KDE 4: Поглед отвътре

Димитър Василев,
20 декември 2008

Цели на презентацията

Как се разработва свободен софтуер?

Влияние на KDE върху други проекти

- Qt
- Subversion
- Cmake
- Webkit
- ...

Какво ново в Qt 4 (1)

- Многоплотформеност – Linux, Mac OS, Windows, Windows CE, Embedded Linux, а скоро и S60
- Tulip
- The Interview framework
- Arthur
- Scribe
- Подръжка на векторно чертаеене и анимация

Какво ново в Qt4 (2)

- PDF backend при печата
- Unit test framework
- Graphics View framework
- Qt Style Sheets
- Поддръжка на D-Bus IPC и RPC
- Undo framework
- QtScript
- Multimedia API

Какво ново Qt 4 (3)

- Добре развит Xml модул
- Webkit интеграция
- ...

Излизане на версии

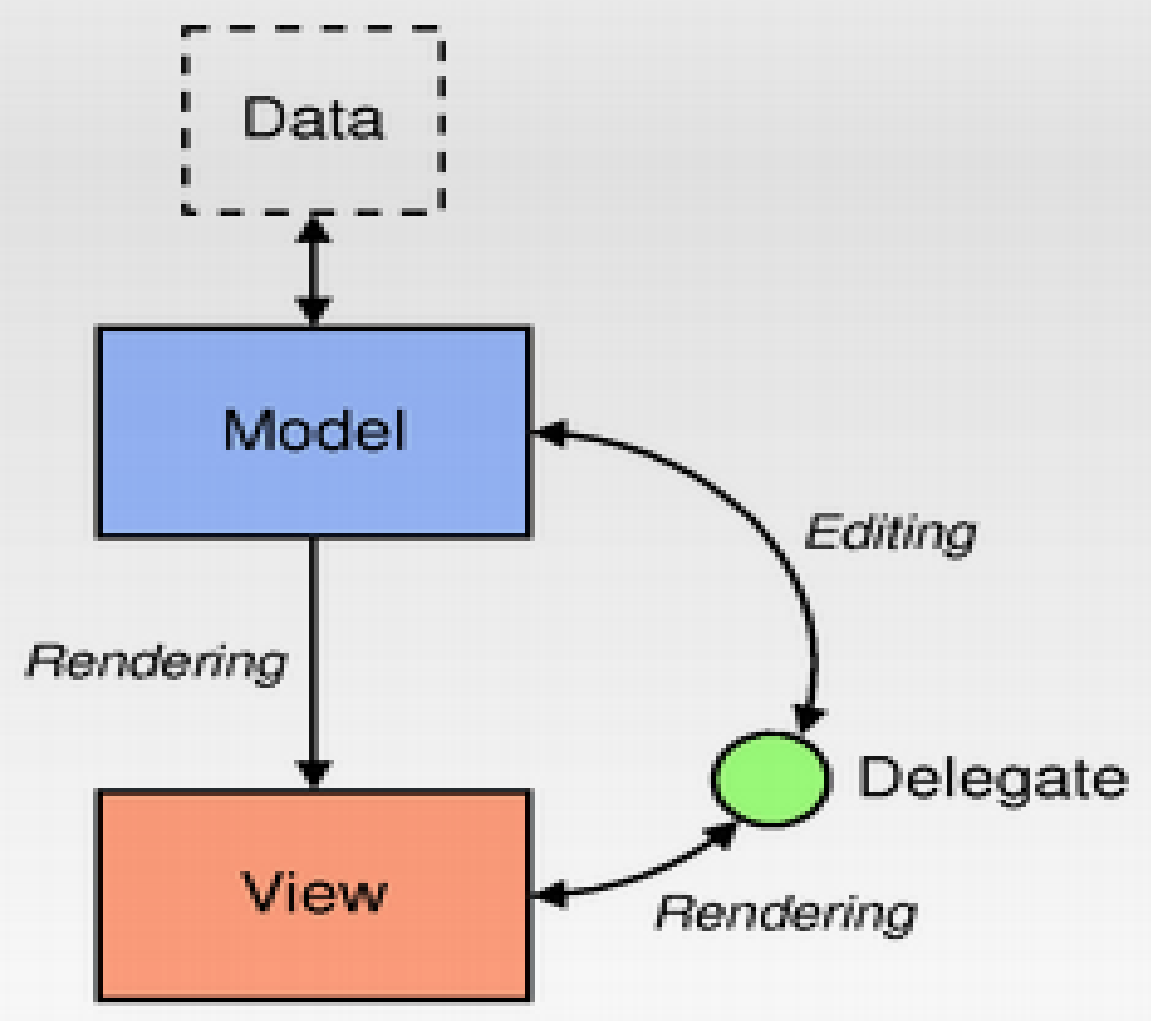
- Qt 4.0 – юни 2005
- Qt 4.1– декември 2005
- Qt 4.2 – октомври 2006
- Qt 4.3 – юни 2007, необходимо за *KDE 4.0*
- Qt 4.4 – май 2008, необходимо за *KDE 4.1*

Какво да очакваме в Qt 4.5

- Много подобрения и разширения във вече наличните функционалности
- По-добра производителност
- Поддръжка на ODF
- QtScript debugger

The Interview Framework

Архитектура



The Interview Framework

Съвместимост с Qt3

- QListBox ==> QListWidget
- QTable ==> QTableWidget
- QListView ==> QTreeWidget

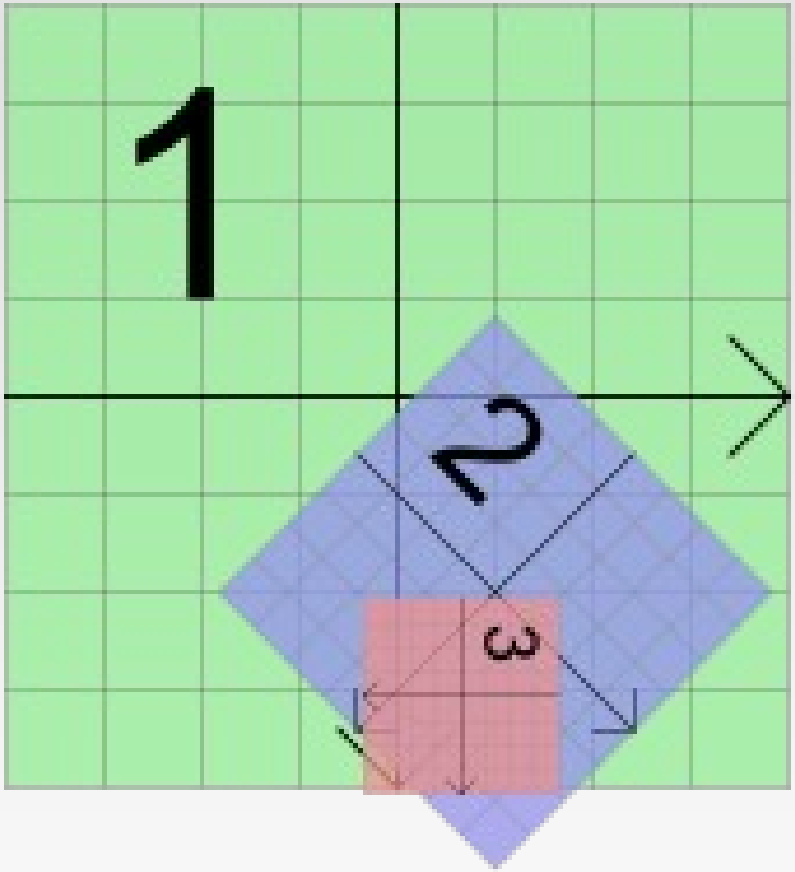
Graphics view framework

Архитектура

- Сцена
- Изглед
- Елементи

Graphics view framework

Координати



Graphics view framework

Основни свойства

- Мащабиране и въртене
- Печат
- Drag&Drop
- Анимация
- OpenGL рендериране
- Групиране на елементите
- Embedded Widget Support

Arthur Paint System

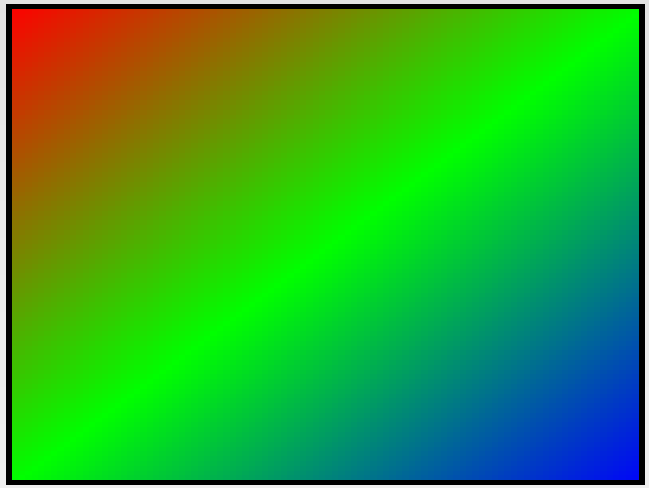
Подходи за „рисуване“:

- Pixel-based engine за Windows и при рисуване на QImages на всички платформи
- OpenGL на всички платформи
- PostScript на Linux, Unix и Mac OS X
- QuickDraw и CoreGraphics на Mac OS X
- X11 и X Render Extension на Linux и Unix системи

Arthur Paint System

диагонален градиент

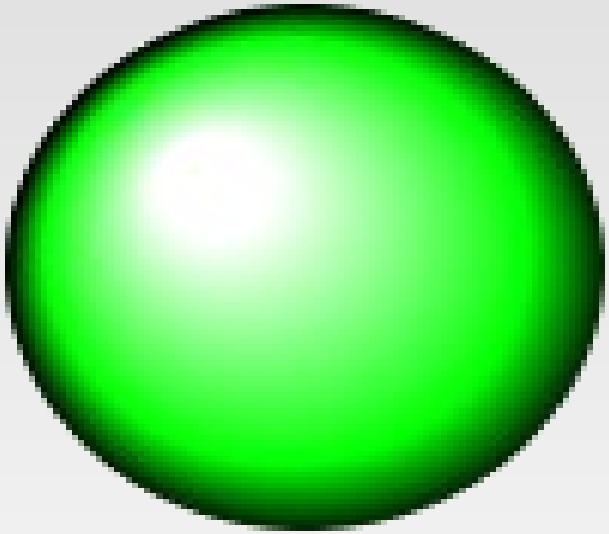
```
QLinearGradient gradient(0, 0, 100, 100);
gradient.setColorAt(0, Qt::red);
gradient.setColorAt(0.5, Qt::green);
gradient.setColorAt(1, Qt::blue);
painter.setBrush(gradient);
painter.drawRect(0, 0, 100, 100);
```



Arthur Paint System

радиален градиент

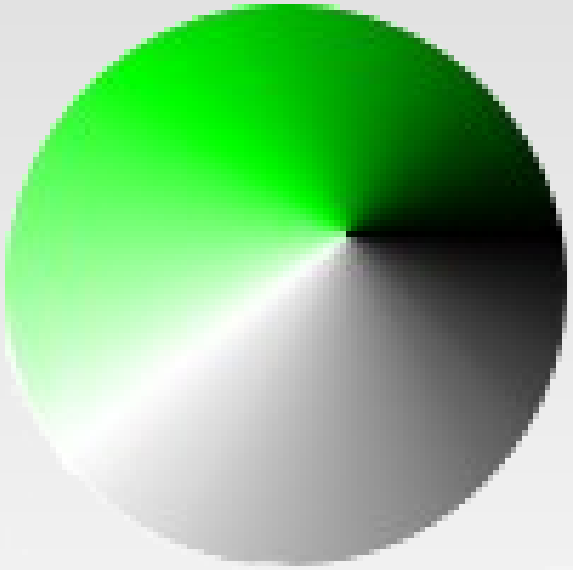
```
QRadialGradient gradient(50, 50, 50, 30, 30);
gradient.setColorAt(0.2, Qt::white);
gradient.setColorAt(0.8, Qt::green);
gradient.setColorAt(1, Qt::black);
painter.setBrush(gradient);
painter.drawEllipse(0, 0, 100, 100);
```



Arthur Paint System

конусен градиент

```
QConicalGradient gradient(60, 40, 0);
gradient.setColorAt(0, Qt::black);
gradient.setColorAt(0.4, Qt::green);
gradient.setColorAt(0.6, Qt::white);
gradient.setColorAt(1, Qt::black);
painter.setBrush(gradient);
painter.drawEllipse(0, 0, 100, 100);
```

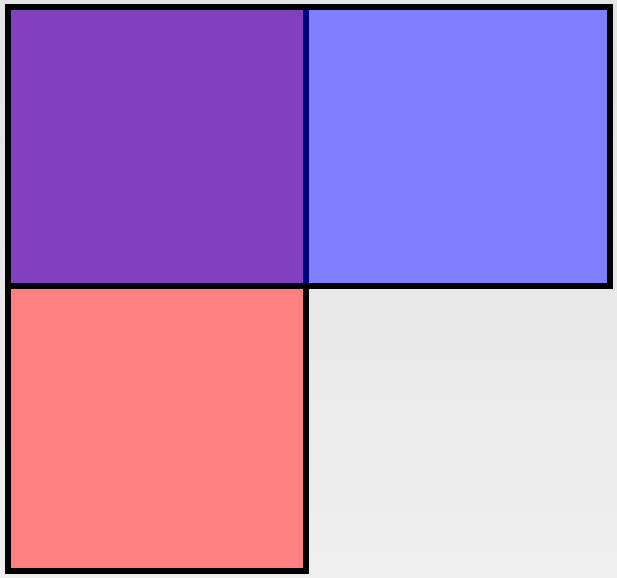


Arthur Paint System

Alpha-Blended рисунка

```
// Specfiy semi-transparent red
 painter.setBrush(QColor(255, 0, 0, 127));
 painter.drawRect(0, 0, width()/2, height());

 // Specify semi-transparend blue
 painter.setBrush(QColor(0, 0, 255, 127));
 painter.drawRect(0, 0, width(), height()/2);
```



От Qt3 към Qt4

Основни трудности

- Чисто синтактични, дължащи се на преименуване на класове, различни параметри в конструктури и функции
- Разширяването на обхвата на благинките идващи от Qt4 предполага, че част от нещата, които сами сме си направили са вече излишни
- Промяна в концепцията как стават нещата налага по-сериозни промени в програмите ни

Предизвикателства пред KDE4

- Многоплатформеност
- Интегрираност
- Разчупване на йерархичния модел

Основни компоненти

- Desktop
  - Plasma
  - Sonnet
  - KParts
- Hardware
  - Solid
  - Phonon
- Communication
  - Decibel
  - Akonadi

Plasma

Основни компоненти

- Corona
- Applet
- Containment
- DataEngine
- Services
- ...

Plasma

Набор от plugin-и

- Plasmoids
- Containments
- Runners
- Wallpapers
- DataEngines
- ....

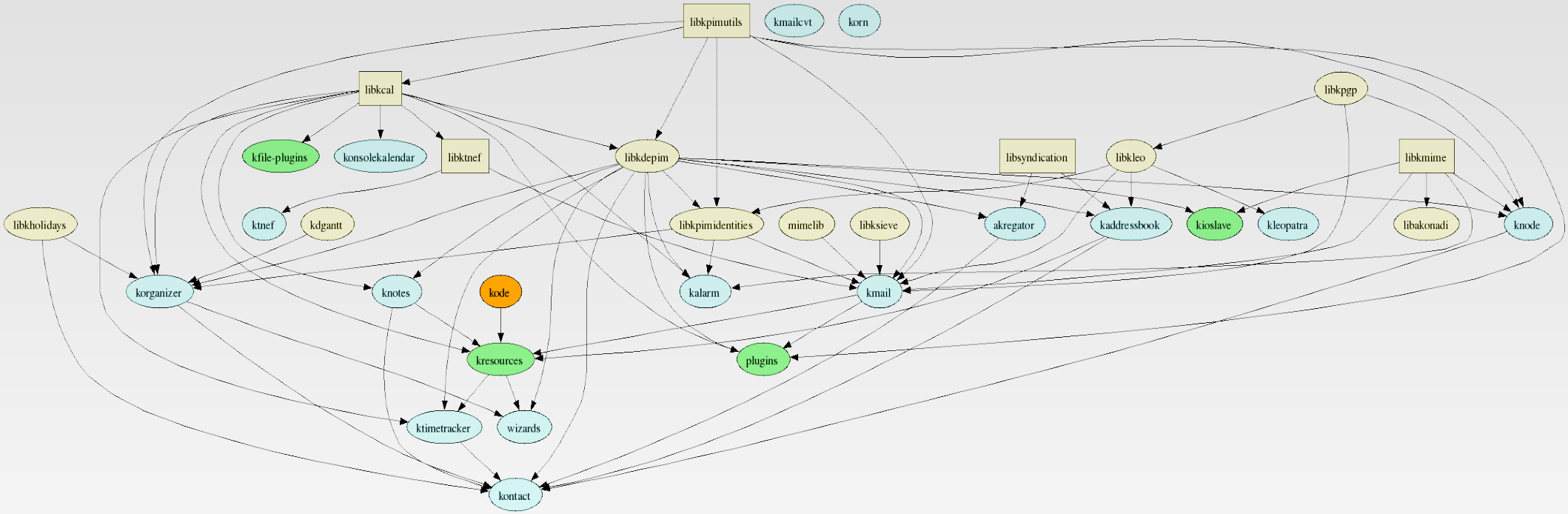
Akonadi

технологията до сега KResource

- Проектирана преди 10 години за малък обем данни
- Синхронен достъп
- Без споделяне на кода
- Няма нотификация при промени
- Труден за разширение
- Няма споделяне на данните

Akonadi

технологията до сега



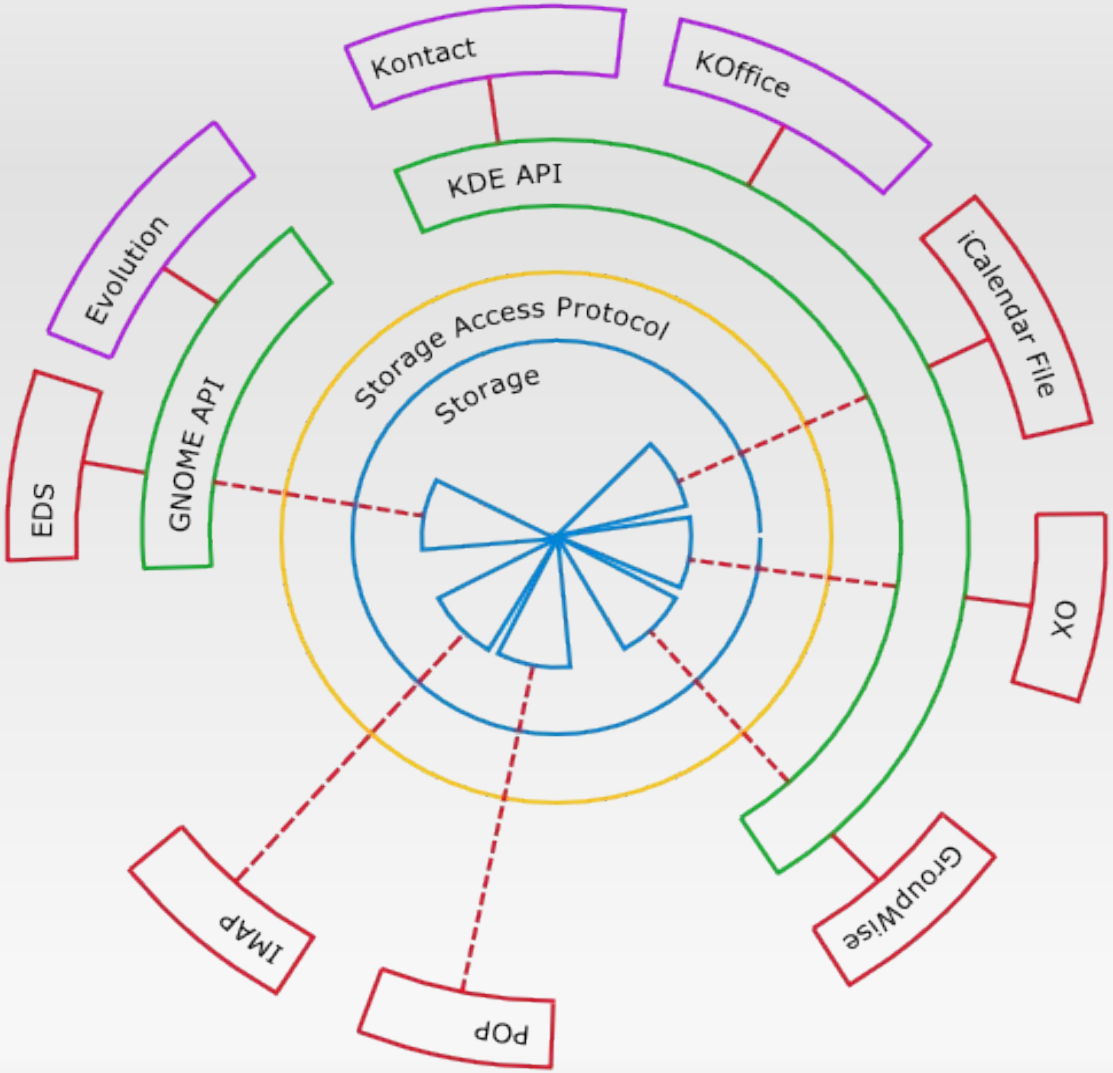
Akonadi

Цели

- Да поправи всички недостатъци
- Да предостави унифицирано API за достъп до PIM данните, независимо от източника на данни
- Да предостави гъвкави правила при кеширането на отдалечени данни
- Бързи виртуални папки, за представяне на резултати от търсене
- Използваемо навсякъде

Akonadi

архитектура



Akonadi

Сървър

- Независим от типа на данните
- Кеширане на отдалечените данни
- Нотификация при промяна
- Откриване на конфликти
- Базиран на MySql
- Зависим само от Qt

Akonadi
Протокол

- Достъп до данните
  - Базиран на IMAP
  - Оптимизиран за обемни трансфери
  - Използване на стандартни формати при кодиране на информацията
- Контролни методи и нотификации
  - D-Bus
  - Оптимизирани за лесна употреба
- Независим от платформата и библиотеките

Akonadi

Основни обекти

- Подобна на файлова система структура
  - Колекции
  - Елементи
- Елементите се състоят от
  - Данни, които могат да бъдат разделени на няколко части
  - Произволни атрибути

Akonadi

Libaconadi API

- На ниско ниво
  - Базирано на задачи
  - Асинхронно изпълнение
- На високо ниво
  - Qt Model/View

Akonadi
Resource agents

- Свързване на Akonadi с външни източници на данни
  - Локални файлове (ical, vcard, maildir,...)
  - Пощенски или groupware сървъри
  - Web услуги (facebook, del.ico.us, ...)
- Конвертиране между различни формати на данните
- Записване на промените извършени offline

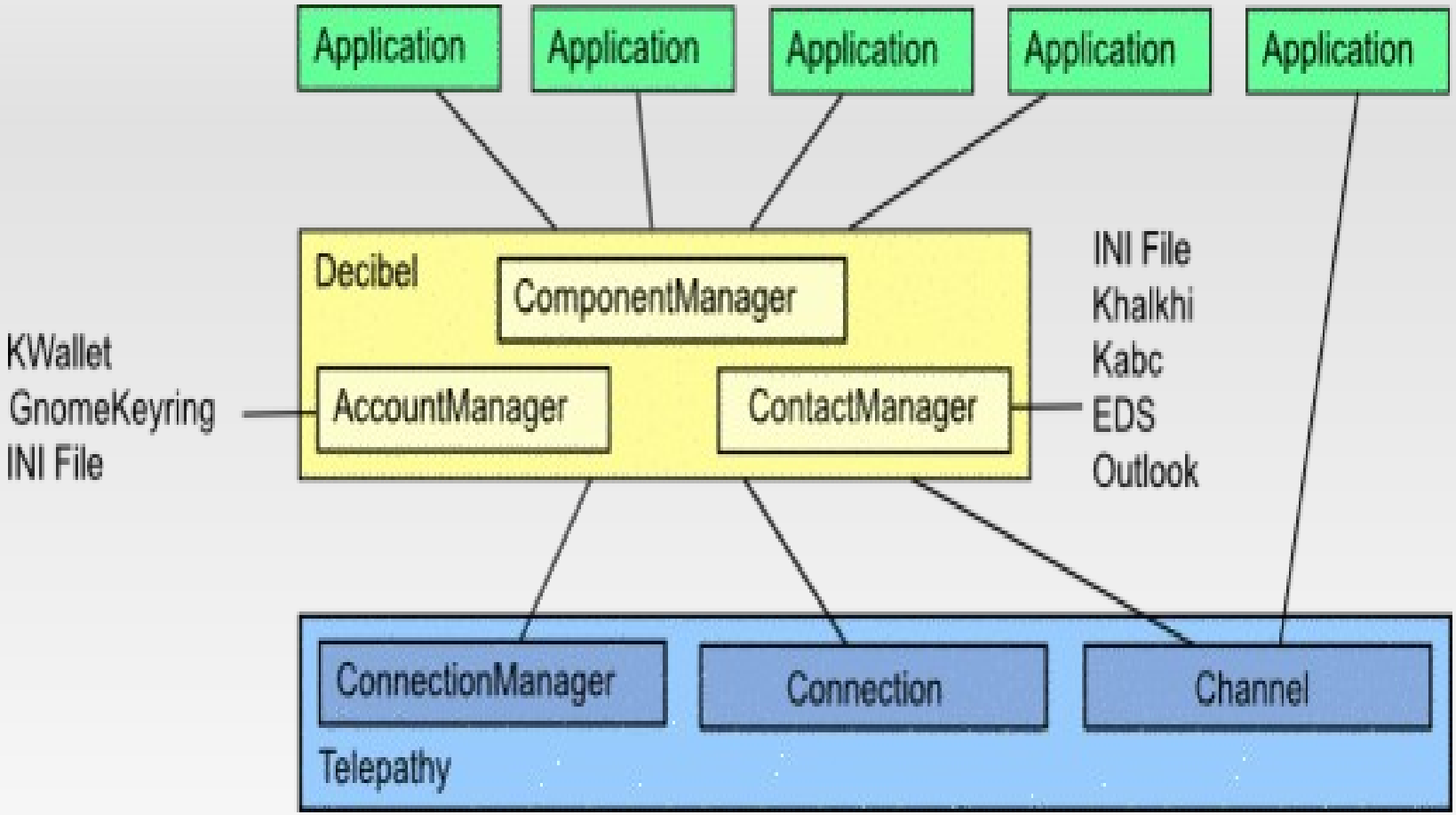
Decibel

Какво е и какво не е?

- Услуга, а не приложение
- Лесна интеграция на комуникациите в реално време в отделните приложения
- Базирано на Telepathy

Decibel

Архитектура



Nepomuk

- Видове метаданни
  - Метаданни съхранявани във файловете (exif, mp3 tags, ...)
  - Метаданни въвеждани от потребителя (етикети, коментари, ...)
  - Метаданни генерирани автоматично
- Nepomuk се грижи за последните две

Nepomuk

извличане на данни

```
Nepomuk::Resource res( uri );
QHash<QString, Variant> properties = res.allProperties();

for( QHash<QString, Variant>::const_iterator it = properties.constBegin();
     it != properties.constEnd(); ++it )
{
   QUrl propertyUri = it.key();
   Variant value = it.value();

   Nepomuk::Types::Class propertyType( propertyUri );

   someList->appendItem( propertyType.label() + ": " + value.toString() );
}
```

Nepomuk
добавяне на данни

- Добавяне на етикет

```
Nepomuk::Tag tag( "This is my nice tag name" );
Nepomuk::Resource res( uri );
res.addTag( tag );
```

- Добавяне на коментар

```
Nepomuk::Resource res( uri );
QString comment = getFancyFileComment();
res.setProperty( Soprano::Vocabulary::NAO::description(), comment );
```

Nepomuk

какво може да добавяме

- setAltLabels (const QStringList &value)
- setAnnotations (const QList< Resource > &value)
- setDescription (const QString &value)
- setIdentifiers (const QStringList &value)
- setIsRelateds (const QList< Resource > &value)
- setIsTopicOfs (const QList< Resource > &value)
- setLabel (const QString &value)
- setProperty (const QUrl &uri, const Variant &value)
- setRating (const quint32 &value)
- setSymbols (const QStringList &value)
- setTags (const QList< Tag > &value)
- setTopics (const QList< Resource > &value)
- setTypes (const QList< QUrl > &types)

Solid

- Позволява на разработчиците да имат лесен достъп до хардуера на всички платформи
- Не предоставя достъп до хардуера, а само информация за него
- За мрежови и bluetooth устройства дава пълен достъп до тях

Solid

един пример

```
QList<Solid::Device> list = Solid::Device::listFromType(Solid::DeviceInterface::Processor, QString());

//take the first processor
Solid::Device device = list[0];
if(device.is<Solid::Processor>()) kDebug() << "We've got a processor!" << list.count() << "to be exact...";
else kDebug() << "Device is not a processor.";

Solid::Processor *processor = device.as<Solid::Processor>();
kDebug() << "This processors maximum speed is: " << processor->maxSpeed();

Solid::Processor::InstructionSets extensions = processor->instructionSets();
kDebug() << "Intel MMX supported:" << (bool)(extensions & Solid::Processor::IntelMmx);
kDebug() << "Intel SSE supported:" << (bool)(extensions & Solid::Processor::IntelSse);
kDebug() << "Intel SSE2 supported:" << (bool)(extensions & Solid::Processor::IntelSse2);
kDebug() << "Intel SSE3 supported:" << (bool)(extensions & Solid::Processor::IntelSse3);
kDebug() << "Intel SSE4 supported:" << (bool)(extensions & Solid::Processor::IntelSse4);
kDebug() << "AMD 3DNOW supported:" << (bool)(extensions & Solid::Processor::Amd3DNow);
kDebug() << "PPC AltiVec supported:" << (bool)(extensions & Solid::Processor::AltiVec);
```

Solid

поддържани устройства

- Solid::AcAdapter
- Solid::AudioInterface
- Solid::Battery
- Solid::Block
- Solid::Button
- Solid::Camera
- Solid::DvbInterface
- Solid::GenericInterface
- Solid::NetworkInterface
- Solid::PortableMediaPlayer
- Solid::Processor
- Solid::StorageAccess
- Solid::StorageDrive
  - Solid::OpticalDrive
- Solid::StorageVolume
  - Solid::OpticalDisc
- Solid::Video

Solid

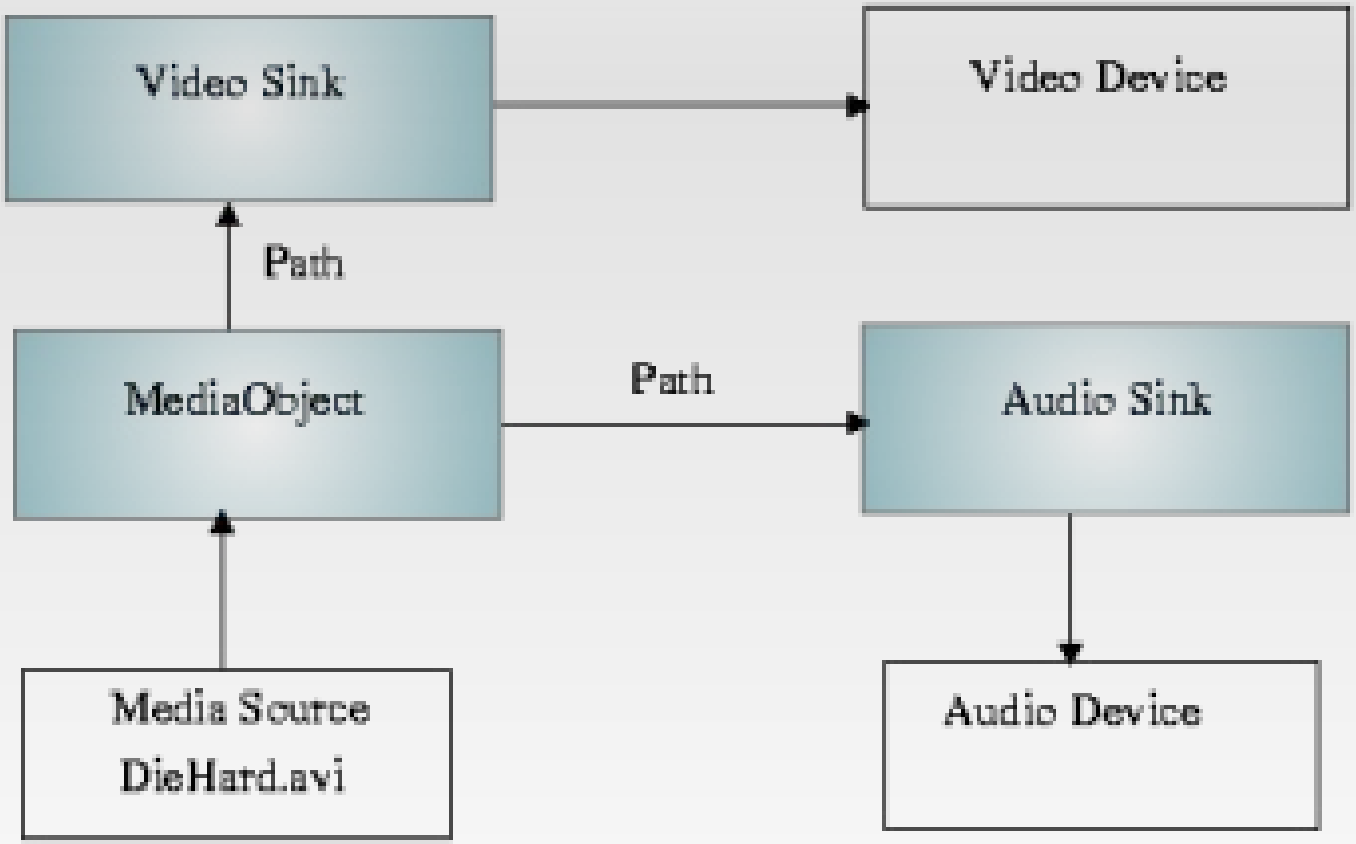
следене на статуса

- Solid::DeviceNotifier
- Solid::Networking::Notifier
- Solid::PowerManagement::Notifier

Phonon

- Мултимедийна библиотека
- Част от Qt
- Използва външни програми
  - DirectShow на Windows
  - QuickTime на Mac Os
  - Gstreamer на Linux

Phonon
архитектура



Phonon

примерче

```
Phonon::MediaObject *mediaObject = new Phonon::MediaObject(this);
mediaObject->setCurrentSource(
    Phonon::MediaSource("/mymusic/barbiegirl.wav"));

Phonon::AudioOutput *audioOutput =
    new Phonon::AudioOutput(Phonon::MusicCategory, this);

Phonon::Path path = Phonon::createPath(mediaObject, audioOutput);
```

Благодаря за вниманието!

Ако имате въпроси, погледнете на:

- http://techbase.kde.org
- http://api.kde.org
- http://websvn.kde.org/trunk