TRANSLATIONS

v0.10a 2013/07/12

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 distinguish between British, Australian, Canadian and US English, say.

[1] onCTAN:beamer [2] onCTAN:babel [3] onCTAN: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 etoolbox[4] package.

3 Usage

3.1 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.

[4] onCTAN:etoolbox

▷ `\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.

▶ `\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 3.4 for an example.

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 section 4 if the language doesn't already exist.

3.2 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

3.3 Usage in Packages

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 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.

4 Defined Languages

TRANSLATIONS currently has these languages defined, "fallback" being a dummy language used for fallback 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. The defined aliases are these (in parentheses the base language name is given):

> Fallback (fallback), Albanian (albanian), Bulgarian (bulgarian), Catalan (catalan), 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. The language to which the dialect belongs to is given in paretheses:

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.

```
31  \def\@trnslt@date{2013/07/12}
32  \def\@trnslt@version{v0.10a}
33
34  \ProvidesPackage{translations}[\@trnslt@date\space \@trnslt@version\space a
       simple translator]
35  \RequirePackage{etoolbox}
36
37  % - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
38  % message handling
39  \def\@trnslt@error@message{%
40    For details have a look at the ‘translations’ manual.}
41
42  \def\@trnslt@create@message#1{%
43    \ifstrequal{#1}{Error}
44      {%
45        \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}}
}

% book keeping: the following macros will be used as ‘etoolbox’ lists that
% keep record of defined languages, dialects and aliases
\def\@trnslt@languages{}
\def\@trnslt@aliases@pair{}
\def\@trnslt@aliases@single{}
\def\@trnslt@dialects@pair{}
\def\@trnslt@dialects@single{}
```

```

% --------------------------------------------------------------------------
% \DeclareLanguage and \DeclareLanguageAlias
% #1: language
\newrobustcmd*\DeclareLanguage[1]{%
  \@trnslt@declare@language{#1}}
\@onlypreamble\DeclareLanguage

\def\@trnslt@declare@language#1{%
  \@trnslt@if@language{#1}
    {}{%
      \csdef{@trnslt@language@#1}{#1}%
      \listeadd\@trnslt@languages{#1}%
    }%
}

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

\def\@trnslt@if@language#1{%
  \ifcsundef{@trnslt@language@#1}
    {\expandafter\@secondoftwo}
    {\expandafter\@firstoftwo}%
}

% #1: dialect
% #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}}

\def\@trnslt@dialect@of#1{%
  \expandafter\expandafter\expandafter
    \@trnslt@dialect@of@aux
    \csname @trnslt@dialect@#1\endcsname\@empty
}
\def\@trnslt@dialect@of@aux#1#2{\ifx\relax#1\@empty\else#1\fi}

\def\@trnslt@if@dialect#1{%
  \ifcsundef{@trnslt@dialect@#1}
    {\expandafter\@secondoftwo}
    {\expandafter\@firstoftwo}%
}

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

\def\@trnslt@declare@languagealias#1#2{%
  \@trnslt@if@language{#2}
    {}{%
      \@trnslt@warn@unknown@lang{#2}%
      \@trnslt@declare@language{#2}%
    }%
  \csletcs{@trnslt@language@#1}{@trnslt@language@#2}%
  \@trnslt@if@dialect{#2}
    {\csletcs{@trnslt@dialect@#1}{@trnslt@dialect@#2}}
    {}%
  \ifinlist{#1}\@trnslt@aliases@single
    {}{%
      \csdef{@trnslt@alias@#1}{#2}%
      \listeadd\@trnslt@aliases@pair{{#1}{#2}}%
      \listeadd\@trnslt@aliases@single{#1}%
    }%
}

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

\def\@trnslt@if@alias#1{%
  \ifcsundef{@trnslt@alias@#1}
    {\expandafter\@secondoftwo}
    {\expandafter\@firstoftwo}%
}

% dummy language: ‘fallback’:
\DeclareLanguage{fallback}
\DeclareLanguageAlias{Fallback}{fallback}

% --------------------------------------------------------------------------

% \DeclareTranslation, \NewTranslation and \RenewTranslation
% #1: language
% #2: word
% #3: replacement
\newrobustcmd*\DeclareTranslation[3]{%
  \@trnslt@declare@translation{#2}{#1}{#3}}
\@onlypreamble\DeclareTranslation

\newrobustcmd*\DeclareTranslationFallback[2]{%
  \@trnslt@declare@translation{#1}{fallback}{#2}}
\@onlypreamble\DeclareTranslationFallback

\newrobustcmd*\NewTranslation[3]{%
  \@trnslt@new@translation{#2}{#1}{#3}}
\@onlypreamble\NewTranslation

\newrobustcmd*\RenewTranslation[3]{%
  \@trnslt@renew@translation{#2}{#1}{#3}}
\@onlypreamble\RenewTranslation

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

\def\@trnslt@if@translation#1#2{%
  \ifcsundef{@trnslt@word@#1@\@trnslt@language{#2}}
    {%
      \@trnslt@if@dialect{#2}
        {%
          \ifboolexpe{
            test {\ifcsundef{@trnslt@word@#1@\@trnslt@dialect{#2}}} or
            test {\ifcsundef{@trnslt@word@#1@\@trnslt@dialect@of{#2}}}
          }
          {\expandafter\@firstoftwo}
          {\expandafter\@secondoftwo}%
        }
        {\expandafter\@secondoftwo}%
    }
    {\expandafter\@firstoftwo}%
}

\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}}}

% -------------------------------------------------------------------------
% \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}}

% #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@info{loading dictionary ‘#1’ for ‘#2’.}}
      {\@trnslt@warning{File ‘#1-\@trnslt@language{#2}.trsl’ not found.}}%
  }}

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

\def\@trnslt@provide@dictionary@for#1#2{%
  \def\@trnslt@dictionary@name{#2}%
  \def\@trnslt@dictionary@lang{#1}%
  \@ifnextchar[
    {\@trnslt@provide@dictionary@version}
    {\ProvidesFile{#2-#1.trsl}[(#1 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]{%
  \@trnslt@declare@translation{#1}{\@trnslt@dictionary@lang}{#2}}
\@onlypreamble\DeclareDictTranslation

% --------------------------------------------------------------------------
% 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}

\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}

\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
```

Index

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