intra-mart PDF-Designer Ver.7.0.2

プログラマーズ・ガイド

＜＜ 変更履歴 ＞＞

変更年月日	変更内容
2008/10/24	初版
2009/6/11	第二版 intra-mart Ver.7.1.0 に対応しました。 「4.3.2 プログラム開発における注意点」に、ファイル出力の競合についての記述を追記しました。
2009/9/10	第三版 JAVADOC（メソッド説明）の誤記を修正しました。 パスワード設定機能を修正（詳細は history.txt に記載）しました。
2010/2/26	第四版 スクリプト開発モデルにおいてセキュリティ設定機能の不具合を修正しました。 分散環境において画像が欠落する不具合を修正しました。 新メソッドを追加しました（詳細は history.txt に記載）。
2010/5/31	第五版 intra-mart Ver.7.2.0 に対応しました。

＜＜ 目 次 ＞＞

1 はじめに

本ドキュメントは、PDF デザイナーを利用した開発手法について説明しています。PDF デザイナーを利用して開発を行う前にお読み下さい。

1.1 用語解説

intra-mart PDF-Designer	以下、PDF-D と略します。
intra-mart WebPlatform	以下、IWP と略します。
intra-mart AppFramework	以下、AFW と略します。
Application Runtime	IWP/AFWのアプリケーション実行サービスです。以下、AppRSrv と略します。
Resource Service	IWP/AFWのリソース管理サービスです。以下、RSrv と略します。
Storage Service	IWP/AFWのファイル管理サービスです。以下、StorageSrv と略します。

1.2 前提

本ドキュメントは、開発をスムースに開始するための手引書となっています。したがって、実際に開発を行うプログラマの方が対象となります。
また、本ドキュメントは、以下に列挙する技術に関する知識を有することを前提として構成されています。これらの技術に関して不明な点がある場合、本ドキュメントの内容を正しく理解することが困難になることがありますので、予めご了承ください。なお、前提知識となる技術に関しては、一般の専門書籍等をご覧ください。

- Java プログラミング言語
- Java Servlet および JSP
- オペレーティングシステム
- intra-mart WebPlatform または intra-mart AppFramework

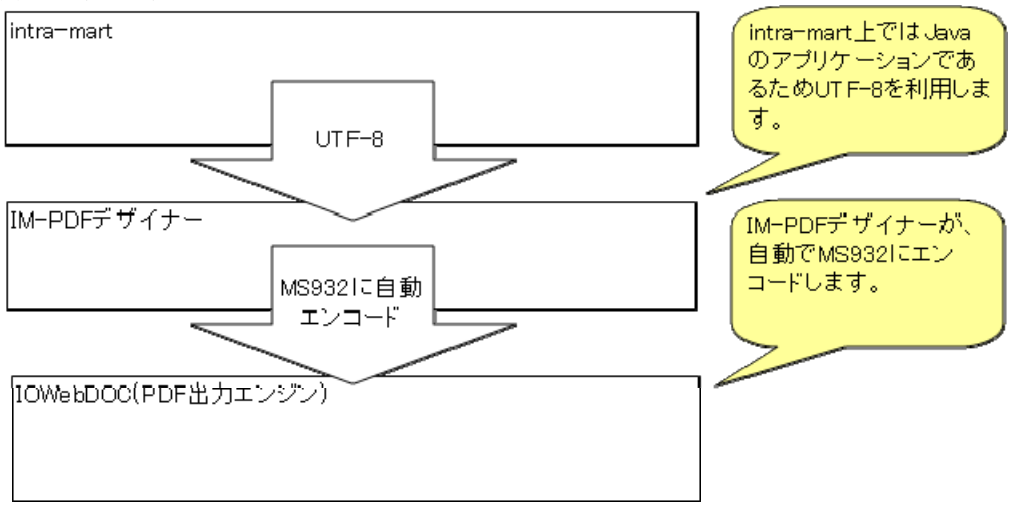
2 UTF-8 環境でのご利用について

PDF デザイナー(Unix/Linux/Windows 版) につきましては、Ver.7.0 より日本語/英数字等の MS932 にて表現できる文字に限定にて UTF-8 対応いたしております。PDF デザイナーVer.7.0 以降を UTF-8 環境にて運用する際には、以下の制限をご理解いただいた上でご利用いただきますようお願い致します。

(1) PDF デザイナーに出力データを設定する方法として、以下 2 種類の手法がございます。
(A) 出力データを CSV ファイルやテキストファイルから読み込む手法
(B) メモリ上で出力データをセットする手法

No	手法	UTF-8 環境での利用時の制限事項
(A)	出力データを CSV ファイルやテキストファイルから読み込む手法	CSV ファイル、テキストファイルは、文字コード MS932 にエンコードして作成する必要があります。
(B)	メモリ上で出力データをセットする手法	UTF-8 の文字データを、PDF デザイナー側で自動で MS932 にエンコードしますので、UTF-8 の文字コードをそのまま利用可能です。 一部制限がありますので、注意事項は必ずお読みください。※詳細は(2)参照

(B)の内部処理概要



(2) (B)メモリ上でデータをセットする場合、MS932 にて使用できない文字は表現できません。UTF8 の一部の文字に、MS932 と互換性のある文字コードと MS932 と互換性の無い文字コードの 2 種類がある文字があります。最近の DB ソフトであれば通常は MS932 と互換性のある文字に変換されて DB に格納されるのが一般的です。ただし、MS932 と互換性の無い文字でデータベースに格納されるリスクもゼロではありません。MS932 と互換性の無い文字コードで DB に格納されてしまうと、PDF デザイナー側では対応できなくなってしまいます。DB ソフト側にて対応できない場合、DB に格納する前に上位のアプリケーションで文字コードを変更してから DB 登録をお願いすることになります。

DB ソフト側で対応できない場合は、文字コードをチェックする仕組みを実装することを推奨いたします。

(a) ユーザからの入力データ受付時に文字コード(MS932 で表現可能か)をチェックする。
(b) プログラムから PDF デザイナーにデータを渡す際に文字コード(MS932 で表現可能か)をチェックする。
(c) UTF-8 から MS932 に変換できない文字については、プログラムにて文字を置換する。
PDF デザイナーに限らず Java の一般的な制約として、UTF-8 から MS932 への変換の際に問題となる文字については、以下の様なものがあります。(他にもある可能性がございますので、事前にご確認頂ければと思います)

∥ (DOUBLE VERTICAL LINE/双柱)
- (MINUS SIGN/負符号, 減算記号)
～ (WAVE DASH/波ダッシュ)
¢ (CENT SIGN/セント記号)
£ (POUND SIGN/ポンド記号)
¬ (NOT SIGN/否定)

※上記はすべて 2 バイト文字の場合です。

注意点・・・・・DB 側の設定や JDBC ドライバの設定方法によって処理結果が異なる場合がありますのでご注意ください。

(3) UTF-8 環境にて、(B)の手法をご利用いただく際には、以下のクラスをご利用ください。下記以外のメソッドは、UTF-8 環境ではご利用いただけません。

No	開発モデル	クラス名
1	JavaEE 開発モデル	AbstractIODOC
2		CSVCela
3		CSVDoc
4		IOIntegration
5		PDFDocumentInformation
6		PDFException
7		PDFIllegalLicenseException
8		PDFIllegalParameterException
9		PDFIllegalStateException
10		PDFIOException
11		PDFLibSecurity
12		PDFMemoryAccessException
13		PDFRuntimeException
14	スクリプト開発モデル	IOCela
15		IODoc
16		IOIntegration

(4) PDF デザイナーの仕様として、PDF の文書情報に 2 バイト文字はご利用頂けません。

(5) 参考資料 PDF デザイナーが利用可能な環境一覧

No	DB	intra-mart	PDF デザイナー	PDF デザイナーの制限事項
1	Shift_JIS	Windows-31J	MS932 (Windows 環境)	(A)(B) (1)～(4)
2		Shift_JIS	SJIS (Unix/Linux 環境)	(A)(B) (1)～(4)
3		UTF-8	MS932 (Windows 環境)	(A)(B) (1)～(4)
4			SJIS (Unix/Linux 環境)	(A)(B) (1)～(4)
5	EUC-JP	EUC-JP	MS932 (Windows 環境)	(A)(B) (1)～(4)
6			SJIS (Unix/Linux 環境)	(A)(B) (1)～(4)
7		UTF-8	MS932 (Windows 環境)	(A)(B) (1)～(4)
8			SJIS (Unix/Linux 環境)	(A)(B) (1)～(4)
9	UTF-8	UTF-8	MS932 (Windows 環境)	(A)(B) (1)～(4)
10			SJIS (Unix/Linux 環境)	(A)(B) (1)～(4)

(A) CSV ファイル、テキストファイルから読み込む手法、(読み込むファイルの文字コードは、MS932 の必要があります)をご利用いただけます。

(B) メモリ上で出力データをセットする手法、UTF-8(MS932 にて表現可能な文字に限定)をご利用いただけます。

(1) から(4)までの制約は当然すべての環境の制限事項として反映されます。

※ PDF デザイナー(EUC)を選択して問題がないのは、現在 EUC 版をご利用いただいているユーザがバージョンアップする場合のみです。また、Ver7.0.0 からの新メソッドは使用できません。EUC を選択する場合、メソッド選択には細心のご注意をお願いします。

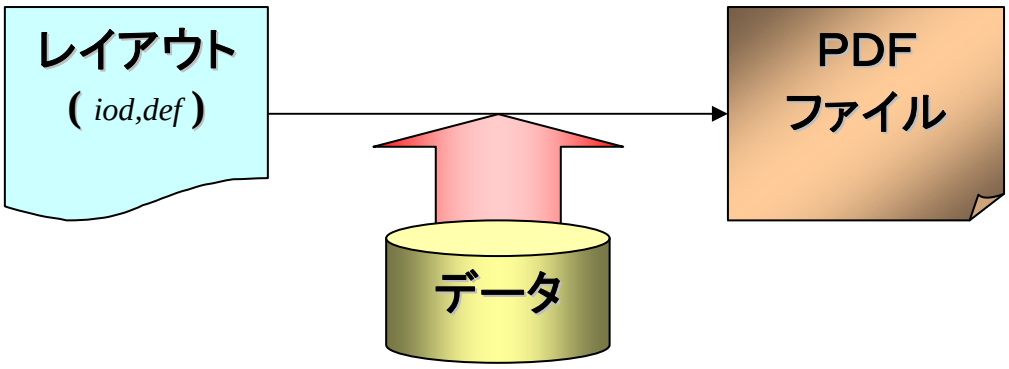
3 開発手順

PDF デザイナーを利用したプログラム開発は以下の流れになります

(1) レイアウトデザインツールでレイアウト定義ファイルを作成します。
CSV 形式のデータファイルを利用する場合は、ここで CSV データとのキーマップ(CDD)を別途作成します。

(2) PDF ファイル作成用のプログラムを開発します。

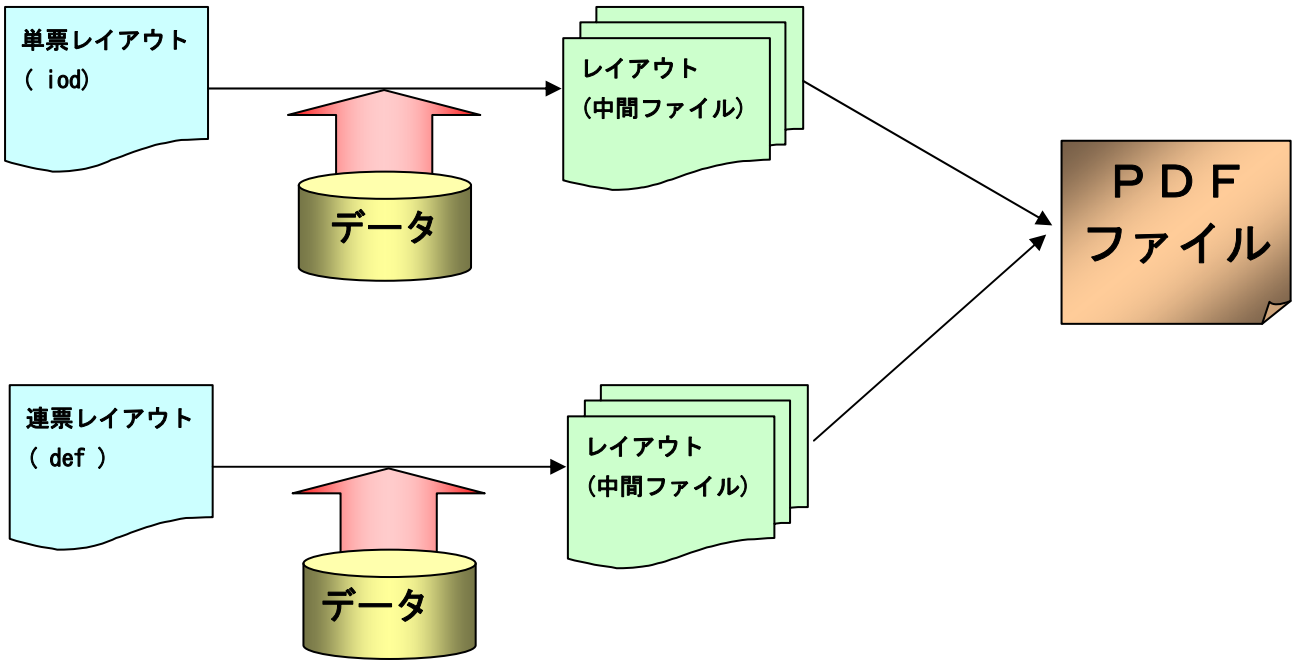
PDF ファイル作成の基本的な概念は、下図のようにレイアウト定義に対してデータを埋め込んで PDF ファイルを作成します。
つまり、1つのレイアウトから1つの PDF ファイルが作成されることが基本となります。



データは、文字列データを指定する以外に制約はありません。従って、データベースから取得したデータをはじめとして、様々なデータソースから取得した値を指定することができます。
データを指定方法としては、以下 3 種類の方法がご利用いただけます。

- CSV ファイルを指定する。
- DAT ファイルを指定する。
- プログラム中でデータを動的に指定する。

PDF ファイルの作成方法としては、複数のレイアウトから1つの PDF ファイルを作成する方法も用意されています。



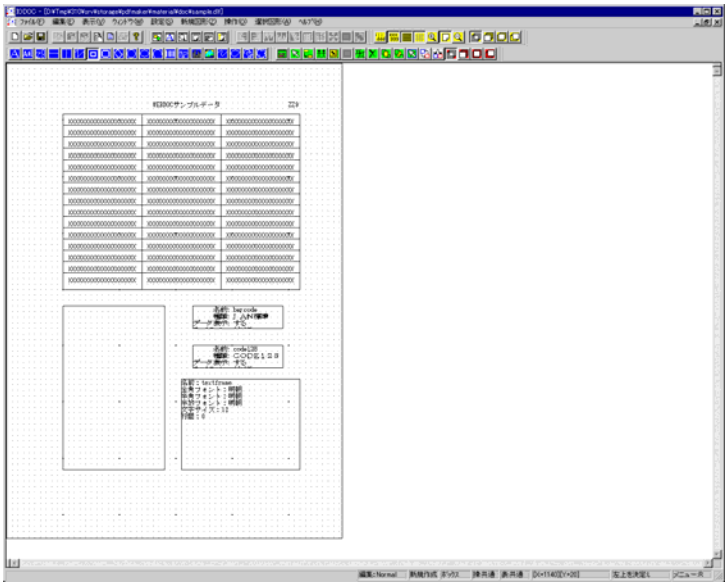
上記のような方法を応用することにより、複数のレイアウトを組み合わせて複雑なドキュメントを作成することも可能です。したがって、本製品を利用して作成した PDF ファイルの表現技法を良く理解した上で、どのようなドキュメントを作成したいのかを設計段階で十分に検討をすることが非常に重要になります。

3.1 レイアウト定義

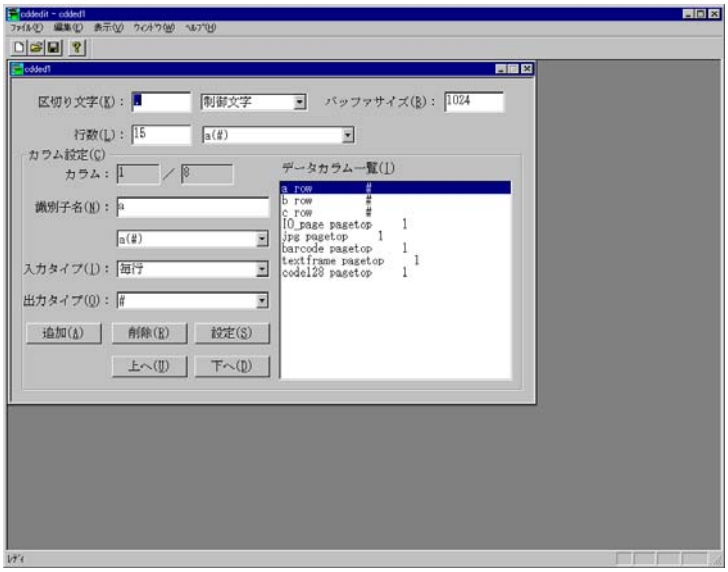
PDF ファイルのレイアウトを設計したら、レイアウトデザインツールを利用してレイアウト定義ファイルを作成します。
レイアウトデザインツールは、Windows 環境で動作します。

3.1.1 単票用（IODodc）

単票用(IODoc)は、複雑なレイアウトのページの作成に適しています。
[スタート]-[プログラム]-[IOWebDOC]-[IODOC]を実行すると、単票用レイアウト定義ファイルを作成するためのツールが起動します。
IODoc の使い方に関しては、専用のマニュアル iothe/iodoc_Tool.pdf をご覧ください。

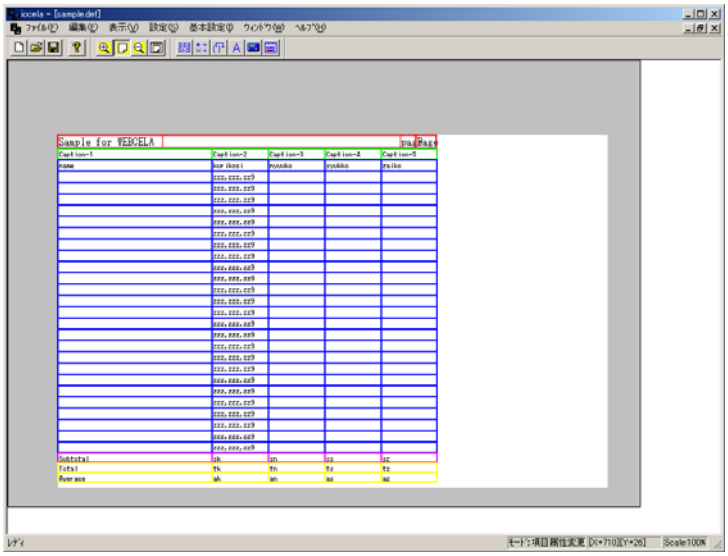


また、PDF ファイルを作成する時に CSV 形式のデータと連携させる場合、変換定義ファイル（cdd）が必要になります。この変換定義ファイルは、CDD エディタを利用すると簡単に作成することができます。
CDD エディタの使い方に関しては、専用のマニュアル iothe/cddedit.pdf をご覧ください。



3.1.2 連票用（IOCela）

連票用(IOCela)は、連続した表形式の帳票の作成に適しています。
[スタート]-[プログラム]-[IOWebDOC]-[IOCELA]を実行すると、連票用レイアウト定義ファイルを作成するためのツールが起動します。
IOCela の使い方に関しては、専用のマニュアル iothe/iocela_Tool.pdf をご覧ください。

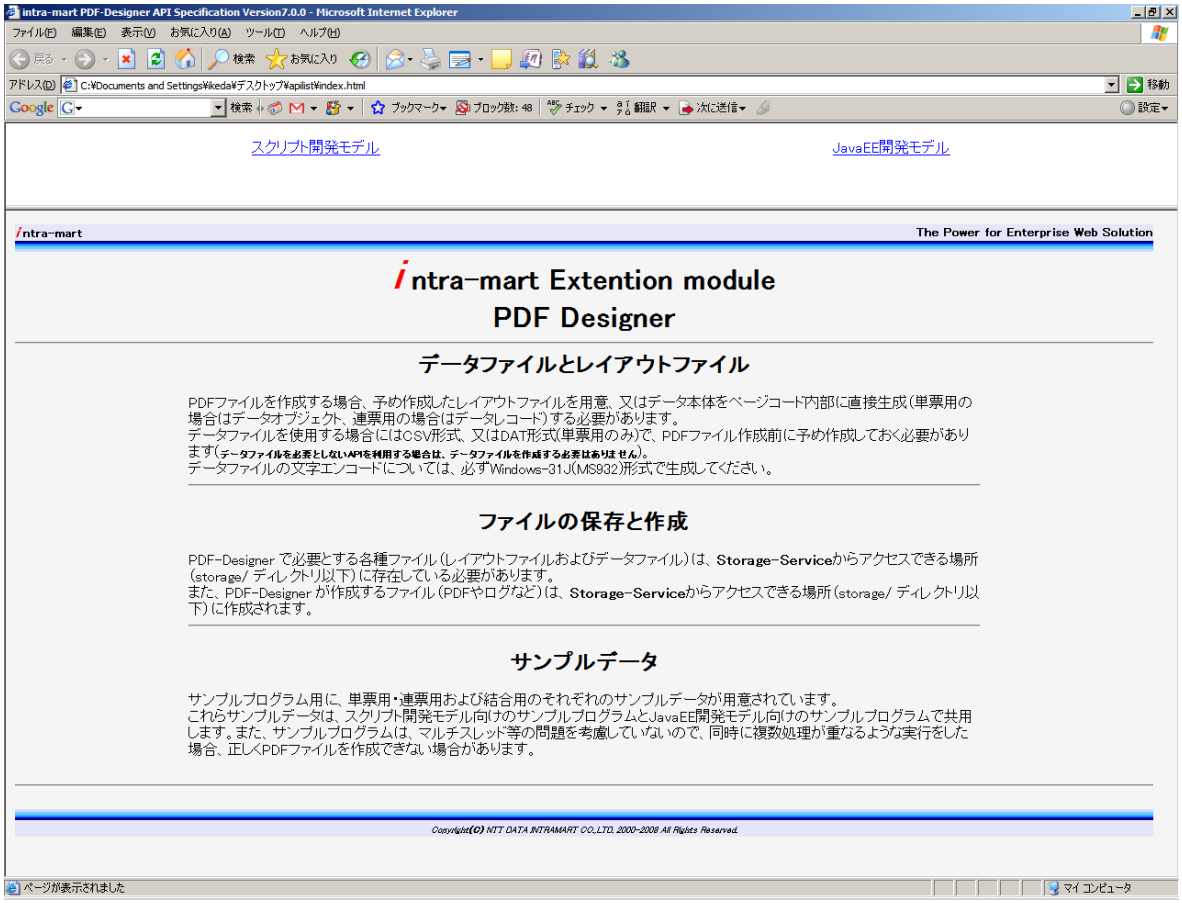


3.2 プログラム開発

本製品で提供されている API を用いて、PDF ファイルを作成するためのプログラムを作成します。
PDF ファイルを作成するための API の利用方法については、PDF デザイナーをインストールすると intra-mart サーバに自動的にプログラムがインストールされ各開発モデル用の API をご利用頂けます。各 API の仕様に関しては、API リスト（apilist.zip）が提供されていますので、そちらをご利用ください。
サンプルプログラムが用意されていますので、API 仕様を学ぶ場合にご利用ください。

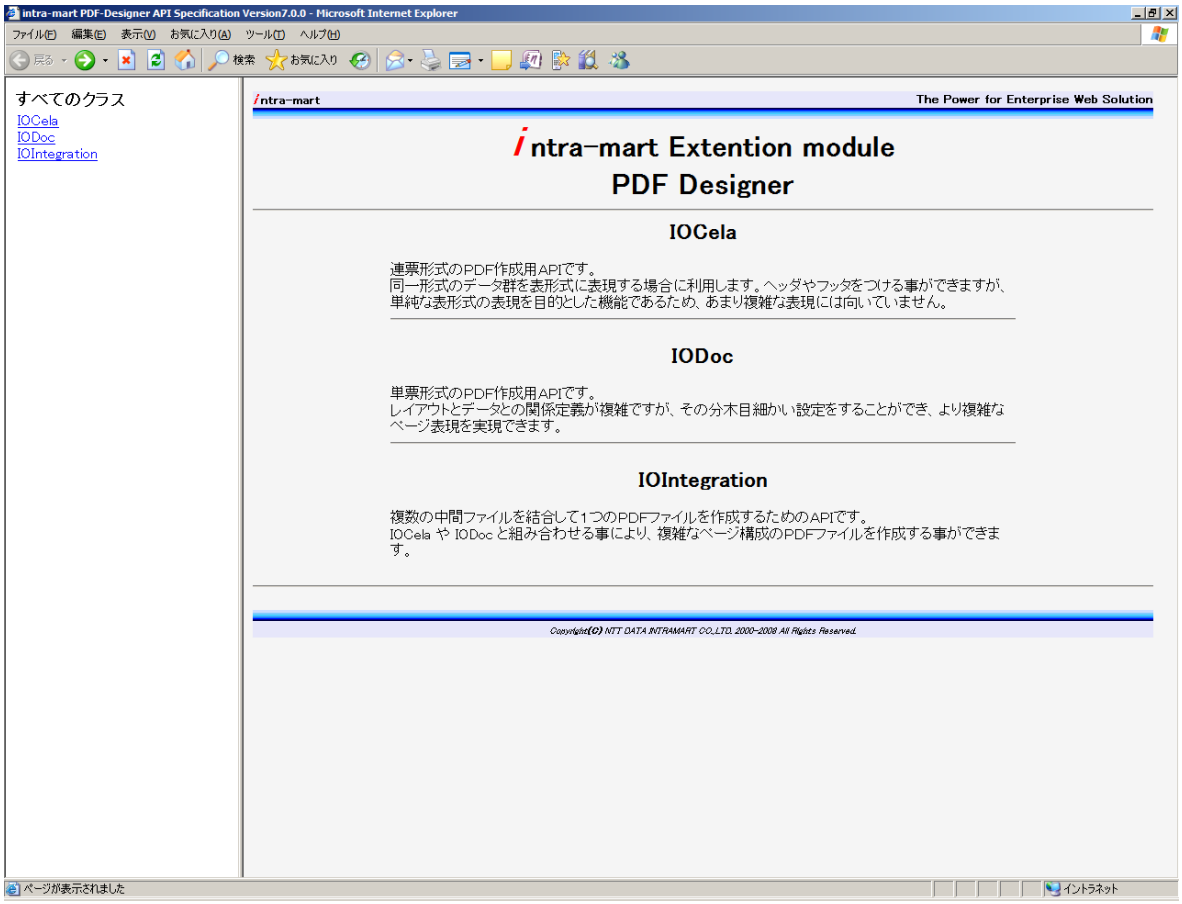
4 APIリスト

本製品には、PDF デザイナー用 API 専用の API リストが付属しています。
API リストは、apilist.zip にあります。このファイルは、ZIP で圧縮されていますので、任意の ZIP 解凍ツールで解凍して下さい（解凍するときは、ディレクトリ付きで解凍して下さい）。
アーカイブファイルを解凍後 apilist/index.html をブラウザで開くことによって、API リストを閲覧することができます。



4.1 スクリプト開発モデル

上部の『スクリプト開発モデル』リンクをクリックすると、FunctionContainer 内で利用可能な JavaScript の API 仕様が表示されます。

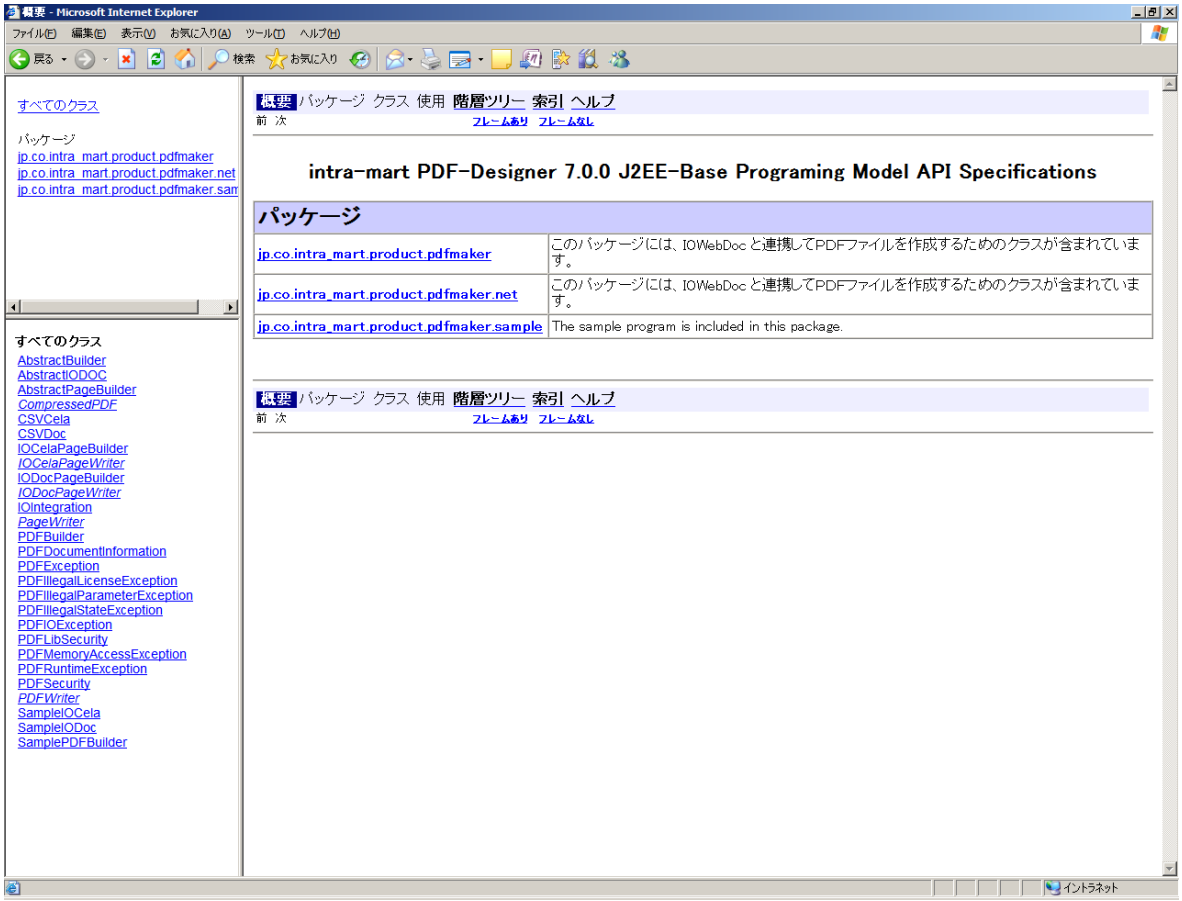


ここに表示された API は、FunctionContainer 内で利用することができます。JavaScript の言語仕様に従って API を利用して下さい。

4.2 JavaEE開発モデル

上部の『JavaEE 開発モデル』リンクをクリックすると、JSP や Servlet などの Java プログラム内で利用可能な Java API（クラス）仕様が表示されます。

また API の他に、JavaEE 開発モデルでのサンプルプログラムとして同梱されているクラス（jp.co.intra_mart.product.pdfmaker.sample.*）の仕様も同時に閲覧することができます。



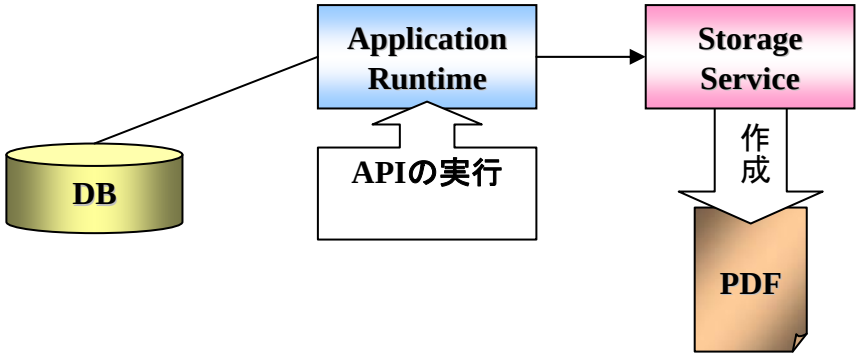
ここに表示された API は、Java プログラム（JSP や Servlet も含む）内で利用することができます。Java の言語仕様に従って API を利用して下さい。

5 プログラミング

ここでは、プログラム開発の際の注意点や、プログラムの方法などを解説します。

5.1 動作概念

通常 FunctionContainer や Java プログラムは、AppRSrv で実行されます。
PDF デザイナーで提供される API も、そのほとんどは AppRSrv で動作しますが、実際に PDF 作成を行うのは StorageSrv になります。したがって、PDF 作成に必要な各ファイルは StorageSrv が動作している Service-Platform プロセスがアクセスできる場所に保存しておく必要があります。また、作成された PDF ファイルは StorageSrv が動作している Service-Platform プロセスがアクセスできるファイルシステム上にファイル保存されます。



intra-mart サーバをネットワーク分散型で運用している場合、上図のように PDF ファイル作成の際にサーバ間でネットワーク通信が発生してしまいます。したがって、PDF 作成用 API のレスポンス時間はネットワーク環境に影響を受けます。

5.2 スクリプト開発モデル

本製品によって IWP/AFW に追加された PDF 作成用の API は、IWP/AFW が標準で提供している他の API と同様にして利用することができます。
PDF 作成用の API を利用した JavaScript プログラムは、AppRSrv で実行されますが、PDF 作成部分のみは、API のメソッドコール時に StorageSrv で PDF 作成ランタイムを起動して PDF ファイルを作成します。したがって、PDF 作成メソッドの実行時は StorageSrv とのネットワーク通信が発生しますので、API のパフォーマンス(レスポンス速度)や実行の成否はネットワークや StorageSrv 等の環境に依存します。

5.3 JavaEE開発モデル

5.3.1 プログラムのコンパイル

PDF デザイナーの API を利用したプログラムをコンパイルする場合、以下のパスをクラスパスに設定して下さい。

- ✓ bin/intramart.jar
- ✓ lib/pdf.jar または doc/imart/WEB-INF/lib/pdf.jar

たとえば、JDK に付属の Java コンパイラを使って Java プログラムファイル(.java)をコンパイルする場合、以下のようにコマンドを実行します（intra-mart サーバをインストールしたディレクトリがカレントディレクトリの場合）。

＞ javac –classpath ./bin/intramart.jar;./lib/pdf.jar;. MyPrg.java

クラスパスの設定にはカレントディレクトリを表す『．』(ピリオド)の指定が必要です。カレントディレクトリにクラスパスが通っていないと、正常にコンパイルできない場合があります。
クラスパスは OS の環境変数 CLASSPATH に設定しておくと、-classpath オプションで指定する必要がなくなります。
javac コマンドの用法に関しては、SUN から提供されている JDK のドキュメントを参照して下さい。なお、プロンプトで javac とだけタイプして（オプション指定なしで）コマンド実行すると、コンソール上に簡単なヘルプが表示されます。

作成した Java プログラムが、JRE に含まれるクラスおよび上記 jar に含まれるクラス以外を利用している場合、それらのクラスの含まれるクラスパスも設定してからコンパイルを行って下さい。

5.3.2 プログラム開発における注意点

- PDF ファイルを作成するプログラムを開発する際に、jp.co.intra-mart.product.pdfmaker パッケージのクラスを利用する場合は、以下の点に関して注意が必要です。
  - ✓ 作成する PDF や IOD ファイルは、PageWriter のインスタンスが生成されるときにファイルがオープンされます。したがって PageWriter のインスタンスを取得した場合、必ず close メソッドが実行されるようにして下さい。close メソッドを実行しないと PageWriter のインスタンスを取得した時にオープンしたファイルが閉じられず、リソース不足を引き起こす要因となります。
  - ✓ PageWriter のインスタンスは、そのインスタンスが生成される時にネイティブメソッド実行用のメモリ領域を占有します。このとき占有したメモリ領域は、release メソッド実行時に開放されます。PageWriter のインスタンスを取得した場合は、必ず release メソッドが実行されるようにプログラミングして下さい。release メソッドが実行されなかった場合、占有したメモリが開放されず、リソース不足を引き起こす要因となります。

- 作成した PDF ファイルのファイルサイズが大きい場合、API のレスポンスと PDF ファイルがディスク上に完全に書き出されるタイミングが大きく異なる場合があります。サイズの大きい PDF ファイルを作成した場合は、十分な時間が経過した後に作成した PDF ファイルにアクセスするようにして下さい。

- PDF デザイナーが提供する API は、指定されたパスを StorageSrv のコンテンツルートディレクトリ（標準では storage ディレクトリ）を親ディレクトリとして解決します。

- PDF デザイナーが提供する API のうち、jp.co.intra-mart.product.pdfmaker パッケージも含まれるクラスの中で java.io.File クラスのインスタンスを受け取るメソッドは、受け取った File オブジェクトが表す絶対パス（File#getAbsolutePath メソッド）によりファイルを決定します。これにより、StorageSrv の動作する Service-Platform プロセスがアクセスできるすべてのファイルシステムに対してファイルアクセスすることができますので、PDF ファイル作成により不用意にファイルを上書きしてしまわないように注意して下さい(指定されたパスが、すでにファイルとして存在していても、API 実行の結果エラーになることはありません)。

- レイアウト定義ファイルに関しては、通常は Shift-JIS で作成されますので StorageSrv の動作環境にコピーする際には Shift-JIS でコピーして下さい。

- PDF デザイナーにおいて、通常の Java アプリケーション同様ファイル出力が競合しないよう、上位アプリケーション側で制御する必要があります。システムを安定して運用するため、
  (1) システムで重複しない出力ファイル名を使用する
  (2) ダブルクリックを防止する
  仕組みを上位アプリケーション側で実装する必要がありますのでご注意ください（未実装の場合、システムが不安定になる可能性があります）。
  出力ファイル名については、上位アプリケーション側でシステムで重複しない ID を生成しファイル名にご利用願います。
  ダブルクリック防止機能については、スクリプト開発モデルであれば isDoubleClicck() がご利用頂けます。JavaEE 開発モデルであれば DblClickForbiden タグがご利用頂けます。ともにイントラマートに標準の API となっております。詳細については、イントラマート API マニュアルをご参照ください。

- PDF デザイナーは、PDF ファイルに 2 種類のパスワードを設定することが可能です。

  (1) オープンパスワード（PDF ファイルの閲覧を制限するためのパスワードです。Adobe Reader 等で開く際にパスワードが要求されます。）
  (2) セキュリティパスワード（PDF ファイルに対して、編集・加工等の操作を制限するためのパスワードです。Adobe Acrobat 等で、PDF ファイルを編集する際にパスワードが要求されます。）

  パスワードを設定する際には、以下 2 点にご注意ください。

  (1) パスワードに空文字は使用しないでください。
  (2) オープンパスワードとセキュリティパスワードに、同じパスワードを設定することはできません。

5.4 体験版ライセンスの注意点

試用版ライセンスでご利用のお客様は、30～60 日間の試用期間が終了すると、PDF 作成 API が自動的に利用できない状態となってしまいますので、その状態で PDF 作成 API を利用したプログラムを実行した場合に、実行時エラーとなってしまいます。
その場合は製品ライセンスをご購入いただき、アンインストール後に再インストールしてください。アンインストール・再インストールの方法は、インストールマニュアルをご確認ください。

6 チュートリアル

ここでは、本製品の API を利用して、スクリプト開発モデル・JavaEE 開発モデルについて、実際にプログラムを作成する過程を学びます。チュートリアルでは、説明を簡素化するためjspからファイルダウンロードを行っていますが、実運用ではサーブレットを使用してファイルをダウンロードして下さい。チュートリアルは SJIS 環境で動作します。

6.1 前提条件

intra-mart WebPlatform がネットワーク分散型でインストールされて、正常に動作していることを前提とします。また、各サーバには PDF デザイナーが正しくインストールされていて、各 API が正常に動作している状態であることが前提となります。サーバは WindowsXP で動作しているものとして説明をします。また、Java は SUN JDK がインストールされているものとします。

6.2 準備

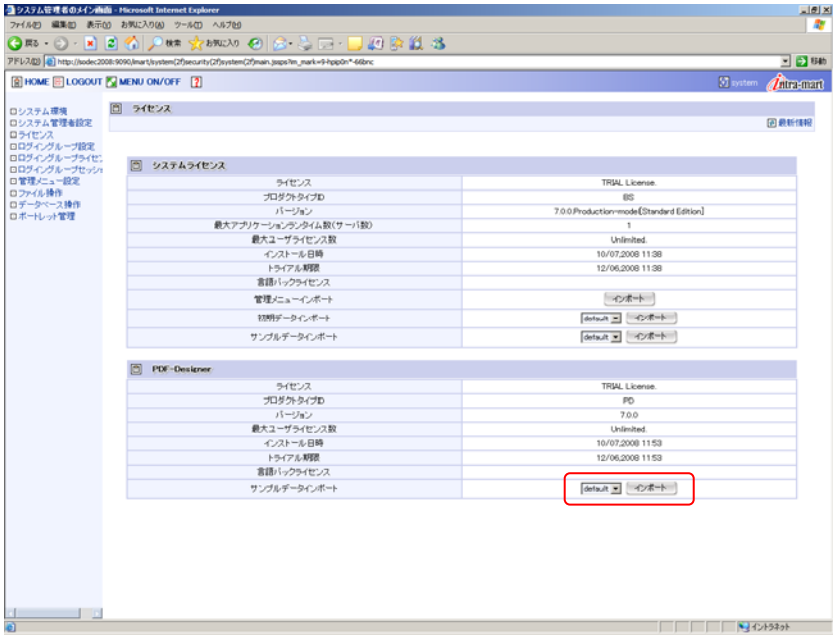
本製品に付属しているサンプルデータをインポートします。

6.2.1.1 サンプルデータのインポート

サンプルデータのインポートは、以下の手順で行って下さい。

a. intra-mart サーバを起動します。
b. ユーザ master（または管理者権限を持つユーザ）でログインします。
c. [システム設定]-[ライセンス]画面を開きます。
d. 『PDF-Designer』にてメニューを登録するログイングループをプルダウンメニューで選択します。
e. 『PDF-Designer』の『インポート』ボタン(サンプルデータインポートの右にあります)をクリックします。

登録が完了したら pdfsuper というロールが追加されていますので、[システム設定]-[ユーザ]画面で、PDF デザイナーのサンプルプログラムを実行したいユーザに pdfsuper ロールを追加して下さい。そのユーザで改めてログインするとメニューに『PDF モジュール』が表示されるようになります。



6.2.1.2 メニュー構成

PDF モジュール
- サンプル
  - スクリプト開発モデル　・・・スクリプト開発モデル用のサンプルプログラムが含まれています
    - 単票用(IODoc 用)
    - 連票用(IOCela 用)
    - PDF ファイル作成(IOIntegration 用)
  - JavaEE 開発モデル　・・・JavaEE 開発モデル用のサンプルプログラムが含まれています
    - 単票用(IODoc 用)
    - 連票用(IOCela 用)
    - PDF ファイル作成(IOIntegration 用)
- チュートリアル
  - スクリプト開発モデル　・・・スクリプト開発モデル用のサンプルプログラムが含まれています
    - 単票用コード(IODoc 用)
    - 連票用コード(IOCela 用)
    - 結合用コード(IOIntegration 用)
  - JavaEE 開発モデル　・・・JavaEE 開発モデル用のサンプルプログラムが含まれています
    - 単票用コード(IODoc 用)
    - 連票用コード(IOCela 用)
    - 結合用コード(IOIntegration 用)

6.3 スクリプト開発モデル

このチュートリアルでは、スクリプト開発モデルにおけるプログラミングの方法について説明します。単票形式、連表形式について、実際に帳票を出力するまでの過程を説明します。スクリプト開発モデルのチュートリアルを始める前に、pages/src/pdfmaker/source-config.xml ファイル内の文字コード設定をインストール環境に合わせて適宜修正してください。

6.3.1 単票形式

ここでは、PDF ファイルを作成するための、スクリプトプログラムを作成します。スクリプト開発では、html ファイルと JavaScript ファイルを作成する必要があります。html ファイル。JavaScript ファイルは AppRSrv 上で動作しますが、実際の PDF 生成処理は StorageSrv 上で実行されます。

6.3.1.1 プログラム作成

テキストエディタを起動して、以下のプログラムを記述します。

```
<!--
// CSVDOC サンプル(PDF-Desiner V7.x)
// IODoc 帳票データをコード内部で生成し、単票両レイアウトの
// PDF 帳票ファイルを生成します。
// PDF ファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
  -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML>
          <HEAD>
          <TITLE>intra-mart[チュートリアルサンプル(IODoc)]</TITLE>
          </HEAD>
          <BODY bgcolor="WhiteSmoke">
          <CENTER>
          <H2>チュートリアルサンプル(IODoc)</H2>

          <TABLE border>
          <TR>
          <TH align="center" nowrap>出力 PDF ファイルは、<BR>
          Storage-Service の[pdfmaker/material/csv/tutorial]<BR>
          フォルダ下に作成され、処理終了後に自動ダウンロードされます。
          </TH>
          </TR>
          <TR>
          <TH align="center" nowrap>処理実行は下のボタンをクリックします。</TH>
          </TR>
          <TR>
          <TD align="center" nowrap>
          <IMART type="form" action="makePDF">
          <INPUT type="submit" value="ＰＤＦ作成">
          </IMART>
          </TD>
```

```
        </TR>
        </TABLE>
        </CENTER>
        </BODY>
</HTML>
```

記述が完了したら C:¥imart¥pages¥src¥pdfmaker¥tutorial ディレクトを作成し docsample.html というファイル名で保存して下さい（AppRSrv の動作する Service-Platform を別なディレクトリにインストールしている場合には、AppRSrv の動作する Service-Platform インストールディレクトリ直下の pages¥src¥pdfmaker¥tutorial ディレクトリ内に保存して下さい）。この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

次に JavaScript ファイルを作成します。

```
// バインド変数宣言

//---
// CSVDOC サンプル(PDF-Desiner V7.x)
//---
// IODoc 帳票データをコード内部で生成し、単票両レイアウトの
// PDF 帳票ファイルを生成します。
// PDF ファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
function makePDF(request){
        // 各ファイルパス設定
        var dirPath = "pdfmaker/material/csv/tutorial";      // ホームフォルダ位置
        var iodPath = dirPath + "/nouhinkensa.iod";          // 入力 IOD

        // 出力 PDF ファイル名編集（ログインユーザ ID 及びセッション ID 付加）
        var userid = AccessSecurityManager.getSessionInfo().user;
        var sessionid = Client.identifier();
        var outPdfName = "nouhinkensa_" + userid + "_" + sessionid + ".pdf";
        var outPdfPath = dirPath + "/" + outPdfName;

        // PDF 文書情報(PDF ファイル生成時に設定されます)
        var docTitle = "PDF デザイナー体験";
        var docAuthor = "ＩＭ 太郎";

        // PDF セキュリティ(PDF ファイル生成時に設定されます)
        var secSecurityPass = "sec1756pswd";
        var printSec = "PRINT_ENABLE";
        var modifySec = "MODIFY_DISABLE";
        var copySec = "COPY_AND_ACCESSBILITY_ENABLE";

        // 処理戻り値
        var resultCode = -9999;

        // 埋め込み識別子及びデータ（本来は DB データ検索等により取得、又はコード内で埋め込みデータを生成し、
```

```
        // 帳票に送ります）
        var doc_data = new Array(2);
        doc_data[0] = new    Array(
        "kyakusaki",OrderComNo","nouhin_No","tantou","nouhinsaki","tyuumon_No",
        "hinmei_code","hinmei","h_memo","syukka_day","suuryou","tani","tanka",
        "j_memo","nouki","shiji_suuryou","nounyu_suuryou","konpou_suuryou","zei",
        "zeinuki","zeikomi","BarCode1","bar1","bar2","bar3"
        );

        doc_data[1] = new Array(
        "NTT データイントラマート","001","001-001","ＩＭ 太郎","ＩＭ商事","C-001-001",
        "YPDFAUTO-001","IM-PDF オートコンバータ","","2004/07/01","2",
        "式","1000000","","2008/07/05","2","2","2","100000","2000000","2100000","CODE39",
        "CODE39","CODE39","CODE39"
        );

        //-----
        // 処理動作
        //-----
        // インスタンス生成
        //（今回はメモリオブジェクト方式のため、入力 IOD のみ指定。CSV ファイル形式の場合は CCD ファイルも指定します）
        var pdf = new IODoc(iodPath, "");

        //-----
        // 文書情報／セキュリティ設定
        //-----
        // 文書情報セット（各項目最大 255 文字まで）
        pdf.defineTitle(docTitle);                  // タイトル
//      pdf.defineSubTitle(docSubTitle);            // サブタイトル（設定無し）
        pdf.defineAuthor(docAuthor);                // 作成者
//      pdf.defineApplication(docApplication);      // 作成アプリケーション名（設定無し）

        // セキュリティ情報セット（パスワードは最大 32 文字まで）
        pdf.setSecurityPassword(secSecurityPass); // セキュリティパスワードを設定
        pdf.printSecurity(printSec);                // 印刷許可設定（印刷可：コメントアウトでも可））
        pdf.modifySecurity(modifySec);              // 変更許可設定（編集不可、注釈不可）
        pdf.copySecurity(copySec);                  // テキスト抽出コピー許可設定（コピー可：コメントアウトでも可）

        //-----
        // ページデータの生成
        //-----
        // 本チュートリアルでは、メモリオブジェクト形式でのデータ設定を実施します。
        // CSV/DAT ファイルオブジェクトでデータを与える場合には、ここでそれぞれ入力ファイルを設定します。
        // テキスト関連識別子データ埋め込み
        // 通常テキスト文字列を埋め込みします。
        //  （※複数行カラムデータの識別子名へは、[識別子#行番号]と編集してセットします）
```

```
	for(var i = 0;i < doc_data[0].length;i++) {
		pdf.setData(doc_data[0][i],doc_data[1][i]);
	}

	// 文字枠データの埋め込み
	// 文字枠への出力は、開始宣言[setTextBoxStart]→データセット[setTextBoxData]→終了宣言[setTextBoxEnd]の順
	// で実施する。
	// データセットは１回で１行分のデータを出力できます（文字枠より大きい文字列長のデータが指定された場合は
	// 自動改行されます）。
	// 複数回データセットを呼び出すことで、改行を含めた文字列のセットが可能です。
	pdf.setTextBoxStart("syoran");
	pdf.setTextBoxData("至急納品");
	pdf.setTextBoxEnd();

	// ページ区切りを出力
	// 複数ページとなる伝票を印刷する場合、印刷ページ区切りを指定することで改ページ位置を指定できます。
//	pdf.setOutPage();

	//-----
	// PDF 出力処理
	//-----
	// PDF ファイルへの出力処理が実行されます。
	// 上記設定内容に従った文書が生成され、正常に処理が完了した場合には指定された PDF ファイル名に
	// 該当の文書が作成されます。
	resultCode = pdf.toPDF(outPdfPath);

	//-----
	// 終了処理
	//-----
	// PDF 作成処理の戻り値が 0 以外である場合は、処理中で何らかのエラーが発生している場合となります
	// （出力ファイルは生成されません）。
	// 戻り値、及び lastMessage メソッドにより取得できるエラーメッセージを参考に、原因を特定し対応します。
	if(resultCode == 0){
		// 結果 PDF ファイルのダウンロード
		var pdfpath = new VirtualFile(outPdfPath);
		Module.download.send(pdfpath.load(),outPdfName);
	}
	else{
		Module.alert.reload("SYSTEM.ERR", "(" + resultCode + ")" + pdf.getMessage());
	}
}
```

記述が完了したら C:¥imart¥pages¥src¥pdfmaker¥tutorial ディレクトに docsample.js というファイル名で保存して下さい（AppRSrv の動作する Service-Platform を別なディレクトリにインストールしている場合には、AppRSrv の動作する Service-Platform インストールディレクトリ直下の pages¥src¥pdfmaker¥tutorial ディレクトリ内に保存して下さい）。この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

6.3.1.2 メニュー設定

a. AppRSrv の動作している環境でブラウザを起動します。
b. master（または、master と同等の権限を有するユーザ）で intra-mart にログインします。
c. [システム設定]-[メニュー]画面を開きます。
d. 『Home』→『PDF モジュール』→『Tutorial』→『スクリプト開発モデル』 フォルダ配下に、以下の設定でメニュー項目を追加します。

表示名	単票用（IODoc 用）
クライアントタイプ	パソコン
URL	docsample.jssp
引数	
アイコン画像ファイルパス	
備考	
ロール	現在のユーザで参照可能なロール

e. ログアウトします。
f. 再び同じユーザでログインします。

6.3.1.3 プログラム実行

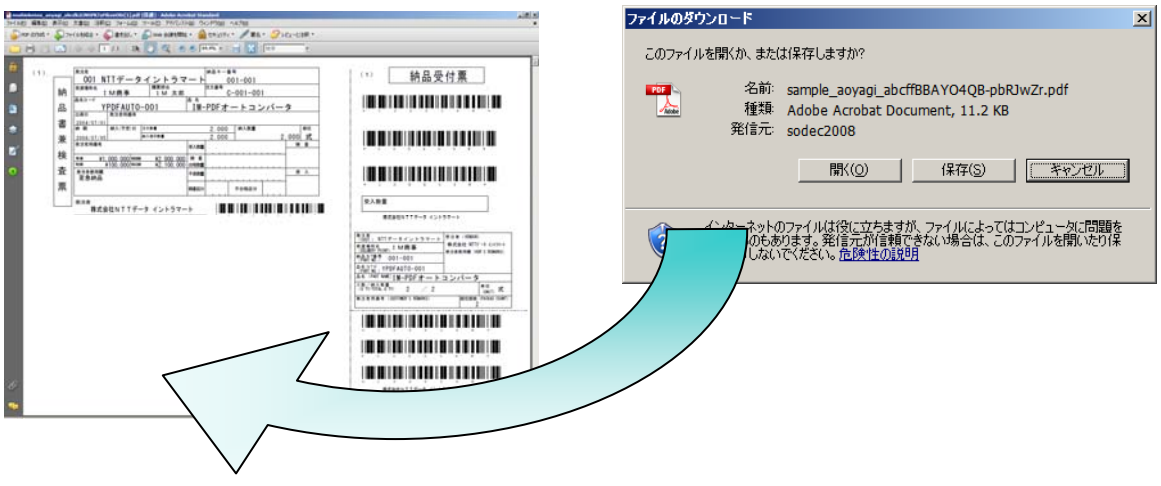
メニューから『Tutorial』→『スクリプト開発モデル』→『単票用（IODoc 用）』を選択して下さい。作成した html ファイルが表示されます。

PDF 作成ボタンを押下すると、PDF ファイルが作成され処理終了後にダウンロードされます。

実行エラーが発生した場合には、エラーメッセージの内容に従い JavaScript ファイルもしくは html ファイルを修正してください。

6.3.1.4 確認

プログラムが正しく実行されると StorageSrv の storage/pdfmaker/tutorial ディレクトリに PDF ファイルが作成されます。このファイルが PDF のビューア（AdobeReader など）で正しく表示できればすべての処理が正しく行われたことになります。



[UNIT]-[ファイル操作]画面で、[pdfmaker]-[material]とディレクトリを移動するとファイルリストに pdf ファイルが作成されています。ファイルが作成されたかどうかだけであれば、この方法で確認することができます。

また、この画面から作成した PDF ファイルをダウンロードすることもできますので、StorageSrv とは別のコンピュータで作成した PDF ファイルの確認が可能です。

6.3.2 連票形式

6.3.2.1 プログラム作成

テキストエディタを起動して、以下のプログラムを記述します。

```
<!--
// CSVCELA サンプル(PDF-Desiner V7.x)
// IOCela 帳票データをコード内部で生成し PDF 帳票ファイルを生成します。
// PDF ファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
  -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML>
          <HEAD>
          <TITLE>intra-mart[チュートリアルサンプル(IOCela)]</TITLE>
          </HEAD>
          <BODY bgcolor="WhiteSmoke">
          <CENTER>
          <H2>チュートリアルサンプル(IOCela)</H2>
          <TABLE border>
          <TR>
          <TH align="center" nowrap>出力 PDF ファイルは、<BR>
          Storage-Service の[pdfmaker/material/csv/tutorial]<BR>
          フォルダ下に作成され、処理終了後に自動ダウンロードされます。
          </TH>
          </TR>
          <TR>
          <TH align="center" nowrap>処理実行は下のボタンをクリックします。</TH>
          </TR>
          <TR>
          <TD align="center" nowrap>
          <IMART type="form" action="makePDF">
          <INPUT type="submit" value="ＰＤＦ作成">
          </IMART>
          </TD>
          </TR>
          </TABLE>
          </CENTER>
          </BODY>
</HTML>
```

記述が完了したら C:¥imart¥pages¥src¥pdfmaker¥tutorial ディレクトを作成し celasample.html というファイル名で保存して下さい（AppRSrv の動作する Service-Platform を別なディレクトリにインストールしている場合には、AppRSrv の動作する Service-Platform インストールディレクトリ直下の pages¥src¥pdfmaker¥tutorial ディレクトリ内に保存して下さい）。この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

次に JavaScript ファイルを作成します。

```
// バインド変数宣言

//---
// CSVCELA サンプル(PDF-Desiner V7.x)
//---
// IOCela 帳票データをコード内部で生成し PDF 帳票ファイルを生成します。
// PDF ファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
function makePDF(request){
        // 各ファイルパス設定
        var dirPath = "pdfmaker/material/csv/tutorial";      // ホームフォルダ位置
        var defPath = dirPath + "/designer.def";             // 入力 DEF ファイル
        var csvPath = dirPath + "/designer_data.csv";        // CSV データファイル

        // 出力 PDF ファイル名編集（ログインユーザ ID 及びセッション ID 付加）
        var userid = AccessSecurityManager.getSessionInfo().user;
        var sessionid = Client.identifier();
        var outPdfName = "designer_" + userid + "_" + sessionid + ".pdf";
        var outPdfPath = dirPath + "/" + outPdfName;

        // PDF 文書情報(PDF ファイル生成時に設定されます)
        var docTitle = "PDF デザイナー体験";
        var docAuthor = "ＩＭ 太郎";

        // PDF セキュリティ(PDF ファイル生成時に設定されます)
        var secSecurityPass = "sec1756pswd";
        var printSec = "PRINT_ENABLE";
        var modifySec = "MODIFY_DISABLE";
        var copySec = "COPY_AND_ACCESSBILITY_ENABLE";

        // 処理戻り値
        var resultCode = -9999;

        //-----
        // 処理動作
        //-----
        // インスタンス生成(DEF ファイル)
        var pdf = new IOCela(defPath);

        //-----
        // 文書情報／セキュリティ設定
        //-----
        // 文書情報セット（各項目最大 255 文字まで）
        pdf.defineTitle(docTitle);                  // タイトル
```

```
//          pdf.defineSubTitle(docSubTitle);          // サブタイトル（設定無し）
            pdf.defineAuthor(docAuthor);              // 作成者
//          pdf.defineApplication(docApplication);    // 作成アプリケーション名（設定無し）

            // セキュリティ情報セット（パスワードは最大 32 文字まで）
            pdf.setSecurityPassword(secSecurityPass); // セキュリティパスワードを設定
            pdf.printSecurity(printSec);              // 印刷許可設定（印刷可：コメントアウトでも可））
            pdf.modifySecurity(modifySec);            // 変更許可設定（編集不可、注釈不可）
            pdf.copySecurity(copySec);                // テキスト抽出コピー許可設定（コピー可：コメントアウトでも可）

            //-----
            // ページデータの生成
            //-----
            // 本チュートリアルでは、レコードオブジェクト形式でのデータ設定を実施します。
            // また IODoc レイアウトの重ね合わせを実施し、内部データをデータオブジェクト形式で設定します。

            // CSV ファイルセット
            // ファイル内容は DEF ファイル上のデータ形式設定に沿って各カラムに値が挿入されます。
            pdf.setCSV(csvPath);

            // CSV データをコード内部でセットする場合では、以下のように setRecord に 1 レコード分のデータ文字列を
            // セットします。
/*
            pdf.setRecord(
            "株式会社 NTT データイントラマート","川崎太郎","2008/10/1","株式会社川崎商事",
            "PDF デザイナー","490000          ","1        ","備考 1   ","備考 2")
            );
*/

            //-----
            // PDF 出力処理
            //-----
            // PDF ファイルへの出力処理が実行されます。
            // 上記設定内容に従った文書が生成され、正常に処理が完了した場合には指定された PDF ファイル名に該当の文書が
            // 作成されます。
            resultCode = pdf.toPDF(outPdfPath);

            //-----
            // 終了処理
            //-----
            // PDF 作成処理の戻り値が 0 以外である場合は、処理中で何らかのエラーが発生している場合となります
            //（出力ファイルは生成されません）。
            // 戻り値、及び lastMessage メソッドにより取得できるエラーメッセージを参考に、原因を特定し対応します。
            if(resultCode == 0){
                        // 結果 PDF ファイルのダウンロード
                        var pdfpath = new VirtualFile(outPdfPath);
```

```
                Module.download.send(pdfpath.load(),outPdfName);
        }
        else{
                Module.alert.reload("SYSTEM.ERR", "(" + result + ")" + pdf.getMessage());
        }
}
```

記述が完了したら C:¥imart¥srv¥doc¥imart ディレクトリ内に celasample.js というファイル名で保存して下さい（AppRSrv の動作する Service-Platform を別なディレクトリにインストールしている場合には、AppRSrv の動作する Service-Platform インストールディレクトリ直下の doc¥imart ディレクトリ内に保存して下さい）。この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

6.3.2.2 メニュー設定

a. AppRSrv の動作している環境でブラウザを起動します。
b. master（または、master と同等の権限を有するユーザ）で intra-mart にログインします。
c. [システム設定]-[メニュー]画面を開きます。
d. 『Home』→『PDF モジュール』→『Tutorial』→『スクリプト開発モデル』 フォルダ配下に、以下の設定でメニュー項目を追加します。

表示名	連票用（IOCela 用）
クライアントタイプ	パソコン
URL	celasample.jssp
引数	
アイコン画像ファイルパス	
備考	
ロール	現在のユーザで参照可能なロール

e. ログアウトします。
f. 再び同じユーザでログインします。

6.3.2.3 プログラム実行

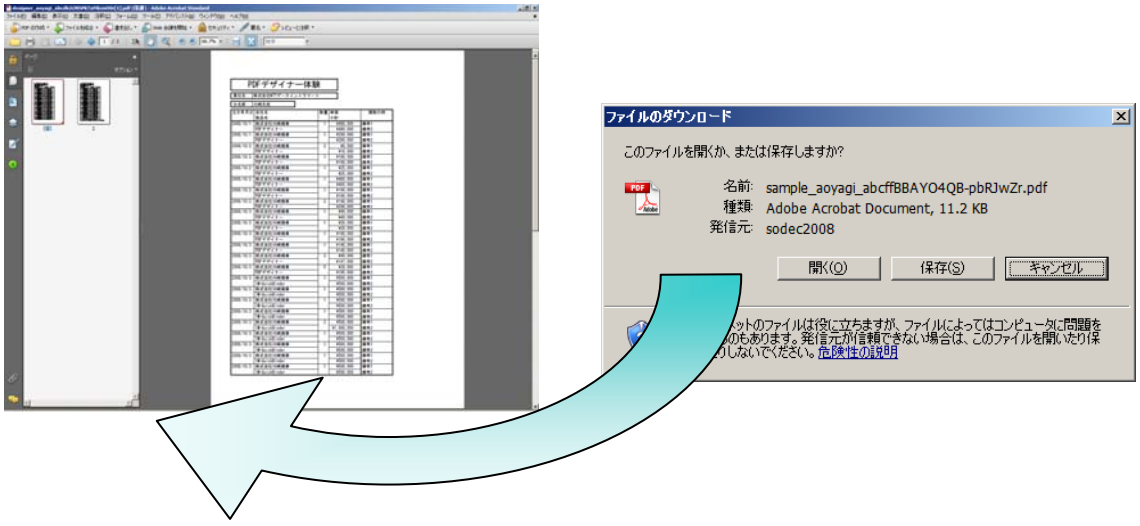
メニューから『Tutorial』→『スクリプト開発モデル』→『連票用（IOCela 用）』を選択して下さい。作成した html ファイルが表示されます。

PDF 作成ボタンを押下すると、PDF ファイルが作成され処理終了後にダウンロードされます。

実行エラーが発生した場合には、エラーメッセージの内容に従い JavaScript ファイルもしくは html ファイルを修正してください。

6.3.2.4 確認

プログラムが正しく実行されると StorageSrv の storage/pdfmaker/tutorial ディレクトリに PDF ファイルが作成されます。このファイルが PDF のビューア（AdobeReader など）で正しく表示できればすべての処理が正しく行われたことになります。



[UNIT]-[ファイル操作]画面で、[pdfmaker]-[material]とディレクトリを移動するとファイルリストに pdf ファイルが作成されています。ファイルが作成されたかどうかだけであれば、この方法で確認することができます。

また、この画面から作成した PDF ファイルをダウンロードすることもできますので、StorageSrv とは別のコンピュータで作成した PDF ファイルの確認が可能です。

6.4 JavaEE開発モデル

このチュートリアルでは、JavaEE 開発モデルにおけるプログラミングの方法について説明します。単票形式、連表形式について、実際に帳票を出力するまでの過程を説明します。

6.4.1 単票形式

ここでは、PDFファイルを作成するための、JSPプログラムを作成します。JSPプログラムに記述されたPDF生成処理は、StorageSrv 上で実行されます。ここでは、JSP プログラムに PDF 生成処理を記載していますが、JSP プログラムから分離して Java プログラムとして作成することも可能です。JSP プログラムは AppRSrv の動作しているコンピュータで作成します。

6.4.1.1 プログラム作成

テキストエディタを起動して、以下のプログラムを記述します。ファイル 1 行目に、環境に合わせて charset、pageEncoding を指定してください。

```
<%@ page language="java" contentType="text/html; charset=" pageEncoding="" %>
<%@ page import="jp.co.intra_mart.foundation.service.client.file.NetworkFile" %>
<%@ page import="jp.co.intra_mart.foundation.security.AccessSecurityManager" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.PDFLibSecurity" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.net.CSVDoc" %>
<%
        //---
        // CSVDOC サンプル(PDF-Desiner V7.x)
        //---
        // IODoc 帳票データをコード内部で生成し、単票両レイアウトの
        // PDF 帳票ファイルを生成します。
        // PDF ファイルへは、文書情報／セキュリティ情報を付加し、出力しています。

        // 各ファイルパス設定
        String dirPath = "pdfmaker/material/tutorial";                    // ホームフォルダ位置
        String iodPath = dirPath.concat("/nouhinkensa.iod"); // 入力 IOD

        // 出力 PDF ファイル名編集（ログインユーザ ID 及びセッション ID 付加）
        String userid = AccessSecurityManager.getInstance().getSessionInfo().getUser();
        String sessionid = session.getId();
        String outPdfPath = dirPath.concat(String.format("/nouhinkensa_%s_%s.pdf", userid, sessionid));

        // PDF 文書情報(PDF ファイル生成時に設定されます)
        String docTitle = "納品書兼検査票";
        String docAuthor = "ＩＭ 太郎";

        // PDF セキュリティ(PDF ファイル生成時に設定されます)
        String secSecurityPass = "sec1756pswd";
        int printSec = PDFLibSecurity.PRINT_ENABLE;
        int modifySec = PDFLibSecurity.MODIFY_DISABLE;
```

```
        int copySec = PDFLibSecurity.COPY_AND_ACCESSBILITY_ENABLE;

        // 処理戻り値とメッセージ
        int resultCode = -9999;
        String resultMessage = "";

        // 埋め込み識別子及びデータ（本来は DB データ検索等により取得、又はコード内で埋め込みデータを生成し、
        // 帳票に送ります）
        String[][] doc_data = {
        {"kyakusaki","OrderComNo","nouhin_No","tantou","nouhinsaki","tyuumon_No","hinmei_code","hinmei",
        "h_memo","syukka_day","suuryou","tani","tanka","j_memo","nouki","shiji_suuryou","nounyu_suuryou",
        "konpou_suuryou","zei","zeinuki","zeikomi","BarCode1","bar1","bar2","bar3"},
        {"NTT データイントラマート","001","001-001","ＩＭ 太郎","ＩＭ商事","C-001-001","YPDFAUTO-001",
        "IM-PDF オートコンバータ","","2004/07/01","2",
        "式","1000000","","2008/07/05","2","2","2","100000","2000000","2100000","CODE39","CODE39","CODE39","CODE39"}
        };

        //-----
        // 処理動作
        //-----
        // インスタンス生成
        // (今回はメモリオブジェクト方式のため、入力 IOD のみ指定。CSV ファイル形式の場合は CCD ファイルも指定します)
        CSVDoc pdf = new CSVDoc(iodPath, "");

        //-----
        // 文書情報／セキュリティ設定
        //-----
        // 文書情報セット（各項目最大 255 文字まで）
        pdf.defineTitle(docTitle);                                          // タイトル
//      pdf.defineSubTitle(docSubTitle);                                    // サブタイトル（設定無し）
        pdf.defineAuthor(docAuthor);                                        // 作成者
//      pdf.defineApplication(docApplication);                  // 作成アプリケーション名（設定無し）

        // セキュリティ情報セット（パスワードは最大 32 文字まで）
        pdf.setSecurityPassword(secSecurityPass); // セキュリティパスワードを設定
        pdf.printSecurity(printSec);              // 印刷許可設定（印刷可：コメントアウトでも可））
        pdf.modifySecurity(modifySec);            // 変更許可設定（編集不可、注釈不可）
        pdf.copySecurity(copySec);                // テキスト抽出コピー許可設定（コピー可：コメントアウトでも可）

        //-----
        // ページデータの生成
        //-----
        // 本チュートリアルでは、メモリオブジェクト形式でのデータ設定を実施します。
        // CSV/DAT ファイルオブジェクトでデータを与える場合には、ここでそれぞれ入力ファイルを設定します。
```

```
		// テキスト関連識別子データ埋め込み
		// 通常テキスト文字列を埋め込みします。
		//  （※複数行カラムデータの識別子名へは、[識別子#行番号]と編集してセットします）
		for(int i = 0;i < doc_data[0].length;i++) {
			pdf.setData(doc_data[0][i],doc_data[1][i]);
		}

		// 文字枠データの埋め込み
		// 文字枠への出力は、開始宣言[setTextBoxStart]→データセット[setTextBoxData]→終了宣言[setTextBoxEnd]の
		// 順で実施する。
		// データセットは１回で１行分のデータを出力できます（文字枠より大きい文字列長のデータが指定された場合は
		// 自動改行されます）。
		// 複数回データセットを呼び出すことで、改行を含めた文字列のセットが可能です。
		pdf.setTextBoxStart("syoran");
		pdf.setTextBoxData("至急納品");
		pdf.setTextBoxEnd();

		// ページ区切りを出力
		// 複数ページとなる伝票を印刷する場合、印刷ページ区切りを指定することで改ページ位置を指定できます。
//		pdf.setOutPage();

		//-----
		// PDF 出力処理
		//-----
		// PDF ファイルへの出力処理が実行されます。
		// 上記設定内容に従った文書が生成され、正常に処理が完了した場合には指定された PDF ファイル名に該当の文書が
		// 作成されます。
		resultCode = pdf.makePDF(outPdfPath);

		//-----
		// 終了処理
		//-----
		// PDF 作成処理の戻り値が 0 以外である場合は、処理中で何らかのエラーが発生している場合となります
		//  （出力ファイルは生成されません）。
		// 戻り値、及び lastMessage メソッドにより取得できるエラーメッセージを参考に、原因を特定し対応します。
		if(resultCode == 0){
			resultMessage = "Success !!";
		}
		else{
			resultMessage = pdf.lastMessage();
		}

// 以下 Web ブラウザ出力 HTML レコードです。
// 当 JSP を呼び出し時に上記 IOCela 帳票からの PDF ファイル生成が実施され、
// 正常に完了した場合には、出力 PDF ファイルをダウンロードする為のリンク
// が表示されます。
```

```
// 出力には処理戻り値、メッセージ取得内容を含みます。
 %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML>
          <HEAD>
          <TITLE>intra-mart[チュートリアルサンプル(IODoc)]</TITLE>
          </HEAD>

          <BODY bgcolor="WhiteSmoke">
          <CENTER>
          <H2>チュートリアルサンプル(IODoc)</H2>
          <TABLE border>
          <TR>
          <TH align="right" nowrap>出力 PDF ファイル</TH>
<%
          if(resultCode == 0) {
%>
          <TD align="left" nowrap><A HREF="./outfile.jsp?file=<%= outPdfPath %>"><%= outPdfPath %></A></TD>
<%
          }
          else {
%>
          <TD align="left" nowrap><%= outPdfPath %></TD>
<%
          }
%>
          </TR>
          <TR>
          <TH align="right" nowrap>戻り値</TH>
          <TD align="left" nowrap><%= resultCode %></TD>
          </TR>
          <TR>
          <TH align="right" nowrap>メッセージ</TH>
          <TD align="left" nowrap><% response.getOutputStream().print(resultMessage); %></TD>
          </TR>
          </TABLE>
          </CENTER>
          </BODY>
</HTML>
```

記述が完了したら C:¥imart¥doc¥imart¥pdfmaker¥tutorial ディレクトリを作成し docsample_act.jsp というファイル名で保存して下さい（AppRSrv の動作する Service-Platform を別なディレクトリにインストールしている場合には、AppRSrv の動作する Service-Platform インストールディレクトリ直下の doc¥imart¥pdfmaker¥tutorial ディレクトリを作成して保存して下さい）。この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

6.4.1.2 メニュー設定

a. AppRSrv の動作している環境でブラウザを起動します。
b. master（または、master と同等の権限を有するユーザ）で intra-mart にログインします。
c. [システム設定]-[メニュー]画面を開きます。
d. 『Home』→『PDF モジュール』→『Tutorial』→『JavaEE 開発モデル』 フォルダ配下に、以下の設定でメニュー項目を追加します。

表示名	単票用（IODoc 用）
クライアントタイプ	パソコン
URL	docsample.jsp
引数	
アイコン画像ファイルパス	
備考	
ロール	現在のユーザで参照可能なロール

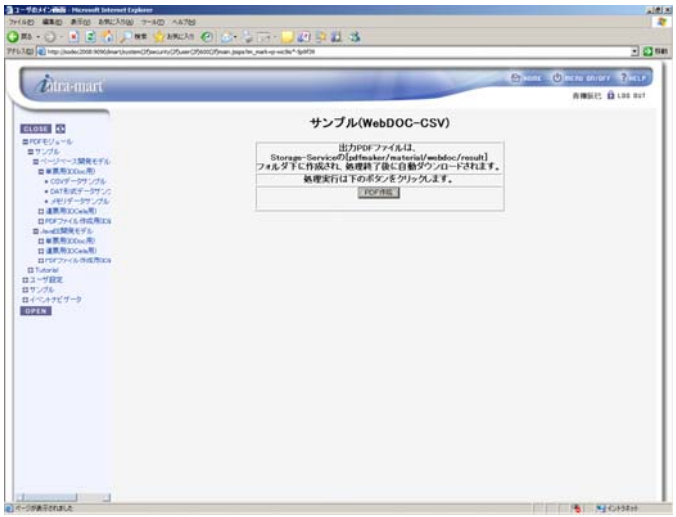
e. ログアウトします。
f. 再び同じユーザでログインします。

6.4.1.3 プログラム実行

メニューから『チュートリアル』→『JavaEE 開発モデル』→『単票用（IODoc 用）』を選択して下さい。作成した jsp ファイルが表示されます。

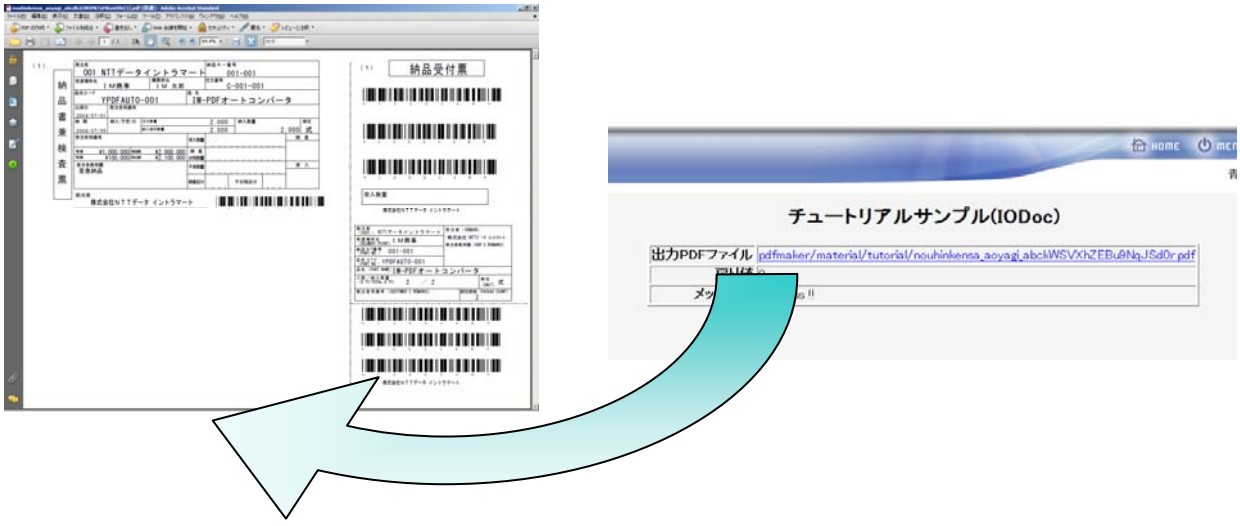
PDF 作成ボタンを押下すると、PDF ファイルが作成され処理終了後にダウンロードされます。

実行エラーが発生した場合には、エラーメッセージの内容に従い jsp ファイルを修正してください。



6.4.1.4 確認

プログラムが正しく実行されると StorageSrv の storage/pdfmaker/tutorial ディレクトリに PDF ファイルが作成されます。このファイルが PDF のビューア（AdobeReader など）で正しく表示できればすべての処理が正しく行われたことになります。



[UNIT]-[ファイル操作]画面で、[pdfmaker]-[material]とディレクトリを移動するとファイルリストに pdf ファイルが作成されています。ファイルが作成されたかどうかだけであれば、この方法で確認することができます。

また、この画面から作成した PDF ファイルをダウンロードすることもできますので、StorageSrv とは別のコンピュータで作成した PDF ファイルの確認が可能です。

6.4.2 連票形式

6.4.2.1 プログラム作成

テキストエディタを起動して、以下のプログラムを記述します。ファイル 1 行目に、環境に合わせて charset、pageEncoding を指定してください。

```
<%@ page language="java" contentType="text/html; charset=" pageEncoding="" %>
<%@ page import="jp.co.intra_mart.foundation.service.client.file.NetworkFile" %>
<%@ page import="jp.co.intra_mart.foundation.security.AccessSecurityManager" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.PDFLibSecurity" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.net.CSVCela" %>
<%
        //---
        // CSVCELA サンプル(PDF-Desiner V7.x)
        //---
        // IOCela 帳票データをコード内部で生成し PDF 帳票ファイルを生成します。
        // PDF ファイルへは、文書情報／セキュリティ情報を付加し、出力しています。

        // 各ファイルパス設定
        String dirPath = "pdfmaker/material/tutorial";        // ホームフォルダ位置
        String defPath = dirPath.concat("/designer.def");     // 入力 DEF ファイル
        String csvPath = dirPath.concat("/designer_data.csv");          // CSV データファイル

        // 出力 PDF ファイル名編集（ログインユーザ ID 及びセッション ID 付加）
        String userid = AccessSecurityManager.getInstance().getSessionInfo().getUser();
        String sessionid = session.getId();
        String outPdfPath = dirPath.concat(String.format("/designer_%s_%s.pdf", userid, sessionid));

        // PDF 文書情報(PDF ファイル生成時に設定されます)
        String docTitle = "PDF デザイナー体験";
        String docAuthor = "ＩＭ 太郎";

        // PDF セキュリティ(PDF ファイル生成時に設定されます)
        String secSecurityPass = "sec1756pswd";
        int printSec = PDFLibSecurity.PRINT_ENABLE;
        int modifySec = PDFLibSecurity.MODIFY_DISABLE;
        int copySec = PDFLibSecurity.COPY_AND_ACCESSBILITY_ENABLE;

        // 処理戻り値とメッセージ
        int resultCode = -9999;
        String resultMessage = "";

        //-----
        // 処理動作
```

```
        //-----
        // インスタンス生成(DEF ファイル)
        CSVCela pdf = new CSVCela(defPath);

        //-----
        // 文書情報／セキュリティ設定
        //-----
        // 文書情報セット（各項目最大 255 文字まで）
        pdf.defineTitle(docTitle);                  // タイトル
//      pdf.defineSubTitle(docSubTitle);            // サブタイトル（設定無し）
        pdf.defineAuthor(docAuthor);                // 作成者
//      pdf.defineApplication(docApplication);      // 作成アプリケーション名（設定無し）

        // セキュリティ情報セット（パスワードは最大 32 文字まで）
        pdf.setSecurityPassword(secSecurityPass); // セキュリティパスワードを設定
        pdf.printSecurity(printSec);                // 印刷許可設定（印刷可：コメントアウトでも可））
        pdf.modifySecurity(modifySec);              // 変更許可設定（編集不可、注釈不可）
        pdf.copySecurity(copySec);                  // テキスト抽出コピー許可設定（コピー可：コメントアウトでも可）

        //-----
        // ページデータの生成
        //-----
        // 本チュートリアルでは、レコードオブジェクト形式でのデータ設定を実施します。
        // また IODoc レイアウトの重ね合わせを実施し、内部データをデータオブジェクト形式で設定します。

        // CSV データファイル設定
        // ファイル内容は DEF ファイル上のデータ形式設定に沿って各カラムに値が挿入されます。
        pdf.setCSV(csvPath);



        // CSV データをコード内部でセットする場合では、以下のように setRecord に 1 レコード分のデータ文字列を
        // セットします。
/*
        pdf.setRecord(String.format("%s %s %s %s %s %s %s %s %s",
        "株式会社 NTT データイントラマート","川崎太郎","2008/10/1","株式会社川崎商事","PDF デザイナー","490000",
        "1","備考 1","備考 2"));
*/

        //-----
        // PDF 出力処理
        //-----
        // PDF ファイルへの出力処理が実行されます。
        // 上記設定内容に従った文書が生成され、正常に処理が完了した場合には指定された PDF ファイル名に該当の文書
        // が作成されます。
        resultCode = pdf.makePDF(outPdfPath);
```

```
		//-----
		// 終了処理
		//-----
		// PDF 作成処理の戻り値が 0 以外である場合は、処理中で何らかのエラーが発生している場合となります
		// （出力ファイルは生成されません）。
		// 戻り値、及び lastMessage メソッドにより取得できるエラーメッセージを参考に、原因を特定し対応します。
		if(resultCode == 0){
			resultMessage = "Success !!";
		}
		else{
			resultMessage = pdf.lastMessage();
		}

// 以下 Web ブラウザ出力 HTML レコードです。
// 当 JSP を呼び出し時に上記 IOCela 帳票からの PDF ファイル生成が実施され、
// 正常に完了した場合には、出力 PDF ファイルをダウンロードする為のリンク
// が表示されます。
// 出力には処理戻り値、メッセージ取得内容を含みます。
 %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML>
		<HEAD>
		<TITLE>intra-mart[チュートリアルサンプル(IOCela)]</TITLE>
		</HEAD>

		<BODY bgcolor="WhiteSmoke">
		<CENTER>
		<H2>チュートリアルサンプル(IOCela)</H2>
		<TABLE border>
		<TR>
		<TH align="right" nowrap>出力 PDF ファイル</TH>
<%
		if(resultCode == 0) {
%>
		<TD align="left" nowrap><A HREF="./outfile.jsp?file=<%= outPdfPath %>"><%= outPdfPath %></A></TD>
<%
		}
		else {
%>
		<TD align="left" nowrap><%= outPdfPath %></TD>
<%
		}
%>
		</TR>
		<TR>
		<TH align="right" nowrap>戻り値</TH>
```

```
		<TD align="left" nowrap><%= resultCode %></TD>
		</TR>
		<TR>
		<TH align="right" nowrap>メッセージ</TH>
		<TD align="left" nowrap><% response.getOutputStream().print(resultMessage); %></TD>
		</TR>
		</TABLE>
		</CENTER>
		</BODY>
</HTML>
```

記述が完了したら C:¥imart¥srv¥doc¥imart ディレクトリ内に celasample_act.jsp というファイル名で保存して下さい（AppRSrv の動作する Service-Platform を別なディレクトリにインストールしている場合には、AppRSrv の動作する Service-Platform インストールディレクトリ直下の doc¥imart ディレクトリ内に保存して下さい）。この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

6.4.2.2 メニュー設定

a. AppRSrv の動作している環境でブラウザを起動します。
b. master（または、master と同等の権限を有するユーザ）で intra-mart にログインします。
c. [システム設定]-[メニュー]画面を開きます。
d. 『Home』→『PDF モジュール』→『Tutorial』→『JavaEE 開発モデル』 フォルダ配下に、以下の設定でメニュー項目を追加します。

表示名	連票用（IOCela 用）
クライアントタイプ	パソコン
URL	celasample.jsp
引数	
アイコン画像ファイルパス	
備考	
ロール	現在のユーザで参照可能なロール

e. ログアウトします。
f. 再び同じユーザでログインします。

6.4.2.3 プログラム実行

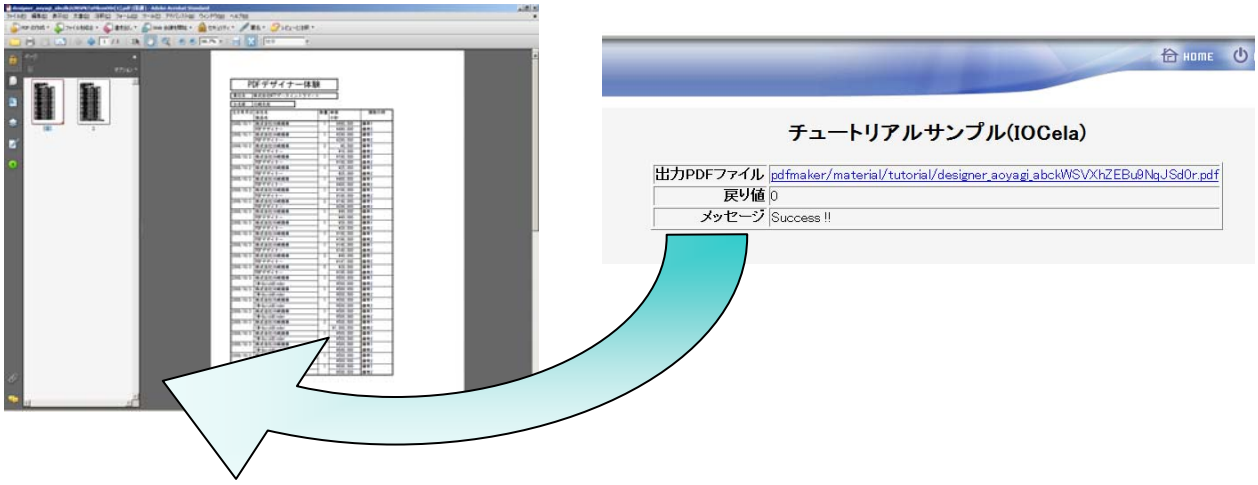
メニューから『チュートリアル』→『JavaEE 開発モデル』→『連票用（IOCela 用）』を選択して下さい。作成した jsp ファイルが表示されます。

PDF 作成ボタンを押下すると、PDF ファイルが作成され処理終了後にダウンロードされます。

実行エラーが発生した場合には、エラーメッセージの内容に従い jsp ファイルを修正してください。

6.4.2.4 確認

プログラムが正しく実行されると StorageSrv の storage/pdfmaker/tutorial ディレクトリに PDF ファイルが作成されます。このファイルが PDF のビューア（AdobeReader など）で正しく表示できればすべての処理が正しく行われたことになります。



[UNIT]-[ファイル操作]画面で、[pdfmaker]-[material]とディレクトリを移動するとファイルリストに pdf ファイルが作成されています。ファイルが作成されたかどうかだけであれば、この方法で確認することができます。

また、この画面から作成した PDF ファイルをダウンロードすることもできますので、StorageSrv とは別のコンピュータで作成した PDF ファイルの確認が可能です。

7 サンプルプログラム

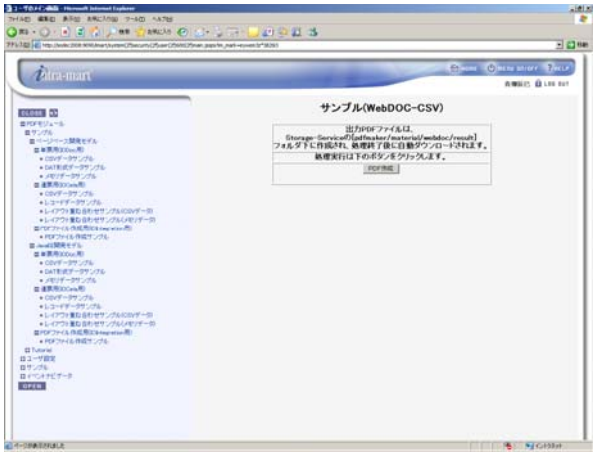
本製品には、PDF デザイナーの API の使用方法を説明したサンプルプログラムが同梱されています。サンプルプログラムは PDF デザイナーインストール時にランタイムと一緒にインストールされます。ここでは、サンプルプログラムの実行方法と内容について説明します。サンプルプログラムは SJIS 環境で動作します。

7.1 サンプルプログラムの保存位置

7.1.1 スクリプト開発モデル

RSrv（スタンドアロンの場合は AppRSrv）のスクリプトプログラム保存ディレクトリ（標準では pages）の src/pdfmaker/sample ディレクトリに保存されています。サンプルプログラムの構成を以下で説明します。

```
pages/src
  └ pdfmaker/
      └ sample/
          ├ doccsv.html        ・・・ CSV ファイルを用いて単票形式の PDF を作成するサンプル
          ├ doccsv.js          ・・・ CSV ファイルを用いて単票形式の PDF を作成するサンプル
          ├ docdat.html        ・・・ DAT ファイルを用いて単票形式の PDF を作成するサンプル
          ├ docdat.js          ・・・ DAT ファイルを用いて単票形式の PDF を作成するサンプル
          ├ docobj.html        ・・・ メモリデータを用いて単票形式の PDF を作成するサンプル
          ├ docobj.js          ・・・ メモリデータを用いて単票形式の PDF を作成するサンプル
          ├ celacsv.html       ・・・ CSV ファイルを用いて連票形式の PDF を作成するサンプル
          ├ celacsv.js         ・・・ CSV ファイルを用いて連票形式の PDF を作成するサンプル
          ├ celarec.html       ・・・ レコードデータを用いて連票形式の PDF を作成するサンプル
          ├ celarec.js         ・・・ レコードデータを用いて連票形式の PDF を作成するサンプル
          ├ celacsvdat.html    ・・・ CSV ファイルを用いてレイアウトを重ね合わせるサンプル
          ├ celacsvdat.js      ・・・ CSV ファイルを用いてレイアウトを重ね合わせるサンプル
          ├ celarecobj.html    ・・・ メモリデータを用いてレイアウトを重ね合わせるサンプル
          ├ celarecobj.html    ・・・ メモリデータを用いてレイアウトを重ね合わせるサンプル
          ├ integration.html   ・・・ PDF ファイルを作成するサンプル
          └ integration.js     ・・・ PDF ファイルを作成するサンプル
```

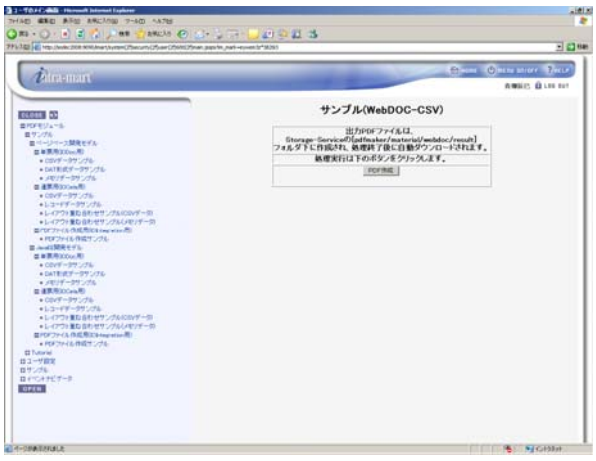


7.1.2 JavaEE開発モデル

サンプルプログラムは AppRSrv の doc/imart/pdfmaker/sample ディレクトリに保存されています。サンプルプログラムのファイル構成を以下で説明します。チュートリアルでは、説明を簡素化するためjspからファイルダウンロードを行っていますが、実運用ではサーブレットを使用してファイルをダウンロードして下さい。

doc/imart/
└ pdfmaker/
　└ sample/

├ doccsv.jsp	・・・ CSV ファイルを用いて単票形式の PDF を作成するサンプル
├ doccsv_act.jsp	・・・ CSV ファイルを用いて単票形式の PDF を作成するサンプル
│ docdat.jsp	・・・ DAT ファイルを用いて単票形式の PDF ファイルを作成するサンプル
│ docdat_act.jsp	・・・ DAT ファイルを用いて単票形式の PDF ファイルを作成するサンプル
│ docobj.jsp	・・・ メモリデータを用いて単票形式の PDF ファイルを作成するサンプル
│ docobj_act.jsp	・・・ メモリデータを用いて単票形式の PDF ファイルを作成するサンプル
│ celacsv.jsp	・・・ CSV ファイルを用いて連票形式の PDF を作成するサンプル
│ celacsv_act.jsp	・・・ CSV ファイルを用いて連票形式の PDF を作成するサンプル
├ celarec.jsp	・・・ レコードデータを用いて連票形式の PDF ファイルを作成するサンプル
├ celarec_act.jsp	・・・ レコードデータを用いて連票形式の PDF ファイルを作成するサンプル
├ celareccsv.jsp	・・・ CSV ファイルを用いてレイアウトを重ね合わせるサンプル
├ celareccsv_act.jsp	・・・ CSV ファイルを用いてレイアウトを重ね合わせるサンプル
├ celarecobj.jsp	・・・ メモリデータを用いてレイアウトを重ね合わせるサンプル
├ celarecobj_act.jsp	・・・ メモリデータを用いてレイアウトを重ね合わせるサンプル
├ integration.jsp	・・・ PDF ファイルを作成するサンプル
├ integration_act.jsp	・・・ PDF ファイルを作成するサンプル
└ outfile.jsp	・・・ ファイルダウンロード用



7.2 サンプルデータ

サンプルデータは StorageSrv のコンテンツ保存ディレクトリ（標準では storage ディレクトリ）の下記に示すディレクトリに保存されています。サンプルデータは、サンプルプログラムで利用します。

```
storage/
  └ pdfmaker/
      └ material/
          ├ tutorial/     ・・・ チュートリアル用のデータ
          ├ webdoc/       ・・・ 単票形式の PDF ファイル作成サンプル用のデータ
          ├ webcela/      ・・・ 連票形式の PDF ファイル作成サンプル用のデータ
          └ integration/  ・・・ 結合サンプル用のデータ
```

7.3 サンプルプログラムの説明

PDF デザイナーに含まれているサンプルプログラムについて説明します。PDF デザイナーはさまざまな形式のデータを指定して PDF ファイルを作成することができます。

単票形式については、以下 3 種類の方法でデータを指定し PDF ファイルを作成することができます。
(1) CSV ファイルを指定して PDF ファイルを作成する。
(2) DAT ファイルを指定して PDF ファイルを作成する。
(3) メモリデータを指定して PDF ファイルを作成する。

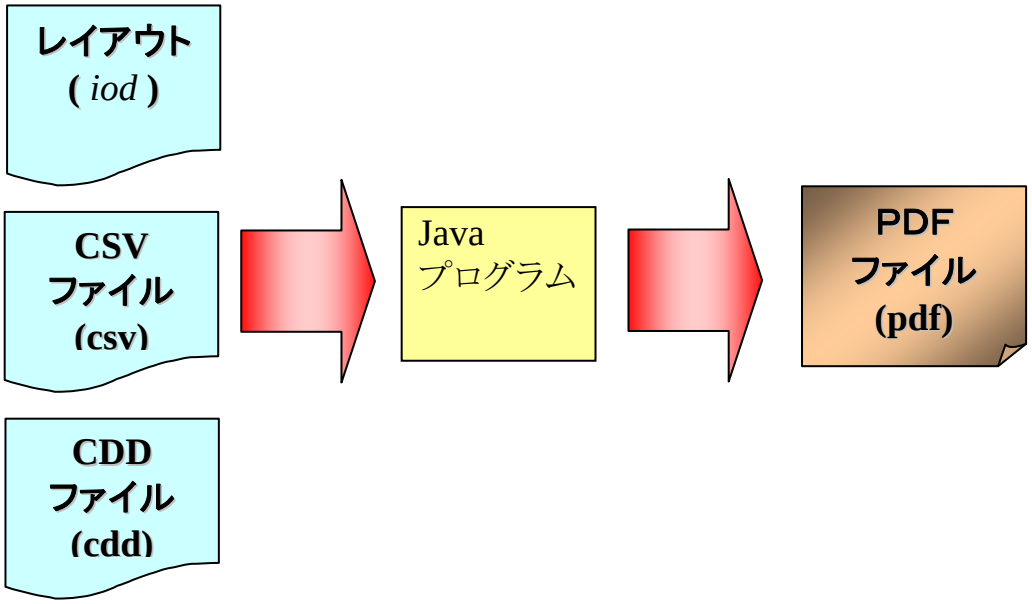
連票形式については、以下 4 種類の方法でデータを指定し PDF ファイルを作成することができます。
(1) CSV ファイルを指定して PDF ファイルを作成する。
(2) レコードデータを指定して PDF ファイルを作成する。
(3) CSV ファイルと単票形式のレイアウトファイルを重ね合わせて PDF ファイルを作成する。
(4) メモリデータと単票形式のレイアウトファイルを重ね合わせて PDF ファイルを作成する。

結合については、単票形式、連票形式で作成した中間ファイル（IOD ファイル）を、1 枚の PDF ファイルとして結合することができます。

以下、サンプルプログラムについて説明します。

7.3.1 CSVファイルを用いて単票形式のPDFを作成するサンプル

帳票レイアウトファイルと CSV ファイルを指定して、PDF ファイルを作成します。CSV ファイルと連携する場合、帳票レイアウトと CSV ファイルのデータを関連付けるキーマップ（cdd ファイル）が必要となります。CDD ファイルについては、CDD エディタを利用すると簡単に作成することができます。CDD エディタの使い方に関しては、専用マニュアル iothe/cddedit.pdf をご覧ください。
帳票レイアウト、CSV ファイル、CDD ファイルから PDF ファイルを作成します。



CSV ファイルを用いて単票形式の PDF を作成する方法については、以下のサンプルプログラムをご覧ください。

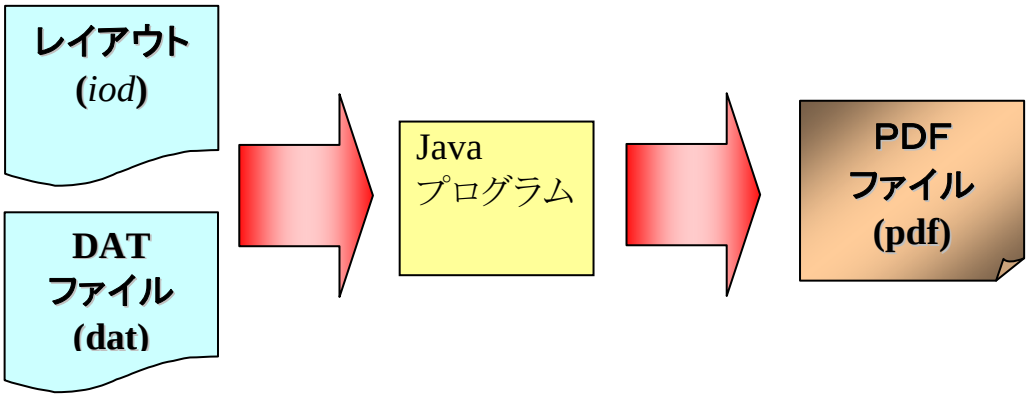
1	JavaEE 開発モデル	doc/imart/pdfmaker/sample/doccsv_act.jsp
2	スクリプト開発モデル	pages/src/pdfmaker/sample/doccsv.html
		pages/src/pdfmaker/sample/doccsv.js

7.3.2 DATファイルを用いて単票形式のPDFを作成するサンプル

帳票レイアウトファイルとDATファイルを指定して、PDFファイルを作成します。DATファイルとは、帳票レイアウトで指定した属性名と、その属性にセットする値を記述したテキスト形式のファイルです。以下にDATファイルのサンプルを示します。

※DATファイルサンプル
DATファイルは、帳票レイアウトで作成した属性名と、その属性にセットする値を記述したテキストファイルです。同名の属性名が複数存在する場合、#(連番)の形式で指定可能です。

```
Kyakusaki 株式会社 yss
NohinshoNo 100
Hinmei#1 EBW-Z1011
Hinmei#2 EBW-Z1210
Hinmei#3 EBW-Z1411
Hinmei#4 EBW-Z1612
Hinmei#5 EBW-Z1712
Hinmei#6 EBW-Z2014
Suryo#1 5
Suryo#2 5
```

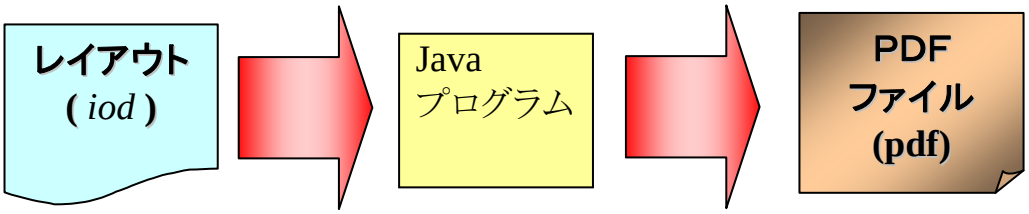


DATファイルを用いて単票形式のPDFを作成する方法については、以下のサンプルプログラムをご覧ください。

1	JavaEE 開発モデル	doc/imart/pdfmaker/sample/docdat_act.jsp
2	スクリプト開発モデル	pages/src/pdfmaker/sample/docdat.html
		pages/src/pdfmaker/sample/docdat.js

7.3.3 メモリデータを用いて単票形式のPDFを作成するサンプル

帳票レイアウトファイルを作成し、プログラム内部でデータを設定して PDF ファイルを作成します。

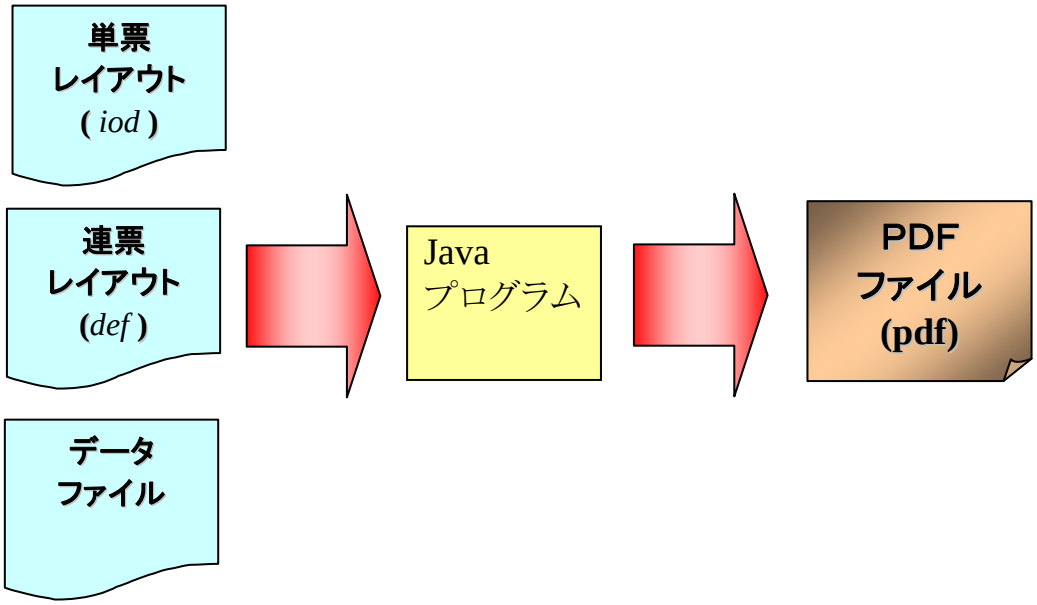


メモリデータを用いて単票形式の PDF を作成する方法については、以下のサンプルプログラムをご覧ください。

1	JavaEE 開発モデル	doc/imart/pdfmaker/sample/docobj_act.jsp
2	スクリプト開発モデル	pages/src/pdfmaker/sample/docobj.html
		pages/src/pdfmaker/sample/docobj.js

7.3.4 レイアウトを重ね合わせるサンプル

単票形式(IODoc 用)の帳票レイアウトと連票形式(IOCela 用)の帳票レイアウトを重ね合わせて、PDF ファイルを作成します。連票形式の帳票に会社ロゴを表示する等、今まで連票形式のみでは実現できなかった帳票を作成可能です。



単票レイアウトと連票レイアウトを重ね合わせて PDF ファイルを作成する方法については、以下のサンプルプログラムをご覧ください。

1	JavaEE 開発モデル	doc/imart/pdfmaker/sample/celacsvdat_act.jsp
2	スクリプト開発モデル	pages/src/pdfmaker/sample/celacsvdat.html
		pages/src/pdfmaker/sample/celacsvdat.js

7.4 サンプルプログラムの実行方法

7.4.1 PDFデザイナー付属のサンプルプログラム

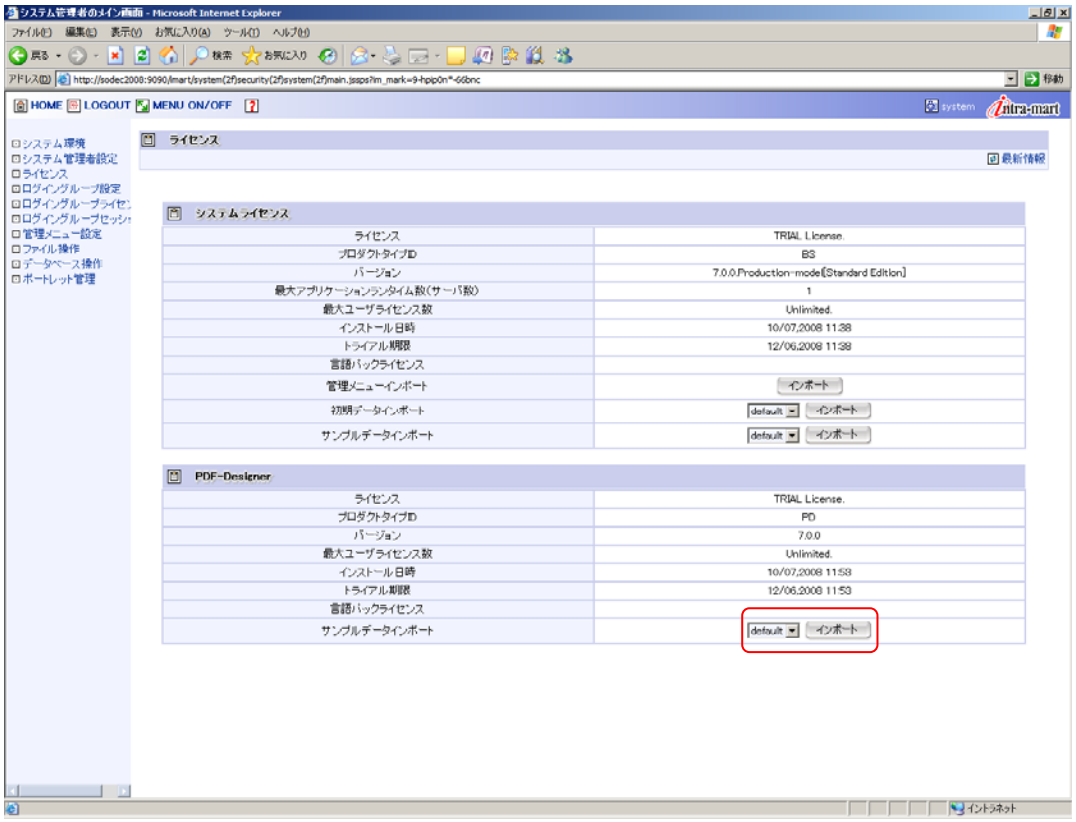
本製品に付属しているサンプルプログラムをメニューに登録することで、サンプルプログラムを実行することができます。

7.4.1.1 サンプルプログラムのメニュー登録

メニューへの登録は、以下の手順で行って下さい。

a. intra-mart サーバを起動します。
b. ユーザ master（または管理者権限を持つユーザ）でログインします。
c. [システム設定]-[ライセンス]画面を開きます。
d. 『PDF-Designer』にてメニューを登録するログイングループをプルダウンメニューで選択します。
e. 『PDF-Designer』の『インポート』ボタン(サンプルデータインポートの右にあります)をクリックします。

登録が完了したら pdfsuper というロールが追加されていますので、[システム設定]-[ユーザ]画面で、PDF デザイナーのサンプルプログラムを実行したいユーザに pdfsuper ロールを追加して下さい。そのユーザで改めてログインするとメニューに『PDF モジュール』が表示されるようになります。



7.4.2 サンプルプログラムに関する注意点

- ✓ 本製品に付属されているサンプルプログラムは、スレッドセーフではありません。従いまして、複数のスレッドで同時にサンプルプログラムを実行した場合、正しく PDF ファイルを作成できない場合があります。

- ✓ 各サンプルプログラムは、プログラムを理解しやすくするためにエラー処理を単純化して簡潔に記述されています。

- ✓ サンプルプログラムにおいて、PDF に埋め込むデータは固定値を使っています。したがって、何回サンプルプログラムを実行しても作成される PDF の内容は変化しません。

- ✓ サンプルプログラム実行時に作成される PDF ファイルは、決められたファイル名で作成されます。すでに同じファイル名のファイルが存在している場合は、作成された PDF の内容で上書きしてしまいます（元のファイルは失われます）。

- ✓ すべてのサンプルプログラムは画面プログラムとして作成されていますが、バッチプログラム内でも同様に（本製品で提供されている）PDF 作成用 API を利用することができます。

- ✓ EUC・UTF-8 環境では、添付のチュートリアル/サンプルは動作しませんのでご注意ください。

- ✓ サンプルプログラム / チュートリアルでは、説明を簡素化するためjspからファイルダウンロードを行っていますが、実運用ではサーブレットを使用してファイルダウンロードして下さい。

7.5 運用環境の構築

7.5.1 サンプルの扱い

アプリケーションの開発が完了して、いざシステムの運用環境を構築となった場合、運用環境ではサンプルプログラムが不要になるケースがでてきます。
こういった場合、サンプルプログラムおよびサンプルデータはすべて削除して下さい。サンプルプログラムおよびサンプルデータを削除したことによりシステムに障害が発生することはありません。また、サンプルプログラムを閲覧するためのメニュー登録も、運用環境では必要ありませんので、インポートする必要はありません。

7.5.2 サンプルとシステムのパフォーマンス

サンプルプログラムやサンプルデータが残っている場合、ファイルの検索速度やディスクスペースといってリソース的な側面において、不利になる場合があります。同様に、サンプルプログラムを実行するためのメニューも、インポートしたままの状態では、intra-mart サーバのパフォーマンスに影響する場合があります。
したがって、運用環境ではサンプル関連のデータやファイルはすべて破棄することを推奨致します。
また、新規に環境構築している場合は、サンプルプログラムを実行するためのメニューを登録する必要はありません。

7.5.3 サンプルの削除方法

7.5.3.1 メニューの削除

『サンプルプログラムのメニュー登録』をしている場合のみ、下記要領にしたがってメニューの削除を行って下さい。
[システム設定]-[メニュー]画面で『PDF Module』を選択して『削除』ボタンをクリックして下さい。

7.5.3.2 ロールの削除

『サンプルプログラムのメニュー登録』をしている場合のみ、下記要領にしたがってロールの削除を行って下さい。
[システム設定]-[ロール]-[ロール設定]画面で『pdfsuper ｜ PDF-Designer』を選択して『削除』ボタンをクリックして下さい。ロール『pdfsuper ｜ PDF-Designer』を他のロールのサブロールに設定している場合、そのロールのサブロール設定から『pdfsuper | PDF-Designer』を削除して下さい。
ロール『pdfsuper ｜ PDF-Designer』をユーザに対して付与していた場合、そのユーザのロール設定から『pdfsuper ｜ PDF-Designer』を削除して下さい。

8 連票（IOCELA）レイアウトの改行コード

連票レイアウトファイル（IOCELA）の改行コードは、標準で CRLF に設定されています。連票（IOCELA）のレイアウトファイルについては、中身がテキストファイルであることから、UNIX/Linux 上でご利用いただく際には改行コードを LF に変更する必要があります。

サーバにアップロードする際には、
(1) ASCII モードでサーバに FTP 転送し、自動的に改行コードを変更する。
もしくは、
(2) テキストエディタ等を利用して改行コードを LF に変更保存してから、バイナリモードで FTP 転送する。
いずれかの手法が必要になりますのでご注意ください。

9 文字枠内での「^」文字の扱いについて

IM-PDF デザイナーで CSV 形式のデータをあつかう場合に、文字枠では「^」は特別な意味をもちます。「^」は、通常の文字コードでは表せない特殊な文字（改行等…）を文字枠で使用するための文字になります。

例えば、文字枠内で改行をする場合には、記号「^」のあとに改行の文字コード「0A」を指定します。「^」のあとの指定は 16 進の文字コードである必要があります（通常は 改行コードとして「0A」を指定しますが、IM-PDF デザイナーですでに「0D」を利用している場合のみ、旧バージョンと互換性をもつために改行コードとして「0D」を指定してください）。

例） 文字枠内で改行する場合

^0A
0A が改行の 16 進の文字コードです。

例） 文字枠内で「^」を表示する場合

^5E
5E が「^」の 16 進の文字コードです。

「^」は、文字枠内での使用禁止文字ではありませんが、使用する際にはご注意ください。

10 レイアウトファイルへの画像の取り込み

PDF デザイナーには、画像を帳票レイアウトに取り込む機能があります。本機能は、変更する必要のない画像（固定の画像、例えば会社ロゴ、シンボルなど…）をあらかじめ帳票レイアウトに埋め込んでおくことで、

(1) 固定の画像を、プログラムで毎回 指定せずとも出力できるようにする。

(2) 画像ファイルと帳票レイアウトを別々のファイルとして管理するのではなく、帳票レイアウトに埋め込んで一元的に管理する。

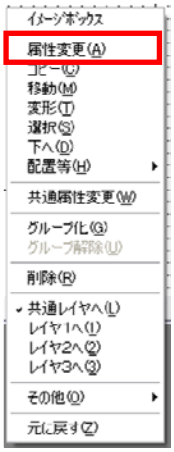
(3) 画像を変更する際も、帳票デザインツール上で変更し、プログラム側の修正が不要である。

ための機能となります。

毎回ことなる画像を指定する場合には、本機能は利用できません。プログラム側で画像ファイルのパスを渡す手法をご利用ください。

10.1 使用方法

(1) 画像領域を右クリックし、属性変更を選択します。



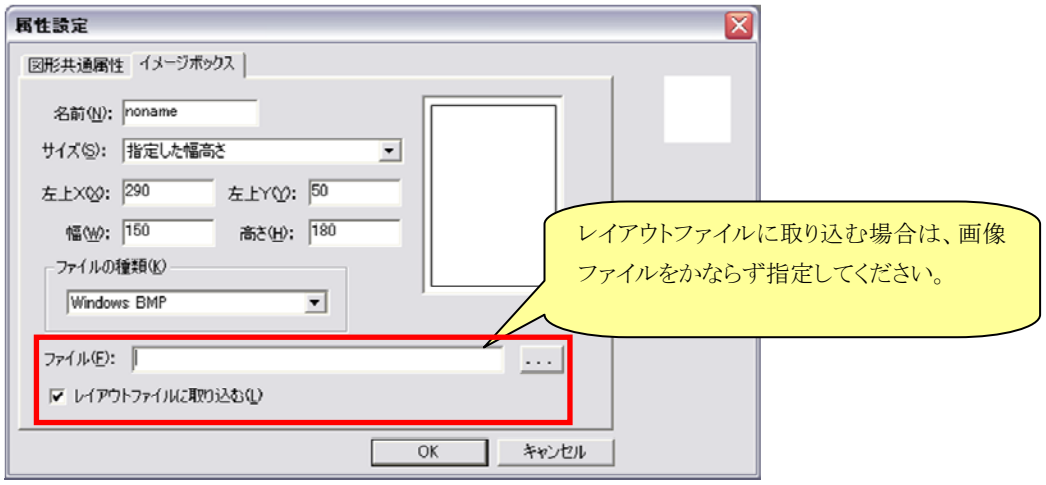
(2) 画像領域のプロパティ画面にて、“埋め込む画像ファイルのパス“ と “レイアウトファイルに埋め込む” にチェックをいれ保存します。

以上で設定は完了です。

＜注意点＞

本機能は、固定画像をあらかじめ指定するための機能ですので、プログラム側で画像ファイルを指定する場合には本設定は不要です。

また、“レイアウトファイルに取り込む” のチェックをいれる場合は、かならず画像ファイルを指定してください。“レイアウトファイルに取り込む” にチェックがはいっており、画像ファイルが指定されていない場合、PDF デザイナーは正常に動作しませんのでご注意ください。



11 エラーコード表

エラー発生時に API のリターンコードとして返されるステータスコード表です。

ステータスコード	内容
0	正常終了（エラーではありません）
－1	MS-DOS の Int21 ファンクションコール 4B00 が無効
－2	実行ファイルが見つからない
－3	パスが見つからない
－4	ファイルのオープン数エラー
－5	ダイナミックリンクライブラリ実行エラー
－6	データセグメントエラー
－7, －9	OS のメモリエラー
－8, －33	システムエラー
－10, －21	現在実行中の OS には未対応
－11, －20	実行に必要なファイルが壊れている
－12, －13	ランタイムのプラットフォームエラー
－14	ファイルタイプエラー
－15	実行ファイルのバージョンエラー
－16, －19	実行ファイルのロードエラー
－17	DLLのロードエラー
－18	アプリケーションのロードエラー
－22 ～ －32	未定義のエラー
－100	ファイルアクセスエラー 連票（IOCELA）でこのエラーが発生した場合は、レイアウトファイル（拡張子が def のファイルです）の改行コードを CRLF から LF に変換してください。改行コードの変更はテキストエディタ等で行えます。
－101	パラメータエラー
－102	メモリーエラー
－103	ランタイムモジュールの起動エラー
－104	IOWebDOC のセットアップエラー
－105	IOWebDOC ライセンスエラー
－106	印刷中のエラー
－107	直接印刷中のキャンセル
－200	セキュリティエラー（パスワードが不正等）
－999	その他のエラー
－1001	レイアウトファイルのパスが未定義
－1002	レイアウトファイルが存在しない
－1003	変換定義ファイル（cdd)ファイルのパスが未定義
－1004	変換定義ファイル（cdd）ファイルが存在しない
－1005	データファイルのパスが未定義
－1006	データファイルが存在しない

ステータスコード	内容
－1007	データが設定されていない
－1008	出力先 PDF ファイルのパスが未定義
－1009	出力先 iod ファイルのパスが未定義
－1010	データファイルのロードに失敗
－1011	iodoc ランタイム実行時エラー
－1012	IOWebDOC Java-Interface ライセンスエラー
－1020	オープンパスワードとセキュリティパスワードに同じパスワードを設定している
－1021	PDF ファイルのセキュリティパスワードが未設定
－1022	PDF ファイルのセキュリティ情報が未設定（印刷可否、編集可否等…）
その他	その他のエラー

なお、ステータスコード 0 は、リクエストされた処理を正常に終了できたことを意味しています。
したがって、ステータスコードが 0 以外の場合、エラーが発生しています。正常終了を示す 0 以外のすべての数値はエラーコードとなります。処理を正常終了できなかった場合は、メッセージ取得メソッドから、返却されたエラーコードに対応するエラーメッセージを取得できます（処理を正常終了している場合は、メッセージ取得メソッドからメッセージを取得する必要はありません）。

12 書式一覧表

レイアウト作成ツール(IODOC)で指定可能な書式の一覧です。

識別子名	書式	データ	変換結果
data1	X(20)	あいうえお	あいうえお
data2	R(20)	あいうえお	あいうえお
data3	C(20)	あいうえお	あいうえお
data4	---,---,--9	-8765	-8,765
data5	ZZZ,ZZZ,ZZ9	0	0
data6	ZZZ,ZZZ,ZZZ-	-8765	8,765-
data7	-ZZZ,ZZZ,ZZZ	-8765	- 8,765
data8	¥¥¥,¥¥¥,¥¥9	8765432	¥8,765,432
data9	¥¥¥,¥¥¥,¥¥9	0	¥0
data10	¥¥¥,¥¥¥,¥¥¥	0	
data11	-¥¥¥,¥¥¥,¥¥¥	-12345	¥-12,345
data12	¥¥¥,¥¥¥,¥¥¥-	-12345	¥12,345-
data13	@@@,@@@,@@@	12345	@12,345
data14	ZZZZZZZZZZ9	1000	1000
data15	ZZZZZZZZZZZ	0	
data16	99999999999	8765432	00008765432
data17	99999999999	0	00000000000
data18	ZZZZZZZZZ9.999	8765432.12	8765432.120
data19	ZZZZZZZZZ9.999	0.0	0.000
data20	99999999999.999	8765432.12	00008765432.120
data21	99999999999.999	0.0	00000000000.000
data22	9999^年^99^月^99^日^	20050101	2005 年 01 月 01 日

13 トラブルシューティング

トラブル	原因	対処方法
java.lang.NoClassDefFoundError が発生する	クラスパスの設定が正しくない	インストールガイドに従ってクラスパスを設定して下さい。
java.lang.UnsatisfiedLinkError が発生する	ネイティブライブラリを呼び出せない	Windows の場合は環境変数 PATH にライブラリのパスを設定して下さい。 Solaris または RedHat の場合は、環境変数 LD_LIBRARY_PATH にライブラリのパスを設定してください。 Resin 以外の WEB サーバを利用している場合は、OS の環境変数とは別に、この WEB サーバ製品固有の環境設定画面から環境変数を設定する必要があります（WebSphere、Weblogic などが該当します）。 ライブラリパスの設定に関しては、インストールガイドを参照して下さい。 JDK の Bit 数（32/64）と、IOWebDOC の Bit 数（32/64）が一致することをご確認ください。IOWebDOC（64Bit）を利用する場合、JDK（64Bit）が必須です。
エラーコード –1012 が返される	IOWebDOC Java-Interface のライセンスが不正または有効期限切れです	IOWebDOC Java-Interface のライセンスを正しく設定して下さい。
エラーコード –103 が返される	PDF 作成ランタイムに実行権限がありません	IOWebDOC をインストールしたディレクトリ直下にある bin ディレクトリ内のすべてのファイルに StorageSrv が動作しているサーバプロセスが実行できる実行権限を設定して下さい。
エラーコード －104 が返される （コンソールに「Please set up environment variable IODOC.」と表示される）	セットアップが不完全です。 （必要な環境変数 IODOC が未設定です。）	インストールガイドにしたがって環境変数を正しく設定して下さい。 （環境変数 IODOC を設定して下さい。）
エラーコード －100 が返される	ディスクがいっぱいか、ファイルにアクセスできません	十分なディスクの空き容量を確保してください。 PDF 作成に必要な各ファイルに参照および書き込み権限を設定してください。
IODoc is undefined というエラーになる IOCela is undefined というエラーになる IOIntegration is undefined というエラーになる	インストールに失敗しているか、または試用期限が切れています	正しくインストールして、ライセンスを登録して下さい。
PDFIllegalLicenseException がスローされる	試用期限が切れています	ライセンスを登録して下さい

14 アップグレード時の注意事項

14.1 廃止メソッド

setLog、setCompression メソッドについては、PDF デザイナー7.0.0 以降の新機能と併用して利用することはできませんのでご注意ください。

14.2 廃止予定のクラス

PDF デザイナー Ver7.0.0 以降、PDF 生成時にメモリデータを使用するクラスは廃止されます。互換性は確保されるため、既存システムから移行する場合は既存のソースをそのままご利用いただけます。ただし、今後 機能追加を行う場合には、廃止予定のクラスを使用しないようお願い致します。

No	クラスの廃止予定	クラス名
1	廃止（使用しないで下さい）	AbstractBuilder
2		AbstractIODOC
3	廃止（使用しないで下さい）	AbstractPageBuilder
4	廃止（使用しないで下さい）	CompressedPDF
5		CSVCela
6		CSVDoc
7	廃止（使用しないで下さい）	IOCelaPageBuilder
8	廃止（使用しないで下さい）	IOCelaPageWriter
9	廃止（使用しないで下さい）	IODocPageBuilder
10	廃止（使用しないで下さい）	IODocPageWriter
11		IOIntegration
12	廃止（使用しないで下さい）	PageWriter
13	廃止（使用しないで下さい）	PDFBuilder
14		PDFDocumentInformation
15		PDFException
16		PDFIllegalLicenseException
17		PDFIllegalParameterException
18		PDFIllegalStateException
19		PDFIOException
20		PDFLibSecurity
21		PDFMemoryAccessException
22		PDFRuntimeException
23	廃止（使用しないで下さい）	PDFSecurity
24	廃止（使用しないで下さい）	PDFWriter
25	廃止（使用しないで下さい）	SampleIOCela
26	廃止（使用しないで下さい）	SampleIODoc
27	廃止（使用しないで下さい）	SamplePDFBuilder

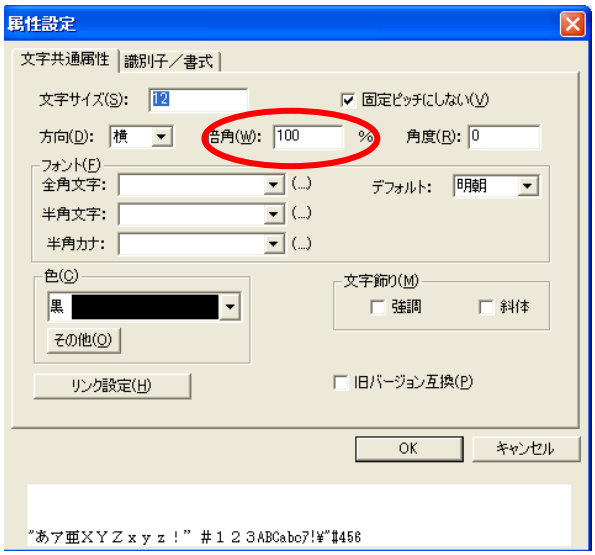
14.3 PDFデザイナー Ver5.0.x(IOWebDOC Ver1.8.x)下位バージョンとの互換性につきまして

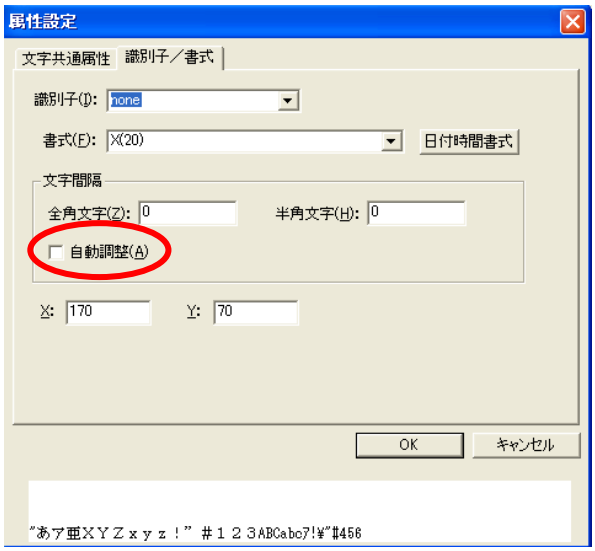
PDF デザイナー Ver7.0.0(IOWebDOC Ver1.9.1)より、単票帳票タイプ作成ツール(IODOC)において、下位バージョンとの互換性につきまして、識別子の文字共通属性の「倍角指定」と「文字間隔の自動調整」の設定において、非常にまれなケースではありますが、調整が必要な場合がありますのでお知らせいたします。

【今回の問題の対象とならないユーザ様】

- PDF デザイナー Ver5.0.x(IOWebDOC Ver1.8.x)以前をご利用で、今回 PDF デザイナー Ver7.0.0(IOWebDOC Ver1.9.1)以降にバージョンアップを予定されている方で、文字共通属性の「倍角指定」が 100%で、「文字間隔の自動調整」にチェックが入っていない場合(デフォルト値・・・変更されていない)。(通常は変更しない箇所です)
- PDF デザイナー Ver5.0.x(IOWebDOC Ver1.8.x)以前をご利用頂いており、バージョンアップの予定の無い方。
- PDF デザイナー Ver7.0.0(IOWebDOC Ver1.9.1)以降のバージョンを新規(システム)にてご利用予定の方

上記の方は今回の問題には該当しませんので、以下の文章は確認しなくても結構です。





【今回の問題の対象となるユーザ様】

- PDF デザイナー Ver5.0.x(IOWebDOC Ver1.8.x)以前をご利用で、今回 PDF デザイナー Ver7.0.0(IOWebDOC Ver1.9.1)にバージョンアップを予定されている方で、文字共通属性の「倍角指定」が 100%以外を指定されていて、「文字間隔の自動調整」にチェックが入っている場合。

【現象】

PDF デザイナー Ver5.0.x(IOWebDOC Ver1.8.x)以前において、「倍角指定」が 100%以外で、且つ「文字間隔の自動調整」にチェックが入っている場合、PDF ファイル上に置かれる位置が本来指定したところとずれているという現象が確認されました。

PDF デザイナー Ver7.0.0(IOWebDOC Ver1.9.1)以降では本来の正確な位置に来るよう修正をいたしております。以前のバージョンにてご利用のユーザ様で、このずれを逆に利用してレイアウトファイルを作成して頂いているお客様がいらっしゃる場合、PDF デザイナー Ver7.0.0(IOWebDOC Ver1.9.1)以降文字の表現位置にずれが発生する可能性があります。

【対処方法】(上記赤字の条件の該当する場合のみ対象となります。)

単票タイプ作成ツール PDF デザイナー Ver5.0.x(IOWebDOC Ver.1.8.x)のレイアウトを PDF デザイナー Ver7.0.0

（IOWebDOC Ver1.9.1）以降にて開いた際、「用紙設定」画面にて「V4.8.4 以前のテキストの互換性を保つ」のチェックを付けて保存をして頂きますと、PDF デザイナー Ver7.0.0（IOWebDOC Ver1.9.1）以降でも以前のバージョンと同じ位置に出力します。

上位バージョンへの移行の際にはご確認をお願いいたします。

15 サポート

弊社では、Web にて弊社製品に対するサポートおよび技術情報の公開を行っております。当製品に関して不明な点などがございましたら、下記 URL にてホームページにアクセスしていただき、情報検索または弊社サポート窓口までご相談下さい。

intra-mart Developer Support Site アドレス

http://www.intra-mart.jp/developer/index.html

intra-mart PDF-Designer Ver.7.0.2 Programmer's GuideBook

初版　:October 24, 2008
二版　:June 11, 2009
三版　:September 10, 2009
四版　:February 26, 2010
五版　:May 31, 2010

TEL: 03-5549-2821
URL: http://www.intra-mart.co.jp