Developing PIM Applications with Akonadi

Till Adam / Volker Krause

Overview

Topics

- Accessing Akonadi
  data types, low-level access to items and collections, change notifications
- High-Level Components
  collection and item models and views
- Extending Akonadi to support new content types
  writing serializer plugins, extending models, developing resources
- How Can I Contribute?

Accessing Akonadi

Topics

- Data Types
- Asynchronous Access
- Accessing and Manipulating Items
- Accessing and Manipulating Collections
- Change Notifications

Accessing Akonadi

Akonadi::Item

- Represents a single object stored in Akonadi
- Generic Properties
  - persistent unique identifier
  - remote identifier
  - mimetype
  - revision number for conflict detection
- Contains a payload object with the actual content
- Can contain additional parts

Accessing Akonadi

Item Payloads

- Retrieving the item payload:
  `Akonadi::Item::payload<T>()`
- Setting the item payload:
  `Akonadi::Item::setPayload<T>(const &T)`
- Payload object properties:
  - Value-based, no pointers
  - Cheap to copy, e.g. implicitly shared classes
  - Can be achieved by using boost::shared_ptr

→ **API completely independent of payload type**

Accessing Akonadi

Item Parts

- Item payload is serialized for storage
- Serialized data can be split into multiple parts
- Client can retrieve only partial payloads to improve performance
- Example: An email message could be split into:
  - header information needed for listing mails in a folder
  - body text for displaying the mail
  - attachments

Accessing Akonadi

Akonadi::Collection

- Represents a collection of items
- Properties
  - persistent unique identifier
  - remote identifier
  - allowed content mimetypes
  - access rights
- Can contain additional attributes

Accessing Akonadi

Akonadi::Job

- Provides asynchronous access to Akonadi
- General usage
  - Create job object:
    ```
    FooJob* job = new FooJob( this );
    ```
  - connect to result signal:
    ```
    connect( job, SIGNAL(result(KJob*)),
             this, SLOT(fooResult(KJob*)) );
    ```
  - job is executed automatically when event loop is entered
  - Result slot is called once the job is finished

Accessing Akonadi

Item Jobs

- Akonadi::ItemFetchJob
  - Retrieve a single item or all items in a collection
  - Allows to specify which parts should be retrieved
  - Supports result streaming
- Akonadi::ItemAppendJob
- Akonadi::ItemStoreJob
  - Supports conflict detection
- Akonadi::ItemDeleteJob

Accessing Akonadi

Collection Jobs

- Akonadi::CollectionListJob
  - Retrieve a single or multiple collections
  - Supports result streaming
- Akonadi::CollectionCreateJob
- Akonadi::CollectionModifyJob
- Akonadi::CollectionDeleteJob

Accessing Akonadi

Akonadi::Monitor

- Emits signals once an item or collection has been added, modified, moved or deleted
- Supports filtering to only monitor objects of interest
- Supports change compression to improve performance

Components

Topics

- High-Level API
- Generic Item Model
- Collection Model
- A mail application in 30 lines ;-)

Components

High-Level API

- Asynchronous API necessary but cumbersome to use
- Too low-level for many common tasks
- Better: Qt's model/view framework
- Provide extensible models that encapsulate most of the commonly needed access of the Akonadi server

Components

Akonadi::ItemModel

- Base class for type specific models
- Takes care of:
  - retrieving items of a selected collection
  - update items on changes
  - copy & paste
  - drag & drop

→ **Functionality is available for all type specific models**

Components

Specialized Item Models

- Type specific models for
  - email messages
  - contacts
- Proxy models
  - email threading proxy model

Components

Akonadi::CollectionModel

- Provides a collection tree
- Takes care of:
  - retrieving collections
  - updating on changes
  - copy & paste
  - drag & drop

Components

Akonadi::CollectionFilterProxyModel

- Can be used on top of the collection model
- Allows to limit the collection tree to only show collection supporting content of a given mimetype
- Example: Show only email collections

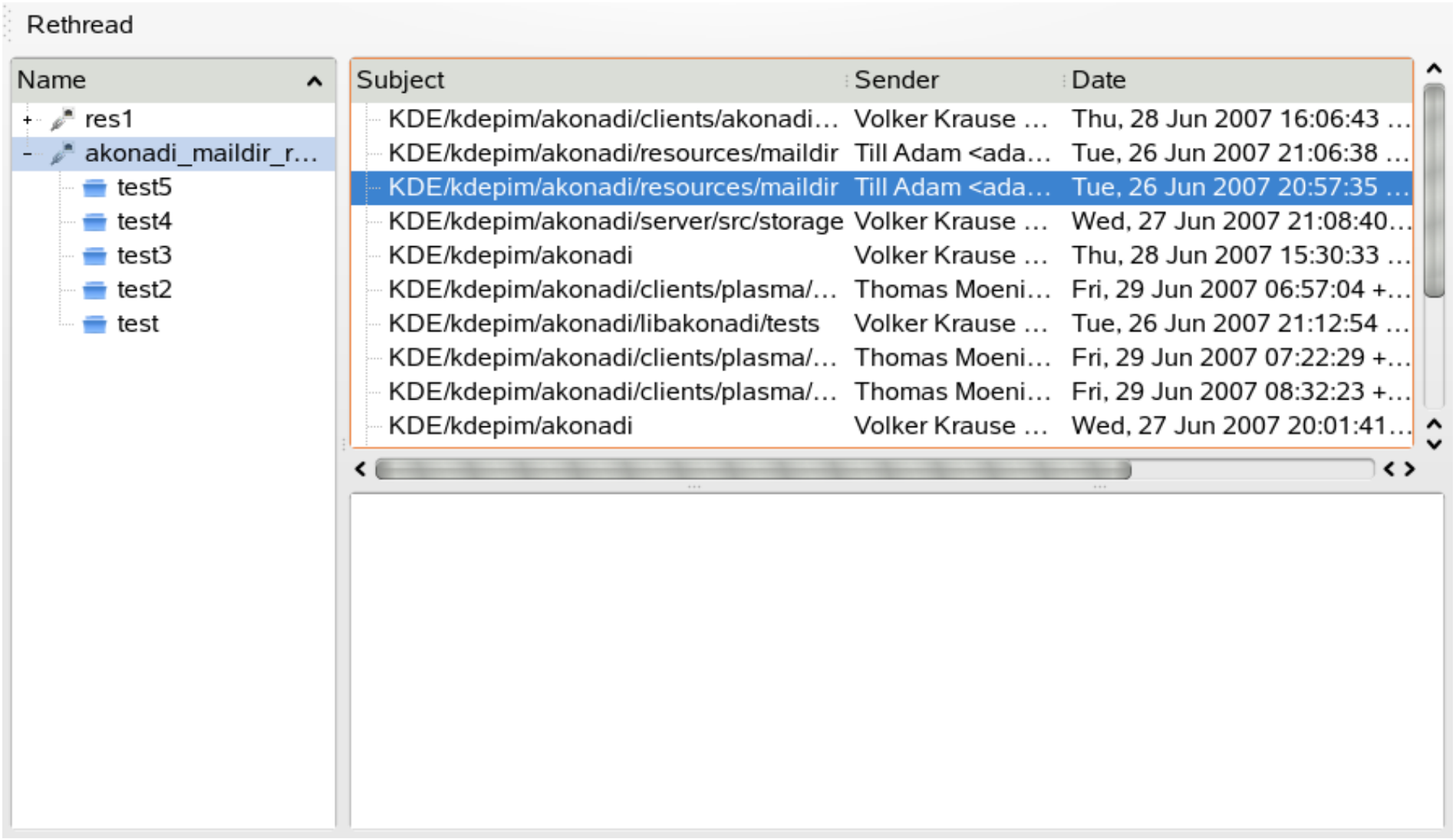
Components

A mail client in 30 lines ;-)

```
// collection view
collectionView = new Akonadi::CollectionView( this );
connect( collectionView, SIGNAL(clicked(QModelIndex)),
         SLOT(collectionActivated(QModelIndex)) );
collectionModel = new Akonadi::CollectionModel( this );
collectionProxy = new Akonadi::CollectionFilterProxyModel( this );
collectionProxy->setSourceModel( collectionModel );
collectionProxy->addMimeType( "message/rfc822" );
collectionView->setModel( collectionProxy );

// message view
messageList = new QTreeView( this );
connect( messageList, SIGNAL(clicked(QModelIndex)),
         SLOT(itemActivated(QModelIndex)) );
messageModel = new Akonadi::MessageModel( this );
messageProxy = new Akonadi::MessageThreaderProxyModel( this );
messageProxy->setSourceModel( mMessageModel );
messageList->setModel( mMessageProxyModel );
```

Components


Rethread
Name
res1
akonadi_maildir_r...
test5
test4
test3
test2
test
Subject
Sender
Date
KDE/kdepim/akonadi/clients/akonadi... Volker Krause ... Thu, 28 Jun 2007 16:06:43 ...
KDE/kdepim/akonadi/resources/maildir Till Adam <ada... Tue, 26 Jun 2007 21:06:38 ...
KDE/kdepim/akonadi/resources/maildir Till Adam <ada... Tue, 26 Jun 2007 20:57:35 ...
KDE/kdepim/akonadi/server/src/storage Volker Krause ... Wed, 27 Jun 2007 21:08:40...
KDE/kdepim/akonadi Volker Krause ... Thu, 28 Jun 2007 15:30:33 ...
KDE/kdepim/akonadi/clients/plasma/... Thomas Moeni... Fri, 29 Jun 2007 06:57:04 +...
KDE/kdepim/akonadi/libakonadi/tests Volker Krause ... Tue, 26 Jun 2007 21:12:54 ...
KDE/kdepim/akonadi/clients/plasma/... Thomas Moeni... Fri, 29 Jun 2007 07:22:29 +...
KDE/kdepim/akonadi/clients/plasma/... Thomas Moeni... Fri, 29 Jun 2007 08:32:23 +...
KDE/kdepim/akonadi Volker Krause ... Wed, 27 Jun 2007 20:01:41...

Extending Akonadi

Topics

- Serializer plugins
- Specialized item models
- Resource agents

Extending Akonadi

Akonadi::ItemSerializer

- Converts between item payload objects and a multipart textual or binary representation
- Mandatory for every payload type
- Derive from Akonadi::ItemSerializer and implement two virtual methods:
  - convert binary representation to payload object
  - convert payload object to binary representation

Extending Akonadi

Example: Deserializing

```
void MailSerializer::deserialize( Akonadi::Item& item,
                                  const QString& part,
                                  QIODevice& data ) {
    KMime::Message::Ptr msg;
    if ( part == Item::PartBody ) {
        msg->setContent( data.readAll() );
        msg->parse();
    } else if ( part == Item::PartEnvelope ) {
        // ...
    }
    item.setPayload<KMime::Message::Ptr>( msg );
}
```

Extending Akonadi

Example: Serializing

```
void MailSerializer::serialize( Akonadi::Item& item,
                                const QString& part,
                                QIODevice& data ) {
    KMime::Message::Ptr msg = item->payload();
    if ( part == Item::PartBody ) {
        msg->assemble();
        data.write( msg->encodedContent() );
    } else if ( part == Item::PartEnvelope ) {
        // ...
    }
}
```

Extending Akonadi

Akonadi::ItemModel

- Generic item model provides many features but usually needs type specific extensions:
  - showing the actual data
  - relations between items (e.g. mail threads)
  - specialized sorting
- Can be implemented as:
  - Akonadi::ItemModel sub-class
  - Proxy model

Extending Akonadi

Example: Specialized Item Model

```
QVariant MessageModel::data( const QModelIndex & index, int role )
  const {
  Akonadi::Item item = itemForIndex( index );
  KMime::Message::Ptr msg = item.payload();
  if ( role == Qt::DisplayRole ) {
    switch ( index.column() ) {
      case Subject:
        return msg->subject()->asUnicodeString();
      // ...
    }
  }
}
```

Extending Akonadi

Resource Agents

- Connect Akonadi to a data source such as:
  - local files (iCal, vCard, maildir, etc.)
  - mail- and groupware servers (Kolab, IMAP, etc.)
  - web services
- Separate process, takes care of:
  - synchronize collection and items with the data source
  - convert between data formats
- Derived from Akonadi::ResourceBase, which provides many convenience methods

Extending Akonadi

Minimal Collection Listing

```
void MyResourceAgent::retrieveCollections()
{
     Akonadi::Collection::List list;
     // retrieve folders from backend and add them to list
     ...
     collectionsRetrieved( list );
}
```

Extending Akonadi

Improved Collection Listing

```
void MyResourceAgent::retrieveCollections()
{
     Akonadi::Collection::List changed, deleted;
     // retrieve changes to folders in the backend
     // and add them to changed and deleted accordingly
     ...
     collectionsRetrievedIncremental( changed, deleted );
}
```

Extending Akonadi

Minimal Collection Content Listing

```
void MyResourceAgent::retrieveItems( const Akonadi::Collection &col
                                     const QStringList &parts )
{
    Akonadi::Item::List list;
    // retrieve information about all items in folder col
    // in the backend, depending on which parts are requested
    ...
    itemsRetrieved( list );
}
```

Extending Akonadi

Improved Collection Content Listing

```
void MyResourceAgent::retrieveItems( const Akonadi::Collection &col
                                     const QStringList &parts )
{
    Akonadi::Item::List changed, deleted;
    // retrieve information about changes to items in folder col
    // in the backend
    ...
    itemsRetrievedIncremental( changed, deleted );
}
```

Extending Akonadi

Retrieving a single Item

```
bool MyResourceAgent::retrieveItem( const Akonadi::Item &item,
                                    const QStringList &parts )
{
    // get data from the backend
    // item.remoteId() helps you to find it
    MyData data = ...
    Akonadi::Item i( item );
    i.setMimeType( "application/x-my-data" );
    i.setPayload( data );
    itemRetrieved( i );
    return true; // no error
}
```

Extending Akonadi

Reacting to local changes

```
void itemAdded( const Akonadi::Item &item,
                const Akonadi::Collection &collection );
void itemChanged( const Akonadi::Item &item,
                  const QStringList &partIdentifiers  );
void itemRemoved( const Akonadi::DataReference &ref );
void collectionAdded( const Collection &collection );
void collectionChanged( const Collection &collection );
void collectionRemoved( int id, const QString &remoteId );
```

Contribute

What can I do?

- Add resource agents for new backends
- Extend Akonadi to support new types
- Using Akonadi in applications
- Additional Client library implementations
- Language bindings
- Contributing to the Akonadi server

Contribute

Ideas

- filter agents
- integrating live search as virtual collections
- support for searches on the backend (LDAP, IMAP)
- (local) subscription / subscription profiles
- OpenSync agent
- cool plasmoids and workflow-centric apps
- ...

Contribute

Contact

- IRC: #kontact on irc.freenode.org
- Mailinglist: kde-pim@kde.org
- http://pim.kde.org/akonadi