Metadata track

Akonadi
the independent solution for PIM data

Will Stephenson

Topics

Akona-what?

Design Overview

What we give you

What you give us

The story so far

- Monolithic apps
  - Own data storage
  - Limited if any external interfaces
- E-D-S
  - Data storage service
  - Limited range of types supported

Why?

Limitations of KDE3

KResource framework limitations:

Data is not shared

Designed for synchronous access

Hard to extend to other data types

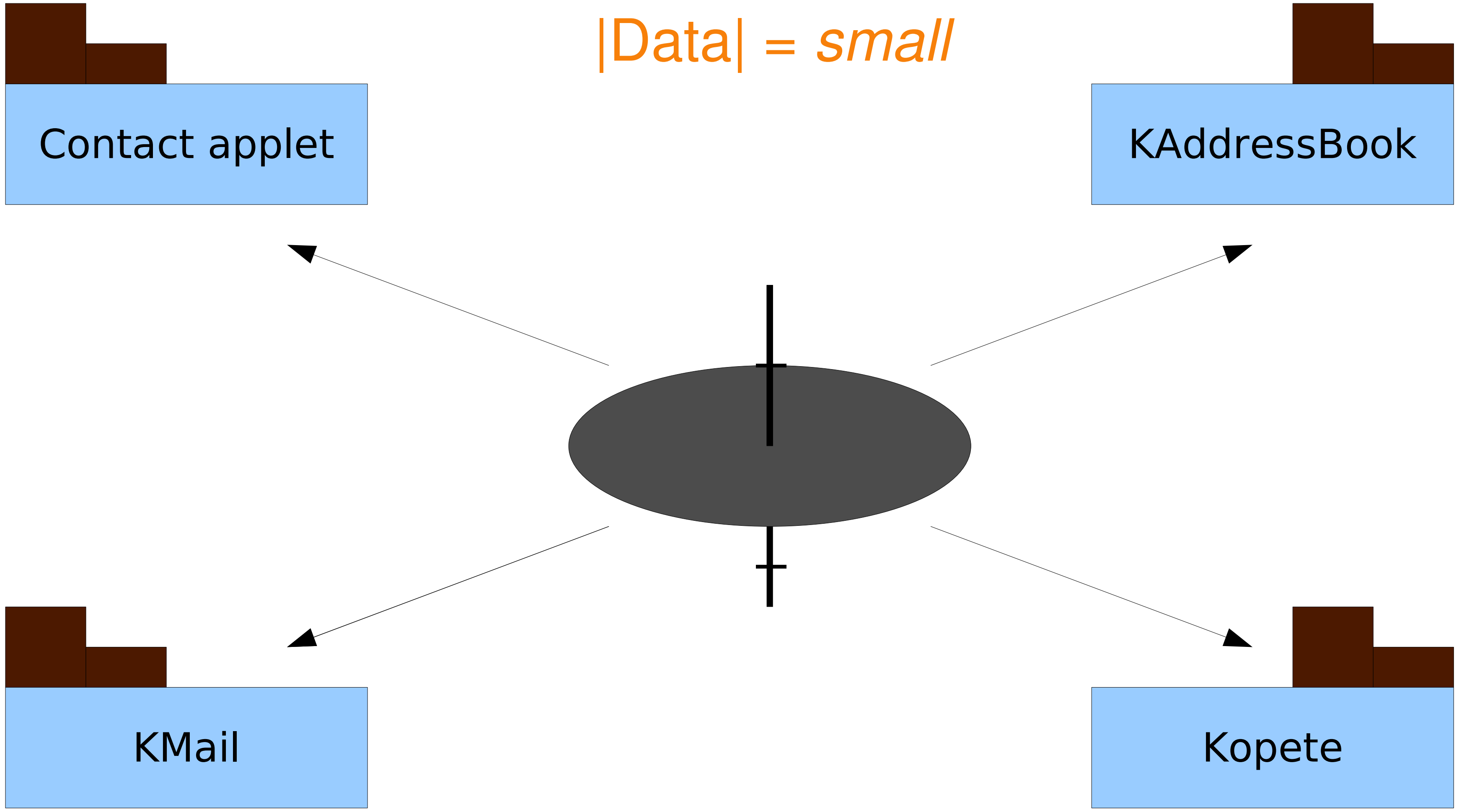
Basically no shared common code

KMail limitations:

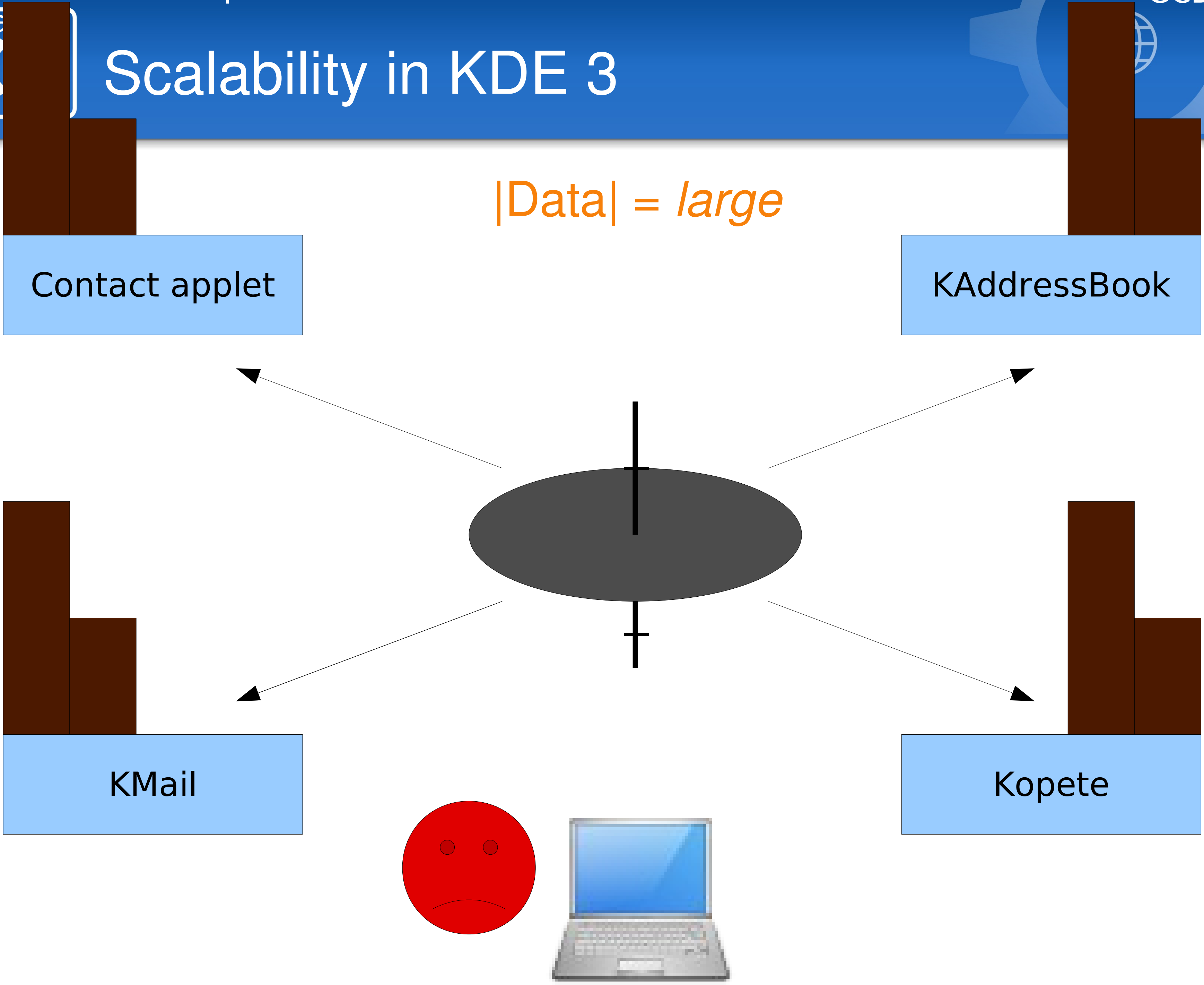
Only limited backend abstraction

Designed for small amounts of local data

Scalability with KDE 3



Scalability in KDE 3



Why?

Goals

As much as possible shared, type independent functionality

Easy to extend to new data types

Unified API to access PIM data, independent of the actual data source

Scalability

Why?

Goals

One synchronization point for mobile devices

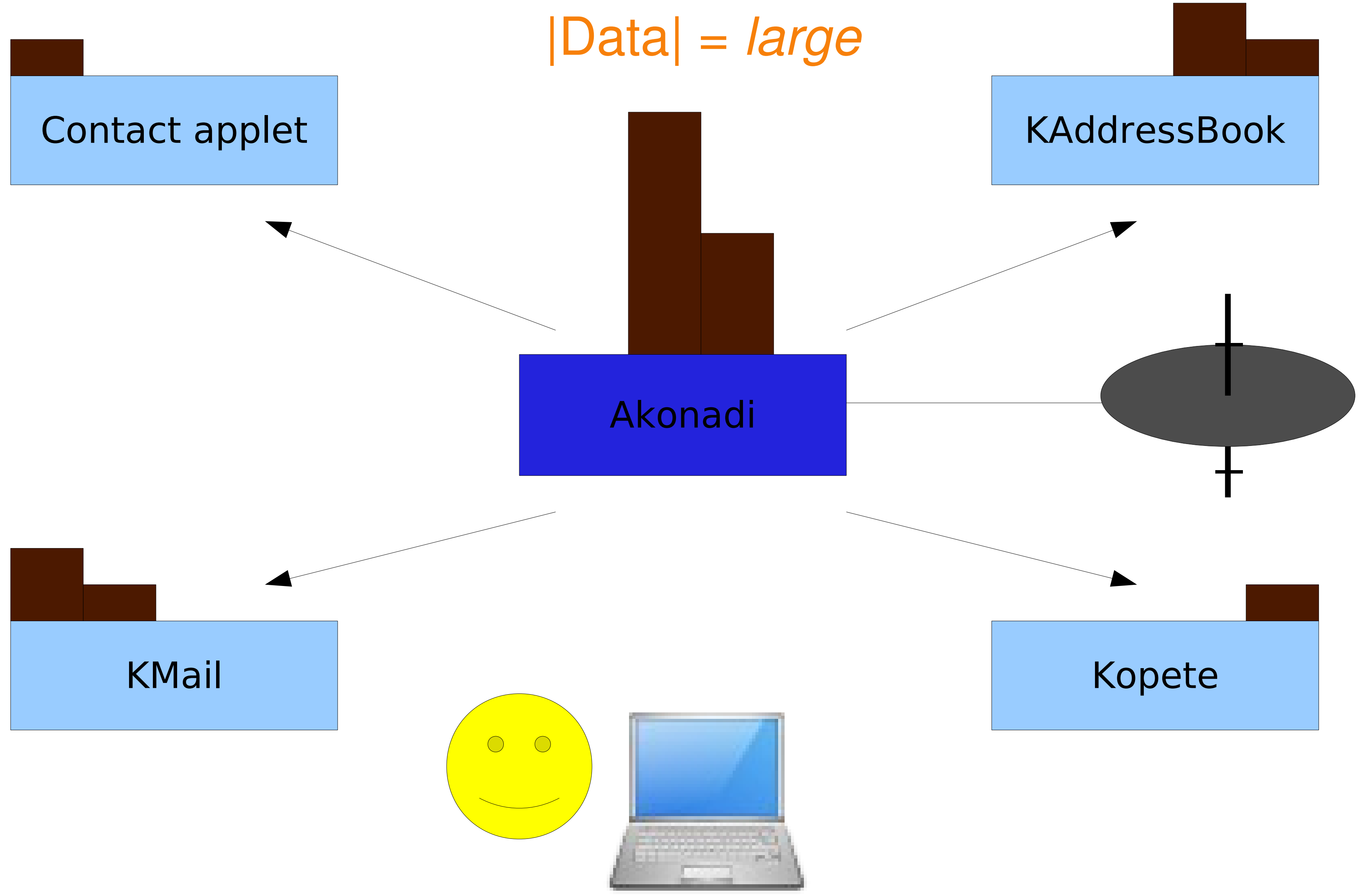
Reliable, desktop wide notification

Clean model/view separation (UI-less data access)

Easy to write access libraries for

Usable for the whole free desktop

Scalability in KDE 4 with Akonadi



Why?

Enabling new use cases

“show me the log of the last IRC chat I had with the person who sent me this mail”

Why?

Enabling new use cases

“show me all mails with pdf attachments mentioning my hamster 'cookie' right here inside my IM client, whenever someone mentions chicken curry”

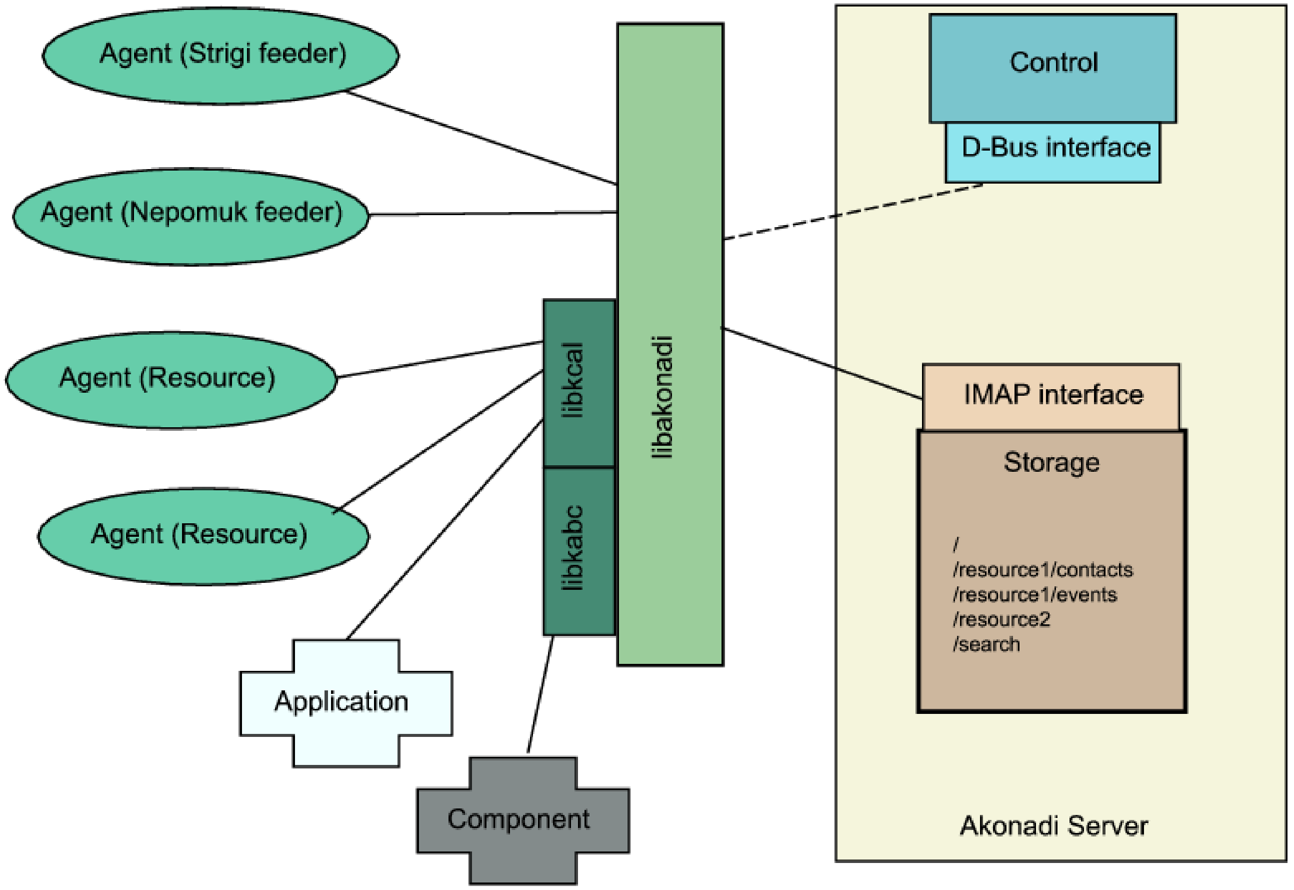
Why?

Enabling new use cases

“tell me when I get new mail in this folder and this other folder, and show it on the desktop, but only if it's not from my mom. show me a picture of the person next to it, and when I have an appointment with them, if I do. allow me to cancel that appointment by dragging it to the trash”

Design

Design Overview



Design

Server

Fully type independent

Cache for remote data with variable cache policies

Change notification

Conflict detection

Design

Basic Objects

Filesystem-like structure:

Collections

Items

Items can consist of multiple parts so clients can access only the actually needed data

Items can be polymorphic

Design

Client/Server Communication

Two communication channels:

D-Bus for control data

IMAP-like protocol for content data

Standard formats for content data (MIME, iCal, vCard, etc.)

Toolkit and language independent interface

Design

Client Libraries

Currently only one: libakonadi, C++/KDE

Consists of type-independent part and type specific plugins

Provides low-level access to Akonadi objects as well as high-level components

Design

Resource Agents

Connect Akonadi to external data sources

local files (maildir, iCal, vCard, ...)

mail- or groupware servers

web services

Translate data formats

Replay offline changes

Design

Other Agents

Implement functionality not limited to one application as separate agents

Existing agents:

Search index feeder

Mail threading

Planned agents:

Filtering

What we give you

Requirements

Server:

D-Bus

Qt 4.5

MySQL Server binary, does not need to be configured and running

Clients:

recent kdelibs + kdepimlibs

What we give you

How do I use it?

Starting/stopping Akonadi:

```
akonadictl start/stop
```

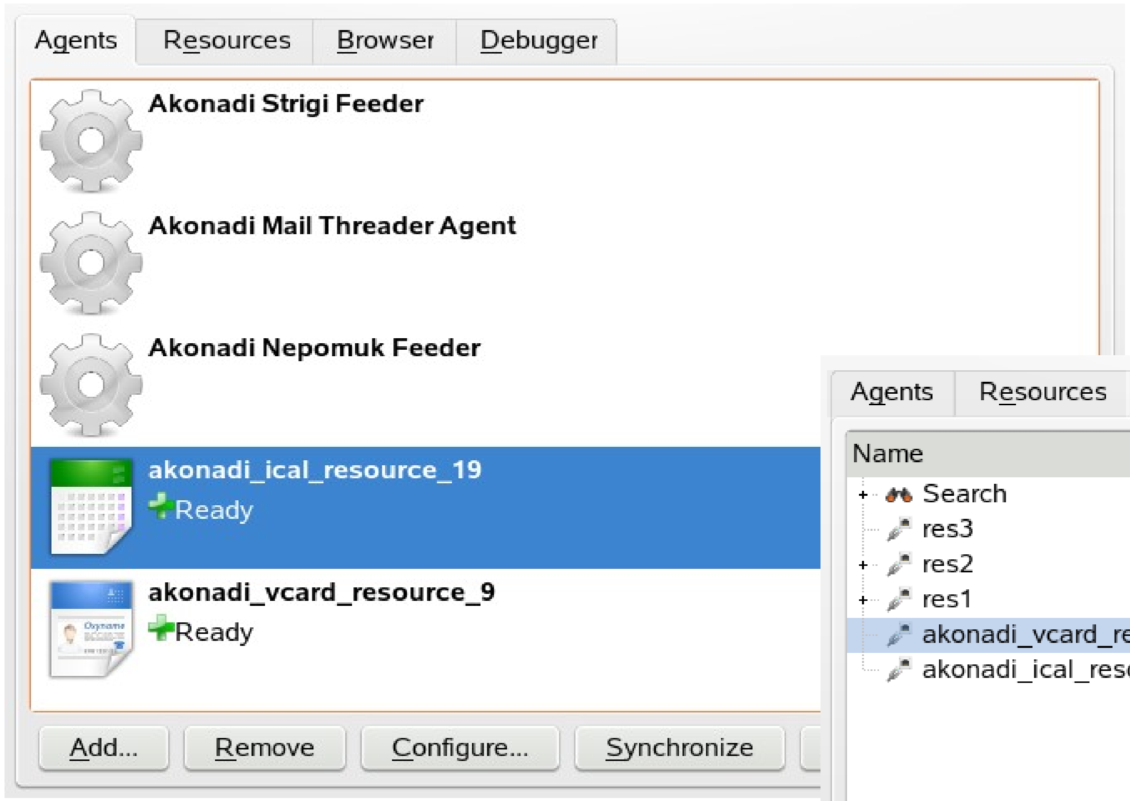
Akonadi Console:

Manage resource agents

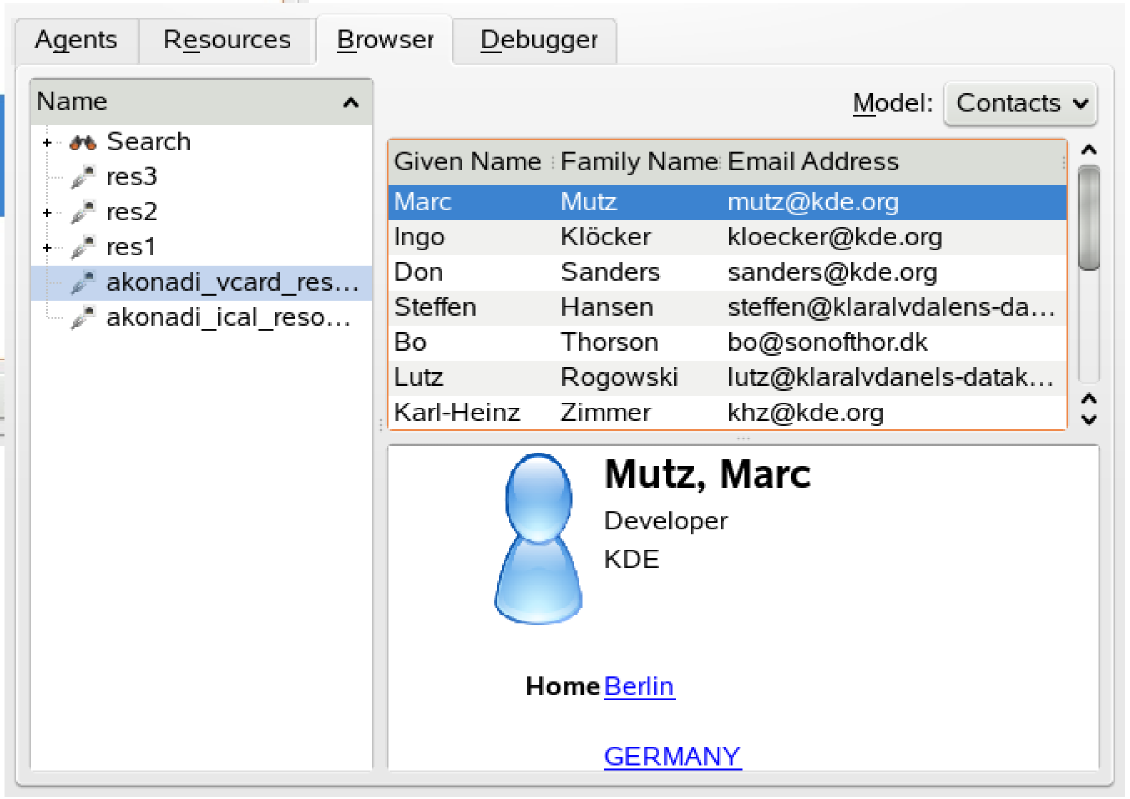
Browse content

Watch client/server communication

What we give you



Akonadi Strigi Feeder
Akonadi Mail Threader Agent
Akonadi Nepomuk Feeder
akonadi_ical_resource_19
Ready
akonadi_vcard_resource_9
Ready



Given Name	Family Name	Email Address
Marc	Mutz	mutz@kde.org
Ingo	Klöcker	kloecker@kde.org
Don	Sanders	sanders@kde.org
Steffen	Hansen	steffen@klaralvdalens-da…
Bo	Thorson	bo@sonofthor.dk
Lutz	Rogowski	lutz@klaralvdanels-datak…
Karl-Heinz	Zimmer	khz@kde.org

Mutz, Marc
Developer
KDE

Home Berlin

GERMANY

The state of the art

Roadmap

Optimisation and server diversification

Port existing KDE applications

More native resources

More client libraries

Optimisation and server work

Optimisations in the API implementation

Making the client API richer based on porting experience

Postgresql backend

Sqlite backend

External mysql instance support (thin client)

Unit tests

KDE Porting progress (1)

Until now, bridge resources access Akonadi via KResource API

Some apps ported completely (KJots, Mailody)

Others in progress (Akregator, KPilot, KNode)

KAddressBook being reimplemented (codename KContactManager)

The state of the art

KDE Porting progress (2)

Big apps (KMail/KOrganizer)

Refactoring to allow port

Simultaneous development on Model/View components for Akonadi

KDE Porting progress (3)

Mail migrator

Akonadi outbox agent – procmail

http://techbase.kde.org/Projects/PIM/Akonadi/PortingStatus

The state of the art

More native resources

SyncML agent

Google data resource

Exchange resource

IMAP resource

Semantic data extraction and semantic search folders

Kolab groupware

Contribute

More client libraries!

Additional Client library implementations

Language bindings

Contribute

Client Libraries

Currently only one: KDE/C++

Possible approaches:

Native implementations:

Native data types, easy integration

Language bindings:

Scripting languages, RAD

Contribute

Extending Akonadi

Support additional backends: groupware servers, web services, ...

Support for additional data types: IM messages, [micro]blogs, CRM/ collaboration

See “How to write an Akonadi resource in 30 minutes” next!

Using Akonadi in Applications

Port existing applications

New possibilities:

Integrate PIM data wherever useful:

Every mail address can be linked to your addressbook

Every date can be linked to your calendar

Plasma applets / Desktop widgets

Contribute

Further Information

IRC: #kontact on irc.freenode.org

Mailinglist kde-pim@kde.org

http://pim.kde.org/akonadi

Next talk!

KMail 2 – The Road to Akonadi – Mon 1715

http://techbase.kde.org/Projects/PIM/Akonadi/PortingStatus

And finally

Take away message

Engineered for performance

Engineered for flexibility

Engineered for independence