High-level documentation

This page describes relevant information about the code structure of JabRef precisely and succinctly. Closer-to-code documentation is available at Code HowTos.

We have been successfully transitioning from a spaghetti to a more structured architecture with the `model` in the center, and the `logic` as an intermediate layer towards the `gui` which is the outer shell. There are additional utility packages for `preferences` and the `cli`. The dependencies are only directed towards the center. We have JUnit tests to detect violations of the most crucial dependencies (between `logic`, `model`, and `gui`), and the build will fail automatically in these cases.

The `model` represents the most important data structures (`BibDatases`, `BibEntries`, `Events`, and related aspects) and has only a little bit of logic attached. The `logic` is responsible for reading/writing/importing/exporting and manipulating the `model`, and it is structured often as an API the `gui` can call and use. Only the `gui` knows the user and their preferences and can interact with them to help them solving tasks. For each layer, we form packages according to their responsibility, i.e., vertical structuring. The `model` should have no dependencies to other classes of JabRef and the `logic` should only depend on `model` classes. The `cli` package bundles classes that are responsible for JabRef's command line interface. The `preferences` package represents all information customizable by a user for her personal needs.

We use an event bus to publish events from the `model` to the other layers. This allows us to keep the architecture but still react upon changes within the core in the outer layers. Note that we are currently switching to JavaFX's observables, as this concepts seems as we aim for a stronger coupling to the data producers.

Package Structure

Permitted dependencies in our architecture are:

```
gui --> logic --> model

gui ------------> model

gui ------------> preferences

gui ------------> cli

gui ------------> global classes


logic ------------> model


global classes ------------> everywhere
```

```
cli ------------> model
cli ------------> logic
cli ------------> global classes
cli ------------> preferences
```

All packages and classes which are currently not part of these packages (we are still in the process of structuring) are considered as gui classes from a dependency stand of view.

Most Important Classes and their Relation

Both GUI and CLI are started via the `JabRefMain` which will in turn call `JabRef` which then decides whether the GUI (`JabRefFrame`) or the CLI (`JabRefCLI` and a lot of code in `JabRef`) will be started. The `JabRefFrame` represents the Window which contains a `SidePane` on the left used for the fetchers/groups Each tab is a `BasePanel` which has a `SearchBar` at the top, a `MainTable` at the center and a `PreviewPanel` or an `EntryEditor` at the bottom. Any right click on the `MainTable` is handled by the `RightClickMenu`. Each `BasePanel` holds a `BibDatabaseContext` consisting of a `BibDatabase` and the `MetaData`, which are the only relevant data of the currently shown database. A `BibDatabase` has a list of `BibEntries`. Each `BibEntry` has an ID, a citation key and a key/value store for the fields with their values. Interpreted data (such as the type or the file field) is stored in the `TypedBibentry` type. The user can change the `JabRefPreferences` through the `PreferencesDialog`.
