CoCoTEX

Simplifying layout development by complicating LATEX

Lupino

Dante Frühjahrstagung 04. 04. 2024

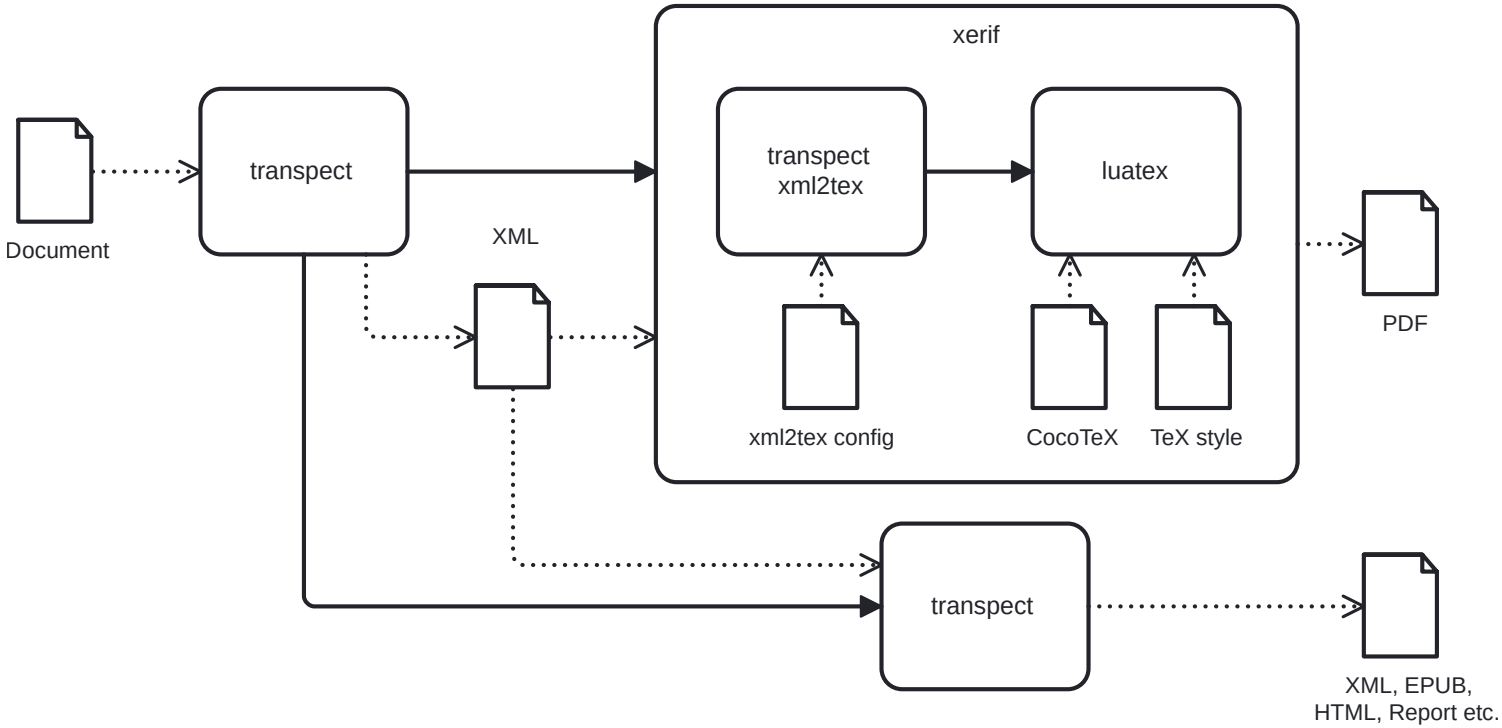
le tex
publishing services

Aussgangslage

le-tex publishing services

- Verlagsdienstleister mit Sitz in Leipzig
- Gegründet 1999
- 2024 über 100 MitarbeiterInnen
- verschiedene Dienstleistungen für hauptsächlich Fachbuch- und Wissenschaftsverlage, u.a.:
  - Verlagsherstellung
  - Lektorat
  - Korrektorat
  - Bildbearbeitung
  - Satz
- Automatisierung von Herstellungsprozessen: xerif

Xerif



- Open-Source Satzautomat
- besteht im Wesentlichen aus zwei Komponenten:
  - XML-Konverter (transpect, genauer: `docx2tex`-Modul)
  - PDF-Renderer (lualaTeX, genauer: CoCoTeX)
- Docx rein, LaTeX-PDF raus
- `.tex`-Datei ist lediglich ein Zwischenprodukt!

Anforderung Verlag 1, Teil 1

„Wir hätten gern einen Satzautomaten, der Monographien setzen kann, und bei dem die Kapitel-Überschriften so aussehen:“

Hauptüberschrift
Unterüberschrift

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

„... die dazugehörigen IHV-Einträge sollen so aussehen:“

„... und im Kolumnentitel soll nur der Haupttitel erscheinen“

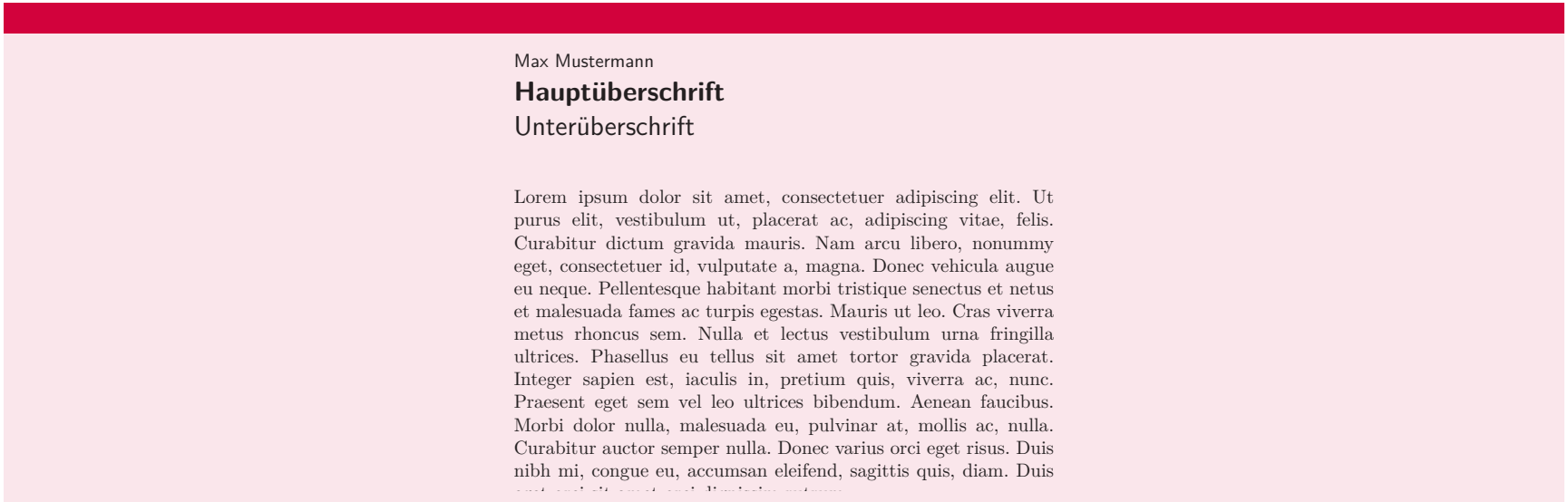
Umsetzung in Standard-LaTeX, Teil 1

machbar in `.tex`-Datei:

```
\chapter[\protect\textbf{Hauptüberschrift} Unterüberschrift]{\protect\textbf{Hauptüberschrift}\\Unterüberschrift}
\chaptermark{Hauptüberschrift}%
```

Anforderung Verlag 1, Teil 2

„und jetzt wollen wir auch Anthologien setzen, bei denen die Kapitel-Überschriften so aussehen sollen:“



„... die dazugehörigen IHV-Einträge sollen so aussehen:“

Umsetzung in Standard-LaTeX, Teil 2

schon komplizierter.
In `.tex`-Datei:

```
\chapauthor{Max Mustermann}
\chapter[\protect\textbf{Hauptüberschrift}]{\textbf{Hauptüberschrift}\\Unterüberschrfit}
```

und im Verlagsstyle:

```
\newif\ifcollection
```

```
\def\chapauthor#1{\def\@chapauthor{#1}}
```

```
\def\@chapter[#1]#2{
  ...
  \ifcollection
    \addcontentsline{toc}{chapter}%
      {\@chapauthor\\#1}%
  \else
    \addcontentsline{toc}{chapter}%
      {\protect\numberline{\thechapter}#1}%
  \fi
  ...}
```

```
\def\@makechapterhead#1{...
  \ifcollection
    {\normalsize \@chapauthor}\par\nobreak
     \global\let\@chapauthor\@undefined
  \fi
  \ifcollection\normalsize\else\Large\fi #1\par\nobreak
  ...}
```

Probleme mit dieser Umsetzung

- Ziemlich fiese Eingriffe in LaTeX-Kernel-Makros
- Teile der Formatierung an den Konverter ausgelagert
- XML-Leuten erklären, wo ein `\protect` erforderlich ist und wo nicht

Alternativ:

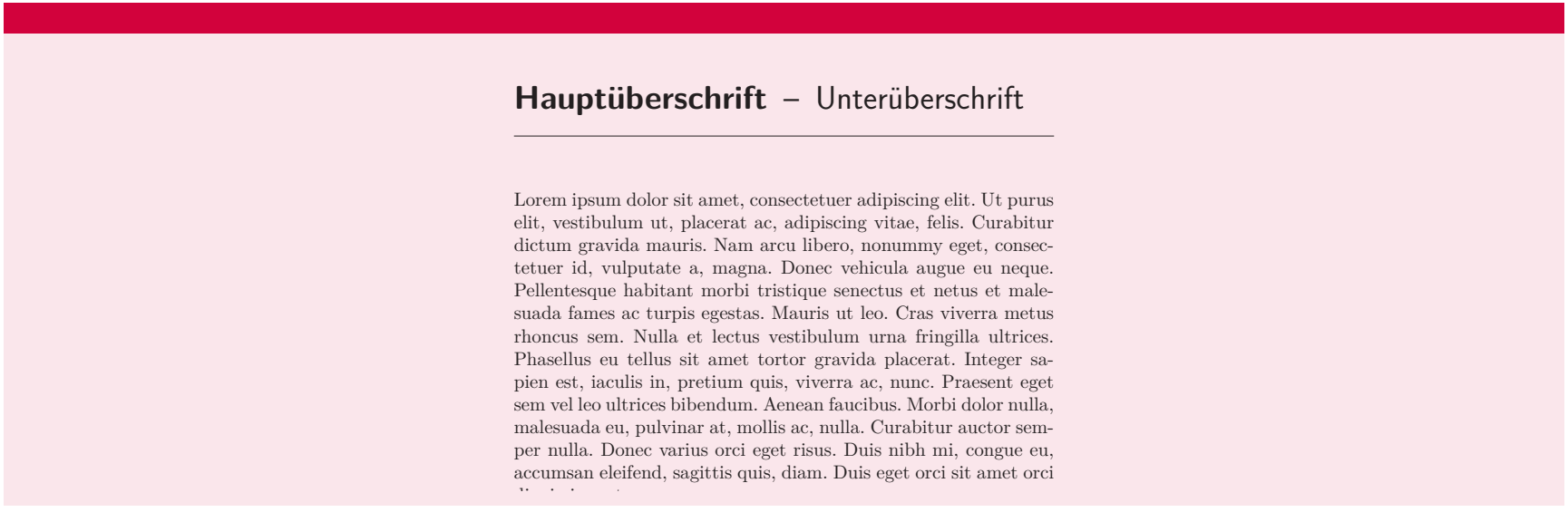
- Neues Überschriften-Makro für Kapitel in Collections vs. Monographien

Aber:

1. Konverter muss für dieselbe logische ÜS-Ebene zwei verschiedene Makros rausschreiben,
2. die sich u. U. optisch nicht weiter unterscheiden, also ggf. Potenzial für Code-Dopplungen

Anforderung Verlag 2, Teil 1

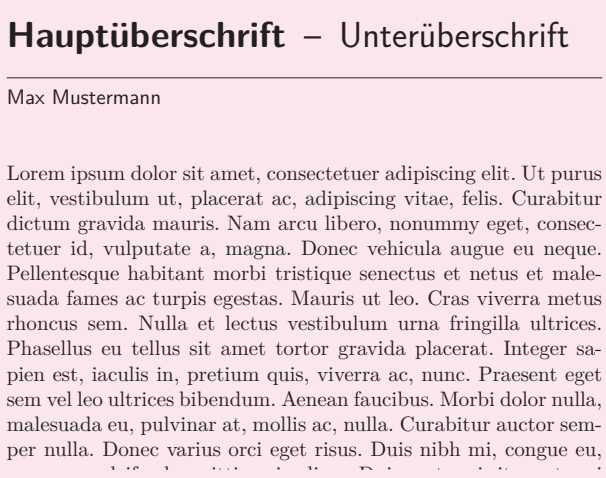
„Wir wollen dasselbe wie Verlag 1, aber bei uns sollen die Kapitel-Überschriften so aussehen:“



„... und die dazugehörigen IHV-Einträge so:“

Anforderung Verlag 2, Teil 2

„*Antologien sind toll, wollen wir auch, aber so:*“



Hauptüberschrift – Unterüberschrift

Max Mustermann

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu,

„*... und die dazugehörigen IHV-Einträge:*“

Umsetzung in Standard-LaTeX, Teil 3

- auch machbar, allerdings viel code-Dopplung zwischen Verlag 1 und Verlag 2
- und vermutlich müssen auch die Konverter-Leute viel zwischen Verlag 1 und Verlag 2 umbauen

Version 1 „Common Framework“

Auslagern der gemeinsam genutzten Definitionen, Schalter, etc. in eigene style-Dateien, die von allen xerif-Kunden gleichermaßen genutzt werden können

Neue Anforderung Verlag 1

„Wir hätten diese Möglichkeit, einen Autorennamen oder Untertitel an eine Überschrift zu pappen, auch bei tieferen Ebenen und auch bei Monographien:“

1 Hauptüberschrift

1.1 Abschnittsüberschrift Ebene 1

Untertitel

1.1.1 Abschnittsüberschrift Ebene 2

Lorem ipsum...

A Zwischentitel Ebene 2 und manueller Nummerierung

(Maxi Musterfrau)

Lorem ipsum...

1.1.2 Abschnittsüberschrift Ebene 2

Lorem ipsum...

Neue Anforderung Verlag 1

„Wir hätten gerne bei Abbildungen eine Möglichkeit, Bildunterschrift, Legende und Quellenangabe separat auszuzeichnen und zu formatieren“

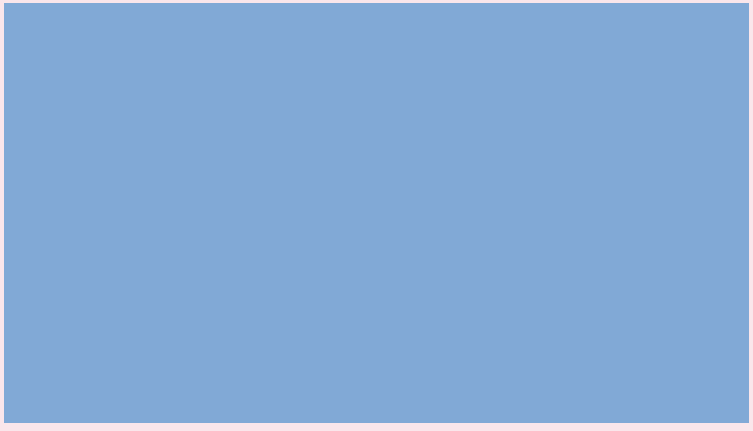


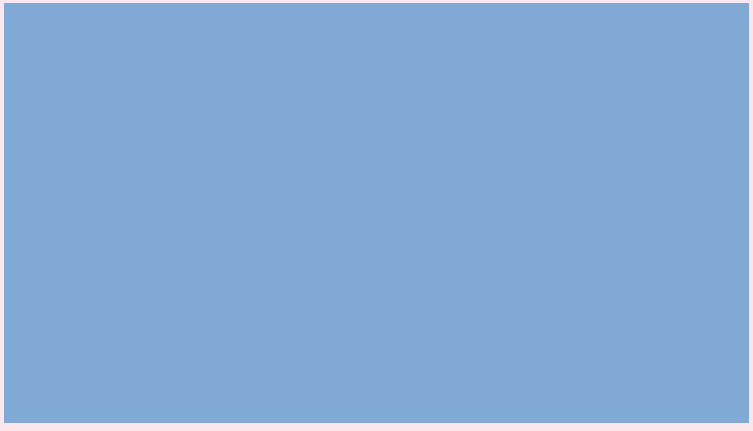
Abbildung 1 Eine Bildbeschreibung.
Legende: Vordergrund – blauer Quader;
Hintergrund – originalgetreue Nachzeichnung der Mona Lisa im Querformat.
Quelle: *Selbst gemalt.*

„Aber im Abbildungsverzeichnis soll natürlich nur die Nummer und die Bildbeschreibung erscheinen, nicht aber die Legende und die Quelle.“

Neue Anforderung Verlag 2

„Wollen wir auch, aber bei uns soll die Caption über das Bild, die Legende und Quelle darunter“

Abbildung 1 Eine Bildbeschreibung.



Legende: Vordergrund – blauer Quader; Hintergrund – originalgetreue Nachzeichnung der Mona Lisa im Querformat.
Quelle: Selbst gemalt.

„Und im Abbildungsverzeichnis soll nur die Nummer, die Bildbeschreibung und die Quelle (aber dort ohne Label!) erscheinen, nicht aber die Legende.“

Neue Anforderung Verlag 3

„Wir finden das alles toll, allerdings wollen wir das nicht für Monographien oder Collections, sondern für Zeitschriftenartikel…“

((Screenshot eines sehr viel komplexeren Article-Beispiels entfernt aus Copyright-Gründen))

„…und später für die komplette Zeitschrift…“

CoCoTeX

Zentrale Fragestellung

Was haben Gleitobjekte, Überschriften, Titeleien, Einträge in Inhalts-, Abbildungs- und Tabellenverzeichnissen, Listen, etc. gemeinsam?

Auf einer abstrakteren Ebene sind das makrotypographische Gestaltungselemente, die aus mehr oder weniger vor-definierten Bestandteilen bestehen, die wiederum auf eine bestimmte Art und Weise formatiert werden

bei Überschriften

- Titel
- Untertitel
- AutorInnen-Namen
- Nummerierung
- Overrides für IHV, Kolumne, Bookmarks

bei Gleitobjekten

- Caption
- Legende und Quellenangabe
- Nummerierung
- das Gleitobjekt selbst (z.B. Abbildung, Tabelle, bunte Infobox)
- Overrides für LoF/LoT

bei Titelseiten

- Buchtitel
- Untertitel
- AutorInnen-Namen
- Reihe und Reihennummer
- Einzelne Angaben im Impressum
- Overrides für Kol-Titel, PDF-Metadaten

„Adding Another Layer of Abstraction“

- Die *Idee* einer „Kapitelüberschrift mit all ihren Bestandteilen“ ist ein Container vom Typ „*Kapitelüberschrift*“
- Eine konkrete Kapitelüberschrift mit AutorInnen-Namen, Titel, Nummerierung etc. in einem Werk ist eine Instanz des Containers vom Typ „*Kapitelüberschrift*“
- Die Idee der Bestandteile („AutorInnen-Namen“, „Titel“, „Nummerierung“) sind Components dieses Containers vom Typ „*Kapitelüberschrift*“
- der konkrete Autor, der an einer konkreten Kapitel-Überschrift in einem konkreten Werk hängt, ist eine Instanz dieser Komponente des Containers vom Typ „*Kapitelüberschrift*“
- Die Anweisungen, wie die Instanzen dieser Komponente zusammengesetzt und gerendert werden, liegt in den Verlagsstyles in Form von LaTeX-Code-Schnipseln vor, die Properties genannt werden

Der Name „CoCoTeX“

steht für

*Co*ntainer and *Co*mponents for *TeX*.

CoCoTEX: Genereller Workflow

- Container sind im wesentlichen LATEX-Umgebungen,
- die komplett aus Components (oder weiteren Containern) bestehen,
- die wiederum LATEX-CSen (oder -Umgebungen) sind, die konrete Komponenten in Makros zwischenspeichern
- die dann am Ende der Umgebung anhand von vordefinierten Properties verarbeitet und letztlich ausgegeben werden

Terminologie

Terminologie I: Begriffe

Container

Idee einer Sammlung von typgraphischen Dingen™, z.B „Überschrift“, „Gleitobjekt“, aber auch „Kapitelüberschrift“ oder „Tabelle“

Type

Sammelbegriff für abstrakte Datenstrukturen, die für jeden Container festgelegt werden (können).

Component

Ding™ innerhalb eines Containers, z.B. „Kapitel-Titel“, „Section-Nummer“, „Subsection-Autorin“, „Buch-Titel“

Property

Formatierung, Aussehen, Reihenfolge und Auswahl von Komponenten

`Components` und `Properties` sind selbst auch Typen, die für nahezu alle CoCoTEX-Container vordefiniert sind.

Terminologie I: Umsetzung

Container-Instanzen

sind in der Regel LaTeX-Umgebungen, seltender Makros

Component-Instanzen

sind in der Regel LaTeX-Makros oder Umgebungen, die nur innerhalb ihres Containers definiert sind

Overrides

sind Komponenten, mit denen sich andernfalls via Properties *generierte* Komponenten lokal überschreiben lassen

Beispiel Container, Components und Overrides

```
\begin{heading}{chapter}
  \tpNumber{Kapitel~1}
  \tpTitle{irgendein langer Überschriftentitel}
  \tpTocTitle{gekürtzte Überschrift}
\end{heading}
```

Terminologie I: Beispiel für Properties

```
\ccDeclareHeading{0}{chapter}{%
  \ccSetProperty{margin-left}{0pt}%
  \ccSetProperty{before-heading}{\cleardoublepage\null}%
  \ccSetProperty{before-skip}{1sp}%
  \ccSetProperty{after-skip}{2\baselineskip}%
  \ccSetProperty{title-face}{\sffamily\LARGE\bfseries}
  \ccSetProperty{heading-block}{{%
    {\ccUseProperty{title-face}\tpUseComp{Title}}%
    \ccWhenComp{Subtitle}{{\enskip--\nobreak\enskip\nobreak\tpUseComp{Subtitle}}}\par
    \vskip.5\baselineskip
    \hrule
    \vskip.25\baselineskip
    \ccWhenComp{AuthorNameList}{{\normalsize\tpUseComp{AuthorNameList}\par}}%
  }}%
}
```

Terminologie II: Container-Deklaration und Vererbung

Deklaration von Containern:

```
\ccDeclareContainer{<container-name>}{<deklaration>}
```

Deklaration von Datentypen:

```
\ccDeclareType{<type-name>}{<deklaration>}
```

Container können die in anderen Containern deklarierten Datentypen „erben“:

```
\ccInherit{<type-list>}{<parent-list>}
```

Deklaration der zum Container gehörigen LaTeX-Umgebung

```
\ccDeclareEnv{begin}{end}
```

```
\ccDeclareContainer{heading}{%
  \ccInherit{Components,Properties}{CommonMeta}%
  \ccDeclareType{Parent}{}%
  \ccDeclareType{Components}{%
    \cch@provide@authors%
    \cch@provide@comp{Title}%
    \cch@provide@comp{Subtitle}%
    \cch@provide@comp{Number}%
    \cch@provide@comp{LicenceLogo}%
    \cch@provide@comp{LicenceName}%
    \ccDeclareComponent{RefLabel}{}{}%
    \cch@provide@quotes
  }%
  \ccDeclareType{Properties}{}%
  \ccDeclareEnv{\heading}{\endheading}%
}
```

Terminologie II: Komplexe Components

Component Groups

Components, die innerhalb eines Containers mehrfach instanziiert und ihrerseits Components beinhalten können, werden als Component groups bezeichnet.

Counted Components

sind die in den Component Groups enthaltenen Components

Collection Components

Beim LaTeX-Lauf werden alle Instanzen der Counted Components einer Container Group eingesammelt und entsprechend der Property zusammengebaut. Das Ergebnis wird in den Collection Components gespeichert,

Anmerkung zu Collection Components

Collection Components sind immer auch gleichzeitig Overrides, d.h. sie können *anstelle* der Counted Components verwendet werden.
Statt z.B. die Autoren einzeln zu listen, kann in einer `heading`-Container auch direkt die `AuthorNameList`-Komponente instanziiert werden.

Terminologie II: Collection Components II

Beispiel 1

Kapitel mit mehreren Autoren

```
\begin{heading}{chapter}
  \begin{tpAuthor}
    \tpFullName{Maxi Musterfrau}
  \end{tpAuthor}
  \begin{tpAuthor}
    \tpFullName{Max Mustermann}
  \end{tpAuthor}
\end{heading}
```

wird beim Rendern entsprechend dem Wert einer Property namens `author-list-print-format` die `AuthorNameList`-Component generiert.
In den Properties, die für das Rendering der Überschrift verantwortlich sind, muss dann nur noch die `AuthorNameList` Component aufgerufen werden.

Beispiel 2

Mehrere Kapitelautoren mit zusätzlichen Metadaten

```
\begin{heading}{chapter}
  \begin{tpAuthor}
    \tpFullName{Maxi Musterfrau}
    \tpEmail{maxi.musterfrau@example.org}
  \end{tpAuthor}
  \begin{tpAuthor}
    \tpFullName{Max Mustermann}
    \tpEmail{max.mustermann@example.org}
  \end{tpAuthor}
\end{heading}
```

Von OOP geklaut...

CoCoTEX	OOP
Container	Klasse, Namespace, Modul
Container-Instanz	Objekt
Typen	Definitionsbody der Klasse
Inherit-Mechanismus	Abgeleitete Klassen, Includes, Mixins
Components	Objektvariable
Wert der Component-CS	Wert der Objektvariable
Properties	Klassenvariable
	Klassenmethode
	Objektmethode

Beispiel: Einträge in ToC

```
\begin{heading}{chapter}
  \tpTitle{Hauptüberschrift}
  \tpSubtitle{Unterüberschrift}
  \tpAuthorNameList{Max Mustermann}
\end{heading}
```

IHV-Einträge:

Hauptüberschrift (Max Mustermann) 21

Max Mustermann
Hauptüberschrift Unterüberschrift 21

```
  \ccSetProperty{toc-heading}{%
    {\bfseries\ccUseComp{TocTitle}}\space%
    (\ccUseComp{TocAuthorNameList})
    \ccUseProperty{toc-page-sep}\ccUseComp{TocPage}%
  }
```

```
  \ccSetProperty{toc-heading}{%
    (\ccUseComp{TocAuthorNameList})\\\nopagebreak
    {\bfseries\ccUseComp{TocTitle}}\space%
    \ccUseComp{TocSubtitle}%
    \ccUseProperty{toc-page-sep}\ccUseComp{TocPage}%
  }
```

Anmerkungen:

- die `ToC*` Components werden automatisch generiert, sofern sie nicht explizit im Input stehen
- Die Konfiguration der IHV-Einträge findet innerhalb der Deklaration der ÜS-Ebene statt, nicht außerhalb wie in Standard-LaTeX,
- d.h., die diversen `\l@<level>` Makros werden *generiert* statt *definiert*.

CoCoTEX im Detail

Terminologie III: Weitere Begrifflichkeiten

Modul

Thematischer Bestandteil des Frameworks, z.B. `coco-headings`, `coco-floats`, `coco-common`, `coco-meta`, etc.

Attribute

Sind spezielle Eigenschaften einer Container-Instanz, die i.d.R. im optionalen Argument des `\begin` der Container-Umgebung mitgegeben werden.

Style-Klassen

Unterschiedliche Ausprägungen desselben Containers.
Nicht zu verwechseln mit dem Klassenbegriff aus der objektorientierten Programmierung; eher vergleichbar mit den style-Klassen in CSS!

Aufbau

cocotex.cls

- gemeinsame Basis für alle xerif-Satzautomaten
- definiert globale documentclass-Optionen und verteilt sie auf die einzelnen cocotex-Module
- lädt die meisten Teil-Module des Frameworks

coco-kernel.sty

- ist das eigentliche C/C/P Framework

coco-common.sty

- stellt wiederkehrende bzw. in mehreren Modulen benutzte Helfer-Makros bereit, die aber nicht Bestandteil der C/C/P-Infrastruktur sind
- z.B.enthält u.a. den Listof-/ToC-Mechanismus, sowie das automatische Generieren von Nummerierungen, Einzügen und Labels

Module

`coco-accessibility.sty`	experimentelle Features für PDF/A-Ausgabe
`coco-endnotes.sty`	End- und Fussnoten
`coco-floats.sty`	Gleitobjekte
`coco-frame.sty`	Layout-Rahmen, Cropmarks, Hilfslinien
`coco-headings.sty`	Überschriften
`coco-lists.sty`	Listen
`coco-scripts.sty`	Schriftverwaltung
`coco-titles.sty`	Titelseiten
`coco-meta.sty`	stellt abstrakte Parent-Container für die `titles` und `headings`-Module bereit

Weitere weiche Abhängigkeiten

`ltpdfa.lua`	wird vom `accessibility`-Modul verwendet
`htmltabs.sty`	special support im `floats`-Modul enthalten
`xerif-fonts`	Ausweichschriften, die im scripts-Modul verwendet werden (enthält die freie Fonts Noto und Junicode in diversen Schnitten)

Namenskonventionen

High-level Makros

- Container, Komponenten nutzen CamelCase
- Endnutzer-Befehle nutzen ebenfalls CamelCase, beginnen aber i.d.R. mit `cc` (und evtl. einem weiteren Buchstaben für die einzelnen Module, etwa cca für Makros aus dem `coco-accessibility` Modul) und
- Properties sind lowercase und nutzen - als Worttrenner

low-level Makros

- nutzen i.d.R. `cc@` oder `cc<modul-buchstabe>@` als Präfix und nutzen `@` als Worttrenner

Features

Globale Features

die CoCoTEX-Module können i.d.R. auch einzeln eingebunden werden, wenn die `cocotex.cls` *nicht* verwendet wird

- Engine-Neutral; CoCoTEX funktioniert mit latex, pdflatex oder lualatex (ausgenommen die Module `accessibility` und `script`, die erfordern beide *zwingend* LuaLATEX. XeLaTeX ungetestet)
- Optional automatische Zählung von Countern für Container-Instanzen
- label-ref-Mechanismus für Gleitobjekten und Überschriften

*No Standard-LATEX-Makros were harmed during development**

(ausgenommen `accessibility`-Modul, aber dort kommt `etoolbox` zum Einsatz)

coco-kernel.sty

beinhaltet im Wesentlichen den kompletten C/C/P Mechanismus

coco-common.sty

- Generalisierter list-of-Mechanismus; erzeugt dynamisch `l@`-Makros
- Generalisierter Mechanismus für hängende Einzüge nach dokument- oder ebenenweit breitesten Countern
- `\CalcRatio{<dimen>}{<dimen>}`: Berechnet das Verhältnis zwischen den Längen als 0..1
- `\CalcModulo{<divisor>}{<divident>}`: Restdivision
- `\minusvspace` Gegenstück zu LaTeX's `\addvspace`

coco-meta.sty

- abstrakte Container, die in den `heading`- und `title`-Modulen verwendet werden
- Role-Mechanismus: Stellt ein einheitliches Interface für Contributer-Rollen wie „author“, „editor“, „series-editor bereit und generiert für diese diverse Counted- und Collection Components.

coco-title.sty

- stellt den `Meta`-Container bereit, der im wesentlichen alle Metadaten des Gesamtwerkes sammelt und verarbeitet
- daraus dynamisches Generieren von `\ccMaketitle`
- einheitliches Handling von Sammelband-, Monographie-, Journal- und Article-Titelseiten
- Einbinden von XMP-Metadaten auch ohne `accessibility`-Modul, sofern vorhanden

coco-headings.sty

Modul für Überschriften

- einheitliches Handling für alle ÜS-Ebenen von part bis subparagraph
- beliebig erweiterbar
- Handling von zusätzlichen Metadaten für Journal- doer Sammelband-Beiträge
- Automatische Generierung von IHV-Einträgen, Kolumnentiteln und pdf-Bookmarks aus vorhandenen Komponenten
- Unterstützung für freistehende Überschriften und Spitzmarken
- generalisiertes Handling von hängenden Einzügen nach ÜS-Ebene, global oder strikt lokal sowohl an der ÜS selbst, als auch – und unabhängig davon – im IHV

coco-floats.sty

Modul für Gleitobjekte (im weitesten Sinne)

- einheitliches Handling für gleitende und nicht-gleitende „Floats“
- Subfloat-Mechanismus für Bilder mit Skalierung auf gemeinsame Höhe
- Caption- und paralleler Subcaption-Mechanismus
- Integrierte Unterstützung für `tabular`, `tabularx`, `tabulary` und `htmltabs`

coco-endnotes.sty

Modul für End- und Fussnoten

- Einfaches Umschalten zwischen Fuss- und Endnoten via Klassen-/Paketoption
- Einfaches Umschalten zwischen kapitelweiser und globaler Nummerierung via Klassen-/Paketoption
- Option zum automatischen Einfügen von Überschriften im Endnoten-Block

coco-script.sty

Handler für nicht-mitteleuropäische Schriftsysteme

- Definiert für vorgegebene Sprachen Fallback-Schriften, die via `\foreignlanguage` oder `\selectlanguage` umgeschaltet werden
- Ordnet Fallback-Schriften automatisiert nach `roman` und `sans-serif` Kontexten

coco-frame.sty

- Darstellung von Cropmarks und diversen PDF-Boxen um die gesamte Seite
- doppelseitiger Satzspiegelrahmen, incl. marginpar, Kopf- und Fusszeilen
- Hilfslinien für Grundlinien

coco-accessibility.sty (WIP!)

Features für barrierearme PDFs

- Im Wesentlichen ein alternatives Interface für das `ltpdfa`-Paket
- Integration von ltpdfa mit den restlichen CoCoTEX-Modulen
- Einbinden von Farbprofilen und XMP-Metadaten
- bei Bedarf auch automatische Generierung der XMP-Datei aus den Angaben in der `Meta`-Umgebung
- Automatisiertes Generieren von Gliederungs-Tagging
- Automatisiertes Taggen von internen und externen Verlinkungen

coco-lists.sty (WIP!!)

Handler für diverse Listenumgebungen

Where can i play with it?

Github-Repo

`https://github.com/transpect/CoCoTeX`

Enthält

- die dtx-Quelldateien aller CoCoTEX-Module
- die Quelldateien für die Endnutzer-Handbuches (W.I.P.!!!)
- ein ruby-Skript zum Generieren der sty-Dateien, Quellcode-Dokumentation und des Nutzerhandbuches
- die vor-gerenderten PDFs beider Dokumenatationen
- die Lua-Dateien des `ltpdfa`-Frameworks
- die `htmltabs.sty`
- Issue-Tracker

Achtung!

CoCoTEX befindet sich aktuell in einer Refactoring-Phase und unterliegt potentially code-breaking changes!
Das Betrifft insbesondere die Namen von End-User Makros!

Lebendbeispiel

Dissertation Katharina Klug

Ende

Vielen Dank für die Aufmerksamkeit!