

translations

v0.10b 2013/07/15

a simple translator

Clemens NIEDERBERGER

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

English documentation

Contents			
		3.3 Usage in Packages	4
		3.3.1 Basic Structure	4
		3.3.2 The ‘fallback’ language	4
		3.4 Dictionaries	5
1 Motivation	1		
2 License and Requirements	1	4 Defined Languages	5
3 Usage	2	5 Implementation	6
3.1 Available Commands	2		
3.2 A Small Example	3	Index	16

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 beamer¹ bundle). Internationalization is also possible with babel² 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 polyglossia³) 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.

2 License and Requirements

TRANSLATIONS is placed under the terms of the L^AT_EX Project Public License, version 1.3 or later (<http://www.latex-project.org/lppl.txt>). It has the status “maintained.”

TRANSLATIONS requires the etoolbox⁴ package.

¹ on CTAN: beamer ² on CTAN: babel ³ on CTAN: polyglossia ⁴ on CTAN: etoolbox

3 Usage

3.1 Available Commands

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

- \blacktriangleright **\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.
- \blacktriangleright **\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.
- \blacktriangleright **\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.
- \blacktriangleright **\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.
- \blacktriangleright **\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.
- \blacktriangleright **\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.
- \blacktriangleright **\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.
- \triangleright **\GetTranslationFor**{<lang>}{<key>}
 Fetches and prints the translation of <key> for the language <lang>. This command is expandable.
- \triangleright **\GetTranslation**{<key>}
 Fetches and prints the translation of <key> for the currently active language (as for example set by babel). This command is expandable.
- \blacktriangleright **\SaveTranslationFor**{<cmd>}{<lang>}{<key>}
 Fetches and saves the translation of <key> for the language <lang> in the macro <cmd>.
- \blacktriangleright **\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: declaring of translations and then retrieving and typesetting them.

```

1 % in the preamble:
2 % \DeclareTranslation{English}{Kueche}{kitchen}
3 % \DeclareTranslation{German}{Kueche}{K\"uche}
4 % \DeclareTranslation{Spanish}{Kueche}{cocina}
5 % \DeclareTranslation{French}{Kueche}{cuisine}
6
7 \GetTranslation{Kueche}
8 \SaveTranslation\kitchen{Kueche}
9 \SaveTranslationFor\cuisine{french}{Kueche}
10
11 \selectlanguage{ngerman}
12 \GetTranslation{Kueche} \kitchen\ \GetTranslationFor{spanish}{Kueche}
13 \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:

```

1 % in the preamble:
2 % \DeclareTranslation{English}{farbe}{color}
3 % \DeclareTranslation{British}{farbe}{colour}
4
5 \GetTranslationFor{English}{farbe} \
6 \GetTranslationFor{British}{farbe} \
7 \GetTranslationFor{American}{farbe}

```

color
colour
color

3.3 Usage in Packages

3.3.1 Basic Structure

A typical usage in a package would look as follows:

```

1 \RequirePackage{translations}
2 \DeclareTranslationFallback{mypackage-title}{Nice Title}
3 \DeclareTranslation{English}{mypackage-title}{Nice Title}
4 \DeclareTranslation{French}{mypackage-title}{Beau Titre}
5 \DeclareTranslation{German}{mypackage-title}{Sch\''{o}ner Titel}
6 ...
7 \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.

3.3.2 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 `\DeclareFallbackTranslation`. If no fallback translation is provided either the translation will expand to the literal string.

The following three examples should make this concept clear:

```

1 \documentclass{article}
2 \DeclareTranslation{German}{foo-literal}{bar}
3 \begin{document}
4 \GetTranslation{foo-literal} => `foo-literal'
5 \end{document}
```

```

1 \documentclass{article}
2 \DeclareTranslationFallback{foo-literal}{foo}
3 \DeclareTranslation{German}{foo-literal}{bar}
4 \begin{document}
5 \GetTranslation{foo-literal} => `foo'
6 \end{document}
```

```

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

```

3.4 Dictionaries

A typical dictionary file should look as follows:

```

1 % this is file housing-german.trsl
2 \ProvideDictionaryFor{German}{housing}[<version info>]
3 \DeclareDictTranslation{kitchen (housing)}{K\"uche}
4 \DeclareDictTranslation{bathroom (housing)}{Bad}
5 \DeclareDictTranslation{living room (housing)}{Wohnzimmer}
6 \DeclareDictTranslation{bedroom (housing)}{Schlafzimmer}
7 ...
8 \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

5 Implementation

(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 parentheses:

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{2013/07/15}
32 \def\@trnslt@version{v0.10b}
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{%
46         \csuse{Package#1}{translations}{##1}{\@trnslt@error@message}}%
47     }{%
48       \lowercase{\csdef{\@trnslt@#1}}##1{%
49         \csuse{Package#1}{translations}{##1}}%
50     }
51 \@trnslt@create@message{Error}
52 \@trnslt@create@message{Warning}
53 \@trnslt@create@message{WarningNoLine}
54 \@trnslt@create@message{Info}
55
```

5 Implementation

```

56 \def\@trnslt@err@unknown@lang#1{%
57   \@trnslt@error{Unknown language `#1'}}
58
59 \def\@trnslt@warn@unknown@lang#1{%
60   \@trnslt@warning{Unknown language `#1'}}
61
62 \def\@trnslt@err@already@defined#1#2{%
63   \@trnslt@error{The #2 translation for `#1' is already defined.}}
64
65 \def\@trnslt@err@not@defined#1#2{%
66   \@trnslt@error{The \@trnslt@language{#2} translation for `#1' is not defined yet.}}
67
68 % -----
69 % check if babel or polyglossia is used
70 \AtEndPreamble{
71   \ifpackageloaded{babel}{}{
72     \ifpackageloaded{polyglossia}{}
73       {\@trnslt@info{No language package found. I am going to use `english'
74         as default language.}}
75   }
76   \ifdef\language{}{
77     {\def\language{english}}
78     \def\@trnslt@current@language{\language}
79     \ifdef\bbl@afterfi{}
80       {\long\def\bbl@afterfi#1\fi{\fi#1}}
81   }
82
83   % book keeping: the following macros will be used as `etoolbox' lists that
84   % keep record of defined languages, dialects and aliases
85   \def\@trnslt@languages{}
86   \def\@trnslt@aliases@pair{}
87   \def\@trnslt@aliases@single{}
88   \def\@trnslt@dialects@pair{}
89   \def\@trnslt@dialects@single{}
90
91   % -----
92   % \DeclareLanguage and \DeclareLanguageAlias
93   % #1: language
94   \newrobustcmd*\DeclareLanguage[1]{%
95     \@trnslt@declare@language{#1}}
96   \@onlypreamble\DeclareLanguage
97
98   \def\@trnslt@declare@language#1{%
99     \@trnslt@if@language{#1}
100     {}{%
101       \csdef{\@trnslt@language@#1}{#1}%
102       \listadd\@trnslt@languages{#1}%
103     }%
104   }
105
106   \def\@trnslt@language#1{%
107     \csuse{\@trnslt@language@#1}}

```

5 Implementation

```

108
109 \def\@trnslt@if@language#1{%
110   \ifcsundef{\@trnslt@language@#1}
111     {\expandafter\@secondoftwo}
112     {\expandafter\@firstoftwo}%
113 }
114
115 % #1: dialect
116 % #2: language
117 \newrobustcmd*\DeclareLanguageDialect[2]{%
118   \@trnslt@declare@language@dialect{#1}{#2}}
119 \@onlypreamble\DeclareLanguageDialect
120
121 \def\@trnslt@declare@language@dialect#1#2{%
122   \@trnslt@if@language{#2}
123     {%
124       \@trnslt@warn@unknown@lang{#2}%
125       \@trnslt@declare@language{#2}%
126     }%
127   \@trnslt@if@dialect{#1}
128     {% => ist schon als dialect definiert => irgendwelche weiteren checks?
129     }
130     {%
131       \@trnslt@if@alias{#2}
132         {%
133           \csedef{\@trnslt@dialect@#1}{\@trnslt@alias{#2}{#1}}%
134           \@trnslt@declare@language{#1}%
135           \listeaddd\@trnslt@dialects@single{#1}%
136           \listeaddd\@trnslt@dialects@pair{{#1}{\@trnslt@alias{#2}}}%
137         }
138         {%
139           \csdef{\@trnslt@dialect@#1}{#{2}{#1}}%
140           \@trnslt@declare@language{#1}%
141           \listeaddd\@trnslt@dialects@single{#1}%
142           \listeaddd\@trnslt@dialects@pair{{#1}{#2}}%
143         }%
144       }%
145     }
146
147 \def\@trnslt@dialect#1{%
148   \csuse{\@trnslt@dialect@#1}}
149
150 \def\@trnslt@dialect@of#1{%
151   \expandafter\expandafter\expandafter
152     \@trnslt@dialect@of@aux
153     \csname \@trnslt@dialect@#1\endcsname\@empty
154 }
155 \def\@trnslt@dialect@of@aux#1#2{\ifx\relax#1\@empty\else#1\fi}
156
157 \def\@trnslt@if@dialect#1{%
158   \ifcsundef{\@trnslt@dialect@#1}
159     {\expandafter\@secondoftwo}

```


5 Implementation

```

160     {\expandafter\@firstoftwo}%
161 }
162
163 % #1: alias
164 % #2: language
165 \newrobustcmd*\DeclareLanguageAlias[2]{%
166   \@trnslt@declare@languagealias{#1}{#2}}
167 \@onlypreamble\DeclareLanguageAlias
168
169 \def\@trnslt@declare@languagealias#1#2{%
170   \@trnslt@if@language{#2}
171   {}{%
172     \@trnslt@warn@unknown@lang{#2}%
173     \@trnslt@declare@language{#2}%
174   }%
175   \csletcs{@trnslt@language@#1}{@trnslt@language@#2}%
176   \@trnslt@if@dialect{#2}
177   {\csletcs{@trnslt@dialect@#1}{@trnslt@dialect@#2}}
178   {}%
179   \ifinlist{#1}\@trnslt@aliases@single
180   {}{%
181     \csdef{@trnslt@alias@#1}{#2}%
182     \listadd\@trnslt@aliases@pair{{#1}{#2}}%
183     \listadd\@trnslt@aliases@single{#1}%
184   }%
185 }
186
187 \def\@trnslt@alias#1{%
188   \csuse{@trnslt@alias@#1}}
189
190 \def\@trnslt@if@alias#1{%
191   \ifcsundef{@trnslt@alias@#1}
192   {\expandafter\@secondoftwo}
193   {\expandafter\@firstoftwo}%
194 }
195
196 % dummy language: `fallback':
197 \DeclareLanguage{fallback}
198 \DeclareLanguageAlias{Fallback}{fallback}
199
200 % -----
201 % \DeclareTranslation, \NewTranslation and \RenewTranslation
202 % #1: language
203 % #2: word
204 % #3: replacement
205 \newrobustcmd*\DeclareTranslation[3]{%
206   \@trnslt@declare@translation{#2}{#1}{#3}}
207 \@onlypreamble\DeclareTranslation
208
209 \newrobustcmd*\DeclareTranslationFallback[2]{%
210   \@trnslt@declare@translation{#1}{fallback}{#2}}
211 \@onlypreamble\DeclareTranslationFallback

```

5 Implementation

```

212
213 \newrobustcmd*\NewTranslation[3]{%
214   \@trnslt@new@translation{#2}{#1}{#3}}
215 \@onlypreamble\NewTranslation
216
217 \newrobustcmd*\RenewTranslation[3]{%
218   \@trnslt@renew@translation{#2}{#1}{#3}}
219 \@onlypreamble\RenewTranslation
220
221 % #1: word
222 % #2: language
223 % #3: replacement
224 \def\@trnslt@declare@translation#1#2#3{%
225   \@trnslt@if@language{#2}
226   {%
227     % save the <word> as <word>:
228     \csdef{@trnslt@word@#1@literal}{#1}%
229     % check if the language is a dialect:
230     \@trnslt@if@dialect{#2}
231     {\csdef{@trnslt@word@#1@\@trnslt@dialect{#2}}{#3}}
232     {}%
233     % check if translation already exists:
234     \@trnslt@if@translation{#1}{#2}
235     {}
236     {\csdef{@trnslt@word@#1@\@trnslt@language{#2}}{#3}}%
237   }
238   {\@trnslt@err@unknown@lang{#2}}%
239 }
240
241 \def\@trnslt@if@translation#1#2{%
242   \ifcsundef{@trnslt@word@#1@\@trnslt@language{#2}}
243   {%
244     \@trnslt@if@dialect{#2}
245     {%
246       \ifboolexpe{
247         test {\ifcsundef{@trnslt@word@#1@\@trnslt@dialect{#2}}} or
248         test {\ifcsundef{@trnslt@word@#1@\@trnslt@dialect@of{#2}}}
249       }
250       {\expandafter\@firstoftwo}
251       {\expandafter\@secondoftwo}%
252     }
253     {\expandafter\@secondoftwo}%
254   }
255   {\expandafter\@firstoftwo}%
256 }
257
258 \def\@trnslt@new@translation#1#2#3{%
259   \@trnslt@if@translation{#1}{#2}
260   {\@trnslt@err@already@defined{#1}{#2}}
261   {\@trnslt@declare@translation{#1}{#2}{#3}}
262
263 \def\@trnslt@renew@translation#1#2#3{%

```

5 Implementation

```

264 \@trnslt@if@translation{#1}{#2}
265 {\@trnslt@declare@translation{#1}{#2}{#3}}
266 {\@trnslt@err@not@defined{#1}{#2}}}
267
268 % -----
269 % \GetTranslationFor and \GetTranslation
270 % these need to be expandable!
271 % #1: language
272 % #2: word
273 \newcommand*\GetTranslationFor[2]{%
274 \@trnslt@checkandget@translation@for{#2}{#1}}
275
276 \newcommand*\GetTranslation[1]{%
277 \@trnslt@checkandget@translation@for{#1}{\@trnslt@current@language}}
278
279 % #1: word #2: language
280 \def\@trnslt@get@translation@for#1#2{%
281 \@trnslt@if@dialect{#2}
282 {%
283 \ifcsdef{@trnslt@word@#1@\@trnslt@dialect{#2}}
284 {\csuse{@trnslt@word@#1@\@trnslt@dialect{#2}}}
285 {\csuse{@trnslt@word@#1@\@trnslt@dialect@of{#2}}}%
286 }
287 {\csuse{@trnslt@word@#1@\@trnslt@language{#2}}}%
288 }
289
290 \def\@trnslt@checkandget@translation@for#1#2{%
291 \@trnslt@if@translation{#1}{#2}
292 {\@trnslt@get@translation@for{#1}{#2}}
293 {%
294 \@trnslt@if@translation{#1}{fallback}
295 {\csuse{@trnslt@word@#1@fallback}}
296 {\csuse{@trnslt@word@#1@literal}}}%
297 }%
298 }
299
300 % this is not expandable!
301 \protected\def\@trnslt@getandwarn@translation@for#1#2{%
302 \@trnslt@if@translation{#1}{#2}
303 {\@trnslt@get@translation@for{#1}{#2}}
304 {%
305 \@trnslt@warning{Translation for `#1' in #2 unknown. You may try to use
306 \string\DeclareTranslation{#2}{#1}{ ... } in your preamble.}%
307 \@trnslt@if@translation{#1}{fallback}
308 {%
309 \@trnslt@info{Using fallback translation for `#1'}%
310 \csuse{@trnslt@word@#1@fallback}
311 }
312 {\csuse{@trnslt@word@#1@literal}}}%
313 }%
314 }
315

```

5 Implementation

```

316 % -----
317 % \SaveTranslationFor and \SaveTranslation
318 \newrobustcmd*\SaveTranslationFor[3]{%
319   \@trnslt@save@translation@for{#1}{#3}{#2}}
320
321 \newrobustcmd*\SaveTranslation[2]{%
322   \@trnslt@save@translation@for{#1}{#2}{\@trnslt@current@language}}
323
324 \def\@trnslt@save@translation@for#1#2#3{%
325   \edef#1{%
326     \@trnslt@if@translation{#2}{#3}
327     {\csuse{\@trnslt@word@#2@\@trnslt@language{#3}}}
328     }%
329   }}
330
331 % -----
332 % \LoadDictionary and \LoadDictionaryFor
333 \newrobustcmd*\LoadDictionary[1]{%
334   \@trnslt@load@dictionary@for{#1}{\@trnslt@current@language}}
335 \@onlypreamble\LoadDictionary
336
337 \newrobustcmd*\LoadDictionaryFor[2]{%
338   \@trnslt@load@dictionary@for{#2}{#1}}
339 \@onlypreamble\LoadDictionaryFor
340
341 % #1: name
342 % #2: lang
343 \def\@trnslt@load@dictionary@for#1#2{%
344   \AtBeginDocument{%
345     \InputIfFileExists{#1-\@trnslt@language{#2}.trsl}
346     {\@trnslt@info{loading dictionary `#1' for `#2'.}}
347     {\@trnslt@warning{File `#1-\@trnslt@language{#2}.trsl' not found.}}%
348   }}
349
350 \newrobustcmd*\ProvideDictionaryFor[2]{%
351   \@trnslt@provide@dictionary@for{#1}{#2}}
352 \@onlypreamble\ProvideDictionaryFor
353
354 \def\@trnslt@provide@dictionary@for#1#2{%
355   \def\@trnslt@dictionary@name{#2}%
356   \def\@trnslt@dictionary@lang{#1}%
357   \@ifnextchar[
358     {\@trnslt@provide@dictionary@version}
359     {\ProvidesFile{#2-#1.trsl}[{#1 translation file `#2'}]}}
360
361 \def\@trnslt@provide@dictionary@version[#1]{%
362   \ProvidesFile
363     {\@trnslt@dictionary@name-\@trnslt@dictionary@lang.trsl}%
364     [(\@trnslt@dictionary@langspace translation file \@trnslt@dictionary@name') #1]}
365
366 % \@trnslt@dictionary@language
367 \newrobustcmd*\DeclareDictTranslation[2]{%

```

5 Implementation

```
368 \@trnslt@declare@translation{#1}{\@trnslt@dictionary@lang}{#2}}
369 \@onlypreamble\DeclareDictTranslation
370
371 % -----
372 % predefined languages
373 \DeclareLanguage{albanian}
374 \DeclareLanguage{bulgarian}
375 \DeclareLanguage{catalan}
376 \DeclareLanguage{croatian}
377 \DeclareLanguage{czech}
378 \DeclareLanguage{danish}
379 \DeclareLanguage{dutch}
380 \DeclareLanguage{english}
381 \DeclareLanguage{finnish}
382 \DeclareLanguage{french}
383 \DeclareLanguage{german}
384 \DeclareLanguage{greek}
385 \DeclareLanguage{hebrew}
386 \DeclareLanguage{hungarian}
387 \DeclareLanguage{icelandic}
388 \DeclareLanguage{italian}
389 \DeclareLanguage{norwegian}
390 \DeclareLanguage{polish}
391 \DeclareLanguage{portuges}
392 \DeclareLanguage{romanian}
393 \DeclareLanguage{russian}
394 \DeclareLanguage{serbocroatian}
395 \DeclareLanguage{slovak}
396 \DeclareLanguage{slovenian}
397 \DeclareLanguage{spanish}
398 \DeclareLanguage{swedish}
399 \DeclareLanguage{turkish}
400 \DeclareLanguage{ukrainian}
401
402 \DeclareLanguageAlias {Albanian}{albanian}
403 \DeclareLanguageAlias {Bulgarian}{bulgarian}
404 \DeclareLanguageAlias {Catalan}{catalan}
405 \DeclareLanguageAlias {Croatian}{croatian}
406 \DeclareLanguageAlias {Czech}{czech}
407 \DeclareLanguageAlias {Danish}{danish}
408 \DeclareLanguageAlias {Dutch}{dutch}
409 \DeclareLanguageAlias {Finnish}{finnish}
410 \DeclareLanguageAlias {français}{french}
411 \DeclareLanguageAlias {Français}{français}
412 \DeclareLanguageDialect{canadien}{french}
413 \DeclareLanguageAlias {Canadien}{canadien}
414 \DeclareLanguageAlias {French}{french}
415 \DeclareLanguageDialect{american}{english}
416 \DeclareLanguageAlias {American}{american}
417 \DeclareLanguageDialect{australian}{english}
418 \DeclareLanguageAlias {Australian}{australian}
419 \DeclareLanguageDialect{british}{english}
```

5 Implementation

```
420 \DeclareLanguageAlias {British}{british}
421 \DeclareLanguageDialect{canadian}{english}
422 \DeclareLanguageAlias {Canadian}{canadian}
423 \DeclareLanguageAlias {English}{english}
424 \DeclareLanguageAlias {UKenglish}{british}
425 \DeclareLanguageAlias {USenglish}{american}
426 \DeclareLanguageDialect{austrian}{german}
427 \DeclareLanguageAlias {Austrian}{austrian}
428 \DeclareLanguageAlias {German}{german}
429 \DeclareLanguageAlias {germanb}{german}
430 \DeclareLanguageDialect{naustrian}{austrian}
431 \DeclareLanguageAlias {ngerman}{german}
432 \DeclareLanguageAlias {Greek}{greek}
433 \DeclareLanguageAlias {polutonikogreek}{greek}
434 \DeclareLanguageAlias {Hebrew}{hebrew}
435 \DeclareLanguageAlias {Hungarian}{hungarian}
436 \DeclareLanguageDialect{magyar}{hungarian}
437 \DeclareLanguageAlias {Magyar}{magyar}
438 \DeclareLanguageAlias {Icelandic}{icelandic}
439 \DeclareLanguageAlias {Italian}{italian}
440 \DeclareLanguageAlias {norsk}{norwegian}
441 \DeclareLanguageAlias {Norsk}{norsk}
442 \DeclareLanguageAlias {Norwegian}{norwegian}
443 \DeclareLanguageAlias {nynorsk}{norwegian}
444 \DeclareLanguageAlias {Nynorsk}{nynorsk}
445 \DeclareLanguageAlias {Polish}{polish}
446 \DeclareLanguageDialect{brazil}{portuges}
447 \DeclareLanguageAlias {Brazil}{brazil}
448 \DeclareLanguageAlias {brazilian}{brazil}
449 \DeclareLanguageAlias {Brazilian}{brazilian}
450 \DeclareLanguageAlias {Portuges}{portuges}
451 \DeclareLanguageAlias {portuguese}{portuges}
452 \DeclareLanguageAlias {Portuguese}{portuguese}
453 \DeclareLanguageAlias {Romanian}{romanian}
454 \DeclareLanguageAlias {Russian}{russian}
455 \DeclareLanguageAlias {Serbocroatian}{serbocroatian}
456 \DeclareLanguageAlias {Slovak}{slovak}
457 \DeclareLanguageAlias {Slovenian}{slovenian}
458 \DeclareLanguageAlias {Spanish}{spanish}
459 \DeclareLanguageAlias {Swedish}{swedish}
460 \DeclareLanguageDialect{swissgerman}{german}
461 % this maybe should be a language of it's own:
462 \DeclareLanguageAlias {Swiss}{swissgerman}
463 \DeclareLanguageAlias {Swissgerman}{swissgerman}
464 \DeclareLanguageAlias {Turkish}{turkish}
465 \DeclareLanguageAlias {Ukrainian}{ukrainian}
466
467 \endinput
468
469 % HISTORY:
470 2012/09/30 v0.2beta - first version (as part of the `exsheets' bundle)
471 2012/10/05 v0.2 - \LoadDictionary and \LoadDictionaryFor added and loads of
```

5 Implementation

languages defined.

2013/03/10 v0.8 - basic dictionaries for English, German, French and Spanish

2013/04/04 v0.8a - new command \DeclareDictTranslation

2013/04/07 v0.9 - bug fix in \DeclareDictTranslation

2013/04/08 v0.9a - slightly improved messages

- 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

2013/07/15 v0.10b - removed from `exsheets' bundle - `translations' should be a package of it's own

Index

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

B	G	P
babel 1 f., 4	\GetTranslation 2	polyglossia 1, 4
beamer 1	\GetTranslationFor 2	\providecaptionname 1
C	I	R
\captions<language> 1	Implementation 6	\RenewTranslation 2
D	L	Requirements 1
\DeclareDictTranslation 3	Languages 5 f.	S
\DeclareFallbackTranslation 4	\LoadDictionary 3	\SaveTranslation 2
\DeclareLanguage 2 f.	\LoadDictionaryFor 3	\SaveTranslationFor 2
\DeclareLanguageAlias 2	M	T
\DeclareLanguageDialect 2	Motivation 1	translator 1
\DeclareTranslation 2, 4	N	U
\DeclareTranslationFallback 2	\NewTranslation 2	Usage 2–5
E		
etoolbox 1		