Lokalisierungen der Website „kleider.herbaer.de”

Inhaltsverzeichnis

Einführung

Manche Dateien der Website hängen nicht von einer (natürlichen) Sprache ab.

Andere Dateien im Zusammenhang mit den Kalendern enthalten einfache Texte in allen angebotenen Sprachen. Wie die Übersetzungen erzeugt und wie neue Sprachen eingefügt werden, ist im Rahmen der Kalender beschrieben.

Andere Dateien nutzen die deutsche Sprache, sollen aber nicht übersetzt werden. Dazu gehören die Sitemap und die Präsentation der Software.

Die meisten Dateien aber sind in deutscher Sprache geschrieben und sollen in andere Sprachen übersetzt werden.

Zu übersetzende Dateien

Die zu übersetzenden XML-Dateien im Dokumenten-Verzeichnis werden daran erkannt, dass der Dateiname auf `.de.` endet.

Die Verarbeitung der XML-Dateien hängt vom XML-Namensraum des Wurzelelements und vom Basis-Dateinamen der XSLT-Transformation ab. Beide Informationen liefert die Transformation `xmlnsss.xslt`. Jedem der drei betroffenen XML-Namensräume ist eine Typbezeichnung *`TYPE`* zugeordnet:

`TYPE`	**XML-Namensraum**
`xhtml`	http://www.w3.org/1999/xhtml
`local`	http://herbaer.de/xmlns/20141210/localization
`imgshow`	http://herbaer.de/xmlns/20080705/imgshow

Der Basisname *`XSLNAME`* der XSLT-Transformation ist der Dateiname ohne das Suffix `.xslt`: Lautet die XSLT-Anweisung zum Beispiel

```
<?xml-stylesheet href = "../style/imgshow.xslt" type = "application/xml"?>
```

so ist *`XSLNAME`* `imgshow`. Wenn es keine XLST-Anweisung gibt, dann ist *`XSLNAME`* die leere Zeichenkette.

Die Übersetzung

Man kann sich den Übersetzungsvorgang wie Zwiebelschalen vorstellen: Die äußere Schale sind die Vor- und Nachbereitung, die zweite Schale ist der „XLIFF-Rahmen”, die dritte Schale die Übersetzung der XLIFF-Dateien und der Kern die Übersetzung einzelner Textteile.

Das Bash-Skript `localize` unterstützt die Übersetzung.

xmlns.xslt

[Quelltext]

Namensräume

Präfix	Namensraum
xml	http://www.w3.org/XML/1998/namespace
d	http://herbaer.de/xmlns/20051201/doc
xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	text
Encoding	utf-8

Eingebundene Stylesheets

/pool/txt.xslt - Hilfsvorlagen zur Ausgabe und Verarbeitung von Text

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage /processing-instruction()[name(.) = 'xml-stylesheet']

Basis-Name der XSLT-Transformation

Aufgerufene benannte Vorlagen:

txt.substring_afterlast

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  XML-Namensraum des Wurzelelements und XML-Stylesheet-Basisname
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  version   = "1.0"
>
<xsl:include href = "/pool/txt.xslt"/>


<xsl:output method = "text" encoding = "utf-8"/>


<xsl:template match = "/">
  <xsl:value-of select = "namespace-uri(*)"/>
  <xsl:text>|</xsl:text>
  <xsl:apply-templates select = "processing-instruction()[name(.) = 'xml-stylesheet']"/>
</xsl:template>


<xsl:template match = "/processing-instruction()[name(.) = 'xml-stylesheet']">
  <xsl:variable name = "n">
    <xsl:call-template name = "txt.substring_afterlast">
      <xsl:with-param name = "txt" select = "
        substring-before (
          substring-after (
            substring-after (., 'href'),
          '&quot;'),
       '&quot;')
      "/>
      <xsl:with-param name = "what" select = "'/'"/>
    </xsl:call-template>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test = "contains ($n, '.xslt')">
      <xsl:value-of select = "substring-before ($n, '.xslt')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select = "$n"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


</xsl:stylesheet>
```

Vor- und Nachbereitung der sprachabhängigen XML-Dateien

Vor der Übersetzung kann eine Datei eine oder zwei XSLT-Transformationen durchlaufen. Falls die Datei *XSLNAME*_pre.xslt existiert, wird diese Transformation angewandt. Falls die Datei *TYPE*_pre.xslt existiert, wird (anschließend) diese Transformation angewandt. Die XSLT-Dateien werden im Skript-Verzeichnis gesucht.

Es gibt die folgenden vorbereitenden Transformationen:

```
desktop_help_pre.xslt
imgshow_pre.xslt
```

Nach der Übersetzung im engeren Sinne kann die übersetzte Datei noch eine oder zwei XSLT-Transformationen durchlaufen. Falls die Datei *TYPE*_post.xslt existiert, wird diese Transformation angewandt. Falls die Datei *XSLNAME*_pre.xslt existiert, wird (anschließend) diese Transformation angewandt. Die XSLT-Dateien werden im Skript-Verzeichnis gesucht.

Es gibt die folgenden nachbereitenden Transformationen:

```
desktop_help_post.xslt
imgshow_post.xslt
xhtml_post.xslt
```

Den nachbereitenden Transformationen werden die folgenden Zeichenketten-Parameter übergeben:

Name	**Wert**
p_localsrc	Pfad der Lokalisierungsdatei der Quellsprache
p_localweb	Pfad der Lokalisierungsdatei der Zielsprache

Das Perl-Programm rmxmlns.pl entfernt nicht benötigte XML-Namensraumknoten, die sich mittels XSLT nicht sicher entfernen lassen.

desktop_help_pre.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://www.w3.org/1999/xhtml
*	d	http://herbaer.de/xmlns/20051201/doc
*	ht	http://www.w3.org/1999/xhtml
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage *

Elemente werden "hohl" kopiert, nicht-leere Bildverweise werden am Ende des Inhalts als leere Elemente angefügt.

Verwendete Modus:

append

Muster-Vorlage processing-instruction() | @*

Verarbeitungsanweisungen und Attribute werden kopiert

Muster-Vorlage ht:span [@class= 'imglink' and string-length(.) > 0]

Nicht-leere Bildverweise werden aus dem laufenden Text entfernt.

Muster-Vorlage ht:span, append

Nicht-leere Bildverweise werden ohne Inhalt am Ende des Inhalts des Elternelements angehängt und bekommen das Attribut `@class = "ximglink"`.

Verwendete Modus:

append

Muster-Vorlage @class, append

Bildverweise, die in der Quelle nicht leer sind, bekommen im Modus "append" das Attribut `@class = "ximglink"`.

Muster-Vorlage @*, append

Andere Attribute werden kopiert.

Modus

Modus append

Die folgenden Vorlagen implementieren den Modus append:

Muster-Vorlage ht:span, append
Muster-Vorlage @class, append
Muster-Vorlage @*, append

Der Modus append wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage *
Muster-Vorlage ht:span, append

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  file KLEIDER/web/src/localization/desktop_help_pre.xslt
  Die Hilfe zur Darstellung "desktop" für die Übersetzung vorbereiten
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:ht  = "http://www.w3.org/1999/xhtml"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns     = "http://www.w3.org/1999/xhtml"
  exclude-result-prefixes = "d ht"
  version   = "1.0"
>
<xsl:output method = "xml" encoding = "utf-8"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "text() | processing-instruction() | comment() | *"/>
</xsl:template>


<xsl:template match = "*">
  <xsl:copy>
    <xsl:apply-templates select = "@* | * | text()"/>
    <xsl:apply-templates
        select = "ht:span [@class= 'imglink' and string-length(.) &gt; 0]"
        mode   = "append"
    />
  </xsl:copy>
</xsl:template>


<xsl:template match = "processing-instruction() | @*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "ht:span [@class= 'imglink' and string-length(.) &gt; 0]">
  <xsl:apply-templates/>
</xsl:template>


<xsl:template match = "ht:span" mode = "append">
  <xsl:copy>
    <xsl:apply-templates select = "@*" mode = "append"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "@class" mode = "append">
  <xsl:attribute name = "class">ximglink</xsl:attribute>
</xsl:template>


<xsl:template match = "@*" mode = "append">
  <xsl:copy-of select = "."/>
</xsl:template>


</xsl:stylesheet>
```

imgshow_pre.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://herbaer.de/xmlns/20080705/imgshow
*	d	http://herbaer.de/xmlns/20051201/doc
*	is	http://herbaer.de/xmlns/20080705/imgshow
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage *

Elemente werden "hohl" kopiert.

Muster-Vorlage processing-instruction() | @*

Verarbeitungsanweisungen und Attribute werden kopiert

Muster-Vorlage is:p

Bildverweise werden aus dem laufenden Text entfernt und am Ende des Abschnitts angefügt.

Verwendete Modus:

append

Muster-Vorlage is:jpg

Bildverweise werden aus dem laufenden Text entfernt.

Muster-Vorlage is:jpg, append

Bildverweise werden ohne Inhalt am Ende des Abschnitts angefügt.

Muster-Vorlage is:jpg/@alt

Das Wort `wttl` wird aus dem Wert des Attributs `is:jpg/@alt` entfernt.

Modus

Modus append

Die folgenden Vorlagen implementieren den Modus append:

Muster-Vorlage is:jpg, append

Der Modus append wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage is:p

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  file KLEIDER/web/src/localization/imgshow_pre.xslt
  Eine Bildergeschichte für die Übersetzung vorbereiten
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:is  = "http://herbaer.de/xmlns/20080705/imgshow"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns     = "http://herbaer.de/xmlns/20080705/imgshow"
  exclude-result-prefixes = "d is"
  version   = "1.0"
>
<xsl:output method = "xml" encoding = "utf-8"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "text() | processing-instruction() | comment() | *"/>
</xsl:template>


<xsl:template match = "*">
  <xsl:copy>
    <xsl:apply-templates select = "@* | * | text()"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "processing-instruction() | @*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "is:p">
  <xsl:copy>
    <xsl:copy-of select = "@*"/>
    <xsl:apply-templates/>
    <xsl:apply-templates select = "is:jpg" mode = "append"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "is:jpg">
  <xsl:apply-templates/>
</xsl:template>


<xsl:template match = "is:jpg" mode = "append">
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "is:jpg/@alt">
  <xsl:attribute name = "alt">
    <xsl:choose>
      <xsl:when test = "contains (., 'wttl')">
        <xsl:value-of select = "normalize-space (
          concat (substring-before (., 'wttl'), ' ', substring-after (., 'wttl'))
        )"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select = "."/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
</xsl:template>


</xsl:stylesheet>
```

desktop_help_post.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://www.w3.org/1999/xhtml
*	d	http://herbaer.de/xmlns/20051201/doc
	ti	http://herbaer.de/xmlns/201500703/transinfo/
*	sk	http://herbaer.de/xmlns/20150106/skeleton
*	ht	http://www.w3.org/1999/xhtml
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage *

Elemente werden "hohl" kopiert. Verschobene Elemente werden entfernt, nur verschobene Verweise werden am Ende angefügt.

Muster-Vorlage processing-instruction() | @*

Verarbeitungsanweisungen und Attribute werden kopiert

Muster-Vorlage ht:span [@class = 'ximglink']

Verschobene Bildverweise werden mit Inhalt gefüllt

Verwendete Modus:

ximglink

Muster-Vorlage @*, ximglink

Attribute verschobener Bildverweise werden kopiert.

Muster-Vorlage @class, ximglink

Nur das Attribut `class` bekommt seinen alten Wert `imglink`

Modus

Modus ximglink

Die folgenden Vorlagen implementieren den Modus ximglink:

Muster-Vorlage @*, ximglink
Muster-Vorlage @class, ximglink

Der Modus ximglink wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage ht:span [@class = 'ximglink']

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  file KLEIDER/web/src/localization/desktop_help_post.xslt
  Hilfe zur Darstellung "desktop" nach der Übersetzung aufbereiten
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:ht  = "http://www.w3.org/1999/xhtml"
  xmlns:sk  = "http://herbaer.de/xmlns/20150106/skeleton"
  xmlns:ti  = "http://herbaer.de/xmlns/201500703/transinfo/"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns     = "http://www.w3.org/1999/xhtml"
  exclude-result-prefixes = "d ht sk"
  version   = "1.0"
>
<xsl:output method = "xml" encoding = "utf-8"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "text() | processing-instruction() | comment() | *"/>
</xsl:template>


<xsl:template match = "*">
  <xsl:copy>
    <xsl:apply-templates select = "@* | * | text()"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "processing-instruction() | @*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "ht:span [@class = 'ximglink']">
  <xsl:text> </xsl:text>
  <xsl:copy>
    <xsl:apply-templates select = "@*" mode = "ximglink"/>
    <xsl:value-of
      select = "concat ('[', count (preceding::ht:span [@class = 'ximglink']) + 1, ']')"
    />
  </xsl:copy>
</xsl:template>


<xsl:template match = "@*" mode = "ximglink">
  <xsl:copy/>
</xsl:template>


<xsl:template match = "@class" mode = "ximglink">
  <xsl:attribute name = "class">imglink</xsl:attribute>
</xsl:template>


</xsl:stylesheet>
```

imgshow_post.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://herbaer.de/xmlns/20080705/imgshow
*	d	http://herbaer.de/xmlns/20051201/doc
*	ti	http://herbaer.de/xmlns/201500703/transinfo/
*	sk	http://herbaer.de/xmlns/20150106/skeleton
*	l	http://herbaer.de/xmlns/20141210/localization
*	is	http://herbaer.de/xmlns/20080705/imgshow
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8

Parameter

Parameter p_localsrc

Dateipfad des Quelltextes der Lokalisierungsdatei

Select: '../style/local.xml.de'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Variable g_locsrcroot

Parameter p_localweb

Dateipfad der übersetzten Lokalisierungsdatei

Select: '../../docroot/local/local.xml.de.'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Variable g_locwebroot

Globale Variable

Variable g_locsrcroot

Wurzelelement des Quelltextes der Lokalisierungsdatei

Select: document($p_localsrc)/*

Verwendete globale Parameter oder Variable:

Parameter p_localsrc

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Benannte Vorlage gettext

Variable g_locwebroot

Wurzelelement der übersetzten Lokalisierungsdatei

Select: document($p_localweb)/*

Verwendete globale Parameter oder Variable:

Parameter p_localweb

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Benannte Vorlage gettext

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage *

Elemente werden "hohl" kopiert.

Muster-Vorlage sk:part

`sk:part`-Elemente dürfte es nicht geben, wenn doch, werden sie ignoriert.

Muster-Vorlage is:f

`f`-Elemente werden aus dem übersetzten Dokument entfernt.

Muster-Vorlage processing-instruction() | @*

Verarbeitungsanweisungen und Attribute werden kopiert

Muster-Vorlage @ti:hint

Hinweise zur Übersetzung werden entfernt.

Muster-Vorlage is:jpg

Bildverweise werden mit Inhalt gefüllt

Muster-Vorlage is:puppe [@ti:hint = 'moved'] | is:peruecke [@ti:hint = 'moved'] | is:kleid [@ti:hint = 'moved']

Verweise auf eine Puppe, eine Perücke oder ein Kleid sollte es im Web nicht geben, trotzdem werden sie hier abgefangen.

Muster-Vorlage is:story [@ti:hint = 'moved']

Verschobener Verweis auf eine Bildergeschichte

Aufgerufene benannte Vorlagen:

gettext

Muster-Vorlage is:sect [@ti:hint = 'moved']

Verschobener Verweis auf einen Textabschnitt

Aufgerufene benannte Vorlagen:

gettext

Benannte Vorlagen

Benannte Vorlage gettext

Parameter

id

ID im Lokalisierungs-Quelltext

Übersetzter Text anhand der ID im Quelltext der Lokalisierungsdatei

Die Vorlage wird aufgerufen in:

Muster-Vorlage is:story [@ti:hint = 'moved']
Muster-Vorlage is:sect [@ti:hint = 'moved']

Verwendete globale Parameter oder Variable:

Variable g_locsrcroot
Variable g_locwebroot

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  file KLEIDER/web/src/localization/imgshow_post.xslt
  Eine Bildergeschichte nach der Übersetzung aufbereiten
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:is  = "http://herbaer.de/xmlns/20080705/imgshow"
  xmlns:l   = "http://herbaer.de/xmlns/20141210/localization"
  xmlns:sk  = "http://herbaer.de/xmlns/20150106/skeleton"
  xmlns:ti  = "http://herbaer.de/xmlns/201500703/transinfo/"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns     = "http://herbaer.de/xmlns/20080705/imgshow"
  exclude-result-prefixes = "d is l sk ti"
  version   = "1.0"
>
<xsl:param name = "p_localsrc" select = "'../style/local.xml.de'"/>


<xsl:param name = "p_localweb" select = "'../../docroot/local/local.xml.de.'"/>


<xsl:variable name = "g_locsrcroot" select = "document($p_localsrc)/*"/>


<xsl:variable name = "g_locwebroot" select = "document($p_localweb)/*"/>


<xsl:output method = "xml" encoding = "utf-8"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "text() | processing-instruction() | comment() | *"/>
</xsl:template>


<xsl:template match = "*">
  <xsl:copy>
    <xsl:apply-templates select = "@* | *[not (@ti:hint = 'moved')] | text()"/>
    <xsl:apply-templates select = "* [@ti:hint = 'moved']"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "sk:part">
  <xsl:apply-templates select = "@* | *[not (@ti:hint = 'moved')] | text()"/>
  <xsl:apply-templates select = "* [@ti:hint = 'moved']"/>
</xsl:template>


<xsl:template match = "is:f">
  <xsl:apply-templates/>
</xsl:template>


<xsl:template match = "processing-instruction() | @*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "@ti:hint"/>


<xsl:template match = "is:jpg">
  <xsl:text> </xsl:text>
  <xsl:copy>
    <xsl:copy-of select = "@*"/>
    <xsl:value-of select = "concat ('[', count (preceding::is:jpg) + 1, ']')"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "
    is:puppe    [@ti:hint = 'moved']
  | is:peruecke [@ti:hint = 'moved']
  | is:kleid    [@ti:hint = 'moved']"
>
  <xsl:text> </xsl:text>
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:text>[</xsl:text>
    <xsl:value-of select = "@ref"/>
    <xsl:text>]</xsl:text>
  </xsl:copy>
</xsl:template>
```

```
<xsl:template match = "is:story [@ti:hint = 'moved']">
  <xsl:text> </xsl:text>
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:text>[</xsl:text>
    <xsl:call-template name = "gettext">
      <xsl:with-param name = "id">bildergeschichte</xsl:with-param>
    </xsl:call-template>
    <xsl:text> </xsl:text>
    <xsl:choose>
      <xsl:when test = "contains (@ref, '#')">
        <xsl:value-of select = "substring-before (@ref, '#')"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select = "@ref"/>
      </xsl:otherwise>
    </xsl:choose>
    <xsl:text>]</xsl:text>
  </xsl:copy>
</xsl:template>


<xsl:template match = "is:sect [@ti:hint = 'moved']">
  <xsl:text> </xsl:text>
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:text>[</xsl:text>
    <xsl:call-template name = "gettext">
      <xsl:with-param name = "id">textabschnitt</xsl:with-param>
    </xsl:call-template>
    <xsl:text> </xsl:text>
    <xsl:choose>
      <xsl:when test = "contains (@ref, '_')">
        <xsl:value-of select = "substring-before (@ref, '_')"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select = "@ref"/>
      </xsl:otherwise>
    </xsl:choose>
    <xsl:text>]</xsl:text>
  </xsl:copy>
</xsl:template>


<xsl:template name = "gettext">


  <xsl:param name = "id"/>
  <xsl:variable name = "pos"
    select = "count ($g_locsrcroot/l:t[@id = $id]/preceding-sibling::l:t) + 1"
  />
  <xsl:value-of select = "$g_locwebroot/l:t[position() = $pos]"/>
</xsl:template>


</xsl:stylesheet>
```

xhtml_post.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	(default)	http://www.w3.org/1999/xhtml
*	ti	http://herbaer.de/xmlns/201500703/transinfo/
*	sk	http://herbaer.de/xmlns/20150106/skeleton
*	l	http://herbaer.de/xmlns/20141210/localization
*	ht	http://www.w3.org/1999/xhtml
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8

Eingebundene Stylesheets

/pool/txt.xslt - Hilfsvorlagen zur Ausgabe und Verarbeitung von Text

Vorlage `txt.split`

/pool/list.xslt - Vorlagen zur Arbeit mit Listen

Vorlage `list.map_once`

Diese Datei enthält benannte Hilfsvorlagen, die die Einträge einer Liste akkumulieren, zu einer neuen Liste bearbeiten, auf zwei Listen aufteilen oder sortieren. Sie nutzt Vorlage aus `txt.xslt`. Wenn ein Stylesheet diese Datei (`list.xslt`) einbindet, sollte es daher auch `txt.xslt` einbinden.

Eine Liste ist eine Zeichenkette, die durch eine feste Zeichenfolge (die Trennzeichenfolge) in einzelne Listeneinträge zerlegt wird. Einige Vorlagen behandeln die leere Zeichenkette als Listeneintrag so, als gäbe es diesen Eintrag nicht. Ein Listeneintrag sollte also, so wie Listen hier behandelt werden, nicht leer sein.

Die Trennzeichenfolge kann jeder Vorlage für jede Liste als Parameter übergeben werden. Wenn eine Vorlage mit mehreren Listen arbeitet, können die Listen verschiedene Trennzeichenfolgen verwenden. Voreingestellt ist ein einzelnes Leerzeichen.

Die Listeneinträge werden von 1 an gezählt (Position eines Listeneintrags).

Konkrete Funktionen für einzelne Listeneinträge werden ausgeführt, indem die Vorlagen im Modus `list.apply` auf einen Parameter ("Funktional") angewandt werden. Typischerweise ist der Parameter eine Vorlage, die auf sich

selbst im Modus `list.apply` passt. Zu jedem "Funktional"-Parameter gibt es einen weiteren Parameter (meist mit dem Namen "*`param`*", der an die Vorlagen im Modus `list.apply` übergeben wird.

Parameter

Parameter p_localsrc

Dateipfad des Quelltextes der Lokalisierungsdatei

Select: '../style/local.xml.de'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Variable g_locsrcroot

Parameter p_localweb

Dateipfad der übersetzten Lokalisierungsdatei

Select: '../../docroot/local/local.xml.de.'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Variable g_locwebroot

Parameter p_excptxslt

Einige XSLT-Transformationen, die auf die übersetzte Datei angewandt werden, beachten nicht die Hinweise zur Übersetzung. Diese Basisnamen dieser Transformationen (Dateinamen ohne das Suffix `.xslt`) sind hier aufgelistet, getrennt durch das Zeichen "`,`". Im Falle einer solchen Transformation wird hier ein Übersetzungshinweis eingefügt.

Select: 'v'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Variable g_handle_trhint

Parameter p_transnames

Das Attribut @ti:machine enthält einen Hinweis auf die benutzte Übersetzungs-Maschine. Wenn der Wert des Attributs eine der nachfolgend aufgeführten Bezeichnungen als Teilzeichenkette enthält, wird die erste passende Bezeichnung als neuer Attributwert ausgegeben. Sonst bleibt das Attribut unverändert, falls es nicht besonders behandelt wird. Dieser Parameter ist eine Komma-getrennte Liste der Bezeichnungen

Select: 'google,mysql'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage @ti:machine

Globale Variable

Variable g_locsrcroot

Wurzelelement des Quelltextes der Lokalisierungsdatei

Select: document($p_localsrc)/*

Verwendete globale Parameter oder Variable:

Parameter p_localsrc

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Benannte Vorlage gettext

Variable g_locwebroot

Wurzelelement der übersetzten Lokalisierungsdatei

Select: document($p_localweb)/*

Verwendete globale Parameter oder Variable:

Parameter p_localweb

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Benannte Vorlage gettext

Variable g_handle_trhint

Verweist das Quelldokument auf eine XSLT-Transformation, die einen Übersetzungshinweis verarbeitet? Dann braucht der Übersetzunghinweis hier nicht behandelt zu werden.

Aufgerufene benannte Vorlagen:

txt.replacechars
txt.basefilename

Verwendete globale Parameter oder Variable:

Parameter p_excptxslt

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage @ti:machine
Muster-Vorlage ht:body

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage *

Elemente werden "hohl" kopiert. Verschobene Elemente werden entfernt, nur verschobene Verweise werden am Ende angefügt.

Muster-Vorlage ht:a [@ti:hint = 'moved']

Ein verschobener Verweis

Muster-Vorlage ht:* [@ti:hint = 'moved']

Andere HTML-Elemente, die aus dem Textfluss herausgenommen sind, werden ignoriert.

Muster-Vorlage sk:part

`sk:part`-Elemente werden zu `span`-Elementen

Muster-Vorlage processing-instruction() | @*

Verarbeitungsanweisungen und Attribute werden kopiert

Muster-Vorlage @ti:machine

ti:machine - Attribute werden nur dann kopiert, wenn der Übersetzunghinweis nicht in dieser Transformation verarbeitet wird.

Aufgerufene benannte Vorlagen:

list.map_once

Verwendete globale Parameter oder Variable:

Parameter p_transnames
Variable g_handle_trhint

Muster-Vorlage @ti:machine, list.apply

Parameter

item

eine Bezeichnung eines Übersetzers

Die passende Bezeichnung finden

Muster-Vorlage @ti:*

Andere ti:* - Attribute werden nicht kopiert.

Muster-Vorlage @translate

Auch das HTML-Attribut `translate` wird nicht kopiert.

Muster-Vorlage ht:body

In das `body`-Element wird ein Verweis auf die Übersetzungsmaschine eingefügt, wenn das Dokument nicht auf ein Stylesheet verweist, das den Übersetzungshinweis verarbeitet.

Verwendete Modus:

hint

Verwendete globale Parameter oder Variable:

Variable g_handle_trhint

Muster-Vorlage @ti:machine, hint

Hinweis auf die maschinelle Übersetzung

Aufgerufene benannte Vorlagen:

gettext

Benannte Vorlagen

Benannte Vorlage gettext

Parameter

id

ID im Lokalisierungs-Quelltext

Übersetzten Text anhand der ID im Quelltext der Lokalisierungsdatei

Die Vorlage wird aufgerufen in:

Muster-Vorlage @ti:machine, hint

Verwendete globale Parameter oder Variable:

Variable g_locsrcroot
Variable g_locwebroot

Modus

Modus list.apply

Die folgenden Vorlagen implementieren den Modus list.apply:

Muster-Vorlage @ti:machine, list.apply

Modus hint

Die folgenden Vorlagen implementieren den Modus hint:

Muster-Vorlage @ti:machine, hint

Der Modus hint wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage ht:body

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  Nachbearbeitung übersetzter XHTML-Dateien
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
  2016-02-10 g_handle_trhint, p_excptxslt, txt.xslt
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:ht  = "http://www.w3.org/1999/xhtml"
  xmlns:l   = "http://herbaer.de/xmlns/20141210/localization"
  xmlns:sk  = "http://herbaer.de/xmlns/20150106/skeleton"
  xmlns:ti  = "http://herbaer.de/xmlns/201500703/transinfo/"
  xmlns     = "http://www.w3.org/1999/xhtml"
  exclude-result-prefixes = "d ht l ti sk"
  version   = "1.0"
>
<xsl:param name = "p_localsrc" select = "'../style/local.xml.de'"/>


<xsl:param name = "p_localweb" select = "'../../docroot/local/local.xml.de.'"/>


<xsl:param name = "p_excptxslt" select ="'v'"/>


<xsl:param name = "p_transnames" select = "'google,mysql'"/>


<xsl:variable name = "g_locsrcroot" select = "document($p_localsrc)/*"/>


<xsl:variable name = "g_locwebroot" select = "document($p_localweb)/*"/>


<xsl:variable name = "g_handle_trhint">
  <xsl:choose>
    <xsl:when test = "/processing-instruction() [name(.) = 'xml-stylesheet']">
      <!--
          Im Wert der Verarbeitungsanweisung werden vor und nach dem Zeichen '='
          Leerzeichen eingefügt.
      -->
      <xsl:variable name = "pi">
        <xsl:call-template name = "txt.replacechars">
          <xsl:with-param name = "txt"   select = "/processing-instruction() [name(.) = 'xml-stylesheet']"
          />
          <xsl:with-param name = "chars" select = "'='"/>
          <xsl:with-param name = "list"  select = "' = '"/>
        </xsl:call-template>
      </xsl:variable>


      <xsl:variable name = "basefn">
        <xsl:text>,</xsl:text>
        <xsl:call-template name = "txt.basefilename">
          <xsl:with-param name = "txt" select = "
            substring-before (
              substring-after (normalize-space ($pi), 'href = &amp;quot;'),
            '&amp;quot;')
          "/>
        </xsl:call-template>
        <xsl:text>,</xsl:text>
      </xsl:variable>
      <!--
        Liste der "Ausnahme-Transformationen" ohne Leerzeichen mit zusätzlichen Kommata
      -->
      <xsl:variable name = "exlist"
        select = "concat (',', translate ($p_excptxslt, ' ', ''), ',')"
      />
      <xsl:choose>
        <xsl:when test = "contains ($exlist, $basefn)">yes</xsl:when>
        <xsl:otherwise>no</xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>yes</xsl:otherwise>
  </xsl:choose>
</xsl:variable>


<xsl:include href = "/pool/txt.xslt"/>
```

```
<xsl:include href = "/pool/list.xslt"/>


<xsl:output method = "xml" encoding = "utf-8"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "text() | processing-instruction() | comment() | *"/>
</xsl:template>


<xsl:template match = "*">
  <xsl:copy>
    <xsl:apply-templates select = "@* | *[not (@ti:hint = 'moved')] | text()"/>
    <xsl:apply-templates select = "* [@ti:hint = 'moved']"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "ht:a [@ti:hint = 'moved']">
  <xsl:text> </xsl:text>
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:text>[</xsl:text>
    <xsl:apply-templates select = "* | text()"/>
    <xsl:text>]</xsl:text>
  </xsl:copy>
</xsl:template>


<xsl:template match = "ht:* [@ti:hint = 'moved']"/>


<xsl:template match = "sk:part">
  <span>
    <xsl:apply-templates select = "@* | * | text()"/>
  </span>
</xsl:template>


<xsl:template match = "processing-instruction() | @*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "@ti:machine">
  <xsl:if test = "$g_handle_trhint != 'yes'">
    <xsl:attribute name = "ti:machine">
      <xsl:call-template name = "list.map_once">
        <xsl:with-param name = "list"   select = "$p_transnames"/>
        <xsl:with-param name = "mapper" select = "."/>
        <xsl:with-param name = "sep"    select = "','"/>
      </xsl:call-template>
    </xsl:attribute>
  </xsl:if>
</xsl:template>


<xsl:template match = "@ti:machine" mode = "list.apply">


  <xsl:param name = "item"/>
  <xsl:if test = "contains (., $item)">
    <xsl:value-of select = "$item"/>
  </xsl:if>
</xsl:template>


<xsl:template match = "@ti:*"/>


<xsl:template match = "@translate"/>
```

```
<xsl:template match = "ht:body">
  <xsl:copy>
  <xsl:apply-templates select = "@*"/>
  <xsl:choose>
    <xsl:when test = "$g_handle_trhint != 'yes'">
      <xsl:apply-templates select = "*"/>
    </xsl:when>
    <xsl:when test = "starts-with (local-name(*[1]), 'h')">
      <xsl:apply-templates select = "*[1]"/>
      <xsl:apply-templates select = "/*/@ti:machine" mode = "hint"/>
      <xsl:apply-templates select = "*[position() &gt; 1]"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates select = "/*/@ti:machine" mode = "hint"/>
      <xsl:apply-templates select = "*"/>
    </xsl:otherwise>
  </xsl:choose>
  </xsl:copy>
</xsl:template>


<xsl:template match = "@ti:machine" mode = "hint">
  <div class = "machine">
    <xsl:choose>
      <xsl:when test = "contains (., 'google')">
        <p>
          <a href = "http://translate.google.com" class = "machine_link" target = "_blank"
          >
            <img src = "/local/trans_by_google.png"/>
          </a>
        </p>
      </xsl:when>
      <xsl:when test = "contains (., 'mysql')">
        <p>
          <a href = "http://translate.google.com" class = "machine_link" target = "_blank"
          >
            <img src = "/local/trans_by_google.png"/>
          </a>
        </p>
      </xsl:when>
    </xsl:choose>
    <p>
      <xsl:call-template name = "gettext">
        <xsl:with-param name = "id" select = "'maschinelle_uebersetzung'"/>
      </xsl:call-template>
    </p>
  </div>
</xsl:template>


<xsl:template name = "gettext">


  <xsl:param name = "id"/>
  <xsl:variable name = "pos"
    select = "count ($g_locsrcroot/l:t[@id = $id]/preceding-sibling::l:t) + 1"
  />
  <xsl:value-of select = "$g_locwebroot/l:t[position() = $pos]"/>
</xsl:template>


</xsl:stylesheet>
```

Der XLIFF-Rahmen

Der Quelldatei im Server-Verzeichnis *WEBDIR*/docroot werden zusätzliche Markierungen hinzugefügt. Die Markierungen kennzeichnen die zu übersetzenden Textteile und ordnen ihnen ID-Werte zu. Das Ergebnis ist eine Gerüst-Datei (XML-Namensraum http://herbaer.de/xmlns/20150106/skeleton, s. skeleton.rng) im Lokalisierung-Arbeitsverzeichnis *WEBDIR*/local statt des Server-Verzeichnisses mit dem zusätzlichen Suffix .skl.

Die Gerüst-Datei wird durch zwei Transformationen erzeugt. Die erste Transformation *TYPE*_clfy.xslt hängt vom XML-Namensraum ab. Die verschiedenen Varianten der ersten Transformation (imgshow_clfy.xslt, local_clfy.xslt, xhtml_clfy.xslt) unterscheiden sich durch die Klassifizierung von Elementen und Attributen bezüglich der Übersetzung. Sie binden den gemeinsamen Teil skeleton_1.xslt ein. Die zweite Transformation ist skeleton_2.xslt.

Aus der Gerüst-Datei wird eine XLIFF-Datei mit den zu übersetzenden Texten erzeugt. Das XLIFF-Dateiformat (s. OASIS XLIFFF TC, XLIFF Version 2 [http://docs.oasis-open.org/xliff/xliff-core/v2.0/os/xliff-core-v2.0-os.html]) ist zum Austausch von Übersetzungsaufträgen zwischen Übersetzungssoftware (zur automatischen Übersetzung oder zur Unterstützung der menschlichen Übersetzung) entwickelt. Die XLIFF-Datei hat das Suffix .xlf statt .skl. Die XLIFF-Datei wird durch die aufeinanderfolgenden Transformationen skeleton_xliff.xslt und xliff_clean.xslt erzeugt.

Die XLIFF-Datei wird in verschiedene Sprachen übersetzt. In den Dateinamen der übersetzten XLIFF-Dateien ist das Suffix, das die Quellsprache bezeichnet, durch die Kennung der Zielsprache ersetzt.

Die Transformation skeleton_merge_xliff.xslt fügt die übersetzte XLIFF-Datei und die Gerüst-Datei zur übersetzten Datei zusammen. Sie wird auf die Gerüst-Datei angewandt; der Dateipfad der übersetzten XLIFF-Datei wird als Parameter übergeben. Im Dateinamen der übersetzten Datei im Server-Verzeichnis ist die Kennung der Quellsprache durch die Kennung der Zielsprache ersetzt. Die übersetzte Datei übernimmt aus der XLIFF-Datei Hinweise auf das Übersetzungsprogramm. Dazu dienen Elemente und Attribute des XML-Namensraums http://herbaer.de/xmlns/201500703/transinfo/ (s. transinfo.rng).

skeleton.rng - Markierungen zu übersetzbaren Inhalten

Ein XML-Dokument, das übersetzt werden soll, wird mit zusätzlichen Markierungen angereichert. Die zusätzlichen Markierungen kennzeichnen die zu übersetzenden Inhalte. Aus dem erweiterten Dokument ("Gerüst") kann eine XLIFF-Datei erstellt werden. Aus der übersetzten XLIFF-Datei und dem Gerüst kann eine übersetzte Version des ursprünglichen Dokuments erstellt werden. Dieses Dokument beschreibt die zusätzlichen Markierungen.

Namespace	http://herbaer.de/xmlns/20150106/skeleton
Wurzelelement	lang, attribute, contents, part, @sk:chk, @sk:wrap
(anything)	Beliebiger Inhalt *Enthält:* (anything) (*) *Enthalten in:* (anything), (foreign_el)
(foreign_att)	Attribute anderer XML-Namensräume *Enthalten in:* lang, attribute, contents, part
(foreign_el)	Elemente anderer XML-Namensräume *Enthält:* (anything) (*) *Enthalten in:* contents, part

lang	Das Element `lang` bezeichnet ein Attribut, das die Sprache bezeichnet. Im übersetzten Dokument müssen sie die Zielsprache angeben. Der Inhalt ist der Wert des Attributs (Quellsprache). Das Attribut `name` gibt den Namen des Attributs an.
	
	Enthält: Datentyp string
	
	Enthalten in: Wurzel
	
	`<element name="lang">`
	`  <ref name="foreign_att"/>`
	`  <ref name="att_name"/>`
	`  <data type="string"/>`
	`</element>`

@name	Der Name eines Attributs *Enthalten in:* lang, attribute

attribute	Das Element `attribute` bezeichnet ein Attribut, dessen Wert zu übersetzen ist. Das Attribut `name` nennt den Namen des Attributs, der Inhalt ist der Wert des Attributs im Quelldokument (der zu übersetzende Wert). Der einzige sinnvolle Wert des Attributs `class` ist hier `text`.
	
	Enthält: Text, (foreign_att), @name, @xliffid, @class (?)
	
	Enthalten in: Wurzel
	
	`<element name="attribute">`
	`  <ref name="foreign_att"/>`
	`  <ref name="att_name"/>`
	`  <ref name="att_xliffid"/>`
	`  <optional>`
	`  <ref name="att_class"/>`
	`  </optional>`
	`  <text/>`
	`</element>`

@xliffid	Eindeutige Kennung eines Textausschnitts *Enthalten in:* attribute, contents, part
@class	Klassifiziert einen Textabschnitt des Quelldokuments bezüglich der Übersetzung.

`text`

Einfacher zu übersetzender Text, der nicht weiter untergliedert ist.

`struct`

Der zu übersetzende Text enhält markierte Ausschnitte.

`move`

Die Markierung des Textinhalts kann beliebig innerhalb des Elternelements verschoben werden, solange die Reihenfolge der Markierungen erhalten bleibt.

`fix`

Der Textinhalt ist wörtlich zu übernehmen (nicht zu übersetzen), z.B. eine E-Mail-Adresse oder eine URL.

Erlaubte Werte: "text", "struct", "move", "fix"

Enthalten in: attribute, contents, part

contents

Der Inhalt eines Elements. Immer wenn ein Element ein Attribut enthält, das die Sprache angibt, oder ein Attribut oder Inhalt, der zu übersetzen ist, wird der Inhalt in ein `contents`-Element gepackt.

Enthält: Text, @class, @unit (?), @wrap (?), @xliffid, @lang, (foreign_att), (foreign_el) (|), part (+)

Enthalten in: Wurzel

```
<element name="contents">
  <ref name="att_class"/>
  <optional>
    <ref name="att_unit"/>
  </optional>
  <optional>
    <ref name="att_wrap"/>
  </optional>
  <ref name="att_xliffid"/>
  <ref name="att_lang"/>
  <ref name="foreign_att"/>
  <choice>
    <interleave>
        <text/>
        <ref name="foreign_el"/>
    </interleave>
    <interleave>
        <oneOrMore>
          <ref name="el_part"/>
        </oneOrMore>
        <ref name="foreign_el"/>
    </interleave>
  </choice>
</element>
```

@unit

Das Attribut kennzeichnet zu übersetzende Texteile, die nicht Teil eines umfassenden zu übersetzenden Textteils sind. ("Units" im Sinne von XLIFF).

Enthalten in: contents

@wrap

Das Attribut zeigt an, dass das Elternelement im Quelldokument ein einziges Kindelement "einhüllt".

Enthalten in: contents

@lang

Die Sprache eines Textteils im Quelldokument.

Enthalten in: contents, part

part

Manche Elemente im Quelldokument zerlegen den Inhalt des Elternelements in Abschnitte. Das ist übel, aber historisch so verwachsen. Die Abschnitte, die eine solche Zerlegung erzeugt, sind in `part`-Elementen enthalten.

Enthält: Text, @class, @xliffid, @lang, (foreign_att), (foreign_el)

Enthalten in: Wurzel, contents

```
<element name="part">
  <ref name="att_class"/>
  <ref name="att_xliffid"/>
  <ref name="att_lang"/>
  <ref name="foreign_att"/>
  <interleave>
    <text/>
    <ref name="foreign_el"/>
  </interleave>
</element>
```

@sk:chk

Das Attribut `sk:chk` gibt die "Erstklassifizierung" eines Elements an:

`s`

Das Element trennt den Inhalt des Elternelements in Teile, die getrennt übersetzt werden (können).

`t`

Der Inhalt des Elements einschließlich seiner Kindelemente ist zu übersetzen.

`l`

Der Inhalt des Elements ist eine unveränderliche Zeichenkette, (ein Name oder eine URL).

`m`

Der Inhalt des Elements kann direkt im Elternelement erscheinen, und das Element selbst kann mit leerem Inhalt verschoben werden, solange die Reihenfolge der "verschiebbaren" Elemente erhalten bleibt.

Erlaubte Werte: "s", "t", "l", "m"

Enthalten in: Wurzel

@sk:wrap

Das Attribut `sk:wrap` in einem Fremdelement zeigt an, dass das Element im Quelldokument ein einziges Kindelement "umhüllt".

Enthalten in: Wurzel

imgshow_clfy.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
*	d	http://herbaer.de/xmlns/20051201/doc
	is	http://herbaer.de/xmlns/20080705/imgshow
	ht	http://www.w3.org/1999/xhtml
	xsl	http://www.w3.org/1999/XSL/Transform

Eingebundene Stylesheets

skeleton_1.xslt - 1. Schritt zur Erzeugung einer Gerüst-Datei

Diese Datei enthält die wesentlichen Vorlagen zur Erzeugung einer "Gerüst"-Datei. Die Transformation, die die Gerüst-Datei erzeugt, definiert Vorlagen des Modus `check` zur Klassifizierung von Attributen und Erstklassifizierung von Elementen und bindet diese Datei ein.

Muster-Vorlagen (matching templates)

Muster-Vorlage is:title | is:p, check

Elemente, die zu übersetzenden Text enthalten.

Muster-Vorlage is:f, check

`f`-Elemente enthalten Text, der nicht zu übersetzen ist.

Muster-Vorlage is:p/@title | is:jpg/@alt, check

Zu übersetzende Attribute

Muster-Vorlage is:puppe | is:peruecke | is:kleid | is:story | is:sect | ht:a, check

Verweis-Elemente sind verschiebbar

Modus

Modus check

Die folgenden Vorlagen implementieren den Modus check:

Muster-Vorlage is:title | is:p, check
Muster-Vorlage is:f, check
Muster-Vorlage is:p/@title | is:jpg/@alt, check
Muster-Vorlage is:puppe | is:peruecke | is:kleid | is:story | is:sect | ht:a, check

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  file KLEIDER/web/src/localization/imgshow_clfy.xslt
  Klassifizierung von imgshow-Elementen für die Übersetzung
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:ht  = "http://www.w3.org/1999/xhtml"
  xmlns:is  = "http://herbaer.de/xmlns/20080705/imgshow"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  exclude-result-prefixes = "d"
  version   = "1.0"
>
<xsl:include href = "skeleton_1.xslt"/>


<xsl:template match = "is:title | is:p" mode  = "check">t</xsl:template>


<xsl:template match = "is:f" mode  = "check">l</xsl:template>


<xsl:template match = "is:p/@title | is:jpg/@alt" mode = "check">t</xsl:template>


<xsl:template
  match = "is:puppe | is:peruecke | is:kleid | is:story | is:sect | ht:a"
  mode  = "check"
>m</xsl:template>


</xsl:stylesheet>
```

local_clfy.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	l	http://herbaer.de/xmlns/20141210/localization
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Eingebundene Stylesheets

skeleton_1.xslt - 1. Schritt zur Erzeugung einer Gerüst-Datei

Diese Datei enthält die wesentlichen Vorlagen zur Erzeugung einer "Gerüst"-Datei. Die Transformation, die die Gerüst-Datei erzeugt, definiert Vorlagen des Modus `check` zur Klassifizierung von Attributen und Erstklassifizierung von Elementen und bindet diese Datei ein.

Muster-Vorlagen (matching templates)

Muster-Vorlage l:t, check

Das Element `l:t` enthält zu übersetzenden Text.

Modus

Modus check

Die folgenden Vorlagen implementieren den Modus check:

Muster-Vorlage l:t, check

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="#" type="application/xml"?>
<!--
  file KLEIDER/web/src/localization/local_clfy.xslt
  Klassifizierung von local-Elementen für die Übersetzung
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:l   = "http://herbaer.de/xmlns/20141210/localization"
  exclude-result-prefixes = "d"
  version   = "1.0"
>
<xsl:include href = "skeleton_1.xslt"/>


<xsl:template match = "l:t" mode = "check">t</xsl:template>


</xsl:stylesheet>
```

xhtml_clfy.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	ht	http://www.w3.org/1999/xhtml
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Eingebundene Stylesheets

skeleton_1.xslt - 1. Schritt zur Erzeugung einer Gerüst-Datei

Diese Datei enthält die wesentlichen Vorlagen zur Erzeugung einer "Gerüst"-Datei. Die Transformation, die die Gerüst-Datei erzeugt, definiert Vorlagen des Modus `check` zur Klassifizierung von Attributen und Erstklassifizierung von Elementen und bindet diese Datei ein.

Muster-Vorlagen (matching templates)

Muster-Vorlage ht:p | ht:title | ht:h1 | ht:h2 | ht:h3 | ht:h4 | ht:h5 | ht:h6 | ht:dd | ht:dt | ht:li | ht:td | ht:label, check

Elemente, die zu übersetzenden Text enthalten.

Muster-Vorlage ht:code, check

Elemente, die nicht zu übersetzenden Text enthalten.

Muster-Vorlage *[@translate = 'no'], check

Elemente, die als nicht zu übersetzen gekennzeichnet sind (z.B. weil sie Eigennamen enthalten)

Muster-Vorlage ht:a [. = @href], check

Verweise, deren Inhalt des Verweisziel ist, sind unveränderlich.

Muster-Vorlage ht:a [not (*) and not (. = @href)], check

Andere Verweise, die nur Text enthalten, können verschoben werden.

Muster-Vorlage ht:em [not (*)], check

Auch an Hervorhebungen und anderen Inline-Markierungen soll die Übersetzung nicht scheitern

Muster-Vorlage ht:strong [not (*)], check

Muster-Vorlage ht:small [not (*)], check

Muster-Vorlage ht:span [not (*)], check

Muster-Vorlage ht:br | ht:img | ht:input, check

Leere Elemente, die Text trennen.

Muster-Vorlage ht:img/@alt | ht:meta[@name='keywords']/@content | ht:a/@title, check

Attribute, die zu übersetzenden Text enthalten.

Modus

Modus check

Die folgenden Vorlagen implementieren den Modus check:

Muster-Vorlage ht:p | ht:title | ht:h1 | ht:h2 | ht:h3 | ht:h4 | ht:h5 | ht:h6 | ht:dd | ht:dt | ht:li | ht:td | ht:label, check
Muster-Vorlage ht:code, check
Muster-Vorlage *[@translate = 'no'], check
Muster-Vorlage ht:a [. = @href], check
Muster-Vorlage ht:a [not (*) and not (. = @href)], check
Muster-Vorlage ht:em [not (*)], check
Muster-Vorlage ht:strong [not (*)], check
Muster-Vorlage ht:small [not (*)], check
Muster-Vorlage ht:span [not (*)], check
Muster-Vorlage ht:br | ht:img | ht:input, check
Muster-Vorlage ht:img/@alt | ht:meta[@name='keywords']/@content | ht:a/@title, check

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet ref="#" type="application/xml"?>
<!--
  Klassifizierung xon XHTML-Elementen für die Übersetzung
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:ht  = "http://www.w3.org/1999/xhtml"
  exclude-result-prefixes = "d"
  version   = "1.0"
>
<xsl:include href = "skeleton_1.xslt"/>


<xsl:template
  match = "
    ht:p | ht:title
    | ht:h1 | ht:h2 | ht:h3 | ht:h4 | ht:h5 | ht:h6 | ht:dd | ht:dt | ht:li | ht:td | ht:label
  "
  mode = "check"
>t</xsl:template>


<xsl:template match = "ht:code" mode = "check">l</xsl:template>


<xsl:template match = "*[@translate = 'no']" mode = "check">l</xsl:template>


<xsl:template match = "ht:a [. = @href]" mode = "check">l</xsl:template>


<xsl:template match = "ht:a [not (*) and not (. = @href)]" mode = "check">m</xsl:template>


<xsl:template match = "ht:em     [not (*)]" mode = "check">m</xsl:template>
<xsl:template match = "ht:strong [not (*)]" mode = "check">m</xsl:template>
<xsl:template match = "ht:small  [not (*)]" mode = "check">m</xsl:template>
<xsl:template match = "ht:span   [not (*)]" mode = "check">m</xsl:template>


<xsl:template match = "ht:br | ht:img | ht:input" mode = "check">s</xsl:template>


<xsl:template
  match = "ht:img/@alt | ht:meta[@name='keywords']/@content | ht:a/@title"
  mode = "check"
>t</xsl:template>


</xsl:stylesheet>
```

skeleton_1.xslt

[Quelltext]

Allgemeines

1. Schritt zur Erzeugung einer Gerüst-Datei

Diese Datei enthält die wesentlichen Vorlagen zur Erzeugung einer "Gerüst"-Datei. Die Transformation, die die Gerüst-Datei erzeugt, definiert Vorlagen des Modus `check` zur Klassifizierung von Attributen und Erstklassifizierung von Elementen und bindet diese Datei ein.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	**Namensraum**
	xml	http://www.w3.org/XML/1998/namespace
*	xl	http://www.w3.org/1999/xlink
	sk	http://herbaer.de/xmlns/20150106/skeleton
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8
Indent	yes

Parameter

Parameter p_comments

Kommentare zur Fehlersuche einfügen? 1 ja, sonst nein

Select: 0

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage *

Muster-Vorlagen (matching templates)

Muster-Vorlage @*, check

Attribute haben für die Übersetzung in der Regel keine Bedeutung.

Muster-Vorlage *, check

Elemente haben für die Übersetzung in der Regel keine Bedeutung.

Muster-Vorlage @xml:lang, check

Das Attribut `xml:lang` bezeichnet die Sprache.

Muster-Vorlage @*, special

Besondere Behandlung von Attributen, die übersetzt werden oder die Sprache angeben.

Verwendete Modus:

check

Muster-Vorlage @*

Attribute, die nicht die Sprache angeben und nicht übersetzt werden, werden kopiert.

Verwendete Modus:

check

Muster-Vorlage /

Alle Kommmentare und Verarbeitungsanweisungen in der Wurzel werden kopiert.

Muster-Vorlage *

Parameter

psep

Default: "

für trennende Elemente: IDs der folgenden Trenner

Elemente.

Zu Elementen, die ein trennendes Kindelement enthalten, werden gruppierende `sk:part`-Elemente eingefügt.

Das Attribut `@sk:chk` enthält die Erstklassifizierung des Elements (`s t l` oder `m`)

Falls am Anfang leere Kindelemente vor weiterem Inhalt vorkommen, werden zwei sk:contents-Elemente angelegt: das erste Element kapselt die leeren Kindelemente, das zweite Element den weiteren Inhalt. So wird die richtige Reihenfolge der Elemente in der Übersetzung sichergestellt.

Verwendete Modus:

check
special

Verwendete globale Parameter oder Variable:

Parameter p_comments

Muster-Vorlage comment() | processing-instruction()

Kommentare und Verarbeitungsanweisungen werden kopiert.

Modus

Modus check

Klassifizierung eines Attributs:

`l`

Das Attribut gibt die Sprache an.

`t`

Der Attributwert ist zu übersetzen.

Erstklassifizierung eines Elements

`s`

Das Element trennt den Inhalt des Elternelements in Teile, die getrennt übersetzt werden (können).

`t`

Der Inhalt des Elements einschließlich seiner Kindelemente ist zu übersetzen.

`l`

Der Inhalt des Elements ist eine unveränderliche Zeichenkette, (ein Name oder eine URL).

`m`

Der Inhalt des Elements kann direkt im Elternelement erscheinen, und das Element selbst kann (mit leerem Inhalt) verschoben werden.

Die folgenden Vorlagen implementieren den Modus check:

Muster-Vorlage @*, check
Muster-Vorlage *, check
Muster-Vorlage @xml:lang, check

Der Modus check wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage @*, special
Muster-Vorlage @*
Muster-Vorlage *

Modus special

Die folgenden Vorlagen implementieren den Modus special:

Muster-Vorlage @*, special

Der Modus special wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage *

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  Rahmen zur Erzeugung einer Gerüst-Datei
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:sk  = "http://herbaer.de/xmlns/20150106/skeleton"
  xmlns:xl  = "http://www.w3.org/1999/xlink"
  exclude-result-prefixes = "d xl"
  version   = "1.0"
>
<xsl:param name = "p_comments" select = "0"/>


<xsl:output method = "xml" encoding = "utf-8" indent = "yes"/>


<xsl:template match = "@*" mode = "check"/>


<xsl:template match = "*" mode = "check"/>


<xsl:template match = "@xml:lang" mode = "check">l</xsl:template>


<xsl:template match = "@*" mode = "special">
  <xsl:variable name = "check">
    <xsl:apply-templates select = "." mode = "check"/>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test = "$check = 'l'">
      <sk:lang name = "{name (.)}">
        <xsl:value-of select = "."/>
      </sk:lang>
    </xsl:when>
    <xsl:when test = "$check = 't'">
      <sk:attribute name = "{name (.)}">
        <xsl:value-of select = "."/>
      </sk:attribute>
    </xsl:when>
    <xsl:otherwise/>
  </xsl:choose>
</xsl:template>


<xsl:template match = "@*">
  <xsl:variable name = "check">
    <xsl:apply-templates select = "." mode = "check"/>
  </xsl:variable>
  <xsl:if test = "string-length ($check) = 0">
    <xsl:copy-of select = "."/>
  </xsl:if>
</xsl:template>


<xsl:template match = "/">
  <xsl:apply-templates select = "comment() | processing-instruction() | text() | *"/>
</xsl:template>


<xsl:template match = "*">
```

```
<xsl:param name = "psep" select = "'|'"/>
<xsl:variable name = "check">
  <xsl:apply-templates select = "." mode = "check"/>
</xsl:variable>
<xsl:variable name = "wrap">
  <xsl:choose>
    <xsl:when test = "not (*)">text</xsl:when>
    <xsl:when test = "text() [string-length(normalize-space(.)) &gt; 0]"
              >mixed</xsl:when>
    <xsl:when test = "count(*) = 1">wrap</xsl:when>
    <xsl:when test = "count(*) &gt; 1">container</xsl:when>
  </xsl:choose>
</xsl:variable>
<xsl:variable name = "sep">
  <xsl:for-each select = "*">
    <xsl:variable name = "c">
      <xsl:apply-templates select = "." mode = "check"/>
    </xsl:variable>
    <xsl:if test = "$c = 's'">
      <xsl:value-of select = "concat (generate-id(.), '||')"/>
    </xsl:if>
  </xsl:for-each>
</xsl:variable>
<xsl:variable name = "sid">
  <xsl:choose>
    <xsl:when test = "contains ($psep, '||')">
      <xsl:value-of select = "substring-before ($psep, '||')"/>
    </xsl:when>
    <xsl:when test = "contains ($sep, '||')">
      <xsl:value-of select = "substring-before ($sep, '||')"/>
    </xsl:when>
    <xsl:otherwise/>
  </xsl:choose>
</xsl:variable>


<xsl:variable name = "ec" select =
  "count( (text() | *)
     [not (
        (
            descendant-or-self::text()
            | preceding-sibling::text()
            | preceding-sibling::*/descendant::text()
        )
        [string-length (normalize-space(.)) &gt; 0]
     )]
   )"
/>


<xsl:variable name = "ece" select =
  "count( *
     [not (
        (
            descendant::text()
            | preceding-sibling::text()
            | preceding-sibling::*/descendant::text()
        )
        [string-length (normalize-space(.)) &gt; 0]
     )]
   )"
/>


<xsl:variable name = "ect" select = "count ( text() | * )"/>
<xsl:copy>
  <xsl:if test = "string-length ($check) &gt; 0">
    <xsl:attribute name = "sk:chk">
      <xsl:value-of select = "$check"/>
    </xsl:attribute>
  </xsl:if>
  <xsl:if test = "string-length ($wrap) &gt; 0">
    <xsl:attribute name = "sk:wrap">
      <xsl:value-of select = "$wrap"/>
    </xsl:attribute>
  </xsl:if>
  <xsl:apply-templates select = "@*"/>
  <xsl:apply-templates select = "@*" mode = "special"/>
  <xsl:if test = "$p_comments">
    <xsl:comment>
      <xsl:value-of
        select = "concat (' ec: ', $ec, ', ece: ', $ece, ', ect: ', $ect, ' ')"
      />
    </xsl:comment>
  </xsl:if>
  <xsl:choose>
    <xsl:when test = "contains ($sep, '||')">
      <sk:contents>
        <sk:part>
          <xsl:apply-templates
            select = "(text() | *) [following-sibling::* [generate-id(.) = $sid]]"
          />
        </sk:part>
        <xsl:apply-templates select = "* [generate-id(.) = $sid]">
          <xsl:with-param name = "psep" select = "substring-after ($sep, '||')"/>
        </xsl:apply-templates>
```

```
      </sk:contents>
    </xsl:when>
    <xsl:when test = "text() [string-length (normalize-space(.)) &gt; 0] | *">
      <sk:contents>
        <xsl:if test = "string-length ($wrap) &gt; 0">
          <xsl:attribute name = "wrap">
            <xsl:value-of select = "$wrap"/>
          </xsl:attribute>
        </xsl:if>
        <xsl:choose>


          <xsl:when test = "string-length ($wrap) = 0 and 0 &lt; $ece and $ec &lt; $ect"
          >
            <sk:part>
              <xsl:apply-templates select = "(text()|*) [position() &lt;= $ec]"/>
            </sk:part>
            <sk:part>
              <xsl:apply-templates select = "(text()|*) [position() &gt; $ec]"/>
            </sk:part>
          </xsl:when>
          <xsl:otherwise>
            <xsl:apply-templates select = "text() | *"/>
          </xsl:otherwise>
        </xsl:choose>
      </sk:contents>
    </xsl:when>
  </xsl:choose>
</xsl:copy>


<xsl:if test = "$check = 's'">
  <sk:part>
    <xsl:choose>
      <xsl:when test = "string-length ($sid) &gt; 0">
        <xsl:apply-templates
        select = "
          (following-sibling::text() | following-sibling::*) [following-sibling::* [generate-id(.) = $sid]]
        "
        />
      </xsl:when>
      <xsl:otherwise>
        <xsl:apply-templates select = "following-sibling::text() | following-sibling::*"
        />
      </xsl:otherwise>
    </xsl:choose>
  </sk:part>
  <xsl:if test = "string-length ($sid) &gt; 0">
    <xsl:apply-templates select = "following-sibling::* [generate-id(.) = $sid]">
      <xsl:with-param name = "psep" select = "substring-after ($psep, '||')"/>
    </xsl:apply-templates>
  </xsl:if>
</xsl:if>
</xsl:template>


<xsl:template match = "comment() | processing-instruction()">
  <xsl:copy-of select = "."/>
</xsl:template>


</xsl:stylesheet>
```

skeleton_2.xslt

[Quelltext]

Allgemeines

2. Schritt zur Erzeugung einer Gerüst-Datei

Diese Datei definiert den zweiten Schritt zur Erzeugung einer Gerüst-Datei. Elemente, die zu übersetzenden Text enthalten, sind zu kennzeichnen und mit einer ID zu versehen. Die Kennzeichnungen (Attribut `@sk:class`) sind:

`text`

Das Element enthält zu übersetzenden Text, der nicht untergliedert ist.

`struct`

Das Element enthält Text mit markierten Textstellen.

`fix`

Der Textinhalt ist unabhängig von der Sprache. Das gilt nicht unbedingt für Attribute.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	**Namensraum**
	xml	http://www.w3.org/XML/1998/namespace
*	xl	http://www.w3.org/1999/xlink
	sk	http://herbaer.de/xmlns/20150106/skeleton
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8
Indent	yes

Parameter

Parameter p_comments

Kommentare zur Fehlersuche einfügen / erhalten?

Select: 1

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage sk:part | sk:contents

Muster-Vorlage *

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage sk:lang

`sk:lang`-Elemente wurden im ersten Schritt eingefügt. Sie stehen für Attribute, die die Sprache bezeichnen.

Muster-Vorlage sk:attribute

Parameter

plang

Default: "

`sk:attribute`-Elemente stehen für Attribute, deren Werte zu übersetzen sind.

Verwendete Modus:

xliffid

Muster-Vorlage sk:part | sk:contents

Parameter

plang

Default: "

Sprache

pclass

Default: "

Einordnung des Elternelements

unit

Default: "

Ist ein sk:unit-Attribut einzufügen?

Zu den eingefügten Elementen `sk:part` und `sk:contents` werden die XLIFF-Kennung und die Einordnung `@class` hinzugefügt. Das Attribut `sk:unit = "unit"` zeigt an, dass der Inhalt als eigener Baustein übersetzt werden kann. Es ist nicht Teil eines umfassenderen zu übersetzenden Bausteins.

Verwendete Modus:

xliffid

Verwendete globale Parameter oder Variable:

Parameter p_comments

Muster-Vorlage *

Parameter

plang

Default: "

Sprache, die auf einer höheren Ebene festgelegt ist

pclass

Default: "

Einordnung zur Übersetzung des übergeordneten Elements

unit

Default: "

Ist nachfolgend ein sk:unit-Attribut einzufügen?

Verwendete globale Parameter oder Variable:

Parameter p_comments

Muster-Vorlage @sk:chk, excl_chk

Im Modus `excl_chk` werden Attribute mit Ausnahme des Attributs `@ck:chk` kopiert

Muster-Vorlage @*, excl_chk

Muster-Vorlage *, xliffid

Die Kennung eines Elements ist seine Position im Quelldokument.

Muster-Vorlage comment() | processing-instruction()

Kommentare und Verarbeitungsanweisungen werden kopiert.

Modus

Modus xliffid

Die folgenden Vorlagen implementieren den Modus xliffid:

Muster-Vorlage *, xliffid

Der Modus xliffid wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage sk:attribute
Muster-Vorlage sk:part | sk:contents

Modus excl_chk

Die folgenden Vorlagen implementieren den Modus excl_chk:

Muster-Vorlage @sk:chk, excl_chk
Muster-Vorlage @*, excl_chk

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  2. Schritt zur Erzeugung einer Gerüst-Datei
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:sk  = "http://herbaer.de/xmlns/20150106/skeleton"
  xmlns:xl  = "http://www.w3.org/1999/xlink"
  exclude-result-prefixes = "d xl"
  version   = "1.0"
>
<xsl:param name = "p_comments" select = "1"/>

<xsl:output method = "xml" encoding = "utf-8" indent = "yes"/>

<xsl:template match = "/">
  <xsl:apply-templates select = "comment() | processing-instruction() | text() | *"/>
</xsl:template>

<xsl:template match = "sk:lang">
  <xsl:copy-of select = "."/>
</xsl:template>

<xsl:template match = "sk:attribute">
  <xsl:param name = "plang" select = "''"/>
  <xsl:copy>
    <xsl:copy-of select = "@*"/>
    <xsl:attribute name = "xliffid">
      <xsl:apply-templates select = "." mode = "xliffid"/>
    </xsl:attribute>
    <xsl:attribute name = "class">text</xsl:attribute>
    <xsl:if test = "string-length ($plang) &gt; 0">
      <xsl:attribute name = "lang">
        <xsl:value-of select = "$plang"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:copy-of select = "text() | *"/>
  </xsl:copy>
</xsl:template>

<xsl:template match = "sk:part | sk:contents">

  <xsl:param name = "plang" select = "''"/>

  <xsl:param name = "pclass" select = "''"/>

  <xsl:param name = "unit" select = "''"/>
  <xsl:variable name = "check">
    <xsl:choose>
      <xsl:when test = "../@sk:chk">
        <xsl:value-of select = "../@sk:chk"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:variable name = "class">
    <xsl:choose>
      <xsl:when test = "$pclass = '' and $check = ''"/>
      <xsl:when test = "$pclass = 'fix' or $check = 'l'">fix</xsl:when>
      <xsl:when test = "$pclass = 'move' or $check = 'm'">move</xsl:when>
      <xsl:when test = "* [not (@sk:chk = 'm')]">struct</xsl:when>
      <xsl:otherwise>text</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:variable name = "thisunit">
    <xsl:if test = "
        (
          ( string-length ($class) &gt; 0 and string-length ($pclass) = 0 )
          or string-length ($unit) &gt; 0
        )
        and not (../@sk:wrap = 'wrap' or ../@sk:wrap = 'container')
      ">
      <xsl:text>unit</xsl:text>
    </xsl:if>
  </xsl:variable>
  <xsl:variable name = "nunit">
    <xsl:if test = "
```

```
        (
          ( string-length ($class) &gt; 0 and string-length ($pclass) = 0 )
          or string-length ($unit) &gt; 0
        )
        and (../@sk:wrap = 'wrap' or ../@sk:wrap = 'container')
      ">
      <xsl:text>unit</xsl:text>
    </xsl:if>
  </xsl:variable>
  <xsl:copy>
    <xsl:copy-of select = "@*"/>
    <xsl:if test = "string-length ($class) &gt; 0">
      <xsl:attribute name = "class">
        <xsl:value-of select = "$class"/>
      </xsl:attribute>
      <xsl:if test = "string-length ($thisunit) &gt; 0">
        <xsl:attribute name = "unit">unit</xsl:attribute>
      </xsl:if>
    </xsl:if>
    <xsl:attribute name = "xliffid">
      <xsl:apply-templates select = "." mode = "xliffid"/>
    </xsl:attribute>
    <xsl:if test = "string-length ($plang) &gt; 0">
      <xsl:attribute name = "lang">
        <xsl:value-of select = "$plang"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:if test = "$p_comments">
      <xsl:comment>
        <xsl:value-of
            select = "concat
            (' pclass = ', $pclass, ', class = ', $class, ', unit = ', $unit)"/>
      </xsl:comment>
      <xsl:apply-templates select = "comment()"/>
    </xsl:if>
    <xsl:apply-templates select = "text() | *">
      <xsl:with-param name = "plang"  select = "$plang"/>
      <xsl:with-param name = "pclass" select = "$class"/>
      <xsl:with-param name = "unit"   select = "$nunit"/>
    </xsl:apply-templates>
  </xsl:copy>
</xsl:template>


<xsl:template match = "*">


  <xsl:param name = "plang"  select = "''"/>


  <xsl:param name = "pclass" select = "''"/>


  <xsl:param name = "unit" select = "''"/>
  <xsl:variable name = "ll" select = "sk:lang"/>
  <xsl:copy>
    <xsl:copy-of select = "@*"/>
    <xsl:if test = "$p_comments">
      <xsl:comment>
        <xsl:value-of
            select = "concat (' plang = ', $plang, ', pclass = ', $pclass, ' ')"/>
      </xsl:comment>
      <xsl:apply-templates select = "comment()"/>
    </xsl:if>
    <xsl:apply-templates select = "text() | *">
      <xsl:with-param name = "plang">
        <xsl:choose>
          <xsl:when test = "string-length ($ll) &gt; 0">
            <xsl:value-of select = "$ll"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select = "$plang"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:with-param>
      <xsl:with-param name = "pclass">
        <xsl:choose>
          <xsl:when test = "string-length ($pclass) = 0"/>
          <xsl:when test = "string-length ($ll) &gt; 0 and not ($plang = $ll)">
            <xsl:text>fix</xsl:text>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select = "$pclass"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:with-param>
      <xsl:with-param name = "unit" select = "$unit"/>
    </xsl:apply-templates>
  </xsl:copy>
</xsl:template>


<xsl:template match = "@sk:chk" mode = "excl_chk"/>
<xsl:template match = "@*"      mode = "excl_chk">
  <xsl:copy-of select = "."/>
</xsl:template>
```

```
<xsl:template match = "*" mode = "xliffid">
  <xsl:text>e</xsl:text>
  <xsl:value-of select = "count (preceding::*) + count (ancestor::*) + 1"/>
</xsl:template>


<xsl:template match = "comment() | processing-instruction()">
  <xsl:copy-of select = "."/>
</xsl:template>


</xsl:stylesheet>
```

skeleton_xliff.xslt

[Quelltext]

Allgemeines

XLIFF aus Skeleton

Markierte Stellen, deren markierendes Element an das Ende des Elternelements verschoben werden kann, sind durch das Attribut `xl:mrk/@type = "herbaer:move"` gekennzeichnet.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	**Namensraum**
	xml	http://www.w3.org/XML/1998/namespace
	(default)	urn:oasis:names:tc:xliff:document:2.0
	sk	http://herbaer.de/xmlns/20150106/skeleton
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Indent	yes

Eingebundene Stylesheets

/pool/txt.xslt - Hilfsvorlagen zur Ausgabe und Verarbeitung von Text

Vorlage `txt.makedotword` und `txt.break`

Parameter

Parameter p_file

Pfad des Quelldokuments, dient zur Erzeugung des Wertes von `file/@id`

Select: ''

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Wurzelelement

Aufgerufene benannte Vorlagen:

txt.makedotword

Verwendete globale Parameter oder Variable:

Parameter p_file

Muster-Vorlage sk:contents [@unit = 'unit'] | sk:part [@unit = 'unit']

Ein zu übersetzender Baustein.

Das erzeugte `unit`-Element kann leer sein. Das verstößt gegen die XLIFF-Spezifikation 2.0. Eine folgende Transformation entfernt `unit`-Elemente, die kein `segment`- oder `ignorable`-Element enthalten.

Verwendete Modus:

segment

Muster-Vorlage sk:contents [@wrap = 'wrap'], segment

"Umhüllungen" werden ignoriert.

Muster-Vorlage sk:contents [@class = 'fix'], segment

Nicht zu übersetzende Teile werden ebenfalls ignoriert.

Muster-Vorlage sk:part [@class = 'fix'], segment

Muster-Vorlage sk:contents | sk:part, segment

Zu übersetzende Teile mit innerer Struktur

Verwendete Modus:

mark

Muster-Vorlage text(), mark

Text

Muster-Vorlage sk:attribute | sk:lang, mark

Attribute tragen nicht zum Textinhalt bei.

Muster-Vorlage sk:contents | sk:part, mark

Parameter

sid

"Umhüllungen" und verschiebbare Markierungen sind "transparent".

Verwendete Modus:

mark

Muster-Vorlage *, mark

Parameter

sid

Trennende Elemente tragen ein Leerzeichen zum Textinhalt bei, andere Elemente sind "transparent".

Verwendete Modus:

mark

Muster-Vorlage @lang

Sprache

Modus

Modus segment

Die folgenden Vorlagen implementieren den Modus segment:

Muster-Vorlage sk:contents [@wrap = 'wrap'], segment
Muster-Vorlage sk:contents [@class = 'fix'], segment
Muster-Vorlage sk:part [@class = 'fix'], segment
Muster-Vorlage sk:contents | sk:part, segment

Der Modus segment wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage sk:contents [@unit = 'unit'] | sk:part [@unit = 'unit']

Modus mark

Der Modus `mark` liefert den Textinhalt mit Markierungen verschachtelter Elemente

Die folgenden Vorlagen implementieren den Modus mark:

Muster-Vorlage text(), mark
Muster-Vorlage sk:attribute | sk:lang, mark
Muster-Vorlage sk:contents | sk:part, mark
Muster-Vorlage *, mark

Der Modus mark wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage sk:contents | sk:part, segment
Muster-Vorlage sk:contents | sk:part, mark
Muster-Vorlage *, mark

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  XLIFF aus Skeleton
  2015, 2016 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:sk  = "http://herbaer.de/xmlns/20150106/skeleton"
  xmlns     = "urn:oasis:names:tc:xliff:document:2.0"
  exclude-result-prefixes = "d"
  version   = "1.0"
>
<xsl:include href = "/pool/txt.xslt"/>


<xsl:param name = "p_file" select = "''"/>


<xsl:output method = "xml" indent = "yes"/>


<xsl:template match = "/">
  <xliff>
    <file>
      <xsl:if test = "string-length ($p_file) &gt; 0">
        <xsl:attribute name = "id">
          <xsl:text>f_</xsl:text>
          <xsl:call-template name = "txt.makedotword">
            <xsl:with-param name = "txt" select = "$p_file"/>
          </xsl:call-template>
        </xsl:attribute>
      </xsl:if>
      <xsl:for-each select = "//sk:attribute">
        <unit id = "u_{@xliffid}">
          <segment id = "s_{@xliffid}">
            <source xml:lang = "{@lang}">
              <xsl:value-of select = "."/>
            </source>
          </segment>
        </unit>
      </xsl:for-each>
      <xsl:apply-templates
        select = "(//sk:contents | //sk:part) [@unit = 'unit' and not (@class = 'fix')]"
      />
    </file>
  </xliff>
</xsl:template>


<xsl:template match = "sk:contents [@unit = 'unit'] | sk:part [@unit = 'unit']">
  <unit id = "u_{@xliffid}">
    <xsl:apply-templates select = ". | .//sk:contents | .//sk:part" mode = "segment"/>
  </unit>
</xsl:template>


<xsl:template match = "sk:contents [@wrap = 'wrap']" mode = "segment"/>


<xsl:template match = "sk:contents [@class = 'fix']" mode = "segment"/>
<xsl:template match = "sk:part     [@class = 'fix']" mode = "segment"/>


<xsl:template match = "sk:contents | sk:part" mode = "segment">
  <xsl:variable name = "sid" select = "@xliffid"/>
  <segment id = "s_{$sid}">
    <source>
      <xsl:apply-templates select = "@lang"/>
      <xsl:apply-templates select = "text() | *" mode = "mark">
        <xsl:with-param name = "sid" select = "$sid"/>
      </xsl:apply-templates>
    </source>
  </segment>
</xsl:template>


<xsl:template match = "text()" mode = "mark">
  <xsl:value-of select = "."/>
</xsl:template>
```

```
<xsl:template match = "sk:attribute | sk:lang" mode = "mark"/>


<xsl:template match = "sk:contents | sk:part" mode = "mark">
  <xsl:param name = "sid"/>
  <xsl:choose>
    <xsl:when test = "@wrap = 'wrap'">
      <xsl:apply-templates select = "text() | *" mode = "mark">
        <xsl:with-param name = "sid" select = "$sid"/>
      </xsl:apply-templates>
    </xsl:when>
    <xsl:when test = "@class = 'move'">
      <mrk id = "m_{@xliffid}_{$sid}" type = "herbaer:move">
        <xsl:apply-templates select = "text() | *" mode = "mark">
          <xsl:with-param name = "sid" select = "$sid"/>
        </xsl:apply-templates>
      </mrk>
    </xsl:when>
    <xsl:when test = "@class = 'fix'">
      <mrk translate = "no" id = "m_{@xliffid}_{$sid}">
        <xsl:apply-templates select = "text() | *" mode = "mark">
          <xsl:with-param name = "sid" select = "$sid"/>
        </xsl:apply-templates>
      </mrk>
    </xsl:when>
    <xsl:otherwise>
      <mrk id = "m_{@xliffid}_{$sid}">
        <xsl:apply-templates select = "text() | *" mode = "mark">
          <xsl:with-param name = "sid" select = "$sid"/>
        </xsl:apply-templates>
      </mrk>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "*" mode = "mark">
  <xsl:param name = "sid"/>
  <xsl:choose>
    <xsl:when test = "@sk:chk = 's'">
      <xsl:value-of select = "$txt.break"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates select = "text() | *" mode = "mark">
        <xsl:with-param name = "sid" select = "$sid"/>
      </xsl:apply-templates>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "@lang">
  <xsl:attribute name = "xml:lang">
    <xsl:value-of select = "."/>
  </xsl:attribute>
</xsl:template>


</xsl:stylesheet>
```

xliff_clean.xslt

[Quelltext]

Allgemeines

Leere unit-Elemente entfernen

Die Transformation `skeleton_xliff.xslt` kann Elemente erzeugen, die nicht den erforderlichen Inhalt enthalten, z.B. leere `source`- oder `mrk`-Elemente. Diese Transformation entfernt diese Elemente.

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
	(default)	urn:oasis:names:tc:xliff:document:2.0
	xl	urn:oasis:names:tc:xliff:document:2.0
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Indent	yes

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Wurzel

Muster-Vorlage processing-instruction() | comment() | text() | @*

Textknoten und Attribute werden kopiert

Muster-Vorlage xl:unit [not (xl:segment/xl:source//text() [string-length(normalize-space(.)) > 0] | xl:ignorable)]

`unit`-Elemente, die kein `segment`- oder `ignorable`-Element enthalten, werden ignoriert.

Muster-Vorlage xl:segment [not (xl:source//text() [string-length(normalize-space(.)) > 0])]

`segment`-Elemente ohne echten source-Inhalt werden ignoriert.

Muster-Vorlage xl:mrk [not (.//text() [string-length(normalize-space(.)) > 0])]

`mrk`-Elemente ohne echten Inhalt werden ignoriert.

Muster-Vorlage *

Andere Elemente werden kopiert.

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  Leere unit-Elemente entfernen
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:xl  = "urn:oasis:names:tc:xliff:document:2.0"
  xmlns     = "urn:oasis:names:tc:xliff:document:2.0"
  exclude-result-prefixes = "d"
  version   = "1.0"
>
<xsl:output method = "xml" indent = "yes"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "processing-instruction() | comment() | text() | *"/>
</xsl:template>


<xsl:template match = "processing-instruction() | comment() | text() | @*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match =
   "xl:unit [ not (
      xl:segment/xl:source//text() [string-length(normalize-space(.)) &gt; 0]
      | xl:ignorable
   )]"
/>


<xsl:template match =
   "xl:segment [ not (xl:source//text() [string-length(normalize-space(.)) &gt; 0]) ]"
/>


<xsl:template match =
   "xl:mrk [ not (.//text() [string-length(normalize-space(.)) &gt; 0]) ]"
/>


<xsl:template match = "*">
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:choose>
      <xsl:when test = "
        text() [string-length(normalize-space(.)) &gt; 0]
        | processing-instruction()
        | comment()
        ">
        <xsl:apply-templates
          select = "text() | processing-instruction() | comment() | *"
          />
      </xsl:when>
      <xsl:otherwise>
        <xsl:apply-templates select = "*"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:copy>
</xsl:template>


</xsl:stylesheet>
```

skeleton_merge_xliff.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	Namensraum
	xml	http://www.w3.org/XML/1998/namespace
*	xlf	urn:oasis:names:tc:xliff:document:2.0
	ti	http://herbaer.de/xmlns/201500703/transinfo/
*	sk	http://herbaer.de/xmlns/20150106/skeleton
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8
Indent	no

Parameter

Parameter p_xliff

Pfad der übersetzten XLIFF-Datei

Select: ''

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Parameter p_machine
Variable g_xlffile

Parameter p_srclang

Quellsprache

Select: /*/sk:lang

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage sk:lang
Muster-Vorlage sk:contents
Muster-Vorlage sk:part

Parameter p_trglang

Zielsprache

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage sk:lang

Parameter p_machine

Übersetzungsmaschine

Select: document($p_xliff)/xlf:xliff/ti:machine

Verwendete globale Parameter oder Variable:

Parameter p_xliff

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /*
Muster-Vorlage sk:lang

Globale Variable

Variable g_xlffile

Das `file`-Element der übersetzten XLIFF-Datei

Select: document($p_xliff)/xlf:xliff/xlf:file[1]

Verwendete globale Parameter oder Variable:

Parameter p_xliff

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage sk:attribute
Muster-Vorlage sk:contents
Muster-Vorlage sk:part

Variable g_sklroot

Das Wurzelelement der Gerüst-Datei

Select: /*

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage /
Muster-Vorlage xlf:mrk

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Wurzel

Verwendete globale Parameter oder Variable:

Variable g_sklroot

Muster-Vorlage /*

Zum Wurzelelement wird ein Attribut eingefügt, das die Übersetzungsmaschine bezeichnet.

Verwendete globale Parameter oder Variable:

Parameter p_machine

Muster-Vorlage processing-instruction()

Verarbeitungsanweisungen werden kopiert.

Muster-Vorlage text()

Text wird kopiert.

Muster-Vorlage *

Parameter

hint

Quelldokument-Elemente

Muster-Vorlage sk:lang

Angabe der Zielsprache

Verwendete globale Parameter oder Variable:

Parameter p_srclang
Parameter p_trglang
Parameter p_machine

Muster-Vorlage sk:attribute

Übersetzte Attribute

Verwendete globale Parameter oder Variable:

Variable g_xlffile

Muster-Vorlage sk:contents

Element-Inhalt. Wenn bei der Übersetzung ein Problem aufgetreten ist, bleibt der Inhalt in der Quellsprache erhalten.

Verwendete Modus:

copyemptyelts

Verwendete globale Parameter oder Variable:

Parameter p_srclang
Variable g_xlffile

Muster-Vorlage sk:part

Teil eines Element-Inhalts Wenn bei der Übersetzung ein Problem aufgetreten ist, wird ein sk:part-Element mit dem Inhalt in der Quellsprache erzeugt.

Verwendete Modus:

copyemptyelts

Verwendete globale Parameter oder Variable:

Parameter p_srclang
Variable g_xlffile

Muster-Vorlage *, copyemptyelts

Leere Elemente werden ohne die zusätzlichen skl-Elemente kopiert. Sie enthalten vielleicht Attribute, die übersetzt werden sollen.

Verwendete Modus:

textcont

Muster-Vorlage *, textcont

Der Textinhalt eines Elements

Verwendete Modus:

textcont

Muster-Vorlage text(), textcont

Textinhalt wird kopiert

Muster-Vorlage sk:lang, textcont

`sk:lang`- und `sk:attribute`-Elemente enthalten keinen Textinhalt des Quell-Elements.

Muster-Vorlage sk:attribute, textcont

Muster-Vorlage @*

Attribute werden im allgemeinen kopiert.

Muster-Vorlage @sk:*

Gerüst-Attribute werden nicht kopiert.

Muster-Vorlage xlf:target

Übersetzung

Muster-Vorlage xlf:mrk

Markierte Textstellen

Verwendete Modus:

upwrap

Verwendete globale Parameter oder Variable:

Variable g_sklroot

Muster-Vorlage sk:contents, upwrap

Parameter

hint

Hinweis zum Übersetzer

"Verpackte" verschachtelte Teilabschnitte werden "eingepackt".

Verwendete Modus:

upwrap

Modus

Modus copyemptyelts

Die folgenden Vorlagen implementieren den Modus copyemptyelts:

Muster-Vorlage *, copyemptyelts

Der Modus copyemptyelts wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage sk:contents
Muster-Vorlage sk:part

Modus textcont

Die folgenden Vorlagen implementieren den Modus textcont:

Muster-Vorlage *, textcont
Muster-Vorlage text(), textcont
Muster-Vorlage sk:lang, textcont
Muster-Vorlage sk:attribute, textcont

Der Modus textcont wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage *, copyemptyelts
Muster-Vorlage *, textcont

Modus upwrap

Die folgenden Vorlagen implementieren den Modus upwrap:

Muster-Vorlage sk:contents, upwrap

Der Modus upwrap wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage xlf:mrk
Muster-Vorlage sk:contents, upwrap

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  Skeleton und XLIFF zur übersetzten Datei zusammenfügen
  2015, 2016 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
  2015-07-27 xlf:target/@ti:error
  2015-08-15 xlf:mrk/@ti:hint
  2016-03-28 upwrap
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:sk  = "http://herbaer.de/xmlns/20150106/skeleton"
  xmlns:ti  = "http://herbaer.de/xmlns/201500703/transinfo/"
  xmlns:xlf = "urn:oasis:names:tc:xliff:document:2.0"
  exclude-result-prefixes = "d sk xlf"
  version   = "1.0"
>


<xsl:param name = "p_xliff" select = "''"/>


<xsl:param name = "p_srclang" select = "/*/sk:lang"/>


<xsl:param name = "p_trglang"/>


<xsl:param name = "p_machine" select = "document($p_xliff)/xlf:xliff/ti:machine"/>


<xsl:variable name = "g_xlffile" select = "document($p_xliff)/xlf:xliff/xlf:file[1]"/>


<xsl:variable name = "g_sklroot" select = "/*"/>


<xsl:output method = "xml" encoding = "utf-8" indent = "no"/>


<xsl:template match = "/">
  <xsl:if test = "$g_sklroot">
    <xsl:apply-templates select = "text() | processing-instruction() | *"/>
  </xsl:if>
</xsl:template>


<xsl:template match = "/*">
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:apply-templates select = "sk:lang"/>
    <xsl:apply-templates select = "sk:attribute"/>
    <xsl:if test = "string-length ($p_machine) &gt; 0">
      <xsl:attribute name = "ti:machine">
        <xsl:value-of select = "$p_machine"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates select = "sk:contents"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "processing-instruction()">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "text()">
  <xsl:copy-of select = "."/>
</xsl:template>
```

```
<xsl:template match = "*">
  <xsl:param name = "hint"/>
  <xsl:copy>
    <xsl:if test = "string-length ($hint) &gt; 0">
      <xsl:attribute name = "ti:hint">
        <xsl:value-of select = "$hint"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates select = "@*"/>
    <xsl:apply-templates select = "sk:lang"/>
    <xsl:apply-templates select = "sk:attribute"/>
    <xsl:apply-templates select = "sk:contents"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "sk:lang">
  <xsl:attribute name = "xml:lang">
    <xsl:choose>
      <xsl:when test = ". = $p_srclang">
        <xsl:value-of select = "$p_trglang"/>
        <xsl:if test = "string-length ($p_machine) &gt; 0">
          <xsl:value-of select = "concat ('-x-mtfrom-', $p_srclang)"/>
        </xsl:if>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select = "."/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
</xsl:template>


<xsl:template match = "sk:attribute">
  <xsl:variable name = "sid" select = "concat ('s_', @xliffid)"/>
  <xsl:attribute name = "{@name}">
    <xsl:choose>
      <xsl:when test = "$g_xlffile//xlf:segment[@id = $sid]/xlf:target">
        <xsl:value-of select = "$g_xlffile//xlf:segment[@id = $sid]/xlf:target"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select = "."/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
</xsl:template>


<xsl:template match = "sk:contents">
  <xsl:variable name = "xliffid"  select = "@xliffid"/>
  <xsl:variable name = "sid"      select = "concat ('s_', $xliffid)"/>
  <xsl:choose>
    <xsl:when test = "sk:part">
      <xsl:apply-templates select = "*"/>
    </xsl:when>
    <xsl:when test = "$g_xlffile//xlf:segment [@id = $sid]/xlf:target [not (@ti:error)]">
      <xsl:apply-templates select = "$g_xlffile//xlf:segment [@id = $sid]/xlf:target">
        <xsl:with-param name = "xliffid" select = "$xliffid"/>
      </xsl:apply-templates>


      <xsl:apply-templates select = "*" mode = "copyemptyelts"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:if test = "$g_xlffile//xlf:segment [@id = $sid]/xlf:target/@ti:error">
        <xsl:attribute name = "xml:lang">
          <xsl:value-of select = "$p_srclang"/>
        </xsl:attribute>
      </xsl:if>
      <xsl:choose>
        <xsl:when test = "text() [string-length(normalize-space(.)) &gt; 0]">
          <xsl:apply-templates select = "text()|*">
            <xsl:with-param name = "xliffid" select = "$xliffid"/>
          </xsl:apply-templates>
        </xsl:when>
        <xsl:otherwise>
          <xsl:apply-templates select = "*">
            <xsl:with-param name = "xliffid" select = "$xliffid"/>
          </xsl:apply-templates>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "sk:part">
  <xsl:variable name = "xliffid"  select = "@xliffid"/>
  <xsl:variable name = "sid"      select = "concat ('s_', $xliffid)"/>
  <xsl:choose>
    <xsl:when test = "sk:part">
      <xsl:apply-templates select = "*"/>
    </xsl:when>
    <xsl:when test = "$g_xlffile//xlf:segment [@id = $sid]/xlf:target [not (@ti:error)]">
      <xsl:apply-templates select = "$g_xlffile//xlf:segment [@id = $sid]/xlf:target">
        <xsl:with-param name = "xliffid" select = "$xliffid"/>
      </xsl:apply-templates>
```

```
      <xsl:apply-templates select = "*" mode = "copyemptyelts"/>
    </xsl:when>
    <xsl:when test = "$g_xlffile//xlf:segment [@id = $sid]/xlf:target/@ti:error">
      <sk:part xml:lang = "{$p_srclang}">
        <xsl:choose>
          <xsl:when test = "text() [string-length(normalize-space(.)) &gt; 0]">
            <xsl:apply-templates select = "text()|*">
              <xsl:with-param name = "xliffid" select = "$xliffid"/>
            </xsl:apply-templates>
          </xsl:when>
          <xsl:otherwise>
            <xsl:apply-templates select = "*">
              <xsl:with-param name = "xliffid" select = "$xliffid"/>
            </xsl:apply-templates>
          </xsl:otherwise>
        </xsl:choose>
      </sk:part>
    </xsl:when>
    <xsl:otherwise>
      <xsl:choose>
        <xsl:when test = "text() [string-length(normalize-space(.)) &gt; 0]">
          <xsl:apply-templates select = "text()|*">
            <xsl:with-param name = "xliffid" select = "$xliffid"/>
          </xsl:apply-templates>
        </xsl:when>
        <xsl:otherwise>
          <xsl:apply-templates select = "*">
            <xsl:with-param name = "xliffid" select = "$xliffid"/>
          </xsl:apply-templates>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "*" mode = "copyemptyelts">
  <xsl:variable name = "textcont">
    <xsl:apply-templates select = "." mode = "textcont"/>
  </xsl:variable>
  <xsl:if test = "string-length (normalize-space ($textcont)) = 0">
    <xsl:apply-templates select = "."/>
  </xsl:if>
</xsl:template>


<xsl:template match = "*" mode = "textcont">
  <xsl:apply-templates mode = "textcont"/>
</xsl:template>


<xsl:template match = "text()" mode = "textcont">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "sk:lang"      mode = "textcont"/>
<xsl:template match = "sk:attribute" mode = "textcont"/>


<xsl:template match = "@*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "@sk:*"/>


<xsl:template match = "xlf:target">
  <xsl:apply-templates/>
</xsl:template>


<xsl:template match = "xlf:mrk">
  <xsl:variable name = "id" select = "substring-before (substring (@id, 3), '_')"/>
  <xsl:choose>
    <xsl:when test = "$g_sklroot//sk:part[@xliffid = $id]">
      <xsl:apply-templates select = "$g_sklroot//sk:part[@xliffid = $id]"/>
    </xsl:when>
    <xsl:when test = "$g_sklroot//sk:contents[@xliffid = $id]">
      <xsl:apply-templates select = "$g_sklroot//sk:contents[@xliffid = $id]" mode = "upwrap"
      >
        <xsl:with-param name = "hint" select = "@ti:hint"/>
      </xsl:apply-templates>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "sk:contents" mode = "upwrap">
```

```
  <xsl:param name = "hint"/>
  <xsl:choose>
    <xsl:when test = "../parent::sk:contents/@wrap = 'wrap'">
      <xsl:apply-templates select = "../parent::sk:contents" mode = "upwrap">
        <xsl:with-param name = "hint" select = "$hint"/>
      </xsl:apply-templates>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates select = "..">
        <xsl:with-param name = "hint" select = "$hint"/>
      </xsl:apply-templates>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


</xsl:stylesheet>
```

transinfo.rng - Metadaten zur Übersetzung

Namespace	http://herbaer.de/xmlns/201500703/transinfo/
(foreign_att)	Attribute anderer XML-Namensräume
	Enthalten in: machine, translator
machine	Bezeichung der benutzen Übersetzungsmaschine, falls Text maschinell übersetzt worden ist.
	Enthält: Datentyp string

```
<element name="machine">
  <ref name="foreign_att"/>
  <ref name="att_name"/>
  <data type="string"/>
</element>
```

translator	Name des Übersetzers, der Text übersetzt hat.
	Enthält: Datentyp string

```
<element name="translator">
  <ref name="foreign_att"/>
  <ref name="att_name"/>
  <data type="string"/>
</element>
```

@machine	In manchen XML-Dokumenten sind Elemente fremder Namensräume nicht oder nur eingeschränkt möglich. Deshalb definiere ich Attribute, die den Elementen entsprechen. Grundsätzlich bevorzuge ich Elemente. Bezeichnung der Übersetzungsmaschine, falls der Inhalt des Elements maschinell übersetzt worden ist.
@translator	Name des Übersetzers, der den Inhalt des Elements übersetzt hat.
@error	Das Attribut bedeutet, dass bei der Übersetzung des Inhalts des Elements ein Fehler oder ein Problem aufgetreten ist. Ein typisches Problem ist, dass zu Textausschnitten, die in der Quellsprache markiert sind, in der Übersetzung die Entsprechung nicht automatisch erkannt wird. Der Wert des Attributs kann einen Hinweis auf das Problem liefern.
@hint	Hinweis zur Übersetzung. Der Wert "moved" bedeutet, dass das Element aus dem Textzusammenhang herausgenommen ist. Der Textinhalt gehört nicht zum Textzusammenhang, der dem Quelltext entspricht.

Übersetzung der XLIFF-Dateien

Die Transformation `xliff_txt.xslt` erzeugt aus einer XLIFF-Datei eine Übersetzungstext-Datei (einfachen Text). Die Übersetzungstext-Datei zur automatischen Übersetzung liegt in demselben Verzeichnis wie die XLIFF-Datei und hat die Dateinamensendung `.mtt` statt `.xlf`.

Für den Fall, dass ein menschlicher Übersetzer mir helfen möchte, habe ich für die Zielsprache eine Übersetzungstext-Dateien mit Kommentaren vorgesehen. Im Dateipfad der XLIFF-Datei wird *`WEBDIR`*`/local` durch *`WEBDIR`*`/local/trtout` ersetzt und die Dateinamensendung *`SRCLANG`*`.xlf` durch *`TGTLANG`*`.txt`. *`SRCLANG`* steht für die Kennung der Quellsprache (de), *`TGTLANG`* für die Kennung der Zielsprache.

Das Perl-Programm `mttext.pl` übersetzt die Textdatei und gibt eine übersetzte Textdatei aus. Die übersetzte Textdatei wird nicht gespeichert, sondern gleich vom Perl-Programm `resstruct.pl` verarbeitet. Dieses Programm versucht, die Verschachtelungen von Textteilen entsprechend den `=struct`-Zeilen zu erkennen. Die Ausgabe ist eine XML-Datei des Namensraums `http://herbaer.de/xmlns/20150127/resstruct#` (`resstruct.rng`). Im Dateipfad der XLIFF-Datei wird die Dateinamensendung *`SRCLANG`*`.xlf` durch *`TGTLANG`*`.rtr` ersetzt.

Vor der Weiterverarbeitung der `*.`*`TGTLANG`*`.rtr`-Datei prüfe ich, ob eine Datei unter dem Pfad *`WEBDIR`*`/local/trtin` statt *`WEBDIR`*`/local` existiert, die auf einer menschlichen Übersetzung basiert. Die menschlich übersetzte Datei wird der maschinell übersetzten Datei vorgezogen. Die Transformation `xliff_merge_rtr.xslt` fügt die XLIFF-Datei `*.`*`SRCLANG`*`.xlf` und die übersetzte Datei `*.`*`TGTLANG`*`.rtr` zur übersetzten XLIFF-Datei `*.`*`TGTLANG`*`.xlf` zusammen.

xliff_txt.xslt

[Quelltext]

Allgemeines

XLIFF zu einfachem Text

Diese Transformation erzeugt aus einer XLIFF-Datei eine Übersetzungstextdatei (s. `trtfileformat.dbk`).

Namensräume

Präfix	Namensraum
xml	http://www.w3.org/XML/1998/namespace
lf	urn:oasis:names:tc:xliff:document:2.0
d	http://herbaer.de/xmlns/20051201/doc
xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	text
Encoding	utf-8

Eingebundene Stylesheets

/pool/txt.xslt - Hilfsvorlagen zur Ausgabe und Verarbeitung von Text

Text-Funktionen

Parameter

Parameter p_targetlang

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage lf:segment

Globale Variable

Variable g_rlist

An einigen Textstellen wird ein Zeilenwechsel eingefügt.

Select: concat ('. |.', $txt.break, '|: |:', $txt.break, '|! |!', $txt.break, '|? |?', $txt.break)

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage xsl:template [@name = 'textline'], txt.apply

Variable g_textline

Vorlage zur Verarbeitung einer Textzeile

Select: document('')/xsl:stylesheet/xsl:template [@name = 'textline']

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage lf:source

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Wurzel des XLIFF-Dokuments

Muster-Vorlage @* | *

Alle Elemente und Attribute, für die es keine spezielle Vorlage gibt, sind "transparent".

Muster-Vorlage lf:file/@id

Muster-Vorlage lf:segment

Eine Übersetzungseinheit

Verwendete globale Parameter oder Variable:

Parameter p_targetlang

Muster-Vorlage lf:source

Zu übersetzender Text

Aufgerufene benannte Vorlagen:

txt.split

Verwendete Modus:

srctxt

Verwendete globale Parameter oder Variable:

Variable g_textline

Muster-Vorlage xsl:template [@name = 'textline'], txt.apply

Name: textline

Parameter

txt

"Callback"-Vorlage für Textzeilen

Aufgerufene benannte Vorlagen:

txt.replacelist

Verwendete globale Parameter oder Variable:

Variable g_rlist

Muster-Vorlage *, srctxt

Verwendete Modus:

srctxt

Muster-Vorlage lf:mrk [@translate = 'no'], srctxt

Nicht übersetzbare oder nicht zu übersetzende Ausschnitte des Quelltextes werden durch Platzhalter ersetzt.

Muster-Vorlage text(), srctxt

Modus

Modus srctxt

Der zu übersetzende Quelltext

Die folgenden Vorlagen implementieren den Modus srctxt:

Muster-Vorlage *, srctxt
Muster-Vorlage lf:mrk [@translate = 'no'], srctxt
Muster-Vorlage text(), srctxt

Der Modus srctxt wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage lf:source
Muster-Vorlage *, srctxt

Modus txt.apply

Die folgenden Vorlagen implementieren den Modus txt.apply:

Muster-Vorlage xsl:template [@name = 'textline'], txt.apply

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  XLIFF zu einfachem Text (Übersetzungstext)
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:lf  = "urn:oasis:names:tc:xliff:document:2.0"
  version   = "1.0"
>
<xsl:include href = "/pool/txt.xslt"/>


<xsl:param name = "p_targetlang"/>


<xsl:output method = "text" encoding = "utf-8"/>


<xsl:variable name = "g_rlist" select = "concat (
  '. |.', $txt.break,
  '|: |:', $txt.break,
  '|! |!', $txt.break,
  '|? |?', $txt.break
)"/>


<xsl:variable
  name   = "g_textline"
  select = "document('')/xsl:stylesheet/xsl:template [@name = 'textline']"
/>


<xsl:template match = "/">
  <xsl:apply-templates select = "*"/>
</xsl:template>


<xsl:template match = "@* | *">
  <xsl:apply-templates select = "@* | *"/>
</xsl:template>


<xsl:template match = "lf:file/@id">
  <xsl:text>=file </xsl:text>
  <xsl:choose>
    <xsl:when test = "starts-with(., 'f_')">
      <xsl:value-of select = "substring (., 3)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select = "."/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:value-of select = "$txt.break"/>
</xsl:template>
```

```
<xsl:template match = "lf:segment">
  <xsl:text>=segment </xsl:text>
  <xsl:choose>
    <xsl:when test = "starts-with (@id, 's_')">
      <xsl:value-of select = "substring (@id, 3)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select = "@id"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:value-of select = "$txt.break"/>
  <xsl:apply-templates select = "*"/>
  <xsl:if test = "string-length ($p_targetlang) &gt; 0">
    <xsl:value-of select = "concat ('=trans ', $p_targetlang, $txt.break)"/>
    <xsl:text># Hier bitte die Übersetzung einfügen</xsl:text>
    <xsl:value-of select = "concat ($txt.break, $txt.break)"/>
  </xsl:if>
</xsl:template>


<xsl:template match = "lf:source">
  <xsl:value-of select = "concat ('=source ', @xml:lang, $txt.break)"/>
  <xsl:variable name = "st">
    <xsl:apply-templates select = "." mode = "srctxt"/>
  </xsl:variable>
  <xsl:call-template name = "txt.split">
    <xsl:with-param name = "txt"  select = "$st"/>
    <xsl:with-param name = "elem" select = "$g_textline"/>
  </xsl:call-template>
  <xsl:if test = "lf:mrk">
    <xsl:text>=struct</xsl:text>
    <xsl:for-each select = "lf:mrk">
      <xsl:text> </xsl:text>
      <xsl:choose>
        <xsl:when test = "starts-with (@id, 'm_')">
          <xsl:value-of select = "substring-before (substring (@id, 3), '_')"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select = "@id"/>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:if test = "@type = 'herbaer:move'">?</xsl:if>
    </xsl:for-each>
    <xsl:value-of select = "$txt.break"/>
  </xsl:if>
</xsl:template>


<xsl:template match = "xsl:template [@name = 'textline']" name = "textline" mode = "txt.apply"
>
  <xsl:param name = "txt"/>
  <xsl:variable name = "t" select = "normalize-space ($txt)"/>
  <xsl:if test = "string-length ($t) &gt; 0">
    <xsl:if test = "starts-with ($t, '=') or starts-with ($t, '#')">
      <xsl:text>\</xsl:text>
    </xsl:if>
    <xsl:call-template name = "txt.replacelist">
      <xsl:with-param name = "txt" select = "$t"/>
      <xsl:with-param name = "sep" select = "'|'"/>
      <xsl:with-param name = "list" select = "$g_rlist"/>
    </xsl:call-template>
    <xsl:value-of select = "$txt.break"/>
  </xsl:if>
</xsl:template>


<xsl:template match = "*" mode = "srctxt">
  <xsl:apply-templates mode = "srctxt"/>
</xsl:template>


<xsl:template match = "lf:mrk [@translate = 'no']" mode = "srctxt">
  <xsl:text>XK</xsl:text>
  <xsl:choose>
    <xsl:when test = "starts-with (@id, 'm_')">
      <xsl:value-of select = "substring-before (substring (@id, 3), '_')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select = "@id"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>KX</xsl:text>
</xsl:template>


<xsl:template match = "text()" mode = "srctxt">
  <xsl:value-of select = "."/>
</xsl:template>


</xsl:stylesheet>
```

Dateiformat für Übersetzungstexte

Anwendung des Dateiformats

Das hier beschriebene Dateiformat (Übersetzungstext) dient als Ausgabe und Eingabe von Übersetzungsprogrammen.

Die Transformation `xliff_txt.xslt` erzeugt aus einem XLIFF-Dokument eine Übersetzungstext-Datei.
Das Programm `mttext.pl` nutzt das Dateiformat zur Ein- und Ausgabe.
Das Programm `resstruct.pl` nutzt das Dateiformat zur Eingabe übersetzter Texte.

Übersetzungstext-Dateiformat

Die Übersetzungstext-Dateien bestehen aus drei Arten von Zeilen: Kommentarzeilen, Strukturzeilen und Textzeilen.

Kommentarzeilen

beginnen mit dem Zeichen #. Sie haben keine Bedeutung für die automatische Verarbeitung der Datei, enthalten aber Hinweise für einen menschlichen Übersetzer.

Strukturzeilen

beginnen mit dem Zeichen `=`, dem ein Schlüsselwort und mögliche weitere Informationen folgen.

`=machine` *`TGTLANG MACHINE`*

Diese Zeile zeigt an, dass im Folgenden zur Übersetzung in die Zielsprache *`TGTLANG`* der Übersetzer *`MACHINE`* benutzt worden ist oder benutzt werden soll. Ich benutze diese Zeile nur in der Ausgabe des Programm `mttext.pl`. Dabei ist *`MACHINE`* die Kennung eines Übersetzungsmoduls.

`=file` *`FILEID`*

`FILEID` ist der Wert des Attributs `file/@id` in der XLIFF-Datei ohne das Präfix `f`. Es gilt für die folgenden Zeilen bis zur nächten `=file`-Zeile.

`=segment` *`ID`*

`ID` ist der Wert des Attributs `segment/@id` ohne das Präfix `s`. Der Quelltext des `segment`-Elements (`segment/source`) enthält kein `mrk`-Element. Die Zeile bezieht sich auf die folgenden Zeilen bis zur nächsten `=segment`- oder `=struct`-Zeile.

`=source` *`SRCLANG`*

Die folgenden Textzeilen bis zur nächsten Strukturzeile enthalten den zu übersetzenden Text (Quelltext) des Segments, dessen Kennung die letzte vorangehende `=segment`-Zeile angibt. *`SRCLANG`* bezeichnet die Sprache des Quelltextes. Die Angabe der Sprache kann entfallen. In diesem Fall gilt die zuletzt angegebene Quellsprache bis zur Angabe einer neuen Quellsprache.

`=trans` *`TGTLANG`*

Die folgenden Textzeilen bis zur nächsten Strukturzeile enthalten den übersetzten Text des Segments, dessen Kennung die letzte vorangehende `=segment`-Zeile angibt. *`TGTLANG`* bezeichnet die Sprache, in die der Quelltext übersetzt ist. Die Angabe der Sprache kann entfallen. In diesem Fallt gilt die zuletzt angegebene Zielsprache bis zur Angabe einer neuen Zielsprache.

`=struct ID?` ...

`ID` und mögliche folgende Werte sind Kennungen von Übersetzungsabschnitten (`section`-Elementen) ohne das Präfix `s`. Die Übersetzungsabschnitte mit der Kennung `ID` und den folgenden Kennungen sind im Übersetzungsabschnitt, den die vorausgehende `=segment`-Zeile bezeichnet, in der genannten Reihenfolge enthalten.

Ein Fragezeichen hinter einer Kennung zeigt an, dass der übersetzte Textteil nicht unbedingt im übersetzten Text gefunden werden muss. Zum Beispiel wird in der chinesischen Übersetzung von „`Guten Tag!`” die chinesische Übersetzung von „`Tag`” nicht gefunden:

```
=segment e1
=source de
Guten Tag!
=trans zh
##
=struct e2?
=segment e2
=source de
Tag
=trans zh
#
```

Das Skript `resstruct.pl` meldet keinen Fehler, wenn es die Übesetzung eines „optionalen” Teiltextes im übesetzten Text nicht findet.

Textzeilen

Zeilen, die weder mit # noch mit = beginnen, (Textzeilen) enthalten den zu übersetzenden Text oder die Übersetzung. Das Zeichen \ am Anfang einer Textzeile wird entfernt. Es „schützt” die Zeichen # und =, die auf diese Weise effektiv am Anfang einer Textzeile möglich sind.

mttext.pl

[Quelltext]

Übersicht

`mttext.pl --help|--version`

`mttext.pl[ --verbose…|--no_verbose ]`
`[--in` *`IN`*`][--out` *`OUT`*`][--trname` *`TRNAME`*`][--srclang` *`SRCLANG`*`][--tgtlang` *`TGTLANG`*`]`

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Erhöht den Umfang der Meldungen nach STDERR.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--in` *`IN`*

Pfad der Eingabedatei oder `-` für `STDIN`. Die Eingabe ist eine Übersetzungstext-Datei (s. Lokalisierung). In der Eingabe bereits enthaltene Übersetzungen werden ignoriert.

`--out` *`OUT`*

Pfad der Ausgabedatei oder `-` für `STDOUT`. Die Eingabe ist eine Übersetzungstext-Datei, die anstelle der Text in der Ausgangssprache übersetzte Texte enthält (`=trans` - Zeilen und folgende Textzeilen). `=segment`- und `=struct`-Zeilen in der Eingabe werden übernommen.

`--trname` *`TRNAME`*

Name des Übersetzers. Er bestimmt das Perl-Modul, das zur Übersetzung benutzt wird. Einzelheiten s. `Herbaer::Translate` (`Translate.pm`).

`--srclang` *`SRCLANG`*

Die Kennung der Quellsprache, sofern in einer `=source`-Zeile nicht eine andere Sprache angegeben ist.

`--tgtlang` *`TGTLANG`*

Die Kennung der Zielsprache, in die die Texte übersetzt werden.

Beschreibung

Die Eingabe und die Ausgabe sind Übersetzungstext-Dateien. Die Ausgabe enthält die Übersetzungen der Texte in der Eingabe.

Software-Voraussetzungen

Das Programm ist mit Perl Version 5.10.1 entwickelt. Es benutzt die folgenden Module:

`Herbaer::Readargs`

Die Funktionen `read_args` aus diesem Modul verarbeitet die Befehlszeilenargumente, die Funktion `print_message_with_values` gibt die Hilfe mit den aktuellen Einstellungen aus.

`Herbaer::Translate`

Die Methode `new` liefert den Übersetzer.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# maschinelle Übersetzung
# 2015-01-24 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


# 2015-07-02 Zeile =machine LANG TRANSLATOR
# 2015-11-02 $trans -> translator_name()


package main;


use utf8 ;
use Herbaer::Readargs ;
use Herbaer::Translate ;


binmode (STDIN,  ":utf8") ;
binmode (STDOUT, ":utf8") ;
binmode (STDERR, ":utf8") ;


# Hash der Kommandozeilen-Argumente
my $args = {
   "[cnt]verbose" => 1,
   "in"           => "-",           # Eingabedatei (Text) oder - für STDIN
   "out"          => "-",           # Ausgabedatei oder - für STDOUT
   "trname"       => "default",     # Übersetzer
   "srclang"      => "de",          # Default-Quellsprache
   "tgtlang"      => "tc",          # Zielsprache
};


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
mttext.pl v20150124
Maschinelle Übersetzung
2015 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
}
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   print_message_with_values (<<'HELP', $args);
mttext.pl [Optionen]
--[no_]verbose    Umfang der Meldungen ${[cnt]verbose}
--in IN           Pfad der Eingabedatei oder "-" für STDIN
                  ${in}
--out OUT         Pfad der Ausgabedatei oder "-" für STDOUT
                  ${out}
--trname TRNAME   Übersetzer ${trname}
--srclang SRCLANG Default-Quellsprache ${srclang}
--tgtlang TGTLANG Zielsprache ${tgtlang}
HELP
   exit 0;
};


read_args ($args);
run ($args);
exit 0;
```

```
sub run {
   my $args = shift ;
   my $verb = $args -> {"[cnt]verbose"};
   my $in   = $args -> {"in"};
   my $dsl  = $args -> {"srclang"};
   my $tgl  = $args -> {"tgtlang"};
   print "mttext.pl\n" if $verb;
   my $trans = new Herbaer::Translate ($args -> {"trname"}) ;
   if (! $trans) {
      print STDERR "Kann Übersetzer " . $args -> {"trname"} . " nicht laden\n";
      exit 1;
   }
   my $hin;
   if ($in eq "-") {
      $hin = STDIN;
   }
   elsif (! open ($hin, "<:encoding(utf-8)", $in)) {
      print STDERR "Kann \"$in\" nicht lesen: $!\n";
      exit 1;
   }
   my $out = $args -> {"out"};
   my $hout;
   if ($out eq "-") {
      $hout = STDOUT;
   }
   elsif (! open ($hout, ">:encoding(utf-8)", $out)) {
      print STDERR "Kann \"$out\" nicht öffnen: $!\n";
      exit 1;
   }
   my $text;     # zu übersetzender Text
   my $srclang;  # Sprache der Quelle
   my $mtt;      # maschinell übersetzter Text
   my $line;     # Zeile der Eingabe
   my $l2;       # Textzeile ohne führendes "\"
   my $read = 0; # Text lesen?

   my $trtext = sub {  # Text übersetzen
      $text =~ s/\s*$//;
      $text =~ s/\s+/ /g;
      $mtt   = undef ;
      $mtt ||= $trans -> translate ($text, $srclang, $tgl);
      if ($mtt) {
         $mtt =~ s/\s+$// ;
         $mtt =~ s/\n([=#])/\n\\$1/g ;
         $mtt =~ s/$([=#])/\\$1/ ;
         print $hout "=trans $tgl\n$mtt\n";
      }
      $text = "";
   };

   while (defined ($line = <$hin>)) {
      if ( $line =~ /^#/ ) {
      }
      elsif ( $line =~ /^=/ ) {
         $read = 0;
         $trtext -> () if $text;
         if ( $line =~ /^=segment/ ) {
            print $hout $line;
         }
         elsif ( $line =~ /^=struct/ ) {
            print $hout $line;
         }
         elsif ( $line =~ /^=source\s+(\S*)/ ) {
            $text = "";
            $read = 1;
            $srclang = $1 || $dsl;
         }
      }
      elsif ( $read ) {
         $l2 = $line;
         $l2 =~ s/^\\//;
         $text .= $l2;
      }
   }
   $trtext -> () if $read && $text;
   print $hout "=machine $tgl ",
     ($trans -> translator_name() || $args -> {"trname"}),
     "\n";
   close $hin  if $in  ne "-";
   close $hout if $out ne "-";
   $trans -> finish ();
} # run
```

resstruct.pl

[Quelltext]

Übersicht

`resstruct.pl --help|--version`

`resstruct.pl[ --verbose… | --no_verbose ]`
`[--in` *`IN`*`][--out` *`OUT`*`][--tgtlang` *`TGTLANG`*`][--ptnfix` *`PTNFIX`*`]`

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Erhöht den Umfang der Meldungen nach STDERR.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--in` *`IN`*

Pfad der Eingabedatei oder `-` für `STDIN`. Die Eingabe ist eine Übersetzungstext-Datei mit Übersetzungen in die Zielsprache (*`TGTLANG`*). Normalerweise ist die Eingabe die Ausgabe des Programms `mttext.pl`.

`--out` *`OUT`*

Pfad der Ausgabedatei oder `-` für `STDOUT`. Die Ausgabe ist ein XML-Dokument, das den Namensraum http://herbaer.de/xmlns/20150127/resstruct# (s. `resstruct.rng`) nutzt.

`--tgtlang` *`TGTLANG`*

Die Kennung der Zielsprache.

`--ptnfix` *`PTNFIX`*

Die Übersetzungstext-Datei enthält Platzhalter für unveränderlichen Text. Der Platzhalter wird gebildet, indem in *`PTNFIX`* der Platzhalter `${id}` durch die ID des unveränderlichen Texts ersetzt wird. Im übersetzten Text wird der Platzhalter (hoffentlich) gefunden und der ID zugeordnet.

Software-Voraussetzungen

Das Programm ist mit Perl Version 5.10.1 entwickelt. Es benutzt die folgenden Module:

`Herbaer::Readargs`

Die Funktionen `read_args` aus diesem Modul verarbeitet die Befehlszeilenargumente, die Funktion `print_message_with_values` gibt die Hilfe mit den aktuellen Einstellungen aus.

Das Modul ist im Zusammenhang mit den Kalendern beschrieben.

`Herbaer::Replace`

Die Funktionen `replace` aus diesem Modul (`Replace.pm`) setzt die ID in den Platzhalter für unveränderlichen Text ein.

`Herbaer::XMLDataWriter`

Wird zur Ausgabe benutzt.

Das Modul ist im Zusammenhang mit den Kalendern beschrieben.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# Verschachtelungen in der Übersetzung auflösen
# 2015-01-27 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


# 2015-07-02 Name der Übersetzungs-Maschine
# 2016-03-25 Wortendungen


package main;


use utf8 ;
use Herbaer::Readargs ;
use Herbaer::Replace ;
use Herbaer::XMLDataWriter ;


binmode (STDIN,  ":utf8") ;
binmode (STDOUT, ":utf8") ;
binmode (STDERR, ":utf8") ;


# Hash der Kommandozeilen-Argumente
my $args = {
   "[cnt]verbose" => 1,
   "in"           => "-",         # Eingabedatei (Text) oder - für STDIN
   "out"          => "-",         # Ausgabedatei oder - für STDOUT
   "tgtlang"      => undef,       # Zielsprache
   "ptnfix"       => 'XK${id}KX', # Platzhalter für nicht übersetzbaren Text mit ${id}
};


my $verbose ;


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
resstruct.pl v20150127
Verschachtelungen in der Übersetzung auflösen
2015 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
}
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   print_message_with_values (<<'HELP', $args);
resstruct.pl [Optionen]
--[no_]verbose     Umfang der Meldungen ${[cnt]verbose}
--in IN            Pfad der Eingabedatei oder "-" für STDIN
                   ${in}
--out OUT          Pfad der Ausgabedatei oder "-" für STDOUT
                   ${out}
--tgtlang TGTLANG  Zielsprache ${tgtlang}
--ptnfix PTNFIX    Platzhalter für unveränderlichen Text ${ptnfix}
HELP
   exit 0;
};


read_args ($args);
$verbose = $args -> {"[cnt]verbose"};


=for comment


Ausgabe:
```

```
<translation>
   <to>zh</to>
   <text id = "4711">übersetzter Text</text>
   <struct id = "4711">
     <part>Teiltext</part>
     <ref>id1</ref>
     <part>Teiltext</part>
     <ref>id2</ref>
     <part>Teiltext</part>
     <moved>id3</moved>
     <moved>id4</moved>
   </struct>
   <error>0815</error>
   <machine>Mudel</machine>
</translation>

=cut

my $translation = {
   "to"      => $args -> {"tgtlang"},
   "text"    => {},    # id als Schlüssel
   "machine" => undef, # maschinelle Übersetzung in die Zielsprache?

   # ebenfalls id als Schlüssel
   # die Werte sind eine Liste von {"part" => "TEXT"} oder {"ref" => "IDREF"}
   "struct"  => {},

   # Liste der ID zu "unlösbaren" Strukturen
   "error"   => [],
};

# Die Werte sind Listen von ID
my $inpstr = {};

readinp   ($args, $translation, $inpstr);
resolve   ($args, $translation, $inpstr);
writeoutp ($args, $translation);
exit 0;

# Die Eingabe lesen
sub readinp {
   my ($args, $translation, $inpstr) = @_;
   my $text = $translation -> {"text"};
   my $in = $args -> {"in"};
   my $hin;
   if ($in ne "-") {
      open ($hin, "<:encoding(utf-8)", $in);
   }
   else {
      $hin = STDIN ;
   }
   my $line;
   my $read = 0;
   my $id;
   my $txt;
   my $remach = "^=machine\\s+" . quotemeta ($args -> {"tgtlang"}) . "\\s+(.*?)\\s*\$" ;
   my $retrans = "^=trans\\s+" . quotemeta ($args -> {"tgtlang"}) ;
   my $refs;      # referenzierte IDs als Textliste
   while ( defined ($line = <$hin>) ) {
      if ( $line =~ /^#/ ) {
      }
      elsif ( $line =~ /^=/ ) {
         if ($id && $read) {
            $txt =~ s/\s*$// ;
            $txt =~ s/\s+/ /g ;
            $text -> {$id} = $txt;
            $txt  = "";
            $read = 0;
         }
         $read = 0;
         if ( $line =~ /$remach/ ) {
            $translation -> {"machine"} = $1 ;
         }
         elsif ( $line =~ /^=segment\s+(\S+)/ ) {
            $id = $1;
         }
         elsif ( $line =~ /$retrans/ ) {
            $read = 1;
         }
         elsif ( $line =~ /^=struct\s+(.+)$/ ) {
            $refs = $1;
            $refs =~ s/\s+$//;
            $inpstr -> {$id} = [split (/\s+/, $refs)] ;
         }
      }
      elsif ( $read ) {
         $line =~ s/^\\// ;
         $txt .= $line;
      }
   }
   if ($id && $read) {
      $txt =~ s/\s*$// ;
      $txt =~ s/\s+/ /g ;
      $text -> {$id} = $txt;
```

```
      $txt = "";
      $read = 0;
   }
   close $hin if $in ne "-";
} # readinp


# Die Verschachtelungen auflösen
sub resolve {
   my ($args, $translation, $inpstr) = @_ ;
   my $ptnfix = $args -> {"ptnfix"};    # Platzhalter für unveränderlichen Text
   my $text   = $translation -> {"text"};
   my $struct = $translation -> {"struct"};
   my $errlst = $translation -> {"error"};
   my $id;
   my $refs;
   my $ref;      # ID-Referenz mit optionalem Fragezeichen
   my $refid;    # die referenzierte ID
   my $rm;       # verbleibender Text
   my $opt;      # Optionale Referenz?
   my $tt;       # Teiltext
   my $rtext;    # referenzierter Text
   my $retext;   # RegEx zum referenzierten Text

   # Ergebnisliste mit Einträgen {"part" => text},  {"ref" => id}, {"moved" => id}
   my $list;

   while ( ($id, $refs) = each %$inpstr ) {

      # Hier sind andere Verfahren nötig.
      # Die Reihenfolge der Kindelemente hat meist keine Bedeutung.
      # Und wenn es auf die Reihenfolge ankommt,
      # kann die richtige Reihenfolge außerhalb des Zusammenhangs der Übersetzung
      # überprüft und sichergestellt werden.

      $list = [];
      $rm = $text -> {$id} or next;

      print STDERR "ID $id\n", "TEXT $rm\n" if $verbose;
      # Für jeden Verweis erzeugen wir einen Hash mit Einträgen
      # refid, rtext, length, moveable, pos, mcount, matches ...
      # matchpos ist eine Liste von Positionen, an denen rtext gefunden wird.
      my $refhsh;

      # Wir erzeugen zu jedem MATCH einen Hash mit Einträgen
      # beg, end, refhsh
      my $matchhsh;

      # Wir erzeugen Listen der $refhsh und $matchhsh
      my $refhlist   = [];
      my $matchhlist = [];

      # Die Datenstruktur muss wegen der Verweiszyklen dekonstruiert werden.

      my $pos = -1;     # Position des Verweises in der Ausgangsliste
      my $matches;      # Liste der Matches
      my $len;          # Länge des Verweistextes  # ***  hängt vom Match ab
      my $mcount;       # Anzahl der Match-Treffer
      my $pend = 0;     # Ende des bevorzugten Matches des vorhergehenden refs
      my $nend;         # Ende des bevorzugten Matches des refs
      my $fend;         # Ende des ersten Matches des refs
      my $nreqrefs = 0; # Anzahl der Verweise, die erforderlich sind
      my $noptrefs = 0; # Anzahl der optionalen (verschiebbaren) Verweise
      for $ref (@$refs) {
         $refid = $ref;
         $opt = 0;
         $matches = [];
         if ($refid =~ s/\?//) { $opt = 1; ++$noptrefs; }
         else                  { ++$nreqrefs; }
         $refhsh = {
            "refid"    => $refid,
            "moveable" => $opt,
            "matches"  => $matches,
            "pos"      => ++$pos,
         };
         push (@$refhlist, $refhsh);
         $rtext = $text -> {$refid};
         if (! defined ($rtext)) {
            $rtext = replace ($ptnfix, {"id" => $refid});
            $refhsh -> {"fix"} = 1;
            $len   = length ($rtext);
            $rtext = quotemeta ($rtext);
            $retext = qr/($rtext)/;
         }
         elsif ($rtext =~ /^(\S{2,})([a-z])([a-z])$/ ) {
            $retext = '\b(' . quotemeta ($1) . $2 . "?" . $3 . "?" . '[a-z]{0,2})\b';
            $retext = qr/$retext/i ;
            # Wort mit Endung
         }
         else {
            $len   = length ($rtext);
            $rtext = quotemeta ($rtext);
            $retext = qr/($rtext)/i;
         }
         $mcount = 0;
         $fend   = 0;
         $nend   = 0;
```

```
         while ( $rm =~ /$retext/g ) {
            $len = length ($1);
            $matchhsh = {
               "beg"    => pos ($rm) - $len,
               "end"    => pos ($rm),
               "length" => $len,
               "refhsh" => $refhsh,
            };
            $fend = $matchhsh -> {"end"} if !$fend;
            if ($matchhsh -> {"beg"} >= $pend && ! $nend) {
               $refhsh   -> {"prefmatch"} = $matchhsh;
               $matchhsh -> {"pref"} = 1;
               $nend = $matchhsh -> {"end"};
            }
            push (@$matches, $matchhsh);
            push (@$matchhlist, $matchhsh);
            ++$mcount;
         }
         if (!$nend && $fend) {
            $refhsh -> {"prefmatch"} = $matches -> [0];
            $pend = $fend;
         }
         else {
            $pend = $nend;
         }
         $refhsh -> {"mcount"} = $mcount;
      }
      print STDERR "ID $id: Verweisliste erzeugt\n" if $verbose;

      # wir sortieren die Matchliste nach Beginn, ...
      $matchhlist = [
         sort
         {
            $a -> {"beg"} <=> $b -> {"beg"}
            || $a -> {"refhsh"} -> {"mcount"}   <=> $b -> {"refhsh"} -> {"mcount"}
            || $a -> {"refhsh"} -> {"moveable"} <=> $b -> {"refhsh"} -> {"moveable"}
            || $a -> {"end"} <=> $b -> {"end"}
            || $a -> {"refhsh"} -> {"pos"} <=> $b -> {"refhsh"} -> {"pos"}
         }
         @$matchhlist
         ];
      print STDERR "ID $id: Verweisliste sortiert\n" if $verbose;

      # Jetzt suchen wir eine "stimmige" Teilliste der Matchliste
      if (! resolve_text ($matchhlist, $nreqrefs, $noptrefs) ) {
         push (@$errlst, $id);
         print STDERR "ID $id: Verweise nicht aufgelöst\n" if $verbose;
      }
      else {
         # wir bauen die Ausgabe-Liste auf
         # Ergebnisliste mit Einträgen {"part" => text},  {"ref" => id}, {"moved" => id}
         my $textpos = 0;
         my $len;
         $struct -> {$id} = $list;
         for $matchhsh (@$matchhlist) {
            if ($matchhsh -> {"selected"}) {
               $len = $matchhsh -> {"beg"} - $textpos;
               push (@$list, {"part" => substr ($rm, $textpos, $len)}) if $len;
               $refhsh = $matchhsh -> {"refhsh"};
               push (@$list, {"ref"  => $refhsh -> {"refid"}});
               if (!$refhsh -> {"fix"}) {
                  $text -> {$refhsh -> {"refid"}} =
                     substr ($rm, $matchhsh -> {"beg"}, $matchhsh -> {"length"});
               }
               $textpos = $matchhsh -> {"end"};
            }
         }
         push (@$list, {"part" => substr ($rm, $textpos)}) if $textpos < length ($rm);
         for $refhsh (@$refhlist) {
            push (@$list, {"moved" => $refhsh -> {"refid"}}) if ! $refhsh -> {"resolved"};
         }
      }

      # wir lösen zyklische Verweise auf
      for $matchhsh (@$matchhlist) {
         $matchhsh -> {"refhsh"} = undef;
      }
   }
} # resolve
```

```
# Wir suchen eine "stimmige" Teilliste der Matchliste
# my $reslist = resolve_text ($matchhlist)
# Die ausgewählten Matches bekommen die Markierung "selected" = 1;
sub resolve_text {
   my ($matchhlist, $nreqrefs, $noptrefs, $pos, $textpos, $allrefs) = @_;
   $pos     ||= 0;
   $textpos ||= 0;
   $allrefs ||= 0;
   # nreqrefs  Anzahl der nicht-optionalen Verweise
   # noptrefs  Anzahl der optionalen Verweise
   # pos       Position in der Liste matchhlist, ab der gesucht werden soll
   # textpos   Textposition, ab der Verweise möglich sind
   # allrefs   Fehler, wenn ein optionaler Verweis nicht gelöst wird

   print STDERR "resolve_text ($nreqrefs, $noptrefs, $pos, $textpos, $allrefs)\n"
      if $verbose;

   # Ende der Liste erreicht
   if ($pos >= @$matchhlist) {
      return ! $nreqrefs && (! $noptrefs || ! $allrefs);
   }

   # alle Verweise gelöst?
   if (!$nreqrefs && !$noptrefs) {
      while ($pos < @$matchhlist) {
         $matchhlist -> [$pos] -> {"selected"} = 0;
         ++$pos;
      }
      return 1;
   }

   my $match  = $matchhlist -> [$pos];
   my $refhsh = $match -> {"refhsh"};

   # Ist der Verweis schon gelöst?
   if ($refhsh -> {"resolved"}) {
      $match -> {"selected"} = 0;
      return resolve_text ($matchhlist, $nreqrefs, $noptrefs, ++$pos, $textpos, $allrefs);
   }

   my $res; # Ergebnis
   # Ist der Match schon "hinter dem Mond"?
   if ($match -> {"beg"} < $textpos) {
      -- $refhsh -> {"mcount"};
      $match -> {"selected"} = 0;
      $res = resolve_text ($matchhlist, $nreqrefs, $noptrefs, ++$pos, $textpos, $allrefs);
      ++ $refhsh -> {"mcount"};
      return $res;
   }

   if ($refhsh -> {"mcount"} == 1) {
      if ($refhsh -> {"moveable"}) {
         # wir versuchen erst, den Match auszuwählen
         $match  -> {"selected"} = 1;
         $refhsh -> {"resolved"} = 1;
         $res = resolve_text ($matchhlist, $nreqrefs, $noptrefs - 1, $pos + 1, $match -> {"end"}, 1);
         if ($allrefs || $res) {
            return $res;
         }
         if (!$res) {
            deselect ($matchhlist, $pos + 1);
            $res = resolve_text ($matchhlist, $nreqrefs, $noptrefs - 1, $pos + 1, $match -> {"end"}, 0);
            if ($res) {
               return $res;
            }
         }
         # Dann nicht
         deselect ($matchhlist, $pos);
         return resolve_text ($matchhlist, $nreqrefs, $noptrefs, $pos + 1, $textpos, 0);
      }
      else {
         # der Match muss ausgewählt werden
         $match  -> {"selected"} = 1;
         $refhsh -> {"resolved"} = 1;
         return resolve_text ($matchhlist, $nreqrefs - 1, $noptrefs, $pos + 1, $match -> {"end"}, $allrefs);
      }
   }

   if ( $refhsh -> {"prefmatch"} -> {"beg"} > $match -> {"beg"} ) {
      # wir probieren erst, den Match nicht auszuwählen
      -- $refhsh -> {"mcount"};
      $res = resolve_text ($matchhlist, $nreqrefs, $noptrefs, $pos + 1, $textpos, $allrefs);
      ++ $refhsh -> {"mcount"};
      if ($res) {
         return $res;
      }
      deselect ($matchhlist, $pos + 1);
   }

   # wir wählen den Match aus
   $match  -> {"selected"} = 1;
   $refhsh -> {"resolved"} = 1;
   -- $refhsh -> {"mcount"};
   $res =
      $refhsh -> {"moveable"} ?
      resolve_text ($matchhlist, $nreqrefs, $noptrefs - 1, $pos + 1, $match -> {"end"}, $allrefs) :
```

```
      resolve_text ($matchhlist, $nreqrefs - 1, $noptrefs, $pos + 1, $match -> {"end"}, $allrefs)
      ;
   ++ $refhsh -> {"mcount"};
   if ($res) {
      return $res;
   }

   if ( $refhsh -> {"prefmatch"} -> {"beg"} <= $match -> {"beg"} ) {
      deselect ($matchhlist, $pos);
      -- $refhsh -> {"mcount"};
      $res = resolve_text ($matchhlist, $nreqrefs, $noptrefs, $pos + 1, $textpos, $allrefs) ;
      ++ $refhsh -> {"mcount"};
   }

   return $res;
} # resolve_text


sub deselect {
   my ($matchhlist, $pos) = @_;
   my $match;
   my $refhsh;
   while ($pos < @$matchhlist) {
      $match  = $matchhlist -> [$pos];
      $refhsh = $match -> {"refhsh"};
      if ($match -> {"selected"}) {
         $refhsh -> {"resolved"} = 0;
         $match  -> {"selected"} = 0;
      }
      ++$pos;
   }
} # deselect


# Ausgabe
sub writeoutp {
   my ($args, $translation) = @_;
   my $writer = new Herbaer::XMLDataWriter ({
      '%struct'  => ["", "segment", '@id'],
      '@segment' => ["struct", "comp"],
      '%comp'    => ["", ""],
      '%text'    => ["", "text", '@id'],
      '@error'   => ["", ""],
   });
   $writer -> open (
      $args -> {"out"}, "utf-8", "http://herbaer.de/xmlns/20150127/resstruct#"
   );
   $writer -> write ("translation", {}, $translation);
   $writer -> close ();
} # writeoutp
```

Herbaer::Replace

[Quelltext]

Übersicht

```
use Herbaer::Replace;
my $vorlage = 'Name: ${name}, Wohnort: ${ort}';
my $data    = {"name" => "Herbert", "ort" => "Dorsten"};
my $text    = replace ($vorlage, $data); # 'Name: Herbert, Wohnort: Dorsten'
```

Zweck

Das Modul `Replace.pm` (Paket `Herbaer::Replace`) exportiert die Funktion `replace`, die Platzhalter in Zeichenketten ersetzt.

Funktionen

`$text` = `replace` (*`$vorlage`*, *`$daten`*)

`$vorlage`

`$vorlage` ist eine Zeichenkette mit Platzhaltern der Form `${`*`NAME`*`}` oder eine Referenz. *`NAME`* steht für eine nicht-leere Kette von Kleinbuchstaben „a” bis „z”, Großbuchstaben „A” bis „Z” oder der Zeichen „_” (Unterstrich), „`-`” (Minus) oder „`.`” (Punkt).

`$daten`

`$daten` ist eine Referenz vom Typ HASH oder ARRAY. Wenn *`$daten`* eine HASH-Referenz ist, wird ein Platzhalter `${`*`NAME`*`}` in der Zeichenkette *`$vorlage`* durch `$daten -> {'`*`NAME`*`'}` ersetzt.

Wenn *`$daten`* keine HASH-Referenz ist oder keinen Eintrag enthält, ist das Ergebnis *`$vorlage`* unverändert.

Wenn *`$vorlage`* eine Referenz ist, dann ist das Ergebnis *`$vorlage`*, aber der referenzierte Wert kann geändert sein.

Wenn *`$vorlage`* eine *`HASH`*-Referenz ist, wird für jeden Schlüssel *`KEY`*, für den der Wert `$vorlage -> {`*`KEY`*`}` definiert ist, dieser Wert durch den Wert `replace (`*`$vorlage -> {KEY}, $daten`*`)` ersetzt.

Wenn *`$vorlage`* eine *`ARRAY`*-Referenz ist, wird jeder definierte Wert *`VALUE`* im Array durch den Wert `replace (`*`VALUE, $daten`*`)` ersetzt.

Wenn *`$vorlage`* eine *`SCALAR`*-Referenz ist, wird der Variablen, auf die *`$vorlage`* verweist, der Wert `replace (`*`$$vorlage, $daten`*`)` zugewiesen.

In den weiteren beschriebenen Fällen ist *`$vorlage`* kein Referenz-Typ. Wenn *`$daten`* eine ARRAY-Referenz ist, dann ist das Ergebnis ebenfalls eine ARRAY-Referenz. Die Komponenten des Ergebnisses sind die Werte `replace (`*`$vorlage, $datum`*`)`, wobei *`$datum`* alle Komponenten von *`$daten`* durchläuft.

Wenn *`$daten`* eine HASH-Referenz ist, dann werden in der Zeichenketten-Darstellung von *`$vorlage`* Platzhalter der Form `${`*`NAME`*`}` ersetzt. Der vorläufige Ersatzwert `$d` ist *`$data`* `-> {"`*`NAME`*`"}`, falls der HASH-Eintrag existiert, sonst der „fallback”-Wert *`$data`* `-> {"*"}`. Solange `$d` eine gültige CODE-Referenz ist, wird `$p` durch das Ergebnis des Funktionsaufrufs ersetzt: `$d = $d -> ("`*`NAME`*`")`. Wenn `$d` schließlich definiert ist und keine Referenz ist, wird der Platzhalter durch den Wert von `$d` ersetzt. Andernfalls bleibt der Platzhalter unverändert. Das Ergebnis ist eine Zeichenkette mit den eingesetzten Werten anstelle der Platzhalter.

Beispiel

Der folgende Code

```
use Herbaer::Replace ;

my $vwnr = 0;
my $vwmodelle = ["Golf", "Passat", "Leguan oder so?"];

sub vw {
   my $m = $vwmodelle -> [$vwnr++];
   $vwnr %= 3;
   $m;
};

my $vorl = "\${auto}: \${modell}\n";
my $ref  = \$vorl;
my $data = [
   {"auto" => "VW - Das Auto", "modell" => \&vw },
   {"auto" => "VW - Das Auto", "modell" => \&vw },
   {"auto" => "VW - Das Auto", "modell" => \&vw },
   {"auto" => "BMW",  "*" => "Aus Freude am Fahren" },
   {"auto" => "Opel", "*" => "\"Opel fahn iss wie wenze fliechs\"" },
   ];
replace ($ref, $data);
for my $t (@$$ref) {
   print $t;
}
```

gibt aus:

```
VW - Das Auto: Golf
VW - Das Auto: Passat
VW - Das Auto: Leguan oder so?
BMW: Aus Freude am Fahren
Opel: "Opel fahn iss wie wenze fliechs"
```

Installation

Die Datei `Replace.pm` wird unter dem Teilpfad `Herbaer/Replace.pm` des Suchpfades für Perl-Module (`@INC`) gesucht. Auf meinem Rechner ist `/usr/local/share/perl/5.10.1/Herbaer/Replace.pm` ein symbolischer Verweis.

Anwendungen

Neben anderen Programmen nutzt `replace.pl` dieses Modul.
x

Quelltext

[Beschreibung]

```
# Einfache Platzhalter in Zeichenketten ersetzen
# 2015-10-16 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Replace ;


BEGIN {
    use Exporter;
    our $VERSION = 20151016;
    our @ISA     = qw (Exporter);
    our @EXPORT  = qw (replace);
}


# "intern" von replace verwendet
sub _r {
   my ($key, $v) = @_;
   my $d = exists $v -> {$key} ? $v -> {$key} : $v -> {"*"};
   while ( defined ($d) && ref $d eq "CODE" ) {
      $d = $d -> ($key);
   }
   defined $d && ! ref $d ? $d : "\$\{$key\}";
}


# füllt Platzhalter
sub replace {
   my ($str, $vals) = @_;
   return $str unless ref $vals eq "HASH" || ref $vals eq "ARRAY" ;
   if (ref $str) {
      if (ref $str eq "HASH") {
         my ($k, $v);
         while ( ($k, $v) = each %$str ) {
            $str -> {$k} = replace ($v, $vals) if $v;
         }
      }
      elsif (ref $str eq "ARRAY") {
         my $v;
         foreach $v (@$str) {
            $v = replace ($v, $vals) if $v;
         }
      }
      elsif (ref $str eq "SCALAR") {
         $$str = replace ($$str, $vals);
      }
      $str;
   }
   elsif (ref $vals eq "ARRAY") {
      [ map { replace ($str, $_) } @$vals ];
   }
   elsif (ref $vals eq "HASH") {
      $str =~ s/\$\{([a-zA-Z0-9_.-]+)\}/_r($1, $vals)/ge if %$vals;
      $str;
   }
   else {
      $str;
   }
}


1;
```

replace.pl

[Quelltext]

Übersicht

`replace.pl --help|--version`

`replace.pl[ --verbose …|--no_verbose ][ --val` *`VAL`* `][ --docroot` *`DOCROOT`* `][ --var` *`VAR`* `…]`

```
cat datei/mit/platzhaltern         \
| replace.pl --val datei/mit/werten \
> datei/mit/eingesetzen/werten      ;
```

Beschreibung

Dieses Skript ersetzt Platzhalter der Form `${`*`NAME`*`}` durch Werte *`WERT`*, die aus einer Werte-Datei gelesen werden. *`NAME`* ist eine Folge von Buchstaben, Ziffern und Zeichen _ (Unterstrich), `.` (Punkt) und `-` (Minus-Zeichen).

Es liest den Text mit Platzhalter aus der Standard-Eingabe und gibt den Text mit den eingesetzten Werten an die Standard-Ausgabe aus.

Die Werte-Datei enthält die Werte zu den Platzhaltern in Zeilen der Form

`NAME=WERT`

Leerzeichen am Anfang und am Ende der Zeile werden entfernt. Folgen von Leerzeichen unmittelbar vor und nach dem Zeichen = werden ignoriert. Zeilen, die nicht so aufgebaut sind, werden ignoriert.

Ein Beispiel für eine Datei mit Platzhaltern ist `MySQLLookup_setup.sql`. Ein Beispiel für eine Werte-Datei ist `secrets`.

Das Skript nutzt das Modul `Herbaer::Replace` (Datei `Replace.pm`). Es ist mit Perl 5.10.1 getestet.

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Erhöht den Umfang der Meldungen nach STDERR.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--val` *`VAL`*

Dateipfad der Datei mit den einzusetzenden Werten (Werte-Datei).

`--docroot` *`DOCROOT`*

`DOCROOT` ist der Wert, der für den Platzhalter *`${docroot}`* eingesetzt wird, falls in der Datei *`VAL`* kein anderer Wert definiert wird.

`--var` *`VAR`*

Der Platzhalter `${`*`PLATZHALTER`*`}` wird durch den Wert ersetzt, der in der Datei *`VAL`* unter dem Schlüssel *`PLATZHALTER.VAR`* eingetragen ist, falls dieser existiert. Andernfalls wird der Wert eingesetzt, der unter dem Schlüsseö *`PLATZHALTER`* eingetragen ist.

Dieses Argument wird z.B. benutzt, um in .htaccess-Dateien abhängig vom Server (lokal oder entfernt) den Pfad für das Dokument-Wurzelverzeichnis einzusetzen, während die meisten anderen Platzhalter gleich sind.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# Platzhalter in einer Datei ersetzen
# 2016-07-18 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


# 2020-05-25 --docroot, --var


package main;


use utf8 ;
use Cwd qw(realpath);
use Encode;
use Herbaer::Readargs ;
use Herbaer::Replace ;


binmode (STDIN,  ":utf8") ;
binmode (STDOUT, ":utf8") ;
binmode (STDERR, ":utf8") ;


my $s = realpath($0);
$s =~ s/\/src\/localization\/replace\.pl.*/\/docroot/;


# Hash der Kommandozeilen-Argumente
my $args = {
   "[cnt]verbose" => 0,
   "val"          => "secrets", # Werte-Datei
   "docroot"      => $s,
   "var"          => [],
};


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
replace.pl
Platzhalter in einer Datei ersetzen
2020-05-25 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
}
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   print_message_with_values (<<'HELP', $args);
replace.pl [Optionen]
--[no_]verbose  Umfang der Meldungen ${[cnt]verbose}
--val  VAL      Pfad der Datei mit den Werten ${val}
--var  VAR ..   Variante ${var}
HELP
   exit 0;
};


read_args ($args);
my $verbose = $args -> {"[cnt]verbose"};


my $vals = { "docroot" => $args -> {"docroot"}, };
my $h;
my $line;
if (!open ($h, "<:encoding(utf-8)", $args -> {"val"} )) {
   print STDERR "Kann Datei \"", $args -> {"val"}, "\" nicht lesen: $!\n"
   if $verbose;
}
else {
   while (defined ($line = <$h>)) {
      print STDERR $line if $verbose;
      $line =~ s/^\s+//;
      $line =~ s/\s+$//;
      if ( $line =~ /\s*([a-zA-Z0-9_.-]+)\s*=\s*(.+)/ ) {
         $vals -> {$1} = $2;
      }
   }
   close $h;
}
print STDERR "Werte gelesen\n" if $verbose;;
```

```
if (@{$args -> {"var"}}) {
   my $r = join ("|", map {quotemeta decode ("utf-8", $_)} @{$args -> {"var"}});
   $r = qr/^(.+)\.$r$/;
   print STDERR "VAR_REGEX $r\n" if $verbose;
   for $h (keys %$vals) {
      $vals -> {$1} = $vals -> {$h} if $h =~ $r;
   }
}


$line = undef;
while (defined ($line = <STDIN>)) {
   print replace($line, $vals);
}
```

resstruct.rng - Verschachtelungen in der Übersetzung

Namespace	http://herbaer.de/xmlns/20150127/resstruct#
Wurzelelement	translation
(foreign_att)	Attribute anderer XML-Namensräume
	Enthalten in: translation, to, machine, struct, text, error, ref, part, moved
(anything)	Beliebiger Inhalt
	Enthält: (anything) (*)
	Enthalten in: (anything), (foreign_el)
(foreign_el)	Elemente anderer XML-Namensräume
	Enthält: (anything) (*)
	Enthalten in: translation, struct
translation	Wurzelelement
	Enthält: (foreign_att), (foreign_el), to, machine (?), struct (+), text (+), error (+)
	Enthalten in: Wurzel

```
<element name="translation">
  <ref name="foreign_att"/>
  <interleave>
    <ref name="foreign_el"/>
    <ref name="el_to"/>
    <optional>
        <ref name="el_machine"/>
    </optional>
    <oneOrMore>
      <ref name="el_struct"/>
    </oneOrMore>
    <oneOrMore>
      <ref name="el_text"/>
    </oneOrMore>
    <oneOrMore>
      <ref name="el_error"/>
    </oneOrMore>
  </interleave>
</element>
```

to	Die Zielsprache, in die der Text übersetzt wurde
	Enthält: Datentyp simpleword
	Enthalten in: translation

```
<element name="to">
  <ref name="foreign_att"/>
  <data type="simpleword"/>
</element>
```

machine	Name der Übersetzungsmaschine, falls maschinell übersetzt
	Enthält: Datentyp text
	Enthalten in: translation

```
<element name="machine">
  <ref name="foreign_att"/>
  <data type="text"/>
</element>
```

struct	Eine Verschachtelung mit Teiltexten und Verweisen auf andere Textteile.
	Enthält: (foreign_att), @id, (foreign_el), ref (*), part (*), moved (*)

Enthalten in: translation

```
<element name="struct">
  <ref name="foreign_att"/>
  <ref name="att_id"/>
  <interleave>
    <ref name="foreign_el"/>
    <zeroOrMore>
        <ref name="el_ref"/>
    </zeroOrMore>
    <zeroOrMore>
      <ref name="el_part"/>
    </zeroOrMore>
    <zeroOrMore>
        <ref name="el_moved"/>
    </zeroOrMore>
  </interleave>
</element>
```

text — Ein übersetzter Text

Enthält: Text, (foreign_att), @id

Enthalten in: translation

```
<element name="text">
  <ref name="foreign_att"/>
  <ref name="att_id"/>
  <text/>
</element>
```

error — ID einer Verschachtelung, die nicht "aufgelöst" werden konnte

Enthält: Datentyp simpleword

Enthalten in: translation

```
<element name="error">
  <ref name="foreign_att"/>
  <data type="simpleword"/>
</element>
```

@id — Die Kennung eines Textes oder einer Verschachtelung

Enthalten in: struct, text

ref — Verweis auf einen Text innerhalb einer Verschachtelung

Enthält: Datentyp simpleword

Enthalten in: struct

```
<element name="ref">
  <ref name="foreign_att"/>
  <data type="simpleword"/>
</element>
```

part — Teiltext in einer Verschachtelung

Enthält: Text, (foreign_att)

Enthalten in: struct

```
<element name="part">
  <ref name="foreign_att"/>
  <text/>
</element>
```

moved — Verweis auf ein verschobenes Element innerhalb einer Verschachtelung

Enthält: Datentyp simpleword

Enthalten in: struct

```
<element name="moved">
  <ref name="foreign_att"/>
  <data type="simpleword"/>
</element>
```

xliff_merge_rtr.xslt

[Quelltext]

Namensräume

Präfix	Namensraum
xml	http://www.w3.org/XML/1998/namespace
(default)	urn:oasis:names:tc:xliff:document:2.0
ti	http://herbaer.de/xmlns/201500703/transinfo/
r	http://herbaer.de/xmlns/20150127/resstruct#
lf	urn:oasis:names:tc:xliff:document:2.0
d	http://herbaer.de/xmlns/20051201/doc
xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Wir wünschen eine schön formatierte Ausgabe.

Method	xml
Indent	yes
Encoding	utf-8

Parameter

Parameter p_verbose

Meldungen bei Problemen?

Select: 'yes'

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage lf:segment/lf:source
Muster-Vorlage r:ref | r:moved, resolve
Muster-Vorlage lf:mrk, resolve

Parameter p_rtrfile

Pfad des Dokuments mit der Übersetzung

Der Parameter wird in den folgenden Toplevel-Elementen benutzt:

Variable g_rtrroot

Globale Variable

Variable g_rtrroot

Wurzelelement der Übersetzung

Select: document($p_rtrfile)/r:translation

Verwendete globale Parameter oder Variable:

Parameter p_rtrfile

Die Variable wird in den folgenden Toplevel-Elementen benutzt:

Muster-Vorlage lf:xliff
Muster-Vorlage lf:segment/lf:source
Muster-Vorlage r:ref | r:moved, resolve
Muster-Vorlage lf:mrk, resolve

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Wurzel des XLIFF-Dokuments

Muster-Vorlage lf:xliff

Dem XLIFF-Wurzelelement wird die Zielsprache hinzugefügt.

Verwendete globale Parameter oder Variable:

Variable g_rtrroot

Muster-Vorlage r:machine

Übersetzungsmaschine

Muster-Vorlage lf:*

XLIFF-Elemente werden kopiert, wenn es keine spezielle Vorlage gibt.

Muster-Vorlage @*

Attribute werden kopiert.

Muster-Vorlage lf:segment/lf:source

Zu `source`-Elementen wird die Übersetzung hinzugefügt.

Verwendete globale Parameter oder Variable:

Parameter p_verbose
Variable g_rtrroot

Muster-Vorlage r:struct

Parameter

source

Das XLIFF-Element mit der Inhalt in der Quellsprache

segid

ID des segment-Elements

Eine übersetzte Struktur wird "aufgelöst".

Verwendete Modus:

resolve

Muster-Vorlage r:part, resolve

Texte innerhalb einer Struktur

Muster-Vorlage r:ref | r:moved, resolve

Parameter

source

Das XLIFF-Element mit dem Inhalt in der Quellsprache

segid

ID des segment-Elements

Verweis zu einem markierten Textausschnitt

Verwendete Modus:

resolve

Verwendete globale Parameter oder Variable:

Parameter p_verbose
Variable g_rtrroot

Muster-Vorlage lf:mrk, resolve

Parameter

moved

Ist das Element verschoben?

Übersetzung eines markierten Textausschnitts

Verwendete globale Parameter oder Variable:

Parameter p_verbose
Variable g_rtrroot

Muster-Vorlage r:error

Hinweis auf einen Fehler als Attribut

Modus

Modus resolve

Die folgenden Vorlagen implementieren den Modus resolve:

Muster-Vorlage r:part, resolve
Muster-Vorlage r:ref | r:moved, resolve
Muster-Vorlage lf:mrk, resolve

Der Modus resolve wird in den folgenden Stylesheet-Elementen benutzt:

Muster-Vorlage r:struct
Muster-Vorlage r:ref | r:moved, resolve

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  XLIFF und Übersetzung zusammenfügen
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
  2015-07-27 r:error: Attribut ausgeben.
  2015-08-15 ti:hint
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:lf  = "urn:oasis:names:tc:xliff:document:2.0"
  xmlns:r   = "http://herbaer.de/xmlns/20150127/resstruct#"
  xmlns:ti  = "http://herbaer.de/xmlns/201500703/transinfo/"
  xmlns     = "urn:oasis:names:tc:xliff:document:2.0"
  version   = "1.0"
>


<xsl:param name = "p_verbose" select = "'yes'"/>


<xsl:param name = "p_rtrfile"/>


<xsl:output method = "xml" indent = "yes" encoding = "utf-8"/>


<xsl:variable name = "g_rtrroot" select = "document($p_rtrfile)/r:translation"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "*"/>
</xsl:template>


<xsl:template match = "lf:xliff">
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:if test = "string-length($g_rtrroot/r:to) &gt; 0">
      <xsl:attribute name = "trgLang">
        <xsl:value-of select = "$g_rtrroot/r:to"/>
      </xsl:attribute>
    </xsl:if>
    <xsl:apply-templates select = "$g_rtrroot/r:machine"/>
    <xsl:apply-templates select = "* | text()"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "r:machine">
  <ti:machine>
    <xsl:value-of select = "."/>
  </ti:machine>
</xsl:template>


<xsl:template match = "lf:*">
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:apply-templates select = "* | text()"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "@*">
  <xsl:copy-of select = "."/>
</xsl:template>
```

```
<xsl:template match = "lf:segment/lf:source">
  <xsl:copy>
    <xsl:apply-templates select = "@*"/>
    <xsl:apply-templates select = "* | text()"/>
  </xsl:copy>
  <xsl:variable name = "id" select = "substring (../@id, 3)"/>
  <xsl:choose>
    <xsl:when test = "string-length ($id) = 0"/>
    <xsl:when test = "$g_rtrroot/r:struct [@id = $id]">
      <target>
        <xsl:apply-templates select = "$g_rtrroot/r:error [ . = $id ]"/>
        <xsl:variable name = "source" select = "."/>
        <xsl:apply-templates select = "$g_rtrroot/r:struct [@id = $id]">
          <xsl:with-param name = "source" select = "$source"/>
          <xsl:with-param name = "segid"  select = "$id"/>
        </xsl:apply-templates>
      </target>
    </xsl:when>
    <xsl:when test = "$g_rtrroot/r:text [@id = $id]">
      <target>
        <xsl:apply-templates select = "$g_rtrroot/r:error [ . = $id ]"/>
        <xsl:value-of select = "$g_rtrroot/r:text [@id = $id]"/>
      </target>
      <xsl:if test = "$p_verbose = 'yes' and lf:*">
        <xsl:message>
          <xsl:text>Kann Struktur </xsl:text>
          <xsl:value-of select = "$id"/>
          <xsl:text> nur als einfachen Text übersetzen.</xsl:text>
        </xsl:message>
      </xsl:if>
    </xsl:when>
    <xsl:otherwise>
      <xsl:if test = "$p_verbose = 'yes'">
        <xsl:message>
          <xsl:text>Keine Übersetzung für Segment </xsl:text>
          <xsl:value-of select = "$id"/>
        </xsl:message>
      </xsl:if>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "r:struct">


  <xsl:param name = "source"/>


  <xsl:param name = "segid"/>
  <xsl:apply-templates select = "*" mode = "resolve">
    <xsl:with-param name = "source" select = "$source"/>
    <xsl:with-param name = "segid"  select = "$segid"/>
  </xsl:apply-templates>
</xsl:template>


<xsl:template match = "r:part" mode = "resolve">
  <xsl:apply-templates/>
</xsl:template>


<xsl:template match = "r:ref | r:moved" mode = "resolve">


  <xsl:param name = "source"/>


  <xsl:param name = "segid"/>
  <xsl:variable name = "refid" select = "."/>
  <xsl:variable name = "mrkid" select = "concat ('m_', ., '_', $segid)"/>
  <xsl:choose>
    <xsl:when test = "$source//lf:mrk [@id = $mrkid]">
      <xsl:apply-templates select = "$source//lf:mrk [@id = $mrkid]" mode = "resolve">
        <xsl:with-param name = "moved" select = "local-name(.)"/>
      </xsl:apply-templates>
    </xsl:when>
    <xsl:otherwise>
      <xsl:if test = "$p_verbose = 'yes' and lf:*">
        <xsl:message>
          <xsl:text>Kein XLIFF-Element zu ref-id </xsl:text>
          <xsl:value-of select = "$refid"/>
        </xsl:message>
      </xsl:if>
      <xsl:value-of select = "$g_rtrroot//r:text [@id = $refid]"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "lf:mrk" mode = "resolve">
```

```
  <xsl:param name = "moved"/>
  <xsl:variable name = "mrkid" select = "substring-before (substring-after (@id, '_'), '_')"
  />
  <xsl:variable name = "segid" select = "substring-after  (substring-after (@id, '_'), '_')"
  />
  <xsl:choose>
    <xsl:when test = "@translate = 'no'">
      <xsl:copy>
        <xsl:apply-templates select = "@*"/>
        <xsl:if test = "$moved = 'moved'">
          <xsl:attribute name = "ti:hint">moved</xsl:attribute>
        </xsl:if>
        <xsl:copy-of select = "ancestor::lf:source/@xml:lang"/>
        <xsl:copy-of select = "text() | *"/>
      </xsl:copy>
    </xsl:when>
    <xsl:when test = "$g_rtrroot/r:struct [@id = $mrkid]">
      <xsl:copy>
        <xsl:apply-templates select = "@*"/>
        <xsl:if test = "$moved = 'moved'">
          <xsl:attribute name = "ti:hint">moved</xsl:attribute>
        </xsl:if>
        <xsl:variable name = "source" select = "."/>
        <xsl:apply-templates select = "$g_rtrroot/r:struct [@id = $mrkid]">
          <xsl:with-param name = "source" select = "$source"/>
          <xsl:with-param name = "segid"  select = "$segid"/>
        </xsl:apply-templates>
      </xsl:copy>
    </xsl:when>
    <xsl:when test = "$g_rtrroot/r:text [@id = $mrkid]">
      <xsl:copy>
        <xsl:if test = "$moved = 'moved'">
          <xsl:attribute name = "ti:hint">moved</xsl:attribute>
        </xsl:if>
        <xsl:apply-templates select = "@*"/>
        <xsl:value-of select = "$g_rtrroot/r:text [@id = $mrkid]"/>
      </xsl:copy>
      <xsl:if test = "$p_verbose = 'yes' and lf:*">
        <xsl:message>
          <xsl:text>Kann Teilstruktur </xsl:text>
          <xsl:value-of select = "$mrkid"/>
          <xsl:text> nur als einfachen Text übersetzen.</xsl:text>
        </xsl:message>
      </xsl:if>
    </xsl:when>
    <xsl:otherwise>
      <xsl:copy-of select = "."/>
      <xsl:if test = "$p_verbose = 'yes'">
        <xsl:message>
          <xsl:text>Keine Übersetzung für Textausschnitt </xsl:text>
          <xsl:value-of select = "$mrkid"/>
        </xsl:message>
      </xsl:if>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match = "r:error">
  <xsl:attribute name = "ti:error">
    <xsl:value-of select = "."/>
  </xsl:attribute>
</xsl:template>


</xsl:stylesheet>
```

Maschinelle Text-Übersetzung

Die maschinelle Text-Übersetzung erfogt durch die Perl-Programme `mttext.pl` und `translate.pl` (interaktiv für kurze Texte). Sie laden zur Übersetzung das Modul `Herbaer::Translate` (Datei `Translate.pm`), das weitere Module lädt, die einen Webdienst zur Übersetzung nutzen oder die Nutzung eines Webdienstes unterstützen.

translate.pl

[Quelltext]

Übersicht

`translate.pl --help|--version`

`translate.pl[ --verbose…|--no_verbose ]`
`[--in` *`IN`*`][--out` *`OUT`*`][--trname` *`TRNAME`*`][--srclang` *`SRCLANG`*`][--tgtlang` *`TGTLANG`*`]`

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Erhöht den Umfang der Meldungen nach STDERR.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--in` *`IN`*

Pfad der Eingabedatei oder `-` für `STDIN`. Die Eingabe (UTF-8-kodierter Text) wird zeilenweise gelesen. Jede Zeile wird als ein Befehl interpretiert.

`--out` *`OUT`*

Pfad der Ausgabedatei oder `-` für `STDOUT`. Die Ausgabe enthält die Übersetzungen.

`--trname` *`TRNAME`*

Name des Übersetzers, Er bestimmt das Perl-Modul, das zur Übersetzung benutzt wird. Einzelheiten s. `Herbaer::Translate` (`Translate.pm`). Ein Befehl kann einen anderen Übersetzer wählen.

`--srclang` *`SRCLANG`*

Die Kennung der Quellsprache, solange nicht eine andere Quellsprache bestimmt ist.

`--tgtlang` *`TGTLANG`*

Die Kennung der Zielsprache, in die die Texte übersetzt werden, solange keine andere Zielsprache bestimmt ist.

Beschreibung

Dieses Programm dient in erster Linie zum Test der Übersetzung.

Die Eingabe ist Textdateien in UTF-8-Kodierung. Jede Zeile wird als ein Befehl verarbeitet. Die möglichen Befehle sind:

`TRANSLATOR` *`trname`*

Im Folgenden wird der Übersetzer mit der Kennung *`trname`* benutzt (s. `Translate.pm`)

`TRANSLATOR_NAME`

Gibt den „Namen” des Übersetzers aus (`translator_name`).

`SRCLANG`

Gibt die Kennung der Quellsprache aus.

`SRCLANG` *`srclang`*

Die folgenden zu übersetzenden Texte sind in der Sprache mit der Kennung *`srclang`*.

`TGTLANG`

Gibt die Kennung der Zielsprache aus.

`TGTLANG` *`tgtlang`*

Die folgenden Texte sind in die Sprache mit der Kennung *`tgtlang`* zu übersetzen.

`TRANSLATE` *`text`*

Gibt die Übersetzung des Textes *`text`* aus.

`FINISH`

Ruft die Methode `finish` des Übersetzers auf.

`HELP` oder `?`

Zeigt die möglichen Befehle an.

`EXIT`

Beendet das Programm.

`text`

Wenn die Zeile nicht einem der oben aufgeführten Befehle entspricht, wird die Übersetzung von *`text`* ausgegeben.

Software-Voraussetzungen

Das Programm ist mit Perl Version 5.10.1 entwickelt. Es benutzt die folgenden Module:

`Herbaer::Readargs`

Die Funktionen `read_args` aus diesem Modul verarbeitet die Befehlszeilenargumente, die Funktion `print_message_with_values` gibt die Hilfe mit den aktuellen Einstellungen aus.

`Herbaer::Translate`

Die Methode `new` liefert den Übersetzer.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# Modul Herbaer::Translate (interaktiv) testen
# 2015-11-14 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package main;


use utf8 ;
use Herbaer::Readargs ;
use Herbaer::Translate ;


binmode (STDIN,  ":utf8") ;
binmode (STDOUT, ":utf8") ;
binmode (STDERR, ":utf8") ;


# Hash der Kommandozeilen-Argumente
my $args = {
   "in"           => "-",       # Eingabedatei (Text) oder - für STDIN
   "out"          => "-",       # Ausgabedatei oder - für STDOUT
   "trname"       => "default", # Übersetzer
   "srclang"      => "de",      # Default-Quellsprache
   "tgtlang"      => "zh",      # Zielsprache
};


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
translate.pl v20151114
Interaktiver Test der Übersetzung
2015 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
}
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   print_message_with_values (<<'HELP', $args);
translate.pl [Optionen]
--in IN           Pfad der Eingabedatei oder "-" für STDIN
                  ${in}
--out OUT         Pfad der Ausgabedatei oder "-" für STDOUT
                  ${out}
--trname TRNAME   Übersetzer ${trname}
--srclang         Quellsprache ${srclang}
--tgtlang         Zielsprache ${tgtlang}

Befehle:
TRANSLATOR <trname>
TRANSLATOR_NAME
SRCLANG <srclang>(?)
TGTLANG <tgtlang>(?)
TRANSLATE <text>
FINISH
HELP | ?
EXIT
<text>

HELP
   exit 0;
};


read_args ($args);


my $trans = new Herbaer::Translate ($args -> {"trname"}) ;
if (! $trans) {
   print STDERR "Kann Übersetzer " . $args -> {"trname"} . " nicht laden\n";
}
```

```
my $in   = $args -> {"in"};
my $hin;
if ($in eq "-") {
   $hin = STDIN;
}
elsif (! open ($hin, "<:encoding(utf-8)", $in)) {
   print STDERR "Kann \"$in\" nicht lesen: $!\n";
   exit 1;
}
my $out = $args -> {"out"};
my $hout;
if ($out eq "-") {
   $hout = STDOUT;
}
elsif (! open ($hout, ">:encoding(utf-8)", $out)) {
   print STDERR "Kann \"$out\" nicht öffnen: $!\n";
   exit 1;
}


my $t;
my $n;
my $line;
while ( defined ($line = <$hin>) )
{
   $line =~ s/^\s+// ;
   $line =~ s/\s+$// ;
   next unless $line;
   next if $line =~ /^#/ ;
   if ( $line =~ /^TRANSLATOR\s+(.+)$/ ) {
      $t = $1;
      if ( $args -> {"trname"} ne $t ) {
         $n = new Herbaer::Translate ($t) ;
         if ($n) {
            $args -> {"trname"} = $t;
            $trans = $n;
            print $hout "Neuer Übersetzer: ", $trans -> translator_name (), "\n";
         }
         else {
            print $hout "Kann Übersetzer \"$t\" nicht laden.\n";
         }
         $n = undef;
      }
   }
   elsif ( $line =~ /^TRANSLATOR_NAME$/ ) {
      if ($trans) {
         print $hout "translator_name: ", $trans -> translator_name (), "\n";
      }
      else {
         print $hout "ERROR: no translator\n";
      }
   }
   elsif ( $line =~ /^SRCLANG\s+(\S+)/) {
      $t = $1;
      if ($args -> {"srclang"} ne $t ) {
         $args -> {"srclang"} = $t;
         print $hout "SRCLANG $t\n";
      }
      else {
         print $hout "SRCLANG $t OK\n";
      }
   }
   elsif ( $line eq "SRCLANG" ) {
      print $hout "SRCLANG ", $args -> {"srclang"}, "\n";
   }
   elsif ( $line =~ /^TGTLANG\s+(\S+)/) {
      $t = $1;
      if ($args -> {"tgtlang"} ne $t ) {
         $args -> {"tgtlang"} = $t;
         print $hout "TGTLANG $t\n";
      }
      else {
         print $hout "TGTLANG $t OK\n";
      }
   }
   elsif ( $line eq "TGTLANG" ) {
      print $hout "TGTLANG ", $args -> {"tgtlang"}, "\n";
   }
   elsif ( $line =~ /^TRANSLATE\s+(.+)/) {
      $t = $1;
      if ($trans) {
         $n = $trans -> translate ($t, $args -> {"srclang"}, $args -> {"tgtlang"});
         print $hout "TRANSLATION ", $n, "\n";
         $n = undef;
      }
      else {
         print $hout "ERROR: no translator\n";
      }
   }
   elsif ( $line eq "FINISH" ) {
      if ($trans) {
         $trans -> finish ();
         print $hout "OK\n";
      }
   }
   elsif ( $line eq "HELP" or $line eq "?" ) {
```

```
      print <<'HLP';
Befehle:
TRANSLATOR <trname>
TRANSLATOR_NAME
SRCLANG <srclang>(?)
TGTLANG <tgtlang>(?)
TRANSLATE <text>
FINISH
HELP | ?
EXIT
<text>
HLP
   }
   elsif ( $line eq "EXIT" ) {
      last;
   }
   else {
      if ($trans) {
         $n = $trans -> translate ($line, $args -> {"srclang"}, $args -> {"tgtlang"});
         print $hout "TRANSLATION ", $n || "", "\n";
         $n = undef;
      }
      else {
         print $hout "ERROR: no translator\n";
      }
   }
}
```

Herbaer::Translate

[Quelltext]

Übersicht

```
use Herbaer::Translate;
my $translator = new Herbaer::Translate ("Übersetzername");
my $hallo_chinesisch = $translator -> translate ("Hallo Welt", "de", "zh");
$translator -> learn ("Hallo Welt", "de", "zh", "####");
```

Einführung

Das Modul `Herbaer::Translate` definiert die Funktion `new`. Die Funktion `new` bestimmt das implementierende Modul, ruft die Funktion `new` des implementierenden Moduls auf und gibt deren Ergebnis zurück.

Nach dem Paketnamen werden der Methode `new` des implementierenden Moduls zunächst die Parameter aus der Datei `Translate.names`, dann die weiteren Parameter nach dem Namen aus dem Aufruf der Funktion `new` aus diesem Modul übergeben.

Zur Ersetzung von Platzhaltern dient das Modul `Herbaer::Replace` (Datei `Replace.pm`).

Datei `Translate.names`

Die Datei `Translate.names` liegt in demselben Verzeichnis wie die Datei `Translate.pm`. Sie ordnet einer Zeichenkette (erstes Argument der Funktion `Herbaer::Translate::new`) ein Modul zu und bestimmt die Argumente des Aufrufs der Methode `new` des Moduls.

Die Datei `Translate.names` wird zeilenweise verarbeitet. Leere Zeilen und Zeilen, die mit dem Zeichen # beginnen, werden ignoriert.

Zeilen, die mit der Zeichenfolge `MODULE` in der ersten Spalte beginnen, bestimmen ein Modul. Nach der Zeichenfolge `MODULE` muss wenigstens ein Leerzeichen folgen. Der Zeilenrest wird duch Folgen von einem oder mehreren Leerzeichen in „Wörter” zerlegt. Das erste Wort ist der vollständige Paketname des Moduls. Die weiteren Wörter können Platzhalter der Form `${`*`INT`*`}` oder `${`*`NAME`*`}` enthalten. Sie stehen für Positions-Matchgruppen oder benannte Matchgruppen des gefundenen Namens-Musters (s. unten). Die Wörter werden nach der Ersetzung der Platzhalter der Methode `new` des angegebenen Moduls als erste Argumente übergeben.

Alle anderen Zeilen werden „getrimmt”: Folgen von Leerraumzeichen am Anfang oder am Ende der Zeile werden entfernt, andere Folgen von mehreren Leerraumzeichen in der Zeile werden durch ein einzelnes Leerzeichen ersetzt. Das Ergebnis ist ein regulärer Ausdruck. Dem regulären Ausdruck wird die letzte vorhergehende `MODULE`-Zeile zugeordnet. Wenn die Zeichenkette, die der Methode `Herbaer::Translate::new` als erster Parameter übergeben wird, (die Bezeichnung des Übersetzers) ganz oder zu einem Teil zu dem regulären Ausdruck passt, wird das in der `MODULE`-Zeile angegebene Modul (das implementierende Modul) geladen. Die Matchgruppen des regulären Ausdrucks bestimmen die Werte der Platzhalter in den Wörtern der `MODULE`-Zeile.

Die regulären Ausdrücke werden in der Reihenfolge, in der sie in der Datei `Translate.names` auftreten, mit der Zeichenkette verglichen.

Die Datei `Translate.names` sollte der Bezeichnung `default` einen Übersetzer zuordnen.

Funktionen

`Herbaer::Translate::new` (*`$trname`*)

> *`$trname`* bezeichnet ein Übersetzungs-Modul (implementierendes Modul). Wenn der Wert zu einem regulären Ausdruck in der Datei `Translate.names` passt, wird das dort bezeichnete Modul geladen und dessen Methode

new mit den in Translate.names angegebenen Parametern aufgerufen. Wenn dieser Methode Herbaer::Translate::new nach *$trname* weitere Argumente übergeben werden, werden diese als weitere Argumente an die Methode new des implementierenden Moduls übergeben.

Wenn kein regulärer Ausdruck aus Translate.names zu *$trname* passt, wird der Wert von *$trname* durch Leerzeichen in Worte zerlegt. Das erste Wort ist der Paketname des implementierenden Moduls. Wenn es nicht die Zeichenfolge :: enthält, wird Herbaer::Translate:: vorangestellt. Die weiteren Wörter werden als Argumente an die Methode new des implementierenden Moduls übergeben, danach folgen die weiteren Argumente des Aufrufs von Herbaer::Translate::new.

Implementierende Module

Das Modul Herbaer::Translate::Base (Base.pm) zeigt die Funktionen, die die Schnittstelle eines Übersetzungsmoduls bilden. Andere Module benutzen Herbaer::Translate::Base als Basis.

Implementierende Module (in alfabetischer Reihenfolge) sind:

- Herbaer::Translate::Ask (Ask.pm)
- Herbaer::Translate::GoogleTranslate (GoogleTranslate.pm)
- Herbaer::Translate::Learnchain (Learnchain.pm)
- Herbaer::Translate::MySQLLookup (MySQLLookup.pm)
- Herbaer::Translate::NameReplacer (NameReplacer.pm)
- Herbaer::Translate::Normalizer (Normalizer.pm)
- Herbaer::Translate::Pipe (Pipe.pm)
- Herbaer::Translate::SystemItemFilter (SystemItemFilter.pm)
- Herbaer::Translate::WordReplacer (WordReplacer.pm)

Diese Module benutze ich tatsächlich. Ich habe eine Reihe weiterer Module ausprobiert. Neben Ask.pm habe ich zum Test weiter Module erstellt, die eine Übersetzung simulieren.

Andere Module nutzen andere Web-Dienste zur Übersetzung. Ein Modul wählt einen Übersetzer abhängig von den Sprachen. Hier liegt der Grund für die Methode translator_name. Ich muss auf der Website natürlich auf den (oder wenigstens einen) tatsächlich benutzten Übersetzungsdienst hinweisen.

Ergänzend oder alternativ zu MySQLLookup.pm gibt es Module, die Übersetzungen in Textdateien oder in „Hash”-Einträgen im Arbeitsspeicher zwischenspeichern. Hier ist der Grund für die Methode finish: Übersetzungen können so dauerhaft gespeichert werden. „DESTROY”-Methoden konnten nicht sicherstellen, dass beim finalen Aufräumen die Daten geschrieben wurden, solange die Dateihandles noch funktionierten. Beim aktuellen Stand der Übersetzung sind die beiden Methoden translator_name und finish nicht nötig.

Quelltext

[Beschreibung]

```
# symlink INC/Herbaer/Translate.pm
# Übersetzer wählen
# 2015-08-03 Herbert Schiemann <h.schiemann@herbaer.de>
# 2020-04-03 Platzhalter ${basedir}
# GPL Version 2 oder neuer


package Herbaer::Translate ;


BEGIN {
   use utf8 ;
   use Cwd qw(realpath);

   binmode (STDOUT, ":utf8") ;
   binmode (STDERR, ":utf8") ;

   my $n = realpath($0);
   $n =~ s/\/web\/src\/.*$//;

   our $basedir = $n;
   our $namelist = [];

   my $fn = $INC{"Herbaer/Translate.pm"};
   $fn =~ s/\.pm$/.names/ ;
   my $fh;
   my $line;
   my $lnr = 0;
   my $module;
   my $modargs = [];
   if ( open ($fh, "<:encoding(utf-8)", $fn) ) {
      while (defined ($line = <$fh>)) {
         ++$lnr;
         next if $line =~ /^#/ ;
         $line =~ s/\s*$//;
         next unless $line;
         if ( $line =~ s/^MODULE// ) {
            if ( $line =~ s/^\s+(\S+)// ) {
               $module = $1;
            }
            else {
               die "Fehler in Datei \"$fn\":\n$line\n";
            }
            $modargs = [];
            while ( $line =~ s/^\s+(\S+)// ) {
               push (@$modargs, $1);
            }
         }
         else {
            $line =~ s/^\s+//;
            $line =~ s/\s+/ /;
            if ( $module ) {
               push (@$namelist, [ qr/$line/, $module, $modargs ]);
            }
            else {
               die "Fehler in Datei \"n$fn\": Modul nicht definiert:\n$line\n";
            }
         }
      }
      close $fh;
   }
} # BEGIN


use utf8 ;
use Herbaer::Replace ;
```

```
sub new {
   my $class = shift;
   my $trans = shift;
   print STDERR "Herbaer::Translate::new (\"$trans\")\n";
   die "Übersetzer nicht angegeben" unless $trans;
   my $impl = "";
   my $p;
   my $ma;         # Argumente aus Translate.names
   my $i;          # Nummer einer Match-Gruppe
   my $b ;         # Beginn einer Match-Gruppe
   my ($n, $v) ;   # Name und Wert einer benannten Match-Gruppe
   my $repl = { "basedir" => $basedir, };  # Ersetzungs-Hash
   for $p (@$namelist) {
      if ($trans =~ $p -> [0]) {
         $impl = $p -> [1];
         $i = 0;
         for $b (@-) {
            if ($i) {
               $repl -> {"$i"} = substr ($trans, $b, $+[$i] - $b) if defined ($b);
            }
            ++$i;
         }
         while ( ($n, $v) = each (%+) ) {
            $repl -> {$n} = $v if defined ($v);
         }
         $ma = replace ([@{$p -> [2]}], $repl);
         last;
      }
   }
   if (!$impl) {
      if ($trans =~ s/\s*(\S+)// ) {
         $impl = $1;
         $ma   = [];
         while ( $trans =~ s/\s*(\S+)// ) {
            push (@$ma, $1);
         }
         replace ($ma, $repl);
      }
   };
   $impl = "Herbaer::Translate::$impl" unless $impl =~ /::/ ;
   no strict "refs";
   # s. perlmod
   unless (exists ${"$impl\::"}{"new"}) {
      my $pmfile = "$impl.pm";
      $pmfile =~ s/::/\//g;
      eval { require $pmfile } ;
      die $@ if $@;
   }
   return $impl -> new (@$ma, @_);
} # new


1;
```

Datei Translate.names

```
# file KLEIDER/web/src/localization/Translate.names
# symlink INC/Herbaer/Translate.names
# 2020-03-04 Platzhalter ${basedir}


# Diese Datei ordnet Übersetzungmodulen Namen zu
# s. Herbaer::Translate, Translate.pm


# Zeilen, die mit dem Zeichen # beginnen oder nur Leerzeichen enthalten,
# sind Kommentare.


# Dem Schlüsselwort MODULE am Zeilenanfang (Modul-Zeile)
# folgt die Bezeichnung eines Moduls,
# gefolgt von weiteren Wörtern, die der Methode new als Parater übergeben werden.
# Diese weiteren Wörter können Platzhalter enthalten:
# ${basedir} für den Dateipfad, unter dem der Datenträger eingebunden ist,
# ${N} für die N-te Matchgruppe
# ${NAME} für die benannte Matchgruppe mit den Namen NAME: (?<NAME>pattern)
# Die Zeile wird an Leerzeichen zerlegt.


# Leerzeichen am Anfang und am Ende anderer Zeilen werden entfernt.
# Der verbleibende Rest ist ein regulärer Ausdruck.
# Wenn ein Name einem regulären Ausdruck entspricht, wird das Modul geladen.
# Die Aufruf-Argumente der Methode new sind die Wörter,
# die nach dem Namen des Moduls angegeben sind,
# nach der Ersetzung der Platzhalter (s.o.),
# gefolgt von weiteren Werten,
# die an Herbaer::Translate::new übergeben sind


MODULE Herbaer::Translate::Normalizer system_bnr_google
  ^default$


MODULE Herbaer::Translate::GoogleTranslate ${basedir}/web/secrets
  ^Google Translate$
  ^google$


# Herbaer::Translate::MySQLLookup pfad/zur/geheimdatei datenbankkennung lerner
MODULE Herbaer::Translate::MySQLLookup  ${basedir}/web/secrets translate
  ^lookup$


# Herbaer::Translate::MySQLLookup pfad/zur/geheimdatei datenbankkennung lerner
MODULE Herbaer::Translate::MySQLLookup  ${basedir}/web/secrets learnbuffer lookup
  ^lookup_buffered$


MODULE Herbaer::Translate::Pipe ${basedir}/web/pipe norm_${1}
  ^pipe_n(\S+)$


MODULE Herbaer::Translate::Pipe ${basedir}/web/pipe ${1}
  ^pipe_(\S+)$


# Nachschlage-Modus
MODULE Herbaer::Translate::Learnchain lookup names_${1} 0
  ^learnchn_n(\S+)$


# Gepuffertes Lernen
MODULE Herbaer::Translate::Learnchain lookup_buffered names_${1} 0
  ^learnbuf_n(\S+)$


# Lern-Modus
MODULE Herbaer::Translate::Learnchain lookup names_${1} 1
  ^learn_n(\S+)$


MODULE Herbaer::Translate::Normalizer system_${1}
  ^norm_s(\S+)$


MODULE Herbaer::Translate::NameReplacer ${basedir}/web/transdb/names replace_${1}
  ^names_r_(\S+)$


MODULE Herbaer::Translate::WordReplacer ${basedir}/web/transdb/replace ${1}
  ^replace_(\S+)$


MODULE Herbaer::Translate::SystemItemFilter learnchn_${1}
  ^system_c(\S+)$
```

```
MODULE Herbaer::Translate::SystemItemFilter learnbuf_${1}
  ^system_b(\S+)$


MODULE Herbaer::Translate::SystemItemFilter learn_${1}
  ^system_l(\S+)$


# zum Test MySQLLookup:
# ********************
MODULE Herbaer::Translate::Learnchain mysql Ask
  ^test_mysql$


# Test MySQLLookup: überschreibendes Lernen
MODULE Herbaer::Translate::Learnchain mysql Ask 1
  ^test_mysql_learn$


MODULE Herbaer::Translate::MySQLLookup ${basedir}/web/secrets translate 1
  ^mysql$
```

Herbaer::Translate::Base

[Quelltext]

Zweck

Das Modul `Herbaer::Translate::Base` (Datei `Base.pm`) definiert „Leerfunktionen” für die Funktionen, die von einer Implementation eines Übersetzungsmoduls erwartet werden. Es kann als Basis-Klasse verwendet werden:

```
# File Herbaer/Translate/Uebersetzungsmodul.pm
package Herbaer::Translate::Uebersetzungsmodul ;
use parent Herbaer::Translate::Base ;
```

Die Implementation (hier die Datei `Herbaer/Translate/Uebersetzungsmodul.pm`) muss die Funktionen implementieren.

Funktionen

`new`

> Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise nicht direkt aufgerufen, sondern von `Herbaer::Translate::new` (s. `Translate.pm`).

`$translator->translate ($text, $quellsprache, $zielsprache)`

> Ergibt die Übersetzung des Textes *`$text`* aus der Sprache *`$quellsprache`* in die Sprache *`$zielsprache`*. ([BCP47] `http://www.ietf.org/rfc/bcp/bcp47.txt`, [IETF RFC 3066] `http://www.ietf.org/rfc/rfc3066.txt`)

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

> Diese Funktion ermöglicht es dem Übersetzer, zu lernen. *`$uebersetzung`* ist die Übersetzung des Textes *`$text`* aus der Sprache *`$quellsprache`* in die Sprache *`$zielsprache`*.

$translator->`finish` ()

> Diese Methode zeigt dem Übersetzer an, dass er sich auf das Ende seiner Arbeit einstellen kann. Ein Übersetzer, der neue Übersetzungen gelernt hat, kann und soll die neu gelernten Übersetzungen dauerhaft speichern. Der Übersetzer funktioniert weiterhin.
>
> Code, der zur normalen Laufzeit ohne Probleme funktioniert, verursacht im globalen "Clean up" beim Aufruf durch den Destruktor Speicherzugriffsfehler. Vielleicht sind die Fehler in einer aktuelleren Perl-Version behoben?

$translator->`translator_name` ()

> Diese Methode liefert eine Kennung der tatsächlichen Übersetzungsmodule. Ein Module kann zum Beispiel abhängig vom zu übersetzenden Text und den Sprachen zwischen verschiedenen Übersetzungsdiensten wählen. Aus dem Ergebnis des Funktionsaufrufs sollte hervorgehen, welcher Übersetzer tatsächlich übersetzt hat.

Quelltext

[Beschreibung]

```
# Basisklasse für Übersetzer
# 2015-11-02 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::Base ;


use utf8;


BEGIN {
   binmode (STDERR, ":utf8");
}


sub new {} # new


# ergibt die Übersetzung von $text aus der Sprache $srl
# in die Sprache $tgl
sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
} # translate


# lernt:
# $tt ist die Übersetzung des Textes $text
# aus der Sprache $tgl in die Sprachen $tt
sub learn {
   my ($self, $text, $srl, $tgl, $tt) = @_ ;
} # learn


# räumt auf,
# der Übersetzer funktioniert aber weiterhin
sub finish {} # finish


# Name des zuletzt aktiven Uebersetzers
# für Module, die abhängig von den Parametern
# die Übersetzung an verschiedene Module weiterleiten.
sub translator_name { ""; } # translator_name


1;
```

Herbaer::Translate::Ask

[Quelltext]

Zweck

Dieses Übersetzungsmodul implementiert die Methoden `translate` und `translator_name`. Die Methode `translate` liest die Übersetzung aus der Standard-Eingabe. Das Modul dient zum Test, s. `MySQLLookup.pm`, `Learnchain.pm`, `Translate.pm`, `Translate.names`, `translate.pl`.

Quelltext

[Beschreibung]

```
# Übersetzung von STDIN lesen
# 2016-07-20 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::Ask ;


use parent Herbaer::Translate::Base ;
use utf8;


sub new {
    my $class = shift;
    my $self = "Ask";
    return bless (\$self, $class);
} # new


sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $trans;
   print "$srl: $text\n";
   print "Übersetzung in $tgl: ";
   $trans = <STDIN>;
   $trans =~ s/^\s*//;
   $trans =~ s/\s*$//;
   print "\n";
   $trans;
} # translate


# Name des zuletzt aktiven Uebersetzers
sub translator_name { "ask"; } # translator_name


1;
# file KLEIDER/web/src/localization/Ask.pm
```

Herbaer::Translate::GoogleTranslate

[Quelltext]

Zweck

Das Modul `Herbaer::Translate::GoogleTranslate` (Datei `GoogleTranslate.pm`) nutzt den Web-Dienst Google Translate zur Übersetzung.

Verzögerung

Um das Netz und den Übersetzungsdienst nicht zu überlasten, kann eine Wartezeit zwischen zwei Requests zur Übersetzung eingestellt werden. Damit die Wartezeit nicht die gesamte lokale Verarbeitung blockiert, wird ein neuer Prozess (Kindprozess) gestartet, der über das Web kommuniziert.

Die Kommunikation des Hauptprozesses mit dem Kindprozess erfolgt über Pipes. Der Hauptprozess sendet zeilenweise „Übersetzungsbefehle” an den Kindprozess. Ein Übersetzungsbefehl hat den Aufbau `tr:`*`SRCTEXT`*`:`*`SRCLANG`*`:`*`TGTLANG`*. *`SRCTEXT`* steht für den zu übersetzdenden Text in der Quellsprache, *`SRCLANG`* für die Kennung der Quellsprache, *`TGTLANG`* für die Kennung der Zielsprache. Der Kindprozess sendet als Antwort eine Zeile mit dem übersetzten Text. Das Zeilenende-Zeichen und der Doppelpunkt im Text werden in der Kommunikation zwischen den Prozessen „geschützt”, also durch andere Zeichenfolgen ersetzt. Dazu dienen die beiden Hilfsfunktionen `_encode` und `_decode`.

Im Hauptprozess schließt der Destructor (`DESTROY`) die Pipes zum Kindprozess und wartet auf dessen Beendigung.

Übersetzung

Ein kurzer zu übersetzender Text wird in der URL eines GET-Requests kodiert, ein längerer Text wird als Rumpf eines POST-Requests.

Die Antwort ist JSON-kodiert.

Der HTTP-Request erfolgt in der Funktion `_translate`, die im Falle einer Verzögerung im Kindprozess, sonst im Hauptprozess aufgerufen wird.

Schlüssel und Einstellungen

Um Google Translate zu nutzen, ist ein Schlüssel (apikey) erforderlich. Er wird aus einer Textdatei (Beispiel `secrets`) gelesen. Die Textdatei enthält eine Zeile

```
google.translate.apikey=mein_geheimer_google_api_schluessel
```

`mein_geheimer_google_api_schluessel` ist ein Platzhalter für den wirklichen Schlüssel. In der Textdatei kann auch die Wartezeit zwischen zwei Übersetzungs-Requests in Mikrosekunden angegeben werden. Die folgende Zeile erzwingt eine halbe Sekunde Wartezeit zwischen zwei Requests:

```
google.translate.wait=500000
```

Funktionen

`$translator = Herbaer::Translate::GoogleTranslate::new ($secrets, $debug)`

> Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise nicht direkt aufgerufen, sondern von `Herbaer::Translate::new` (s. `Translate.pm`).

`$secrets` ist der Dateipfad der Datei, die den Google-Anwendungsschlüssel enthält.

Wenn der Wert von *`$debug`* logisch wahr ist, werden Meldungen nach STDERR ausgegeben.

`$translator->translate ($text, $quellsprache, $zielsprache)`

Ergibt die Übersetzung des Textes *`$text`* aus der Sprache *`$quellsprache`* in die Sprache *`$zielsprache`*. ([BCP47] `http://www.ietf.org/rfc/bcp/bcp47.txt`, [IETF RFC 3066] `http://www.ietf.org/rfc/rfc3066.txt`)

In den Kennungen der Sprachen werden die Zeichen vom ersten „-”-Zeichen an entfernt. An die Kennung `zh` wird `-CN` angehängt.

Im übersetzten Text sind in asiatischen Sprachen die „Null-Leerzeichen” ein Problem. Sie zeigen dem Browser Stellen an, an denen ein Zeilenwechsel möglich ist. Aber ein „Null-Leerzeichen” unmittelbar bei einem anderen Leerzeichen ist nicht nötig und wird entfernt. An einigen Stellen zwischen lateinischen Buchstaben und asiatischen Satzzeichen werden „Null-Leerzeichen” eingefügt.

`$translator->languages ($quellsprache)`

Ergibt eine Liste (ARRAY-Ref) der Kennungen der Sprachen, in die Text aus der Sprache mit der Kennung *`$quellsprache`* übersetzt werden kann. Der dazu nötige HTTP-Request erfolgt im Hauptprozess unabhängig von der Wartezeit zwischen zwei Übersetzungs-Requests.

$translator->`translator_name` ()

Ergibt die Zeichenkette `google`.

Quelltext

[Beschreibung]

```
# Übersetzung durch GoogleTranslate
# 2015-08-09 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::GoogleTranslate ;


use parent Herbaer::Translate::Base ;
use utf8 ;
use Herbaer::Replace ;
use IO::Handle ;
use JSON ;
use LWP ;
use Time::HiRes qw ( usleep );
use URI::Escape ;


# "\" wird durch "\\",
# ":" wird durch "\c",
# Zeilenende durch "\n" ersetzt
sub _encode {
   my $t = shift;
   $t =~ s/\\/\\\\/g;
   $t =~ s/\n/\\n/g;
   $t =~ s/:/\\c/g;
   return $t;
} # _encode


# macht _encode rückgängig
sub _decode {
   my $t = shift;
   $t =~ s/\\(.)/$1 eq "n" ? "\n" : $1 eq "c" ? ":" : $1/ge;
   return $t;
} # _decode


sub new {
   my ($class, $secrets, $debug) = @_;
   print STDERR "new Herbaer::Translate::GoogleTranslate\n" if $debug;
   my $hnd;
   if (! open ($hnd, "<:encoding(utf-8)", $secrets)) {
      print STDERR "Kann Datei $secrets nicht lesen:$!\n" if $debug;
      return undef;
   }
   my $line;
   my $key;
   my $url;
   my $post_url;
   my $wait = 0;
   while (defined ($line = <$hnd>)) {
      $line =~ s/\s*$//;
      if ($line =~ /^\s*google\.translate\.apikey\s*=\s*(.*)/) {
         $key = $1;
         $url =
            "https://www.googleapis.com/language/translate/v2?key=${key}"
            . "\&format=text\&q=\$\{text\}\&source=\$\{srl\}\&target=\$\{tgl\}"
            ;
         $post_url =
            "https://www.googleapis.com/language/translate/v2?key=${key}"
            . "\&format=text\&source=\$\{srl\}\&target=\$\{tgl\}"
            ;
      }
      if ($line =~ /^\s*google\.translate\.wait\s*=\s*([1-9][0-9]+)/) {
         $wait = $1 + 0;
      }
   }
   close $hnd;
   my $self = {
      "dbg"    => $debug,
      "ua"     => LWP::UserAgent -> new (),
      "u"      => $url,
      "p_url"  => $post_url,
      "p_cont" => "q=\${text}",
      "wait"   => $wait,
      "pid"    => undef,
   };
   my $ua = $self -> {"ua"};
   $ua -> agent ($ua -> _agent . " (+http://kleider.herbaer.de)");
   $self = bless ($self, $class);
   # Kindprozess starten
   if ($wait) {
      my ($cw, $cr, $pw, $pr);
      pipe ($cr, $pw);
      pipe ($pr, $cw);
      my $pid = fork ();
      if (!$pid) {
         close ($pr);
```

```
         close ($pw);
         binmode ($cr, ":encoding(utf-8)");
         binmode ($cw, ":encoding(utf-8)");
         $cw -> autoflush ();
         my $line;
         my $st;   # source text
         my $sl;   # source language
         my $tl;   # target language
         my $tt;   # translated text
         while (defined ($line = <$cr>)) {
            if ($line =~ /^tr:([^:]+):([^:]+):([^:]+)$/) {
               $st = $1;
               $sl = $2;
               $tl = $3;
               $tt = $self -> _translate (_decode ($st), $sl, $tl) || "";
               print $cw _encode ($tt) . "\n";
               usleep ($self -> {"wait"});
            }
         }
         close ($cr);
         close ($cw);
         exit 0;
      }
      else {
         close ($cr);
         close ($cw);
         binmode ($pr, ":encoding(utf-8)");
         binmode ($pw, ":encoding(utf-8)");
         $pw -> autoflush ();
         $self -> {"pid"} = $pid;
         $self -> {"r"}   = $pr;
         $self -> {"w"}   = $pw;
      }
   }
   return bless ($self, $class);
} # new


sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $dbg = $self -> {"dbg"};
   if ($dbg) {
      print STDERR "Herbaer::Translate::GoogleTranslate::translate\n";
   }
   $srl =~ s/-.+//;
   $srl = lc ($srl);
   $tgl =~ s/-.+//;
   $tgl = lc ($tgl);
   $tgl = "zh-CN" if $tgl eq "zh";
   my $tt;
   if ($self -> {"pid"}) {
      my $read  = $self -> {"r"};
      my $write = $self -> {"w"};
      print $write join (":", "tr", _encode ($text), $srl, $tgl), "\n";
      $tt = <$read> || "";
      $tt =~ s/\n//;
      $tt =  _decode ($tt);
   }
   else {
      $tt = $self -> _translate ($text, $srl, $tgl);
   }
   $tt =~ s/(\s)\x{200b}+/$1/g;            # Null-Zwischenraum nach Leerraum
   $tt =~ s/\x{200b}+(\s)/$1/g;            # Null-Zwischenraum vor Leerraum
   $tt =~ s/\x{200b}\x{200b}+/\x{200b}/g;  # Folgen von Null-Zwischenraum
   # Null-Zwischenraum vor öffnender Klammer (chinesisch, japanisch)
   $tt =~ s/([a-zäöüßA-ZÄÖÜ])([#][a-zäöüßA-ZÄÖÜ])/${1}\x{200b}${2}/g;
   # Null-Zwischenraum nach schließender Klammer, Komma oder Punkt (zh, ja)
   $tt =~ s/([a-zäöüßA-ZÄÖÜ][####])([a-zäöüßA-ZÄÖÜ])/${1}\x{200b}${2}/g;
   $tt;
} # translate
```

```
sub _translate {
   require bytes ;
   my ($self, $text, $srl, $tgl) = @_ ;
   my $input =
   {
      "text" => uri_escape_utf8 ($text),
      "srl"  => $srl,
      "tgl"  => $tgl,
   };
   my $dbg = $self -> {"dbg"};
   if ($dbg) {
      print STDERR "Herbaer::Translate::GoogleTranslate::_translate\n";
   }
   my $ua = $self -> {"ua"};
   my $resp;
   my $data;
   my $url = replace ($self -> {"u"}, $input);
   if (bytes::length ($url) < 2000) {
      print STDERR "URL $url\n" if $dbg;
      $resp = $ua -> get ($url);
   }
   else {
      $url  = replace ( $self -> {"p_url"},  $input);
      $cont = replace ( $self -> {"p_cont"}, $input);
      my $req = HTTP::Request -> new ( "POST" => $url );
      $req -> header       ( "X-HTTP-Method-Override" => "GET" );
      $req -> content_type ("application/x-www-form-urlencoded");
      $req -> content      ($cont);
      $resp = $ua -> request ($req);
   }
   print STDERR $resp -> as_string(), "\n" if $dbg;
   if ( ! $resp -> is_success () ) {
      print STDERR "Kann URL $url nicht laden\n" if $dbg;
   }
   else {
      $data = decode_json ($resp -> decoded_content ());
   }
   $tt = $data -> {"data"} -> {"translations"} [0] -> {"translatedText"} || "";
} # _translate


sub languages {
   my ($self, $srl) = @_ ;
   my $dbg = $self -> {"dbg"};
   if ($dbg) {
      print STDERR "Herbaer::Translate::GoogleTranslate::languages\n";
   }
   my $url = $self -> {"u"};
   $url =~ s/\/v2\?/\/v2\/languages\?/;
   $url =~ s/&.*/&source=/;
   $srl =~ s/-.*//;
   $url .= $srl;
   my $ua = $self -> {"ua"};
   print STDERR "URL $url\n" if $dbg;
   my $resp = $resp = $ua -> get ($url);
   my $data;
   if ( ! $resp -> is_success () ) {
      print STDERR "Kann URL $url nicht laden\n" if $dbg;
   }
   else {
      $data = decode_json ($resp -> decoded_content ());
   }
   my $l;
   my $ret = [];
   for $l (@{$data -> {"data"} -> {"languages"}}) {
      push @$ret, $l -> {"language"};
      print STDERR $l -> {"language"}, "\n" if $dbg;
   }
   $ret;
} # languages


# Name des Uebersetzers
sub translator_name { "google"; } # translator_name


sub DESTROY {
   my $self = shift;
   my $dbg = $self -> {"dbg"};
   if ($dbg) {
      print STDERR "Herbaer::Translate::GoogleTranslate::DESTROY\n";
   }
   if ($self -> {"pid"}) {
      close ($self -> {"w"});
      close ($self -> {"r"});
      wait;
   }
   1;
} # DESTROY


1;
```

Datei secrets

```
# file KLEIDER/web/src/localization/secrets
# Beispiel für eine Geheimnis-Datei


# FTP-Zugangsdaten
ftp.host=kleider.herbaer.de
ftp.user=meine_heimliche_kennung
ftp.password=mein_ganz_geheimes_kennwort


google.translate.apikey=mein_geheimer_google_api_schluessel
# Wartezeit in Mikrosekunden zwischen zwei Requests zur Übersetzung
google.translate.wait=500000


# translate.mysql.DATENBANKKENNUNG=DATENBANKNAME
translate.mysql.translate=translate
translate.mysql.learnbuffer=translrnbuf


# mysql.DATENBANKNAME.*
mysql.translate.user=mein_heimliches_login
mysql.translate.password=noch_ein_ganz_geheimes_kennwort
mysql.translrnbuf.user=mein_anderes_login
mysql.translrnbuf.password=noch_ein_ganz_anderes_kennwort
mysql.likedb.user=auswerter
mysql.likedb.password=denk_es_dir_aus
```

Herbaer::Learnchain

[Quelltext]

Zweck

Das Modul `Herbaer::Translate::Learnchain` (Datei `Learnchain.pm`) (Paket `Herbaer::Translate::Learnchain`) implementiert die Übersetzer-Schnittstelle (s. `Tranlate.pm`, `Base.pm`)

Es verknüpft zwei Übersetzer: einen „Lehrling” und einen „Meister”. Ein typischer „Lehrling” sucht Übersetzungen in einer Datenbank, der „Meister” befragt einen Web-Dienst. Details sind zur Methode `translate` beschrieben.

Funktionen

`$translator = Herbaer::Translate::Learnchain->new (`*`$learner`*`, `*`$master`*`, `*`$update`*`, `*`$debug`*`)`

Ergibt einen neuen Übersetzer als Verknüpfung zweier Übersetzer: eines „Lehrlings” und eines „Meisters”.

`$learner` ist die Kennung des „Lehrlings”

`$master` ist die Kennung des „Meisters”

Der optionale Parameter *`$update`* steuert die Arbeitsweise der Methode `translate []`. Ein logisch wahrer Wert stellt den „Aktualisierungsmodus” ein.

Ein logisch wahrer Wert des optionalen Parameters *`$debug`* führt zu Meldungen nach STDERR.

`$translator->translate ($text, $quellsprache, $zielsprache)`

Zunächst soll der „Lehrling” den Text *`$text`* aus der Sprache *`$quellsprache`* in die Sprache *`$zielsprache`* übersetzen. Wenn er keine Übersetzung liefert oder wenn der „Aktualisierungmodus” eingeschaltet ist, übersetzt der „Meister”. Wenn der Meister eine andere Übersetzung findet als der „Lehrling”, lernt der „Lehrling” die Übersetzung der Meisters. Das Ergebnis ist die Übersetzung des Meisters, falls der Meister gefragt wurde, sonst die Übersetzung der Lehrlings.

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

Ruft die gleichnamigen Methoden des „Lehrlings” und des „Meisters” auf.

`$translator->finish ()`

Ruft die gleichnamigen Methoden des „Lehrlings” und des „Meisters” auf.

`$translator->translator_name ()`

Ergibt im Aktualisierungmodus die Zeichenkette `lu_`*`MASTER_LEARNER`*, sonst die Zeichenkette `lc_`*`MASTER_LEARNER`*. Der Platzhalter *`MASTER`* steht für das Ergebnis der Methode `translator_name` des „Meisters”, *`LEARNER`* für das Ergebnis der Methode `translator_name` des „Lehrlings”.

Quelltext

[Beschreibung]

```
# Zwei Übersetzer verbinden
# 2015-08-03 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::Learnchain ;


use parent Herbaer::Translate::Base ;
use Herbaer::Translate ;
use utf8;


# update: "Lerner" wird aktualisiert
sub new {
   my ($class, $lrn, $mtr, $update, $dbg) = @_;
   if ($dbg) {
      print STDERR "new Herbaer::Translate::Learnchain ($lrn, $mtr)\n";
   }
   my $self = {
      "l"   => new Herbaer::Translate ($lrn, $dbg),
      "m"   => new Herbaer::Translate ($mtr, $dbg),
      "upd" => $update || 0,
      "dbg" => $dbg,
   };
   return bless ($self, $class);
} # new


sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $d   = $self -> {"dbg"};
   my $upd = $self -> {"upd"};
   if ($d) {
      print STDERR "Herbaer::Translate::Learnchain::translate\n";
   }

   $text || return;
   $srl  || return;
   $tgl  || return;
   $srl =~ s/-.*// ;
   $srl = lc ($srl);
   $tgl =~ s/-.*// ;
   $tgl = lc ($tgl);

   my $tt = $self -> {"l"} -> translate ($text, $srl, $tgl);
   if (!$tt || $upd) {
      my $tt2 = $self -> {"m"} -> translate ($text, $srl, $tgl);
      if ($tt2 && (!$tt || $tt ne $tt2) ) {
         print STDERR "LCHN LEARN $text\n" if $d;
         $self -> {"l"} -> learn ($text, $srl, $tgl, $tt2);
      }
      return $tt2;
   }
   else {
      print STDERR "LCHN FOUND $text\n" if $d;
      return $tt;
   }
} # translate


sub learn {
   my ($self, $text, $srl, $tgl, $tt) = @_ ;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Learnchain::learn $tt\n";
   }
   $self -> {"m"} -> learn ($text, $srl, $tgl, $tt);
   $self -> {"l"} -> learn ($text, $srl, $tgl, $tt);
} # learn


# Name des Uebersetzers
sub translator_name {
   my $self = shift;
   ($self -> {"upd"} ? "lu" : "lc")
      . "_" . $self -> {"m"} -> translator_name()
      . "_" . $self -> {"l"} -> translator_name();
} # translator_name


sub finish {
   my $self = shift;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Learnchain::finish\n";
   }
   $self -> {"m"} -> finish ();
   $self -> {"l"} -> finish ();
} # finish


1;
```

Herbaer::Translate::MySQLLookup

[Quelltext]

Zweck

Ein Webdienst zur Übersetzung kostet Zeit und Geld. Übersetzungen speichere ich in einer MySQL-Datenbank, um Zeit und Geld zu sparen.

Die Datenbank

Die Datenbank umfasst zwei Tabellen: eine Tabelle für die Texte in einer Quellsprache (bisher nur Deutsch) und eine Tabelle für die übersetzten Texte. Das Skript `MySQLLookup_setup` richtet die Datenbank und die Zugangsdaten zur Datenbank ein.

Die Texte speichere ich als Binär-Daten ("BLOB") So vermeide ich Komplikationen mit Lokale-Einstellungen. (Um Texte in der Datenbank zu suchen, ist ein Index erforderlich. Textfelder implizieren eine alphabetische Sortierung. Wer nur einmal die Regeln für die Sortierung von Text in lateinischen Buchstaben mit Akzenten und einigen Sonderzeichen gelesen hat, der nimmt noch einen doppelten Esspresso, bevor chinesische Namen und japanische Zeichen einsortiert werden.)

Die Zugangsdaten zur Datenbank werden aus einer „Geheimnis-Datei” gelesen. Ein Beispiel für eine „Geheimnis-Datei” ist `secrets`.

Dieses Übersetzungsmodul wird normalerweise mit anderen Modulen verknüpft. Ein vorgeschaltetes Modul sollte gewährleisten, dass die Funktionen mit sinnvollen Parameter-Werte aufgerufen werden.

Dieses Modul kann auch als „Lern-Zwischenspeicher” einem Übersetzer („Lerner”) vorgeschaltet werden. Die Methode `finish` ruft dann für alle gespeicherten Übersetzungen die Methode `learn` des „Lerners” auf und löscht die Inhalte der Datenbanktabellen. Das ist nützlich, wenn eine bereits übersetzte Datei neu übersetzt werden soll (zum Beispiel mit einem besseren „End”-Übersetzer). Die Verbindung dieses Moduls und eines Übersetzungs-Webdienstes mittels `Herbaer::Translate::Learnchain` im Update-Modus würde für jeden Text den (langsamen und kostenpflichtigen) Webdienst aufrufen. Es ist besser, eine Instanz dieses Moduls und den Webdienst mittels `Herbaer::Translate::Learnchain` im Normal-Modus zu verbinden und der Instanz dieses Moduls eine zweite Instanz als „Lerner” zuzuordnen, die die Übersetzungen persistent in einer Datenbank speichert. Die folgenden Abschnitte verdeutlichen die Möglichkeiten.

Normale Übersetzung

```
                  Learnchain

"Meister"         "Lehrling"
Webdienst         MySQLLookup(Haupt-Datenbank)
```

Learnchain leitet die Übersetzungsanfrage an den "Lehrling". Wenn der Lehrling keine Übersetzung weiß, fragt Learnchain den Meister. Der Lehrling speichert dann die Übersetzung des Meisters in der Haupt-Datenbank.

Neuübersetzung ohne Zwischendatenbank

```
                  Learnchain(Update-Modus)

"Meister"         "Lehrling"
Webdienst         MySQLLookup(Haupt-Datenbank)
```

Learnchain leitet die Übersetzungsanfrage an den "Meister" und den "Lehrling". Wenn der Lehrling keine Übersetzung weiß oder der Meister eine andere Übersetzung liefert, speichert der Lehrling die Übersetzung des Meisters in der Haupt-Datenbank.

Neuübersetzung mit Zwischendatenbank

```
                  Learnchain

"Meister"         "Lehrling"
Webdienst         MySQLLookup(Zwischen-Datenbank)

                        "Lerner"
                        MySQLLookup(Haupt-Datenbank)
```

Der "Lehrling" ist hier mit einer zunächst leere Zwischen-Datenbank verbunden. Die Übersetzungsanfrage geht an Learnchain. Diese fragt den Lehrling. Wenn der Lehrling keine Antwort weiß, fragt Learnchain den "Meister". Der Lehrling speichert die Übersetzung des Meisters dann in der Zwischen-Datenbank. Am Ende überträgt der Lehrling die Datenbankeinträge von der Zwischen-Datenbank an den "Lerner", der die Übersetzungen in der Haupt-Datenbank speichert.

Funktionen

`$translator = Herbaer::Translate::MySQLLookup->new (`*`$secrets`*`, `*`$dbkennung`*`, `*`$lerner`*`, `*`$debug`*`)`

Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise nicht direkt aufgerufen, sondern von `Herbaer::Translate::new` (s. `Translate.pm`).

`$secrets` ist der Pfad der „Geheimnis-Datei”. Diese Datei enthält den Datenbanknamen und die Login-Daten. Ein Beispiel ist `secrets`. Die Daten stehen in Zeilen der Form

```
translate.mysql.DBKENNUNG=DBNAME
mysql.DBNAME.user=USER
mysql.DBNAME.password=PASSWORD
```

In der Zeile

```
translate.mysql.DBKENNUNG=DBNAME
```

in der „Geheimnis-Datei” steht der Platzhalter *`DBKENNUNG`* für den Wert von *`$dbkennung`*. Der Platzhalter *`DBNAME`* steht für den Namen der Datenbank. Eine Änderung der „Geheimnis-Datei” genügt, um eine andere Übersetzungs-Datenbank zu benutzen.

Wenn der Wert des optionalen Parameters *`$lerner`* die Kennung eines Übersetzers („Lerners”, s. `Translate.pm`) ist, ruft die Methode `finish` für jede gelernte Übersetzung die Methode `learn` des Lerners auf und löscht die Inhalte der Datenbank.

Ein logisch wahrer Wert des optionalen Parameters *`$debug`* führt zu Meldungen nach STDERR.

Diese Funktion stellt die Verbindung zur Datenbank her und bereitet die nötigen Abfragen für die Methoden `translate` und `learn` vor.

`$translator->translate ($text, $quellsprache, $zielsprache)`

Sucht die Übersetzung des Textes *`$text`* von der Sprache *`$quellsprache`* in die Sprache *`$zielsprache`* in der MySQL-Datenbank.

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

Speichert die Übersetzung *`$uebersetzung`* des Textes *`$text`* aus der Sprache *`$quellsprache`* in die Sprache *`$zielsprache`* in der MySQL-Datenbank. Eine bereits gespeicherte Übersetzung wird ersetzt.

`$translator->finish ()`

Wenn der Parameter *`$lerner`* der Methode `new` die Kennung eines Übersetzers („Lerners”) ist, ruft diese Methode (`finish`) für jede gelernte Übersetzung die Methode `learn` der Lerners auf und löscht die in der Datenbank gespeicherten Übersetzungen.

$translator->`translator_name` ()

Ergibt `mysql_`*`DBNAME`*. *`DBNAME`* ist den Name der Datenbank, der unter dem Schlüssel `translate.mysql_dbname` aus der „Geheimnis-Datei” gelesen wird. Wenn der Parameter *`$lerner`* der Methode `new` die Kennung eines Übersetzers („Lerners”) ist, wird ein Unterstrich und der Nahme des Lerners (Ergbnis der Methode `translator_name`) angehängt.

$translator->`DESTROY` ()

Der Destruktor schließt die Verbindung zur Datenbank.

Quelltext

[Beschreibung]

```
# Übersetzung in MySQL-Datentenbanksdatenbank suchen und einfügen
# 2016-07-18, Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::MySQLLookup ;


use parent Herbaer::Translate::Base ;
use Encode;
use utf8;


sub new {
   use DBI;
   my ($class
       , $secrets
       , $dbkey
       , $learner
       , $dbg
      ) = @_;
   $dbg ||= 0;
   if ($dbg) {
      print STDERR "Herbaer::Translate::MySQLLookup::new\n";
   }
   my $dbname;

   my $hnd;
   if (! open ($hnd, "<:encoding(utf-8)", $secrets)) {
      print STDERR "Kann Datei $secrets nicht lesen:$!\n" if $dbg;
      return undef;
   }
   my $line;
   my $user;
   my $password;
   while (defined ($line = <$hnd>)) {
      $line =~ s/\s*$//;
      if ( $line =~ /^\s*translate.mysql.([a-z0-9]+)\s*=\s*(.+)/ ) {
         if ($1 eq $dbkey) {
            $dbname = $2;
         }
      }
      elsif ($line =~ /^\s*mysql\.([a-z0-9]+)\.([a-z]+)\s*=\s*(.+)/) {
         if ($dbname && $1 eq $dbname) {
            if ($2 eq "user") {
               $user = $3;
            }
            elsif ($2 eq "password") {
               $password = $3;
            }
         }
      }
   }
   close $hnd;
   my $db = DBI -> connect ("DBI:mysql:$dbname", $user, $password);
   if (!$db) {
      print STDERR "Keine Verbindung zur MySQL-Datenbank $dbname\n" if $dbg;
      return undef;
   }
   my $self = {
      # Datenbank-Handles
      "db"  => $db,
      "ss"  => $db -> prepare (
         "SELECT id FROM src WHERE lang = ? AND txt = ?"
         ),
      "st"  => $db -> prepare (
         "SELECT txt FROM translation WHERE src_id = ? AND lang = ?"
         ),
      "is"  => $db -> prepare (
         "INSERT INTO src (lang, txt) VALUES (?, ?)"
         ),
      "it"  => $db -> prepare (
         "INSERT INTO translation (txt, src_id, lang) VALUES (?, ?, ?)"
         ),
      "ut"  => $db -> prepare (
         "UPDATE translation SET txt = ? WHERE src_id = ? AND lang = ?"
         ),
      "dbg"     => $dbg,
      "dbname"  => $dbname,
      "learner" => $learner,
   } ;
   return bless ($self, $class);
} # new
```

```
sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $dbg = $self -> {"dbg"};
   print STDERR "Herbaer::Translate::MySQLLookup::translate\n  $srl -> $tgl: $text\n"
      if $dbg;

   # hier keine Prüfung der Parameter
   my $h = $self -> {"ss"};
   my $ar;
   my $id;
   my $tt;
   $h -> execute ($srl, encode ("utf-8", $text));
   if ($ar = $h -> fetchrow_arrayref()) {
      $id = $ar -> [0];
      print STDERR "  Gefunden id: $id\n" if $dbg;
   }
   if ($id) {
      $h = $self -> {"st"};
      $h -> execute ($id, $tgl);
      if ($ar = $h -> fetchrow_arrayref()) {
         $tt = decode ("utf-8", $ar -> [0]);
         print STDERR "  Übersetzung gefunden\n" if $dbg;
      }
   }
   $tt;
} # translate


sub learn {
   my ($self, $text, $srl, $tgl, $trans) = @_ ;
   my $dbg = $self -> {"dbg"};
   print STDERR
      "Herbaer::Translate::MySQLLookup::learn\n",
      "  $srl: $text\n",
      "  $tgl: $trans\n"
      if $dbg;
   # hier keine Parameter-Prüfung

   my $id;
   my $ar;
   my $h = $self -> {"ss"};
   $text  = encode ("utf-8", $text);
   $trans = encode ("utf-8", $trans);
   $h -> execute ($srl, $text);
   if ($ar = $h -> fetchrow_arrayref()) {
      $id = $ar -> [0];
      print STDERR "  Gefunden id: $id\n" if $dbg;
   }
   if (!$id) {
      $h = $self -> {"is"};
      $h -> execute ($srl, $text);
      $id = $h -> {"mysql_insertid"};
      print STDERR "  Eingefügt id: $id\n" if $dbg;
   }
   return 0 if !$id;
   my $tt;
   $h = $self -> {"st"};
   $h -> execute ($id, $tgl);
   if ($ar = $h -> fetchrow_arrayref()) {
      $tt = $ar -> [0] || "";
      $self -> {"ut"} -> execute ($trans, $id, $tgl) if $tt ne $trans;
   }
   else {
      $self -> {"it"} -> execute ($trans, $id, $tgl);
   }
   return 1;
} # learn


sub _learner {
   my $self = shift;
   my $lrn;
   if (! ( $lrn = $self -> {"lrn"}) && $self -> {"learner"} ) {
      $self -> {"lrn"} ||= new Herbaer::Translate ($self -> {"learner"});
   }
   $lrn;
} # _learner
```

```
sub finish {
   my $self = shift;
   my $dbg = $self -> {"dbg"};
   print STDERR "Herbaer::Translate::MySQLLookup::finish\n" if $dbg;
   if ($self -> {"learner"}) {
      my $lrn = $self -> _learner ();
      my $db = $self -> {"db"};
      if (!$lrn) {
         print STDERR
            "Herbaer::Translate::MySQLLookup::finish\n",
            "Kein Übersetzer mit Kennung \"", $self -> {"learner"}, "\"\n"
            if $dbg;
      }
      elsif (!$db) {
         print STDERR
            "Herbaer::Translate::MySQLLookup::finish\n",
            "Keine Verbindung zur Datenbank \"", $self -> {"dbname"}, "\"\n"
            if $dbg;
      }
      else {
         my $ss = $db -> prepare ("SELECT id, lang, txt FROM src");
         my $st = $db -> prepare ("SELECT lang, txt FROM translation where src_id = ?");
         my $id;
         my $sr;
         my $tr;
         my $srl;
         my $text;
         my $tgl;
         my $trans;
         $ss -> execute ();
         while ($sr = $ss -> fetchrow_arrayref()) {
            ($id, $srl, $text) = @$sr;
            $text = decode ("utf-8", $text);
            $st -> execute ($id);
            while ($tr = $st -> fetchrow_arrayref()) {
               ($tgl, $trans) = @$tr;
               $lrn -> learn ($text, $srl, $tgl, decode ("utf-8", $trans));
            }
         }
         $ss -> finish ();
         $st -> finish ();
         $db -> do ("DELETE FROM src");
         $db -> do ("DELETE FROM translation");
         $lrn -> finish ();
      }
   }
   $self;
} # finish


# Name des Uebersetzers
sub translator_name {
   my $self = shift;
   my $lrn  = $self -> _learner ();
   my $tn   = "mysql_" . $self -> {"dbname"};
   if ($lrn) {
      $tn . "_" . $lrn -> translator_name ();
   }
   else {
      $tn;
   }
} # translator_name


sub DESTROY {
   my $self = shift;
   my $dbg = $self -> {"dbg"};
   print STDERR "Herbaer::Translate::MySQLLookup::DESTROY\n" if $dbg;
   if ($self -> {"db"}) {
      $self -> {"ss"} -> finish();
      $self -> {"st"} -> finish();
      $self -> {"is"} -> finish();
      $self -> {"it"} -> finish();
      $self -> {"ut"} -> finish();
      $self -> {"db"} -> disconnect();
   }
} # DESTROY


1;
```

MySQLLookup_setup

[Quelltext]

Übersicht

`MySQLLookup_setup --help|--version`

`MySQLLookup_setup [ --verbose … | --no_verbose ][ --withcred | --no_withcred ]`
`[ --srcdir` *`SRCDIR`* `][ --user` *`USER`* `]`
`[ --password` *`PASSWORD`* `][ --secrets` *`SECRETS`* `]`
`[ --dbs ][ --users ]`

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus.

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Meldungen werden nach STDOUT ausgegeben.

`--no_verbose`

Diese Option hebt die Wirkung der Option `--verbose` auf.

`--withcred`

Zum Anlegen der Datenbank wird ein Kennwort benötigt.

`--no_withcred`

Diese Option hebt die Wirkung der Option `--withcred` auf.

`--srcdir` *`SRCDIR`*

Im Verzeichnis `SRCDIR` werden das Skript `MySQLLookup_setup.sql` und das Perl-Skript `replace.pl` gesucht.

`--user` *`USER`*

`USER` ist der Benutzername zur Anmeldung am Datenbank-Server. Der Wert wird nur benutzt, wenn die Option `--withcred` gesetzt ist.

`--password` *`PASSWORD`*

`PASSWORD` ist das Kennwort zur Anmeldung am Datenbank-Server. Es wird nur benutzt, wenn die Option `--withcred` gesetzt ist.

`--secrets` *`SECRETS`*

Aus der Datei *`SECRETS`* werden Benutzernamen und Kennwörter zur Anmeldung an den Datenbanken translate und translrnbuf gelesen.

`--dbs`

Zum Anlegen der Datenbanken wird das Skript `MySQLLookup_dbs.sql` ausgeführt.

`--users`

Zum Anlegen der Datenbank-Nutzer wird das Skript `MySQLLookup_users.sql` ausgeführt. Das ist nötig, wenn der Datenträger an einen neuen Rechner angeschlossen wird.

Beschreibung

Das Skript `MySQLLookup_setup` richtet die Übersetzungs-Datenbanken für das Übersetzungsmodul `MySQL-Lookup.pm` ein. Dazu werden zwei SQL-Skripte ausgeführt: `MySQLLookup_dbs.sql` legt die Datenbanken an, `MySQLLookup_users.sql` legt die Zugangsdaten für den Zugriff auf die Datenbanken fest.

`MySQLLookup_dbs.sql` wird nur einmal zum Einrichten der Datenbanken ausgeführt. Solange es kein ernsthaftes Datenbankproblem gibt, sollte dieses Skript nicht ausgeführt werden.

`MySQLLookup_users.sql` ist immer auszuführen, wenn der Datenträger mit den Datenbanken an einen neuen Rechner angeschlossen wird.

Ohne die Optionen `--dbs` oder `--users` (oder mit beiden Optionen) werden beide Skripte nach einander ausgeführt.

Dateien

`SRCDIR``/MySQLLookup_dbs.sql`

SQL-Skript legt die Datenbanken an.

`SRCDIR``/MySQLLookup_setup.sql`

SQL-Skript legt die Zugangsdaten (Benutzername und Kennwort) zum Zugriff auf die Datenbanken fest. Es enthält Platzhalter, deren Werte das Skript `replace.pl` aus der „Geheimnis-Datei” *`SECRETS`* liest.

`SRCDIR``/replace.pl`

Das Perl-Skript ersetzt die Platzhalter in den SQL-Skripten durch die Werte, die es aus *`SECRETS`* liest.

Quelltext

[Beschreibung]

```
#!/bin/bash
# -*- coding:utf-8 -*-
# Übersetzungs-Datenbank einrichten
# 2016-07-20 Herbert Schiemann <h.schiemann@herbaer.de>
# 2020-04-01 --withcred, --users, --dbs


# Zähler, Variable, Aktionen
declare_vars ()
{
   # Zähler
   g_counters="  \
      verbose    \
      withcred   "

   # Variable
   g_variables=" \
      srcdir     \
      user       \
      password   \
      secrets    ";

   # Aktionen
   g_actions="   \
      users      \
      dbs        ";
} # declare_vars


# setzt Vorgabe-Werte
set_defaults ()
{
   local basedir=$(realpath $0) ;
   basedir=${basedir%/src/localization/MySQLLookup_setup};
   [[ -n "$verbose"       ]] || verbose=1 ;
   [[ -n "$withcred"      ]] || withcred=0 ;
   [[ -n "$user"          ]] || user=root ;
   [[ -n "$password"      ]] || password= ;
   [[ -n "$srcdir"        ]] || srcdir="$basedir/src/localization" ;
   [[ -n "$secrets"       ]] || secrets="$basedir/secrets" ;
} # set_defaults


# Zeigt eine kurze Hilfe an
show_help ()
{
   local cmd=${0#*/} ;
   set_defaults ;
   cat << .HELP ;
$cmd --version
$cmd --help
$cmd OPTION ..

Optionen
--[no_]verbose        Erhöht den Umfang der Ausgabe des Scripts ($verbose)
--[no_]withcred       Benutzername und Kennwort benutzen ($withcred)
--user     USER       MySQL-Root-User ($user)
--password PASSWORD   MySQL-Root-Kennwort
--srcdir   SRCDIR     Skript-Verzeichnis ($srcdir)
--secrets  SECRETS    Pfad der "Geheimnis"-Datei ($secrets)
--users               Datenbank-Nutzer anlegen
--dbs                 Datenbanken anlegen
.HELP
} # show_help


# Zeigt die Version an
show_version ()
{
   cat << .VERSION ;
web/src/localization/MySQLLookup_setup
Übersetzungsdatenbank einrichten
2020-04-01, Herbert Schiemann, h.schiemann@herbaer.de
GPL Version 2 oder neuer
siehe MySQLLookup.pm
.VERSION
} # show_version
```

```
# Variable und Zähler initialisieren
init_vars () {
   local v;
   declare_vars ;
   for v in $g_counters $g_variables $g_actions; do
      eval "$v=" ;
   done;
} # init_vars


# Argumente verarbeiten
read_args ()
{
   local wd ;
   local lastwd ;
   local var ;
   local ok ;

   has_actions=0 ;
   for wd in "$@"; do
      if [[ "$lastwd" = "--" ]]; then
          _argv="$_argv $wd";
      elif [[ -n "$lastwd" ]]; then
         if [[ "$wd" =~ ^[\ a-zA-Z0-9./_#-]+$ ]]; then
            ok=0 ;
            for var in $g_variables; do
               if [[ "$var" == "$lastwd" ]]; then
                  (( ++ok )) ;
                  eval "$var=\"$wd\"" ;
                  break ;
               fi ;
            done ;
            if (( ! ok )); then
               (( verbose )) && echo "Unbekannte Option --$lastwd $wd" ;
               exit 11 ;
            fi ;
         else
            (( verbose )) && echo "Ungültiger Optionswert --$lastwd $wd" ;
            exit 12;
         fi;
         lastwd= ;
      else
         case "$wd" in
            --version )
               show_version ;
               exit 0 ;
               ;;
            --help )
               show_version ;
               show_help ;
               exit 0 ;
               ;;
            -- )
               if [[ -n "$_argv" ]]; then
                  lastwd=--;
                  continue;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 13 ;
               fi ;
               ;;
            --* )
               if [[ "$wd" =~ ^--[a-z][a-z0-9_]*$ ]]; then
                  lastwd=${wd#--} ;
                  ok=0 ;
                  for var in $g_counters ; do
                     if   [[ "$lastwd" == $var    ]]; then
                        eval "(( ++$lastwd ))" ;
                     elif [[ "$lastwd" == "no_$var" ]]; then
                        eval "${lastwd#no_}=0" ;
                     else
                        continue;
                     fi;
                     (( ++ok )) ;
                     break ;
                  done;
                  if (( !ok )); then
                     for var in $g_actions; do
                        if [[ "$lastwd" == "$var" ]]; then
                           eval "(( ++$var ))" ;
                           (( ++ok ));
                           has_actions=1;
                           break;
                        elif [[ "$lastwd" == "no_$var" ]]; then
                           eval "(( ++no_$var ))" ;
                           (( ++ok ));
                           break;
                        fi;
                     done;
                  fi;
                  (( ok )) && lastwd=;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 14 ;
               fi ;
               ;;
```

```
            * )
               (( verbose )) && echo "Ungültige Option $wd" ;
               exit 15 ;
               ;;
         esac ;
      fi ;
   done ;
   if [[ -n $lastwd && "$lastwd" != "--" ]]; then
      (( verbose )) && echo "Unverarbeitete Option --$lastwd";
      exit 16 ;
   fi ;
} # read_args


# Aktionen ausführen
run_actions ()
{
   local act ;
   for act in $g_actions; do
     eval "(( ! has_actions && ! no_$act || $act )) && process_$act";
   done;
} # run_actions


# show_variables VARNAME1 VARNAME2
# Werte der Variablen anzeigen
show_variables ()
{
   local v ;
   for v in $g_counters $g_variables $1; do
      eval "echo \"$v = \$$v\"" ;
   done;
} # show_variables


# Können die Eingabedateien gelesen werden?
# check_infiles first/path/to/file path/to/second_file ;
check_infiles ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -s "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist leer";
         return 1;
      fi;
      if [[ ! -r "$f" ]]; then
         (( verbose )) && echo "Kann Datei \"$f\" nicht lesen";
         return 1;
      fi;
   done;
   return 0;
} # check_infiles


# Sind die Dateien ausführbar?
# check_executeable first/path/to/script path/to/second_srcipt ;
check_executeable ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -x "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine ausführbare Datei";
         return 1;
      fi;
   done;
   return 0;
} # check_executeable


# run_sql name (users, dbs)
run_sql ()
{
   sql=$srcdir/MySQLLookup_$1.sql ;
   check_infiles $sql || return 1;
   (( verbose )) && echo $sql;
   cat $sql | $rpl --val $secrets | mysql $cre ;
} # run_sql


# Datenbanknutzer anlegen
process_users ()
{
   run_sql users ;
} # process_users


# Datenbanken anlegen
process_dbs ()
{
   run_sql dbs ;
} # process_dbs
```

```
# Sicherheit
export PATH=/bin:/usr/bin ;
IFS=$' \t\n' ;
init_vars ;
set -o noclobber ;   # existierende Dateien werden nicht überschrieben
shopt -s extglob nullglob ;
read_args "$@" ;
set_defaults ;
cre= ;
if (( withcred )); then
   if [[ -z "$password" ]]; then
      echo "PASSWORD nicht angegeben";
      show_help;
      exit 101;
   fi;
   if [[ -z "$user" ]]; then
      echo "USER nicht angegeben";
      show_help;
      exit 102;
   fi;
   cre="-u $user --password=$password" ;
fi;


(( verbose > 1 )) && show_variables;
rpl=$srcdir/replace.pl ;
check_executeable $rpl  || exit 103;
check_infiles  $secrets || exit 104;


run_actions;
exit 0;
```

Datei MySQLLookup_dbs.sql

```
# file KLEIDER/web/src/localization/MySQLLookup_dbs.sql
# 2020-04-01 Herbert Schiemann <h.schiemann@herbaer.de>
# Legt die Datenbanken "translate" und "translrnbuf" an.


CREATE DATABASE translate;
USE translate;
CREATE TABLE src (
       id   INT        NOT NULL AUTO_INCREMENT,
       lang VARCHAR(5) NOT NULL,
       txt  BLOB       NOT NULL,
       PRIMARY KEY (id),
       INDEX txt (txt(100))
);
CREATE TABLE translation (
       src_id      INT        NOT NULL,
       lang        VARCHAR(5) NOT NULL,
       txt         BLOB       NOT NULL,
       PRIMARY KEY (src_id, lang)
);


CREATE DATABASE translrnbuf;
USE translrnbuf;
CREATE TABLE src (
       id   INT        NOT NULL AUTO_INCREMENT,
       lang VARCHAR(5) NOT NULL,
       txt  BLOB       NOT NULL,
       PRIMARY KEY (id),
       INDEX txt (txt(100))
);
CREATE TABLE translation (
       src_id      INT        NOT NULL,
       lang        VARCHAR(5) NOT NULL,
       txt         BLOB       NOT NULL,
       PRIMARY KEY (src_id, lang)
);
```

Datei MySQLLookup_users.sql

```
# file KLEIDER/web/src/localization/MySQLLookup_users.sql
# 2020-04-01 Herbert Schiemann <h.schiemann@herbaer.de>
# Legt die Benutzer der Datenbanken "translate" und "translrnbuf" an.


USE mysql;
DELETE FROM user WHERE user = '${mysql.translate.user}';
INSERT INTO user
       (user, host, password)
       VALUES (
              '${mysql.translate.user}',
              'localhost',
              PASSWORD('${mysql.translate.password}')
       );
FLUSH PRIVILEGES;
GRANT ALL PRIVILEGES ON translate.* TO ${mysql.translate.user}@localhost;


DELETE FROM user WHERE user = '${mysql.translrnbuf.user}';
INSERT INTO user
       (user, host, password)
       VALUES (
              '${mysql.translrnbuf.user}',
              'localhost',
              PASSWORD('${mysql.translrnbuf.password}')
       );
FLUSH PRIVILEGES;
GRANT ALL PRIVILEGES ON translrnbuf.* TO ${mysql.translrnbuf.user}@localhost;
```

Herbaer::Translate::NameReplacer

[Quelltext]

Zweck

Möglicherweise erkennt ein Übersetzer einen Namen nicht als Namen, sondern versucht, ihn als ein (deutsches) Wort mit einer Bedeutung zu übersetzen. Dieses Modul ersetzt Namen vor der Übersetzung durch sinnfreie Zeichenfolgen, die im übersetzten Text unverändert erscheinen. Die Zeichenfolgen werden dann in der Übersetzung durch die Namen ersetzt.

Als Name gelten Textteile, die auf einen regulären Ausdruck passen. Abhängig von den Sprachen werden Listen regulärer Ausdrücke aus Textdateien („Namensdateien”) gelesen. Die Namensdateien werden zeilenweise verarbeitet. Eine Zeile, die mit dem Zeichen # beginnt, ist ein Kommentar. Zeilen, die nur aus Leerzeichen bestehen, werden ignoriert. Von allen anderen Zeilen werden Leerzeichen am Anfang und am Ende der Zeile entfernt. Die verbleibende Zeichenfolge wird als regulärer Ausdruck interpretiert. Die regulären Ausdrücke werden in der Reihenfolge der Datei mit dem zu übersetzenden Text abgeglichen. Ein passender Textteil (Name) wird durch die Zeichenfolge XKC*n*CKX (Platzhalter) ersetzt. *n* steht für einen fortlaufenden Zähler. Jeder reguläre Ausdruck wird so oft mit dem Text abgeglichen, bis kein passender Textteil gefunden wird. Die Platzhalter (XKC*n*CKX) erscheinen unverändert im übersetzten Text. Im entsprechenden Namen werden Folgen aufeinanderfolgender Leerzeichen durch ein einzelnes Leerzeichen ersetzt. Dann wird der Name an Stelle des Platzhalters in der Übersetzung eingesetzt.

Für Übersetzungen aus der Sprache *SRCLANG* in die Sprache *TARGETLANG* werden der Reihe nach die Namensdateien mit dem Dateinamen *SRCLANG_TARGETLANG*, *SRCLANG* und `default` im angegebenen Verzeichnis (s. `new`) gesucht. Die erste gefundene Datei bestimmt die Liste der regulären Ausdrücke.

`NameReplacer.beispiel` ist ein Beispiel für eine Namensdatei.

Funktionen

`$translator = Herbaer::Translate::NameReplacer->new (`*`$directory, $translator_name, $debug`*`)`

Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise nicht direkt aufgerufen, sondern von `Herbaer::Translate::new` (s. `Translate.pm`).

`$directory` ist der Pfad des Verzeichnisses, in dem die Namensdateien (mit Listen regulärer Ausdrücke) gesucht werden.

`$translator_name` ist die Kennung des „beauftragten” Übersetzers, s. `Herbaer::Translate` (`Translate.pm`).

Ein logisch wahrer Wert des optionalen Parameters *`$debug`* führt zu Meldungen nach STDERR.

`$translator->translate ($text, $quellsprache, $zielsprache)`

Ergibt die Übersetzung des Textes *`$text`* aus der Sprache *`$quellsprache`* in die Sprache *`$zielsprache`* nach der Ersetzung von Namen im Quelltext *`$text`* durch Platzhalter und der „Rückersetzung” der Platzhalter im übersetzten Text durch die Namen.

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

Ruft die Methode `learn` des „beauftragten” Übersetzers auf.

$translator->`finish` ()

Ruft die Methode `finish` des „beauftragten” Übersetzers auf.

$translator->`translator_name` ()

Ergibt `nr_`*`TRANSTOR_NAME`*. *`TRANSTOR_NAME`* steht für das Ergebnis der Methode `translator_name` des „beauftragten” Übersetzers.

Quelltext

[Beschreibung]

```
# Namen durch abstrakte Zeichenfolgen ersetzen
# 2015-08-13 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::NameReplacer ;


use parent Herbaer::Translate::Base ;
use File::Spec::Functions qw (catfile) ;
use Herbaer::Translate ;
use utf8;


sub new {
   my ($class
       , $dir
       , $trname,
       , $dbg
      ) = @_;
   use File::Spec::Functions qw(rel2abs) ;
   $dbg ||= 0;
   if ($dbg) {
      print STDERR "Herbaer::Translate::NameReplacer::new\n";
   }
   $dir ||= "names";
   $dir = rel2abs($dir);
   if (! -d $dir) {
      print STDERR
         "  $dir ist kein Verzeichnis,\n",
         "  Wörter werden nicht ersetzt\n";
      $dir = undef;
   }
   my $self = {
      "dbg"    => $dbg,
      "dir"    => $dir,      # Verzeichnis der Wortersetzungstabellen
      "names"  => {},        # lang -> [REGEX, ...]
      "trname" => $trname,
   } ;
   if ($dbd && $dbg >= 1) {
      print STDERR "  Verzeichnis der Namendateien: $dir\n" ;
   }
   return bless ($self, $class);
} # new


# liest eine Namensliste
sub _getdata {
   my ($self, $key) = @_ ;
   my $dbg = $self -> {"dbg"};
   print STDERR "Herbaer::Translate::NameReplacer::_getdata (\"$key\")\n" if $dbg;
   my $n = $self -> {"names"};
   my $l = [];
   $n -> {$key} = $l;
   my $k2;
   if ($key =~ /^([^_])+_/) {
      $k2 = $1;
   }
   my $p = catfile ($self -> {"dir"}, $key) ;
   if (! -f $p) {
      if ($k2) {
         if ($n -> {$k2}) {
            $n -> {$key} = $n -> {$k2};
            print STDERR
               "Herbaer::Translate::NameReplacer::_getdata: ",
               "\"$k2\" für Schlüssel \"$key\"\n"
               if $dbg;
            return $n -> {$key};
         }
         $n -> {$k2} = $l;
      }
      $p = catfile ($self -> {"dir"}, $k2) ;
   }
   if (! -f $p) {
      print STDERR
         "Herbaer::Translate::NameReplacer::_getdata: ",
         "\"default\" für Schlüssel \"$key\"\n"
         if $dbg;
      if ($k2) {
         $n -> {$k2} = $n -> {"default"};
         print STDERR
            "Herbaer::Translate::NameReplacer::_getdata: ",
            "\"default\" für Schlüssel \"$k2\"\n"
            if $dbg;
      }
      if ($n -> {"default"}) {
         $n -> {$key} = $n -> {"default"};
         $n -> {$k2}  = $n -> {"default"} if $k2;
         return $n -> {$key};
```

```
      }
      $n -> {"default"} = $l;
      $p = catfile ($self -> {"dir"}, "default");
   }
   my $h;
   my $ln;
   if (! open ($h, "<:encoding(utf-8)", $p)) {
      print STDERR "Herbaer::Translate::NameReplacer Kann Datei $p nicht lesen:$!\n"
         if $dbg;
      return $l;
   }
   while (defined ($ln = <$h>)) {
      next if $ln =~ /^#/ ;
      $ln =~ s/\s*$// ;
      $ln =~ s/^\s*// ;
      next unless $ln ;
      push (@$l, qr($ln) );
   }
   close $h;
   $l;
} # _getdata


sub _gettrans {
   my $self = shift;
   my $trans = $self -> {"trans"};
   if (! $trans) {
      $trans = new Herbaer::Translate ($self -> {"trname"});
      if (! $trans) {
         print STDERR "Kann Übersetzer \"", $self -> {"trname"}, "\" nicht laden.\n"
            if $self -> {"dbg"};
         return undef;
      }
      $self -> {"trans"} = $trans;
   }
   $trans;
} # _gettrans


sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $dbg = $self -> {"dbg"};
   if ($dbg) {
      print STDERR "Herbaer::Translate::NameReplacer::translate\n";
   }
   # Parameter können sinnvoller geprüft werden.
   $text || return;
   $srl  || return;
   $tgl  || return;

   $srl =~ s/-.*// ;
   $tgl =~ s/-.*// ;
   my $key = "${srl}_${tgl}" ;
   my $xkc = {};
   my $names = $self -> {$key} || $self -> _getdata ($key);
   my $li;
   my $re;
   my $name;
   my $n = 1;
   for $li (@$names) {
      $re = $li;
      $key = "XKC${n}CKX";
      while ( $text =~ s/$re/$key/e ) {
         $name = $&;
         $name =~ s/\s+/ /;
         $xkc -> {$key} = $name ;
         if ($dbg) {
            print STDERR "KEY $key\n";
            print STDERR "NAME $name\n";
            print STDERR "TEXT $text\n";
         }
         ++$n;
         $key = "XKC${n}CKX";
      }
   }
   my $trans = $self -> _gettrans () or return "";
   my $tt = $trans -> translate ($text, $srl, $tgl);
   while ( ($key, $name) = each %$xkc ) {
      if ($dbg) {
         print STDERR "KEY $key\n";
         print STDERR "NAME $name\n";
         print STDERR "TT $tt\n";
      }
      $tt =~ s/$key/$name/e ;
   }
   return $tt;
} # translate
```

```
sub learn {
   my ($self, $text, $srl, $tgl, $tt) = @_ ;
   my $dbg = $self -> {"dbg"};
   if ($dbg) {
      print STDERR "Herbaer::Translate::NameReplacer::learn\n";
   }
   # Parameter können sinnvoller geprüft werden.
   $text || return;
   $srl  || return;
   $tgl  || return;
   $tt   || return;

   my $trans = $self -> _gettrans ();
   $trans -> learn ($text, $srl, $tgl, $tt) if $trans;
} # learn


# Name des Uebersetzers
sub translator_name {
   my $self = shift;
   my $tn = "nr" ;
   my $trans = $self -> _gettrans ();
   $tn .= "_" . $trans -> translator_name () if $trans;
   $tn;
} # translator_name


# speichert die gelernten Daten
sub finish {
   my $self = shift;
   my $dbg = $self -> {"dbg"};
   if ($dbg) {
      print STDERR "Herbaer::Translate::NameReplacer::finish\n";
   }
   my $trans = $self -> {"trans"};
   $trans -> finish () if $trans;
} # finish


1;
```

Datei NameReplacer.beispiel

```
# file KLEIDER/web/src/localization/NameReplacer.beispiel
# Beispiel einer Namensersetzungsdatei für deutsche Quelltexte
# zum Perl-Modul Herbaer::Translate::NameReplacer
# Originaldatei im Verzeichnis KLEIDER/web/transdb/names
# Originaldateiname z.B. "de" für Übersetzungen aus der deutschen Sprache
# in irgendeine andere Sprache
# oder "de_zh" für Übersetzungen vom Deutschen ins Chinesische.


# Personennamen
\bHerbert\s+Schiemann
\bHeinz\s+Oestergaard
\bOestergaard


# Ortsnamen und andere Bezeichner
Borkener\s+Str(?:\.|aße)\s+167
# Ortsname mit oder ohne Postleitzahl
\d{5}\s+Dorsten
\bDorsten\b


# allgemein Folgen von Großbuchstaben
\b[A-ZÄÖÜ]{3,4}\b
\b[A-ZÄÖÜ]{3}[A-ZÄÖÜ ]*?[A-ZÄÖÜ]{2}\b
```

Herbaer::Translate::Normalizer

[Quelltext]

Zweck

Text, der nicht übersetzt werden soll oder kann, wird vorher durch Platzhalter XK*id*KX ersetzt. Der Wert *id* hängt von der Position des Elements im XML-Dokument ab. Er ändert sich, wenn ein Element oder ein Attribut eingefügt wird. Ohne dieses Modul würde der Text mit dem neuen *id*-Wert in der Übersetzungsdatenbank gesucht. Der Text mit dem alten *id*-Wert würde nicht gefunden.

Die Methode `translate` ersetzt die Zeichenfolgen XK*id*KX in der Quellsprache durch WX*nn*XW, läßt den „normalisierten” Text durch den beauftragten Übersetzer übersetzen und ersetzt im übersetzten Text die Zeichenfolgen WX*nn*XW durch XK*id*KX. *nn* steht hier für eine fortlaufende Zahl, die für jede neue Übersetzung bei 1 beginnt.

Funktionen

`$translator = Herbaer::Translate::Normalizer->new (`*`$trname`*`, `*`$debug`*`)`

> Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise von `Herbaer::Translate::new` aufgerufen (s. `Translate.pm`).
>
> *`$trname`* ist die Kennung des „beauftragten” Übersetzers.
>
> Ein logisch wahrer Wert des optionalen Parameters *`$debug`* führt zu Meldungen nach STDERR.

`$translator->translate ($text, $quellsprache, $zielsprache)`

> Ersetzt Platzhalter für nicht übersetzbare Textteile, bevor der „beauftragte” Übersetzer arbeitet, siehe Abschnitt „Zweck”.

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

> Ruft die Methode `learn` des beauftragten Übersetzers auf.

$translator->`translator_name` ()

> Ergibt `norm_`*`TRANSLATOR_NAME`*. Der Platzhalter *`TRANSLATOR_NAME`* steht für das Ergebnis der Methode `translator_name` des beauftragten Übersetzers.

$translator->`finish` ()

> Ruft die Methode `finish` des beauftragten Übersetzers auf.

Quelltext

[Beschreibung]

```
# Platzhalter durch Platzhalter in Normalform ersetzen
# 2015-09-28 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::Normalizer ;


use parent Herbaer::Translate::Base ;
use utf8;


sub new {
   my ($class
       , $trname
       , $debug
      ) = @_;
   $debug ||= 0;
   if ($debug) {
      print STDERR "Herbaer::Translate::Normalizer::new\n";
   }
   my $self = {
      "t"   => new Herbaer::Translate ($trname),
      "dbg" => $debug,
   };
   return bless ($self, $class);
} # new


sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::Normalizer::translate\n";
   }

   # Parameter prüfen
   $text || return;
   $srl  || return;
   $tgl  || return;
   $srl =~ s/-.*// ;
   $srl = lc ($srl);
   $tgl =~ s/-.*// ;
   $tgl = lc ($tgl);

   my $subs = {};
   my $bsbs = {};

   my $num = 0;
   my $k;
   my $s;
   my $repl = sub {
      $k = shift;
      if (! ($s = $subs -> {$k}) ) {
         $s = "WX" . ++$num . "XW";
         $subs -> {$k} = $s;
         $bsbs -> {$s} = $k;
      }
      $s;
   };
   print STDERR "NORM IN:$text\n" if $debug;
   # XKe315KX
   $text =~ s/(XK[a-z0-9]+KX)/$repl -> ($1)/ge;
   print STDERR "NORM MOD:$text\n" if $debug;
   my $tt = $self -> {"t"} -> translate ($text, $srl, $tgl);
   if ($tt) {
      print STDERR "NORM TRT:$tt\n" if $debug;
      while ( ($k, $s) = each %$bsbs ) {
         $tt =~ s/$k/$s/ge ;
      }
      print STDERR "NORM OUT:$tt\n" if $debug;
   }
   $tt;
} # translate


sub learn {
   my ($self, $text, $srl, $tgl, $tt) = @_ ;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Normalizer::learn\n"
   }
   $self -> {"t"} -> learn ($text, $srl, $tgl, $tt);
} # learn


# Name des Uebersetzers
sub translator_name {
   my $self = shift;
   "norm_" . $self -> {"t"} -> translator_name () ;
} # translator_name
```

```
sub finish {
   my $self = shift;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Normalizer::finish\n";
   }
   $self -> {"t"} -> finish ();
} # finish


1;
```

Herbaer::Translate::Pipe

[Quelltext]

Beschreibung

Damit nicht zu viel Zeit benötigt wird, Übersetzer-Prozesse zu starten und zu beenden, kann dieses Modul mit einem „persistenten” Übersetzerprozess. über benannte Pipes kommunizieren. Das Programm `pipe_srv.pl` startet den Übersetzerprozess, das Programm `pipe_srv_stop.pl` beendet den Übersetzerprozess.

Die Methode `_send_request` dieses Moduls `Herbaer::Translate::Pipe` (Datei `Pipe.pm`) sendet einen einzeiligen Befehl an den Übersetzer über die Pipe *`BASIS`*`/request` und liest die Antwort aus der Pipe *`BASIS`*`/response`. Die Parameter einer Methode entsprechen Feldern, die durch einen Doppelpunkt getrennt werden. Der Doppelpunkt und das Zeilenendezeichen innerhalb von Parameterwerten werden daher anders kodiert („geschützt”). Dazu dient die Funktion `_encode`. Die Funktion `_decode` macht die Kodierung durch `_encode` rückgängig.

Funktionen

`$translator = Herbaer::Translate::Pipe->new (`*`$basis`*`, `*`$trname`*`, `*`$debug`*`)`

Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise nicht direkt aufgerufen, sondern von `Herbaer::Translate::new` (s. `Translate.pm`).

Der Dateipfad für eine Anfrage an den Übersetzerprozess ist *`$base`*`/request`, der Dateipfad der Antwort ist *`$base`*`/response`.

`$trname` ist die Kennung des Übersetzers, den der Übersetzerprozess verwenden soll. Dem Übersetzerprozess wird die Anfrage `trname:`*`$trname`* gesendet.

Ein logisch wahrer Wert des optionalen Parameters *`$debug`* führt zu Meldungen nach STDERR.

`$translator->translate ($text, $quellsprache, $zielsprache)`

Sendet die Anfrage `translate:`*`$text`*`:`*`$quellsprache`*`:`*`$zielsprache`* an den Übersetzer. Der sendet die kodierte Übersetzung als Antwort.

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

Sendet die Anfrage `learn:`*`$text`*`:`*`$quellsprache`*`:`*`$zielsprache`*`:`*`$uebersetzung`* an den Übersetzerprozess und gibt dessen Antwort zurück.

$translator->`finish` ()

Sendet die Anfrage `finish` an den Übersetzer und gibt dessen Antwort zurück.

$translator->`trname` ($trname)

Sendet die Anfrage `trname:`*`$trname`* an den Übersetzerprozess und gibt dessen Antwort zurück. Der Übersetzerprozess soll ab jetzt den Übersetzer mit der Kennung *`$trname`* verwenden.

$translator->`translator_name` ()

Sendet die Anfrage `translator_name` an den Übersetzerprozess. Wenn dessen Antwort mit der Zeichenfolge `OK` beginnt, werden diese Zeichenfolge und mögliche weitere Leerzeichen am Anfang der Antwort durch `pipe_` ersetzt. Die resultierende Zeichenfolge wird als Ergebnis zurückgegeben. Wenn die Anwort des Übersetzerprosesses nicht mit `OK` beginnt, wird `pipe_error` zurückgegeben.

$translator->`stop` ()

Sendet die Anfrage `stop` an den Übersetzerprozess und gibt dessen Antwort zurück. Der Übersetzerprozess soll sich beenden.

Quelltext

[Beschreibung]

```
# symlink KLEIDER/web/src/localization/Translate/Pipe.pm
# Übersetzer in separatem Prozess über Pipes ansprechen
# 2015-08-09 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::Pipe ;
use parent Herbaer::Translate::Base ;
use Cwd qw(getcwd) ;
use File::Spec::Functions qw(catfile) ;
use utf8;


sub new {
   my ($class
       , $base
       , $trname
       , $debug
      ) = @_;
   $base ||= getcwd();
   if ($debug) {
      print STDERR "Herbaer::Translate::Pipe::new\n";
   }
   my $self = {
      "dbg"  => $debug,
      "req"  => catfile ($base, "request"),
      "resp" => catfile ($base, "response"),
   };
   trname ($self, $trname) if $trname;
   return bless ($self, $class);
}


sub _send_request {
   my ($self, $request) = @_;
   my $debug = $self -> {"dbg"};
   my $reqp  = $self -> {"req"};
   my $respp = $self -> {"resp"};
   print STDERR "_send_request $request\n" if $debug;
   if (! -p $reqp) {
      # Der Server-Prozess braucht die Zeit, die Pipes anzulegen.
      sleep (1);
      if (! -p $reqp) {
         print STDERR "$reqp ist keine Pipe\n" if $debug;
         return;
      }
   }
   if (! -p $respp) {
      print STDERR "$respp ist keine Pipe\n" if $debug;
      return;
   }
   my $reqh;   # request handle
   if (!open ($reqh, ">:encoding(utf-8)", $reqp)) {
      print STDERR "Kann request-Pipe $reqp nicht öffnen: $!\n" if $debug;
      return;
   }
   my $resph; # response handle
   if (!open ($resph, "<:encoding(utf-8)", $respp)) {
      print STDERR "Kann response-Pipe $respp nicht öffnen: $!\n" if $debug;
      close $reqh;
      return;
   }
   print $reqh $request;
   close $reqh;
   my $response = <$resph>;
   close $resph;
   $response =~ s/\n$// ;
   return $response;
} # _send_request


# "\" wird durch "\\",
# ":" wird durch "\c",
# Zeilenende durch "\n" ersetzt
sub _encode {
   my $t = shift;
   $t =~ s/\\/\\\\/g;
   $t =~ s/\n/\\n/g;
   $t =~ s/:/\\c/g;
   return $t;
} # _encode


# macht _encode rückgängig
sub _decode {
   my $t = shift || "";
   $t =~ s/\\(.)/$1 eq "n" ? "\n" : $1 eq "c" ? ":" : $1/ge;
   return $t;
} # _decode
```

```
sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Pipe::translate\n";
   }
   # Parameter prüfen
   $text || return;
   $srl  || return;
   $tgl  || return;
   $srl =~ s/-.*// ;
   $srl = lc ($srl);
   $tgl =~ s/-.*// ;
   $tgl = lc ($tgl);
   return _decode ($self -> _send_request (join (":",
      "translate", _encode ($text), $srl, $tgl
   )));
} # translate


sub learn {
   my ($self, $text, $srl, $tgl, $tt) = @_ ;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Pipe::learn\n";
   }
   return $self -> _send_request (join (":",
      "learn", _encode ($text), $srl, $tgl, _encode ($tt)
   ));
} # learn


sub finish {
   my $self = shift;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Pipe::finish\n";
   }
   return $self -> _send_request ("finish");
} # finish


# setzt die Übersetzer-Kennung
sub trname {
   my ($self, $trname) = @_;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Pipe::trname $trname\n";
   }
   return _send_request ($self, "trname:$trname");
} # trname


# liefert den Übersetzer-Namen
sub translator_name {
   my $self = shift;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Pipe::translator_name\n";
   }
   my $r = _send_request ($self, "translator_name") || "";
   if ( $r =~ s/^OK\s+// ) {
      return "pipe_$r";
   }
   else {
      return "pipe_error";
   }
} # translator_name


sub stop {
   my $self = shift;
   if ($self -> {"dbg"}) {
      print STDERR "Herbaer::Translate::Pipe::stop\n";
   }
   return $self -> _send_request ("stop");
} # stop


1;
```

Herbaer::Translate::SystemItemFilter

[Quelltext]

Beschreibung

Manchmal kommt die Anforderung, einen Dateisystempfad, einen URL oder eine E-Mail-Adresse zu übersetzen. Dieses Modul erkennt Dateipfade, URL und E-Mail-Adressen und liefert sie unverändert als „Übersetzung” zurück.

Funktionen

`$translator = Herbaer::Translate::SystemItemFilter->new (`*`$trname`*`, `*`$debug`*`)`

Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise von `Herbaer::Translate::new` aufgerufen (s. `Translate.pm`).

`$trname` ist die Kennung des „beauftragten” Übersetzers.

Ein logisch wahrer Wert des optionalen Parameters *`$debug`* führt zu Meldungen nach STDERR.

`$translator->translate ($text, $quellsprache, $zielsprache)`

Dateisystem-Pfade, URL und E-Mail-Adressen, Dateinamen und Namen von Perl-Modulen werden (hoffentlich) erkannt und unverändert als „Übersetzung” zurückgegeben. Sonst wird die Methode `translate` des beauftragten Übersetzers aufgerufen.

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

Ruft die Methode `learn` des beauftragten Übersetzers auf.

$translator->`translator_name` ()

Ergibt `sif_`*`TRANSLATOR_NAME`*. Der Platzhalter *`TRANSLATOR_NAME`* steht für das Ergebnis der Methode `translator_name` des beauftragten Übersetzers. Das Ergebnis ist `sif_error`, wenn der beauftragte Übersetzer nicht verfügbar ist.

$translator->`finish` ()

Ruft die Methode `finish` des beauftragten Übersetzers auf.

Quelltext

[Beschreibung]

```
# URIs und Dateipfade herausfiltern
# 2015-08-14 Herbert Schiemann <h.schieman@herbaer.de>
# GPL Version 2 oder neuer

package Herbaer::Translate::SystemItemFilter ;

use parent Herbaer::Translate::Base ;
use Herbaer::Translate ;
use utf8;

BEGIN {
   binmode (STDERR, ":utf8");
}

sub new {
   my ($class, $trname, $debug) = @_;
   my $self = {
      "dbg"    => $debug,
      "trname" => $trname,
      "trans"  => undef,
   } ;
   return bless ($self, $class);
} # new

# lädt den Übersetzer
sub _translator {
   my $self = shift;
   my $trans = $self -> {"trans"};
   if (! $trans) {
      $trans = new Herbaer::Translate ($self -> {"trname"});
      if (! $trans) {
         print STDERR "Kann Übersetzer \"", $self -> {"trname"}, "\" nicht laden.\n"
            if $self -> {"dbg"};
         return undef;
      }
      $self -> {"trans"} = $trans;
   }
   return $trans;
}

sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::SystemItemFilter::translate\n";
   }
   # Web-Adressen und Dateinamen werden nicht übersetzt
   if ($text =~ /^\S*$/) {
      return $text if
            $text =~ /^(?:http:|https:|ftp:|mailto:)/         # Web-Adresse
         || $text =~ /\.[a-z]{2,5}$/                          # Dateiname mit Suffix
         || $text =~ /^\.*(?:::)?(?:[A-Za-z]+::)+[A-Za-z]+$/  # Perl-Modul
         || $text =~ /^[^@]{3,}@[^.@]{3,}\.[^@]{3,}$/         # E-Mail-Adresse
         || $text =~ /^.$/                                    # einzelnes Zeichen
         ;
      return $text if
            $text =~ /^[a-z0-9A-Z_.\/]+$/                     # Dateipfade
         && $text =~ /\//
         ;
   }
   my $trans = $self -> _translator ();
   return $trans -> translate ($text, $srl, $tgl) if $trans;
} # translate

sub learn {
   my ($self, $text, $srl, $tgl, $tt) = @_ ;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::SystemItemFilter::learn\n";
   }
   my $trans = $self -> _translator ();
   $learn -> learn ($text, $srl, $tgl, $tt) if $trans;
} # learn
```

```
# speichert die gelernten Daten
sub finish {
   my $self = shift;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::SystemItemFilter::finish\n";
   }
   my $trans = $self -> {"trans"};
   $trans -> finish () if $trans;
} # finish


# Name des zuletzt aktiven Uebersetzers
sub translator_name {
   my $self = shift;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::SystemItemFilter::translator_name\n";
   }
   my $trans = $self -> {"trans"};
   if ($trans) {
      return "sif_" . $trans -> translator_name ();
   }
   else {
      return "sif_error";
   }
} # translator_name


1;
# file KLEIDER/web/src/localization/SystemItemFilter.pm
```

Herbaer::Translate::WordReplacer

[Quelltext]

Zweck

Es kann (deutsche) Wörter geben, die der Übersetzer nicht versteht. Das Wort erscheint dann mehr oder weniger unverändert in der Zielsprache. Möglicherweise ist die Großschreibung geändert, oder anstelle von Umlauten oder Buchstaben mit Akzenten erscheinen die Grund-Buchstaben. Dieses Modul versucht, das Wort durch ein anderes Wort zu ersetzen, ohne den Sinn wesentlich zu ändern, damit der Übersetzer den Text übersetzen kann. Die wesentliche Arbeit dieses Moduls liegt also in der Funktion `translate`.

Dieses Modul verarbeitet Listen von Paaren von regulären Ausdrücken und Ersetzungsausdrücken. Wenn ein ein regulärer Ausdruck zu einem übersetzten Text und zum originalen Text in der Quellsprache passt, dann werden im Text der Quellsprache die Teile, die der reguläre Ausdruck beschreibt, durch den Ersetzungsausdruck ersetzt. Der Ersetzungsausdruck kann Platzhalter der Form `${n}` für das n-te Teilmuster im Text in der Quellsprache enthalten. Der modifizierte Text wird neu übersetzt. Wenn der reguläre Ausdruck noch immer zu dem neu übersetzten Text passt, werden die früheren Versionen des Quelltextes und der Übersetzung wiederhergestellt. Die Verarbeitung eines Listeneintrags ergibt eines von drei möglichen Ergebnistypen:

- Der reguläre Ausdruck passt nicht zur Übersetzung.
- Der reguläre Ausdruck passt zur Übersetzung. Die passenden Stellen im Quelltext werden ersetzt. Die Übersetzung des geänderten Quelltextes passt nicht mehr zum regulären Ausdruck.
- Der reguläre Ausdruck passt zur Übersetzung, passt aber auch zur Übersetzung des geänderten Quelltextes. Die Änderung des Quelltextes wird zurückgenommen und die frühere Übersetzung wieder gewählt.

In allen drei Fällen wird anschließend der nächste Listeneintrag verarbeitet, bis die Liste abgearbeitet ist.

Eine besondere Rolle hat der Text `/RETURN` als Ersetzungstext. Wenn der reguläre Ausdruck nicht zur Übersetzung passt, wird die Übersetzung als Ergebnis der Funktion `translate` zurückgegeben. Die weitere Listeneinträge werden nicht mehr verarbeitet. Wenn der reguläre Ausdruck zur Übersetzung passt, wird die Verarbeitung mit dem nächsten Listeneintrag fortgesetzt.

Funktionen

`$translator = Herbaer::Translate::WordReplacer->new ($directory, $translator_name, $debug)`

Ergibt ein neues Übersetzer-Objekt. Diese Funktion wird normalerweise nicht direkt aufgerufen, sondern von `Herbaer::Translate::new` (s. `Translate.pm`).

`$directory` ist der Pfad des Verzeichnisses, in dem die Dateien mit den regulären Ausdrücken und Ersetzungstexten gesucht werden. Für die Übersetzung eines Textes aus der Sprache mit der Kennung *`SRCLANG`* in die Sprache mit der Kennung *`TARGETLANG`* werden zunächst die Einträge in der Datei *`SRCLANG_TARGETLANG`*, dann die Einträge in der Datei *`SRCLANG`*, gelesen.

Die Dateien im Verzeichnis *`$directory`* sind einfache Textdateien, die zeilenweise gelesen werden. Zeilen, die mit dem Zeichen # beginnen oder nur Leerzeichen enthalten, sind Kommentare. Andere Zeilen bestehen aus beliebig vielen Leerzeichen am Zeilenanfang, dem regulären Ausdruck aus Nicht-Leerzeichen, beliebig vielen Leerzeichen und dem Ersetzungstext, gefolgt von beliebig vielen Leerzeichen und dem Zeilenende. Ein Beispiel ist die Datei `WordReplacer.beispiel`.

`$translator_name` ist die Kennung des „beauftragten” Übersetzers, s. `Herbaer::Translate` (`Translate.pm`).

Ein logisch wahrer Wert des optionalen Parameters *$debug* führt zu Meldungen nach STDERR.

`$translator->translate ($text, $quellsprache, $zielsprache)`

Ergibt die Übersetzung des Textes *$text* aus der Sprache *$quellsprache* in die Sprache *$zielsprache* nach der Ersetzung oder dem Versuch der Ersetzung von Teilen des Quelltextes abhängig von den Sprachen gemäß den Ersetzungsdateien (s. `new`, Parameter *$directory*).

`$translator->learn ($text, $quellsprache, $zielsprache, $uebersetzung)`

Ruft die Methode `learn` des „beauftragten” Übersetzers auf.

$translator->`finish` ()

Ruft die Methode `finish` des „beauftragten” Übersetzers auf.

$translator->`translator_name` ()

Ergibt `wr_`*TRANSTOR_NAME*. *TRANSTOR_NAME* steht für das Ergebnis der Methode `translator_name` des „beauftragten” Übersetzers.

Quelltext

[Beschreibung]

```
# Texteile ersetzen zur Erleichterung der Übersetzung
# 2015-08-03 Herbert Schiemann, <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package Herbaer::Translate::WordReplacer ;


use parent Herbaer::Translate::Base ;
use File::Spec::Functions qw (catfile) ;
use Herbaer::Translate ;
use utf8;


sub new {
   my ($class
       , $dir
       , $trname
       , $debug
      ) = @_;
   use File::Spec::Functions qw(rel2abs) ;
   $debug ||= 0;
   if ($debug) {
      print STDERR "Herbaer::Translate::WordReplacer::new\n";
   }
   $dir ||= "replacements";
   $dir = rel2abs($dir);
   if (! -d $dir) {
      print STDERR
         "  $dir ist kein Verzeichnis,\n",
         "  Wörter werden nicht ersetzt\n";
      $dir = undef;
   }
   my $self = {
      "dbg"    => $debug,
      "dir"    => $dir,      # Verzeichnis der Wortersetzungstabellen
      "repl"   => {},        # lang, global -> [ [pattern, replacement | /RETURN ],]
      "trname" => $trname,
   } ;
   if ($dbd && $debug >= 1) {
      print STDERR "  Verzeichnis der Ersetzungen: $dir\n" ;
   }
   return bless ($self, $class);
} # new


# lädt den Übersetzer
sub _translator {
   my $self = shift;
   my $trans = $self -> {"trans"};
   if (! $trans) {
      $trans = new Herbaer::Translate ($self -> {"trname"});
      if (! $trans) {
         print STDERR "Kann Übersetzer \"", $self -> {"trname"}, "\" nicht laden.\n"
            if $self -> {"dbg"};
         return undef;
      }
      $self -> {"trans"} = $trans;
   }
   return $trans;
}
```

```
# liest eine Ersetzungstabelle
sub _getdata {
   my ($self, $key) = @_ ;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::WordReplacer::_getdata (\"$key\")\n";
   }
   my $path = catfile ($self -> {"dir"}, $key) ;
   my $h;
   my $list = [];
   my $line;
   my $re;
   open ($h, "<:encoding(utf-8)", $path) or return ;
   while (defined ($line = <$h>)) {
      next if $line =~ /^#/ ;
      $line =~ s/\s*$// ;
      if ( $line =~ /\s*(\S+)\s+(.*)/ ) {
         $re = $1;
         print STDERR "REGEX $re\n" if $debug;
         push (@$list, [qr($re), $2]);
      }
   }
   close $h;
   if ($debug) {
      print STDERR "Herbaer::Translate::WordReplacer data\n";
      for $h (@$list) {
         print STDERR " ", $h->[0], " --> ", $h->[1], "\n";
      }
   }
   $self -> {"repl"} -> {$key} = $list;
} # _getdata


sub translate {
   my ($self, $text, $srl, $tgl) = @_ ;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::WordReplacer::translate\n";
   }
   # Parameter können sinnvoller geprüft werden.
   $text || return;
   $srl  || return;
   $tgl  || return;

   my $trans = $self -> _translator ();
   return "" unless $trans;

   $srl =~ s/-.*// ;
   $tgl =~ s/-.*// ;
   my $tt;
   $tt = $trans -> translate ($text, $srl, $tgl);
   return "" unless $tt;

   my $key;    # Schlüssel zur Ersetzungsliste
   my $list;   # Liste der Ersetzungen
   my $stext;  # Text-Sicherung
   my $stt;    # Ersetzungstest-Sicherung
   my $li;     # Listeneintrag
   my $re;     # Regex
   my $repl;   # Ersetzungsausdruck
   for $key ("${srl}_${tgl}", $srl) {
      my $list = $self -> {"repl"} -> {$key}; # Liste der sprachabhängigen Ersetzungen
      if (!$list) {
         $self -> _getdata ($key);
         $list = $self -> {"repl"} -> {$key};
      }
      for $li (@$list) {
         $re   = $li -> [0];
         $repl = $li -> [1];
         if ($debug) {
            print STDERR
               "WordReplacer RE $re\n",
               "WordReplacer TT $tt\n";
         }
         if ($tt =~ $re) {
            next if $repl eq "/RETURN" ;
            $stext = $text;
            $stt   = $tt;
            if ($debug) {
               print STDERR
                  "WordReplacer TRY TEXT REPLACE $text\n",
                  "WordReplacer EVAL " . "\$text =~ s/$re/\"$repl\"/ge" . "\n";
            }
            if (eval ("\$text =~ s/$re/\"$repl\"/ge")) {
               $tt = $trans -> translate ($text, $srl, $tgl);
               if ($debug) {
                  print STDERR
                     "WordReplacer TEXT REPLACE $text\n",
                     "WordReplacer NEW TT $tt\n";
               }
               if ( $tt =~ $re ) {
                  # Die Ersetzung hat nicht genutzt
                  print STDERR "WordReplacer ROLLBACK\n" if $debug;
                  $tt   = $stt;
                  $text = $stext;
               }
```

```
            }
         }
         else {
            print STDERR "WordReplacer NO MATCH $repl\n" if $debug;
            last if $repl eq "/RETURN" ;
         }
      }
   }
   return $tt;
} # translate


sub learn {
   my ($self, $text, $srl, $tgl, $tt) = @_ ;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::WordReplacer::learn\n";
   }
   my $trans = $self -> _translator ();
   $learn -> translate ($text, $srl, $tgl, $tt) if $trans;
} # learn


sub finish {
   my $self = shift;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::WordReplacer::finish\n";
   }
   my $trans = $self -> {"trans"};
   $trans -> finish () if $trans;
} # finish


# Name des zuletzt aktiven Uebersetzers
sub translator_name {
   my $self = shift;
   my $debug = $self -> {"dbg"};
   if ($debug) {
      print STDERR "Herbaer::Translate::WordReplacer::translator_name\n";
   }
   my $trans = $self -> {"trans"};
   if ($trans) {
      return "wr_" . $trans -> translator_name ();
   }
   else {
      return "wr_error";
   }
} # translator_name


1;
```

Datei WordReplacer.beispiel

```
# file KLEIDER/web/src/localization/WordReplacer.beispiel
# Beispiel einer Wortersetzungsdatei für deutsche Quelltexte
# zum Perl-Modul Herbaer::Translate::WordReplacer
# Originaldatei im Verzeichnis KLEIDER/web/transdb/replace
# Originaldateiname z.B. "de" für Übersetzungen aus der deutschen Sprache
# in irgendeine andere Sprache
# oder "de_zh" für Übersetzungen vom Deutschen ins Chinesische.


# "Kleidchen" wird durch "Kleid" ersetzt
Kleidchen          Kleid


# "ungefährlich" wird durch "nicht gefährlich" ersetzt
un(ge[a-zäöüß]*)   nicht ${1}


# keine weiteren Ersetzungen,
# wenn die Übersetzung nicht wenigstens fünf aufeinanderfolgende lateinische Buchstaben
# oder kleine deutsche Sonderbuchstaben enthält
[a-zäüößA-Z]{5}                 /RETURN


# "Tüllkleid" wird durch "Tüll-kleid" ersetzt
Tüll([[a-zäüößA-Z]{3,15})       Tüll-${1}
```

localize

[Quelltext]

Übersicht

`localize --help|--version`

`localize [ --verbose … --no_verbose][ --overwrite --no_overwrite]`
`[ --story --no_story][ --zipfiles --no_zipfiles]`
`[ --srclang` *`SRCLANG`* `][ --trlang` *`TRLANG`* `]`
`[ --basedir` *`BASEDIR`* `][ --docroot` *`DOCROOT`* `][ --locdir` *`LOCDIR`* `]`
`[ --trtoutdir` *`TRTOUTDIR`* `][ --trtindir` *`TRTINDIR`* `][ --srcdir` *`SRCDIR`* `]`
`[ --stylesrc` *`STYLESRC`* `][ --trname` *`TRNAME`* `]`
`[ --storyptn` *`STORYPTN`* `][ --ftpmode` *`FTPMODE`* `][ --runonce` *`RUNONCE`* `]`
`[ --reminter --skl --xliff --mtt --trt`
`--mtrans --merge --finish --upload --once`
`|--no_skl --no_xliff --no_mtt --no_trt --no_mtrans`
`--no_merge --no_finish --no_upload --no_once ]`
`FILE` …

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus.

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Meldungen über den Programmablauf werden nach STDOUT ausgegeben, Programme werden mit dem Argument `--verbose` aufgerufen.

`--no_verbose`

Diese Option hebt die Wirkung der Option `--verbose` auf.

`--overwrite`

Existierende Dateien werden überschrieben.

`--no_overwrite`

Existierende Dateien werden nicht überschrieben.

`--story`

Positionsargumente *`FILE`* werden als Kennungen von Bildergeschichten behandelt. Wenn kein Positionsargument angegeben ist, werden alle Bildergeschichten verarbeitet, s. *`STORYPTN`*.

`--no_story`

Diese Option hebt die Wirkung der Option `--story` auf. Alle Optionen werden ausgewertet, bevor die Positionsargumente verarbeitet werden. Die letzte der Optionen `--story` oder `--no_story` gilt daher für alle Positionsargumente *`FILE`*.

`--zipfiles`

Die übersetzten Dateien werden gzip-komprimiert.

`--no_zipfiles`

Diese Option hebt die Wirkung der Option `--zipfiles` auf. Die übersetzten Dateien werden nicht nicht gzip-komprimiert.

`--srclang` *`SRCLANG`*

Kennung der Ausgangssprache. Die Kennung bezeichnet als Dateinamenssuffix die Dateien im Verzeichnis *`DOCROOT`*, die zu übersetzen sind. Sie wird auch als Default-Quellsprache an das Übersetzungsprogramm `mttext.pl` übergeben.

`--basedir` *`BASEDIR`*

Der Wert *`BASEDIR`* beeinflusst die Voreinstellung anderer Optionen. Er wird nicht direkt verwendet.

`--docroot` *`DOCROOT`*

Das Dokument-Verzeichnis des Webservers.

`--locdir` *`LOCDIR`*

Im Verzeichnis *`LOCDIR`* werden alle Zwischendateien im Zusammenhang mit der Übersetzung gespeichert. Die Zwischendateien zu einer Quelldatei im Verzeichnis *`DOCROOT`* liegen unter dem gleichen Unterpfad, aber einem anderen Suffix im Verzeichnis *`LOCDIR`*.

`*.`*`SRCLANG`*`.skl`

Die Gerüst-Datei

`*.`*`SRCLANG`*`.xlf`

XLIFF-Datei mit den zu übersetzenden Texten in der Quellsprache.

`*.`*`SRCLANG`*`.mtt`

Übersetzungstext-Datei mit den zu übersetzenden Texten in der Quellsprache.

`*.`*`TGTLANG`*`.rtr`

XML-Datei mit den übersetzten Texten

`*.`*`TGTLANG`*`.xlf`

XLIFF-Datei mit den übersetzten Texten

`--trtoutdir` *`TRTOUTDIR`*

Im Verzeichnis *`TRTOUTDIR`* werden Übersetzungstextdateien zur Übersetzung durch einen menschlichen Übersetzer gespeichert. Im relativen Dateipfad einer XLIFF-Datei im Verzeichnis *`LOCDIR`* wird die Dateinamensendung *`SRCLANG`*`.xlf` durch *`TGTLANG`*`.txt` ersetzt.

`--trtindir` *`TRTINDIR`*

Im Verzeichnis *`TRTINDIR`* werden übersetzte Dateien gesucht (Textdateien `*.`*`TGTLANG`*`.txt` oder XML-Dateien `*.`*`TGTLANG`*`.rtr`). Übersetzte Dateien im Verzeichnis *`TRTINDIR`* haben Vorrang vor Dateien im Verzeichnis *`LOCDIR`*.

`--srcdir` *SRCDIR*

Verzeichnis der XSLT-Dateien und Perl-Skripte, die dieses Skript benutzt.

`--stylesrc` *STYLESRC*

Verzeichnis der Quelldateien zum Stil. Der Pfad der Quelldatei der Stil-Lokalisierung wird als Parameter den Stylesheets zur Nachbereitung der Überseztung übergeben.

`--trname` *TRNAME*

Bezeichnung der Übersetzungsmaschine. Sie wird als Parameter an `mttext.pl` übergeben.

Wenn *TRNAME* die Zeichenfolge `pipe` enthält, wird zu Anfang des Übersetzungen das Diensprogramm `pipe_srv.pl` gestartet und am Ende mit dem Programm `pipe_srv_stop.pl` beendet.

`--trlang` *TRLANG*

TRLANG ist Liste der Kennungen der Zielsprachen. Trennzeichen ist das Leerzeichen.

In der Voreinstellung ist *TRLANG* die Liste der Kennungen *TGTLANG* aller Sprachen, für die eine Stil-Lokalisierungsdatei *DOCROOT*`/local/local.xml.`*TGTLANG*`.` existiert.

`--storyptn` *STORYPTN*

STORYPTN ist das Muster des relativen Dateipfades einer Bildergeschichte in der Quellsprache. Es enthält die Platzhalter `${sid}` und `${srclang}` und wird in Verbindung mit der Option `--story` benutzt.

`--ftpmode` *FTPMODE*

FTPMODE beeinflusst den Updload der übersetzten Dateien. Die möglichen Werte sind:

`put`

Existierende Dateien auf dem Server werden ersetzt.

`putnewer`

Existierende Dateien auf dem Server werden ersetzt, wenn die Dateien im lokalen Verzeichnis *DOCROOT* neuer sind.

`putnotex`

Existierende Dateien auf dem Server werden nicht ersetzt, nur Dateien im lokalen Verzeichnis *DOCROOT*, die auf dem Server nicht existieren, werden hochgeladen.

`--runonce` *RUNONCE*

RUNONCE ist der Pfad eines Programms, das die Option `--once` aufruft. Ein relativer Pfad bezieht sich auf das Verzeichnis *SRCDIR*. Dem Programm werden die Optionen `--verbose` und `--[no_]overwrite`, die Argumente *TRLANG*, *SRCLANG* und *SRCDIR* sowie der Dateipfad einer zu übersetzenden Datei als Positionsargument übergeben.

Das Bash-Skript `runonce` ist ein Beispiel für ein solches Programm.

`--reminter`

Die Zwischendateien zu einer zu übersetzenden Datei *DOCROOT*`/*.`*SRCLANG*`.` werden gelöscht:

LOCDIR`/*.`*SRCLANG*`.skl`

LOCDIR/.SRCLANG*.xlf
LOCDIR/.SRCLANG*.mtt
LOCDIR/.TGTLANG*.rtr
LOCDIR/.TGTLANG*.xlf

TGTLANG steht hier für die Kennung einer Zielsprache.

--skl

Erzeugt zu einer zu übersetzenden Datei *DOCROOT/*.SRCLANG*. die Gerüst-Datei *LOCDIR/*.SRCLANG*.skl. Zu diesem Schritt gehören

- eine vorbereitende Transformation *XSLNAME*_pre.xslt abhängig vom Stylesheet (optional),
- eine vorbereitende Transformation *TYPE*_pre.xslt abhängig vom XML-Namensraum (optional),
- die Transformatioen *TYPE*clfy.xslt,
- die Transformatioen skeleton_2.xslt.

--xliff

Erzeugt aus einer Gerüst-Datei *LOCDIR/*.SRCLANG*.skl eine XLIFF-Datei *LOCDIR/*.SRCLANG*.xlf. Das schaffen die beiden Transformationen skeleton_xliff.xslt und xliff_clean.xslt.

--mtt

Erzeugt aus einer XLIFF-Datei *LOCDIR/*.SRCLANG*.xlf eine Übersetzungstext-Datei *LOCDIR/*.SRCLANG*.mtt zur maschinellen Übersetzung.

--trt

Erzeugt aus einer XLIFF-Datei *LOCDIR/*.SRCLANG*.xlf für jede Zielsprache *TGTLANG* in *TRLANG* eine Übersetzungstext-Datei *TRTOUTDIR/*.TGTLANG*.txt zur maschinellen Übersetzung.

In der Voreinstellung bleibt diese Aktion aus.

--mtrans

Die Dateien werden maschinell übersetzt. Wenn zu einer XLIFF-Datei *.xlf eine Übersetzungstextdatei *.mtt existiert, wird die Übersetzungstextdatei weiter verarbeitet, sonst wird auf die XLIFF-Datei die Transformation *SRCDIR*/xliff_txt.xslt angewandt. Eine Übersetzungstext-Datei *LOCDIR/*.SRCLANG*.mtt wird automatisch übersetzt. Das Perl-Programm mttext.pl übersetzt die Texte, das Programm resstruct.pl versucht, verschachtelte Strukturen wiederzuerkennen. Das Ergebnis ist eine XML-Datei *LOCDIR/*.TGTLANG*.rtr für jede Zielsprache *TGTLANG* in *TRLANG*.

--merge

Diese Aktion erzeugt übersetzte XLIFF-Dateien *LOCDIR/*.TGTLANG*.xlf.

Wenn die Datei *TRTINDIR/*.TGTLANG*.rtr existiert, fügt die Transformation xliff_merge_rtr.xslt die XLIFF-Datei *LOCDIR/*.SRCLANG*.xlf und diese Datei *TRTINDIR/*.TGTLANG*.rtr zusammen.

Wenn andernfalls die übersetzte Textdatei *TRTINDIR/*.TGTLANG*.txt existiert, erzeugt das Programm resstruct.pl daraus die Datei *TRTINDIR/*.TGTLANG*.rtr, die mit der XLIFF-Datei *LOCDIR/*.SRCLANG*.xlf zur übersetzten XLIFF-Datei *LOCDIR/*.TGTLANG*.xlf zusammengefügt wird.

Andernfalls fügt die Transformation xliff_merge_rtr.xslt die XLIFF-Datei mit der maschinell übersetzten Datei *LOCDIR/*.TGTLANG*.rtr zusammen.

`--finish`

Im letzten Schritt der automatischen Übersetzung führt die Transformation `skeleton_merge_xliff.xslt` die Gerüst-Datei *`LOCDIR/*.SRCLANG`*`.skl` und die übersetzten XLIFF-Dateien *`LOCDIR/*.TGTLANG`*`.xlf` zu den übersetzten Dateien *`DOCROOT/*.TGTLANG.`* zusammen. Auf die übersetzten Dateien werden optional die Transformationen *`TYPE`*`_post.xslt` abhängig vom XML-Namensraum und *`XSLNAME`*`_post.xslt` abhängig vom Stylesheet angewandt, und abschließend entfernt das Programm `rmxmlns.pl` überflüssige XML-Namensraum-Präfixe.

Fehlende Zwischendateien aus den vorhergehenden Verarbeitungsschritten werden bei Bedarf erzeugt.

`--upload`

Die übersetzten Dateien werden auf den Webserver hochgeladen. Dazu dient das Skript `ftp.pl`. Das Argument *`FTPMODE`* steuert, ob existierende Dateien ersetzt werden.

`--once`

Diese Aktionsoption führt das Programm *`RUNONCE`* für jede zu übersetzende Datei *`DOCROOT/*.SRCLANG.`* aus.

Die Aktion ist für einmalige Nachträge gedacht und bleibt in der Voreinstellung aus.

`--no_*`

Wenn keine der „Aktionsoptionen## `--skl`, `--xliff`, `--mtt`, `--trt`, `--mtrans`, `--merge`, `--finish`, `--upload`, `--once` genutzt wird, können die Optionen mit dem Präfix `no_` (`--no_skl`, `--no_xliff`, `--no_mtt`, `--no_trt`, `--no_mtrans`, `--no_merge`, `--no_finish`, `--no_upload`, `--no_once`) genutzt werden. Diese schließen die jeweilige Aktion aus. Alle nicht ausgeschlossenen Aktionen werden ausgeführt.

`FILE`

Abhängig von der Option `--story` wird das Positionsargument *`FILE`* unterschiedlich interpretiert.

Zusammen mit der Option `--story` ist *`FILE`* die Kennung einer Bildergeschichte. Das Programm bearbeitet die Bildergeschichten mit den angegebenen Kennungen. Wenn die Option `--story` ohne ein Positionsargument *`FILE`* benutzt wird, bearbeitet das Programm alle Bildergeschichten.

Ohne die Option `--story` ist *`FILE`* der relative Pfad einer zu übersetzenden Datei unterhalb des Verzeichnisses *`DOCROOT`* ohne das Suffix `.xhtml.`*`SRCLANG.`* oder das Suffix *`.SRCLANG.`* oder der relative Pfad eines Unterverzeichnisses von *`DOCROOT`*. Das Programm bearbeitet alle aufgeführten zu übersetzenden Dateien und alle zu übersetzenden Dateien unterhalb eines aufgeführten Unterverzeichnisses. Wenn weder die Option `--story` noch ein Positionsargument *`FILE`* benutzt wird, bearbeitet das Programm alle zu übersetzenden Dateien unterhalb von *`DOCROOT`*.

Beschreibung

Das Skript `localize` unterstützt der Erstellung weiterer Lokalisierungen der Website `http://kleider.herbaer.de`.

Existierende Dateien werden nur dann überschrieben, wenn die Option *`--overwrite`* benutzt wird. Wenn eine Datei, die für eine Aktion nötig ist, nicht existiert, dann wird diese Datei nach Möglichkeit erzeugt.

Quelltext

[Beschreibung]

```
#!/bin/bash
# -*- coding:utf-8 -*-
# Lokalisierung der Website http://kleider.herbaer.de
# 2015-07-02 Herbert Schiemann <h.schiemann@herbaer.de>


# Zunächst Funktionen,
# die an die konkrete Anwendung anzupassen sind.


# Zähler, Variable, Aktionen
declare_vars ()
{
   # Ein Leerzeichen als Wert bedeutet, dass Positionsargumente verarbeitet werden
   _argv=" ";

   # Suchpfad für rc-Dateien, : - getrennte Liste von Verzeichnispfaden
   # Falls leer, wird die Option --rc nicht speziell behandelt
   g_configpath= ;

   # Zähler
   g_counters="  \
      verbose    \
      overwrite  \
      story      \
      zipfiles   "

   # Variable
   g_variables=" \
      srclang    \
      basedir    \
      docroot    \
      locdir     \
      trtoutdir  \
      trtindir   \
      srcdir     \
      stylesrc   \
      trname     \
      trlang     \
      storyptn   \
      ftpmode    \
      runonce    ";

   # Aktionen
   g_actions="   \
      reminter   \
      skl        \
      xliff      \
      mtt        \
      trt        \
      mtrans     \
      merge      \
      finish     \
      upload     \
      once       ";
} # declare_vars


# setzt Vorgabe-Werte
set_defaults ()
{
   local b=$(realpath $0);
   b=${b%/src/localization/localize};
   [[ -n "$no_trt"        ]] || no_trt=1 ;
   [[ -n "$no_once"       ]] || no_once=1 ;
   [[ -n "$verbose"       ]] || verbose=1 ;
   [[ -n "$overwrite"     ]] || overwrite=0 ;
   [[ -n "$story"         ]] || story=0 ;
   [[ -n "$zipfiles"      ]] || zipfiles=1 ;
   [[ -n "$srclang"       ]] || srclang="de" ;
   [[ -n "$basedir"       ]] || basedir="$b" ;
   [[ -n "$docroot"       ]] || docroot="$basedir/docroot" ;
   [[ -n "$locdir"        ]] || locdir="$basedir/local" ;
   [[ -n "$trtoutdir"     ]] || trtoutdir="$locdir/trtout" ;
   [[ -n "$trtindir"      ]] || trtindir="$locdir/trtin" ;
   [[ -n "$srcdir"        ]] || srcdir="$basedir/src/localization" ;
   [[ -n "$stylesrc"      ]] || stylesrc="$basedir/src/style" ;
   # pipe_nscnr_google normale Anwendung
   # pipe_nsbnr_google gepuffertes Lernen
   # pipe_nslnr_google hartes Lernen
   [[ -n "$trname"        ]] || trname="pipe_nscnr_google" ;
   [[ -n "$trlang"        ]] || set_default_trlang ;
   [[ -n "$trlang"        ]] || trlang=" \
     eu ca zh hr cs da nl en eo et fi fr gl el hu is ga it ja ko la lv lt  ms mt \
     no pl pt ro sk sl es sv th vi ";
   [[ -n "$storyptn"      ]] || storyptn="s\${sid}/story.xml.\${srclang}.";
   [[ "$ftpmode" == "put" ]] || [[ "$ftpmode" == "putnotex" ]] || ftpmode="putnewer" ;
   [[ -n "$runonce"       ]] || runonce="$srcdir/runonce";
   [[ $runonce =~ ^/      ]] || runonce="$srcdir/$runonce";
```

```
# Basque        eu
# Catalan       ca
# Chinese Simplified    zh
# Croatian      hr
# Czech         cs
# Danish        da
# Dutch         nl
# English       en
# Esperanto     eo
# Estonian      et
# Finnish       fi
# French        fr
# Galician      gl
# Greek         el
# Hungarian     hu
# Icelandic     is
# Irish         ga
# Italian       it
# Japanese      ja
# Korean        ko
# Latin         la
# Latvian       lv
# Lithuanian    lt
# Malay         ms
# Maltese       mt
# Norwegian     no
# Polish        pl
# Portuguese    pt
# Romanian      ro
# Slovak        sk
# Slovenian     sl
# Spanish       es
# Swedish       sv
# Thai  th
# Vietnamese    vi
} # set_defaults


# setzt die Default-Sprachen
set_default_trlang () {
   trlang="" ;
   local f;
   local l;
   for f in $docroot/local/local.xml.*.; do
      l=${f#$docroot/local/local.xml.};
      l=${l%%.*};
      [[ $l == $srclang ]] || trlang="$trlang $l";
   done;
} # set_default_trlang


# Zeigt eine kurze Hilfe an
show_help ()
{
   local cmd=${0#*/} ;
   set_defaults ;
   cat << .HELP ;
$cmd --version
$cmd --help
$cmd ([Aktion] | [Option])*  FILE ..

Aktionen
--reminter            Zwischendateien löschen
--skl                 Gerüstdateien erzeugen
--xliff               XLIFF-Dateien erzeugen
--mtt                 Übersetzungstexte für die maschinelle Übersetzung
--trt                 Textdateien zur menschlichen Übersetzung
--mtrans              Texte maschinell übersetzen
--merge               Übersetzten Text in XLIFF-Dateien einfügen
--finish              Übersetzte XLIFF und Skeleton zum Dokument zusammenfügen
--upload              Übersetzte Dateien hochladen
--once                ein Skript für jede zu übersetzende Datei ausführen

FILE                  Rel. Pfad einer zu übersetzenden Datei ohne Sprachsuffix
                      ($_argv)
Optionen
--[no_]verbose        Erhöht den Umfang der Ausgabe des Scripts ($verbose)
--[no_]overwrite      Existierende Dateien überschreiben ($overwrite)
--[no_]story          Positionsargumente als Bildergeschichten ($story)
--[no_]zipfiles       Dateien gzip-komprimieren ($zipfiles)
--srclang SRCLANG     Kürzel der Ausgangssprache ($srclang)
--basedir BASEDIR     Basisverzeichnis
                      ($basedir)
--docroot DOCROOT     Document Root des Servers
                      ($docroot)
--locdir LOCDIR       Verzeichnis für Lokalisierungs-Zwischendateien
                      ($locdir)
--trtoutdir TRTOUTDIR Verzeichnis der Dateien für die menschliche Übersetzung
                      ($trtoutdir)
--trtindir TRTINDIR   Verzeichnis der übersetzten Dateien zur Weiterverarbeitung
                      ($trtindir)
--srcdir SRCDIR       Verzeichnis der Skripte
                      ($srcdir)
--stylesrc STYLESRC   Verzeichnis der Stil-Quelldateien
                      ($stylesrc)
--trname TRNAME       Name der Übersetzungmaschine ($trname)
--trlang TRLANG       Zielsprachen ($trlang)
```

```
--storyptn STORYPTN   Muster eines Bildergeschichten-Pfads ($storyptn)
--ftpmode FTPMODE     putnewer, putnotex oder put ($ftpmode)
--runonce RUNONCE     Skript, das für jede Datei ausgeführt wird
                      ($runonce)
.HELP
} # show_help


# Zeigt die Version an
show_version ()
{
   cat << .VERSION ;
web/src/localization/localize
Lokalisierung der Website http://kleider.herbaer.de
2015-01-09, Herbert Schiemann, h.schiemann@herbaer.de
GPL Version 2 oder neuer
.VERSION
} # show_version


# Variable und Zähler initialisieren
init_vars () {
   local v;
   declare_vars ;
   for v in $g_counters $g_variables $g_actions; do
      eval "$v=" ;
   done;
} # init_vars


# is_secure path/to/file
# Ist der Dateipfad sicher, d.h
# - Ist die Datei eine gewöhnliche Datei, lesbar und nicht leer?
# - Ist die Datei nicht im Wurzelverzeichnis?
# - Hat nur der Besitzer mehr als nur Leserecht für die Datei?
# - Hat nur der Besitzer Schreibrecht für das Verzeichnis?
# - Ist der Besitzer der Datei auch der Besitzer des Verzeichnisses?
is_secure ()
{
   local chk ;
   (( verbose )) && echo "prüfe Sicherheit $1" ;
   [[ -f "$1" && -r "$1" && -s "$1" ]] || return 1 ;
   [[ $(stat --format=%A "$1") =~ ^.{4}[r-]{6}$ ]] || return 1 ;
   chk=${1%/*};
   [[ -n "$chk" ]] || return 1;
   [[ "$chk" != "$1" ]] || chk=$(pwd);
   [[ -d "$chk" ]] || return 1 ;
   [[ $(stat --format=%u "$chk") == $(stat --format=%u "$1") ]] || return 1;
   if [[ $(stat --format=%A "$chk") =~ ^.{4}[rx-]{6}$ ]]; then
      (( verbose )) && echo "Datei $1 scheint sicher";
      return 0;
   fi;
   return 1;
} # is_secure


# read_configuration file
# file: Dateiname ("rc"-Datei) ohne Endung ".rc"
read_configuration ()
{
   local path=$g_configpath ;
   local file=$1.rc ;
   local dir;
   while [[ -n "$path" ]]; do
      dir=${path%%:*} ;
      if is_secure "$dir/$file"; then
         source "$dir/$file";
         return 0;
      fi;
      path=${path#*:} ;
      [[ "$dir" == "$path" ]] && break;
   done;
   return 1;
} # read_configuration


# Liste der existierenden Konfigurationen
list_configurations ()
{
   local path=$g_configpath ;
   local cfg=" ";
   local dir;
   local f;
   while [[ -n "$path" ]]; do
      dir=${path%%:*} ;
      for f in $dir/*.rc; do
         [[ -f $f && -r $f && -s $f ]] || continue ;
         f=${f##*/};
         f=${f%.rc};
         [[ -n $f ]] || continue ;
         [[ "$cfg" == "${cfg#* $f }" ]] && cfg="$cfg $f " ;
      done ;
      path=${path#*:} ;
      [[ "$dir" == "$path" ]] && break;
   done;
   cfg=${cfg# };
   echo ${cfg% };
} # list_configurations
```

```
# Argumente verarbeiten
read_args ()
{
   local wd ;
   local lastwd ;
   local var ;
   local ok ;

   has_actions=0 ;
   for wd in "$@"; do
      if [[ "$lastwd" = "--" ]]; then
          _argv="$_argv $wd";
      elif [[ -n "$lastwd" ]]; then
         if [[ "$wd" =~ ^[\ a-zA-Z0-9./_#-]+$ ]]; then
            if [[ "$lastwd" == "rc" && -n "$g_configpath" ]]; then
               if ! read_configuration $wd; then
                  (( verbose )) && echo "Kann Konfiguration $wd nicht lesen" ;
                  exit 10 ;
               fi ;
            else
               ok=0 ;
               for var in $g_variables; do
                  if [[ "$var" == "$lastwd" ]]; then
                     (( ++ok )) ;
                     eval "$var=\"$wd\"" ;
                     break ;
                  fi ;
               done ;
               if (( ! ok )); then
                  (( verbose )) && echo "Unbekannte Option --$lastwd $wd" ;
                  exit 11 ;
               fi ;
            fi ;
         else
            (( verbose )) && echo "Ungültiger Optionswert --$lastwd $wd" ;
            exit 12;
         fi;
         lastwd= ;
      else
         case "$wd" in
            --version )
               show_version ;
               exit 0 ;
               ;;
            --help )
               show_version ;
               show_help ;
               exit 0 ;
               ;;
            -- )
               if [[ -n "$_argv" ]]; then
                  lastwd=--;
                  continue;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 13 ;
               fi ;
               ;;
            --* )
               if [[ "$wd" =~ ^--[a-z][a-z0-9_]*$ ]]; then
                  lastwd=${wd#--} ;
                  ok=0 ;
                  for var in $g_counters ; do
                     if   [[ "$lastwd" == $var    ]]; then
                        eval "(( ++$lastwd ))" ;
                     elif [[ "$lastwd" == "no_$var" ]]; then
                        eval "${lastwd#no_}=0" ;
                     else
                        continue;
                     fi;
                     (( ++ok )) ;
                     break ;
                  done;
                  if (( !ok )); then
                     for var in $g_actions; do
                        if [[ "$lastwd" == "$var" ]]; then
                           eval "(( ++$var ))" ;
                           (( ++ok ));
                           has_actions=1;
                           break;
                        elif [[ "$lastwd" == "no_$var" ]]; then
                           eval "(( ++no_$var ))" ;
                           (( ++ok ));
                           break;
                        fi;
                     done;
                  fi;
                  (( ok )) && lastwd=;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 14 ;
               fi ;
               ;;
            * )
               if [[ -n $_argv ]]; then
                   _argv="$_argv $wd";
```

```
               else
                   (( verbose )) && echo "Ungültige Option $wd" ;
                   exit 15 ;
               fi;
               ;;
         esac ;
      fi ;
   done ;
   if [[ -n $lastwd && "$lastwd" != "--" ]]; then
      (( verbose )) && echo "Unverarbeitete Option --$lastwd";
      exit 16 ;
   fi ;
   [[ "$_argv" =~ ^[[:space:]]+$ ]] && _argv="" ;
} # read_args


# Aktionen ausführen
run_actions ()
{
   local act ;
   for act in $g_actions; do
     eval "(( ! has_actions && ! no_$act || $act )) && process_$act";
   done;
} # run_actions


# show_variables VARNAME1 VARNAME2
# Werte der Variablen anzeigen
show_variables ()
{
   local v ;
   for v in $g_counters $g_variables $g_actions $1; do
      eval "echo \"$v = \$$v\"" ;
   done;
} # show_variables


# Können die Eingabedateien gelesen werden?
# check_infiles first/path/to/file path/to/second_file ;
check_infiles ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -s "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist leer";
         return 1;
      fi;
      if [[ ! -r "$f" ]]; then
         (( verbose )) && echo "Kann Datei \"$f\" nicht lesen";
         return 1;
      fi;
   done;
   return 0;
} # check_infiles


# Können die Ausgabedateien erstellt werden?
# erstellt fehlende Verzeichnisse und löscht existierende Dateien
# nach Maßgabe der Variablen overwrite
# check_outfiles first/path/to/file path/to/second_file ;
check_outfiles ()
{
   local fp;
   local dir;
   local verb;
   (( verbose )) && verb=--verbose ;
   for fp in "$@"; do
      if [[ ! -e $fp ]]; then
         dir=${fp%/*};
         if [[ -n $dir && ! -e $dir ]]; then
            mkdir -p $verb $dir ;
            if [[ ! -d $dir ]]; then
               (( verbose )) && echo "$dir ist kein Verzeichnis";
               return 1;
            fi;
         fi;
      elif [[ -d $fp ]]; then
         (( verbose )) && echo "$fp ist ein Verzeichnis";
         return 1;
      elif (( overwrite )); then
         (( verbose )) && echo "lösche $fp";
         rm $fp;
      else
         (( verbose )) && echo "$fp existiert";
         return 1;
      fi;
      (( verbose )) && echo "$fp";
   done;
   return 0;
} # check_outfiles
```

```
# Sind die Dateien ausführbar?
# check_executeable first/path/to/script path/to/second_srcipt ;
check_executeable ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -x "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine ausführbare Datei";
         return 1;
      fi;
   done;
   return 0;
} # check_executeable


# Hilfsfunktion: Zwischendateien zu einer Quelldatei löschen
# procfile_reminter srcfile
procfile_reminter ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_reminter $q" ;
   local verb= ;
   (( verbose )) && verb=--verbose ;
   [[ -e $q ]] || q=$q. ;
   check_infiles $q || return;
   q=${q%.} ;
   [[ $q =~ \.$srclang$ ]] || return ;   # Kennung der Quellsprache?
   local s=$locdir/${q#$docroot/} ;      # Lokalisierungsverzeichnis statt docroot
   [[ -e $s.skl ]] && rm $verb $s.skl ;
   [[ -e $s.xlf ]] && rm $verb $s.xlf ;
   [[ -e $s.mtt ]] && rm $verb $s.mtt ;
   s=${s%.$srclang};
   local l;
   for l in $trlang; do
      [[ $l == $srclang ]] && continue;
      [[ -e $s.$l.rtr ]] && rm $verb $s.$l.rtr ;
      [[ -e $s.$l.xlf ]] && rm $verb $s.$l.xlf ;
   done;
} # procfile_reminter


# Hilfsfunktion: eine einzelne Gerüstdatei erzeugen
# procfile_skl srcfile
procfile_skl ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_skl $q" ;
   [[ -e $q ]] || q=$q. ;
   check_infiles $q || return;
   local s=$locdir/${q#$docroot/} ;      # Lokalisierungsverzeichnis statt docroot
   local ns ;                            # Namensraum des Wurzelelements der Quelldatei
   local ft ;                            # Kennung des Dateityps
   local sn ;                            # Name des Stylesheets
   s=${s%.}.skl ;                        # Suffix .skl für Gerüstdateien
   [[ $s =~ \.$srclang\. ]] || return ;  # Kennung der Quellsprache?
   check_outfiles $s        || return ;  # Kann die Gerüstdatei erstellt werden?
   ns=$(xsltproc $srcdir/xmlnsss.xslt $q) ;
   sn=${ns#*|};
   ns=${ns%|*};
   if [[ $ns == "http://www.w3.org/1999/xhtml" ]]; then
      ft=xhtml ;
   elif [[ $ns == "http://herbaer.de/xmlns/20141210/localization" ]]; then
      ft=local ;
   elif [[ $ns == "http://herbaer.de/xmlns/20080705/imgshow" ]]; then
      ft=imgshow ;
   else
      (( verbose )) && echo "Kann XML-Namensraum $ns nicht übersetzen";
      return ;
   fi;
   local tpx=$srcdir/${sn}_pre.xslt ;    # Vor-Transformation abhängig vom Stylesheet
   local tpf=$srcdir/${ft}_pre.xslt ;    # Vor-Transformation abhängig vom Namensraum
   local t=$srcdir/${ft}_clfy.xslt ;     # 1. Schritt zur Erzeugung der Gerüst-Datei
   local t2=$srcdir/skeleton_2.xslt ;    # 2. Schritt
   local c="cat $q";
   [[ -f $tpx ]] && c="$c | xsltproc $tpx -" ;
   [[ -f $tpf ]] && c="$c | xsltproc $tpf -" ;
   eval "$c | xsltproc $t - | xsltproc $t2 - > $s" ;
} # procfile_skl
```

```
# Hilfsfunktion: eine einzelne XLIFF-Datei erzeugen
# procfile_xliff srcfile
procfile_xliff ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_xliff $q" ;
   local b=${q#$docroot/} ;              # relativer Dateipfad ohne Punkt am Ende
   b=${b%.} ;
   local ow=$overwrite ;
   local s=$locdir/$b.skl ;              # Gerüstdatei
   local x=$locdir/$b.xlf ;              # XLIFF-Datei
   check_outfiles $x || return ;
   if ! check_infiles $s; then
      overwrite=0 ;
      procfile_skl $q ;
      overwrite=$ow;
      check_infiles $s || return 1 ;
   fi;
   local tx=$srcdir/skeleton_xliff.xslt ;
   local tc=$srcdir/xliff_clean.xslt ;
   xsltproc --stringparam p_file "$b" $tx $s | xsltproc -o $x $tc - ;
} # procfile_xliff


# Hilfsfunktion:
# eine einzelne Übersetzungstext-Datei für die automatische Übersetzung
# procfile_mtt srcfile
procfile_mtt ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_mtt $q" ;
   local b=${q#$docroot/} ;              # relativer Dateipfad ohne Punkt am Ende
   b=${b%.} ;
   local ow=$overwrite ;
   local x=$locdir/$b.xlf ;              # XLIFF-Datei
   local m=$locdir/$b.mtt ;              # Übersetzungstext-Datei
   check_outfiles $m || return ;
   if ! check_infiles $x; then
      overwrite=0 ;
      procfile_xliff $q ;
      overwrite=$ow;
      check_infiles $x || return 1 ;
   fi;
   local tt=$srcdir/xliff_txt.xslt ;
   xsltproc -o $m $tt $x ;
} # procfile_mtt


# Hilfsfunktion:
# Übersetzungstext-Dateien für die menschliche Übersetzung
# zu einer einzelnen Quelldatei
# procfile_trt srcfile
procfile_trt ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_trt $q" ;
   local b=${q#$docroot/} ;              # relativer Dateipfad ohne Punkt am Ende
   b=${b%.} ;
   local ow=$overwrite ;
   local x=$locdir/$b.xlf ;              # XLIFF-Datei
   if ! check_infiles $x; then
      overwrite=0 ;
      procfile_xliff $q ;
      overwrite=$ow;
      check_infiles $x || return 1 ;
   fi;
   local o=$trtoutdir/${b%.$srclang} ;   # Ausgabedatei ohne Sprach/Typ-Suffix
   local l ;                             # Zielsprache
   local o2;                             # Ausgabedatei
   local tt=$srcdir/xliff_txt.xslt ;
   for l in $trlang; do
      [[ $l == $srclang ]] && continue;
      o2=$o.$l.txt ;
      check_outfiles $o2 || continue ;
      xsltproc --stringparam p_targetlang $l $tt $x > $o2 ;
   done;
} # procfile_trt
```

```
# Hilfsfunktion:
# maschinelle Übersetzungen zu einer einzelnen Quelldatei
# procfile_mtrans srcfile
procfile_mtrans ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_mtrans $q" ;
   local b=${q#$docroot/} ;              # relativer Dateipfad ohne Punkt am Ende
   b=${b%.} ;
   local ow=$overwrite ;
   local m=$locdir/$b.mtt ;              # XLIFF-Datei
   if ! check_infiles $m; then
      overwrite=0 ;
      procfile_mtt $q ;
      overwrite=$ow;
      check_infiles $m || return 1 ;
   fi;
   local verb;
   (( verbose )) && verb=--verbose ;
   local o=$locdir/${b%.$srclang} ;      # Ausgabedatei ohne Sprach/Typ-Suffix
   local p1=$srcdir/mttext.pl ;
   local p2=$srcdir/resstruct.pl ;
   local l ;                             # Zielsprache
   local o2 ;                            # Ausgabedatei
   for l in $trlang; do
      [[ $l == $srclang ]] && continue;
      o2=$o.$l.rtr ;
      check_outfiles $o2 || continue ;
      $p1 $verb --in $m --srclang $srclang --tgtlang $l --trname $trname \
      | $p2 $verb --tgtlang $l > $o2                                      ;
   done;
} # procfile_mtrans


# Hilfsfunktion:
# rtr-Datei und XLIFF-Datei zusammenfügen
# procfile_merge srcfile
procfile_merge ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_merge $q" ;
   local verb ;
   (( verbose )) && verb=--verbose ;
   local b=${q#$docroot/} ;              # relativer Dateipfad ohne Punkt am Ende
   b=${b%.} ;
   local bb=${b%.$srclang} ;             # relativer Dateipfad ohne Sprach/Typ-Suffix
   local ow=$overwrite ;
   local tr=$srcdir/xliff_merge_rtr.xslt ;
   local l ;                             # Zielsprache
   local rt ;                            # übersetzte Übersetzungstext-Datei
   local x=$locdir/$b.xlf ;              # XLIFF - Datei der Quellsprache
   local o ;                             # Ausgabe: übersetzte XLIFF-Datei
   if ! check_infiles $x ; then
      overwrite=0 ;
      procfile_xliff $q ;
      overwrite=$ow;
      check_infiles $x || return ;
   fi;
   for l in $trlang; do
      [[ $l == $srclang ]] && continue;
      o=$locdir/$bb.$l.xlf ;
      check_outfiles $o || continue ;
      if check_infiles $trtindir/$bb.$l.rtr ; then
         rt=$trtindir/$bb.$l.rtr ;
      elif check_infiles $trtindir/$bb.$l.txt ; then
         rt=$trtindir/$bb.$l.rtr ;
         if check_outfiles $rt ; then
            $srcdir/resstruct.pl $verb --in $trtindir/$bb.$l.txt --tgtlang $l > $rt ;
         else
            rt=$locdir/$bb.$l.rtr ;
         fi ;
      else
         rt=$locdir/$bb.$l.rtr ;
      fi;
      if ! check_infiles $rt ; then
         overwrite=0 ;
         procfile_mtrans $q ;
         overwrite=$ow;
         check_infiles $rt || continue ;
      fi;
      xsltproc --stringparam p_rtrfile $rt $tr $x > $o ;
   done;
} # procfile_merge
```

```
# Hilfsfunktion:
# Skeleton und übersetzte XLIFF-Datei zum übersetzten Dokument zusammenfügen
# procfile_finish srcfile
procfile_finish ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_finish $q" ;
   local b=${q#$docroot/} ;              # relativer Dateipfad ohne Punkt am Ende
   b=${b%.} ;
   local bb=${b%.$srclang} ;             # relativer Dateipfad ohne Sprach/Typ-Suffix
   local ow=$overwrite ;
   local tr=$srcdir/skeleton_merge_xliff.xslt ;
   local s=$locdir/$b.skl ;              # Skeleton-Datei
   local x;                              # übersetzte XLIFF-Datei
   local o;                              # übersetztes Dokument

   local tp ;                            # Kennung für nachgeordnete Transformation

   local ns ;                            # XML-Namensraum
   local sn ;                            # Stylesheet-Basisname
   local tf ;                            # Kennung des Dateityps
   if ! check_infiles $s; then
      overwrite=0 ;
      procfile_skl $q;
      overwrite=$ow;
      check_infiles $s || return ;
   fi;

   ns=$(xsltproc $srcdir/xmlnsss.xslt $q) ;
   sn=${ns#*|};
   ns=${ns%|*};
   if [[ $ns == "http://www.w3.org/1999/xhtml" ]]; then
      ft=xhtml ;
   elif [[ $ns == "http://herbaer.de/xmlns/20141210/localization" ]]; then
      ft=local ;
   elif [[ $ns == "http://herbaer.de/xmlns/20080705/imgshow" ]]; then
      ft=imgshow ;
   else
      ft=none ;
      return ;
   fi;
   local tpx=$srcdir/${sn}_post.xslt ;  # Nach-Transformation abhängig vom Stylesheet
   local tpf=$srcdir/${ft}_post.xslt ;  # Nach-Transformation abhängig vom Namensraum
   # Befehl zur Transformation
   local c="xsltproc             \
     --stringparam p_xliff   \$x \
     --stringparam p_trglang \$l \
     $tr $s                      ";
   # Parameter für die Nach-Transformation
   local pparm="                                          \
     --stringparam p_localsrc $stylesrc/local.xml.$srclang  \
     --stringparam p_localweb $docroot/local/local.xml.\$l. "
   [[ -f $tpf ]] && c="$c | xsltproc $pparm $tpf -" ;
   [[ -f $tpx ]] && c="$c | xsltproc $pparm $tpx -" ;
   if check_executeable $stylesrc/rmxmlns.pl ;
   then
      c="$c | $stylesrc/rmxmlns.pl" ;
      (( verbose )) && c="$c --verbose";
   fi;
   c="$c > \$o.";
   for l in $trlang; do
      [[ $l == $srclang ]] && continue;
      x=$locdir/$bb.$l.xlf ;
      if ! check_infiles $x ; then
         overwrite=0 ;
         procfile_merge $q ;
         overwrite=$ow;
         check_infiles $x || continue ;
      fi;
      o=$docroot/$bb.$l ;
      check_outfiles $o. || continue ;
      eval $c;
      if (( zipfiles )); then
         [[ -f $o.gz ]] && rm $o.gz ;
         gzip --best --stdout $o. > $o.gz ;
      fi;
      continue;
   done;
} # procfile_finish
```

```
procfile_upload ()
{
   local q=$1 ;              # Quelldatei im Dokument-Verzeichnis
   local b=${q#$docroot/} ;  # relativer Dateipfad
   b=${b%.} ;                # ohne Punkt
   b=${b%.$srclang} ;        # ohne Sprach-Suffix
   local o;                  # übersetztes Dokument
   for l in $trlang; do
      [[ $l == $srclang ]] && continue;
      o=$docroot/$b.$l ;
      [[ -f $o.   ]] && echo "$ftpmode $b.$l.";
      [[ -f $o.gz ]] && echo "$ftpmode $b.$l.gz";
   done;
} # upload


# Hilfsfunktion:
# Einmal-Programm aufrufen
# procfile_once srcfile
procfile_once ()
{
   local q=$1 ;                          # Quelldatei im Dokument-Verzeichnis
   (( verbose )) && echo "procfile_once $q" ;
   local verb;
   (( verbose )) && verb=--verbose ;
   local ow ;
   if [[ -n $overwrite ]] ;
   then
      if (( overwrite )) ;
      then
         ow=--overwrite ;
      else
         ow=--no_overwrite ;
      fi;
   fi;
   $runonce $verb $ow --trlang "$trlang" --srclang $srclang --srcdir $srcdir $q;
} # procfile_once


# Hilfsfunktion: die Verarbeitungsschleife
# procloop STEP $docroot/subdir
# STEP: skl ...
procloop ()
{
   local s=$1
   local dir=$2 ;
   (( verbose )) && echo "procloop $s $dir";
   local f  ; # Dateipfad im Verzeichnis
   local n  ; # Dateiname
   for f in $dir/* ; do
      if [[ -d $f ]]; then
         n=${f##*/};
         [[ $n == images   ]] && continue;
         [[ $n == smallimg ]] && continue;
         [[ $n == thumbs   ]] && continue;
         [[ $n == montage  ]] && continue;
         procloop $s $f;
         continue;
      elif [[ $f =~ \.$srclang\.$ ]]; then
         procfile_$s $f;
      fi;
   done;
} # procloop


# Hilfsfunktion: einzelne Datei verarbeiten oder Schleife
# Falls (( story )), werden alle Bildergeschichten verarbeitet
# proctry STEP
proctry ()
{
   local s=$1 ;
   local f ;
   local sid ;
   local ss ;
   (( verbose )) && echo "proctry $s";
   if [[ -n "$_argv" ]]; then
      if (( story )); then
         for sid in $_argv; do
            eval f=$docroot/$storyptn;
            procfile_$s $f ;
         done;
      else
         for f in $_argv; do
            if [[ -d "$docroot/$f" ]]; then
               procloop $s $docroot/$f ;
            elif [[ -f $docroot/$f.xhtml.$srclang. ]]; then
               procfile_$s $docroot/$f.xhtml.$srclang. ;
            else
               procfile_$s $docroot/$f.$srclang. ;
            fi;
         done;
      fi;
   elif (( story )); then
      sid='*';
      eval ss=$storyptn;
      for f in $docroot/$ss; do
```

```
         procfile_$s $f ;
      done;
   else
      procloop $s $docroot ;
   fi;
} # proctry


# Zwischendateien löschen
process_reminter ()
{
   (( verbose )) && echo "process_reminter" ;
   proctry reminter ;
}


# Gerüstdateien erzeugen
process_skl ()
{
   (( verbose )) && echo "process_skl" ;
   if check_infiles               \
      $srcdir/imgshow_clfy.xslt   \
      $srcdir/local_clfy.xslt     \
      $srcdir/xhtml_clfy.xslt     \
      $srcdir/skeleton_2.xslt     ;
   then
      proctry skl ;
   else
      (( verbose )) && echo "Kann skl-Dateien nicht erzeugen";
      return 1;
   fi;
} # process_skl


# XLIFF-Dateien erzeugen
process_xliff ()
{
   (( verbose )) && echo "process_xliff" ;
   if check_infiles               \
      $srcdir/imgshow_clfy.xslt   \
      $srcdir/local_clfy.xslt     \
      $srcdir/xhtml_clfy.xslt     \
      $srcdir/skeleton_2.xslt     \
      $srcdir/skeleton_xliff.xslt \
      $srcdir/xliff_clean.xslt    ;
   then
      proctry xliff ;
   else
      (( verbose )) && echo "Kann XLIFF-Dateien nicht erzeugen";
      return 1;
   fi;
} # process_xliff


# Übersetzungstext-Dateien für die maschinelle Übersetzung
process_mtt ()
{
   (( verbose )) && echo "process_mtt" ;
   if check_infiles               \
      $srcdir/imgshow_clfy.xslt   \
      $srcdir/local_clfy.xslt     \
      $srcdir/xhtml_clfy.xslt     \
      $srcdir/skeleton_2.xslt     \
      $srcdir/skeleton_xliff.xslt \
      $srcdir/xliff_clean.xslt    \
      $srcdir/xliff_txt.xslt      ;
   then
      proctry mtt ;
   else
      (( verbose )) && echo "Kann mtt-Dateien nicht erzeugen";
      return 1;
   fi;
} # process_mtt
```

```
# Übersetzungstext-Dateien für die menschliche Übersetzung
process_trt ()
{
   (( verbose )) && echo "process_mtt" ;
   if check_infiles               \
      $srcdir/imgshow_clfy.xslt   \
      $srcdir/local_clfy.xslt     \
      $srcdir/xhtml_clfy.xslt     \
      $srcdir/skeleton_2.xslt     \
      $srcdir/skeleton_xliff.xslt \
      $srcdir/xliff_clean.xslt    \
      $srcdir/xliff_txt.xslt      ;
   then
      proctry trt ;
   else
      (( verbose )) && echo "Kann trt-Dateien nicht erzeugen";
      return 1;
   fi;
} # process_trt

# maschinelle Übersetzung
process_mtrans ()
{
   (( verbose )) && echo "process_mtrans" ;
   if check_executeable     \
      $srcdir/mttext.pl     \
      $srcdir/resstruct.pl  ;
   then
      local verb;
      (( verbose )) && verb=--verbose ;
      [[ $trname =~ "pipe" ]] && $srcdir/pipe_srv.pl $verb &
      proctry mtrans ;
      [[ $trname =~ "pipe" ]] && $srcdir/pipe_srv_stop.pl $verb ;
   else
      (( verbose )) && echo "Kann rtr-Dateien nicht erzeugen";
      return 1;
   fi;
} # process_mtrans

# rtr-Dateien und XLIFF-Dokumente zusammenfügen
process_merge ()
{
   (( verbose )) && echo "process_merge" ;
   if check_infiles $srcdir/xliff_merge_rtr.xslt; then
      proctry merge ;
   else
      (( verbose )) && echo "Kann übersetzte XLIFF-Dateien nicht erzeugen";
      return 1;
   fi;
} # process_merge

# Übersetzte XLIFF und Skeleton zum Dokument zusammenfügen
process_finish ()
{
   (( verbose )) && echo "process_finish" ;
   if check_infiles $srcdir/skeleton_merge_xliff.xslt; then
      proctry finish ;
   else
      (( verbose )) && echo "Kann Skeleton- und XLIFF-Dateien nicht zusammenfügen";
      return 1;
   fi;
}

# Übersetzte Dateien hochladen
process_upload ()
{
   (( verbose )) && echo "process_upload" ;
   if check_executeable \
      $srcdir/ftp.pl    ;
   then
      local verb;
      (( verbose )) && verb=--verbose
      local v=$verbose;
      verbose=0 ;
      { proctry upload; }                        \
      | $srcdir/ftp.pl $verb --putbase $docroot   ;
      verbose=$v ;
   fi;
} # process_upload

# Einmal-Programm ausführen
process_once ()
{
   (( verbose )) && echo "process_once" ;
   if check_executeable $runonce ;
   then
      proctry once ;
   fi;
} # process_mtrans
```

```
# Sicherheit
export PATH=/bin:/usr/bin ;
IFS=$' \t\n' ;


init_vars ;


set -o noclobber ;   # existierende Dateien werden nicht überschrieben
shopt -s extglob nullglob ;


read_args "$@" ;
set_defaults ;


(( verbose > 1 )) && show_variables;


run_actions ;
exit 0;
```

pipe_srv.pl

[Quelltext]

Übersicht

`pipe_srv.pl --help|--version`

`pipe_srv.pl [ --verbose… | --no_verbose ]`
`[--base` *BASE*`] [--req` *REQ*`] [--resp` *RESP*`] [--trname` *TRNAME*`] [--trname` *ERRSLEEP*`]`

Zweck

Dieses Programm kommuniziert über benannte Pipes mit anderen Prozessen und stellt die Funktionen eines Übersetzers (`Herbaer::Translate`, `Translate.pm`) zur Verfügung.

Eine Anfrage erfolgt in einer Zeile über die benannte Pipe *REQ*. Die Anfrage wird durch den Doppelpunkt `:` in Teile (Methodenname und Argumente) zerlegt. Die Anwort erfolgt ebenfalls in einer Zeile über die benannte Pipe *RESP*.

In den Teilen einer Anfrage oder in der Antwort müssen das Zeilenendezeichen und der Doppelpunkt ersetzt („geschützt”) werden. Zunächst wird das Zeichen \ („Rückschrägstrich”) durch `\\` ersetzt. Das Zeichen `:` wird durch `\c`, das Zeilenendezeichen durch `\n` ersetzt.

Anfragen (Requests)

Die erste Komponente (Zeichen bis zum ersten Doppelpunkt oder bis zum Zeilenende) ist der Name einer Methode des Übersetzers oder eines Befehls für dieses Steuerprogramm. Abhängig vom Namen können oder müssen ein oder mehrere Argumente (durch einen Doppelpunkt vom vorhergehende Text abgegrenzte Zeichenketten) folgen. Die Anfragen werden (normalerweise) vom Perl-Modul `Herbaer::Translate::Pipe`, Datei `Pipe.pm`, gesendet. Die möglichen Anfragen sind:

`trname:`*TRNAME*

> Dieser Befehl setzt einen neuen Übersetzernamen. Der Konstruktor `Herbaer::Translate::new` wird mit der Zeichenkette *NAME* als Argument aufgerufen, wenn ein Übersetzer benötigt wird. Das Übersetzer-Objekt wird erst dann erzeugt, wenn es benötigt wird, also bei den Anforderungen `translate` und `learn`. Wenn *NAME* vom bisherigen Namen abweicht und ein Übersetzer-Objekt existiert, dann wird die Methode `finish` aufgerufen und das Übersetzer-Objekt freigegeben.

`translate:`*TEXT*`:`*SRL*`:`*TGL*

> Die Anfrage ruft die Methode `translate` des Übersetzers auf. Der Text *TEXT* wird aus der Sprache *SRL* in die Sprache *TGL* übersetzt. Die Antwort ist der übersetzte Text oder die leere Zeichenkette, falls der Übersetzer den Text nicht übersetzen kann. Falls nötig, wird der Übersetzer neu angelegt.

`learn:`*TEXT*`:`*SRL*`:`*TGL*`:`*TRTEXT*

> Diese Anfrage ruft die Methode `learn` des Übersetzers auf. Der Übersetzer wird bei Bedarf angelegt. Er lernt, dass die Übersetzung des Textes *TEXT* aus der Sprache *SRL* in die Sprache *TGL* der Text *TRTEXT* ist. Die Antwort ist `OK`, es sei denn, der Übersetzer kann nicht angelegt werden. In diesem Fall ist die Antwort `ERROR no translator`.

`finish`

> Wenn ein Übersetzer-Objekt existiert, wird dessen Methode `finish` aufgerufen. Der Übersetzer kann dann (z.B.) die neu gelernten Übersetzungen dauerhaft speichern. Die Antwort ist `OK`. Falls kein Übersetzer-Objekt existiert, ist die Antwort `WARNING no translator`.

`translator_name`

Wenn ein Übersetzer-Objekt existiert, wird dessen Methode `translator_name` aufgerufen. Die Antwort ist die Zeichenfolge `OK` *`TRANSLATOR_NAME`*, wobei der Platzhalter *`TRANSLATOR_NAME`* für das Ergibnis der Methode `translator_name` steht. Falls kein Übersetzer-Objekt existiert, ist die Antwort `WARNING no translator`.

`stop`

Die Anfrage ist ein Befehl an diesen Dienst-Prozess, sich zu beenden. Wenn ein Übersetzer-Objekt existiert, wird dessen Methode `finish` aufgerufen.

Diese Anfrage wird normalerweise vom Programm `pipe_srv_stop.pl` gesendet.

`verbose`

Erhöht den Umfang der Meldungen dieses Prozesses nach STDERR. Diese Anfrage entspricht der Befehlszeilen-Option `--verbose`.

`silent`

Unterbindet Meldungen des MAIN-Moduls nach STDERR. Diese Anfrage entspricht der Befehlszeilen-Option `--no_verbose`.

Optionen

Die Werte der Platzhalter *`BASE`*, *`REQ`*, *`RESP`* und *`TRNAME`* können die Platzhalter *`${base}`*, *`${req}`*, *`${resp}`*, *`${trname}`* und *`${errsleep}`* für die Werte der (anderen) Platzhalter enthalten. Zum Einsetzen der Platzhalterwerte dient das Modul `Herber::Replace` (`Replace.pm`).

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Meldungen über den Ablauf, alle Anfragen und die Antworten werden auch nach STDERR ausgegeben.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen nach STDERR. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--base` *`BASE`*

Die Werte der Platzhalter *`REQ`*, *`RESP`* und *`TRNAME`* können Die Zeichenfolge *`${base}`* enthalten. Die Zeichenfolgen *`${base}`* wird durch den Wert von *`BASE`* ersetzt.

`--req` *`REQ`*

`REQ` ist der Dateipfad der benannten Pipe, über die in UTF-8-Kodierung die Anfragen gestellt werden. Das Verzeichnis muss existieren.

`--resp` *`RESP`*

`RESP` ist der Dateipfad der benannten Pipe, über die in UTF-8-Kodierung die Antworten gesendet werden. Das Verzeichnis muss existierten.

`--trname` *TRNAME*

TRNAME ist der Name des Übersetzers, der gestartet wird, wenn eine `translate`-Anforderung vor der ersten `trname`-Anforderung erfolgt.

`--errsleep` *ERRSLEEP*

Wenn eine Anfrage nicht gelesen werden kann, wartet der Übersetzerprozess *ERRSLEEP* Sekunden, bevor er wieder versucht, eine Anfrage zu lesen.

Software-Voraussetzungen

Das Programm ist mit Perl Version 5.10.1 entwickelt. Es benutzt die folgenden Module:

`Herbaer::Readargs`

Die Funktionen `read_args` aus diesem Modul verarbeitet die Befehlszeilenargumente, die Funktion `print_message_with_values` gibt die Hilfe mit den aktuellen Einstellungen aus.

Das Modul ist im Zusammenhang mit den Kalendern beschrieben.

`Herbaer::Replace` (Datei `Replace.pm`)

Die Funktion `replace` ersetzt Platzhalter.

`Herbaer::Translate` (Datei `Translate.pm`)

Die Methode `new` liefert den Übersetzer.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# Server-Prozess für die maschinelle Übersetzung, Kommunikation mittels benannter Pipes
# 2015-08-07 Herbert Schiemann <h.schiemann@herbaer.de>
# 2020-04-02 Anpassung an Perl 5.24.1
# GPL Version 2 oder neuer


package main;


use utf8 ;
use Cwd qw(realpath) ;
use Herbaer::Readargs ;
use Herbaer::Replace ;
use Herbaer::Translate ;
use POSIX ;


binmode (STDOUT, "encoding(utf-8)");
binmode (STDERR, "encoding(utf-8)");


my $basedir = realpath ($0);
$basedir =~ s/\/src\/localization\/.*// ;


# Hash der Kommandozeilen-Argumente
my $args = {
   "[cnt]verbose" => 0,
   "base"         => "$basedir/pipe",     # Basisverzeichnis für Pipes
   "req"          => "\${base}/request",  # Request-Pfad
   "resp"         => "\${base}/response", # Response-Pfad
   "trname"       => "default",           # Übersetzer
   "errsleep"     => 1 ,                  # Warte-Sekunden im Fehlerfall
};


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
pipe_srv.pl v20200402
Server-Prozess für die maschinelle Übersetzung, Kommunikation mittels benannter Pipes
(C) 2015 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
};
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   set_default ($args);
   print_message_with_values (<<'HELP', $args);
pipe_srv.pl [Optionen]
--[no_]verbose      Umfang der Meldungen ${[cnt]verbose}
--base     BASE     Platzhalter in den anderen Argumenten ${base}
--req      REQ      Request-Pfad ${req}
--resp     RESP     Response-Pfad ${rest}
--trname   TRNAME   Übersetzer ${trname}
--errsleep ERRSLEEP Wartezeit in Sekunden, nachdem kein Request ansteht ${errsleep}
HELP
   exit 0;
};


# Platzhalter in den Argumenten einsetzen
sub set_default {
   my $args = shift;
   my $key;
   for $key (qw(base req resp trname)) {
      $args -> {$key} = replace ($args -> {$key}, $args);
   }
} # set_default


my $trans;             # der Übersetzer
my $wait_request = 1;  # auf einen weiteren Request warten?


read_args ($args);
set_default ($args);


# "\" wird durch "\\",
# ":" wird durch "\c",
# Zeilenende durch "\n" ersetzt
sub encode {
   my $t = shift;
   $t =~ s/\\/\\\\/g;
   $t =~ s/\n/\\n/g;
   $t =~ s/:/\\c/g;
   return $t;
} # encode
```

```
# macht _encode rückgängig
sub decode {
   my $t = shift;
   $t =~ s/\\(.)/$1 eq "n" ? "\n" : $1 eq "c" ? ":" : $1/ge;
   return $t;
} # decode


sub check_pipes {
   my $args = shift;
   my $verb = $args -> {"[cnt]verbose"};
   my $k;
   for $k (qw(req resp)) {
      my $p = $args -> {$k};
      if (! -p $p) {
         if (! mkfifo ($p, 0700)) {
            print STDERR "Kann Pipe $p nicht erstellen\n" if $verb;
            return 0;
         }
      }
   }
   1;
} # check_pipes


sub rm_pipes {
   my $args = shift;
   my $verb = $args -> {"[cnt]verbose"};
   my $k;
   for $k (qw(req resp)) {
      my $p = $args -> {$k};
      if (-p $p && ! unlink ($p)) {
         print STDERR "Kann Pipe $p nicht entfernen\n" if $verb;
      }
   }
   1;
} # rm_pipes


# behandelt eine Anfrage
sub handle_request  {
   my $args = shift ;
   my $verb = $args -> {"[cnt]verbose"};
   my $reqp  = $args -> {"req"};
   my $respp = $args -> {"resp"};
   my $reqh;
   print STDERR "wait for request\n" if $verb > 1;
   if (! open ($reqh, "<:encoding(utf-8)", $reqp)) {
      print STDERR "Kann Request-Pipe $reqp nicht öffnen\n" if $verb;
      sleep ($args -> {"errsleep"});
      return 0;
   }
   my $resph;
   if (! open ($resph, ">:encoding(utf-8)", $respp)) {
      print STDERR "Kann Response-Pipe $respp nicht öffnen\n" if $verb;
      return 0;
   }
   my $req = <$reqh>;
   close $reqh;
   if ($req) {
      print STDERR "REQUEST $req\n" if $verb;
      my $resp = response_to_request ($args, $req);
      print STDERR "RESPONSE $resp\n" if $verb;
      print $resph "$resp\n";
   }
   close $resph;
   return 1;
} # handle_request
```

```
sub response_to_request {
   my ($args, $req) = @_;
   my $p;
   my $st;
   my $sl;
   my $tt;
   my $tl;
   if ($req =~ /^trname:(.*)$/) {
      $p = $1;
      if ($args -> {"trname"} ne $p) {
         if ($trans) {
            $trans -> finish ();
            $trans = undef;
         }
         $args -> {"trname"} = $p;
         return "OK trname $p";
      }
   }
   elsif ($req =~ /^translate:([^:]+):([^:]+):([^:]+)$/) {
      $trans = new Herbaer::Translate ($args -> {"trname"}) unless $trans;
      if ($trans) {
         $st = $1;
         $sl = $2;
         $tl = $3;
         $tt = $trans -> translate (decode ($st), $sl, $tl) || "";
         return encode ($tt);
      }
      else {
         return "";
      }
   }
   elsif ($req =~ /^learn:([^:]+):([^:]+):([^:]+):([^:]+)$/) {
      $trans = new Herbaer::Translate ($args -> {"trname"}) unless $trans;
      if ($trans) {
         $st = $1;
         $sl = $2;
         $tl = $3;
         $tt = $4;
         $trans -> learn (decode ($st), $sl, $tl, decode ($tt));
         return "OK learn";
      }
      else {
         return "ERROR no translator";
      }
   }
   elsif ($req eq "finish") {
      if ($trans) {
         $trans -> finish ();
         return "OK finish";
      }
      else {
         return "WARNING no translator";
      }
   }
   elsif ($req eq "translator_name") {
      if ($trans) {
         return "OK " . $trans -> translator_name ();
      }
      else {
         return "WARNING no translator";
      }
   }
   elsif ($req eq "stop") {
      $trans -> finish () if $trans;
      $wait_request = 0;
      return "OK";
   }
   elsif ($req eq "verbose") {
      ++$args -> {"[cnt]verbose"};
      return "OK " . $args -> {"[cnt]verbose"};
   }
   elsif ($req eq "silent") {
      $args -> {"[cnt]verbose"} = 0;
      return "OK silent";
   }
   else {
      return "ERROR unknown command";
   }
} # response_to_request


check_pipes ($args);
while ($wait_request) {
  handle_request ($args);
}
rm_pipes ($args);
```

pipe_srv_stop.pl

[Quelltext]

Übersicht

`pipe_srv_stop.pl --help | --version`

`pipe_srv_stop.pl [ --verbose ... | --no_verbose ] [ --base` *BASE* `]`

Zweck

Dieses Programm beendet den Übersetzerprozess (`pipe_srv.pl`). Mittels des Moduls `Herbaer::Translate::Pipe` (Datei `Pipe.pm`) sendet er dem Übersetzerprozess ein `stop`-Anfrage.

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Meldungen werden nach STDERR ausgegeben.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen nach STDERR. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--base` *BASE*

Verzeichnis der benannten Pipes. *BASE* wird als Parameter an `Herbaer::Translate::Pipe -> new ()` übergeben.

Software-Voraussetzungen

Das Programm ist mit Perl Version 5.10.1 entwickelt. Es benutzt die folgenden Module:

`Herbaer::Readargs`

Verarbeitet die Befehlszeilenargumente.

`Herbaer::Translate::Pipe` (Datei `Pipe.pm`)

Die Methode `stop` sendet die `stop`-Anfrage an den Übersetzerprozess.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# Beendet den Server-Prozess für die maschinelle Übersetzung
# 2015-08-11 Herbert Schiemann <h.schiemann@herbaer.de>
# GPL Version 2 oder neuer


package main;


use utf8 ;
use Cwd qw(realpath) ;
use Herbaer::Readargs ;
use Herbaer::Translate::Pipe ;


binmode (STDOUT, "encoding(utf-8)");
binmode (STDERR, "encoding(utf-8)");


my $basedir = realpath ($0);
$basedir =~ s/\/src\/localization\/.*// ;


# Hash der Kommandozeilen-Argumente
my $args = {
   "[cnt]verbose" => 0,
   "base"         => "$basedir/pipe",  # Basisverzeichnis für Pipes
};


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
pipe_srv_stop.pl v20150811
Beendet den Server-Prozess für die maschinelle Übersetzung
(C) 2015 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
};
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   print_message_with_values (<<'HELP', $args);
pipe_srv.pl [Optionen]
--[no_]verbose    Umfang der Meldungen ${[cnt]verbose}
--base   BASE     Verzeichnis der Pipes ${base}
HELP
   exit 0;
};


read_args ($args);
my $translator = new Herbaer::Translate::Pipe ($args -> {"base"});
my $ok = $translator -> stop () || "";
if ($ok ne "OK") {
   print STDERR "Kann Übersetzungsprozess nicht beenden.\n" if $args -> {"[cnt]verbose"};
}
else {
   print STDERR "OK\n";
}
```

runonce

[Quelltext]

Übersicht

`runonce --help | --version`

`runonce [ --verbose ... | --no_verbose ] [ --overwrite | --no_overwrite ]`
`[ --srcdir` *`SRCDIR`* `] [ --trlang` *`TRLANG`* `] [ --srclang` *`SRCLANG`* `]` *`FILE`* ...

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus.

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Meldungen über den Programmablauf werden nach STDOUT ausgegeben.

`--no_verbose`

Diese Option hebt die Wirkung der Option `--verbose` auf.

`--overwrite`

Existierende Dateien werden überschrieben.

`--no_overwrite`

Existierende Dateien werden nicht überschrieben.

`--srcdir` *`SRCDIR`*

Verzeichnis der XSLT-Dateien und Perl-Skripte, die dieses Skript benutzt.

`--trlang` *`TRLANG`*

`TRLANG` ist Liste der Kennungen der Zielsprachen. Trennzeichen ist das Leerzeichen.

`--srclang` *`SRCLANG`*

Kennung der Ausgangssprache. Die Kennung bezeichnet als Dateinamenssuffix die Dateien, die zu übersetzen sind.

`FILE`

Der Pfad einer zu übersetzenden Datei in der Quellsprache mit dem Sprach-Suffix.

Beschreibung

Das Skript `runonce` ist für eine Aktion gedacht, die für eine übersetzte Datei einmal auszuführen ist. Es wird (normalerweise) über das Skript `localize` aufgerufen.

Es ist ein Rahmenprogramm für Einmal-Skripte.

Dieses Skript filtert die Übersetzungen der Quelldatei(en) durch das Programm `rmxmlns.pl`. Lokalisierungsdateien (XML-Namensraum `http://herbaer.de/xmlns/20141210/localization`) durchlaufen zuvor die Transformation `local_post.xslt`. Das Ergebnis wird unter dem Pfad der übersetzten Datei mit dem zusätzlichen Suffix `.2` gespeichert. Wenn das Ergebnis mit der ursprünglichen Übersetzung übereinstimmt, wird die Datei mit dem Suffix `.2` gelöscht. Andernfalls wird die ursprüngliche Übersetzung gelöscht und das Suffix `.2` des Ergebnisses entfernt.

Quelltext

[Beschreibung]

```
#!/bin/bash
# -*- coding:utf-8 -*-
# 2015-10-06 Herbert Schiemann <h.schiemann@herbaer.de>
# Programm zur einmaligen Anwendung im Zusammenhang mit der Übersetzung
# als Vorlage für andere Einmal-Programme


# Zähler, Variable, Aktionen
declare_vars ()
{
   # Ein Leerzeichen als Wert bedeutet, dass Positionsargumente verarbeitet werden
   _argv=" ";

   # Suchpfad für rc-Dateien, : - getrennte Liste von Verzeichnispfaden
   # Falls leer, wird die Option --rc nicht speziell behandelt
   g_configpath= ;

   # Zähler
   g_counters=" \
    verbose     \
    overwrite   " ;

   # Variable
   g_variables=" \
     srcdir      \
     trlang      \
     srclang     ";

   # Aktionen
   g_actions="" ;
} # declare_vars


# setzt Vorgabe-Werte
set_defaults ()
{
   [[ -n $overwrite ]] || overwrite=0 ;
   if [[ -z $srcdir ]]; then
      srcdir=$(realpath $0);
      srcdir=${srcdir#/*};
   fi;
   [[ -n $srclang   ]] || srclang=de ;
} # set_defaults


# Zeigt eine kurze Hilfe an
show_help ()
{
   local cmd=${0#*/} ;
   set_defaults ;
   cat << .HELP ;
$cmd --version
$cmd --help
$cmd [Option]*  FILE ..

FILE                  Rel. Pfad einer zu übersetzenden Datei ohne Sprachsuffix
                      ($_argv)
Optionen
--[no_]verbose        Erhöht den Umfang der Ausgabe des Scripts ($verbose)
--[no_]overwrite      Existierende Dateien überschreiben ($overwrite)
--srcdir SRCDIR       Verzeichnis der Skripte
                      ($srcdir)
--trlang TRLANG       Zielsprachen ($trlang)
--srclang SRCLANG     Kürzel der Ausgangssprache ($srclang)
.HELP
} # show_help


# Zeigt die Version an
show_version ()
{
   cat << .VERSION ;
$0
Programm zur einmaligen Anwendung im Zusammenhang mit Lokalisierungen
2015-10-06, Herbert Schiemann, h.schiemann@herbaer.de
GPL Version 2 oder neuer
.VERSION
} # show_version
```

```
# Variable und Zähler initialisieren
init_vars () {
   local v;
   declare_vars ;
   for v in $g_counters $g_variables $g_actions; do
      eval "$v=" ;
   done;
} # init_vars


# Argumente verarbeiten
read_args ()
{
   local wd ;
   local lastwd ;
   local var ;
   local ok ;

   has_actions=0 ;
   for wd in "$@"; do
      if [[ "$lastwd" = "--" ]]; then
          _argv="$_argv $wd";
      elif [[ -n "$lastwd" ]]; then
         if [[ "$wd" =~ ^[\ a-zA-Z0-9./_#-]+$ ]]; then
            if [[ "$lastwd" == "rc" && -n "$g_configpath" ]]; then
               if ! read_configuration $wd; then
                  (( verbose )) && echo "Kann Konfiguration $wd nicht lesen" ;
                  exit 10 ;
               fi ;
            else
               ok=0 ;
               for var in $g_variables; do
                  if [[ "$var" == "$lastwd" ]]; then
                     (( ++ok )) ;
                     eval "$var=\"$wd\"" ;
                     break ;
                  fi ;
               done ;
               if (( ! ok )); then
                  (( verbose )) && echo "Unbekannte Option --$lastwd $wd" ;
                  exit 11 ;
               fi ;
            fi ;
         else
            (( verbose )) && echo "Ungültiger Optionswert --$lastwd $wd" ;
            exit 12;
         fi;
         lastwd= ;
      else
         case "$wd" in
            --version )
               show_version ;
               exit 0 ;
               ;;
            --help )
               show_version ;
               show_help ;
               exit 0 ;
               ;;
            -- )
               if [[ -n "$_argv" ]]; then
                  lastwd=--;
                  continue;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 13 ;
               fi ;
               ;;
            --* )
               if [[ "$wd" =~ ^--[a-z][a-z0-9_]*$ ]]; then
                  lastwd=${wd#--} ;
                  ok=0 ;
                  for var in $g_counters ; do
                     if   [[ "$lastwd" == $var    ]]; then
                        eval "(( ++$lastwd ))" ;
                     elif [[ "$lastwd" == "no_$var" ]]; then
                        eval "${lastwd#no_}=0" ;
                     else
                        continue;
                     fi;
                     (( ++ok )) ;
                     break ;
                  done;
                  if (( !ok )); then
                     for var in $g_actions; do
                        if [[ "$lastwd" == "$var" ]]; then
                           eval "(( ++$var ))" ;
                           (( ++ok ));
                           has_actions=1;
                           break;
                        elif [[ "$lastwd" == "no_$var" ]]; then
                           eval "(( ++no_$var ))" ;
                           (( ++ok ));
                           break;
                        fi;
                     done;
```

```
                  fi;
                  (( ok )) && lastwd=;
               else
                  (( verbose )) && echo "Ungültige Option $wd" ;
                  exit 14 ;
               fi ;
               ;;
            * )
               if [[ -n $_argv ]]; then
                   _argv="$_argv $wd";
               else
                    (( verbose )) && echo "Ungültige Option $wd" ;
                    exit 15 ;
               fi;
               ;;
         esac ;
      fi ;
   done ;
   if [[ -n $lastwd && "$lastwd" != "--" ]]; then
      (( verbose )) && echo "Unverarbeitete Option --$lastwd";
      exit 16 ;
   fi ;
   [[ "$_argv" =~ ^[[:space:]]+$ ]] && _argv="" ;
} # read_args


# Aktionen ausführen
run_actions ()
{
   local act ;
   for act in $g_actions; do
     eval "(( ! has_actions && ! no_$act || $act )) && process_$act";
   done;
} # run_actions


# show_variables VARNAME1 VARNAME2
# Werte der Variablen anzeigen
show_variables ()
{
   local v ;
   for v in $g_counters $g_variables $g_actions $1; do
      eval "echo \"$v = \$$v\"" ;
   done;
} # show_variables


# Können die Eingabedateien gelesen werden?
# check_infiles first/path/to/file path/to/second_file ;
check_infiles ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -s "$f" ]]; then
         (( verbose )) && echo "\"$f\" ist leer";
         return 1;
      fi;
      if [[ ! -r "$f" ]]; then
         (( verbose )) && echo "Kann Datei \"$f\" nicht lesen";
         return 1;
      fi;
   done;
   return 0;
} # check_infiles
```

```
# Können die Ausgabedateien erstellt werden?
# erstellt fehlende Verzeichnisse und löscht existierende Dateien
# nach Maßgabe der Variablen overwrite
# check_outfiles first/path/to/file path/to/second_file ;
check_outfiles ()
{
   local fp;
   local dir;
   local verb;
   (( verbose )) && verb=--verbose ;
   for fp in "$@"; do
      if [[ ! -e $fp ]]; then
         dir=${fp%/*};
         if [[ -n $dir && ! -e $dir ]]; then
            mkdir -p $verb $dir ;
            if [[ ! -d $dir ]]; then
               (( verbose )) && echo "$dir ist kein Verzeichnis";
               return 1;
            fi;
         fi;
      elif [[ -d $fp ]]; then
         (( verbose )) && echo "$fp ist ein Verzeichnis";
         return 1;
      elif (( overwrite )); then
         (( verbose )) && echo "lösche $fp";
         rm $fp;
      else
         (( verbose )) && echo "$fp existiert";
         return 1;
      fi;
      (( verbose )) && echo "$fp";
   done;
   return 0;
} # check_outfiles


# Sind die Dateien ausführbar?
# check_executeable first/path/to/script path/to/second_srcipt ;
check_executeable ()
{
   local f ;
   for f in "$@"; do
      if [[ ! -f "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine gewöhnliche Datei";
         return 1;
      fi;
      if [[ ! -x "$f" ]]; then
         (( verbose )) && echo "$f\" ist keine ausführbare Datei";
         return 1;
      fi;
   done;
   return 0;
} # check_executeable


# Sicherheit
export PATH=/bin:/usr/bin ;
IFS=$' \t\n' ;


init_vars ;


set -o noclobber ;   # existierende Dateien werden nicht versehentlich überschrieben
shopt -s extglob nullglob ;


read_args "$@" ;
set_defaults ;


(( verbose > 1 )) && show_variables;
```

```
# hier folgt die auszuführende Aktion
for file in $_argv; do
   if check_infiles $file;
   then
      xmlns=$(xsltproc $srcdir/xmlnsss.xslt $file);
      xmlns=${xmlns%|*}
   else
      xmlns=unknown ;
   fi;
   if [[ $xmlns == "http://herbaer.de/xmlns/20141210/localization" ]]; then
      posttrans=$srcdir/local_post.xslt ;
   else
      posttrans= ;
   fi;
   b=${file%.$srclang.} ;
   if (( verbose )); then
      verb=--verbose ;
   else
      verb= ;
   fi;
   for lang in $trlang; do
      (( verbose )) && echo "LANG $lang";
      f=$b.$lang. ;
      if check_infiles $f; then
         f2=$f.2 ;
         if [[ -n $posttrans ]]; then
            xsltproc $posttrans $f | $srcdir/rmxmlns.pl $verb > $f2 ;
         else
            cat $f | $srcdir/rmxmlns.pl $verb > $f2 ;
         fi;
         if diff $f $f2 > /dev/null 2>&1 ; then
            (( verbose )) && echo "NOCHNG $f";
            rm $f2;
         else
            rm $f;
            mv $f2 $f;
            [[ -f $f.gz ]] && rm $f.gz ;
            gzip --best --stdout $f > $f.gz ;
         fi;
      fi;
   done;
done;


exit 0;
```

local_post.xslt

[Quelltext]

Namensräume

Die Namensraum-Präfixe, die aus dem erzeugten Dokument ausgeschlossen sind, sind durch einen Stern (*) in der ersten Spalte gekennzeichnet.

	Präfix	**Namensraum**
	xml	http://www.w3.org/XML/1998/namespace
*	ti	http://herbaer.de/xmlns/201500703/transinfo/
*	d	http://herbaer.de/xmlns/20051201/doc
	xsl	http://www.w3.org/1999/XSL/Transform

Ausgabe (output)

Method	xml
Encoding	utf-8

Muster-Vorlagen (matching templates)

Muster-Vorlage /

Muster-Vorlage *

Elemente werden "hohl" kopiert.

Muster-Vorlage processing-instruction() | @*

Verarbeitungsanweisungen und Attribute werden kopiert

Muster-Vorlage @ti:*

ti:* - Attribute werden nicht kopiert.

Quelltext

[Beschreibung]

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="/pool/xslt_ht.xslt" type="application/xml"?>
<!--
  file KLEIDER/web/src/localization/local_post.xslt
  Nachbearbeitung übersetzter Lokalisierungs-Dateien
  2015 Herbert Schiemann <h.schiemann@herbaer.de>
  Borkener Str. 167, 46284 Dorsten, Germany
  Diese Datei wird unter den Bedingungen der GPL Version 2 oder
  einer neueren Version weitergegeben.
  Jede Gewährleistung ist ausgeschlossen
-->
<xsl:stylesheet
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:d   = "http://herbaer.de/xmlns/20051201/doc"
  xmlns:ti  = "http://herbaer.de/xmlns/201500703/transinfo/"
  exclude-result-prefixes = "d ti"
  version   = "1.0"
>
<xsl:output method = "xml" encoding = "utf-8"/>


<xsl:template match = "/">
  <xsl:apply-templates select = "text() | processing-instruction() | comment() | *"/>
</xsl:template>


<xsl:template match = "*">
  <xsl:copy>
    <xsl:apply-templates select = "@* | * | text()"/>
  </xsl:copy>
</xsl:template>


<xsl:template match = "processing-instruction() | @*">
  <xsl:copy-of select = "."/>
</xsl:template>


<xsl:template match = "@ti:*"/>


</xsl:stylesheet>
```

ftp.pl

[Quelltext]

Übersicht

`ftp.pl --help|--version`

`ftp.pl[ --verbose… | --no_verbose ]`
`[ --secrets` *`SECRETS`* `][ --putbase` *`PUTBASE`* `][ --msec` *`MSEC`* `]`

Optionen

`--help`

Gibt eine kurze Hilfe mit den aktuellen Einstellungen aus

`--version`

Gibt kurze Hinweise zum Programm und die Version aus.

`--verbose`

Erhöht den Umfang der Meldungen nach STDERR.

`--no_verbose`

Unterdrückt die Ausgabe von Meldungen. Die Optionen `--verbose` und `--no_verbose` werden der Reihe nach ausgewertet.

`--secrets` *`SECRETS`*

Dateipfad der Datei mit den FTP-Zugangsdaten.

`--putbase` *`PUTBASE`*

Lokales Basis-Verzeichnis für PUT-Befehle.

`--msec` *`MSEC`*

Wartezeit zwischen zwei FTP-Befehlen in Millisekunden. Der Wert wird an `Herbaer::Upload` (`Upload.pm`) weitergegeben.

Beschreibung

Dieses Programm dient zum Upload von Dateien. Es liest die Standardeingabe zeilenweise. Leere Zeilen oder Zeilen, die mit dem Zeichen # beginnen, werden übergangen. Zeilen, die aus dem Wort „`quit`”, umgeben von beliebig vielen Leerzeichen, beenden das Programm. Alle anderen Zeilen werden als Befehl an das Modul `Herbaer::Upload` (`Upload.pm`) gesendet.

Das Programm ist mit Perl Version 5.10.1 entwickelt.

Quelltext

[Beschreibung]

```
#!/usr/bin/perl -w
# 2016-03-22 Herbert Schiemann <h.schiemann@herbaer.de>
# Datei-Upload
# GPL Version 2 oder neuer


# 2020-04-17 quit, Meldungen [cnt]verbose > 1, msec


package main;


use utf8 ;
use Cwd qw(realpath) ;
use Herbaer::Readargs ;
use Herbaer::Upload ;


binmode (STDIN,  ":utf8") ;
binmode (STDOUT, ":utf8") ;


my $basedir = realpath($0);
$basedir =~ s/\/src\/localization\/.*// ;


# Hash der Kommandozeilen-Argumente
my $args = {
   "[cnt]verbose" => 1,
   "secrets"      => "$basedir/secrets",
   "putbase"      => "$basedir/docroot",
   "msec"         => 300,
};


# gibt die Version nach STDOUT aus
sub version {
   print <<'VERSION' ;
KLEIDER/web/src/localization/ftp.pl
FTP für die Website http://kleider.herbaer.de
2016 Herbert Schiemann <h.schiemann@herbaer.de>
VERSION
}
$args -> {"[sr]version"} = sub { version (); exit 0; };


$args -> {"[sr]help"} = sub {
   version ();
   print_message_with_values (<<'HELP', $args);
ftp.pl [Optionen]
--[no_]verbose     Umfang der Meldungen ${[cnt]verbose}
--secrets SECRETS  Datei mit Zugangsdaten
                   ${secrets}
--putbase PUTBASE  Lokales Basisverzeichnis für den Upload
                   ${putbase}
--msec    MSEC     Wartezeit zwischen FTP-Befehlen in ms
HELP
   exit 0;
};


read_args ($args);


my $ftp = Herbaer::Upload -> new
  ($args -> {"secrets"}, $args -> {"[cnt]verbose"}, $args -> {"msec"});
exit 1 unless $ftp;
$ftp -> putbase ($args -> {"putbase"});


my $line;
my $verb = $args -> {"[cnt]verbose"};
while ( defined ($line = <STDIN>) ) {
   print STDERR "ftp.pl cmd $line" if $verb > 1;
   $line =~ s/^\s+// ;
   $line =~ s/\s+$// ;
   next unless $line;
   next if ($line =~ /^#/);
   last if $line eq "quit";
   $ftp -> cmd ($line);
   print STDERR "ftp.pl cmd done\n" if $verb > 1;
}
print STDERR "ftp.pl exit\n" if $verb > 1;
```

Herbaer::Upload

[Quelltext]

Übersicht

```
#!/usr/bin/perl

my $ftp = new Herbaer::Upload ("/geheimer/pfad/zu/zugangsdaten");
$ftp -> putbase ("/das/lokale/dokumentenverzeichnis");
$ftp -> cmd ("putbase /das/lokale/dokumentenverzeichnis");
$ftp -> cmd ("put datei_1");
$ftp -> cmd ("putnotex dir/datei_2");
$ftp -> cmd ("putnewer dir/neu/datei_3");
```

Beschreibung

Das Modul `Herbaer::Upload` kapselt das Standard-Modul `Net::FTP` für Anwendungen zur Pflege meiner Website.

Die Datei `Upload.pm` wird unter dem Teilpfad `Herbaer/Upload.pm` des Suchpfades für Perl-Module (`@INC`) gesucht. Auf meinem Rechner ist `/usr/local/share/perl/5.10.1/Herbaer/Upload.pm` ein symbolischer Verweis.

Funktionen

`my $ftp = new Herbaer::Upload ($secrets_file, $verbose, $millisec)`

Ergibt eine neues Obekt für den FTP-Upload. *`$secrets_file`* ist der Dateipfad der Datei mit den Zugangsdaten. Die weiteren Parameter sind optional. Ein positiver Wert von *`$verbose`* führt zu (mehr) Meldungen nach `STDERR`. *`$millisec`* ist die Wartezeit in Millisekunden am Ende einer Ausführung der Prozedur `cmd`.

Das Objekt speichert die folgenden Daten (Eigenschaften):

`ftp`

Die Net::FTP - Instanz

`verbose`

Eine nicht negative Zahl, bestimmt den Umfang der Meldungen an `STDERR`.

`putbase`

Der Pfad eines lokalen Verzeichnisses. Dieses Verzeichnis ist die Basis für alle relativen Dateipfade in Put-Befehlen. Siehe `putbase`.

`remotedir`

Das aktuelle Verzeichnis auf dem FTP-Server mit abschließendem Schrägstrich.

`usleep`

Die Wartezeit nach der Ausführung eines FTP-Befehl durch `cmd` in Mikrosekunden.

FTP-Zugangsdaten

Die FTP-Zugangsdaten werden aus einer Textdatei gelesen. Die Textdatei wird zeilenweise verarbeitet. Zeilen, die mit dem Zeichen # beginnen, sind Kommentarzeilen. Zeilen, die nur Leerraum-Zeichen enthalten, werden übergangen. Andere Zeilen haben den Aufbau

key = value

Vor und nach dem Gleichheitszeichen können beliebig viele Leerzeichen stehen. Der Wert besteht aus Nicht-Leer-Zeichen. Nach dem Wert können ein Leerzeichen und weitere Zeichen als Kommentar folgen. Hier werden die Werte zu den folgenden Schlüsseln (*key*) gelesen:

- ftp.host
- ftp.user
- ftp.password

Ein Beispiel für eine Zugangsdaten-Datei ist secrets.

```
$ftp -> putbase ($local_put_dir)
```

$local_put_dir ist ein absoluter oder relativer Verzeichnispfad. Ein relativer Verzeichnispfad bezieht sich auf das bisherige lokale Basisverzeichnis für put-Befehle. Wenn das Verzeichnis zu dem Pfad existiert, wird es das neue lokale Basisverzeichnis für put-Befehle. Der Rückgabewert ist der Pfad des bisherigen Basisverzeichnisses für put-Befehle (s. putbase).

Wenn *$local_put_dir* nicht definiert oder leer ist, wird nur der Pfad des bisherigen Basisverzeichnisses für put-Befehle zurückgegeben.

```
$ftp -> cmd ($command_line)
```

$command_line ist ein Befehl, der ausgeführt sind. Nach der Ausführung wird usleep Mikrosekunden gewartet. Die möglichen Befehle sind nachfolgend beschrieben.

```
$ftp -> cmd ("put $path_to_file_or_dir")
```

$path_to_file_or_dir ist ein relativer Pfad bezüglich des lokalen Basisverzeichnisses für put-Befehle (putbase) und serverseitig bezüglich des FTP-Wurzelverzeichnisses.

Wenn *$path_to_file_or_dir* ein lokaler Verzeichnispfad ist, wird die Funktion rekursiv für alle Unterpfade des Verzeichnisses aufgerufen.

Wenn unter dem Pfad eine lokale Datei existiert, wird sie hochgeladen. Eine existierende Datei auf dem Server wird überschrieben. Fehlende Verzeichnisse auf dem Server werden bei Bedarf angelegt, aber nicht, wenn eine gleichnamige Datei existiert.

```
$ftp -> cmd ("putnotex $path_to_file_or_dir")
```

Dieser Befehl ist ähnlich dem Befehl put $path_to_file. Eine Datei wird nur dann hochgeladen, wenn auf dem Server keine Datei unter dem Pfad existiert.

```
$ftp -> cmd ("putnewer $path_to_file_or_dir")
```

Dieser Befehl ist ähnlich dem Befehl put $path_to_file. Eine Datei wird nur hochgeladen, wenn auf dem Server keine Datei unter dem Pfad existiert oder wenn die existierende Datei älter ist als die lokale Datei.

Problem

Wenn viele Dateien schnell hochgeladen worden sind, dauert die Ausführung eines nächsten put-Befehls manchmal Minuten. Vielleicht kann eine Wartezeit zwischen zwei Uploads dem Server etwas mehr Ruhe verschaffen (2020-04-17).

Quelltext

[Beschreibung]

```
# symlink Herbaer/Upload.pm
# 2020-04-17 Herbert Schiemann <h.schiemann@herbaer.de>


package Herbaer::Upload ;


use Cwd qw(getcwd realpath) ;
use File::Spec::Functions qw(catfile catdir file_name_is_absolute) ;
use Net::FTP;
use Time::HiRes qw(usleep);
use utf8 ;


binmode (STDERR, ":utf8") ;


sub new {
   my ($class, $secrets, $verbose, $mssleep) = @_;
   # mssleep: Schlafdauer in Millisekunden
   my $hnd;
   if (! open ($hnd, "<:encoding(utf-8)", $secrets)) {
      print STDERR "Kann Datei $secrets nicht lesen:$!\n" if $verbose;
      return undef;
   }
   my $line;
   my $host;
   my $user;
   my $password;
   while (defined ($line = <$hnd>)) {
      $line =~ s/\s*$//;
      if ($line =~ /^\s*ftp\.user\s*=?\s*(\S*)/) {
         $user = $1;
      }
      elsif ($line =~ /^\s*ftp\.password\s*=?\s*(\S*)/) {
         $password = $1;
      }
      elsif ($line =~ /^\s*ftp\.host\s*=\s*(\S*)/) {
         $host = $1;
      }
   }
   my $ftp = Net::FTP->new ($host);
   if (!$ftp) {
      print STDERR "Keine Verbindung zu $host\n" if $verbose;
      return undef;
   }
   if (!$ftp -> login ($user, $password)) {
      print STDERR "LOGIN Fehler\n", $ftp -> message() if $verbose;
      return undef;
   }
   $ftp -> binary ();
   my $self = {
      "ftp"       => $ftp,
      "verbose"   => $verbose,
      "putbase"   => getcwd() ,
      "remotedir" => "",   # das aktuelle FTP-Remote-Verzeichnis
      "usleep"    => ($mssleep && $mssleep > 0 ? $mssleep * 1000 : 0) # Mikrosekunden
   };
   return bless ($self);
} # new


sub DESTROY {
   my $self = shift;
   my $ftp = $self -> {"ftp"};
   if ($ftp) {
      $ftp -> quit ();
   }
} # DESTROY


# Wechselt ein lokales Basisverzeichnis
sub _base {
   my ($self, $dir, $pg) = @_;
   my $verb = $self -> {"verbose"};
   my $n = "$pg" . "base";
   my $rv = $self -> {$n};
   if ($dir) {
      $dir = realpath (file_name_is_absolute ($dir) ? $dir : catdir ($rv, $dir));
      if (-d $dir) {
         $self -> {$n} = $dir;
         print STDERR "FTP $n: neues lokales Basisverzeichnis \"$dir\"\n" if $verb > 1;
      }
      else {
         print STDERR "FTP $n: \"$dir\" ist kein Verzeichnis.\n" if $verb > 1;
         return undef;
      }
   }
   return $rv;
} # _base
```

```
# stellt das Remote-FTP-Verzeichnis ein
# pth : Dateipfad der betroffenen Datei
# ergibt den Dateinamen ohne den Verzeichnispfad
sub _set_remote_dir {
   my ($self, $pth) = @_;
   my $rd   = $self -> {"remotedir"};
   my $verb = $self -> {"verbose"};
   $pth =~ s/([^\/]+)$//;
   my $file = $1; # Dateiname wird von $pth abgetrennt
   return $file  if $pth eq $rd;
   my $cr = $pth; # zu erzeugender Verzeichnispfad
   my $cm = "" ;  # gemeinsamer Teilpfad von $rd und $pth;
   my $d;         # Pfadkomponente
   my $q = 0;     # Schleife beenden?
   while ($cr =~ s/^([^\/]*\/)//) {
      $d = $1;
      if ( $rd =~ s/^([^\/]*\/)// ) {
         if ($d ne $1) {
            $cr = "$d$cr";
            $rd = "../$rd";
            last;
         }
         else {
            $cm .= $d;
         }
      }
      else {
         $cr = "$d$cr";
         last;
      }
   }
   my $ftp = $self -> {"ftp"};
   if ($rd) {
      $rd =~ s/[^\/]+/../g;
      if ( $ftp -> cwd ($rd) ) {
         print STDERR "FTP cwd $rd\n" if $verb > 1;
      }
      else {
         print STDERR "FTP ERROR: cwd $rd\n" if $verb;
         return "";
      }
   }
   if ($cr) {
      $cr =~ s/\/$// ;
      if ( $ftp -> mkdir ($cr, 1) ) {
         print STDERR "FTP mkdir $cr\n" if $verb > 1;
      }
      else {
         print STDERR "FTP ERROR: mkdir $cr\n" if $verb;
         return "";
      }
      if ( $ftp -> cwd ($cr) ) {
         print STDERR "FTP cwd $cr\n" if $verb > 1;
      }
      else {
         print STDERR "FTP ERROR: cwd $cr\n" if $verb;
         return "";
      }
   }
   $self -> {"remotedir"} = $pth;
   return $file;
} # _set_remote_dir


# Wechselt das lokale Basisverzeichnis für Uploads
sub putbase {
   my ($self, $dir) = @_;
   $self -> _base ($dir, "put");
} # putbase
```

```
# Upload einer Datei oder rekursiv eines Verzeichnisses,
# eine existierende Datei wird ersetzt nach Maßgabe des Parameters $mode
# $path ist in jedem Fall relativ zum lokalen put-Basisverzeichnis
# und zum entfernten FTP-Basisverzeichnis
# $mode
#    notex    nur Dateien, die nicht existieren, hochladen
#    newer    nur Dateien, die neuer sind, hochladen
sub _put {
   my ($self, $path, $mode) = @_;
   my $verb = $self -> {"verbose"};
   $mode //= "";
   $path =~ s/^\/+// ; # Schrägstriche am Anfang
   $path =~ s/\/+$// ; # und am Ende entfernen
   my $lp;             # lokaler Dateipfad
   $lp = catfile ($self -> {"putbase"}, $path);
   if (-d $lp) {
      my $dh;
      my $de;
      if (! opendir ($dh, $lp)) {
         print STDERR "FTP put: kann Verzeichnis $lp nicht lesen:\$!\n";
         return;
      }
      while ( $de = readdir ($dh) ) {
         next if $de eq "." || $de eq "..";
         next if $de =~ /~$/;
         $self -> _put (catfile ($path, $de), $mode);
      }
      close $dh;
   }
   elsif (-s $lp) {
      my $file = $self -> _set_remote_dir ($path);
      return unless $file ;
      my $ftp = $self -> {"ftp"};
      my $pt = 0;
      if ( $mode eq "notex" ) {
         my $sz = $ftp -> size ($file);
         if ($verb > 1) {
            print STDERR "Size $file: ", (defined $sz ? $sz : "undef"), "\n";
         }
         if (!$sz) {
            $pt = 1;
         }
         elsif ($verb) {
            print STDERR "FTP $path exists\n" if $verb;
         }
      }
      elsif ( $mode eq "newer" ) {
         my $mdt = $ftp -> mdtm ($file);
         my $ltm = (stat ($lp))[9];
         if ($verb > 1) {
            print STDERR "Modtime $file: ", (defined $mdt ? $mdt : "undef"), "\n";
            print STDERR "Loctime $lp: ", (defined $ltm ? $ltm : "undef"), "\n";
         }
         if (!$mdt || !$ltm || $mdt < $ltm + 60 * 60) {
            $pt = 1;
         }
         elsif ($verb) {
            print STDERR "FTP skip $lp\n" if $verb;
         }
      }
      else {
         $pt = 1;
      }
      if ($pt) {
         my $res;
         $res = $ftp -> put ($lp, $file) || "";
         print STDERR "FTP put $lp $file: $res\n" if $verb > 1;
      }
   }
} # _put
```

```
# verarbeitet ein Kommando
sub cmd {
   my ($self, $cmd) = @_;
   print STDERR "FTP cmd $cmd\n" if $self -> {"verbose"};
   my $list = [ split (' ', $cmd) ];
   return unless @$list ;
   my $c = shift (@$list);
   if ($c eq "put") {
      $self -> _put (@$list);
   }
   elsif ($c eq "putnewer") {
      $self -> _put (@$list, "newer");
   }
   elsif ($c eq "putnotex") {
      $self -> _put (@$list, "notex");
   }
   elsif ($c eq "putbase") {
      $self -> putbase (@$list);
   }
   else {
      print STDERR "ungültiger Befehl: $cmd\n" if $self -> {"verbose"};
   }
   my $sl = $self -> {"usleep"};
   usleep ($sl) if $sl;
} # cmd

1;
```