Dokumentation zu `hbrs-thesis`

Modern und wissenschaftlich

Version 1.0

29. August 2023

Inhaltsverzeichnis

1 Vorwort

Diese Dokumentation ist nicht vollumfänglich. Das bedeutet, dass sie nicht auf alle Eigenheiten der einzelnen verwendeten Pakete eingeht. Für weitere und aktuelle Informationen empfehle ich immer die Recherche auf `https://www.ctan.org/`, wo alle Pakete mit ihren Dokumentationen hinterlegt sind.

In diesem Dokument werden die wichtigsten Befehle gezeigt und erklärt. Diese Dokumentation kann und sollte gerne durch die Hilfe der Community korrigiert und erweitert werden.

Viel Erfolg beim Schreiben!

2 Verwendung der LaTeX-Klasse

Um die Klasse als Vorlage für Dokumente zu verwenden, empfiehlt es sich, den kompletten Ordner `template` zu kopieren und entsprechend dieser Dokumentation zu verwenden. In diesem Kapitel werden die einzelnen Bestandteile des Ordners kurz erklärt. Die Verwendung der einzelnen Ordner und Dateien wird im späteren Verlauf des Dokuments näher erläutert.

2.1 Bestandteile des Templates

Die Konfiguration aller Informationen auf dem Deckblatt, sowie die Widmung können über die Datei `assets/utility/meta.tex` angepasst werden. Innerhalb des Ordners `assets/utility` finden sich außerdem noch eine Datei für Worttrennungen, die von LaTeXnicht korrekt erkannt werden (`hyphenation.tex`), Akronyme (`acronyms.tex`) und Glossareinträge (`glossary.tex`). `assets/utility/*`

In `assets/images` werden Bilder hinterlegt. Je nach Anzahl der Bilder im Dokument empfiehlt es sich, diese nach Kapitel zu sortieren. Im Ordner `assets/images` befindet sich bereits ein Ordner `titlepage`, welcher die Bilder für die Titelseite enthält. Diese sollten nicht gelöscht werden, da sonst die Titelseite und somit das Dokument nicht mehr gebaut werden kann. `assets/images/*`

Der Ordner `chapter` ist dafür gedacht, die Kapitel oder Abschnitte (je nach Verwendung der Dokumentklasse (siehe Kapitel 3)) aufzuteilen und die entsprechenden Dateien dort abzulegen. Diese Aufteilung bringt den Vorteil, dass Kapitel sehr schnell neu sortiert werden können. Zusätzlich enthalten die Dateien weniger Text und sind damit übersichtlicher. `chapter`

Zu einem wissenschaftlichen Dokument gehört auch Literatur. Die Informationen zu dieser Literatur werden als BibTeX oder BibLaTeX in der Datei `bibliography.bib` hinterlegt. Ich persönlich bevorzuge es die PDF-Dokumente lokal mit abzuspeichern. Mit lokalen Dateien habe ich die Möglichkeit Anmerkungen zu machen und die Dateien mit anderer Software besser zu durchsuchen (siehe `https://github.com/freedmand/semantra`). Die PDF-Dokumente kommen dann in den dieser Ordner `literature`. Je nach Literaturverwaltungssoftware können Einstellungen getroffen werden, um diese Dokumente direkt in der Software aufzurufen. `bibliography.bib` `literature`

Für den Bau des Dokuments mit `latexmk` wird die Datei `.latexmkrc` benötigt, da sie Informationen zum Bauen des Dokumentes mit Glossar und Akronymen enthält. Wird kein Glossar- und/oder Akronymverzeichnis benötigt, empfiehlt es sich, die entsprechende Option in der Klasse zu verwenden (siehe Kapitel 3). `.latexmkrc`

Beim Kompilieren des Dokumentes entsteht ein neuer Ordner `out`. Dieser Ordner enthält verschiedene Kompilierungsschritte, Log- und Synchronisierungsdateien von LaTeX. Im Zweifel kann dieser Ordner gelöscht und der Buildprozess neu gestartet werden. Die darin befindliche PDF-Datei (das gebaute Dokument) sollte jedoch vorher gespeichert werden, um einen Verlust der Daten zu vermeiden. Vorschläge für eine Konfiguration von LaTeX-Umgebungen in verschiedenen IDEs werden auch noch vervollständigt.

`out`

Zuletzt befindet sich in `hbrs-thesis.cls` sämtliche Konfiguration für die Klassendatei. Diese Konfiguration kann nach Belieben angepasst werden. Sollten Grundsätzliche Änderungs- oder Verbesserungsvorschläge an dieser Datei entstehen, bitte ich darum, diese auch bei GitHub (siehe `https://github.com/blackapple113/H-BRS-Thesisvorlage`) einzureichen.

`hbrs-thesis.cls`

3 Optionen der Klasse hbrs-thesis

Für die Modularität der Klasse wurden einige Optionen eingebaut, die die Verwendung des Templates vereinfachen sollen. Im Folgenden werden die Optionen mit Beispielen erläutert.

3.1 Spracheinstellungen

Eine Pflichtoption, welche für eine korrekte Worttrennung verwendet werden muss ist die Einstellung der Sprache. Die Sprache kann aktuell nur zwischen englisch und deutsch unterschieden werden. Weitere Anpassungsmöglichkeiten für komplexere bilinguale Dokumente oder andere Spracheinstellungen müssen über die Datei `hbrs-thesis.cls` manuell eingestellt werden. Angenommen es wird ein Dokument auf Deutsch geschrieben, so muss in Kombination mit dieser Klasse die Option auf `german` gesetzt werden. Intern werden dann Optionen für die Deutsche Sprache gesetzt. Wird das Dokument auf Englisch geschrieben, so muss `english` angegeben werden.

```
1  \documentclass[german]{hbrs-thesis}
2  ...
```

Werden bilinguale Spracheinstellungen im Dokument benötigt, müssen in der Klassendatei `hbrs-thesis.cls` Einstellungen für das Paket `babel` getroffen werden. Das Paket befindet sich unter der Region `Required packages`. Ist die Hauptsprache des Dokuments Englisch, kommen aber durchaus Deutsche Begriffe darin vor, eignet sich die Konfiguration `\RequirePackage[ngerman,english]{babel}`. Die zuletzt angegebene Sprache ist die Hauptsprache. **Mit dieser extra Konfiguration darf keine Sprache in den Optionen der Klasse gesetzt werden!**

3.2 Stiloptionen

Die Klasse besitzt ein paar Optionen, um verschiedene Stile anzubieten. Mithilfe der Option `classicstyle` bzw. `modernstyle` kann zwischen einer Schriftart mit Serifen und einer serifenlosen Schriftart (was sind Serifen: `https://de.wikipedia.org/wiki/Serife`) unterschieden werden. Serifen sollen dem Lesenden helfen die Zeilen besser zu verfolgen, um während des Lesens nicht in der Zeile zu verrutschen, sind aber auf Bildschirmen teilweise schlechter darzustellen. Der Standard ist hier `modernstyle` und muss somit nicht explizit angegeben.

```
\documentclass[german,classicstyle]{hbrs-thesis}
...
```

Abbildung 3.1: Beispielbild für die Option `classicstyle`.

Neben dem Stil der Schriftart kann auch zwischen `report` und `article` unterschieden werden. Report bietet einzelne Kapitel beschrieben durch `\chapter{...}` welche immer auf einer neuen Seite anfangen. Im Gegensatz dazu bietet `article` lediglich Abschnitte beschrieben durch `\section{...}`, die nicht jedes Mal auf einer neuen Seite beginnen. Diese beiden Stile können durch die Größe des entstehenden Dokumentes unterschieden werden. Zum Beispiel wird so bei einem Praxisprojektbericht `article` verwendet wohingegen für eine Thesis `report` geeignet erscheint. Da die Dokumente nicht doppelseitig ausgedruckt werden entfällt die Option für ein Buchlayout. Diese wird vielleicht später noch hinzugefügt. **Die Option `article` ist standardmäßig konfiguriert.** Für eine Thesis sollten die Optionen für die Klasse `hbrs-thesis` also z. B. wie folgt verwendet werden:

```
\documentclass[german,classicstyle,report]{hbrs-thesis}
...
```

Für den Fall, dass der zweispaltige Stil wie in IEEE- oder ACM-Dokumenten optisch bevorzugt wird, existiert die Option `twocolumn`. Diese Eignet sich am besten in Kombination der Optionen `classicstyle` und `article`. Die entsprechende Konfiguration sieht dann beispielsweise folgendermaßen aus, wobei `article` Standard ist, also nicht angegeben werden muss:

```
\documentclass[german,classicstyle,article,twocolumn]{hbrs-thesis}
...
```

Abbildung 3.2: Beispielbild für die Option `twocolumn` in Kombination mit `article` und `classicstyle`.

3.3 Glossar und Akronymverzeichnis

Werden kein Glossar und/oder Akronymverzeichnis verwendet empfiehlt es sich die Option `noglossaries` in der Klasse zu verwenden. Damit wird das entsprechende Paket nicht geladen, es wird keine Warnung diesbezüglich ausgegeben und das Kompilieren geht eventuell schneller.

```
1  \documentclass[german,noglossaries]{hbrs-thesis}
2  ...
```

4 Funktionen der Klasse

In diesem Kapitel werden die wichtigsten integrierten Funktionen der Klasse erklärt und mit Beispielen veranschaulicht. Nochmal der Hinweis, dass nicht sämtliche Funktionen der integrierten Pakete erläuter werden, sondern eine Verwendung innerhalb dieser Klasse für den einfachen Gebrauch dargestellt wird. Optionen der einzelnen Pakete können in `hbrs-thesis.cls` geändert werden. Dokumentation zu den einzelnen Paketen findet sich auf `https://www.ctan.org/`.

4.1 Code

Diese Klasse verwendet das Paket `minted` (siehe `https://www.ctan.org/pkg/minted`) für Syntax Highlighting. Das Paket basiert auf *Pygments* (siehe `https://pygments.org/`) welches mit Python auf dem System installiert sein sollte, solange kein Docker-Container zum Bauen bereitsteht. Außerdem muss für das Kompilieren die Option `-shell-escape` hinzugefügt werden, also `latexmk -shell-escape file.tex`.

4.1.1 Inline Code

Code, der im Text stehen soll, kann auf verschiedene Arten erreicht werden. Die einfachste ist die Verwendung von `\texttt{...}`. Dabei wird allerdings kein Syntax Highlighting verwendet. Der Text wird innerhalb dieser Umgebung nur an Leerzeichen umgebrochen und nicht innerhalb der Wörter.

Eine weitere Möglichkeit für Syntax Highlighting im Fließtext bietet `minted` mit dem Befehl `\mintinline{sprache}{code}`. Dieser Befehl kann zusätzlich an vorgegebenen Stellen den Text umbrechen, was dennoch zu Problemen bei der Formatierung am Rand führen kann.

4.1.2 Block Code

Oft ist es notwendig Programmcode im Dokument darstellen zu können. Für diesen Zweck gibt es zwei verschiedene Möglichkeiten. Zum einen kann der Code direkt in die LaTeX-Datei geschrieben werden, was bei Änderungen zu manuellen Anpassungen führt. Zum anderen kann der Code auch aus den Dateien importiert werden. **Aufgrund von Limitierungen von minted und TeX kann es zur Ausführung von Code kommen, weshalb nur bekannter Code geladen werden sollte!**

4.1.2 Block Code

Code kann wie in Kapitel 3 ohne starkes visuelles Absetzen direkt unter den Text geschrieben werden, in dem die Umgebung `code` verwendet wird.

```
\begin{code}[python]
    if __name__ == "__main__":
        print("Hello World!")
\end{...} % ersetze ... durch "code"
```

Um den Code optisch durch Linien klar abzugrenzen, wird die Umgebung `codeblock` verwendet. Dieser Codeblock eignet sich vor allem für größere Codebereiche. Zusätzlich kann ein Label für die Referenzierung auf den Code hinzugefügt werden.

```
\begin{codeblock}[python,label={code:python-class}]
    class Test:
        __init__(self):
            self.x = 5

    if __name__ == "__main__":
        print("Hello World!")
\end{codeblock}
```

Wie oben schon beschrieben kann Code auch aus anderen Dateien geladen werden. Dies geschieht mithilfe des Befehls `\inputcode{language}{path/to/file}{title}`.

```
\inputcode{java}{assets/code/Test.java}{Java Testdatei}
```

Code 4.1: Java Testdatei

```
public class Test{
    public static void main(String[] args){
        System.out.println("Hello World!");
    }
}
```

Soll nur ein bestimmter Bereich an Zeilennummern aus Code-Dateien angezeigt werden können die Zeilen als Zeilennummern (jeweils inklusiv) mit dem Befehl `\inputcodelinerange{language}{path/to/file}{title}{startline}{endline}` angegeben werden.

Weniger kompliziert machen mit NewTCBInputListing und xparse für optionale Angaben

```
\inputcodelinerange{java}{assets/code/Test.java}{Java Testdatei Zeilen 2 - 4}{2}{4}
```

Code 4.2: Java Testdatei Zeilen 2 - 4

```
public static void main(String[] args){
    System.out.println("Hello World!");
}
```