LyX-Anpassung: Möglichkeiten für fortgeschrittene Benutzer

vom LyX-Team*

Version 2.3.x

13. Mai 2024

*Übersetzung: Peter Sütterlin, Leif Albers und Hartmut Haase (HHa, bis März 2010).

Inhaltsverzeichnis

1. Einleitung

In diesem Teil der Dokumentation wird beschrieben, welche Möglichkeiten LyX bietet, um es den eigenen Wünschen anzupassen. Es werden Dinge wie Tastaturkürzel, Vorschau am Bildschirm, Optionen zum Drucken, das Senden von Befehlen an LyX durch den LyX-Server, Internationalisierung, Installation neuer LaTeX-Klassen und LyX-Layouts usw. behandelt. Es kann hier nicht alles beschrieben werden, das an LyX individuell eingestellt und verändert werden kann — die Entwickler fügen Neuerungen schneller ein, als wir sie dokumentieren können — doch werden die grundlegenden Fähigkeiten von LyX dokumentiert sowie für einige der etwas obskuren Hinweise gegeben.

2. Die Konfigurationsdateien von LyX

Dieses Kapitel soll Ihnen dabei helfen, sich mit den Konfigurationsdateien von LyX vertraut zu machen. Bevor Sie jedoch weiterlesen, sollten Sie herausfinden, wo sich das System- und das Benutzerverzeichnis von LyX auf Ihrem Rechner befindet. Sie erfahren dies über den Menüpunkt Hilfe ▷ Über LyX. Im Systemverzeichnis speichert LyX alle systemweiten Konfigurationsdateien; wir werden es im weiteren LyXDir nennen. Im Benutzerverzeichnis können Sie an Ihre Bedürfnisse angepasste Versionen ablegen; wir werden dieses im weiteren UserDir nennen.

2.1. Was befindet sich in LyXDir?

Das Verzeichnis LyXDir sowie seine Unterverzeichnisse enthalten eine Anzahl Dateien, mit denen das Verhalten von LyX beeinflusst werden kann. Diese Dateien können direkt von LyX aus über den Dialog Werkzeuge ▷ Einstellungen... gelesen und geändert werden. Fast alles, was Sie möglicherweise an LyX ändern wollen, können Sie hier einstellen. Darüber hinaus können auch viele interne Dinge in LyX angepasst werden, indem man Dateien in LyXDir verändert. Die Dateien können in verschiedene Kategorien unterteilt werden, die in den folgenden Unterabschnitten behandelt werden.

2.1.1. Automatisch erzeugte Dateien

Diese Dateien, die sich im UserDir finden, werden automatisch bei der Konfiguration von LyX erzeugt. Sie enthalten verschiedene Standardwerte, die durch Untersuchung des Systems ermittelt werden. Normalerweise sollte man sie nicht verändern, da sie jederzeit von LyX überschrieben werden können.

`lyxrc.defaults` enthält Standardwerte für diverse Befehle. Einstellungen, die Ihnen nicht zusagen, können einfach über Werkzeuge ▷ Einstellungen... verändert werden.

`packages.lst` enthält eine Auflistung aller LaTeX-Pakete, die von LyX erkannt wurden. Derzeit wird diese Liste von LyX selber nicht benutzt, jedoch ist die Information, zusammen mit einigem anderen, über den Menüpunkt Hilfe ▷ LaTeX Konfiguration zugänglich.

`textclass.lst` ist eine Liste mit den im Verzeichnis `layout` gefundenen Textklassen, zusammen mit den entsprechenden LaTeX-Dokumentklassen und einer kurzen Beschreibung.

`lyxmodules.lst` ist eine Liste mit den im Verzeichnis `layout` gefundenen Modulen.

`*Files.lst` sind Listen von verschiedenen LaTeX-bezogenen Dateien, die auf Ihrem System gefunden wurden.

`doc/LaTeXConfig.lyx` wird bei der Konfiguration aus der Datei `LaTeXConfig.lyx.in` erzeugt. Das Dokument enthält Informationen über Ihre LaTeX-Konfiguration (bspw. darüber, welche Pakete Sie installiert haben).

2.1.2. Verzeichnisse

Diese Verzeichnisse finden sich sowohl in `LyXDir` als auch in `UserDir`. Wenn eine bestimme Datei beiden Verzeichnissen gefunden wird, wird jene in `UserDir` verwendet.

`bind/` Dieses Verzeichnis enthält Dateien mit der Endung `.bind`. In ihnen werden die Tastenkürzel festgelegt, die in LyX verwendet werden können. Falls im Unterverzeichnis `doc/xx`, wobei „xx“ jeweils des ISO-Sprachcode ist (für Deutsch: `de`), eine Datei mit einer an die internationalisierte Version von LyX angepassten Tastenbelegung existiert, wird diese bevorzugt geladen. Näheres dazu finden Sie in Kapitel 4.

`citeengines/` enthält Dateien mit der Endung `.citeengine`, in welchen die verschiedenen Literaturverweis-Methoden (Natbib, Biblatex usw.) spezifiziert werden. Siehe Kap. 5.2.6 für weitere Informationen.

`clipart/` Hier sind einige Grafiken gespeichert, die Sie in Ihre Dokumente einbinden können.

`doc/` enthält die Dateien der LyX-Dokumentation (einschließlich jener, die Sie gerade lesen). Eine Besonderheit stellt die bereits beschriebene Datei `LaTeXConfig.lyx` dar, da Sie bei der Konfiguration von LyX jeweils neu erzeugt wird. Die Übersetzungen der Dokumente (falls vorhanden) findet sich in Unterverzeichnissen `doc/xx`, wobei „xx“ jeweils des ISO-Sprachcode ist (für Deutsch: `de`). Sie werden in der jeweiligen Lokalisierung automatisch bevorzugt. Siehe dazu Kapitel 4.

`examples/` enthält Beispieldateien, die erläutern, wie Sie die unterschiedlichen Möglichkeiten von LyX nutzen können. Verwenden Sie die Schaltfläche Beispiele im Dateiauswahlmenü, um in dieses Verzeichnis zu gelangen.

`images/` enthält Bilddateien, die im Dialog Dokument ▷ Einstellungen verwendet werden. Außerdem finden Sie hier die unterschiedlichen Symbole für die Werkzeugleisten sowie das Bild für den Startbildschirm.

`kbd/` Hier sind die Definitionsdateien für die Tastaturbelegung gespeichert. Näheres dazu finden Sie im Kap. 4.2.

`layouts/` Hier werden die in Kapitel 5 beschriebenen Layoutdateien für die unterschiedlichen Dokumentenklassen sowie die Module gespeichert.

`lyx2lyx/` enthält Python-Skripte, die für die Konvertierung zwischen verschiedenen LyX-Versionen benötigt werden. Diese können auch von der Kommandozeile aus aufgerufen werden, etwa, wenn Sie mehrere Dateien gebündelt konvertieren wollen.

`scripts/` Hier sind einige Python-Skripte abgelegt, die LyX für bestimmte interne Operationen benötigt.

`templates/` enthält die Vorlagendateien, die Ihnen bei Datei ▷ Neu von Vorlage präsentiert werden, siehe Kap. 5.2.4.

`ui/` Hier finden Sie Dateien mit der Endung `.ui`, in denen die Benutzerschnittstelle von LyX genauer definiert ist, insbesondere, welche Einträge in welchen Menüs und Werkzeugleisten zu finden sind.

`xtemplates/` enthält Dateien mit der Endung `.xtemplate`, in welchen Vorlagen für die Einfügung von „externem Material“ in LyX-Dokumente definiert werden; siehe Kapitel 6.

2.1.3. Dateien, die Sie nicht verändern sollten

Die folgenden Dateien werden intern von LyX verwendet. Sie sollten im Normalfall nur von den Entwicklern editiert werden.

`CREDITS` Diese Datei enthält eine Liste der Entwickler. Ihr Inhalt wird über die Menüauswahl Hilfe ▷ Über LyX angezeigt.

`chkconfig.ltx` ist ein LaTeX-Skript, das bei der Konfiguration verwendet wird. Starten Sie es nie direkt.

`configure.py` ist ein Python-Skript, das zur Neukonfiguration von LyX verwendet wird. Es erzeugt die Konfigurationsdateien in dem Verzeichnis, von dem aus es aufgerufen wurde.

2.1.4. Andere Dateien

`encodings` beschreibt, wie die unterschiedlichen Zeichenkodierungen in Unicode dargestellt werden.

`languages` Eine Liste mit allen derzeit von LyX unterstützten Sprachen.

`latexfonts` enthält Informationen über die unterstützen LaTeX-Schriften.

`layouttranslations` enthält Übersetzungen für lokalisierbare Absatzstile (siehe Kap. 5.3.8).

`unicodesymbols` enthält Informationen über Unicode-kodierte Glyphen (Zeichen) und die Art und Weise, wie diese in LyX mit Hilfe von LaTeX unterstützt werden.

2.2. Das lokale Konfigurationsverzeichnis

Eventuell benutzen Sie LyX als normaler Benutzer und wollen dennoch einige Einstellungen der Konfiguration ändern. Zu diesem Zweck gibt es das Verzeichnis `UserDir`, in dem Ihre gesamte persönliche Konfiguration gespeichert wird. Dieses Verzeichnis wird als Klon des systemweiten Verzeichnisses verwendet. Das bedeutet, dass jede Datei, die Sie dort speichern, die entsprechende Datei im Systemverzeichnis ersetzt. Jede der im vorigen Abschnitt beschriebenen Konfigurationsdateien kann sich entweder im Systemverzeichnis `LyXDir` oder aber in Ihrem privaten Verzeichnis `UserDir` befinden. Im ersten Fall gelten die Einstellungen für alle Benutzer, im zweiten Fall nur für Sie.

Dies lässt sich an einigen Beispielen leichter erklären:

- Alle Änderungen, die über den Dialog Werkzeuge ▷ Einstellungen gemacht werden, werden in der Datei `preferences` gespeichert, die sich im `UserDir` findet.
- Wenn Sie mit dem Menüpunkt Werkzeuge ▷ Neu konfigurieren eine Neukonfiguration von LyX durchführen, werden die dabei erzeugten Dateien in Ihrem privaten Konfigurationsverzeichnis `UserDir` gespeichert. Das bedeutet, dass etwaige neue Dokumentenklassen, die Sie in Ihrem Verzeichnis `UserDir/layouts` gespeichert haben, im Feld Dokumentklasse des Dialoges Dokument ▷ Einstellungen... erscheinen.
- Wenn Sie sich von einem LyX-FTP-Server eine aktuellere Version (oder zum Beispiel diese deutsche Version) der Dokumentation besorgt haben, sie aber nicht *offiziell* installieren können, da Sie keine Systemadministratorrechte haben, können Sie diese Dateien einfach nach `UserDir/doc` kopieren, und sie werden automatisch über das Hilfe-Menü geöffnet.

2.3. LyX mit mehreren Konfigurationen

Die hochgradige Konfigurierbarkeit von LyX durch das lokale Verzeichnis wird für diejenigen nicht ausreichend sein, die parallel mehrere unterschiedliche Konfigurationen verwenden wollen, zum Beispiel unterschiedliche Tastaturkürzel und/oder Druckerkonfigurationen. Sie können dies durch das Anlegen von mehreren Konfigurationsverzeichnissen erreichen und LyX jeweils beim Start mitteilen, welches davon verwendet werden soll.

Indem Sie LyX mit der Option `-userdir` *<Verzeichnis>* starten, erreichen Sie, dass die Konfiguration aus diesem Verzeichnis anstelle des Standardverzeichnisses

gelesen wird (das Standardverzeichnis ermitteln Sie, indem Sie LyX ohne diese Option starten). Falls das so angegebene Verzeichnis noch nicht existiert, fragt LyX, ob es angelegt werden soll. Die Konfiguration in diesem Verzeichnis können Sie dann entsprechend verändern. Sie ist unabhängig von der Standardkonfiguration (aber lesen Sie weiter!). Anstelle der Kommandozeilenoption können Sie übrigens auch die Umgebungsvariable `LYX_USERDIR_VER` auf das zu verwendende Verzeichnis setzen.

Unterschiedliche Konfigurationsverzeichnisse bedeuten aber auch zusätzlichen Wartungsaufwand: Wenn Sie etwa eine neue Layoutdatei in `UserDir/layouts` hinzufügen und diese für alle Konfigurationen sichtbar sein soll, müssen Sie sie in *allen* Verzeichnissen separat hinzufügen. Sie können das jedoch mit einem Trick umgehen: Nachdem LyX das neue UserDir angelegt hat, sind praktisch alle Unterverzeichnisse (siehe oben) leer. Sie können also all diese Verzeichnis durch einen symbolischen Link auf das entsprechende Verzeichnis im originalen `UserDir` ersetzen. Lediglich mit dem Verzeichnis `doc` müssen Sie vorsichtig sein, denn dort wird eine Datei durch das Konfigurationsskript (Werkzeuge ▷ Neu konfigurieren) abgelegt, die konfigurationsabhängig ist.

3. Der Dialog Werkzeuge ▷ Einstellungen

Alle Optionen im Dialog Werkzeuge ▷ Einstellungen sind im Anhang *Der Einstellungen-Dialog* des Benutzerhandbuchs beschrieben. Hier finden Sie darüber hinaus gehende Informationen für einige Dinge.

3.1. Formate

Hier können Sie existierende Dateiformate verändern oder neue definieren. Für Letzteres öffnen Sie Werkzeuge ▷ Einstellungen ▷ Datei-Handhabung ▷ Dateiformate und klicken auf Neu. Das Format-Feld enthält den Namen, unter dem das Format in LyX erscheint. Im Feld Einsortieren als steht der Name, mit dem das Format intern identifiziert wird. Außerdem muss eine Dateiendung festgelegt werden. Diese drei Felder sind erforderlich. Zusätzlich kann ein Tastenkürzel definiert werden. Zum Beispiel ruft Strg+D das Menü Dokument ▷ Ansicht ▷ DVI auf.

Ein Format kann ein Bearbeitungsprogramm und ein Anzeigeprogramm haben. Sie können zum Beispiel für JPEG-Dateien gimp als Bearbeitungsprogramm und xv als Betrachter angeben. Zum Definieren des Befehls können auch die vier Variablen, die im nächsten Abschnitt beschrieben werden, benutzt werden. Das Anzeigeprogramm wird verwendet, wenn Sie ein Bild in LyX ansehen oder das Menü Dokument ▷ Ansicht verwenden. Das Bearbeitungsprogramm wird aufgerufen, wenn Sie nach einem Rechtsklick auf ein Bild Datei extern bearbeiten auswählen.

Der MIME-Typ[1] eines Formats muss nicht zwingend angegeben werden, wenn er aber angegeben wird, dann sollte dies einheitlich über alle Formatvarianten hinweg geschehen. Der MIME-Typ wird verwendet, um ein Dateiformat über den Dateiinhalt zu erkennen. Für einige wichtige Dateiformate wurde von der zuständigen Organisation (IANA) noch kein offizielles MIME-Typ festgelegt. LyX verwendet daher die erweiterte inoffizielle Liste, die von freedesktop.org festgelegt wurde.

Wenn Dokumentformat angekreuzt ist, weiß LyX, dass das Format für den Dokumentexport geeignet ist. Wenn dann auch noch ein geeigneter Konverter existiert (siehe Kap. 3.3), erscheint das Format unter Datei ▷ Exportieren. Außerdem erscheint es im Menü Dokument ▷ Ansicht, wenn ein Anzeigeprogramm angegeben wurde. Für reine Grafikformate wie `PNG` sollten Sie diese Option nicht benutzen, dagegen aber

[1] MIME (*Multipurpose Internet Mail Extensions*) ist ein Kodierstandard, der ursprünglich entwickelt wurde, um die Struktur und den Aufbau von E-Mails festzulegen. Er wird mittlerweile aber auch zur generellen Bestimmung von Dateiformaten eingesetzt.

für Formate, die sowohl Vektorgrafiken als auch Dokumente repräsentieren (etwa PDF).

Die Option Vektorgrafik-Format sagt LyX, dass ein Format Vektorgrafiken enthalten kann. Diese Information wird benutzt, um ein geeignetes Zielformat für eingefügte Grafiken für den PDFLaTeX-Export zu bestimmen. Eingefügte Grafiken müssen gegebenenfalls in PDF, PNG oder JPG konvertiert werden, weil PDFLaTeX keine anderen Grafikformate handhaben kann. Hat eine eingefügte Grafik nicht bereits eines dieser Formate, wird sie nach PDF konvertiert, falls Vektorgrafik-Format angekreuzt ist, andernfalls nach PNG.

3.2. Kopierer

Weil alle Konvertierungen im temporären Verzeichnis von LyX stattfinden, muss eine Datei manchmal geändert werden bevor sie ins temporäre Verzeichnis kopiert wird, damit die Konvertierung durchgeführt werden kann.[2] Das macht ein Kopierer: Er kopiert eine Datei ins (oder vom) temporären Verzeichnis und verändert sie gegebenenfalls dabei.

Die Definitionen der Kopierer können acht Variablen benutzen:

`$$s` ist das Systemverzeichnis von LyX (zum Beispiel `/usr/local/share/lyx`).

`$$i` ist die Eingabedatei.

`$$o` ist die Ausgabedatei.

`$$b` Der Basisname (ohne Dateinamenerweiterung), wie er im temporären LyX-Verzeichnis verwendet wird.

`$$p` ist der vollständige Dateipfad des temporären LyX-Verzeichnisses.

`$$r` ist der vollständige Dateipfad zur LyX-Datei.

`$$f` ist der Dateiname der LyX-Datei (ohne Verzeichnispfad).

`$$l` ist der *LaTeX-Name*. Dies sollte der Dateiname sein, den Sie in LaTeX im `\include`-Befehl benutzen würden. Er ist nur dann relevant, wenn die exportierten Dateien für diesen Befehl geeignet sind.

Kopierer können benutzt werden, um *fast* alles mit Ausgabedateien zu machen. Wenn Sie zum Beispiel PDF-Dateien in ein spezielles Verzeichnis (bspw. `/home/ich/pdf/`) kopieren wollen, können Sie ein Shell-Skript wie das folgende schreiben:

[2] Wenn die Datei beispielsweise auf andere Dateien mit relativen Pfaden verweist – vielleicht Bilder – und diese Pfade beim Kopieren ungültig werden.

```
#!/bin/bash
FROMFILE=$1
TOFILE=‘basename $2‘
cp $FROMFILE /home/ich/pdf/$TOFILE
```

Speichern Sie das Skript ausführbar in Ihrem lokalen LyX-Verzeichnis (`UserDir`) – etwa `/home/ich/.lyx/scripts/pdfkopierer.sh`. Dann wählen Sie in Werkzeuge ▷ Einstellungen ▷ Datei-Handhabung ▷ Dateiformate das Format PDF (pdflatex) und tragen im Kopierer-Feld `pdfkopierer.sh $$i $$o` ein.

Kopierer werden von LyX in vielen eigenen Konvertierungsprozessen benutzt. So erzeugt LyX, wenn auf Ihrem Rechner geeignete Programme installiert sind, automatisch Kopierer für die Formate HTML und HTML (MS Word). Wenn diese Formate exportiert werden, sieht der Kopierer, dass nicht nur die Haupt-HTML-Datei, sondern auch verschiedene zugehörige Dateien (Stildateien, Bilder usw.) kopiert werden müssen. All diese Dateien werden in ein Unterverzeichnis des Verzeichnisses geschrieben, in dem die LyX-Datei steht.[3]

3.3. Konverter

Sie können eigene Konverter in Werkzeuge ▷ Einstellungen ▷ Datei-Handhabung ▷ Konverter definieren. Dazu wählen aus Von Format und In Format jeweils ein Format aus, schreiben den benötigten Befehl ins Feld Konverter und klicken auf Hinzufügen rechts oben. Sie können im Befehl mehrere Variablen benutzen:

`$$s` ist das Systemverzeichnis von LyX

`$$i` ist die Eingabedatei.

`$$o` ist die Ausgabedatei.

`$$b` ist der Dateiname der Eingabedatei ohne Erweiterung.

`$$p` ist der Pfad zur Eingabedatei.

`$$r` ist der Pfad zur ursprünglichen Eingabedatei. Wenn eine Kette von Konvertern aufgerufen wird, weicht er von `$$p` ab.

`$$e` Der `Iconv`-Name der Kodierung des Dokuments.

Ins Feld Zusatz-Flag können Sie folgende durch Kommata getrennte Flags schreiben:

[3] Kopierer können angepasst werden. Der optionale Parameter `-e` kann eine durch Kommata getrennte Liste von Erweiterungen enthalten, die mit kopiert werden sollen. Wenn es fehlt, werden alle Dateien kopiert. Der Parameter `-t` bestimmt die Namenserweiterung, die an den erzeugten Verzeichnisnamen angehängt werden soll. Standardmäßig ist es LyXconv, so dass die aus `/Pfad/nach/Datei.lyx` erzeugte HTML-Datei im Verzeichnis `/Pfad/nach/Datei.html.LyXconv` landet.

`latex=flavor` teilt LyX mit, dass der Konverter eine Variante von LaTeX darstellt. Dies macht die LaTeX-Fehlermeldungen von LyX verfügbar. Der optionale Wert `flavor` legt fest, welche Variante von LaTeX (`latex, pdflatex, platex, xetex, luatex`) verwendet wird. Wenn kein Wert angegeben wird, wird `latex` verwendet.

`needauth` Dieser Konverter wird als nicht sicher angesehen und benötigt eine Autorisierung durch die Benutzer:innen. Je nach Einstellung in Werkzeuge▷Einstellungen▷Datei-Handhabung▷Konverter werden Benutzer:innen (a.) gefragt, ob sie dem aktuellen Dokument temporär, permanent oder gar nicht vertrauen, (b.) darüber informiert, dass eine Konvertierung aus Sicherheitsgründen nicht möglich ist oder (c.) nicht informiert, da sie eine grundsätzliche Vertrauenserklärung abgegeben haben. Verwenden Sie diesen Flag für Konverters, die arbiträre Programme ausführen könnten.

`needaux=flavor` teilt LyX mit, dass der Konverter eine LaTeX-Hilfsdatei (Dateiendung `.aux`) zur Konvertierung benötigt. Der optionale Wert `flavor` legt fest, welche Variante von LaTeX (`latex, pdflatex, platex, xetex, luatex`) verwendet wird. Wenn kein Wert angegeben wird, wird `latex` verwendet.

`nice` teilt LyX mit, dass der Konverter eine „schöne“ Datei benötigt, also eine, die so aussieht, wie die, die man über das Menü exportiert (ohne interne Hilfsbefehle wie `input@path`).

`xml` teilt LyX mit, dass der Konverter ein XML-Format ausgibt.

Die folgenden Flags sind keine richtigen, weil sie ein Argument der Form `key=value` benutzen:

`hyperref-driver` Der Name der Treiberdatei, die für diesen Konverter mit dem Hyperref-Paket geladen werden soll. Dies ist nötig, um bestimmte PDF-Features verwenden zu können. Konsultieren Sie das Hyperref-Handbuch für Einzelheiten.

`parselog` Wenn das gesetzt ist, werden der Standardfehler des Konverters in die Datei `Eingabedatei.out` umgeleitet, und das Skript wird so ausgeführt: `script < Eingabedatei.out > Eingabedatei.log`. Das Argument kann `$$s` enthalten.

`resultdir` ist der Name des Verzeichnisses, in dem der Konverter die erzeugten Dateien ablegen soll. LyX legt das Verzeichnis nicht an und kopiert auch nichts hinein, aber es kopiert dieses Verzeichnis an seinen Bestimmungsort. Das Argument kann `$$b` enthalten, was durch die Basisnamen von Ein- oder Ausgabedatei ersetzt wird, wenn das Verzeichnis kopiert wird. Beachten Sie, das `resultdir` und `usetempdir` zusammen keinen Sinn ergeben. Wenn das erste definiert wurde, wird das zweite ignoriert.

`resultfile` ist der Name der Ausgabedatei und kann `$$b` enthalten. Er wird nur zusammen mit `resultdir` benutzt und ist auch dann nur optional. Wenn er nicht angegeben wird, wird `index` benutzt.

Ein passender Hyperref-Treiber wird für einige mit LyX installierten Konverter definiert. Die zuletzt aufgeführten drei Flags hingegen werden zurzeit von keinem der vorinstallierten Konverter verwendet.

Sie müssen nicht für alle Formate, zwischen denen Sie konvertieren wollen, Konverter definieren. Zum Beispiel gibt es keinen Konverter von LyX nach PostScript, aber LyX kann dennoch PostScript exportieren. Dies geschieht, indem zunächst eine LaTeX-Datei erzeugt wird – dafür wird auch kein Konverter benötigt –, die dann mit dem Konverter von LyX nach DVI in eine DVI-Datei konvertiert wird, die schließlich nach PostScript konvertiert wird. LyX findet solche Konverter-Ketten automatisch und wird immer die kürzeste wählen.

Sie können auch mehrere Konversionsvarianten zwischen Dateiformaten definieren. Zum Beispiel liefert die Standardkonfiguration von LyX fünf Möglichkeiten, um von LaTeX nach PDF zu konvertieren:

1. direkt mit `pdflatex`
2. mit `ps2pdf` über DVI und PostScript
3. mit `dvipdfm(x)` über DVI
4. direkt mit `XeTeX`
5. direkt mit `LuaTeX`

Um solche alternativen Ketten zu definieren, müssen Sie alternative Ziel-*Dateiformate* definieren, wie in Kap. 3.1 beschrieben. Zum Beispiel enthält die Standardkonfiguration verschiedene Formate für PDF-Dateien, die `pdf` (für `ps2pdf`), `pdf2` (für `pdflatex`), `pdf3` (für `dvipdfm`), `pdf4` (für `XeTeX`) und `pdf5` (für `LuaTeX`) heißen.

4. Internationales LyX

LyX unterstützt die Übersetzung der Benutzerschnittstelle in beliebige Sprachen. Als dieser Text erstellt wurde, wurden neben dem Englischen bereits 32 Sprachen (in unterschiedlichem Ausmaß) unterstützt. Die jeweils benutze Sprache nennt man *locale* (Lokalisierung). (Für weitere Informationen über Lokalisierungseinstellungen lesen Sie bitte die entsprechende Dokumentation Ihres Betriebssystems. Im Fall von Linux ist die Manpage `locale (5)` ein guter Startpunkt.)

Bitte beachten Sie, dass die Übersetzungen zwar funktionieren, aber zuweilen ein paar Einschränkungen unterliegen. Insbesondere wurde das Design der Popup-Menüs auf den englischen Text zugeschnitten. Das bedeutet, dass der übersetzte Text an einigen Stellen mehr Platz benötigt als dort zur Verfügung steht. Dies ist natürlich nur ein Darstellungsproblem und schränkt nicht die Funktionsweise von LyX ein. Sie werden auch feststellen, dass einige Übersetzungen nicht für alle Menüpunkte Tastenkürzel definieren. Manchmal stehen einfach nicht genügend freie Buchstaben zur Verfügung, manchmal hatte der Übersetzer bisher keine Zeit, sich darum zu kümmern. Unser Lokalisierungsteam, dem Sie vielleicht beitreten möchten,[1] ist natürlich bemüht, diese Dinge in einer späteren Version zu korrigieren.

4.1. LyX übersetzen

4.1.1. Die Benutzerschnittstelle übersetzen (Textmeldungen)

LyX verwendet die `GNU-Gettext`-Bibliothek, um die Internationalisierung der Benutzerschnittstelle zu verwalten. Um LyX dazu zu bringen, in allen Menüs und Dialogen Ihre bevorzugte Sprache zu verwenden, müssen Sie eine `po`-Datei für diese Sprache erstellen. Anschließend müssen Sie daraus eine `mo`-Datei erzeugen und diese installieren. Eine umfassende Anleitung dazu finden Sie in der Dokumentation für `GNU Gettext`.[2] Natürlich können Sie das einfach nur für sich selbst tun, aber wenn Sie es schon tun, können Sie die Früchte Ihres Fließes auch mit dem Rest der LyX-Gemeinschaft teilen. Schicken Sie einfach eine Nachricht an die LyX-Entwicklerliste, um weitere Informationen zum Vorgehen zu erhalten.

Kurz gesagt müssen Sie folgendes tun (`xx` bezeichnet den Sprachcode der neuen Sprache):

[1] Wenn Sie eine andere Sprache als Englisch gut beherrschen, ist dies ein guter Weg, der LyX-Gemeinschaft etwas zurückzugeben.

[2] Natürlich nur auf englisch. Die Veränderungen, die an der `po`-Datei durchgeführt werden müssen, sind allerdings recht intuitiv.

- Laden Sie den LyX-Quellcode herunter. (Siehe die Informationen im Netz.)
- Kopieren Sie die Datei `lyx.pot` in das Verzeichnis der .po-Dateien. Benennen Sie anschließend die Datei in `xx.po` um. (Falls `lyx.pot` nicht existiert, kann sie mit dem Befehl `make lyx.pot` neu erzeugt werden. Sie können alternativ auch eine beliebige existierende `po`-Datei als Vorlage verwenden.)
- Bearbeiten Sie `xx.po`.[3] Für einige Menüeinträge und Dialogelemente gibt es Tastenkürzel, die ebenfalls angepasst werden sollten. Diese Kürzel werden mit '|' markiert und sollten so übersetzt werden, dass Sie zum übersetzten Ausdruck passen. Sie sollten auch das Informationsfeld am Anfang der neuen `po`-Datei ausfüllen (mit Ihrer E-Mail-Adresse usw.), damit Sie für andere Leute erreichbar sind, die Ihnen Vorschläge oder unterhaltsame Kommentare schicken möchten.

Wenn Sie dies alles nur für den eigenen Gebrauch tun, dann:

- Erzeugen Sie in die Datei `xx.mo` mithilfe des Befehls `msgfmt -o xx.mo < xx.po`.
- Kopieren Sie diese Datei in ihr Lokalisierungsverzeichnis, in das entsprechende Unterverzeichnis für die Sprach `xx` und unter dem Namen `lyx.mo` (bspw. `/usr/local/share/locale/xx/LC_MESSAGES/lyx.mo`).

Wie gesagt wäre es aber besser, wenn Sie die neue `po`-Datei zur LyX-*Distribution* beisteuern würden, damit auch andere sie verwenden können. Wenden Sie sich an die LyX-Entwicklerliste, wenn Sie das tun möchten.

4.1.1.1. Mehrdeutige Texte

Manchmal muss ein- und derselbe englische Ausdruck in verschiedene Varianten übersetzt werden. Ein Beispiel ist der englische Ausdruck `To`, der im Deutschen je nach Kontext entweder *Nach* oder *Bis* lauten muss. `GNU Gettext` unterstützt solche ambigen Übersetzungen nicht. Daher werden in LyX Kontextinformationen beigefügt. Statt `To` heißt es `To[[as in 'From format x to format y']]` und `To[[as in 'From page x to page y']]`. Diese beiden Varianten werden von `gettext` als zwei verschiedene Nachrichten behandelt, sie können daher jeweils wie gewünscht als *Nach* bzw. *Bis* übersetzt werden (die Kontextinformationen werden in der Übersetzung weggelassen).

Natürlich müssen diese Kontextinformationen auch von der ursprünglichen Nachricht getilgt werden, wenn diese (in der englischen Lokalisierung) verwendet wird. Deshalb erscheinen sie in doppelten eckigen Klammern am Ende des Textes. Der Übersetzungsmechanismus von LyX unterdrückt alles in doppelten eckigen Klammern am Ende von Nachrichten, wenn die Nachricht selbst angezeigt wird.

[3] Für diese Aufgabe gibt es spezielle Programme, wie Poedit (für alle Plattformen) oder KBabel (für KDE). Emacs hat ebenfalls einen Modus, der Sie bei dieser Arbeit unterstützt, siehe https://www.gnu.org/software/gettext/manual/html_node/PO-Mode.html#PO-Mode.

4.1.2. Die Dokumentation übersetzen

[Anmerkung des Übersetzers: wenn *Sie die Dokumentation übersetzen wollen, benutzen Sie als Vorlage auf jeden Fall das englische Original. Übersetzungen – diese eingeschlossen – sind oft nicht ganz auf dem neuesten Stand.]*

Auch die LyX-Dokumentation (die Sie im Menü Hilfe finden) kann (und sollte!) übersetzt werden. Wenn übersetzte Versionen verfügbar sind und die Lokalisierung entsprechend eingestellt wurde, werden diese automatisch von LyX benutzt.[4] LyX sucht nach übersetzten Versionen in `LyXDir/doc/xx/DocName.lyx`, wobei `xx` wiederum für den Code der aktuellen Sprache steht.

Falls solche Dateien nicht existieren, wird die englische Version verwendet. Die übersetzten Versionen müssen den gleichen Dateinamen (im Beispiel oben `DocName`) tragen wie die englischen Originale. Wenn Sie gerne die Dokumentation übersetzen möchten (übrigens ein guter Weg, um die Originale Korrektur zu lesen!), geben wir Ihnen hier ein paar Tipps, die Ihnen möglicherweise etwas Arbeit ersparen:

- Werfen Sie einen Blick auf die Seiten des Übersetzungsteams auf der Homepage des LyX-Entwickler-Teams: https://www.lyx.org/Translation. Dort erfahren Sie, welche Texte bereits in Ihre Sprache übersetzt sind, auch sehen Sie, ob jemand (und wenn ja, wer) die Übersetzungsaktivitäten koordiniert. Wenn niemand dies tut, lassen Sie uns bitte wissen, falls Sie an dieser Aufgabe interessiert sind.

Wenn Sie dann mit der eigentlichen Übersetzungsarbeit beginnen, sind hier einige Tipps, die Ihnen vielleicht helfen, einige Schwierigkeiten zu überwinden:

- Machen Sie im Dokumentationsteam mit! Informationen dazu gibt es in Hilfe ▷ Einführung. Dies ist übrigens das erste Dokument, das Sie übersetzen sollten.

- Machen Sie sich mit den typographischen Konventionen der Sprache vertraut, in die Sie übersetzen möchten. Typographie ist eine alte Kunst, und in verschiedenen Teilen der Welt wurden verschiedene Konventionen eingeführt. Auch sollten Sie die typographische Terminologie in Ihrem Land lernen. Eine eigene Terminologie würde nur die Leser verwirren. *(Warnung: Typographie macht süchtig!)*

- Legen Sie eine Kopie des Originaldokumentes an. Dies wird Ihre Arbeitskopie. Sie können diese auch als persönliche Hilfe-Datei in LyX verwenden, indem Sie sie in den Ordner `UserDir/doc/xx/` kopieren.
  Achtung: Für komplexe Dokumente mit externem Material (Bilder usw.) werden die Dateipfade von relativ auf absolut geändert, wenn man das Dokument verschiebt. Daher ist es das Beste, LyX mittels Git zu beziehen (siehe `https://www.lyx.org/HowToUseGIT`) und das Dokument im Verzeichnis zu belassen.

[4] Zurzeit sind Übersetzungen in etwa 20 Sprachen vorhanden.

- Manchmal wird das Originaldokument (vom LyX-Team) aktualisiert. Verwenden Sie den Quelltextbetrachter unter https://www.lyx.org/trac/timeline, um nachzuvollziehen, was verändert wurde. So können Sie leicht erkennen, welche Teile des Dokuments bearbeitet wurden.

Wann immer Sie einen Fehler im Originaltext entdecken, korrigieren Sie ihn bitte und teilen Sie dem Rest des Dokumentationsteams Ihre Veränderungen mit. (Sie haben nicht vergessen, dem Dokumentationsteam beizutreten, oder?)

4.2. Internationale Tastaturtabellen: *Keymaps*

Die nächsten beiden Abschnitte beschreiben detailliert die Syntax der `.kmap`- und `.cdef`-Dateien. Diese Abschnitte sollten Ihnen dabei helfen, Ihre eigene Tastaturtabelle zu entwerfen, wenn die vorhandenen nicht ganz Ihren Bedürfnissen entsprechen.

4.2.1. Die `.kmap`-Datei

Eine .`kmap`-Datei transformiert gedrückte Tasten zu Buchstaben oder Zeichenketten – es definiert ein *keyboard mapping*. Im Folgenden werden die Schlüsselwörter `kmap`, `kmod`, `kxmod` und `kcomb` beschrieben.

`kmap` transformiert einen Buchstaben zu einer Zeichenkette

`\kmap` *Zeichen Zeichenkette*

Dieser Ausdruck definiert, dass *Zeichen* zu *Zeichenkette* transformiert werden soll. Dabei müssen in *Zeichenkette* die Zeichen Backslash „\“ und Anführungszeichen „`"`“ mit einem vorangehenden Backslash versehen werden.

Als Beispiel ein Ausdruck, der das Zeichen „`/`“ ausgibt, wenn die Taste „`&`“ gedrückt wurde:

```
\kmap & /
```

`kmod` spezifiziert ein Akzentzeichen

`\kmod` *Zeichen Akzent erlaubt*

Dieser Ausdruck sorgt dafür, dass *Zeichen* als ein bestimmter *Akzent* interpretiert wird, und zwar bei allen Zeichen, die in *erlaubt* aufgeführt sind. Dies ist der Mechanismus toter Tasten *(dead keys)*.[5]

Wenn Sie die Taste *Zeichen* drücken, gefolgt von einem Zeichen, das *nicht* in *erlaubt* aufgeführt wurde, werden einfach beide Zeichen einzeln ausgegeben. Beachten Sie,

[5] Der Ausdruck *tote Taste* kommt daher, dass diese Taste allein kein Zeichen erzeugt, aber, gefolgt von einer anderen Taste, akzentuierte Zeichen erzeugt. Zum Beispiel kann auf diese Weise é“ erzeugt werden.

dass die Rücktaste eine vorangehende tote Taste nivelliert; wenn Sie also *Zeichen Rücktaste* eingeben, wird der Cursor nicht eine Position rückwärts gehen, sondern wird statt dessen den Effekt nivellieren, den Zeichen auf den folgenden Buchstaben gehabt hätte.

Der folgende Ausdruck definiert, dass die Taste „^“ zum Zirkumflex-Akzent wird, wenn er von einem der Buchstaben a, e, i, o, u, A, E, I ,O oder U gefolgt wird:

```
\kmod ^ circumflex aeiouAEIOU
```

`kxmod` definiert eine Ausnahme zu einem Akzentzeichen

`\kxmod` *Akzent Zeichen Ausgabe*

Dieser Ausdruck definiert eine Ausnahme für die Wirkung, die *Akzent* in Verbindung mit *Zeichen* haben soll. Dabei muss *Akzent* vorher mit Hilfe einer `\kmod`-Zeile einer Taste zugewiesen worden sein. Wenn Sie die Sequenz *Akzent Zeichen* drücken, wird *Ausgabe* produziert. Falls solch eine Definition *nicht* existiert und Sie *Akzent Zeichen* eingeben, erhalten Sie das *Akzent Zeichen* als Ergebnis, wobei Akzent das erste Argument der `\kmod`-Definition ist.

Der folgende Ausdruck sorgt dafür, dass LaTeX bei einem „i“ mit circumflex den I-Punkt entfernt, bevor das Akzentsymbol eingefügt wird:

```
\kxmod circumflex i "\\^{\\i}"
```

`kcomb` kombiniert zwei Akzentsymbole

`\kcomb` *Akzent1 Akzent2 erlaubt*

Hier wird es ziemlich esoterisch. Dieser Ausdruck erlaubt die Kombination der Effekte von *Akzent1* und *Akzent2* (in dieser Reihenfolge!) bei allen *erlaubten* Zeichen. Die Bedeutungen von *Akzent1* und *Akzent2* müssen zuvor mit Hilfe von `\kmod` definiert worden sein.

Folgendes Beispiel stammt aus der Datei `greek.kmap`:

```
\kmod ; acute aeioyvhAEIOYVH \kmod : umlaut iyIY \kcomb acute umlaut iyIY
```

Diese Zeilen erlauben es, „`;:i`“ einzugeben und auf diese Weise „`\'{\"{i}}`“ zu erzeugen (ǐ). In diesem Fall löscht die Backspace-Taste das letzte gedrückte Zeichen. Wenn Sie also `;:` Backspace `i` eingeben, erhalten Sie „`\'(i)`“ (í).

4.2.2. Die `.cdef`-Datei

Nachdem LyX die `.kmap`-Datei verarbeitet hat, legt eine `.cdef`-Datei fest, wie die einzelnen Symbole im gegenwärtigen Zeichensatz dargestellt werden sollen. Die LyX-Distribution enthält wenigstens die Dateien `iso8859-1.cdef` und `iso8859-2.cdef`.

Generell besteht eine `.cdef`-Datei aus einer Reihe von Deklarationen der folgenden Form:

Position_im_Zeichensatz Zeichenkette

Um beispielsweise der Zeichenkette (*Ausgabe* im vorigen Abschnitt) „`'{e}`“ das entsprechende Zeichen im Zeichensatz ISO-8859-1 (233) zuzuweisen, benutzt man folgenden Ausdruck:

```
233 "\\'{e}"
```

Wieder müssen den Zeichen „\“ und „`"`“ ein Backslash vorangestellt werden. Beachten Sie, dass es durchaus möglich ist, dass dasselbe Zeichen (sinnvoll) zwei verschiedene Strings repräsentieren kann. Zum Beispiel gibt es in `iso-8859-7.cdef` die Zeilen:

```
192 "\\'{\\\"{i}}"
192 "\\\"{\\'{i}}"
```

Wenn LyX kein passendes Zeichen für einen String finden kann, der durch eine Tastensequenz erzeugt wurde, wird es versuchen, falls der String wie ein akzentuierter Buchstabe aussieht, auf dem Bildschirm den Buchstaben mit Akzent selbst zu zeichnen.

4.2.3. Tote Tasten definieren

Es gibt noch eine zweite Möglichkeit, internationale Buchstaben mit Hilfe von *toten Tasten (dead keys)* zu erzeugen. Eine tote Taste erzeugt in Kombination mit einem Buchstaben ein akzentuiertes Zeichen. Im Folgenden erläutern wir für Illustrationszwecke, wie man eine wirklich simple tote Taste definiert.

Nehmen wir an, Sie benötigen das Zirkumflex-Zeichen („ˆ“). Hierzu können Sie der Taste `^` den LyX-Befehl `accent-circumflex` zuweisen. Wann immer Sie danach diese Taste gefolgt von einem Buchstaben drücken, wird dieser Buchstabe einen Zirkumflex haben. Die Sequenz `^E` produziert also den Buchstaben ê. Wenn Sie nach der Taste `^` die Leertaste drücken, wird nur der Akzent ausgegeben. Beachten Sie diesen letzten Punkt! Wenn Sie eine Taste an eine tote Taste anbinden, müssen Sie das Zeichen, das diese Taste normalerweise erzeugt, an eine andere Taste anbinden (oder immer zusätzlich die Leertaste drücken). Daher ist es keine gute Idee, die Taste `,` an `accent-cedilla` anzubinden, da Sie dann Kommata sehr mühsam eingeben müssen.

Ein üblicher Weg, tote Tasten zu belegen, ist es, die Modifizierungstasten (`Meta/Alt`, `Strg` oder `Umschalt`) mit einem Zeichen (wie `~` oder `,` oder `^`) zu kombinieren.

Ein anderer setzt die Verwendung der Linux-Programme `xmodmap` und `xkeycaps` voraus, mit denen die spezielle Taste `Mode_Switch` definiert wird. `Mode_Switch` verhält sich ähnlich wie `Umschalt` und erlaubt es Ihnen, Tasten an Akzentzeichen zu binden. Sie können damit auch Tasten in tote Tasten umwandeln, indem Sie diese Tasten an Tastensymbole wie `usldead_cedilla` anbinden und diese dann an einen entsprechenden LyX-Befehl.[6] Sie können im Grunde jede Taste als `Mode_Switch` definieren: Eine der `Strg`-Tasten, eine freie Funktionstaste usw. Die LyX-Befehle, die Akzente produzieren, sind im Handbuch LyX-Funktionen dokumentiert. Schauen Sie nach dem Eintrag `accent-acute`. Sie finden dort eine komplette Liste.

4.2.4. Ihre Sprachkonfiguration einstellen

Über können Sie die Sprache der Benutzeroberfläche konfigurieren.

[6] Hinweis von John Weiss: Genau das mache ich in meinen Dateien `~/.lyx/lyxrc` und `~/.xmodmap`. I habe meine Taste `Scroll Lock` (`Rollen`) als `Mode_Shift` definiert und eine Reihe der Tastensymbole mit dem Präfix `usldead_` an Prozesse wie die folgenden gebunden: Scroll Lock-^ und Scroll Lock-~. So komme ich zu meinen Akzentbuchstaben.

5. Installieren neuer Textklassen, Layouts und Vorlagen

In diesem Kapitel wird beschrieben, wie Sie beim Installieren neuer Layout- und Vorlagendateien vorgehen müssen. Außerdem geben wir eine kleine Auffrischung, wie man neue Dokumentenklassen für LaTeX korrekt installiert.

Zunächst möchten wir noch einmal ein paar Worte darüber verlieren, wie LyX und LaTeX miteinander verbunden sind, da dies für das Folgende wichtig ist. Wichtig zu wissen ist vor allem, dass LyX selbst im Grunde sehr wenig Konkretes über LaTeX weiß. Tatsächlich ist LaTeX aus der Sicht von LyX nur eines mehrerer „Backend-Formate“ für die es eine Ausgabe erzeugen kann. Andere solche Formate sind DocBook, einfacher Text und XHTML. Nun ist LaTeX aber natürlich ein besonders wichtiges Format. Der Punkt jedoch ist, dass sehr wenig der Informationen, die LyX über LaTeX hat, im Programm selbst festgeschrieben sind.[1] Vielmehr bezieht es diese Informationen, selbst im Fall der Standardklassen wie `article.cls`, aus so genannten „Layout-Dateien“. Genauso wenig weiß LyX über DocBook oder XHTML. Was es weiß, bezieht es aus Layout-Dateien.

Eine Layout-Datei können Sie sich als eine Art Übersetzungshandbuch vorstellen, in dem für eine bestimmte Dokumentklasse festgelegt wird, wie LyX-Konstrukte – Absätze mit den entsprechenden Stilen, bestimmte Einfügungen usw. – und LaTeX-DocBook- oder XHTML-Konstrukte korrespondieren. Fast alles, was LyX beispielsweise über die Standard-Artikelklasse von LaTeX (`article.cls`) weiß, ist in der Layout-Datei `article.layout` und in verschiedenen anderen Dateien, die diese einbindet, festgeschrieben. Daher sollten Sie, wenn Sie vorhaben, selbst eine Layout-Datei zu schreiben, zuallererst die vorhandenen Layout-Dateien studieren. Am Besten beginnen Sie mit der Datei `stdsections.inc`, die in `article.layout`, `book.layout` und vielen anderen Layout-Dateien für Dokumentklassen eingebunden wird. Dort werden die Abschnittsüberschriften und ähnliches definiert: `stdsections.inc` informiert also LyX darüber, wie Absätze, die als Abschnitt, Unterabschnitt usw. markiert sind, in LaTeX, DocBook und XHTML ausgegeben werden sollen. Die Datei `article.layout` bindet im Grunde einfach nur verschiedene solcher `std*.inc`-Dateien ein.

Layout-Dateien definieren aber nicht nur die LyX-LaTeX-Korrespondenz, sondern sie legen auch fest, wie die LyX in LyX selbst (auf dem Bildschirm) dargestellt werden. Die Tatsache, dass Layout-Dateien diese beiden Aufgaben gleichzeitig haben, führt

[1] Manche Befehle sind so komplex, dass sie in LyX festgeschrieben sind. Aber die Entwickler betrachten dies allgemein schlechte Lösung.

häufig zu Verwirrung, denn die Aufgaben sind strikt getrennt. Wenn Sie LyX sage, wie ein bestimmtes Konstrukt in LaTeX dargestellt werden soll, haben Sie dem Programm noch nicht gesagt, wie das Konstrukt in LyX dargestellt werden soll. Umgekehrt weiß LyX, wenn Sie festlegen, wie ein Konstrukt in LyX darzustellen ist, damit nicht, wie es in LaTeX übersetzt werden muss (und erst recht nicht LaTeX, wie es darzustellen ist). Das heißt, Sie müssen, wenn Sie ein neuen LyX-Konstrukt definieren, immer zwei verschiedene und getrennte Dinge tun: (1.) LyX instruieren, wie dieses in LaTeX übersetzt werden soll und (2.) LyX instruieren, wie es das Konstrukt darstellen soll.

Analoges gilt natürlich für die anderen „Backend-Formate“, die LyX unterstützt. Allerdings ist XHTML in mancher Hinsicht eine Ausnahme, denn in diesem Fall *ist* LyX bis zu einem gewissen Grad in der Lage, aus den Informationen über die Darstellung in LyX Informationen über die Ausgabe in einem Browser (über CSS) abzuleiten. Aber auch in diesem Fall bleibt die prinzipielle Trennung in Kraft und sollte berücksichtigt werden. Weitere Ausführungen dazu finden Sie in Abschnitt Kap. 5.4.

5.1. Installation eines neuen LaTeX-Paketes

Bei manchen TeX-Installationen fehlt möglicherweise das eine oder andere Paket, das Sie gerne mit LyX verwenden würden. Zum Beispiel wollen Sie vielleicht FoilTeX verwenden, ein Paket zur Erstellung von Dias und Folien für Overheadprojektoren. Moderne LaTeX-Distributionen wie TeXLive (2008 oder neuer) oder MiKTeX besitzen ein grafisches Programm, um solche Pakete zu installieren. Bei MiKTeX etwa starten Sie das Programm „Package Manager“, um eine Liste mit den verfügbaren Paketen zu bekommen. Um eines davon zu installieren, rechtsklicken Sie oder benutzen den entsprechenden Werkzeugleistenknopf.

Falls Ihre LaTeX-Distribution keinen Paketmanager besitzt, oder falls das Paket nicht direkt über Ihre Distribution verfügbar ist, folgen Sie diesen Schritten um es manuell zu installieren:

1. Besorgen Sie sich das Paket von CTAN oder einer anderen Quelle.

2. Falls das Paket eine Datei mit der Endung „`.ins`“ enthält (was bei FoilTeX der Fall ist), dann öffnen sie eine Kommandozeile wechseln in das Verzeichnis der Datei und führen den Befehl `latex foiltex.ins` aus. Sie haben damit dass Paket entpackt und haben alle Dateien, um es zu installieren. Die meisten LaTeX-Pakete sind nicht gepackt und man kann direkt mit dem nächsten Schritt weitermachen.

3. Nun müssen Sie entscheiden, ob das Paket für alle Nutzer oder nur für Sie verfügbar sein soll.

   a) Bei *nix Systemen (Linux, OSX, etc.) installieren Sie, wenn Sie das Paket für alle Nutzer installieren möchten, dieses in den „lokalen“ TeX Ordner; anderenfalls installieren Sie es in den eigenen „Benutzer“-TeX-Ordner. Wo man diese Ordner anlegt, sofern sie nicht schon existieren, hängt von Ihrem

System ab. Dazu schauen Sie in die Datei `texmf.cnf`.[2] Der Ort des lokalen TeX-Ordners ist in der Variable `TEXMFLOCAL` definiert; es ist üblicherweise der Pfad `/usr/local/share/texmf/` oder `/usr/local/texlive/XXXX`, wobei XXXX das Jahr der installierten TeXLive-Distribution ist. Der Ort des Benutzer-TeX-Ordners ist in der Variable `TEXMFHOME` definiert und ist üblicherweise der Pfad `$HOME/texmf/` oder `$HOME/.texliveXXXX`. (Wenn diese Variablen nicht vordefiniert sind, müssen Sie diese selbst definieren.) Sie brauchen wahrscheinlich Administrator-Rechte um in den lokalen TeX-Ordner zu schreiben, beim Benutzer-TeX-Ordner ist die nicht nötig.
Allgemein empfiehlt es sich, Pakete in den Benutzer-TeX-Ordner zu installieren, da dieser nicht verändert oder gar überschrieben wird, wenn Sie ihr System aktualisieren. Des Weiteren wird er zusammen mit Ihren Nutzerdaten gesichert, wenn Sie ein Backup machen (was Sie natürlich regelmäßig tun).

b) Unter Windows gehen Sie, wenn Sie das Paket für alle Nutzer installieren möchten, in den Ordner, in dem LaTeX installiert ist und wechseln dort in das Verzeichnis `~tex\latex` (verwendet man MiKTeX, wäre es standardmäßig der Ordner `~:\Programme\MiKTeX\tex\latex`). Legen Sie dort einen neuen Ordner mit dem Namen `foiltex` an und kopieren Sie alle Dateien des Pakets hinein. Wenn das Paket nur für den aktuellen Benutzer verfügbar sein soll bzw. Sie keine Administrator-Rechte haben, tun Sie dasselbe, aber im lokalen LaTeX-Ordner. Bei MiKTeX 2.9 wäre das unter WinXP der Ordner
~:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten\MiKTeX\2.9\tex\latex
, unter WinVista wäre es der Ordner
~:\Users\<Benutzername>\AppData\Roaming\2.9\MiKTeX\tex\latex .

4. Jetzt muss man LaTeX nur noch mitteilen, dass es neue Dateien gibt. Die ist je nach LaTeX-Distribution anders:

   a) Bei TeXLive führen Sie von einer Kommandozeile den Befehl `texhash` aus. Wenn Sie das Paket für alle Nutzer installiert haben, brauchen sie dazu wahrscheinlich Administrator-Rechte.

   b) Bei MiKTeX starten Sie, wenn Sie das Paket für alle Nutzer installiert haben, das Programm „Settings (Admin)“ und drücken dann auf den Kopf „Refresh FNDB“. Anderenfalls starten Sie das Programm „Settings“ und machen dasselbe.

5. Nun muss man LyX noch mitteilen, dass es neue Pakete gibt. Verwenden Sie dazu in LyX das Menü und starten LyX danach neu.

[2] Diese befindet sich normalerweise im Ordner `$TEXMF/web2c`. Falls nicht, führen Sie den Befehl `kpsewhich texmf.cnf` aus, um sie zu lokalisieren.

Nun ist das Paket installiert. In unserem Beispiel wird nun die Dokumentklasse `FoilTeX` im Menü Dokument ▷ Einstellungen ▷ Dokumentklasse (unter „Präsentationen“) verfügbar sein.

Möchten sie eine LaTeX-Dokumentklasse verwenden, die generell nicht im Menü Dokument ▷ Einstellungen ▷ Dokumentklasse gelistet ist, müssen Sie dafür selbst eine Layout-Datei erstellen. Dies ist das Thema des nächsten Abschnitts.

5.2. Layout-Dateitypen

Dieser Abschnitt beschreibt die verschiedenen Arten von LyX-Dateien, die Layout-Informationen enthalten können. Dort werden verschiedene Absatz- und Zeichenstile definiert, es wird bestimmt, wie LyXdiese darstellt und wie sie in LaTeX, DocBook, XHTML oder sonstige Formate exportiert werden sollen.

Wir bieten Ihnen hier eine umfassende Dokumentation zum Verfassen von Layout-Dateien. Da es aber so viele verschiedene nur schon von LaTeX unterstütze Dokumenttypen gibt, können wir nicht jedes Problem, vor dem Sie vielleicht stehen, behandeln. Die LyX-Benutzer-Mailingliste wird aber von vielen Leuten gelesen, die Erfahrung mit dem Layout-Design haben und die Ihnen gerne helfen. Zögern Sie also nicht, offene Fragen dort zu stellen.

Wenn Sie vor der Aufgabe stehen, eine neue Layout-Datei zu schreiben, ist es zunächst einmal sehr hilfreich, die Layouts zu studieren, die LyX bereits enthält. Wenn Sie eine Layout-Datei für eine LaTeX-Dokumentklasse geschrieben haben, die auch von andern verwendet wird, sollten Sie überlegen, dieses Layout auf der Seite ‚Layouts‘ des LyX-Wiki zu verlinken oder es vielleicht sogar an die LyX-Entwicklerliste zu schicken, damit es in LyX selbst aufgenommen wird.[3]

5.2.1. Layout-Module

Wir haben bislang immer von ‚Layout-Dateien‘ gesprochen, tatsächlich gibt es aber unterschiedliche Arten von Dateien, die Layout-Informationen enthalten. Layout-Dateien im engeren Sinn haben die Endung `.layout` und stellen LyX Information über Dokumentklassen zur Verfügung. Seit LyX 1.6 gibt es jedoch auch so genannte Layout-Module, die zusätzliche Layout-Informationen enthalten können. Sie haben die Endung `.module`. Module verhalten sich, grob gesprochen, so zu LaTeX-Paketen, wie sich Layout-Dateien zu LaTeX-Klassen verhalten, und manche Module – wie etwa das Modul `endnotes` – bieten spezifische Unterstützung für ein bestimmtes Paket. In einiger Hinsicht sind Module ganz ähnlich wie die von Layout-Dateien eingebundene, Dateien[4] – bspw. `stdsections.inc`. So sind sie nicht an eine bestimmte Dokumentklasse gebunden, sondern können in verschiedenen Klassen genutzt werden. Ein

[3] Beachten Sie, dass LyX der *General Public License* (GPL) unterliegt. Ihr Beitrag müsste auch dieser Lizenz unterworfen werden.

[4] Diese haben normalerweise die Endung `.inc`.

wichtiger Unterschied ist jedoch, dass die eingebundenen Dateien nur genutzt werden können, wenn die Layout-Datei hierfür bearbeitet wird. Module hingegen können einfach im Dialog ausgewählt werden.

Die Erstellung eines Moduls ist der einfachste Weg, um mit dem Verfassen von Layouts zu beginnen, denn Module können sehr einfach sein und etwa nur einen einzelnen Absatzstil oder eine benutzerdefinierte Einfügung definieren. Auf der anderen Seite können Module im Prinzip aber alles enthalten, was auch Layout-Dateien enthalten.

Nachdem Sie ein neues Modul geschrieben und dieses in den Ordner `layouts/` in `UserDir` kopiert haben, müssen Sie LyX rekonfigurieren () und dann neu starten, damit das Modul in der Benutzeroberfläche zugänglich wird. Falls Sie existierende Module modifizieren, ist diese Prozedur nicht nötig: Änderungen werden verfügbar, sobald Sie in irgend etwas ändern und OK drücken. *Wir raten aber dringend, dass Sie Ihre Dokumente sichern, bevor Sie dies tun.* Um sicherzugehen, *sollten Sie lieber nicht an Dokumenten arbeiten, während Sie dort verwendete Module edieren.* Obwohl die Entwickler natürlich alles tun, um LyX in solchen Situationen stabil zu halten, können Fehler in Ihren Modulen manchmal ungewollte Effekte haben.

5.2.1.1. Lokales Format

Module sind für LyX das, was Pakete für LaTeX sind. Manchmal braucht man jedoch möglicherweise eine spezifische Einfügung oder einen Absatzstil nur für ein bestimmtes Dokument. Hierfür extra ein Modul zu schreiben, ist mit Kanonen auf Spatzen geschossen. Viel besser für solche Zwecke ist das, was wir „lokales Format“ nennen.

Sie finden dieses in unter Lokales Format. In das große Eingabefeld dort können Sie alles eingeben, was Sie in eine Layout-Datei oder ein Modul eingeben können. Das lokale Format eines Dokuments können Sie sich im Grunde als Modul vorstellen, das nur dem entsprechenden Dokument zugänglich ist (alle Definitionen werden direkt im Dokument gespeichert und können so auch leicht mit dem Dokument weitergegeben werden). Wie in Layout-Dateien und Modulen auch sollten Sie das `Format` spezifizieren (s. u.). Dabei ist jedes (bekannte) Format möglich, sinnvollerweise sollten Sie aber das jeweils aktuelle verwenden (das aktuelle Format in LyX ist).

Sobald Sie etwas in das Eingabefeld in `Lokales Format` eingegeben haben, aktiviert LyX den Knopf „Validieren“ unter diesem Feld. Wenn Sie darauf klicken (was Sie tun müssen, bevor Sie OK klicken können), überprüft LyX, ob das, was Sie eingegeben haben, im spezifizierten Format auch korrekt ist. LyX meldet, ob dies der Fall ist, aber leider nicht, *was* gegebenenfalls falsch ist. Wenn Sie LyX von einer Konsole gestartet haben, wird dies aber dort ausgegeben. Solange es Fehler in der Definition gibt, können Sie Ihr lokales Format nicht speichern.

Die Warnungen, die wir am Ende des vorherigen Abschnitts ausgesprochen haben, gelten auch hier: Spielen Sie nicht mit dem lokalen Format herum, während Sie am Dokument arbeiten, besonders nicht, wenn Sie ungesicherte Änderungen haben. Wenn Sie dies beachten, kann das lokale Format (in einem Testdokument) aber als sehr hilfreiche Methode dienen, um Layout-Ideen auszuprobieren oder etwa Modulideen auszuprobieren.

5.2.2. Layout für .sty-Dateien

Wenn Sie eine neue LaTeX-Klasse unterstützen wollen, heißt das in der Regel, dass Sie entweder einen LaTeX-Stil (`.sty`) oder eine LaTeX 2_ε-Klasse (`.cls`) vorliegen haben, für den es ein LyX-Layout zu schreiben gilt. Ersteres ist normalerweise relativ einfach. Letzteres ist zumeist etwas anspruchsvoller. Wenn Sie eine neue DocBook-DTD unterstützen wollen, können Sie die Anmerkungen teilweise übertragen.

Der einfachere Fall ist, wie gesagt, dass die Dokumentklasse, die Sie unterstützen wollen, als Stildatei vorliegt, die ihrerseits auf eine bereits unterstützte LaTeX-Klasse aufsetzt. Zur Illustration nehmen wir an, dass die Stildatei `myclass.sty` heißt und dass Sie zusammen mit der LaTeX-Klasse `report.cls`, die eine Standardklasse ist, verwendet werden soll.

Beginnen Sie damit, dass Sie eine Kopie der existierenden Layout-Datei `report.layout` in Ihrem lokalen Verzeichnis unter dem Namen der neuen Klasse, `myclass.layout`, ablegen:[5]

```
cp report.layout ~/.lyx/layouts/myclass.layout
```

Dann bearbeiten Sie `myclass.layout` und ändern die Zeile

```
\DeclareLaTeXClass{Report (Standard Class)}
```

in

```
\DeclareLaTeXClass[report, myclass.sty]{Report (My Class)}
```

Dann fügen Sie gegen Anfang der Datei ein:

```
Preamble
    \usepackage{myclass}
EndPreamble
```

Starten Sie LyX und wählen Sie . Dann starten Sie LyX neu und beginnen ein neues Dokument. Sie sollten nun „Report (My Class)“ in der Liste der auswählbaren Dokumentklassen in finden. Es ist wahrscheinlich, dass sich einige der Überschriftenbefehle und andere Dinge in Ihrer neuen Klasse anders verhalten als in der Basisklasse (hier: Report). Sie können mit den vorhandenen Definitionen spielen, um dies anzupassen. Die Layout-Informationen für Überschriften sind nicht in `report.layout` selbst, sondern in `stdsections.inc` enthalten, aber Sie müssen deren Inhalt nicht kopieren, um die Definitionen zu verändern. Statt dessen können Sie Änderungen einfach in Ihrer neuen Layout-Datei vornehmen, und zwar nach der Zeile `Input stdclass.inc`, die ihrerseits `stdsections.inc` einbindet. Sie können zum Beispiel folgenden Zeilen einfügen:

[5] Mit dem lokalen Verzeichnis meinen wir das `UserDir`, siehe oben Abschnitt Kap. 2.2.

```
Style Chapter
    Font
        Family Sans
    EndFont
End
```

Dies ändert die Schrift, die in LyX zur Darstellung von Kapitelüberschriften verwendet wird, in eine serifenlose. Die bestehende Definition des Kapitelstils wird so überschrieben (bzw. ergänzt).

Ihr neues Paket stellt vielleicht auch neue Befehle oder Umgebungen bereit, die nicht in der Basisklasse enthalten sind. In diesem Fall können Sie diese zur Layout-Datei hinzufügen Abschnitt Kap. 5.3 erläutert dies genauer.

Falls `myclass.sty` mit verschiedenen Dokumentklassen verwendet werden kann, und vielleicht auch falls nicht, ist es vielleicht noch einfacher, statt einer Layout-Datei ein Modul zu schreiben. Das einfachst mögliche Modul würde so aussehen:

```
#\DeclareLyXModule{My Class}
#DescriptionBegin
#Support for myclass.sty.
#DescriptionEnd
Format
Preamble
    \usepackage{mypkg}
EndPreamble
```

Komplexere Module modifizieren vielleicht die Definition vorhandener Konstrukte oder ergänzen neue. Auch dies wird in Abschnitt Kap. 5.3 ausführlicher besprochen.

5.2.3. Layout für .cls-Dateien

Hier gibt es zwei Möglichkeiten. Eine ist, dass die LaTeX-Klasse selbst auf einer bestehenden anderen LaTeX-Klasse basiert. Viele Klassen für Dissertationen basieren etwa auf `book.cls`. Um zu prüfen, ob das in Ihrem Fall so ist, schauen Sie in Ihrer Klasse, ob es eine Zeile wie diese gibt:

```
\LoadClass{book}
```

Falls ja, können Sie im Großen und Ganzen wie im vorherigen Abschnitt beschrieben vorgehen, mit Ausnahme der Zeile `DeclareLaTeXClass`. Falls Ihre neue Klasse `thesis` heißt und auf `book` basiert, sollte diese Zeile lauten:[6]

```
\DeclareLaTeXClass[thesis,book]{Thesis}
```

[6] Und am einfachsten ist es, wenn Sie die Layout-Datei `thesis.layout` nennen: LyX nimmt an, wenn nicht anders angegeben (siehe unten), dass die Layout-Datei denselben Namen hat wie die LaTeX-Klasse, die sie unterstützt.

Die zweite Möglichkeit ist, dass die neue Klasse auf keiner anderen basiert. Dann müssen Sie wohl ein komplett neues Layout erstellen. Aber auch dann können Sie Bestandteile aus anderen Layouts übernehmen, die sich (partiell) ähnlich verhalten und diese dann gegebenenfalls modifizieren. Zumindest sollten Sie eine existierende Layout-Datei als Startpunkt nehmen, damit Sie sehen, um was Sie sich alles kümmern müssen. Die Einzelheiten folgen unten.

5.2.4. Vorlagen erstellen

Sobald Sie eine Layout-Datei für eine neue Dokumentklasse geschrieben haben, werden Sie vielleicht auch eine Vorlage dafür schreiben wollen. Eine Vorlage ist eine Art Tutorium für Ihr Layout, sie zeigt, wie diese verwendet wird (unter Verwendung von Blindtext). Schauen Sie sich einfach bestehende Vorlagen an, um zu sehen, wie man dies gestalten kann.

Vorlagen sind im Grunde normale LyX-Dokumente und werden auch so erstellt. Der einzige Unterschied ist, dass normale LyX-Dokumente alle möglichen Einstellungen enthalten, beispielsweise hinsichtlich Schriftarten und Seitengrößen. Vorlagen enthalten diese oft nicht, um die Anwender nicht zu sehr einzuschränken. Daher werden bei Vorlagen die entsprechenden Befehle oft entfernt (wie `\font_roman` oder `\papersize`). Dies können Sie mit einem einfachen Text-Editor, etwa `vi` oder `notepad`, erledigen.

Legen Sie die fertige Vorlage in `UserDir/templates/` ab, kopieren Sie ggf. die, die Sie verwenden, von `LyXDir/templates/` in dasselbe Verzeichnis und ändern Sie den Pfad zu den Vorlagen in unter Pfade.

Beachten Sie übrigens, dass es eine Vorlage mit besonderer Funktion gibt: `defaults.lyx`. Diese wird jedes Mal geladen, wenn Sie mit ein neues Dokument erstellen, um sinnvolle Voreinstellungen vorzunehmen. Um diese Vorlage zu verändern, müssen Sie nur ein Dokument mit den gewünschten Einstellungen öffnen und in den Knopf Als Dokument-Voreinstellung speichern betätigen.

5.2.5. Alte Layout-Dateien auf den neuesten Stand bringen

Das Format der Layout-Dateien ändert sich mit jeder LyX-Version. Daher müssen die Layout-Dateien in das neue Format konvertiert werden. Wenn LyX eine Layout-Datei eines älteren Formats liest, ruft es automatisch das Skript `layout2layout.py` auf um es in eine temporäre Datei im aktuellen Format zu konvertieren. Die Originaldatei wird nicht verändert. Wenn Sie die Layout-Datei öfter verwenden, dann können Sie sie permanent in das neue Format konvertieren, so dass LyX dies nicht jedes Mal tun muss. Um das zu tun, machen sie Folgendes:

1. Benennen Sie `MeineKlasse.layout` in `MeineKlasse.alt` um.
2. Rufen Sie den Befehl
   `python LyXDir/scripts/layout2layout.py myclass.old myclass.layout`
   auf, wobei `LyXDir` der Name Ihres LyX-Systemverzeichnisses ist.

Beachten Sie, das manuelle Konvertierungen keine eingefügten Dateien mit konvertieren. Diese müssen separat konvertiert werden.

5.2.6. Cite-Engine-Dateien

Cite-Engine-Dateien (Endung `*.citeengine`) stellen eine besondere Form von Layout-Dateien dar. Sie finden diese im Unterordner `citeengines/`. Sie dienen dazu, die Spezifika von LaTeX-Paketen zu definieren, welche zur Erstellung von Literaturverzeichnissen verwendet werden (etwa Natbib, Jurabib oder Biblatex), legen aber auch fest, wie normale BibTeX-Literaturverweise (ohne zusätzliche Pakete) in LyX dargestellt werden.

Genauer wird dort spezifiziert, welche Pakete LyX laden muss, welche Befehle für Literaturverweise verfügbar sind, wie diese in LyX dargestellt werden (im Arbeitsbereich, den Dialogen oder im Kontextmenü) und wie sie in der XHTML- und Textausgabe erscheinen. Außerdem legen diese Dateien Stilvarianten (Autor-Jahr, Nummerisch etc.) und deren Spezifika fest. Die Cite-Engine-Dateien werden auch verwendet, um die Optionen zu bestimmten, die in Dokument ▷ Einstellungen ▷ Literaturverzeichnis erscheinen.

Auch wenn eine Cite-Engine-Datei im Grunde eine normale Layout-Datei ist, die theoretisch jede beliebige Layout-Information enthalten könnte, enthält sie üblicherweise nur literaturspezifische Parameter wie `MaxCiteNames`, `CiteFramework`, `CiteEngine` und `CiteFormat`-Blöcke. Die Syntax der letzten beiden wird in Abschnitt Kap. 5.3.14 und Kap. 5.3.15 sowie auch in den Dateien selbst beschrieben.

5.3. Das Layout-Dateiformat

Die folgenden Abschnitte beschreiben wie Layout-Dateien aufgebaut sind und erstellt werden. Wir empfehlen bei der Erstellung von Layouts langsam zu beginnen und sich Stück für Stück vorzuarbeiten. Es ist nicht wirklich schwer, jedoch sind die möglichen Optionen manchmal etwas erschlagend, besonders wenn man zu viele davon auf einmal ausprobiert. Am einfachsten ist es, wenn man bestehende Layout-Dateien von LyX als Beispiel nimmt oder diese umgestaltet.

Beachten Sie dass alle Marken in Layout-Dateien nicht durch Groß- und Kleinschreibung zu unterscheiden sind. Das bedeutet dass `Style`, `style` und `StYlE` dieselbe Marke sind. Die möglichen Argumente für die Marken sind hinter ihnen in eckigen Klammern angegeben. Das voreingestellte Argument ist *`hervorgehoben`*. Wenn das Argument einen Datentyp hat wie „string“ oder „float“, wird die Voreinstellung so angezeigt: `float=`*`default`*.

5.3.1. Deklaration einer neuen Textklasse und Klassifikation

Zeilen, die mit einem `#` beginnen, sind Kommentare. Mit einer Ausnahme: alle Textklassen sollten mit Zeilen ähnlich wie den folgenden beginnen:

```
#% Do not delete the line below; configure depends on this[7]
#  \DeclareLaTeXClass{Article (Standard Class)}
#  \DeclareCategory{Articles}
```

Die zweite und dritte Zeile wird benötigt, wenn Sie LyX konfigurieren. Die Textklassen-Datei wird von dem LaTeX-Skript chkconfig.ltx gelesen, und zwar in einem speziellen Modus, in dem #-Zeichen ignoriert werden. Die erste Zeile ist einfach ein LaTeX-Kommentar, in der zweiten muss die Textklasse deklariert werden und die dritte Zeile enthält die optionale Klassifikation der Klasse. Eine Datei namens `article.layout`, die mit diesen beiden Zeilen beginnt, definiert eine Textklasse mit dem Namen article (der Name der Layout-Datei) und benutzt die LaTeX-Dokumentenklasse `article.cls` (Standard ist, denselben Namen wie das Layout zu verwenden). Die Zeichenkette „Article (Standard Class)“, die oben erscheint, ist auch die Beschreibung, die später im Popup-Menü Dokument ▷ Einstellungen auftaucht. Die Kategorie („Articles“ im Beispiel) wird auch im Dialog Dokument ▷ Einstellungen verwendet: Die Textklassen werden nach diesen Kategorien gruppiert (was üblicherweise Genres sind, typische Kategorien sind also „Artikel“, „Bücher“, „Berichte“, „Briefe“, „Präsentationen“, „Lebensläufe“ usw.). Wenn keine Kategorie deklariert wurde, wird die Klasse in die Gruppe „Nicht kategorisiert“ getan.

Angenommen, Sie möchten Ihre eigene Textklasse schreiben, welche die LaTeX-Dokumentklasse `article` benutzt, in der Sie aber das Aussehen der Kopfzeile verändert haben. Wenn Sie dann Ihre Textklasse in eine Datei namens `myarticle.layout` schreiben, sollten die ersten beiden Zeilen der Datei etwa so aussehen:

```
#% Do not delete the line below; configure depends on this
#  \DeclareLaTeXClass[article]{Article (with My Own Headings)}
#  \DeclareCategory{Articles}
```

Auf diese Weise deklarieren Sie eine Textklasse `myarticle`, die die LaTeX-Dokumentklasse `article.cls` verwendet und (im Popup-Menü) beschrieben wird mit: Article (with My Own Headings). Falls Ihre Textklasse auch noch von weiteren Paketen abhängt, können Sie das so angeben:

```
#% Do not delete the line below; configure depends on this
#  \DeclareLaTeXClass[article,foo.sty]{Article (with My Own Headings)}
#  \DeclareCategory{Articles}
```

Dadurch wird angezeigt, dass Ihre Klasse auch das Paket `foo.sty` verwendet. Schließlich können Sie auch Klassen für DocBook festlegen. Eine typische Deklaration sieht so aus:

```
#% Do not delete the line below; configure depends on this
# \DeclareDocBookClass[article]{SGML (DocBook Article)}
```

[7] zu Deutsch: Löschen Sie die folgenden Zeilen nicht, da die Konfiguration davon abhängt

Diesen Deklarationen kann außerdem ein optionaler Parameter zugefügt werden, der den Namen der Dokumentenklasse festlegt (hier ist aber keine Liste erlaubt).

Eine Layout-Deklaration hat demnach, noch einmal zusammengefasst, die Form

```
#  \DeclareLaTeXClass[Klasse,Paket.sty]{Layout-Beschreibung}
#  \DeclareCategory{Kategorie}
```

Die Klasse muss nur dann explizit genannt werden, wenn der Name der LaTeX-Klasse und der der LyX-Layoutdatei divergieren, oder wenn Sie Pakete laden. Wenn kein optionales Argument angegeben ist, nimmt LyX an, dass die Klasse denselben Namen wie die Layout-Datei hat.

Wenn Sie eine Textklasse nach Ihrem Geschmack erstellt haben, müssen Sie die Datei nur noch in das Verzeichnis `LyXDir/layouts/` oder nach `UserDir/layouts` kopieren und unter LyX den Menüpunkt Werkzeuge ▷ Neu konfigurieren auswählen. Nach dem Neustart von LyX sollte Ihre Textklasse im Popup-Menü Dokument ▷ Einstellungen auswählbar sein.

Ist das Layout einmal von LyX erkannt, können Sie Änderungen direkt auch ohne Neustart sehen.[8] Sie können ein Neuladen des Layouts mithilfe der LyXFunktion `layout-reload` erzwingen. Für diese Funktion gibt es standardmäßig kein Tastenkürzel. Sie können selbst eines definieren oder aber einfach die Funktion in den *Minibuffer* eingeben.

Aber Achtung: `layout-reload` ist ein komplexes Feature. Wir ersuchen Sie dringend, Ihr Dokument zu speichern, bevor Sie die Funktion verwenden. Eigentlich sollten Sie keine Layout-Dateien bearbeiten, während Sie an einem wichtigen Dokument arbeiten. Verwenden Sie ein Textdokument. Fehler in der Layout-Datei können üble Effekte haben. Insbesondere können sie dazu führen, dass LyX das aktuelle Layout für ungültig hält und in ein anderes wechselt.[9] Zwar tut das LyX-Team alles, um LyX in solchen Situationen stabil zu halten, aber Vorsicht ist bekanntlich die Mutter der Porzellankiste.[10]

5.3.2. Die Modul-Deklaration

Ein Modul muss mit einer Zeile wie die folgende beginnen:

```
#\DeclareLyXModule[endnotes.sty]{Endnotes}
```

Das benötigte Argument in geschweiften Klammern ist der Name des Moduls, wie es in Dokument ▷ Einstellungen ▷ Module erscheinen soll. Das Argument in eckigen Klammern ist optional: Es deklariert alle LaTeX-Pakete, die das Modul benötigt. Es ist

[8] Das ist erst ab LyX 1.6 der Fall. Vorher war die Bearbeitung von Layout-Dateien mühsam und zeitraubend, weil man LyX ständig neu starten musste.

[9] Ganz schlimme Fehler können sogar dazu führen, dass LyX abstürzt, da manche Fehler dazu führen, dass LyX *gar keine* Layout-Informationen mehr lesen kann. Seien Sie also bitte vorsichtig.

[10] Wo wir gerade beim Rat geben sind: Machen Sie regelmäßig Backups. Und seien Sie nett zu Ihren Nachbarn.

außerdem möglich die Form `von->zu` als optionales Argument zu verwenden, das angibt, dass das Modul nur verwendet werden kann, wenn es eine Konvertierungsmöglichkeit zwischen den Formaten „`von`“ und „`zu`“ gibt.

Der Modul-Deklaration sollten Zeilen zur Beschreibung wie die folgenden folgen:[11]

```
#DescriptionBegin
#Adds an endnote command, in addition to footnotes.
#You will need to add  \theendnotes in TeX code where you
#want the endnotes to appear.
#DescriptionEnd
#Requires: somemodule | othermodule
#Excludes: badmodule
```

Die Beschreibung wird in Dokument ▷ Einstellungen ▷ Module verwendet, um dem Nutzer zu erläutern, was das Modul macht. Die Zeile mit `Requires` wird verwendet, um andere Module anzugeben, die dieses Modul verwenden muss; die Zeile mit `Excludes` wird verwendet, um Module anzugeben, die mit diesem Modul nicht verwendet werden dürfen. Beide Zeilen sind optional und mehrere Module müssen , wie gezeigt, mit einem „|“ getrennt werden. Beachten Sie dass die benötigten Module disjunktiv behandelt werden: *Mindestens eins* der benötigten Module muss verwendet werden. Dementsprechend darf *keines* der ausgeschlossenen Modul verwendet werden. Beachten Sie auch, dass Module durch ihren Dateinamen ohne die Dateiendung `.module` angegeben werden. Daher ist `EinModul` ist in Wirklichkeit `EinModul.module`.

5.3.3. Die CiteEngine-Dateideklaration

Eine Cite-Engine-Datei muss mit einer Zeile wie dieser beginnen:

```
#\DeclareLyXCiteEngineModule[biblatex.sty]{Biblatex}
```

Das benötigte Argument in geschweiften Klammern ist der Name des Zitierstils, wie er in in Dokument ▷ Einstellungen ▷ Literaturverzeichnis erscheinen soll. Das Argument in eckigen Klammern ist optional: Es deklariert alle LaTeX-Pakete, die die Cite Engine benötigt.

Der CiteEngine-Deklaration sollten Zeilen zur Beschreibung wie die folgenden folgen:[12]

```
# DescriptionBegin
#   Biblatex supports many author-year and numerical styles.
```

[11] Vorzugsweise in Englisch, wenn das Modul als Teil von LyX veröffentlicht werden soll. Diese Beschreibung wird dann in LyX' Liste der zu übersetzenden Zeichenketten erscheinen und übersetzt werden.

[12] Vorzugsweise in Englisch, wenn die Cite Engine als Teil von LyX veröffentlicht werden soll. Diese Beschreibung wird dann in LyX' Liste der zu übersetzenden Zeichenketten erscheinen und übersetzt werden.

```
#   It is mainly aimed at the Humanities. It is highly
#   customizable, fully localized and provides many features
#   that are not possible with BibTeX. The use of 'biber' as
#   bibliography processor is advised.
# DescriptionEnd
```

Die Beschreibung wird in Dokument ▷ Einstellungen ▷ Literaturverzeichnis verwendet, um dem Nutzer zu erläutern, was der Stil bietet.

5.3.4. Dateiformat

Die erste Zeile, die kein Kommentar ist, muss die Dateiformatnummer enthalten:

`Format` [`int`] Die Nummer des Formats der Layout-Datei.

Diese Marke wurde mit LyX 1.4.0 eingeführt. Layout-Dateien älteren LyX-Versionen haben kein explizites Format und werden als `Format 1` behandelt. Das Format dieser LyX-Version ist 60. Aber jede LyX-Version kann ältere Versionen von Layout-Dateien lesen, so wie es ältere LyX-Dateien lesen kann. Es gibt jedoch keine Unterstützung in ältere Formate zu konvertieren.

5.3.5. Allgemeine Parameter für Textklassen

Nachfolgend allgemeine Parameter, die die Form der gesamten Dokumentklasse beschreiben. (Dies bedeutet *nicht* dass sie nur in `.layout`-Dateien und nicht in Modulen erscheinen müssen. Ein Modul kann jede Layout-Marke enthalten.)

`AddToCiteEngine <engine>` Erweitert die Möglichkeiten der Darstellung von Literaturverweisen. Siehe Abschnitt Kap. 5.3.14 für Details. Muss mit `End` beendet werden.

`AddToHTMLPreamble` fügt Informationen hinzu, die im `<head>`-Block ausgegeben werden, wenn das Dokument als XHTML ausgegeben wird. Typischerweise wird dies verwendet werden, um CSS-Stilinformationen auszugeben, aber es kann auch für alles Andere verwendet werden, dass in `<head>` zulässig ist. Muss mit `EndPreamble` beendet werden.

`AddToPreamble` fügt Informationen zum LaTeX-Vorspann hinzu. Muss mit `EndPreamble` beendet werden.

`BibInToc` [*`0`*, `1`] Falls die Dokumentklasse das Literaturverzeichnis per Voreinstellung im Inhaltsverzeichnis aufführt, fügen Sie diese Option mit dem Wert `1` (oder `true`) hinzu. Dies verhindert, dass das Literaturverzeichnis zweimal erscheint.

`CiteEngine <engine>` Definiert die Möglichkeiten der Darstellung von Literaturverweisen. Siehe Abschnitt Kap. 5.3.14 für Details. Muss mit `End` beendet werden. Vor allem von Cite-Engine-Dateien verwendet (siehe Abschnitt Kap. 5.2.6).

Beachten Sie, dass die Definitionen von Cite-Engine-Dateien komplett überschrieben werden, wenn Sie dies in einer Klasse oder einem Modul verwenden. Siehe auch `AddToCiteEngine`.

`CiteFormat` Definiert Formate die in der Anzeige von Bibliographie-Informationen verwendet werden. Siehe Kap. 5.3.15 für Details. Muss mit `End` beendet werden. Vor allem von Cite-Engine-Dateien verwendet (siehe Abschnitt Kap. 5.2.6). Ein Format, das in einer Klasse oder einem Modul definiert wurde, überschreibt das der Cite-Engine-Dateien.

`CiteFramework` [*`bibtex`*,`biblatex`] Bestimmt, ob Biblatex oder BibTEX verwendet wird, um ein Literaturverzeichnis zu erzeugen. Vor allem von Cite-Engine-Dateien verwendet (siehe Abschnitt Kap. 5.2.6).

`ClassOptions` Dieser Abschnitt beschreibt verschiedene globale Optionen, die von der Dokumentenklasse unterstützt werden. Eine detaillierte Beschreibung finden Sie in Kap. 5.3.6. Muss mit `End` beendet werden.

`Columns` [*1*, 2] Gibt an, ob die Textklasse standardmäßig ein- oder zweispaltig gesetzt wird. Kann im Menü geändert werden.

`Counter` [`string`] definiert die Eigenschaften für einen Zähler. Wenn der Zähler noch nicht existiert, wird er erstellt; wenn er bereits existiert, wird er modifiziert. Muss mit „`End`“ beendet werden.
Siehe Kap. 5.3.12 für Details zu Zählern.

`DefaultFont` Definiert den Standardzeichensatz für die Anzeige des Dokuments. Eine genauere Beschreibung finden Sie in Kap. 5.3.13. Muss mit `EndFont` beendet werden.

`DefaultModule` [`<Modul>`] spezifiziert ein Modul, das standardmäßig zu dieser Dokumentklasse hinzugefügt wird. <Modul> ist der Dateiname ohne die Dateiendung `.module`. Der Nutzer kann das Modul zwar immer noch entfernen, aber es bleibt von Beginn an aktiv. (Dies gilt nur für neue Dateien oder wenn diese Klasse für ein existierendes Dokument gewählt wird.)

`DefaultStyle` [`<Stil>`] Dies ist das Layout bzw. der Stil, der für neu angelegte Absätze verwendet wird. Normalerweise ist das Standard. Fehlt dieser Eintrag, wird das erste definierte Layout verwendet; dennoch ist es ratsam `DefaultStyle` anzugeben.

`ExcludesModule` [`<Modul>`] zeigt an, dass das genannte Modul (das durch den Dateinamen ohne die Endung `.module` angegeben wird) in dieser Dokumentklasse nicht benutzt werden kann. Dies könnte in einem Journal-spezifischen Layout benutzt werden, um zum Beispiel die Verwendung des Moduls `theorems-sec` zu verhindern, das Theoreme abschnittsweise nummeriert. Diese Marke darf *nicht*

in einem Modul benutzt werden. Module haben ihre eigene Methode andere Module auszuschließen (siehe Kap. 5.2.1).

Float definiert ein neues Gleitobjekt. Siehe Kap. 5.3.9 für Details. Muss mit **End** beendet werden.

HTMLPreamble Informationen, die im **<head>**-Block ausgegeben werden, wenn das Dokument als XHTML ausgegeben wird. Beachten Sie, dass dies jede vorhergehende **HTMLPreamble** oder **AddToHTMLPreamble**-Deklaration überschreibt. (Verwenden Sie **AddToHTMLPreamble** wenn Sie Material zum Vorspann hinzufügen wollen.) Muss mit **EndPreamble** beendet werden.

HTMLTOCSection [**<Stil>**] Das Layout bzw. der Stil, der für das Inhaltsverzeichnis, das Literaturverzeichnis etc. verwendet werden soll, wenn das Dokument als HTML exportiert wird. Für Artikel sollte dies normalerweise **Section** sein und für Bücher **Chapter**. Wenn es nicht angegeben wird, wird LyX versuchen herauszufinden, welches Layout zu benutzen ist.

IfCounter [**<Zähler>**] Ändert die Eigenschaften des angegebenen Zählers. Wenn dieser nicht existiert, wird die Anweisung ignoriert. Muss mit **End** beendet werden.
Siehe Kap. 5.3.12 für Details zu Zählern.

Input [**<Dateiname>**] Hiermit können Sie andere Dateien einbinden, die Definitionen für Textklassen enthalten. Damit können Sie unnötige Mehrfachdefinitionen vermeiden. Beispiele sind die Standard-Layout-Dateien, z. B. **stdclass.inc**, die ein Großteil der Standardlayouts enthalten.

InputGlobal [**<Dateiname>**] ist eine Variante des Befehls **Input**, die nicht nach Dateien im Benutzerverzeichnis sucht. Das erlaubt es Ihnen, im Benutzerverzeichnis eine Datei **name.layout** oder **name.inc** anzulegen, die mit **InputGlobal name** bzw. **InputGlobal name.inc** eine globale Datei desselben Namens einbindet (mit **Input** würde die Datei sich selbst einbinden). Damit können Sie globale Dateien modifizieren, ohne sie komplett kopieren zu müssen.

InsetLayout [**<Typ>**] Dieser Abschnitt definiert das Layout einer Einfügung (neu). Es kann auf eine vorhandene Einfügung angewendet werden oder eine neue, benutzerdefinierte, zum Beispiel einen neuen Zeichenstil. Muss mit **End** beendet werden.
Kap. 5.3.10 enthält weitere Einzelheiten.

LeftMargin [**string**] ist ein String dessen Länge die Breite des linken Randes festlegt, zum Beispiel **MMMMM**. (Beachten Sie, dass hier keine Längenangabe wie **2ex** eingegeben wird.)

MaxCiteNames [integer] Eine Ganzzahl, die festlegt, wie viele Autoren maximal in einem Autor-Jahr-Zitat angezeigt werden, bevor der Verweis zu „Erstautor et al.“ wechselt. Vor allem von Cite-Engine-Dateien verwendet (siehe Abschnitt Kap. 5.2.6).

ModifyInsetLayout [<Typ>] Ändert die Eigenschaften des angegebenen Layouts einer Einfügung. Wenn dieses nicht existiert, wird die Anweisung ignoriert. Muss mit **End** beendet werden.

ModifyStyle [<Stil>] Ändert die Eigenschaften des angegebenen Absatzstils. Wenn dieser nicht existiert, wird die Anweisung ignoriert. Muss mit **End** beendet werden.

NoCounter [<Zähler>] Löscht einen existierenden Zähler; üblicherweise einen, der in einer eingefügten Datei definiert wurde.

NoFloat [<Gleitobjekt>] Löscht ein vorhandenes Gleitobjekt. Dies ist dann nützlich, wenn Sie ein Gleitobjekt, das in einer eingefügten Datei definiert wurde, nicht verwenden wollen.

NoStyle [<Stil>] Löscht ein existierendes Layout bzw. einen Stil.

OutputFormat [<Format>] Das Dateiformat (wie es in den LyX-Voreinstellungen definiert ist) das von dieser Dokumentklasse erzeugt wird. Es ist hauptsächlich nützlich wenn **OutputType** auf **literate** gesetzt ist und man einen neuen Typ eines „Literate“-Dokuments definieren will. Das Format wird auf **docbook** oder **latex** zurückgesetzt wenn der entsprechende **OutputType**-Parameter gefunden wird.

OutputType [*latex*, **docbook**, **literate**] Gibt an welche Dokumentart diese Klasse erzeugt.

PackageOptions [**string string**] Der zweite String gibt Optionen für das Paket im ersten String an. Zum Beispiel lädt **PackageOptions natbib square** **natbib** mit der Option **square**. (Für TeXperten: Dies bewirkt, dass LyX **\PassOptionsToPackage{natbib}{square}** vor dem Laden von **natbib** ausgibt.)

PageSize [*custom*, **letter**, **legal**, **executive**, **a0**, **a1**, **a2**, **a3**, **a4**, **a5**, **a6**, **b0**, **b1**, **b2**, **b3**, **b4**, **b5**, **b6**, **c0**, **c1**, **c2**, **c3**, **c4**, **c5**, **c6**, **b0j**, **b1j**, **b2j**, **b3j**, **b4j**, **b5j**, **b6j**] Die Standard-Seitengröße. Dies wird von einigen Konvertern verwendet.

PageStyle [*plain*, **empty**, **headings**] Der Standard-Seitenstil. Kann im Menü geändert werden.

Preamble Definiert den Vorspann für das LaTeX-Dokument. Beachten Sie, dass dies jede vorhergehende `Preamble` oder `AddToPreamble`-Deklaration überschreibt. (Verwenden Sie `AddToPreamble` wenn Sie Material zum Vorspann hinzufügen wollen.) Muss mit `EndPreamble` beendet werden.

ProvideInsetLayout [`<Typ>`] Erstellt das Layout einer Einfügung, falls es noch nicht existiert. Existiert es bereits, wird `ProvideInsetLayout` ignoriert. Muss mit `End` beendet werden.

Provides [`string`] [*0*, 1] zeigt an, ob die Klasse bereits die Funktion `string` liefert. Eine Funktion ist im Allgemeinen der Name eines Paketes (z. B. `amsmath` oder `makeidx`) oder ein Makro (z. B. `url` oder `boldsymbol`). Siehe Anhang A für eine Liste der Funktionen.

ProvidesModule [`string`] zeigt an, dass dieses Layout die Funktionalität des Moduls `string` anbietet, das als Dateiname ohne die Erweiterung `.module` angegeben wird. Dies wird typischerweise benutzt, wenn das Layout das Modul direkt benutzt statt die Marke `DefaultModule` zu benutzen. Es könnte auch in einem Modul benutzt werden, das eine andere Implementation derselben Funktion liefert.

ProvideStyle [`<Stil>`] Erstellt einen neuen Absatzstil, falls er noch nicht existiert. Existiert er bereits, wird `ProvideStyle` ignoriert. Muss mit `End` beendet werden.

Requires [`string`] zeigt an, ob die Klasse die Funktion `string` benötigt. Mehrfache Funktionen müssen durch Komma getrennt werden Beachten Sie, dass Sie nur unterstützte Funktionen anfordern können. (Siehe Anhang A für eine Liste der Funktionen.) Wenn Sie ein Paket mit bestimmten Optionen anfordern müssen, können Sie zusätzlich `PackageOptions`verwenden.

RightMargin [`string`] ist eine Zeichenkette, deren Länge die Breite des rechten Randes festlegt, zum Beispiel `MMMMM`.

SecNumDepth [`int=3`] legt die Nummerierungstiefe fest; korrespondiert mit dem LaTeX-Zähler `secnumdepth`.

Sides [*1*, 2] Gibt an, ob der Text standardmäßig für ein- oder für zweiseitigen Druck gesetzt wird. Kann im Dialog geändert werden.

Style [`<Name>`] definiert einen neuen Absatzstil. Wenn er bereits existiert, werden stattdessen einige seiner Parameter neu definiert. Muss mit `End` beendet werden. Siehe Kap. 5.3.7 für mehr über Absatzstile.

TableStyle [`<name>`] legt den Standardstil fest, der für eingefügte Tabellen verwendet wird. Die folgenden Stile stehen zur Verfügung:

- `Formal_with_Footline`: formaler Stil ("booktabs") nur mit horizontalen Linien und fetter Linie am Kopf und Fuß der Tabelle. Zusätzlich sind die Kopf- und Fußzeile mit einer dünnen Linie vom Rest der Tabelle abgegrenzt.
- `Formal_without_Footline`: ähnlich wie der vorherige Stil, allerdings ist die letzte Zeile *nicht* mit einer dünnen Linie vom Rest der Tabelle abgegrenzt (also nicht als Fußzeile markiert).
- `Simple_Grid`: Einfache Tabellenlinien.
- `Grid_with_Head`: Wie `Simple_Grid`, aber mit einer zusätzlichen horizontalen Linie unter der Kopfzeile. Das ist auch der Standard-Tabellenstil von LyX.
- `No_Borders`: Tabelle ohne Rahmenlinien.

`TitleLatexName` [`string="maketitle"`] ist der Name des Befehls oder der Umgebung, der für `TitleLatexType` benutzt werden soll.

`TitleLatexType` [*`CommandAfter`*, `Environment`] gibt an, wie der Dokumenttitel aussehen soll. `CommandAfter` bedeutet, dass das Makro namens `TitleLatexName` nach dem letzten Absatzstil mit `InTitle 1` gesetzt werden soll. `Environment` ist für den Fall, dass alle Absatzstile mit `InTitle 1` in die `TitleLatexName`-Umgebung gesetzt werden sollen.

`TocDepth` [`int=3`] legt fest, bis zu welcher Tiefe das Inhaltsverzeichnis gehen soll; korrespondiert mit dem LaTeX-Zähler `tocdepth`.

5.3.6. Der Abschnitt `ClassOptions`

Der Abschnitt `ClassOptions` kann folgende Einträge enthalten:

`FontSize` [`string="10|11|12"`] Eine Liste verfügbarer Größen für den Hauptzeichensatz; die Einträge werden mit „`|`“ getrennt. Neben den genannten sind auch andere Größen möglich.

`FontSizeFormat` [`string`] Das Format der Schriftgrößen-Option. Voreinstellung: `$$spt`. `$$s` ist ein Platzhalter für die Schriftgröße.

`Header` wird benutzt, um die DTD-Zeile mit XML-basierten Klassen zu setzen. Zum Beispiel PUBLIC „-//OASIS//DTD DocBook V4.2//EN“.

`Other` [`string=""`] Sonstige Optionen für die Dokumentenklasse, die durch Komma getrennt werden. Sie werden in dem `\documentclass`-Befehl als optionales Argument übergeben.

PageSize [string="letter|legal|executive|a0|a1|a2|a3|a4|a5|a6|b0|b1|b2|b3|b4|b5|b6|c0|c1|c2|c3|c4|c5|c6|b0j|b1j|b2j|b3j|b4j|b5j|b6j"] Eine Liste verfügbarer Seitengrößen; die Einträge werden mit „|“ getrennt. Nur die aufgeführten Größen werden zurzeit unterstützt. Weitere Größen können ggf. über die benutzerdefinierten Klassenoptionen eingegeben werden.

PageSizeFormat [string] Das Format der Seitengrößen-Option. Voreinstellung: $$spaper. $$s ist ein Platzhalter für die Papiergröße.

PageStyle [string="empty|plain|headings|fancy"] Eine Liste verfügbarer Seitenstile; die Einträge werden mit „|“ getrennt.

Der Abschnitt ClassOptions muss mit End beendet werden.

5.3.7. Einzelne Absatz-Layouts

Eine Layoutbeschreibung für einen Absatz sieht wie folgt aus:[13]

```
Style Name
  ...
End
```

Innerhalb des Blocks sind folgende Befehle erlaubt:

AddToToc [string=""] Dieser Abschnitt erscheint im Inhaltsverzeichnis des spezifizierten Typs. Eine leere Zeichenkette deaktiviert die Anzeige dort. Siehe auch OutlinerName und den Befehl IsTocCaption. Voreinstellung: deaktiviert.

Align [*block*, left, right, center] Gibt an, ob der Text im Blocksatz linksbündig, rechtsbündig oder zentriert gesetzt wird.

AlignPossible [*block*, left, right, center] Eine Liste von möglichen Textausrichtungen, die durch Kommata voneinander getrennt werden. (Einige LaTeX-Stile verbieten bestimmte Ausrichtungen, weil sie keinen Sinn machen. Beispielsweise sollte in einer nummerierten Aufzählung der Text nicht rechtsbündig oder zentriert gesetzt werden.)

Argument [int] Definiert Argument Nummer <int> eines Befehls/einer Umgebung, der/die im aktuellen Stil definiert ist. Die Definition muss mit EndArgument enden. Siehe Kap. 5.3.11 für Details.

AutoNests beinhaltet eine mit Kommata separierte Liste von Layouts, welche in und nach dem aktuellen Layout automatisch eingebettet werden sollen. Dies ist nur für Layouts sinnvoll, die auch einbetten können (etwa Umgebungen). Muss mit EndAutoNests beendet werden. Siehe auch IsAutoNestedBy.

[13] Sie können mit diesem Ausdruck entweder einen neuen Absatzstil definieren oder aber einen bereits definierten umdefinieren.

BabelPreamble Beachten Sie, dass dies alle vorhergehenden BabelPreamble-Deklaration für diesen Stil überschreibt. Muss mit „EndBabelPreamble“ beendet werden. Siehe Kap. 5.3.8 für Details zur Verwendung.

BottomSep [float=0][14] Der vertikale Abstand, der die letzte Serie von Absätzen vom folgenden Text trennt. Wenn der nächste Paragraph einen anderen Stil hat, werden die Abstände nicht einfach addiert, sondern das Maximum wird verwendet.

Category [string] ist die Kategorie für diesen Stil. Sie wird benutzt, um zugehörige Stile im Absatzstil-Auswahlfeld der Werkzeugleiste zu gruppieren. Jede beliebige Zeichenkette kann benutzt werden, aber es ist sinnvoll, vorhandene Kategorien zu verwenden.

CommandDepth ist die Tiefe des XML-Befehls und wird nur für XML-Formate benutzt.

CopyStyle [string] Kopiert alle Eigenschaften eines bereits definierten Layouts in das aktuelle.

DependsOn [<Name>] ist der Name eines Stils, dessen Vorspann *vor* diesem ausgegeben werden soll. Dadurch wird eine Reihenfolge von Vorspannteilen bewirkt, wenn Makro-Definitionen voneinander abhängen.[15]

EndLabelType [*No_Label*, Box, Filled_Box, Static] ist der Markentyp, der am Ende eines Absatzes steht (oder mehrerer Absätze, wenn LatexType auf Environment, Item_Environment oder List_Environment gesetzt ist).
No_Label bedeutet „nichts“, Box oder Filled_Box ist ein weißes oder schwarzes Quadrat, das für das Markieren eines Beweis-Endes geeignet ist. Static ist eine explizite Zeichenkette.

EndLabelString [string=""] ist eine Zeichenkette, die für einen Static EndLabelType benutzt wird.

Font Der Zeichensatz, der für den Textkörper *und* die Marke verwendet wird, siehe Kap. 5.3.13. Wird Font gesetzt, dann erhält LabelFont automatisch denselben Wert. Daher sollte Font zuerst definiert werden.

ForceLocal [int=0] Wird benutzt um neue Stile für stabile LyX-Versionen zu konvertieren. Die erste stabile Version, die das unterstützt, ist LyX 2.1.0. Das Argument ist eine Nummer, die entweder 0, -1 oder irgend eine Zahl größer Null sein kann. Wenn ForceLocal eines Stils größer als Null ist, wird er immer in den Dokumentkopf geschrieben. Wenn eine .lyx-Datei gelesen wird, werden die

[14] „float“ ist eine Gleitkommazahl, wie „1.5“.

[15] Beachten Sie, dass es außer dieser Funktionalität keine andere Möglichkeit gibt, Vorspanne zu ordnen. Die Reihenfolge, die Sie in einer LyX-Version sehen, kann sich in späteren Versionen ohne Warnung ändern.

Stil-Definitionen aus dem Dokumentkopf zur Dokumentklasse hinzugefügt. Dadurch können sogar ältere LyX-Versionen den Stil handhaben. Das Argument von `ForceLocal` ist eine Versionsnummer: Wenn der Stil gelesen wird, und die Versionsnummer ist kleiner als die Versionsnummer des bereits existierenden Stils der Dokumentklasse, wird der neue Stil ignoriert. Wenn die Versionsnummer größer ist, ersetzt der neue Stil den bestehenden. Der Wert `-1` steht für eine unendliche Versionsnummer, das heißt der Stil wird immer benutzt.

`FreeSpacing` [*0*, 1] Normalerweise erlaubt es LyX nicht, mehr als ein Leerzeichen zwischen Wörtern einzufügen. Diese Eigenschaft kann in bestimmten Fällen umständlich sein, zum Beispiel, wenn ein Programmcode eingegeben werden soll. In solchen Fällen kann `FreeSpacing` auf `1` gesetzt werden. LyX erzeugt in diesem Falls sich LyX nicht im LaTeX-Modus befindet, erzeugt es für jedes zusätzliche Leerzeichen ein geschütztes Leerzeichen.

`HTML*` Diese Marken kontrollieren die XHTML-Ausgabe. Siehe Kap. 5.4.

`InnerTag` [FIXME] (Wird nur für XML-Formate benutzt.)

`InPreamble` [*0*, 1] Wenn auf `1` gesetzt, wird der Stil in den LaTeX-Vorspann gesetzt und nicht in den eigentlichen Dokumenttext. Dies ist nützlich für Dokumentklassen, die Informationen wie den Titel und Autor im Vorspann erwarten. Beachten Sie, dass dies nur für Stile funktioniert, deren `LatexType Command` oder `Paragraph`ist.

`InTitle` [*0*, 1] Wenn auf `1` gesetzt, wird der Stil als Teil des Titel-Abschnitts behandelt (siehe auch die allgemeinen Textklassen-Parameter `TitleLatexType` und `TitleLatexName`).

`IsAutoNestedBy` beinhaltet eine mit Kommata separierte Liste von Layouts, nach welchen Absätze mit dem aktuellen Layout automatisch eingebettet werden sollen. Sinnvolle Layouts für diese Liste sind nur solche, die auch einbetten können (etwa Umgebungen). Muss mit `EndIsAutoNestedBy` beendet werden. Siehe auch `AutoNests`.

`IsTocCaption` [*0*, 1] Wenn dies auf `1` gesetzt ist und `AddToToc` aktiviert ist, fügt der Absatz eine Zusammenfassung seines Inhalts zum Eintrag im Inhaltsverzeichnis. Ansonsten wird nur die Marke, falls existent, dort aufscheinen.

`ItemCommand` [`string="item"`] Der LaTeX-Befehl, der ein Item in einer Liste definiert. Dieser Befehl muss ohne den Backslash am Anfang angegeben werden (die Voreinstellung ist '`item`', was in der LaTeX-Ausgabe als `\item` erscheint).

`ItemSep` [`float=0`] Ein zusätzlicher Abstand zwischen Absätzen desselben Layouts. Wenn in einer Umgebung andere Layouts integriert werden, so werden diese mit dem `ParSep` der Umgebung getrennt. Die kompletten Unterpunkte der

Umgebung werden jedoch *zusätzlich* mit `ItemSep` getrennt. Man beachte, dass `ItemSep`ein *Multiplikator* ist.

`ItemTag` [FIXME] (Wird nur für XML-Formate benutzt.)

`KeepEmpty` [*0*, 1] Normalerweise ist es in LyX nicht möglich, einen Absatz leer zu lassen, da das zu einer leeren LaTeX-Ausgabe führen würde. In manchen Fällen ist das aber durchaus gewünscht: So können beispielsweise in einer Briefvorlage die benötigten Felder leer voreingestellt werden, damit keiner vergisst, sie anzugeben; in speziellen Klassen wird ein Absatz als Unterbrechung verwendet, der keinen Text enthält.

`LabelBottomsep` [`float=0`] Der vertikale Abstand zwischen der Marke und dem folgenden Text. Wird nur für Marken benutzt, die über dem folgenden Text stehen (`Top_Environment` und `Centered_Top_Environment`).

`LabelCounter` [`string=""`] ist der Name des Zählers zur automatischen Nummerierung. Um den Zähler einer Marke zuzuordnen, muss er im `LabelString` referenziert werden. Dies funktioniert zumindest mit `LabelType`en, `Static`, `Above` und `Centered`.
Er *kann* angegeben werden, wenn `LabelType Enumerate` ist. In diesem Fall ist es etwas kompliziert: Angenommen Sie haben "`LabelCounter MeinZaehler`" angegeben, dann lauten die eigentlichen Zähler `MeinZaehleri`, `MeinZaehlerii`, `MeinZaehleriii` und `MeinZaehleriv`; so wie in LaTeX. Diese Zähler müssen alle separat deklariert werden.
Siehe Kap. 5.3.12 für Einzelheiten zu Zählern.

`LabelFont` Der Zeichensatz, der für die Marke verwendet wird. Siehe Kap. 5.3.13.

`LabelIndent` [`string=""`] Text der angibt, wie weit die Marke eingerückt werden soll.

`Labelsep` [`string=""`] Text der den horizontalen Abstand zwischen der Marke und dem folgenden Text angibt. Wird nur für Marken benutzt, die nicht über dem folgenden Text stehen.

`LabelString` [`string=""`] Die Zeichenkette, die für den `LabelType Static` verwendet wird. Wenn `LabelCounter` gesetzt wurde, kann die Zeichenkette spezielle Formatierungsbefehle enthalten, wie sie in Kap. 5.3.12 beschrieben sind.

`LabelStringAppendix` [`string=""`] wird im Anhang anstatt `LabelString` benutzt. Beachten Sie, dass jede Definition von `LabelString` auch `LabelStringAppendix` zurücksetzt.

`LabelTag` [FIXME] (Wird nur für XML-Formate benutzt.)

LabelType [*No_Label*, Manual, Static, Above, Centered, Sensitive, Enumerate, Itemize, Bibliography]

Manual bedeutet: die Marke ist das erste Wort (bis zum ersten echten Leerzeichen). Verwenden Sie geschützte Leerzeichen wenn Sie mehr als ein Wort als Marke haben wollen.

Static bedeutet: die Marke ist, was als **LabelString** definiert wurde. Die Marke wird interlinear zu Beginn des Absatzes angezeigt. Wenn der **LatexType Environment** ist, wird sie nur im ersten Absatz von aufeinanderfolgenden Absätzen mit demselben **Style**. angezeigt.

Above und Centered sind Spezialfälle von **Static**. Die Marke erscheint über dem Absatz, entweder am Anfang der Zeile oder zentriert.

Sensitive ist ein Spezialfall für Beschriftungsmarken für Abbildungen und Tabellen-Gleitobjekte. **Sensitive** bedeutet, dass der gedruckte Text von der Art des Gleitobjekts abhängt: Er ist fest einprogrammiert als ‚GleitobjektTyp N', wobei N der Wert des Zählers des Gleitobjekttyps ist. Für den Fall, dass die Beschriftungsmarke außerhalb eines Gleitobjekts eingefügt wird, erscheint der **LabelString** als „Sinnlos!".

Enumerate erzeugt die üblichen Marken für Nummerierungen. Die Art der Nummerierung muss im Zähler festgelegt werden, siehe Kap. 5.3.12.

Itemize erzeugt je nach Schachtelungstiefe verschiedene Auflistungszeichen. Die Auflistungszeichen können über das Menü Dokument ▷ Einstellungen ▷ Auflistungszeichen eingestellt werden.

Bibliography sollte nur zusammen mit **LatexType BibEnvironment** verwendet werden.

LangPreamble Beachten Sie, dass dies alle vorhergehenden **LangPreamble**-Deklaration für diesen Stil überschreibt. Muss mit **EndLangPreamble** beendet werden. Siehe Kap. 5.3.8 für Details zur Verwendung.

LatexName [<Name>] Der LaTeX-Name für dieses Layout. Das bedeutet entweder der Name eines LaTeX-Befehls oder der einer LaTeX-Umgebung.

LatexParam [<Parameter>] Ein optionaler Parameter für den entsprechenden **LatexName**. Dieser Parameter kann innerhalb von LyX nicht mehr geändert werden (man verwendet **Argument** für anpassbare Parameters). Dieser wird nach allen anderen LaTeX-**Argument**en ausgegeben.

LatexType [*Paragraph*, Command, Environment, Item_Environment, List_Environment, Bib_Environment] Legt fest, wie das Layout in LaTeX übersetzt wird.[16]

[16] **LatexType** mag irreführend sein, denn dessen Regeln gelten auch für DocBook-Klassen. Siehe die DocBook Klassendateien (Dateinames **db_*.inc**) für spezielle Beispiele.

Paragraph bewirkt nichts besonderes – der Text wird als *normaler Absatz* übernommen.

Command behandelt den Text als Argument eines LaTeX-Befehls (`\`*LatexName*`{...}`).

Environment behandelt den Text als Kern einer LaTeX-Umgebung (`\begin{`*LatexName*`}...\end{`*LatexName*`}`).

Item_Environment bewirkt dasselbe wie `Environment`, nur dass vor jedem Absatz ein `\item` eingefügt wird.

List_Environment funktioniert wie `Item_Environment`, nur dass `LabelWidthString` als Argument an die Umgebung übergeben wird. `LabelWidthString` kann im Menü Bearbeiten ▷ Absatz-Einstellungen definiert werden.

Bib_Environment ist wie `Environment` aber fügt zusätzlich das notwendige Argument (die längste Marke) zum Startbefehl der Bibliografie-Umgebung ein:
`\begin{thebibliography}{99}` Es ist daher nur für die Bibliografie-Umgebung nützlich. Die voreingestellte längste Marke „99“ kann vom Nutzer in den Absatzeinstellungen eines Bibliografie-Eintrags geändert werden.

Fasst man die letzten Sachen zusammen, wird die LaTeX-Ausgabe entweder so:

```
\LatexName[LatexParam]{...}
```

oder so:

```
\begin{LatexName}[LatexParam] ... \end{LatexName}.
```

aussehen, abhängig vom LaTeX-Typ.

LeftDelim [`string`] Eine Zeichenkette, die zu Beginn des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

LeftMargin [`string=""`] Wenn ein Layout in ein anderes Layout für Umgebungen eingefügt wird, werden die Breiten der verschiedenen `LeftMargin` nicht einfach addiert, sondern vorher in Abhängigkeit zur Schachtelungstiefe mit dem Faktor $\frac{4}{\text{Tiefe}+4}$ multipliziert. Dieser Parameter wird auch dann benutzt, wenn `Margin` als `Manual` oder `Dynamic` definiert wurde. In diesem Fall wird der Wert zu den gegebenen manuellen oder dynamischen Rändern hinzugefügt.
Zum Beispiel bedeutet „`MM`“, dass der Absatz mit der Breite eingerückt wird, die die Buchstaben „`MM`“ in der normalen Schriftart haben. man kann negative Breite erzeugen, indem man den String mit „`-`“ beginnt. Diese Art der Angabe wurde gewählt, damit der Text unabhängig von der verwendeten Bildschirmschriftart wie vorgesehen aussieht.

Margin [*Static*, Manual, Dynamic, First_Dynamic, Right_Address_Box]
legt fest, wie der linke Rand des Textes bestimmt wird.

Static wählt feste Randbreiten.

Manual bedeutet, dass der Rand von der Einstellung der Ausrichtung im Menü Bearbeiten ▷ Absatz-Einstellungen abhängt. Dies wird für hübsche Listen ohne Tabulatoren benutzt.

Dynamic bedeutet, der linke Rand hängt von der Größe der verwendeten Markierung ab. Dies wird zum Beispiel bei automatisch nummerierten Überschriften verwendet. Es leuchtet ein, dass die Überschrift „5.4.3.2.1 Sehr lange ... Überschrift“ einen größeren linken Rand benötigt, als „3.2 Sehr lange ... Überschrift“.

First_Dynamic arbeitet ähnlich wie Dynamic, aber nur die erste Zeile wird dynamisch gesetzt, die anderen statisch. Dies wird für die LaTeX-Umgebung description benutzt.

Right_Address_Box bedeutet, dass der Rand so gewählt wird, dass die längste Zeile des Absatzes gerade den rechten Rand berührt. Dies wird zum Setzen einer Adresse am rechten Rand der Seite eingesetzt.

NeedProtect [*0*, 1] Gibt an, ob „zerbrechliche“ LaTeX-Befehle innerhalb dieses Layouts durch \protect geschützt werden müssen. (Achtung: Diese Einstellung sagt nichts darüber aus, ob der Befehl an sich geschützt werden soll.)

NeedCProtect [-1, *0*, 1] Wert 1 bewirkt, dass Makros, die dieses Layout enthalten, mittels \cprotect (Paket cprotect) geschützt werden, falls nötig. Damit wird die Verwendung (mancher) Verbatim-Dinge in Makros ermöglicht. In der Voreinstellung (Wert 0) wird \cprotect verwendet, sobald ein eingebettetes Element dies verlangt. Der Wert -1 unterbindet die Verwendung von \cprotect selbst dann, wenn eingebettete Elemente dies verlangen.

NeedMBoxProtect [*0*, 1] legt fest, ob bestimmte Befehle in diesem Stil (bspw. \cite und \ref) in einer \mbox geschützt werden sollen. Das ist vor allem für Stile nötig, die auf Befehle der Pakete ulem oder soul aufbauen, welche ihren Inhalt in kompexer Art und Weise auslesen.

Newline [0, *1*] Gibt an, ob Zeilenumbrüche in LaTeX als „\\“ dargestellt werden, oder nicht. Man kann dies ausschalten (Wert: 0), um TeX-Code in LyX komfortabler editieren zu können.

NextNoIndent [*0*, 1] Gibt an, ob der nachfolgende Absatz einen Erstzeileneinzug haben darf oder nicht. 1 heißt, der Absatz erhält auf keinen Fall einen Einzug (z. B. nach einer Überschrift), wenn DefaultStyle- (normalerweise Standard-) Paragraphen einen Einzug haben. (Daher beeinflusst die Einstellung nur Standard-Paragraphen.)

ObsoletedBy [<Name>] Der Name eines Layouts, das durch dieses ersetzt wurde. So können Sie ein Layout umbenennen und die Rückwärtskompatibilität erhalten.

ParagraphGroup [0, 1] Legt fest ob aufeinanderfolgende Absätze desselben Typs als zusammengehörend behandelt werden. Das hat den Effekt, dass GuiLabel nur einmalig vor einer solchen Gruppe ausgegeben wird. Dies ist standardmäßig der Fall für LaTeXType Environment und Bib_Environment und nicht der Fall für alle anderen Typen.

ParbreakIsNewline [*0*, 1] Gibt an, dass ein Paragraph nicht durch eine leere Zeile in der LaTeX-Ausgabe abgesetzt wird, sondern nur durch einen Zeilenumbruch. Zusammen mit PassThru 1 erlaubt dies die Emulation eines reinen Texteditors (so wie die TeX-Code Einfügung).

ParIndent [string=""] Der Einzug der ersten Zeile eines Absatzes. Parindent bleibt für ein bestimmtes Layout fest. Eine Ausnahme ist das Standard-Layout, denn dort kann der Einzug vom vorherigen Layout mit NextNoIndent verboten werden. Außerdem benutzt das Standard-Layout innerhalb von Umgebungen den Parindent der Umgebung und nicht den eigenen. Zum Beispiel haben Standard-Absätze innerhalb einer Aufzählung keinen Einzug.

ParSep [float=0] Der vertikale Anstand zwischen den Absätzen dieses Layouts.

Parskip [float=0] Benutzer können in LyX wählen, ob Absätze durch Einrückung oder Vertikaler Abstand getrennt werden. Wenn Einrückung gewählt ist, wird Parskip ignoriert. Ist Vertikaler Abstand gewählt wird ParIndent ignoriert und alle Absätze durch den vertikalen Abstand von Parskip getrennt. Die Länge dieses Abstands berechnet sich mit Parskip * DefaultHeight wobei DefaultHeight die Höhe einer Zeile in der normalen Schrift ist. Dadurch bleibt das Aussehen mit verschiedenen Schriften gleich.

PassThru [*0*, 1] Legt fest, ob der Absatzinhalt unverändert ausgegeben werden soll, also ohne diverse von LaTeX benötigte Ersetzungen durchzuführen.

PassThruChars [string] Definiert Zeichen, die unverändert ausgegeben werden sollen. Das bedeutet, dass sie nicht in einen LaTeX-Befehl übersetzt werden, falls das normalerweise der Fall wäre.

Preamble Befehle und Definitionen, die in die Präambel (vor \begin{document}) eingefügt werden, wenn dieses Layout benutzt wird. Kann verwendet werden um Pakete zu laden, Makros zu definieren usw. Muss mit EndPreamble beendet werden.

RefPrefix [string] Der Präfix, der verwendet werden soll, wenn auf Marken dieses Absatzes verwiesen wird. Dies erlaubt die Verwendung von formatierten Querverweisen.

Requires [string] legt fest, dass das Layout die Funktion `string` benötigt (siehe Anhang A für eine List der Funktionen). Wenn Sie ein Paket mit bestimmten Optionen anfordern müssen, können Sie zusätzlich `PackageOptions`als allgemeiner Textklassen-Parameter verwenden (siehe Kap. 5.3.5).

ResetArgs [*0*,1] Setzt die LaTeX-Argumente dieses Stils zurück (der via `Argument` definiert wurde). Dies ist nützlich, wenn man einen Stil mit `CopyStyle` kopiert hat, aber nicht dessen (benötigte und optionale) Argumente übernehmen will.

ResumeCounter [*0*,1] Behält den letzten Zählerstand bei Zählern bei, die normalerweise am Beginn einer neuen Sequenz von Absatzstilen zurückgesetzt würde. Momentan ist das nur sinnvoll, wenn `LabelType` auf `Enumerate` gesetzt ist.

RightDelim [string] Eine Zeichenkette, die am Ende des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

RightMargin [string=""] Ähnlich wie `LeftMargin`.

Spacing [*single*, onehalf, double, other <Wert>] Dies definiert die Voreinstellung für den Zeilenabstand des Layouts. Die Argumente *single*, `onehalf` und `double` entsprechen den Multiplikatoren 1, 1.25 und 1.667. Wenn Sie als Argument `other` angeben, müssen Sie als *Wert* einen konkreten Multiplikator angeben. Im Gegensatz zu anderen Parametern erzeugt LyX, wenn `Spacing` gesetzt wird, spezifischen LaTeX-Code, der das LaTeX-Paket `setspace` verwendet.

Spellcheck [0, *1*] Erlaubt es, den Inhalt des Absatzes auf Rechtschreibung zu überprüfen.

StepMasterCounter [*0*,1] Erhöht den Hauptzähler eines gegebenen Zählers am Anfang einer neuen Sequenz von Layouts. Momentan ist das nur sinnvoll, wenn `LabelType` auf `Enumerate` gesetzt ist.

TextFont Der Zeichensatz, der für den Textkörper verwendet wird.
Siehe Kap. 5.3.13.

TocLevel [int=3] ist die Stufe des Stils im Inhaltsverzeichnis und wird zur automatischen Nummerierung von Abschnittsüberschriften benutzt.

ToggleIndent [*default*, always, never] Dieser Befehl legt fest, ob die Einrückung der ersten Absatzzeile über den Absatz-Dialog ein/ausgeschaltet werden kann. Wenn *default* gesetzt ist, kann umgeschaltet werden, wenn in den Dokumenteinstellungen für die Absatztrennung „Einrückung“ gewählt ist, Mit `always` kann immer umgeschaltet werden, unabhängig von den Dokumenteinstellungen. Mit `never`kann nie umgeschaltet werden.

TopSep [float=0] Der vertikale Abstand, der die erste Serie von Absätzen vom vorangehenden Text trennt.

5.3.8. Internationalisierung von Absatz-Stilen

Wenn ein Absatzstil (`Style`) Text definiert, der im ausgegebenen Dokument erscheinen soll, kann `LangPreamble` und `BabelPreamble` verwendet werden, um nichtenglische und sogar mehrsprachige Dokumente korrekt zu bedienen. Der folgende Auszug (aus der Datei `theorems-ams.inc`) zeigt, wie das geht:

```
Preamble
      \theoremstyle{remark}
      \newtheorem{claim}[thm]{\protect\claimname}
    EndPreamble
    LangPreamble
      \providecommand{\claimname}{_(Claim)}
    EndLangPreamble
    BabelPreamble
        \addto\captions$$lang{\renewcommand{\claimname}{_(Claim)}}
    EndBabelPreamble
```

Im Grunde kann jeder korrekte LaTeX-Code in die `LangPreamble` und `BabelPreamble` eingefügt werden, normalerweise sieht er aber so aus wie hier gezeigt. Der Schlüssel zur korrekten Übersetzung des ausgegebenen Texts ist hier die Definition des LaTeX-Befehls `\claimname` und seine Verwendung in `\newtheorem`.

`LangPreamble` erlaubt Internationalisierung mit Blick auf die Hauptsprache des Dokuments. Der Inhalt wird im Vorspann des LaTeX-Dokuments ausgegeben, wie im Fall von `Preamble`. Spezifisch ist allerdings die Verwendung der „Funktion“ `_()`, die mit der Übersetzung seines Inhalts in die Dokumentsprache ersetzt wird, wenn LyX eine LaTeX-Ausgabe erzeugt.

`BabelPreamble` ist etwas komplexer, da sie auf mehrsprachige Dokumente abzielt und eine Schnittstelle zum Paket `babel` bietet. Sein Inhalt wird jeweils einmal für jede im Dokument verwendete Sprache in den Vorspann eingefügt. Das Argument von `_()` wird dabei jeweils durch die Übersetzung in die betreffende Sprache ersetzt; der Ausdruck `$$lang` wird vom Babel-Namen für die Sprache ersetzt (z. B. `ngerman` für Deutsch).

Ein deutschsprachiges Dokument, das auch einen französischen Abschnitt enthält, würde also etwa folgendes im Vorspann stehen haben:

```
\addto\captionsfrench{\renewcommand{\claimname}{Affirmation}}
\addto\captionsngerman{\renewcommand{\claimname}{Behauptung}}
\providecommand{\claimname}{Behauptung}
```

LaTeX und `babel` werden gemeinsam den jeweils korrekten Text in der Ausgabe erzeugen.

Wichtig zu wissen ist, dass die Übersetzungen selbst von LyX beigesteuert werden. Dabei wird die Datei `layouttranslations` konsultiert. Das heißt, dass

`LangPreamble` und `BabelPreamble` nur richtig sinnvoll im Fall von Layout-Dateien sind, die mit LyX geliefert werden, da benutzereigene Layout-Dateien von LyX' Übersetzungsmechanismus nicht berücksichtigt werden, wenn die Datei `layouttranslations` nicht entsprechend bearbeitet wurde. Umgekehrt sollten alle Layout-Dateien, die in LyX aufgenommen werden sollen, diese Marken auch konsequent einsetzen. Beachten Sie, dass die Übersetzungen, die LyX bietet, niemals mit kleineren Versionssprüngen geändert werden (bspw. zwischen Version 2.1.x und 2.1.y). Es ist aber damit zu rechnen, dass dies im Rahmen eines Hauptversionssprungs (z. B. von 2.2.x auf 2.3.0) geschieht.

5.3.9. Gleitobjekte

Es ist nötig, Gleitobjekte (Abbildung, Tabelle,...) in der Textklasse selber zu definieren. Standardgleitobjekte sind in der Datei `stdfloats.inc` enthalten, so dass Sie sie nur noch

```
Input stdfloats.inc
```

zu Ihrer Layoutdatei hinzufügen müssen. Wenn Sie eine Textklasse implementieren wollen, die andere Gleitobjekttypen enthält (wie zum Beispiel die AGU-Klasse), werden Ihnen die folgenden Informationen helfen:

`AllowedPlacement` [`string`=!htbpH] Erlaubte Platzierungsoptionen für den Gleitobjekttyp. Der Wert ist eine Zeichenkette aus Platzierungszeichen. Mögliche Zeichen sind: `h` (*here* = „hier wenn möglich“), `t` (*top* = „oben auf Seite“), `b` (*bottom* = „unten auf Seite“), `p` (*page* = „auf Seite nur mit Gleitobjekten“), `H` („hier, unbedingt“) und `!` („ignoriere LaTeX-Regeln”). Die Reihenfolge der Zeichen in der Zeichenkette ist egal. Wenn keine Platzierungsoptionen erlaubt sind, verwendet man stattdessen die Zeichenkette `none`.

`AllowsSideways` [`0`, *`1`*] Definiert ob das Gleitobjekt mit Hilfe des LaTeX-Pakets `rotfloat` (sidewaysfloat) rotiert werden kann. Falls das nicht der Fall ist, setzt man es auf `0`.

`AllowsWide` [`0`, *`1`*] Definiert ob das Gleitobjekt eine Sternversion hat, die in einem zweispaltigen Dokument die komplette Seitenbreite einnimmt. Falls das nicht der Fall ist, setzt man es auf `0`.

`Extension` [`string`=""] Die Dateinamenserweiterung einer zusätzlichen Datei für das Abbildungsverzeichnis (oder andere). LaTeX schreibt die Beschriftungen in diese Datei.

`GuiName` [`string`=""] Die Zeichenkette, die in den Menüs und für die Beschriftung benutzt wird. Dies wird in die aktuelle Sprache übersetzt, wenn babel verwendet wird.

HTML* Diese Marken kontrollieren die XHTML-Ausgabe. Siehe Kap. 5.4.

IsPredefined [*0*, 1] Gibt an, ob das Gleitobjekt bereits in der Dokumentklasse definiert ist oder ob das LaTeX-Paket `float` geladen werden muss, um es zu definieren. Die Voreinstellung ist `0`, was bedeutet, dass `float` verwendet wird. Es sollte auf `1` gesetzt werden, wenn das Gleitobjekt bereits in der Dokumentklasse definiert ist.

ListCommand [string=""] Der Befehl der verwendet wird, um eine Liste der Gleitobjekte dieses Typs zu generieren; das `\` muss weggelassen werden. Der Befehl *muss* angegeben werden, wenn `UsesFloatPkg` auf `0` gesetzt ist, da es sonst keine Möglichkeit gibt, diesen Befehl zu erstellen. Er wird ignoriert, falls `UsesFloatPkg` auf `1` gesetzt ist, da es dann eine Möglichkeit gibt.

ListName [string=""] Die Überschrift für das Gleitobjekt-Verzeichnis (z. B. „Abbildungsverzeichnis“). Sie wird für die Bildschirmmarke in LyX verwendet, von LaTeX für den Titel verwendet und als Titel in der XHTML-Ausgabe. Sie wird in die Dokumentsprache übersetzt.

NumberWithin [string=""] Dieses optionale Argument bestimmt, ob Gleitobjekte dieser Klasse mit der Abschnittsnummer dieses Dokuments nummeriert werden. Wenn zum Beispiel `NumberWithin` auf `chapter` gesetzt ist, werden die Gleitobjekte mit den Kapitelnummern nummeriert.

Placement [string=""] Die Standardplatzierung für die Gleitobjektklasse. `string` sollte die Standard-LaTeX-Werte `t`, `b`, `p` und `h` für *oben*, *unten*, *Seite* und *hier* enthalten.[17] Zusätzlich gibt es den neuen Typ `H`, der nicht wirklich für ein Gleitobjekt steht, denn er bedeutet: drucke es *hier* und nirgendwo sonst. Beachten Sie, dass `H` besonders ist und wegen der Implementierungsdetails nicht bei nicht-eingebauten Gleitobjekttypen benutzt werden kann. Wenn Sie die Platzierung nicht verstehen, benutzen Sie einfach `tbp`.

PrettyFormat [string=""] Ein Format, das für *formatierte Querverweise* auf einen Zähler verwendet wird. Möchte man zum Beispiel Verweise auf Tabellen als „Tabelle 2“ ausgegeben haben, kann die Zeichenkette `##` oder eine Zählerspezifikation enthalten (für letzteres siehe die Dokumentation von `LabelString` in Kap. 5.3.12). Im ersteren Fall werden die Zeichen `##` später durch die referenzierte Nummer ersetzt. Für Abschnitte würde man also beispielsweise `Abschnitt ##` verwenden. Eine andere Möglichkeit wäre `\S\,\arabic{section}`; dies würde als § 2.7 ausgegeben).

RefPrefix [string] Das Präfix, das verwendet werden soll, wenn auf Marken dieser Gleitobjekte verwiesen wird. Dies erlaubt die Verwendung von Formatierten Querverweisen. Man kann das `RefPrefix` eines kopierten Stils entfernen, indem `string` auf `OFF` gesetzt wird.

[17] Wie in LaTeX ist die Reihenfolge der Buchstaben unerheblich.

Requires [string] Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

Style [string=""] ist der Gleitobjektstil, wenn er mit `\newfloat` definiert wird.

Type [string=""] ist der „Typ“ der neuen Gleitobjektklasse, wie z. B. *Programm* oder *Algorithmus*. Nach dem entsprechenden `\newfloat` stehen Befehle wie `\begin{program}` oder `\end{algorithm*}` zur Verfügung.

UsesFloatPkg [0, *1*] Gibt an, ob dieses Gleitobjekt mit Hilfe des LaTeX-Pakets `float` definiert wurde, entweder durch die Dokumentklassen, ein anderes Paket oder durch LyX.

Anmerkung: Wenn ein Gleitobjekt vom Typ *type* definiert wurde, gibt es automatisch einen dazugehörigen Zähler namens *type*.

5.3.10. Flexible Einfügungen und InsetLayout

Es gibt drei Arten von flexiblen Einfügungen:

- Zeichenstil (`CharStyle`): diese definieren semantische Textauszeichnungen, die mit LaTeX-Befehlen wie `\noun` oder `\code` korrespondieren.
- Benutzerdefiniert (`Custom`): diese können benutzt werden, um benutzerdefinierte einklappbare Einfügungen zu definieren, ähnlich wie TeX-Code, Fußnote usw. Ein Beispiel ist Endnote, die im `endnote`-Modul definiert ist.
- XML-Element (`Element`): diese werden mit DocBook-Klassen benutzt.

Flexible Einfügungen werden mit der `InsetLayout`-Marke definiert, die weiter unten erklärt wird.

Die `InsetLayout`-Marke besitzt noch eine andere Funktion: sie kann benutzt werden, um das allgemeine Aussehen vieler verschiedener Einfügungstypen anzupassen. Zurzeit kann `InsetLayout` benutzt werden, um die Layout-Parameter für Fußnoten, Randnoten, eingefügten Noten, TeX-Code (ERT), Zweige, Stichwortverzeichnisse, Boxen, Tabellen, Algorithmen, URLs und Legenden anzupassen, ebenso um flexible Einfügungen zu definieren.

Die `InsetLayout`-Definition muss mit folgender Zeile beginnen:

```
InsetLayout <Typ>
```

Hier bezeichnet `<Typ>` die Einfügung, deren Layout definiert wird. Es gibt vier Möglichkeiten.

1. Das Layout für eine existierende Einfügung wird geändert. In diesem Fall kann `<Typ>` folgendes sein: `Algorithm`, `Branch`, `Box`, `Box:shaded`, `Caption:Standard`, `ERT`, `Figure`, `Foot`, `Index`, `Info`, `Info:menu`, `Info:shortcut`, `Info:shortcuts`, `Listings`, `Marginal`, `Note:Comment`, `Note:Note`, `Note:Greyedout`, `Table`, oder `URL`.

2. Das Layout für eine neue flexible Einfügung wird definiert. In diesem Fall muss `<Typ>` in der Form `Flex:<Name>` sein, wobei `Name` ein beliebiger gültiger Bezeichner sein kann, der in keiner anderen existierenden Einfügung benutzt wird. Der Bezeichner darf Leerzeichen enthalten, dann muss aber der komplette Typ in Anführungszeichen gesetzt werden. Beachten Sie, dass die Definition einer flexiblen Einfügung auch einen `LyXType`-Eintrag enthalten *muss*, der festlegt, welcher Einfügungstyp definiert wird.

3. Das Layout für einen benutzerdefinierten Zweig wird definiert. In diesem Fall muss `<Typ>` die Form `Branch:<Name>` haben, wobei `Name` ein beliebiger existierender Bezeichner für einen im Dokument definierten Zweig sein muss. Der Bezeichner darf Leerzeichen enthalten, dann muss aber der komplette Typ in Anführungszeichen gesetzt werden. Der Hauptzweck einer solchen Definition ist es, spezifische Zweige in LaTeX-Code einzubetten.

4. Das Layout einer benutzer- oder klassenspezifischen Legende wird definiert. In diesem Fall muss `<Typ>` die Form „`Caption:<Name>`“ haben, wobei `Name` den Namen der Legende spezifiziert, wie er im Menü erscheint. Schauen Sie sich die Standard-Legende (`Caption:Standard`), die spezifischen Legenden der KOMA-Script-Klassen (`Caption:Above`, `Caption:Below`) oder das Modul Mehrsprachige Legenden (`Caption:Bicaption`) für Beispiele an.

Die `InsetLayout`-Definition kann folgende Einträge enthalten:

`AddToToc` [`string=""`] Diese Einfügung erscheint im Inhaltsverzeichnis des spezifizierten Typs. Bei einer leeren Zeichenkette erscheint sie nirgends. Siehe auch `OutlinerName` und `IsTocCaption`. Dies ist nur für Flex-Einfügungen implementiert. Standard: deaktiviert.

`AllowedInInsets` enthält eine kommaseparierte Liste von Einfügungen, in die diese Einfügung platziert werden kann. Muss mittels “`EndAllowedInInsets`” beendet werden. Falls Sie die Einfügung (auch) in bestimmten Argumenten der entsprechenden Einfügung erlauben möchten, hängen Sie den Namen des Arguments nach @ an (bspw. `Meine_Einfuegung@post:1`). Beachten Sie, dass im Moment nur Einfügungen berücksichtigt werden, die eine Hierarchiestufe höher stehen (keine tiefere Verschachtelung). Siehe auch `AllowedInLayouts`.

`AllowedInLayouts` enthält eine kommaseparierte Liste von Stilen, in die diese Einfügung platziert werden kann. Muss mittels “`EndAllowedInLayouts`” beendet werden. Beachten Sie, dass im Moment nur Stile berücksichtigt werden, die die Einfügung unmittelbar enthalten (keine tiefere Verschachtelung). Siehe auch `AllowedInInsets`.

`AllowedOccurrences` [`int`] Falls `AllowedInInsets` oder `AllowedInLayouts` definiert wurde, kann hiermit eine maximale Anzahl von Einfügungen pro Absatz festgelegt werden.

AllowedOccurrencesPerItem [*0*, 1] Wenn dies auf `true` gesetzt wird, zählen die `AllowedOccurrences` je Absatz, sofern wir uns in einer listenartigen Umgebung (mit `\item`s) befinden.

Argument [int] Definiert die Argumentnummer eines Befehls/einer Umgebung, die im aktuellen Layout definiert ist. Die Definition muss mit `EndArgument` beendet werden. Siehe Kap. 5.3.11 für Details.

BabelPreamble Präambel um Sprachbefehle zu modifizieren; siehe Kap. 5.3.8.

BgColor [<Name>] ist die Hintergrundfarbe der Einfügung. Siehe Anhang B für eine Liste von verfügbaren Farbnamen.

ContentAsLabel [*0*, 1] Ob der Inhalt der Einfügung als Marke verwendet werden soll, wenn die Einfügung geschlossen ist.

CopyStyle [<Typ>] Wie bei Absatz-Layouts, siehe Kap. 5.3.7. Beachten Sie, dass der komplette Typ angegeben werden muss, z. B. `CopyStyle Flex:<Name>`.

CustomPars [*0*, 1] zeigt an, ob der Benutzer in dieser Einfügung den Absatzeinstellungen-Dialog benutzen darf.

Decoration kann `Classic`, `Minimalistic`, oder `Conglomerate` sein. Es beschreibt den Darstellungsstil für den Einfügungsrahmen und die -knöpfe. Fußnoten benutzen im allgemeinen `Classic`, TEX-Code `Minimalistic` und Zeichenstile `Conglomerate`.

Display [0, *1*] Nur sinnvoll wenn der `LatexType Environment` ist. Gibt an, ob die Umgebung in der Ausgabe abgesetzt erscheint oder in einer Zeile mit dem umgebenden Text. Wenn auf `0` gesetzt, wird angenommen, dass die LATEX-Umgebung Leerraum nach den `\begin{`*LatexName*`}` und `\end{`*LatexName*`}` Befehlen ignoriert (inklusive des Zeilenumbruchzeichens).

EditExternal [*0*, 1] Erlaubt es, dass der Inhalt der Einfügung (mit dem für das Dokument-Ausgabeformat definierten Bearbeitungsprogramm) extern bearbeitet wird.

End beendet die `InsetLayout`-Definition.

Font wird für den Text *und* die Marke benutzt (siehe Kap. 5.3.13). Beachten Sie, dass die Definition dieses Fonts automatisch dem `LabelFont` denselben Wert zuweist, das heißt `Font` muss zuerst definiert werden und `LabelFont` danach, wenn sie unterschiedlich sein sollen.

FixedWidthPreambleEncoding [*0*, 1] Ob eine Zeichenkodierung mit „fester Breite“ für den übersetzten Inhalt von `BabelPreamble` und `LangPreamble` erzwungen wird. Dies wird für spezielle LATEX-Pakete wie `listings` benötigt, die keine variable Zeichenkodierung wie `utf8` erlauben. Diese Einstellung wird ignoriert,

wenn LaTeX-Varianten wie XeTeX oder LuaTeX verwendet werden, die Unicode voll unterstützen.

ForceLocalFontSwitch [*0*, 1] Legt fest, ob, wenn Babel verwendet wird, in dieser Einfügung immer eine lokale Umschaltung der Sprache erfolgen soll (mittels `\foreignlanguage`), also nie eine globale (mittels `\selectlanguage`).

ForceLTR erzwingt die „LaTeX-Sprache“ und führt zu einer Links-nach-rechts-Ausgabe, zum Beispiel bei TeX-Code oder URL. `ForceLTR` ist eine Behelfslösung.

ForceOwnlines [*0*, 1] erzwingt einen Zeilenumbruch in der LaTeX-Ausgabe vor und nach der Einfügung. Dies stellt sicher, dass die Einfügung in eigenen Zeilen ausgegeben wird, um die Ausgabe später besser anderweitig einfacher verändern zu können.

ForcePlain [*0*, 1] erzwingt die Verwendung von `PlainLayout`, wobei der Benutzer den Absatzstil der Einfügung nicht ändern darf.

FreeSpacing [*0*, 1] Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

HTML* Diese Tags kontrollieren die XHTML-Ausgabe. Siehe Kap. 5.4.

InToc [*0*, 1] Legt fest, ob der Inhalt der Einfügung für die Einträge im Fenster „Gliederung des Dokuments“ verwendet werden soll, und zwar für alle Inhaltsverzeichnisse, unabhängig von der Einstellung `AddToToc`. Zum Beispiel sollte der Inhalt einer Fußnote in einer Überschrift nicht im Abschnittstitel im Inhaltsverzeichnis des Gliederungs-Fensters erscheinen, sehr wohl aber der Inhalt bestimmter Zeichenstile.

IsTocCaption [*0*, 1] Wenn dies auf `1` gesetzt ist und `AddToToc` aktiviert ist, fügt die Einfügung eine Zusammenfassung ihres Inhalts zum spezifizierten Inhaltsverzeichnis. Ansonsten scheint dort nur die Marke auf.

KeepEmpty [*0*, 1] Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

LabelFont ist die für die Marke benutze Schrift (siehe Kap. 5.3.13). Beachten Sie, dass diese Definition niemals vor `Font` erscheinen darf, weil sie sonst unwirksam ist.

LabelString [`string=""`] wird auf dem Knopf der Einfügung und anderswo als Marke angezeigt. Einige Einfügungstypen (TeX-Code und Zweig) ändern diese Marke temporär.

LangPreamble Sprachabhängige Präambel; siehe Kap. 5.3.8.

LatexName [`<Name>`] ist der Name der LaTeX-Umgebung oder des LaTeX-Befehls.

LatexParam [<Parameter>] ist ein optionaler Parameter für den zugehörigen LatexName, einschließlich möglicher Klammerpaare wie []. Dieser Parameter kann in LyX nicht geändert werden (man verwendet **Argument** für anpassbare Parameter). Dieser wird nach allen anderen LaTeX-**Argument**en ausgegeben.

LatexType [Command, Environment, None] Wie der Stil in LaTeX übersetzt wird.[18]

None bedeutet nichts Spezielles

Command bedeutet *LatexName*{...}

Environment bedeutet \begin{*LatexName*}...\end{*LatexName*}.

Zusammenfassend bedeutet das, dass die LaTeX-Ausgabe entweder:

```
\LatexName[LatexParam]{...}
```

oder:

```
\begin{LatexName}[LatexParam] ... \end{LatexName}
```

sein wird, je nach LaTeX-Typ.

LeftDelim [string] Eine Zeichenkette, die zu Beginn des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit
 angegeben.

LyxType kann charstyle, custom, element oder end (zeigt das Ende einer Definition an) sein. Dieser Eintrag wird für flexible Einfügungen benötigt und ist nur dort sinnvoll. Neben anderen Dingen legt er fest, in welchem Menü diese Einfügung erscheinen wird. Wird LyxType auf charstyle gesetzt, wird MultiPar automatisch auf 0 und ForcePlain automatisch auf 1 gesetzt. MultiPar kann auf 1 oder ForcePlain auf 0 für charstyle-Einfügungen gesetzt werden, indem es *nach* dem LyxType spezifiziert wird.

MenuString [string] Eine spezifische Zeichenkette für das Menü. Sie können ein Tastenkürzel definieren, indem Sie das entsprechende Zeichen an die Zeichenkette mit „|“ abgetrennt anhängen (z. B. Meine Einfügung|M). Diese Spezifikation ist optional. Wenn Sie fehlt, wird der Name der Einfügung (aus der Typ-Spezifikation) für das Menü verwendet.

MultiPar [*0*, 1] zeigt an, ob in dieser Einfügung mehrere Absätze erlaubt sind. Dadurch wird CustomPars auf denselben Wert gesetzt und ForcePlain auf den anderen. Diese können auf andere Werte gesetzt werden, wenn sie *nach* MultiPar benutzt werden.

NeedProtect [*0*, 1] zeigt an, ob *zerbrechliche Befehle* in diesem Layout (mit \protect) geschützt werden sollen. Es zeigt *nicht* an, ob der Befehl selber geschützt werden soll.

[18] LatexType ist vielleicht etwas missverständlich, da diese Regeln auch für SGML-Klassen gelten. Siehe die SGML-Klassendateienfür spezielle Beispiele.

NewlineCmd [string] erlaubt es, ein (von \\) abweichendes Makro für Zeilenumbrüche zu verwenden. Das \ am Anfang des Makros müssen Sie bei der Spezifizierung nicht eingeben.

NoInsetLayout [<Layout>] Löscht ein vorhandenes InsetLayout.

ObsoletedBy [<Layout>] Name eines InsetLayout, das dieses InsetLayout ersetzt. Dies wird verwendet um ein InsetLayout umzubenennen und dabei Rückwärtskompatibilität zu gewähren.

ParbreakIgnored [*0*,1] Wenn dies auf 1 gesetzt wird, werden Absatzwechsel in der Ausgabe ignoriert. Das kann dann sinnvoll sein, wenn der Inhalt von Einfügungen im LyX-Arbeitsbereich auf verschiedenen Zeilen angeordnet werden soll, ohne dass dies eine Wirkung in der Ausgabe hat.

ParbreakIsNewline [*0*, 1] Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

PassThru [*0*, 1] Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

Preamble Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

RefPrefix [string] Das Präfix, das verwendet werden soll, wenn auf Marken dieser Einfügung verwiesen wird. Dies erlaubt die Verwendung von formatierten Querverweisen.

Requires [string] Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

ResetArgs [*0*, 1] Setzt die LaTeX-Argumente dieses Stils zurück (der via Argument definiert wurde). Dies ist nützlich, wenn man einen Stil mit CopyStyle kopiert hat, aber nicht dessen (benötigten und optionalen) Argumente übernehmen will.

ResetsFont [*0*, 1] Wenn dies 1 gesetzt ist, werden Schriftänderungen in der betreffenden Einfügung (in der Ausgabe) neu initiiert, auch dann, wenn die Einfügung in einem Kontext ist, in dem die Schriftänderungen bereits initiiert sind (bspw. `\textbf{Umgebender Text \meineEinfuegung{\textbf{Inhalt}}...}` statt `\textbf{Umgebender Text \meineEinfuegung{Inhalt}...}`. Diese Einstellung ist für solche Makros sinnvoll, die intern Schrifteinstellungen zurücksetzen (bspw. \footnote). Beachten Sie, dass eine nicht angebrachte Aktivierung dieser Einstellung zu unerwünschten Ergebnissen führen kann (so wird z. B. mit `\emph{Umgebender Text \myinset{\emph{Inhalt}}...}` Inhalt nicht kursiv ausgegeben, da \emph zwischen Kursiv und Nicht-Kursiv hin- und herschaltet. Die Voreinstellung ist 0: Schriftänderungen werden innerhalb der Einfügung nicht neu initiiert.

RightDelim [string] Eine Zeichenkette, die am Ende des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit
 angegeben.

Spellcheck [0, *1*] Erlaubt es, den Inhalt der Einfügung auf Rechtschreibung zu überprüfen.

5.3.11. Argumente

Sowohl Absatzlayouts als auch Inset-Layouts erlauben die Definition von Argumenten. Dies ist nützlich für Dinge wie Abschnittsüberschriften und nur mit LaTeX als Ausgabeformat sinnvoll. Jedes Argument (optional oder erforderlich) eines Befehls oder einer Umgebung hat eine eigene Definition (ausgenommen das erforderliche Haupt-Argument des Absatzes). Die Nummer gibt die Reihenfolge des Arguments an. Die Definition muss mit `EndArgument` enden. Ein Befehl mit 2 optionalen Argumenten hat somit diese Struktur:

```
Argument 1
    ...
    EndArgument
    Argument 2
    ...
    EndArgument
```

Innerhalb einer `Argument`-Definition sind die folgenden Spezifikationen möglich:

- `LabelString [string]` Die Zeichenkette, die sowohl im Menü (um dieses Argument einzufügen) als auch auf dem Einfügungsknopf des Arguments erscheint (falls Sie keinen separaten `MenuString` definieren). Für das Menü können Sie ein Tastenkürzel definieren, indem Sie das entsprechende Zeichen an die Zeichenkette mit „|“ abgetrennt anhängen (z. B. `Kurztitel|K`).
- `MenuString [string]` Eine separate Zeichenkette für das Menü. Sie können ein Tastenkürzel definieren, indem Sie das entsprechende Zeichen an die Zeichenkette mit „|“ abgetrennt anhängen (z. B. `Kurztitel|K`). Diese Spezifikation ist optional. Wenn Sie fehlt, wird `LabelString` auch für das Menü verwendet.
- `NewlineCmd [string]` erlaubt es, ein (von `\\`) abweichendes Makro für Zeilenumbrüche zu verwenden. Das `\` am Anfang des Makros müssen Sie bei der Spezifizierung nicht eingeben.
  - `Tooltip [string]` Ein ausführlicherer erklärender Text, der im Werkzeughinweis erscheint, wenn man die Maus über die Argumenteinfügung bewegt.
- `Mandatory` [*0*, 1] Deklariert, ob es sich um ein obligatorisches (1) oder ein optionales (0) Argument handelt. Obligatorische Argumente werden, wenn Sie nicht eingegeben werden, leer ausgegeben, optionale werden in dem Fall unterdrückt. Voreingestellt ist, dass obligatorische Argumente mit `{...}` begrenzt werden, optionale mit `[...]`.

- `Requires [int=0]` Definiert ein anderes Argument bzw. mehrere andere Arguments (mittels ihrer Nummer), welche(s) das vorliegende voraussetzt, wenn es selbst ausgegeben wird. So verlangen LaTeX-Befehle häufig, dass optionale Argumente in jedem Fall ausgegeben werden (notfalls leer), wenn ein weiteres optionales Argument folgt, wie in `\command[][Argument]{Text}`. Das erreicht man durch die Angabe `Requires 1` innerhalb von `Argument 2`. Wenn mehrere Argumente vorausgesetzt werden, separieren Sie diese durch Komma (bspw. `Requires 1,2`)

- `LeftDelim [string]` Definiert ein eigenes linkes Begrenzungszeichen (statt `{` oder `[`). Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

- `RightDelim [string]` Definiert ein eigenes rechtes Begrenzungszeichen (statt `}` oder `]`). Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

- `DefaultArg [string]` Definiert ein Argument, das nur eingefügt wird, wenn der Nutzer kein Argument eingefügt hat. Das heißt, wenn keine Argument-Einfügung eingefügt wurde oder sie eingefügt wurde, aber leer ist. Mehrere Argumente werden durch Kommas getrennt.

- `PresetArg [string]` Definiert ein Argument, das in jedem Fall eingefügt wird (allein oder zusätzlich zu benutzerdefinierten Argumenten). Mehrere Argumente werden durch Kommas getrennt.

- `Font` Die Schrift, die für den Argumentinhalt verwendet wird; siehe Kap. 5.3.13.

- `FreeSpacing` [*0*, 1] Wie bei Absatz-Layouts, siehe Kap. 5.3.7.

- `LabelFont` Die Schrift, die für die Marke verwendet wird; siehe Kap. 5.3.13.

- `Decoration` [*`Classic`*, `Minimalistic, Conglomerate`] legt den Anzeigestil für den Rahmen und Knopf der Einfügung fest.

- `AutoInsert [int=0]` Wenn dies auf `1` gesetzt wird, wird dieses Argument automatisch eingefügt, sobald der betreffende Absatzstil ausgewählt wird.

- `InsertOnNewline [int=0]` Wenn dies auf `1` gesetzt ist, wird dieses Argument mit `AutoInsert` auf eine neue Zeile gesetzt (nur mit Flex-Einfügungen verfügbar).

- `InsertCotext [int=0]` Wenn dies auf `1` gesetzt wird, wird dieses Argument mit einer Kopie des umgebenden Texts (entweder der ausgewählte Text oder der ganze Absatz) als Inhalt eingefügt.

- `PassThru` [*`inherited`*, `true, false`] bestimmt, ob der Inhalt dieses Arguments in unbearbeiteter Form, ohne spezifische Bearbeitung, die LaTeX verlangen würde, ausgegeben wird. In der Voreinstellung wird der Status von

`PassThru` von der Einfügung oder dem Absatzstil, zu dem das Argument gehört, übernommen. Wenn `true` oder `false` angegeben ist, wird der Status für das Argument verändert.

- `PassThruChars` [`Buchstabenkette`] bestimmt einzelne Zeichen, für die der Inhalt dieses Arguments in unbearbeiteter Form, ohne spezifische Bearbeitung, die LaTeX verlangen würde, ausgegeben wird. Beachten Sie, dass anders als bei `PassThru` ein Wert für Argumente explizit spezifiziert werden muss. Argumente übernehmen also nicht die `PassThruChars` des zugehörigen Absatzstils oder der zugehörigen Einfügung.

- `IsTocCaption` [*0*, 1] Wenn dies `1` gesetzt wird, wird dieses Argument seinen Inhalt im zugehörigen Inhaltsverzeichnis ausgeben. Siehe `AddToToc`.

Standardmäßig ist der Text, der in die LyX-Arbeitsumgebung (außerhalb von Argument-Einfügungen) im entsprechenden Format eingegeben wird, zugleich das letzte (obligatorische) Argument eines Befehls, sofern der `LatexType` auf `Command` gesetzt ist. Argumente mit dem Präfix `post:` werden jedoch nach diesem Arbeitsumgebungs-Argument ausgegeben. Beachten Sie, dass die Nummerierung solcher Post-Argumente wieder bei 1 beginnt. Das erste Argument, das dem Arbeitsumgebungs-Argument folgt, ist somit `post:1`. Post-Argumente werden in allen anderen `LatexType` außer `Command` ignoriert.

Argumente für Listen-`\item`s (wie in `\item[foo]`) haben das Präfix `item:` gefolgt von der Nummer (z. B. `Argument item:1`)

Schließlich gibt es noch einen Argumenttyp mit dem Präfix `listpreamble:`. Strenggenommen ist das, was damit erfasst wird, kein Argument, aber der Mechanismus der LyX-Argumente wird verwendet (daher folgt dem Präfix auch wie üblich eine Nummer, bspw. `Argument listpreamble:1`). Wie der Name andeutet, zielt dieser Argumenttyp auf Listen wie Auflistung, Aufzählung, Beschreibung oder Literaturverzeichnis. Sein Inhalt wird am Start der Liste, vor dem ersten `\item`, auf einer eigenen Zeile ausgegeben (eine Stelle, die sonst in LyX nicht zugänglich ist). Auf diese Weise können Benutzer in individuellen Listen Re-Definitionen (bspw. von Längen) vornehmen. In der Voreinstellung haben diese Argumente keine Begrenzungszeichen.

5.3.12. Zähler

Zähler (CHAPTER, FIGURE,...) müssen in der Textklasse selbst definiert werden. Die Standardzähler sind in der Datei `stdcounters.inc` definiert, so dass Sie nur die Zeile

```
Input stdcounters.inc
```

zu Ihrer Layout-Datei hinzufügen müssen, damit diese arbeiten.Wenn Sie darüber hinaus eigene Zähler definieren wollen, können Sie das wie folgt tun. Zähler-Deklarationen beginnen mit

```
Counter <Name>
```

wobei `<Name>` der Name Ihres Zählers ist. Die Deklaration endet mit `End`.
Folgende Parameter können auch benutzt werden:

`InitialValue` [`int=1`] Setzt den Startwert für einen Zähler, auf den er zurückgesetzt wird. Meist entspricht die Voreinstellung `1` bereits dem Gewünschten.

`LabelString` [`string=""`] definiert, wie der Zähler dargestellt wird. Hierdurch wird `LabelStringAppendix` auf denselben Wert gesetzt. In der Zeichenkette können folgende Konstrukte benutzt werden:

- `\thecounter` wird durch den Wert von `LabelString` (oder `LabelStringAppendix`) des Zählers `counter` ersetzt.
- Zählerwerte können durch LaTeX-ähnliche Makros wie `\`*`numbertype`*`{`*`counter`*`}` ausgedrückt werden, wobei *`numbertype`* Folgendes sein kann:[19] `arabic` für arabische Ziffern: 1, 2, 3, ...; `alph` für Kleinbuchstaben: a, b, c, ...; `Alph` für Großbuchstaben: A, B, C, ...; `roman` für kleine römische Ziffern: i, ii, iii, ...; `Roman` für große römische Ziffern: I, II, III, ...

Wenn `LabelString` nicht definiert ist, wird ein Standardwert wie folgt gesetzt: Wenn der Zähler einen Hauptzähler `master` (über `Within` definiert) hat, in Form der Zeichenkette `\themaster.\arabic{counter}` benutzt, ansonsten als `\arabic{counter}`.

`LabelStringAppendix` [`string=""`] ist dasselbe wie `LabelString`, aber für den Anhang.

`PrettyFormat` [`string=""`] Ein Format, das für *formatierte Querverweise* auf einen Zähler verwendet wird. Möchte man z. B. Verweise auf Abschnitte in der Form „Abschnitt 2.4“ haben, kann die Zeichenkette `##` oder eine Zählerspezifikation wie in `LabelString` enthalten. Im ersteren Fall werden die Zeichen `##` später durch die referenzierte Abschnittsnummer ersetzt. Für Abschnitte würde man also beispielsweise `Abschnitt ##` verwenden. Eine andere Möglichkeit wäre `\S\,\arabic{section}`; dies würde als § 2.7 ausgegeben).

`RefFormat` [`string`, `string`] Für die Verwendung mit „Formatierten Querverweisen“ gedacht, vor allem dann, wenn ein einzelner Zähler mit verschiedenen Stilvarianten verwendet wird. So wird etwa der Zähler `theorem` oft mit verschiedenartigen Theoremvarianten verwendet: Theorem, Lemma etc. Das erste String-Argument spezifiziert ein Präfix, das in den Marken verwendet wird (bspw. „thm“ oder „lem“), das zweite eine Formatierungsangabe, wie Sie auch für `LabelString` oder `PrettyFormat` verwendet wird. Wenn `RefFormat` nicht definiert ist, wird auf `PrettyFormat` zurückgegriffen.

[19] Genau genommen ist die Situation etwas komplizierter: jeder `numbertype` mit Ausnahme der im Folgenden beschriebenen generiert arabische Ziffern. Möglicherweise ändert sich das in der Zukunft, verlassen Sie sich also nicht darauf.

Within [string=""] Wenn dies auf den Namen eines anderen Zählers gesetzt wird, wird der gegenwärtige Zähler jedes Mal zurückgesetzt, wenn der andere erhöht wird. Zum Beispiel wird `subsection` zurückgesetzt, wenn `section` erhöht wird.

5.3.13. Beschreibung des Zeichensatzes

Eine Zeichensatzbeschreibung sieht folgendermaßen aus:

```
Font oder LabelFont oder DefaultFont
  ...
EndFont
```

Folgende Parameter sind verfügbar:

Color [string] Siehe Anhang Anhang B für zulässige Argumente.

Family [*Roman*, Sans, Typewriter]

Misc [string] Zulässige Argumente sind: `emph`, `noun`, `strikeout`, `underbar`, `uuline`, `uwave`, `no_emph`, `no_noun`, `no_strikeout`, `no_bar`, `no_uuline` und `no_uwave`. Jedes schaltet die entsprechende Eigenschaft an oder aus; zum Beispiel führt `emph` zum Stil *Hervorhebung* und `no_emph` schaltet diesen aus.
Falls Sie Letzteres verwirrt, erinnern Sie sich, dass die Schrifteinstellungen standardmäßig von den umgebenden Stilen übernommen wird. Daher schaltet `no_emph` die *Hervorhebung* aus, die z. B. in einer Theorem-Umgebung aktiv ist.

Series [*Medium*, Bold]

Shape [*Up*, Italic, SmallCaps, Slanted]

Size [tiny, small, *normal*, large, larger, largest, huge, giant]

5.3.14. Beschreibung der *Cite Engine*

Die `CiteEngine`-Blöcke, wie sie vor allem in den Cite-Engine-Dateien verwendet werden (siehe Abschnitt Kap. 5.2.6), legen fest, wie sich die Literaturverweisbefehle, die von einer bestimmten „Cite Engine“ unterstützt werden, verhalten. Als „Cite Engine“ wird in LyX eine spezifische Art und Weise, Literaturverweise zu gestalten, bezeichnet. Dabei können etwa Nummern, Autorennamen und/oder Jahresangaben oder anderes zum Einsatz kommen. Momentan unterstützt LyX drei Engine-Typen, nämlich:

1. `default`: die Standardmethode von BibTeX, Literaturverweise darzustellen, ein einfacher nummerischer Stil (bspw. „[1]“)

2. `authoryear`: Literaturverweise im „Harvard-Stil“ mithilfe von Autorennamen und Publikationsjahren (bspw. „Schmidt und Müller (2017b)“)

3. `numerical`: erweiterte nummerische Literaturverweise, bei denen auch Autorennamen oder Titel neben der Nummer ausgegeben werden können (bspw. „Schmidt und Müller [1]“)

`CiteEngine`-Blöcke sehen so aus:

```
CiteEngine default
 cite
 Citep*[][]
 citeyearpar[][]=parencite*
 ...
End
```

Das Argument, dass der Marke `CiteEngine` folgt, bezeichnet den Engine-Typ. Die folgenden Zeilen definieren jeweils einen Literaturverweisbefehl, der von der Engine unterstützt wird, bzw. eine komplexere Literaturverweisangabe. Die Zeilen können im einfachsten Fall einfach einen Literaturverweisbefehl enthalten, der in dieser Form sowohl in der LyX-Datei als auch in der LaTeX-Ausgabe verwendet wird, oder auch komplexere Angaben, bei denen weiter differenziert wird. Die vollständige Syntax ist:

```
LyXName|alias$*<!_stardesc!_stardesctooltip>[][]=latexcmd
```

- `LyXName`: Der Name des Befehl, wie es in der LyX-Datei verwendet wird.

  Um den Wechsel zwischen verschiedenen Engines möglichst zu erleichtern, versuchen wir, für Befehle in verschiedenen Paketen, die dieselbe Ausgabe erzeugen, denselben `LyXName` zu verwenden, auch wenn die hierfür verwendeten LaTeX-Befehle sich unterscheiden (viele sind nach den Befehlen in Natbib benannt). Wenn sich der LaTeX-Befehl vom `LyXName` unterscheidet, wird ein `latexcmd` spezifiziert.

- `alias`: eine (kommaseparierter) Liste von Befehlen, die in der aktuellen Engine auf den aktuellen `LyXName` zurückgesetzt werden. Auch dies dient der Erleichterung eines Wechsels zwischen Engines. Der `alias` ist vergleichbar mit der Marke `ObsoletedBy` in Layout-Definitionen.

- `latexcmd`: Der auszugebende LaTeX-Befehl.

`Alias` und `latexcmd` sind optional. Wenn kein `latexcmd` spezifiziert ist, wird der `LyXName` als LaTeX-Befehl verwendet.

Beachten Sie außerdem:

- Großschreibung zeigt an, dass der Befehl auch eine großgeschriebene Variante hat (`\Latexcmd` vs. `\latexcmd`). Diese Varianten erzwingen üblicherweise die Großschreibung von Namenspräfixen (*von Goethe* ⇒ *Von Goethe*).

- Eckige Klammern `[]` zeigen die Zahl der optionalem Argumente an (0 bis 2).
- Ein Sternchen `*` zeigt an, dass es auch eine Sternvariante des Befehls gibt (`\latexcmd*` vs. `\latexcmd`).

  In der Voreinstellung hat ein Sternbefehl die Bedeutung: Gib eine vollständige Liste der Autorennamen aus, selbst wenn diese Liste aufgrund der Angabe `MaxCiteNames` mittels „et al.“ gekürzt werden müsste.

  Sollte das Sternchen für den aktuellen Befehl eine andere Bedeutung haben, kann diese in spitzen Klammern spezifiziert werden: `<!_stardesc!_stardesctooltip>`. Maximal zwei übersetzbare Makro-Schlagwörter, durch das Präfix `!_` markiert, können hier angegeben werden. Das erste weist auf die Zeichenkette, mit der der Text „Full aut&hor list“ (dt. „Alle Autoren“) zum entsprechenden Ankreuzfeld im Literaturverweisdialog ersetzt werden soll, das zweite zu einem optionalen Werkzeughinweis für dieses Ankreuzfeld.

  Die Makros selbst müssen in einem `CiteFormat` (siehe nächster Abschnitt) definiert werden, wobei `!` vom Präfix weggelassen wird, etwa so:

  ```
  _stardesc Marke für den S&ternbefehl
  _stardesctooltip Hinweise für das Sternbefehl-Ankreuzfeld.
  ```

- Ein Dollar-Zeichen `$` zeigt an, dass dieser Befehl „qualifizierte Literaturverweislisten“ (*qualified citation lists*) unterstützt. Dies ist ein Biblatex-spezifisches Feature für Mehrfachverweise. Dabei kann jeder Verweis in einer solchen Mehrfachverweisliste einen individuellen Text vor und nach dem Verweis haben. Bitte lesen Sie das Biblatex-Handbuch für Einzelheiten.

Wenn Sie einen Literaturverweisbefehl zu einer Engine hinzufügen möchten (zum Beispiel einen speziellen von der Dokumentklasse bereitgestellten Befehl), können Sie `AddToCiteEngine <Engine-Typ> ... End` verwenden. Beachten Sie, dass nur Literaturverweisbefehle hinzugefügt werden, die noch nicht existieren.

5.3.15. Beschreibung des Literaturverweisformats

`CiteFormat`-Blöcke werden verwendet um zu beschreiben, wie bibliographische Informationen dargestellt werden sollen, und zwar sowohl in LyX selbst (bspw. im Literaturverweis-Dialog und in Werkzeughinweisen) als auch in der XHTML-Ausgabe. Diese Blöcke sehen etwa so aus:

```
CiteFormat
 article ...
 book ...
End
```

oder so:

```
CiteFormat
 cite ...
 citet*[][] ...
End
```

Im ersten Fall definieren die einzelnen Zeilen, wie bibliographische Informationen bestimmter Eintragstypen (wie `article` oder `book`) dargestellt werden sollen; solche Definitionen können für jeden Eintragstyp gegeben werden, die in einer BibTEX-Datei enthalten sein können. LYX definiert ein eintragstypunabhängiges Standardformat im Quellcode, das verwendet wird, sollte keine andere Definition gegeben werden. Außerdem definiert LYX mehrere Formate in der Datei `stdciteformats.inc` vor, die in den meisten Dokumentklassen eingebunden ist.

Im zweiten Fall definieren die Zeilen, wie ein bestimmter Literaturverweisbefehl (im Beispiel oben `\cite`, `\citet`) auf der Literaturverweiseinfügung, im Literaturverweisdialog, im Menü oder in der XHTML-Ausgabe dargestellt werden soll. LYX bringt Definitionen für die Zitierstilvarianten, die über Dokument ▷ Einstellungen ▷ Literaturverzeichnis unterstützt werden, mit. Sie sind in spezifischen `*.citeengine`-Dateien, die mit LYX mitgeliefert werden, enthalten (siehe Abschnitt Kap. 5.2.6).

Die Definitionen verwenden eine einfache Notation, bei denen Platzhalter mit entsprechenden Werten aus der BibTEX-Datei ersetzt werden. Die Platzhalter müssen in %-Zeichen eingeschlossen werden, bspw. `%author%`. Eine einfache Definition würde also etwa so aussehen:

```
misc %author%, "%title%".
```

Das würde den Autornamen, gefolgt von einem Komma, gefolgt vom Titel in Anführungszeichen, gefolgt von einem Punkt ausgeben.

Manchmal möchte man einen Wert aber nur dann ausgeben, wenn er existiert. Hierfür kann man eine Bedingungskonstruktion verwenden, bspw.: `{%volume%[[Bd. %volume%]]}`.
Das bedeutet: Wenn der Wert `volume` für den entsprechenden Eintrag existiert, dann gib „Bd. “ gefolgt von dem Wert aus. Es ist auch möglich, eine „Sonst“-Klausel anzugeben, etwa:
`{%author%[[%author%]][[%editor% (Hg.)]]}`
Hier wird der Wert `author` ausgegeben, wenn er existiert, sonst `editor` gefolgt von „ (Hg.)“. Beachten Sie, dass der Platzhalter wieder in %-Zeichen eingeschlossen ist; die gesamte Bedingungskonstruktion ist in geschweiften Klammern eingeschlossen, die „Wenn-“ und „Sonst-“Klauseln in eckigen Klammern, „`[[`“ und „`]]`“. Zwischen diesen darf es keine Leerzeichen geben.

Neben den Wertplatzhaltern gibt es spezielle Marken, die für solche Bedingungen verwendet werden können:

- `{%dialog%[[wahr]][[falsch]]}`: führt den „Wahr“-Teil in Dialogen aus, den „Falsch“-Teil in anderen Kontexten (Arbeitsbereich, Export)

- `{%export%[[wahr]][[falsch]]}`: führt den „Wahr“-Teil beim Export und in Menüs aus, den „Falsch“-Teil in anderen Kontexten (Arbeitsbereich, Dialoge)

- `{%next%[[wahr]]}`: führt den „Wahr“-Teil aus, wenn ein weiterer Eintrag folgt (in einem Mehrfachliteraturverweis)

- `{%second%[[wahr]][[falsch]]}`: führt den „Wahr“-Teil aus, wenn dies der zweite von mehreren Einträgen ist, sonst den „Falsch“-Teil.

- `{%ifstar%[[wahr]][[falsch]]}`: führt den „Wahr“-Teil für Sternchenbefehle (wie `\cite*`) aus, den „Falsch“-Teil für andere.

- `{%ifentrytype:<type>%[[wahr]][[falsch]]}`: führt den „Wahr“-Teil aus, wenn der aktuelle Eintragstyp `<type>` ist, sonst den „Falsch“-Teil (bspw. in einer Literaturverweisdefinition: `{%ifentrytype:book%[[das ist ein Buch]][[das ist kein Buch]]}`)

- `{%ifmultiple:<authortype>%[[wahr]][[falsch]]}`: führt den „Wahr“-Teil aus, wenn der aktuelle Autortyp (`author`, `editor` etc.) mehrere Autoren hat, sonst den „Falsch“-Teil (bspw. in einer Literaturverzeichnisdefinition: `{%ifmultiple:editor%[[Hgg.]][[Hg.]]}`)

- `{%ifqualified%[[wahr]][[falsch]]}`: führt den „Wahr“-Teil aus, wenn der aktuelle Literaturverweis eine qualifizierte Literaturverweisliste ist (ein spezifisches Biblatex-Format für Mehrfachverweise), sonst den „Falsch“-Teil.

Wir haben oben gesagt, dass `%author%` den Wert des Autoreintrags ausgibt, wie er in der Bibliographiedatenbank gespeichert ist. Das ist vielleicht nicht die Ausgabe, die Sie wünschen, denn das Ergebnis sieht möglicherweise so aus: „Müller, Peter and Schmidt, Maria und Weiß, Jana“ (da „and“ von BibTEX verwendet wird, um Autoren voneinander abzugrenzen). LyX bietet daher einige Methoden, um korrekt formatierte (und auch übersetzte) Autorenlisten zu bekommen. Die folgenden Möglichkeiten stehen zur Verfügung:

1. Für Namenlisten mit Vor- und Nachnamen, geeignet für die Hauptautoren/-herausgeber eines Literatureintrags. Der Teil `<nametype>` zeigt die Art der verlangten Liste an (bspw. `<nametype:author>`):
   - `%abbrvnames:<nametype>%`: Bietet eine ggf. abgekürzte Namenliste (mit „et al.“), nämlich dann, wenn `MaxCiteNames` erreicht ist.
   - `%fullnames:<nametype>%`: Bietet eine volle Namenliste (niemals mit „et al.“ abgekürzt).
   - `%forceabbrvnames:<nametype>%`: Bietet eine in jedem Fall abgekürzte Namenliste (mit „et al.“), unabhängig von `MaxCiteNames`.

2. Alternative Namenlisten mit Vor- und Nachnamen, für den Fall, dass die Reihenfolge der Vor- und Nachnamen innerhalb des Literaturverweises sich ändern (wie in: „Müller, Josef: Ein Aufsatz, in: Maria Schmidt (Hg.): Ein Sammelband"):
   - `%abbrvbynames:<nametype>%`: Bietet eine ggf. abgekürzte Namenliste (mit „et al."), nämlich dann, wenn `MaxCiteNames` erreicht ist.
   - `%fullbynames:<nametype>%`: Bietet eine volle Namenliste (niemals mit „et al." abgekürzt).
   - `%forceabbrvbynames:<nametype>%`: Bietet eine in jedem Fall abgekürzte Namenliste (mit „et al."), unabhängig von `MaxCiteNames`.
3. Schließlich Namelisten, die nur aus Nachnamen bestehen, so wie sie etwa in Autor-Jahr-Verweisen verwendet werden. Hier gibt es keinen `<nametype>`: es wird immer eine Liste mit Autoren ausgegeben, wenn es solche gibt, ansonsten mit Herausgebern (genau so, wie es für Autor-Jahr-Verweise auch üblich ist):
   - `%abbrvciteauthor%`: Bietet eine ggf. abgekürzte Namenliste (mit „et al."), nämlich dann, wenn `MaxCiteNames` erreicht ist.
   - `%fullciteauthor%`: Bietet eine volle Namenliste (niemals mit „et al." abgekürzt).
   - `%forceabbrvciteauthor%`: Bietet eine in jedem Fall abgekürzte Namenliste (mit „et al."), unabhängig von `MaxCiteNames`.

Die Reihenfolge der Vor- und Nachnamen in den ersten beiden Listenvarianten können mit den folgenden Makros angepasst werden:

- `!firstnameform %surname%, %prename%` (der erste Autor in Listen des Typs 1)
- `!othernameform %surname%, %prename%` (weitere Autoren in Listen des Typs 1)
- `!firstbynameform %prename% %surname%` (der erste Autor in Listen des Typs 2)
- `!otherbynameform %prename% %surname%` (weitere Autoren in Listen des Typs 2)

Damit können Sie Namenlisten wie die folgenden bekommen: „Müller, Peter und Maria Schmidt: . . . , in: Jonas Damm und Patricia Grün (Hgg.):. . . ".

In den Definitionen gibt es noch weitere Angaben, die so aussehen: `{!<i>!}`. Hier wird die Formatierung definiert, die für die Erzeugung von „Rich Text" (mit typografischen Auszeichnungen usw.) verwendet wird. Da wir keine HTML-Tags ausgeben wollen, wenn wir reinen Text ausgeben, müssen diese Angaben in `{!` und `!}` eingeklammert werden.

In `CiteFormat`-Blöcken sind auch zwei spezielle Formen von Definitionen möglich. Ein Beispiel für die erste ist:

```
!quotetitle "%title%"
```

Das ist die Definition eines Makros. Es kann verwendet werden wie ein Platzhalter: `%!quotetitle%`. LyX behandelt `%!quotetitle%` genau so wie seine Definition. Das heißt aber auch, dass Sie niemals so etwas tun sollten:

```
!funfun %funfun%
```

LyX wird zwar nicht in eine endlos rekursive Schleife gehen, aber in eine lange, bevor es aufgibt.

Die zweite spezielle Form sieht so aus:

```
B_pptext pp.
```

Dies definiert ein übersetzbares Textsegment, womit es möglich wird, relevante Teile der bibliographischen Angaben zu übersetzen. Auch dies kann in Definitionen wie ein Platzhalter verwendet werden: `%B_pptext%`. Beachten Sie, dass es zwei verschiedene Arten der Übersetzung gibt. Alle Definitionen, die wie das obige Beispiel mit `B_` beginnen werden in die jeweils aktuelle Sprache des Dokuments oder Abschnitts übersetzt (so dass die Übersetzung mit dem ausgegebenen Dokument übereinstimmt). Alle Definitionen, die nur mit Unterstrich beginnen, werden in die aktuelle Sprache der Benutzeroberfläche übersetzt. Das ist die angemessene Übersetzung für Textsegmente, die nur in Dialogfenstern oder auf Knöpfen in der Arbeitsfläche erscheinen, wie etwas diese:

```
_addtobib Add to bibliography only.
```

Viele von diesen übersetzbaren Textsegmenten sind in `stdciteformats.inc` und den verschiedenen `*.citeengine`-Dateien vordefiniert. Beachten Sie, dass dies keine Makros im eben definierten Sinn sind. Sie werden nicht expandiert.

Hier ist abschließend ein Beispiel, das mehrere Möglichkeiten ausschöpft:

```
!authoredit {%author%[[%author%, ]][[{%editor%[[%editor%, %B_edtext%, ]]}]]}
```

Dies definiert ein Makro das den Autornamen ausgibt, gefolgt von einem Komma, falls der Autorname existiert, sonst wird der Name des Herausgebers ausgegeben, gefolgt von der Definition von `B_edtext` bzw. seiner Übersetzung (in der englischen Voreinstellung „ed.“, im Deutschen „Hg.“), falls `editor` definiert ist. Genau diese Definition findet sich auch in `stdciteformats.inc`, sie können sie also in Ihren eigenen Definitionen (oder Re-Definitionen) verwenden, wenn Sie diese Datei zuvor einbinden.

5.4. Spezifikationen der XHTML-Ausgabe

Wie bei LaTeX oder DocBook wird auch das Format von LyX' XHTML-Ausgabe durch Layout-Informationen kontrolliert. Grundsätzlich bietet LyX sinnvolle Voreinstellungen und wie bereits erwähnt generiert es sogar CSS-Stilregeln aus den anderen Layout-Angaben. So nimmt LyX etwa die Information, die für einen Kapitelstil in der `Font`-Deklaration gegeben wird, um CSS zu erzeugen, die die Kapitelüberschriften entsprechend formatiert.

In vielen Fällen müssen Sie also gar nichts tun, um eine akzeptable XHTML-Ausgabe für ihre eigenen Umgebungen, benutzerdefinierten Einfügungen usw. zu bekommen. Aber manchmal ist das vielleicht nötig, und daher bietet LyX eine Reihe von Layout-Tags für die Anpassung der XHTML- und CSS-Ausgabe.

Es gibt zwei Marken, `HTMLPreamble` und `AddToHTMLPreamble`, die außerhalb von Stil- und Einfügungsdeklarationen verwendet werden können. Diese werden in Abschnitt Kap. 5.3.5 besprochen.

5.4.1. Absatzstile

Die Art von XHTML, die LyX für einen jeweiligen Absatz ausgibt, hängt davon ab, ob wir es mit einem normalen Absatz zu tun haben, mit einem Befehl oder einer Umgebung, für die die entsprechende `LaTeXType`-Spezifikation die Ausgabe bestimmt.

Im Fall eines Befehls oder eines normalen Absatzes sieht die XHTML-Ausgabe so aus:

```
<tag attr="Wert">
<labeltag attr="Wert">Marke</labeltag>
Inhalt des Absatzes.
</tag>
```

Die `labeltag`s werden natürlich nur ausgegeben, wenn der Absatz auch eine Marke hat.

Im Fall einer Umgebung, die *keine* Liste ist, sieht die XHTML-Ausgabe so aus:

```
<tag attr="Wert">
<itemtag attr="Wert"><labeltag attr="Wert">Marke der Umge-
bung</labeltag>
Erster Absatz.</itemtag>
<itemtag>Zweiter Absatz.</itemtag>
</tag>
```

Beachten Sie, dass die Marke nur für den ersten Absatz ausgegeben wird, wie es bspw. für Theoreme auch sein sollte.

Im Fall von Listen wird eine der folgenden Ausgabeformen generiert:

```
<tag attr="Wert">
<itemtag attr="Wert"><labeltag attr="Wert">
Listenmarke</labeltag>Erster Listeneintrag.</itemtag>
<itemtag attr="Wert"><labeltag attr="Wert">Listenmarke</labeltag>
Zweiter Listeneintrag.</itemtag>
</tag>
<tag attr="Wert">
<labeltag attr="Wert">Listenmarke</labeltag><itemtag attr="Wert">
Erster Listeneintrag.</itemtag>
<labeltag attr="Wert">Listenmarke</labeltag><itemtag attr="Wert">
Zweiter Listeneintrag.</itemtag>
</tag>
```

Beachten Sie hierbei die unterschiedliche Reihenfolge von `labeltag` und `itemtag`. Welche Reihenfolge ausgegeben wird, hängt von der Einstellung von `HTMLLabelFirst` ab: Wenn `HTMLLabelFirst` auf `false` gesetzt ist (das ist die Voreinstellung), bekommen Sie die erste Variante, mit dem `labeltag` innerhalb des `itemtag`; ist sie hingegen auf `true` gesetzt, bekommen Sie die zweite, mit dem `labeltag` außerhalb des `itemtag`.

Die spezifischen Tags und Attribute, die für Absatztypen ausgegeben werden können mithilfe der Absatzspezifikationen kontrolliert werden, die wir im Folgenden beschreiben. Wir erwähnt verwendet LyX für viele von diesen sinnvolle Voreinstellungen, oftmals müssen Sie also für eine gute XHTML-Ausgabe wenig tun. Die verfügbaren Spezifikationsmöglichkeiten sind also eher dazu da, Dinge zu verändern, wo Sie dies für nötig halten.

`HTMLAttr` [`string`] Legt die Attribute fest, die mit dem Haupt-Tag ausgegeben werden, bspw. `class='meindiv'`. In der Voreinstellung gibt LyX `class='layoutname'` aus, wobei `layoutname` der LyX-Name des Absatzstils in Kleinbuchstaben ist, etwa `chapter`. Hier sollten Sie *keine* Stilinformationen übergeben. Verwenden Sie dafür `HTMLStyle`.

`HTMLForceCSS` [*0*, `1`] Legt fest, ob die Standard-CSS-Information, die LyX für diesen Absatzstil generiert, auch dann ausgegeben werden, wenn Informationen mittels `HTMLStyle` explizit übergeben werden. Wenn Sie dies auf `1` setzen, können Sie diese CSS modifizieren, statt sie komplett zu überschreiben. Voreinstellung: `0`.

`HTMLItem` [`string`] Dieser Tag wird für einzelne Absätze in Umgebungen verwendet. Er ersetzt `itemtag` in den Beispielen oben. Voreinstellung: `div`.

`HTMLItemAttr` [`string`] Attribute für den `itemtag`.
Voreinstellung: `class='layoutname_item'`. Hier sollten Sie *keine* Stilinformationen übergeben. Verwenden Sie dafür `HTMLStyle`.

`HTMLLabel` [`string`] Dieser Tag wird für Absatz- und Listeneintrags-Marken verwendet. Er ersetzt `labeltag` in den Beispielen oben. Voreinstel-

lung: `span`, sofern `LabelType` nicht entweder `Top_Environment` oder `Centered_Top_Environment` ist; die Voreinstellung in diesen Fällen ist `div`.

`HTMLLabelAttr` [`string`] Attribute für den Marken-Tag. Voreinstellung: `class='layoutname_label'`. Hier sollten Sie *keine* Stilinformationen übergeben. Verwenden Sie dafür `HTMLStyle`.

`HTMLLabelFirst` [*0*, 1] Dies ist nur für listenähnliche Umgebungen relevant. Die Spezifikation legt fest, ob der `labeltag` innerhalb oder außerhalb des `itemtag` ausgegeben wird. Dies wird beispielsweise in der Umgebung *Beschreibung* verwendet, bei der wir folgende Ausgabe haben wollen: `<dt>...</dt><dd>...</dd>`. Voreinstellung: `0`, d. h., der `labeltag` wird außerhalb des `itemtag` ausgegeben.

`HTMLPreamble` Informationen, die im Abschnitt `<head>` ausgegeben werden, wenn dieser Absatzstil verwendet wird. Dies könnte man etwa verwenden, um einen `<script>`-Block auszugeben, der eine `onclick`-Routine definiert.

`HTMLStyle` CSS-Stilinformationen, die ausgegeben werden sollen, wenn dieser Absatzstil verwendet wird. Beachten Sie, dass dies automatisch in einen vom Absatzstil generierten `<style>`-Block eingefügt wird, Sie müssen also nur die CSS selbst festlegen. Muss mit `EndHTMLStyle` beendet werden.

`HTMLTag` [`string`] Der Tag, der für die Hauptmarke verwendet wird. In den obigen Beispielen `tag`. Voreinstellung: `div`.

`HTMLTitle` [*0*, 1] Markiert diesen Stil als einen, der in der Titelei (`<title>`) der XHTML-Datei enthalten sein soll. Voreinstellung: `0`. Die Datei `stdtitle.inc` aktiviert dies für den Absatzstil `title`.

5.4.2. InsetLayout und XHTML

Auch die XHTML-Ausgabe von Einfügungen kann durch Layout-Dateien kontrolliert werden.[20] Auch hier versucht LyX sinnvolle Voreinstellungen anzubieten, und es generiert Standard-CSS-Informationen. Aber alles kann angepasst werden.

Die XHTML-Ausgabe für Einfügungen hat folgende Form:

```
<tag attr="Wert">
<labeltag>Marke</labeltag>
<innertag attr="Wert">Inhalt der Einfügung.</innertag>
</tag>
```

Wenn die Einfügung mehrere Absätze zulässt – das heißt, wenn `MultiPar` auf `true` eingestellt ist –, wird der Inhalt der Einfügung in Form von Absätzen ausgegeben, die nach den Vorgaben für diese Absätze formatiert sind (Standard, Zitat usw.). Das

[20] Momentan gilt dies nur für „Text“-Einfügungen (Einfügungen, in die Sie Text eingeben können) und nicht für „Befehls“-Einfügungen (Einfügungen, die mit Dialogfenstern verbunden sind).

`labeltag` wird natürlich nur ausgegeben, wenn die Einfügung eine Marke hat, und aktuell ist sie stets `span`. Das `innertag` ist optional und wird in der Voreinstellung nicht ausgegeben.

Die spezifischen Tags und Attribute, die für Einfügen ausgegeben werden können mithilfe der folgenden Absatzspezifikationen kontrolliert werden.

`HTMLAttr` [`string`] Legt die Attribute fest, die mit dem Haupt-Tag ausgegeben werden, beispielsweise `class='meininset' onclick='...'`. In der Voreinstellung gibt LyX `class='insetname'` aus, wobei `insetname` der LyX-Name der Einfügung in Kleinbuchstaben ist, etwa `footnote`. Nicht-alphabetische Zeichen werden hierbei zu Unterstrichen umgewandelt.

`HTMLForceCSS` [*`0`*, `1`] Legt fest, ob die Standard-CSS-Information, die LyX für diesen Absatzstil generiert, auch dann ausgegeben werden, wenn Informationen mittels `HTMLStyle` explizit übergeben werden. Wenn Sie dies auf `1` setzen, können Sie diese CSS modifizieren, statt sie komplett zu überschreiben. Voreinstellung: `0`.

`HTMLInnerAttr` [`string`] Attribute für den inneren Tag. Voreinstellung: `class='insetname_inner'`.

`HTMLInnerTag` [`string`] Der innere Tag; ersetzt `innertag` in den Beispielen oben. Standardmäßig nicht ausgegeben.

`HTMLIsBlock` [`0`, *`1`*] Legt fest, ob diese Einfügung einen alleinstehenden Textblock repräsentiert (wie etwa eine Fußnote) oder Text, der Teil des umgebenden Textes ist (etwa ein Zweig). Voreinstellung: `1`.

`HTMLLabel` [`string`] Eine Marke für diese Einfügung, möglicherweise mit einem Verweis aus einen Zähler (für Fußnoten etwa `\arabic{footnote}`). Diese Angabe ist optional, es gibt keine Voreinstellung.

`HTMLPreamble` Informationen, die im Abschnitt `<head>` ausgegeben werden, wenn dieser Absatzstil verwendet wird. Dies könnte man etwa verwenden, um einen `<script>`-Block auszugeben, der eine `onclick`-Routine definiert.

`HTMLStyle` CSS-Stilinformationen, die ausgegeben werden sollen, wenn diese Einfügung verwendet wird. Beachten Sie, dass dies automatisch in einen vom Absatzstil generierten `<style>`-Block eingefügt wird, Sie müssen also nur die CSS selbst festlegen. Muss mit `EndHTMLStyle` beendet werden.

`HTMLTag` [`string`] Der Tag, der für die Hauptmarke verwendet wird. In den obigen Beispielen `tag`. Voreinstellung: `div`, wenn `MultiPar` auf `true` eingestellt ist, sonst `span`.

5.4.3. Gleitobjekte und XHTML

Auch die XHTML-Ausgabe für Gleitobjekte kann durch Layout-Dateien kontrolliert werden. Die Ausgabe hat folgende Form:

```
<tag attr="Wert">
Inhalt des Gleitobjekts.
</tag>
```

Die Legende ist, falls eine existiert, eine separate Einfügung und wird als solche ausgegeben. Seine Ausgabe wird somit durch `InsetLayout`-Spezifikationen für Legenden-Einfügungen bestimmt.

`HTMLAttr` [`string`] Legt die Attribute fest, die mit dem Haupt-Tag ausgegeben werden, beispielsweise `class='meingleitobjekt' onclick='...'`. In der Voreinstellung gibt LyX `class='float float-floattype'` aus, wobei `floattype` der LyX-Name dieses Gleitobjekttyps ist, wie er in der Gleitobjekt-Definition festgelegt ist (siehe Abschnitt Kap. 5.3.9), allerdings in Kleinbuchstaben und mit nicht-alphabetischen Zeichen zu Unterstrichen umgewandelt. Beispielsweise `float-table`.

`HTMLStyle` CSS-Stilinformationen, die ausgegeben werden sollen, wenn dieses Gleitobjekt verwendet wird. Beachten Sie, dass dies automatisch in einen vom Absatzstil generierten `<style>`-Block eingefügt wird, Sie müssen also nur die CSS selbst festlegen. Muss mit `EndHTMLStyle` beendet werden.

`HTMLTag` [`string`] Der Tag, der für die Hauptmarke verwendet wird. In den obigen Beispielen `tag`. Voreinstellung: `div`; dies sollten Sie nur in Ausnahmefällen ändern.

5.4.4. Formatierung des Literaturverzeichnisses

Das Literaturverzeichnis kann mittels `CiteFormat`-Blöcken formatiert werden. Siehe Abschnitt Kap. 5.3.15 für weitere Informationen.

5.4.5. Von LyX generierte CSS

Wir haben oft genug erwähnt, dass LyX auf der Grundlage der Absatzstildeklarationen Standard-CSS-Informationen sowohl für Einfügungen als auch für Absatzstile erzeugt. In diesem Abschnitt diskutieren wir kurz, welche Layout-Informationen LyX dabei verwendet, und wie dies geschieht.

Gegenwärtig erzeugt LyX CSS nur für Schriftinformationen. Es greift dabei auf `Family`, `Series`, `Shape`, und `Size` in der `Font`-Deklaration zurück (siehe Abschnitt Kap. 5.3.13). Die Übertragung ist zumeist ziemlich offensichtlich. So wird

aus `Family Sans` im CSS `font-family: sans-serif;`. Das Verhältnis von LyX-Schriftgrößen und CSS-Schriftgrößen ist etwas weniger offensichtlich, aber trotzdem nachvollziehbar. Schauen Sie sich bei Interesse die Funktion `getSizeCSS()` in `src/FontInfo.cpp` an.

6. Externes Material einfügen

ACHTUNG: Dieser Teil der Dokumentation wurde lange nicht aktualisiert. Wir hoffen, dass sie noch akkurat ist, garantieren das aber nicht.

Die Verwendung von „externem Material“ in LyX wird ausführlich im Handbuch *Eingebettete Objekte* beschrieben. Im Folgenden geht es nur darum zu beschreiben, wie das intern funktioniert und wie man neue Vorlagen erstellen kann.

6.1. Wie funktioniert das?

Die Einfügung Externes Material basiert auf dem Konzept der *Vorlage*. Eine Vorlage ist eine Spezifikation, wie LyX mit einer bestimmten Art von Material umgehen soll. Derzeit gehören zu LyX derartige Vorlagen für XFig-Abbildungen, Dia-Diagramme, diverse Abbildungen im Rasterformat, Gnuplot und noch ein paar mehr. Die vollständige Liste sehen Sie in Einfügen ▷ Datei ▷ Externes Material. Darüber hinaus ist es möglich, durch eigene Vorlagen beliebige andere Formate einzubinden. Wir werden weiter unten beschreiben, was genau Sie dazu machen müssen und hoffen, dass Sie derartig erstellte Vorlagen an das LyX-Team schicken, damit sie in kommenden LyX-Versionen integriert werden können.

Ein weiteres Merkmal der Idee der externen Einfügung ist die Unterscheidung zwischen der ursprünglichen Datei, die als Grundlage für das eingefügte Material dient, und der erzeugten Datei, die dann letztendlich in Ihr Dokument eingebunden wird. Wir wollen dies am Beispiel einer XFig-Abbildung erläutern.

Das Programm XFig bearbeitet eine speziell formatierte Datei mit der Endung `.fig`. In XFig können Sie Ihre Abbildung editieren und ändern, und zum Schluss speichern Sie diese `.fig`-Datei. Wenn Sie nun eine derartige Abbildung in LyX einbinden wollen, müssen Sie zunächst `transfig` starten, um eine PostScript-Datei zu erzeugen, die von LaTeX eingebunden werden kann. In diesem Fall ist also die `.fig`-Datei die oben erwähnte Originaldatei, und die `.ps`-Datei die tatsächlich eingebundene Datei.

Diese Unterscheidung ist wichtig, denn Sie erlaubt das einfache Ändern und Aktualisieren des Materials, während Sie an Ihrem Text schreiben. Außerdem ist erst so die Flexibilität gegeben, die benötigt wird, um unterschiedliche Exportformate für die LyX-Datei zu ermöglichen.

So ist es im Falle einer Ausgabe als reiner (ASCII) Text sicher nicht sinnvoll, eine PostScript-Datei im Rohformat einzubinden. In diesem Fall wird dann entweder nur eine Referenz auf die Bilddatei angegeben, oder aber es wird ein Konverter gestartet, der eine ASCII-Darstellung erzeugt, die in etwa so aussieht wie die ursprüngliche Gra-

fik. Genau dies ist mit der Einfügung Externes Material möglich, denn sie kennt all die notwendigen Befehle für derartige Konvertierungen (sofern sie von LyX unterstützt werden).

Darüber hinaus erlaubt die Einfügung Externes Material aber auch die einfache Integration mit externen Betrachtern und Editoren. So sind Sie bei einer XFig-Abbildung in der Lage, mit einem einigen Klick XFig zu starten, um die Abbildung zu bearbeiten oder die erstellte PostScript-Datei mit `ghostview` zu betrachten. Kein langes Herumsuchen mit Dateimanagern nach den Original- und Grafikdateien mehr, und Sie müssen sich nicht mehr an die unterschiedlichen Parameter erinnern, die vielleicht für diese Abbildung notwendig sind, um sie in der richtigen Größe zu erstellen. Sie haben ohne viel Aufwand Zugriff auf eine Vielzahl von Applikationen und können so Ihre Produktivität ungemein steigern.

6.2. Die Konfigurationsdateien für externe Vorlagen

Es ist ziemlich einfach, eigene externe Vorlagen zu LyX hinzuzufügen. Beachten Sie aber, dass dies, wenn es sorglos geschieht, ziemlich wahrscheinlich ein Sicherheitsproblem mit sich bringt. Bevor Sie dies also tun, raten wir Ihnen dringend, die Sicherheitshinweise in Abschnitt Kap. 6.4 zu lesen.

Trotzdem ermutigen wir Sie, interessante Vorlagen an uns zu schicken.

Die externen Vorlagen sind in Dateien mit der Endung `*.xtemplate` definiert, die im Verzeichnis `LyXDir/lib/xtemplates/` zu finden sind. Jede Vorlage ist in einer eigenen Datei definiert. Ihre eigenen Vorlagen können Sie in `UserDir/xtemplates/` ablegen; Sie können auch existierende Vorlagen dorthin kopieren, um sie zu modifizieren.

Eine typische Vorlage sieht so aus:

```
Template XFig
GuiName "XFig: $$AbsOrRelPathParent$$Basename"
HelpText
An XFig figure.
HelpTextEnd
InputFormat fig
FileFilter "*.fig"
AutomaticProduction true
Transform Rotate
Transform Resize
Format LaTeX
TransformCommand Rotate RotationLatexCommand
TransformCommand Resize ResizeLatexCommand
Product "$$RotateFront$$ResizeFront
         \\input{$$AbsOrRelPathMaster$$Basename.pstex_t}
         $$ResizeBack$$RotateBack"
```

```
UpdateFormat pstex
UpdateResult "$$AbsPath$$Basename.pstex_t"
Requirement "graphicx"
ReferencedFile latex "$$AbsOrRelPathMaster$$Basename.pstex_t"
ReferencedFile latex "$$AbsPath$$Basename.eps"
ReferencedFile dvi "$$AbsPath$$Basename.eps"
FormatEnd
Format PDFLaTeX
TransformCommand Rotate RotationLatexCommand
TransformCommand Resize ResizeLatexCommand
Product "$$RotateFront$$ResizeFront
         \\input{$$AbsOrRelPathMaster$$Basename.pdftex_t}
         $$ResizeBack$$RotateBack"
UpdateFormat pdftex
UpdateResult "$$AbsPath$$Basename.pdftex_t"
Requirement "graphicx"
ReferencedFile latex "$$AbsOrRelPathMaster$$Basename.pdftex_t"
ReferencedFile latex "$$AbsPath$$Basename.pdf"
FormatEnd
Format Ascii
Product "$$Contents(\"$$AbsPath$$Basename.asc\")"
UpdateFormat asciixfig
UpdateResult "$$AbsPath$$Basename.asc"
FormatEnd
Format DocBook
Product "<graphic fileref=\"$$AbsOrRelPathMaster$$Basename.eps\">
         </graphic>"
UpdateFormat eps
UpdateResult "$$AbsPath$$Basename.eps"
ReferencedFile docbook "$$AbsPath$$Basename.eps"
ReferencedFile docbook-xml "$$AbsPath$$Basename.eps"
FormatEnd
Product "[XFig: $$FName]"
FormatEnd
TemplateEnd
```

Wie Sie sehen, ist die Vorlage in `Template` ... `TemplateEnd` eingeschlossen. Sie enthält einen Kopf, in dem generelle Einstellungen vorgenommen werden und Abschnitte `Format` ... `FormatEnd` für jedes unterstützte primäre Zielformat des Dokuments.

6.2.1. Der Vorlagenkopf

`AutomaticProduction true|false` Legt fest, ob die Datei, die von der Vorlagen repräsentiert wird, von LyX erzeugt werden muss. Diese Spezifikation darf nur

einmal vorkommen.

FileFilter <pattern> Ein Glob-Muster, das im Datei-Dialog verwendet wird, um die gewünschten Dateien zu filtern. Wenn es mehr als eine mögliche Dateiendung gibt (tgif etwa hat `.obj` und `.tgo`), verwenden Sie so etwas wie `"*.{obj,tgo}"`. Diese Spezifikation darf nur einmal vorkommen.

GuiName <guiname> Der Text, der auf dem Knopf der externen Einfügung angezeigt wird. Diese Spezifikation darf nur einmal vorkommen.

HelpText <text> HelpTextEnd Der Hilfetext, der im Dialog „externes Material“ angezeigt wird. Geben Sie hier genügend Informationen, um den Benutzern zu verdeutlichen, was die Vorlage bietet. Diese Spezifikation darf nur einmal vorkommen.

InputFormat <format> Das Format der Quelldatei. Das muss der Name eines Formats sein, das LyX kennt (siehe Abschnitt Kap. 3.1). Verwenden Sie `*`, wenn die Vorlage Quelldateien von mehr als einem Format bearbeiten kann. LyX wird dann versuchen, über die Datei selbst herauszufinden, um welches Format es sich aktuell handelt. Diese Spezifikation darf nur einmal vorkommen.

Template <id> Ein eindeutiger Name für die Vorlage. Er darf keine Ersetzungsmakros (siehe unten) enthalten.

Transform Rotate|Resize|Clip|Extra Diese Spezifikation legt fest, welche Transformationen von dieser Vorlage unterstützt werden. Es kann keinmal oder mehrmals vorkommen. Dieser Befehl aktiviert die entsprechenden Reiter im Dialog *Externes Material*. Jede `Transform`-Spezifikation benötigt eine korrespondierende `TransformCommand`- oder `TransformOption`-Spezifikation im `Format`-Abschnitt. Falls dies nicht der Fall ist, wird die Transformation von diesem Format nicht unterstützt.

6.2.2. Der Format-Abschnitt

Format LaTeX|PDFLaTeX|PlainText|DocBook|XHTML Das primäre Dokument-Zielformat, für den diese Formatdefinition gedacht ist. Nicht jede Vorlage bietet sinnvolle Ausgaben in jedes Zielformat. Definieren Sie aber bitte dennoch für alle Zielformat einen `Format`-Abschnitt. Verwenden Sie einen Blindtext, wenn keine vernünftige Definition möglich ist. So können Sie im exportierten Dokument wenigstens einen Hinweis auf die externe Vorlage sehen.

Option <Name> <Wert> Diese Spezifikation definiert ein zusätzliches Makros `$$<Name>` zur Ersetzung in `Product`. `<Wert>` selbst kann Ersetzungsmakros enthalten. Der Vorteil gegenüber der Verwendung von `<Wert>` direkt in `Product` ist, dass der ersetzte Wert von`$$<Name>` validiert wird, so dass es ein gültiges optionales Argument im Dokumentformat ist. Diese Spezifikation kann keinmal oder mehrmals verwendet werden.

`Product <Text>` Der Text, der in das exportierte Dokument eingefügt wird. Das ist die wichtigste Spezifikation überhaupt, und sie kann sehr komplex sein. Diese Spezifikation darf nur einmal vorkommen.

`Preamble <Name>` spezifiziert einen Textbaustein für den LaTeX-Vorspann. Er muss mit `PreambleDef ... PreambleDefEnd` definiert werden. Diese Spezifikation kann keinmal oder mehrmals verwendet werden.

`ReferencedFile <Format> <Dateiname>` Diese Spezifikation benennt Dateien, die vom Konversionsprozess erzeugt werden und die für ein bestimmtes Ausgabeformat benötigt werden. Wenn der Dateiname relativ ist, wird er relativ zum Hauptdokument interpretiert. Diese Spezifikation kann keinmal oder mehrmals verwendet werden.

`Requirement <Paket>` Der Name eines benötigten LaTeX-Pakets. Dieses wird mittels `\usepackage{}` in den LaTeX-Vorspann eingebunden. Diese Spezifikation kann keinmal oder mehrmals verwendet werden.

`TransformCommand Rotate RotationLatexCommand` Diese Spezifikation legt fest, dass der eingebaute LaTeX-Befehl für die Drehung verwendet werden soll. Diese Spezifikation kann keinmal oder einmal verwendet werden.

`TransformCommand Resize ResizeLatexCommand` Diese Spezifikation legt fest, dass der eingebaute LaTeX-Befehl für die Skalierung verwendet werden soll. Diese Spezifikation kann keinmal oder einmal verwendet werden.

`TransformOption Rotate RotationLatexOption` Diese Spezifikation legt fest, dass die Drehung über ein optionales Argument erfolgt. Diese Spezifikation kann keinmal oder einmal verwendet werden.

`TransformOption Resize ResizeLatexOption` Diese Spezifikation legt fest, dass die Skalierung über ein optionales Argument erfolgt. Diese Spezifikation kann keinmal oder einmal verwendet werden.

`TransformOption Clip ClipLatexOption` Diese Spezifikation legt fest, dass das Zuschneiden über ein optionales Argument erfolgt. Diese Spezifikation kann keinmal oder einmal verwendet werden.

`TransformOption Extra ExtraLatexOption` Diese Spezifikation legt fest, dass ein zusätzliches optionales Argument verwendet wird. Diese Spezifikation kann keinmal oder einmal verwendet werden.

`UpdateFormat <Format>` Das Dateiformat der konvertierten Datei. Das muss der Name eines Formats sein, das LyX kennt (siehe Abschnitt Kap. 3.1). Diese Spezifikation darf nur einmal vorkommen. Soll das Ergebnis eine PDF-Datei sein, müssen Sie das Format `pdf6` wählen. Das ist das PDF-Format, das für eingebundene Grafiken verwendet wird. Die anderen definierten PDF-Formate sind für die Dokumentausgabe gedacht.

UpdateResult <filename> Der Dateiname der konvertierten Datei. Der Dateiname muss absolut sein. Diese Spezifikation darf nur einmal vorkommen.

6.2.3. Vorspann-Definitionen

Die Konfigurationsdatei für externe Vorlagen kann zusätzliche Vorspann-Definitionen enthalten, die in `PreambleDef ... PreambleDefEnd` eingeschlossen sind. Sie können von den Vorlagen im jeweiligen `Format`-Abschnitt verwendet werden.

6.3. Der Ersetzungsmechanismus

Wenn über die externe Einfügung ein externes Programm gestartet wird, geschieht dies anhand eines Befehls, der in der Vorlage festgelegt wurde. Ein solcher Befehl kann diverse Makros enthalten, die vor dem eigentlichen Aufruf ausgewertet und ersetzt werden. Die Ausführung erfolgt dabei immer in demjenigen Verzeichnis, das auch das LyX-Dokument enthält.

Außerdem wird der Name des externen Materials für die Ausgabe durch den Ersetzungsmechanismus generiert; auch die meisten anderen Spezifikationen in der Vorlage können den Ersetzungsmechanismus nutzen.

Hier finden Sie eine Liste der Makros für die Ersetzung:

$$AbsOrRelPathMaster Der absolute oder relative Dateipfad zum LyX-Hauptdokument.

$$AbsOrRelPathParent Der absolute oder relative Dateipfad zum LyX-Dokument.

$$AbsPath Der absolute Dateipfad.

$$Basename Der Dateiname ohne Pfad und Endung.

$$Contents("Dateiname.ext") Dieses Makro gibt den Inhalt der Datei mit dem Namen `Dateiname.ext` aus.

$$Extension Die Dateiendung (inklusive Punkt).

$$pngOrjpg Die Zeichenkette „jpg“, wenn die Datei das JPEG-Format hat, sonst „png“. Das ist hilfreich, um unnötige Konversionen für solche Ausgabeformate zu vermeiden, die sowohl PNG als auch JPEG unterstützen. Die vordefinierte Rastergrafik-Vorlage verwendet dies für die Ausgabe in das PDFTEX-Format.

$$FName Der Name der Datei, die im Dialog *Externes Material* ausgewählt wurde. Das ist entweder ein absoluter Dateiname oder ein zum LyX-Dokument relativer.

$$FPath Der Pfad-Teil von `$$FName` (absoluter Dateipfad oder ein zum LyX-Dokument relativer).

$$RelPathMaster Der Dateipfad, relativ zum LyX-Hauptdokument.

$$RelPathParent Der Dateipfad, relativ zum LyX-Dokument.

$$Sysdir Der absolute Pfad zum Systemverzeichnis. Das wird üblicherweise verwendet, um auf die Hilfsskripte zu verweisen, die LyX selbst enthält.

$$Tempname Voller Dateiname (mit Pfad) im temporären Verzeichnis, das automatisch gelöscht wird, wenn das entsprechende Dokument geschlossen oder das externe Material entfernt wird.

Alle Pfad-Makros enthalten einen angehängten Verzeichnis-Trenner, so dass Sie z. B. einen absoluten Dateipfad mittels `$$AbsPath$$Basename$$Extension` generieren können.

Die obigen Makros werden in allen Spezifikationen ersetzt, sollte dies nicht explizit anders vermerkt worden sein. Die Spezifikation `Product` unterstützt zusätzlich noch die folgenden Ersetzungen, sofern sie durch `Transform` und `TransformCommand` aktiviert wurden:

$$ResizeFront Der vordere Teil des Skalierungsbefehls.

$$ResizeBack Der hintere Teil des Skalierungsbefehls.

$$RotateFront Der vordere Teil des Drehungsbefehls.

$$RotateBack Der hintere Teil des Drehungsbefehls.

Die Spezifikation `Option` unterstützt zusätzlich noch die folgenden Ersetzungen, sofern sie durch `Transform` und `TransformCommand` aktiviert wurden:

$$Clip Die Option für das Zuschneiden.

$$Extra Die zusätzliche Option.

$$Resize Die Option für das Skalieren.

$$Rotate Die Option für die Drehung.

Sie fragen sich vielleicht, warum es so viele Pfad-Makros gibt. Vor allem gibt es dafür zwei Gründe:

1. Relative und absolute sollten jeweils relativ oder absolut bleiben, denn die Anwender haben vermutlich Gründe, warum sie diese oder jene Form verwenden. Relative Pfade sind bspw. hilfreich für Dokumente, die weitergegeben werden und die auf verschiedenen Geräten funktionieren sollten. Absolute Pfade werden von manchen Programmen vorausgesetzt.

2. LaTeX behandelt relative Dateinamen anders als LyX und andere Programme in eingebetteten eingebundenen Dateien. Für LyX ist ein relativer Dateiname immer relativ zum Dokument, das den Dateinamen enthält. Für LaTeX ist er immer relativ zum Hauptdokument. Diese zwei Angaben sind identisch, wenn sie keine Unterdokumente haben, aber sie unterscheiden sich, wenn Sie ein Hauptdokument mit Unterdokumenten haben. Das heißt, dass relative Dateinamen umgewandelt werden müssen, wenn Sie für LaTeX aufbereitet werden. Zum Glück macht LyX das automatisch für Sie, wenn Sie die richtigen Makros verwenden.

Welche Pfad-Makros sollten Sie also in neuen Vorlagen-Definitionen verwenden? Die Regel ist nicht schwer:

- Verwenden Sie `$$AbsPath`, wenn ein absoluter Pfad benötigt wird.
- Verwenden Sie `$$AbsOrRelPathMaster`, wenn die ersetzte Zeichenkette eine Form von if LaTeX-Eingabe ist.
- Sonst verwenden Sie `$$AbsOrRelPathParent`, um die Präferenzen der Anwender zu erhalten.

Es gibt Spezialfälle, bei denen diese Regel nicht funktioniert und bei denen bspw. relative Namen gebraucht werden, aber der Normalfall ist das nicht. Ein Beispiel für einen solchen Ausnahmefall isz die Spezifikation

```
ReferencedFile latex "$$AbsOrRelPathMaster$$Basename.pstex_t"
```

in der XFig-Vorlage oben. Wir können hier keinen absoluten Namen verwenden, weil der Kopierer für `.pstex_t`-Dateien relative Namen braucht, um den Dateiinhalt bearbeiten zu können.

6.4. Sicherheitshinweise

Das Feature für externes Material interagiert viel mit externen Programmen, und zwar automatisch. Das bringt potentielle Sicherheitsprobleme mit sich, die zu beachten sind. Da die Möglichkeit besteht, in den Vorlagen beliebige Dateinamen und/oder Parameter zu definieren, und da diese in Befehlsdeklarationen eingefügt werden, ist es theoretisch möglich, bösartige Dokumente zu erzeugen, die beliebige Befehle ausführen können, wenn jemand das Dokument exportiert oder ansieht. Das wollen wir definitiv verhindern.

Da nun aber die externen Programmbefehle ausschließlich in den Konfigurationsdateien für die Vorlagen spezifiziert sind, gibt es keine Sicherheitsprobleme, wenn LyX richtig konfiguriert wurde und nur sichere Vorlagen verwendet werden. Denn externe Programme werden mit dem Systembefehl `execvp` und nicht mit `system` aufgerufen. Das verunmöglicht die Ausführung beliebiger Befehle über die Shell mithilfe von Dateinamen oder -parametern.

Das heißt aber auch, dass die Auswahl der Befehle, die Sie in den Vorlagendefinitionen verwenden können, beschränkt ist. Insbesondere *Pipes* und *Weiterleitungen* sind nicht ohne Weiteres verfügbar. Das ist der Preis für die Sicherheit. Wenn Sie Shell-Features verwenden wollen, sollten Sie ein sicheres Skript schreiben, um dies in einer kontrollierten Art und Weise zu tun, und dann dieses Skript über die Befehlsspezifikation aufrufen.

Es ist zwar möglich eine Vorlage zu entwerfen, die direkt mit der Shell interagiert, aber da dies bösartigen Anwendern erlauben würde, über clevere Dateinamen und/oder Parameter beliebige Befehle auszuführen, empfehlen wir grundsätzlich, dass Sie sichere Skripte verwenden, die den Systembefehl `execvp` in kontrollierter Form einsetzen. Wir verstehen, dass es innerhalb kontrollierter Umgebungen verlockend sein kann, einfach normale Shell-Skripte zu verwenden. Aber seien Sie sich bewusst, dass Sie, wenn Sie dies tun, eine leicht ausnutzbare Sicherheitslücke in ihr System einfügen. Natürlich werden solche unsicheren Vorlagen niemals in die LyX-Distribution aufgenommen, auch wenn wir immer dazu ermuntern, neue Vorlagen in der Open-Source-Tradition beizusteuern.

Die Einbindung von externem Material ist ein mächtiges Werkzeug, und Sie müssen aufpassen, dass Sie diese Macht nicht unterschätzen und Sicherheitsprobleme schaffen. Ein subtiler Fehler in einer einzigen Zeile in einem unschuldig aussehenden Skript kann die Tür für riesige Sicherheitsprobleme öffnen. Wenn Sie also etwas nicht vollständig verstehen, empfehlen wir, dass Sie einen Sicherheitsexperten oder das LyX-Team konsultieren, sobald sie nicht sicher sind, ob eine externe Vorlage sicher ist oder nicht. Und tun Sie dies, bevor Sie die vorlage in einer unkontrollierten Umgebung verwenden!

A. Liste der Funktionen für die Verwendung in Layout-Dateien

accents	booktabs	feyn	listings	natbib	rsphrase	tfrupee	wasysym
amsbsy	calc	fixltx2e	longtable	nomencl	setspace	tipa	wrapfig
amscd	CJK	float	lyxskak	pdfpages	shapepar	tipx	xargs
amsmath	color	framed	makeidx	pifont	slashed	tone	xcolor
amssymb	covington	graphicx	marvosym	pmboxdraw	soul	txfonts	xy
amstext	csquotes	hhline	mathdesign	polyglossia	splitidx	ulem	yhmath
amsthm	dvipost	hyperref	mathdots	prettyref	subfig	undertilde	
array	endnotes	ifsym	mathrsfs	pxfonts	subscript	units	
ascii	enumitem	ifthen	mhchem	refstyle	tcolorbox	url	
bbding	esint	jurabib	multicol	rotating	textcomp	varioref	
bm	fancybox	latexsym	multirow	rotfloat	textgreek	verbatim	

B. Namen von verfügbaren Farben für die Verwendung in Layout-Dateien

Im Folgenden sind die Standardfarben und die Farben, die man in den LyX-Voreinstellungen festlegen kann, aufgelistet.

B.1. Farbfunktionen

Das folgende sind keine echten Farben, sondern eher Funktionen, die Farbdefinitionen betreffen.

ignore Die Farbe wird ignoriert

inherit Die Farbe wird vom Kontext geerbt

none Keine spezielle Farbe – setzt definierte Farben zurück.

B.2. Statische Farben

Die folgenden Farben sind fest definiert und können nicht verändert werden. Bitte verwenden Sie diese Farben *nicht* in Layoutdefinitionen, denn sie harmonieren nicht gut mit manchen Farbschemata (bspw. dunkle Schemata):

black Schwarz

white Weiß

blue Blau

brown Braun

cyan Cyan

darkgray Dunkelgrau

gray Grau

green Grün

lightgray Hellgrau

lime Neongrün

magenta Magenta

olive Olivgrün

orange Orange

pink Pink

purple Purpur

red Rot

teal Blaugrün

violet Violett

yellow Gelb

B.3. Dynamische Farben

Dies sind die Farben, die in Werkzeuge ▷ Einstellungen spezifischen Elementen zugeordnet werden:

added_space Farbe der Markierung von eingefügten Leerzeichen (in der Änderungsmarkierung)

addedtext Farbe von eingefügtem Text (in der Änderungsmarkierung)

appendix Farbe der Anhangmarkierung

background Generelle Farbe des Hintergrunds

bookmark Farbe der Lesezeichen

bottomarea Farbe des Bereichs nach dem Ende von LyX-Dokumenten

branchlabel Farbe der Marke von Zweigen

buttonbg Farbe des Hintergrund von (Einfügungs-)Knöpfen

buttonframe Farbe des Rahmens von (Einfügungs-)Knöpfen

buttonhoverbg Farbe des Hintergrund von (Einfügungs-)Knöpfen, wenn die Maus darüberfährt

buttonhoverbg_broken Farbe des Hintergrund von (Einfügungs-)Knöpfen für ungültige Verweis-Einfügungen, wenn die Maus darüberfährt

changebar Farbe des Änderungsbalkens

changedtextauthor1 Geänderter Text des 1. Autors

changedtextauthor2 Geänderter Text des 2. Autors

changedtextauthor3 Geänderter Text des 3. Autors

changedtextauthor4 Geänderter Text des 4. Autors

changedtextauthor5 Geänderter Text des 5. Autors

changedtextcomparison Geänderter Text beim Dokumentvergleich (Arbeitsbereich)

collapsible Textfarbe in einklappbaren Einfügungen

collapsibleframe Rahmenfarbe in einklappbaren Einfügungen

command Textfarbe in Befehls-Einfügungen

commandbg Hintergrundfarbe in Befehls-Einfügungen

commandframe Rahmenfarbe in Befehls-Einfügungen

command_broken Textfarbe für ungültige Verweis-Einfügungen (bspw. Literatur- oder Querverweis)

commandbg_broken Hintergrundfarbe für ungültige Einfügungen

commandframe_broken Rahmenfarbe für ungültige Einfügungen

comment Farbe der Marke von Kommentaren

commentbg Hintergrundfarbe von Kommentaren

cursor Farbe des Cursors

deletedtext Farbe von gelöschtem Text (im Änderungsmodus)

deletedtextmodifier Modifizierungsfarbe für gelöschten Text (um die Helligkeit anzupassen)

depthbar Farbe der Einrückungsmarkierung am Rand

eolmarker Farbe der Zeilenendemarkierung

error Farbe der LaTeX-Fehler-Box

footlabel Farbe der Marke für Fußnoten

foreground generelle Vordergrundfarbe

graphicsbg Hintergrundfarbe von Grafik-Einfügungen

greyedoutbg Hintergrundfarbe für Grauschrift-Einfügungen

greyedoutlabel Farbe der Marke für Grauschrift-Einfügungen

greyedouttext Textfarbe für Grauschrift-Einfügungen

indexlabel Farbe der Marke für Stichworteinfügungen

inlinecompletion Farbe der Wortvervollständigung

insetbg Hintergrundfarbe von Einfügungen

insetframe Rahmenfarbe von Einfügungen

language Farbe zur Markierung fremdsprachigen Texts,

latex Textfarbe im LaTeX-Modus

listingsbg Hintergrundfarbe von Programmlisting-Einfügungen

marginlabel Farbe der Marke von Randnotiz-Einfügungen

math Textfarbe von Mathe-Einfügungen

mathbg Hintergrundfarbe von Mathe-Einfügungen

mathcorners Rahmenfarbe von Mathe-Einfügungen, die nicht ediert werden

mathframe Rahmenfarbe von Mathe-Einfügungen, die ediert werden

mathline Linienfarbe im Mathe-Modus

mathmacrobg Hintergrundfarbe der Mathe-Makro-Einfügung

mathmacroblend Ausgeblendete Farbe der Mathe-Makro-Einfügung

mathmacroframe Rahmenfarbe der Mathe-Makro-Einfügung

mathmacrohoverbg Hintergrundfarbe der Mathe-Makro-Einfügung, wenn die Maus darüberfährt

mathmacrolabel Farbe der Marke der Mathe-Makro-Einfügung

mathmacronewarg Farbe für neue Parameter der Mathe-Makro-Definition

mathmacrooldarg Farbe für alte Parameter der Mathe-Makro-Definition

newpage Farbe der Seitenumbruchsmarkierung (neue Seite)

nonunique_inlinecompletion Farbe der Wortvervollständigung (nicht-eindeutiger Teil)

note Farbe der Marke von Notiz-Einfügungen

notebg Hintergrundfarbe von Notiz-Einfügungen

pagebreak Farbe der Seitenumbruchsmarkierung (Seitenumbruch) und von Zeilenumbrüchen

paragraphmarker Farbe für die Absatzmarkierung

phantomtext Textfarbe für Phantom-Einfügungen

preview Farbe für den Hintergrund der eingebetteten Vorschau

previewframe Rahmenfarbe der eingebetteten Vorschau

regexpframe Rahmenfarbe für die Einfügung für reguläre Ausdrücke

scroll Farbe, die anzeigt, dass eine Tabellenspalte gescrollt werden kann

selection Hintergrundfarbe des ausgewählten Texts

selectionmath Vordergrundfarbe von ausgewähltem Text in Mathe-Einfügungen

selectiontext Vordergrundfarbe des ausgewählten Texts

shadedbg Hintergrundfarbe einer schattierten Box

special Textfarbe von Sonderzeichen

tabularline Farbe von Tabellenlinien

tabularonoffline Farbe von Tabellenlinien

textlabel1 Farbe 1 von der Marke von Absatzlayouts

textlabel2 Farbe 2 von der Marke von Absatzlayouts

textlabel3 Farbe 3 von der Marke von Absatzlayouts

urllabel Farbe der Marke von URL-Einfügungen

urltext Textfarbe in URL-Einfügungen