Welcome

Kexi - Database Environment and Introduction to KDE Database Layer

Jarosław Staniek
js at iidea.pl
Overview

1. Kexi in Details
2. Kexi & KOffice
3. Reusable Components
4. Plans
What is Kexi?

- **Integrated data management application** for KDE (KOffice member)
  - We are talking about *desktop databases*, geared to vertical solutions, user interaction

- Direct competition for:
  - MS Access
  - Filemaker
  - Oracle Forms
  - OpenOffice.org Base
What Kexi is not?

- **It's not a database administering tool**
  (there are many specific administering tasks a database designer is not willing to perform)

- **It's not a database frontend specialized for a single database engine**
  (specific tasks will be always better handled by specialized tools dedicated for a single engine)
- **Late 2002:** project started by Lucijan Bush; contributions from Peter Simonsson and Joseph Wenninger

- **Feb 2003:** joined Jarosław Staniek, current lead developer sponsored by OpenOffice Polska

- **Dec 2003:** joined Cédric Pasteur, now core developer (forms)

- **Jun 2004:** joined Martin Ellis (databases)

- **Sep 2004:** joined Sebastian Sauer (scripting)
Important Milestones

- September 2003: High-level generic database layer (KexiDB)
- May 2004: Visual table and query designers
- October 2004: Graphical support for server databases
- July 2005: Final version of database forms with advanced designer
- June 2005: First version officially bundled with KOffice
- **Database Library (KexiDB)**
  uniform schema designing and database accessing API

- **Data Sheet and Table Designer**
  more flexible than QTable or MS Access Data Sheet:
  - uses KDE editor widgets for custom data types (extendable via plugins)
  - uses data model for database table
  - Record-oriented with navigator, not spreadsheet-like
Main Modules – Stable (2)

- **Query Designer**
  - visual mode
  - SQL mode with SQL parser

- **Database Forms**
  - built-in designer
  - Data Source pane
  - plugins (factories)
  - more user friendly than Qt Designer
Main Modules - In Progress

- **Database reports** with built-in designer
- **Scripting with Python**
- **Macros** (simplified method of scripting)
- **Form templates and autoforms**
Database support - KexiDB

- **Plugin-based** driver system on higher and wider level than QtSQL
- Detailed information about status of every operation (full i18n)
- **KEXISQL** dialect is translated for backends
- **SQLite** as default built-in file-based SQL database engine:
  - no need to use a server
  - endianess-neutral
  - empty database file takes about 10 kilobytes
- **MySQL and PostgreSQL** database servers are currently supported
Opening Data Table View (looks similar in C++)

```python
# setup data

driverManager = kexidb.DriverManager()
driver = drivermanager.driver("SQLite3")
connData = drivermanager.createConnectionData()
db="/home/foo/project.kexi"
connData.setFileName(db)
connection = driver
    .createConnection(connData)
connection.connect()
connection.useDatabase(db)

# setup gui

carsTable = connection.tableSchema("cars")
cursor = connection.prepareQuery( carsTable )
tableView = KexiDataTableView(0, cursor)
tableView.show()
```
Features

- Plugin-based data and project migration tools
  - with wizards, built on top of KexiDB library
  - user can quickly migrate a project from one database engine to another
    - MS Access files supported as data source
- Portable to win32
  (using KDElibs for win32)
- Good database engine-independency (KexiDB)
  - db engines have 1) different APIs, 2) features
    3) SQL dialects
  - libraries like QtSQL only support independency for 1)
A set of formats defined for Kexi database projects:

- table metadata: represented as a simple tabular data
- query: represented as a XML string + SQL statement compatible with KexiSQL parser
- form: XML format inherited from Qt Designer's format
- report: inherited form's XML format
- script: XML string + source code string
- connection data: ini-like format used to store information required to perform a server connection
File Types

- **.kexi** – **project** saved as SQLite-based database file
- **.kexis** – **shortcut to a project** on a database server
- **.kexic** – **shortcut** to a database server connection

- **Note**: if server databases are used, small shortcut files are the only data stored locally.
Proposals to making OpenDocument more generic have been recently submitted to OASIS

- Specific information bits (like Java-dependency) is now removed
- This will allow to exchange schema and data to a larger extent (OpenOffice.org Base, Kexi, others?)
By default, **Kexi stores database schema data in "system" database tables** within the same database as user-defined tables. Advantages:

- If SQLite is used, this gives a **single file** containing both schema and data
- Good level of database independence (easy migration)
- Efficiency, multiuser access and data integrity inherited from database engines
On the contrary:
OpenOffice.org Base stores database schema data in a separate "XML Archive", very much like e.g. OpenOffice.org Writer does. (lack of mentioned advantages)
Integration with KOffice

- **KOffice features used in Kexi**
  - **CSV Import** shared with KSpread (improved, support for Fixed Width Text is planned)
  - Planned: **Embedding KOffice documents** (via KoPart) as frames in Kexi forms and reports
Kexi features provided for other KDE applications:

- Planned: integration with mail merge for KWord
- Integration with KSpread
  (mapping spreadsheet's cells to database rows)
In real world, KOffice documents are in most cases shared by sending them within email.

Documents have to support OASIS specifications. Moreover, they are often converted to Microsoft's proprietary formats.

Data sheets or forms containing "live" data (loaded from a database on demand) require database access.
Possible solution:

- When a document containing live data (loaded from a database) is saved, just save a copy of the data within the document.

Pro:

- Compatibility with OASIS and MS formats can be maintained
Cons:

- This can extremely increase document size
- No client-server/multiuser advantages: data will not be "live" anymore nor editable (unless replication feature is implemented)
- Problems with data updates when database schema or related data has been changed in the meantime

Conclusion:

- Embedding a rich, custom content (like KParts or MS OLE interface) within exchangeable document formats could be a design flaw, inherited from MS formats
Using Database Framework by Other KDE Apps

- **Using Kexi data sources**
  - One shortcut file with description format for data source connection available for all KDE apps (proposed as freedesktop.org specification)
  - Planned: configuring KDE database access in KControl
  - Predefined “open” dialogs for server based data sources:

  ![Conventional file dialog](image)
  ![Dialogs for server-based data source](image)
Using Database Framework by Other KDE Apps (2)

- **Reuse path for developers**
  1. Prepares logical design for app's data structures
  2. Create database schema using Kexi
  3. Add data input and output functions for using KexiDB connectivity library
  4. Reuse data-aware widgets, like data sheet views and forms, if needed

- **Further possibilities**
  - Reuse reporting functionality
  - Enable scripting so data processing can be extended by users
Other Reusable Components

- **Data migration framework**
  e.g. to add CSV files import/export functionality to other applications

- **Form Designer plugin**
  to allow designing data-aware forms within other applications (not yet in public API)

- **Property library with editor widget**
  - supports grouping and custom property values
  - currently merged with KOffice libs as KoProperty
  - reused by Kexi Form/Report Designer and Kugar
using namespace KoProperty;

// setup data model
Set* set = new Set();
set->addProperty(
    new Property("color", Qt::black, i18n("Color"))) ;
set->addProperty(
    new Property("name", "abc", i18n("Name"))) ;
set->addProperty( new Property("size", // composed
    QSize(367,530), i18n("Size")) ) ;

// setup GUI
Editor* editor = new Editor();
editor->changeSet(propSet);
Plans: KPart Database Components
Provided by Kexi

- Data sheet inside Konqueror
- KPart Form based on data source (query) inside Konqueror

- Database components are wrapped to be usable via KParts interface.

- Many actions like design mode are hidden: even if data is read-write, db schema is usually read-only.
- Deploying Kexi projects
  - User mode (i.e. with design actions disabled)
  - Precompiler (creating standalone executables)
- More database drivers: Firebird, Oracle, DB2 and ODBC/JDBC bridges
- More data/schema migration tools
- Scripting with Java Script (KJSEmbed)
- Fine-grained permissions management
- User-defined i18n
More TODOs

- **Asynchronous connections and threading**
  - Multiple processing tasks can be executed concurrently in using kioslaves-like API
  - Support for interrupting operations (reusing database transactions)
  - Better user experience for large data sets or low bandwidth and higher latency

- **Server-side extensions, middleware**
  - Triggers and data processing scripts
  - Data-driven web applications and services
Our Needs

- **Increase interest** in KexiDB within KDE apps and desktop
  - gnome-db is going surprisingly well in this department - about 15 GNOME applications use gnome-db API...

- **Usability studies**, especially for KDE-wide database support

- **Developers!** Developers! Developers! Developers! Wanted for:
  - developing report module, GUIs, especially wizards
  - developing database drivers
  - integrating with other KOffice/KDE apps
· **More Information**
  
  - **Project Home**
    http://www.kexi-project.org/
  
  - **Development Pages**
    http://www.kexi-project.org/wiki/wikiview/
  
  - **KOffice Page**
    http://koffice.org/kexi/