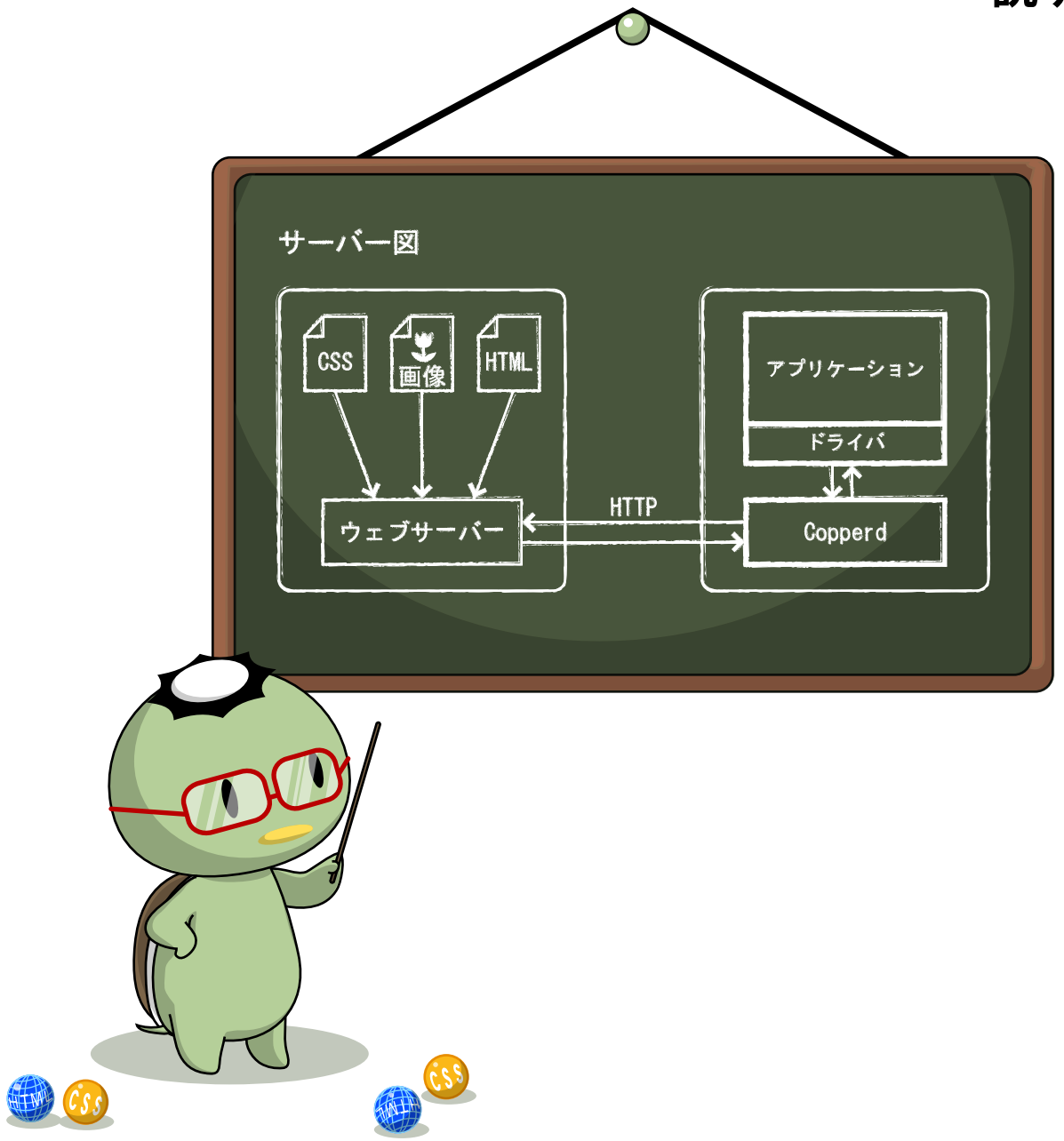
HTML→PDF変換サーバー

Copper PDF

説明書



株式会社 GNN

2009-2-19

NOTICE

Copper PDF 2008 GNN & Co.,Ltd. All rights reserved.

This product includes software developed by Andy Clark.

This product includes software developed at The Apache Software Foundation (http://www.apache.org/).

This software contains code from the World Wide Web Consortium (W3C) for the Document Object Model API (DOM API), SVG Document Type Definition (DTD) and the The Simple API for CSS (SAC API).

目次

1.はじめに 1
1.1 Copper PDFについて 1
1.2 動作環境 2
1.2.1 サーバー 2
1.2.2 プログラミングインターフェース(ドライバ) 2
Javaドライバ 2
Perlドライバ 2
PHPドライバ 2
1.3 機能一覧 3
1.3.1 入力データ形式 3
ドキュメント 3
スタイルシート 3
ベクター画像データ 3
ラスター(ビットマップ)画像データ 3
1.3.2 PDF出力 3
1.3.3 HTTP接続 4
1.3.4 印刷サポート 4
1.3.5 その他の機能 4
1.4 CSSJ 1.x系統からの変更点 5
1.4.1 Copper PDFへの移行 5
1.4.2 主な変更 5
配布パッケージ 5
設定ファイル 5
改ページ可能な場所 5
1.4.3 廃止・置き換えられた機能 5
ライブラリとしてリンクするJavaドライバの廃止 5
'-cssj-regeneratable'の非推奨 6
<cssj:toc>要素の廃止 6
ウィンドウアプリケーションと直接印刷の廃止 6

2.管理者ガイド 7
2.1 セットアップ 7
2.1.1 Java実行環境 のインストール 7
2.1.2 Copper PDF の配布パッケージ 7
2.1.3 Windows 2000/XP/Vista 7
サービスのインストール(Windows 2000 / Windows XP) 8
Windows Vistaでサービスをインストールする場合 8
サービスの動作状態の確認 8
2.1.4 Red Hat Enterprise Linux(RHEL) 8
Copper PDFサーバーの起動・停止 9
2.1.5 Debian 9
Copper PDFサーバーの起動・停止 10
2.1.6 その他のOS 11
2.1.7 ディレクトリ構成 11
アーカイブ内のディレクトリ構成 11
RPMまたはDebianパッケージの構成 11
2.1.8 ライセンスキーファイルの配置 12

2.2 Copper PDFのツール 13
2.2.1 copper コマンドラインアプリケーション 13
形式 13
概要 14
オプション 14
説明 14
入力について 14
出力について 15
javaコマンドで実行する方法 15
2.2.2 copper-webapp ウェブアプリケーション 15
概要 15
javaコマンドで実行する方法 15
2.2.3 copperd ドキュメント変換サーバー 16
形式 16
概要 16
オプション 16
説明 17
javaコマンドで実行する方法 17
2.3 設定ファイル 18
2.3.1 copperdの動作設定(copperd.properties) 18
2.3.2 ログの設定(logging.properties) 19
2.3.3 アクセス制御の設定(access.txt) 20
2.3.4 profilesディレクトリ 20
2.3.5 デフォルトの入出力プロパティ(default.properties) 21
2.3.6 fontsディレクトリ 21
2.4 フォントの設定 22
2.4.1 ドキュメント中でのフォントの利用 22
2.4.2 フォント設定の更新 23
2.4.3 コアフォント 23
2.4.4 CIDフォント 26
埋め込みフォント 27
CID Identity 27
CID Keyedフォント 27
2.4.5 フォントファイル 27
2.4.6 フォント設定ファイル 28
コアフォントのエンコーディング(encodings要素) 28
encodingsに含まれる要素 28
cmapファイル(cmaps要素) 28
cmapsに含まれる要素 29
コアフォント(core-fonts要素) 29
core-fontsに含まれる要素 29
letter-font 29
symbol-font 29
letter-fontおよびsymbol-fontに含まれる要素 30
CIDフォント(cid-fonts要素) 30
cid-fontsに含まれる要素 30
cid-keyed-font 30
font-file 31
font-dir 32
system-font 32
all-system-fonts 33

cid-keyed-font, font-file, system-fontに含まれる要素 33
一般フォントファミリ(generic-fonts要素) 34
generic-fontsに含まれる要素 34

3.開発者ガイド 36
3.1 プログラムインターフェース(ドライバ)の概要 36
3.1.1 サーバーへの接続・認証 37
3.1.2 エラーハンドラ・プログレスリスナの設定 37
エラーハンドラ 37
プログレスリスナ 37
3.1.3 出力先の設定 38
3.1.4 入出力プロパティの設定 38
3.1.5 リソースの送信・アクセス許可 38
リソースをサーバーに送る 39
リソースにサーバーからアクセスする場合 40
URLパターン 40
両者の組み合わせ 41
3.1.6 ドキュメント本体の送信 41
3.1.7 通信の終了 41
3.2 Javaドライバ 42
3.2.1 使用方法 42
3.2.2 APIの概要 42
サーバーへの接続・認証 42
エラーハンドラ・プログレスリスナの設定 42
出力先の設定 42
プロパティの設定 43
リソースの送信・アクセス許可 43
本体の送信 43
通信の終了 43
3.2.3 サンプル 43
3.2.4 サーブレットの作成 44
3.2.5 フィルターを使ったServlet/JSPの変換 45
3.2.6 ソースコード 45
3.3 Perlドライバ 46
3.3.1 使用方法 46
3.3.2 APIの概要 46
サーバーへの接続・認証 46
エラーハンドラ・プログレスリスナの設定 46
出力先の設定 46
プロパティの設定 46
リソースの送信・アクセス許可 46
本体の送信 46
通信の終了 47
3.3.3 サンプル 47
3.4 PHPドライバ 49
3.4.1 使用方法 49
3.4.2 APIの概要 49
サーバーへの接続・認証 49
エラーハンドラ・プログレスリスナの設定 49
出力先の設定 49
プロパティの設定 49

リソースの送信・アクセス許可 49
本体の送信 50
通信の終了 50
3.4.3 サンプル 50
3.4.4 Content-Lengthヘッダの送信 51
3.5 copper Antタスク 52
3.5.1 使用方法 52
3.5.2 copper タスク 52
3.5.3 うまく動かない場合 53
Can't connect to X11... というエラーが表示される 53
ライセンスが認証されない場合 54

4.デザイナーガイド 55
4.1 Copper PDFによるPDFの作成 55
4.1.1 Copper PDFでドキュメントをレイアウトするには 55
4.1.2 入出力プロパティ 55
4.1.3 ページの参照 56
4.1.4 2パス以上の変換処理 56
4.2 CSSによるドキュメントのレイアウト 57
4.2.1 スタイルシートの型 57
4.2.2 メディアタイプ 57
4.2.3 ドキュメント中にスタイルシートを記述する 57
HTMLのstyle属性 57
HTMLのstyle要素 58
jp.cssj.stylesheet処理命令 59
4.2.4 外部のCSSの使用 59
HTMLのlink要素 59
CSSの@import指示子 60
xml-stylesheet処理命令 60
4.2.5 デフォルトのスタイルシート 61
4.3 CSSの拡張機能 62
4.3.1 名前空間 62
4.3.2 ページカウンタ 63
4.4 HTML/XMLの処理 64
4.4.1 ドキュメントの判別 64
4.4.2 文書情報 64
4.4.3 見出し 65
ブックマーク 65
現在ページのセクション 65
4.4.4 目次の生成 65
4.4.5 リンクとフラグメント 66
フラグメント識別子によるリンク 66
-cssj-page-ref関数 67
4.4.6 Internet Explorerとの互換モード 67
4.4.7 XSLT スタイルシートの適用 68
4.4.8 注釈 68
4.4.9 XML中でHTMLの要素と属性を使用する 68
4.5 画像 69
4.5.1 Copper PDFがサポートする画像 69
4.5.2 他の画像形式の利用 69

4.5.3 ラスター(ビットマップ/ピクセルマップ)画像 69
GIF / PNG画像 69
JPEG / JPEG 2000画像 69
その他の画像 69
4.5.4 SVG画像 70
SVG画像ファイルの参照 70
インラインSVG 70
4.5.5 画像を読み込めない場合 71
4.6 ページ処理機能 72
4.6.1 ページのレイアウト 72
4.6.2 片面印刷と両面印刷 74
4.6.3 ページごとに生成されるコンテンツ 74
4.7 改ページ制御 77
4.7.1 用語の定義 77
4.7.2 強制改ページ 77
4.7.3 'orphans'と'windows' 78
'orphans' 79
'windows' 79
'orphans'と'windows'の競合 80
4.7.4 改ページの抑制 81
内部の改ページ抑制 82
前後の改ページ抑制 82
4.7.5 自動改ページ 82
通常のフローのブロック 83
浮動ボックス 83
4.8 テーブル内での改ページ 84
4.8.1 改ページされない場所 84
4.8.2 page-break-XXXの適用 84
4.8.3 テーブル行内部(セル内部)での改ページ 84
4.8.4 デフォルトの改ページ禁止 85
4.9 PDFの機能 86
4.9.1 PDFのバージョンと機能 86
4.9.2 暗号化 86
4.9.3 ファイルの添付 86
4.10 一般的なブラウザとの互換性 87
4.10.1 互換性モードの切り替え 87
4.10.2 標準モードと'msie'モードの違い 87
CSSで数字のクラス名が認識される 87
CSSで':'の代わりに'='が使用できる 87
CSSの色指定で#を省略しても認識される 88
CSSの長さ指定で単位を省略しても認識される 88
フォームの前後にマージンが設定される 89
段落と見出しのマージン上下のマージンをなくす場合がある 89
ボックスの幅または高さに100%が指定された場合、外側のボックスをはみ出さない 89
絶対位置指定ボックス、浮動ボックスの大きさが内容により拡張される 89
通常のフローのボックスが、幅が指定されたボックスにより拡張される 89
幅がautoの固定レイアウトテーブルが固定レイアウトのまま処理される 89
テーブル行に対する'max-height'が適用されない 89
ブロックに対する'line-height'による高さが確保されない 89
全角スペースの間で折り返しされない 90
input要素の高さが強制される 90

マージンの設定に関わらず中央寄せされる 90
匿名のテーブルセルにより補完されない 90
インラインボックスに'width'が適用される 90
'text-align'がブロックの配置にも適用される 90
テーブルカラムのプロパティがセルに継承される 91
'width'が設定されたテーブルセルには{white-space: nowrap;}が適用されない 91
4.10.3 既知の制限事項 91
ＭＳ明朝系フォントの文字幅について 91
サポートしていないCSSプロパティ 91

5.資料集 92
5.1 入出力プロパティ 92
5.1.1 文書中で設定できないプロパティ 102
5.1.2 個人利用ライセンスの制限 103
5.2 エラーハンドラから取得できる情報 104
5.3 CSSプロパティのサポート状況 105
5.4 HTMLの各要素・属性のサポート状況 110
5.5 拡張機能 115
5.5.1 処理命令の拡張 115
5.5.2 CSSプロパティの拡張 116
CSSプロパティ 116
CSS関数 117
5.5.3 XMLの拡張 117
XML要素 118
XML属性 118

1.はじめに

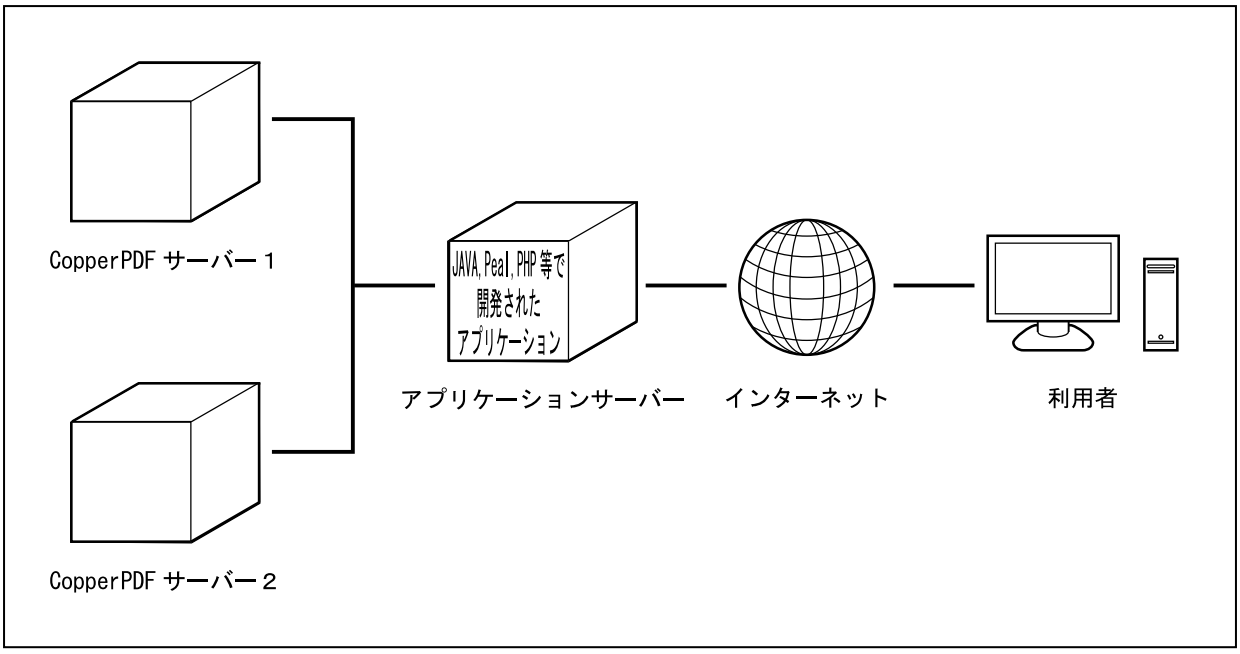
1.1 Copper PDFについて

Copper PDFはHTML+CSSによる文書をサーバー側でPDFに変換するソフトウェアです。一般的なブラウザでページ分割された文書を表示する場合に比べて、高速・高品質でレイアウトできるように独自に機能強化されています。また文書を表示するブラウザ等のクライアントの環境に関係なく同一の表示が保障され、単なるHTML文書では実現できない、PDF特有の機能を利用することができます。

ウェブアプリケーションの開発言語として一般的なJava, Perl, PHPをサポートしており、HTML以外にもXML, SVG, XSLTといったウェブ向けの標準規格に対応しています。動的なウェブページを生成するための技術を、そのまま印刷向け文書を生成するために活用することができます。ECサイト等での帳票類の出力、統計資料やマニュアル等の文書生成、官公庁や銀行等での各種申請書類の出力等、幅広く利用が可能です。

Copper PDFは100% Pure Javaアプリケーションであり、Java実行環境を利用できる、Linux, Windowsをはじめとする様々なOS上で実行可能です。Copper PDFへ各プログラミング言語からネットワークを介してアクセスできるため、付加分散により非常な高負荷が予想されるシステムにも対応可能です。

図 1.1 システムの構成例



1.2 動作環境

1.2.1 サーバー

CopperPDF サーバー (copperd)、 copper (コマンドラインインターフェース)、 copper-webapp(ウェブアプリケーション)および copper Antタスクの実行には最低限以下の環境が必要です。

- Java JRE/JDK(国際化対応版) バージョン1.4.2, 5.0, 6.0
- 128MB以上の空きメモリ
- 50MB以上の空きディスクスペース

以下の環境を推奨します。

- Java JDK バージョン1.4.2, 5.0, 6.0
- 1024MB(1GB)以上の空きメモリ
- 2GB以上の空きディスクスペース

以下のOS上での動作を確認しています。

- Windows 2000/XP/Vista
- Red Hat Enterprise Linux(RHEL) 4/5
- Debian 4.0 "etch"
- Solaris 10
- MacOS X

1.2.2 プログラミングインターフェース(ドライバ)

Copper PDFを利用するアプリケーションの開発には、各プログラミング言語向けのドライバが必要です。各ドライバはCopper PDF本体とは別に配布しています。

Javaドライバ

Java バージョン1.4.2, 5.0, 6.0

Perlドライバ

Perl バージョン5.6.1以降

PHPドライバ

PHP バージョン4.3.0以降またはPHP バージョン5.0.0以降

1.3 機能一覧

1.3.1 入力データ形式

ドキュメント

- HTML/XHTML (64ページ) - HTML文書をレイアウトします。
- XML (64ページ) - XML文書をレイアウトします。

スタイルシート

- CSS (57ページ) - HTMLやXML文書をCSSでスタイル付けします。
- XSLT (60ページ) - XML文書をXSLTで変換してレイアウトします。

ベクター画像データ

- SVG (70ページ) - SVGに対応したドローイングツールで作成した絵・図を埋め込みます。

ラスター(ビットマップ)画像データ

画像を文書中に埋め込む機能と、レイアウト結果を画像として出力する機能があります。

- JPEG (69ページ)
- GIF
- PNG
- その他Java実行環境が対応可能な画像形式 (69ページ) - JPEG2000等、最新の画像形式にも対応可能です。

1.3.2 PDF出力

- PDFバージョン1.2から1.7に対応 (86ページ) - 各バージョンのPDFに対応した出力結果が得られます。
- 添付ファイル (86ページ) - PDFに別のファイルを添付します。
- ブックマーク (65ページ) - HTMLの見出し(H1〜H5)からブックマーク(しおり)を生成します。
- 暗号化 (86ページ) - 40〜128ビットの暗号化に対応しています。
- パスワード・パーミッションの指定 (86ページ) - 暗号化したPDFにパスワードと利用制限をかけることができます。
- 埋め込みフォント (22ページ) - TrueType,OpenType フォントを埋め込みます。
- ハイパーリンク (66ページ) - HTMLのリンク(A)をPDFにも適用します。
- 文書内リンク (66ページ)

1.3.3 HTTP接続

Copper PDFが外部のサーバーからHTTP接続で文書を取得する機能です。

- ヘッダ(Referer, User-Agent等)の変更 (93ページ)
- プロクシを介した接続 (93ページ)
- BASIC認証/Digest認証 (94ページ)
- クッキー (94ページ)

1.3.4 印刷サポート

- トンボの出力 (72ページ) - 断裁して製本するためのトンボを表示します。
- 目次の生成 (65ページ) - HTMLの見出し(H1～H6)から目次を生成します。
- ノンブルの生成 (74ページ) - 欄外にページ番号や、見出しを出力します。
- グレイスケール変換 (95ページ) - カラーの文書をモノトーンに変換します。

1.3.5 その他の機能

- IE互換モード (67ページ) - InternetExplorerの表示に近いレイアウトモードです。
- 出力サイズ制限 (97ページ) - DoS攻撃対策等のため出力サイズを制限できます。
- ページ数制限 (96ページ) - DoS攻撃対策等のため出力ページ数を制限できます。
- 実行経過情報取得 (104ページ) - リアルタイムで文書の変換状況を表示します。

1.4 CSSJ 1.x系統からの変更点

1.4.1 Copper PDFへの移行

CSSJ 1.x向けのJava, PHP, Perlのドライバは、そのままCopper PDFでも利用可能です。以前のドライバを利用したアプリケーションは、接続先をCopper PDFにするだけで、PDFの出力サーバーをCSSJ 1.xからCopper PDFに以降することができます。

ただし、文書のレイアウト機能、サーバー側の設定ファイルについては上位互換性を維持していません。Copper PDFへの移行は、新しい機能を使う場合だけ行ってください。なお、CSSJ 1.x系統のメンテナンス期限は**2013年3月31日まで**です。

1.4.2 主な変更

配布パッケージ

CSSJ 1.xではサーバーとドライバを一緒に配布していましたが、Copper PDFではドライバは別途配布しています。弊社サイト(http://copper-pdf.com/download/) から、利用環境に合うものをダウンロードしてください。

設定ファイル

CSSJ 1.xのresourcesディレクトリがなくなり、全ての設定はconfディレクトリにまとめられました。また、fonts.xml設定ファイルの仕様が大きく変更されたため、CSSJ 1.xのものをCopper PDFで使用することはできません。詳細はフォントの設定 (22ページ)を参照してください。

改ページ可能な場所

Copper PDFでは改ページ制御関連の機能が強化されました。詳細は「改ページ制御」(77ページ)の章を参照してください。

CSSJ 1.xでは常に禁止されていた、テーブルセル内部、浮動ボックス内部での改ページができるようになりました。また、CSSJ 1.xでは`{page-break-inside: avoid;}`で改ページが禁止された領域が ページ高さを超えてしまう場合は内容がページからはみ出していましたが、Copper PDFでは最悪の場合は改ページするように改善されています。

1.4.3 廃止・置き換えられた機能

ライブラリとしてリンクするJavaドライバの廃止

Copper PDFからは、ライブラリとしてリンクさせて動作するJavaドライバはサポートしません。

Copper PDFはサーブレットコンテナ等で広く使用されている各種ライブラリに依存するため、他のアプリケーションから利用する場合に、ライブラリの競合が問題となるこ

とがあることと、近年はマルチコアやハイパースレッディングに対応したCPUが普及したため、ソケット通信により処理を分散したほうが高速に動作するのが実情であり、ライブラリとしてリンクさせることのメリットがないためです。

'-cssj-regeneratable'の非推奨

'-cssj-regeneratable'、'-cssj-regeneratable-clear'は下位互換性のためCopper PDFでも使用できますが、代わりに'-cssj-page-content'、'-cssj-page-content-clear'の使用を推奨します。詳細は「ページごとに生成されるコンテンツ」 (74ページ)を参照してください。

なお、{display: fixed;}が指定されたボックスの 再生成は完全に廃止されました。

<cssj:toc>要素の廃止

<cssj:toc>要素が廃止され、<cssj:make-toc>に置き換えられ、仕様が変更されました。詳細は「目次の生成」 (65ページ)を参照してください。

また、目次の生成のタイミングを指定する、以下の入出力プロパティは廃止されました。

- "processing.make-toc-before"

ウィンドウアプリケーションと直接印刷の廃止

CSSJ無料版として配布していたウィンドウアプリケーションをCopper PDFでは廃止し、PDFを出力するウェブアプリケーションに置き換えました。それに伴い、ウィンドウアプリケーションで表示する際の見栄えを設定する、以下の入出力プロパティを廃止しました。

- "output.j2d.antialiasing"
- "output.j2d.pdf-layout"

2.管理者ガイド

2.1 セットアップ

この章ではCopper PDFのセットアップ(インストール)方法について説明します。

2.1.1 Java実行環境 のインストール

Copper PDFをインストールするためには、Java実行環境(JREまたはJDK)が必要です。Java実行環境は、各OSベンダが配布しているものをインストールするか、http://www.java.com/download/ で配布されているものをインストールしてください。

必要なJava実行環境はバージョン1.4.2以降の国際化対応Java実行環境です。JDK(開発環境およびサーバー用Java VMを含むもの)を推奨します。

2.1.2 Copper PDF の配布パッケージ

Copper PDF本体は、以下のパッケージが配布されています。利用する環境に合ったパッケージをダウンロードしてください。(2.x.xの部分はCopper PDFのバージョンにより異なります)

copper-pdf-2_x_x.zip
ZIPアーカイブ

copper-pdf-2.x.x-x.noarch.rpm
RPMパッケージ

copper-pdf_2.x.x_all.deb
Debianパッケージ

copper-pdf-2.x.x.tar.gz
tar.gzアーカイブ

2.1.3 Windows 2000/XP/Vista

http://copper-pdf.com/download/ で配布されているZIPアーカイブ copper-pdf-2_x_x.zip をダウンロードしてください。

copper-pdf-2_x_x.zipを展開し、CopperPDFディレクトリを適当な場所に配置してください。アンイストールはCopperPDFディレクトリを削除するだけです



CopperPDFディレクトリまでのファイルパスに半角英数字以外の文字(かな、漢字など)が含まれている場合は、次に説明するサービスの起動ができません。Cドライブ直下、あるいはC:\Program Filesなど、半角英数字の文字だけで構成されるファイルパスに格納してください。

サービスのインストール(Windows 2000 / Windows XP)

Windows版にはCopper PDFサーバーをサービスとしてインストールするためのスクリプトが用意されています。スクリプトのインストール方法は、Administrator権限を持つユーザーでInstallService.batを実行するだけです。サービスを削除する場合はRemoveService.batを実行してください。

Windows Vistaでサービスをインストールする場合

アクセサリの「コマンドプロンプト」を管理者で実行し、Copper PDFのディレクトリに移動してからInstallService.batを実行してください。サービスを削除する場合も同様にRemoveService.batを実行してください。

例 2.1 Windows Vistaでサービスをインストールする

```
C:\>cd [Copper PDFのインストールディレクトリ]
C:\[Copper PDFのインストールディレクトリ]>.\InstallService.bat
```

サービスの動作状態の確認

サービスの動作状態を確認するには、copperd.exeの-statusオプションを利用することができます。

例 2.2 サービスの動作状態確認

```
C:\>cd [Copper PDFのインストールディレクトリ]
C:\[Copper PDFのインストールディレクトリ]>copperd.exe -status
* Status Report *
[Summary]
Uptime:0 days 0 h 0 min 3 s
AccessCount:0

[Threads]
Total:10
Busy:0
Free:10
Max:50

[Memory]
Total:127.06MB
Using:3.96MB
Free:123.1MB
Max:1016.13MB
```

2.1.4 Red Hat Enterprise Linux(RHEL)

RHEL向けにはRPMパッケージを配布しています。



RHELでは、Java実行環境としてjava-(バージョン)-sun が必要です。Red Hat NetworksのSupplementaryチャンネルからyumでインストールするか、Supplement CDに収録されているものをインストールしてください。java-(バージョン)-gcj-compatでは動作しません。alternatives --config javaコマンドでjava-(バージョン)-sunに切り替えてください。

copper-pdf-2.x.x-0.noarch.rpmをrpmコマンドでインストールしてください。アンインストールの方法は通常のRPMパッケージの場合と同じです。

例 2.3 RPMパッケージのインストール

```
# sudo rpm -ivh copper-pdf-2.0.0-0.noarch.rpm
```

例 2.4 RPMパッケージのアンインストール

```
# sudo rpm -e copper-pdf
```

Copper PDFサーバーの起動・停止

RHELではchkconfigおよびserviceコマンドでCopper PDFサーバーを管理できます。Copper PDFサーバーのサービス名はcopperdです。インストール直後は、Copper PDFサーバーの自動起動は無効化されています。

例 2.5 copper-pdfの起動

```
# sudo service copperd start
```

例 2.6 copper-pdfの動作状態確認

```
# sudo copperd -status
```

例 2.7 copper-pdfの停止

```
# sudo service copperd stop
```

例 2.8 copper-pdfの再起動

```
# sudo service copperd restart
```

例 2.9 copper-pdfの自動起動の有効化

```
# sudo chkconfig copperd on
```

例 2.10 copper-pdfの自動起動の無効化

```
# sudo chkconfig copperd off
```

2.1.5 Debian

Debian向けにはdebパッケージを配布しています。



Debianでは、Java実行環境としてsun-java5-jdk [non-free]が必要です。kaffe または java -gcj -compat では動作しません。 sun -java5 -jdk をインストールし、update- alternatives - - config java コマンドで java コマンドを /usr/lib/jvm/java-1.5.0-sun/jre/bin/javaに切り替えてください。

他の方法でインストールしたJava実行環境を使う場合は、/etc/profileファイル等で、JAVA_HOME 環境変数にJavaのインストールディレクトリのパスを設定してください。

copper-pdf_2.x.x_all.debをdpkgコマンドでインストールしてください。アンインストールの方法は通常のdebパッケージの場合と同じです。

例 2.11 debパッケージのインストール

```
# sudo dpkg -i copper-pdf_2.0.0_all.deb
```

例 2.12 debパッケージのアンインストール

```
# sudo dpkg -r copper-pdf
```

Copper PDFサーバーの起動・停止

DebianではCopper PDFのサービスは/etc/init.d/copperdとして配置されます。インストール直後は、Copper PDFサーバーの自動起動は無効化された状態です。サービスの管理方法は以下の通り、通常のDebianの手法に従います。

例 2.13 copper-pdfの起動

```
# sudo /etc/init.d/copperd start
```

例 2.14 copper-pdfの動作状態確認

```
# sudo copperd -status
```

例 2.15 copper-pdfの停止

```
# sudo /etc/init.d/copperd stop
```

例 2.16 copper-pdfの再起動

```
# sudo /etc/init.d/copperd restart
```

例 2.17 copper-pdfの自動起動の有効化

```
# sudo update-rc.d /etc/init.d/copperd defaults
```

例 2.18 copper-pdfの自動起動の無効化

```
# sudo update-rc.d /etc/init.d/copperd remove
```

2.1.6 その他のOS

Javaを実行可能なOS全般のために、tar.gzアーカイブのパッケージを配布しています。

copper-pdf-2.x.x.tar.gzを適当なディレクトリに展開してください。

2.1.7 ディレクトリ構成

アーカイブ内のディレクトリ構成

ZIPまたはtar.gzアーカイブを展開すると、copper-pdf-2_x_xというディレクトリができます。各ツールの実行ファイルは、このディレクトリの直下にあります。その他のディレクトリ構成は次の通りです。

図 2.2 ディレクトリ構成

```
copper-pdf-2_x_x
|-- conf        設定ディレクトリ
|   `-- profiles   プロファイル
|       `-- fonts  フォント設定
|-- docs        ドキュメント
|-- extras      アイコン、各プラットフォーム向けのファイル等
|-- legal       付属ライブラリのライセンス文書
|-- jetty       サーブレットコンテナ(copper-webapp用)
|-- lib         ライブラリ
|-- logs        ログディレクトリ
`-- webapp      copper-webapp
```

RPMまたはDebianパッケージの構成

RPMまたはDebianパッケージでインストールした場合は、Copper PDFの各コマンドラインツールは/usr/binと/usr/sbinに配置されます。

また、他のディレクトリの配置場所は次のとおりです。

表 2.1 RPM/Debianのディレクトリ構成

ディレクトリ	配置場所
lib	/usr/share/copper-pdf/lib
docs	/usr/share/doc/copper-pdf
conf	/var/lib/copper-pdf/conf (/etc/copper-pdfにシンボリックリンク)
jetty	/var/lib/copper-pdf/jetty
webapp	/var/lib/copper-pdf/webapp
logs	/var/log/copper-pdf (/var/lib/copper-pdf/logsにシンボリックリンク)

/etc/init.d/copperdはcopperユーザーで実行されます。ファイルの読み書きが行われる /usr/share/copper-pdf/conf/profiles, /var/log/copper-pdf の各ディレクトリはcopperユーザーの所有となります。

2.1.8 ライセンスキーファイルの配置

Copper PDFを使用するためには、ライセンスキー・ファイル(license-key)をconfディレクトリに配置する必要があります。ライセンスキーはhttp://copper-pdf.com/buy/で購入してください。また、機能・期限に制限のある試用ライセンスキーを無料で発行しています。

2.2 Copper PDFのツール

Copper PDFはプログラムとして実行可能な以下のツールで構成されています。それぞれのツールにはLinux/Solaris(拡張子のない実行ファイル), Windows(拡張子が.exeのもの)向けの実行ファイルがあります。

Linux/Solaris環境では、次の環境変数が使用されます。

JAVA_HOME
: 使用するJava実行環境のディレクトリ("/usr/java/jdk1.6.0_05"等)です。設定されていない場合は、パスが通ったjavaコマンドの実行環境が使用されます。

JAVA_OPTS
: javaの実行オプションです。

Linux/Solaris環境でメモリの割り当てを設定する場合はJAVA_OPTS 環境変数を設定してください。また、Java実行環境としてJDKを使用する場合は、-serverオプションを付けることで若干パフォーマンスが向上します。

以下の例ではメモリの割り当てを最大1024MBとし、-serverオプションをつけて各ツールを実行します。

例 2.19 JAVA_OPTS の設定

```
export JAVA_OPTS="-Xmx1024m -server"
```

付属の実行ファイルを使えない環境等で、Javaコマンドで実行する方法については、各ツールの説明を参照してください。

copper
: コマンドラインアプリケーション

copper-webapp
: ウェブアプリケーション

copperd
: ドキュメント変換サーバー

2.2.1 copper コマンドラインアプリケーション



形式

```
copper [-h | -in <入力ファイル> | -uri <入力URI> | -v] [-if <入力形式>] [-out <出力ファイル>] [-p <プロパティ名=値>] [-pf <プロパティファイル>]
```

概要

コマンドラインでドキュメントを変換するアプリケーションです。

オプション

-in, --input-file ファイル
入力ファイルを指定します。

-uri, --input-uri URI
入力ファイルのURIを指定します。

-ie, --input-encoding ファイル
入力ファイルのキャラクタ・エンコーディングを指定します。

-if, --input-format ファイル
入力ファイル形式を指定します。デフォルトはtext/htmlです。

-out, --output-file ファイル
出力先ファイルを指定します。

-p 名前=値
プロパティを指定します。

-pf, --properties-file ファイル
プロパティファイルを指定します。

説明

コマンドラインで手軽に実行できるドキュメント変換ツールです。

ツールの実行ごとにJavaVM とCopper PDFを起動するため、プログラムインターフェースを用いた処理より遅くなります。また、マニュアルの作成等、複数のドキュメントの一括変換を行う場合はcopper Antタスクを使う方が効率的です。

Linux版のcopperおよびcopper-webappは実行ユーザーのホームディレクトリに置かれた設定を用います。詳細はprofilesディレクトリの説明を参照してください。

入力について

- -inオプションが省略された場合は、標準入力を使用します。
- -inオプションと-uriオプションが両方指定された場合、-inで指定されたファイルのデータが使われ、ドキュメントのURIは-uriで指定されたものと見なします。
- -inオプションだけ指定された場合、-inで指定されたファイルのデータが使われ、ドキュメントのURIはそのファイルのシステムURIとします。
- -uriオプションだけが指定された場合、そのURIから取得できるデータが使われ、ドキュメントのURIも同じURIとします。**このとき-ie, -ifオプションは無視されます。**

出力について

- -outオプションが省略された場合は、標準出力を使用します。
- -outオプションが指定された場合は、指定されたファイルに出力します。

javaコマンドで実行する方法

javaコマンドで直接実行するには、libディレクトリ内のboot.jarを-jarオプションにより実行してください。また、以下のシステムプロパティを-Dオプションにより設定してください。

java.awt.headless
Java 1.4.2でディスプレイのない環境で実行する場合は"true"を設定してください。

jp.cssj.boot.lib
libディレクトリのパスを設定してください。

jp.cssj.boot.main
"jp.cssj.driver.cli.Main" を設定してください。

jp.cssj.driver.default
default.propertiesファイルのパスを設定してください。

jp.cssj.copper.config
confディレクトリのパスを設定してください。

2.2.2 copper-webapp ウェブアプリケーション



概要

ウェブアプリケーションとして動作し、ブラウザ上でドキュメントを変換することができます。

プログラムを起動後、ブラウザで http://localhost:8803/ にアクセスすると、Copper PDFのウェブアプリケーションが利用できます。なお、ブラウザが実行できる環境では、プログラムの起動後に自動的にブラウザが開きます。

javaコマンドで実行する方法

javaコマンドで直接実行するには、jettyディレクトリ内のstart.jarを-jarオプションにより実行してください。また、以下のシステムプロパティを-Dオプションにより設定してください。

java.awt.headless
Java 1.4.2でディスプレイのない環境で実行する場合は"true"を設定してください。

jetty.home
: jettyディレクトリのパスを設定してください。

START
: jettyディレクトリ内のstart.configファイルのパスを設定してください。

jp.cssj.driver.default
: default.propertiesファイルのパスを設定してください。

jp.cssj.copper.config
: confディレクトリのパスを設定してください。

jp.cssj.webapp.config
: copper-webappの設定ファイルのパスを設定してください。*[2.0.1]*

2.2.3 copperd ドキュメント変換サーバー



形式

```
copperd [-start|-stop|-kill|-restart|-status] [-cc <制御コマンド>] [-cf <ディレクトリ>] [-cp <ポート番号>] [-h | -v] [-p <ポート番号>] [-passwd <パスワード>] [-user <ユーザー名>] [-userdel <ユーザー名>]
```

概要

ドキュメント変換サーバーの制御と、パスワードの設定ツールです。

オプション

-start
: サーバーを起動します。

-stop
: サーバーを停止します。

-kill
: サーバーを強制停止します。

-status
: 動作中のサーバーの状態を表示します。

-cc, --control-command 制御コマンド文字列
: 制御コマンドを指定します。

-cf, --config ディレクトリ
: 設定ディレクトリを指定します。

-cp, --control-port ポート番号
: 制御ポートを指定します。

-p, --port ポート番号
サービスポートを指定します。

-passwd パスワード
接続パスワードを変更します。

説明

各プログラミング言語向けのインターフェースを利用するには、ドキュメント変換サーバーを常駐させる必要があります。また、ドキュメント変換サーバーへはネットワークを介してアクセスできるため、ドキュメント変換サーバーと、それを利用するアプリケーションを別々のマシン上で動かすことができます。

copperdの設定ファイルはcopperd.propertiesです。サービスポート(-p)、制御ポート(-cp)、制御コマンド(-cc)はデフォルトでは設定ファイルのものが使われ、オプションは設定ファイルによる設定を上書きします。

サーバーの停止(-stop)および状態の取得(-status)には制御用ポートと、制御用コマンドを用います。起動時(-start)の設定と、実行中のサーバーを制御する際の設定は同じである必要があります。

-stopコマンドは、現在接続中のプログラミングインターフェースの処理が全て完了してからサーバーを停止します。-killコマンドは全ての処理を強制的に中断してサーバーを停止します。

copperdにはパスワード設定機能(-passwd)があります。パスワードファイルは暗号化されており、直接編集できないため、このツールを利用して編集してください。

javaコマンドで実行する方法

javaコマンドで直接実行するには、libディレクトリ内のboot.jarを実行してください。また、以下のシステムプロパティを-Dオプションにより設定してください。

java.awt.headless
Java 1.4.2でディスプレイのない環境で実行する場合は"true"を設定してください。

user.home
logsディレクトリが置かれているディレクトリのパスを設定してください。

java.util.logging.config.file
logging.propertiesファイルのパスを設定してください。

jp.cssj.boot.lib
libディレクトリのパスを設定してください。

jp.cssj.boot.main
"jp.cssj.copper.Main"を設定してください。

jp.cssj.driver.default
default.propertiesファイルのパスを設定してください。

jp.cssj.copper.config
confディレクトリのパスを設定してください。

2.3 設定ファイル

confディレクトリの構成は次の通りです。

図 2.3 設定ディレクトリの構成

```
conf
|-- license-key
|-- copperd.properties
|-- logging.properties
|-- password.txt
|-- access.txt
`-- profiles
    |-- default.properties
    `-- fonts
        |-- fonts.xml
        |-- truetype
        |-- warrays
        |-- afms
        |-- cmaps
        `-- encodings
```

confディレクトリの直下には次の設定ファイルが収められています。

license-key
ライセンスキーファイルです。

copperd.properties
copperdの動作設定です。

logging.properties
ログの設定です。

password.txt
パスワードの設定です。編集にはcopperdコマンドを使ってください。

access.txt
アクセス制御設定です。

また、profilesディレクトリ内には次のファイルがあります。

default.properties
デフォルトの入出力プロパティです。

fonts/fonts.xml
使用するフォントの設定です。

2.3.1 copperdの動作設定(copperd.properties)

copperd.propertiesはcopperdサーバーの動作に関する設定です。この設定の変更を反映するにはcopperdの再起動が必要です。

ファイルの形式は Javaのプロパティファイル です。各プロパティの説明は以下の通りです。

表 2.2 copperd.propertiesのプロパティー覧

プロパティ	説明
jp.cssj.cssjd.port	サービスポートです。 ドライバの接続先のポートです。
jp.cssj.cssjd.control-command	制御コマンドです。
jp.cssj.cssjd.control-port	制御用ポートです。
jp.cssj.cssjd.timeout	接続タイムアウト(秒数)です。 プログラムインターフェースとの間で指定された時間データがやりとりされなかった場合、接続を切断します。
jp.cssj.cssjd.minThreads	最小スレッド数です。 プログラムインターフェースの接続を待ち受けるために準備しておく最小限のスレッド数です。高負荷が予想される状況では大きめの値を設定しておくと処理が効率化しますが、使用するメモリは増大します。
jp.cssj.cssjd.maxThreads	最大スレッド数です。 copperdが同時に処理できるプログラムインターフェースからの接続の最大数です。高負荷が予想され、なおかつシステムのリソースに余裕がある場合は大きな値を設定するのが有効ですが、リソースに余裕がない環境では、大きな値を設定することで逆に非効率になることがあります。
jp.cssj.cssjd.backlog	接続のバックログ数です。 未処理の接続がこの数を超えた場合、処理に空きができるまで以降の接続は拒否します。

2.3.2 ログの設定(logging.properties)

logging.propertiesはcopperdのログ設定ファイルです。この設定の変更を反映するにはcopperdの再起動が必要です。ログ設定ファイルは java.util.logging のログプロパティファイル の形式で記述してください。

例 2.20 ログの設定例

```
# Logging levels by categories.
jp.cssj.handlers = jp.cssj.logging.FileHandler, \
    java.util.logging.ConsoleHandler
jp.cssj.level = INFO

# File
java.util.logging.FileHandler.level = INFO
java.util.logging.FileHandler.formatter =
java.util.logging.SimpleFormatter
java.util.logging.FileHandler.encoding = UTF-8

# Console
java.util.logging.ConsoleHandler.level = OFF
java.util.logging.ConsoleHandler.formatter =
java.util.logging.SimpleFormatter
```

Copper PDF には日付でログをローテーションするログハンドラ(jp.cssj.logging.FileHandler) が用意されています。ハンドラのプロパティは次のとおりです。

表 2.3 jp.cssj.logging.FileHandler のプロパティ

名前	デフォルト	説明
directory	copperdの標準のlogsディレクトリ	ログの出力先ディレクトリ
prefix	copperd.	ログファイル名の先頭の文字列
suffix	.log	ログファイル名の末尾の文字列
level	ALL	ログレベル
filter	N/A	ログのフィルタ
formatter	java.util.logging.SimpleFormatter	ログのフォーマッタ
encoding	UTF-8	ログファイルのキャラクタエンコーディング

2.3.3 アクセス制御の設定(access.txt)

access.txtはIPアドレスでアクセスできるクライアントを制限する設定です。このファイルの変更は自動的に検出されるため、変更後にcopperdを再起動する必要はありません。各行のルールを

`[allow|deny] IPアドレス`

という形式で記述します。IPアドレスはIPv4, IPv6のどちらの形式も使用可能です。

allowは指定されたアドレスのアクセスを許可し、denyはアクセスを拒否するルールです。クライアントからのアクセスがあった場合は、先頭から順に検索し、最初に一致したルールが適用されます。

IPアドレスを * にすると、全てのクライアントにルールを適用します。

例 2.21 アクセス制御の設定例

```
allow   127.0.0.1
deny    *
```

2.3.4 profilesディレクトリ

profilesディレクトリ内の設定は、Copper PDFの出力結果に関係するものです。デフォルトの入出力プロパティやフォントの情報が含まれます。

Linux版のcopperdおよび、Windows版の各種ツールはconfディレクトリ内のprofilesディレクトリの設定を使用しますが、Linux版の copper copper-webapp は初回実行時に実行ユーザーのホームディレクトリに.copperという名前のディレクトリをつくり、そこにprofilesディレクトリを作成します。copper, copper-webappを使用する各ユーザーはホームディレクトリの設定を編集する必要があります。

copper-webappの設定ファイルは、初回実行時に /WEB-INF/client.propertiesから各ユーザーの.copper/webapp.proprtiesにコピーされます。*[2.0.1]*

2.3.5 デフォルトの入出力プロパティ(default.properties)

default.propertiesはデフォルトの入出力プロパティを設定するものです。この設定の変更を反映するためにcopperdを再起動する必要はありません。

入出力プロパティはcopperコマンドの引数やcopper-webappの設定ウィンドウ、プログラムインターフェースによって変更できますが、default.propertiesに書いておくことにより、あらかじめ入出力プロパティを設定した状態にすることができます。

ファイルの形式は Javaのプロパティファイル です。各プロパティは入出力プロパティの節で解説されていますが、１つだけsystem.fontsという特別なプロパティがあります。system.fontsは後で次に説明するフォント設定ファイルをdefault.propertiesファイルからの相対パスで指定したものです。

2.3.6 fontsディレクトリ

fontsディレクトリにはドキュメント中で使用することができるフォントの詳細情報が含まれます。また、出荷時の設定ではfonts/truetypeディレクトリ内のフォントファイルを利用するようになっています。使用したいフォントをfonts/truetypeディレクトリにコピー(あるいはシンボリックリンク)してください。ディレクトリ名はtruetypeですが、Copper PDF 2.0.3からOpenType CFF/Type2(.otf)フォントがサポートされました。

フォント関連の設定方法の詳細は次の章で解説します。

2.4 フォントの設定

Copper PDFは外部のフォントを使用するPDF、あるいはフォント情報を埋め込んだPDFを出力することができます。フォント設定ファイル(fonts.xml)に、ドキュメント中で利用できるフォントの設定が記述されています。

2.4.1 ドキュメント中でのフォントの利用

ドキュメント中で使用するフォントを特定するCSSプロパティは 'font-family'(ファミリ名), 'font-weight'(太さ), 'font-size'(大きさ), 'font-style'(スタイル) の４つです('font'プロパティでまとめて指定することもできます)。指定されたファミリ名, 太さ, スタイルから, 記述された文字を表記できる最も近いフォントが選択され, 使用されます。

該当する太さのフォントが見つからない場合は, 機械的に文字の輪郭を拡張して太いフォントがつくられます (逆に指定した太さより細いフォントがない場合, 機械的に細いフォントをつくることはできません)。

また, 'font-style'にitalicまたはobliqueが指定されたとき、斜体スタイルのフォントが見つからない場合は, フォントを機械的に傾けます。同時に'font-weight'が指定されている場合は、以下の表のとおり, italicとobliqueの場合で選択されるフォントが異なることがあります。

表 2.4 フォントスタイルの選択

条件	italic指定の場合	oblique指定の場合
太さとスタイルが一致するフォントがある	太さとスタイルが一致するフォントを使用する	太さとスタイルが一致するフォントを使用する
太さだけ一致するフォントがある	太さが一致するフォントを傾けて使用する	太さが一致するフォントを傾けて使用する
スタイルだけ一致するフォントがある	スタイルが一致するフォントを太くして使用する	スタイルが一致するフォントを太くして使用する
太さだけ一致するフォントとスタイルだけ一致するフォントがある	スタイルが一致するフォントを太くして使用する	太さが一致するフォントを傾けて使用する
太さもスタイルも一致するフォントがない	他の条件が一致するフォントを太くして傾けて使用する	他の条件が一致するフォントを太くして傾けて使用する

ドキュメント中で指定されたフォントが見つからない場合は、output.default-font-familyにより設定されたフォントが使われますが, どうしても表示できない文字がドキュメント中に記述された場合は、代わりに１６進ユニコードを表す組文字(　　のような文字)が表示されます。

Copper PDFがサポートするフォントには コアフォント、埋め込みフォント、CID Identityフォント、CID Keyedフォント の４種類があります。コアフォントは常に利用可能ですが, 埋め込みフォント, CID Identityフォント, CID Keyedフォントのどれを利用するかは output.pdf.fonts.policy によって選択可能です。また, Copper PDFの拡張CSSプロパティ -cssj-font-policy によってドキュメント中で指定することもできます。

2.4.2 フォント設定の更新

Copper PDFは実行中にfonts.xmlの更新を検出して再読み込みします。ただし、システムへのフォントのインストールや、truetypeディレクトリ内の変更を自動検出しません。フォント設定を再読み込みさせるためには、fonts.xmlファイルを再保存するか、touchコマンド等でファイルのタイムスタンプを更新してください。

2.4.3 コアフォント

コアフォントは、ほとんどのPDF表示環境が標準的にサポートしているフォントで、フォントの埋め込みをせずに表示することができます。コアフォントはさらにletter(欧文文字・記号だけで構成されるもの)、symbol(欧文文字・記号以外の文字を含むもの)の2種類に分けられます。

以下はコアフォントの一覧です。

表 2.5 コアフォント

正式名称	略称	ファミリ名	太字	斜体	種類
Times Roman	Times-Roman	Times			letter
Times Bold	Times-Bold	Times	✔		letter
Times Italic	Times-Italic	Times		✔	letter
Times Bold Itatdc	Times-BoldItalic	Times	✔	✔	letter
Helvetica	Helvetica	Helvetica			letter
Helvetica Bold	Helvetica-Bold	Helvetica	✔		letter
Helvetica Oblique	Helvetica-Oblique	Helvetica		✔	letter
Helvetica Bold Oblique	Helvetica-BoldOblique	Helvetica	✔	✔	letter
Courier	Courier	Courier			letter
Courier Bold	Courier-Bold	Courier	✔		letter
Courier Oblique	Courier-Oblique	Courier		✔	letter
Courier Bold Oblique	Courier-BoldOblique	Courier	✔	✔	letter
Symbol	Symbol	Symbol			symbol
ITC Zapf Dingbats	ZapfDingbats	ZapfDingbats			symbol

CSSの`'font-family'`プロパティによるフォントの指定は、正式名称、略称、ファミリ名のいずれでも可能です。ファミリ名を使用した場合は、`'font-style'`, `'font-weight'` プロパティの指定により、同じファミリ名を持つフォントのうち、スタイルと太さが最も一致するフォントが自動的に選択されます。

コアフォントのフォントメトリックス情報(文字の幅などの情報)がAFM(Adobe Font Metrics)ファイルとして、fonts/afmsディレクトリに収められています。これらのファイルをユーザーが変更する必要はありません。

各文字とPDF内で使用される文字コードとの対応表が fonts/encodingsディレクトリに収められています。UNICODE.txtはletterフォントの文字名とユニコードとの対応表です。symbol.txtとzdingbat.txtはそれぞれSymbol, ITC Zapf Dingbatsのためのユニコード対応表です。これらのファイルをユーザーが変更する必要はありません。

以下はletterフォントで使用できる文字セット(WinAnsiEncodingエンコーディング)の文字一覧表です。それぞれの文字は8ビット(16進数で2桁)のユニコードに対応しており、縦が上位桁, 横が下位桁です。ただし、80から9Fまでのコードで空いている部分は対応する文字がないことを表し、下に4桁の16進数字がある文字には別のコードがあります。例えばダブルダガー(‡)を表示する場合はドキュメント中で‡または&#0x2021と表記してください。

表 2.6 WinAnsiEncodingの文字一覧

-	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8	€ 20AC		‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	Š 0160	‹ 2039	Œ 0152		Ž 017D	
9		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	˜ 02DC	™ 2122	š 0161	› 203A	œ 0153		ž 017E	Ÿ 0178
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

以下はSymbolで使用できる文字の一覧です。文字の下の数字は16進ユニコードです。

表 2.7 Symbolの文字一覧

0020 00A0	! 0021	∀ 2200	# 0023	∃ 2203	% 0025	& 0026	∍ 220B	(0028	) 0029	* 2217	+ 002B	, 002C	− 2212	. 002E	/ 002F
0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
≅ 2245	Α 0391	Β 0392	Χ 03A7	Δ 0394 2206	Ε 0395	Φ 03A6	Γ 0393	Η 0397	Ι 0399	ϑ 03D1	Κ 039A	Λ 039B	Μ 039C	Ν 039D	Ο 039F
Π 03A0	Θ 0398	Ρ 03A1	Σ 03A3	Τ 03A4	Υ 03A5	ς 03C2	Ω 03A9 2126	Ξ 039E	Ψ 03A8	Ζ 0396	[005B	∴ 2234	] 005D	⊥ 22A5	¯ 005F
‾ F8E5	α 03B1	β 03B2	χ 03C7	δ 03B4	ε 03B5	φ 03C6	γ 03B3	η 03B7	ι 03B9	φ 03D5	κ 03BA	λ 03BB	μ 00B5 03BC	ν 03BD	ο 03BF
π 03C0	θ 03B8	ρ 03C1	σ 03C3	τ 03C4	υ 03C5	ϖ 03D6	ω 03C9	ξ 03BE	ψ 03C8	ζ 03B6	{ 007B	| 007C	} 007D	~ 223C	
€ 20AC	ϒ 03D2	′ 2032	≤ 2264	⁄ 2044 2215	∞ 221E	ƒ 0192	♣ 2663	♦ 2666	♥ 2665	♠ 2660	↔ 2194	← 2190	↑ 2191	→ 2192	↓ 2193
° 00B0	± 00B1	″ 2033	≥ 2265	× 00D7	∝ 221D	∂ 2202	• 2022	÷ 00F7	≠ 2260	≡ 2261	≈ 2248	… 2026	| F8E6	— F8E7	↵ 21B5
ℵ 2135	ℑ 2111	ℜ 211C	℘ 2118	⊗ 2297	⊕ 2295	∅ 2205	∩ 2229	∪ 222A	⊃ 2283	⊇ 2287	⊄ 2284	⊂ 2282	⊆ 2286	∈ 2208	∉ 2209
∠ 2220	∇ 2207	® F6DA	© F6D9	™ F6DB	∏ 220F	√ 221A	⋅ 22C5	¬ 00AC	∧ 2227	∨ 2228	⇔ 21D4	⇐ 21D0	⇑ 21D1	⇒ 21D2	⇓ 21D3
◊ 25CA	〈 2329	® F8E8	© F8E9	™ F8EA	∑ 2211	⎛ F8EB	⎜ F8EC	⎝ F8ED	⎡ F8EE	⎢ F8EF	⎣ F8F0	⎧ F8F1	⎨ F8F2	⎩ F8F3	⎪ F8F4
	〉 232A	∫ 222B	⌠ 2320	⎮ F8F5	⌡ 2321	⎞ F8F6	⎟ F8F7	⎠ F8F8	⎤ F8F9	⎥ F8FA	⎦ F8FB	⎫ F8FC	⎬ F8FD	⎭ F8FE	

以下はZapfDingbatsで使用できる文字の一覧です。文字の下の数字は16進ユニコードです。

表 2.8 ZapfDingbatsの文字一覧

0020 00A0	2701	2702	2703	2704	260E	2706	2707	2708	2709	261B	261E	270C	270D	270E	270F
2710	2711	2712	2713	2714	2715	2716	2717	2718	2719	271A	271B	271C	271D	271E	271F
2720	2721	2722	2723	2724	2725	2726	2727	2605	2729	272A	272B	272C	272D	272E	272F
2730	2731	2732	2733	2734	2735	2736	2737	2738	2739	273A	273B	273C	273D	273E	273F
2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	274A	274B	25CF	274D	25A0	274F
2750	2751	2752	25B2	25BC	25C6	2756	25D7	2758	2759	275A	275B	275C	275D	275E	
F8D7	F8D8	F8D9	F8DA	F8DB	F8DC	F8DD	F8DE	F8DF	F8E0	F8E1	F8E2	F8E3	F8E4		
	2761	2762	2763	2764	2765	2766	2767	2663	2666	2665	2660	2460	2461	2462	2463
2464	2465	2466	2467	2468	2469	2776	2777	2778	2779	277A	277B	277C	277D	277E	277F
2780	2781	2782	2783	2784	2785	2786	2787	2788	2789	278A	278B	278C	278D	278E	278F
2790	2791	2792	2793	2794	2192	2194	2195	2798	2799	279A	279B	279C	279D	279E	279F
27A0	27A1	27A2	27A3	27A4	27A5	27A6	27A7	27A8	27A9	27AA	27AB	27AC	27AD	27AE	27AF
	27B1	27B2	27B3	27B4	27B5	27B6	27B7	27B8	27B9	27BA	27BB	27BC	27BD	27BE	

2.4.4 CIDフォント

欧文以外の文字をPDFに含める場合は、CIDフォントを用います。フォントを埋め込む方法と、フォントを埋め込まない方法があり、フォントを埋め込まない方法には、さらにCID IdentityとCID Keyedフォントの２種類の方法があります。どの種類のフォントを利用するかは、入出力プロパティoutput.pdf.fonts.policyの設定によります。

埋め込みフォントまたはCID Identityを使用する場合、Copper PDFにフォントをインストールする必要があります。Copper PDFの出荷時の状態では、truetypeディレクトリにフォントファイルを配置すると フォントが利用可能になるようにfonts.xmlが設定されています。

埋め込みフォント

PDFにフォントを字体のデータを埋め込む方式で、環境に関係なく、確実に同じ字体で文字が表示されることが保証されます。

Copper PDFはフォントの埋め込みをサポートしています。フォントを埋め込む場合、文書で使用する文字のフォントだけをPDFに含めるため、ファイルサイズは最小限に抑えられますが、編集や加工には適しません。

表示や印刷の見栄えに厳密さが求められる場合や、広く配布する文書、あるいは長期保存する文書に適しています。

CID Identity

フォントの埋め込みをせず、グリフID(フォントファイルに含まれる文字のインデックス)をそのままPDF内に記述する方式です。Copper PDFの動作環境と、PDFを表示する環境に同一のフォントがインストールされている必要があります。グリフIDは特定のフォントファイルに依存するため、PDFを表示する環境にインストールされたフォントファイルが異なれば、似たような書体のフォントであっても文字化けが発生するか、全く表示できなくなります。

同一のマシン上や、同じ組織内での編集や加工、印刷を目的とする文書に適しています。フォントの埋め込みをする場合にくらべて、出力されるPDFのサイズは小さくなります。

CID Keyedフォント

CID Identityフォント同様にフォントの埋め込みをしませんが、Adobe社により公開されているコード体系(CMap)を使用します。

PDFを表示する環境で利用可能な、書体の近いフォントが自動的に選ばれるので、特定のフォントファイルに依存することはありません。ただし、使用できる文字はAdobe社が提供するCMapファイルで定義された文字に限られ、全ての環境でPDFを表示できることが保障されるものではありません。

特定の言語環境のWindowsやMacOS、特定のバージョン以降のAdobe Readerなど、表示・編集・加工する環境が比較的限られ、表示や印刷の見栄えに厳密さが求められない文書には有効です。

2.4.5 フォントファイル

埋め込みフォント、CID Identityフォントを使用する場合、あるいはCID Keyedフォントのためにフォントの幅情報を抽出するためには、Copper PDFが動作する環境にフォントファイルがインストールされている必要があります。

Copper PDFがサポートするフォントファイルは以下のフォーマットです。

TrueType "OpenType(TrueType flavor)"
一般的にTrueTypeと呼ばれるフォントファイルです。.ttfまたは.ttc(複数のフォントを含むもの)という拡張子のファイルとして配布されています。

OpenType CFF/Type2
一般的に単にOpenTypeと呼ばれる、CFF/Type2形式のOpenTypeフォントファイルです。.otfという拡張子のファイルとして配布されています。Copper PDF 2.0.3からサポートされました。

またCopper PDFはJava実行環境によりサポートされるフォントを埋め込むことができます。SunのJava実行環境はTrueTypeに加えてType1フォント(.pfa, .pfb)、F3フォント(.f3b)をサポートしています。

2.4.6 フォント設定ファイル

PDF出力に使用するフォントの情報はフォント設定ファイル(fonts.xml) に記述してください。フォント設定ファイルはXML形式で、ルート要素はfontsです。フォント設定ファイルには各種ファイルのファイルパスの情報も含まれており、ファイルパスはフォント設定ファイルからの相対パスとなります。

fonts要素には次の要素が含まれています。

コアフォントのエンコーディング(encodings要素)

コアフォントのエンコーディング情報のファイルを設定します。通常は編集する必要はありません。

letterフォントを使用する場合、使用できる文字セットにはStandardEncoding, MacRomanEncoding, WinAnsiEncodingの3種類があります。Copper PDFの出荷時にはWinAnsiEncodingが設定されています。core-fonts要素のencoding属性 の指定を変更することで切り替えることができます。

encodingsに含まれる要素

表 2.9 encoding要素

属性	必須	説明
src	✔	グリフコードとグリフ名の対応を記述したファイルです。

cmapファイル(cmaps要素)

等幅和文フォントのエンコーディング情報のファイルを設定します。通常は編集する必要はありません。

cmapsに含まれる要素

表 2.10 cmap要素

属性	必須	説明
src	✔	Adobe Japan 1-4コードと文字コードの対応を記述したファイルです。
java-encoding	✔	文字コードのJavaエンコーディング名です。

コアフォント(core-fonts要素)

コアフォントの設定です。通常は編集する必要はありませんが、使用するエンコーディングを変更したり、ドキュメントから参照する際の別名を追加することができます。

表 2.11 core-fonts要素

属性	必須	説明
unicode-src	✔	ユニコードと文字名の対応を記述したファイルです。
encoding	✔	エンコーディング名です。encodingsで設定されたものの中から選択できます。

core-fontsに含まれる要素

core-fonts要素にはletter-font, symbol-fontのいずれかを含むことができます。

letter-font

letter-font要素は通常の欧文フォントの設定です。

表 2.12 letter-font要素

属性	必須	説明
src	✔	AFMファイルです。
encoding		core-fontsのencoding属性を上書きします。

symbol-font

symbol-font要素は記号フォントの設定です。

表 2.13 symbol-font要素

属性	必須	説明
src	✔	AFMファイルです。
encoding-src	✔	ユニコードとグリフコードの対応を記述したファイルです。

letter-fontおよびsymbol-fontに含まれる要素

core-fontおよびsymbol-font内にalias要素を追加することで、ドキュメント中の'font-family'で参照することができる別名が追加されます。

表 2.14 alias要素

属性	必須	説明
name	✔	フォントの別名です。

CIDフォント(cid-fonts要素)

cid-fontsに含まれる要素

cid-fonts要素にはcid-keyed-font, font-file, font-dir, system-font, all-system-fontsのいずれかを含むことができます。

cid-keyed-font

cid-keyed-font要素はフォントファイルを使用する代わりに、フォントの幅情報を記述したファイルを利用してCID-Keyedフォントを定義するものです。

表 2.15 cid-keyed-font要素

属性	必須	説明
name	✔	フォント名です。
italic		フォントを斜体にする場合はtrue、そうでない場合はfalseを設定してください。
weight		フォントの太さです。100から900まで100刻みの値で設定してください。
panose		PDFのFontDescripterのPanoseフィールドに対応する値です。クラスID、サブクラスと10のPANOSE-1コードをスペース区切りで記述した12の数字から構成されます。
cmap	✔	横書きのCMap名です。
vcmap		縦書きのCMap名です。
warray	✔	フォントの幅情報のファイルです。

フォントの幅情報のファイルはwarraysディレクトリに等幅フォント用の既定のものが用意されています。

また、手持ちのフォントから幅情報を抽出するツールが用意されています。以下のように、javaコマンドで直接jp.cssj.sakae.pdf.tools.WArrayTool を実行してください。クラスパスとクラス名に続く引数はそれぞれ、cmapファイル、Javaエンコーディング名、フォントファイルです。複数のフォントを含むファイル(.ttc)の場合、さらにフォントの番号を続けることができます。

例 2.22 幅情報の抽出

```
java -cp lib/cssj-sakae.jar:lib/port4-fragmental.jar \
    jp.cssj.sakae.pdf.tools.WArrayTool \
    conf/profiles/fonts/cmaps/UniJIS-UTF16-H \
    UTF-16BE \
    /usr/share/fonts/truetype/kochi/kochi-mincho.ttf \
    > conf/profiles/fonts/warrays/kochi-mincho.txt
```

font-file

font-file要素はフォントファイルを直接指定します。

表 2.16 font-file要素

属性	必須	説明
name		フォント名フォントデータから取得されますが、ここで上書きすることもできます。
src	✔	フォントファイルです。
index		複数のフォントを含むTTCファイルの中で、使用するフォントの番号です。省略した場合は0です。TTCファイルでない場合は無視されます。
types	✔	フォントのタイプをスペース区切りで記述します。値は次のいずれかです。 **embedded** 埋め込みフォント **cid-identity** CID Identityフォント **cid-keyed** CID-Keyedフォント
italic		フォントが斜体かどうかはフォントデータから取得されますが、ここで上書きすることもできます。斜体にする場合はtrue、そうでない場合はfalseを設定してください。
weight		フォントの太さはフォントデータから取得されますが、ここで上書きすることもできます。100から900まで100刻みの値で設定してください。
cmap	✔ (type="cid-keyed"の場合)	横書きのCMap名です。
vcmap		縦書きのCMap名です。

font-dir

font-dir要素は指定したディレクトリに存在するフォントファイルを直接まとめて読み込みます。

表 2.17 font-dir要素

属性	必須	説明
dir	✔	フォントファイルが格納されるディレクトリです。
types	✔	フォントのタイプをスペース区切りで記述します。値は次のいずれかです。 **embedded** 埋め込みフォント **cid-identity** CID Identityフォント

system-font

system-fontはJava実行環境を利用してフォントを読み込みます。OSやウィンドウシステムにインストールされたフォントを名前で指定できますが、縦書きなどフォントの一部の機能に制約があります。

表 2.18 system-font要素

属性	必須	説明
name		フォント名フォントデータから取得されますが、ここで上書きすることもできます。
src	✔	フォント名です。
types	✔	フォントのタイプをスペース区切りで記述します。値は次のいずれかです。 **embedded** 埋め込みフォント **cid-identity** CID Identityフォント **cid-keyed** CID-Keyedフォント
italic		フォントが斜体かどうかはフォントデータから取得されますが、ここで上書きすることもできます。斜体にする場合はtrue、そうでない場合はfalseを設定してください。

属性	必須	説明
weight		フォントの太さはフォントデータから取得されますが、ここで上書きすることもできます。100から900まで100刻みの値で設定してください。
cmap	✔(type="cid-keyed"の場合)	横書きのCMap名です。
vcmap		縦書きのCMap名です。

all-system-fonts

all-system-fontsはJava実行環境が利用可能なフォントを全て読み込みます。

表 2.19 all-system-fonts要素

属性	必須	説明
types	✔	フォントのタイプをスペース区切りで記述します。値は次のいずれかです。 **embedded** 埋め込みフォント **cid-identity** CID Identityフォント

cid-keyed-font, font-file, system-fontに含まれる要素

フォントの名前はフォントデータから取得されますが、cid-keyed-font, font-file, system-fontにalias要素を追加することにより、さらにフォントの別名を追加できます。記述方法はcore-font, symbol-fontのalias と同じです。

Copper PDF 2.0.3からinclude, exclude要素により、フォントが使用される文字範囲を指定できるようになりました。includeにより、フォントが使用される文字範囲を明示することができ、excludeにより除外する文字範囲を明示することができます。include, excludeの記述がない場合は、利用可能な全ての文字でフォントが利用されます。

表 2.20 include要素

属性	必須	説明
unicode-range	✔	カンマで区切ったユニコード範囲(詳細は後述)。

表 2.21 exclude要素

属性	必須	説明
unicode-range	✔	カンマで区切ったユニコード範囲(詳細は後述)。

unicode-rangeはU+に続く16進数によるユニコードとハイフンを用います。例えば、日本語のかなを含める文字範囲の指定は次の通りです。

例 2.23 ハイフンによる文字範囲

```
<include unicode-range="U+3030-30FF" />
```

また下位の桁をワイルドカードに置き換えることもできます。以下の記述は U+1F00-1FFFと書くのと等価です。

例 2.24 ワイルドカードによる文字範囲

```
<include unicode-range="U+1F??" />
```

複数の文字範囲はカンマで区切ってください。

例 2.25 複数の文字範囲

```
<include unicode-range="U+3030-30FF,U+1F??" />
```

一般フォントファミリ(generic-fonts要素)

font-familyで指定できるserif,sans-serif,monospace,fantasy,cursive という 5種類の一般フォント・ファミリに対応するフォントを指定するものです。

generic-fontsに含まれる要素

generic-fontsにはCSSの一般フォントファミリ名に対応する5つの名前の要素、serif, sans-serif, monospace, fantasy, cursiveが含まれます。

表 2.22 serif要素

属性	必須	説明
font-family	✔	カンマで区切ったフォント名を優先順位の高いものから順に記述します。

表 2.23 sans-serif要素

属性	必須	説明
font-family	✔	カンマで区切ったフォント名を優先順位の高いものから順に記述します。

表 2.24 monospace要素

属性	必須	説明
font-family	✔	カンマで区切ったフォント名を優先順位の高いものから順に記述します。

表 2.25 fantasy要素

属性	必須	説明
font-family	✔	カンマで区切ったフォント名を優先順位の高いものから順に記述します。

表 2.26 cursive要素

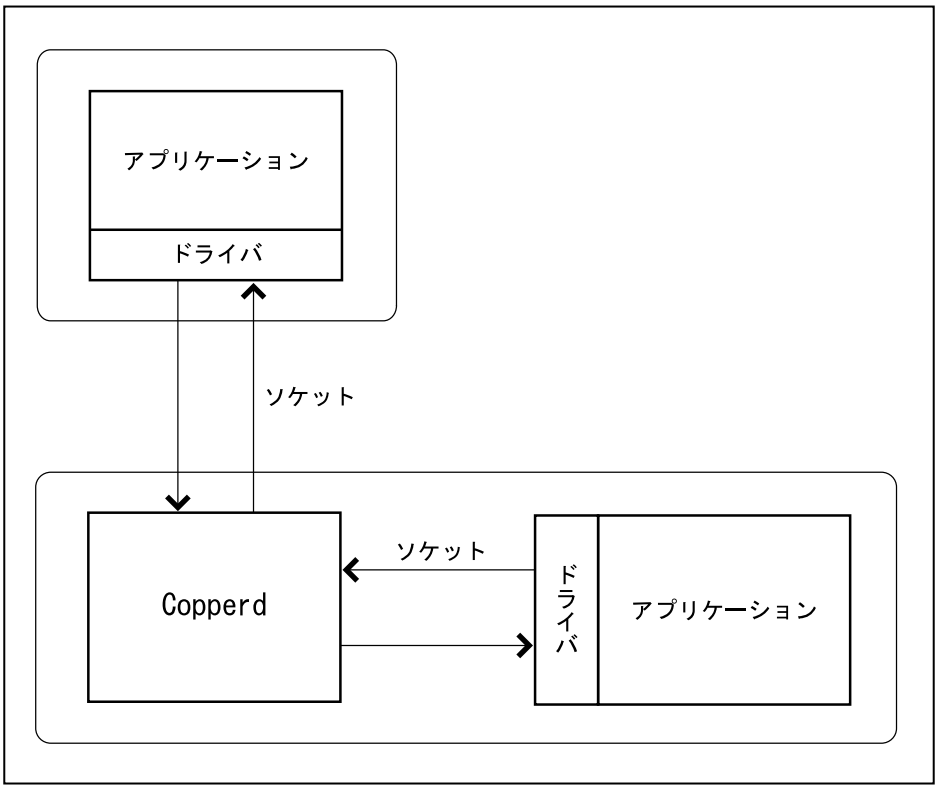
属性	必須	説明
font-family	✔	カンマで区切ったフォント名を優先順位の高いものから順に記述します。

3.開発者ガイド

3.1 プログラムインターフェース(ドライバ)の概要

Copper PDF自体はJavaによるプログラムですが、Java以外にも各プログラミング言語向けのドライバが用意されています。ドライバは動作中のCopper PDFサーバー(copperd)にソケット通信により接続します (一般的なデータベースサーバーと同様のしくみです)。

図 3.4 copperdとドライバ



ドライバとCopper PDFとの通信は、以下の流れが基本となります。

1. サーバーへの接続・認証
2. エラーハンドラ・プログレスリスナの設定
3. 変換結果の出力先の設定
4. プロパティの設定
5. リソースの送信・アクセス許可
6. ドキュメント本体の送信
7. 通信の終了

上記のうち、2,3,4は順序が入れ替わっても構いませんが、他の手順は正しい順序で実行することが必要です。各手順について順を追って説明します。

3.1.1 サーバーへの接続・認証

copperdにアクセスするためには、copperdのホスト名、ポート番号を知る必要があります。これらはcopperdの設定(copperd.properties)によります。ローカルマシンで初期設定のままのCopper PDFを動かしている場合、ホスト名はlocalhost(あるいは127.0.0.1(IPv4)または::1(IPv6))、ポート番号は8099です。

簡単なセキュリティ機能として、ユーザーIDとパスワードにより認証があります。ユーザーIDは"user"で固定です。サーバーの初期設定の状態ではパスワードは"kappa"ですが、サーバー側でcopperdコマンドにより変更することができます。

copprdはクライアントのIPアドレスによりアクセスを制限します。初期設定の状態ではローカルマシン(127.0.0.1(IPv4)または::1(IPv6))からのアクセスだけが許可されています。これはサーバー側のアクセス制御の設定(access.txt)を編集することで起動中に変更することができます。

3.1.2 エラーハンドラ・プログレスリスナの設定

エラーハンドラ

エラーハンドラは、変換処理の過程で出力された警告やエラー、処理情報を受け取るためのインターフェースです (処理情報を受け取る機能は後で追加されたため「エラーハンドラ」という名前になっています)。エラーハンドラが受け取ることができるメッセージは以下の4つに分類されます。

種類	コード	説明
警告	1	処理の続行が可能なエラーです。
エラー	2	処理の続行が不可能になるか、出力結果を得られなくなるエラーです。
致命的エラー	3	通信障害など、システムの問題に起因する深刻なエラーです。
処理情報	4	これはエラーメッセージではありません。出力済みのページ数、処理中のセクションなどの情報です。

プログレスリスナ

プログレスリスナはサーバー側でのデータの読み込み状況や最終的なデータサイズをプログラムが知るためのインターフェースです。

プログレスリスナには、サーバー側で処理済の入力データのバイト数が渡されます。また、プログレスリスナが要求した場合は、変換結果の先頭が得られる前に、結果全体のデータのバイト数が渡されます。前者はプログレスバーなどでユーザーに進行状況を伝えるために利用できます。後者はHTTPのContent-Lengthヘッダを送るために利用できます。

3.1.3 出力先の設定

変換結果はストリーム、ファイル等に出力することができます。出力先の指定方法は、プログラミング言語によります。Copper PDFはウェブ上での利用を重視しているため、クライアントのブラウザに送る方法は必ず用意されています。

3.1.4 入出力プロパティの設定

ドキュメントの変換方法の詳細は入出力プロパティによって細かく指定することができます。利用可能なプロパティのリストはプロパティ一覧を参照してください。

3.1.5 リソースの送信・アクセス許可

Copper PDFで文書を変換する場合、CSSファイルや画像ファイルといったリソースがどこにあるかが問題となります。リソースがサーバー側からアクセスできる場所(例:copperdが動作しているマシンのディスク上)にある場合は、copperdが直接リソースを取得できます。リソースがクライアント側(ドライバを使用するアプリケーションが動いているマシン)からアクセスでき、サーバー側からアクセスできない場所にある場合は、ドライバが事前にリソースをサーバーに送る必要があります。

文書内から画像などがURLによって参照されている場合、copperdはまず、そのURLで表されるリソースが既にドライバにより送られているかどうかを調べます。送られている場合は、そのリソースを使います。そうでない場合は、サーバー側でディスクやネットワーク等からリソースを取得しようと試みます。

図 3.5 copperdがリソースにアクセスする場合(1)

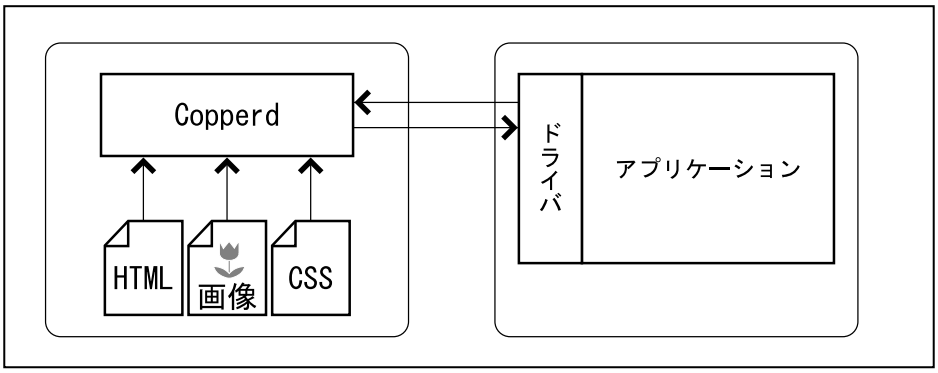


図 3.6 copperdがリソースにアクセスする場合(2)

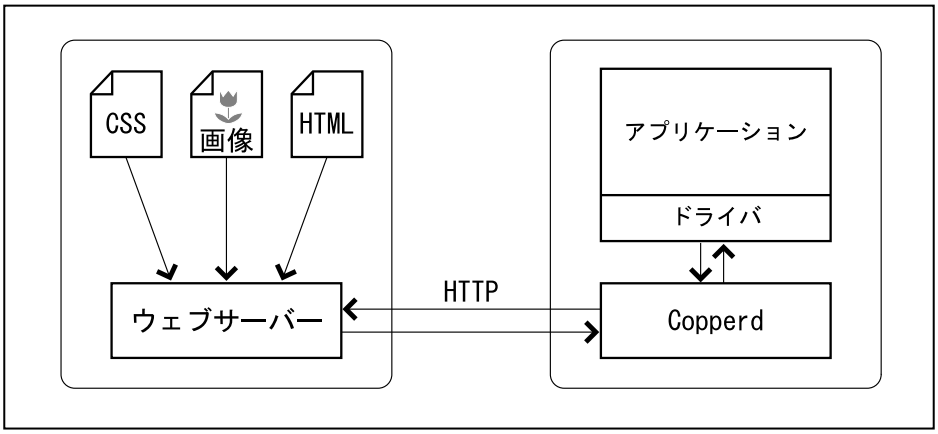
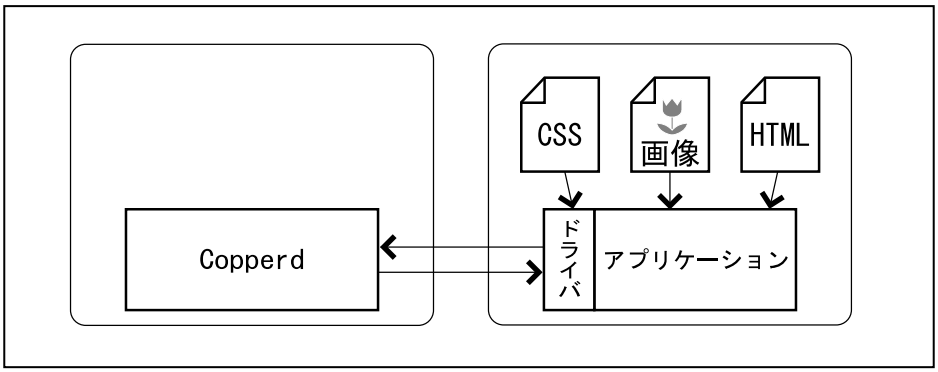


図 3.7 クライアント側からcopperdにリソースを送る場合



リソースをサーバーに送る

ドライバがリソースをcopperdに送る際には、リソース本体のデータとリソースの仮想的なURLに加え、リソースのMIME型およびキャラクタ・エンコーディングを送ることができます。MIME型とキャラクタ・エンコーディングは必須ではなく、省略された場合は拡張子やファイルの内容をもとにサーバー側で自動的に判断されます。また、画像などのバイナリデータではキャラクタ・エンコーディングは無意味です。

仮想的なURLは、*file:///var/data/image.gif*や、*http://host/style.css*といった文字列です。copperdはこれらのURLで表されるリソースが必要になった場合、そのURLで表される実際の場所にアクセスすることはせず。クライアント側から送られたデータを使います。

この方法は、事前に画像などのデータをサーバーに送り出す手間がかかる分、パフォーマンス上不利になります。また、アプリケーションは、本文から参照されているリソースを事前に把握している必要があります。ドライバには必要なリソースを自動的に判断する機能がないため、事前にどのリソースを送るかはアプリケーション側に委ねられます。

一方で、変換対象やリソースを1つ1つアプリケーションで指定するため、予期しなかったファイルにアクセスされてしまうといった、セキュリティ上の危険は少なくなります。

リソースにサーバーからアクセスする場合

必要とするリソースがドライバから送られていなかった場合、copperdは実際にそのURLで表される場所にアクセスしてデータを取得しようと試みます。

この方法の利点は、変換対象の文書から参照されているリソースを、copperdその都度自動的に集めてくることです。また、リソースがサーバーのディスク上にある場合は、事前にドライバがリソースを送る(この作業は実質的には、copperdがアクセスできる場所にファイルをコピーする作業です)手間が省けるため、パフォーマンス上有利です。

欠点として、セキュリティの問題が挙げられます。変換対象となる文書を誰でも編集できる場合、そこにサーバー上のファイル名を指定することで、サーバー上のファイルが盗まれてしまう可能性があります。また、http://で始まるURLを使うことで、他のサーバーへの不正なアクセスのための踏み台にされる恐れがあります。そのため、ネットワークの構成を含めて十分に注意して運用することが必要となります。

無制限にサーバー上のリソースにアクセスされるのを防止するため、copperdがアクセスするリソースを制限する簡単なセキュリティ機能が用意されています。サーバー上のリソースを利用するには、適宜アクセス許可を行う必要があります。



デフォルトの状態では、サーバーから全てのリソースへのアクセスが禁止されています copperdが文書に関連するCSSや画像等にアクセスするためにはアプリケーションで明示的に許可する必要があります。

URLパターン

リソースへのアクセス許可・制限は、URLパターンによって指定します。本文の変換を始める前に、クライアントは、利用できるURLと除外するURLのパターンを指定します。

URLパターンにはワイルドカードを使うことができます。"*"というワイルドカードは、'/'(スラッシュ)以外の任意の文字列を表します。"**"というワイルドカードは、それに加えて'/'も含めることを表します。ワイルドカードの例は以下の通りです。

- *http://www.company.com/** というパターンは、*http://www.company.com/* 直下の全てのリソースを現します。
- *http://www.company.com/image/*** は、*http://www.company.com/image/* 以下の全てのリソースを表します。
- *http://www.company.com/**.css* は、*http://www.company.com/* 以下の全てのCSSファイルを表します。

アクセスの制御は指定された順に行われます。

例えば、最初に次のパターンへのアクセスを禁止したとします。

- *http://www.company.com/secret*

次に以下のパターンへのアクセスを許可したとします。

- *http://www.company.com/***

このとき、*http://www.company.com/style.css* へのアクセスは許可されますが、*http://www.company.com/secret/image.jpeg* へのアクセスは禁止されます。

逆に、最初に以下のパターンへのアクセスを許可したとします。

- *http://www.company.com/***

この場合は、後の指定に関係なく *http://www.company.com/* 以下へのアクセスが全て許可されてしまいます。

両者の組み合わせ

クライアントからリソースを事前に送る方法と、サーバー側から積極的にリソースを取得する方法は、組み合わせて使うことができます。前に説明した通り、copperdはまず、クライアントから送られたリソースを利用しようと試みるため、前者の方法が優先されます。従って、URLパターンによるアクセス制御はクライアントから送られたリソースには適用されません。例えば、*file:///var/data/*.gif* へのアクセスが禁止されていたとしても、クライアントから仮想URL *file:///var/data/image.gif* として送られたリソースは有効です。

3.1.6 ドキュメント本体の送信

最後に変換対象となるドキュメント本体の送信を行います。リソースの場合と同様に、ドキュメントの仮想的なURLを送る必要があります。このURLは、文書中で使われる相対パスを解決するために使われます。また、同様に必須ではありませんが、文書のMIME型とキャラクタ・エンコーディングを明示することができます。サーバー側での文書の変換処理は、本体の送信と並行して行われます。

あるいは、ドライバから対象となる文書のURLだけを送って、リソースのときと同様にサーバー側で変換対象の文書を取得することもできます。

3.1.7 通信の終了

最後に通信を完了する操作を実行する必要があります。このとき、copperdは全ての変換結果をドライバに送り返し、接続を切断します。

3.2 Javaドライバ

3.2.1 使用方法

Java用ドライバはCopper PDF本体とは別に配布されています。http://www.cssj.jp/download/driver.html からダウンロードしてください。libディレクトリ内のcssj-driver-1.x.x.jarがドライバのライブラリです。このファイルをクラスパスに追加(あるいはアプリケーションのライブラリディレクトにコピー)してください。

ドライバの窓口となるクラスはjp.cssj.cti.CTIDriverManagerです。例えばlocalhostの8099番ポートで起動しているcopperdに、ユーザーID"user"、パスワード"kappa"で接続するには、以下のようにします。

例 3.1 ソケット通信インターフェース

```
//ドライバクラスのインポート
import jp.cssj.cti.CTIDriver;
import jp.cssj.cti.CTIDriverManager;
import jp.cssj.cti.CTISession;

...

CTIDriver driver = CTIDriverManager.createDriverFor("localhost",
"8099");
CTISession session = driver.createSession("user", "kappa");

//各種操作
...
```

3.2.2 APIの概要

ここではAPIによるアクセスの概要で説明した各手順に対応する関数を列挙します。各関数の詳細はapidocディレクトリ内のJavadocか、オンラインのAPIドキュメントを参照してください。

サーバーへの接続・認証

- public static CTIDriver createDriverFor(String host, int port)
- public CTISession createSessio n(String user, String password) throws IOException, SecurityException

エラーハンドラ・プログレスリスナの設定

- public void setErrorHandler(ErrorHandler eh)
- public void setProgressListener(ProgressListener l)

出力先の設定

- public void setOutput(OutputStream out, String mimeType) throws IOException

プロパティの設定

- public void setProperty(String name, String value) throws IOException

リソースの送信・アクセス許可

- public void includeResource(String uriPattern) throws IOException
- public void excludeResource(String uriPattern) throws IOException
- public OutputStream sendResourc e(String uri, String mimeType, String encoding) throws IOException

本体の送信

- public void formatMain(String uri) throws IOException
- public OutputStream sendMai n(String uri, String mimeType, String encoding) throws IOException

通信の終了

- public void close() throws IOException

3.2.3 サンプル

以下は、サーバー側から取り出したデータを変換するサンプルです。

例 3.2 jp.cssj.cti.examples.ServerResource

```
package jp.cssj.cti.examples;

import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.OutputStream;

import jp.cssj.cti.CTIDriver;
import jp.cssj.cti.CTIDriverManager;
import jp.cssj.cti.CTISession;
import jp.cssj.cti.helpers.StdioErrorHandler;

/**
 * サーバーでデータを取得して変換します。
 */
public class ServerResource {
    /** 接続先のホスト名。 */
    private static final String HOST = "localhost";

    /** 接続先のポート番号。 */
    private static final int PORT = 8099;

    /** パスワード。 */
    private static final String PASSWORD = "kappa";

    public static void main(String[] args) throws Exception {
```

```
        //ドライバを取得
        CTIDriver driver = CTIDriverManager.createDriverFor(HOST,
PORT);
        //接続する(ユーザー名は"user"で固定)
        CTISession session = driver.createSession("user",
PASSWORD);
        try {
            //test.pdfに結果を出力する
            OutputStream out = new BufferedOutputStream(new
FileOutputStream(
                    "test.pdf"));
            try {
                session.setOutput(out, "application/pdf");
                //エラーメッセージを標準出力に表示する
                session.setErrorHandler
(StdioErrorHandler.getInstance());

                //ハイパーリンクとブックマークを作成する
                session.setProperty("output.pdf.hyperlinks",
"true");
                session.setProperty("output.pdf.bookmarks",
"true");

                // http://www.cssj.jp/以下にあるリソースへアクセスする
                session.includeResource("http://www.cssj.jp/**");
                // index.htmlを変換
                session.formatMain
("http://www.cssj.jp/index.html");
            } finally {
                out.close();
            }
        } finally {
            //セッションを閉じる(忘れやすいので注意!)
            session.close();
        }
    }
}
```

クライアント側のデータを変換するサンプルを含め、これらのファイルはsrc/examplesに収められています。

3.2.4 サーブレットの作成

webappにウェブアプリケーションのサンプルが収められています。このサンプルのソースコードはwebapp/WEB-INF/srcにあります。

このサンプルは、index.pdfにアクセスすると、index.jspの出力結果をPDFに変換されたものが 表示されるというものです。webappディレクトリをサーブレット・コンテナに配備することで実際に動かすことができます。web.xml内のcontext-paramの部分を接続先のcopperdに合わせて設定してください。

サンプルのjp.cssj.cti.servlet.AbstractCSSJServletを継承することで、PDFを出力するサーブレットを簡単に作ることができます。ドライバを利用してユーザーが全く独自に

サーブレットを作ることも可能ですが、変換結果を直接クライアントを送る場合は、以下のようにContent-Lengthヘッダを送ることを 忘れないで下さい。Content-Lengthヘッダを送らないと、Acrobat Readerプラグインで表示されない場合があります。

例 3.3 Content-Lengthヘッダの送出

```
...
import jp.cssj.cti.helpers.ProgressAdapter;
...セッションの作成...
session.setProgressListener(new ProgressAdapter(true) {
    public void contentLength(long contentLength) {
        response.setContentLength((int) contentLength);
    }
});
...変換処理...
```

3.2.5 フィルターを使ったServlet/JSPの変換

ウェブアプリケーションのサンプルに含まれるjp.cssj.cti.servlet.CSSJFilterは、Servlet 2.3の「フィルタ」を利用してServletまたはJSPの出力結果をPDFに変換するものです。

サンプルでは、filterディレクトリ内に置かれたファイルをPDFに変換します。ファイルは静的なファイルのほか、JSPなど動的なファイルでも構いません。

CSSなどのリソースはresourcesファイルに置いています。これらのファイルに直接アクセスされるのを防ぐために、jp.cssj.cti.servlet.AccessFilterを使うことが出来ます。このフィルタは、CSSJFilter以外からのアクセスに対して、404エラーを返します。

3.2.6 ソースコード

ドライバのソースはドライバの配布ファイルのsrcディレクトリに収められています。このソースは、JDK1.4.2以降でコンパイルすることができます。

3.3 Perlドライバ

3.3.1 使用方法

Perl用ドライバはCopper PDF本体とは別に配布されています。http://www.cssj.jp/download/driver.html からダウンロードしてください。アプリケーションは、codeディレクトリをライブラリパスに含め、`use CSSJ::Driver;`でモジュールをインポートしてください。

3.3.2 APIの概要

ここではAPIによるアクセスの概要で説明した各手順に対応する関数を列挙します。各関数の詳細はapidoc内のAPIドキュメントか、オンラインのAPIドキュメントを参照してください。

サーバーへの接続・認証

- create_driver_for HOST PORT [ENCODING]
- CSSJ::Driver->create_session USER PASSWORD

エラーハンドラ・プログレスリスナの設定

- CSSJ::Session->set_error_func FUNCTION
- CSSJ::Session->set_progress_func FUNCTION
- CSSJ::Session->set_content_length_func FUNCTION

出力先の設定

- CSSJ::Session->set_output OUTPUTHANDLE [MIME_TYPE]

プロパティの設定

- CSSJ::Session->set_property NAME VALUE

リソースの送信・アクセス許可

- CSSJ::Session->include_resource URI_PATTERN
- CSSJ::Session->exclude_resource URI_PATTERN
- CSSJ::Session->start_resource FILEHANDLE URI [MIME_TYPE ENCODING]
- CSSJ::Session->end_resource FILEHANDLE

本体の送信

- CSSJ::Session->format_main URI
- CSSJ::Session->start_main FILEHANDLE URI [MIME_TYPE ENCODING]
- CSSJ::Session->end_main FILEHANDLE

通信の終了

- CSSJ::Session->close

3.3.3 サンプル

Perl用インターフェースは、ファイルハンドルに対する出力をキャプチャしてサーバーに送ります。以下の例では$session->start_mainと$session->flush_mainの間で出力されたHTMLが変換されます。

例 3.4 content.pl

```
#!/usr/bin/perl
=head1 NAME

コンテンツ変換サンプル

=head2 概要

start_mainとend_mainの間の出力結果をPDFに変換します。

=cut
use lib '../code';
# ドライバのインポート
use CSSJ::Driver(create_driver_for);

# ドライバの作成
$driver = create_driver_for('localhost', 8099, 'EUC-JP');
# 接続
$session = $driver->create_session('user', 'kappa');

# Content-Lengthヘッダの送信
$session->set_content_length_func (sub {
    my $length = shift;
    print "Content-Length: $length\n\n";
});

# Content-Typeヘッダの送信
print "Content-Type: application/pdf\n";

# リソースの送信
$session->start_resource(STDOUT, 'file:skin.css');
print << 'EOF';
  p {
    background-color: Gray;
  }
EOF
$session->end_resource(STDOUT);

# 本体の送信
$session->start_main(STDOUT, 'file:test.html', 'text/html', 'EUC-
JP');
print << 'EOF';
<html>
```

```
  <head>
    <title>テストドキュメント</title>
    <link rel="StyleSheet" type="text/css" href="skin.css">
  </head>
  <body>
    <p>こんにちは</p>
  </body>
</html>
EOF
$session->end_main(STDOUT);

# セッションを閉じる
$session->close();
```

start_resource,start_mainにファイルハンドルを渡すと、それぞれend_resource,end_mainが呼び出されるまで、ファイルハンドルに出力されたデータをcopperdに送ります。その間、ファイルハンドルの本来の機能(STDOUTでは標準出力にデータを送るなど)は使えなくなります。

3.4 PHPドライバ

3.4.1 使用方法

PHP用ドライバはCopper PDF本体とは別に配布されています。http://www.cssj.jp/download/driver.html からダウンロードしてください。アプリケーションは、codeディレクトリ内のcssj_driver.phpをインクルード(require_once)してください。PHP用ドライバではcssj_driver.php(driverパッケージ)の関数を用いてください。他の関数は低レベルな処理をするもので、通常は必要ありません。

また、ソース中の日本語のコメントのために、ファイルはEUC-JPエンコーディングとなっています。PHPの内部エンコーディングにEUC-JP以外を使う場合は、asciiディレクトリの中にASCIIコードに変換したものがありますので、そちらを利用してください。

3.4.2 APIの概要

ここではAPIによるアクセスの概要で説明した各手順に対応する関数を列挙します。各関数の詳細はapidocディレクトリ内のAPIドキュメント(phpDocumentor) またはオンラインのAPIドキュメントを参照してください。

サーバーへの接続・認証

- mixed &cssj_create_driver_for ($host $host, $port $port, [$encoding $encoding = 'ISO-8859-1'])
- mixed &cssj_create_session ($driver &$driver, $user $user, $password $password)

エラーハンドラ・プログレスリスナの設定

- void cssj_set_error_func ($session &$session, $errorFunc &$errorFunc)
- void cssj_set_progress_func ($session &$session, $progressFunc &$progressFunc)

出力先の設定

- boolean cssj_set_output ($session &$session, $out &$out, [$mimeType $mimeType = 'application/pdf'])

プロパティの設定

- boolean cssj_set_property ($session &$session, $name $name, $value $value)

リソースの送信・アクセス許可

- boolean cssj_include_resource ($session &$session, $uriPattern $uriPattern)
- boolean cssj_exclude_resource ($session &$session, $uriPattern $uriPattern)
- boolean cssj_ob_start_resource ($session &$session, $uri $uri, [$mimeType $mimeType = 'text/css'], [$encoding $encoding = ''])
- boolean cssj_ob_end_flush_resource ()

本体の送信

- boolean cssj_format_main ($session &$session, $uri $uri)
- boolean cssj_ob_start_main ($session &$session, $uri $uri, [$mimeType $mimeType = 'text/html'], [$encoding $encoding = ''])
- boolean cssj_ob_end_flush_main ()

通信の終了

- boolean cssj_close ($session &$session)

3.4.3 サンプル

PHP用インターフェースは、出力バッファを介してドキュメントを変換するのが特徴です。以下の例ではcssj_ob_start_mainとcssj_ob_end_flush_mainの間に記述されたHTMLが変換されます。

例 3.5 ob.php

```
<?php
require_once ('../code/cssj_driver.php');
header("Content-Type: application/pdf");

//ドライバの作成
$driver = cssj_create_driver_for('localhost', 8099);

//セッションの開始
$session = cssj_create_session($driver, 'user', 'kappa') or die('
サーバーに接続できません');

//リソースの送信
cssj_ob_start_resource($session, 'file:test.css');
readfile('test.css');
cssj_ob_end_flush_resource();

//出力結果の変換の開始
cssj_ob_start_main($session, 'file:ob.html');
?>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=EUC-JP">
    <link rel="StyleSheet" type="text/css" href="test.css">
    <title>Hello CSSJ</title>
  </head>
  <body>
    <h2>ただいまの時刻</h2>
    <p><?php echo date("l dS of F Y h:i:s A") ?></p>
  </body>
```

```
</html>

<?php
//出力結果の変換の終了
cssj_ob_end_flush_main();

//セッションの終了
cssj_close($session);

?>
```

単純に変数から変数に変換する場合は以下のような関数を定義しておくと便利です。実際の使用例はvar.phpを参照してください。

例 3.6 変数の変換

```
/**
 * 与えられたデータをPDFに変換して返します。
 *
 * @param $session セッション
 * @param $input 元のデータ
 * @return 変換結果PDF
 */
function &toPDF(&$session, $input) {
  //出力先
  $output = '';//nullの場合標準出力となるので、必ず文字列を代入しておく必要がある
  cssj_set_output($session, $output);

  //変換
  cssj_ob_start_main($session, 'file:test.html');
  echo $input;
  cssj_ob_end_flush_main();

  //セッションの終了
  cssj_close($session);

  return $output;
}
```

3.4.4 Content-Lengthヘッダの送信

変換結果の出力先はデフォルトではクライアントにそのまま返されます。このとき、自動的にContent-Lengthヘッダが送信されます。

それ以外の出力先を設定した場合はContent-Lengthヘッダは出力されません。最終的にクライアントにPDFを送信する場合は、Content-Lengthヘッダを送らないとAdobe Readerプラグインで表示されない場合がありますのでご注意ください。

3.5 copper Antタスク

頻繁に更新されるドキュメントを素早くまとめて変換するために、Apache Antによるバッチ処理をサポートしています。AntはJavaで開発されたフリーのビルド・ツールです。Antについての詳細は書籍などをご参照下さい。

3.5.1 使用方法

Copper PDFのAntタスクは、Copper PDF本体がインストールされたサーバー上で実行することができます。別途ライブラリをダウンロードする必要はありません。

3.5.2 copper タスク

copper タスクはドキュメントの一括変換を行うためのタスクです。なお、CSSJ 1.x系のcssjというタスク名も使用できます。copper タスクを使用するためには、Antのbuild.xml中で次のように宣言する必要があります。

例 3.7 copper タスクの宣言

```
<path id="lib.path">
  <fileset dir="Copper PDFのlibディレクトリのパス" includes="*.jar"/>
</path>
<typedef classpathref="lib.path"
resource="jp/cssj/driver/anttask/tasks.properties"/>
```

以降、copperという要素名でタスクを使えるようになります。

表 3.1 copper タスクの属性

属性名	説明
configFile	Copper PDFの設定ファイル(profiles/default.properties)です。
configDir	Copper PDFの(license-keyファイルが置かれた)設定ディレクトリです。
srcDir	変換前のXML,HTMLファイルなどが格納されたディレクトリです。省略した場合、カレントディレクトリとなります。
includes	srcDir中の変換対象となるファイルのパターンです。
excludes	srcDir中の変換対象外となるファイルのパターンです。
destDir	出力先のディレクトリです。省略するとsrcDirと同じディレクトリになります。
suffix	出力結果ファイルの拡張子です。省略した場合は.pdfとなります。

copper要素内で、property要素を使って入出力プロパティを設定することができます。プロパティ名はname属性、値はvalue属性となります。

以下の例では、docs/manual.htmlからdocs/manual.pdfを出力します。

例 3.8 copper タスクの使用例

```
<copper includes="docs/manual.html" suffix=".pdf"
    configFile="conf/profiles/default.properties">
    <includeresource pattern="**" />
    <property name="output.pdf.bookmarks" value="true" />
    <property name="output.pdf.hyperlinks" value="true" />
    <property name="output.pdf.font.policy" value="cid-keyed" />
</copper>
```

上記のサンプルは実際にドキュメントディレクトリ(docsあるいは/usr/share/doc/copper-pdf)に収められています。一旦manual.pdfを削除して、ドキュメントディレクトリ内でantを実行することで試すことができます。

3.5.3 うまく動かない場合

Can't connect to X11... というエラーが表示される

JDK 1.4.2では以下のようなエラーが発生することがあります。

例 3.9 エラーメッセージの例

```
   [copper] java.lang.InternalError: Can't connect to X11 window
server using ':0.0' as the value of the DISPLAY variable.
   [copper]     at sun.awt.X11GraphicsEnvironment.initDisplay
(Native Method)
   [copper]     at sun.awt.X11GraphicsEnvironment.<clinit>
(X11GraphicsEnvironment.java:134)
   [copper]     at java.lang.Class.forName0(Native Method)
...省略...
   [copper]     at org.apache.tools.ant.launch.Launcher.run
(Launcher.java:246)
   [copper]     at org.apache.tools.ant.launch.Launcher.main
(Launcher.java:67)
   [copper] Transcoded 0 file(s).

BUILD FAILED
/home/miyabe/workspaces/cssj/cssj/build/release/docs/build.xml:11:
java.lang.InternalError: Can't connect to X11 window server using
':0.0' as the value of the DISPLAY variable.
```

これは、Copper PDFが一部でグラフィック環境を必要とする処理を行っているため、ディスプレイが接続されていない環境で発生するものです。以下のように、ANT_OPTS環境変数でjava.awt.headless システムプロパティをtrueにするように設定することで、問題が解消されます。

例 3.10 java.awt.headlessの設定

```
export ANT_OPTS=-Djava.awt.headless=true
```

ライセンスが認証されない場合

「ライセンスファイルが存在しないか期限切れです」と表示される場合は、ライセンスキーのインストールか、confディレクトリ(license-keyファイルが置かれたディレクトリ)のパスを指定する必要があります。

confディレクトリはcopperタスクのconfigDir属性で指定するか、ANT_OPTS環境変数によりjp.cssj.copper.configシステムプロパティで指定することもできます。

例 3.11 設定ディレクトリの指定(ANT_OPTS環境変数)

```
export ANT_OPTS=-Djp.cssj.copper.config=/etc/copper-pdf
```

例 3.12 設定ディレクトリの指定(configDir属性)

```
<copper includes="docs/manual.html" suffix=".pdf"
    configFile="conf/profiles/default.properties"
    configDir="/etc/copper-pdf">
    <includeresource pattern="**" />
    <property name="output.pdf.bookmarks" value="true" />
    <property name="output.pdf.hyperlinks" value="true" />
    <property name="output.pdf.font.policy" value="cid-keyed" />
</copper>
```

4.デザイナーガイド

4.1 Copper PDFによるPDFの作成

Copper PDFはHTMLやCSS等、ウェブコンテンツの作成で標準となっている方法を使用するため、ウェブコンテンツの作成の知識があれば、PDFの作成も同様にできるのが特徴です。このドキュメントでは、Copper PDF独自の機能や、注意点について解説します。

4.1.1 Copper PDFでドキュメントをレイアウトするには

ドキュメントをレイアウトする方法は、通常のウェブアプリケーションの作成と大差ありません。HTMLエディタ等を使ってHTMLや、プログラムで処理するためのテンプレートを作成し、それをプログラムによりPDFに変換するという手順となります。

Copper PDFによる出力結果を手軽に確認するために、Copper PDFにはcopper-webappというツールが用意されています。このツールを起動すると、ローカルマシン上にあるHTMLファイルをウェブベースのツールでPDFに変換することができます。

4.1.2 入出力プロパティ

Copper PDF独自の機能や、一般的なブラウザの環境設定やオプションに相当する部分を設定するためには、名前と値の組み合わせである「入出力プロパティ」を設定します。入出力プロパティの一覧は資料集の入出力プロパティの章を参照してください。

入出力プロパティはシステム管理者やプログラマにより設定されますが、管理者やプログラマにより許可されている場合は (input.property-piがtrueに設定されている)、HTMLやXML文書中でjp.cssj.property処理命令により設定することもできます。処理命令のname属性にプロパティ名、value属性に値を記述してください。また、必ずドキュメントの先頭から最初の要素の間に記述してください。

例 4.1 jp.cssj.property処理命令による入出力プロパティの設定

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<?jp.cssj.property name="output.pdf.encryption" value="v2"?>
<?jp.cssj.property name="output.pdf.encryption.user-password"
value="user"?>
<?jp.cssj.property name="output.pdf.encryption.owner-password"
value="owner"?>
<?jp.cssj.property name="output.pdf.encryption.permissions.print"
value="false"?>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>暗号化1</title>
    <style type="text/css">
    </style>
```

```
  </head>

  <body>
    <h1>暗号化1</h1>
    <p>
      この文書のユーザーパスワードは"user" オーナーパスワードは"owner"です。
      印刷が禁止されています。
    </p>
  </body>
</html>
```

なお、jp.cssj.property処理命令により設定できないプロパティがあります。詳細は資料集を参照してください。

4.1.3 ページの参照

Copper PDFには、目次をつくる機能(cssj:make-toc要素)と、ある内容が印刷されるページ番号を表示する機能(-cssj-page-ref関数)があります。

これらの機能を利用するためには、processing.page-references をtrueに設定し、ページ参照情報を収集する機能を有効にしてください。



CSSJ 1.xのcssj:toc要素、"processing.make-toc-before"は廃止されました。

4.1.4 2パス以上の変換処理

Copper PDFには、ページ番号による参照を可能にするために、同じドキュメントを複数回処理する(それぞれの処理を「パス」と呼びます)機能が用意されています。複数パスが必要になるのは、ドキュメントの最後以外の場所に目次を入れる場合と、-cssj-page-ref関数を使用する場合です。

文書中にページ番号を表示する場合、1度目のパスでアンカーや見出し等が表示されるページの番号を求めて、2度目のパスで実際に参照のためのページ番号を挿入します。また、目次と本文を通しでページ番号を振る場合や、非常に大きなドキュメントで、ページ番号の挿入のためにページ数が増加する場合は、3パス以上必要になることがあります。

パスの数は入出力プロパティ processing.pass-count により設定可能です。

4.2 CSSによるドキュメントのレイアウト

Copper PDFはHTMLとXMLをCSSによってスタイル付けすることができます。

Copper PDFは CSS Level 2 Revision 1 に準拠しています。Copper PDFがサポートするCSSプロパティの一覧は資料集を参照してください。

4.2.1 スタイルシートの型

Copper PDFは、特に型が指定されていないスタイルシートをCSS(MIME型text/css)として認識します。Content-Style-Type ヘッダ (HTML中では<meta http-equiv="Content-Style-Type" content="...といった記述) やstyle要素のtype属性等によってtext/css以外がスタイルシートの型として指定されている場合は、スタイルシートをCSSとして認識しません。

4.2.2 メディアタイプ

style要素のmedia属性、あるいはCSSの@media指示子等でスタイルシートが適用されるメディアタイプが限定されている場合、通常Copper PDFは印刷メディア向けのスタイルだけを適用します。すなわち、Copper PDFのメディアタイプはall print paged visual bitmap staticのいずれかです。

なお、Copper PDFのメディアタイプは output.media_types 入出力プロパティによって変更することができます。これによって、印刷メディア向け以外のスタイルを適用することができます。

4.2.3 ドキュメント中にスタイルシートを記述する

ドキュメント中にスタイルおよびスタイルシートを記述する方法は以下の3通りがあります。

1. HTMLのstyle属性
2. HTMLのstyle要素
3. jp.cssj.stylesheet処理命令

HTMLのstyle属性

style属性により、各要素に直接スタイルを指定する方法です。style属性には、適用したいCSSのプロパティを記述してください。

例 4.2 style属性によるスタイルの指定

```
<html>
<body style="border: 1px Black dashed;">
<h1 style="font-style: italic;">題名</h1>
<p style="text-decoration: underline;">本文<span style="font-size:
x-large;">大きなテキスト</span>本文</p>
</body>
</html>
```

上記の例は以下のように表示されます。

例 4.3 style属性によるスタイルの指定(表示)



style要素はHTMLだけではなく、XML文書内でも利用可能です。ただし、style要素が名前空間"http://www.w3.org/1999/xhtml" に属するように、名前空間宣言を行ってください。

例 4.4 style属性によるスタイルの指定(XML)

```
<?xml version="1.0">
<body xmlns:html="http://www.w3.org/1999/xhtml" html:style="border:
 1px Black dashed;">
<h1 html:style="font-style: italic;">題名</h1>
<p html:style="text-decoration: underline;">本文<span
html:style="font-size: x-large;">大きなテキスト</span>本文</p>
</body>
```

HTMLのstyle要素

一般的なブラウザと同じくstyle要素内にスタイルシートを記述することができます。

例 4.5 style要素によるスタイルシートの記述

```
<html>
<head>
<style type="text/css" media="print">
    body {
        border: 1px Black dashed;
    }
    h1 {
        font-style: italic;
    }
    p {
        text-decoration: underline;
    }
    span {
        font-size: x-large;
    }
</style>
</head>
<body>
<h1>題名</h1>
<p>本文<span>大きなテキスト</span>本文</p>
</body>
</html>
```

style要素はなるべくhead要素内に記述してください。Copper PDFは他の場所のstyle要素も認識しますが、ドキュメント中でstyle要素が現れる以前の部分には、記述されたスタイルが適用されなくなります。

また、XMLドキュメント中ではstyle要素によるスタイル指定は行わず、次に説明するjp.cssj.stylesheet処理命令を使用するか、あるいは一般的なブラウザでもサポートされているxml-stylesheet処理命令を使用してください。

jp.cssj.stylesheet処理命令

jp.cssj.stylesheet処理命令はHTMLのstyle要素に相当する機能を処理命令 (<?で始まり?>で終わるHTMLやXML中の特別な記述。) によって実現したものです。HTML, XMLの両方のドキュメントの先頭で使用することができます。

style要素のtype, media属性に相当する部分はHTMLの属性と同様の形式で記述し、スタイルシートは'[]'で囲みます。

例 4.6 jp.cssj.stylesheet処理命令によるスタイルシート記述

```
<?jp.cssj.stylesheet type="text/css" media="print" [
    body {
        border: 1px Black dashed;
    }
    h1 {
        font-style: italic;
    }
    p {
        text-decoration: underline;
    }
    span {
        font-size: x-large;
    }
] ?>
<html>
<body>
<h1>題名</h1>
<p>本文<span>大きなテキスト</span>本文</p>
</body>
</html>
```

4.2.4 外部のCSSの使用

ドキュメントと外部のスタイルシートを結びつける方法は以下の3通りがあります。

1. HTMLのlink要素
2. CSSの@import指示子
3. xml-stylesheet処理命令

HTMLのlink要素

一般的なブラウザと同様に、link要素によってスタイルシートをドキュメントに関連付けることができます。

例 4.7 linkタグによるスタイルシートの指定

```
<link rel="stylesheet" type="text/css" media="print" href="スタイルシートのURL">
```

rel属性にキーワードalternateを加えることで、1つのドキュメントに対して、複数の代替スタイルシートを用意することができます。title属性は代替スタイルシートの名前です。

例 4.8 alternateスタイル

```
<link rel="alternate stylesheet" title="a" type="text/css"
media="print" href="a.css">
<link rel="alternate stylesheet" title="b" type="text/css"
media="print" href="b.css">
```

代替スタイルシートは通常は適用されませんが、入出力プロパティ input.stylesheet.titles に代替スタイルシートのtitleを設定することで、適用するスタイルシートを選択することができます。

CSSの@import指示子

これは、HTMLのstyle要素とCSSの@importを組み合わせる方法です。

例 4.9 @import指示子によるスタイルシートの指定

```
<style type="text/css" media="print">
    @import url(スタイルシートのURL);
</style>
```

HTMLのstyle要素の代わりに、jp.cssj.stylesheet処理命令を使うこともできます。

例 4.10 @import指示子によるスタイルシートの指定(jp.cssj.stylesheet処理命令)

```
<?jp.cssj.stylesheet type="text/css" media="print" [
    @import url(スタイルシートのURL);
] ?>
```

xml-stylesheet処理命令

xml-stylesheet処理命令を使う方法は、一般的なブラウザでXMLとスタイルシートを結びつけるために広くサポートされている方法です。

スタイルシートを適用したいドキュメントの冒頭で、以下のように記述してください。

例 4.11 xml-stylesheet処理命令によるスタイルシートの指定

```
<?xml-stylesheet type="text/css" media="print" href="スタイルシートのURL"?>
```

alternate属性はlink要素のrel属性に対するalternateの指定に相当するものです。xml-stylesheet処理命令でalternate="yes"を指定することは、link要素にrel="alternate stylesheet"を設定するのと同じ意味です。

例 4.12 xml-stylesheet処理命令による代替スタイルシート

```
<?xml-stylesheet alternate="yes" title="a" type="text/css"
media="print" href="a.css"?>
<?xml-stylesheet alternate="yes" title="b" type="text/css"
media="print" href="b.css"?>
```

xml-stylesheetのtype属性にはCSSだけでなくXSLTスタイルシートを指定することもできます。この場合、type属性にtext/xslを設定し、href属性にはXSLTファイルの指定してください。

4.2.5 デフォルトのスタイルシート

Copper PDFでは、ドキュメント内の記述に関係なく、特定のスタイルシートを適用する機能が用意されています。input.default-stylesheet 入出力プロパティで指定されたスタイルシートが最初に適用されます。

4.3 CSSの拡張機能

4.3.1 名前空間

Copper PDFはCSSの名前空間のための拡張機能 (CSS Namespace Enhancements) をサポートしており、複数の名前空間が混在するXMLをスタイル付けすることができます。

CSSスタイルシート中で名前空間の接頭辞(prefix)とURIを指定するためには、@namespace指示子を使って以下のように宣言してください。

例 4.13 名前空間の宣言

```
/* デフォルトの名前空間のURIをhttp://www.w3.org/1999/xhtmlとする。 */
@namespace "http://www.w3.org/1999/xhtml";

/* 接頭辞rdfのURIをhttp://www.w3.org/1999/02/22-rdf-syntax-ns#とする。
*/
@namespace rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
```

なお、互換性のためにURIの部分はurl(http://www.w3.org/1999/xhtml) という書き方も許されています。

スタイルシートの選択子(selector)で接頭辞を使う場合は、'|'で区切ります。(':'でないことに注意してください。)

例 4.14 接頭辞を含む選択子

```
/* <pdf:Description>要素のスタイルを指定する。 */
rdf|Description { display: block; }

/* ref:about属性がhttp://foo.com/barであるitem要素のスタイルを指定する。
*/
item[rdf|about=http://foo.com/bar] { color: Red; }
```

選択子の記述方法と、意味は次のとおりです。

prefix|ELEMENT
: prefixが指す名前空間に属するELEMENT

|ELEMENT
: どの名前空間にも属さないELEMENT

***|ELEMENT**
: 任意の名前空間にも属すか、どの名前空間にも属さないELEMENT

ELEMENT
: デフォルトの名前空間に属するELEMENT

4.3.2 ページカウンタ

Copper PDFは、ページの番号付けのために ページの生成ごとに処理されるページカウンタを用意しています。ページカウンタは、以下のように@page ルール内の 'counter-increment' プロパティにより宣言します。

例 4.15 ページカウンタ

```
@page {
    counter-increment: page;
}
```

ページカウンタの処理は、ページの内容が処理される直前に行われます。また、ページカウンタは通常のカウンタと同様に、'content' プロパティ内でcounter関数により参照可能です。従って、上記の宣言を行った場合、最初のページで {content: counter(page);} という宣言が処理されるとき、1が出力されます。

途中でページカウンタをリセットする場合 (例えば、目次が終わった後、本文で改めて番号を振りなおすなど) は、通常のカウンタと同様に'counter-reset' を使うことができます。例えば、pageという名前を1に設定しなおす場合は {counter-reset: page 1;} と宣言します。

4.4 HTML/XMLの処理

4.4.1 ドキュメントの判別

Copper PDFはドキュメントがHTMLかXMLであるかを、以下の情報をもとに判別します。

1. プログラマが指定したドキュメントのMIMEタイプ
2. HTTPのContent-Type ヘッダ

1の情報は、2より優先されます。上記の情報以外によりCopper PDFがドキュメントの型を判別することはありません。HTMLとして判別されたドキュメントは、<?xml〜で開始していても、HTMLとして認識します。ただし、HTMLと認識されても、XMLやXHTMLがサポートしている名前空間は認識されます。

HTMLと認識された文書は、ゆるやかに解釈されるため、文法ミスが許容されます。

一方、XMLと認識された文書は厳密に解釈されます。XHTMLを記述する場合、全ての要素名と属性名は小文字で記述してください。文法エラーがあった場合、処理が停止します。

4.4.2 文書情報

Copper PDFはHTMLのmeta要素から取得した情報を、PDFの文書情報として使用します。文書情報はHTMLの<meta name="名前" content="値">要素によって設定することができます。ただし、TITLEはHTMLのtitle要素の内容も使われます。

表 4.1 meta要素による文書情報の設定

名前	PDFの属性名	説明
TITLE	Title/タイトル	文書の表題。
DESCRIPTION SUBJECT	Subject/サブタイトル	文書に内容についての簡潔な説明。
KEYWORDS	Keywords/キーワード	スペースまたはカンマ区切りで羅列した、文書の内容に関連するキーワード。
AUTHOR	Author/作成者	文書の作成者。
PRODUCER	Producer/PDF変換	PDFを生成したプログラム。省略した場合はCopper PDFの名前とバージョンが入ります。
GENERATOR CREATOR	Creator/アプリケーション	HTML文書を生成したプログラム、エディタ、オーサリングツールなど。

なお、meta要素のname属性は大文字小文字を区別しません。<meta name="AUTHOR" content="作者名">としても、<meta name="Author" content="作者名">としても、PDF文書に作者名が設定されます。

4.4.3 見出し

HTMLのh1〜h6要素は、見出しとして特別な意味を持ちます。h1〜h6要素の数字は見出しのレベルとして認識され、数字が大きいものほど深い階層にあるものとして処理されます。

また、任意の要素に`html:header="レベル"`という属性を付与することで、任意の要素をHTMLのh1〜h6要素と同様の意味を持たせることができます。

ブックマーク

output.pdf.bookmarks をtrueにすると、PDFのブックマーク(しおり)が生成されます。ブックマークは、見出しのレベルによって自動的に階層化されます。

現在ページのセクション

ページの内容として出力済みの見出しを、`'content'` プロパティ内で-cssj-heading関数によって生成することができます。

例 4.16 見出しの表示

```
content: -cssj-heading(1) ' - ' -cssj-heading(2);
```

上の例はレベル1とレベル2の見出しを表示します。すなわち、ドキュメントの先頭から、前ページの最後までの間で、一番最後のh1とh2の内容を表示します。

-cssj-heading関数を -cssj-page-content が設定されたページで使用すると、ドキュメントの先頭から現在のページの最後までの間で、一番最後の見出しを表示することができます。これは、ノンブルに見出しを表示する場合に便利です。

4.4.4 目次の生成

見出しは、目次の生成に利用することができます。目次を生成するためには、Copper PDFが文書全体の見出しと、見出しのあるページ番号を収集している必要があります。

例 4.17 目次の生成

```
<cssj:make-toc xmlns:cssj="http://www.cssj.jp/ns/cssjml"
               counter="page-number" type="decimal"/>
```

ドキュメントの末尾に目次を生成する場合は、目次の生成の開始時点で本文が全て処理されていますが、ドキュメントの先頭や途中に目次を生成する場合は、2パス以上の処理が必要です。

counter属性はページ番号を振るために使用するページカウンタの名前です。typeは、ページ番号のタイプです。CSSの`'list-style-type'` で使われるのと同じタイプ名が利用可能です。

生成される目次は、次のようなリストの形式をしています。

例 4.18 生成される目次の例

```
  <ul class="cssj-toc">
    <li><a href="#cssj-heading-1"><span class="cssj-title">1. タイト
ル1</span><span class="cssj-page">1</span></a></li>
    <li><a href="#cssj-heading-2"><span class="cssj-title">2. タイト
ル2</span><span class="cssj-page">5</span></a></li>
    <li><a href="#cssj-heading-3"><span class="cssj-title">3. タイト
ル3</span><span class="cssj-page">8</span></a></li>
    <ul>
      <li><a href="#cssj-heading-4"><span class="cssj-title">3.1.
タイトル4</span><span class="cssj-page">8</span></a></li>
      <li><a href="#cssj-heading-5"><span class="cssj-title">3.2.
タイトル5</span><span class="cssj-page">9</span></a></li>
    </ul>
  </ul>
```

例として、以下のようなスタイルシートを使うことで目次を整形してください。

例 4.19 目次を整形するCSSの例

```
ul.cssj-toc, ul.cssj-toc ul {
  list-style: none;
}
ul.cssj-toc {
  margin: 0;
}
ul.cssj-toc ul {
  margin: 1em;
}
ul.cssj-toc li {
  margin: 1em 1em 1.5em 0;
  font-family: sans-serif;
  height: 0.5em;
  border-bottom: 1pt dotted;
}
ul.cssj-toc ul li {
  margin: 0 0 0.5em 1em;
  font-family: serif;
}
ul.cssj-toc span {
  background-color: White;
  padding: 0 0.5em;
}
ul.cssj-toc span.cssj-page {
  position: absolute;
  right: 0;
}
```

4.4.5 リンクとフラグメント

フラグメント識別子によるリンク

output.pdf.hyperlinks をtrueに設定すると、文書内のハイパーリンクがPDFに反映されます。

output.pdf.hyperlinks.fragment をtrueに設定すると、ドキュメントの一部に対してフラグメント識別子によるリンクが可能になります。フラグメントの作成方法は通常のHTMLと同じで、任意の要素のid属性と、a要素のname属性をサポートしています。

-cssj-page-ref関数

Copper PDFは、フラグメントが存在するページを表示するために、'content'内で使用できる-cssj-page-ref関数を用意しています。

例 4.20 フラグメントのページ番号の表示

```
@page {
    counter-increment: page;
}
span:before {
    content: 'p' -cssj-page-ref(frag, page, decimal, ', p') ' ';
}
a:after {
    content: ' (' -cssj-page-ref(attr(href), page, decimal, ', ') '
ページ)';
}
```

-cssj-page-ref関数の最初の引数は、参照するフラグメント識別子です。2番目の引数はページ番号の表示に使うページカウンタです。3番目の引数は数字のスタイルで、'list-style-type'で使われるのと同じスタイル名です。3番目の引数を省略するとdecimalスタイルで表示されます。4番目の引数(Copper PDF 2.0.2以降)は同じ名前のフラグメントが複数存在する場合、複数のページ番号を表示するための区切り記号です。4番目の引数を省略すると、最初のフラグメントのページ番号だけが表示されます。

フラグメント識別子の部分はattr関数を使うことができます。上の例のように、HTMLのa要素でhref属性の値を使うと、フラグメントへリンクすると同時に、フラグメントが存在知るページの番号を表示することができます。なお、フラグメント識別子の最初の'#'はあってもなくても構いません。

-cssj-page-ref関数を使うためには、一般的に2パス以上の処理の処理が必要です。ドキュメント全体の処理が一度終わらないと、ドキュメントの後の方にあるフラグメントのページが分からないためです。

4.4.6 Internet Explorerとの互換モード

完全ではありませんが、Copper PDFはWindows Internet Explorerに近いレイアウトを再現します。このモードを有効にするには、output.compatible_mode にtrueを設定してください。ただし、次のいずれかのpublic idを持つDOCTYPE宣言がある場合は、通常のモードでレイアウトします。

- -//W3C//DTD HTML 4.01//EN
- -//W3C//DTD XHTML 1.0 Transitional//EN
- -//W3C//DTD XHTML 1.0 Strict//EN
- -//W3C//DTD XHTML 1.1//EN

4.4.7 XSLTスタイルシートの適用

XSLTスタイルシートはxml-stylesheet処理命令により適用されます。詳細はxml-stylesheet処理命令の節を参照してください。

4.4.8 注釈

cssj:annot属性は、Copper PDFを利用するアプリケーションに、メッセージを送るものです。例えば、日付順に内容が並んでいる文書を処理する場合に、cssj:annot属性に日付を入れることで、ドキュメント中でCopper PDFが現在処理中の部分をアプリケーションが知ることができます。

4.4.9 XML中でHTMLの要素と属性を使用する

名前空間 http://www.w3.org/1999/xhtml を使用することで、画像(img要素)、テーブルセルの結合(colspan, rowspan属性)といったHTMLの一部の要素や属性はXML内でも使用することができます。

利用可能な要素や属性のリストは資料集のXMLの拡張の章を参照してください。

4.5 画像

4.5.1 Copper PDFがサポートする画像

Copper PDFはJPEG, GIF, PNG, SVG形式の画像を標準でサポートしています。各画像は通常のブラウザ同様にimg要素によってドキュメント中に含めることができます。また、object, embed要素の使用もサポートされています。

CSSの'background-image'プロパティ、あるいは'content'プロパティによっても画像の表示が可能です。

4.5.2 他の画像形式の利用

より多種多様な画像フォーマットを使用する場合はサンマイクロシステムズ社により配布されている Java Advanced Imaging Image I/O Tools(JAI) をCopper PDFを動作させるJava環境に管理者がインストールする必要があります。JAIをインストールすることにより、BMP, JPEG 2000, PNM, TIFF, WBMPの各画像フォーマットが利用可能になります。また、JAI以外にもJava Image I/Oに対応した拡張ライブラリをインストールすることで利用可能な画像を追加することができます。

4.5.3 ラスター(ビットマップ/ピクセルマップ)画像

GIF / PNG画像

GIFおよびPNG画像の透明化効果はPDF内でも有効です。ただし、PDF 1.3以前のバージョンのPDFを出力する場合、PNG画像の半透明化効果が正確に表現されず、透明・不透明だけとなります。

アニメーションGIFの場合、最初のコマだけが使われ、アニメーションしません。

JPEG / JPEG 2000画像

JPEG画像は処理を加えずに、そのままPDFに含めることができます。入出力プロパティ output.pdf.jpeg-image にto-flateを設定することで画像を展開してからPDFに含むこともできますが、PDFのサイズは大きくなります。

JAIがインストールされている場合は、JPEG 2000にも同様の処理が適用されます。JAIがない場合はJPEG 2000は全く利用できませんのでご注意ください。

その他の画像

JAI等のライブラリで他の画像形式が利用可能な場合、それらの扱いはGIF /PNG画像に準じます。半透明効果をサポートする画像形式の場合はPDF内でも再現され、アニメーションする画像では最初のコマだけが使われます。

4.5.4 SVG画像

SVG画像ファイルの参照

Copper PDFはSVG画像ファイル(.svg)またはGZIPで圧縮されたSVG画像ファイル(.svgz)をサポートしています。ただし、**グラデーション塗りとSVG埋め込みフォントには対応していません。**

一般的なブラウザではobject要素によりSVG画像サポートされていますので、Copper PDFでも同様の方法を推奨します。

例 4.21 object要素によるSVG画像の埋め込み

```
<object data="image.svg" type="image/svg+xml">
SVGをサポートしないブラウザで表示されるテキスト。
</object>
```

Copper PDFでは、SVG画像を他の画像と同様に扱うことができます。img要素を利用することができ、'background-image'プロパティにより背景に設定することもできます。

インラインSVG

SVGを別ファイルに分けずに、文章中に直接記述することができます。グラフ等、動的に変化する画像をドキュメントに含める場合に適しています。インラインSVGは、ドキュメント中でhttp://www.w3.org/2000/svg と名前空間宣言したSVGを記述するだけです。

例 4.22 インラインSVG

```
<html>
<head>
  <title>インラインSVGを含むドキュメント</title>
</head>
<body>
<p>すぐ下に○が表示されます。</p>
<svg:svg xmlns:svg="http://www.w3.org/2000/svg"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         preserveAspectRatio="none"
         width="100" height="100"
         viewBox="0 0 100 100"
         xml:space="preserve">
    <svg:g>
        <svg:circle cx="50" cy="50" r="45" stroke="Blue"
fill="White" stroke-width="10"/>
    </svg:g>
</svg:svg>
</body>
</html>
```

4.5.5 画像を読み込めない場合

画像ファイルが存在しない、データの破損、Copper PDFがサポートしない画像形式といった原因で 画像を読み込めない場合は、通常はalt属性で指定された代替テキストが表示されます。入出力プロパティ output.broken-image の設定により、画像の形に×印を表示するか、空白を空けることができます。

4.6 ページ処理機能

4.6.1 ページのレイアウト

生成されるページは以下の部分から構成されています。

用紙
ドキュメントが印刷される用紙です。

印刷面
ドキュメントの内容が印刷される部分です。

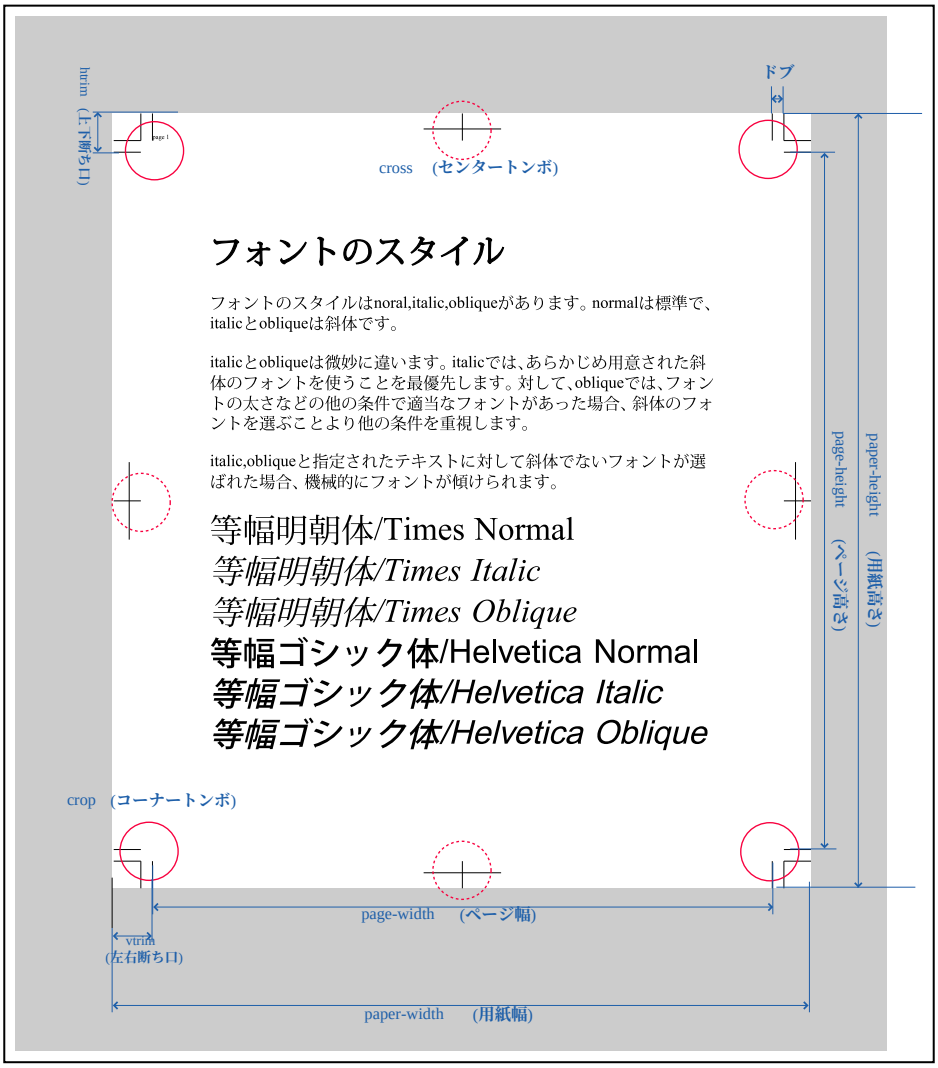
マージン
通常は内容がみだすことのない、ページの余白部分です。

トンボ
製本する際、断裁の目印となる印です。

裁ち口
断裁されて切り落とされる部分です。

ドブ
断裁される可能性のある範囲です。

図 4.8 ページのレイアウト



ページのレイアウトは入出力プロパティにより設定されます。対応する入出力プロパティは次の通りです。

用紙のサイズ

output.paper-width, output.paper-height

印刷面のサイズ

output.page-width, output.page-height

マージン

output.page-margins

トンボ

output.marks

裁ち口

output.htrim, output.vtrim

用紙のサイズの指定がない場合、用紙の幅と高さは、それぞれページの幅と高さに断ち口の幅を足したものに自動的に設定されます。用紙のサイズを指定した場合、output.fit-to-paper がtrueの場合は用紙に合わせて内容が拡大され、falseの場合は中央に寄せられます。

output.page-margins の設定はページマージンのデフォルト値として使用されるものです。ページのマージンは、CSS の 'margin-top', 'margin-right', 'margin-bottom', 'margin-left' プロパティにより@pageルール内で上書きすることができます。

Copper PDFのデフォルトの設定では、ページの高さは297mm、ページの幅は210mm(A4サイズ)です。上下左右のマージンはいずれも12.7mm(3pc)です。

用紙の縦横の長さはいずれも1ptから14400pt(5080mm)の 間である必要があります。用紙サイズがこの範囲を超えた場合は、超えた部分は切り落とされます。また、ページの高さは

4.6.2 片面印刷と両面印刷

デフォルトではCopper PDFによるページ生成は横書き・両面印刷として行われます。従って、CSSの@pageルールにおいて、最初のページは:firstまたは:right擬似要素として扱われ、以降は:left, :right擬似要素のページが交互に現れます。

output.print-mode にsingle-sideを設定することにより、片面印刷に切り替えることができます。片面印刷では最初のページは:first擬似要素として扱われ、以降はどの擬似要素にも属さないページが生成されます。

4.6.3 ページごとに生成されるコンテンツ

Copper PDFは「ノンブル」のようなページごとに生成されるコンテンツをサポートするために、独自の'-cssj-page-content'プロパティを用意しています。

'-cssj-page-content'は要素をページ毎に生成するボックスとして配置するものです。ボックスの配置方法は{position: fixed;}と同じですが、ページ毎にボックスが再生成され、'content'プロパティが処理される点が異なります。

'-cssj-page-content' 最初の値は、生成されるコンテンツの識別名です。2つめの以降の値は、コンテンツを最初のどちらのページに再生成するかの指定です。first(最初), left(左ページ), right(右ページ), single(片面印刷のページ)のいずれかです。2つめの値を省略すると、全てのページで再生成されます。2つめ以降に複数の値を指定すると、いずれかの条件にマッチする場合にページを再生成します。例えばleft singleでは両面印刷の左か、片面印刷のページで再生成します。

識別名は再生成の停止や置き換えのためのものです。'-cssj-page-content-clear' により、ボックスの再生成を止めることができます。また、'-cssj-page-content'で指定した名前が既に使われている場合、古いボックスの再生成を止め、新しいボックスに置き換えられます。

以下は再生成ボックスを利用して、ページごとに「ノンブル」を振る例です。各ページの下部中央にページ番号が表示されます。

例 4.23 ページ番号の生成

```
<style type="text/css">
  @page {
    margin: 2cm 2cm 5cm 2cm;
    counter-increment: page;
  }
  #page-number {
    -cssj-page-content: footer;
    bottom: -1cm;
    text-align: center;
    width: 100%;
  }
  #page-number:before {
    content: counter(page);
  }
</style>
<div id="page-number"></div>
... 本文 ...
```

左右のページでアラインメントを変えることにより、ページの外側にノンブルを振ることができます。また、-cssj-heading関数を利用することで、現在ページのセクションの見出しを一緒に表示することができます。

例 4.24 見出しの表示

```
<style type="text/css">
  @page {
    margin: 2cm 2cm 5cm 2cm;
    counter-increment: page;
  }
  #nombre-left, #nombre-right {
    bottom: -1cm;
    width: 100%;
  }
  #nombre-left {
    -cssj-page-content: nombre-left left;
    text-align: left;
  }
  #nombre-right {
    -cssj-page-content: nombre-right right;
    text-align: right;
  }
  #nombre-left:before {
    content:  counter(page) ' - ' -cssj-heading(1);
  }
  #nombre-right:after {
    content: -cssj-heading(2) ' - ' counter(page);
  }
</style>
<div id="nombre-left"></div>
<div id="nombre-right"></div>
... 本文 ...
```

CSSJ 1.x系の'-cssj-regeneratable'も互換性のためサポートされていますが、パフォーマンスの面から'-cssj-page-content'の使用を推奨します。

4.7 改ページ制御

4.7.1 用語の定義

絶対配置ボックス
{position: absolute;}が指定された要素です。{position: fixed;}が指定された要素やページごとに生成されるコンテンツも 広義の絶対配置ボックスです。

浮動ボックス
{float: left;}または{float: right;} が指定された要素です。

通常のフローのブロック
何も指定されていない<p>要素や 要素や{display: block;}が 指定された要素で絶対配置ボックスでも浮動ボックスでもないものです。

テーブルは次の部分からなっています。

テーブルキャプション
HTMLのcaptionタグ、あるいはdisplayがtable-captionと指定された部分。

テーブルヘッダ
HTMLのtheadタグ、あるいはdisplayがtable-header-groupと指定された部分。

テーブルフッタ
HTMLのtfootタグ、あるいはdisplayがtable-footer-groupと指定された部分。

テーブル行グループ
HTMLのtbodyタグ、あるいはdisplayがtable-row-groupと指定された部分。ただし、tbodyやtable-row-groupを省略して、テーブルの中に直接存在する行も行グループに属すると見なされます。

4.7.2 強制改ページ

強制改ページは、指定した場所で強制的に改ページを発生させる機能です。強制改ページを指定できるのは次の場所です。

- 通常のフローのブロックの直前
- 通常のフローのブロックの直後
- 浮動ボックスの直前
- 浮動ボックスの直後
- テーブルの直前
- テーブルの直後
- テーブル行の直前
- テーブル行の直後
- テーブルセルの直前
- テーブルセルの直後

ただし、上記の場所であっても浮動ボックス内、絶対配置ボックス内、テーブルセル内では強制改ページを発生することはできません。

要素の直前の強制改ページの指定は{page-break-before: always;}です。要素の直後の強制改ページの指定は{page-break-after: always;}です。

以下は強制改ページを使って表紙を作る例です。

例 4.25 強制改ページ

```
<html>
  <head>
    <title>ドキュメント</title>
  </head>
  <body>
    <h1 style="page-break-after: always;">表紙</h1>
    <p>本文...</p>
  </body>
</html>
```

また、単純に改ページするためではなく、改ページした直後のページが右になるか、左になるかを指定することができます。この場合、調整のために空白のページがつくられる可能性があります。

強制改ページの後のページが右になるか、左になるかを指定するには、page-break-beforeおよびpage-break-afterプロパティの値として、alwaysの代わりに left(左ページにする場合)またはright(右ページにする場合)を指定します。

以下の例では、必ず右側になる中表紙を生成しています。

例 4.26 空ページが生じるケース

```
<html>
  <head>
    <title>ドキュメント</title>
  </head>
  <body>
    <h1 style="page-break-after: always;">表紙</h1>
    <p style="page-break-after: always;">本文1...</p>
    <p>本文2...</p>
    <h1 style="page-break-before: right;">中表紙</h1>
  </body>
</html>
```

4.7.3 'orphans'と'windows'

'orphans'と'windows'プロパティは、段落(ここでは通常のフローのブロックを指し、
による空行等は段落の区切りとは認識されません) の途中で改ページが発生する場合、必ず前のページに残す行数と、後のページに表示される行数を指定するものです。

'orphans'

ある段落がページの下端にかかっている場合、段落を途中で分割して改ページする必要があります。'orphans'はそのような場合に、改ページされる前のページに最低限残さなければならない行数です。例えば、以下の例では'orphans'が3に対して 改ページ前のページに3行があるので、条件を満たしています。

例 4.27 orphansが3の場合

(段落1)1行目... 2行目... 3行目... 4行目... 5行目... 6行目... 7行目... 8行目... (段落2)1行目... 2行目... 3行目...	4行目... 5行目...
1ページ目	**2ページ目**

同じ文書で'orphans'を4に指定すると、そのままでは'orphans'を満たすことができないため、段落をまるごと次ページに移動してしまいます。

例 4.28 orphansが4の場合

(段落1)1行目... 2行目... 3行目... 4行目... 5行目... 6行目... 7行目... 8行目...	(段落2)1行目... 2行目... 3行目... 4行目... 5行目...
1ページ目	**2ページ目**

'windows'

文書の内容の高さがページの高さよりわずかに高い場合、次のページに文書の内容のうち何行かを先送りしなければなりません。'windows'は改ページされた後のページに最低限表示されなければならない行数で、Copper PDFは'windows'を満たすように先送りする行数を調整します。例えば、'windows'が2の場合、以下の例では2ページ目に2行存在するので条件を満たしています。

例 4.29 windowsが2の場合

(段落1)1行目... 2行目... 3行目... 4行目... 5行目... 6行目... 7行目... 8行目... (段落2)1行目... 2行目... 3行目...	4行目... 5行目...
1ページ目	**2ページ目**

同じ文書で'windows'を3に指定すると、以下のように前のページから次のページへ行を移動して、'windows'を満たすようにします。

例 4.30 windowsが3の場合

(段落1)1行目... 2行目... 3行目... 4行目... 5行目... 6行目... 7行目... 8行目... (段落2)1行目... 2行目...	3行目... 4行目... 5行目...
1ページ目	**2ページ目**

'orphans'と'windows'の競合

'orphans'と'windows'の両方の条件を同時に満たすことができない場合も、'orphans'を満たせなかった場合と同様に段落を丸ごと次ページに移動します。

例えば、以下の状況では'orphans'と'windows'の両方が満たされています。

例 4.31 orphansが3でwindowsが2の場合

(段落1)1行目... 2行目... 3行目... 4行目... 5行目... 6行目... 7行目... 8行目... (段落2)1行目... 2行目... 3行目...	4行目... 5行目...
1ページ目	**2ページ目**

この状態で'windows'を3に設定すると、'orphans'は満たせるが'windows'は 満たせない状態になります。

例 4.32 orphansが3でwindowsが3の場合

(段落1)1行目... 2行目... 3行目... 4行目... 5行目... 6行目... 7行目... 8行目...	(段落2)1行目... 2行目... 3行目... 4行目... 5行目...
1ページ目	**2ページ目**

4.7.4 改ページの抑制

次の場所には、改ページの抑制を指定することができます。

- 通常のフローのブロックの内部
- 通常のフローのブロックの直前
- 通常のフローのブロックの直後
- 浮動ボックスの内部
- テーブルの内部
- テーブルの直前
- テーブルの直後
- テーブル行の内部
- テーブル行の直前
- テーブル行の直後
- テーブルセルの内部
- テーブルセルの直前
- テーブルセルの直後

内部の改ページ抑制

内部での改ページを抑制するには、{page-break-inside: avoid;}という指定をします。

改ページ抑制されたボックスがページをはみ出す場合は、ボックスが丸ごと次のページの先頭に先送りされます。ただし、ボックスが(先送りの結果か、元々そこにあるかに関わらず)ページの先頭にある場合、かつボックスの高さがページの高さを超えてしまう場合は、改ページの抑制を無視してページ分割されます。

前後の改ページ抑制

ボックスの前後での改ページを抑制するには、{page-break-before: avoid;}(ボックスの前)あるいは{page-break-after: avoid;}(ボックスの後)を指定します。Copper PDFは改ページが抑制された箇所での改ページを避け、前後の何行かを必ず1つのページに含めるようにします。改ページが抑制されたポイントの前に入れる行数は'orphans'に依存します。

強制改ページと自動改ページが競合する場合は、強制改ページが優先されます。。例えば、{page-break-after: always;}と指定された段落の直後に、{page-break-before: avoid;}と指定された段落がある場合です。このような競合が起こった場合、常に強制改ページが優先されます。つまり、このケースでは{page-break-before: avoid;}は無視されて改ページが発生します。

なお、HTMLのh1～h6要素にはデフォルトで{page-break-before: avoid;}が指定されています。

4.7.5 自動改ページ

Copper PDFは、文書の内容がページの下端にさしかかった部分で、自動的に改ページします。自動改ページが発生するのは次の場所です。

- 通常のフローのブロックの内部
- 通常のフローのブロックの間
- 浮動ボックスの内部
- テーブル行の間
- テーブル行の内部

逆に、以下の場所では絶対に改ページされることはありません。

- 行の内部
- 画像の内部
- テーブルキャプションの内部
- テーブルキャプションとテーブルの間
- テーブルヘッダの内部
- テーブルフッタの内部
- テーブルヘッダと行グループの間
- テーブルフッタと行グループの間

通常のフローのブロック

通常のフローでは 'orphans', 'windows' を尊重して改ページが行われます。ブロックに境界線がある場合に境界線の直後、あるいは高さが指定されていて内容がないブロックの内部では なるべく改ページを避けますが、ブロックがページの先頭ある場合は改ページが発生します。

浮動ボックス

浮動ボックスがページの下端をはみ出した場合、浮動ボックスは分割され、次ページに送られた部分は浮動ボックスとして再配置されます。浮動ボックスの分割でも、'orphans', 'windows' の指定は尊重されますが、条件を満たせない場合であっても浮動ボックスを丸ごと次ページに送られることはなく、その場合は'orphans', 'windows'を無視して分割されます。

画像および{page-break-inside: avoid;}が指定された浮動ボックスは分割されることはなく、ページの下端をはみ出した場合は丸ごと次のページに持ち越されます。ただし、浮動ボックスの中に入れ子になった浮動ボックスでは{page-break-inside: avoid;}は無効です。

4.8 テーブル内での改ページ

Copper PDFはテーブルの行間、テーブルの行の途中(セルの途中)での改ページが可能です。また、テーブルのヘッダとテーブルのフッタは各ページで繰り返し表示されます。

4.8.1 改ページされない場所

テーブル内の次の場所では、いかなる状況であっても改ページされることはありません。

- テーブルのキャプション内
- テーブルのキャプションとテーブルの間
- テーブルのヘッダ内
- テーブルのフッタ内
- テーブルのヘッダと行グループの間
- テーブルのフッタと行グループの間

従って、上記の部分に指定された'page-break-after', 'page-break-before', 'page-break-inside' は無視されます。また、上記の規則が適用された結果、テーブルの行グループが存在する限り、どのように改ページが発生する場合も、テーブルの行グループの一部が常に表示され、ヘッダかフッタだけのテーブルが現れることはありません。上記の部分がページの高さを超える場合は、内容がページをはみ出します。

4.8.2 page-break-XXXの適用

改ページに関する特性は行の間には通常のフローのブロックと同様に適用されます。改ページに関する特性の指定は行グループの間にも適用されます。ただし、強制改ページでleft, rightを指定した場合の効果は、alwaysと同じになります。

セルに対する'page-break-inside'は、行に適用されます。同じ行内でautoとavoidが指定されたセルが競合する場合、avoidが優先されます。すなわち、行に属するセルのうち１つでもavoidが指定された場合、行全体の中での改ページが禁止されます。また、セルがrowspanで連結されている場合、セルが属する全ての行の内部で改ページが抑制されるのに加えて、行の間で 'page-break-after'および'page-break-before'にavoidが指定されたものと見なされます。

セルに対する'page-break-after'および 'page-break-before'は、行に適用されます。この場合の優先順位は次の順になります。

always > avoid > auto

4.8.3 テーブル行内部(セル内部)での改ページ

テーブル行内に、{page-break-inside: avoid;}が指定されたセルがなく、テーブル行がページの下端に差し掛かっている場合、そのテーブル行の分割が試みられます。このとき、'orphans'と'windows'が尊重され、行に属する全てのセルが分割不可能な場合は、行全体が次のページに先送りされます。１つでも分割可能なセルがあった場合は、行が分割されます。この場合、他のセルに対しては'orphans'と'windows' を無視した分割が起こる可能性があります。



分割されたセルに対しては'vertical-align'による垂直アラインメントの指定が無効となり、セルの内容は全てセルの上端につけられます。

4.8.4 デフォルトの改ページ禁止

HTMLのtd,th要素は、デフォルトで {page-break-inside: avoid;} が設定されています。テーブルセル内の改ページを有効にするには、HTMLのtd, th要素に対して明示的に {page-break-inside:: auto;} を指定する必要があります。(td, th以外の要素に対して {display: table-cell;} を指定したことによるテーブルセルは、この限りではありません。)

例 4.33 テーブルセル内での改ページを有効にするCSS

```
td, th {
  page-break-inside: auto;
}
```

上のスタイルシートをデフォルトのスタイルシートとして指定しておけば、あらゆるページでテーブルセル内での改ページがされるようになります。

4.9 PDFの機能

4.9.1 PDFのバージョンと機能

出力するPDFのバージョンは output.pdf.version で設定することができます。デフォルトではバージョン1.5のPDFが出力されます。PDF 1.5以降であれば、Copper PDFの全ての機能を利用することができます。

PDF 1.4以前では、使用できる機能に制限があります。詳細は資料集のoutput.pdf.versionの説明を参照してください。

4.9.2 暗号化

Copper PDFは暗号化したPDFを出力することができます。パスワードを設定して暗号化したPDFは、パスワードがなければ閲覧不可能になります。また、内容のコピーなどPDFの利用制限を設定するためにも暗号化が必要で、広く配布するPDFのためにパスワードを設定せず、暗号化だけしたPDFを出力することができます。

暗号化は、output.pdf.encryption の設定で有効となります。暗号化方式はv1とv2の2種類がありますが、通常はPDF 1.3以降でサポートされているv2暗号化を使用してください。

閲覧のためのパスワードは output.pdf.encryption.user-password で設定することができます。このプロパティを設定しなかった場合は、暗号化だけされて誰でも開くことができるPDFが生成されます。さらに output.pdf.encryption.owner-password でAdobe Acrobat等で文書の編集をするためのパスワードを設定することができます。

PDFを暗号化する場合は、パーミッションを設定することができます。パーミッションはoutput.pdf.encryption.permissionsで開始する8種類の入出力プロパティで設定することができます。詳細は 資料集 (98ページ) を参照してください。

> **パーミッションを設定することにより、Adobe Reader等のPDF閲覧ソフトは設定に沿った動作をしますが、他のツール等でPDFに対する該当する操作が行われないことを保証するものではありません。**

4.9.3 ファイルの添付

PDF 1.4以降では、PDFにファイルを添付することができます。ファイルの添付はoutput.pdf.attachments で始まる4つのプロパティを使います。

詳細は資料集のoutput.pdf.attachmentsの説明を (97ページ) 参照してください。

4.10 一般的なブラウザとの互換性

4.10.1 互換性モードの切り替え

Copper PDFは、デフォルト(標準モード)ではCSS 2.1に準拠したレンダリングをしますが、実用上の問題から一般的なブラウザとの互換モードが用意されています。output.compatible_mode に'msie'を指定することにより、Copper PDFはInternet Explorer 7に近いレイアウトをするように試みます。このモードはCopper PDFのバージョンアップごとに一般的なブラウザと互換性を持つように改善が試みられますが、完全な互換性を保証するものではありません。また、制限事項があり、デフォルトのモードより非効率な処理が行われることがあるため、若干パフォーマンスが劣ります。

4.10.2 標準モードと'msie'モードの違い

CSSで数字のクラス名が認識される

標準モードでは数字のクラス名はエラーとなり無視されますが、'msie'モードでは有効です。

例 4.34 数字のクラス名を使用する

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>数字のクラス名</title>
    <style type="text/css">
.A {
    color: Red;
}
.1 {
    color: Red;
}
    </style>
  </head>

  <body>
    <p class="a">このテキストは赤字です</p>
    <p class="1">'msie'モードではここも赤くなります</p>
  </body>
</html>
```

CSSで':'の代わりに'='が使用できる

CSSでプロパティ名と値の区切り文字は':'ですが、'msie'モードでは'='も使用することができます。

例 4.35 ':'の代わりに'='を使用する

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>':'の代わりに'='を使用する</title>
    <style type="text/css">
.A {
    color: Red;
}
.B {
    color=Red;
}
    </style>
  </head>

  <body>
    <p class="a">このテキストは赤字です</p>
    <p class="b">'msie'モードではここも赤くなります</p>
  </body>
</html>
```

CSSの色指定で#を省略しても認識される

'msie'モードではRGBコードで色を指定する場合、#を省略できます。

例 4.36 色指定で#を省略する

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>色指定で#を省略する</title>
    <style type="text/css">
.A {
    color: #FF0000;
}
.B {
    color: FF0000;
}
    </style>
  </head>

  <body>
    <p class="a">このテキストは赤字です</p>
    <p class="b">'msie'モードではここも赤くなります</p>
  </body>
</html>
```

CSSの長さ指定で単位を省略しても認識される

'msie'モードでは長さ指定で単位を省略した場合、px単位による指定と認識されます。

フォームの前後にマージンが設定される

'msie'モードでは、デフォルトでフォーム(form要素)の前後に1.12emのマージンが設定されます。

段落と見出しのマージン上下のマージンをなくす場合がある

'msie'モードでは、段落または見出しが(p, h1～h6要素)の、ページまたはテーブルセルの先頭にある場合、前のマージンがなくなります。同様に末尾にある場合は後のマージンがなくなります。

ボックスの幅または高さに100%が指定された場合、外側のボックスをはみ出さない

'width'または'height'のパーセント指定は、標準モードでは外側のボックスの幅に対して、ボックスの内側の幅を指定するため、境界のあるボックスに対して100%を指定すると、ボックスが外側のボックスの内部からはみ出します。'msie'モードでは100%指定された場合にボックスの外側の幅が外側のボックスの内側の幅に一致し、0%指定された場合にボックスの内側の幅がゼロになるように計算します。

絶対位置指定ボックス、浮動ボックスの大きさが内容により拡張される

絶対位置指定ボックス、浮動ボックスに{overflow: visible;}が指定され、かつボックスの幅と高さが設定されている場合、標準モードではボックスのサイズは常に固定で、内容がはみ出した部分はボックスの境界にかかりますが、'msie'モードではボックスの内容によってボックスの幅と高さが拡張されます。

通常のフローのボックスが、幅が指定されたボックスにより拡張される

幅指定されたボックスに、より大きな幅のボックスが含まれている場合、標準モードでは内部のボックスは外側のボックスをはみ出しますが、'msie'モードでは外側のボックスの幅が、内側のボックスが収まるように拡張されます。

幅がautoの固定レイアウトテーブルが固定レイアウトのまま処理される

標準モードでは、{table-layout: fixed;}が指定されているテーブルであっても、'width'がautoであれば{table-layout: auto;} が指定されたのと同様に扱われます。

'msie'モードでは、'width'がautoの場合は、テーブルの外側の幅を、外側のボックスの内側の幅に合わせて固定し、固定レイアウトとして処理します。

テーブル行に対する'max-height'が適用されない

'msie'モードではテーブル行に対する'max-height'の設定は無視されます。

ブロックに対する'line-height'による高さが確保されない

ブロックに対して'line-height'が指定されている場合、標準モードではブロック内の行に対して指定した高さが必ず確保されますが、'msie'モードでは行内にインライン

画像、インラインボックスまたはインラインテーブルだけが含まれ、かつそれらの高さが'line-height'より小さい場合は、'line-height'で指定された高さは確保されません。

全角スペースの間で折り返しされない

'msie'モードでは、全角スペースが連続している場合、その中での折り返しは発生しません。また、全角スペースにより外側のボックスが拡張されることはありません。

input要素の高さが強制される

'msie'モードでは、input要素で配置されるフォームの高さが1.12emに強制されます。'height'は適用されません。

マージンの設定に関わらず中央寄せされる

img, input, applet, object, iframe, tableのalign属性による中央寄せは、標準モードでは左右のマージンをautoにした状態{margin: 0 auto;}と同等ですが、'msie'モードでは左右のマージンが設定された場合も中央寄せになります。

例えば、標準モードでは{margin-left: 100pt;}が指定されれば、中央寄せが設定された要素でも常に左から100ptの位置に配置されますが、'msie'モードでは中央寄せから左に100ptの1/2(つまり50pt)ずれた位置に配置されます。

匿名のテーブルセルにより補完されない

テーブル内で、かつテーブルセルの外にインライン、インラインブロックまたはインラインテーブルが配置された場合、標準モードでは自動的に上位に匿名のセル要素が挿入されますが、'msie'モードでは全てテーブルの前に表示されます。

インラインボックスに'width'が適用される

標準モードでは'width'はインラインボックスには適用されませんが、'msie'モードでは'width'の設定によりインラインボックスの幅が変化します。

また、'width'がauto以外に設定されたインラインボックスはインラインボックスと同様に配置されます。すなわち、{display: inline;}が指定されている要素であっても、'width'がauto以外であれば {display: inline-block;}が適用されているのと同じことになります。

'text-align'がブロックの配置にも適用される

標準モードでは'text-align'はテキストの配置にだけ適用されますが、'msie'モードでは内部にある通常のフローのボックスの配置にも適用されます。これはp要素やdiv要素等のalign属性と同様の挙動をします。

テーブルカラムのプロパティがセルに継承される

'msie'モードではテーブルカラムに設定された以下のプロパティがテーブルセルに継承されます。

- text-align
- color
- font-style
- font-weight
- font-family
- font-size
- text-transform
- letter-spacing
- word-spacing

上記のプロパティがテーブルセルと、テーブル行の両方に設定されている場合は、テーブル行のプロパティの方が継承されます。

'width'が設定されたテーブルセルには{white-space: nowrap;}が適用されない

標準モードでは{white-space: nowrap;}が指定された領域ではテキストの折り返しが行われないため、'width'で幅が指定されたテーブルセルでは、内容が幅をはみ出すことがありますが、'msie'モードでは'width'で幅が指定されたテーブルセルでは{white-space: nowrap;}が適用されず、内容がはみ出さないようにテキストの折り返しが発生します。

4.10.3 既知の制限事項

ＭＳ明朝系フォントの文字幅について

ＭＳ明朝、ＭＳゴシックのフォントを使用する場合、一般的なブラウザでは小さなフォントサイズが指定された場合、一番近いビットマップフォントが使用されるため、指定された文字サイズと実際に使われる文字サイズが異なることがありますが、Copper PDFでは全てアウトラインフォントを使用するため、指定したとおりの文字サイズとなります。そのため、文字幅が一般的なブラウザで表示する場合と異なり、文字列の折り返し位置等が異なることがあります。

サポートしていないCSSプロパティ

Internet Explorerで縦書きを行うための'writing-mode' プロパティはサポートしていません。

5.資料集

5.1 入出力プロパティ

以下は、各種プログラミング言語からCSSJにアクセスする際に設定できるプロパティの一覧です。プロパティの設定方法の詳細は開発マニュアルをご参照ください。

表 5.1 入力関連プロパティ

名前	デフォルト	バージョン	説明
input.default-encoding	JISAutoDetect	1.0.0	HTMLのMETA 要素でキャラクタ・エンコーディングを判断できない場合に使用するキャラクタ・エンコーディング名です。。
input.default-stylesheet	-	1.0.0	デフォルトのCSSスタイルシートのURIです。このプロパティが指定されている場合、最初にデフォルトのスタイルシートが読み込まれます。
input.filters	xslt default-to-xhtml loose-html	1.0.0	入力HTML(XML)のフィルタリングです。適用される順にフィルタ名をスペース区切りで並べます。以下のフィルタ名が用意されています。 **xslt** <xml-stylesheet...処理命令で指定されているXSL変換スタイルシートを適用します。 **default-to-xhtml** XMLで名前空間が指定されていない要素をXHTML名前空間に変換します。 **loose-html** 一般的なHTMLを解釈できるようにするためのフィルタです。
input.property-pi	false	2.0.0	true を設定するとドキュメント中でjp.cssj.property-pi処理命令を使うことができるようになります。
input.stylesheet.titles	-	1.0.0	適用するCSSスタイルシートのタイトルをスペース区切りで並べます。link要素またはxml-stylesheet処理命令で関連付けられたスタイルシートについて、デフォルトでは代替スタイル以外が全て適用されますが、このプロパティを用いて適用するスタイルシートを指定することができます。

名前	デフォルト	バージョン	説明
input.xslt.default-stylesheet	-	1.2.0	デフォルトのXSLTスタイルシートのURIです。このプロパティが指定されている場合、最初にデフォルトのスタイルシートが読み込まれます。input.filtersにxsltフィルタが存在するとき場合のみ有効です。

表 5.2 HTTPアクセス関連プロパティ

名前	デフォルト	バージョン	説明
input.http.referer	true	1.0.1	HTTP通信でサーバー側のデータを取得するときにRefererヘッダを送るかどうかの指定です。trueまたはfalseで指定します。 falseを指定すると、Refererを用いて画像などのリソースへの直接アクセスを規制しているサイトでリソースにアクセスできなくなります。
input.http.proxy.host	-	1.2.6	プロクシのホスト名です。 この設定するとHTTP通信でプロクシを用います。
input.http.proxy.port	8080	1.2.6	プロクシを使う際のポート番号です。 この設定はinput.http.proxy.hostが設定されている場合のみ有効です。
input.http.header.*n*.name input.http.header.*n*.value	-	2.0.0	HTTP接続で送信するヘッダです。 nは0から始まる通し番号で、nが同じ2つのプロパティで一組です。 nameはヘッダ名でvalueはヘッダの値です。 通し番号は0から開始してカウントしていき、必要な情報(name)が欠けていた時点で以降のパラメータは無効となります。
input.http.authentication.preemptive	false	1.2.6	HTTP通信で認証を行う場合に、最初から認証情報を送るかどうかの設定です。trueまたはfalseで指定します。 trueを指定すると、Authorizationヘッダ等の認証情報を最初の接続で送ります。 falseを指定すると、最初にサーバーから401レスポンスを受け取ってレルムや認証スキーマ等の情報を取得します。 trueを指定した場合、Digest認証や複数のレルムが存在するサーバーで認証が行われなくなります。

名前	デフォルト	バージョン	説明
input.http.proxy.authentication.user input.http.proxy.authentication.password	-	1.2.6	認証が必要なプロクシサーバーでの認証情報(user,password)です。この設定はinput.http.proxy.hostが設定されている場合のみ有効です。
input.http.authentication.*n*.host input.http.authentication.*n*.user input.http.authentication.*n*.password input.http.authentication.*n*.port input.http.authentication.*n*.realm input.http.authentication.*n*.schema	-	1.2.6	認証が必要なサイトでの認証情報です。 nは通し番号で、nが同じ4つのプロパティで一組です。hostは対象となるホストで、user,passwordが認証に用いられるユーザー名とパスワードです。 portはポート番号、realmは認証領域のレルム、schemaは認証スキーマ(basicまたはdigest)です。 port,realm,schemaは省略可能で、省略した場合はそれぞれ全てのポート、レルム、スキーマでの認証が行われます。 通し番号は0から開始してカウントしていき、必要な情報(hostおよびuser)が欠けていた時点で以降のパラメータは無効となります。
input.http.cookie.*n*.domain input.http.cookie.*n*.name input.http.cookie.*n*.value input.http.cookie.*n*.path	-	1.2.6	送信するクッキーです。nは通し番号で、nが同じ4つのプロパティで一組です。domain,name,valueはそれぞれクッキーのドメインと名前、値です。pathはクッキーのパスで、省略した場合はルート(/)となります。 通し番号は0から開始してカウントしていき、必要な情報(domainおよびname)が欠けていた時点で以降のパラメータは無効となります。

表 5.3 出力関連プロパティ

名前	デフォルト	バージョン	説明
output.auto-height	false	1.0.0	自動高さの指定です。falseまたはtrueで指定します。 trueにすると、自動改ページをせず、ページの高さを文書の内容の高さに合わせます。このとき、output.page-heightプロパティは無効になります。 出力可能なページのサイズには制限があります。
output.broken-image	none	1.2.2 noneは2.0.0	壊れた画像の表示方法。none,hidden,crossで指定します。noneでは代替テキスト(alt属性の値)だけが挿入されます。hiddenではwidth, height属性による矩形の範囲が空白となります。crossでは、width, height属性による矩形の範囲に×印が表示されます。

名前	デフォルト	バージョン	説明
output.clip	true	2.0.3	trueに設定した場合、印刷面の外側(トンボのドブの外側、あるいはページの外側)を描画しません。falseに設定した場合、印刷面の外側を描画します。
output.color	rgb	1.2.1	出力結果のカラー・タイプです。rgb,grayで指定します。rgbではフルカラーで出力されます。grayでは、グレイスケールに変換されます。
output.compatible_mode	copper	2.0.0	レイアウトの互換モードです。copper,msieで指定します。msieを指定すると、Internet Explorer 7に近いレイアウトを再現します。
output.default-font-family	serif	2.0.0	デフォルトのフォントファミリです。ドキュメント中でフォントが指定されていない場合、あるいは該当するフォントが見つからない場合、このフォントを使用します。 CSSの'font-family'と同じ形式で複数のフォントを指定することができます。空白を含むフォント名はクウォート('または")で囲うことに注意してください。
output.fit-to-paper	false	2.0.0	trueに設定した場合、内容が用紙に合わせて拡大されます。falseに設定した場合、中央寄せされます。
output.marks	none	1.0.0 hiddenは1.2.1	トンボおよび裁ち口の表示です。none,crop,cross,both,hiddenのいずれかを指定します。 それぞれ、トンボ・裁ち口なし、コーナートンボを表示、センタートンボを表示、両方のトンボを表示、裁ち口だけを表示する、という意味になります。
output.media_types	all print paged visual bitmap static	2.0.0	適用するスタイルシートのメディアタイプです。 http://www.w3.org/TR/CSS21/media.html#media-types
output.meta.*n*.name output.meta.*n*.value	-	2.0.3	文書情報をあらかじめ設定します。nは0から始まる通し番号で、nが同じ2つのプロパティで一組です。 文書情報はドキュメント内の<meta name="名前" content="値">要素によって上書きされます。詳細は文書情報 (64ページ)の節を参照してください。
output.no-page-break	false	2.0.3	trueに設定すると改ページを全くしなくなります。output.auto-heightをtrueに設定するのと異なり、ページの高さを内容に合わせて拡大しません。

名前	デフォルト	バージョン	説明
output.page-height	297mm	1.0.0	ページの高さです。デフォルトはA4の高さです。 CSSの長さの単位(mm,cm,in,pt,pc,px)を使ってください。 出力可能なページのサイズには制限があります。この設定は文書中の@pageルール内で上書きできます。
output.page-limit	-	1.2.0	最大ページ数です。ページ数が限界に達すると、処理が中断されます。デフォルトでは無制限です。
output.page-margins	12.7mm	2.0.0	ページの余白です。CSSの'margin'プロパティと同じ形式で記述します。長さの単位は(mm,cm,in,pt,pc,px)が使用可能です。この設定は文書中の@pageルール内で上書きできます。
output.page-width	210mm	1.0.0	ページの幅です。デフォルトはA4の横幅です。 CSSの長さの単位(mm,cm,in,pt,pc,px)を使ってください。 出力可能なページのサイズには制限があります。この設定は文書中の@pageルール内で上書きできます。
output.paper-height	output.page-heightの値	2.0.0	用紙の高さです。デフォルトはページの高さです。用紙とページの大きさが異なる場合の動作は"output.fit-to-paper"の設定によります。 CSSの長さの単位(mm,cm,in,pt,pc,px)を使ってください。 出力可能なページのサイズには制限があります。
output.paper-width	output.paper-widthの値	2.0.0	用紙の幅です。デフォルトはページの横幅です。 用紙とページの大きさが異なる場合の動作は"output.fit-to-paper"の設定によります。 CSSの長さの単位(mm,cm,in,pt,pc,px)を使ってください。 出力可能なページのサイズには制限があります。
output.print-mode	double-side	2.0.0	印刷モードです。single-side, double-sideのいずれかを指定します。 single-sideでは片面印刷となり、@pageルールの:left, :right擬似クラスは適用されなくなります。
output.resolution	96	2.0.0	px単位の基準となる解像度です。 ppi(1インチあたりのピクセル数)を指定します。 一般的なブラウザでは96という値が使われます。72を指定すると1pt(PDFの基本単位)と1pxの長さが同じになります。

名前	デフォルト	バージョン	説明
output.image.resolution	72	2.0.4	"output.type"の設定によりラスター画像を出力する際の解像度(dpi)です。
output.size-limit	-	1.2.0	出力データの最大サイズ(バイト)です。サイズが限界に達すると、処理が中断されます。デフォルトでは無制限です。
output.htrim	1cm	2.0.0	左右の裁ち口の幅です。 CSSの長さの単位(mm,cm,in,pt,pc,px)を使ってください。
output.vtrim	1cm	2.0.0	上下の裁ち口の幅です。 CSSの長さの単位(mm,cm,in,pt,pc,px)を使ってください。
output.type	application/pdf	1.0.0	出力(MIME)形式です。"application/pdf"(PDFファイル)は必ず利用することができます。 Copper PDF 2.0.3から画像の出力に対応しました。画像の出力はJava Image I/Oに依存しており、Java実行環境がサポートする画像形式("image/png"など)を利用することができます。また、JAI等のプラグインをJava実行環境にインストールすることで、利用可能な画像形式を追加することができます。 画像の出力では最後のページだけが出力されます。また、コア14フォントとフォント設定ファイルのcid-keyed-font要素によるCID-Keyedフォントは正確に描画できません。

表 5.4 PDF出力関連プロパティ

名前	デフォルト	バージョン	説明
output.pdf.attachments.*n*.name output.pdf.attachments.*n*.description output.pdf.attachments.*n*.mime-type output.pdf.attachments.*n*.uri	なし	1.2.0 (PDF1.4)	添付ファイルです。 *n*は通し番号で、*n*が同じ4つのプロパティで一組です。nameはファイル名、descriptionはファイルについての説明で、省略可能です。mime-typeはファイルのMIME型で、省略可能です。uriはファイルの内容が置かれたURIで、省略できません。 通し番号は0から開始してカウントしていき、必要な情報(uri)が欠けていた時点でファイルの添付を終わります。nameにはASCII文字だけを使うことを推奨します(マルチバイト文字を含むことはできますが、表示環境によっては文字化けします)。マルチバイト文字が含まれる場合は、URLエンコードなどでASCIIに変換したものをnameに使い、実際のファイル名をdescriptionにセットしてください。

名前	デフォルト	バージョン	説明
output.pdf.bookmarks	false	1.0.0 (PDF1.2)	ブックマーク機能です。falseまたはtrueで指定します。 trueにすると、H1～H6要素をもとにブックマーク(アウトライン)を生成します。
output.pdf.compression	binary	1.0.0 (PDF1.2)	圧縮方法です。none,ascii,binaryで指定します。 後者ほど圧縮効率がよくなります。noneでは画像以外は圧縮せず、asciiでは画像以外の内容も圧縮されますが、生成されるPDFはテキストファイルとなります。binaryの場合、生成されるPDFは圧縮され、かつバイナリ形式となります。 ただし、暗号化を行う場合は、結果的に全てバイナリとなることに注意してください。
output.pdf.encryption	none	1.2.0 (PDF 1.2) (v2はPDF 1.3)	暗号化方式です。none,v1,v2で指定します。 noneは暗号化なし、v1は40ビットArcfour暗号、v2は40-128ビットArcfour暗号です。
output.pdf.encryption.length	128	1.2.0 (PDF1.3)	暗号化キーの長さ(ビット)です。 output.pdf.encryption=v1では40で固定です。 v2では40から128の間で、8ビット刻みで指定可能です。
output.pdf.encryption.user-password	空	1.2.0 (PDF1.2)	文書を開くためのパスワードです。このパスワードを使って文書を閲覧する場合は、文書に設定されたパーミッションによる制限がかかります。
output.pdf.encryption.owner-password	ユーザーのパスワード	1.2.0 (PDF1.2)	文書の権限を変更するためのパスワード(マスタパスワード)です。文書に対するあらゆる操作を可能にします。
output.pdf.encryption.permissions.print	true	1.2.0 (PDF1.2)	文書を印刷する権限です。 true=許可,false=禁止 です。
output.pdf.encryption.permissions.modify	true	1.2.0 (PDF1.2)	文書中の内容を変更をする権限です。 true=許可,false=禁止 です。
output.pdf.encryption.permissions.copy	true	1.2.0 (PDF1.2)	文書中のテキストや画像をコピーする権限です。 true=許可,false=禁止 です。
output.pdf.encryption.permissions.add	true	1.2.0 (PDF1.2)	注釈を追加・変更する、あるいはフォームに入力する権限です。 output.pdf.encryption.permissions.modify=trueであればフォームの追加・変更も許可されます。 true=許可,false=禁止 です。
output.pdf.encryption.permissions.fill	true	1.2.0 (PDF1.3)	フォームに入力する権限です。 output.pdf.encryptionがv2のときだけ有効です true=許可,false=禁止 です。

名前	デフォルト	バージョン	説明
output.pdf.encryption.permissions.extract	true	1.2.0 (PDF1.3)	障害のあるユーザーのために文書中のテキストや画像を抽出する権限です。 output.pdf.encryptionがv2のときだけ有効です true=許可,false=禁止 です。
output.pdf.encryption.permissions.assemble	true	1.2.0 (PDF1.3)	文書中に新しいページ、ブックマーク、サムネイル画像を追加する権限です。 output.pdf.encryptionがv2のときだけ有効です true=許可,false=禁止 です。
output.pdf.encryption.permissions.print-high	true	1.2.0 (PDF1.3)	文書を高画質で印刷する権限です。 output.pdf.encryptionがv2のときだけ有効です true=許可,false=禁止 です。
output.pdf.fonts.policy	cid-keyed	1.1.0 (PDF1.2)	デフォルトのフォントの埋め込みポリシーです。cid-keyed,cid-identity,embeddedで指定します。指定する値と、使用するフォントの対応は以下の通りです。 **cid-keyed** コア14フォント、CID-Keyed外部フォント **cid-identity** コア14フォント、CID Identity外部フォント **embedded** コア14フォント、埋め込みフォント なお、CSSJ 1.x系ではそれぞれgeneric, external, embedというキーワードが使われていました。互換性のため、このキーワードはCopper PDFでも利用可能です。 CopperPDF 2.0.1からは、スペース区切りで複数の指定が可能になりました。例えば"embedded cid-keyed"という指定をすると、埋め込みフォントが見つからない場合はCID Keyedフォントを使用します。
output.pdf.hyperlinks	false	1.0.0 (PDF1.2)	ハイパーリンク機能です。falseまたはtrueで指定します。 trueにすると、PDFからWWWなどへのハイパーリンクが有効になります。
output.pdf.hyperlinks.href	relative	1.1.0 (PDF1.2)	ハイパーリンクのアドレスの記述方法です。relativeまたはabsoluteで指定します。 relativeでは相対アドレス指定となり、HTMLのa要素のhref属性がそのまま使われます。absoluteでは絶対URIに変換されてPDFに反映されます。

名前	デフォルト	バージョン	説明
output.pdf.hyperlinks.base	ドキュメントのURI	2.0.0 (PDF1.2)	output.pdf.hyperlinks.hrefにrelativeを指定した場合の、基準となるURIです。output.pdf.hyperlinks.hrefがabsoluteの場合は、このプロパティは無効です。
output.pdf.hyperlinks.fragment	true	2.0.0 (PDF1.2)	trueを設定するとHTMLの<a name～あるいはid属性によりドキュメントフラグメントが配置され、URLのフラグメント識別子によってドキュメント内の特定の場所へリンクすることができるようになります。falseを設定した場合はドキュメントフラグメントを配置しません。
output.pdf.image.compression	flate	2.0.3	画像をPDFに埋め込むときの圧縮形式です。以下の値を指定可能です。 **flate** FlateDecode形式です。ただしoutput.pdf.compressionがnoneのときは圧縮されずHEX形式になります。 可逆圧縮のため高画質ですが、出力されるファイルのサイズは大きくなります。 **jpeg** JPEG形式です。 **jpeg2000** JPEG2000形式です。PDF 1.5以降の出力で、Java実行環境にJAIがインストールされている必要があります。
output.pdf.image.compression.lossless	200	2.0.3	output.pdf.image.compressionにより非可逆圧縮(JPEG形式等)を使用する場合、非可逆圧縮を適用する画像サイズの閾値です。指定されたサイズ(縦のピクセル数と横のピクセル数を足したもの)より小さければ可逆圧縮(FlateDecode)を使用します。

名前	デフォルト	バージョン	説明
output.pdf.jpeg-image	raw	1.1.0 (PDF1.2)	JPEG画像の埋め込み方法です。raw,to-flate,recompress(Copper PDF 2.0.3以降)で指定します。 rawでは元のデータをそのまま使います。 to-flateまたはrecompressを設定すると、データを再圧縮します(Copper PDF 2.0.2以前ではto-flateしか使用できず、文字通りFlateDecode形式に再圧縮していました。Copper PDF 2.0.3以降では他の圧縮形式も使用できますが、to-flateとrecompressは全く同じ意味です)。 Copper PDF 2.0.3以降ではシステムにJAIがインストールされ、かつPDF 1.5以降の出力でJPEG 2000に対しても有効になります。
output.pdf.platform-encoding	MS932	1.2.0 (PDF1.2)	PDFを表示する環境のプラットフォームのキャラクタ・エンコーディングです。 PDF1.2以前ではフォント名が影響を受けます。PDF1.3以降ではユニコードが使われるため無関係です。 PDF1.6以前では添付ファイル名が影響を受けます。ファイル名にマルチバイト文字が使われている場合、このエンコーディングが表示するプラットフォームのものと一致しないと文字化けします。日本語の文書であればMS932(Windows版Shift_JIS)、韓国語であればEUC-KR、繁体字中国語ではBig5といった指定をしてください。 PDF1.7以降ではユニコードが使われるため無関係です(Copper PDF 2.0.3)。

名前	デフォルト	バージョン	説明
output.pdf.version	1.5	1.1.0	出力されるPDFファイルのバージョンです。1.2,1.3,1.4,1.5,1.6,1.7が指定可能です(1.5,1.6,1.7はCSSJ 1.xでは対応していません)。PDFのバージョンによって利用できる機能が変わります。指定されたバージョンで未対応の機能を使おうとすると警告が出力され、PDFに反映されません。 PDFの一定のバージョンで有効になる機能は以下の通りです。 **1.3以降** 40から128ビットの暗号化とfill,extract,assemble,print -high権限。 **1.4以降** ファイルの添付、PNG半透明化、SVG透明度。 **1.5以降** JPEG 2000 画像の使用(要JAI)。 **1.7以降** 添付ファイル名にユニコードを使用(Copper PDF 2.0.3)。

表 5.5 その他のプロパティ

名前	デフォルト	バージョン	説明
processing.page-references	false	2.0.0	trueを設定すると、目次、ページ参照のための情報を収集します。falseを設定すると、目次、ページ参照のための情報を収集しないため一部の機能が利用できなくなります。詳細はページの参照を参照してください。
processing.pass-count	1	1.2.0	1回のフォーマット処理のために、文書を処理する回数です。詳細は2パス以上の変換処理を参照してください。

5.1.1 文書中で設定できないプロパティ

以下のリストにあるプロパティは"input.property-pi"の設定とは関係なく、jp.cssj.property処理命令による設定ができません。

- input.http.で始まるプロパティ
- input.default-encoding
- input.property-pi
- output.type
- processing.pass-count

5.1.2 個人利用ライセンスの制限

個人利用ライセンスの場合、原則として入出力プロパティはデフォルトのままで固定されます。ただし、以下のプロパティを除外します。

- input.default-encoding
- output.pdf.bookmarks
- output.pdf.hyperlinks
- processing.page-references
- processing.pass-count

また、利用できるフォントがCID-Keyedフォントのみに固定されているのに伴い、'-cssj-font-policy'プロパティを利用できません。

5.2 エラーハンドラから取得できる情報

エラーハンドラから得ることができる情報の一覧です。

名前	値の形式	説明
page-number	数値	これから生成されるページの番号です。最終的なページ番号が総ページ数となります。
heading-title	文字列	見出し(h1～h6)として認識された文字列です。
broken -image-uri	文字列	表示できない画像のURIです。
pass-count	数値	残りパス数です。
annot	文字列	cssj:annot属性で任意の要素に指定された注釈です。

5.3 CSSプロパティのサポート状況

以下の表はW3C CSS2.1仕様の各プロパティのサポート状況です。

表 5.6 HTML/XML要素に対するCSSプロパティ

特性	サポート	備考
azimuth	しない	音声スタイルのため、印刷には無関係です。
background-attachment	する	
background-color	する	
background-image	する	
background-position	する	
background-repeat	する	
background	する	
border-collapse	する	
border-color	する	
border-spacing	する	
border-style	する	
border-top	する	
border-right	する	
border-bottom	する	
border-left	する	
border-top-color	する	
border-right-color	する	
border-bottom-color	する	
border-left-color	する	
border-top-style	する	
border-right-style	する	
border-bottom-style	する	
border-left-style	する	
border-top-width	する	
border-right-width	する	
border-bottom-width	する	

特性	サポート	備考
border-left-width	する	
border-width	する	
border	する	
bottom	する	
caption-side	する	
clear	する	
clip	する	
color	する	
content	する	
counter-increment	する	
counter-reset	する	
cue-after	しない	音声スタイルのため、印刷には無関係です。
cue-before	しない	音声スタイルのため、印刷には無関係です。
cue	しない	音声スタイルのため、印刷には無関係です。
cursor	しない	インタラクティブスタイルのため、印刷には無関係です。
direction	しない	左から右へ書く言語(アラビア語・ヘブライ語など)はサポートしていません。
display	する	
elevation	しない	音声スタイルのため、印刷には無関係です。
empty-cells	する	
float	する	
font-family	する	
font-size	する	
font-style	する	
font-variant	しない	スモール・キャップフォントは利用できません。
font-weight	する	
font	する	
height	する	
left	する	
letter-spacing	する	
line-height	する	
list-style-image	する	

特性	サポート	備考
list-style-position	する	
list-style-type	する	
list-style	する	
margin-right	する	
margin-left	する	
margin-top	する	
margin-bottom	する	
margin	する	
max-height	する	
max-width	する	
min-height	する	
min-width	する	
orphans	する	
outline-color	しない	インタラクティブスタイルのため、印刷には無関係です。
outline-style	しない	インタラクティブスタイルのため、印刷には無関係です。
outline-width	しない	インタラクティブスタイルのため、印刷には無関係です。
outline	しない	インタラクティブスタイルのため、印刷には無関係です。
overflow	する	
padding-top	する	
padding-right	する	
padding-bottom	する	
padding-left	する	
padding	する	
page-break-after	する	
page-break-before	する	
page-break-inside	する	
pause-after	しない	音声スタイルのため、印刷には無関係です。
pause-before	しない	音声スタイルのため、印刷には無関係です。
pause	しない	音声スタイルのため、印刷には無関係です。
pitch-range	しない	音声スタイルのため、印刷には無関係です。
pitch	しない	音声スタイルのため、印刷には無関係です。

特性	サポート	備考
play-during	しない	音声スタイルのため、印刷には無関係です。
position	する	
quotes	する	
richness	しない	音声スタイルのため、印刷には無関係です。
right	する	
speak-header	しない	音声スタイルのため、印刷には無関係です。
speak-numeral	しない	音声スタイルのため、印刷には無関係です。
speak-punctuation	しない	音声スタイルのため、印刷には無関係です。
speak	しない	音声スタイルのため、印刷には無関係です。
speech-rate	しない	音声スタイルのため、印刷には無関係です。
stress	しない	音声スタイルのため、印刷には無関係です。
table-layout	する	
text-align	する	
text-decoration	する	
text-indent	する	
text-transform	する	
top	する	
unicode-bidi	しない	
vertical-align	する	
visibility	する	
voice-family	しない	音声スタイルのため、印刷には無関係です。
volume	しない	音声スタイルのため、印刷には無関係です。
white-space	する	
windows	する	
width	する	
word-spacing	する	
z-index	する	

表 5.7 ページに対するCSSプロパティ

特性	サポート	備考
margin-top	する	
margin-right	する	

特性	サポート	備考
margin-bottom	する	
margin-left	する	
margin	する	

5.4 HTMLの各要素・属性のサポート状況

以下の表はHTMLの各要素のサポート状況です。

表 5.8 HTMLサポート状況一覧

要素・機能	属性	サポート	備考
a	href name	する	name属性を用いた文書内リンクとハイパーリンクをサポートします。XML文書中で使用可能です。
abbr		する	
acronym		する	
address		する	
applet	width height hspace vspace alt align	しない	枠だけが表示されます。
area		しない	
b		する	
base	href	する	以降のハイパーリンクなどは、hrefからの相対パスになります。XML文書中で使用可能です。
basefont	size color face	しない	font要素と同じ働きをします。
bgsound		しない	
bdo		しない	
big		する	
blink		しない	
blockquote		する	
body	marginheight marginwidth topmargin leftmargin rightmargin bottommargin bgcolor background bgproperties text link	する	alink,vlink属性はサポートしません。
br	clear	する	
button	disabled	しない	
caption	align valign	する	
center		する	
cite		する	
code		する	

要素・機能	属性	サポート	備考
colgroup		する	align,bgcolor,charoff,span,valign,width 属性はサポートされません。
col		する	charoff属性はサポートされません。
comment		しない	
dd		する	
del		する	
dfn		する	
dir	type	する	compact属性はサポートしません。
div	align	する	
dl		する	compact属性はサポートしません。
dt		する	
em		する	
embed	border width height hspace vspace alt hidden frameborder units	しない	画像の表示に使用できます。
fieldset	align	する	
font	size color face font -weight point-size	する	
form		しない	枠だけ表示されます。
frame		しない	
framset		しない	
h1 h2 h3 h4 h5 h6	align	する	
head		する	
hr	align color noshade size width	する	
html		する	
i		する	
iframe		しない	
ilayer	background bgcolor clip height src visibility width left pagex pagey top z-index	する	above,below属性はサポートしません。
img	src alt border width height hspace vspace align	する	XML文書中で使用可能です。ただし、XML文書中ではsrc,width,height属性のみ有効です。
input	disabled	しない	枠だけ表示されます。

要素・機能	属性	サポート	備考
input [type=checkbox]	size	しない	枠だけ表示されます。
input[type=text]	size	しない	枠だけ表示されます。
input [type=password]	size	しない	枠だけ表示されます。
input[type=file]	size	しない	枠だけ表示されます。
input [type=radio]		しない	枠だけ表示されます。
input [type=reset]		しない	枠だけ表示されます。
input [type=button]		しない	枠だけ表示されます。
input [type=submit]		しない	枠だけ表示されます。
input [type=image]	src border width height align	しない	img要素と同様に表示されます。
ins		する	
isindex		しない	
kbd		しない	
keygen		しない	
label		しない	
layer	background bgcolor clip height src visibility width left pagex pagey top z-index	する	above,below属性はサポートしません。
legend		しない	ブロックとして表示します。
li	type	する	value属性はサポートしません。
listing		する	
link	rel type media href	する	rel="StyleSheet" type="text/css" の場合にhrefのCSSスタイルシートをリンクします。これはCSSスタイルシートに対するxml-stylesheet処理命令と同様に動作します。link要素でXSLTスタイルシートをリンクすることはできません。
map		しない	
marquee	bgcolor width height hspace vspace	しない	スクロールはしません。
menu	type	する	compact属性はサポートしません。

要素・機能	属性	サポート	備考
meta	http-equiv name content	する	XML文書中で使用可能です。 HTML文書では、http-equiv="Content-Type" content="text/html; charset=エンコーディング名" という指定を行うことでエンコーディングを指定可能です。 また、PDF出力の際の文書情報を設定することができます。詳細は「デザイナーガイド」を参照してください。
multicol		しない	ブロックとして表示されます。
nextid		しない	
nobr		する	
noembed		しない	表示しません。
noframes		する	表示します。
nolayer		しない	表示しません。
noscript		する	表示します。
object	border width height hspace vspace alt align (absbottom,absmiddle,texttop を除く)	しない	画像の表示に使用できます。
ol	type	する	compact,start属性はサポートしません。
optgroup		しない	
option		しない	
p	align	する	
param		しない	
plaintext		する	
pre	cols width wrap	する	
q		する	
ruby rb rt rp		しない	
s		する	
samp		する	
script		しない	表示しません。
select		しない	枠だけ表示されます。
server		しない	
small		する	
spacer		しない	

要素・機能	属性	サポート	備考
span		する	
strike		する	
strong		する	
style	disabled type media	する	XML文書中で使用可能です。
sub		する	
sup		する	
table	width height bgcolor bgackground align hspace vspace border frame rules cellspacing cellpadding	する	bordercolordark,bordercolorlight,cols,summary属性はサポートされません。
tbody	align bgcolor valign	する	charoff属性はサポートされません。
thead	align bgcolor valign	する	charoff属性はサポートされません。
tfoot	align bgcolor valign	する	charoff属性はサポートされません。
td	bordercolor background bgcolor align valign height width nowrap colspan rowspan	する	charoff,bordercolordark,bordercolorlight属性はサポートされません。colspan,rowspan属性は"display: table-cell"スタイルが指定されている要素に付けることでXML文書でも利用可能です。
th	bordercolor background bgcolor align valign height width nowrap colspan rowspan	する	charoff,bordercolordark,bordercolorlight属性はサポートされません。colspan,rowspan属性は"display: table-cell"スタイルが指定されている要素に付けることでXML文書でも利用可能です。
tr	bordercolor background bgcolor align valign height	する	charoff,bordercolordark,bordercolorlight属性はサポートされません。
textarea	cols rows disabled	しない	枠だけ表示されます。wrap属性は無視されます。
title		する	タイトルバーまたは, PDF文書情報のタイトルとして使われます。
u		する	
ul	type	する	compact属性はサポートされません。
var		する	
wbr		しない	
xmp		する	
dir一般属性		しない	
style一般属性		する	XML文書中で使用可能です。

5.5 拡張機能

Copper PDFには独自の処理命令、CSSプロパティ、CSS関数、XML要素、XML属性があります。また、CSS 2.1ではサポートされず、CSS 3で追加されるプロパティを先行して実装したものがあります。

5.5.1 処理命令の拡張

表 5.9 処理命令一覧

名前	バージョン	説明
jp.cssj.default-encoding	1.1.0	これはHTMLの<meta http -equiv="Content -Type" content="text/html; charset=...">要素の代替機能を提供するものです。エンコーディング名を値に使用します。
jp.cssj.default-style-type	1.1.0	これはHTMLの<meta name="content-style-type" content="...">要素の代替機能を提供するものです。MIME型を値に使用します。
jp.cssj.document-info	1.1.0	これはHTMLの<meta name="..." content="...">要素の代替機能を提供するものです。name,content属性に対して、name,value擬似属性が用意されています。
jp.cssj.property	2.0.0	input.property-piがtrueのときだけ利用可能です。 入出力プロパティをドキュメント中で再設定します。name,value擬似属性が用意されています。valueを省略すると、デフォルト値に設定されます。
jp.cssj.stylesheet	1.1.0	これはHTMLのstyle要素の代替機能を提供するものです。type,media属性に対して、同名の擬似属性が用意されています。スタイルシートは'[]'で囲って記述します。

5.5.2 CSSプロパティの拡張

CSSプロパティ

表 5.10 CSSプロパティー覧

名前	バージョン	継承	デフォルト値	適用対象	説明
-cssj-font-policy	2.0.0	✔	cid-keyed	すべての要素	使用するフォントの種類を指定します。指定できる値は cid -keyed, cid -identity, embeddedのいずれかです。詳細はフォントの設定の章を参照してください。CopperPDF 2.0.1以降では、複数の値を指定可能になりました。例えば"embedded cid-keyed"という指定をすると、埋め込みフォントが見つからない場合はCID Keyedフォントを使用します。
-cssj-page-content	2.0.0		none	すべての要素	要素をページごとに生成します。1つめの値は、ページごとに生成されるコンテンツの名前です。2つめ以降の値は、コンテンツを生成するページ(first, left, right, singleのいずれか)で、省略するか複数列挙することができます。詳細はページごとに生成される内容の章を参照してください。
-cssj-page-content-clear	2.0.0		none	すべての要素	'-cssj-page-content'によりページごとに生成されるコンテンツの再生成を停止します。指定する値は、ページごとに生成されるコンテンツの名前で、複数列挙することができます。詳細はページごとに生成される内容の章を参照してください。

CSS関数

表 5.11 CSS関数一覧

名前	バージョン	引数の数	引数の型	適用プロパティ	説明
-cssj-heading	1.2.0	1	整数	content	いちばん最後に表示された見出し(HTMLのh1～h6または cssj:header 属性がついた要素)の内容を出力します。引数の値は、見出しのレベルです。
-cssj-page-ref	2.0.0	2,3,4	文字列[, 文字列, 文字列]	content	指定したドキュメントフラグメントでのカウンタの値を出力します。詳細はリンクとフラグメントの節を参照してください。
-cssj-cmyk	1.0.0	3	整数 小数 パーセント値	color 他、色を指定するプロパティ	CSSのrgbカラーの代わりに、CMYKで色を指定します。引数の値はそれぞれ Cyan, Magenta, Yellow, Blackの順です。
-cssj-gray	2.0.0	1	整数 小数 パーセント値	color 他、色を指定するプロパティ	CSSのrgbカラーの代わりに、グレイスケールで色を指定します。引数の値は黒味の強さです。

5.5.3 XMLの拡張

Copper PDFはXML中で特別の意味をもつ要素、属性を処理します。

以降の表では、便宜上以下のように接頭辞と名前空間が対応しているものとして記述します。当然、実際のドキュメント中では別の接頭辞を使うことができます(xmlで始まる接頭辞を除く)。なお、接頭辞のないものは、任意の名前空間に属することを意味します。

接頭辞	名前空間
cssj	http://www.cssj.jp/ns/cssjml
html	http://www.w3.org/1999/xhtml
svg	http://www.w3.org/2000/svg

XML要素

表 5.12 XML要素一覧

名前	バージョン	属性	説明
cssj:make-toc	2.0.0	counter, type	目次を生成します。counterはページ番号付けに使用するページカウンタの名前で、typeはページ番号のスタイルです。詳細は目次の章を参照してください。
html:img	1.0.0	alt, src, width, height	HTMLのimg要素と同等の働きをします。
html:a	1.0.0	href,name	HTMLのa要素と同等の働きをします。
html:br	2.0.0		HTMLのbr要素と同等の働きをします。
html:h1〜html:h6	1.0.0		HTMLのh1〜h6要素と同等の働きをします。
svg:svg	1.2.0		SVG画像として処理します。詳細はインラインSVGの節を参照してください。

XML属性

表 5.13 XML属性一覧

名前	バージョン	説明
cssj:annot	1.2.0	処理中に注釈メッセージを出力します。設定された値がannotメッセージとしてドライバに送り返されます。
cssj:header	1.2.0	一般的な要素にHTMLのh1〜h6と同じ意味を持たせ、見出しとして目次やブックマークの生成に使います。値は見出しのレベルです。
html:style	1.0.0	HTMLのstyle属性と同等の働きをします。
html:class	1.0.0	HTMLのclass属性と同等の働きをします。
html:colspan	1.0.0	HTMLのcolspan属性と同等の働きをします。
html:rowspan	1.0.0	HTMLのrowspan属性と同等の働きをします。
xml:lang	2.0.0	HTMLのlang属性と同等の働きをします。
id	1.0.0	HTMLのid属性と同等の働きをします。

GNN **株式会社** GNN
〒103-0022 **東京都中央区日本橋室町**1-10-10 LXS**室町**803**号**
e-mail: info@cssj.jp
ホームページ: http://www.gnn.co.jp/
Copper PDF**サイト**: http://copper-pdf.com/