The multibabel Package

v0.9.0d 2025/08/10

Generierung verschiedener Sprachversionen aus einer Quelle

Matthias Werner[1]

https://github.com/tuc-osg/osglecture

Dieses Paket bietet ein Interface zu Babel für die Unterstützung der Generierung von Dokumenten in mehreren Sprachvarianten aus einem gemeinsamen Quelldokument. Polyglossia wird nicht unterstützt. Das Paket gehört zum osglecture-Bundle und wird in die osglecture-Klasse integriert und vom Buildscript `ollm` unterstützt, kann aber auch eigenständig genutzt werden.

Table of Contents

1 Einleitung

Mitunter werden mehrere Sprachversionen des gleichen Dokuments benötigt. Im Anwendungsgebiet des osglecture-Bündel sind dies beispielsweise die Lehrskripte, Vorlesungsfolien und Handouts einer Lehrveranstaltung. Es fällt häufig schwer, mehrere parallel existierende Sprachversionen konsistent zu halten. Diese Aufgabe ist etwas einfacher, wenn alle Versionen in einem gemeinsamen Quelldokument enthalten sind.

Dieses Paket unterstützt die Generierung solcher Sprachversionen aus einem gemeinsamen LaTeX-Quelldokument und ermöglicht für die verschiedenen Sprachversionen automatisch die entsprechenden Babel-Optionen.

1. matthias.werner@informatik.tu-chemnitz.de

1.1 Alternativen

Es gibt eine Reihe von Alternativen zu multibabel.

Adhoc-Macros Man kann relativ einfach adhoc Lösungen schaffen, wie z. B.

```
\newif\ifenglish
\englishtrue
\ifenglish
    Welcome!
\else
    Willkommen!
\fi

```

Tatsächlich folgt multibabel grundsätzlich diesem Ansatz, automatisiert aber die Erstellung der Sprachmakros, sorgt für die korrekte Babeloptionen, so dass Wordtrennung und anderes korrekt ist, und vereinfacht die gemeinsame Nutzung von nichtsprachabhängigen Elementen wie z. B. Formeln.

multilang Das ausgeklügelte Paket multilang[2] von Richard Grewe erstellt Sprachversionen von gegebenen Makros, wobei die aktive Sprache direkt aus Babel oder Polyglossia übernommen wird. Wenn Sie den Ansatz von multibabel zu sperrig finden, sollten Sie sich unbedingt dieses Paket anschauen.

comment Victor Eijkhout hat des Paket comment[3] geschrieben, das auf einfache Weise ermöglicht, nur bestimmte Abschnitte im Dokument auszugeben. Dies kann auch sehr gut für eine Sprachauswahl genutzt werden.

translations/translator/xt_capts Diese drei Pakete translations[4]/translator[5]/xt_capts[6] addressieren Programmierer von LaTeX-Paketen, um fixe Textelemente wie die Überschriften von Verzeichnissen zu internationalisieren. Für Autoren mehrsprachiger Dokumente sind diese Pakete nur bedingt geeignet.

1.2 Sprachen

Das Konzept von multibabel beinhaltet die Nutzung von Sprachen auf verschiedenen Ebenen mit unterschiedlicher Wirkung. Um Verwechselungen zu vermeiden, wollen wir hier die Begriffe eindeutig definieren.

2. on CTAN as multilang: http://mirrors.ctan.org/macros/latex/contrib/multilang/
3. on CTAN as comment: http://mirrors.ctan.org/macros/latex/contrib/comment/
4. on CTAN as translations: http://mirrors.ctan.org/macros/latex/contrib/translations/
5. on CTAN as translator: http://mirrors.ctan.org/macros/latex/contrib/translator/
6. on CTAN as xt_capts: http://mirrors.ctan.org//macros/latex/contrib/xtcapts/xt_capts/

1. Die Menge *möglicher* Zielsprachen des Dokuments. Ein Dokument, das multibabel nutzt, enthält typischerweise Abschnitte in verschiedenen Zielsprachen. Die möglichen Zielsprachen werden über eine Paketoption gesetzt, siehe Abbschnitt 2.2. Im weiteren Text dieser Dokumentation nennen wir die möglichen Zielsprachen *Auswahlsprachen*, da für die LaTeX-Übersetzung eine dieser Sprachen ausgewählt wird.

2. Die tatsächliche bei einer LaTeX-Übersetzung gebrauchte Zielsprache. Diese muss eine der Sprachen aus 1. sein. Typischerweise wird der Übersetzungsvorgang für jede der tatsächlichen Zielsprachen einzeln ausgeführt. Für die Auswahl der tatsächlichen Zielsprache gibt es mehrere Methoden, siehe Abschnitt 2.1.

3. Die Sprache, in der Textteile oder einzelne Wörter tatsächlich geschrieben sind. Beispielsweise kann ein Dokument mit den Auswahlsprachen Deutsch und Englisch durchaus im Text lateinische Wörter oder Sätze enthalten. Diese Auswahl wird über die üblichen Babelmechanismen wie `\selectlanguage`, `\foreignlanguage` oder `\text`⟨*sprache*⟩ vorgenommen. Wir nennen eine so ausgewählte Sprache (lokal) *aktive* Sprache.

Sprachen werden entweder mit ihrem Babelnamen (”‘english”’, ”‘french”’, ”‘ngerman”‘ …), über ISO 639-1-Codes (z. B. ”‘en”’, ”‘fr”’, ”‘de”’, …) oder BCP47-Codes (z. B. ”‘de-AT”’, ”‘fr-Fr”’, ”‘en-US”’, …) bezeichnet.

2 Anwendung

Das Paket wird auf die übliche Weise geladen: `\usepackage[`⟨*options*⟩`]{multibabel}`
Zu diesem Zeitpunkt darf Babel noch nicht geladen sein, da es sonst zu einen Optionskonflikt kommen kann.

Das Paket multibabel ruft selbst Babel auf. Entsprechend wird

2.1 Auswahl der Zielsprache

Da die Auswahl der Zielsprache sowohl über Optionen als auch auf anderem Weg erfolgen kann, werden hier zunächst diese verschiedenen Wege beschrieben. Weitere Optionen wie das Setzen der Auswahlsprachen werden in Abschnitt 2.2 besprochen. Um eine Zielsprache für einen LaTeX-Übersetzungslauf zu bestimmen, gibt es verschiedene Möglichkeiten. Die Reihenfolge in der folgenden Liste spiegelt auch die Reihenfolge der Auswertung. Sobald damit eine Zielsprache ermittelt wird, wird eine weitere Auswertung abgebrochen.

1. Definition eines Makros `\omdTargetLanguage` *vor* Laden des Pakets. Damit ist es beispielsweise möglich, über

   ```
   > latexmk -e"\def\omdTargetLanguage{fr}" document.tex
   ```

   die Zielsprache Französisch für den Übersetzungslauf festzulegen.

2. Setzen der Paketoption `targetlang` = `{`⟨*Zielsprache*⟩`}`

3. Wenn die Option `targetlang = {job=⟨n⟩}` gesetzt wird, wird die Zielsprache aus dem n. Element des Jobnamens bestimmt. Ein Element ist ein mit "'-"' (Minus) abgetrennter Teil des Jobnamens, siehe die Dokumentation zum Paket varsfromjobname,[7] das dafür geladen wird. Beispielsweise kann bei `targetlang = {job=2}` über

   ```
   > latex -jobname doc-ru doc.tex
   ```

   Russisch als Zielsprache eingestellt werden.

4. Mit Setzen von `targetlang = {meta}` wird die in `\DocumentMetaData` angegebene Sprache als Zielsprache verwendet.

5. Wenn alles andere versagt, wird Englisch als Zielsprache gesetzt.

2.2 Options

Es können folgende Optionen gesetzt werden:

`languages = {`⟨*Liste von Auswahlsprachen*⟩`}` (required)
Gibt die Auswahlsprachen an. Sinnvollerweise sollten wenigstens zwei Sprachen angegben werden. Die Reihenfolge ist relevant für die Generierung der Sprachmakros (siehe Abschnitt 3).

Alle hier eingegebenen Sprachen werden in Babel geladen. Deshalb können hier auch Sprachen angegeben werden, die nicht als Zielsprache gedacht sind, sondern später über `\foreignlanguage` als aktive Sprache genutzt werden sollen.

`targetlang =` ⟨*Sprache*⟩`|{job=`⟨*n*⟩`}|meta`
Siehe Abbschnitt 2.1

`prefix = {`⟨*prefix*⟩`}` Default: `l`
Setzt den

3 Makros

4 Beispiele

5 Implementation

```
\NeedsTeXFormat{LaTeX2e}[2022/06/01]
\def\packagename{multibabel}
\def\packageversion{2025/08/10 v0.9.0d}
\ProvidesPackage{\packagename}[\packageversion\space babel interface for language versions]
```

Im Moment brauchen wir noch LuaLaTeX für die Sternvariante der Sprachmakros.

7. on CTAN as `varsfromjobname`: `http://mirrors.ctan.org/macros/latex/contrib/varsfromjobname/`

```
\ExplSyntaxOn
\sys_if_engine_luatex:TF{
  \RequirePackage{luacode}
}{
 \ClassError{\packagename}{\MessageBreak
  **********************************************\MessageBreak
  *~LuaLaTeX~is~required~to~use~this~package. \MessageBreak
  *~Sorry! \MessageBreak
  **********************************************
 }{Use~this~package~with~LuaLaTex.}
\ExplSyntaxOff
```

Wir definieren zwei Befehle, um die Korrektheit der Sprachnamen bzw. Tags zu prüfen.

```
\newcommand\IfValidBabelName[1]{%
  \begingroup
```

Catch babel error

```
  \def\PackageError##1##2##3{\endgroup\@secondoftwo}%
  % Test to load:
  \babelprovide{#1}%
  \endgroup\@firstoftwo
}
\newcommand\IfValidBabelTag[1]{%
  \begingroup
  \edef\bbl@tempa{#1}%
    % We use an internal babel function, not an official API.
  \expandafter\bbl@bcplookup\bbl@tempa-\@empty-\@empty-\@empty\@@
  \ifx\bbl@bcp\relax
      \endgroup\@secondoftwo%
  \else
      \endgroup\@firstoftwo%
  \fi
}
\newcommand*\ombLoadLanguageForTag[2][]{%
  \begingroup
  \edef\bbl@tempa{#2}%
  % We use an internal babel function, not an official API
  \expandafter\bbl@bcplookup\bbl@tempa-\@empty-\@empty-\@empty\@@
  \ifx\bbl@bcp\relax
    \ifx#1\relax
      \global\cslet{#1}{\@empty}
    \fi
```

```
    \endgroup
  \else
    % Read the corresponding ini file.
    \bbl@read@ini{\bbl@bcp}\m@ne
    \ifx#1\relax
      \csxdef{#1}{\languagename}
    \fi
    \endgroup
  \fi
}
```

Das Flag `\ombLangsSet` soll gesetzt werden, sobald eine Liste von Auswahlsprachen angegeben wird.

```
\newtoggle{ombLangsSet}\togglefalse{ombLangsSet}
\ExplSyntaxOn
\DeclareKeys[multibabel]{
  languages.code = {
    \clist_new:N\cl_lang
    \clist_gset:Nn{\cl_lang}{#1}
    \clist_map_inline:Nn{\cl_lang}{
      \IfValidBabelName{#1}{
        \babelprovide{#1}
      }{
        \IfValidBabelTag{#1}{
          \mbLoadLanguageForTag{#1}
        }{
          \PackageError{\packagename}{Couldn't~resolve~language~'#1'}{
            Use~valid~language~selector~in~option~'languages'.}
          \aftergroup\endinput
        }
      }
      \toggletrue{ombLangsSet}
    }
  },
  languages.usages=load,
  targetlang.code = {
    \ifundef{omdTargetLanguage}{
          \IfValidBabelName{#1}{% valid babel language
            \edef\omdTargetLanguage{#1}
          }{% no babel language
            \IfValidBabelTag{#1}{% valid language tag
              \edef\omdTargetLanguage{#1}
            }{% no valid tag
              \str_if_eq:eeTF{\str_range:Nnn{#1}{1}{3}}{job}{% targetlang =job n
```

```
                \RequirePackage{varsfromjobname}
                \edef\omdTargetLanguage{\getfromjobname{\str_range:Nnn{#1}{-1}{-1}}}
              }{% targetlang != job
                \str_if_eq:nnTF{#1}{meta}{% destlang=meta
                  \IfDocumentMetadataTF{% \DocumentMetadata is loaded.
                    % No check for correct language is needed, since at this point in
                    % time, babel is already loaded and has checked the language, if
                    % \DocumentMetadata is loaded.
                    \edef\omdTargetLanguage{ \g_document_properties_prop { document/lang }}
                  }{%
                    \PackageError{\packagename}{Option~destlang=meta,~but~no~
                      \string\DocumentMetadata~is~provided}{Provide~\string\DocumentMetadata.}
                  }
                }{% targetlang != meta
                  \PackageWarnigNoLine{\packagename}{I~wasn't~able~to~recognize~the~
                    destination~language.\MessageBreak
                    I'll~go~with~'english'
                  }
                  \def\omdTargetLanguage{en}
                }
              }
            }
          }
        }
      },
  targetlang.usage=load,
  static prefix.store=\mb@sprefix,
  static prefix.usage=load,
}%



\ProcessOptions\relax
\iftoggle{mbLangsSet}{}{
  \PackageError{\packagename}{Option 'languages' not set.}{You have to provide a
  non-empty list of languages.}
}
```

−−−−−− Ende Paketcode −−−−−−−−−−-

6 Anwendung

Laden und verwenden:

```
\usepackage{mypkg}
```

Das war’s.