qt 3 implementation but fix the problems
- Same design goals as Qt 3
- Make embedding the interpreter as easy as possible
- Keep things that worked well
- Try to make as much use of the QMetaObject as possible
- Keep the event handling mechanism as before

KJSEmbed/Qt4 Implementation

- Heavily template based
- Templates retain the real type information
- Uses template specialisation for types like QObject
- Retaining type information avoids multiple inheritance problem
- Lots of effort to ensure that startup is very fast
- Most of the features of the Qt 3 version are already implemented
- Simple scripts port with minimal changes

KJSEmbed/Qt4 Work In Progress

- Plasma integration
- Getting people to use the code!
- Ability to load custom bindings on demand
- Isolate the user from direct interaction with KJS
- New automatic binding generator
- Adding a standard library of utility functions
- Improving documentation
- Creating some real world examples
- Considering the relationship with QSA 2

Questions