translations

v1.0 2013/07/16

a simple translator

Clemens Niederberger

https://github.com/cgnieder/translations/
contact@mychemistry.eu

English documentation

Contents

1 Motivation

This package provides means for package authors to have an easy interface for internationalization of their packages. The functionality of this package is in many parts also covered by the package translator (part of the beamer[1] bundle). Internationalization is also possible with babel[2] and it's `\addto\captions<language>` mechanism or KOMA-Script's `\providecaptionname`. However, I believe that translations is more flexible than all of these. Unlike translator it detects the used (babel or polyglossia[3]) language itself and provides expandable retrieving of the translated key. translations also provides support for language dialects which means package authors can for example distinguish between British, Australian, Canadian and US English.

The first draft of the package was written since I missed an expandable version of translator's `\translate` command. Once I had the package available I began using it in various of my other packages so it got extended to the needs I faced there.

[1] on CTAN: beamer [2] on CTAN: babel [3] on CTAN: polyglossia

2 License and Requirements

TRANSLATIONS is placed under the terms of the LaTeX Project Public License, version 1.3 or later (`http://www.latex-project.org/lppl.txt`). It has the status "maintained."

TRANSLATIONS requires the packages etoolbox[4] and scrlfile (part of the KOMA-Script bundle[5]).

3 Usage

3.1 Background

The TRANSLATIONS package enables the author of a package or a class (or a document) to declare translations in different languages of key words and fetch these translations in the document depending on the active language as set by babel or polxglossia.[6] Since TRANSLATIONS checks which language is active it is generally not necessary to specify the language for which a translation should be fetched manually.

TRANSLATIONS knows of three types of languages: main languages (see table 2 on page 9), language aliases (see table 3 on page 9) and language dialects (see table 4 on page 10). For the commands declaring or fetching a translation base languages and language aliases are equivalent. Dialects are similar to aliases but there are a few important differences.

Figure 1 shows what happens if TRANSLATIONS is asked to fetch a translation for a given key.

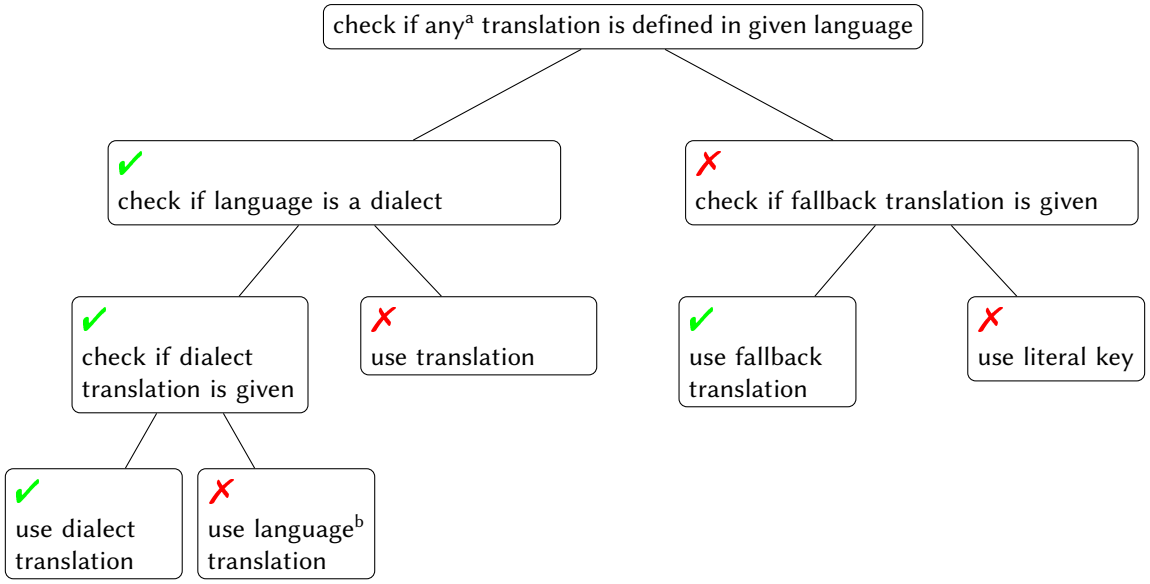

FIGURE 1: Schematic representation of TRANSLATIONS' translating mechansim. Notes: [a] except for a possible fallback translation. [b] i.e., the base language of the dialect.

What happens if you declare a translation? There are three cases:

1. You declare a translation for a base language: this is the normal case where an internal macro is defined which can be fetched by the `\GetTranslation` command (see section 3.2).

[4] on CTAN: etoolbox [5] on CTAN: koma-script [6] on CTAN: polxglossia

2. You declare a translation for a language alias: this is the very same as the first case since the same internal macro is defined.

3. You declare a translation for a dialect: this is two-fold. Either a translation for the base language exists so only the translation for the dialect is saved. If the translation for the base language does not exist it is defined to be the same as the one for the dialect.

3.2 Available Commands

Below the commands provided by TRANSLATIONS are explained. The symbol ⋆ means that the command is expandable, ⋆ means that it isn't.

⋆ `\DeclareLanguage{<lang>}`
Declare a language that can be used by TRANSLATIONS. If the language already exists it will be silently redefined. This command can only be used in the preamble.

⋆ `\DeclareLanguageAlias{<lang2>}{<lang1>}`
Declares `<lang2>` to be an alias of `<lang1>`. If `<lang1>` doesn't exist yet a warning will be raised and it will be defined. This command can only be used in the preamble.

⋆ `\DeclareLanguageDialect{<dialect>}{<lang>}`
Declares `<dialect>` to be a dialect of language `<lang>`. If a translation for `<dialect>` is provided it is used by the translation macros. If there is none the corresponding translation for `<lang>` is used instead.

⋆ `\NewTranslation{<lang>}{<key>}{<translation>}`
Defines a translation of key `<key>` for the language `<lang>`. An error will be raised if a translation of `<key>` already exists. This command can only be used in the preamble.

⋆ `\RenewTranslation{<lang>}{<key>}{<translation>}`
Redefines a translation of key `<key>` for the language `<lang>`. An error will be raised if no translation of `<key>` exists. This command can only be used in the preamble.

⋆ `\DeclareTranslation{<lang>}{<key>}{<translation>}`
Defines a translation of key `<key>` for the language `<lang>`. No error will be raised if a translation of `<key>` already exists. This command can only be used in the preamble.

⋆ `\DeclareTranslationFallback{<key>}{<fallback>}`
Defines a fallback translation for key `<key>` that is used in case no translation of `<key>` for the currently active language has been provided. No error will be raised if a fallback for `<key>` already exists. This command can only be used in the preamble.

⋆ `\GetTranslationFor{<lang>}{<key>}`
Fetches and prints the translation of `<key>` for the language `<lang>`. This command is expandable.

⋆ `\GetTranslation{<key>}`
Fetches and prints the translation of `<key>` for the currently active language (as for example set by babel). This command is expandable.

★ \GetTranslationForWarn{<lang>}{<key>}

Introduced in version 1.0

Fetches and prints the translation of <key> for the language <lang>. Issues a warning if no translation is available.

★ \GetTranslationWarn{<key>}

Introduced in version 1.0

Fetches and prints the translation of <key> for the currently active language (as for example set by babel). Issues a warning if no translation is available.

★ \SaveTranslationFor{<cmd>}{<lang>}{<key>}

Fetches and saves the translation of <key> for the language <lang> in the macro <cmd>.

★ \SaveTranslation{<cmd>}{<key>}

Fetches and saves the translation of <key> for the currently active language (as for example set by babel) in the macro <cmd>.

★ \LoadDictionary{<name>}

Loads a file named <name>-<lang>.trsl where <lang> corresponds to the lowercase name of the current language as defined with \DeclareLanguage. This file should contain the translations for the specified language.

★ \LoadDictionaryFor{<lang>}{<name>}

Loads a file named <name>-<lang>.trsl.

★ \DeclareDictTranslation{<key>}{<translation>}

This command is to be used in a dictionary file and picks up the language of that file, see section 3.5 for an example.

★ \ProvideDictionaryFor{<lang>}{<name>}[<date>]

Needs to be in a dictionary file. This command tells TRANSLATIONS that the file indeed is a dictionary and also sets the language for the dictionary which is used by \DeclareDictTranslation.

★ \PrintDictionaryFor{<lang>}{<name>}{<pre>}{<mid>}{<post>}

Introduced in version 1.0

Prints all entries of dictionary <name> in language <lang> in the order the entries have been declared. For every entry the code

```
<pre><key><mid><translation><post>
```

is printed. The dictionary must have been loaded of course.

3.3 A Small Example

This section demonstrates with two short examples how the macros are used. The first example covers the basics: dlaring of translations and then retrieving and typesetting them.

```
% in the preamble:
% \DeclareTranslation{English}{Kueche}{kitchen}
% \DeclareTranslation{German}{Kueche}{K\"uche}
% \DeclareTranslation{Spanish}{Kueche}{cocina}
% \DeclareTranslation{French}{Kueche}{cuisine}

\GetTranslation{Kueche}
\SaveTranslation\kitchen{Kueche}
\SaveTranslationFor\cuisine{french}{Kueche}

\selectlanguage{ngerman}
\GetTranslation{Kueche} \kitchen\ \GetTranslationFor{spanish}{Kueche}
\cuisine
```

kitchen
Küche kitchen cocina cuisine

The next example demonstrates the use of dialects and how they fall back to the translation for the main language if no extra translation was declared:

```
% in the preamble:
% \DeclareTranslation{English}{farbe}{color}
% \DeclareTranslation{British}{farbe}{colour}

\GetTranslationFor{English}{farbe} \\
\GetTranslationFor{British}{farbe} \\
\GetTranslationFor{American}{farbe}
```

color
colour
color

3.4 Usage in Packages

3.4.1 Basic Structure

A typical usage in a package would look as follows:

```
\RequirePackage{translations}
\DeclareTranslationFallback{mypackage-title}{Nice Title}
\DeclareTranslation{English}{mypackage-title}{Nice Title}
\DeclareTranslation{French}{mypackage-title}{Beau Titre}
\DeclareTranslation{German}{mypackage-title}{Sch\"{o}ner Titel}
...
\def\mypackage@title{\GetTranslation{mypackage-title}}
```

That is, a package defines some unique key for an expression and at least defines a fallback translation. Additionally translations for as many languages as the author wants are defined. A user then may add `\DeclareTranslation{<language>}{<translation>}` if they find their translation missing.

3.4.2 The 'fallback' language

If a user has neither loaded babel nor polyglossia TRANSLATIONS will use English as language and translate to English if the translation was provided. If the user *has* loaded one of the language packages but has chosen a language for which no translation is defined the language 'fallback' will be used, i.e., the translation provided with `\DeclareTranslationFallback`. If no fallback translation is provided either the translation will expand to the literal string.

The following three examples should make this concept clear:

```
\documentclass{article}
\DeclareTranslation{German}{foo-literal}{bar}
\begin{document}
\GetTranslation{foo-literal} => `foo-literal'
\end{document}
```

```
\documentclass{article}
\DeclareTranslationFallback{foo-literal}{foo}
\DeclareTranslation{German}{foo-literal}{bar}
\begin{document}
\GetTranslation{foo-literal} => `foo'
\end{document}
```

```
\documentclass{article}
\usepackage[ngerman]{babel}
\DeclareTranslation{German}{foo-literal}{bar}
\begin{document}
\GetTranslation{foo-literal} => `bar'
\end{document}
```

3.5 Dictionaries

3.5.1 Own Dictionaries

A typical dictionary file should look as follows:

```
% this is file housing-german.trsl
\ProvideDictionaryFor{German}{housing}[<version info>]
\DeclareDictTranslation{kitchen (housing)}{K\"uche}
\DeclareDictTranslation{bathroom (housing)}{Bad}
\DeclareDictTranslation{living room (housing)}{Wohnzimmer}
\DeclareDictTranslation{bedroom (housing)}{Schlafzimmer}
...
\endinput
```

The usage is similar to the one in a package: unique keys are given translations, this time for the language the dictionary file is declared for only.

3.5.2 TRANSLATIONS' Basic Dictionaries

TRANSLATIONS already provides a basic dictionary for the languages English, French, German and Spanish. This dictionary is loaded automatically if the document language is one of these four. If you'd like to contribute and add the basic dictionary in your language this is more than welcome and highly appreciated! The easiest way to do this would be to copy one of the existing files `translations-basic-dictionary-<lang>.trsl` and modify the file accordingly. You can then send me the file via email and I'll add it to TRANSLATIONS.

Table 1 lists all words provided by the basic dictionary for German.

Table 1: All entries of TRANSLATIONS' basic dictionary in German.

key	translation
`Abstract`	Zusammenfassung
`Addresses`	Adressen
`addresses`	Adressen
`Address`	Adresse
`address`	Adresse
`and`	und
`Appendix`	Anhang
`Authors`	Autoren
`authors`	Autoren
`Author`	Autor
`author`	Autor
`Bibliography`	Literaturverzeichnis
`cc`	Verteiler
`Chapters`	Kapitel
`chapters`	Kapitel
`Chapter`	Kapitel
`chapter`	Kapitel
`Conclusion`	Zusammenfassung
`conclusion`	Zusammenfassung
`Contents`	Inhaltsverzeichnis
`Continuation`	Fortsetzung
`continuation`	Fortsetzung
`cont`	Forts
`encl (plural)`	Anlagen
`encl (singular)`	Anlage
`encl`	Anlage(n)
`Figures`	Abbildungen
`figures`	Abbildungen
`Figure`	Abbildung

continues

key	translation
`figure`	Abbildung
`From`	Von
`from`	von
`Glossary`	Glossar
`Index`	Index
`Introduction`	Einleitung
`introduction`	Einleitung
`List of Figures and Tables`	Abbildungs- und Tabellenverzeichnis
`List of Figures`	Abbildungsverzeichnis
`List of Tables`	Tabellenverzeichnis
`or`	oder
`Outline`	Gliederung
`Overview`	Übersicht
`Pages`	Seiten
`pages`	Seiten
`Page`	Seite
`page`	Seite
`Paragraphs`	Absätze
`paragraphs`	Absätze
`Paragraph`	Absatz
`paragraph`	Absatz
`Parts`	Teile
`parts`	Teile
`Part`	Teil
`part`	Teil
`Preface`	Vorwort
`Proofs`	Beweise
`proofs`	Beweise
`Proof`	Beweis
`proof`	Beweis
`References`	Literatur
`Related work`	Verwandte Arbeiten
`Related Work`	Verwandte Arbeiten
`Sections`	Abschnitte
`sections`	Abschnitte
`Section`	Abschnitt
`section`	Abschnitt
`See also`	Siehe auch
`see also`	siehe auch
`See`	Siehe
`see`	siehe
`Sketch of Proofs`	Beweisskizzen
`Sketch of proofs`	Beweisskizzen

continues

key	translation
`Sketch of Proof`	Beweisskizze
`Sketch of proof`	Beweisskizze
`Subsections`	Unterabschnitte
`subsections`	Unterabschnitte
`Subsection`	Unterabschnitt
`subsection`	Unterabschnitt
`Summary`	Zusammenfassung
`Tables`	Tabellen
`tables`	Tabellen
`Table`	Tabelle
`table`	Tabelle
`To`	An
`to`	an

4 Defined Languages

Quite a number of languages already are defined, either directly or via an alias. So, before you define a language you should take a look at the tables below if the language doesn't already exist. Table 2 lists all base languages, "fallback" being a dummy language used for fallback translations.

Table 2: Base languages defines by translations.

fallback	albanian	bulgarian	catalan
croatian	czech	danish	dutch
english	finnish	french	german
greek	hebrew	hungarian	icelandic
italian	norwegian	polish	portuges
romanian	russian	serbocroatian	slovak
slovenian	spanish	swedish	turkish
ukrainian	canadien	american	australian
british	canadian	austrian	naustrian
magyar	brazil	swissgerman	

To every one of these languages at least one alias exists, the uppercase variant. This is due to the fact that it is common to write language names uppercased. All defined aliases are listed in table 3.

Table 3: All language aliases defined by translations.

alias	language	alias	language
Fallback	fallback	Albanian	albanian
Bulgarian	bulgarian	Catalan	catalan

continues

alias	language	alias	language
Croatian	croatian	Czech	czech
Danish	danish	Dutch	dutch
Finnish	finnish	francais	french
Francais	francais	Canadien	canadien
French	french	American	american
Australian	australian	British	british
Canadian	canadian	English	english
UKenglish	british	USenglish	american
Austrian	austrian	German	german
germanb	german	ngerman	german
Greek	greek	polutonikogreek	greek
Hebrew	hebrew	Hungarian	hungarian
Magyar	magyar	Icelandic	icelandic
Italian	italian	norsk	norwegian
Norsk	norsk	Norwegian	norwegian
nynorsk	norwegian	Nynorsk	nynorsk
Polish	polish	Brazil	brazil
brazilian	brazil	Brazilian	brazilian
Portuges	portuges	portuguese	portuges
Portuguese	portuguese	Romanian	romanian
Russian	russian	Serbocroatian	serbocroatian
Slovak	slovak	Slovenian	slovenian
Spanish	spanish	Swedish	swedish
Swiss	swissgerman	Swissgerman	swissgerman
Turkish	turkish	Ukrainian	ukrainian

TRANSLATIONS also defines a few dialects. They are listed in table 4.

Table 4: All dialects defined by TRANSLATIONS.

dialect	language	dialect	language
canadien	french	american	english
australian	english	british	english
canadian	english	austrian	german
naustrian	austrian	magyar	hungarian
brazil	portuges	swissgerman	german

These languages should cover all languages which are currently covered by babel and polyglossia.

5 Implementation

In the following code the lines 1–30 have been omitted. They only repeat the license statement which has already been mentioned in section 2.

```
\def\@trnslt@date{2013/07/16}
\def\@trnslt@version{v1.0}

\ProvidesPackage{translations}[\@trnslt@date\space \@trnslt@version\space a simple
  translator]
\RequirePackage{etoolbox,scrlfile}

% ------------------------------------------------------------------------
% message handling
\def\@trnslt@error@message{%
  For details have a look at the `translations' manual.}

\def\@trnslt@create@message#1{%
  \ifstrequal{#1}{Error}
    {%
      \lowercase{\csdef{@trnslt@#1}}##1{%
        \csuse{Package#1}{translations}{##1}{\@trnslt@error@message}}%
    }{%
      \lowercase{\csdef{@trnslt@#1}}##1{%
        \csuse{Package#1}{translations}{##1}}%
    }}
\@trnslt@create@message{Error}
\@trnslt@create@message{Warning}
\@trnslt@create@message{WarningNoLine}
\@trnslt@create@message{Info}

\def\@trnslt@err@unknown@lang#1{%
  \@trnslt@error{Unknown language `#1'}}

\def\@trnslt@warn@unknown@lang#1{%
  \@trnslt@warning{Unknown language `#1'}}

\def\@trnslt@err@already@defined#1#2{%
  \@trnslt@error{The #2 translation for `#1' is already defined.}}

\def\@trnslt@err@not@defined#1#2{%
  \@trnslt@error{The \@trnslt@language{#2} translation for `#1' is not defined yet.}}

% ------------------------------------------------------------------------
% check if babel or polyglossia is used
\AtEndPreamble{
  \@ifpackageloaded{babel}{}{
    \@ifpackageloaded{polyglossia}{}
      {\@trnslt@warning{No language package found. I am going to use `english'
        as default language.}}
  }
  \ifdef\languagename{}
    {\def\languagename{english}}
  \def\@trnslt@current@language{\languagename}
  \ifdef\bbl@afterfi{}
    {\long\def\bbl@afterfi#1\fi{\fi#1}}
}
```

```
82
83  % ---------------------------------------------------------------------------
84  % book keeping: the following macros will be used as `etoolbox' lists that
85  % keep record of defined languages, dialects and aliases
86  \def\@trnslt@languages{}% all languages
87  \def\@trnslt@aliases@pair{}% all aliases and their base
88  \def\@trnslt@aliases@single{}% all aliases
89  \def\@trnslt@dialects@pair{}% all dialects and their base
90  \def\@trnslt@dialects@single{}% all dialects
91
92  % ---------------------------------------------------------------------------
93  % define \@trnslt@if@<name> conditionals that don't leave the checked macro as
94  % \relax behind and check for \@trnslt@<name>@#1.  These conditionals should
95  % also be expandable in an \edef-like context:
96  \def\@trnslt@newif#1{%
97    \csdef{@trnslt@if@#1}##1{%
98      \expandafter\expandafter\expandafter\expandafter
99      \expandafter\expandafter\expandafter
100     \@firstofone
101     {\expandafter\expandafter\expandafter}%
102     \ifcsname @trnslt@#1@##1\endcsname
103       \expandafter\@firstoftwo
104     \else
105       \expandafter\@secondoftwo
106     \fi
107   }%
108 }
109
110 % ---------------------------------------------------------------------------
111 % \DeclareLanguage
112 % #1: language
113 \newrobustcmd*\DeclareLanguage[1]{%
114   \@trnslt@declare@language{#1}}
115 \@onlypreamble\DeclareLanguage
116
117 \def\@trnslt@declare@language#1{%
118   \@trnslt@if@language{#1}
119     {}{%
120       \csdef{@trnslt@language@#1}{#1}%
121       \listeadd\@trnslt@languages{#1}%
122     }%
123 }
124
125 \def\@trnslt@language#1{%
126   \csuse{@trnslt@language@#1}}
127
128 \@trnslt@newif{language}
129
130 % ---------------------------------------------------------------------------
131 % \DeclareLanguageDialect
132 % #1: dialect
133 % #2: language
```

```
\newrobustcmd*\DeclareLanguageDialect[2]{%
  \@trnslt@declare@languagedialect{#1}{#2}}
\@onlypreamble\DeclareLanguageDialect

\def\@trnslt@declare@languagedialect#1#2{%
  \@trnslt@if@language{#2}
    {}{%
      \@trnslt@warn@unknown@lang{#2}%
      \@trnslt@declare@language{#2}%
    }%
  \@trnslt@if@dialect{#1}
    {% => ist schon als dialect definiert => irgendwelche weiteren checks?
    }
    {%
      \@trnslt@if@alias{#2}
        {%
          \csedef{@trnslt@dialect@#1}{{\@trnslt@alias{#2}}{#1}}%
          \@trnslt@declare@language{#1}%
          \listeadd\@trnslt@dialects@single{#1}%
          \listeadd\@trnslt@dialects@pair{{#1}{\@trnslt@alias{#2}}}%
        }
        {%
          \csdef{@trnslt@dialect@#1}{{#2}{#1}}%
          \@trnslt@declare@language{#1}%
          \listeadd\@trnslt@dialects@single{#1}%
          \listeadd\@trnslt@dialects@pair{{#1}{#2}}%
        }%
    }%
}

\def\@trnslt@dialect#1{%
  \csuse{@trnslt@dialect@#1}}

% this macros fetches the base language for a given dialect, expandably:
\def\@trnslt@dialect@of#1{%
  \@trnslt@if@dialect{#1}
    {%
      \expandafter\expandafter\expandafter
      \@firstoftwo
      \csname @trnslt@dialect@#1\endcsname
    }{}%
}

\@trnslt@newif{dialect}

% ------------------------------------------------------------------------------
% \DeclareLanguageAlias
% #1: alias
% #2: language
\newrobustcmd*\DeclareLanguageAlias[2]{%
  \@trnslt@declare@languagealias{#1}{#2}}
\@onlypreamble\DeclareLanguageAlias
```

```
186
187 \def\@trnslt@declare@languagealias#1#2{%
188   \@trnslt@if@language{#2}
189     {}{%
190       \@trnslt@warn@unknown@lang{#2}%
191       \@trnslt@declare@language{#2}%
192     }%
193   \csletcs{@trnslt@language@#1}{@trnslt@language@#2}%
194   \@trnslt@if@dialect{#2}
195     {\csletcs{@trnslt@dialect@#1}{@trnslt@dialect@#2}}
196     {}%
197   \ifinlist{#1}\@trnslt@aliases@single
198     {}{%
199       \csdef{@trnslt@alias@#1}{#2}%
200       \listeadd\@trnslt@aliases@pair{{#1}{#2}}%
201       \listeadd\@trnslt@aliases@single{#1}%
202     }%
203 }
204
205 \def\@trnslt@alias#1{%
206   \csuse{@trnslt@alias@#1}}
207
208 \@trnslt@newif{alias}
209
210 % ------------------------------------------------------------------------
211 % dummy language: `fallback':
212 \DeclareLanguage{fallback}
213 \DeclareLanguageAlias{Fallback}{fallback}
214
215 % ------------------------------------------------------------------------
216 % \DeclareTranslation, \NewTranslation and \RenewTranslation
217 % #1: language
218 % #2: word
219 % #3: replacement
220 \newrobustcmd*\DeclareTranslation[3]{%
221   \@trnslt@declare@translation{#2}{#1}{#3}}
222 \@onlypreamble\DeclareTranslation
223
224 \newrobustcmd*\DeclareTranslationFallback[2]{%
225   \@trnslt@declare@translation{#1}{fallback}{#2}}
226 \@onlypreamble\DeclareTranslationFallback
227
228 \newrobustcmd*\NewTranslation[3]{%
229   \@trnslt@new@translation{#2}{#1}{#3}}
230 \@onlypreamble\NewTranslation
231
232 \newrobustcmd*\RenewTranslation[3]{%
233   \@trnslt@renew@translation{#2}{#1}{#3}}
234 \@onlypreamble\RenewTranslation
235
236 % #1: word
237 % #2: language
```

```
% #3: replacement
\def\@trnslt@declare@translation#1#2#3{%
  \@trnslt@if@language{#2}
    {%
      \@trnslt@if@dialect{#2}
        {%
          \csdef{@trnslt@word@#1@\@trnslt@dialect{#2}}{#3}%
          \@trnslt@if@word\@trnslt@dialect@of{#1}{#2}
            {}
            {\csdef{@trnslt@word@#1@\@trnslt@dialect@of{#2}}{#3}}%
        }
        {\csdef{@trnslt@word@#1@\@trnslt@language{#2}}{#3}}%
      % save the <word> as <word>:
      \csdef{@trnslt@word@#1@literal}{#1}%
    }
    {\@trnslt@err@unknown@lang{#2}}%
}

\def\@trnslt@new@translation#1#2#3{%
  \@trnslt@if@translation{#1}{#2}
    {\@trnslt@err@already@defined{#1}{#2}}
    {\@trnslt@declare@translation{#1}{#2}{#3}}}

\def\@trnslt@renew@translation#1#2#3{%
  \@trnslt@if@translation{#1}{#2}
    {\@trnslt@declare@translation{#1}{#2}{#3}}
    {\@trnslt@err@not@defined{#1}{#2}}}

% ------------------------------------------------------------------------
% now let's go through some trouble to check if a translation exists:
\def\@trnslt@if@word#1#2#3{%
  \expandafter\expandafter\expandafter\expandafter
  \expandafter\expandafter\expandafter
  \@firstofone
  {\expandafter\expandafter\expandafter}%
  \ifcsname @trnslt@word@#2@#1{#3}\endcsname
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
}%

\def\@trnslt@if@translation#1#2{%
  \@trnslt@if@word\@trnslt@language{#1}{#2}
    {\expandafter\@firstoftwo}
    {%
      \@trnslt@if@dialect{#2}
        {%
          \@trnslt@if@word\@trnslt@dialect{#1}{#2}
            {\expandafter\@firstoftwo}
            {%
              \@trnslt@if@word\@trnslt@dialect@of{#1}{#2}
```

```
                {\expandafter\@firstoftwo}
                {\expandafter\@secondoftwo}%
            }
        }
        {\expandafter\@secondoftwo}%
    }%
}

% ---------------------------------------------------------------------------
% \GetTranslationFor and \GetTranslation
% these need to be expandable!
% #1: language
% #2: word
\newcommand*\GetTranslationFor[2]{%
  \@trnslt@checkandget@translation@for{#2}{#1}}

\newcommand*\GetTranslation[1]{%
  \@trnslt@checkandget@translation@for{#1}{\@trnslt@current@language}}

% unexpandable version of the commands that raise a warning if no translation
% is available:
\newcommand*\GetTranslationForWarn[2]{%
  \@trnslt@getandwarn@translation@for{#2}{#1}}

\newcommand*\GetTranslationWarn[1]{%
  \@trnslt@getandwarn@translation@for{#1}{\@trnslt@current@language}}

% #1: word #2: language
\def\@trnslt@get@translation@for#1#2{%
  \@trnslt@if@dialect{#2}
    {%
      \ifcsdef{@trnslt@word@#1@\@trnslt@dialect{#2}}
        {\csuse{@trnslt@word@#1@\@trnslt@dialect{#2}}}
        {\csuse{@trnslt@word@#1@\@trnslt@dialect@of{#2}}}%
    }
    {\csuse{@trnslt@word@#1@\@trnslt@language{#2}}}%
}

\def\@trnslt@checkandget@translation@for#1#2{%
  \@trnslt@if@translation{#1}{#2}
    {\@trnslt@get@translation@for{#1}{#2}}
    {%
      \@trnslt@if@translation{#1}{fallback}
        {\csuse{@trnslt@word@#1@fallback}}
        {\csuse{@trnslt@word@#1@literal}}%
    }%
}

% this is not expandable!
\protected\def\@trnslt@getandwarn@translation@for#1#2{%
  \@trnslt@if@translation{#1}{#2}
    {\@trnslt@get@translation@for{#1}{#2}}
```

```
    {%
      \@trnslt@warning{Translation for `#1' in #2 unknown. You may try to use
        \string\DeclareTranslation{#2}{#1}{ ... } in your preamble.}%
      \@trnslt@if@translation{#1}{fallback}
        {%
          \@trnslt@info{Using fallback translation for `#1'}%
          \csuse{@trnslt@word@#1@fallback}
        }
        {\csuse{@trnslt@word@#1@literal}}%
    }%
}

% ---------------------------------------------------------------------------
% \SaveTranslationFor and \SaveTranslation
\newrobustcmd*\SaveTranslationFor[3]{%
  \@trnslt@save@translation@for{#1}{#3}{#2}}

\newrobustcmd*\SaveTranslation[2]{%
  \@trnslt@save@translation@for{#1}{#2}{\@trnslt@current@language}}

\def\@trnslt@save@translation@for#1#2#3{%
  \edef#1{%
    \@trnslt@if@translation{#2}{#3}
      {\csuse{@trnslt@word@#2@\@trnslt@language{#3}}}
      {}%
  }}

% ---------------------------------------------------------------------------
% \LoadDictionary and \LoadDictionaryFor
\newrobustcmd*\LoadDictionary[1]{%
  \@trnslt@load@dictionary@for{#1}{\@trnslt@current@language}}
\@onlypreamble\LoadDictionary

\newrobustcmd*\LoadDictionaryFor[2]{%
  \@trnslt@load@dictionary@for{#2}{#1}}
\@onlypreamble\LoadDictionaryFor

% #1: name
% #2: lang
\def\@trnslt@load@dictionary@for#1#2{%
  \AtBeginDocument{%
    \InputIfFileExists{#1-\@trnslt@language{#2}.trsl}
      {\@trnslt@check@dictionary{#1}{#2}}
      {\@trnslt@warning{dictionary file `#1-\@trnslt@language{#2}.trsl' not
          found.}}%
  }}

\def\@trnslt@check@dictionary#1#2{%
  \AfterFile{#1-\@trnslt@language{#2}.trsl}
    {%
      \ifcsdef{@trnslt@dictionary@#1@\@trnslt@language{#2}}
        {\@trnslt@info{loading dictionary `#1' for `#2'.}}
```

```
        {%
          \@trnslt@warning{file `#1-\@trnslt@language{#2}.trsl' does not
            appear to be a dictionary}%
        }%
    }%
}

\def\@trnslt@load@dictionary@silent@for#1#2{%
  \AtBeginDocument{\InputIfFileExists{#1-\@trnslt@language{#2}.trsl}{}{}}}

\newrobustcmd*\ProvideDictionaryFor[2]{%
  \@trnslt@provide@dictionary@for{#1}{#2}}
\@onlypreamble\ProvideDictionaryFor

\def\@trnslt@provide@dictionary@for#1#2{%
  \def\@trnslt@dictionary@name{#2}%
  \edef\@trnslt@dictionary@lang{\@trnslt@language{#1}}%
  % this macro can be used to check if we have a dictionary and will also be
  % used as a list for the dictionary entries:
  \csdef{@trnslt@dictionary@\@trnslt@dictionary@name @\@trnslt@dictionary@lang}{}%
  \@ifnextchar[
    {\@trnslt@provide@dictionary@version}
    {
      \ProvidesFile
        {#2-\@trnslt@dictionary@lang.trsl}%
        [(\@trnslt@dictionary@lang\space translation file `#2')]
    }%
}

\def\@trnslt@provide@dictionary@version[#1]{%
  \ProvidesFile
    {\@trnslt@dictionary@name-\@trnslt@dictionary@lang.trsl}%
    [(\@trnslt@dictionary@lang\space translation file `\@trnslt@dictionary@name') #1]}

% \@trnslt@dictionary@language
\newrobustcmd*\DeclareDictTranslation[2]{%
  \listcsadd
    {@trnslt@dictionary@\@trnslt@dictionary@name @\@trnslt@dictionary@lang}
    {{#1}{#2}}%
  \@trnslt@declare@translation{#1}{\@trnslt@dictionary@lang}{#2}%
}
\@onlypreamble\DeclareDictTranslation

% \PrintDictionaryFor
% #1: lang
% #2: name
% #3: pre
% #4: mid
% #5: post
\newcommand*\PrintDictionaryFor[5]{%
  \@trnslt@print@dictionary@for{#1}{#2}{#3}{#4}{#5}}
```

```
% #1: lang
% #2: name
% #3: pre
% #4: mid
% #5: post
\def\@trnslt@print@dictionary@for#1#2#3#4#5{%
  \forlistcsloop
    {\@trnslt@print@dictionary@entry{#3}{#4}{#5}}
    {@trnslt@dictionary@#2@\@trnslt@language{#1}}%
}

% #1: pre
% #2: mid
% #3: post
% #4: {key}{translation}
\def\@trnslt@print@dictionary@entry#1#2#3#4{%
  \@trnslt@print@dictionary@entry@aux{#1}{#2}{#3}#4}

% #1: pre
% #2: mid
% #3: post
% #4: key
% #5: translation
\def\@trnslt@print@dictionary@entry@aux#1#2#3#4#5{#1#4#2#5#3}

% ---------------------------------------------------------------------------
% predefined languages
\DeclareLanguage{albanian}
\DeclareLanguage{bulgarian}
\DeclareLanguage{catalan}
\DeclareLanguage{croatian}
\DeclareLanguage{czech}
\DeclareLanguage{danish}
\DeclareLanguage{dutch}
\DeclareLanguage{english}
\DeclareLanguage{finnish}
\DeclareLanguage{french}
\DeclareLanguage{german}
\DeclareLanguage{greek}
\DeclareLanguage{hebrew}
\DeclareLanguage{hungarian}
\DeclareLanguage{icelandic}
\DeclareLanguage{italian}
\DeclareLanguage{norwegian}
\DeclareLanguage{polish}
\DeclareLanguage{portuges}
\DeclareLanguage{romanian}
\DeclareLanguage{russian}
\DeclareLanguage{serbocroatian}
\DeclareLanguage{slovak}
\DeclareLanguage{slovenian}
\DeclareLanguage{spanish}
```

```
\DeclareLanguage{swedish}
\DeclareLanguage{turkish}
\DeclareLanguage{ukrainian}

% ------------------------------------------------------------------------
% aliases and dialects:
\DeclareLanguageAlias  {Albanian}{albanian}
\DeclareLanguageAlias  {Bulgarian}{bulgarian}
\DeclareLanguageAlias  {Catalan}{catalan}
\DeclareLanguageAlias  {Croatian}{croatian}
\DeclareLanguageAlias  {Czech}{czech}
\DeclareLanguageAlias  {Danish}{danish}
\DeclareLanguageAlias  {Dutch}{dutch}
\DeclareLanguageAlias  {Finnish}{finnish}
\DeclareLanguageAlias  {francais}{french}
\DeclareLanguageAlias  {Francais}{francais}
\DeclareLanguageDialect{canadien}{french}
\DeclareLanguageAlias  {Canadien}{canadien}
\DeclareLanguageAlias  {French}{french}
\DeclareLanguageDialect{american}{english}
\DeclareLanguageAlias  {American}{american}
\DeclareLanguageDialect{australian}{english}
\DeclareLanguageAlias  {Australian}{australian}
\DeclareLanguageDialect{british}{english}
\DeclareLanguageAlias  {British}{british}
\DeclareLanguageDialect{canadian}{english}
\DeclareLanguageAlias  {Canadian}{canadian}
\DeclareLanguageAlias  {English}{english}
\DeclareLanguageAlias  {UKenglish}{british}
\DeclareLanguageAlias  {USenglish}{american}
\DeclareLanguageDialect{austrian}{german}
\DeclareLanguageAlias  {Austrian}{austrian}
\DeclareLanguageAlias  {German}{german}
\DeclareLanguageAlias  {germanb}{german}
\DeclareLanguageDialect{naustrian}{austrian}
\DeclareLanguageAlias  {ngerman}{german}
\DeclareLanguageAlias  {Greek}{greek}
\DeclareLanguageAlias  {polutonikogreek}{greek}
\DeclareLanguageAlias  {Hebrew}{hebrew}
\DeclareLanguageAlias  {Hungarian}{hungarian}
\DeclareLanguageDialect{magyar}{hungarian}
\DeclareLanguageAlias  {Magyar}{magyar}
\DeclareLanguageAlias  {Icelandic}{icelandic}
\DeclareLanguageAlias  {Italian}{italian}
\DeclareLanguageAlias  {norsk}{norwegian}
\DeclareLanguageAlias  {Norsk}{norsk}
\DeclareLanguageAlias  {Norwegian}{norwegian}
\DeclareLanguageAlias  {nynorsk}{norwegian}
\DeclareLanguageAlias  {Nynorsk}{nynorsk}
\DeclareLanguageAlias  {Polish}{polish}
\DeclareLanguageDialect{brazil}{portuges}
\DeclareLanguageAlias  {Brazil}{brazil}
```

```
\DeclareLanguageAlias  {brazilian}{brazil}
\DeclareLanguageAlias  {Brazilian}{brazilian}
\DeclareLanguageAlias  {Portuges}{portuges}
\DeclareLanguageAlias  {portuguese}{portuges}
\DeclareLanguageAlias  {Portuguese}{portuguese}
\DeclareLanguageAlias  {Romanian}{romanian}
\DeclareLanguageAlias  {Russian}{russian}
\DeclareLanguageAlias  {Serbocroatian}{serbocroatian}
\DeclareLanguageAlias  {Slovak}{slovak}
\DeclareLanguageAlias  {Slovenian}{slovenian}
\DeclareLanguageAlias  {Spanish}{spanish}
\DeclareLanguageAlias  {Swedish}{swedish}
\DeclareLanguageDialect{swissgerman}{german}
% this maybe should be a language of it's own:
\DeclareLanguageAlias  {Swiss}{swissgerman}
\DeclareLanguageAlias  {Swissgerman}{swissgerman}
\DeclareLanguageAlias  {Turkish}{turkish}
\DeclareLanguageAlias  {Ukrainian}{ukrainian}

% ---------------------------------------------------------------------------
% load basic dictionary if available
\AtBeginDocument{%
  \@trnslt@load@dictionary@silent@for
    {translations-basic-dictionary}
    {\@trnslt@current@language}%
}

\endinput

% ---------------------------------------------------------------------------
% HISTORY:
2012/09/30 v0.2beta - first version (as part of the `exsheets' bundle)
2012/10/05 v0.2     - \LoadDictionary and \LoadDictionaryFor added and loads of
                      languages defined.
2013/03/10 v0.8     - basic dictionaries for English, German, French and Spanish
                    - new command \DeclareDictTranslation
2013/04/04 v0.8a    - bug fix in \DeclareDictTranslation
2013/04/07 v0.9     - slightly improved messages
2013/04/08 v0.9a    - changed fallback warning into info
                    - synchronized version number with `exsheets' until now but
                      won't any more
2013/06/22 v0.9b    - added Swiss
2013/06/28 v0.10    - declaring aliases of dialects now works as expected
                    - declarings dialects of an alias now correctly declares
                      the dialect to the correct base language
                    - corrected a few erroneous language declarations
2013/07/12 v0.10a   - \GetTranslation gets two-folded fallback: use
                      fallback-translation if no translation for the current
                      language has been defined; use literal string if /no/
                      language is used - this should never happen but /will/
                      happen if neither `babel' nor `polyglossia' have been
                      loaded, i.e., no language has been chosen /and/ the
```

```
                        package writer did not provide an English translation
2013/07/16 v1.0      - removed from `exsheets' bundle - `translations' should
                        be a package of it's own
                     - load basic dictionary automatically if available
                     - rudimentary check in \LoadDictionary if loaded file is a
                        dictionary
                     - new command \PrintDictionaryFor
                     - redefined conditionals; they still seemed to make
                        trouble in some cases
```

Index

Section titles are indicated **bold**, packages sans serif and commands \brown.