Launching an application

How hard can it be?

2022/10/02, Akademy

Nicolas Fella

nicolas.fella@kde.org
@nicofee

Imagine

- Imaging you are writing a chat app
- Clicking on the file should open it

Starting a process

```
QProcess process("gwenview", {url});
process.startDetached();
```

- Starts a new Gwenview process
- Pass file URL as argument
- Start detached so it can outlive the parent process

Desktop Files

- Current approach assumes gwenview is in $PATH
- Does not work if shipped as Flatpak
- Use desktop files instead of commands

Desktop Files II

- Key-Value pairs describing app metadata
- Name, description, icon etc
- Exec key describes how to launch
  - Non-Flatpak: gwenview %U
  - Flatpak: /usr/bin/flatpak run --branch=stable --arch=x86_64 --command=gwenview --file-forwarding org.kde.gwenview @@u %U @@

Placeholders

- When opening files/URLs placeholders in Exec are replaced:
  - %f a single file
  - %F multiple files
  - %u a single URL
  - %U multiple URLs

KService

KService parses desktop files:

```
KService::Ptr gwenview = KService::serviceByDesktopName("org.kde.gwenview");

QString exec = gwenview.exec();
exec.replace("%u", url);
exec.replace("%f", url.toLocalFile());

QStringList args = KShell::splitArgs(exec);

QProcess::startDetached(args.takeFirst(), args);
```

Preferred Applications

- Some people prefer Koko over Gwenview
- How do we find the user’s preferred image viewer?
- XDG spec: “Association between MIME types and applications”

Which config file?

- $XDG_CONFIG_HOME/$desktop-mimeapps.list
- $XDG_CONFIG_HOME/mimeapps.list
- $XDG_CONFIG_DIRS/$desktop-mimeapps.list
- $XDG_CONFIG_DIRS/mimeapps.list
- $XDG_DATA_HOME/applications/$desktop-mimeapps.list
- $XDG_DATA_HOME/applications/mimeapps.list
- $XDG_DATA_DIRS/applications/$desktop-mimeapps.list
- $XDG_DATA_DIRS/applications/mimeapps.list

KApplicationTrader

Default application for PNG images:

```
KService::Ptr preferredImageViewer =
KApplicationTrader::preferredService("image/png");
```

All applications for PNG images, sorted by preference:

```
KService::List lst =
KApplicationTrader::queryByMimeType("image/png");
```

Window activation

- Sometimes opening a file doesn’t open a new window but a new tab
- Existing window should be raised
- On Wayland windows can’t raise themselves
- On X11 they can, except they can’t

XDG Activation

- Client A requests a token from the compositor
- Compositor grants token to A
- A transfers the token to B
- B uses the token to raise itself

cgroups

- cgroups are great for many things
- See “Next Generation Application Management” from Akademy 2020
- Applications should be launched into their own cgroup
- Needs talking to the cgroup controller (i.e. systemd)

Non-KIO Apps

- KIO-aware apps can handle many different protocols (ftp, sftp, nfs, webdav etc)
- Third-party apps usually don’t
- Problem: Passing KIO URLs from KIO to non-KIO apps
- KIOExec downloads file to a temp location
- KIO-Fuse uses a virtual filesystem

Portals

- Flatpak apps have limited access to the host system
- Interaction is done using portals
- OpenURI/OpenFile portal allows opening URLs/files externally

Launching an app

```
KService::Ptr app = ...;
auto *job = new KIO::ApplicationLauncherJob(app);
job->setUrls({url});
job->start();
```

- Runs the given KService
- Optionally with URLs
- Passing an empty service asks the user which app to use

Open URLs

```
auto *job = new KIO::OpenUrlJob(url);
job->start();
```

- Determines mimetype for the URL
- Launches default application

Run Commands

```
auto *job = new KIO::CommandLauncherJob(“kate”, {“-n”, url});
job->start();
```

- Executes a command + arguments
- Does cgroup integration + window activation
- Use when launching GUI apps without desktop files or complex arguments

Write Emails

```
auto *job = new KEMailClientLauncherJob;
job->setTo({"nicolas.fella@kde.org"});
job->setSubject("Nice Talk!");
job->setAttachments({someUrl});
job->start();
```

- Opens the preferred email app with prefilled data
- Works around quirks in various clients

Summary

- We learned that
  - Launching applications is hard
  - Why doing it properly is important
  - How to use KDE Frameworks to make it easy
- In the future
  - Fix all places to do this properly
  - Add support for other OS to KService/KIO?

Questions?