openLilyLib Integration Roadmap

Bird-Eye's view of the integration of openLilyLib as LilyPond's extension mechanism, plus an outline of a possible design. The actual integration of the extension mechanism seems to be a pretty clearly-cut, disappointingly small task.

Current State

- openLilyLib is a set of extension packages, providing high-level functionality
- `oll-core` provides the basic interface and implements required and additional functionality.

Overall Outline

- Relevant functionality will be moved from `oll-core` to LilyPond proper, providing the main extension mechanism as well as some of the additional functionality (option handling, logging ["oll:" function to report status and errors]).
- The extension mechanism provides the commands `\loadPackage` and `\loadModule`, and it allows configuration options to be passed to either. It also handles package dependencies and implicit loading. (one example for using options (just for getting an idea) would be `\loadPackage \with { modules = annotate.choice } scholarly`.)
- This will *not* be called openLilyLib anymore (but "is" LilyPond's unnamed extension mechanism).
- A core extension library shipping with LilyPond will be initiated. Extensions that are considered core functionality (prime candidates: edition-engraver, stylesheets) will eventually be moved here from openLilyLib, additionally special functionality (e.g. gregorian.ly, arabic.ly) may over time be moved there to expose the difference between core functionality and use-case specific modules more clearly. These tools will then be called through `\loadModule` instead of `\include`, which will be easy to handle with convert-ly rules. Probably it would be a good idea to eventually expose *all* non-standard notation through explicit packages and have that nicely describe in the LM. This too will not be called openLilyLib.
- openLilyLib will remain similar to what it is now, a curated collection of standard packages that are not mature enough or too specific to be included in LilyPond itself. The extension mechanism will find openLilyLib packages through LilyPond's search path.
- Packages can be stored anywhere on disk (=> LilyPond search path), so on the long term it may be interesting to set up an uncurated package manager like npm or CTAN/TeX Live.
- Frescobaldi will provide a user interface to manage extensions but also for using them in documents (e.g. it will know about available configuration options and provide an action (context menu/dialog) to use a given package).