KDE4 Workshop - CLUG

Tobias Koenig

May 9, 2008

- Plasma - Die Desktop Shell
  - Aufbau
  - Ein einfaches Plasmoid
  - Ein dynamisches Plasmoid
  - Ein nützliches Plasmoid
- KDE 4.1
  - GUI Neuerungen
  - Technische Neuerungen
- Akonadi - Der PIM Layer
  - Aufbau
  - Features
  - Zugriff
- Fragen und Antworten

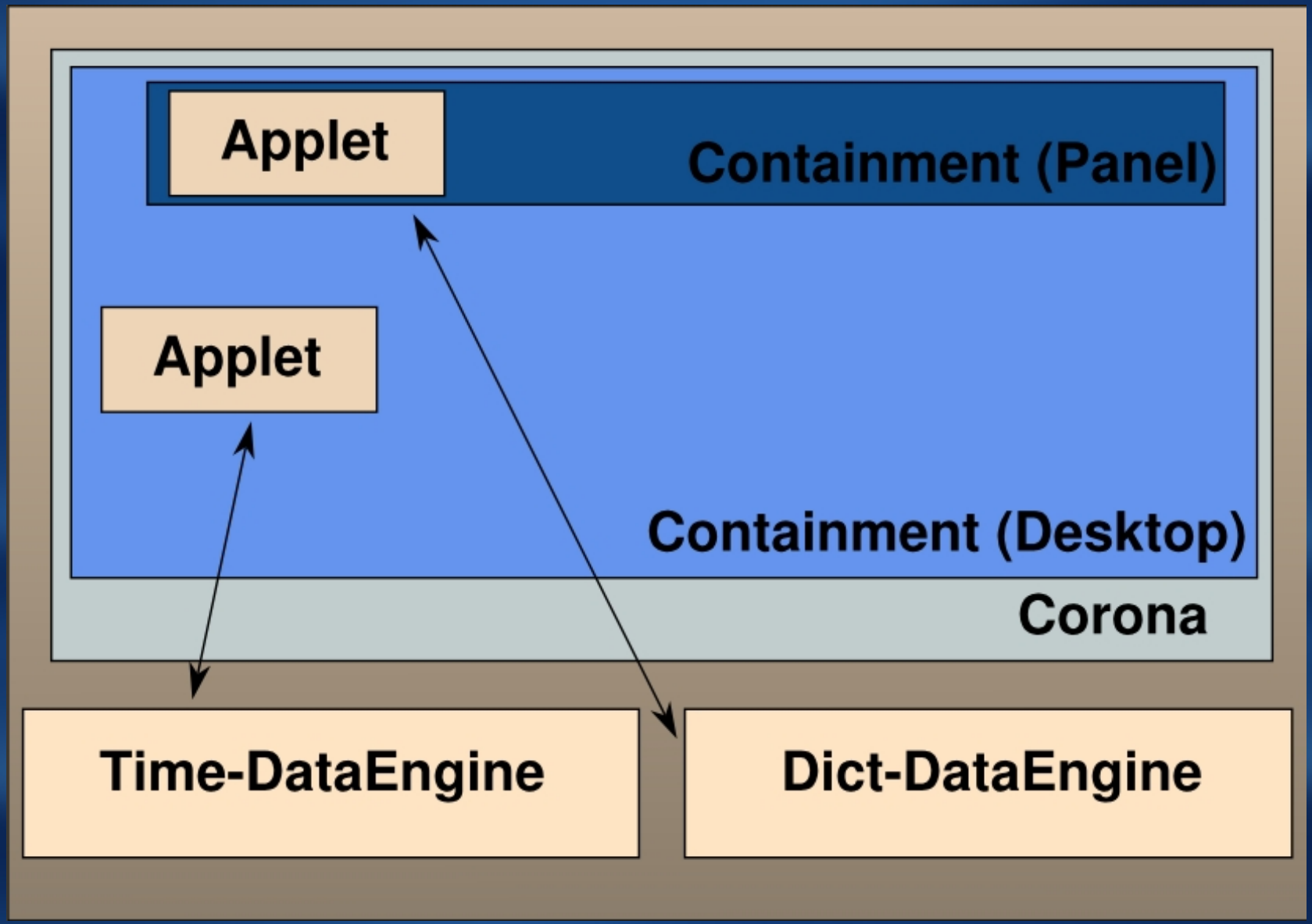
Plasma - Die Desktop Shell

Plasma - Die Desktop Shell

- Desktop und Panels zu einem Programm verschmolzen
- Graphische Effekte mittels QGraphicsView
- Funktionalität durch Plasmoide erweiterbar
- Trennung zwischen Daten und Darstellung (MVC)

Plasma - Aufbau

Plasma - Aufbau



Plasma - Aufbau

Corona

- Erbt von QGraphicsScene
- Steuert Veränderbarkeit der Applets/Containments
- Enthält alle Applets/Containments

Containment

- Dient Gruppierung von Applets
- Typen
  - Desktop
  - Panel
  - Custom
- Bietet ToolBox an

Plasma - Aufbau

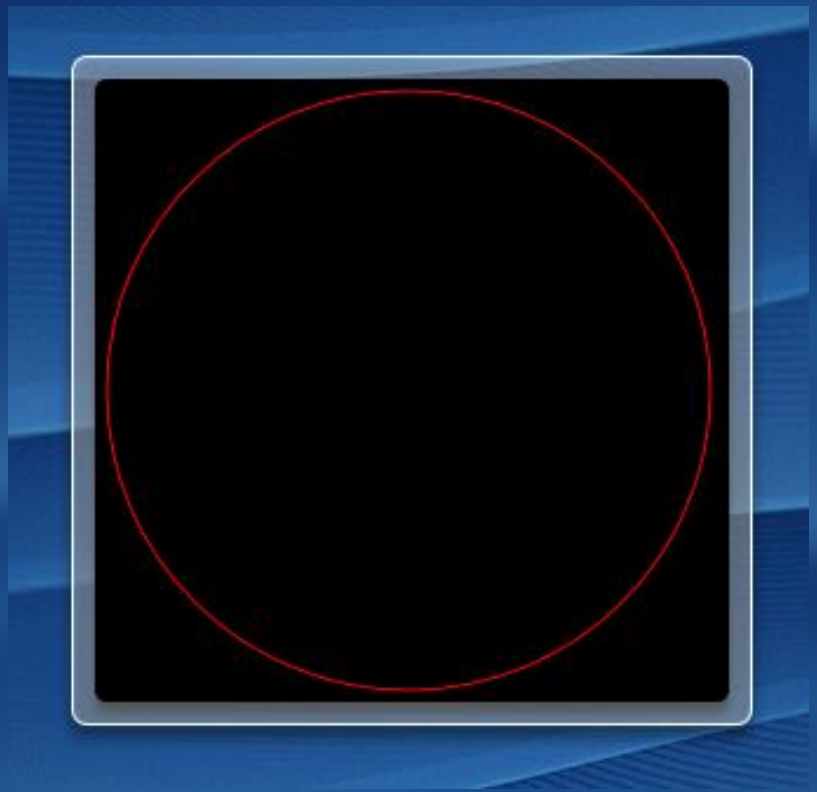
Applet

- Basis für alle Applets und Containments
- Erbt von QGraphicsWidget
  $\rightarrow$ Kann beliebige QGraphicsLayouts/Widgets beinhalten
- Alle graphischen Transformationen können angewendet werden
- Bietet Funktionalität zur Steuerung/Verwaltung von:
  - Layout
  - SVG-Themes
  - Packages

Plasmoid - "Workshop"

- Ruby-Bindings bis SVN Rev. #804508 unter trunk/KDE/kdebindings/
- Neue Ruby-Bindings erst ab KDE 4.2
- Beispiel-Code unter http://wgess16.dyndns.org/~tobias/static/plasma_ruby.tar.bz2

Plasmoid - Das Ziel



Plasmoid - Das Drumherum

Verzeichnisstruktur

- plasma_circle
  - CMakeLists.txt
  - circle.rb
  - plasma-applet-circle.desktop

Plasmoid - CMakeLists.txt

```
PROJECT( plasma_circle )

FIND_PACKAGE( KDE4 REQUIRED )
FIND_PACKAGE( Plasma REQUIRED )
FIND_PACKAGE( RUBY REQUIRED )

INCLUDE_DIRECTORIES( ${RUBY_INCLUDE_PATH} )

INSTALL( FILES circle.rb
         DESTINATION ${DATA_INSTALL_DIR}/plasma-ruby-circle )
INSTALL( FILES plasma-applet-circle.desktop
         DESTINATION ${SERVICES_INSTALL_DIR} )
```

Plasmoid - plasma-applet-circle.desktop

```
[Desktop Entry]
Encoding = UTF-8
Name = Circle Plasmoid
Comment = Shows an animated circle
Icon = face-smile-big
Type = Service
X-KDE-ServiceTypes = Plasma/Applet

X-KDE-Library = krubypluginfactory
X-KDE-PluginKeyword = plasma-ruby-circle/circle.rb
X-KDE-PluginInfo-Author = Tobias Koenig
X-KDE-PluginInfo-Email = tokoe@kde.org
X-KDE-PluginInfo-Name = circle
X-KDE-PluginInfo-Version = 0.1
X-KDE-PluginInfo-Website = http://plasma.kde.org/
X-KDE-PluginInfo-Category = Graphics
X-KDE-PluginInfo-License = GPL
X-KDE-PluginInfo-EnabledByDefault = true
```

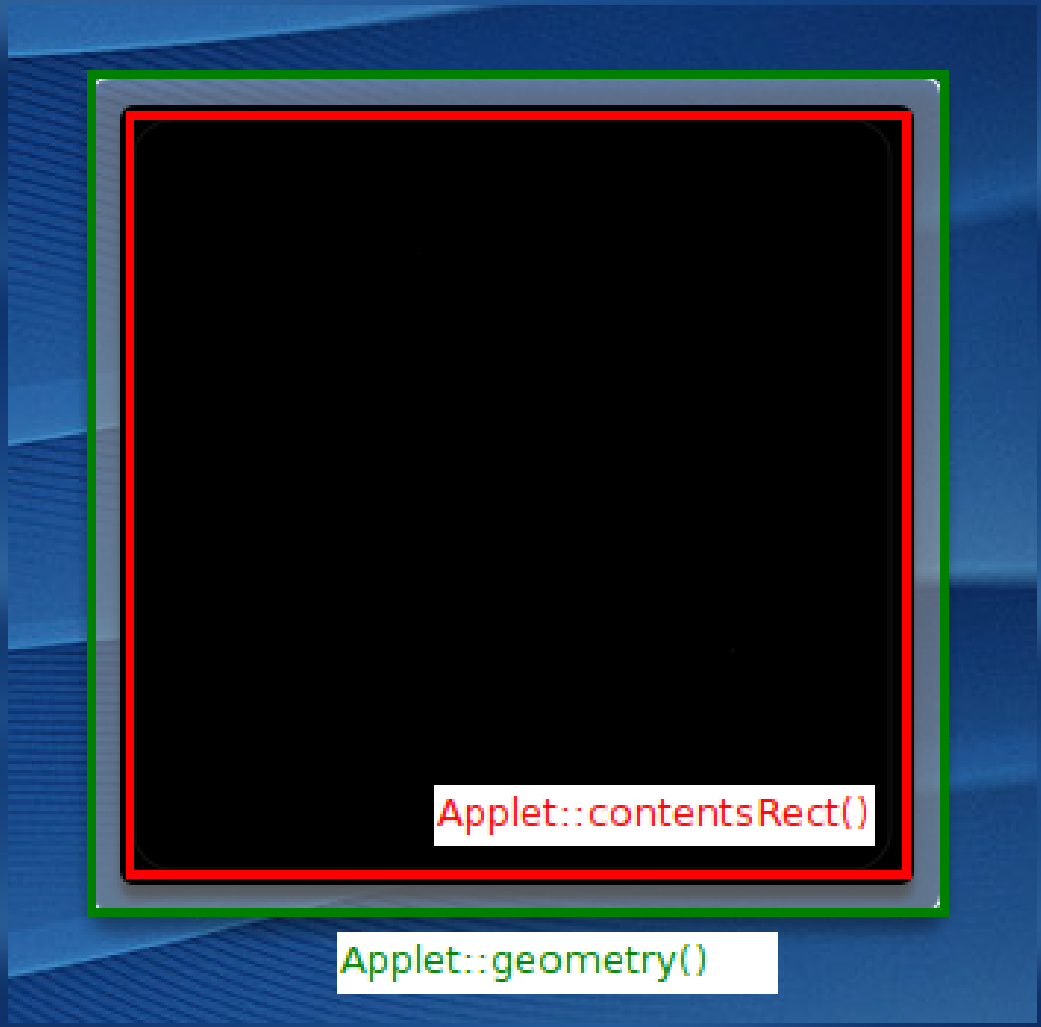
Plasmoid - circle.rb

```
require 'plasma_applet'

module PlasmaRubyCircle

class Circle < Plasma::Applet
  def initialize( parent, args )
    super
  end

  def init
    resize( 300, 300 )
  end

  def paintInterface( painter, option, contentsRect )
    painter.setRenderHint( Qt::Painter::Antialiasing )
    painter.pen = Qt::Color.new( "red" )
    painter.drawEllipse( contentsRect );
  end
end

end
```

Plasmoid - Installieren

- cd plasma_circle
- cmake -DCMAKE_INSTALL_PREFIX=/opt/kde4 .
- make
- make install

Plasmoid - Geometrie



Plasmoid - Das Ziel

Plasmoid - circle.rb (I)

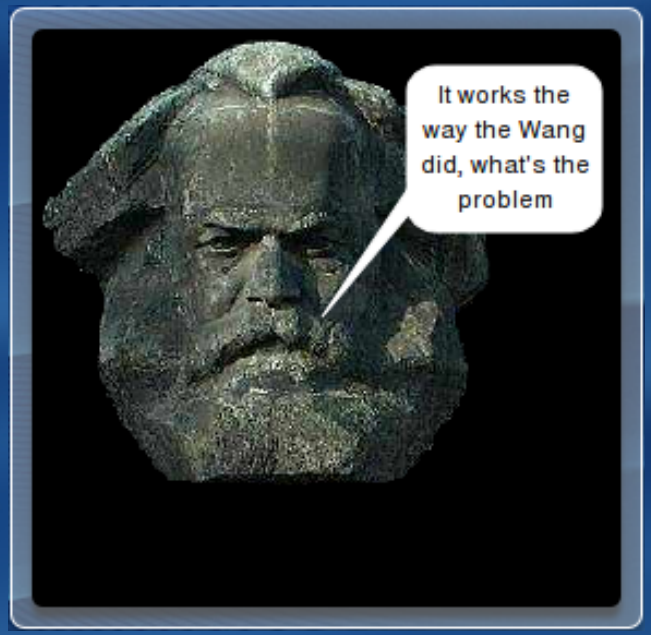
```
require 'plasma_applet'

module PlasmaRubyCircle

class Circle < Plasma::Applet
  slots 'dataUpdated(QString, Plasma::DataEngine::Data)'

  def initialize( parent, args )
    super
  end

  def init
    resize( 300, 300 )

    @radius = 0

    timeEngine = dataEngine( "time" )
    timeEngine.connectSource( "Local", self, 1000 )
  end

  def dataUpdated( source, data )
    @time = data[ "Time" ].toTime()
    @radius = contentsRect().width()/2 * (@time.second/60.0)
    update()
  end
```

Plasmoid - circle.rb (II)

```
    def paintInterface( painter, option, contentsRect )
      painter.setRenderHint( Qt::Painter::Antialiasing )
      painter.pen = Qt::Color.new( "red" )

      midX = contentsRect.left() + contentsRect.width() / 2
      midY = contentsRect.top() + contentsRect.height() / 2

      painter.drawEllipse( Qt::PointF.new( midX, midY ),
                           @radius, @radius )
      painter.drawText( midX, midY, @time.toString() )
    end
  end

  end
```

Ein nützliches Plasmoid

Ein nützliches Plasmoid



Plasmoid - Verzeichnisstruktur

- plasma_nischel
  - CMakeLists.txt
  - applet
    - CMakeLists.txt
    - background.png
    - nischel.rb
    - plasma-applet-nischel.desktop
  - dataengine
    - CMakeLists.txt
    - bofh.cpp
    - bofh.h
    - nischel.cpp
    - nischel.h
    - plasma-dataengine-nischel.desktop

Applet - nischel.rb (I)

```
require 'plasma_applet'

module PlasmaRubyNischel

class Nischel < Plasma::Applet
  slots 'dataUpdated( QString, Plasma::DataEngine::Data )'

  def initialize( parent, args )
    super( parent, args )
  end

  def init
    resize( 300, 300 )
    nischelEngine = dataEngine( "nischel" )
    nischelEngine.connectSource( "currentExcuse", self, 0 )
    @currentExcuse = nischelEngine.query( "currentExcuse" )
                                  ["currentExcuse"].toString()
  end

  def dataUpdated( identifier, data )
    if identifier == "currentExcuse"
      @currentExcuse = data["currentExcuse"].toString()
    end
    update()
  end
```

Applet - nischel.rb (II)

```
def paintInterface( painter, option, contentsRect )
  painter.setRenderHint( Qt::Painter::Antialiasing )
  painter.setRenderHint( Qt::Painter::TextAntialiasing )

  img = Qt::Image.new( KDE::StandardDirs.locate( "data",
                              "plasma-ruby-nischel/background.png" ) )

  x = contentsRect.left()
  y = contentsRect.top()
  width = contentsRect.width()
  height = contentsRect.height()

  imageRect = Qt::Rect.new( x, y, width*0.8, height*0.8 )

  painter.drawImage( imageRect, img )

  painter.brush = Qt::Brush.new( Qt::white, Qt::SolidPattern )
  painter.pen = Qt::Color.new( "white" )

  poly = Qt::Polygon.new( 3 )
  poly.setPoint( 0, Qt::Point.new( x+width*0.5, y+height*0.5 ) )
  poly.setPoint( 1, Qt::Point.new( x+width*0.7, y+height*0.2 ) )
  poly.setPoint( 2, Qt::Point.new( x+width*0.75, y+height*0.2 ) )
```

Applet - nischel.rb (III)

```
      rect = Qt::Rect.new( x+width*0.65, y+height*0.05,
                           width*0.35, height*0.3 )
      painter.drawRoundRect( rect, 30, 30 )
      painter.drawPolygon( poly )

      fontSize = height*10/300
      if fontSize < 1
        fontSize = 1
      end

      font = painter.font
      font.pointSize = fontSize
      painter.font = font

      painter.setPen( Qt::Color.new( "black" ) )

      painter.drawText( rect, Qt::AlignCenter | Qt::AlignVCenter |
                        Qt::TextWordWrap, @currentExcuse )
    end
  end

  end
```

DataEngine - CMakeLists.txt

```
SET( nischel_engine_SRCS
     bofh.cpp
     nischel.cpp
)

KDE4_ADD_PLUGIN( plasma_engine_nischel ${nischel_engine_SRCS} )
TARGET_LINK_LIBRARIES( plasma_engine_nischel ${KDE4_KDEUI_LIBS}
                                             ${PLASMA_LIBS} )

INSTALL( TARGETS plasma_engine_nischel
         DESTINATION ${PLUGIN_INSTALL_DIR} )
INSTALL( FILES plasma-dataengine-nischel.desktop
         DESTINATION ${SERVICES_INSTALL_DIR} )
```

DataEngine - plasma-dataengine-nischel.desktop

```
[Desktop Entry]
Encoding=UTF-8
Name=Nischel Data Engine
Comment=Data Engine for Nischel Applet
Type=Service
Icon=face-smile-big

X-KDE-ServiceTypes=Plasma/DataEngine
X-KDE-Library=plasma_engine_nischel
X-Plasma-EngineName=nischel
```

DataEngine - nischel.h

```
#ifndef NISCHEL_DATAENGINE_H
#define NISCHEL_DATAENGINE_H

#include <plasma/dataengine.h>

class NischelEngine : public Plasma::DataEngine
{
  Q_OBJECT

  public:
    NischelEngine( QObject* parent, const QVariantList& args );
    ~NischelEngine();
  protected:
    void init();
    bool sourceRequestEvent( const QString &identifier );
  protected Q_SLOTS:
    bool updateSourceEvent( const QString &identifier );
  private:
    QStringList mExcuses;
};

K_EXPORT_PLASMA_DATAENGINE( nischel, NischelEngine )

#endif
```

DataEngine - nischel.cpp (I)

```
#include <QtCore/QDateTime>

#include "bofh.h"
#include "nischel.h"

NischelEngine::NischelEngine( QObject* parent,
                              const QVariantList &args )
  : Plasma::DataEngine( parent, args )
{
  setMinimumPollingInterval( 0 );
  setPollingInterval( 30000 );
}

NischelEngine::~NischelEngine()
{
}

void NischelEngine::init()
{
  // Initialize random number generator
  qsrand( QDateTime::currentDateTime().toTime_t() );

  Bofh bofh( "/usr/share/games/fortunes/bofh-excuses" );
  mExcuses = bofh.excuses();
}
```

DataEngine - nischel.cpp (II)

```
bool NischelEngine::updateSourceEvent( const QString &id )
{
  if ( id == "currentExcuse" ) {
    setData( id, mExcuses.at( qrand() % mExcuses.count() ) );
    return true;
  } else {
    return false;
  }
}

bool NischelEngine::sourceRequestEvent( const QString &id )
{
  setData( id, DataEngine::Data() );

  return updateSourceEvent( id );
}

#include "nischel.moc"
```

KDE 4.1

Plasma - Der Desktop

- Device Notifier Applet
- Unterstützung für MacOS X und SuperKaramba Applets
- Installation von Plasma-Themes via GetHotNewStuff
- Mehrere Panels
- Classic-Style Menü

Konqueror - Der Datei/Webbrowser

- ACID3 Test mit 73/100 (Konqueror 3.5.9 42/100)
- Theming für Fehlerseiten
- History für zuletzt besuchte und geschlossene Seiten

Okular - Der Dokumentenbetracher

- Erweiterte Unterstützung von Anmerkungen
- Zugriff auf Datei-Anhänge
- Neuerungen im Präsentationsmodus
  - Zuweisung des Bildschirms
  - (BlackScreen-Modus)
- Unterstützung des EPub-Formats

Allgemeines

- Ozone Widget-Style
- Drucken von 'Cheat-Sheets' für Tastenkürzel
- Modul zur Konfiguration des Autostarts

Neue Programme

- Dragon Player (Video-Player)
- Okteta (Hex-Editor)
- Skanlite (Scan-Programm)
- KDiamond (Spiel)

Technische Neuerungen

- WMI Interface für Solid
- Modularer Bindings Generator
- PcmIO-Backend für Phonon
- (VLC/MPlayer-Backend für Phonon)
- Akonadi

Akonadi - Der PIM Layer

Akonadi - Der PIM Layer

- PIM in KDE3 für kleine Datenmengen konzipiert
- Datenhaltung realisiert durch Bibliotheken
- Probleme:
  - Jedes Programm hält alle Daten im Speicher
  - Keine Änderungsbenachrichtigungen
  - Schlechte Integration
- Akonadi als Lösung

Akonadi - Der PIM Layer

Was ist Akonadi?

- Eine asynchrone API zum einheitlichen Zugriff auf PIM Daten
- Ein Cache für PIM Daten
- Eventueller Ersatz für EDS
- Ein cooles Projekt ;)

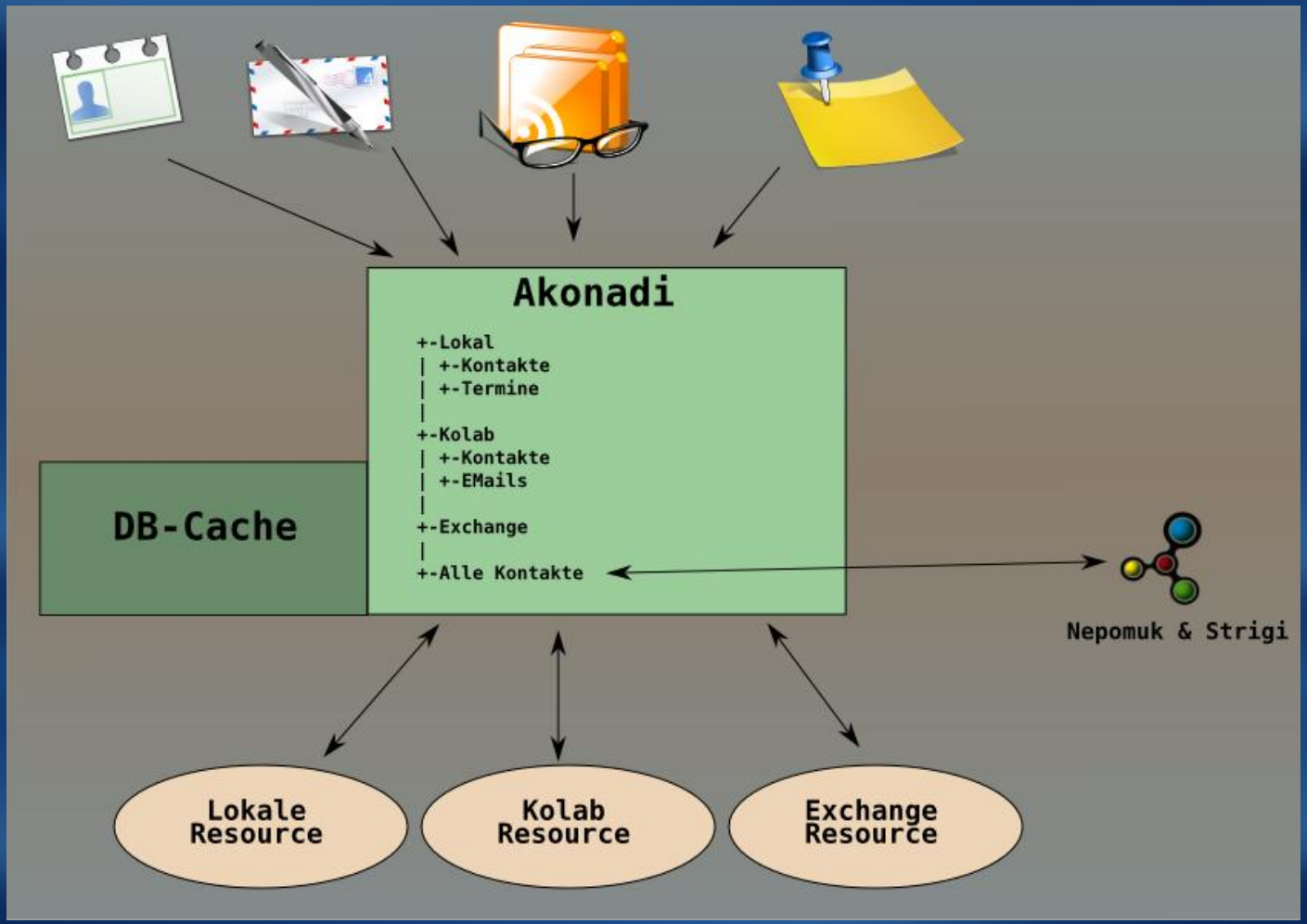
Was ist Akonadi **nicht**?

- Ein Groupware Server

Akonadi - Aufbau

- Nur abhängig von Qt 4.4
- Besteht aus mehreren Prozessen
  Ready for SMP [tm] ;)
  - Control: Startet/Stoppt alle Akonadi-Prozesse
  - Storage: Cached Daten mittels MySQL-Datenbank
  - Agents: Erledigen Routineaufgaben
  - Ressourcen: Führen Datentransfer durch
- Läuft einmal pro KDE Session

Akonadi - Aufbau



Akonadi - Features

- Zentraler Zugriff auf alle PIM Daten
- Globale Änderungsbenachrichtigungen
- Asynchroner/Synchroner Zugriff auf Daten
- Einheitliche, desktopübergreifende Schnittstelle

Akonadi - Features

- Bietet hierarchische Struktur aller PIM Objekte
  - Collection: Verzeichnis/Ordner/Zweig
  - Item: Datei/Blatt
- Jede Ressource bietet eine Basis-Collection
- Collection kann Sub-Collections beinhalten
- Collections können typisiert werden
- Integration mit Nepomuk

Akonadi - Zugriff

- Zugriff über IMAP-ähnliches Protokoll bzw. DBus
- Client-Bibliothek für KDE verfügbar
- GLib-Bindings erwünscht, Interesse? ;)
- Kommunikation nach Anfrage-Antwort Prinzip

Akonadi - Zugriff

Beispiel:

```
#include <akonadi/collectionfetchjob.h>

CollectionFetchJob *job =
    new CollectionFetchJob( Collection::root(),
                            CollectionFetchJob::Recursive );

job->exec();
foreach( Collection &col, job->collections() ) {
  qDebug() << "Collection:" << col.name();
}
```

Akonadi - Zugriff

Beispiel:

```
#include <akonadi/itemfetchjob.h>
#include <kabc/addressee.h>

ItemFetchJob *job = new ItemFetchJob( collection );
job->fetchScope().fetchFullPayload();
job->exec();
foreach( Item &item, job->items() ) {
  qDebug() << "Item:" << item.mimeType();
  if ( item.mimeType() == "text/directory" ) {
    KABC::Addressee addr = item.payload<KABC::Addressee>();

    qDebug() << "Name:" << addr.givenName()
                        << addr.familyName();
  }
}
```

Akonadi - Zugriff

Beispiel:

```
#include <akonadi/itemcreatejob.h>
#include <kabc/addressee.h>

KABC::Addressee addr = ...

Item item;
item.setMimeType( "text/directory" );
item.setPayload<KABC::Addressee>( addr );

ItemCreateJob *job = new ItemCreateJob( item, collection );
job->exec();
```

Akonadi - Zusammenfassung

- Service für desktopweiten Zugriff auf PIM Daten
- Integriert PIM Daten mit semantischer Suche
- Starke Verwendung von Programmen ab KDE 4.2 geplant

6. Linux-Info-Tag Dresden

- Wann: Am Samstag 8. November, 2008
- Wo: Fakultätsgebäude Informatik, TU Dresden
- CfP: bereits freigegeben ;)
- http://www.linux-info-tag.de

Fragen und Antworten

Danke!