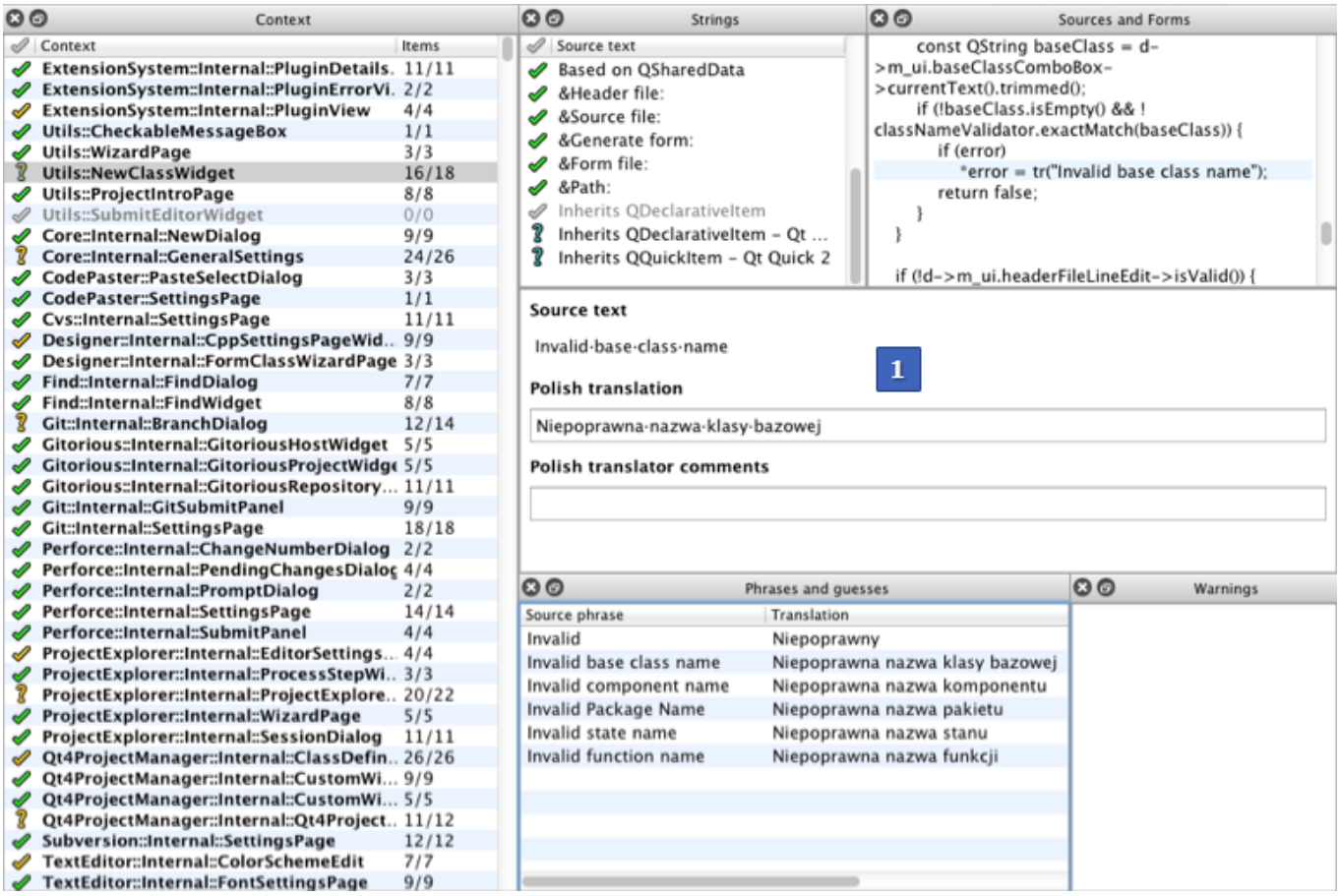
Qt 语言学家手册：译员

*Qt语言学家*是一个为Qt应用程序添加翻译的工具。安装 Qt 后，您可以像启动开发主机上的任何其他应用程序一样启动 *Qt 语言学家*。

*Qt 语言学家*主窗口包含一个菜单栏和以下视图：

- **上下文** （F6），用于从显示要翻译的字符串的上下文列表中进行选择。
- **字符串** （F7），用于查看在上下文中找到的可翻译字符串。
- **源和表单** （F9），用于查看使用当前字符串的上下文（如果上下文的源代码可访问）。
- 用于翻译字符串的翻译区域。
- **短语和猜测** （F10），用于查看当前字符串的可能翻译。
- **用于**查看未通过验证测试的已翻译字符串的警告 （F8）。



翻译区域 （1） 始终可见。若要显示或隐藏其他视图，请选择"**查看**>**视图**"，或使用上面列出的键盘快捷方式。您可以拖动视图的标题栏，并将它们排列在翻译区域周围，甚至在主窗口之外。

翻译字符串

您可以在 *Qt 语言学家*中打开翻译源 （TS） 文件进行翻译。TS 文件是人类可读的 XML 文件，其中包含源短语及其翻译。TS文件通常由羽毛创建和更新。如果您没有 TS 文件，请参阅发布管理器以了解如何生成一个。

您还可以使用 *Qt 语言学家*翻译由其他程序生成的国际 XML 本地化交换文件格式 （XLIFF） 中的文件。但是，对于标准 Qt 项目，仅使用 TS 文件格式。XLIFF 格式文件支持的最低版本是 1.1。

*Qt语言学家*在翻译区域显示目标语言，并相应地调整复数形式的输入字段数量。当您同时打开多个 TS 文件进行翻译时，**将显示**每种语言的**“转换器”和“转换器”注释**字段。有关设置位置信息的详细信息，请参阅更改目标区域设置。

如果开发人员提供了消除歧义的注释，则该注释将显示在**“开发人员注释”**字段中。

要翻译字符串：

1. 选择**“文件”>“打开”**以加载 TS 文件。
2. 在**“上下文”**视图中选择一个上下文，以将在该上下文中找到的可翻译字符串加载到**“字符串”**视图中。
3. 选择一个字符串以将其复制为翻译区域中的**源文本**。源文本中的空格是可视化的，以便您可以看到文本中所需的间距。
4. 在“翻译”字段中输入当前字符串的**翻译**。

   您还可以通过双击**“短语和猜测”**视图从该翻译中选择现有翻译。这些短语是从短语书中读取的，并且猜测是TS文件中类似短语的现有翻译。

5. （可选）在译员注释字段中输入要由其他译员阅读的**注释**。
6. 若要接受翻译，请按 **Ctrl+Enter**，选择“  ”，或单击字符串列表中所选源字符串左侧的图标。
7. 选择**“文件”>“保存”**以保存您的工作。

重复此过程，直到字符串列表中的所有字符串都标记为“✔”（**已接受/正确**）”或“✔”（**已接受/警告**）”。然后选择下一个上下文并继续。

要查看源文本和已翻译文本中的字数和字符数，请选择**“查看”>“统计信息”**。

选择**“文件”>“发布”**以创建与当前翻译源文件具有相同基本名称的 Qt 消息 QM 文件。发布管理器的命令行工具 lrelease 对应用程序*的所有*翻译源文件执行相同的功能。

要打印翻译源和翻译，请选择**文件>打印**。

要退出 *Qt 语言专家*，请选择**文件>退出**。

Leaving a Translation for Later

If you wish to leave a translation, press **Ctrl+L** (Next Unfinished) to move to the next unfinished translation. To move to the next translation (whether finished or unfinished) press **Shift+Ctrl+L**. You can also navigate using the **Translation** menu. If you want to go to a different context entirely, click the context you want to work on in the **Context** list, then click the source text in the **Strings** view.

Phrases That Require Multiple Translations Depending on Context

The same phrase may occur in two or more contexts without conflict. Once a phrase has been translated in one context, *Qt Linguist* notes that the translation has been made and when the translator reaches a later occurrence of the same phrase, *Qt Linguist* provides the previous translation as a possible translation candidate in the **Phrases and Guesses** view.

such comments are used, the duplicate phrases appear in the **Context** view. The developer's comments appear in the translation area on a light blue background.

Changing Keyboard Accelerators

A keyboard accelerator is a key combination that, when pressed, causes an application to perform an action. There are two kinds of keyboard accelerators: Alt key and Ctrl key accelerators.

Alt Key Accelerators

Alt key accelerators are used in menu selection and on buttons. The underlined character in a menu item or button label signifies that pressing the Alt key with the underlined character will perform the same action as clicking the menu item or pressing the button. For example, most applications have a *File* menu with the "F" in the word "File" underlined. In these applications the *File* menu can be invoked either by clicking the word "File" on the menu bar or by pressing *Alt+F*. To identify an accelerator key in the translation text ("File") precede it with an ampersand: *&File*. If a string to be translated has an ampersand in it, then the translation for that string should also have an ampersand in it, preferably in front of the same character.

The meaning of an Alt key accelerator can be determined from the phrase in which the ampersand is embedded. The translator can change the character part of the Alt key accelerator, if the translated phrase does not contain the same character or if that character has already been used in the translation of some other Alt key accelerator. Conflicts with other Alt key accelerators must be avoided within a context. Note that some Alt key accelerators, usually those on the menu bar, may apply in other contexts.

Ctrl Key Accelerators

Ctrl key accelerators can exist independently of any visual control. They are often used to invoke actions in menus that would otherwise require multiple keystrokes or mouse clicks. They may also be used to perform actions that do not appear in any menu or on any button. For example, most applications that have a *File* menu have a *New* submenu item in the *File* menu. The *New* item might appear as "New Ctrl+N" in the *File* menu, meaning the *New* menu can be invoked by simply pressing **Ctrl+N**, instead of either clicking *File* with the mouse and then clicking *New* with the mouse, or by entering *Alt+F* and *N*.

Each Ctrl key accelerator is shown in the **Strings** view as a separate string, for example **Ctrl+Enter**. Since the string does not have a context to give it meaning, such as the context of the phrase in which an Alt key accelerator appears, the translator must rely on the UI developer to include a disambiguation comment to explain the action the Ctrl key accelerator is meant to perform. This disambiguating comment (if provided by the developer) will appear under **Developer comments** in the translation area below the **Source text** field.

Ideally, translations for Ctrl key accelerators are simply copied by selecting **Translation** > **Copy from source text**. However, in some cases the character will not make sense in the target language, and it must be changed. Whichever character (alpha or digit) is chosen, the translation must be in the form "Ctrl+" followed by the upper case character. Qt will automatically display the correct name at run-time. As with Alt key accelerators, if the translator changes the character, the new character must not conflict with any other Ctrl key accelerator.

> **Warning:** Do not translate the "Alt", "Ctrl" or "Shift" parts of the accelerators. Qt relies on these strings being there. For supported languages, Qt automatically translates these strings.

Handling Numbered Arguments and Plurals

Some phrases contain numbered arguments. A numbered argument is a placeholder that will be replaced with text at run-time. A numbered argument appears in a source string as a percent sign followed by a digit. Consider an example: . In this string to be translated, and are numbered arguments. At run-time, and will be replaced with the first and second file names respectively. The same numbered arguments must appear in the translation, but not

string.`After processing file %1, file %2 is next in line%1%2%1%2Datei %2 wird bearbeitet, wenn Datei %1 fertig ist%i`

The use of numbered arguments is often accompanied by the use of plurals in the source text. In many languages, the form of the text will depend on the value shown, and more than one translation is required. If the developers have marked up the source text in correct way, fields for each of the possible plural forms will be available in the translation area. For more information, see Writing Source Code for Translation.

Changing the Target Locale

You can set the locale information explicitly in **Edit** > **Translation File Settings**. If the target language and country are not explicitly set when you open a translation source file, *Qt Linguist* attempts to deduct them from the translation source file name. This requires that the translation files adhere to the following file name convention: , where:`appname_language[_country].ts`

- `language` is an ISO 639 language code in lowercase.
- `country` is an ISO 3166 two-letter country code in uppercase.

If this attempt to resolve the target language and country fails, the **Translation File Settings** window opens.

For example, sets the target language to German, and sets the target language to German and the target country to Switzerland. This also helps loading translations for the current locale automatically. For more information, see Developers.`app_de.tsapp_de_ch.ts`

Selecting Context to Translate

The **Context** view lists the contexts in which strings to be translated appear. The column labeled **Context** lists the context names in alphabetical order. Each context is the name of a subclass of QObject. There can also be a context for QObject itself, which contains strings passed to the static function QObject::tr(). There can also be an *<unnamed context>*, which contains strings that are not in a subclass of QObject.

The following icons indicate the current translation state for each context:

State	Icon	Description
Accepted/Correct	✔	All strings in the context have been translated, and all the translations passed the validation tests.
Accepted/Warnings	✔	All strings in the context have been translated or marked as translated, but at least one translation failed the validation tests. In the **Strings** view, you can see which string failed the test.
State	Icon	Description

Obsolete		None of the translated strings appears in the context any more. This usually means the context itself no longer exists in the application.

The **Items** column displays the total number of translatable strings in the context and the number of translated strings, separated by a slash (/). If the numbers are equal, all the translatable strings in the context have translations.

Selecting String to Translate

The **Strings** view lists all the translatable strings found in the current context and their translation acceptance state. Selecting a string makes that string the current string in the translation area.

Click the icon in front of a string to change its translation acceptance state. A tick mark, green or yellow, means the string has been translated and the translation has been accepted. A question mark means either that the translation has not been accepted or that the string does not have a translation.

The following icons indicate the current translation state for each string:

State	Icon	Description
Accepted/Correct		The source string has a translation (possibly empty). The user has accepted the translation, and the translation passes all the validation tests. If the translation is empty, the user has chosen to leave it empty. Click the icon to revoke acceptance of the translation and decrement the number of accepted translations in the **Items** column of the gui Context view by 1. The state is reset to **Not Accepted** if the string has a translation, or to **No Translation** if the string's translation is empty. If changes the contents of a string, its acceptance state is automatically reset to **Not Accepted**.`lupdate`
Accepted/Warnings		The user has accepted the translation, but the translation does not pass all the validation tests. The validation test failures are shown in the **Warnings** view. Click the icon to revoke acceptance of the translation. The state is reset to **Validation Failures**, and the number of accepted translations in the **Items** column of the **Context** view is decremented by 1.
Not Accepted		The string has a translation that passes all the validation tests, but the user has not yet accepted the translation. Click the icon or press **Ctrl+Enter** to accept the translation. The state is reset to **Accepted/Correct**, and the number of accepted translations in the **Items** column of the **Context** view is incremented by 1.
No Translation		The string does not have a translation. Click the icon to accept the empty translation anyway. The state is reset to **Accepted/Correct**, and the number of accepted translations in the **Items** column of the **Context** view is incremented by 1.
Validation Failures		The string has a translation, but the translation does not pass all the validation tests. Validation test failures are shown in the **Warnings** view. Click on the icon or press **Ctrl+Return** to accept the translation even with validation failures. The state is reset to **Accepted/Warnings**. We recommended editing the translation to fix the causes of the validation failures. The state will reset automatically to **Not Accepted**, when all the failures have been fixed.
Obsolete		The string is obsolete. It is no longer used in the context. See the Release Manager for instructions on how to remove obsolete messages from the file.

If the source files containing the translatable strings are available to *Qt Linguist*, the **Sources and Forms** view shows the source context of the current string in the **Strings** view. The source code line containing the current string should be shown and highlighted. If the file containing the source string is not found, the expected absolute file path is shown.

If the source context shows the wrong source line, it probably means the translation file is out of sync with the source files. For more information about how to re-sync the translation file with the source files, see Using lupdate.

Forms created by *Qt Designer* are stored in special UI files. *Qt Linguist* can make use of these UI files to show the translations done so far on the form itself if the UI files are available to it during the translation process.

Reusing Translations

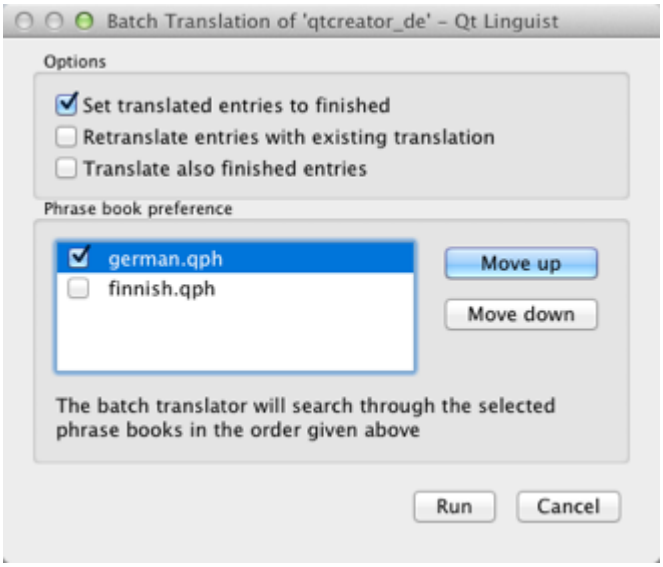
If the translated text is similar to the source text, select **Translation** > **Copy from source text** (or press **Ctrl+B**) to copy the source text into the translation area.

In *Qt Linguist*, *phrase books* are used to provide a common set of translations to help ensure consistency. A phrase book is a set of source phrases, target (translated) phrases, and optional definitions. Typically, one phrase book is created per language and family of applications. Phrase books can also be used to avoid duplication of effort since the translations for a family of applications can be produced once in the phrase book.

If the current string in the **Strings** view appears in one or more of the phrase books that have been loaded, the current string and its phrase book translations are listed in the **Phrases and Guesses** view. If the current string is the same as, or similar to, another string that has already been translated, that other string and its translation are also listed in this view.

To copy a translation from the **Phrases and Guesses** view to the translation area, double-click it or select it and press **Enter**.

Batch Translation

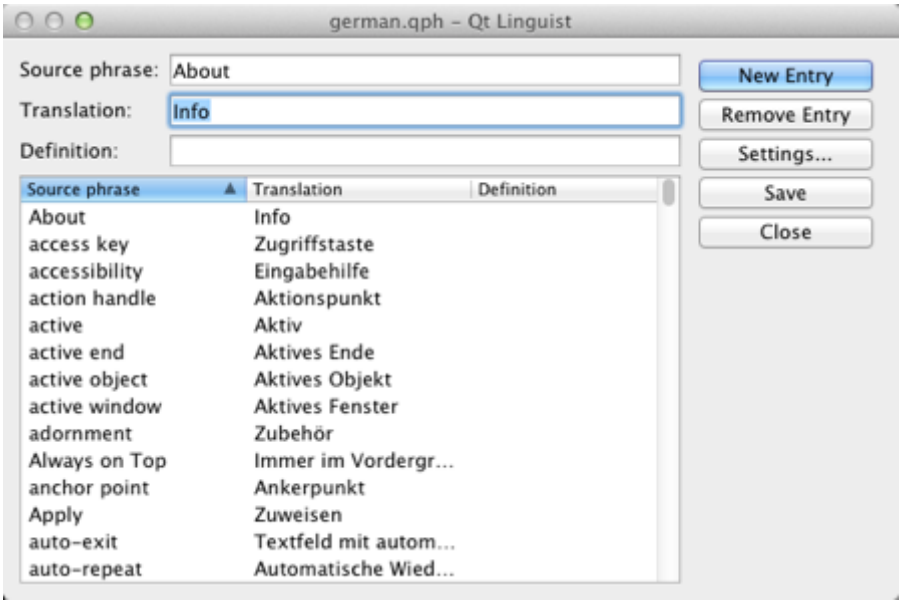


Use the batch translation feature of *Qt Linguist* to automatically translate source texts that are also in a phrase book. To configure which phrase books to use in what order during the batch translation process, select **Edit** > **Batch Translation**. You can specify whether only entries with no current translation should be considered, and whether batch translated entries should be marked as **Accepted**.

Creating and Editing Phrase Books

Phrase book files are human-readable XML files containing standard phrases and their translations. These files are

To create a new phrase book, select **Phrases** > **New Phrase Book**.



To open a phrase book, select **Phrases** > **Open Phrase Book**, and then select the Qt phrase book file (.qph) to open.

To view and change open phrase books, select **Phrases** > **Edit Phrase Book**.

To add a new phrase, select **New Entry** (or press **Alt+N**) and type in a new source phrase, the translation, and an optinal definition. This is useful to distinguish different translations of the same source phrase.

To add the translation you are working on to the current phrase book, select **Phrases** > **Add to Phrase Book** or press **Ctrl+T**. If multiple phrase books are loaded, you have to select one.

If you detect an error in a phrase book entry that is shown in the **Phrases and Guesses** view, you can also edit it in place by right clicking on the entry, and selecting **Edit**. After fixing the error press **Return** to leave the editing mode.

To delete a phrase, select it in the **Source phrase** list, and then select **Remove Entry**.

To print an open phrase book, select **Phrases** > **Print Phrase Book**.

Validating Translations

Qt Linguist provides the following validation tests for translations:

- *Accelerator validation* detects translated phrases that do not have an ampersand when the source phrase does and vice versa.
- *Punctuation validation* detects differences in the terminating punctuation between source and translated phrases when this may be significant. For example, warns if the source phrase ends with an ellipsis, exclamation mark or question mark, and the translated phrase does not, and vice versa.
- *Phrases validation* detects source phrases that are also in the phrase book but whose translation differs from that given in the phrase book.
- *Place marker validation* detects whether the same variables (like ,) are used both in the source text and in the translation.%1%2

To switch validation tests on or off, select **Validation** or use the toolbar buttons.

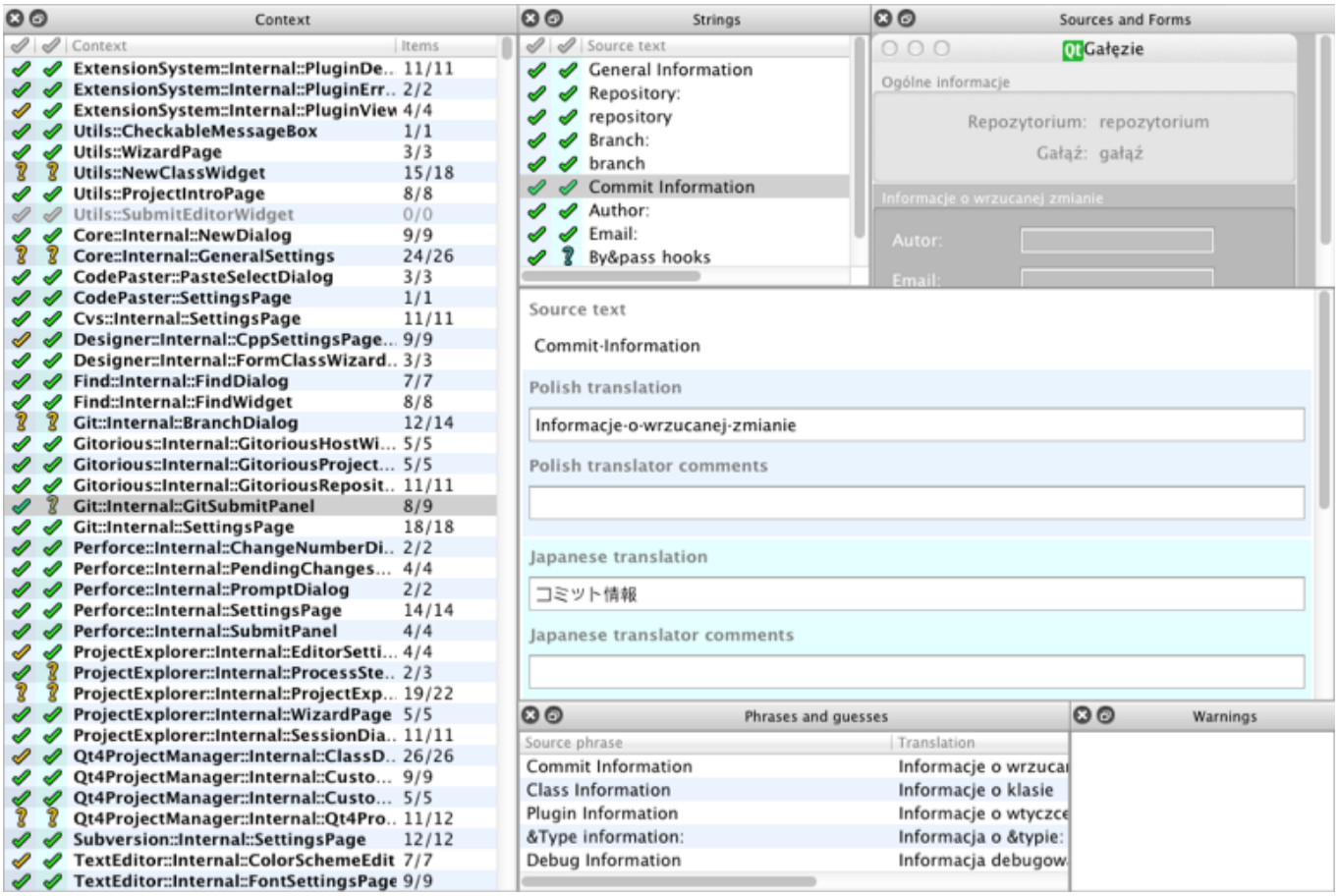
Not accepted strings that fail validation tests are marked with the **Validation Failures** icon in the **Strings** view. Accepted strings are marked with **Accepted/Warnings**.

If the translation you enter for the current string fails any of the active validation tests, the failures are listed in the **Warnings** view. The first of these failure messages is also shown in the status bar at the bottom of the main window.

> **Note:** Only results of *active* validation tests are reported.

Translating Multiple Languages Simultaneously

You can load and edit multiple translation files simultaneously. The following screen shot displays *Polish* and *Japanese* translation files loaded.



The translation area has text editing areas for both Polish and Japanese, and these are color-coded for easier separation. The **Context** view and the **Strings** view both have two status columns instead of one, color-coded with the same colors. The left-most column is related to the top-most language area (Polish above) in the translation area, and the right-most column is related to the bottom language area.

The **Items** column in the **Context** view combines the values for both languages. If the number of translatable strings does not match the number of accepted strings, either or both languages have strings that need to be translated or accepted. The **Strings** view shows the translation acceptance state of each string for each language.

‹ Qt Linguist Manual: Release Manager

Qt Linguist Manual: Developers ›

Qt DOCUMENTATION

The Qt Company

Contact Us

Company

About Us
Investors
Newsroom
Careers
Office Locations

Licensing

Terms & Conditions
Open Source
FAQ

Support

Support Services
Professional Services
Partners
Training

For Customers

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In