Das ltxdoclet-Paket: Java-Dokumentation (und Quelltexte) in LaTeX. *

Paul Ebermann†

3. Mai 2011

Inhaltsverzeichnis

1 Nutzerdoku

Springe zum Ziel

2 Implementation

81 ⟨*package⟩

2.1 Package-Optionen

Bisher gibt es keine.

*Dieses Dokument gehört zu ltxdoclet v0.0, von 2010/02/14.

†Paul-Ebermann@gmx.de

2.2 Geladene Pakete

Wir laden das color-Paket, um Farben verwenden zu können.

```
%\RequirePackage[dvipsnames]{color}
```

```
\raggedbottom%    Wir wollen nicht den Inhalt der Seiten strecken, damit es passt.
```

2.3 Diverse Makros

\noprint

\noprint ist ein Klon des \@gobble-Makros aus dem LaTeX-Kernel. Es dient dazu, im generierten LaTeX-Quelltext Debug-Informationen auszugeben, ohne dass sie in der Ausgabe auftauchen.

\noprint

```
\newcommand*\noprint[1]{}%
```

\textlangle
\textrangle

Wir stellen hier Defaults für diese Makros zur Verfügung, damit unsere String-Literale mit ⟨⟩ nicht kaputtgehen, falls sie nicht da sind.

\textlangle
\textrangle

```
\DeclareTextCommandDefault{\textlangle}{~\rlap{$\langle$}}
\DeclareTextCommandDefault{\textrangle}{\llap{$\rangle$}~}
```

2.4 Makros aus TeXDoclet

Diese Makros sind direkt kopiert aus dem von TeXDoclet generierten Code, damit die HTML-Umwandlung funktioniert. Ich sollte noch einmal drübergehen und diese Makros anpassen bzw. zumindest ordentlich dokumentieren.

\bl

```
\def\bl{\mbox{}\newline\mbox{}\newline{}}
```

\hide

```
\newcommand{\hide}[2]{%
  \ifthenelse{\equal{#1}{inherited}}%
  {}%
  {}%
}
```

\isep

```
\newcommand{\isep}[0]{%
\setlength{\itemsep}{-.4ex}
}
```

\sld

```
\newcommand{\sld}[0]{%
\setlength{\topsep}{0em}
\setlength{\partopsep}{0em}
\setlength{\parskip}{0em}
\setlength{\parsep}{-1em}
}
```

\headref

```
\newcommand{\headref}[3]{%
\ifthenelse{#1 = 1}{%
\addcontentsline{toc}{section}{\hspace{\qquad}\protect\numberline{}{%
      #3}}%
}{}%
\ifthenelse{#1 = 2}{%
```

```
\addcontentsline{toc}{subsection}{\hspace{\qquad}\protect\numerline{%
      }{#3}}%
}{}%
\ifthenelse{#1 = 3}{%
\addcontentsline{toc}{subsubsection}{\hspace{\qquad}\protect%
      \numerline{}{#3}}%
}{}%
\label{#3}%
\makebox[\textwidth][l]{#2 #3}%
}%
\membername  \newcommand{\membername}[1]{{\it #1}\linebreak}
\divideents  \newcommand{\divideents}[1]{\vskip -1em\indent\rule{2in}{.5mm}}
\refdefined  \newcommand{\refdefined}[1]{
\expandafter\ifx\csname r@#1\endcsname\relax
\relax\else
{$($ in \ref{#1}, page \pageref{#1}$)$}
\fi}
\startsection  \newcommand{\startsection}[4]{
\gdef\classname{#2}
\subsection{\label{#3}{\bf {\sc #1} #2}}{
\rule[1em]{\hsize}{4pt}\vskip -1em
\vskip .1in
#4
}%
}
tsubsubsection  \newcommand{\startsubsubsection}[2]{
\subsubsection{\sc #1}{%
\rule[1em]{\hsize}{2pt}%
#2}
}
\chardef\bslash=`\\
```

2.5 Text ausrichten

\clap Dieses Makro habe ich aus mathtools geklaut. Es ist ein Verwandter der bekannten \llap und \rlap. Es setzt eine horizontale Box mit dem Argument als Inhalt zentriert an der aktuellen Stelle, ohne dass sie Platz verwendet.

```
\clap  \newcommand*\clap[1]{%
  \hb@xt@\z@{\hss#1\hss}
}
```

\clapon {⟨*text1*⟩}{⟨*text2*⟩} Verwandt mit \clap, zentriert dieses Makro ⟨*text1*⟩ nicht mit Breite

```
%   0, sondern über ⟨text2⟩. (D.h. beides wird gesetzt, relativ zueinander zentriert, und
%   das Ergebnis hat die Breite von ⟨text2⟩.)
\clapon 189 \providecommand*{\clapon}[2]{%
190   \setbox\@tempboxa\hbox{#2}%   Wir merken uns ⟨text2⟩ in einer Box.
191   \hbox to\wd\@tempboxa{%   Dann öffnen wir eine Box mit der Breite von
          %   ⟨text2⟩, ...
193     \hss#1\hss}%   ... und setzen darin ⟨text1⟩, mit beidseitig flexiblem Platz
            %   (d.h. zentriert).
195   \kern-\wd\@tempboxa%   Dann gehen wir wieder zurück zum Anfang (mit einem
          %   negativen Abstand).
197   \unhbox\@tempboxa}%   und hier setzen wir ⟨text1⟩, außerhalb seiner Box. (Es
          %   ist zu überlegen, ob statt \unhbox eher \box sinnvoller wäre, denn nun
          %   kann ⟨text2⟩ noch vom Zeilenpasser bearbeitet werden, und damit die
          %   Zentrierung kaputtgehen.)
```

2.6 Literale hervorheben

\markString
\markNumber
\LiteralKeyword

Diese drei Makros sind Deklarationen, die den Bereich bis zum nächsten Gruppenende als Literal markieren. Sie werden von unserem Quelltext-Drucker in der Form `{\markNumber 20}` verwendet: 20

```
\markString 214 \providecommand*{\markString}{%
215   \color{blue}}%   Strings markieren wir blau.
\markNumber 216 \providecommand*{\markNumber}{%
217   \color[named]{ForestGreen}}%   Zahlen sind grün.
\LiteralKeyword 218 \providecommand*{\markLiteralKeyword}{%
219   \color[named]{Brown}}%   Und Schlüsselwortliterale wie null, true, false
          %   sind braun.
```

2.7 sourcecode-Umgebung

sourcecode

Diese Umgebung verwenden wir, um Quelltext zu setzen.

```
\begin{sourcecode}
  \ltdIndent.\textbf{private}
     DocletStart()%
     \ltdSetIndent{4}%
     \ltdSetIndent{0}
  \ltdIndent.\{\ltdSetIndent{%
     4}
  \ltdIndent    .super()~%
     \clap{\textbf{;}}
  \ltdSetIndent{0}\ltdIndent.%
     \}
\end{sourcecode}
```

```
private DocletStart()
{
    super() ;
}
```

Und ein längeres Beispiel:

```
public void writeDoku()
{
    writePackages() ;
    configuration.root.printNotice("ltxdoclet: doku-main.tex wird erstellt
...") ;
    println("   % Damit beim Compilieren nicht bei jedem Fehler angehalten
wird") ;
    println("\\scrollmode") ;
    println() ;
    writePreamble() ;
    println("\\begin{document}") ;
    println() ;
    chapter("Übersicht", false) ;
    ltxwrite(configuration.doctitle+ " besteht aus den folgenden Packages.
Eine") ;
    ltxwrite(" kurze Beschreibung folgt danach.") ;
    section("Package-Liste") ;
    writePackageList() ;
    section("Beschreibung") ;
    writeOverview() ;
    println("\\setcounter{chapter}{0}") ;
    writePackageImports() ;
    println("\\appendix") ;
    println("\\end{document}") ;
    close() ;
    configuration.root.printNotice("ltxdoclet: ... doku-main.tex fertig.") ;
    configuration.root.printNotice("ltxdoclet: warte auf Beendigung der anderen
Dateien ...") ;
    waitForAllThreads() ;
    configuration.root.printNotice("ltxdoclet: Fertig!") ;
}
```

Der Inhalt dieser Umgebung wird von unserem LaTeX-doclet automatisch generiert, indem ein Syntaxbaum des Compilers abgelaufen wird.

Hier die Definition:

sourcecode

```
\newenvironment*{sourcecode}%
{%   Einstellungen am Anfang:
  \par%   zuerst beenden wir den Absatz, falls da einer ist.
  \setlength{\parindent}{0pt}%   Wir wollen keine Paragraphen-Einrückung
      %   hier.
  \ttfamily\small%   Alles soll in einer nichtproportionalen Schrift und etwas
      %   kleiner sein.

  \catcode`\ =\active%   Leerzeichen wollen wir aktiv machen, und ...
  %\expandafter\let\ltd@activeSpace=\ltd@DiscrSpace%   (\let
      =\ltd@DiscrSpace)

  \obeylines%   Zeilenumbrüche sollen bitte behalten werden. (Alternativ
      %   könnten wir unseren Quelltext-Formatierer immer \par ausgeben
      %   lassen, aber so ist es einfacher.)

  \raggedright%   das bringt etwas weniger Overfull hbox-Meldungen, und
      %   erlaubt einen gewissen Zeilenumbruch im Quelltext. Wir müssen uns
      %   da aber noch etwas besseres ausdenken (siehe oben der Versuch mit
      %   den aktiven Leerzeichen.
```

```
    \setlength{\baselineskip}{0.7\baselineskip}%  Den Zeilenabstand setzen
      %  wir auch runter, da wir ja eine kleinere Schriftart nehmen.
}{%  Am Ende der Umgebung muss nicht so viel gemacht werden:
    \par%  wir beenden nur noch den Absatz (und alle Definitionen von oben
      %  werden rückgängig gemahcht, weil die Gruppe endet).
}%
```

In unseren `sourcecode`-Umgebungen soll aktiv sein und wie `\ltdBreak{~}` funktionieren.

```
\catcode`\ =\active%  Wir lassen  aktiv sein, und ...
\def\ltd@activeSpace{ }%  merken uns in diesem Makro ein aktives Leerzeichen.
\catcode`\ =10%  Hier schalten wir wieder zurück.
```

2.8 Einrückungen

`\ltdSetIndent` `{`⟨*anzahl*⟩`}` Setzt die Einrückung für folgende Zeilen auf den angegebenen Wert (in Zeichenbreiten).

```
\newcounter{ltxdoclet@indent}
\providecommand*{\ltdSetIndent}[1]{%
  \setcounter{ltxdoclet@indent}{#1}%
}%
```

```
\newlength{\ltxdoclet@indentlen}
```

`\ltdIndent` ⟨*space*⟩.
Setzt soviel Platz wie nach aktueller Einrückungslänge notwendig.

```
\newcommand*{\ltdIndent}{}%
\def\ltdIndent#1.{%
  \settowidth{\ltxdoclet@indentlen}{~}%
  \multiply\ltxdoclet@indentlen by\value{ltxdoclet@indent}\relax%
  \leavevmode\kern\ltxdoclet@indentlen
}
```

`\ltdBreak` Setzt ein Element mit erlaubtem Umbruch danach.

```
\providecommand*{\ltdBreak}[1][]{%
  #1\discretionary{}{\ltdIndent.}{}%
}
```

`\ltd@DiscrSpace` Dieses Kommando ist ein Leerzeichen, welches zu einer Einrückung umbrechen kann. Die Definition ist analog zu der in gmdoc-enhance.

```
\def\ltd@DiscrSpace{%
  \ifx\protect\@typeset@protect%
    \ifinner%
```

```
365         \space%   Wenn wir in einer inneren Box sind, soll das aktive Leerzeichen
                %     wie ein normales Leerzeichen funktionieren, da es ja hier
                %     sowieso keinen Zeilenumbruch gibt. Oder? Egal, unsere
                %     Code-Kommentare sollten nie im inneren Modus auftauchen.
369     \else%
370       \ifhmode%
371         \unskip%   Hmm, ich bin mir nicht ganz sicher, warum dieser Befehl hier
                %     notwendig ist. Ohne gibt es jedenfalls gelegentlich
                %     Zeilenumbrüche, die nicht an einen dieser \discretionarys
                %     fallen (und entsprechend kein % in der nächsten Zeile haben.)
376         \discretionary{%   Die »Expansion« dieses \discretionary-Objektes:
                %     Falls es hier einen Zeilenumbruch gibt, ist am Ende der Zeile leer.
379         }{\ltdIndent.%   Dafür taucht am Anfang der nächsten Zeile die Box
                %     auf, die durch \ltdIndent gesetzt wird.
381         }{%   Falls kein Umbruch an dieser Stelle erfolgte, ist es ebenfalls leer.
383         }%
384         ~%   Jetzt setzen wir noch ein nicht-umbrechbares Leerzeichen. Falls es
                %     einen Umbruch gab, war der ja davor.
386       \else%
387         \space%   Außerhalb des horizontalen Modus (d.h. im Mathe- und im
                %     vertikalen Modus) hat unsere Spezialfunktion auch nichts zu
                %     suchen.
390       \fi%
391     \fi%
392     \afterfi{%
393       \ltd@gobbleActiveSpaces}%   Am Ende fressen wir alle weiteren direkt
                %     folgenden derartigen aktiven Leerzeichen auf, damit es nicht
                %     mehrere auf einmal gibt.
```

Eigentlich sorgt das obige `\unskip` ja schon dafür, dass mehrere aufeinanderfolgende aktive Leerzeichen (die ja prinzipiell ein `~` als skip einfügen), keine Probleme bereiten, nur das letzte bleibt übrig. Damit ist das `\gmd@gobbleActiveSpaces` eigentlich nicht mehr nötig. Aber wir entlasten damit den Zeilenumbruch-Algorithmus etwas, wenn nicht mehrere `\discretionary`s hintereinander kommen.

```
403   \else%
404     \space%   Falls wir nicht im Typeset-Modus sind, sondern etwa in eine Datei
                %     oder auf den Bildschirm schreiben, soll unser  auch wie ein normales
                %     Leerzeichen wirken. Schließlich findet da kein Zeilenumbruch statt.
407   \fi%
408 }%
```

`\afterfi`

```
412 \def\afterfi#1#2\fi{\fi#1}
```

Hier die Definition unseres Space-Fressers.

`leActiveSpaces`

```
415 \def\ltd@gobbleActiveSpaces{%
```

```
  \expandafter\@ifnextchar\ltd@activeSpace%    Wir überprüfen, ob das
        %    nächste Token ein aktives Leerzeichen ist.
  {%    Falls ja, ...
    \expandafter\ltd@gobbleActiveSpaces\@gobble%    entsorgen wir es mit
          %    \@gobble und rufen uns dann selbst rekursiv auf.
  }{}%    Im anderen Fall machen wir gar nichts, womit die Rekursion beendet ist.
}%
```

2.9 Hyperlinks

`\hypertarget` setzt das Target leider auf die Höhe der Grundlinie, so dass es gerade nicht zu sehen ist, wenn man es von einem `\hyperlink` anspringt.

Hier ein Workaround, der bei PDFs in Adobe Reader und Okular das Richtige macht. (Bei xdvi verschlechtert es die Situation für Links im unteren Teil der Seite, die sind jetzt gerade unterhalb des sichtbaren Bereiches.)

ltdHypertarget

```
\newcommand*\ltdHypertarget[2]{%
  \setbox\@tempboxa\hbox{#2}%
  \@tempdima\ht\@tempboxa
  \raisebox{1.5\@tempdima}[\@tempdima][0pt]%
  {%
    \hypertarget{#1}%
    {%
      \raisebox{-1.5\@tempdima}%
      {%
        \unhbox\@tempboxa%
      }%    (raisebox)
    }%    (hypertarget)
  }%    (raisebox)
}
```

2.10 Ende

```
\endinput
⟨/package⟩
```