Okular
Simply a document viewer?

Pino Toscano, pino@kde.org

June 30th, 2007
Outline

1. Introduction
2. Expanding Okular
1 Introduction

2 Expanding Okular
What is okular?

The KDE 4 document viewer
Included with kdegraphics.
Based on the awarded technology called KPDF.

Different document types in a single application
No more need for different applications to read different types of documents.

Reading aids
Different ways to ease the reading of a document.
Features
What does okular bring to the user?

“Old“ KPDF features . . .

- Thumbnails
  Filtering by text search
- Table of Contents
- Links and hyperlinks
- Presentation mode
- Customizable view
  Single/facing, continuous
- Text extraction
Features (2)

What does okular bring to the user?

... and new features

- Multicolumns view
- Pages rotation
- Annotations
- Bookmarks
- Preliminary forms support
  no ActionScript, no possibility to submit
- Embedded files in documents
- Multimedia (sounds) support
- And much more ...
Introduction

Expanding Okular

!okular

What okular is not

- A document manipulator (a.l.a. pdftk)
- A document collector (a.l.a. digiKam)
- An image viewer
### Formats

The supported document formats

<table>
<thead>
<tr>
<th>Format</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF</td>
<td>PostScript</td>
</tr>
<tr>
<td>OpenDocument</td>
<td>DVI</td>
</tr>
<tr>
<td>Text</td>
<td>DjVu</td>
</tr>
<tr>
<td>TIFF</td>
<td>XPS</td>
</tr>
<tr>
<td>CHM</td>
<td>FictionBook</td>
</tr>
<tr>
<td>ComickBook</td>
<td>images</td>
</tr>
<tr>
<td>Plucker</td>
<td></td>
</tr>
</tbody>
</table>
Continuous collaboration with the OpenUsability.org project, thanks to the great help of Florian Grässle

KPDF 0.5 got already many usability fixes.

Many of the new solutions were developed taking into account usability.

Okular and the Summer Of Usability

Okular took part in the recent SoU contest.

An Indian student, Sharad Baliyan, was selected.

Results so far:

- surveys about user interface
- a proposal about the toolbar
People
The people behind Okular

- Pino Toscano
- Albert Astals Cid
- Tobias König
- Brad Hards
- Luigi Toscano
- Jiri Klement
- Piotr Szymański
- Florian Grässle
Future
Planned work on Okular

- Better support of forms
- Better handling or annotations
- Possible integration with Nepomuk/Strigi
- OpenDocument generator using the KOffice/Flake library
- More and better document supports
1 Introduction

2 Expanding Okular
Okular provides a public API for writing generators for document types.

Why writing an Okular backend?

- No worries about the view of the document
  Think about writing a “model”…
- Common UI for reading the document
  Single place for usability enhancements and fixes
- New features and bugfixes become available for free for any generator
Okular API
General structure of the Okular API

- Core library
- GUI
- Generators
  - Poppler
  - DjVu
  - ...

Pino Toscano, pino@kde.org — Okular
All in the Okular namespace.

- Document
- Page
- Generator & TextDocumentGenerator
- PixmapRequest
- Action & its hierarchy
- TextPage
- Annotation & its hierarchy
- FormField & its hierarchy
- ...

Pino Toscano, pino@kde.org — Okular
Okular::Generator

bool loadDocument(const QString &fileName, QVector<Okular::Page*> &pagesVector);
bool closeDocument();

Easy job: opening a document and loading the pages from it.

Should also add all the page objects to the Page.
protected:
    QImage image(PixmapRequest *page);

- synchronous
- activating the Threaded feature may help...

Own rendering strategy

bool canGeneratePixmap() const;
void generatePixmap(PixmapRequest *request);
image() is not needed in this case
protected:
    TextPage* textPage(Page *page);
    
    sync/async depending on the type

Own text extraction strategy

bool canGenerateTextPage() const;
void generateTextPage(Page *page, GenerationType type);
textPage() is not needed in this case
- general document information
  (generateDocumentInfo())
- table of contents (generateDocumentSynopsis())
- Fonts in the document (generateDocumentFonts())
- Embedded files (embeddedFiles())
- DRM (isAllowed())
- ...and metaData() for any other kind of metadata
Implementing the printing

bool print(KPrinter &printer);

- synchronous (no workaround at the moment)
- printer has all the needed stuff (size, margins, etc...)
- printing configuration via PrintInterface
class SimpleGenerator : Okular::Generator() {
public:
    SimpleGenerator();
    ~SimpleGenerator();
    bool loadDocument(
        const QString &fileName,
        QVector<Okular::Page*> &pages) {
        // open the document, fill the pages vector
        return whether the loading succeeded;
    }
    bool closeDocument() {
        // cleanup your open document
        return true;
    }
};
Example of more “complicated“ generators (TIFF & Poppler)
Questions

Pino Toscano and the Okular team

okular-devel@kde.org