Code Howtos

This page provides some development support in the form of howtos. See also High Level Documentation.

Generic code how tos

We really recommend reading the book Java by Comparison.

Please read https://github.com/cxxr/better-java.

- try not to abbreviate names of variables, classes or methods
- use lowerCamelCase instead of snake_case
- name enums in singular, e.g. `Weekday` instead of `Weekdays` (except if they represent flags)

Dependency injection

JabRef uses a fork of the afterburner.fx framework by Adam Bien.

The main idea is to get instances by using `Injector.instantiateModelOrService(X.class)`, where `X` is the instance one needs. The method `instantiateModelOrService` checks if there is already an instance of the given class. If yes, it returns it. If not, it creates a new one. A singleton can be added by `com.airhacks.afterburner.injection.Injector#setModelOrService(X.class, y)`, where X is the class and y the instance you want to inject.

Cleanup and Formatters

We try to build a cleanup mechanism based on formatters. The idea is that we can register these actions in arbitrary places, e.g., onSave, onImport, onExport, cleanup, etc. and apply them to different fields. The formatters themselves are independent of any logic and therefore easy to test.

Example: NormalizePagesFormatter

Drag and Drop

Drag and Drop makes usage of the Dragboard. For JavaFX the following tutorial is helpful. Note that the data has to be serializable which is put on the dragboard. For drag and drop of Bib-entries between the maintable and the groups panel, a custom Dragboard is used, `CustomLocalDragboard` which is a generic alternative to the system one.

For accessing or putting data into the Clipboard use the `ClipboardManager`.

Get the JabRef frame panel

`JabRefFrame` and `BasePanel` are the two main classes. You should never directly call them, instead pass them as parameters to the class.

Get Absolute Filename or Path for file in File directory

JabRef stores files relative to one of multiple possible directories. The convert the relative path to an absolute one, there is the `find` method in `FileUtil`:

```
org.jabref.logic.util.io.FileUtil.find(org.jabref.model.database.BibDatabaseContext, java.lang.String, o
```

`String path` Can be the files name or a relative path to it. The Preferences should only be directly accessed in the GUI. For the usage in logic pass them as parameter

Get a relative filename (or path) for a file

JabRef offers multiple directories per library to store a file.. When adding a file to a library, the path should be stored relative to “the best matching” directory of these. This is implemented in `FileUtil`:

```
org.jabref.logic.util.io.FileUtil.relativize(java.nio.file.Path, org.jabref.model.database.BibDatabaseCo
```

Setting a Directory for a .bib File

- `@comment{jabref-meta: fileDirectory:<directory>`
- “fileDirectory” is determined by Globals.pref.get(“userFileDir”) (which defaults to “fileDirectory”
- There is also “fileDirectory-<username>”, which is determined by Globals.prefs.get(“userFileDirIndividual”)
- Used at DatabasePropertiesDialog

How to work with Preferences

`model` and `logic` must not know `JabRefPreferences`. See `ProxyPreferences` for encapsulated preferences and https://github.com/JabRef/jabref/pull/658 for a detailed discussion.

See https://github.com/JabRef/jabref/blob/master/src/main/java/org/jabref/logic/preferences/TimestampPreferences.java (via https://github.com/JabRef/jabref/pull/3092) for the current way how to deal with preferences.

Defaults should go into the model package. See Comments in this Commit

UI

Global variables should be avoided. Try to pass them as dependency.

“Special Fields”

keywords sync

`Database.addDatabaseChangeListener` does not work as the `DatabaseChangedEvent` does not provide the field information. Therefore, we have to use `BibtexEntry.addPropertyChangeListener(VetoableChangeListener listener)`.

Working with BibTeX data

Working with authors

You can normalize the authors using `org.jabref.model.entry.AuthorList.fixAuthor_firstNameFirst(String)`. Then the authors always look nice. The only alternative containing all data of the names is `org.jabref.model.entry.AuthorList.fixAuthor_lastNameFirst(String)`. The other `fix...` methods omit data (like the “von” parts or the junior information).

Benchmarks

- Benchmarks can be executed by running the `jmh` gradle task (this functionality uses the JMH Gradle plugin)
- Best practices:
  - Read test input from `@State` objects
  - Return result of calculations (either explicitly or via a `BlackHole` object)
- List of examples

Measure performance

Try out the YourKit Java Profiler.

equals

When creating an `equals` method follow:

1. Use the `==` operator to check if the argument is a reference to this object. If so, return `true`.
2. Use the `instanceof` operator to check if the argument has the correct type. If not, return `false`.
3. Cast the argument to the correct type.

4. For each “significant” field in the class, check if that field of the argument matches the corresponding field of this object. If all these tests succeed, return `true` otherwise, return `false`.
5. When you are finished writing your `equals` method, ask yourself three questions: Is it symmetric? Is it transitive? Is it consistent?

Also, note:

- Always override `hashCode` when you override equals (`hashCode` also has very strict rules) (Item 9 ofEffective Java)
- Don’t try to be too clever
- Don’t substitute another type for `Object` in the equals declaration

Files and Paths

Always try to use the methods from the nio-package. For interoperability, they provide methods to convert between file and path.
https://docs.oracle.com/javase/tutorial/essential/io/path.html Mapping between old methods and new methods https://docs.oracle.com/javase/tutorial/essential/io/legacy.html#mapping

---

TABLE OF CONTENTS