Localization

More information about this topic from the translator side is provided at Translating JabRef Interface.

All labeled UI elements, descriptions and messages shown to the user should be localized, i.e., should be displayed in the chosen language.

JabRef uses ResourceBundles (see Oracle Tutorial) to store `key=value` pairs for each String to be localized.

Localization in Java code

To show a localized String the following `org.jabref.logic.l10n.Localization` has to be used. The Class currently provides three methods to obtain translated strings:

```
    public static String lang(String key);

    public static String lang(String key, String... params);

    public static String menuTitle(String key, String... params);
```

The actual usage might look like:

```
    Localization.lang("Get me a translated String");
    Localization.lang("Using %0 or more %1 is also possible", "one", "parameter");
    Localization.menuTitle("Used for Menus only");
```

Localization in FXML

To write a localized string in FXML file, prepend it with `%`, like in this code:

```
<HBox alignment="CENTER_LEFT">
    <Label styleClass="space-after" text="%Want to help?" wrapText="true"/>
    <Hyperlink onAction="#openDonation" text="%Make a donation"/>
    <Label styleClass="space" text="%or" wrapText="true"/>
    <Hyperlink onAction="#openGithub" text="%get involved"/>
</HBox>
```

General hints

- Use the String you want to localize directly, do not use members or local variables: `Localization.lang("Translate me");` instead of `Localization.lang(someVariable)` (possibly in the form `someVariable = Localization.lang("Translate me")`
- Use `%x`-variables where appropriate: `Localization.lang("Exported %0 entry(s).", number)` instead of `Localization.lang("Exported ") + number + Localization.lang(" entry(s).");`
- Use a full stop/period (“.”) to end full sentences
- For pluralization, use a combined form. E.g., `Localization.lang("checked %0 entry(s)")`.

Checking for correctness

The tests in `org.jabref.logic.l10n.LocalizationConsistencyTest` check whether translation strings appear correctly in the resource bundles.

Adding a new key

1. Add new `Localization.lang("KEY")` to Java file. Run the `org.jabref.logic.LocalizationConsistencyTest`.
2. Tests fail. In the test output a snippet is generated which must be added to the English translation file.
3. Add snippet to English translation file located at `src/main/resources/l10n/JabRef_en.properties`
4. Please do not add translations for other languages directly in the properties. They will be overwritten by Crowdin

Adding a new Language

1. Add the new Language to the Language enum in https://github.com/JabRef/jabref/blob/master/src/main/java/org/jabref/logic/l10n/Language.java
2. Create an empty <locale code>.properties file
3. Configure the new language in Crowdin

If the language is a variant of a language `zh_CN` or `pt_BR` it is necessary to add a language mapping for Crowdin to the crowdin.yml file in the root. Of course the properties file also has to be named according to the language code and locale.

Background information

The localization is tested via the class LocalizationConsistencyTest.