Handle different bibentry formats of fetchers by adding a layer

Context and Problem Statement

All fetchers (except IDFetchers) in JabRef return BibEntries when fetching entries from their API. Some fetchers directly receive BibTeX entries from their API, the other fetchers receive their entries in some kind of exchange format such as JSON or XML and then parse this into BibEntries. Currently, all fetchers either return BibEntries in BibTeX or BibLaTeX format. This can lead to importing BibEntries of one format in a database of the other format. How can this inconsistency between fetchers, and their used formats be addressed?

Considered Options

- Pass fetchers the format, they have to create entries accordingly (in the correct format).
- Pass fetchers the format, they have to call a conversion method if necessary (in the correct format).
- Let the caller handle any format inconsistencies and the conversion.
- Introduce a new layer between fetchers and caller, such as a FetcherHandler, that manages the conversion

Decision Outcome

Chosen option: “Introduce a new layer between fetchers and caller, such as a FetcherHandler, that manages the conversion”, because it can compose all steps required during importing, not only format conversion of fetched entries. As described here (comment)

Pros and Cons of the Options

Introduce a new layer between fetchers and caller, such as a FetcherHandler, that manages the conversion

- Good, because fetchers do not have to think about conversion (Separation of concerns)
- Good, because no other code that currently relies on fetchers has to do the conversion
- Good, because this layer can be used for any kind of import to handle all conversion steps (not only format). As described here (comment)
- Good, because this layer can easily be extended if the import procedure changes

- Bad, because this requires a lot of code changes
- Bad, because this has to be tested extensively

Pass fetchers the format, they have to call a conversion method if necessary

- Good, because less code has to be written than with option “Pass fetchers the format, they have to create entries accordingly”
- Good, because code is already tested
- Good, because keeps all conversion code centralized (code reuse)
- Bad, because fetcher first creates the BibEntry in a possibly “wrong” format, this can easily lead to bugs due to e.g. code changes
- Bad, because adds dependency

Pass fetchers the format, they have to create entries accordingly

- Good, because fetchers already handle BibEntry creation (in their format of choice). This is part of his responsibility.
- Good, because fetchers only create BibEntries of the “correct” format. At no point there exists the chance of the wrong format being passed on due to e.g. code changes.
- Good, because the conversion does not have to take place
- Bad, because fetcher has to “know” all differences of the formats -> clutters the code.
- Bad, because this code has to be tested. Conversion already exists.

Let the caller handle any format inconsistencies and the conversion

- Good, because fetcher code does not have to change
- Good, because fetcher only has to fetch and does not need to know anything about the formats
- Bad, because programmers might assume that a certain format is used, e.g. the preferred format (which would not work as the databases that imports the entries does not have to conform to the preferred format)
- Bad, because at every place where fetchers are used, and the format matters, conversion has to be used, creating more dependencies

---