The langselect Package

v0.9.0d 2025/08/10

Generierung verschiedener Sprachversionen aus einer Quelle

Matthias Werner[1]

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

Dieses Paket unterstützt die Generierung von Dokumenten in zwei oder drei Sprachvarianten aus einem gemeinsamen Quelldokument. Das Paket gehört zum osglecture-Bundle und wird in die osglecture-Klasse integriert und vom Buildscript `ollm` unterstützt, kann aber auch eigenständig genutzt werden.

Inhaltsverzeichnis

1 Einleitung

1.1 TL;DR

Angenommen Sie wollen von einen Text eine englische und ein deutsche Version erhalten, wobei Sie ein gemeinsames Quelldokument nutzen wollen..

1. Binden Sie @1 ein

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

2. Nutzen Sie im Quelldokument `\lende{`$\langle Text_{en}\rangle$`}{`$\langle Text_{de}\rangle$`}`. $\langle Text_{en}\rangle$ wird in der englischen Version erscheinen, während Text_{de} in der deutschen Version erscheinen wird.
   - Text, der in beiden Versionen vorkommt, schreiben Sie außerhalb des Makros oder nutzen die Sternversion, um gemeinsame Daten im Fließtext nur einmal schreiben zu müssen, siehe Abschnitt @1.
3. Übersetzen Sie das Dokument mit @1, um die englische Version zu erhalten, und mit @2 für die deutsche.
   - Es gibt noch einige andere Möglichkeiten,die Zielsprache zu bestimmen, u. a. über den Jobnamen und über @1.

1.2 Motivation

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 indem es Makros für Sprachvarianten und Mechanismen zur Auswahl der Zielsprache bereitstellt.

1.3 Alternativen

Es gibt eine Reihe von Alternativen zu langselect.

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

```
\newif\ifenglish
\englishtrue
\ifenglish
    Welcome!
\else
    ¡Bienvenido!
\fi

```

Tatsächlich folgt langselect grundsätzlich diesem Ansatz, automatisiert aber die Erstellung der Sprachmakros, vereinfacht die gemeinsame Nutzung von nichtsprachabhängigen Elementen wie z. B. Formeln und ermöglicht die Auswahl der Sprachvariante über verschiedene Methoden.

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. langselect hat ein ähnliches Ziel, aber ein anderes (aus Sicht des Paketautoren: praktischeres) Interface. Falls Sie aber den Ansatz von langselect nicht mögen, sollten Sie sich unbedingt multilang 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.

bicaption/translations/translator/xt_capts Die Pakete @1 addressieren Programmierer von LaTeX-Paketen, um fixe Textelemente wie die Überschriften von Verzeichnissen zu internationalisieren. Für Dokumentautoren sind diese Pakete nur bedingt geeignet.

1.4 Sprachen

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

1. Die Menge *möglicher* Zielsprachen des Dokuments. Ein Dokument, das langselect nutzt, enthält typischerweise Abschnitte in verschiedenen Zielsprachen. Die möglichen Zielsprachen werden über eine Paketoption gesetzt, siehe Abbschnitt @1. 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. Dies ist das Konzept von Sprache, das Pakete wie Babel oder Polyglossia haben.[4] Bitte beachten Sie, dass ein mehrsprachiges Dokument aus der Sicht von langselect ein monosprachiges Dokument aus der Sicht von Babel sein kann. Allerdings können durchaus noch weitere Sprachen hinzukommen: Beispielsweise kann ein Dokument mit den Auswahlsprachen Deutsch und Englisch im Text lateinische Wörter oder Sätze enthalten. Diese Auswahl wird über die üblichen Babelmechanismen wie `\selectlanguage`, `\foreignlanguage` oder `\text`⟨*sprache*⟩ vorgenommen. Wenn Babel oder Polyglossia geladen wird, was auf Wunsch auch direkt über langselect erfolgen kann, wird als aktive Sprache die Zielsprache ausgewählt.

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. Wobei auch in diesen Pakten „Sprache“ auch für mehrere unterschiedliche und z. T. voneinander unabhängige Konzepte wie Trennungsmuster, zu nutzende Fonts etc. steht

Sprachen werden in langselectüber ISO 639-1-Codes (z. B. „en“, „fr“, „de“, …) bezeichnet. Wenn für die Sprachbehandlung auf der Ebene von Babel/Polyglossia ein bestimmter Dialekt oder eine bestimmte Variante gewünscht ist, kann ein Mapping erfolgen.

2 Anwendung

Das Paket wird auf die übliche Weise geladen:

`\usepackage[`⟨*options*⟩`]{langselect}`

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 @1 besprochen.

2.1 Auswahl der Zielsprache

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 `\olsTargetLanguage` *vor* Laden des Pakets. Damit ist es beispielsweise möglich, über

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

   die Zielsprache Französisch für den Übersetzungslauf festzulegen.

2. Setzen der Paketoption @1. Verschiedene Werte sind hier möglich:

`targetlang` = ⟨*ISO-Code*⟩
: Die Zielsprache wird direkt durch den Code angegeben.

`targetlang` = `{job=`⟨n⟩`}`
: Die Zielsprache wird aus dem n. Element des Jobnamens bestimmt. Ein Element ist ein mit „-“ (Minus) abgetrennter Teil des Jobnamens, siehe die Dokumentation zum Paket varsfromjobname,[5] das dafür geladen wird.Beispielsweise kann bei `targetlang = {job=2}` über

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

  Russisch als Zielsprache eingestellt werden.

`targetlang` = `babel|polyglossia`
: Für die Nutzung dieses Schlüsselwerts muss Babel oder Polyglossia *vor* langselect geladen sein. Die Zielsprache wird dann auf die durch das Sprachpaket eingestellte (Haupt-)Sprache gesetzt.[6]

`targetlang` = `meta`
: Die in `\DocumentMetaData` angegebene Sprache als Zielsprache verwendet.

5. on CTAN as `varsfromjobname`: `http://mirrors.ctan.org/macros/latex/contrib/varsfromjobname/`
6. Beide Werte bewirken exakt das Gleiche: Da Babel und Polyglossia die gleiche Schnittstelle anbieten, ist es egal, welches von beiden Paketen geladen ist. `targetlang = polyglossia` findet also auch Babel.

3. Wenn weder `\olsTargetLanguage` definiert noch die Option `targetlang` gesetzt wird, versucht langselect selbständig die Zielsprache zu bestimmen, indem es erst die Option `targetlang = babel` und dann (wenn kein Sprachpaket gefunden wurde) die Option `targetlang = meta` ausprobiert.

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

2.2 Optionen

Folgende Optionen stehen zur Verfügung:

`languages` = {⟨*Liste von Auswahlsprachen*⟩} (erforderlich)
Gibt zwei oder drei Auswahlsprachen an. Die Reihenfolge ist relevant für die automatische Generierung der Sprachmakros (siehe Abschnitt @1).

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

`prefix` = {⟨*prefix*⟩} Voreinstellung: `l`
Setzt einen Prefix für die Multisprachmakros. Es kann auch ein leerer Prefix gesetzt werden.

`auto` = true|false Voreinstellung: `true`
Die eigentlichen Sprachmakros werden automatisch angelegt, siehe Abschnitt @1. Wird `auto = false` gesetzt, werden nur die Makros zur Erzeugung der Sprachmakros zur Verfügung gestellt, nicht die Sprachmakros selbst.

`trim` = true|false Voreinstellung: `true`
Enternt Whitespaces am Anfang und Ende der Argumente der Sprachmakros.

`load babel` = {⟨*Babel Optionen*⟩} Voreinstellung:

`load Polyglossia` = {⟨*Polyglossia Optionen*⟩} Voreinstellung:
Lädt entweder Babel oder Polyglossia so, dass die Zielsprache von langselect als Hauptsprache gesetzt wird. Die hier übergebenen Optionen werden an das jeweilige Sprachpaket weitergeleitet. Wenn eine dieser Option genutzt wird, darf nicht `targetlang = babel` oder `targetlang = polyglossia` gesetzt sein und langselect muss vor dem jeweiligen Sprachpaket geladen werden.

`map` = {⟨*Sprache*⟩=⟨*Variante*⟩,…}
Legt ein Mapping von Sprachen zu einer Sprachvariante fest. Wenn langselect mit anderen Paketen interagiert, wird dann die spezifische Sprachvariante statt des ISO-Codes weitergegeben.

Falls Sie beispielsweise Babel von langselect aus laden und Ihr Dokument explizit *Britisches* Englisch und *Schweizerdeutsch* nutzt, sichern Sie mit folgendem Code ab, dass Babel korrekte Optionen erhält.

```
\documentclass{article}
\usepackage[
  languages={en,de},
  load=babel,
  map={en=british,de=nswissgerman}
]
\begin{document}
 \title{\lende{Hallo}{Grüezi}}

```

unified shorthands = true|false

3 Nutzermakros

3.1 Automatisch Sprachmakros

Wenn Sie nicht die Option auto = false gesetzt haben, wird ein Multisprachmakro angelegt. Dessen Name wird durch die Liste der Auswahlsprachen festgelegt. Je nachdem, ob Sie zwei oder mehr Auswahlsprachen angegeben haben, ist ist der Name

- \⟨*Prefix*⟩⟨*Sprache*$_1$⟩⟨*Sprache*$_2$⟩{⟨*Text*$_1$⟩}{⟨*Text*$_2$⟩}

oder

- \⟨*Prefix*⟩⟨*Sprache*$_1$⟩⟨*Sprache*$_2$⟩⟨*Sprache*$_3$⟩{⟨*Text*$_1$⟩}{⟨*Text*$_2$⟩}{⟨*Text*$_3$⟩}

Beispielsweise für für diese Dokumentation das Paket mit folgenden Optionen geladen:

```
\usepackage[
  languages={de,en},
  targetlang={job=2}
]{langselect}
```

Damit steht das Makro \ldeen{⟨*Text*$_1$⟩}{⟨*Text*$_1$⟩} zur Verfügung. Wenn jetzt die Zielsprache „de“ ausgewählt wird (hier über den Jobnamen, aber es gibt auch andere Möglichkeiten, vgl. Abschnitt @1), wird nur ⟨*Text*$_1$⟩ in das PDF-Dokument übernommen und ⟨*Text*$_2$⟩ verworfen. Bei der Zielsprache „en“ ist es umgekehrt.

Achtung!
Die automatische Generierung der Sprachmakros kann manchmal zu Problemen führen. Beispielsweise ergibt Abchasisch (Code: „ab“) als Erst- und Griechisch (Code: „el“) als

Zweisprache den automatischen Namen `\label`, was zu Fehlermeldungen führt. Nutzen Sie in solchen Fällen die im Abschnitt 3.3 beschriebenen Möglichkeiten.

3.2 Sternvariante

Ein häufiges Problem bei mehreren Sprachversionen ist die Konsistenz nichtsprachabhängiger Daten, insbesondere wenn diese Daten sich öfter ändern. (Denken Sie z. B.an Stundenpläne.)

Für solche Fälle stellt langselect eine Sternvariante der Sprachmakros bereit. In dieser Sternvariante können die Textargumente Platzhalter der Form „@⟨*n*⟩“ enthalten, wobei ⟨*n*⟩ einen Wert zwischen 1 und 9 darstellt. Hinter dem Makro müssen dann weitere Argumente angegeben werden, und zwar so viele, wie es dem höchsten verwendeten ⟨*n*⟩ entspricht. Die Sternvariante ersetzt die Platzhalter mit den entsprechenden Argumenten.

Die Sternvariante der Sprachmakros ist „zerbrechlich“, bitte verwenden Sie sie nicht in einem beweglichen Kontext, z. B. innerhalb eines `\section`-Arguments.

```
\documentclass{article}
\usepackage[languages={en,fr},
targetlang=fr]{langselect}

\begin{document}
\lenfr*{
  The formula @1 generally refers to the Pythagorean theorem.
}{
  La formule @1 fait généralement référence au théorème de
  Pythagore.
}{$a^{2}+b^{2}=c^{2}$}

\lenfr*{
  Calculate the length of the third side if one side is @1 units long
  and the hypotenuse is @2 units long.
}{
  Calculez la longueur du troisième
  côté si un côté mesure @1 unités et l'hypoténuse @2 unités.
}{$4$}{$5$}
\end{document}
```

3.3 Individuelle Sprachmakros

Wenn Ihnen die generierten Namen der Sprachmakros nicht gefallen oder diese sogar zu Kollisionen führen, können Sie diese ändern. Am einfachsten ist es, die Paketoption `prefix` zu ändern, so dass beispielsweise mit `prefix = mv` das Sprachmakro für die Deutsch/Englisch-Kombination `\mvdeen` heisst.

Wenn dies nicht ausreichend ist, geben Ihnen die folgenden zwei Makros die vollständige Kontrolle über die Namensbildung.

`\olsMakeBilangualMacro[`⟨*Prefix*⟩`]{`⟨*Sprache*$_1$⟩`}{`⟨*Sprache*$_2$⟩`}[`⟨*Basisname*⟩`]`

`\olsMakeTrilangualMacro[`⟨*Prefix*⟩`]{`⟨*Sprache*$_1$⟩`}{`⟨*Sprache*$_2$⟩`}{`⟨*Sprache*$_3$⟩`}[`⟨*Basisname*⟩`]`

Diese Makros legen ein bilinguales bzw. trilinguales Sprachmakro für die angegebenen Sprachen an. Wenn ⟨*Basisname*⟩ angegeben wird, ist der Name des Sprachmakros `\`⟨*Prefix*⟩⟨*Basisname*⟩. Wenn das ⟨*Prefix*⟩-Argument benutzt wird, bestimmt dieses den Prefix, sonst gelten die Paketoptionen.

4 Interaktion mit anderen Paketen

4.1 Babel/Polyglossia

Das Paket langselect ist zunächst einmal unabhängig von Sprachpaketen wie Babel oder Polyglossia. Es kann ohne oder neben diesen Paketen eingesetzt werden. Wenn man sich jedoch die Mühe eines mehrsprachigen Dokuments macht, will man in der Regel auch, dass für die jeweilige Sprache die richtigen Trennungsregeln, Fonteinstellungen, Datumsformate etc. benutzt werden.[7]

Bitte beachten Sie, dass, solange Sie nicht *innerhalb* eines Textes eine andere Sprache (vielleicht ja die Sprache der anderen Sprachversion) nutzen, ihr Dokument aus der Sicht von Babel ein *einsprachiges* Dokument ist. Es besteht die Gefahr, dass Sie Babel auf die gewohnte Weise mit den verschiedenen Sprachen ihrer Sprachversion aufrufen, aber zur Bestimmung der Zielsprache nicht die Option `targetlang = babel` nutzen. Als Ergebnis stimmt u.U. die Zielsprache nicht mit Babels aktiver Sprache überein, und können Sie inkorrekte Trennungen und ähnliche Fehler erhalten.

Um solche Probleme zu vermeiden, bieten langselect verschiedene Möglichkeiten:

1. Wie oben erwähnt kann man die Paketoption `targetlang = babel` nutzen. Dann wird die Zielsprache durch die Hauptsprache von Babel bestimmt. Allerdings muss man dann auf die Möglichkeit verzichten, die Zielsprache „von außen“ (also mit `\de` oder über den Jobnamen) zu steuern.

2. Umgekehrt lädt langselect Babel oder Polyglossia mit Hilfe der Optionen `load babel`/`load polyglossia` landen und setzt die Zielsprache als Hauptsprache. Weitere Optionen an die Sprachpakete werden als Optionsargument übergeben.

3. Schließlich kann mit dem Hook `langselect/language` beliebiger Code vor Ausgabe der Argumente des Sprachmakros eingefügt werden, z. B. kann hier `\foreignlanguage` genutzt werden. Der Hook hat als Parameter die Zielsprache.

7. Im Folgenden werde ich mich ausschließlich auf Babel beziehen. Das Gesagte gilt jedoch entsprechend auch für Polyglossia.

4.2 Csquotes

5 FAQ

6 Implementation

```
\NeedsTeXFormat{LaTeX2e}[2022/06/01]
\def\packagename{langselect}
\def\packageversion{2025/08/10 v0.9.0d}
\ProvidesPackage{\packagename}[\packageversion\space support for different
languages versions of a document]
\ExplSyntaxOn
```

\olsIsoTags enthält alle gültigen ISO 639-1-Codestags.

```
\seq_const_from_clist:cn{olsIsoTags}{aa,ab,ae,af,ak,am,an,ar,as,av,ay,az,ba,be,bg,
  bh,bi,bm,bn,bo,br,bs,ca,ce,ch,co,cr,cs,cu,cv,cy,da,de,dv,dz,ee,el,en,eo,es,et,eu,
  fa,ff,fi,fj,fo,fr,fy,ga,gd,gl,gn,gu,gv,ha,he,hi,ho,hr,ht,hu,hy,hz,ia,id,ie,ig,ii,
  ik,io,is,it,iu,ja,jv,ka,kg,ki,kj,kk,kl,km,kn,ko,kr,ks,ku,kv,kw,ky,la,lb,lg,li,ln,
  lo,lt,lu,lv,mg,mh,mi,mk,ml,mn,mr,ms,mt,my,na,nb,nd,ne,ng,nl,nn,no,nr,nv,ny,oc,oj,
  om,or,os,pa,pi,pl,ps,pt,qu,rm,rn,ro,ru,rw,sa,sc,sd,se,sg,si,sk,sl,sm,sn,so,sq,sr,
  ss,st,su,sv,sw,ta,te,tg,th,ti,tk,tl,tn,to,tr,ts,tt,tw,ty,ug,uk,ur,uz,ve,vi,vo,wa,
  wo,xh,yi,yo,za,zh,zu}

\newcommand\IfTagIsValidF[2]{%
  \seq_if_in:ceF{olsIsoTags}{#1}{#2}
}
\RequirePackage{varsfromjobname}[2025/08/03]
```

Das Ergebnisse von \varsfromjobname und \str_range:Nnn haben Catcode 12, aber Iso-Tags und Optionen haben Catcode 11. Wir vereinheitlichen durch einen Wrapper auf Catcode 11.

```
\newcommand\olsChangeCatEleven[1]{
  \def\tmpa{#1}
  \edef\tmpb{
    \noexpand\scantokens{
      \noexpand\edef\noexpand\olsCatElven{\tmpa}
    }
  }
  \tmpb
}
```

Bei Bedarf bilden wir Spachcode auf Sprachvarianten ab. Bei Interaktionen mit externen Code wir dann die Sprachvariante weitergegeben.

```
\cs_new:Npn \ols_use_map:n #1 {
```

```
  \cs_if_exist_use:cF { olslangmap / #1 }{ #1 }
}
```

Wir definieren einige Optionen. Alle außer `language` sind optional. Einige Ergebnisse werden wir in Variablen spreichern, die wir zuerst anlegen.

```
\bool_new:N\ols_load_babel
\bool_new:N\ols_load_polyglossia
\DeclareKeys{
  languages.clist_gset:N = \ols_langs,
  languages.usage=load,
  targetlang.code = {
    \cs_if_exist:NF\olsTargetLanguage{
      \str_if_eq:eeT{\str_range:Nnn{#1}{1}{3}}{job}{%
        \olsChangeCatEleven{\getfromjobname{\str_range:Nnn{#1}{-1}{-1}}}
        \xdef\olsTargetLanguage{\olsCatElven}
      }
      \bool_if:nTF{
        \str_if_eq_p:nn{#1}{babel} || \str_if_eq_p:nn{#1}{polyglossia}
      }{
        \str_if_empty:eF{\BCPdata{main.language}}{
          \edef\olsTargetLanguage{\BCPdata{main.language}}
        }
      }
      \str_if_eq:nnTF{#1}{meta}{
        \IfDocumentMetadataTF{
          \edef\tmpa{\GetDocumentProperty{document/lang}}
          \edef\tmpb{\str_range:Nnn\tmpa{1}{2}}
          \olsChangeCatEleven{\tmpb}
          \edef\olsTargetLanguage{\olsCatElven}
        }{
          \PackageWarningNoLine{\packagename}{No~meta~data~provided}
        }
      }
    }
  },
  targetlang.usage=load,
  prefix.store=\ols_sprefix,
  prefix.initial:n=l,
  prefix.usage=load,
  auto.bool_set:N=\ols_generate,
  auto.usage=load,
  auto.initial:n=true,
  trim.bool_set:N=\ols_trim,
  trim.usage=load,
```

```
  trim.initial:n=true,
  map.code:n = {
    \prop_new:N \ols_lang_pl
    \prop_clear:N \ols_lang_pl
    \prop_set_from_keyval:Nn \ols_lang_pl {#1}
```

Da Property-Listen nicht expandierbar sind, setzen wir die Einträge auf gewöhnlich Makros um.

```
    \prop_map_inline:Nn \ols_lang_pl
      { \cs_gset:cpn { olslangmap / \tl_to_str:n {##1} } {##2} }
  },
  load~babel .code:n = {
    \bool_set_true:N \ols_load_babel
    \PassOptionsToPackage{#1}{babel}
  },
  load~polyglossia .code:n = {
    \bool_set_true:N \ols_load_polyglossia
    \PassOptionsToPackage{#1}{polyglossia}
  },
  load.usage = load,
  unified~shorthands.bool_set:N=\ols_shorthands,
  unified~shorthands.initial:n=true,
  unified~shorthands.usage=load
}%
\ProcessKeyOptions\relax
```

Wenn über `\olsTargetLanguage` oder die `targetlang` keine Zielsprache zugewiesen wurde, untersuchen wir noch einmal, ob über Sprachpakete oder `\DocumentMetadata` eine Sprache identifiziert werden kann.

```
\cs_if_exist:NF \olsTargetLanguage {
  \str_if_eq:eeF{\BCPdata{main.language}}{}{\edef\olsTargetLanguage{\BCPdata{main.language}}}
}
\cs_if_exist:NF \olsTargetLanguage {
  \IfDocumentMetadataT{
    \edef\tmpa{\GetDocumentProperty{document/lang}}
    \edef\tmpb{\str_range:Nnn\tmpa{1}{2}}
    \olsChangeCatEleven{\tmpb}
    \edef\olsTargetLanguage{\olsCatElven}
  }
}
\cs_if_exist:NF \olsTargetLanguage {
  \PackageWarningNoLine{\packagename}{Can't~identify~any~target~language.
    \MessageBreak Falling~back~to~'en'}
```

```
  \def\olsTargetLanguage{en}
}
```

Wir führen verschiedene Checks durch. Als erstes prüfen wir, ob eine valide Zielsprache ermittelt werden konnte.

```
\IfTagIsValidF{\olsTargetLanguage}{
  \PackageWarningNoLine{\packagename}{Do~not~recognize~language~'\olsTargetLanguage'.
    \MessageBreak Take~'en'~as~replacement}
  \def\olsTargetLanguage{en}
}
```

Sind mindestes zwei Auswahlsprachen angegeben?

```
\int_compare:nNnT{\clist_count:N{\ols_langs}} < {2}{
  \PackageError{\packagename}{No~sufficient~number~of~selectable
    \MessageBreak~languages~provided}{
    You~have~to~provide~a~list~of~two~or~three\MessageBreak
    languages~via~'language'~option.}
  \aftergroup\endinput
}
```

Im Moment kann langselect nicht mehr als drei Auswahlsprachen verarbeiten. Wir warnen daher, wenn mehr als drei Sprachen angegeben werden.

```
\int_compare:nNnT{\clist_count:N\ols_langs} > {3}{
  \PackageWarningNoLine{\packagename}{Too~many~selectable~languages~provided.
    \MessageBreak I~will~ignore~the~superflous~languages}
}
```

Als nächstes überprüfen wir, ob alle Auswahlsprachen gültige ISO 639 1 Codes sind.

```
\clist_map_inline:Nn\ols_langs {
  \IfTagIsValidF{#1}{
    \PackageError{\packagename}{Couldn't~resolve~selectable~language~'#1'}{
      Use~valid~ISO-639-1~code~in~option~'languages'.}
    \aftergroup\endinput
  }
}
```

Die Zielsprache sollte in der Liste von Auswahlsprachen enthalten sein.

```
\clist_if_in:NVF\ols_langs{\olsTargetLanguage}{
  \PackageError{\packagename}{Target~language~'\olsTargetLanguage'~is~not\MessageBreak
    in~the~list~of~selectable~languages}{
    Check~option~'language'~and~'targetlang'.}
}
```

Babel und Polyglossia können nicht beiden geladen werden.

```
\bool_if:nT{
  \ols_load_babel && \ols_load_polyglossia
}{
  \bool_set_false:N \ols_load_babel
  \bool_set_false:N \ols_load_polyglossia
  \PackageError{\packagename}{The~options~'load~babel'~and~'load~polyglossia'
    \MessageBreak are~mutual~exclusive,~you~can't~use~both}{Use~one~load~option,~only.}
}
\bool_if:nT{
  ( \cs_if_exist_p:c{ver@babel.sty} || \cs_if_exist_p:c{ver@polyglossia.sty} )
  &&
  ( \ols_load_babel && \ols_load_polyglossia )
}{
  \bool_set_false:N \ols_load_babel
  \bool_set_false:N \ols_load_polyglossia
  \PackageWarningNoLine{\packagename}{Can't~load~babel/polyglossia~for~you,since\MessageBreak
    a~language~package~is~already~loaded.\MessageBreak
    I'll~ignore~that~option}
}
```

@1 ist die Bezeichnung der Zielsprache, wie sie ggf. an externen Code weitergegeben wird, d.,h. `\DocumentMetadata`, Sprachpakete und Hooks.

```
\edef\ols_mapped_target_lang{\ols_use_map:n{\olsTargetLanguage}}
```

Wir korrigieren noch ggf. die über `\DocumentMetadata` gesetzen Metadaten auf die der Zielsprache, so dass beispielsweise Babel keine falsche Hauptsprache ableitet.

```
\IfDocumentMetadataT{
  \edef\olsDocLang{\exp_args:Ne\str_range:nnn{\GetDocumentProperty{document/lang}}{1}{2}}
  \bool_if:nF{
    \str_if_eq_p:ee{\olsDocLang}{\olsTargetLanguage}
  }{
    \PackageWarningNoLine{\packagename}{Target~language~'\olsTargetLanguage'~
      doesn't~comply~with\MessageBreak \c_backslash_str
      DocumentMetadata~'\GetDocumentProperty{document/lang}'.\MessageBreak
      I'll~try~to~overwrite~the~meta~data}
    \DocumentMetadata{lang=\ols_mapped_target_lang}
  }
}
```

Da der Name des Sprachmakros nicht bekannt ist, sind Patchen oder generische Hooks komplizierter. Wir stellen deshalb eigene Hooks zur Verfügung.

```
\NewHookWithArguments{langselect/language}{1}
\NewHookWithArguments{langselect/argument}{2}
```

Ein Boolean zur Markierung, ob mehr als zwei Auswahlsprachen angegeben wurden.

```
\bool_new:c{ols_trilang}
\int_compare:nNnTF{\clist_count:N{\ols_langs}} > {2}{
  \bool_set_true:c{ols_trilang}
}{
  \bool_set_false:c{ols_trilang}
}
\def\olsFrstLanguage{\clist_item:Nn \ols_langs{1}}
\def\olsScndLanguage{\clist_item:Nn \ols_langs{2}}
\bool_if:cT{ols_trilang}{
  \def\olsThrdLanguage{\clist_item:Nn \ols_langs{3}}
}
\bool_if:NTF\ols_trim{
  \let\ols_trim:n=\tl_trim_spaces:n
}{
  \let\ols_trim:n=\relax
}
```

Die Makros `\olsMakeBilangualMacro` bzw. `\olsMakeTrilangualMacro` dienen zur Generierung der eigentlichen Sprachmakros.

```
 % #1: prefix, #2,#3: selectable languages, #4: base name
\NewDocumentCommand{\olsMakeBilangualMacro}{O{\ols_sprefix} m m o}{
  \str_if_eq:eeT{\olsTargetLanguage}{#2} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#4}{s +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\olsProcessArgs{\ols_trim:n{##2}}}{\ols_trim:n{##2}}
      }
    }{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3}{s +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\olsProcessArgs{\ols_trim:n{##2}}}{\ols_trim:n{##2}}
      }
    }
  }
  \str_if_eq:eeT{\olsTargetLanguage}{#3}{
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#4}{s +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:n{\olsProcessArgs{##3}}}{\ols_trim:n{##3}}
```

```
      }
    }{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3}{s +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:n{\olsProcessArgs{##3}}}{\ols_trim:n{##3}}
      }
    }
  }
}
```

Nun das Gleiche nochmal für *drei* Sprachen.

```
 % #1: prefix, #2...#4: selectable languages, #5: base name
\NewDocumentCommand{\olsMakeTrilangualMacro}{O{\ols_sprefix} m m m o}{
  \str_if_eq:eeT{\olsTargetLanguage}{#2} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##2}}
      }
    }{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##2}}
      }
    }
  }
  \str_if_eq:eeT{\olsTargetLanguage}{#3} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##3}}
      }
    }{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##3}}
      }
    }
  }
  \str_if_eq:eeT{\olsTargetLanguage}{#4} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##4}}
```

```
      }
    }{
      \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
        \UseHook{langselect/language}{\ols_mapped_target_lang}
        \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##4}}
      }
    }
  }
}
```

Falls die automatische Erzeugung nicht deaktivert ist, werden die konkreten Sprachmakros angelegt. Der Name sind \⟨*Prefix*⟩⟨*Sprache*$_1$⟩⟨*Sprache*$_2$⟩ bzw. \⟨*Prefix*⟩⟨*Sprache*$_1$⟩⟨*Sprache*$_2$⟩⟨*Sprache*$_2$⟩.

```
\bool_if:cTF{ols_generate}{
  \bool_if:cTF{ols_trilang}{
    \olsMakeTrilangualMacro{\olsFrstLanguage}{\olsScndLanguage}{\olsThrdLanguage}
  }{
    \olsMakeBilangualMacro{\olsFrstLanguage}{\olsScndLanguage}
  }
}{
  \PackageInfoNoLine{\packagename}{No~language~macros~generated.\MessageBreak
  Use~\string\olsMakeBilangualMacro~or~\string\olsMakeTrilangualMacro~to~generate.}
}
```

Für die Sternvarianten der Sprachmakros legen wir uns ein Makro an, dass die folgenden Argumente verarbeitet.

```
\seq_new:N \ols_args_seq
\int_new:N \ols_num_of_args
\tl_new:N \ols_format_str_tl
```

Da das Platzhalterzeichen @ im folgendem Code als Literal auftritt, muss es den gleichen Catcode wie im Dokument haben.

```
\makeatother
\NewDocumentCommand \olsProcessArgs { +m } {
  \tl_gset:Nn \ols_format_str_tl{#1}
  \int_zero:N \ols_num_of_args
```

Suche den höchsten Wert @1, für den @@1 im Formatstring vorkommt.

```
  \int_step_inline:nnnn { 9 } { -1 } { 1 }
  {
    \regex_if_match:nnT {@##1}{ #1 }
```

```
    {
      \int_set:Nn \ols_num_of_args { ##1 } \prg_break:
    }
  }
```

Wenn kein Platzhalterzeichen gefunden wurde, gib den Text in Originalform aus. Ansonsten wird das Ersetzungmakro gerufen.

```
  \int_compare:nNnTF {\ols_num_of_args} = { 0 }
  { #1 }{
    \seq_clear:N \ols_args_seq
    \ols_collect_replace:n \ols_num_of_args
  }
}
```

@1 erhält *ein* Argument übergeben, nämlich die Anzahl der verbleibenden Ersetzungstexte. Es konsumiert aber *zwei* Argumente. Dieses zweite holt es sich aus dem nach dem Makro liegenden Tokenstrom, wo es den jeweilig nächsten Ersetzungstext findet. Dieser wird in einer Tokensequent gespeichert und @1 erneut aufgerufen.

```
\cs_new_protected:Npn \ols_collect_replace:n #1 #2{
  \IfHookEmptyTF{langselect/argument}{
    \seq_put_right:Nn \ols_args_seq { #2 }
  }{
    \seq_put_right:Nn \ols_args_seq {
      \UseHook{langselect/argument}{\ols_mapped_target_lang}{#2}
    }
  }
  \int_compare:nNnTF { #1 } = { 1 }
  {
```

Wenn alle Ersetzungstexte eingelesen wurden, werden sie nacheinander auf die Platzhalterausdrücke angewendet.

```
    \int_step_inline:nn { \ols_num_of_args }
    {
      \tl_replace_all:Nnn \ols_format_str_tl
      { @##1 }
      { \seq_item:Nn \ols_args_seq { ##1 } }
    }
    \ols_format_str_tl
  }{
    \ols_collect_replace:n { \int_eval:n { #1 - 1 } }
  }
}
\makeatletter
```

```
\bool_if:NT \ols_load_babel{
  \RequirePackage[main=\ols_mapped_target_lang]{babel}
  \bool_if:nT{
    \ols_shorthands && !\str_if_p:ee{olsTargetLanguage}{de}{
      \languageshorthands{ngerman}
    }
  }
}
\cs_if_exist:cT{ver@babel.sty}{
  \AtBeginDocument{
    \selectlanguage{\ols_mapped_target_lang}
  }
}
\bool_if:NT\ols_load_polyglossia{
  \RequirePackage{polyglossia}
}
\cs_if_exist:cT{ver@polyglossia.sty}{
  \bool_if:NTF\ols_shorthands{
   \setdefaultlanguage[babelshorthands=true]{\ols_mapped_target_lang}
  }{
    \setdefaultlanguage{\ols_mapped_target_lang}
  }
}
\bool_if:NT\ols_shorthands {
  \RequirePackage[autostyle=true]{csquotes}
  \AtBeginDocument{
    \cs_if_exist:cT{useshorthands}{%
      \useshorthands*{"}
      \defineshorthand{"`}{\openautoquote}%
      \defineshorthand{"'}{\closeautoquote}%
    }
  }
}
```