# Integration von Python in TEX am Beispiel von Katalogeinträgen

Lukas C. Bossert, Uwe Ziegenhagen

Viele Dissertationen in der Archäologie enthalten am Ende der Arbeit einen Katalog, in dem die untersuchten Daten in einem bestimmten System aufgeschlüsselt präsentiert werden. Ein solcher Katalog kann aus Bildern, Bohrproben, Architekturelementen etc. bestehen.

Eine händische Erstellung der einzelnen Einträge bspw. über `\section` oder `\subsection` und anschließend in einer `items`-Umgebung ist nicht effizent, 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 bspw. 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 TEX überforderte.

Es musste also eine andere Lösung her und ich (Lukas) erinnerte mich an einen Beitrag, den Uwe Ziegenhagen in dtk 1/2015 vorgestellt hatte: Damals ging es darum, bei einem Werkkatalog die Erwähnung des Stückes im Haupttext anzugeben. Prima, genau das, was ich gesucht habe und den damals verwendeten Code fand ich auf seiner Webseite.[1] Allerdings sah ich nun, dass die Aufgabe damals darin bestand, nur *eine* Erwähnung im Haupttext anzugeben, was mit `label` und `ref` bewerkstelligt werden konnte. Ich brauche jedoch alle Erwähnungen im Haupttext. Meine E-Mail-Anfrage bei Uwe nach einer Möglichkeit, mehr als nur eine Erwähnung anzeigen zu lassen, beantwortete er mit dem Vorschlag, mittels Python das Problem zu lösen.

Um gemeinsam an dem hybriden Konstrukt von XELATEX und Python arbeiten zu können, wurde ein Repository auf `github`[2] erstellt, einer Plattform, die unabhängig von der Programmiersprache einen exzellenten Austausch und eine detaillierte Versionskontrolle ermöglicht.

[1] `http://uweziegenhagen.de/?p=3020`

[2] `https://github.com/LukasCBossert/DTK-TeX-Python`

Während des Schreibprozesses an diesem Artikel und am Python-Code haben wir Schützenhilfe von Herbert Voß erhalten, der uns eine TEX-immanente Lösung erarbeitet hat, die als weitere Möglichkeit zu unserem konkreten Beispiel präsentiert wird.

## Katalogeintrag

Als Aufgabe soll ein Katalog zu den Häusern in Pompeji angelegt werden, 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 `keyval`-Pakets.[3] Dafür werden in der Präambel verschiedene Schlüssel (`keys`) definiert. In der Grundversion sieht die Definition eines Eintrags wie folgt aus:[4]

```
\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

```
\makeatletter
\define@key{catalogue}{house}{\def\KVhouse{#1}}
\define@key{catalogue}{label}{\def\KVlabel{#1}}
\define@key{catalogue}{description}{\def\KVdescription{#1}}
\define@key{catalogue}{location}{\def\KVlocation{#1}}
\define@key{catalogue}{size}{\def\KVsize{#1}}
\define@key{catalogue}{interior}{\def\KVinterior{#1}}
\define@key{catalogue}{interiorM}{\def\KVinteriorM{#1}}
\define@key{catalogue}{interiorW}{\def\KVinteriorW{#1}}
\define@key{catalogue}{interiorS}{\def\KVinteriorS{#1}}
\makeatother
```

[3] Basierend auf der Idee vorgestellt auf: http://tex.stackexchange.com/a/254336/98739

[4] Vgl. http://www.tug.org/tugboat/tb30-1/tb94wright-keyval.pdf

Das Aussehen eines Katalogeintrages wird separat mit dem Makro `\newcommand\catalogueentry[1]…` definiert: 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 Schlüsselfamilie (hier `catalogue`) auszulesen ist.

Listing 2: Definition der Katalogeinträge, Anfang

```
\newcommand\catalogueentry[1]{%
\begingroup
\RaggedRight
\setkeys{catalogue}{#1}
...
```

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

```
...
\ifdef{\KVhouse}{\section{\KVhouse
      \ifdef{\KVlabel}{\label{\KVlabel}}{}}}{}
\begin{labeling}{Beschreibung}
  \ifdef{\KVdescription}{\item[Beschreibung] \KVdescription}{}
  \ifdef{\KVlocation}{\item[Verortung] \KVlocation}{}
  \ifdef{\KVinterior}{\item[Ausstattung] \KVinterior
      \begin{labeling}{Wandgemälde}
        \ifdef{\KVinteriorM}{\item[Mosaike:] \KVinteriorM}{}
        \ifdef{\KVinteriorW}{\item[Wandgemälde:] \KVinteriorW}{}
        \ifdef{\KVinteriorS}{\item[Statuen:] \KVinteriorS}{}
      \end{labeling}
      }{}
  \ifdef{\KVsize}{\item[Größe] \SI{\KVsize}{\meter\squared}}{}
\end{labeling}
\endgroup
}
```

Mit dieser Einstellung lassen sich die Katalogeinträge schon sehr gut darstellen. Im Fließtext des Hauptdokuments kann man an gewünschter Stelle mit `\cata-`

`logueentry` 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. | |
| | Mosaike: | S/W-Mosaik. |
| | Wandgemälde: | Dionysius mit einer Mänade, Narzissus und ein Cupido, Hercules und Deinira etc. |
| | Statuen: | Bronzene Statue eines Epheben. |
| Größe | $172\,\mathrm{m}^2$ | |

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 ($\mathrm{m}^2$) übergeben. Ähnlich kann auch mit anderen Angaben verfahren werden (bspw. bei Abbildungen können die `\label` an ein vordefiniertes Macro `\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-Ebers2},
  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 | $54\,\mathrm{m}^2$ | |

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: Einsetzten der Booleschen Operatoren in die Schlüssel

```
\define@key{catalogue}{interiorM}{\def\KVinteriorM{#1}\booltrue{@KVinteriorM}}
\define@key{catalogue}{interiorW}{\def\KVinteriorW{#1}\booltrue{@KVinteriorW}}
\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: Einsetzten der Booleschen Operatoren in den Katalogeintrag

```
\ifdef{\KVinterior}{%
  \item[Ausstattung] \KVinterior
  \ifboolexpr{bool{@KVinteriorM}
          or bool{@KVinteriorW}
          or bool{@KVinteriorS}}{%
      \begin{labeling}{Wandgemälde}
        \ifdef{\KVinteriorM}{\item[Mosaike] \KVinteriorM}{}
        \ifdef{\KVinteriorW}{\item[Wandgemälde] \KVinteriorW}{}
        \ifdef{\KVinteriorS}{\item[Statuen] \KVinteriorS}{}
      \end{labeling}
      }{}}
  {\ifboolexpr{bool{@KVinteriorM}
          or bool{@KVinteriorW}
          or bool{@KVinteriorS}}{%
    \item[Ausstattung]%
  \begin{labeling}{Wandgemälde}
        \ifdef{\KVinteriorM}{\item[Mosaike] \KVinteriorM}{}
        \ifdef{\KVinteriorW}{\item[Wandgemälde] \KVinteriorW}{}
        \ifdef{\KVinteriorS}{\item[Statuen] \KVinteriorS}{}
      \end{labeling}
  }{}}
```

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 TEX-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

Tabulator-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. `pandas`, ursprünglich zur Analyse von Finanzdaten entwickelt, erleichtert die Arbeit mit strukturierten Daten aller Art. Das Einlesen, Filtern und Aggregieren von Daten jeder Art geht damit erstaunlich einfach, dem interessierten Leser sei daher eine nähere Beschäftigung ans Herz gelegt.

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

Dazu bedienen wir uns des `\newwrite` Befehls, der eine neue Ausgabedatei öffnet. In der Präambel des TEX-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 `\ausgabe`{Stichwort}{ } Makro 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 `\ausgabe` Makros 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 \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 einen entsprechenden Auszug aus der Datei, den kompletten Quellcode mit einem ausführ-

lich dokumentierten Beispiel findet der geneigte Leser unter `www.uweziegenhagen.de/?p=3513`. Die durch LaTeX erstellte `‹Stichwort:Seite›` Datei wird hier über den `read_csv` Befehl 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 »`result =`« Zeile 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 Befehlt 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 LaTeX-Tabelle geschrieben, die dann direkt in das Dokument importiert werden kann (über den selbstdefinierten `\stichworttabelle` Befehl). Zum Schluss werden in der LaTeX-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 eine 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 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 LaTeX-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 `\stichworttabelle` tabellerisch 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:

```
\catalogueentry{%
  backref={\ausgabe{Haus}{}},
}
```

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:

```
\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.},
  backref={\ausgabe{Haus-Fabius}{}},
}
```

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

```
\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.},
}
```

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 | $172\,\mathrm{m}^2$ |
| Erwähnungen | S. |

### Haus des Wilden Ebers

| | |
|---|---|
| Beschreibung | Renovierung nach Erdbeben 62 n. Chr. |
| Verortung | Regio VII, Insula 4, Eingang 48, 43. |
| Größe | $54\,\mathrm{m}^2$ |
| Erwähnungen | S. |

Sobald diese Vorarbeiten getan sind, alle Stichwörter entsprechend mit `\eintrag` gesetzt sind, dann muss einmal (bspw. mit XeLaTeX) kompiliert werden, anschließend der Python-Code im entsprechenden externen Programm ausgeführt und schließlich zurück im TEX-Editor wiederum einmal kompiliert werden.

## Problemlösung mit TEX

Die andere, TEX-immanente Lösung von Herbert Voß arbeitet über `\labelcref`. Das heißt, dass man im Gegensatz zur Python-Variante die Rückverweise nicht manuell setzen kann, sondern sie werden automatisch über die im Fließtext gesetzten

`\labelcref` erstellt. Dafür muss der Code des Katalogeintrages `backref` geändert werden:

Listing 10: Modifkation im Katalogeintrag

```
\ifKVmark \item[Erwähnungen] S.~\expandafter\csname\KVlabel-pages\endcsname \
 ↪fi
  ...
\define@boolkey{catalogue}[KV]{backref}[true]{}
```

Zudem muss das Makro `\labelcref` bearbeitet werden, sodass TEX die gesetzten Einträge auslesen kann.

Listing 11: Ergänzung in der Präambel

```
\let\LabelCref\labelcref
\renewcommand\labelcref[1]{\expandafter\label@cref#1,,\@nil}
\def\label@cref#1,#2,#3\@nil{%
  \expandafter\ifx\csname#1-pages\endcsname\relax %
    \expandafter\xdef\csname#1-pages\endcsname{\arabic{page}}%
  \else
    \expandafter\xdef\csname#1-pages\endcsname{\csname#1-pages\endcsname, \
 ↪arabic{page}}%
  \fi
  \LabelCref{#1}%
  \ifx\relax#2\relax %
    \def\next{}%
  \else
    \def\next{\label@cref#2,#3\@nil}%
  \fi
  \next}
```

Bei den Katalogeinträgen reicht es, am Ende `backref` einzutragen (Listing 12):

Listing 12: Ergänzung im Katalogeeintrag

```
\catalogueentry{%
...
  backref,
}
```

Für das aufgeführt Beispiel wurde die `\section` des Katalogteils geändert, damit die `\labelcref`-Einträge das Kürzel ›Kat. A.1‹, etc. bekommen (Listing 13).

Listing 13: Modifkation der Namensgebung für Katalogeeinträge

```
\section{Katalogteil}
\renewcommand\thesection{Kat. \Alph{section}}
```

Die Seiten, auf denen über ein `\labelcref` auf ein Katalogeintrag verwiesen wird, werden nun beim jeweiligen Katalogeintrag eingetragen und bei jedem Kompilieren ggf. auf den aktuellen Stand gebracht.

## 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 Minimales Working Example erstellt, sodass man diese Varianten ausprobieren kann.[5]

[5] Für 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`.