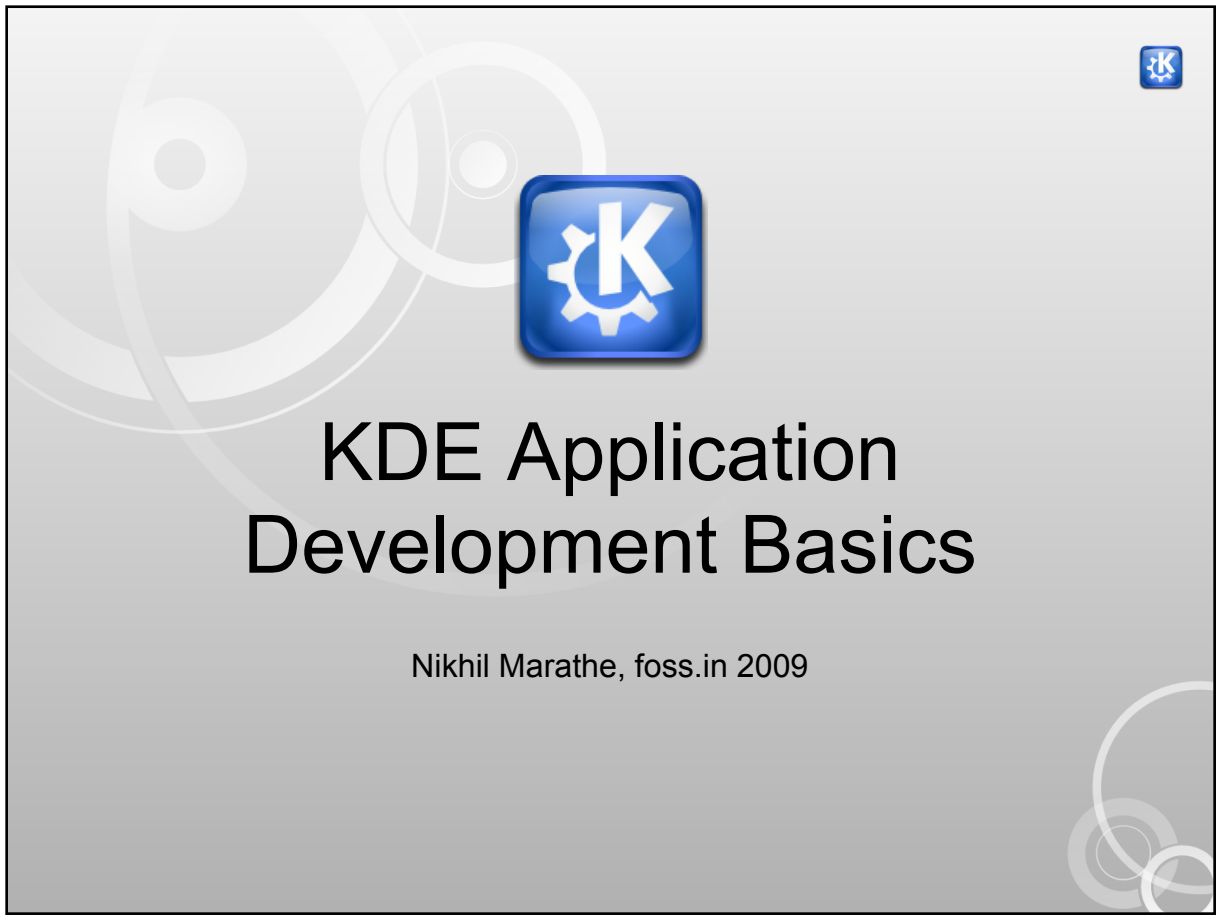


Good <time>

I'm Nikhil Marathe.
Currently in my second year in DA-IICT, Gandhinagar.

Got into KDE 7 years ago.
Developer since May. I am working on getting tiling window management into KWin.

Speaking of KDE, its really not just a desktop environment. With the recent rebranding, the term KDE Platform is appropriate.

For developers KDE provides a superb technology stack that allows you to write Killer applications in days.

Today we are going to get started with creating windows and managing user settings.

Whats here?

- Fast, automatic application creation with KXmlGuiWindow
- Saving configuration data with KConfig
- A quick look at KIO
- Twitter is the new Hello World

This is how the talk will proceed.

First we will take a look at creating an application, with a window, and compiling it.

Then we will take a look at how we can use KConfig for persistence and saving user preferences.

KIO is the KDE input output framework. It provides network-transparent asynchronous access to files. This way you don't have to care about the source and format of your data.

Finally we will combine these to create a simple application which will display the users twitter statuses.

Prologue

- Install CMake
- Install KDE development packages for your distribution
- Grab the code
  - http://github.com/nikhilm/foss.in

Before we begin, there are certain prerequisites.

I assume you are comfortable with C++ and have a basic idea of how Qt's signals and slots system works.

You should have a working KDE development environment. This usually only needs CMake and the relevant KDE development packages.

Finally grab the source code for the talk.

If you are creating a normal, window based application, you need a main window. In KDE you do this by subclassing the KMainWindow class. But KXmlGuiWindow is the recommended way to go, since it offers some niceties.

So before we take a look at the features, let's set us some basic code and have it compiled

: Go to twitterdemo_mainwin
: Show main.cpp
So here we have basic application setup, with metadata and so on. TwitterDemo is our subclass of KXmlGuiWindow, so we instantiate it.
: Show twitterdemo.cpp
 : constructor
    Here we make the call to the superclass then create our main widget and set up actions. We'll come back to this later. Here is the relevant CMakeLists to build this application

: Show CMakeLists.txt
so let's build this. The CMAKE_INSTALL_PREFIX is important, KXmlGuiWindow REQUIRES that the application be INSTALLED, but we really don't want it while in the development phase

So let's run our application and see what we get.

: cd /tmp/twitterdemo_mainwin
: ./bin/twitterdemo

: hover over menus
As you can see we have the about dialog, and configuration settings, and the message in the status bar.

KXmlGuiWindow gives you a lot of free stuff. For example, it will have an About dialog for your application, basic shortcut configuration, ability to switch languages on the fly and links to your application's Documentation.

In addition it provides a status bar widget and allows you to plug in a widget to occupy the window.

KDE/Qt has concepts of Actions. Actions are user triggered mechanisms which are connected to a slot, when the action is activated the slot will be called.

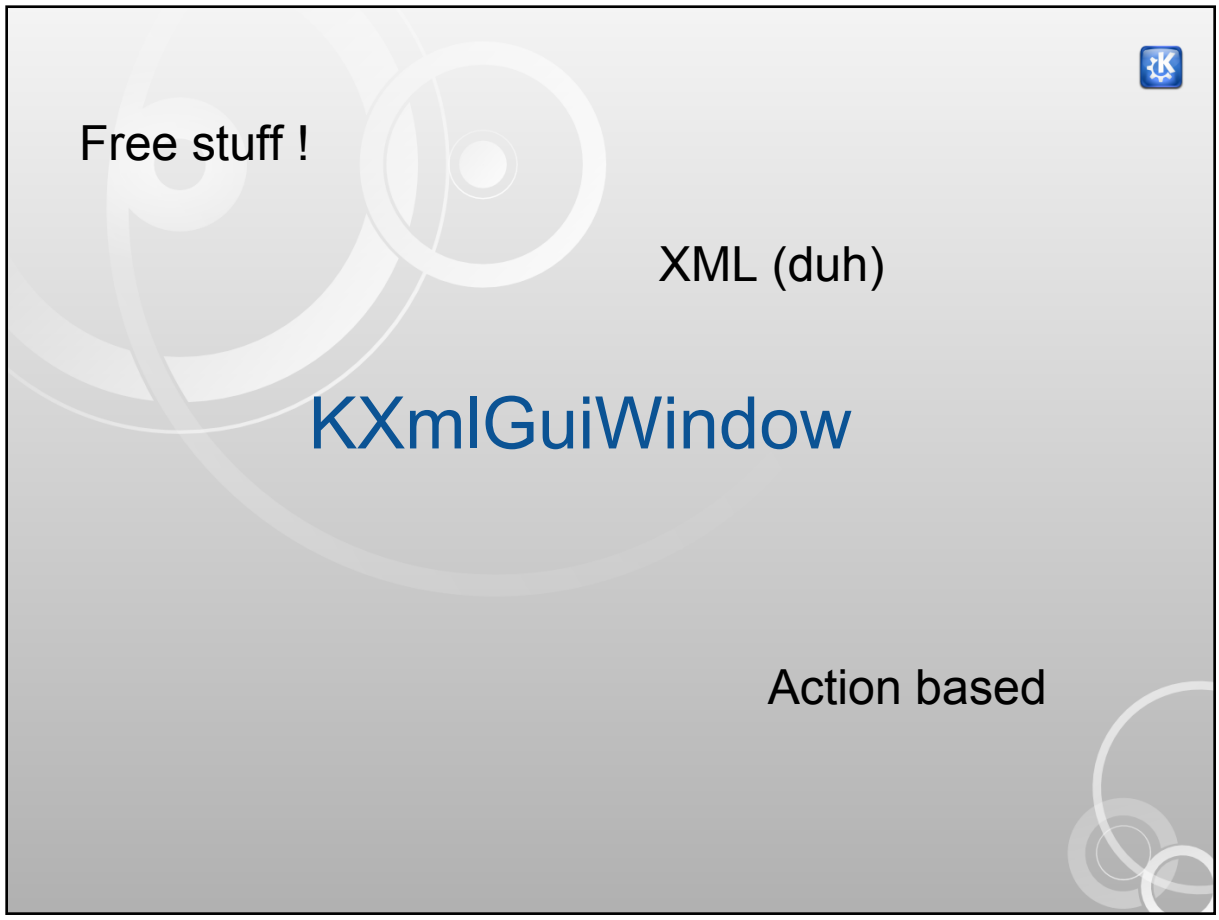
It has a description, icon and a shortcut

What makes it different is that an Action by itself is not shown to the user, it is an abstract concept.

You can take this action and add it to the menu or toolbars or any widget so that the user can interact with it.

: open twitterdemo.cpp again to setupActions()
So lets create a custom action for refresh. When creating an action, remember two things
1. Pass it the action collection
2. Remember the name you set in the addAction call, otherwise the action won't show up.

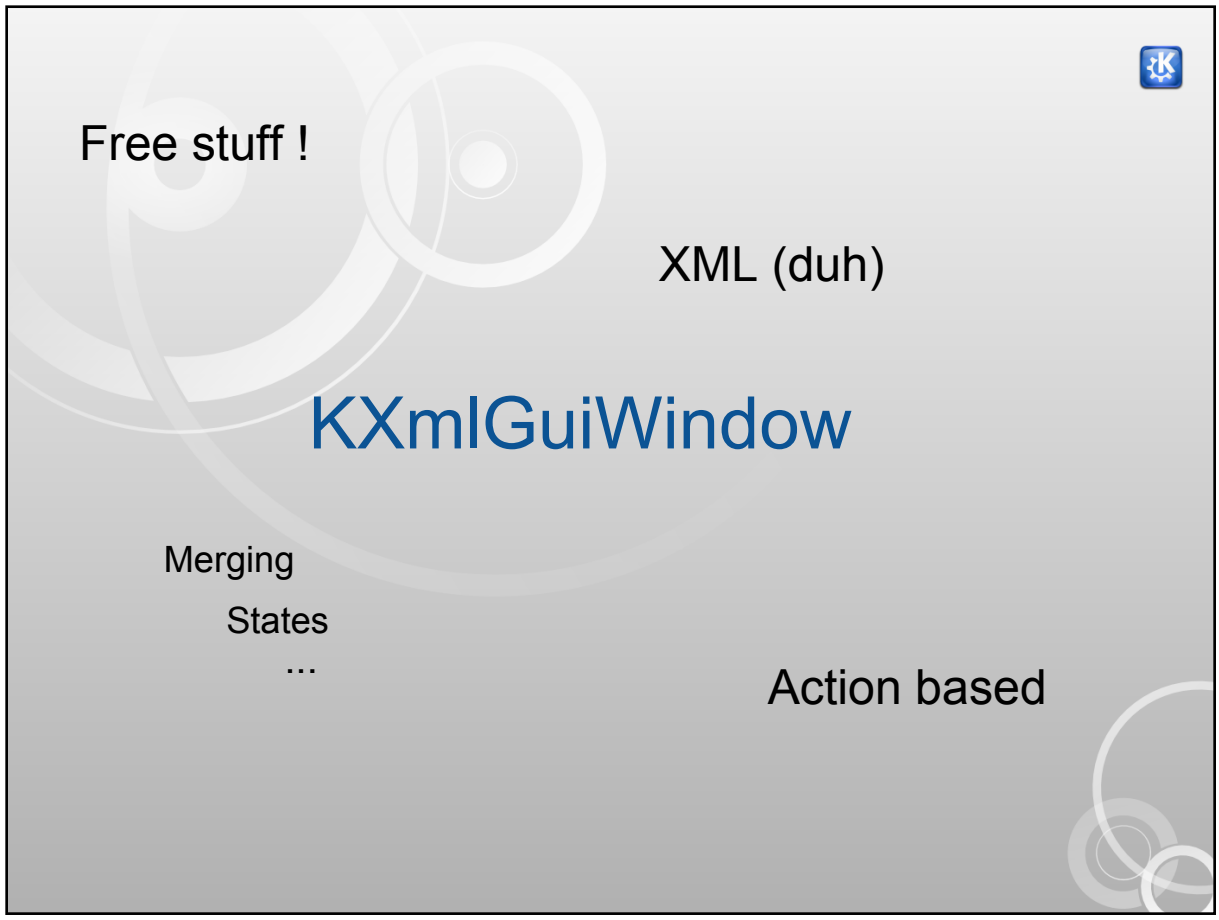
The view-refresh icon is a standard icon, so we don't need to pass any path.



When using KXmlGuiWindow, you usually won't express any layout details in your code. The preferred way to go is to create actions and assign slots in the code. Then an XML description file actually specifies which toolbars and menu receive which actions and so on. If you ever need to change the UI, there is no need to change the code or recompile it. The XML file also allows creation of menus and toolbars which don't require controller logic.

: switch to the xml file
: uncomment the part and explain
: edit CMakeLists and uncomment the line
File is a standard menu
: recompile and run

As an example of creating menus, lets move our refresh action to a new menu.
: uncomment the menu part
: Run make install

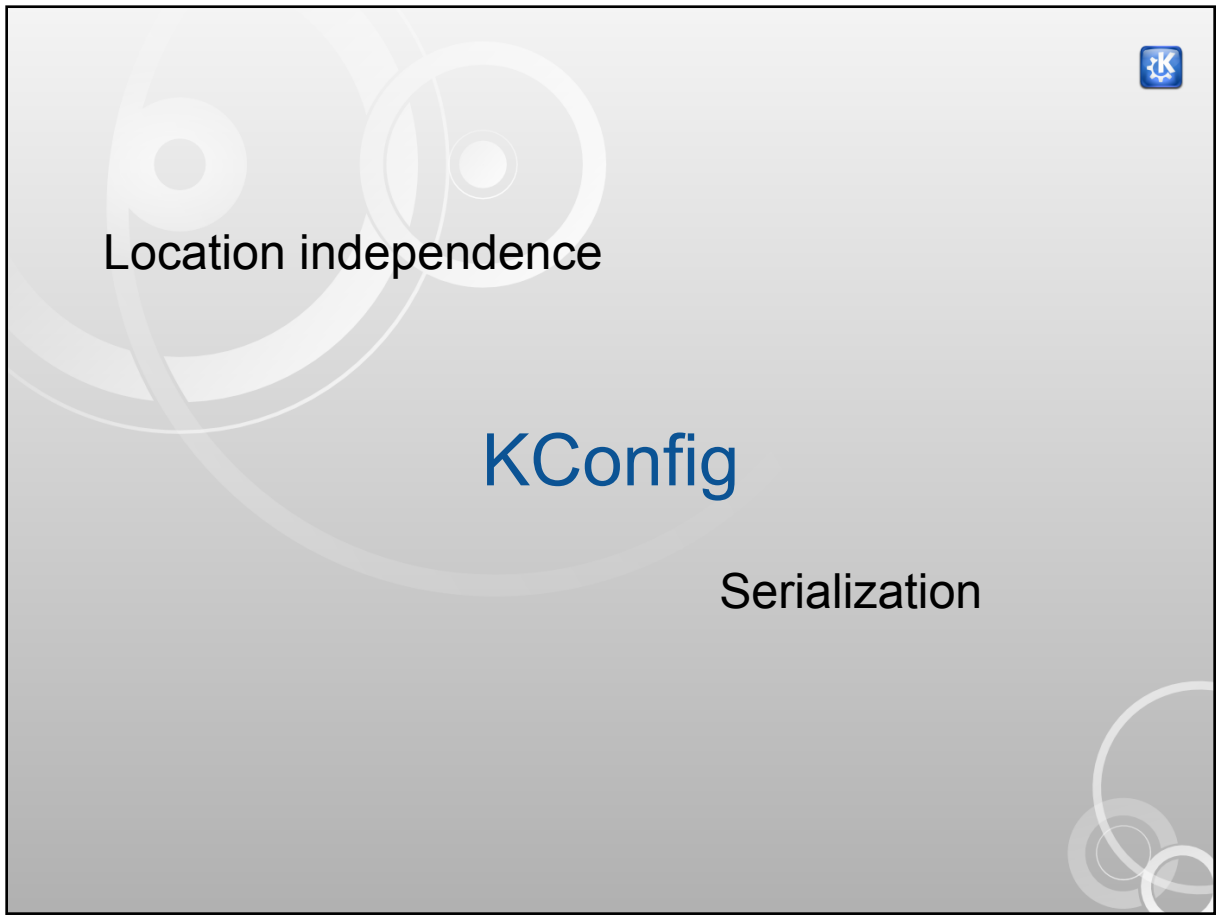


KXmlGuiWindow offers some advanced features like a more declarative action mechanism and merging of multiple files. I encourage you to take a look at that.

The traditional UNIX model is to use dot files for user preferences and their are a mindboggling number of formats for the files. The KConfig framework hides all of this platform and user specific details. Using the data you provided as part of the application about data, it will create the appropriately named configuration file in the user's KDEHOME.

Location independence

KConfig



Using the QVariant from Qt, it also supports saving and loading of arbitrary data types, including custom data types. So you can just dump the settings as long as you read them back as the same type.

Simple .ini configuration

```
[Compositing]
AnimationSpeed=3
Backend=OpenGL
...

[Plugins]
Plugin1=shutdown
...
```

So lets look at the code to use KConfig
: cd to twitterdemo_kconfig
: show twitterdemo.h, the KConfig and KConfigGroup members
: open twitterdemo.cpp, in constructor show config creation

In displayTweets we will load configuration entries using readEntry. The first argument is the key name, the second is the default value. The default value also decides the data type. See how we can read and write complex types like QDateTime and QRect.

The saveConfig slot writes using writeEntry.

: run and click on "Configure Tweet Fetcher" once
Now lets see how our file looks
: open ~/.kdemod4/share/config/twitterdemorc

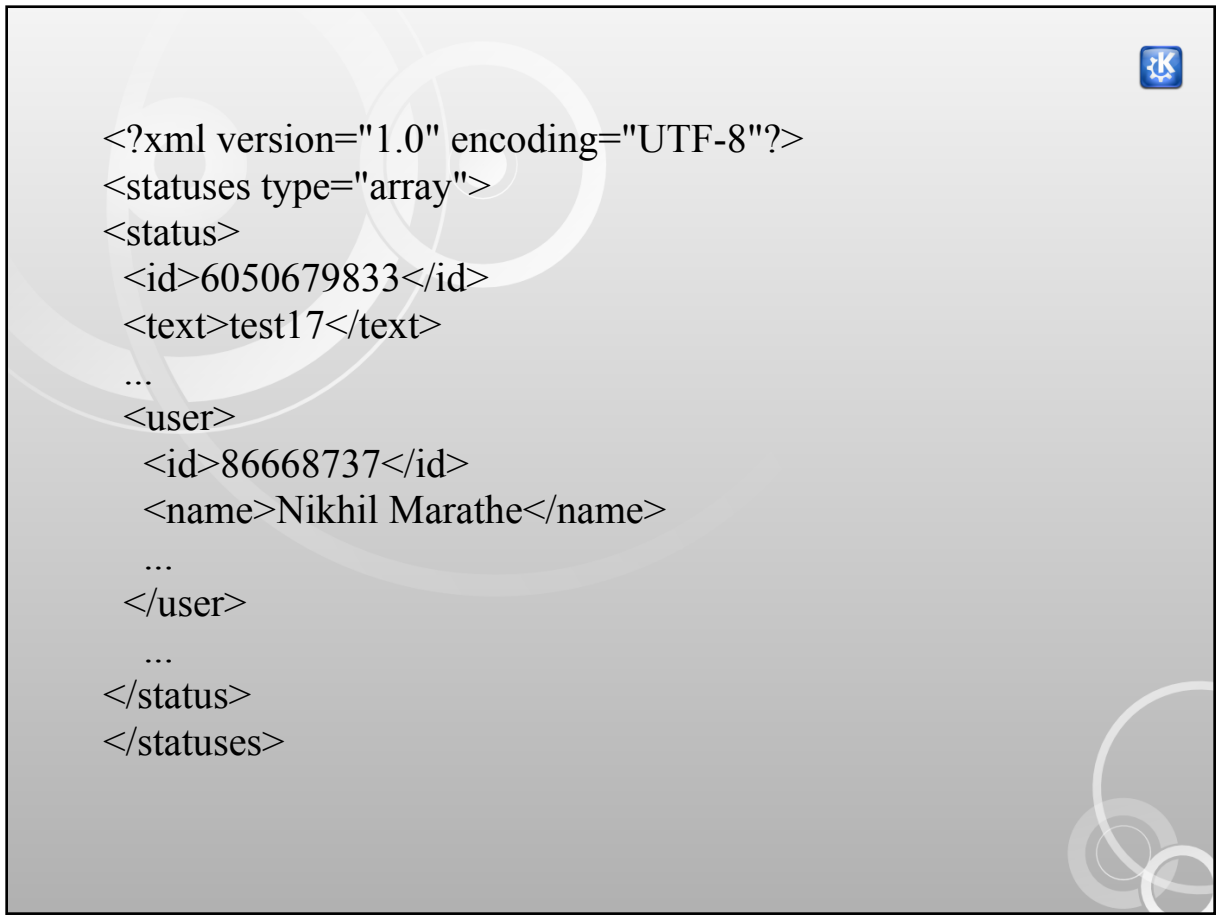
As you can see basic configuration is very easy.

While KConfig is great, with complex configuration you want some more features, for example, you would like to associate keys with certain widgets in your settings dialogs < fill this >. KConfigXT supports all this, the techbase has some great tutorials for this.

Put it together with a dash of KIO

Mix in Twitter

- Simple API over HTTP
- No complicated authentication
- KIO → XML → Tweets



: Show sample twitter output
The ID is important, since that allows us to fetch entries only after a certain point.

Lets go over to the final code now

: cd twitterdemo_final
: open twitterdemo.h
: explain methods
: explain m_responseData array.

: open twitterdemo.cpp
: constructor
First we create two groups, the general group will have the user name and the ID of the last tweet.
We parse any already existing xml using QDomDocument and load it into the list.

: go to loadTweet
load Tweet goes over the statuses and inserts them into the list with a bit of formatting


¿ Questions ?

Thank You

Nikhil Marathe

nsm.nikhil@gmail.com
http://kodeclutz.blogspot.com
http://22bits.exofire.net