Spendenbescheinigungen erstellen mit LaTeX, SQL und Python

Uwe Ziegenhagen

In der Funktion als Kassenwart der Kölner Dingfabrik e. V. muss ich spätestens im Frühjahr eine Vielzahl von Spendenbescheinigungen erstellen. Der bisher gelebte Prozess, die händische Aggregation der Daten in Excel und Fertigstellung von MS Word Schreiben, war keine Option für einen überzeugten TeXie; eine automatisierte Lösung musste gefunden werden.

Im folgenden Artikel beschreibe ich den Workflow der Erstellung der Spendenformulare mittels LaTeX sowie ihre Befüllung aus einer Datenbank über selbstgeschriebener Python-Skripte.

Erstellung der Formulare

Seit dem 01. Januar 2013 sind vom Bundesministerium für Finanzen neue Muster für Zuwendungsbestätigungen[1] vorgeschrieben.

Auf den offiziellen Webseiten[2] werden leider nur Word- und PDF-Vorlagen angeboten, sodass die Vorlagen in LaTeX nachgebaut werden müssen.

Als Alternative könnte man zwar die PDF Formulare als Hintergrundbild in einer LaTeX-Datei nutzen und beispielsweise mit den Befehlen des `eso-pic` Pakets [**esopic**] entsprechende Textteile auf der Seite positionieren [**forms**], ich wollte aber die Dokumente aber von Grund auf neu umsetzen. Einerseits geschah dies, um eine wirklich saubere LaTeX-Lösung zu haben und andererseits, um etwas mehr über die automatisierte Erstellung von Dokumenten zu lernen.

Inhaltlich unterscheiden muss man zwischen drei verschiedenen angebotenen Vorlagen, die sich aber zu großen Teilen überlappen:

- **Zuwendungsbestätigung - Geldzuwendung** für einmalige monetäre Spenden
- **Zuwendungsbestätigung - Sachzuwendung** für Sachspenden

[1] Beamtendeutsch für »Spendenquittungen«

[2] `http://www.finanzamt.bayern.de/Informationen/Formulare/Weitere_Themen_A_bis_Z/Spenden/default.php`

- **Sammelbestätigung über Geldzuwendungen / steuerbegünstigte Einrichtung** für aggregierte Spendenbescheinigungen von z. B. Mitgliedsbeiträgen

Da das grundsätzliche Vorgehen bei allen Dokumententypen identisch ist, werde ich mich im folgenden auf die Sammelbestätigung konzentrieren, die gezeigte Vorgehensweise ist dann auf die anderen Vorlagen übertragbar.

Aufbau des Formulars

Bei der Analyse des Formulars, siehe Abbildung 1, sah man eigentlich nichts, was LaTeX grundsätzlich *nicht* konnte. Einzelne Details erwiesen sich dann jedoch doch als trickreich.

Da das Design soweit nur irgendwie möglich den originalen Formularen der Finanzverwaltung entsprechen sollte, wollte ich auch die Boxen mit den Beschreibungen in den linken oberen Ecken originalgetreu setzen.

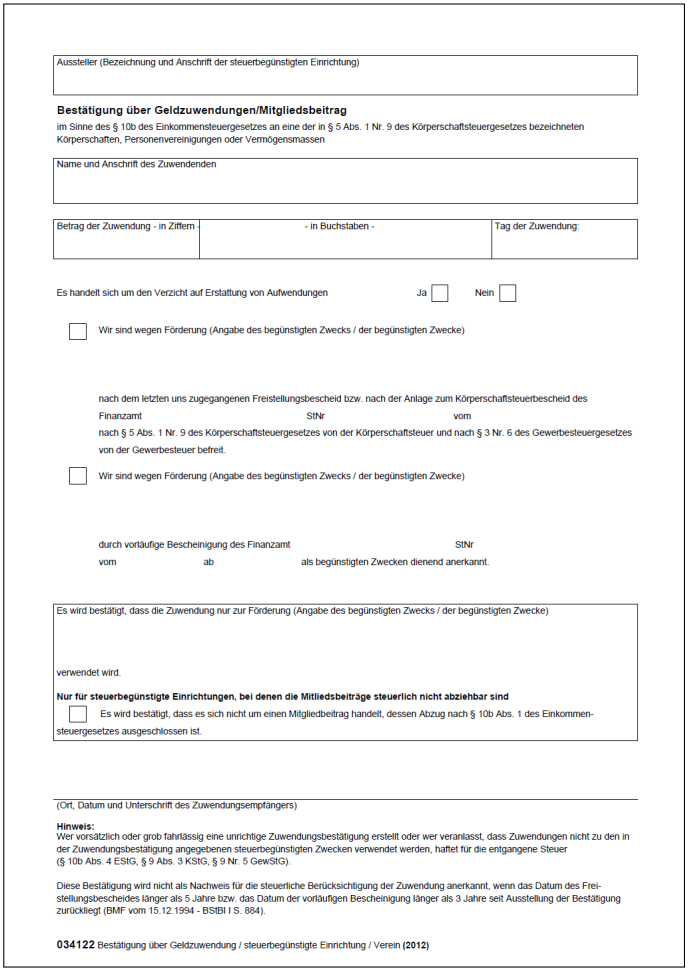
Die Lösung fand sich wie so oft bei `tex.stackexchange.com`[3], in der Peter Grill mit Hilfe des `mdframed` Pakets [**mdframed**] von Marco Daniel eine entsprechende Box kreierte. Listing 1 zeigt den Quellcode eines einfachen Beispiels.

```
\documentclass {article }
\usepackage [a6paper ]{geometry }
\usepackage [utf 8]{ inputenc }
\usepackage [T1]{ fontenc }
\usepackage {mdframed }

\mdfdefinestyle {MyFormStyle }{%
    linewidth =1.25 pt ,
    skipbelow =\topskip ,
    skipabove =\topskip
}

\newcommand {\MyFormBox }[3][1.0 cm]{%
    \begin {mdframed }[style =MyFormStyle ]%
    {\noindent \footnotesize #2 \vspace *{1 em}%
    \par \normalsize  #3}\ vspace *{#1} %
    \end {mdframed }%
}
```

[3] `http://tex.stackexchange.com/questions/111079/creating-form-boxes-with-labels`



Aussteller (Bezeichnung und Anschrift der steuerbegünstigten Einrichtung)

Bestätigung über Geldzuwendungen/Mitgliedsbeitrag

im Sinne des § 10b des Einkommensteuergesetzes an eine der in § 5 Abs. 1 Nr. 9 des Körperschaftsteuergesetzes bezeichneten Körperschaften, Personenvereinigungen oder Vermögensmassen

Name und Anschrift des Zuwendenden

Betrag der Zuwendung - in Ziffern -	- in Buchstaben -	Tag der Zuwendung:

Es handelt sich um den Verzicht auf Erstattung von Aufwendungen Ja ☐ Nein ☐

☐ Wir sind wegen Förderung (Angabe des begünstigten Zwecks / der begünstigten Zwecke)

nach dem letzten uns zugegangenen Freistellungsbescheid bzw. nach der Anlage zum Körperschaftsteuerbescheid des Finanzamt StNr vom
nach § 5 Abs. 1 Nr. 9 des Körperschaftsteuergesetzes von der Körperschaftsteuer und nach § 3 Nr. 6 des Gewerbesteuergesetzes von der Gewerbesteuer befreit.

☐ Wir sind wegen Förderung (Angabe des begünstigten Zwecks / der begünstigten Zwecke)

durch vorläufige Bescheinigung des Finanzamt StNr
vom ab als begünstigten Zwecken dienend anerkannt.

Es wird bestätigt, dass die Zuwendung nur zur Förderung (Angabe des begünstigten Zwecks / der begünstigten Zwecke)

verwendet wird.

Nur für steuerbegünstigte Einrichtungen, bei denen die Mitliedsbeiträge steuerlich nicht abziehbar sind

☐ Es wird bestätigt, dass es sich nicht um einen Mitgliedbeitrag handelt, dessen Abzug nach § 10b Abs. 1 des Einkommensteuergesetzes ausgeschlossen ist.

(Ort, Datum und Unterschrift des Zuwendungsempfängers)

Hinweis:
Wer vorsätzlich oder grob fahrlässig eine unrichtige Zuwendungsbestätigung erstellt oder wer veranlasst, dass Zuwendungen nicht zu den in der Zuwendungsbestätigung angegebenen steuerbegünstigten Zwecken verwendet werden, haftet für die entgangene Steuer (§ 10b Abs. 4 EStG, § 9 Abs. 3 KStG, § 9 Nr. 5 GewStG).

Diese Bestätigung wird nicht als Nachweis für die steuerliche Berücksichtigung der Zuwendung anerkannt, wenn das Datum des Freistellungsbescheides länger als 5 Jahre bzw. das Datum der vorläufigen Bescheinigung länger als 3 Jahre seit Ausstellung der Bestätigung zurückliegt (BMF vom 15.12.1994 - BStBl I S. 884).

034122 Bestätigung über Geldzuwendung / steuerbegünstigte Einrichtung / Verein **(2012)**

Abb. 1: Spendenformular, Quelle: `http://www.finanzamt.bayern.de`

```
\begin{document}
\MyFormBox [1.0 cm]{Headline }{Inhalt  der  Box}
\end{document}
```

Listing 1: `mdframed` Beispiel, Ausgabe siehe Abbildung 2 auf Seite 4

Eine weitere Herausforderung war das Ausschreiben des gespendeten Betrags als Wort. Hier hätte Nicola Talbots `fmtcount` Paket [**fmtcount**] sehr nützlich

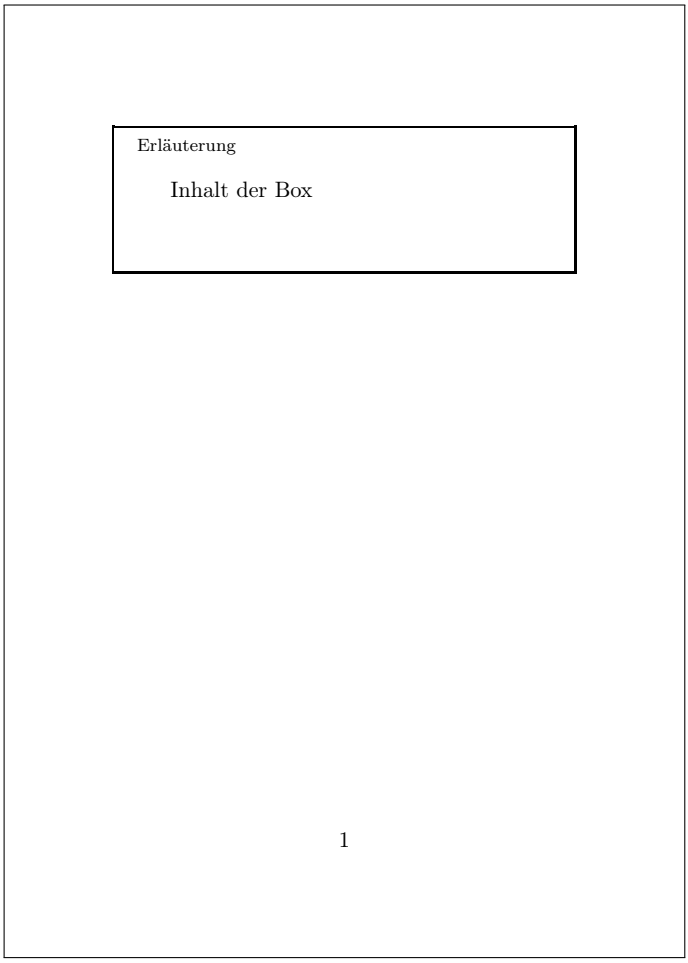


Abb. 2: Ergebnis von Listing 1

können, dessen `\Numberstringnum` Befehl eine ganze Zahl (ohne Dezimalteil) entgegennimmt und die textliche Repräsentation als Zeichenkette zurückgibt. Leider kann das Paket mit den im Deutschen anzutreffenden Sprach-Feinheiten nicht umgehen, so wird aus »123« »Einshundertdreiundzwanzig« statt »Einhundertdreiundzwanzig«.

Die finale Lösung bestand dann darin, eine weitere Tabelle in der Datenbank mit fertigen Zahlwörtern für jeden ganzzahligen Betrag zwischen 1 und 9999 zu befüllen und diese weiteren Programmablauf auszuwerten.

Erstellung der Datenbankabfragen

Nach der Erstellung des Layouts war es nun Zeit, die gewohnten LaTeX-Pfade zu verlassen, um die Daten für die Spendenquittungen aufzubereiten.

Zum Skripten der verschiedenen benötigten Funktionen wurde Python genutzt, das sich bei mir persönlich durch einen hohen Funktionsumfang bei guter Lesbarkeit des Quellcodes und leichter Erlernbarkeit beliebt gemacht hat.

Die Buchungen der Dingfabrik werden in Lexware Quicken verwaltet, das neben seiner wichtigsten Funktion – dem Abholen der Kontoauszüge von der Bank mittels HBCI – verschiedene Möglichkeiten zur Klassifikation und Auswertung der Daten bietet.

Aus Quicken werden die klassifizierten Buchungen dann für die Weiterverarbeitung exportiert. Der CSV-Export[4] aus Quicken heraus war leider unbrauchbar, da die Daten im CP1252 (Latin1) Encoding ausgegeben wurden und erste Versuche, die Daten mittels Python einzulesen, in defekten Umlauten resultierten.

Der Excel-Export war deutlich besser zu nutzen, da Excel schon seit mehreren Version alle Daten in Unicode abspeichert. Wirklich »perfekt« ist jedoch auch dieser Export nicht, da Quicken in den ersten und letzten Zeilen des entsprechenden Arbeitsblattes verschiedene Meta-Informationen einfügt, die sich auch nicht wegkonfigurieren lassen.

```
import xlrd
workbook = xlrd.open_workbook('Buchungen_20131129.xlsx')
worksheet = workbook.sheet_by_name('Sheet')
num_rows = worksheet.nrows - 10
num_cells = worksheet.ncols - 1
curr_row = 8

while curr_row < num_rows:
  curr_row += 1
  datum = str(worksheet.cell_value(curr_row, 1))
  beschreibung = worksheet.cell_value(curr_row, 5)
  zweck = worksheet.cell_value(curr_row, 5)
  kategorie = worksheet.cell_value(curr_row, 6)
```

[4] Comma-Separated Values

```
  klasse  = worksheet.cell_value(curr_row, 7)
  betrag  = worksheet.cell_value(curr_row, 9)
  print(datum, " ", beschreibung," ", zweck," ", kategorie," ",
↪ betrag)
```

Listing 2: Python Quellcode zum Auslesen von Excel-Daten

Das von mir genutzte Python-Modul `xlrd` [**xlrd**] verfügt jedoch über entsprechende Funktionen, den auszulesenden Teil einer Excel-Datei genau zu definieren. Listing 2 zeigt ein Beispiel von der `xlrd`-Webseite, das etwas angepasst wurde, um die Daten aus der exportierten Quicken-Datei »Buchungen.xlsx« auszulesen.

Das Auslesen der Daten aus Excel ist jedoch nur der erste Schritt, im nächsten Schritt werden die ausgelesenen Daten in eine Datenbank geschrieben, um die Daten mit SQL-Abfragen passend aufzubereiten und zu extrahieren.

Anfänglich habe ich SQLite genutzt, da Python dieses nicht nur standardmäßig mitbringt, es leicht zu konfigurieren ist, und sogenannte »In-Memory« Datenbanken unterstützt. »In-Memory« bedeutet, dass keine Datenbank-Datei auf der Dateiablage angelegt werden muss, alle Daten landen im Arbeitsspeicher.

Dies bedeutet zwar, dass sie nach dem Lauf des Python-Skripts verloren sind, da die Datenmenge in meinem Anwendungsfall überschaubar groß ist und keine Persistenz der Daten über den Python-Lauf hinweg benötigt wird, war »In-Memory« hier vollkommen ausreichend.

Im Laufe des Projekts kam noch der Wunsch nach der Verbindung zu den Stammdaten auf, die in MySQL gehalten werden, aus Gründen der Vereinheitlichung wurde daher auf MySQL umgestellt. Als Datenbank-Treiber diente das offizielle von Oracle angebotene Python Modul[5].

```
import mysql.connector

db = mysql.connector.connect(host="localhost",user="root",
                        passwd="uweuwe", db="TestDB")

cur = db.cursor()
cur.execute("insert into test values('Max','Hase')")
db.commit()
cur.execute("SELECT * FROM test")

for row in cur.fetchall():
    print (row)
```

[5] `http://dev.mysql.com/downloads/connector/python`

Listing 3: Python Quellcode für eine MySQL Datenbank

Listing 3 zeigt ein einfaches Beispiel, wie man mit Python Daten in eine MySQL Datenbank schreiben und auch wieder auslesen kann.

Mehr zum Thema SQLite und MySQL findet man beispielsweise unter [**sqlite**] oder in [**pypro**].

Über eine Kombination der Skripte aus den Listings 2 und 3 wurden dann die Kontoauszüge aus der Excel-Datei in die MySQL Datenbank geladen, weitere SQL Statements holen die Daten dann auch wieder aus der Datenbank, um sie in die LaTeX Dokumente einzufügen.

Erzeugen der Dokumente

Für die Kombination des LaTeX-Dokuments mit den SQL-Daten nutze ich `Jinja2`, eine auf Python basierende Template-Engine.

Template Engines machen nichts anderes, als in einer Vorlage bestimmte Platzhalter mit Werten zu ersetzen. Dies lässt sich zwar auch mit Bordmitteln erreichen, Template Engines sind aber üblicherweise deutlich schneller und der Quellcode bleibt eleganter und übersichtlicher.

Python besitzt zwar seit Version 2.4 auch eine eingebaute Template Engine, Jinja2 bietet jedoch noch einige nützliche Zusatzfunktionen wie eine eingebaute Skriptsprache, mit der z. B. Aufzählungen oder Tabellenzeilen recht einfach gesetzt werden können.

```
import  jinja2
import  os

latex_jinja_env   = jinja2.Environment (
    block_start_string   = '\BLOCK{',
    block_end_string   = '}',
    variable_start_string   = '\VAR{',
    variable_end_string   = '}',
    comment_start_string   = '\#{',
    comment_end_string   = '}',
    line_statement_prefix   = '%-',
    line_comment_prefix   = '%#',
    trim_blocks   = True,
    autoescape   = False,
    loader   = jinja2.FileSystemLoader  (os.path.abspath ('.'))
)
```

Listing 4: `Jinja2` Beispiel

Jinja2 nutzt standardmäßig doppelte geschweifte Klammern zur Kennzeichnung von Platzhaltern, die bekanntermaßen auch in LaTeX eine zentrale Rolle spielen. Listing 4 zeigt daher ein entsprechendes Minimalbeispiel, das zuerst LaTeX-Kompatibilität[6] sowie die Funktionalität zum Laden von Vorlagen aus Dateien herstellt.

Aus der Beispieldatei von Listing 5 wird beim Python-Lauf der LaTeX-Code von Listing 6 erzeugt, Damit haben wir nun alles an Funktionalität, was wir im weiteren Verlauf für unsere LaTeX-Dokumente benötigt wird.

Die Vorlage für die Spendenquittung wird im nächsten Schritt um die Jinja2-Variablen erweitert und mit dem Datenbank-Code verbunden.

```
\section {\VAR{headline }}

\begin{itemize }
\BLOCK{for  item  in  liste }
 \item  \VAR{item }
\BLOCK{endfor }
\end{itemize }
```

Listing 5: Beispieldatei `test-itemize.tex` für Jinja2

```
\section {Hello }

\begin{itemize }
 \item  first
 \item  second
 \item  third
\end{itemize }
```

Listing 6: Ausgabe von Listing 4, generierter LaTeX-Code

Auf den Abdruck des kompletten Quellcodes für das finale Skript (80 Zeilen) soll an dieser Stelle verzichtet werden, der geneigte Leser findet ihn zusammen mit den anderen Python-Skripten und Beschreibungen unter `http://code.google.com/p/spendenquittungen-mit-latex/`.

[6] Code urprünglich von `http://e6h.de/post/11/`

Fazit

Die vorgestellte Lösung vereint die elegante Mächtigkeit von Python mit LaTeX, um mit – im Vergleich zur manuellen Erstellung von mehreren Dutzend Spendenbescheinigungen – recht geringem Aufwand professionelle Dokumente automatisiert erstellen zu können.

Für Feedback, Ideen und Wünsche bin ich dankbar.

Literatur

[1] Rolf Niepraschk, »eso-pic«, `http://www.ctan.org/pkg/eso-pic`

[2] Marco Daniel/Elke Schuber, »mdframed«, `http://www.ctan.org/pkg/mdframed`

[3] `http://www.python-excel.org`

[4] `http://www.zetcode.com/db/sqlitepythontutorial/`

[5] Mark Lutz, »Programming Python«, O'Reilly

[6] Uwe Ziegenhagen, »Formulare ausfüllen mit LaTeX«, `http://uweziegenhagen.de/?p=1402`

[7] Nicola Talbot, »fmtcount«, `http://www.ctan.org/pkg/fmtcount`