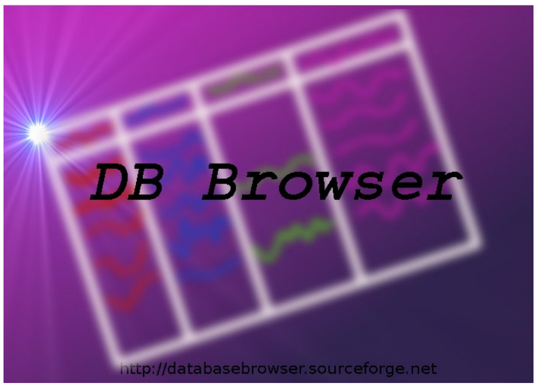

DB Browser
http://databasebrowser.sourceforge.net

Contents

Introduction to DBBrowser

DBBrowser is an open source, cross-platform tool which can be used to view the contents of a database. It supports CLOBS, BLOBS and Oracle XMLTypes. It is designed to work with all the major DBMS (Oracle, MySQL, SQLServer). The user should never have to write SQL to view the data although a SQL window is provided. Support for ER (Entity Relationship) diagrams is planned for the next version.

The main features are:

- View all data in tables without writing SQL
- Support for CLOBs and BLOBS in Oracle and MySQL
- SQL syntax highlighting
- Add, remove, update records without writing SQL
- Add, remove columns from tables without writing SQL
- Export data as CSV, HTML Excel, Word or PDF files
- More than 10 GUI skins to change the look and feel
- Context Sensitive Help
- Internationalized with 2 language packs. Support for plugging in more language packs

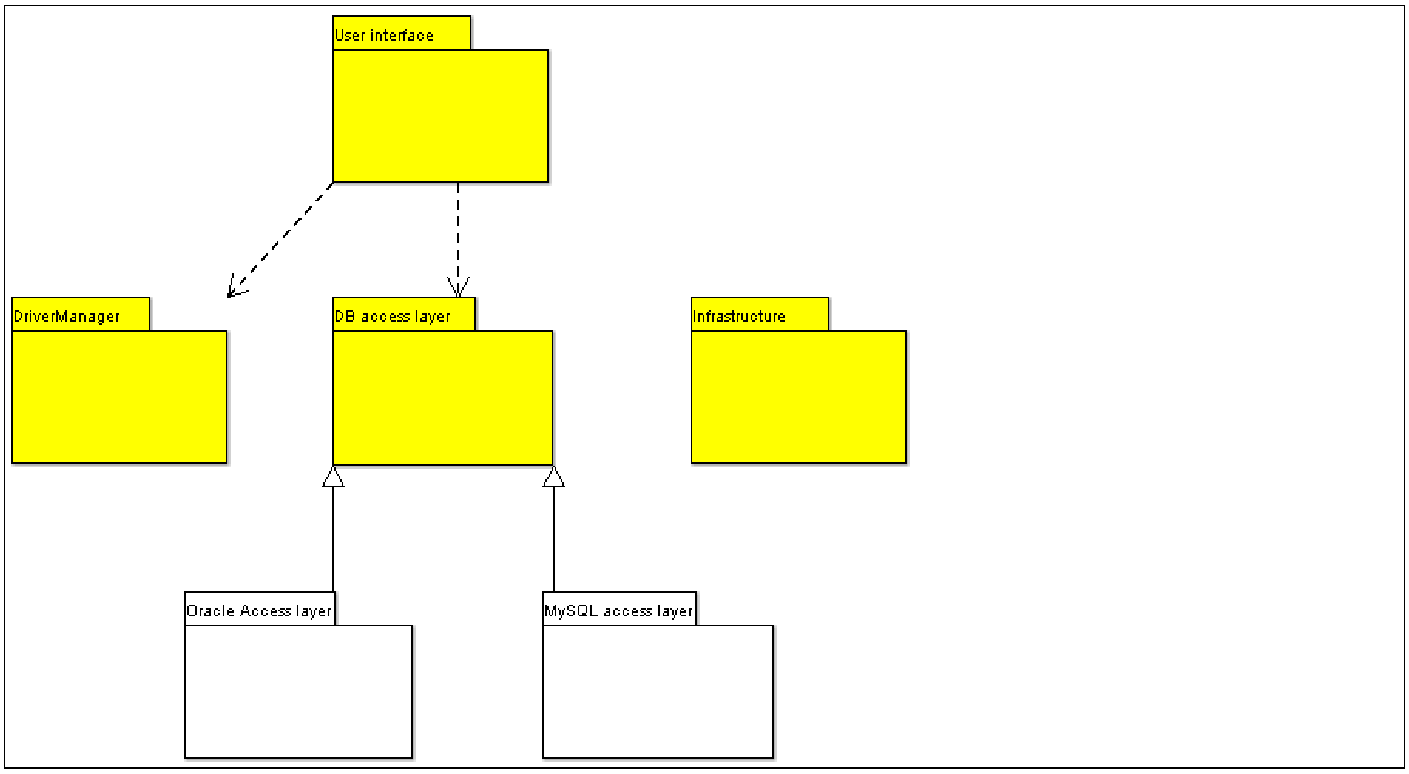
DBBrowser is hosted on the SourceForge website (http://databasebrowser.sourceforge.net) and on Hercules worldwide website (www.herculesworldwide.com/dbbrowser).

Architecture

DBBrowser is divided into 4 layers.

DBBrowser layers

Layer	Function	java package
DB access layer	DBBrowser Engine - with different engine for various DBMS and different engines for Query, update and running raw SQL	org.dbbrowser.db.engine
Infrastructure	Infrastructure services such as set look and feel, logging, property management, internationalization	infrastructure.jar
User interface	User interface	org.dbbrowser.ui
Driver manager	Setup connection to DBMS	org.dbbrowser.drivermanager



Installer and executable

DBBrowser can be downloaded as a Windows installer ('exe' file). The user can download this 'exe' file and double click on it to install DBBrowser. An installer is launched which allows the user to specify some options and install DBBrowser. If a JVM is not present, it will download one for the user.
DBBrowser uses Izpack, an open source installer from IzForge. DBBrowser uses JSmooth, an open source tool to wrap the installer in an 'exe' file so that the installer can be launched on Windows by double clicking on the 'exe' file.
For non-windows and windows platform, a java installer is also provided which can be run from the command line as shown below:

```
java -jar DBBrowser-installer-v0.1-build-1.jar
```

Once the installer has finished, DBBrowser is installed. On Windows, the installer creates an 'exe' file which the user can double click to launch DBBrowser. The DBBrowser launcher for windows is created using launch4j, an open source wrapper for executable jar files.
The whole process of building the executables and installer is automated using Ant. The Ant build file (build.xml) has a target 'build-and-generate-windows-exe-and-installer' which will 'clean', 'compile' and 'build' the project files. The target will then create a windows 'exe' wrapper for the executable jar file, create the installer and wrap the installer in another 'exe' file using JSmooth. The Ant build file uses configuration files to build installer and executables. These files are:

- dbbrowser-launch-4j.cfg - config file for launch4j
- dbbrowser-config-file-for-izpack-installer.xml - config file for izpack installer
- dbbrowser.jsmooth.xml - config file for jsmooth wrapper - wraps installer in 'exe' file

RPM installer for Linux is planned for version 0.2.

Internationalization

Support for internationalisation is implemented through standard Java Resource bundles. DBBrowser includes resource bundle for English only. Version 0.2 onwards will have language packs for other languages as well. The InternationalizationManager class is initialized using an input stream. The InternationalizationManager also has the concept of categories. Messages belong to a specific category. A category has a list of messages. The list of messages is a Java Resource bundle.

Internationalization in DBBrowser

Category	Messages(Resource Bundle)
dbbrowser-core	Starting DBBrowser..., etc
dbbrowser-ui	Please complete all the fields, Check this box to say you have read this, etc

Connection info

Information about the database connections are stored as an XML file. The list of ConnectionInfo objects are converted to XML using Castor(http://www.castor.org). Default Castor mapping is used to convert Java objects to XML.
The purpose of storing the connection info in XML is to make the connection info portable across different database browsing tools. As there is no standard format for storing information about connection to a database, the XML produced using default castor mapping is sufficient for the task. There is no point in using customized XML format. A sample XML fragment with connection info for 2 connections is shown below:

```
<?xml version="1.0" encoding="UTF-8"?> <array-list> <connection-info
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="java:org.dbbrowser.drivermanager.ConnectionInfo"> <DBMSType>MySQL</DBMSType>
<password>rfvyhn573</password> <name>mysql</name>
<driver-class-name>com.mysql.jdbc.Driver</driver-class-name>
<last-used>2006-01-10T15:31:12.078+05:30</last-used>
<database-uRL>jdbc:mysql://localhost:3306/</database-uRL> <username>root</username>
</connection-info> <connection-info> <DBMSType>MsSQL</DBMSType>
<password>september</password> <name>sqltest</name>
<driver-class-name>net.sourceforge.jtds.jdbc.Driver</driver-class-name>
<last-used>2005-12-29T16:30:24.000+05:30</last-used>
<database-uRL>jdbc:jtds:sqlserver://localhost/testdatabase</database-uRL>
<username>gaurav</username> </connection-info> </array-list>
```

Password in the connection info is encrypted and encoded using BASE64 encoding. (TODO - add more info about how to achieve this)
A side effect of using Castor for persisting Connection info as a XML string is that the ConnectionInfo class has Java Bean style getXXX() and setXXX() methods for all the attributes. This is a deviation from the use of Immutable pattern in all other DTOs such as DBTableCell and ColumnInfo. DTOs in DBBrowser such as DBTableCell and ColumnInfo are immutable objects and the values cannot be changed after they have been created.
A connection info represents all the information DBBrowser needs to setup a connection. The connection infos can be setup via the UI when DBBrowser starts up. The XML file with the ConnectionInfos can be loaded into DBBrowser via the File->Open menu. The user can change the connection info via the UI.

JDBC Drivers

DBBrowser should work with MySQL and Oracle DBMS. Support for MS-SQL is planned for version 0.2. DBBrowser does not include any JDBC drivers and the user must download the appropriate JDBC driver. A 'first-time' dialog reminds the user to download the JDBC driver for the DBMS and ‘register’ it with DBBrowser. During registration, DBBrowser associates the JDBC driver with a connection info. When setting a connection, DBBrowser uses the JDBC driver classes in the jar file to setup the connection. The registration can be done through the UI when defining a new connection. The location of the jdbc drive is stored in the 'ConnectionInfo'.

User Interface

The user interface is built using Java Swing. The only other option is to use IBM SWT. A brief comparison of Swing and SWT is shown below:

Comparision of Swing and SWT	
Swing	SWT
Well know in the open source community	Not that well known compared to Swing
Number of Look and Feels available	No know look and feels available except the native look and feel
Designed for building GUIs	Designed for building Eclipse and oriented to building Eclipse although other applications have been written (e.g. Azureus)

Retrieving data

DBBrowser relies on several SQL engines to retrieve data. The data is always returned in the form of a table. DBBrowser has knowledge of the different dialects of SQL understood by various DBMS. Currently, MySQL and Oracle is supported. Other DBMS will be supported in the future. The common SQL commands are run by a Generic SQL Query engine and the DBMS specific commands are run by DBMS specific engines. A good example is the SQL commands to retrieve a subset of the data in the table (paging). They are different for different DBMS.

See 2 examples below for a table with name 'tablename'.
For Oracle, the command is:

```
select * from tablename where rownum <= 100 and ID in ( select ID from tablename group by ID ) minus
select * from tablename where rownum < 1 and ID in ( select ID from tablename group by ID )
```

...where id is the primary key of the table. This SQL will return the first 100 records in the table.

For MySQL, the SQL is:

```
SELECT * FROM tablename LIMIT 1, 100
```

This will also return the first 100 records in the table.

DBBrowser generates different dialects of SQL depending on the DBMS.

Export data

Data retrieved from a database can be exported to various formats. They are:

- CSV files
- PDF (using iText library)
- Microsoft Excel (using jxl library)
- HTML
- SQL

The SQL wizard generates SQL 'insert' statements which can be used to transfer data from one database to another.

A Wizard based interface is used to get the user input. For all formats, the wizard asks the user to specify the following parameters:

- Location of the file
- Format for date (e.g. 04-JAN-06 or 4/1/2006)
- The user can select columns to include in the report(BLOBS and CLOBS are not included)

For PDF and Excel files, the user can specify additional parameters:

- Header and footer
- Background colour for column head
- Portrait or Landscape orientation
- Paper size - A4, Letter, A3, A5 etc
- PDF document meta data such as author name etc.

The wizard will present different options to the user depending on the data format required. The requirements of a wizard are:

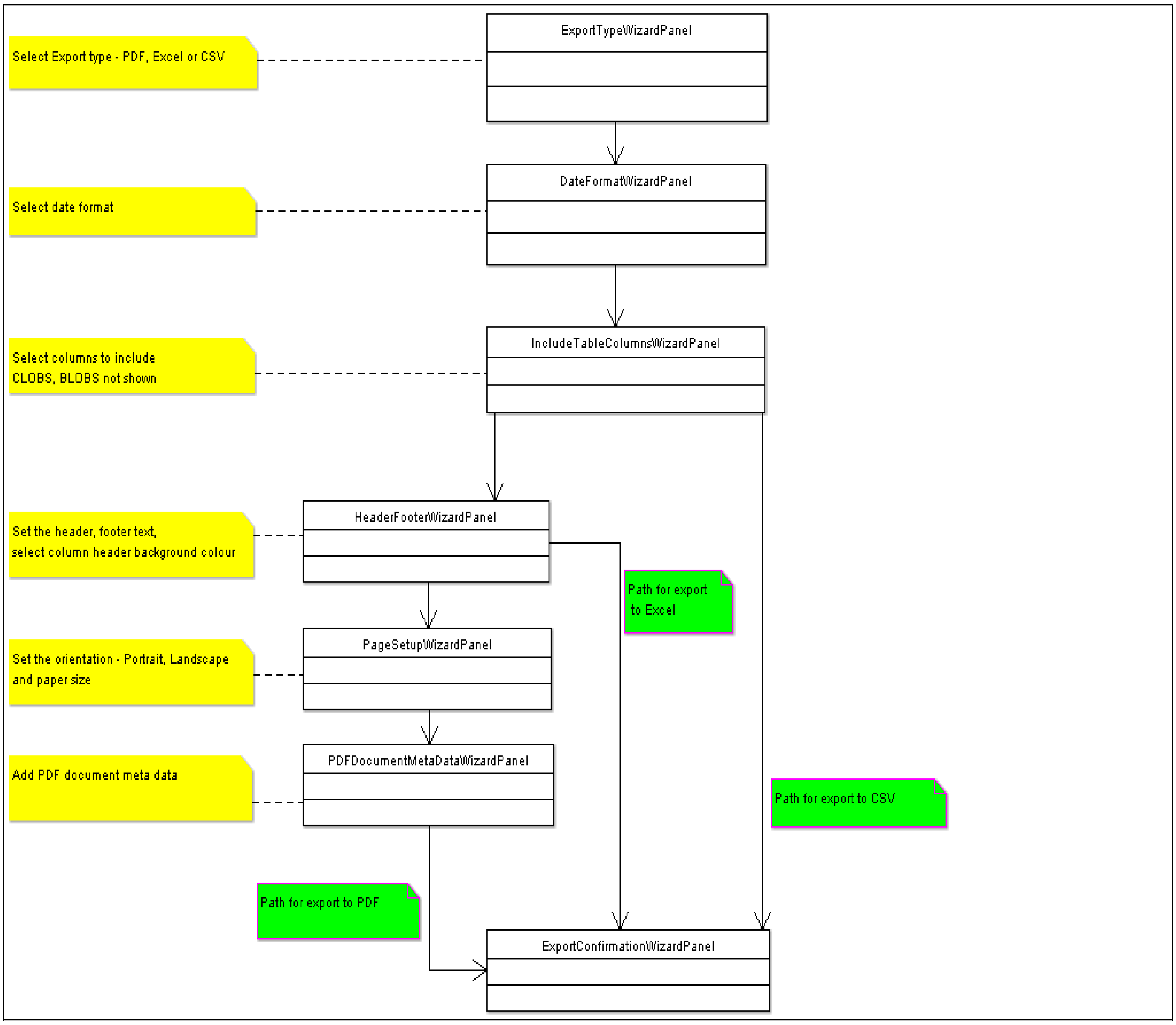
- Wizard should provide means to navigate between panels, handle layout using layout managers
- Allow conditional branching - different panels may have to be shown depending on the user input
- Should be stateful - Data should not be lost while the user navigates between panels.
- The Wizard show conform to standard Java/Sun guidelines for wizards
- It should be well-documented

'Step by Step' tool was used initially but it was difficult to use and comes without any documentation (http://sourceforge.net/projects/stepbystep). The problems faced were:

- Poor documentation

- Badly written code
- Difficult to change wizard navigation once the wizard has been built

Wizards are now built using sample code taken from Sun website (http://java.sun.com/developer/technicalArticles/GUI/swing/wizard/). For more information about wizards, see the Java Look and Feel guidelines (http://java.sun.com/products/jlf/at/book/index.html).


Select Export type - PDF, Excel or CSV
ExportTypeWizardPanel
Select date format
DateFormatWizardPanel
Select columns to include CLOBS, BLOBS not shown
IncludeTableColumnsWizardPanel
Set the header, footer text, select column header background colour
HeaderFooterWizardPanel
Path for export to Excel
Set the orientation - Portrait, Landscape and paper size
PageSetupWizardPanel
Add PDF document meta data
PDFDocumentMetaDataWizardPanel
Path for export to CSV
Path for export to PDF
ExportConfirmationWizardPanel