

AKADEMY

GRAZ 2026

Harald Sitter

David Redondo

KDE Development Best Practices

Harald

- KDE Developer
- Works for Techpaladin
- Likes Bananas
- Award Winning

David

- KDE Developer
- Works for Techpaladin
- Doesn't like Bananas
- Not Award Winning

DISCLAIMER

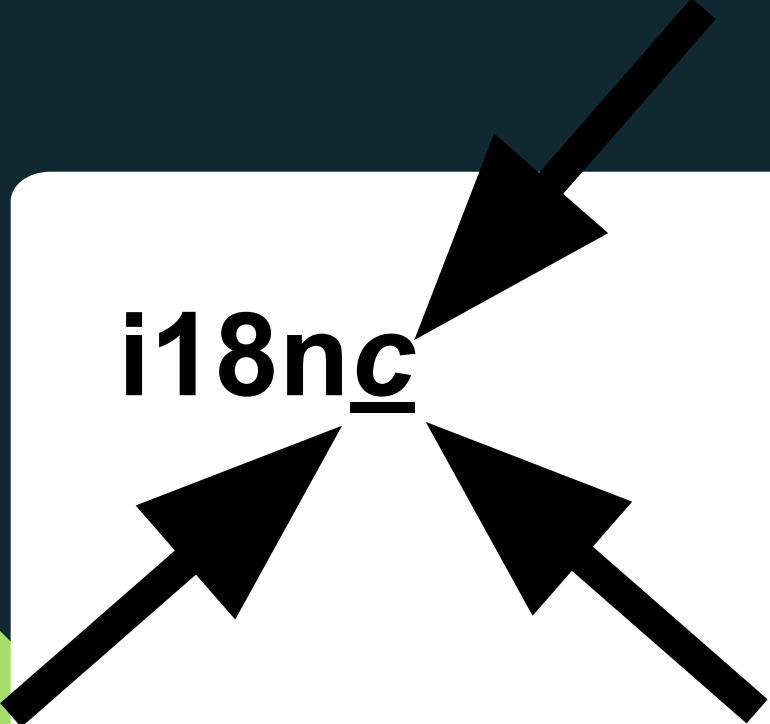
- We had better talks proposed
- They got rejected
- 💔🌹
- We are honest presenters

AGENDA

- Internationalization
- Linting
- QML
- Licensing
- CI
- Compiler Warnings
- KAuth
- C++
- Comments and Commit Messages
- Miscellanea

Internationalization

- `i18nc`
- `Messages.sh`
- `KI18n` (new!) in QML



i18nc

- Gives context to translators
- Be brief but expressive
- Also has semantic impact

https://techbase.kde.org/Development/Tutorials/Localization/i18n_Semantics

Messages.sh

- Give it a license ;)
- Make sure it works actually
- Simple check:

```
rm -rf po/*.pot;
```

```
../l10n-scripty/extract-messages.sh;
```

```
echo $?
```

- Try the xx language after pushing and waiting a day for it to magically appear

KI18n in QML

- Somewhat new addition
- Simply import and use `KI18n.i18nc` etc.

```
import QtQuick
```

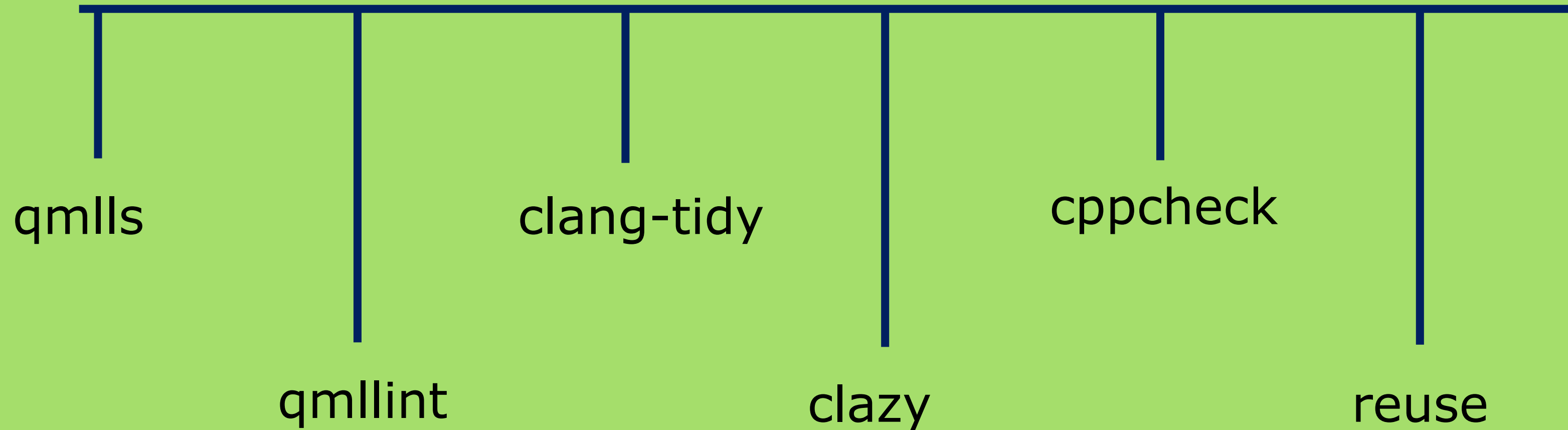
```
import org.kde.ki18n
```

```
Text {
```

```
    text: KI18n.i18nc("@label", "Hello World")
```

```
}
```

Lint all the Things



QML Thingamagos

- Use declarative registration so tooling can see things
- Use qmlsp and qmlint
- Always expected the Spanish Inquisition

Licensing

- all new files must have SPDX markers
- reuse lint
- Sunday lightning talk 17:15

Commit d26bb3d5 authored 1 week ago by 

Browse files

Options ▾

add manual test



3 files +107 -0



> CMakeLists.txt

+1 -0

> tests/CMakeLists.txt 0 → 100644

+9 -0

▼ tests/dbus-test.cpp 0 → 100644

+97 -0

```
1 + /*
2 + *   SPDX-License-Identifier: BSD-3-Clause
3 + *   SPDX-FileCopyrightText: 2026 David Redondo <kde@david-redondo.de>
4 + */
```

 **Merge request pipeline #1067349 failed**
Merge request pipeline failed for `6ec9185d` 10 months ago 
Test coverage 37.00% from 1 job 



8✓ Approved by 



 Merged by  10 months ago

Revert

Merge details

- Changes merged into master with `6ec9185d`.
- Deleted the source branch.

 **Pipeline #1067435 failed**
Pipeline failed for `6ec9185d` on `master` 10 months ago
Test coverage 37.00% from 1 job 



Don't merge on red

- Enforce passing tests
- Fix flakey ones ❄️





Don't merge on red



⚙️ Merge request pipeline #1077066 waiting for manual action



Merge request pipeline waiting for manual action for 3d511118 ?

8✓ Approved by [redacted]



🔗 Merged by [redacted] 10 months ago

Revert

Cherry-pick

Merge details

- Changes merged into master with [3d511118](#).
- Deleted the source branch.

❌ Pipeline #1077082 failed



Pipeline failed for 3d511118 on master 10 months ago

Let the tools help

- The compiler wants to help you
- Use KDECMakeSettings and KDECompilerSettings

```
warning: ignoring return value of 'virtual bool  
QFile::open(QIODeviceBase::OpenMode)', declared with attribute 'nodiscard'  
[-Wunused-result]  
38 |         f.open(QFile::WriteOnly);
```

KAuth/Polkit - Here be Dragons

- Beware the shortcomings (no concurrent actions in KAuth)
- Pay attention to security boundary
- Get security review
- Consider using polkit(qt) directly



Let the language help

- `std::span`
- `std::unique_ptr<T, D>`
- **views and ranges**
- **auto**
- `qScopeGuard`
- `std::to_underlying`
- `[[nodiscard]]`
- **exhaustive** switch
- `std::inout_ptr`

auto

- `auto` is cool, usually we don't care about the specific type but rather its semantics
- `QString` is the exception! Beware the `QStringBuilder` trap



`std::unique_ptr<T, D>`

- Everybody loves it but...
- Do not use it in API!
- The `D`!
- Mind the `D`!
- Prefer `shared_ptr`

std::span

```
int len = 0;  
auto some_c_array = foobar(&len);  
std::span fancy_span{some_c_array, len};
```

ranges and views

- everybody hates writing `foo.begin()` and `foo.end()`
- more algorithms than in the past
- projections are really useful
- combine with views to build expressive pipelines
- `std::views::filter`
- `std::views::reverse`
- `std::views::enumerate`
- `std::views::transform`

ranges and views

```
std::ranges::fold_left geometries, QRect(),
&QRect::united);
auto validServices = services |
    std::views::filter(&KService::Ptr::data);
for (auto &&[index, device] :
    std::views::enumerate(m_devices))
for (const QString &lang : languages |
    std::views::reverse) {
```

qScopeGuard

- Fancy defer pattern from another famous language

```
auto fd = open("/foo");  
auto close_it = qScopeGuard([fd] {  
    close(fd);  
});  
// closes when out of scope  
// can also be disarmed!
```

std::to_underlying

- Cool new trick to get the underlying type of an enum

```
enum Foo char {  
    One,  
    Two  
};  
  
char x = std::to_underlying(Foo::Two);
```

exhaustive switch

- avoid `default` by default
- list all values of the `enum`
- notable exception: C enums, veeeeeeeeery expansive enums where you definitely will only ever care about one or two things

[[nodiscard]]

- Bit contentious
- But prevents you from unexpectedly not doing anything with a return type
- Super amazing for error returns
- But also for others because you have to be explicit
- Combine with `std::ignore`
- Should really be a language feature :(

std::(in)out_ptr

- Very useful for C APIs that want foo**

```
std::unique_ptr<pam_handle_t> handle;  
pam_start(s, nullptr, &c, std::out_ptr(handle));  
// handle now contains the pam_handle_t*
```

Comments and Commit Messages

- good comments can be helpful
- bad comments will confuse you

```
QString UserInfoDialog::image() const
{
    // TODO: Ideally plasma-welcome and kcm_users should always set an avatar (if need be to a render of Kirigami.Avatar)
    // such that the user always has a valid avatar 99.9% of the time.
    if (!m_userDetails->m_iconFileExists) {
        // Always provide a fallback. We **must** provide an avatar to XDP per its documentation and code.
        // Legacy users may not have one, systems without Accounts service may not have one, some software may clear it...
        static const auto iconURI = [] {
            const auto iconPath = KIconLoader::global()->iconPath(u"user"_s, -KIconLoader::SizeEnormous, /* canReturnNull = */ false);
            return u"file://"_s + QFileInfo(iconPath).canonicalFilePath();
        }();
        // WARNING: do not use symlinks. They make XDP forward incorrect paths to clients because it will portal the canonical path
        // but forward the original (symlink) path
        return iconURI;
    }
    return QUrl::fromLocalFile(m_userDetails->m_iconFile).toString();
}
```

```
if (mFocus) {
    // FIXME!
    if (auto *ww = mFocus->waylandWindow())
        window = ww->window();
}
```

```
// Dave, this is wrong
auto *ww = surface->waylandWindow();
if (!ww || !ww->window())
    return;

tp.area = QRectF(0, 0, 8, 8);
const QPointF localPosition = ww->mapFromWlSurface(surfacePosition);
```

- comments vanish – the commit message endures

Commit Messages

- Explain the What and the Why, not the How
- Be concise but descriptive
- Do not repeat what the code already says
- Will the commit message still make sense in a year?
- What about in  years?

Miscellanea

- Use `-symbolic` icons when you want symbolic icons
- Reach out to domain experts
- write unit testable code so the CI can run sanitizers for you

Checklist

- ☐ Are all user facing strings translatable and do they have context?
- ☐ Do linters pass?
- ☐ Are all files licensed? Does reuse lint pass?
- ☐ Does it compile without warnings?
- ☐ Can qmls resolve types?
- ☐ Using modern C++?
- ☐ Good commit messages?

QUESTIONS





Thank You

sitter@kde.org
david.redondo@kde.org

