

Bretter, die die Welt bedeuten

Integration von Python in T_EX am Beispiel von Katalogeinträgen

Lukas C. Bossert, Uwe Ziegenhagen, Herbert Voß

Viele Dissertationen in der Archäologie enthalten am Ende der Arbeit einen Katalog, in dem die untersuchten Daten in einem bestimmten System aufgeschlüsselt präsentiert werden. In diesem Beitrag wird eine effiziente Gestaltung eines Kataloges und dessen Einbindung in den Fließtext vorgestellt.

Ein Katalog kann aus verschiedenen Einträgen bestehen, wie Bildern, Bohrproben, Architekturelementen etc., die aufgelistet werden. Eine händische Erstellung dieser einzelnen Einträge des Kataloges, beispielsweise über `\section` oder `\subsection` und anschließend in einer Umgebung `itemize`, ist nicht effizient, fehleranfällig und nur bei wenigen Katalogeinträgen einsetzbar. Es muss zudem berücksichtigt werden, dass die vorgegebenen Kategorien nicht für jeden Katalogeintrag passend sind, sodass Kategorien leer bleiben und dann im Katalogeintrag nicht auftauchen sollen. Dabei soll der Code des Katalogeintrags möglichst viel redundante Tipp-Arbeit abnehmen, wie sie beispielsweise bei Maßeinheiten vorkommt. Darüber hinaus sollen alle Einträge immer gleich formatiert sein und ihr Aussehen global verändert werden können.

Nachdem eine Lösung gefunden wurde, die allen bisher genannten Anforderungen entspricht (siehe unten), sollten in einer Kategorie alle Seitenzahlen enthalten sein, auf denen im Haupttext auf den Katalogeintrag verwiesen wird. Ein Lösungsansatz sah die Nutzung von `glossaries` vor: Dafür musste allerdings für jeden Katalogeintrag ein eigenes Glossar angelegt werden, was nicht nur viel händische Arbeit bedeutet, sondern auch die Zählerkapazitäten von T_EX überforderte.

Es musste also eine andere Lösung her und die Erinnerung an einen Beitrag in »»Die T_EXnische Komödie«« half. [1] Damals ging es darum, bei einem Werkkatalog die Erwähnung des Stückes im Haupttext anzugeben. Es war genau das, was gesucht wurde und den damals verwendeten Code gab es noch auf <http://uweziegenhagen.de/?p=3020>. Allerdings bestand die damalige Aufgabe darin, nur *eine* Erwähnung

im Haupttext anzugeben, was mit `label` und `ref` bewerkstelligt werden konnte. Im vorliegenden Fall wurden jedoch alle Erwähnungen im Haupttext benötigt. Es wurde entschieden, dieses Problem erst einmal mit der Skriptsprache Python zu lösen.

Um gemeinsam an dem hybriden Konstrukt von X_YL^AT_EX und Python arbeiten zu können, wurde ein Repository auf [github](https://github.com)¹ erstellt, einer Plattform, die unabhängig von der Programmiersprache einen exzellenten Austausch und eine detaillierte Versionskontrolle ermöglicht.

Katalogeintrag

Die Aufgabe war einen Katalog zu den Häusern in Pompeji anzulegen, der Auskunft über den Namen des Hauses, dessen Verortung und Grundstücksgröße geben soll. Zudem soll eine kurze Beschreibung enthalten sein und die Angabe über die Innenausstattung, die wiederum auf die Untergruppen Mosaik, Wandgemälde und Skulptur aufgeschlüsselt werden kann. Zum Schluss soll im Katalogeintrag angezeigt werden, auf welchen Seiten des Fließtextes das Haus genannt wird.

Die gefundene Lösung für die Umsetzung der Anforderungen funktioniert mit Hilfe des Pakets `keyval`.² Dafür werden in der Präambel verschiedene Schlüssel (`keys`) definiert. In der Grundversion sieht die Definition eines Eintrags wie folgt aus:³

```
\define@key{family}{key}{#1}
```

Für das konkrete Beispiel wird die `key`-Familie (`family`) mit `catalogue` angegeben, der Schlüssel (`key`), was einer Kategorie im Katalog entspricht, als `house` bezeichnet und als Resultat soll der Wert im Makro `\KVhouse` gespeichert werden:

```
\define@key{catalogue}{house}{\def\KVhouse{#1}}
```

Dem Beispiel entsprechend, können alle Kategorien als Schlüssel angelegt werden (Listing 1):

Listing 1: Definition der Schlüssel

```
1 \makeatletter
2 \define@key{catalogue}{house}{\def\KVhouse{#1}}
3 \define@key{catalogue}{label}{\def\KVlabel{#1}}
4 \define@key{catalogue}{description}{\def\KVdescription{#1}}
5 \define@key{catalogue}{location}{\def\KVlocation{#1}}
6 \define@key{catalogue}{size}{\def\KVsize{#1}}
```

¹ <https://github.com/LukasCBossert/DITK-TeX-Python>

² Basierend auf der Idee, vorgestellt auf: <http://tex.stackexchange.com/a/254336/98739>

³ Vgl. <http://www.tug.org/tugboat/tb30-1/tb94wright-keyval.pdf>

```

7 \define@key{catalogue}{interior}{\def\KVinterior{#1}}
8 \define@key{catalogue}{interiorM}{\def\KVinteriorM{#1}}
9 \define@key{catalogue}{interiorW}{\def\KVinteriorW{#1}}
10 \define@key{catalogue}{interiorS}{\def\KVinteriorS{#1}}
11 \makeatother

```

Das Aussehen eines Katalogeintrages wird separat mit einem Makro definiert: `\newcommand\catalogueentry[1]...` Darin wird zunächst festgelegt, dass eine neue Gruppe beginnt (`\begingroup`), sodass es zu keinen Problemen mit den jeweils definierten Schlüsseln kommt, da diese für jeden Katalogeintrag neu definiert werden. Dann sollen die Einträge im Flattersatz gesetzt werden (`\RaggedRight`) und schließlich die Angabe, welche Schlüssel Familie (hier `catalogue`) auszulesen ist.

Listing 2: Definition der Katalogeinträge, Anfang

```

1 \newcommand\catalogueentry[1]{%
2 \begingroup
3 \RaggedRight
4 \setkeys{catalogue}{#1}
5 ...

```

Es folgt in der Definition die Verarbeitung der einzelnen Schlüssel: Da nur die Ausgabe einer Kategorie erfolgen soll, wenn diese auch mit Informationen versehen ist, wird dies über die Abfrage `\ifdef` erledigt. Der Inhalt der Kategorie `house` wird als `\section`-Titel verwendet und, wenn vorhanden, mit einem `\label` versehen. Die weiteren Kategorien sollen in einer `labeling`-Umgebung aufgelistet werden. Die Definition wird mit `\endgroup` geschlossen.

Listing 3: Definition der Katalogeinträge, Fortsetzung

```

1 ...
2 \ifdef{\KVhouse}{\section{\KVhouse
3   \ifdef{\KVlabel}{\label{\KVlabel}}{}}{}}
4 \begin{labeling}{Beschreibung}
5   \ifdef{\KVdescription}{\item[Beschreibung] \KVdescription}{ }
6   \ifdef{\KVlocation}{\item[Verortung] \KVlocation}{ }
7   \ifdef{\KVinterior}{\item[Ausstattung] \KVinterior
8     \begin{labeling}{Wandgemälde}
9       \ifdef{\KVinteriorM}{\item[Mosaike:] \KVinteriorM}{ }
10      \ifdef{\KVinteriorW}{\item[Wandgemälde:] \KVinteriorW}{ }
11      \ifdef{\KVinteriorS}{\item[Statuen:] \KVinteriorS}{ }
12      \end{labeling}
13    }{ }
14   \ifdef{\KVsize}{\item[Größe] \KVsize\,$\text{m}^2$}{ }
15 \end{labeling}

```

```

16 \endgroup
17 }

```

Mit dieser Einstellung lassen sich die Katalogeinträge schon sehr gut darstellen. Im Fließtext des Hauptdokuments kann man an gewünschter Stelle mit `\catalogueentry` einen Katalogeintrag eintragen. So wird aus den folgenden Angaben (Listing 4)

Listing 4: Katalogeintrag für das Haus des M. Fabius Rufus

```

\catalogueentry{%
  house={Haus des M. Fabius Rufus},
  label={haus:M-Fabius-Rufus},
  size={172},
  description={Haus besteht aus mehreren Einzelgebäuden.},
  location={Regio VII, Insula 16, Eingang 17--22.},
  interior={Reicher Fundkomplex.},
  interiorM={S/W-Mosaik.},
  interiorW={Dionysius mit einer Mänade, Narzissus und ein Cupido, Hercules und
↪Deinira etc.},
  interiorS={Bronzene Statue eines Epheben.},
}

```

ein Katalogeintrag, der so aussieht:

Haus des M. Fabius Rufus

Beschreibung Haus besteht aus mehreren Einzelgebäuden.

Verortung Regio VII, Insula 16, Eingang 17–22.

Ausstattung Reicher Fundkomplex.

Mosaik: S/W-Mosaik.

Wandgemälde: Dionysius mit einer Mänade, Narzissus und ein Cupido, Hercules und Deinira etc.

Statuen: Bronzene Statue eines Epheben.

Größe 172 m²

Wie man bei diesem Beispiel sieht, ist die Reihenfolge, in der die Kategorien angegeben werden, irrelevant, da die Definition in der Präambel entscheidend ist. Der Wert bei `size` wird intern sogleich an das vordefinierte `\SI`-Makro mit entsprechender Einheit (m²) übergeben. Ähnlich kann auch mit anderen Angaben verfahren werden (bspw. bei Abbildungen können die `\label` an ein vordefiniertes Makro `\cref` übergeben werden).

Bei dem oben beschriebenen Katalogeintrag zum »Haus des M. Fabius Rufus« sind alle Kategorien ausgefüllt. Wenn eine Kategorie nicht ausgefüllt ist, wird sie nicht ausgegeben. Allerdings besteht die Kategorie `interior` [Ausstattung] zusätzlich aus drei Unterkategorien (`interiorM` [Mosaike], `interiorW` [Wandgemälde], `interiorS` [Statuen]). Diese sollen auch dann angezeigt werden, wenn `interior` selbst nicht definiert ist.

Ein solcher Fall tritt beim »Haus des Wilden Ebers« auf. Im Fließtext ist der Katalogeintrag wie folgt ausgefüllt (Listing 5):

Listing 5: Katalogeintrag für das Haus des Wilden Ebers

```
\catalogueentry{%
  house={Haus des Wilden Ebers},
  label={Haus-des-Wilden-Ebers},
  size={54},
  description={Renovierung nach Erdbeben 62\,n.\,Chr.},
  location={Regio VII, Insula 4, Eingang 48, 43.},
  interiorM={S/W-Mosaik.},
  interiorW={Venus, Leda und der Schwan, Ariadne und Theseus.},
}
```

So wird daraus:

Haus des Wilden Ebers

Beschreibung	Renovierung nach Erdbeben 62 n. Chr.
Verortung	Regio VII, Insula 4, Eingang 48, 43.
Ausstattung	Mosaike: S/W-Mosaik.
	Wandgemälde: Venus, Leda und der Schwan, Ariadne und Theseus.
Größe	54 m ²

Um zu diesem Ergebnis zu kommen, ist es notwendig, mit Booleschen Operatoren zu arbeiten. Dafür müssen bei den `key`-Definitionen drei Operatoren eingeführt werden (Listing 6):

Listing 6: Definition der Booleschen Operatoren

```
1 \newbool{@KVinteriorM}%Mosaik
2 \newbool{@KVinteriorW}%Wandgemälde
3 \newbool{@KVinteriorS}%Statue
```

Diese Booleschen Operatoren werden bei der Definition der einzelnen Einträge eingebaut und als true gesetzt, wenn dieser Eintrag mit Informationen versehen wird. Konkret sieht die Modifikation so aus (Listing 7):

Listing 7: Einsetzen der Booleschen Operatoren in die Schlüssel

```

1 \define@key{catalogue}{interiorM}{\def\KVinteriorM{#1}\booltrue{@KVinteriorM}}
2 \define@key{catalogue}{interiorW}{\def\KVinteriorW{#1}\booltrue{@KVinteriorW}}
3 \define@key{catalogue}{interiorS}{\def\KVinteriorS{#1}\booltrue{@KVinteriorS}}

```

Zudem müssen die Booleschen Operatoren bei der Ausgabe der Katalogeinträge eingesetzt werden, sodass geprüft wird (`\ifboolexpr`), ob einer oder mehrere der Operatoren auf true gesetzt ist (Listing 8).

Listing 8: Einsetzen der Booleschen Operatoren in den Katalogeintrag

```

1 \ifdef{\KVinterior}{%
2   \item[Ausstattung] \KVinterior
3   \ifboolexpr{bool{@KVinteriorM}
4     or bool{@KVinteriorW}
5     or bool{@KVinteriorS}}{%
6     \begin{labeling}{Wandgemälde}
7       \ifdef{\KVinteriorM}{\item[Mosaike] \KVinteriorM}{
8         \ifdef{\KVinteriorW}{\item[Wandgemälde] \KVinteriorW}{
9           \ifdef{\KVinteriorS}{\item[Statuen] \KVinteriorS}{
10            \end{labeling}
11          }{}}
12      {\ifboolexpr{bool{@KVinteriorM}
13        or bool{@KVinteriorW}
14        or bool{@KVinteriorS}}{%
15        \item[Ausstattung]%
16        \begin{labeling}{Wandgemälde}
17          \ifdef{\KVinteriorM}{\item[Mosaike] \KVinteriorM}{
18            \ifdef{\KVinteriorW}{\item[Wandgemälde] \KVinteriorW}{
19              \ifdef{\KVinteriorS}{\item[Statuen] \KVinteriorS}{
20                \end{labeling}
21              }{}}

```

Mit dieser modifizierten Ergänzung der Katalogeintragsdefinition werden die Unterkategorien von interior ausgegeben, auch wenn interior selbst nicht definiert ist.

Was noch fehlt, ist die Ausgabe der Seiten, auf denen der Katalogeintrag erwähnt wird. Dies erfolgt zunächst über Python, anschließend wird die T_EX-Variante vorgestellt. Lösungen über andere Programmiersprachen sind natürlich auch denkbar.

Problemlösung mit Python

Python ist eine interpretierte Programmiersprache, die den Python-Neuling durch weitestgehende Nicht-Nutzung von Klammern verwirrt. Stattdessen nutzt Python Leerzeichen als Einrückungen, um zusammengehörige Code-Teile zu kennzeichnen. Was erst einmal gewöhnungsbedürftig klingt, hat in der Praxis Vorteile, denn selbst fremder Python-Code ist erstaunlich gut lesbar. Im aktuellen Anwendungsfall beschränken wir uns jedoch nicht auf Python, sondern nutzen mit pandas eine Erweiterung zur Datenanalyse.⁴ pandas, ursprünglich zur Analyse von Finanzdaten entwickelt, erleichtert die Arbeit mit strukturierten Daten aller Art. Das Einlesen, Filtern und Aggregieren von Daten geht damit erstaunlich einfach, dem interessierten Leser sei daher eine nähere Beschäftigung ans Herz gelegt.

Da Python auf Basis des reinen L^AT_EX-Quelltextes keine Information hat, auf welchen Seiten ein Stichwort erwähnt wird, muss dies auf L^AT_EX-Ebene weggeschrieben werden.

Dazu bedienen wir uns des Befehls `\newwrite`, der eine neue Ausgabedatei öffnet. In der Präambel des T_EX-Dokuments muss aufgeführt werden, dass eine neue Datei geschrieben wird, die den gleichen Namen hat, aber die Endung `.sti`. In diese Datei werden alle Stichwörter und die dazu gehörende Seitenzahl geschrieben, die mit dem Makro `\eintrag{Stichwort}` definiert sind.

```
\newwrite\myfile
\immediate\openout\myfile=\jobname.sti
\newcommand{\eintrag}[1]{\immediate\write\myfile{#1:\thepage}}
```

Für die spätere gesammelte Anzeige der Stichwörter und die entsprechende Seitenzahl müssen zwei weitere Befehle definiert werden:

```
\newcommand{\ausgabe}[2]{#1 was referenced on page(s): #2}
\newcommand{\stichworttabelle}{\IfFileExists{\jobname.tab}{\input{\jobname.tab}}{
↪Tab file not found}}
```

Die erste sorgt dafür, dass bei dem Makro `\ausgabe{Stichwort}{ }` das Stichwort sowie der Text *was referenced on page(s):* mit den Seitenzahlen gesetzt wird. Der zweite Befehl gibt die Stichwörter und die Seitenzahlen als Tabelle aus.

Zusammengesetzt sieht der Text in der Präambel wie folgt aus, wobei die Ausgabe des Makros `\ausgabe` zugleich für das konkrete Beispiel abgeändert wurde.

```
1 \newwrite\myfile
2 \immediate\openout\myfile=\jobname.sti
3 \newcommand{\eintrag}[1]{\immediate\write\myfile{#1:\thepage}}
```

⁴<http://pandas.pydata.org>

```

4 \newcommand{\ausgabe}[2]{#2}
5 \newcommand{\stichworttabelle}{\IfFileExists{\jobname.tab}{\input{\jobname.tab
  ↵}{Tab file not found}}

```

Schauen wir uns nun den Python-Code dazu an. Das folgende Listing enthält als Auszug nur die `process`-Funktion aus der Klasse, den kompletten Quellcode mit einem ausführlich dokumentierten Beispiel findet man im bereits erwähnten Github-Repository. Die durch L^AT_EX erstellte `<Stichwort:Seite>` Datei wird hier über den Befehl `read_csv` in einen pandas-DataFrame geladen. Diesen DataFrame kann man sich als Tabelle mit zwei Spalten vorstellen, in einer Spalte die Stichwörter, in der anderen die jeweilige Seitenzahl, auf der das Stichwort genannt wurde.

Die Zeile mit `»result = df.groupby«` enthält recht komplexe Logik. Hier wird nach dem jeweiligen Stichwort gruppiert und ein String erstellt, der die Seitenzahlen, per Komma und Leerzeichen getrennt, enthält. Dabei werden Seiten, auf denen ein Stichwort mehrfach erscheint, auch nur einmal ausgegeben. Was von diesem Befehl zurückgegeben wird, wird dann wieder in einen pandas-DataFrame verwandelt, um leichter damit arbeiten zu können. Über eine weitere pandas-Funktion wird dieser DataFrame dann in eine L^AT_EX-Tabelle geschrieben, die dann direkt in das Dokument importiert werden kann (über den selbstdefinierten Befehl `\stichworttabelle`). Zum Schluss werden in der L^AT_EX-Datei mittels eines regulären Ausdrucks die `\ausgabe`-Befehle gesucht und die in den Befehlen vorhandenen Seitenzahlen durch die (unter Umständen) aktuellere Version des Strings im DataFrame ersetzt.

Listing 9: Auszug aus dem Python-Code

```

def process(self, jobname):
    # Lies die Datei im Stichwort:Seite Format
    df = pd.read_csv(jobname+'.sti', sep=':', names=['Stichwort', 'Seite'])

    # Erstelle die Liste der einzigartigen Stichwörter
    result = df.groupby(['Stichwort'], as_index=False)['Seite'].agg(
        lambda x: ', '.join(np.unique(x).astype(str)))

    # Erstelle einen pandas Dataframe aus der Datenserie
    result = pd.DataFrame(result)

    # setze den Index auf die Spalte 'Stichwort'
    # wird später als Schlüsselfeld genutzt
    result = result.set_index('Stichwort')

    # exportiere den pandas DataFrame in eine LaTeX Tabelle
    result.to_latex(jobname+'.tab', index=True)

```



```
# Gehe durch die Liste der Stichwort/Seiten Tupel und
# suche in der TeX-Datei den \Ausgabe{<Stichwort>}{<Seiten>}
# und ersetze den <Seiten> Teil durch die aktuelle Version
for index, row in result.iterrows():
    suchmuster = re.compile('(\\|\\|\\|\\|ausgabe{'+ index + '})([0-9, ]*)')
    self.regReplace(jobname+'.tex', suchmuster,
        '\\|\\|\\|\\|ausgabe{'+ index + '}' + row[0] + ')')
    print(index, ':', row[0])
```

Integration

Im L^AT_EX-Dokument muss nun an allen Stellen, an denen das Stichwort auftaucht oder der Rückverweis stattfinden soll, der Befehl `\eintrag{Stichwort}` gesetzt werden. Wenn alle Stichwörter gesetzt sind, kann man die Art der Ausgabe wählen. Entweder werden alle Seitenzahlen zu einem Stichwort angezeigt, dies erfolgt über `\ausgabe{Stichwort}{}` oder Stichwörter und Seiten werden mit `\stichworttable` tabellarisch aufgelistet.

Für unser konkretes Beispiel der Katalogeinträge bedeutet dies, dass das Makro `\ausgabe` in die Katalogeintragdefinition integriert werden muss.

```
\define@key{catalogue}{backref}{\def\KVbackref{#1}}
```

Damit wird der Wert, der im `\catalogueentry` unter `backref` angegeben ist (hier *Haus*), an den Befehl `\ausgabe` übergeben:

```
1 \catalogueentry{%
2   backref={\ausgabe{Haus}}},
3 }
```

Das bedeutet selbstverständlich, dass im Fließtext entsprechend `\eintrag{Haus}` gesetzt werden muss. Wenn etwas im `backref` des Katalogeintrags steht, dann wird dieser Eintrag übernommen, ansonsten wird geschaut, ob das `label` definiert ist, wenn auch dies nicht der Fall ist, bleibt dieser Katalogeintrag undefiniert und es findet auch keine Weiterverarbeitung statt.

Für die Darstellung des Katalogeintrages muss ebenfalls eine Neudefinition eingeführt werden:

```
\ifdef{\KVbackref}{\item[Erwähnungen] \KVbackref}{}
```

Ein vollständiger Katalogeintrag:

```
1 \catalogueentry{%
2   house={Haus des M. Fabius Rufus},
```

```

3  label={haus:M-Fabius-Rufus},
4  size={172},
5  description={Haus besteht aus mehreren Einzelgebäuden.},
6  location={Regio VII, Insula 16, Eingang 17--22.},
7  interior={Reicher Fundkomplex.},
8  backref={\ausgabe{Haus-Fabius}{}},
9  }

```

Nun wird *Haus* an `\ausgabe` übergeben. Im folgenden Fall ist dies der Wert vom Katalogeintrag `label`:

```

1  \catalogueentry{%
2  house={Haus des Wilden Ebers},
3  label={Haus-des-Wilden-Ebers},
4  size={54},
5  description={Renovierung nach Erdbeben 62\n.\,Chr.},
6  location={Regio VII, Insula 4, Eingang 48, 43.},
7  }

```

Während im ersten Fall im Fließtext an gewünschter Stelle `\eintrag{Haus-Fabius}` gesetzt werden muss, gilt für die zweite Variante `\eintrag{Haus-des-Wilden-Ebers}`.

Haus des M. Fabius Rufus

Beschreibung Haus besteht aus mehreren Einzelgebäuden.

Verortung Regio VII, Insula 16, Eingang 17–22.

Ausstattung Reicher Fundkomplex.

Größe 172 m²

Erwähnungen S.

Haus des Wilden Ebers

Beschreibung Renovierung nach Erdbeben 62 n. Chr.

Verortung Regio VII, Insula 4, Eingang 48, 43.

Größe 54 m²

Erwähnungen S.

Sobald diese Vorarbeiten getan sind, alle Stichwörter entsprechend mit `\eintrag` gesetzt sind, dann muss einmal (bspw. mit X_YL^AT_EX) kompiliert werden, anschließend

der Python-Code im entsprechenden externen Programm ausgeführt und schließlich zurück im T_EX-Editor wiederum einmal kompiliert werden.

Problemlösung mit T_EX

Bei einer Lösung über T_EX muss beachtet werden, dass die jeweiligen Seitenzahlen der Referenzen eventuell schon benötigt werden, wenn der eigentliche Referenzbefehl noch gar nicht ausgeführt wurde. Dies entspricht faktisch dem gleichen Vorgang wie bei der Erstellung eines Inhaltsverzeichnisses. Da dieses im Allgemeinen am Anfang des Dokumentes steht, kann es auch erst nach einem zweiten Durchlauf ausgegeben werden. Die gleiche Vorgehensweise wird für die Verweise gewählt:

- Erstellen einer eigenen externen Referenzdatei.
- Beim Programmstart Einlesen dieser Datei, sofern sie von einem vorhergehenden Lauf vorhanden ist.
- Interpretieren der in der Datei vorhandenen Referenzbefehle.
- Löschen der Datei und erneutes Schreiben der Referenzen in diese, um Änderungen zu erfassen.
- Am Ende schließen und speichern der externen Datei.

Die externe Datei bekommt den Namen `\jobname.ctg`:

```
\newwrite\catalog%   Neues write-Register
\inputIfFileExists{\jobname.ctg}{}{}% Datei einlesen, falls schon vorhanden
\immediate\openout\@catalog\jobname.ctg% Zum Überschreiben öffnen
```

Das heißt, dass man im Gegensatz zur Python-Variante die Rückverweise nicht manuell setzen muss, sondern sie werden automatisch über die im Fließtext gesetzten `\labelcref` erstellt. Ein entsprechender Eintrag sieht dann wie folgt aus:

```
\catalogueentry{%
  house=Haus des Wilden Ebers,
  label=haus:Haus-des-Wilden-Ebers,
  size=54,
  description={Renovierung nach Erdbeben 62\,n.\,Chr.},
  location={Regio VII, Insula 4, Eingang 48, 43},
  interiorM=S/W-Mosaik,
  interiorW={Venus, Leda und der Schwan, Ariadne und Theseus},
  mark%   Gib Seitenzahlen aus
}
```

Allerdings muss das Makro `\labelcref` modifiziert werden, sodass es zusätzlich die entsprechenden Einträge in die externe Datei schreiben kann:

```

\let\LabelCref\labelcref
\renewcommand\labelcref[1]{\expandafter\label@cref#1,,\@nil}
\def\label@cref#1,#2,#3\@nil{%
  \immediate\write\@catalog{\string\catalog@ref{#1-pages}{\arabic{page}}}
  \LabelCref{#1}%
  \ifx\relax#2\relax % kein label mehr da
    \def\next{%
  \else
    \def\next{\label@cref#2,#3\@nil}%
  \fi
  \next}
\AtEndDocument{\closeout\@catalog}

```

Die Datei `\jobname.ctg` könnte dann beispielsweise folgende Einträge enthalten:

```

\catalog@ref{haus:Haus-des-Wilden-Ebers-pages}{2}
\catalog@ref{haus:M-Fabius-Rufus-pages}{3}
\catalog@ref{haus:Haus-des-Wilden-Ebers-pages}{3}
\catalog@ref{haus:M-Fabius-Rufus-pages}{4}
\catalog@ref{haus:Haus-der-Venus-pages}{4}
\catalog@ref{haus:Haus-des-Wilden-Ebers-pages}{4}
\catalog@ref{haus:Haus-der-Venus-pages}{5}

```

Diese werden durch die entsprechende Definition von `\catalog@ref` interpretiert:

```

\def\catalog@ref#1#2{%
  \expandafter\ifx\csname#1\endcsname\relax % schon definiert??
    \@namedef{#1}{#2}% Nein, also erstellen
  \else% anderenfalls Seitenzahl an existierenden Befehl anhängen
    \expandafter\edef\csname#1\endcsname{\csname#1\endcsname, #2}%
  \fi}

```

Es existieren danach jeweils Makros, die für die zugehörigen Seitenzahlen stehen. Beispielsweise enthält das Makro `\haus:Haus-des-Wilden-Ebers-pages` als Wert die Seitenangaben 2, 3, 4. Auf diese Makros muss dann einfach bei den Katalogeinträgen Bezug genommen werden:

```

\newcommand\catalogueentry[1]{%
  ...
  \ifKVmark \item[Erwähnungen] S.~\expandafter\csname\KVlabel-pages\endcsname \fi
  ...
}

```

Die Seiten, auf denen über ein `\labelcref` auf ein Katalogeintrag verwiesen wird, werden nun beim jeweiligen Katalogeintrag eingetragen und bei jedem *zweiten*

Kompilieren auf den aktuellen Stand gebracht, insofern sich zwischenzeitlich etwas geändert hat. Als Beispiel sei die erste Seite eines Dokumentes ausgegeben, welches alle Verweise grundsätzlich erst nach der Definition der einzelnen Katalogeinträge vornimmt.

Zusammenfassung

Ausgehend von dem Wunsch, einen Katalog in einer wissenschaftlichen Arbeit möglichst effizient zu gestalten, wurde die Lösung über die Definition verschiedener keys präsentiert, die eine einheitliche Programmierung der Eingabe und Gestaltung der Ausgabe ermöglichen.

Zwei Möglichkeiten wurden vorgestellt, wie man einen Rückverweis im jeweiligen Katalogteil auf die Seite mit der Erwähnung einbauen kann. Die erste Lösung arbeitet mit einem Python-Code, bei dem man den Marker `\eintrag{Stichwort}` im Text selbst setzen kann und bei dem die Seitenzahl über `\ausgabe{Stichwort}` im Katalogeintrag ausgegeben wird.

Die zweite Möglichkeit arbeitet mit TEX -eigenen Mitteln, indem es die `\labelcref` nutzt, mit denen auf die Katalogeinträge verwiesen wird. Diese Variante erfordert kein händisches Setzen der Marker im Text, führt dann jedoch nur die Seiten auf, auf denen auch ein entsprechendes `\labelcref` gesetzt ist.

Für beide Lösungen wurde ein Minimalbeispiel erstellt, sodass man diese Varianten ausprobieren kann.⁵

Literatur und Software

- [1] Uwe Ziegenhagen: »Größere Dokumente mit $\text{L}^{\text{A}}\text{T}\text{E}\text{X}$ erstellen«, *Die TEX nische Komödie*, 1 (Feb. 2015), 25–29.

⁵Für den Python-Code siehe <https://github.com/LukasCBossert/DTK-TeX-Python/blob/master/catalogueentry-python.tex>, für den TEX -Code siehe <https://github.com/LukasCBossert/DTK-TeX-Python/blob/master/catalogueentry-tex.tex>.

1 Katalogteil

Kat. A.1 Haus des M. Fabius Rufus

Beschreibung Haus besteht aus mehreren Einzelgebäuden.

Verortung Regio VII, Insula 16, Eingang 17–22.

Ausstattung Reicher Fundkomplex.

Mosaike: S/W-Mosaik

Wandgemälde: Dionysius mit einer Mänade, Narzissus und ein Cupido, Hercules und Deinira, etc.

Statuen: Bronzene Statue eines Epheben

Größe 172 m²

Erwähnung S. 2, 3

Kat. A.2 Haus des Wilden Ebers

Beschreibung Renovierung nach Erdbeben 62 n. Chr.

Verortung Regio VII, Insula 4, Eingang 48, 43

Ausstattung Mosaike: S/W-Mosaik

Wandgemälde: Venus, Leda und der Schwan, Ariadne und Theseus

Größe 54 m²

Erwähnung S. 1, 2, 3

Kat. A.3 Haus der Venus

Beschreibung Schönes großes Haus

Verortung Regio VII, Insula 4, Eingang 12, 23

Ausstattung Keine Innenausstattung

Größe 123 m²

Erwähnung S. 3, 4

2 Text

In meiner Arbeit beschäftige ich mit drei Häusern in Pompeji (Kat. A.2).