Valence

Valenceにおける帳票出力について

株式会社ミガロ. RAD事業部技術支援課 尾崎 浩司



略歴

牛年月日:1973年8月16日 最終学歴:1996年 三重大学 工学部卒業 ミガロ入社年月:1999年10月 株式会社ミガロ、入社 社内経歴:1999年10月 システム事業部配属 2013年04月 RAD事業部配属

現在の仕事内容: Delphi/400を中心としたテクニカルサポート対応や 製品セミナーの講師などを担当している。

1.はじめに

- 2.Valenceから帳票を出力する従来の方法
- 3.Valence6.0における新しいPDF帳票出力
- 4.pdfmakeを使用したPDF帳票作成方法
- 4-1. シンプルなPDF出力処理の作成
- 4-2. 画面入力値や変数値を出力する PDF作成
- 4-3. 画像を含むPDF作成
- 4-4. QRコードやバーコードを含む PDF作成
- 5.Gridウィジェットデータを活用した 一覧帳票作成
- 5-1. ウィジェットデータの取得方法
- 5-2. 画像を追加した一覧帳票のカスタマイズ

6.さいごに

1.はじめに

Valenceは、IBM iに「最高のユーザーエクスペリエンスをも たらす」ことをコンセプトに、米CNX社が開発し、2008年よ り販売している製品である。2017年より日本国内での製品 販売と技術サポートを当社ミガロ、が担当している。取り扱 いを開始した2017年当時からIBM iを使用してモダンな Webアプリが構築できる環境として好評であったが、2018 年に登場したValence5.2でローコード開発機能App Builderが追加された事により、飛躍的にアプリの開発生産 性が向上した。さらに2020年以降はユーザーの要望を取り 入れながら、短いスパンでバージョンアップを行っており、機 能が強化されている。【図1】

Valence バージョンアップの変遷



近年Valenceは順調なバージョンアップにより、 Valenceは画面を最新化するもので、帳票出力の機能が IBM i Web化開発ツールとして、特にUI(画面)の最適化 無いと思われているかもしれないが、最新のValence6.0 が進んでいるが、最近サポートへの問合せで多いのが、 ではPDF帳票を出力する事が可能である。本稿では、 Valenceにおける帳票出力についてである。一般的には ValenceからPDF帳票を出力する手法について紹介する。

2. Valenceから帳票を出力する従来の方法

Valenceには、従来よりRPGを使用して独自ロジックを追 ToolKitの中には、動的にPDFを作成する為のAPIも用意 されている。このAPIを使用した例がサンプルプログラム1 加できるRPG ToolKitというAPIが用意されている。この ToolKitの活用例については、2019年度テクニカルレ である。これは、Formウィジェット上で商品カテゴリーを ポート『「Valence App Builder」 RPG連携テクニック』で 選択し、選択したカテゴリーに合致する商品マスターの データを一覧形式でPDFファイルに出力する処理である。 詳しく紹介しているので、そちらを参照してほしい。この 【図2】

サンプルプログラム1:RPG ToolKitの例



函 S S ЯI Д rtPad4i 杏二

Delphi/400 畑中 侑

Delphi/400 福井和彦石I

E

깸

Ð

口府

)elphi/400 E田雄一

RPGコーディング例は、【ソース1】となる。

Vース1 RPG ToolKitを使用したPDF作成(TEC21PG10)

0001.00	/copy_gcpylesrc.vvHspec
0002.00	**
0003.00	** テクニカルレポート2021
0004.00	** TEC21PG10:カテゴリ別商品情報出力
0005.00	**
0006 00	**
0007 00	/define includePDF 1
0008 00	
0009.00	**
0010 00	** program start
0011 00	**
0012 00	/free
0013 00	Initialize()
0014 00	
0015 00	
0016.00	tinlr=ton
0017.00	/end_free
0018 00	**
0019_00	n Process h
0020_00	d pi
0020.00	d pdfDoo s like(dooument)
0021.00	
0022.00	
0023.00	
0024.00	/free
0025.00	//166
0020.00	//フォームエで入りされに進を取得 VCATG = GotFormChar (' CATEGO') : //カテゴルのD
0027.00	
0028.00	//データの取得(商日マスタから指定されたわテゴルを抽出)
0029.00	
0030.00	+ (WLEDE ODOTOD = ('' + VONTO + ('')) = 1-(2)
0031.00	
0032.00	· UNDER DI UDUIUD, UDIDUD ,
0034 00	//DNFファイル保友生パスの指定
0035 00	TMDDATH - willtility getValenceSetting('TEMD_DATH')
0036.00	ImitAtti - Wottitity_BetWatenoeoetting(TEmi_IAtti);
0030.00	//デーク取得結果を示に動めにDDEを作成
0038.00	vyPDE path = %trim(TMPPATH) + 'TEMP MPRONP pdf'
0030.00	pdfDoc = wPDE newDocument(wPDE):
0040_00	vvPdf_addTablefromSQL(vvPDF;pdfDoc:SQLSTR);
0041 00	vvPDF_closeDocument(odfDoc):
0042 00	
0043_00	//動的に作成されたPDFをダウンロード
0044 00	$y/0 \pm down \log d = '1'$:
0045_00	v_{0} t_{1} t_{1} t_{2} t_{1} t_{2} t_{1} t_{2} t_{1} t_{2} t_{1} t_{2} t_{1} t_{2} t_{2
0046.00	what file (wPDE path what)
0040.00	
0047_00	
0047.00	//動的に作成されたPDFをIFSとから削除
0047.00 0048.00 0049.00	//動的に作成されたPDFをIFS上から削除 wylfs_deleteFile(wPDF_path):
0047.00 0048.00 0049.00	//動的に作成されたPDFをIFS上から削除 vvlfs_deleteFile(vvPDF.path); /end=free
0047.00 0048.00 0049.00 0050.00	<pre>//動的に作成されたPDFをIFS上から削除 vvlfs_deleteFile(vvPDF.path); /end-free p</pre>
0047.00 0048.00 0049.00 0050.00 0051.00 0052.00	<pre>//動的に作成されたPDFをIFS上から削除 vvlfs_deleteFile(vvPDF.path); /end-free p e /include.gcpvlesrc.vvNabBtp</pre>

 PDF出力には、vvPDFというAPIを使用するが、このAPIを
 SQL実行結果より一覧表PDFが作成できる。最後にIFS上

 使用する場合、ソースの宣言部に1-①のような宣言を追加
 のPDFをブラウザにダウンロードさせる処理が1-④である。

 すればよい。1-②で商品マスターからカテゴリーCDが合致
 このように、従来からValenceでは、RPG ToolKitを使用す

 するデータを抽出する為のSQL文を作成している。そして、
 る事で、動的なPDF帳票が作成できた。ただ、この方法に

 1-③がPDFを作成しIFS上に保存する処理である。
 は、課題があり、残念ながら日本語を含む文字列は、ブラン

 [vvPdf_addTablefromSQL]というAPIを使用すれば、
 クで出力されてしまうのである。【図3】





Valence 尾崎 浩司

SmartPad4i 國元祐二

Delphi/400 俎中 侑

Delphi/400 福井和彦石山

깸

Ð

Delphi/400 佐田 雄一



このAPIは、開発元CNX社によるとIBM i側のPDF作成エン ジンを使用しているとの事なのだが、このエンジンには残念 ながら、日本のIBM iマシンであっても日本語フォントが標準 搭載されておらず、日本語文字が欠落してしまうとの事であ る。折角ValenceにはPDF作成機能が用意されているにも関 わらず、これまでこの機能を積極的にアピールしてこなかっ たのは、この為である。 もちろん、ValenceはIBM iを使用している為、従来通りス プールを使用した帳票は活用できるわけだが、PDFを作成 したり、表現力のある帳票を作成したい場合、【図4】のよう に外部の帳票ツールと連携する、或いは【図5】のように Delphi/400等を組み合わせて独自の帳票処理を作成する のが、これまでは一般的であった。



図5 ValenceとDelphi/400との連携イメージ



3. Valence6.0における新しいPDF帳票出力

Valenceは、2020年12月にバージョン6.0となったが、この バージョンからGridウィジェットのダウンロード機能にお いて従来のExcelに加えてPDF形式が選択できるように なった。【図6】

ウィジ	ェットの編集 "TE	CREP21_商	品一覧グリッドPDFタ	ブウンロード _.
	_		Grid ぇ「ダウ	ウィジェット ンロード」項
カラム	カテゴリー	斎品CD ↑	商品名	商品力ナ
\$	01:家電製品	K-10001	全自動洗濯機	27.2971-7922
RE	01:家電製品	K-10002	スチームアイロン	スチームアイロン
רא ג אוג א	01:家電製品	K-10003	4 ドア冷蔵庫レッド	4ኑ" 7ሁለን" ዕጋ
ß	01:家電製品	K-10004	4 ドア冷蔵車ブラック	4ኮ* 7ሁብሃ* 9ጋ
OF設定	01:家電製品	K-10005	3 ドア冷蔵庫ホワイト	31*71/49*03
œ	01:家電製品	K-10006	電気ジャーボットホワイト	F">\$>">=#
版送	02: 寶趨製品	O-10001	A V サラウンドテレビ 3 3	サラウント** デルと**33
	02:音響製品	O-10002	A Vサラウンドテレビ 2 9	サラウント" デルと" 29

このPDF形式を選択したGridウィジェットを使用してアプ リケーションを作成し、実行すると一覧データがPDF形式 でダウンロードできる事が分かる。しかも、生成された PDFを確認すると、日本語文字列を含めて正しく出力され ているのである。【図7】



画面



留中 宿	Delphi/400
福井 和彦 石山 智也	Delphi/400
依田 雄 —	Delphi/400
國元 祐二	SmartPad4i
尾崎 浩司	Valence

なぜ、日本語を含む文字列が正しく出力できるようになった のか再度開発元に確認したところ、バージョン6.0では従来 のIBM iエンジンを使用する方法とは別にクライアント(ブ ラウザ)の機能を使用してPDFを作成できる方法を採用した ことがわかった。

詳細を確認してみたところ、新たにpdfmakeと呼ばれるライ ブラリがValence環境に追加されたのである。pdfmakeは、 JavaScriptを使用してクライアントサイドで動的なPDFを作 成できるオープンソースライブラリである。【図8】

JavaScriptベースのPDF作成ライブラリ 図8

 ・ ・	il pdfmake.org/index.html#/ 🗮 88 🏠 🗘
раттаке	PEATORES DOCUMENTATION PLATGROUND SOF
ndfmako	
punnake	
Client/server side PDF	printing in pure JavaScript
dient-version bower install pdfr server-version npm install pdfm	ake Valence6.0より、Valence環境にpdfmakeライブラルが 組み込まれるようになった。これにより、クライアントサイドで 動的なPDF作成が可能になった。
In-browser Print PDFs directly in the browser or	【pdfmake公式サイト】 http://pdfmake.org/index.html#/ delegate it to your Nodej5 backend. Use the same document definition in both cases.
Fully declarative Forget about manual x, y calculation	s. Declare document structure and let pdfmake do the rest.

実はpdfmakeも標準では日本語の出力には対応していない のだが、独自のフォントを追加できるようになっている。日本 版のValence環境にはデフォルトで「あおぞら明朝フォン ト」と呼ばれる日本語フォントが組み込まれている為、日本 語を含む文字列のPDFが出力できるのである。 この新しいPDF出力機能をApp Builderの標準機能として 使用できるのは、2021年8月現在Grid/Edit Gridウィジェッ トに追加されたPDFダウンロード機能のみであるが、このラ イブラリはJavaScriptを使用する事で簡単にPDFを作成で きる為、独自のPDF出力処理をApp Builderアプリケーショ ンに追加する事ができる。具体的な作成方法を次節で紹介 する。

4. pdfmakeを使用したPDF帳票作成方法

App Builderにおいて、ユーザーの操作に応答するアク プトの実行」が定義できるようになり、クライアントブラウ ション(処理)は「動作内容」画面で設定できる。従来からこ ザ上で実行可能な処理(JavaScript)が定義できるように の機能を設定する事で、例えばRPGプログラムの連携等 なった。**【図9】** が実装できたわけだが、Valence6.0からは新たに「スクリ



この機能を使用すれば、pdfmakeを使用した独自の帳票が 作成できる。ここでは具体的な作成方法の紹介として、まず 始めにシンプルなPDF出力アプリケーションを作成する。

Valence6.0より、アクションを定義する「動作内容」 画面において、新たに「スクリプトの実行」が定義



4-1. シンプルなPDF出力処理の作成

サンプルプログラム2は、Formウィジェットに配置した [PDF出力]ボタンをクリックすると、「ミガロ. テクニカルレ

ポート」という日本語文字列を持つPDFファイルをダウン ロードするというシンプルなPDF出力処理である。【図10】

図10 サンプルプログラム2:シンプルなPDF出力処理



App Builderの「アプリケーション作成」ステップにおける 「動作内容」定義画面で、[PDF出力]ボタンに対し、アクショ ン追加より「スクリプトの実行」を選択すると、スクリプトを 記述する為のスクリプトエディタ画面が開く。エディタ上で "loadpdf"と入力すると、エディタに搭載された「入力コード 支援機能」により、入力候補として、"load Pdf Make"が表示 されるので、それを選択する。するとライブラリpdfmakeを 読み込む為のテンプレートソースが自動的に追加される。 【図11】

図11 pdfmake 読込処理をスクリプトに追加



084 MIGARO Technical Report

"//ロード済み"と書かれたコメント部分がpdfmakeの機	
能が記述できる部分となる。ここに独自のPDFを生成し、	

ソース2 サンプルプログラム2:シンプルなPDF出力処理

functio	n(rec, success){		
1	// PDFMakeを起動		
2 *	<pre>loadPdfMake().then(function () {</pre>		
3	// Valenceに導入済みの日本語フォントを指定		
4 -	pdfMake.fonts = {		
5 -	Aozora: {		
6	normal: 'AozoraMinchoRegular.ttf',		
7	bold: 'AozoraMincho-bold.ttf'		
8	}		100
9	};		_
10			
11	// PDFレイアウト作成		
12 -	<pre>var docDefinition = {</pre>		
13	//用紙指定		
14	pageSize: 'A4',	11	用紙
15	pageOrientation: 'landscape',	11	向ぎ
16	pageMargins: [40, 60, 40, 60],	11	余白
17			
18	//出力する内容		
19	content:		
20 -	[
21	{text: 'ミガロ・テクニカルレポート'}	11	文字
22],		
23			
24	//スタイル (書式) 指定		
25 *	defaultStyle: {		
26	font: 'Aozora'	11	フォ
27	}		
28	};		
29			
30	// PDF生成		
31	<pre>var pdfDocGenerator = pdfMake.createPdf(docDefi</pre>	nit:	ion);
32	// 生成したPDFを出力(下記のいすれかを実行)	- 17	
33	pdfDocGenerator.download(' T X F.pdf'); //PDF?	89	0.01
34			
35 -	}, function (err) {		
36	//エラー		
37	3);		

出力する処理を記述すればよい。実際に記述したソース例が【ソース2】である。







2-①の部分が、pdfmakeに組み込まれた日本語フォントを 指定する部分である。これを書かないと日本語文字列が出 力できないので、このまま埋め込むようにしてほしい。 2-②が帳票を作成する部分で、この例のようにpdfmakeで は、帳票のレイアウトの作成をdocDefinitionという変数に 対し、JSON形式で値をセットする形で行う。一見複雑そうだ が、実際のPDFレイアウトは、この中の"content"という要素 である。この例では一行だけ「ミガロ.テクニカルレポート」と いう文字列をテキスト出力項目(text)として定義しているの である。このソース例では追加情報として、別途用紙サイズ や向き、余白を設定しているが、これは無くても良い。また最 後の部分で"defaultStyle"を定義しているが、これは2-①で 定義したフォントをこのレイアウトで使用するという設定な ので、日本語を出力する際にはこの指定は必須となる。 2-③がPDFを生成し、結果をダウンロードさせる部分であ る。"pdfMake.createPdf(docDefinition)"というメソッド を呼び出す事で、2-②で定義したレイアウトを使ってPDFを 生成でき、生成された結果をpdfDocGeneratorという変数 に代入している。あとは生成されたPDFをどのように処理す るかを記述すればよく、この例ではdownload()というメ ソッドを使用してブラウザ上にダウンロードさせるようにし ている。この部分はdownload()以外にopen()あるいは print()といった指定もできる。それぞれの実行結果は【図 12】のようになる。

図12 pdfmakeを使用したPDF出力方法



4-2. 画面入力値や変数値を出力するPDF作成

次にウィジェット上で入力した値やValenceで定義できる アプリ変数の値等をPDF帳票に埋め込む方法を紹介する。 サンプルプログラム3は、Formウィジェット上で選択した 商品カテゴリーの値(フィールド名:CATEGO)とValence にデフォルトで定義されたアプリ変数であるIBMiユー ザー/Valenceユーザーの値をPDFに出力するものであ る。【図13】

図13 サンプルプログラム3:画面入力値/



このPDF出力処理のソース例は、【ソース3】である。

function(rec, success){	
1 /	/ PDFMakeを記動	
2 - 10	padPdfMake().then(function () {	
3	// 日本語フォント定義	
4 -	pdfMake.fonts = {	
5 -	Aozora: {	
6	normal: 'AozoraMinchoRegular	.ttf'.
7	bold: 'AozoraMincho-bold.t	tf'
8	}	
9	};	
10	// PDFレイアウト作成	
11 -	<pre>var docDefinition = {</pre>	
12	//出力する内容	
13 -	content: [
14	{text: 'ウィジェット上の値	や変数の値を出力', fontSize: 24},
15	{text: ' '},	
16	{text: '選択カテゴリーは【	' + rec.get('CATEGO') + '] です。'}
17	{text: 'IBM i User = ' + g	etAppVar('nabIbmI') },
18	{text: 'Valence User = ' +	<pre>getAppVar('nabUser') }</pre>
19	1.	
20	//スタイル(書式)指定	
21 -	defaultStyle: {	
22	font: 'Aozora'	// 日本語フォント
23	}	
24	};	
25	// PDF生成	
26	var pdfDocGenerator = pdfMake.crea	tePdf(docDefinition);
27	// 生成したPDFを出力	
28	pdfDocGenerator.open(); // PDFを	開く
29 -	<pre>}, function (err) {</pre>	
30	//エラー	
31	});	

先程と同様"content"要素内でPDFレイアウトを作成して また、getAppVerメソッドがApp Builderのアプリケー いるのだが、ここでは、recというオブジェクト変数や ションに定義された「アプリ変数」値を取得するものであ る。このようにアプリケーション実行時にユーザーがウィ ジェットに入力した値や、内部的に保持しているアプリ変 スする為に用意された変数で、getメソッドを使用すれば かの値を使用してPDF帳票を作成するのも容易であるこ とがわかる。

´変数の値をPDF出力	
されたPDFファイル (画面プレビュー) +	I
ト上の値や変数の値を出力	
ロットで指定した条件値や 値がPDF出力される。	
値/変数の値をPDF出力	
3-①	
, //ウィジェット上のフィールド //アブリ変数 (IBMiユーザー) //アブリ変数 (Valenceユーザー)	

尾崎 浩司	國元 祐二	存田 雄 –	福井 哲彦 石山 踏也	笛 中 街
Valence	SmartPad4i	Delphi/400	Delphi/400	Delphi/400

4-3. 画像を含むPDF作成

次は、画像をPDF出力する方法を紹介する。Valenceを使用 して画像情報を取得する為には、Valence専用のメソッドを 使用する。スクリプトエディタ上で"getImage"と入力する と、「入力コード支援機能」により画像取得処理のテンプレートが追加できる。【図14】

図14 画像読み込み処理(getImage)



【図14】の処理だが、(a)の部分が画像を読み込むメソッドと なっている。画像の読込が完了すると、取得した画像情報が imageDataにセットされるので、取得後の画像処理を(c)の 部分に記述すればよい。但し、このメソッドには注意点があ る。画像の読込処理自体は非同期で行われる為、(a)の後続 処理部分である(b)の部分は、(a)の実行後に即実行されて しまう点である。(c)の部分は、画像情報が取得された後に 実行される為、(c)より(b)の部分の方が先に実行されてし まうのである。この処理順序に対する対処例は、本稿の後半 (サンプルプログラム8)で紹介する。

ここでは、getImageメソッドを使用した具体的なサンプル を紹介する。サンプルプログラム4は、Formウィジェット上 の[PDF出力]ボタンをクリックすると、IFS上に保存された Valenceのロゴ画像をPDF出力するものである。【図15】

図15 サンプルプログラム4:画像データをPDF出力



サンプルプログラム4のソース記述例は【ソース4】である。



4-①が画像を読み込む部分となる。IFS上にあるValence びPDF生成を行っている。pdfmakeでは4-③のように画の口ゴ画像(/resources/images/valence_logo.png)
 を取得している。画像が取得できれば、4-②の部分が実行
 される為、この部分でPDFファイルのレイアウト作成およ
 びPDF生成を行っている。pdfmakeでは4-③のように画のように画の口ゴ画像(/resources/images/valence_logo.png)



088 MIGARO Technical Report

ゑデータから PI	DF出力
	- 1
nction (imageData) {	4-①
4-3	- 1
語フォント	- 1
tion);	- 1
	- 1

peipni/400 peipni/400 smartPad41 國元 祐二
--

4-4. QRコードやバーコードを含むPDF作成

帳票出力においてニーズが高いのが、QRコード/バーコード の出力である。pdfmakeはQRコードも容易に作成できる。 サンプルプログラム5は、Formウィジェット上に入力した値

を使用して動的にQRコードを作成し、それをPDF出力する ものである。【図16】

図16 サンプルプログラム5:QRコードをPDF出力

Formウィジェット ^図 Wakes 60 ← → ♂ △ ▲ MIRATAUS QRコードを含むPDFを作成し出力	ールド: CODE) リンをクリックすると、 する。 生成されたPDFファイル (画面プレビュー)
 ■ valence ORコードを含むPDFを作成 QRコード出力 *ORCODE に出力したい文字がを入力してください。 https://www.migard.co.jp 	 ● ● ○ ○ ▲ #####1/121 beam#y/1221 ● ■ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●

サンプルプログラム5のソース記述例は【ソース5】である。

unction	n(rec, success){	
1	// PDFMakeを起動	
2 -	<pre>loadPdfMake().then(function () {</pre>	
3	// 日本語フォント定義	
4 -	pdfMake.fonts = {	
5 *	Aozora: {	
6	normal: 'AozoraMinchoRegular.ttf'	,
7	bold: 'AozoraMincho-bold.ttf'	
8	}	
9	};	
10		
11	// フォーム上の入力値を取得	- (1)
12	<pre>const CodeVal = rec.get('CODE'); // image: const CodeVal = rec.get('CODE');</pre>	· 面入力值 5-(1)
13		
14	// PDFレイアウト作成	
15 *	<pre>var docDefinition = {</pre>	
16	//出力する内容	
17 -	content: [
18	{text: 'QRコードの出力', fontSi	ize: 24},
19	{text: '入力值'},	
20	<pre>{text: CodeVal, fontSize: 16},</pre>	
21	{text: '},	- 0
22	{qr: CodeVal}	// QR⊐ - F 5-(2)
23],	
24	//スタイル (書式) 指定	
25 *	defaultStyle: {	
26	font: 'Aozora'	// 日本語フォント
27	}	
28);	
29	// PUF 主 放	(1-0-(1-1+1))
30	var patDocGenerator = pdtMake.createPdt	r(docuerinition);
31	// 主成したPDFを出力	
32	patuocGenerator.open(); // PDF @ M <	
33 *	}, function (err) {	
34	//17-	
35	333	

ソース5 サンプルプログラム5:QRコードをPDF出力

おり、取得した変数を使用して5-②のようにQRコード項目 (qr)に文字列をセットするだけなので、とても簡単である。 次にバーコードだが、pdfmakeには残念ながらバーコード

5-①でFormウィジェット上のフィールド入力値を取得して



このライブラリを使用すれば、任意の文字列から動的に が、App Builderの場合、「ポータル管理」の「設定」ページ バーコード画像を作成する事ができる。こういった外部の の中で外部ライブラリを追加できるようになっている。今 ライブラリを組み込む場合、一般的にはHTMLの中に 回は、JsBarcodeのCDNを設定した。【図18】 JavaScriptが参照するライブラリのCDNを定義するのだ



090 MIGARO Technical Report

を出力する機能は含まれていない。従ってバーコードを出 力するには一工夫必要となる。今回は、JsBarcodeと呼ば れるJavaScriptでバーコードを作成できるライブラリを活 用する。【図17】

Q 22 0 弱合 スを指定する。 おれているCDNを設定。		
Q № 0 第合 スを指定する。 おれているCDNを設定。 sealtrins		
易合 スを指定する。 れているCDNを設定。 see all min.1s	٩	8 0
合 スを指定する。 されているCDNを設定。 sealtrin.is		
スを指定する。 れているCDNを設定。		
わているCDNを設定。	スを指定する	5.
de alt min.is	れているCD	Nを設定。
	ide. all. min. js	

phi/400 E	Delphi/400	Delphi/400	SmartPad4i	Valence
	量井 和彦 石山 智也	佐田 雄 —	國元祐二	尾崎 浩司

留り

サンプルプログラム6は、Formウィジェット上でコードの値 とバーコードの種類を指定して、動的にバーコードを作成 し、それをPDF出力するものである。【図19】



サンプルプログラム6のソース例は【ソース6】である。

inction(r	ec, success){
1 //	/ PDFMakeを記動
2 - 10	padPdfMake().then(function () {
3	// 日本語フォント定義
4 -	pdfMake.fonts = {
5 -	Aozora: {
6	normal: 'AozoraMinchoRegular.ttf',
7	bold: 'AozoraMincho-bold.ttf'
8	}
9	};
10	
11	// フォーム上の入力値を取得]
12	const CodeVal = rec.get('CODE'); // 入力したコード
13	const TypeVal = rec.get('TYPE'); // バーコード種類 (EAN13, CODE39, CODE128, CODABAR,)
14	
15	// キャンパスを作成しパーコードを発行
16	var canvas = document.createElement('canvas'); - 6-(2)
17	JsBarcode(canvas, CodeVal, {format: TypeVal});
18	
19	// PDF U 1 2 U F TERX
20 -	Var docuetinition = {
21	//표기 9 이사용
22 *	(fort; [
25	(text: // = = = = = = = = = = = = = = = = = =
24	(text: A) in 5
25	(text: Turola) fontSize: 16)
20	(text: 'y)eval, foncsize: 10/,
28	(reaction); (rea
29	[[[moge. canvas.cobacaone(), widen. zoo, neight. zoo, neight. zoo, neight.
30	//スタイル (書式) 指定
31 -	defaultStyle: {
32	font: 'Aozora'//日本語フォント
33	}
34	}
35	// PDF生成
36	<pre>var pdfDocGenerator = pdfMake.createPdf(docDefinition);</pre>
37	// 生成したPDFを出力
38	pdfDocGenerator.open(); // PDFを開く
39 -)	<pre>+, function(err) {</pre>
40	//エラー

Delphi/400 俎中 侑 6-①でFormウィジェット上で入力したコードとバーコー にcanvas画像をセットすればよい。 ドの種類を取得している。6-②は、動的に画像要素 本節ではpdfmakeを使用して動的にPDFを生成する例を (canvas)を生成し、JsBarcodeメソッドを使用してバー 紹介したが、pdfmakeは帳票レイアウト作成にJSONを使 コード画像を生成している。このメソッドではパラメータに 用する。定義の詳細は公式サイトのドキュメントに詳しく 画像要素、コード文字列、そしてバーコードの種類を指定 載っているので、そちらを参照してほしいのだが、サイトの する。すると画像要素に生成された画像データが格納され 中に[PLAYGROUND]というページがあり、ここでは色々 Delphi/400 福井和彦石山 るので、後は6-③のようにpdfmakeの画像項目(image) な要素の実装例を確認する事ができる。【図20】 pdfmake PLAYGROUNDページ 図20 깸 Ð pdfmake.org/playground.html x + \star 🖆 🚇 🙎 pdfmake FEATURES DOCUMENTATION PLAYGROUND SOURCES Delphi/400 佐田 雄一 /8 Q - + - + Tables Official documentation pdfmake and its layout A simple table (no headers, no width specified, no sp Column 2 Column 3 here Another one here OK? Column 1 A simple table with nested elements Column 2 SmartPad4i 國元祐二 Defini pdfmake 公式サイト: [PLAYGROUND]ページ 画面左側には、定義の記述例、右側に生成されるPDF例が olumo will I am auto sized. 表示される。 左側の定義は編集でき、リアルタイムに変更点が確認できるよう になっている。 Valence 尾崎 浩司 この[PLAYGROUND]では左側の定義部分を直接変更 する事もでき、その結果はリアルタイムに右側の帳票レイ アウトに反映される。実際に帳票レイアウト設計する際に は、まずここで定義を試してみるとよいだろう。





5. Gridウィジェットデータを活用した一覧帳票作成

本節では、ウィジェット上に表示されているデータソースの する。さらに一覧表に画像情報を追加するカスタマイズ方法 内容から動的に一覧表形式のPDFを生成する方法を紹介 も紹介する。

5-1. ウィジェットデータの取得方法

サンプルプログラム7は、Gridウィジェットに表示されてい のである。【図21】 る商品マスターの内容を一覧表の形式でPDFに出力するも

サンプルプログラム7:ウィジェットデータのPDF出力 図21



ウィジェット上のデータは次のようにすれば取得できる。ス クリプトエディタ上で"getWidgetData"と入力すると表示 される「入力コード支援機能」の候補を選択する事でテンプ レートが自動的に追加される。追加されたメソッドの中にあ る"getWidget('WidgetIdOrName')"の部分に、データを

取得したいウィジェットを指定すればよいのだが、これはス クリプトエディタ右側の「ウィジェット」タブに表示される ウィジェット一覧から目的のウィジェットを選択すればよ い。選択したウィジェットが内部で保持しているウィジェット IDが自動的にセットされる。【図22】

ウィジェットデータの取得処理(getWidgetData) 図22



094 MIGARO Technical Report

これで目的のウィジェット上のデー タがスクリプトの中で取得できるよ うになり、取得したGridウィジェッ トの一覧データがwidgetData変 数にレコード件数分の配列として 格納される。

取得したデータから一覧形式の PDFを作成する際に使用するの が、pdfmakeの表(table)という要 素である。先程紹介したpdfmake の[PLAYGROUND]で"table"要 素の基本形を作成してみたのが、 【図23】である。このように表を示す "table"要素の子要素としてbody 部を定義し、その中に表のレイアウ トを作成すればよい。

この仕組みを使用して作成したサ ンプルプログラム7のソース例が 【ソース7】である。

7-①で宣言しているtableBody変 数が、動的にbody部を作成する為 に用意した配列である。7-②では、 表のタイトル行を作成している。 7-③では、forループを使用して、 wigdetData変数に格納された ウィジェットデータの配列をレコー ド単位で順番に取得しながら、 tableBody変数に値をセットして いる。そして、7-④の部分で tableBody変数をpdfmakeの表 項目(table)のbody部にセットし ている。





【ソース7】を組み込んだアプリケーションを実行すると、 Gridウィジェット上に一覧表示された商品マスターの情報

を元に、表形式の一覧PDFファイルを動的に作成する事が できる。【図24】



5-2. 画像を追加した一覧帳票のカスタマイズ

次に、この一覧表PDFをカスタマイズする例を紹介する。 今回はIFSの中にある"/resources/images/products/" ディレクトリに"[商品CD].jpg"という形式で格納された商 品画像を使用する。先程作成した商品マスターの一覧表に

画像列を追加して、画像付きの一覧表が出力できるように変 更する。完成したサンプルプログラム8からPDFを出力す ると次のような実行結果となる。【図25】

図25	
-----	--

サンプルプログラム8によって生成されたPDF



096 MIGARO Technical Report

前節で紹介した画像の取得方法であるgetImageメソッド の部分でgetImageメソッドを使用してIFS上にある"[商品 を【ソース7】に追加すると、【ソース8】のような形になると考 CD].jpg"というファイル名の画像を取得し、画像が取得で えられるだろう。処理を追加した部分は次のとおりである。 きた場合に、画像フィールドの列情報をtableBody変数へ 8-①でタイトル行に"画像"列を追加している。そして、8-② 追加している。

ソース8 サンプルプログラム8:画像項目を追加(※動作しない)

<pre>var tableBody = []; // 表の要素を保持する変数 // VDFMakeを読込 4 loadDoffMake().then(function () { 5 // Valencell、導入深かの日本語フォントを指定 6 pdfMake.fonts = { 7 accors: { 7 normal: 'AccorationchoRegular.ttf', 9 bold: 'AccorationchoRegular.ttf, accorationch</pre>	functio	n(rec. success){
<pre>// PDFLAGE REAL // Valencel: #\.\Argebols = { // Fights = for (1 = 0; i < widgetData.length; i++) { var rowData.publ(text: widgetData[]].F1_ODPDCD]); // #G rowData.publ(text: widgetData[]].F1_ODPDCN]); // #G rowData.publ(text: widgetData[]].F1_ODPDCN]); // #G rowData.publ(text: widgetData[]].F1_ODPDCN]); // #G rowData.publ(text: widgetData[]].F1_ODPNN]; // #G rowData.publ(text: widgetData[]].F1_ODTANN, alignment: rowData.publ(text: widgetData[]</pre>	1	var tableBody = []: // 表の要素を保持する変数
<pre>3 // PDFMakeを読込 4 loadD4TMake().then(function () {</pre>	2	
<pre>4 loadPdfNake().then(function () { 5</pre>	3	// PDFMakeを読込
<pre>> // Valencel:導入済みの日本語フォントを指定 > pdfNake.forms = { Aozora: { normal: 'AozoraMinchoRegular.ttf', bold: 'AozoraMincho-bold.ttf' bold: 'AozoraMincho-bold.ttf' j } // フィジェットデータの取得 retuidgetData(getWidget('1279')).then(function (widgetData) { //取得したデータを支加をBookの規列へ移始 // 別タイトルを要素に追加 rowData.push(Teal: vidgetData[1].fl_OOPDCN)); // 商品 rowData.push(Text: widgetData[1].fl_OOPDCN)); // #位 //</pre>	4 -	<pre>loadPdfMake().then(function () {</pre>
<pre>6 pdfNake.fonts = { Accora! { Accora! { Accora! {</pre>	5	// Valenceに導入済みの日本語フォントを指定
<pre>/* Accoral: 'AccoraMinchoRegular.ttf', bold: 'AccoraMincho-bold.ttf' } //ワイジェットデータの取得 //股得したデータをtableBody配列へ結約 //取得したデータをtableBody配列へ結約 //取得したデータをtableBody配列へ結約 //</pre>	6 *	pdfMake.fonts = {
<pre>bold: 'Accorabincho-bold.ttf' bold: 'Accorabincho-bold.ttf' i i i i i i i i i i i i i i i i i i i</pre>	7.	Aozora: {
<pre>> Dol: According (Dollar) ************************************</pre>	8	normal: 'AozoraMinchoKegular.ttf',
<pre>10 }; 11 }; 12 // クィジェットデータの取得 13 // ジョクイトルを要素に追加 14 beBody.publ(「商品CD', 商品名', カナ', 単価', 単位', 雪 15 // 取得したデータを要素に追加 17 tableBody.publ(「商品CD', 商品名', カナ', 単価', 単位', 雪 18 // 取得したデータを要素に追加 19 for (i = 0; i < widgetData.length; i++) { 19 for (i = 0; i < widgetData.length; i++) { 10 varouData.publ([text: widgetData[i].F1_OOPON]); // 商品 10 rowData.publ([text: widgetData[i].F1_OOPON]); // 商品 10 rowData.publ([text: widgetData[i].F1_OOPON]); // 商品 10 rowData.publ([text: widgetData[i].F1_OOPON]); // 単位 11 rowData.publ([text: widgetData[i].F1_OOPON]); // 単位 12 rowData.publ([text: widgetData[i].F1_OOPON]); // 単位 13 rowData.publ([text: widgetData[i].F1_OOPON]); // 単位 14 rowData.publ([text: widgetData[i].F1_OOPON]); // 単位 15 rowData.publ([text: widgetData[i].F1_OOPON]); // 単位 16 rowData.publ([text: widgetData[i].F1_OOPON]); // 単位 17 mGata.publ([text: widgetData[i].F1_OOPON]); // 単位 18 rowData.publ([text: widgetData]].F1_OOPON]); // 単位 19 rowData.publ([text: widgetData]].F1 20 rowData.publ([text: widgetData]].F1 21 rowData.publ([text: widgetData]].F1 22 rowData.publ([text: widgetData]]. 23 rowData.publ([text: "面像Data] { 24 rowData.publ([text: "面像Data] { 25 rowData.publ([text: "面像Data] { 26 rowData.publ([text: "面像Data] { 27 rowData.publ([text: "面像Data] { 28 rowData.publ([text: "面像Data] { 29 rowData.publ([text: "面像Data] { 20 rowData.publ([text: "面像Data] { 20 rowData.publ([text: "面像Data] { 21 rowData.publ([text: "面像Data] { 22 rowData.publ([text: "面像Data] { 23 rowData.publ([text: "面像Data] { 24 rowData.publ([text: "面像Data] { 25 rowData.publ([text: "mamedata] { 27 rowData.publ([text: "面像Data] { 28 rowData.publ([text: "面像Data] { 29 rowData.publ([text: "mamedata] { 20 rowData.publ([text: "mamedata] { 21 rowData.publ([text: "mamedata] { 22 rowData.publ([text: "mamedata] { 23 rowData.publ([text: "mamedata] { 24 rowData.publ([text: "mamedata] { 25 rowData] { 26 rowData.publ([text: "mamedata] { 27 rowData] { 28 rowData.publ([text: "mamedata] { 29 rowData] { 20 rowData] { 20</pre>	10	Bold: Adzonamineno-bold.cer
<pre>// ウィジェットデータの取得 // 取得したデータをtableBody配別へ格納 // 取得したデータをtableBody配別へ格納 // 取得したデータを要素に違加 tableBody.push(「商品でD・,商品名、,カナ,,単価,,単位,」 // 取得したデータを要素に違加 rowData.push(text: widgetData[1].F1_OOPDNN); // 商品 rowData.push(text: widgetData[1].F1_OOPDNN); // 商品 rowData.push(text: widgetData[1].F1_OOPNNI); // 商品 rowData.push(text: widgetData[1].F1_OONNI); // 第合 rowData.push(text: widgetData[1].F1_OONNI]; // 単位 // 商品面像の取得(商品でD.jgをじうフィルをで getImage(ingPath).then(function (imageData) { // 画像取得成功の場合 rowData.push(text: "面像素し")); // , function (err) { // 画像取得成功の場合 rowData.push(text: "面像素し")); // }, function (err) { // この取得(rowData); } // I 面像取得の現者、TomBata(I); // } // I 面像取得の取者(rowData); } // I 面像取得の現者、TomBata(I); // // I 面像取得の開合 i // I = // I // I // I // I // I // I /</pre>	11	3:
<pre>// クィジェットデータの取得 getWidgetData(getWidget('1279')).then(function (widgetData) { // 取得したデータをableBody配列へ格納 // 別タイトルを要素に追加 tableBody.push('m品CD','商品名','カナ','単価','単位','単位 war rowData = []; rowData.push('text: widgetData[].F1_OOPDCD]); // 商品 rowData.push('text: widgetData[].F1_OOPDKN}); // 商品 rowData.push('text: widgetData[].F1_OOPDKN}); // 商品 rowData.push('text: widgetData[].F1_OOPDKN}); // 商品 rowData.push('text: widgetData[].F1_OOPNKN}); // 商品 rowData.push('text: widgetData[].F1_OOPNKN}); // 「商品 rowData.push('text: widgetData[].F1_OOPNKN}); // 「商品 rowData.push('text: widgetData[].F1_OOPNKN]); // 「商品 rowData.push('text: widgetData[].F1_OOPNKN]); // 「商品 rowData.push('text: widgetData[].F1_OONNNI); // 単位 // 商品固像の取得 (商品CD.jpgというファイル名で getImage(imgPath).then(function (imageData) { // 「一個像取得な方の場合 rowData.push((text: "面像是し")); // // 面像取得な方の場合 rowData.push((text: "面像是し")); // }, function (err) { // 「一個像取得太子-の場合 rowData.push((rowData); // 「一個像取得なう-の場合 rowData.push((rowData); // 「一の目をしかいため(rowData); // 「一の目をしかいため(rowData); // LableBody.push(rowData); // 「一般電報でのます。', fontSize: 16), (table: { headerRows: 1, width: ['auto','aut</pre>	12	
<pre>14- getWidgetData(getWidget(1279')).then(function (widgetData) { 15 // 取得したデータをtableBody配列へ格給 17 tableBody.push(「商品 CD', 「商品名', 'カナ', '単価', '単位', 'E 18 // 取得したデータを要素に追加 19- for (i = 0; i < widgetData.length; i++) { 19-</pre>	13	// ウィジェットデータの取得
<pre>15 // 取得したデータをtableBody配列へ格納 16 // 列タイトルを要素に追加 17 tableBody.push([*簡品CD', *商品名', *力ナ', 準価', 準位', *重 18 // 取得したデータを要素に追加 19 for (i = 0; i < uidgetData.length; i++) { 20 var rowData.push({text: widgetData[i].F1_OUPDKN}; // 商品 21 rowData.push({text: widgetData[i].F1_OUPDKN}; // 商品 22 rowData.push({text: widgetData[i].F1_OUPDKN}; // 常品 23 rowData.push({text: widgetData[i].F1_OUPDKN}; // #位 24 rowData.push({text: widgetData[i].F1_OUPDKN}; // #位 25 rowData.push({text: widgetData[i].F1_OUPDKN}; // #位 26 const imgPath = <u>//resources/imageS/products/' + widgetData[i]</u> 27 rowData.push({text: widgetData[i].F1_OUDNNT}; // #位 28 const imgPath = <u>//resources/imageS/products/' + widgetData[i]</u> 29 rowData.push({image: imageData, width: 85}); // 31 rowData.push({image: imageData, width: 85}); // 32 rowData.push({image: imageData, width: 85}); // 33 rowData.push({text: "面像素し"}); // 34 rowData.push({text: "面像素し"}); // 35 rowData.push((rowData); 36 47 rowData.push((rowData); 39 } 40 41 //PDFレイアウト作成 42 var docDefinition = { 43 //LäbieBody.push(rowData); 44 //LäbieBody: tableBody 55 j. 56 //Lapie: { 57 rowData.push(est) fig: 58 rowData.push(with: [*auto', *auto', *auto'</pre>	14 -	<pre>getWidgetData(getWidget('1279')).then(function (widgetData) {</pre>
<pre>16</pre>	15	// 取得したデータをtableBody配列へ格納
<pre>17 17 17 17 17 17 17 17 17 17 17 17 17 1</pre>	16	// 列タイトルを要素に追加
<pre>13 // 取得したテータを要素に追加 19 - for (= 0; 1 < widgetData.length; 1++) { 19 - var rowData = []; 10 var rowData.push({text: widgetData[].F1_OOPDCD)}; // 商品 10 rowData.push({text: widgetData[].F1_OOPNN}); // 商品 10 rowData.push({text: widgetData[].F1_OOPNN}; // 商品 10 rowData.push({text: widgetData[].F1_OOPNN}; // 商品 10 rowData.push({text: widgetData[].F1_OOPNN}; // 単位 10 rowData.push({text: widgetData[].F1_OOPNN}; // 単位 10 rowData.push({text: widgetData[].F1_OOPNN}; // 単位 10 rowData.push({text: widgetData[].F1_OOPNN}; // 単位 11 rowData.push({text: widgetData[].F1_OOPNN}; // 単位 12 rowData.push({text: widgetData[].F1_OOPNN}; // 単位 13 rowData.push({text: mageData, width: 85}); // 14 rowData.push({text: "面像取得本Data]; 15 rowData.push({text: "面像無見#Data]; 16 rowData.push((text: "面像無L"]); // 17 rowData.push((text: "面像無L"]); // 18 rowData.push(rowData); 19 } 10 rowData.push(rowData); 11 rowData.push(rowData); 12 rowData.push(rowData]; 13 rowData.push(rowData]; 14 rowData.push(rowData]; 15 rowData.push(rowData]; 16 rowData.push(rowData]; 17 rowData.push(rowData]; 18 rowData.push(rowData]; 19 } 10 rowData.push(rowData]; 10 rowData.push(rowData]; 10 rowData.push(rowData]; 11 rowData.push(rowData]; 12 rowData.push(rowData]; 13 rowData.push(rowData]; 14 rowData.push(rowData]; 15 rowData.push(rowData]; 16 rowData.push(rowData]; 17 rowData.push(rowData]; 18 rowData.push(rowData]; 19 rowData.push(rowData]; 10 rowData.push(rowData]; 11 rowData.push(rowData]; 12 rowData.push(rowData]; 13 rowData.push(rowData]; 14 rowData.push(rowData]; 15 rowData.push(rowData]; 16 rowData.push(rowData]; 17 rowData.push(rowData]; 18 rowData.push(rowData]; 19 rowData.push(rowData]; 10 rowData.push(rowData]; 10 rowData.push(rowData]; 11 rowData.push(rowData]; 12 rowData.push(rowData]; 13 rowData.push(rowData]; 14 rowData.push(rowData]; 15 rowData.push(rowData]; 16 rowData.push(rowData]; 17 rowData.push(rowData]; 18 rowData.push(rowData]; 19 rowData.push(rowData]; 19 rowData.push(rowData]; 10</pre>	17	tableBody.push(['商品CD','商品名','カナ','単価','単位','@
<pre>19 - for (1 = 0; 1 < widgetData.length; 1++) % 20 21</pre>	18	// 取得したテータを要素に追加
20 Var rowData.push({text: widgetData[i].F1_OOPDCD)); // 商品 rowData.push({text: widgetData[i].F1_OOPDCN)); // 商品 rowData.push({text: widgetData[i].F1_OOPNN); // 商品 rowData.push({text: widgetData[i].F1_OONN, alignment: rowData.push({text: midgetData[i].F1_OONN, alignment: rowData.push({text: "mages/products/" + widgetD getImage(imgPath).then(function (imageData) { // imgtpPath).then(function (imageData) { // imgtpPath).then(function (imageData) { // imgtpPath).then(function { imgtpAth, alignment); }); 33 // imgtpPath.then(function { imgtpAth, alignment); }); // imgtpPath imgtpAth, alignment); }); 34 // imgtpPath, alignment); // imgtpAth, alignment); }); // imgtpAth, alignment); // imgtpAth, alignment, alignment); // imgtpAth, alignment, alignmen, alignment, alignmen	19 -	<pre>for (i = 0; i < widgetData.length; i++) {</pre>
<pre>11 12 12 12 13 13 14 14 14 15 15 14 15 15 15 15 15 15 15 15 15 15</pre>	20	var rowData = [];
<pre>22</pre>	21	<pre>rowData.push({text: widgetData[i].ri_ODPDLD}); // monopulata.push({text: widgetData[i].ri_ODPDNM}); // mplane</pre>
<pre>1</pre>	22	nowData nuch(/text: widgetData[i].ri_ODPD(N)); // m.m.
<pre>25 25 26 27 28 29 29 29 30 31 31 32 32 33 34 35 34 35 34 35 35 36 37 38 39 39 30 30 30 30 31 32 33 34 35 35 36 37 38 49 40 40 41 41 41 42 42 43 44 44 44 44 45 45 44 45 44 45 45</pre>	24	rowData.push({text: widgetData[i].F1_ODTANK, alignment:
26 // 商品画像の取得(商品CD.jpgというファイル名で更 29- getImage(imgPath).then(function (imageData) { // 画像取得成功の場合 rowData.push((image: imageData, width: 85}); // 33 , function (err) { // 画像取得エラーの場合 rowData.push((image: imageData, width: 85}); // 35 36 rowData.push((image: imageData, width: 85}); // 37 // 画像取得エラーの場合 70 71 72 73 74 75 76 77 78 79 70 71 72 73 74 75 76 77 77 77 78 79 70 71 <th>25</th> <th>rowData.push({text: widgetData[i].F1 ODUNIT}): // 単位</th>	25	rowData.push({text: widgetData[i].F1 ODUNIT}): // 単位
<pre>// 商品画像の取得(商品CD.jpgというファイル名で通 const imgPath = '/resources/imageS/products/' + widgetD getImage(imgPath).then(function (imageData) {</pre>	26	
<pre>28 ▲ 29 - 30 30 31 ▲ 32 - 31 ▲ 32 - 33 34 35 35 36 37 38 39 39 30 30 30 30 31 30 31 32 33 34 35 35 36 37 37 38 40 41 41 41 42 - var docDefinition = { 72 //tableBodyに要素を追加 42 - var docDefinition = { 72 //tbJす & PJ # 44 - content: [45 - { tableBody.r要素を追加 44 - content: [45 - { tableBody.rwsh(rowData); 46 - { tableBody.rwsh(rowData); 47POFL 4 7 7 } ftk 48 - { tableBody.rwsh(rowData); 49</pre>	27	// 商品画像の取得(商品CD.jpgというファイル名で通
<pre>▲ 29 - 30 31 32 - 33 33 34 35 34 35 35 36 4 3 3 3</pre>	28	<pre>const imgPath = '/resources/images/products/' + widgetD</pre>
<pre>30 31 32 33 34 35 35 35 36 36 36 36 37 37 38 38 48 40 41 38 40 41 41 41 41 42 42 42 43 44 44 44 44 44 44 44 44 45 45 45 46 47 45 46 47 46 47 48 48 48 49 49 40 41 49 50 50 50 50 50 50 50 50 50 50 50 50 50</pre>	🛕 29 🕶	<pre>getImage(imgPath).then(function (imageData) {</pre>
<pre>31 32 33 34 34 35 35 36 36 37 7 //tableBodyl.要素を追加 tableBody.push(rowData); 39 39 3 4 4 4 4 4 5 4 4 5 5 5 5 5 5 5 5 5 5 5</pre>	30	// 画像取得成功の場合
<pre> 32 33 34 35 34 35 35 36 37 36 37 38 38 38 38 39 39 40 41 39 40 41 39 40 41 39 40 41 39 40 41 39 40 41 42 42 42 42 44 44 44 44 44 44 45 45 45 45 45 45 45</pre>	31	<pre>rowData.push({image: imageData, width: 85}); //</pre>
<pre>>>> // 回藤秋(年よ) ~ (金) >>> // 回藤秋(日よ) ~ (金) >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	A 32 *	}, function (err) {
<pre>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	30	// 回版取得エジーの場合 nouData nuch(/taxt: "面像無」。"})・ //
<pre>//tableBodyに要素を追加 tableBody.push(rowData); //tableBody.push(rowData); //tableBody.push(rowData); //tableBody.push(rowData); //tableBody.push(rowData); //tableBody.push(rowData); //tableBody.tableBody //tableSody.tableBody //tableSody.t</pre>	35	}):
<pre>//tableBodyに要素を追加 tableBody.push(rowData); } //tableBody.push(rowData); } //tableBody.push(rowData); //tableBody.push(rowData); //tableSometry { //table</pre>	36	10
<pre>38 tableBody.push(rowData); 39 40 41 //PDFレイアウト作成 42 var docDefinition = { 43</pre>	37	//tableBodyに要素を追加
<pre>39 } 40 //PDFレイアウト作成 41 //PDFレイアウト作成 42 var docDefinition = { 43 //比力する内容 44 content: [45 {text: 商品一覧表', fontSize: 16}, 46 {table: { 60 {table: { 61 {table: { 62 {table: { 63 {table: { 64 {table: { 64 {table: { 65 {table: { 65 {table: { 65 {table: { 77 {table: { 78 {table: { 78 {table: { 79 {table: { 70 {table: { 70 {table: { 71 {table: { 73 {table: { 73 {table: { 73 {table: { 73 {table: { 74 {table: { 75 {table: { 75 {table: { 75 {table: { 76 {table: { 77 {table: { 77 {table: { 78 {table: { 79 {table: { 79 {table: { 70 {table: { 70 {table: { 71 {table: { 71 {table: { 71 {table: { 72 {table: { 73 {table: { 73 {table: { 73 {table: { 74 {table: { 75 {table: { 75 {table: { 75 {table: { 76 {table: { 76 {table: { 77 {table: { 76 {table: { 77 {table: { 76 {table: { 77 {table: { 77 {table: { 77 {table: { 77 {table: { 76 {table: { 77 {table: { 76 {table: { 77 {table: { 77 {table: { 76 {table: { 77 {table: { 76 {table: { 77 {table: { 76 {table: { 76 {table: { 77 {table: { 76 {table: { 76 {table: { 76 {table: { 76 {table: { 77 {table: { 76 {table: {</pre>	38	tableBody.push(rowData);
<pre>40 41 41 42 41 42 43 44 44 44 44 44 45 45 45 46 45 46 46 47 48 48 49 49 49 49 49 49 49 50 51 51 52 54 55 54 5 55 55 55 55 55 55 55 55 55 5</pre>	39	}
<pre>41 //PDFレイアフト作成 42 var docDefinition = { 43</pre>	40	
<pre>42 var docDefinition = { //出力する内容 content: [ftext: '商品一覧表', fontSize: 16}, ftext: '面品一覧表', function (err) { // エラー ftext: '面品一覧表', fontSize: 16}, ftext: '面品」表示 ftext: '面品」 ftext: '面は ftext: '面は</pre>	41	//PDFレイアウト作成
<pre>43 //エリタ OVP 44 - content: [45 { 45 { 45 { 46 - { 47 { 48 { 49 { 50 { 51 { 53 { 53 { 54 - { 53 { 53 { 54 - { 55 { 55 { 56 { 57 { 58 { 59 { 59 { 50 { 50 { 50 { 50 { 51 { 52 { 53 { 53 { 54 - { 55 { 55 { 56 { 57 { 58 { 59 { 59 { 50 { 50 { 50 { 50 { 50 { 50 { 51 { 52 { 53 { 54 - { 55 { 55 { 56 { 57 { 58 { 59 { 59 { 59 { 50 { 50 { 50 { 50 { 50 { 50 { 50 { 50 { 51 { 52 { 53 { 54 - { 55 { 55 { 56 { 57 { 58 { 59 { 59 { 59 { 59 { 59 { 59 { 59 { 50 { 59 { 50 { 59 { 59 { 59 { 59 { 59 { 50 { 59 { 59 { 59 { 50 {</pre>	42 *	var docDefinition = {
<pre>24* 25 46* 45 46* 46* 47 7 7 8 48 49 49 49 50 51 52 53 7 53 7 54 7 55 5 7 55 7 55 7 55 7 55</pre>	43	//出力する内容
<pre>46 - {table: {</pre>	44 *	content: [/text: '商品一覧表' fontSize: 16]
<pre>47 48 47 48 49 49 49 50 51 52 53 53 54 55 55 55 55 55 55 55 55 55 55 55 55</pre>	45	{table: {
<pre>48 49 49 49 49 50 50 51 52 53 53 7 54 64 64 55 5 6 7 55 5 7 5 8 7 5 7 5 8 7 5 7 5 8 7 5 1 9 5 7 5 1 9 5 7 5 5 7 5 5 7 5 5 7 5 7 5 7 5 8 7 7 5 8 7 7 5 8 7 7 7 7</pre>	47	headerRows: 1.
<pre>49 body: tableBody } 50 } 51 } 52], 53 //スタイル(書式)指定 64 defaultStyle: { 55 } 56 } 57 }; 58 // PDF生成 59 var pdfDocGenerator = pdfMake.createPdf(docDefinition); 60 // 生成したPDFを出力 61 pdfDocGenerator.open(); // PDFを開く 62 }; function (err) { 63 //エラー 64 }); 65 }; 71 //エラー 67 });</pre>	48	width: ['auto', 'auto', 'auto', 'auto', 'auto', 'auto', 'aut
50 51 52 53 53 53 54 54 55 56 57 57 58 77 58 77 58 77 59 59 59 59 59 59 59 59 59 59	49	body: tableBody
<pre>51 } 52 } 53 //スタイル (書式) 指定 54 defaultStyle: { 55</pre>	50	}
<pre>52], //スタイル(書式)指定 53 //スタイル(書式)指定 54 defaultStyle: { 55 defaultStyle: { 56 } 57 }; 58 // PDF生成 59 var pdfDocGenerator = pdfMake.createPdf(docDefinition); 60 // 生成したPDFを出力 61 pdfDocGenerator.open(); // PDFを開く 62 }, function (err) { 63 //エラー 64 }); 65 }, function (err) { 66 //エラー 67 });</pre>	51	}
<pre>53</pre>	52],
54* defaultStyle: { font: 'Aozora' // 日本語フォント 55 } 57 }; 58 // PDF生成 59 var pdfDocGenerator = pdfMake.createPdf(docDefinition); 60 // 生成したPDFを出力 61 pdfDocGenerator.open(); // PDFを開く 62 - }, function (err) { 63 //エラー 64 }); 65 - }, function (err) { 66 //エラー 77 };	53	//スタイル (書式) 指定
<pre>55</pre>	54 *	defaultStyle: {
<pre>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	55	
<pre>5% // PDF生成 5% var pdfDocGenerator = pdfMake.createPdf(docDefinition); 60 // 生成したPDFを出力 61 pdfDocGenerator.open(); // PDFを開く 62 - }, function (err){ 63 //エラー 64 }); 65 - }, function (err) { 66 //エラー 67 });</pre>	50	1.
<pre>var pdfDocGenerator = pdfMake.createPdf(docDefinition); 60 // 生成したPDFを出力 61 pdfDocGenerator.open(); // PDFを開く 62 - }, function (err) { 63 //エラー 64 }); 65 - }, function (err) { 66 //エラー 67 });</pre>	58	// PDF生成
<pre>60 // 生成したPDFを出力 61 pdfDocGenerator.open(); // PDFを開く 62 }, function (err){ 63 //エラー 64 }); 65 }, function (err) { 66 //エラー 67 });</pre>	59	<pre>var pdfDocGenerator = pdfMake.createPdf(docDefinition);</pre>
<pre>61 pdfDocGenerator.open(); // PDFをM< 62</pre>	60	// 生成したPDFを出力
62 · }, function (err){ 63 //⊥ラ− 64 }); 65 · }, function (err) { 66 //⊥ラ− 67 });	61	pdfDocGenerator.open(); // PDFを開く
63 //Iラ→ 64 }); 65 - }, function (err) { 66 //Iラ→ 67 });	62 -	<pre>}, function (err){</pre>
64 }); 65-}, function (err) { 66 //エラ− 67 });	63	//エラー
65 °), tunction (err) { 66 //I ∋ − 67 });	64	<pre>});</pre>
67 });	65 -	<pre>}, function (err) {</pre>
<i>ii j</i> / <i>i</i>	65	11-5-
	0/	1/)

	- 1
	- 1
<u>\$(\$``]</u>); 8-(1)	
CD 名 カナ 'right'}); // 単価	
可像を取得) ata[i].F1_ODPDCD + '.jpg';	8-2
画像を出力	
画像無しメッセージ	
	- I
// 列タイトル行 o'], // 列幅指定 // tableBady(赤の専	*)
// capiebooy(4X/0/#	**)
	- 1
	- 1

始中 宿	Delphi/400	
福井和彦 石山 智也	Delphi/400	
佐田 雄一	Delphi/400	
國元 祐二	SmartPad4i	
尾崎 浩司	Valence	

一見この【ソース8】は正しい処理に見えるかもしれないが、 残念ながら、このプログラムは実行しても帳票は出力されな い。なぜならば、getImageメソッドの項で紹介した通り、画 像の取得処理自体は非同期で行われる為、この記述方法だ と画像が取得されるより前にループが繰り返されてしまうの である。

この課題を解決する為に処理を書き直したのが、【ソース 9-1&2】である。ウィジェットデータを取得する getWidgetDataメソッドの処理結果部分を大きく変更し た。

今回、1明細分のtableBodyを作成する為のmakeDetailサ ブルーチン(9-②)と、PDFを作成、出力する為のmakePDF

サブルーチン(9-④)を新たに定義して処理を分割した。この 処理では、まず始めに9-①が実行されて、タイトル行を設定 した後に、9-②を呼び出している。9-②では画像取得処理で あるgetImageメソッドを使用しているが、今回は画像取得 が完了してから、次の明細行の処理に進むように、9-③のよ うな形でmakeDetailサブルーチンを再帰呼び出しする実 装としている。明細行数分だけ再帰呼び出しを行い、最終行 の処理が終わったら、最後に9-④を呼び出して、PDFを生成 しているのである。少し複雑な処理となってしまったが、今 回のような場合に、再帰呼び出し処理は有用なので、是非こ のソースを参考にしてほしい。

ソース9-1 サンプルプログラム8:再帰呼び出し処理に修正





6. さいごに

本稿では、Valenceで実現可能なPDF帳票作成方法につ 面からのリスト出力や各種定型帳票の出力等には充分活 いて紹介してきた。従来バージョンのValenceでは、日本 用できる。 語を含むPDFを出力する事ができず、帳票ツール等との組 JavaScriptを使用したコーディングによる帳票作成となる 為、若干のJavaScriptスキルが必要とはなるが、pdfmake による帳票レイアウト作成の手順さえ習得できれば、今回 のサンプルソースを参考に十分活用できるので、皆様も ValenceからのPDF帳票出力にチャレンジして頂ければ 幸いである。

み合わせが現実的な手法であったが、Valence6.0では Valence単体で日本語を含むPDFが動的に生成できるよ うになった。 クライアントサイドでのPDF作成となる為、バッチ処理の 実行結果を帳票印刷するような用途にはそぐわないが、画



Valen 尾崎 ※
