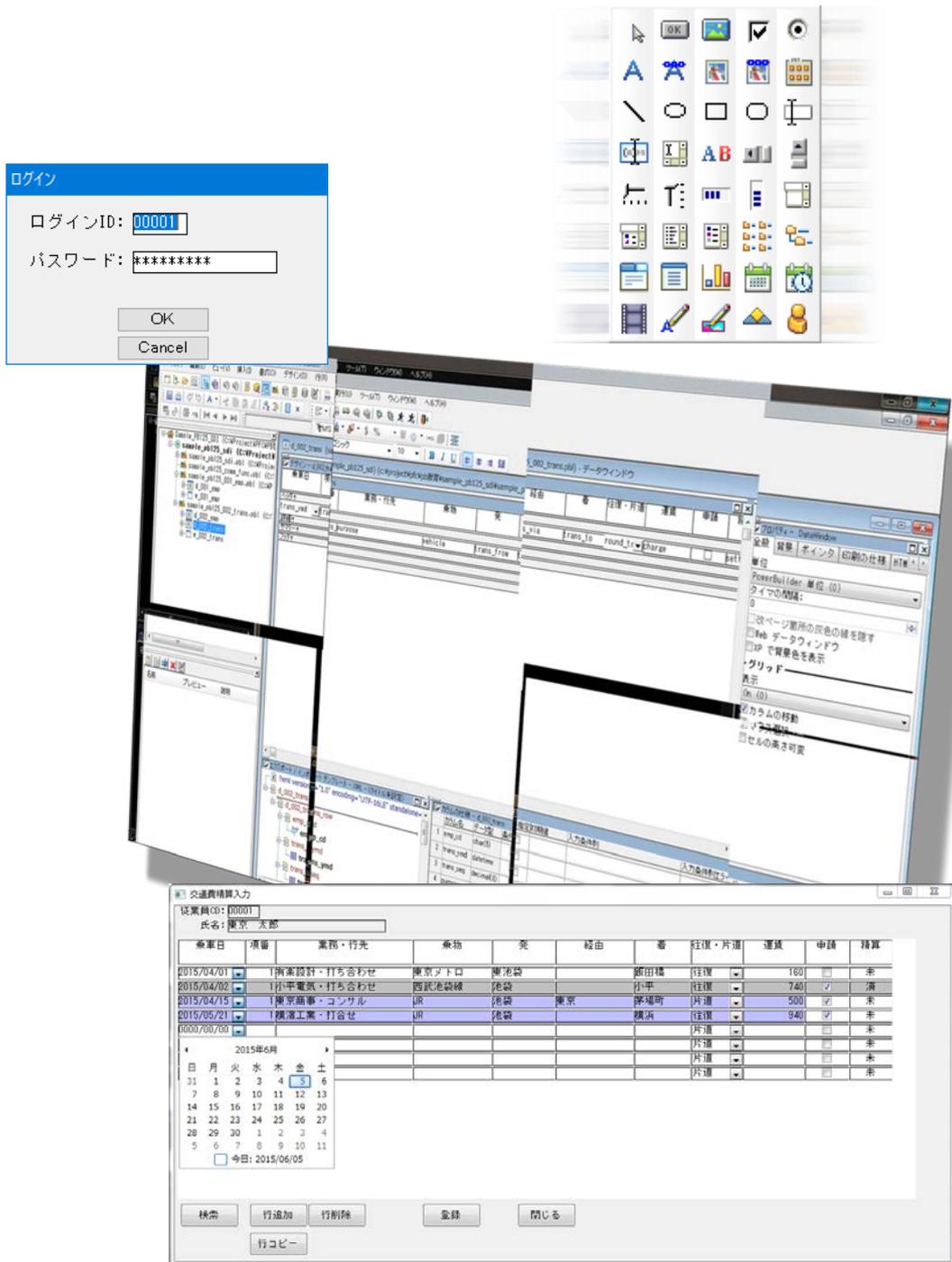


Appeon PowerBuilder 2017 入門コース



パワーフューチャー株式会社

目次

はじめに	3
1. Appeon PowerBuilder 2017 ソースの構成	4
(1) PBLファイル	4
(2) オブジェクトファイル	4
(3) オブジェクトファイルの概要	5
(4) オブジェクトへのプログラミング	8
(5) 変数の宣言	9
(6) 外部関数の宣言	10
(7) 命名規約	10
2. データウインドウ	12
(1) データウインドウとは	12
(2) データウインドウの特徴	13
(3) データウインドウの種類	14
(4) データウインドウのオブジェクト (dwo)	16
(5) データウインドウのカラムの表現方法	18
3. Appeon PowerBuilder 2017 開発手順	19
(1) ワークスペースの作成	19
(2) ターゲットの作成	20
(3) システムオプションのシステムフォント	22
(4) アプリケーションの付加的なプロパティの設定	23
(5) DB 接続	25
(6) アプリケーションのコーディング	29
(7) ログイン画面の作成	33
(8) メニュー画面 (暫定) の作成	40
(9) アプリ、ログイン、メニューの連結	41
(10) PBL ファイルの追加	42
(11) 従業員メンテナンスの作成	44
(12) 従業員マスタメンテナンスをメニューから起動	53
(13) 交通費精算画面の作成	54
(14) 交通費精算画面をメニューから起動	67
4. まとめ	68

はじめに

本コースは、これから **Appoon PowerBuilder 2017** の開発や運用保守を行う技術者向けに、入門レベル以上の達成度を目標に下記 3 ステップにより構成している。

ステップ 1 では、**Appoon PowerBuilder 2017** を用いてプログラミングを行うために、最低限の知識として必要となるソース構成について説明する。

ソース構成を知らずしてプログラムを組み立てることはできないため、**Appoon PowerBuilder 2017** が初めての場合は必読とする。

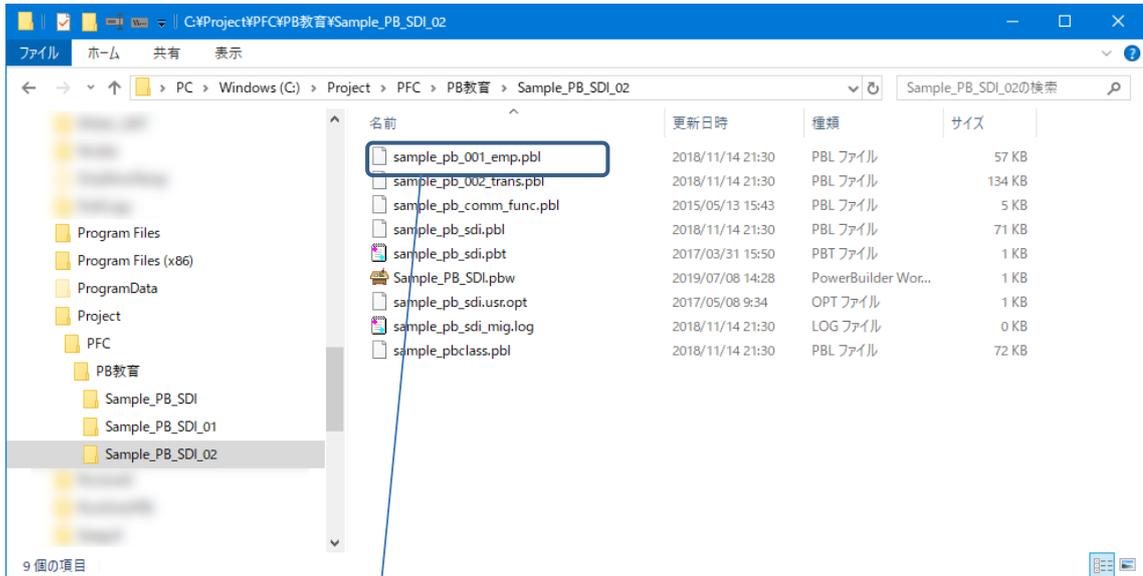
ステップ 2 では、データウインドウのリッチなユーザインタフェースを理解し、システム設計・構築を行う上で、優位性と効率的な視点に立てる解説を行う。

ステップ 3 では、**Appoon PowerBuilder 2017** アプリケーションの作成方法を説明する。IDE の操作に力点を置かず、実際にプログラムを作りながら操作方法を交えた解説をする。本コースでは、**Appoon PowerBuilder 2017** のクラスライブラリ機能を使用せずに、DB 操作アプリケーションの構築方法に注力する。

1. PowerBuilder ソースの構成

(1) PBLファイル

Apeon PowerBuilder 2017 のソースは、PBL ファイル（バイナリー）で PC 上に保存される。

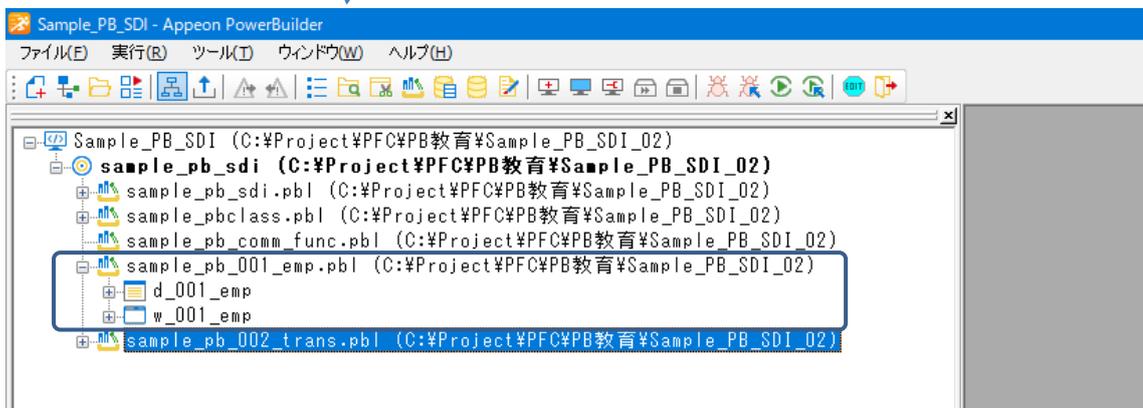


(2) オブジェクトファイル

PBL ファイルの中には、ウインドウやデータウインドウ、関数など、アプリケーション構築に必要なオブジェクトが登録され、Apeon PowerBuilder 2017 IDE によりソースの編集が可能になる。

各オブジェクトは、共通機能や処理画面ごとに PBL ファイルにまとめることで管理しやすくなる。

また、ソース管理ツール（VSS）を用いた開発においても、PBL 単位で役割分担がしやすくなるメリットがある。



(3) オブジェクトファイルの概要

本書では、イメージし易いように SDI (シングルドキュメントインタフェース) アプリケーションを前提に記載する。

MDI (マルチプルドキュメントインタフェース) は Office 製品のようにフレーム内で複数のウインドウが操作でき、メニュー項目を選択して操作する構成のため、説明範囲が広がり難解になるため割愛する。

① アプリケーション

アプリケーションを定義する非表示オブジェクトである。

アプリケーションの開始と終了の処理、後述するグローバル変数やグローバル外部関数の宣言を行う。

② トランザクション

Appeon PowerBuilder 2017 アプリケーション内で DB 接続、DB 操作内容を保持する非表示オブジェクトである。

デフォルトでは SQLCA トランザクションが提供されている。

複数のトランザクションが必要な際には、別名で Create して使用することができる。

その際には、ソースのハードコーディングが必要で、事前にユーザオブジェクト (後述) として作成することもできる。

CONNECT によりトランザクションが開始され、COMMIT で DB へ書込み情報を確定させ、ROLLBACK で書込み情報を破棄、DISCONNECT でトランザクションを終了する。

※重要

CONNECT 文の実行における負荷は、DB ベンダー製品によって異なるが、DB 接続が完了するまでに要する時間は大きい。

最善策としては、環境にもよるがアプリケーションの最初に 1 回だけ発行する。(但し、将来的な Web 化を前提に、画面ごとでも OK)

③ 関数

特定のプログラムを部品化し、共通関数などとして作成することができる。

④ 構造体

関数の引数や、画面間の引数用に利用できる。

WinAPI 関数 (DLL) を利用する際の構造体定義もここで行う。

①⑥⑦のオブジェクト内で構造体を定義できるが、将来廃止予定のため、オブジェクト内に作成することは推奨しない。

⑤ データウインドウ

Appeon PowerBuilder 2017 の特許オブジェクトであり、DB アクセスを中心に、さまざまな表現、アクション操作が実現できる。

※「2. データウインドウ」で詳細を説明

⑥ ユーザオブジェクト

- ビジュアル： 画面上の共通コントロールをデザインできる。
(コントロールの集合体としてデザインが可能)
- ノンビジュアル： トランザクションや、共通関数群として作成することができる。

⑦ ウィンドウ

- ユーザインタフェースを提供する主要なオブジェクトである。
データの表示、ユーザのマウス操作、キーボード操作など、様々なイベント駆動に対応した動作を実現できる。

ウィンドウの種類

- ・メインウィンドウ
- ・チャイルドウィンドウ
- ・ポップアップウィンドウ
- ・レスポンスウィンドウ

ウィンドウに配置できるコントロール

- ・ **コマンドボタン**
- ・ **ピクチャーボタン**
- ・ **チェックボックス**
- ・ **ラジオボタン**
- ・ **スタティックテキスト**
- ・ ピクチャー
- ・ 直線、楕円、長方形、丸長方形
- ・ シングルラインエディット
- ・ エディットマスク
- ・ マルチラインエディット
- ・ リッチテキストエディット
- ・ ドロップダウンリストボックス
- ・ ドロップダウンピクチャーリストボックス
- ・ **リストボックス**
- ・ ピクチャーリストボックス
- ・ リストビュー
- ・ ツリーリストビュー
- ・ **タブコントロール**
- ・ **データウィンドウ**
- ・ グラフ
- ・ など・・・

ウィンドウにユーザインタフェース機能を搭載させるのに、上記の様々なコントロールが配置できる。

そのコントロールの一つであるデータウィンドウには、ウィンドウ上に配置できるコントロールと同等で、しかも DB と密接に連携したコントロール（正式には `dwo`）が提供できる。

DB 連携ができていないことと、ウィンドウ上のコントロール数を削減できることから、ユーザインタフェースの作成には、データウィンドウを中心に設計することを推奨する。

1. Appeon PowerBuilder 2017 ソースの構成

コントロールの種類と数が減ることで、スクリプトを記載する場所が限定でき、かつ、不具合発生率の低減、メンテナンス性の向上が期待できる。

上記 **dwo** のコントロールがあれば、ユーザインタフェースはほぼ網羅でき、チェックボックス、ラジオボタン、ドロップダウンなどは、データウインドウでリッチなユーザインタフェースとして提供することができる。

これまでの開発実績から、必要に応じて **dwo** があれば十分なユーザインタフェースを提供することが可能である。

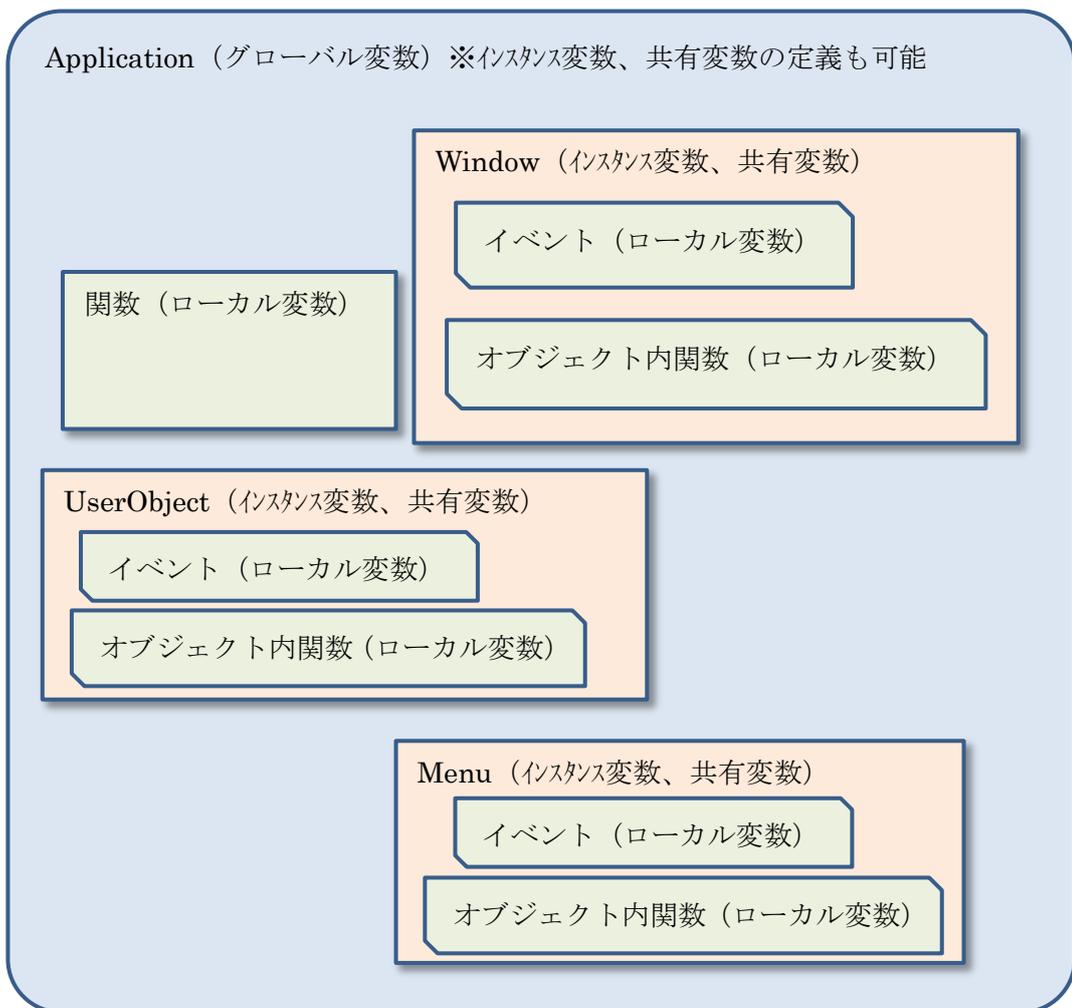
⑧ メニュー

MDI アプリケーション用のため、本書では取り上げない。

(5) 変数の宣言

変数にはスコープの範囲がある。

- ① グローバル変数
アプリケーション内のどのスクリプトからでも利用できる。
- ② 共有変数
オブジェクト内で利用できる変数で、インスタンスを閉じて再度開いたときに値が保持されている。
- ③ インスタンス変数
オブジェクト内のスクリプト、その子孫のスクリプトで使用できる。
- ④ ローカル変数
ローカルの関数、イベント内でのみ利用できる。



(6) 外部関数の宣言

WindowsAPI など、外部 DLL の関数を利用することができる。

スコープの範囲を持っており、グローバルまたはローカル（オブジェクト内）で利用可能になる。

(7) 命名規約

アプリケーション開発時では、命名規約を定め、それに従い実施することを推奨する。先頭文字の代表例を以下に掲載する。

- ① オブジェクト名
- | | |
|-------------------|------|
| A) 関数オブジェクト | f_ |
| B) 構造体オブジェクト | str_ |
| C) データウインドウオブジェクト | d_ |
| D) ユーザオブジェクト | uo_ |
| E) ウインドウオブジェクト | w_ |
- 他割愛
- ② 変数名
- | | |
|-----------------------|-------|
| A) グローバル変数 (String) | gs_ |
| グローバル変数 (Integer) | gi_ |
| グローバル変数 (Long) | gl_ |
| グローバル変数 (Decimal) | gdc_ |
| グローバル変数 (DateTime) | gdt_ |
| グローバル変数 (Date) | gd_ |
| グローバル変数 (Time) | gt_ |
| グローバル変数 (Boolean) | gb_ |
| グローバル変数 (Transaction) | gt_ |
| グローバル変数 (構造体) | gstr_ |
| グローバル変数 (Datawindow) | gdw_ |
| グローバル変数 (DataStore) | gs_ |
| グローバル変数 (UserObject) | guo_ |
| グローバル変数 (Window) | gw_ |
- 他割愛
- | | |
|----------------------|------|
| B) インスタンス変数 (String) | is_ |
| インスタンス変数 (Integer) | ii_ |
| インスタンス変数 (Long) | il_ |
| インスタンス変数 (Decimal) | idc_ |
| インスタンス変数 (DateTime) | idt_ |
| インスタンス変数 (Date) | id_ |
| インスタンス変数 (Time) | it_ |
- 他割愛
- | | |
|------------------|-----|
| C) 共有変数 (String) | ss_ |
| 共有変数 (Integer) | si_ |
| 共有変数 (Long) | sl_ |

1. Appeon PowerBuilder 2017 ソースの構成

	共有変数 (Decimal)	sdc_
	共有変数 (DateTime)	sdt_
	共有変数 (Date)	sd_
	共有変数 (Time)	st_
	他割愛	
D)	ローカル変数 (String)	ls_
	ローカル変数 (Integer)	li_
	ローカル変数 (Long)	ll_
	ローカル変数 (Decimal)	ldc_
	ローカル変数 (DateTime)	ldt_
	ローカル変数 (Date)	ld_
	ローカル変数 (Time)	lt_
	他割愛	
D)	引数 (String)	as_
	引数 (Integer)	ai_
	引数 (Long)	al_
	引数 (Decimal)	adc_
	引数 (DateTime)	adt_
	引数 (Date)	ad_
	引数 (Time)	at_
	他割愛	
③	オブジェクト内のイベント名や関数	
A)	アプリケーション (ユーザ定義イベント)	ue_
	アプリケーション (関数)	af_
B)	ユーザオブジェクト (ユーザ定義イベント)	ue_
	ユーザオブジェクト (関数)	uf_
C)	ウインドウ (ユーザ定義イベント)	ue_
	ウインドウ (関数)	wf_

2. データウインドウ

(1) データウインドウとは

SQL SELECT の結果集合カラムを用いて、フリーフォームや一覧形式のユーザインタフェースを提供することができる。

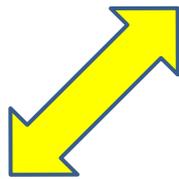
更新特性を持たせることにより、データの追加・変更・削除が可能になり、ソースに SQL 文を記載しなくてもエントリー画面が簡単に作成することができる。

また、SQL SELECT の結果集合を帳票イメージにデザインでき、印刷プロパティを持っていることから、柔軟かつ効率よくアプリケーションを作成することができる。

SQL 文を記載する画面「データソース」と、デザインを行う画面「データウインドウペインタ」を自由に行き来しながら開発することができる。



データソース (SQL 文記載画面)



SELECT 文 - orcl.ORA Oracle

```

SELECT
  A.CUSTXXXX,
  B.CUSTSEQX,
  A.PJSEQXXX,
  A.PJXXXXXX,
  A.COSTMMXX,
  A.DELFLGXX,
  A.CONTTYPE,
  '0' AS new_flg
FROM   M_PJXXXXXX A,
       M_CUSTXXXX B
WHERE  A.CUSTXXXX = B.CUSTXXXX
        
```

グラフィック表示でデザインが可能だが、検索レスポンスを考慮したSQL記載を推奨する。

データウインドウペインタ (デザイン画面)

データウインドウペインタ (デザイン画面)

ウインドウに張り付けることでリッチなユーザインタフェースを提供できる。

更新特性

更新特性の指定

更新可能(A) 更新可能なテーブル(T): M_PJXXXXXX

Update/Delete 文の Where 句 キーカラムの更新

キーカラム(K) Delete 文の後 Insert 文を使用(D)
 キーカラムと更新可能カラム(P) Update 文を使用(U)
 キーカラムと修正したカラム(M)

更新可能なカラム(C):

- custxxxx
- custseqx
- pjseqxxx
- pjxxxxxx
- costmmxx
- delflgxx
- conttype

固有キーカラム(Q):

- custseqx
- pjseqxxx
- delflgxx
- conttype

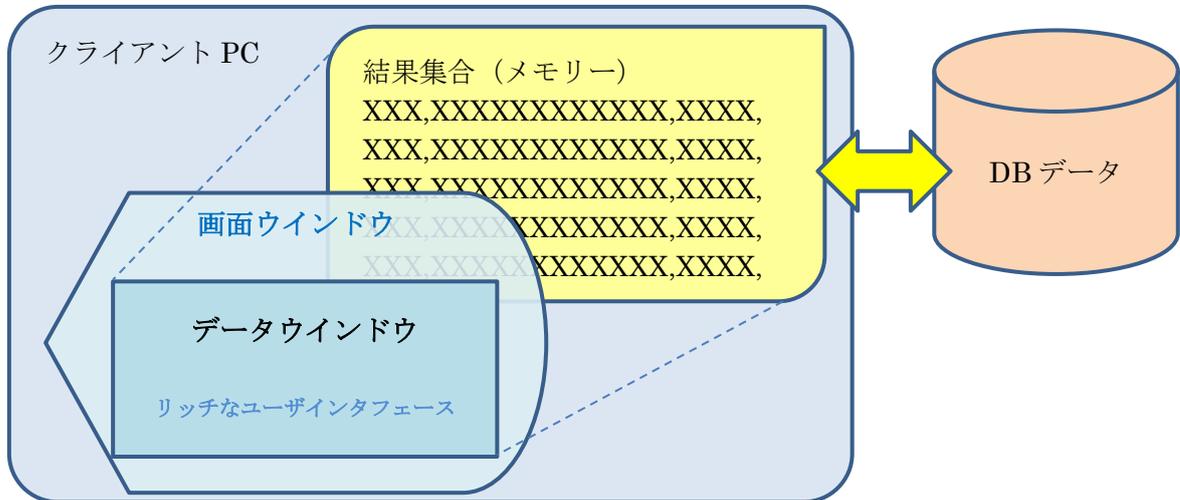
Identity カラム(I): (なし)

OK キャンセル ヘルプ(H) 主キー(P)



(2) データウインドウの特徴

DB から検索した結果集合はクライアント PC のメモリー上で管理され、高いレスポンスビリティでリッチなユーザインタフェースを提供することができる。



データウインドウのメモリー管理にはユーザが操作した情報を保持する領域があり、更新可能なデータウインドウでは、その情報を用いて DB サーバに SQL 文の自動送信を行っている。

① データソース

- Current : データウインドウコントロール内の現在のデータ
- Original : DB から最初に取得されたデータ

検索直後は、Current = Original
入力操作で、Current ≠ Original

② バッファ

- Primary : 行削除またはフィルタで除外されていないデータ
- Delete : 行削除されたデータ
- Filter : フィルタで除外されたデータ

データウインドウには Update 関数があり、上記①②の状態を判断して DB サーバに SQL 文を自動送信する。

データソース Current ≠ Original のとき

値に変更があったと判断し、Update 文が送信される。
但し、更新特性の設定によっては、Delete 文&Insert 文が送信される。

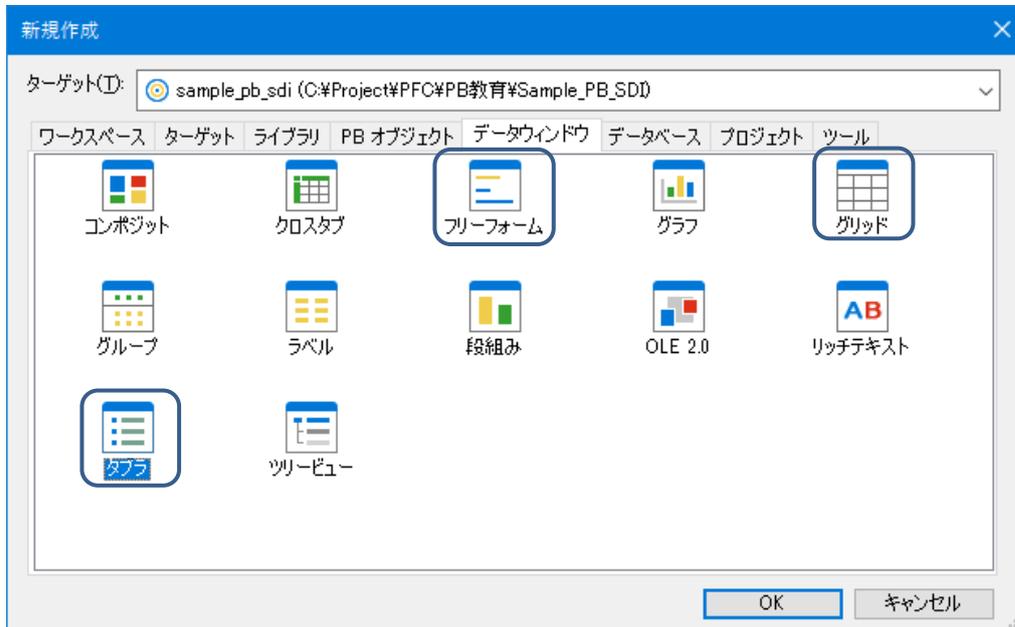
新しい行のとき、Insert 文が送信される。

バッファ Delete に行があるとき

Delete 文が送信される。

(3) データウインドウの種類

データウインドウのデザイン時に下図の選択肢があり、使用頻度が高いものを枠で囲んだ。



- ① フリーフォーム
結果集合 1 行分をページ内に自由に配置するイメージ
- ② グリッド
結果集合の一覧イメージで、列幅の調整が可能（エクセルイメージ）
明細行のデザインは 1 行分のイメージとなる。

行	見出し 1	見出し 2	見出し 3
1	データ 1	データ 2	データ 3
2	データ 1	データ 2	データ 3
3	データ 1	データ 2	データ 3

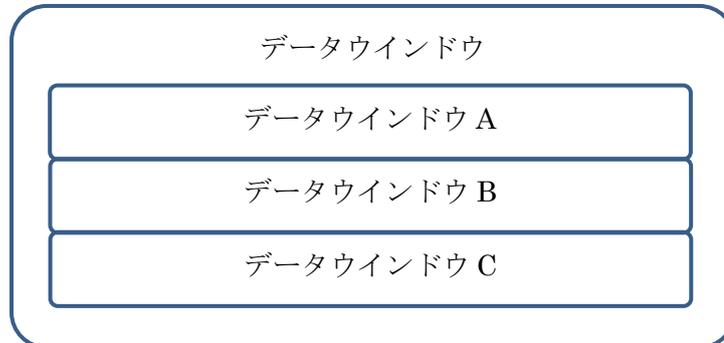
- ③ タブラー
結果集合の一覧イメージで、列幅と位置が固定
ヘッダ見出しを上下に重ねたり、明細行のデザインを複数行のイメージにできる。

行	見出し 1	
	見出し 2	
	見出し 3	
1	データ 1	
		データ 2 データ 3
2	データ 1	
		データ 2 データ 3

2. データウインドウ
(3) データウインドウデザインの種類

④ コンポジット

データウインドウの中にデータウインドウを組み込む。
異なる帳票を左右、上下にまとめて印刷したいときに利用できる。



⑤ クロスタブ

クロスタブ集計用のイメージとなり、ウィザード形式で初期デザインが行われる。

⑥ グラフ

棒グラフ、折れ線、3Dグラフの表現が可能になる。

⑦ グループ

結果集合をデザイン画面でグループ集計して表現できる。
ウィザード形式で初期デザインが行われる。
実際には、タブラーで手動デザインすることが多い。

⑧ ラベル

ウィザード形式でラベル用紙を指定してデザインが行われる。

⑨ 段組み

結果集合を横方向（段）に並べて配置する。

1行目データ	2行目データ	3行目データ
4行目データ	5行目データ	6行目データ
7行目データ		

⑩ ツリービュー

ツリービューイメージの表現ができ、ウィザード形式で初期デザインが行われる。

検索されてメモリーにあるデータをツリービューで表示できることから、.NETのツリービュー表示のレスポンスとは比べ物にならないぐらいに高速である。

(4) データウインドウのオブジェクト (dwo)

データウインドウをデザインすると、SQL の結果集合カラムに相当するカラムとテキストオブジェクトが自動的に初期デザインされる。

タブラーやグリッドの場合、ヘッダには見出し (テキストオブジェクト)、詳細にカラムが配置される。

★

```
SELECT 文 - orcl.ORA Oracle.pfc_pj
SELECT
ACUSTXXXX,
BCUSTSEQX,
APJSEQXXX,
APJXXXXXX,
ACOSTMMXX,
ADEFLGXX,
ACONTTYPE,
' AS new_flg
FROM   M_PJXXXXXX A,
       M_CUSTXXXX B
WHERE  ACUSTXXXX = BCUSTXXXX
```

★

カラム名	データ型	条件指定	初期値	入力条件別	入力条件別はラ - メッセージ
m_pjxxxxxx_custxxxx	char(30)	<input type="checkbox"/>			
m_custxxxx_custseqx	decimal(0)	<input type="checkbox"/>			
m_pjxxxxxx_pjseqxxx	decimal(0)	<input type="checkbox"/>			
m_pjxxxxxx_pjxxxxxx	char(30)	<input type="checkbox"/>			
m_pjxxxxxx_costmxxx	decimal(2)	<input type="checkbox"/>			

カラム、テキストオブジェクトの他に、計算フィールド、図形 (直線、楕円、長方形、丸長方形)、グラフ、OLE、レポート (データウインドウの入れ子) などがあ、総称して dwo と呼ぶ。

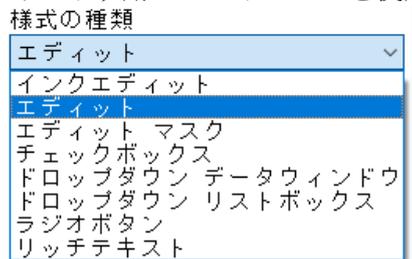
計算フィールドではカラムや計算式を用いてその結果を表示することができる。

タイトルにページ数や日付、フッタに金額の合計や、データの件数を表示できることから、レポート処理で有効利用できる。

2. データウインドウ (4) データウインドウのオブジェクト (dwo)

カラムは、プロパティにある「様式の種類」を選択することで、様々な表現が可能になる。

ウインドウ用のコントロールを使用せずに、リッチなデザインが可能である。



例) 自社プロジェクト管理ツール for PowerBuilder

で囲んでいる箇所はデータウインドウ
データウインドウだからこそ、DB データをベースとした自由な表現ができる。

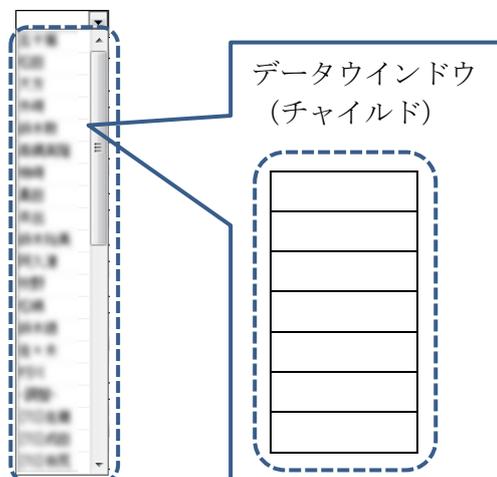


(5) データウインドウのカラムの表現方法

データウインドウカラムのプロパティを設定することで、リッチなユーザインタフェースを提供できる。

主要なものを紹介する。

- ① エディット (デフォルト)
作成時のデフォルトであり、最も多く使われる様式である。
書式を指定し、1,000,000 や、2015/12/31 23:59:59 などの表示が可能になり、ユーザにとって見やすい表現方法が豊かである。
- ② エディットマスク
入力編集がし易いフォーマットを有している。
※カラムの Null 判定ができないため注意が必要
- ③ チェックボックス
フラグの ON/OFF などのチェック欄に用いる。
- ④ ラジオボタン
カラム1つに、複数の選択肢をラジオボタンで設けることができる。
(選択肢は手入力で設定する)
- ⑤ ドロップダウンリスト
カラム1つに、複数の選択肢をドロップダウンリストで設けることができる。(選択肢は手入力で設定する)
- ⑥ ドロップダウンデータウインドウ
カラム1つに、複数の選択肢をドロップダウンリストで設けることができる。(選択肢はチャイルドデータウインドウ)
チャイルドデータウインドウには、データウインドウオブジェクトを指定する。
DB データを用いてドロップダウンのリストを生成できる。



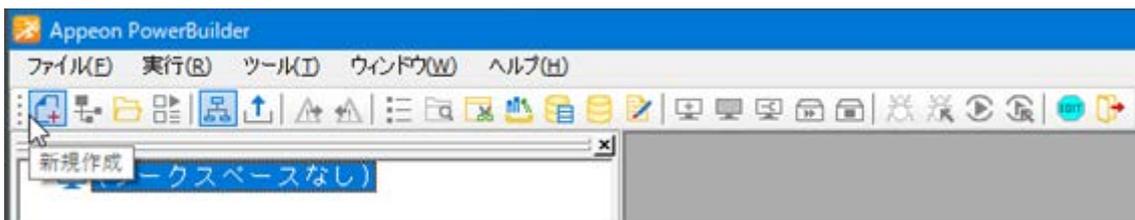
3. Apeon PowerBuilder 2017 開発手順 (1) ワークスペースの作成

3. Apeon PowerBuilder 2017 開発手順

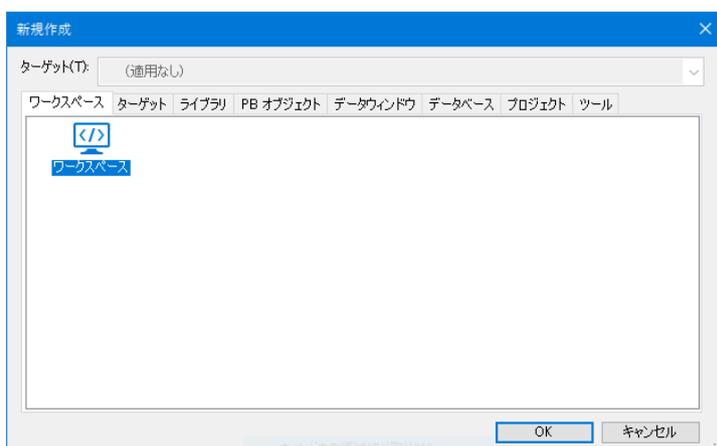
(1) ワークスペースの作成

最初に、ソースを管理するための領域を作成する。

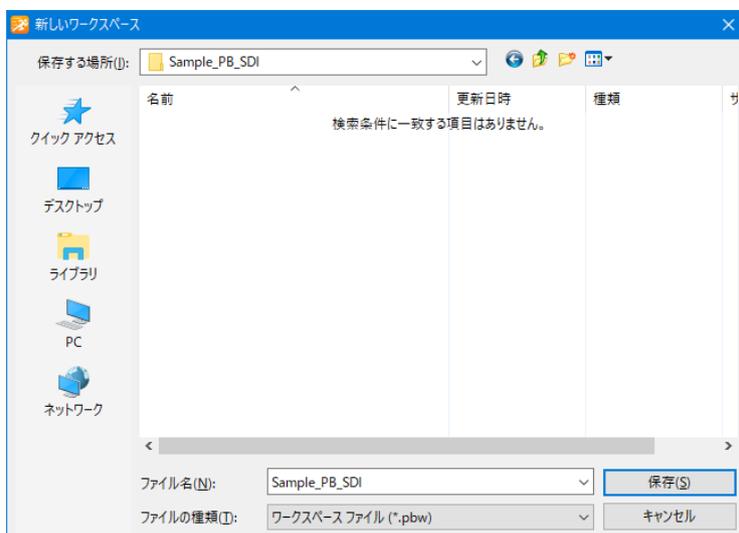
「新規作成」を押下する。



新規作成ダイアログから、ワークスペースを選択して「OK」を押下する。



ワークスペースを保存するフォルダを指定し、ファイル名（ワークスペース名）を入力して「保存」を押下する。



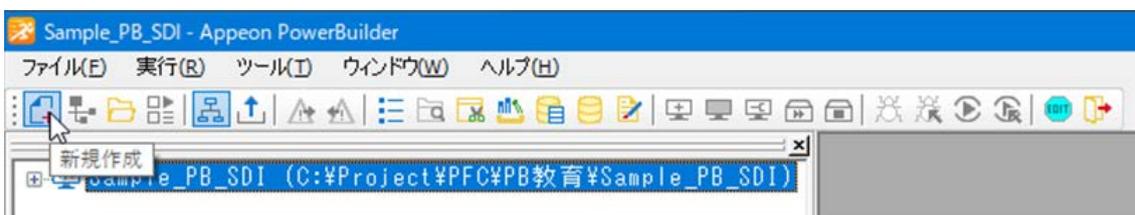
(2) ターゲットの作成

ターゲットはアプリケーションを指し、ワークスペース内に複数登録することができる。

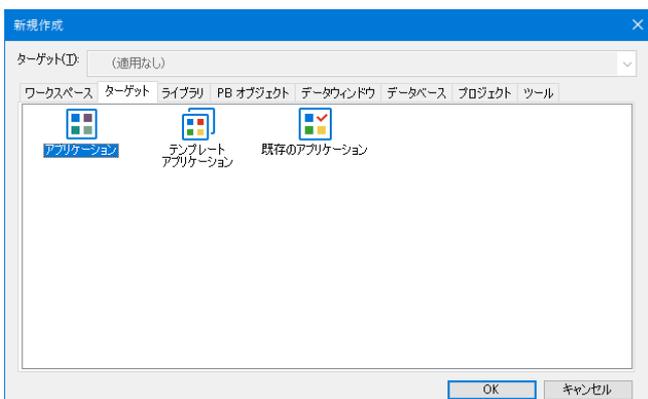
但し、複数ターゲットはソースの管理が煩雑になる懸念があるため、1つのターゲットを推奨する。

ターゲットには、アプリケーションが1つ作成され、PBL ライブラリリストを定義する。(初期は自身のPBLのみとなり、開発を進めながら追加する)

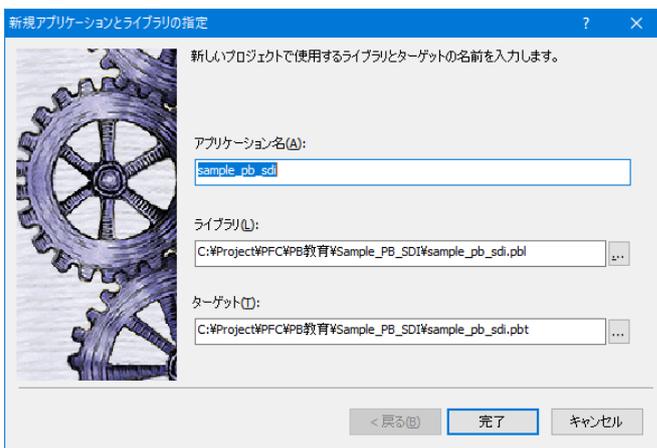
「新規作成」を押下する。



新規作成ダイアログから、ターゲット、アプリケーションを選択して「OK」を押下する。



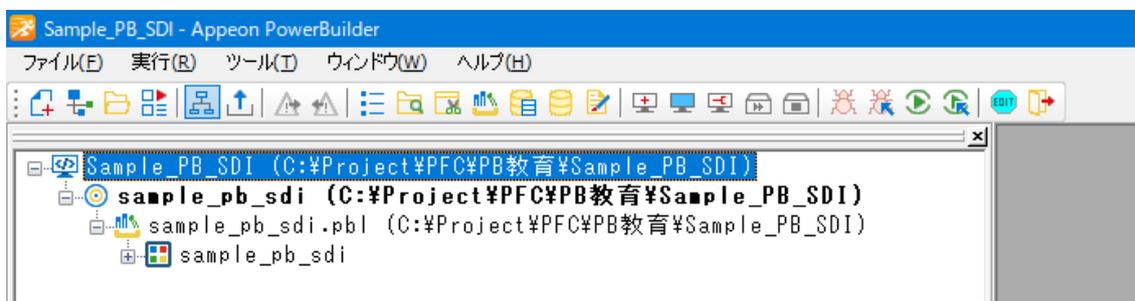
アプリケーション名を入力し、「Tab」押下により、ライブラリ、ターゲットが自動的に展開表示されるので、そのまま「完了」を押下する。



3. Appeon PowerBuilder 2017 開発手順 (2) ターゲットの作成

ターゲットの追加により、以下の構成が作成される。

ワークスペース	Sample_PB_SDI
ターゲット	sample_pb_sdi
PBL ファイル	sample_pb_sdi.pbl
アプリケーションオブジェクト	sample_pb_sdi

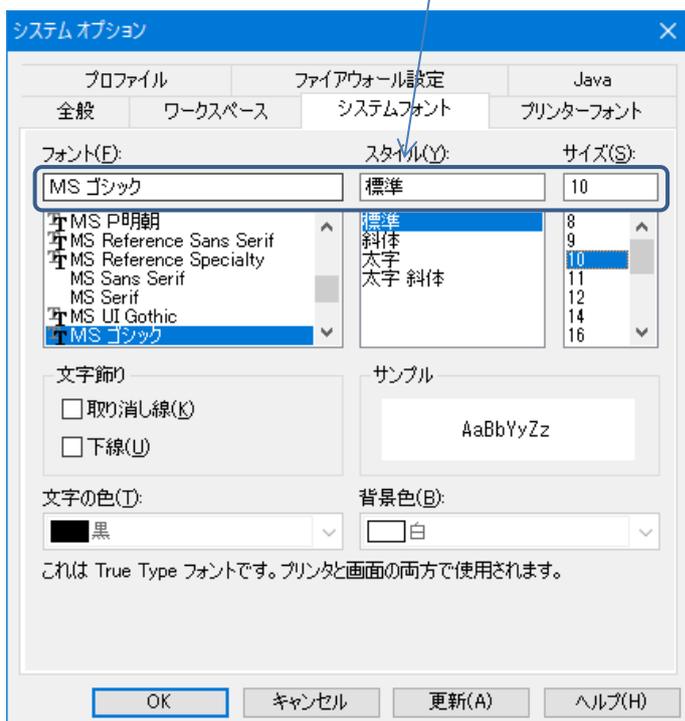
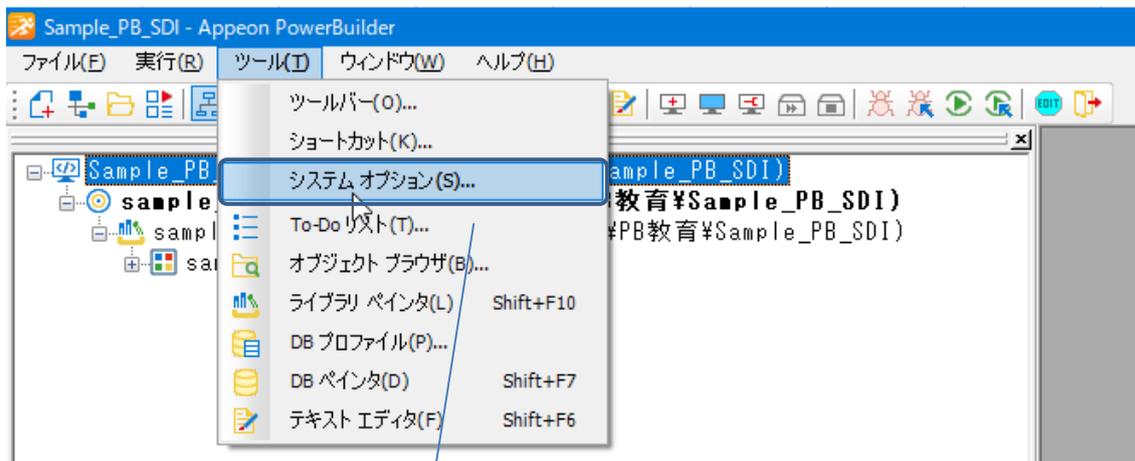


3. Appeon PowerBuilder 2017 開発手順 (3) システムオプションのシステムフォント

(3) システムオプションのシステムフォント

インストール後のデフォルトの状態では、スクリプトの文字が小さく、文字幅が異なるため開発し辛いため、ゴシックにして文字を大きくすることを推奨する。

※設定後、Appeon PowerBuilder 2017 を再起動する。

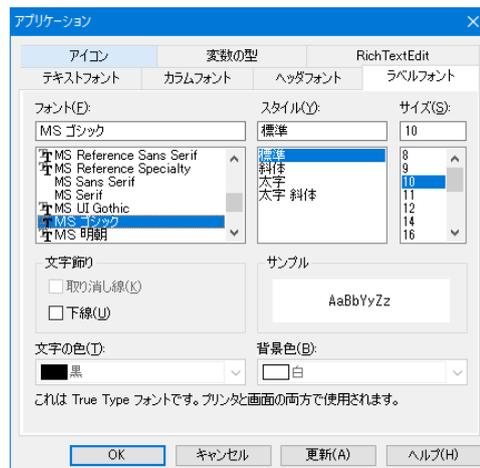
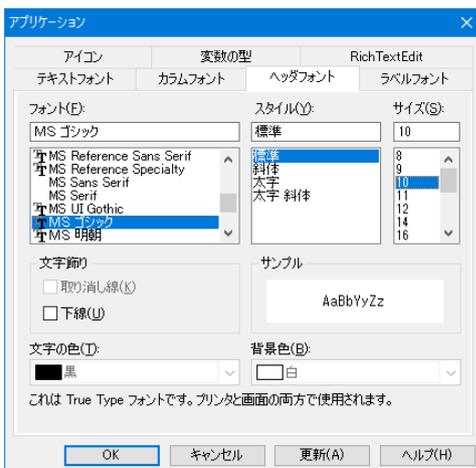
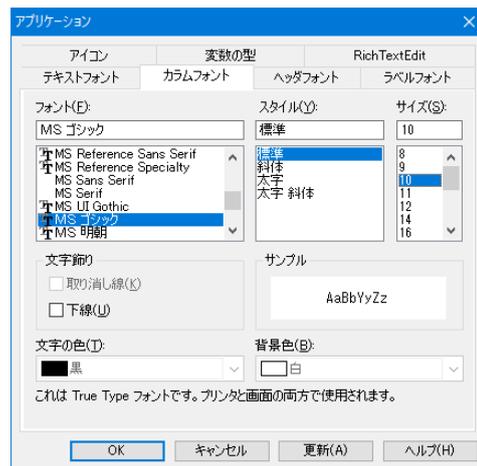
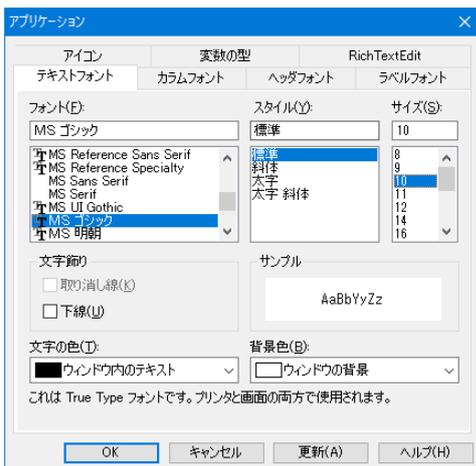
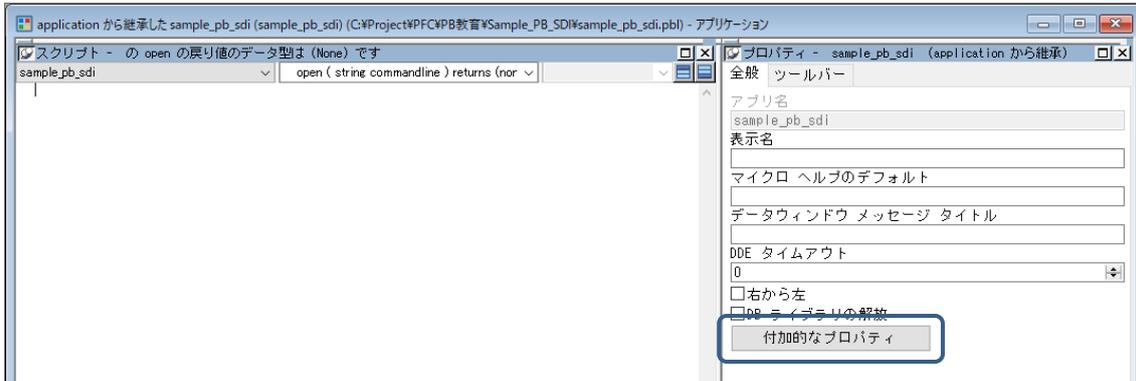


3. Appeon PowerBuilder 2017 開発手順 (4) アプリケーションの付加的なプロパティの設定

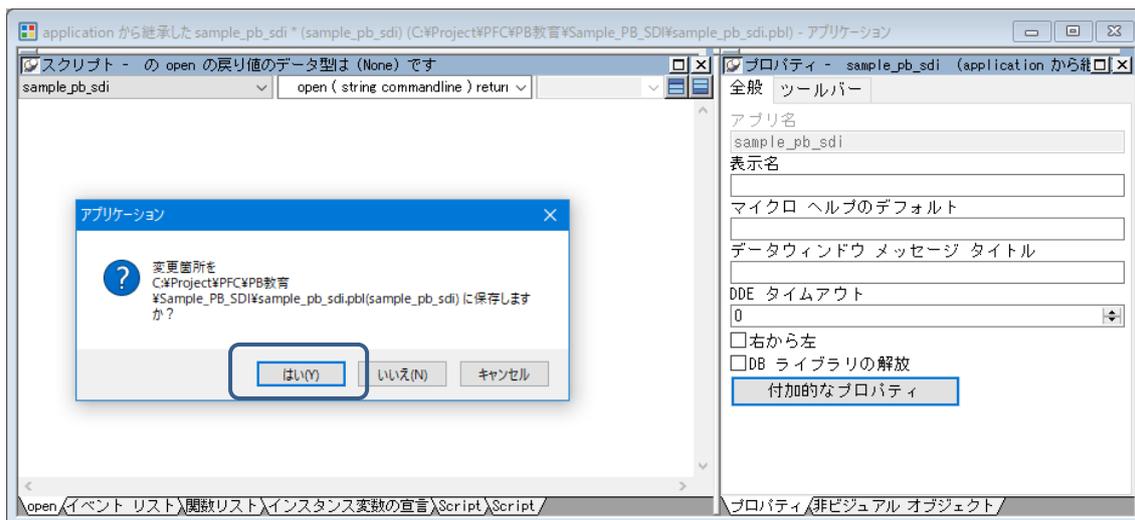
(4) アプリケーションの付加的なプロパティの設定

アプリケーションが持つ「付加的なプロパティ」を設定する。
デザイン時の文字の大きさをここで統一する。

※設定後、変更箇所を保存して Appeon PowerBuilder 2017 を再起動する。



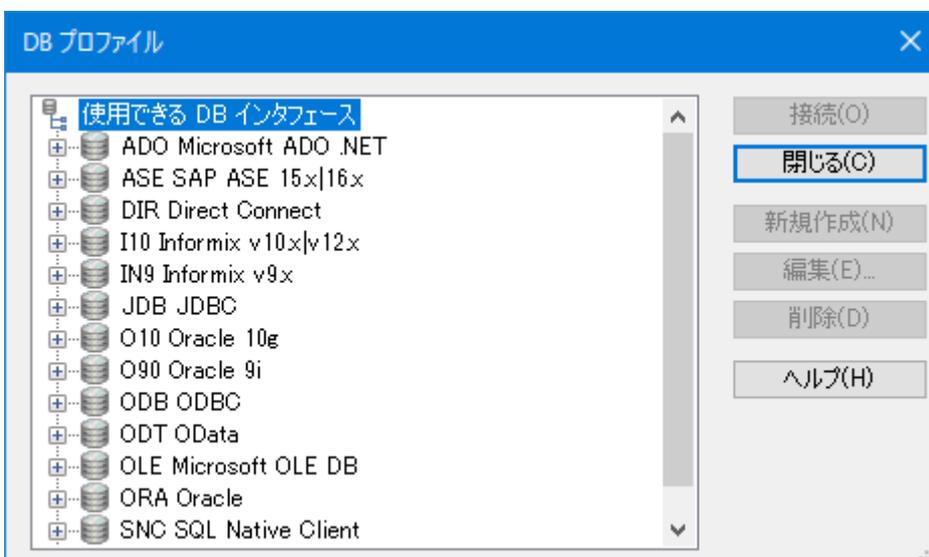
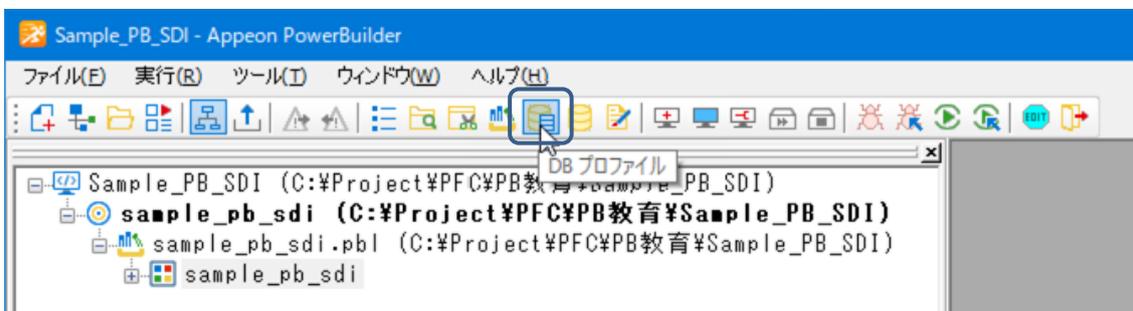
3. Apeon PowerBuilder 2017 開発手順 (4) アプリケーションの付加的なプロパティの設定



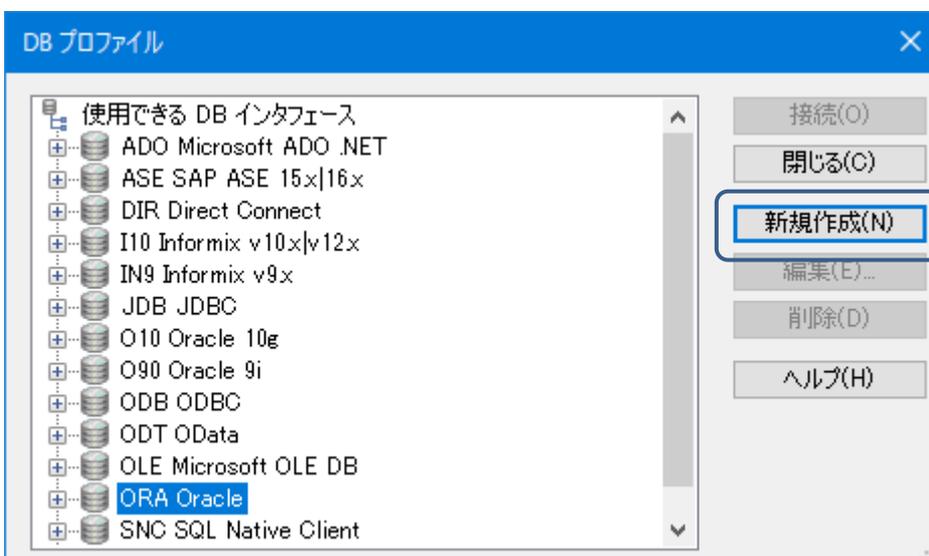
(5) DB 接続

コーディングの前に、IDE を DB に接続する。

「DB プロファイル」を押下する。



Oracle11g への接続として設定する。



3. Appeon PowerBuilder 2017 開発手順 (5) DB 接続

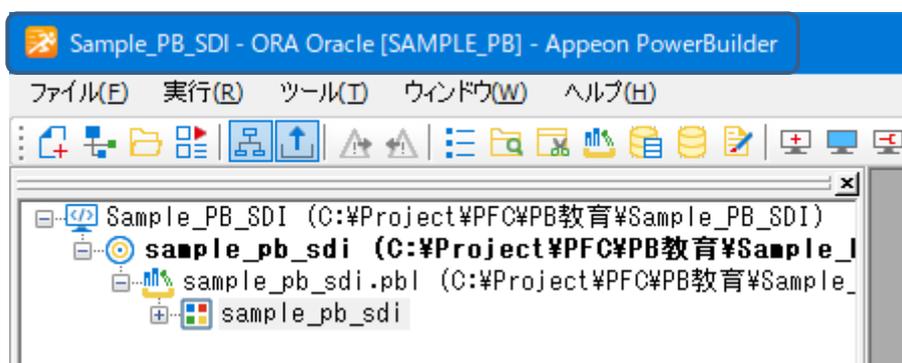
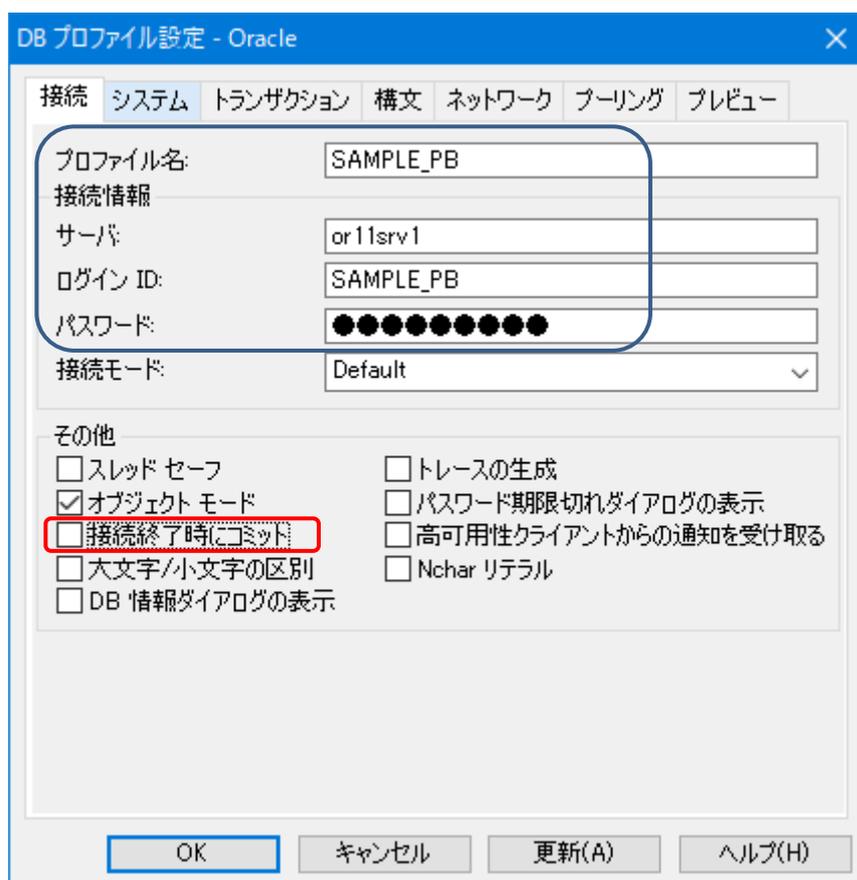
「接続」の設定を行う。

プロファイル名： DB プロファイルの一覧に表示される。

接続情報： DB ベンダーごとに設定する。

その他： 接続終了時にコミット

ON のままでアプリケーションが異常終了した場合、その時点まで実行された SQL 文が COMMIT されるため、外すことを推奨する。

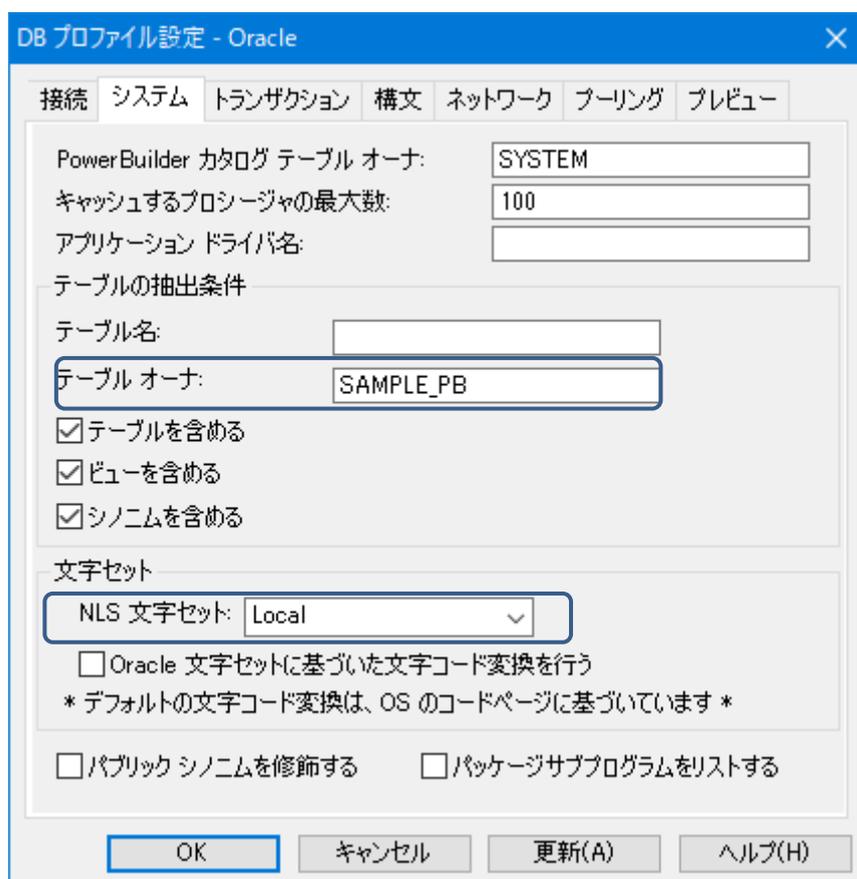


3. Apeon PowerBuilder 2017 開発手順 (5) DB 接続

「システム」の設定を行う。

テーブル抽出条件： テーブルオーナーを指定する。
テーブルオーナーを指定しておくことで、DB ペインターでテーブル操作する際に、他のオーナーのテーブルが表示されないため、目的のテーブルが探しやすくなる。

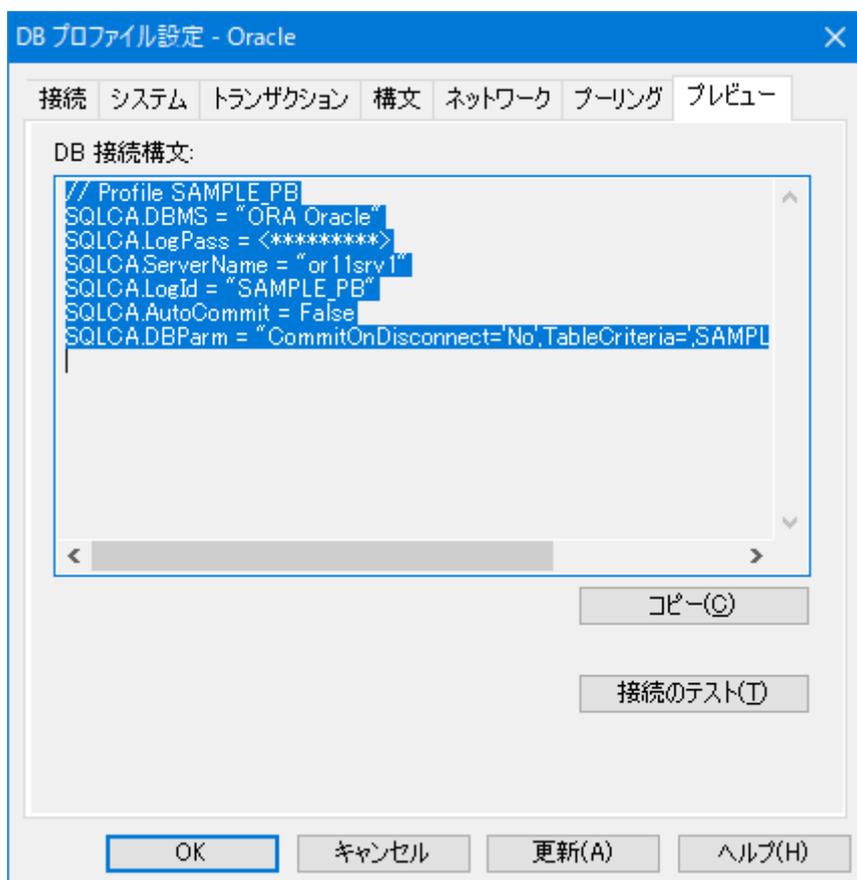
文字セット： NLS 文字を指定する。
DB の文字コードが SJIS のとき、Local を指定する。
但し、Build 5006J は不具合があるため、開発時は Unicode を指定し、実行時は Local を指定すること。



3. Appeon PowerBuilder 2017 開発手順
(5) DB 接続

「プレビュー」について

DB 接続構文をプログラミングに利用することができる。



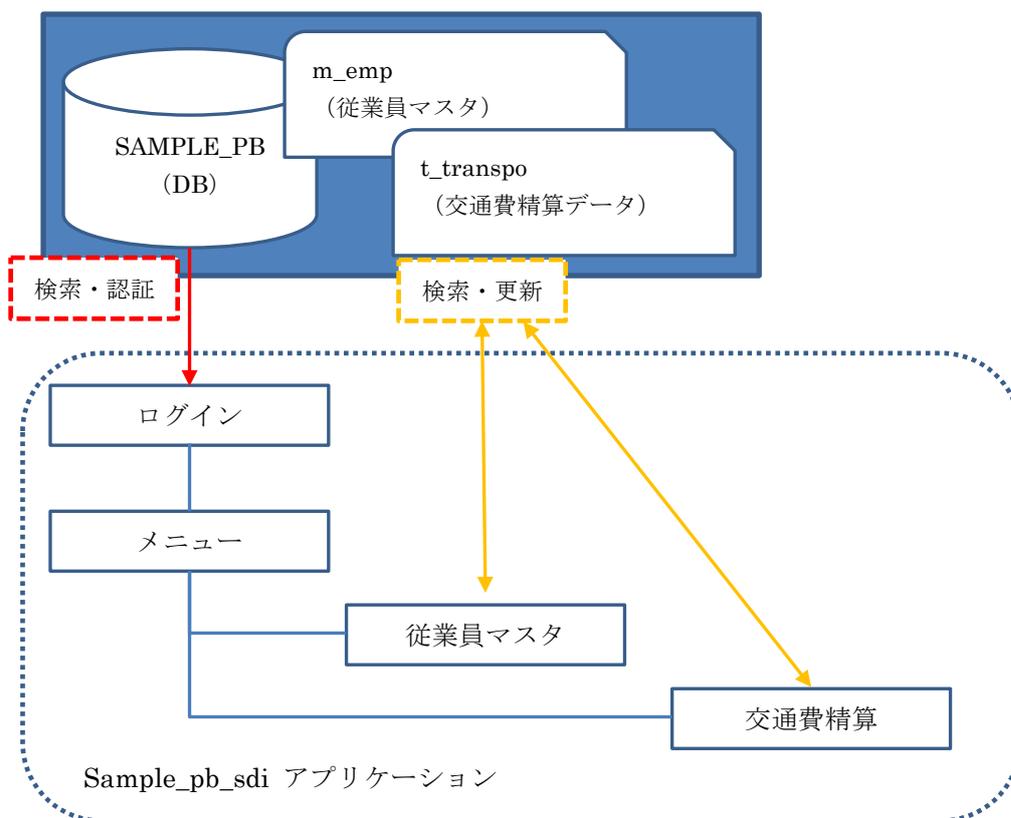
(6) アプリケーションのコーディング

上記手順により作成されたアプリケーションオブジェクトに、基本ソースをコーディングする。

■アプリケーションの概要

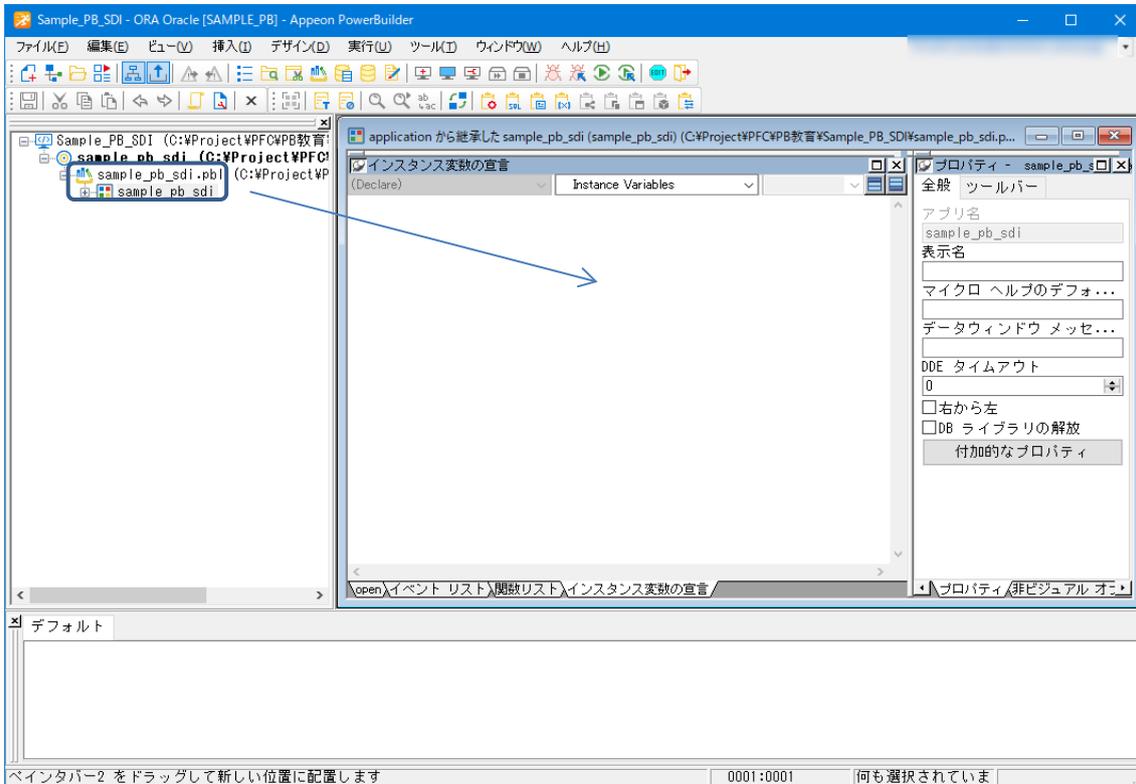
DB (Sample_PB) に用意されたテーブル『従業員マスタ』『交通費精算』からデータの検索や更新 (登録・変更・削除) を行う画面を作成する。

■画面フローイメージ



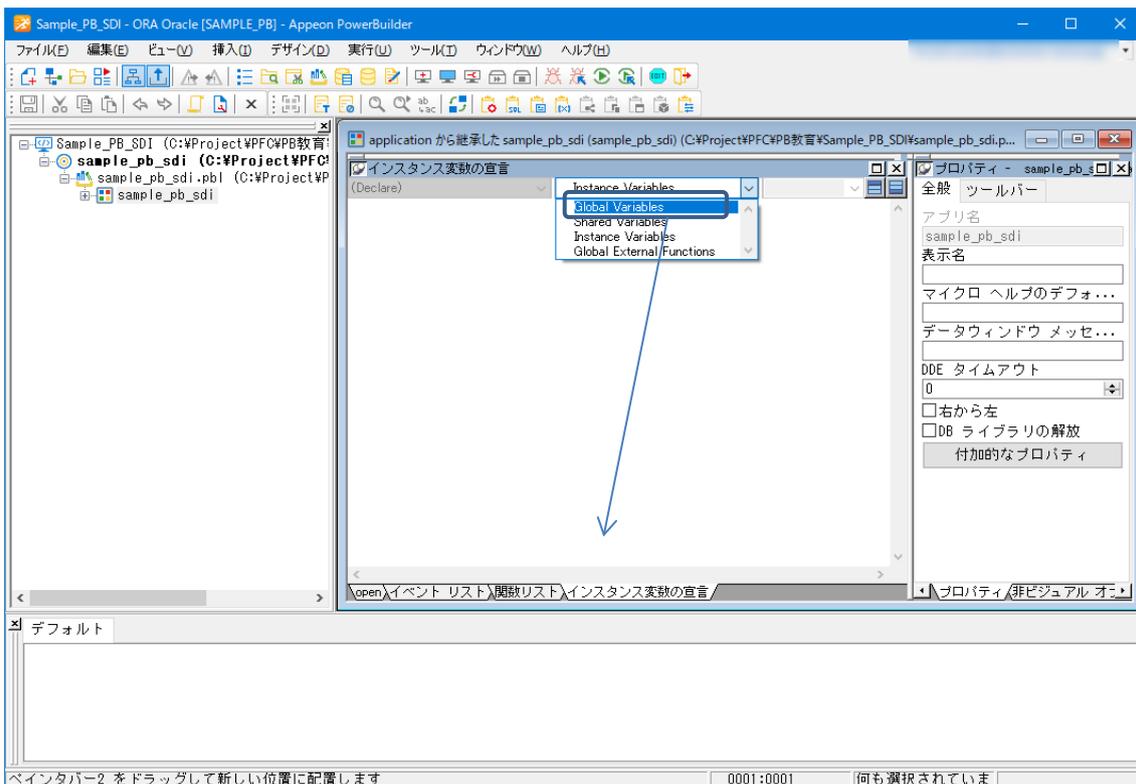
3. Apeon PowerBuilder 2017 開発手順 (6) アプリケーションのコーディング

アプリケーションオブジェクトは、PB アプリケーションの入り口であり、出口である。



① グローバル変数宣言

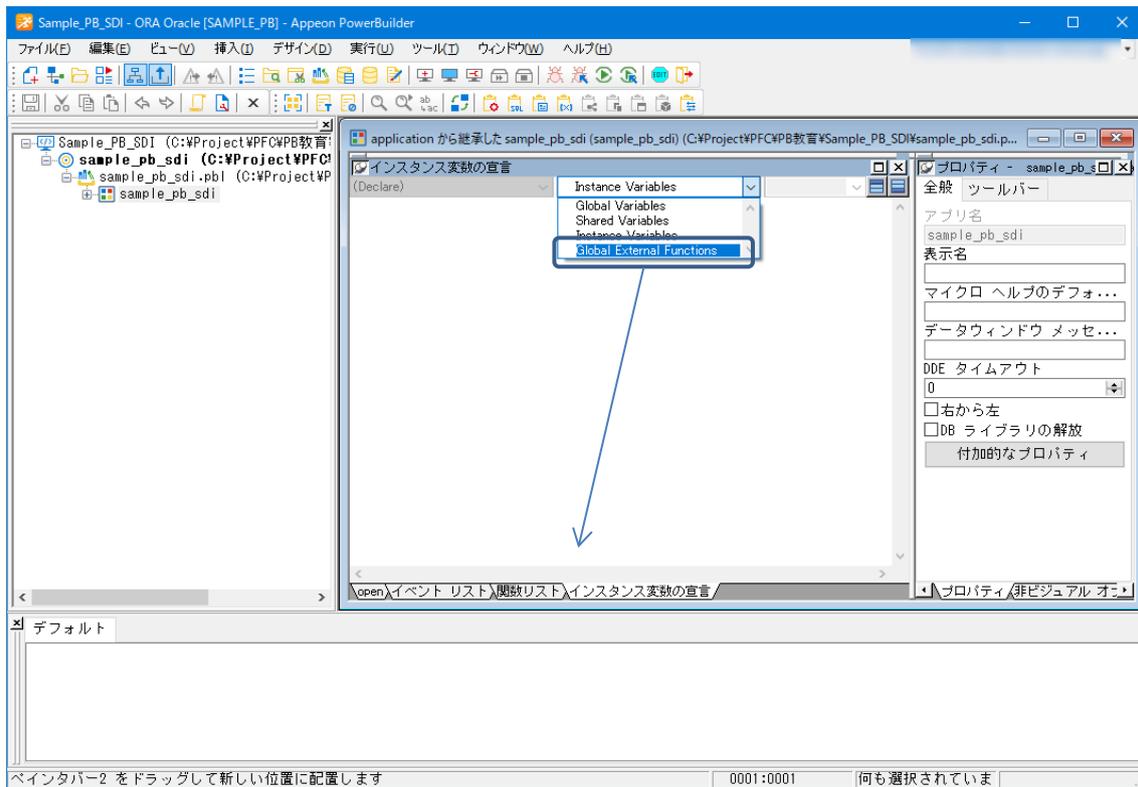
下図操作により、グローバル変数宣言が可能になる。



3. Apeon PowerBuilder 2017 開発手順 (6) アプリケーションのコーディング

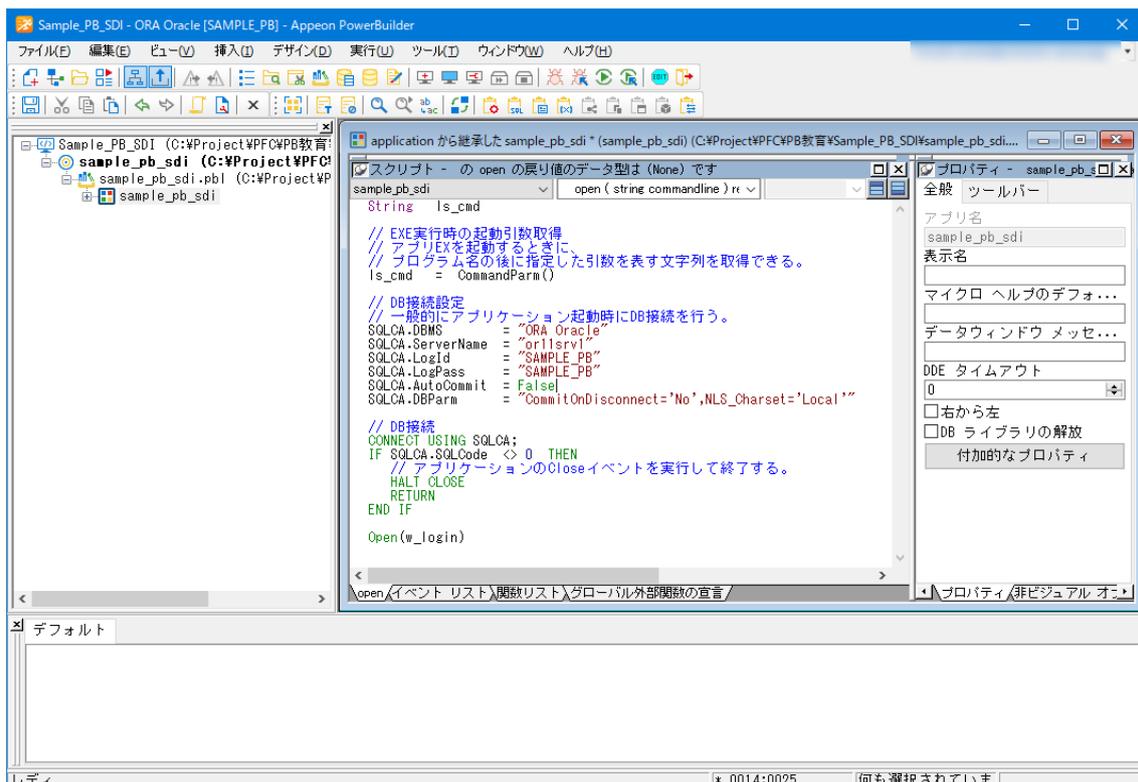
② グローバル外部関数宣言

下図操作により、グローバル外部関数変数宣言が可能になる。



③ Open イベント

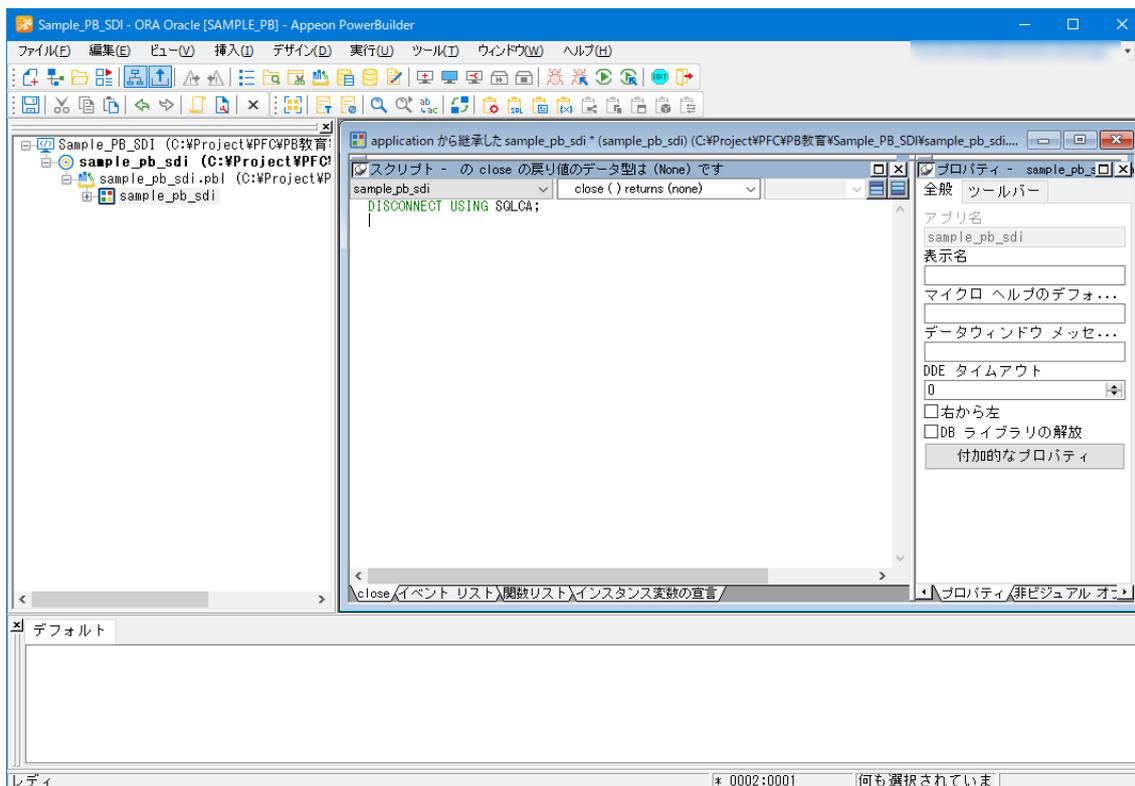
アプリケーション起動時の処理を記載する。



3. Apeon PowerBuilder 2017 開発手順 (6) アプリケーションのコーディング

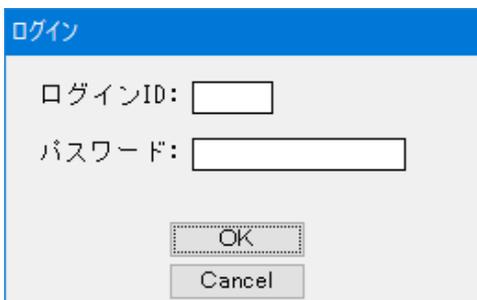
Close イベント

アプリケーション終了時の処理を記載する。



(7) ログイン画面の作成

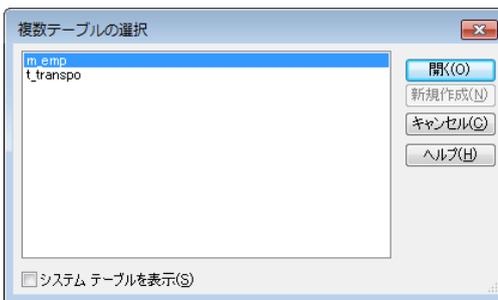
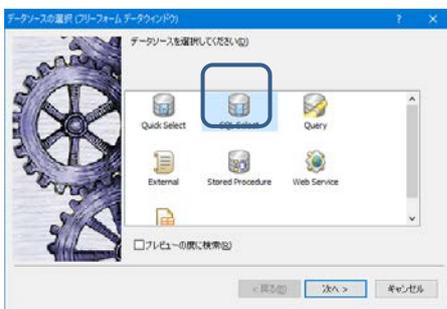
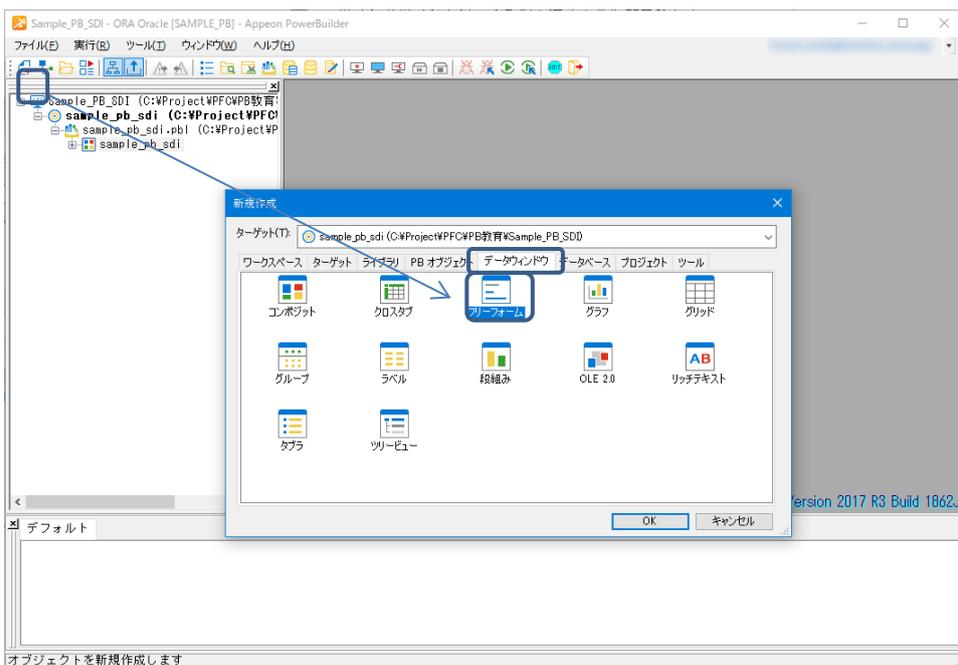
作成するウィンドウのイメージ



① データウィンドウの作成

画面に配置できるコントロールは様々あるが、今回はデータウィンドウで入力フィールドを作成する。

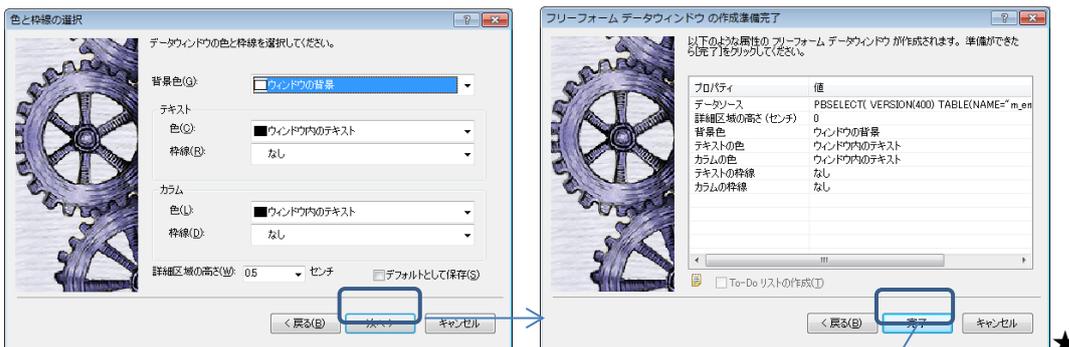
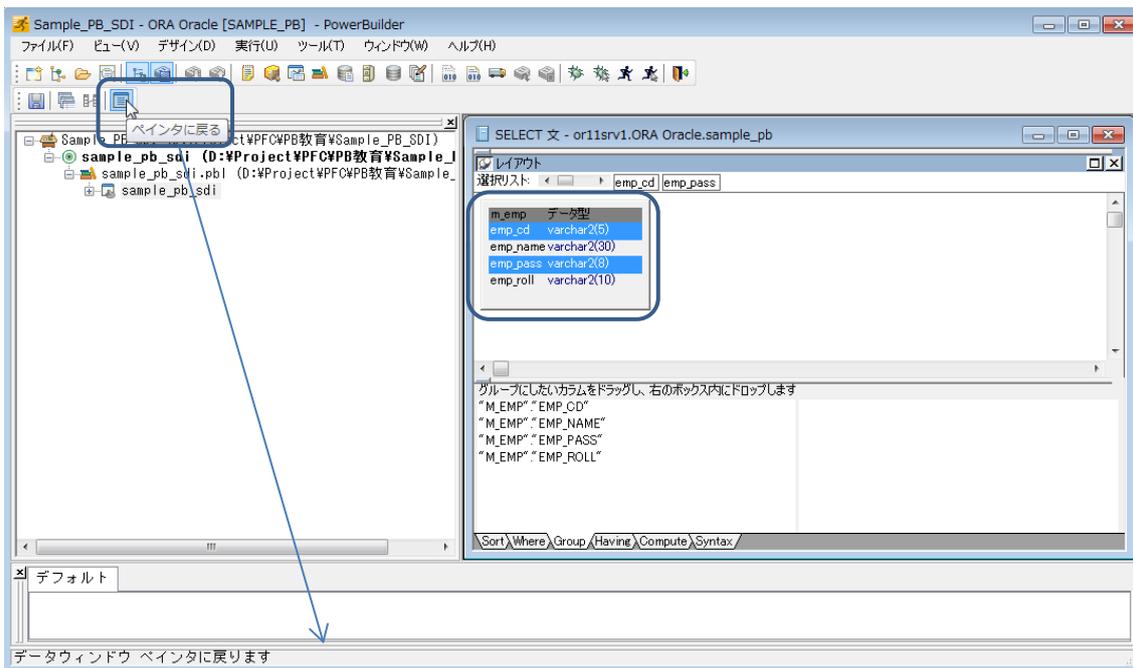
検索・更新がなく入力専用であっても、データウィンドウを用いることでフィールドデザインが容易に行えるメリットがある。



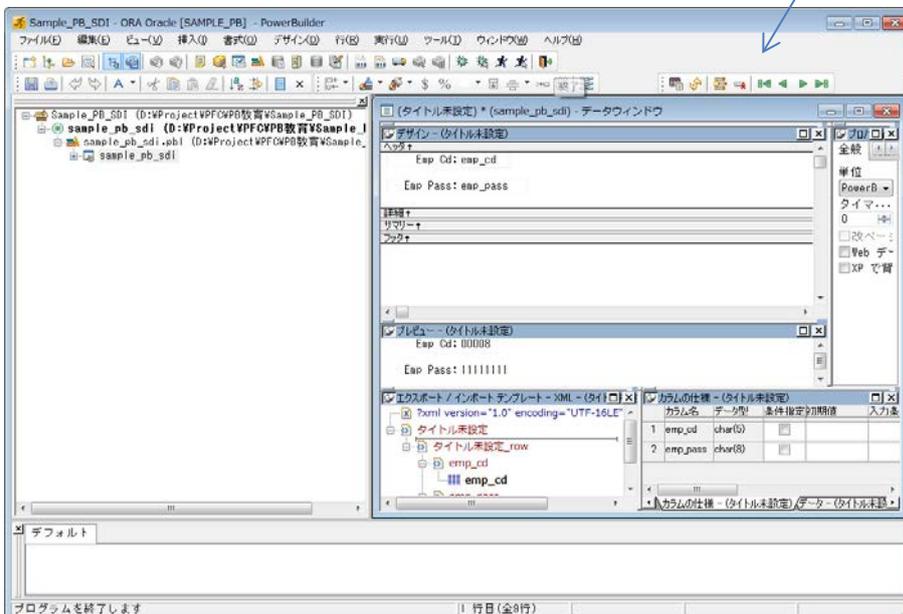
3. Apeon PowerBuilder 2017 開発手順

(7) ログイン画面の作成

テーブルからカラムを指定して、ペインタに戻る。



ウィンドウコントロールを配置するよりも簡単である。
よって、開発スピードとメンテナンス性が向上する。★



3. Apeon PowerBuilder 2017 開発手順 (7) ログイン画面の作成

体裁を整える。★

プロパティ - emp_cd

IME
オフ (2)

自動選択
 自動選択

パスワード
 パスワード

プロパティ - emp_pass

IME
オフ (2)

自動選択
 自動選択

パスワード
 パスワード

【カラムのプロパティ設定】

- ・IME オフ (2)
入力値を半角英数のみ可能とする
- ・パスワード
入力値をアスタリスク (*) で表示する

保存する。

データウィンドウの保存

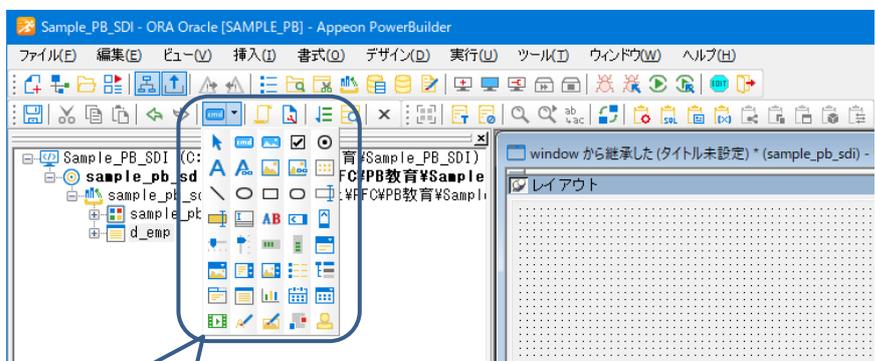
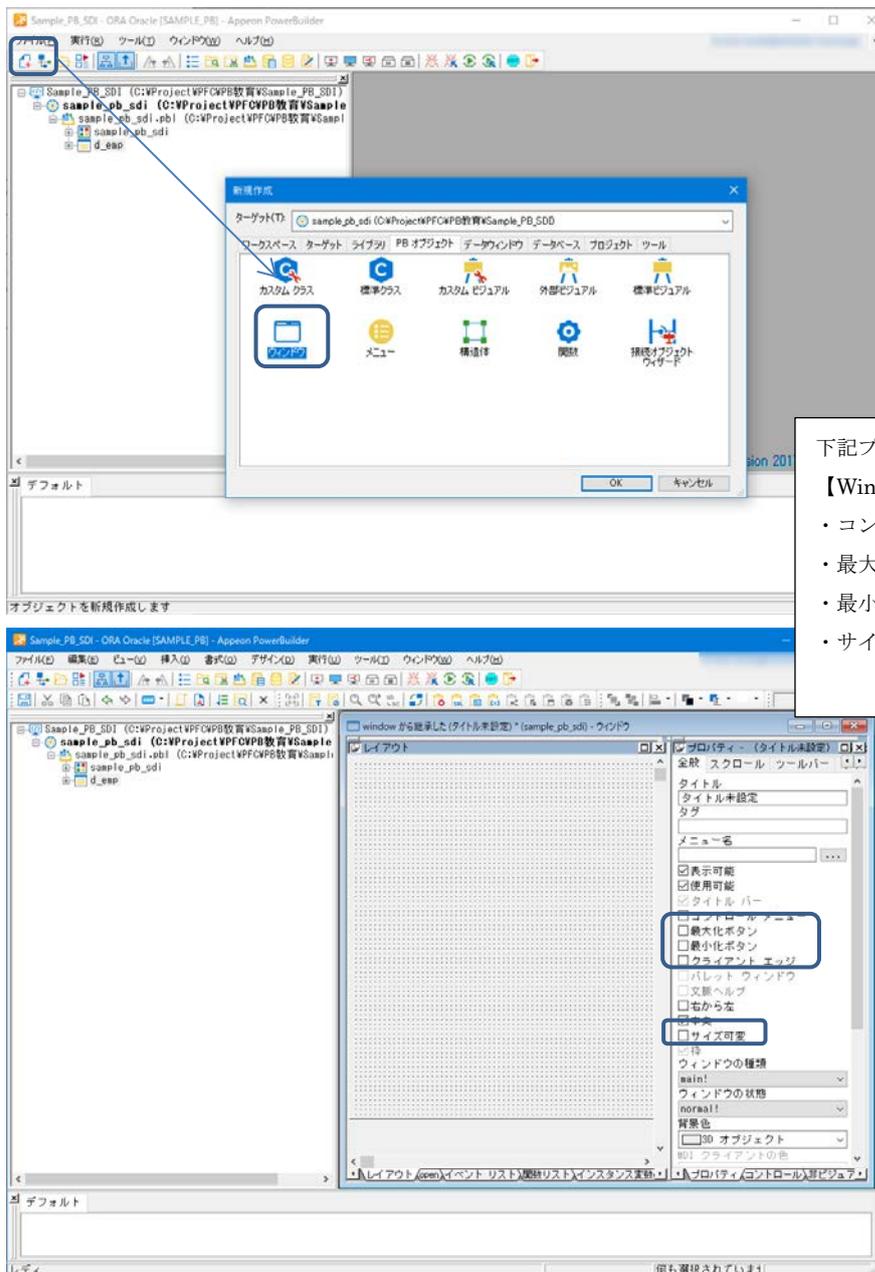
データウィンドウ(D):
d_emp

アプリケーション ライブラリ(A):
C:\Project\PF\CPB教育\Sample_PB_SDI\sample_pb_sdi.pbl

3. Apeon PowerBuilder 2017 開発手順 (7) ログイン画面の作成

② ウィンドウの作成

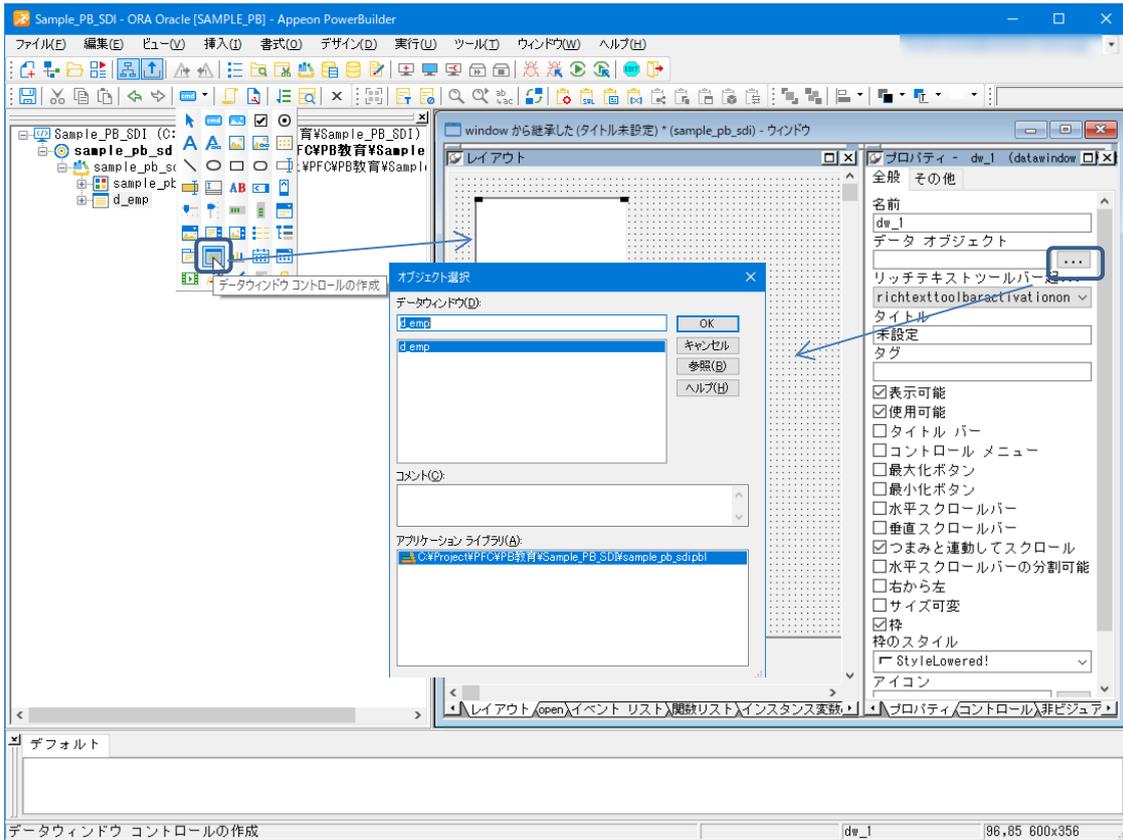
ウィンドウを新規で開き、コントロールを配置する。



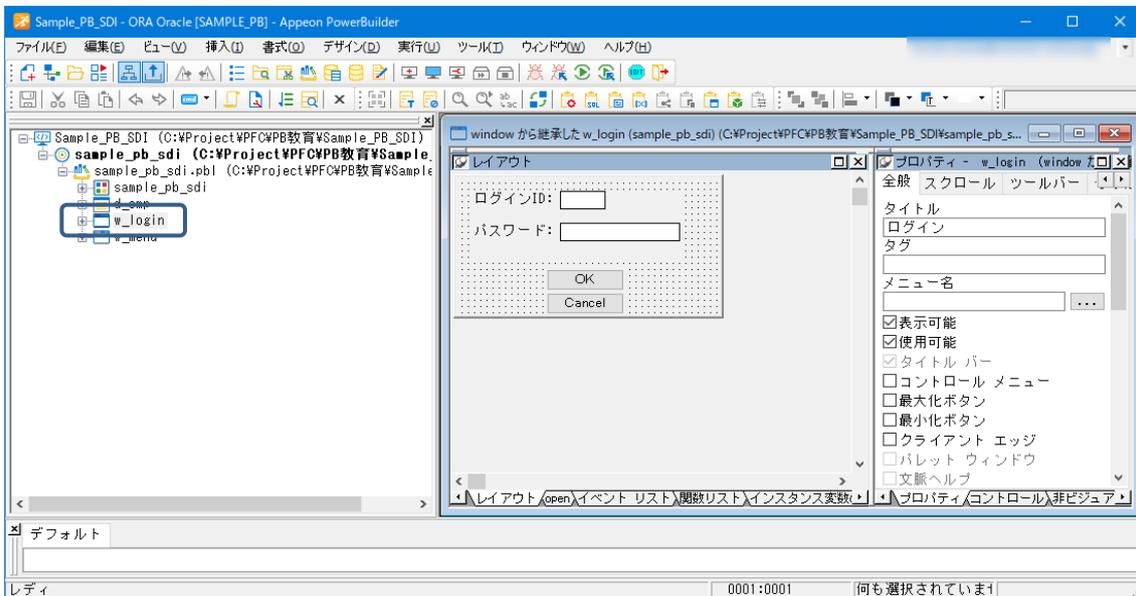
配置できるコントロール

3. Apeon PowerBuilder 2017 開発手順 (7) ログイン画面の作成

データウィンドウを配置する。



コマンドボタンも貼り付け、体裁を整え、保存する。(w_login)



- ③ スクリプトの記載箇所
- w_login : ウィンドウのイベント
 - dw_1 : データウィンドウのイベント
 - cb_1 : OK ボタンのイベント
 - cb_2 : Cancel ボタンのイベント

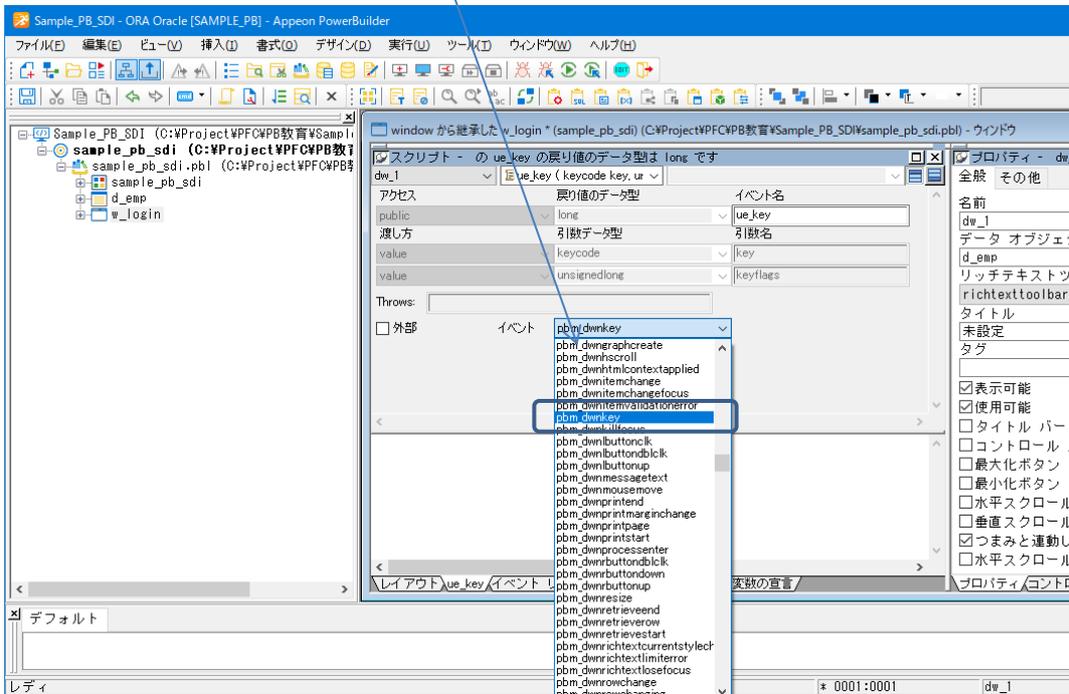
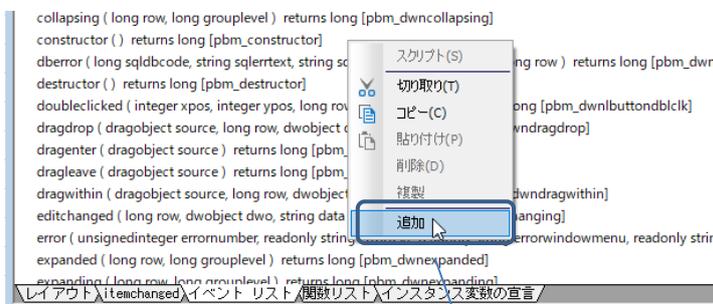
④ w_login
 A) open イベント

```

window から継承した w_login (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_s
スクリプト - の open の戻り値のデータ型は long です
w_login
open ( ) returns long [pbm_open]
// 入力データウインドウの表示設定
// 検索・更新用ではない
// 入力可能にするため、1行空欄を用意する
dw_1.InsertRow(0)
dw_1.SetFocus()
    
```

⑤ dw_1
 A) ユーザイベントの作成 ue_key

Enter キー押下でカラム移動する仕組みを入れる。
 ウィンドウ標準では、Enter キー押下で次行移動となり操作性
 が悪いので、Enter キー押下で Tab キーを発行する。



```

window から継承した w_login (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI
スクリプト - の ue_key の戻り値のデータ型は long です
dw_1
IF key = KeyEnter! THEN
//改行制御
Send(Handle(This), 258, 9, Long(0,0) )
RETURN 1
END IF
    
```

Enter キーが無効になる。

⑥ cb_1 : OK

A) clicked イベント

ログイン、パスワードのチェックを行う。

```

window から継承した w_login (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_sdi.pbl) - ウィンドウ
スクリプト - の clicked の戻り値のデータ型は long です
cb_1
String ls_emp_cd
String ls_emp_pass
String ls_emp_pass_db
Long ll_count

dw_1.AcceptText()

// 従業員コード
ls_emp_cd = dw_1.Object.emp_cd[1]
IF IsNull(ls_emp_cd) OR ls_emp_cd = "" THEN
    MsgBox("Error", "従業員コードを入力してください。")
    dw_1.SetColumn("emp_cd")
    dw_1.SetFocus()
    RETURN 0
END IF

SELECT
A.emp_pass
INTO
:ls_emp_pass_db
FROM m_emp A
WHERE A.emp_cd = :ls_emp_cd
IF SQLCA.SQLCode <> 0 THEN
    MsgBox("Error", "従業員コード[" + ls_emp_cd + "]が登録されていません。")
    dw_1.SetColumn("emp_cd")
    dw_1.SetFocus()
    RETURN 0
END IF

// パスワード
ls_emp_pass = dw_1.Object.emp_pass[1]
IF IsNull(ls_emp_pass) OR ls_emp_pass = "" THEN
    MsgBox("Error", "パスワードを入力してください。")
    dw_1.SetColumn("emp_pass")
    dw_1.SetFocus()
    RETURN 0
END IF
IF ls_emp_pass <> ls_emp_pass_db THEN
    MsgBox("Error", "パスワードが違います。")
    dw_1.SetColumn("emp_pass")
    dw_1.SetFocus()
    RETURN 0
END IF
RETURN 0
    
```

データウィンドウの基本動作
 ≪重要キーワード≫
 ・データ管理に使用するバッファ
 Original / Primary / Filter /Delete バッファ
 ・データ操作スクリプト
 カラムデータの取得方法
 ①関数によるアクセス
 dw_1.GetItemX[データ型](行番号, "カラム名")
 ②ドット表記によるアクセス
 dw_1.Object.カラム名[行番号]

スクリプト内で SQL 文を記載時
 取得値を定義済の変数「ls_emp_pass_db」に
 格納できる。

⑦ cb_2 : Cancel

A) clicked イベント

画面を閉じる。

Close(ウインドウ名)になるが、Parent 代名詞を使用している。
 Parent : コントロールから見て、張り付いている親 (Parent) を
 意味する。(自分自身は This になる)

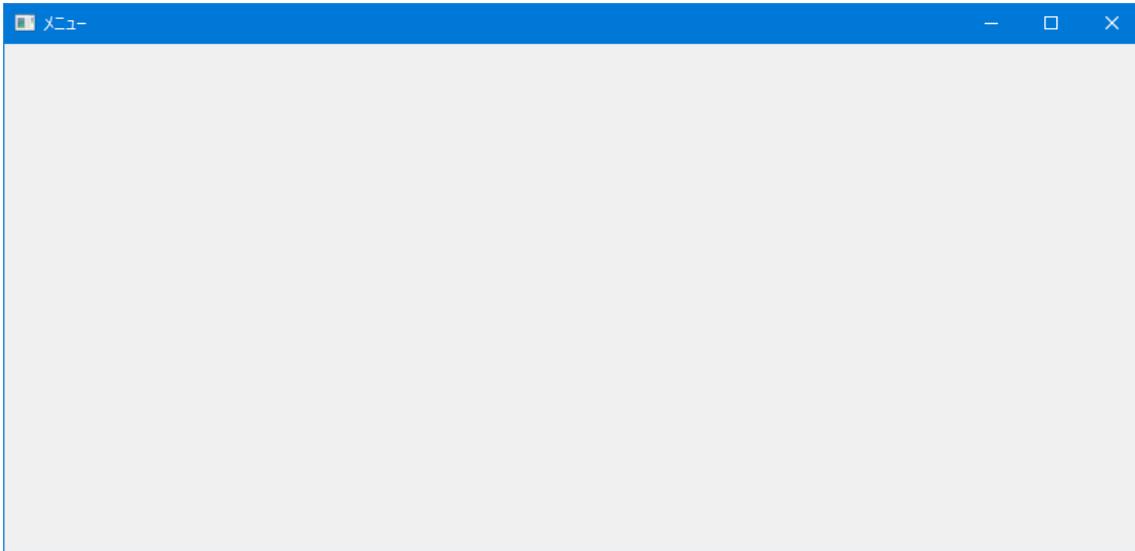
```

window から継承した w_login (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_sdi.pbl) - ウィンドウ
スクリプト - の clicked の戻り値のデータ型は long です
cb_2
Close(parent)
    
```

3. Appeon PowerBuilder 2017 開発手順
(8) メニュー画面 (暫定) の作成

(8) メニュー画面 (暫定) の作成

作成するメニューのイメージ (暫定)



w_menu で保存する。

3. Apeon PowerBuilder 2017 開発手順 (9) アプリ、ログイン、メニューの連結

(9) アプリ、ログイン、メニューの連結

これまで作成したオブジェクトの動作を連結する。

① アプリケーション : sample_pb_sdi

```
application から継承した sample_pb_sdi (sample_pb_sdi) (C:\Project\PFCH\PB教育\Sample_PB_SDI\sample_pb_sdi.pbl)
スクリプト - の open の戻り値のデータ型は (None) です
sample_pb_sdi
String ls_cmd
// EXE実行時の起動引数取得
// アプリEXを起動するときに、
// プログラム名の後に指定した引数を表す文字列を取得できる。
ls_cmd = CommandParm()

// DB接続設定
// 一般的にアプリケーション起動時にDB接続を行う。
SQLCA.DBMS = "ORA Oracle"
SQLCA.ServerName = "or11srv1"
SQLCA.LogId = "SAMPLE_PB"
SQLCA.LogPass = "SAMPLE_PB"
SQLCA.AutoCommit = False
SQLCA.DBParm = "CommitOnDisconnect='No',NLS_Charset='Local'"

// DB接続
CONNECT USING SQLCA;
IF SQLCA.SQLCode <> 0 THEN
    // アプリケーションのCloseイベントを実行して終了する。
    HALT CLOSE
    RETURN
END IF

Open(w_login)
```

② ログインウィンドウ : w_login

```
window から継承した w_login (sample_pb_sdi) (C:\Project\PFCH\PB教育\Sample_PB_SDI\sample_pb_sdi.pbl) - ウィンドウ
スクリプト - の clicked の戻り値のデータ型は long です
cb_1
String ls_emp_cd
String ls_emp_pass
String ls_emp_pass_db
Long ll_count

dw_1.AcceptText()

// 従業員コード
ls_emp_cd = dw_1.Object.emp_cd[1]
IF IsNull(ls_emp_cd) OR ls_emp_cd = "" THEN
    MessageBox("Error", "従業員コードを入力してください。")
    dw_1.SetColumn("emp_cd")
    dw_1.SetFocus()
    RETURN 0
END IF

SELECT
A.emp_pass
INTO
:ls_emp_pass_db
FROM m_emp A
WHERE A.emp_cd = :ls_emp_cd
;
IF SQLCA.SQLCode <> 0 THEN
    MessageBox("Error", "従業員コード[" + ls_emp_cd + "]が登録されておりません。")
    dw_1.SetColumn("emp_cd")
    dw_1.SetFocus()
    RETURN 0
END IF

// パスワード
ls_emp_pass = dw_1.Object.emp_pass[1]
IF IsNull(ls_emp_pass) OR ls_emp_cd = "" THEN
    MessageBox("Error", "パスワードを入力してください。")
    dw_1.SetColumn("emp_pass")
    dw_1.SetFocus()
    RETURN 0
END IF
IF ls_emp_pass <> ls_emp_pass_db THEN
    MessageBox("Error", "パスワードが違います。")
    dw_1.SetColumn("emp_pass")
    dw_1.SetFocus()
    RETURN 0
END IF

Open(w_menu)

RETURN 0
```

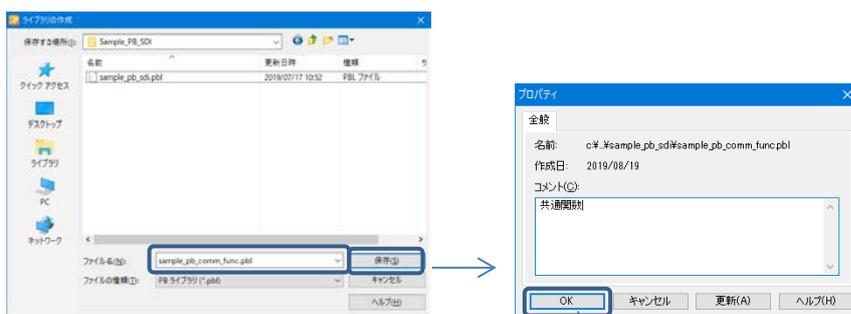
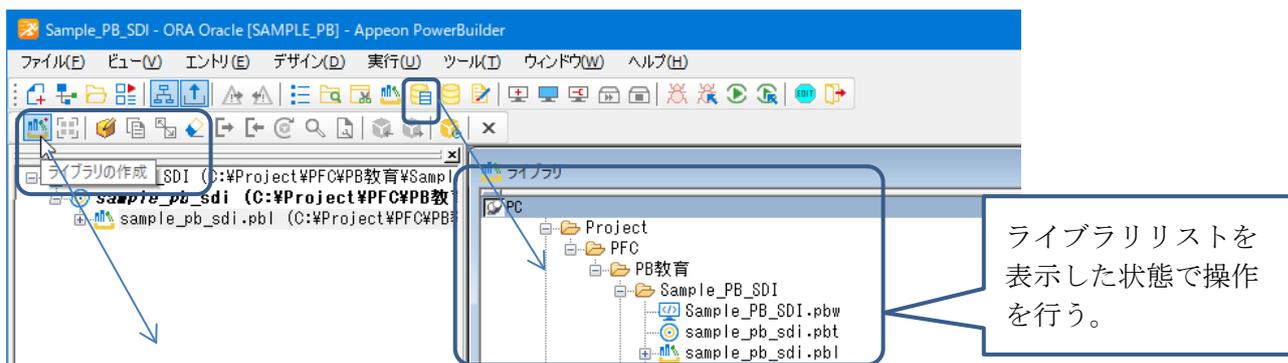
3. Apeon PowerBuilder 2017 開発手順 (10) PBL ファイルの追加

(10) PBL ファイルの追加

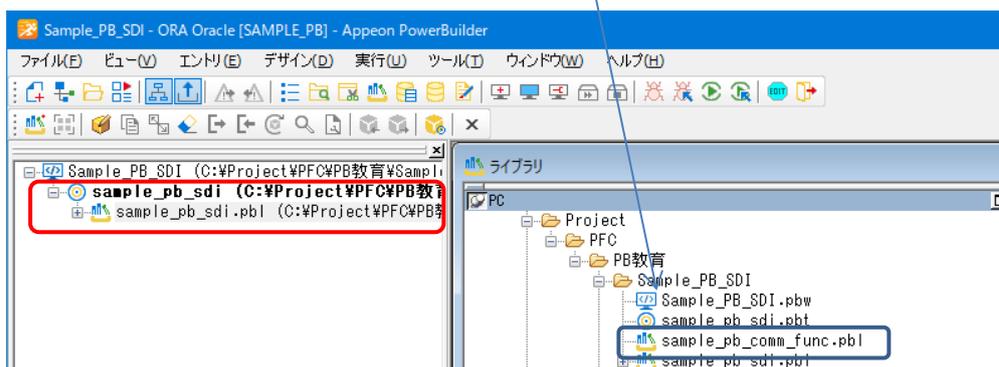
暫定メニューまで作成が済んだ段階で、次のウインドウ作成のために、PBL ファイルを追加する。

機能単位、画面や処理の単位に PBL ファイルを分けた開発を推奨する。

PBL ファイルを追加した後、既存ターゲットのライブラリリストに PBL パスを追加する必要がある。

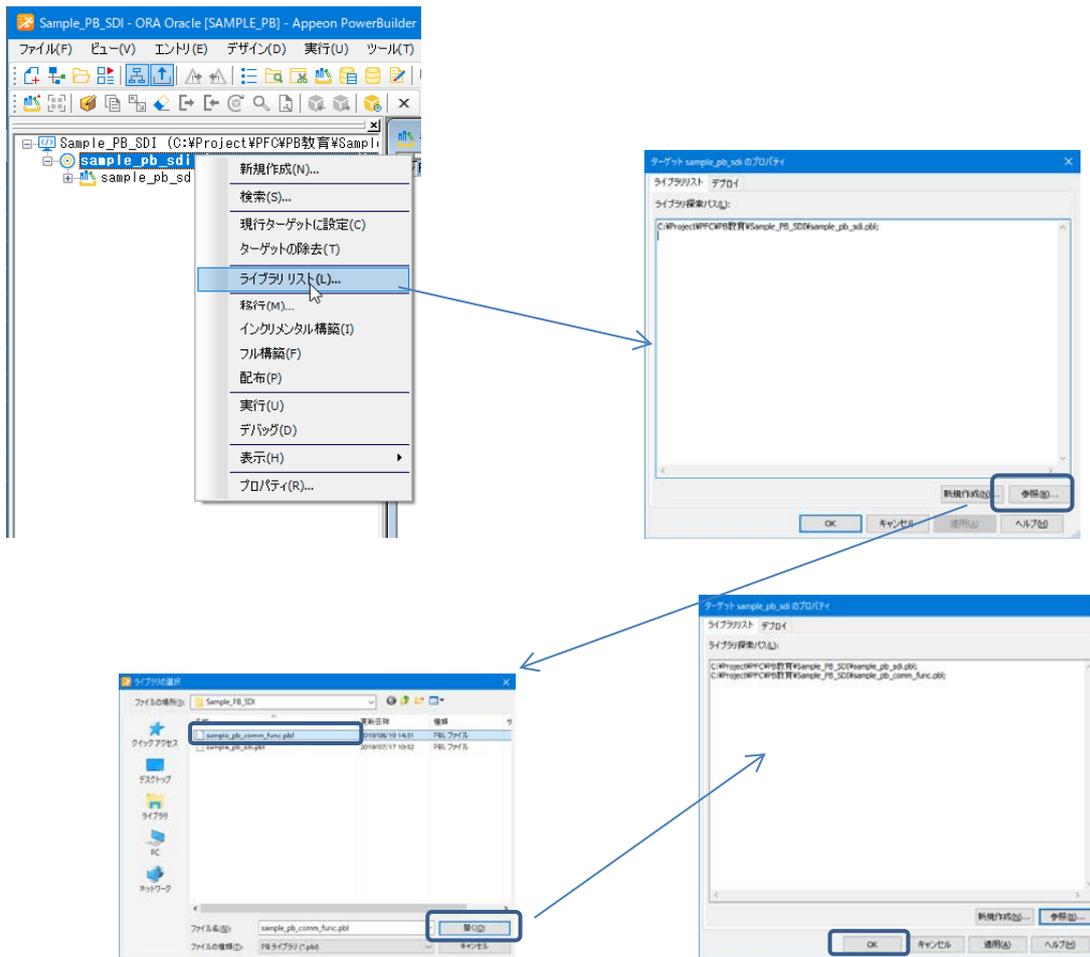


PBL は作成されるが、ライブラリリストにない状態

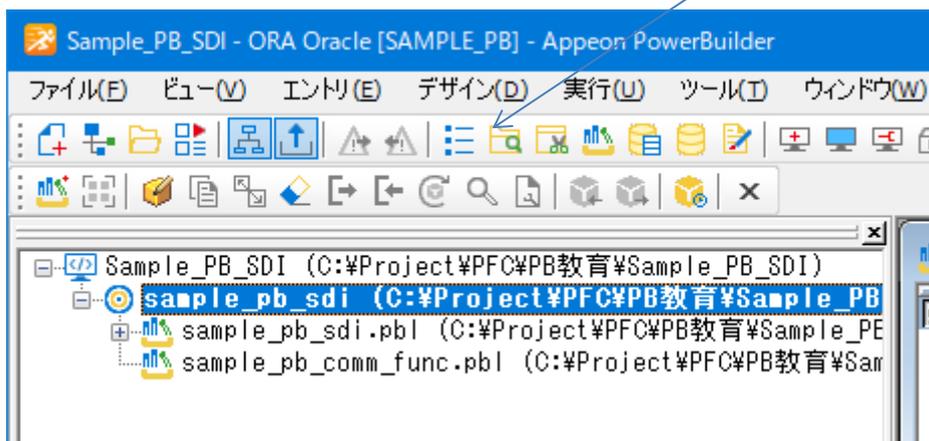


3. Apeon PowerBuilder 2017 開発手順 (10) PBL ファイルの追加

ライブラリリストに登録する。



ライブラリリストに登録された。

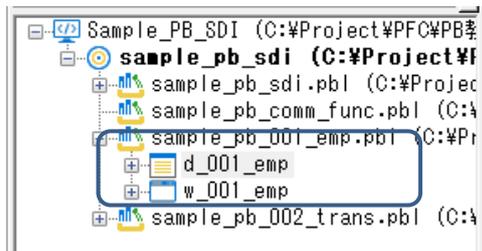


sample_pb_comm_func.pbl と同じ手順で下記 PBL を追加する。

- sample_pb_001_emp.pbl
- sample_pb_002_trans.pbl

3. Apeon PowerBuilder 2017 開発手順 (11) 従業員メンテナンスの作成

(11) 従業員メンテナンスの作成 オブジェクト保存の例



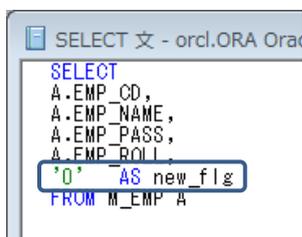
作成するウィンドウのイメージ



① データウィンドウの作成 (d_001_emp)

グリッドで作成する。

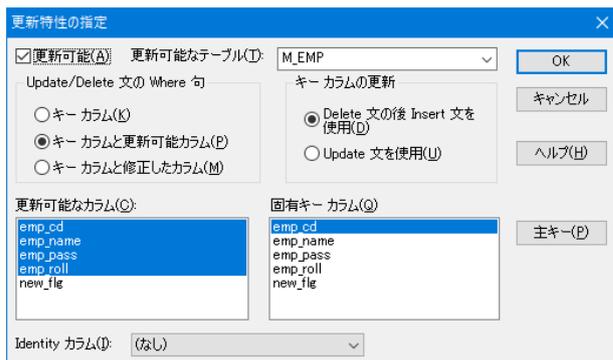
ここでは、制御用のフラグ new_flg を select 文に組み込む。★



見出しの体裁を整え、制御用のフラグはデザイン画面から削除する。

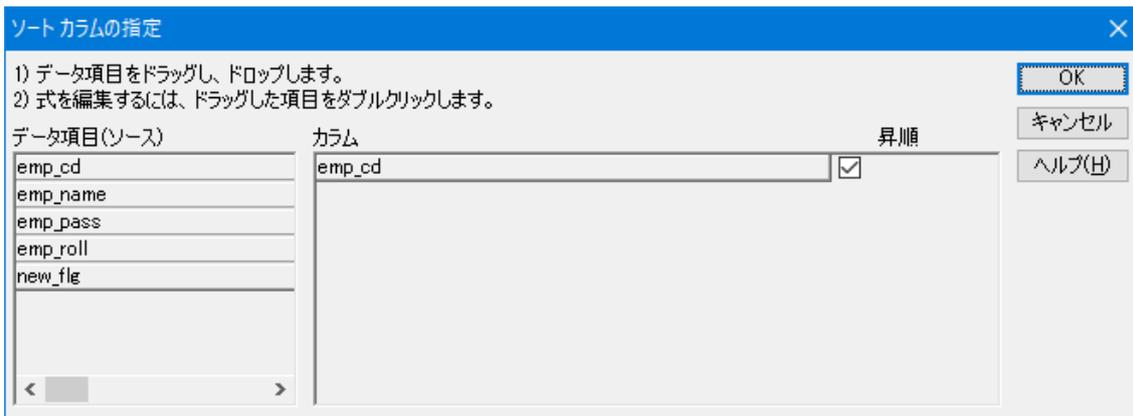


制御用のフラグ (new_flg) は更新特性から外す。



3. Appeon PowerBuilder 2017 開発手順 (11) 従業員メンテナンスの作成

ソートカラムの指定



制御用のフラグ (new_flg) の初期値に 1 を設定する。

カラムの仕様 - d_001_emp

カラム名	データ型	条件指定	初期値	入
1 emp_cd	char(5)	<input type="checkbox"/>		
2 emp_name	char(30)	<input type="checkbox"/>		
3 emp_pass	char(8)	<input type="checkbox"/>		
4 emp_rol	char(10)	<input type="checkbox"/>		
5 new_flg	char(1)	<input type="checkbox"/>	1	

カラムの仕様 - d_001_emp / データ - d_001_emp

制御用のフラグ (new_flg) の意味

一つの開発手法であり、マスタメンテナンス画面用にデータウインドウを作成するとき役に立つ。

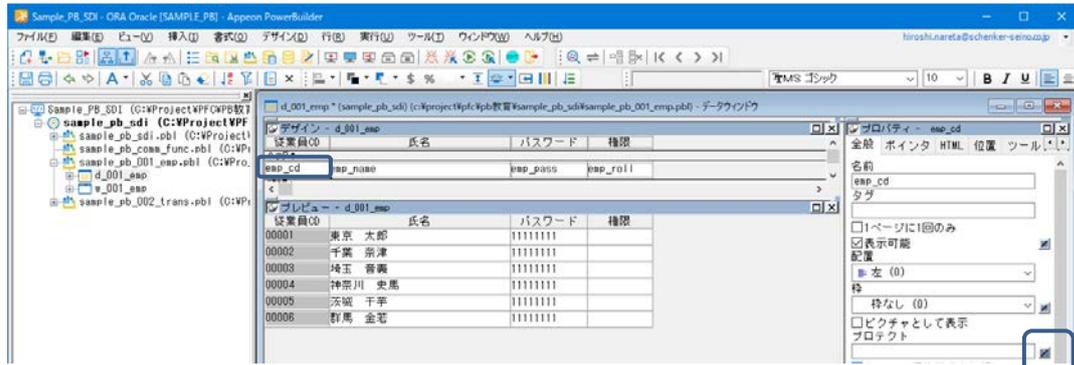
SQL SELECT で検索した行は、new_flg = 0 である。

新規に行追加したときは、new_flg = 1 になる。

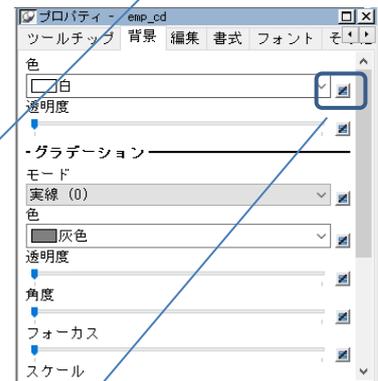
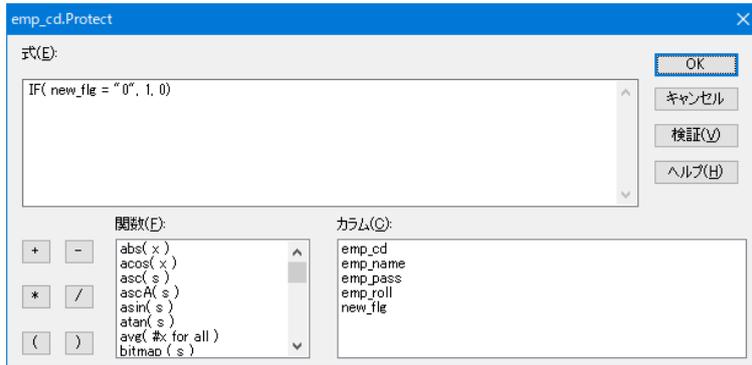
この情報を用いて、主キーに相当するカラムをプロテクトし、登録済のデータの主キー変更を許可しない方式をとれる。

また、背景色を変えることで既存行、新規行の区別が確認できる。

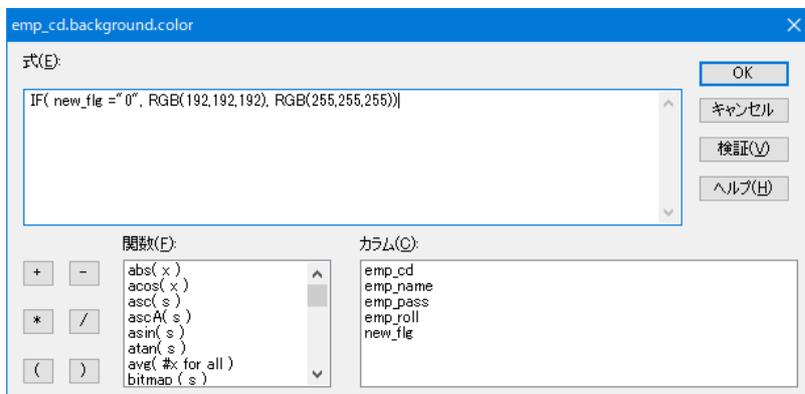
3. Apeon PowerBuilder 2017 開発手順 (11) 従業員メンテナンスの作成



new_flg = "0" のとき、プロテクトする。

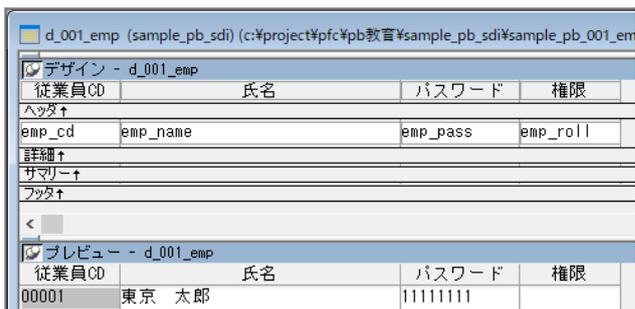


背景色を変更する。



3. Apeon PowerBuilder 2017 開発手順 (1) 従業員メンテナンスの作成

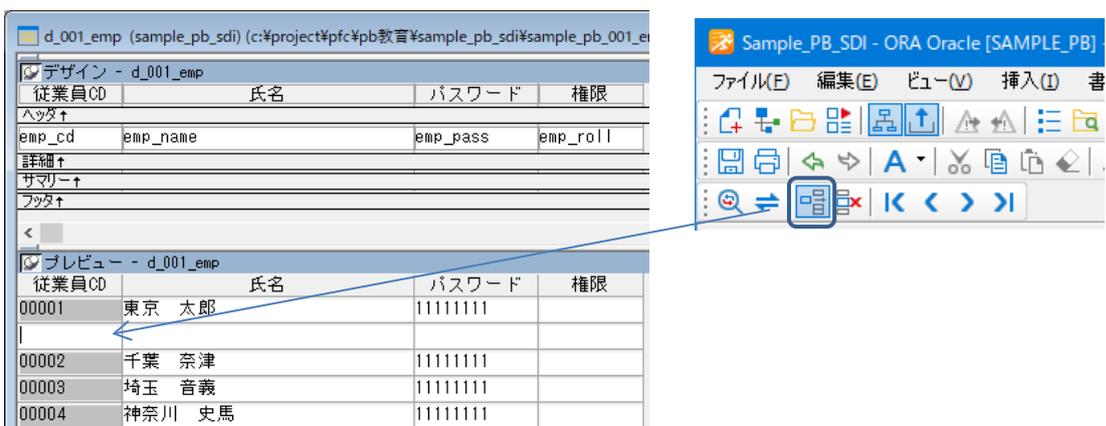
検索データの従業員 CD は、入力できなく灰色になっている。



The screenshot shows a design window for a table named 'd_001_emp'. The table has four columns: '従業員CD', '氏名', 'パスワード', and '権限'. Below the design view, a preview window shows the table data. The '従業員CD' field in the preview is highlighted in grey, indicating it is disabled for input.

従業員CD	氏名	パスワード	権限
00001	東京 太郎	11111111	

行追加された場合は、従業員 CD が入力可能になっている。



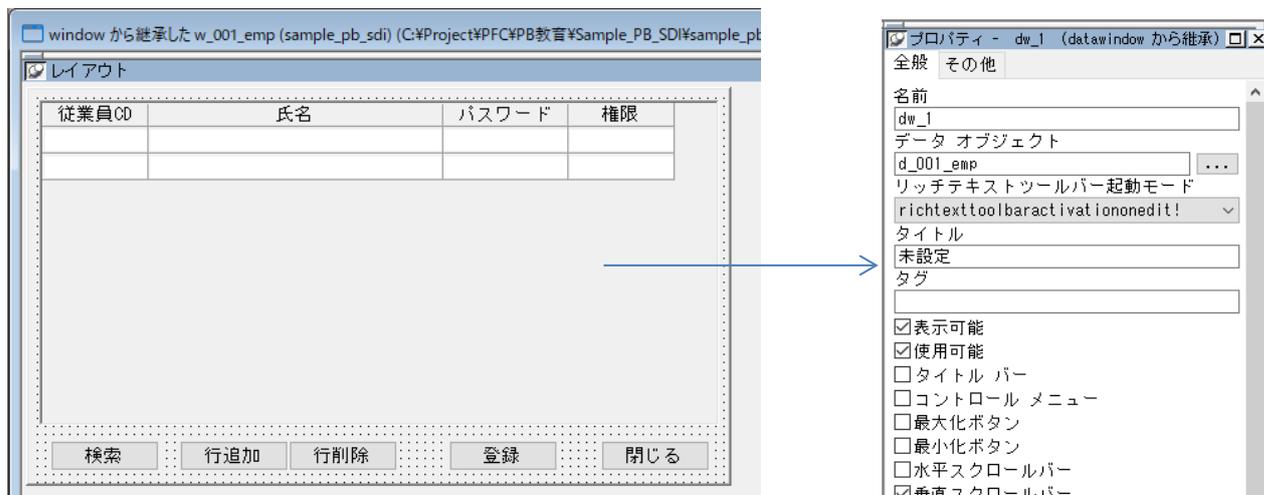
The screenshot shows the same design window as above, but now with multiple rows of data. The '従業員CD' field in the preview is no longer grey. A blue arrow points from the '従業員CD' field in the preview to the '検索' (Search) button in the Oracle menu bar of the 'Sample_PB_SDI - ORA Oracle [SAMPLE_PB]' window, indicating that the search functionality is active.

従業員CD	氏名	パスワード	権限
00001	東京 太郎	11111111	
00002	千葉 奈津	11111111	
00003	埼玉 音義	11111111	
00004	神奈川 史馬	11111111	

3. Apeon PowerBuilder 2017 開発手順 (11) 従業員メンテナンスの作成

② ウィンドウの作成 (w_001_emp)

新規作成し、コントロールを配置する。



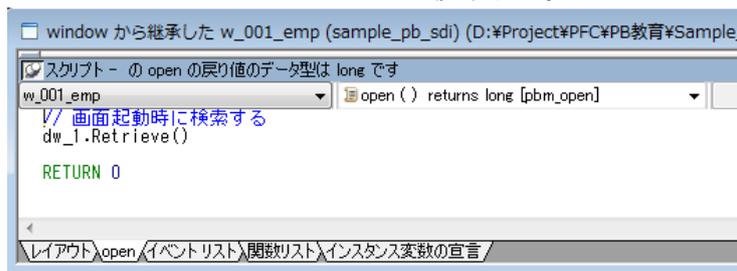
③ スクリプトの記載箇所

- w_001_emp : ウィンドウのイベント
- dw_1 : データウィンドウのイベント
- cb_1 : 検索ボタンのイベント
- cb_2 : 行追加ボタンのイベント
- cb_3 : 行削除ボタンのイベント
- cb_4 : 登録ボタンのイベント
- cb_5 : 閉じるボタンのイベント

④ w_001_emp

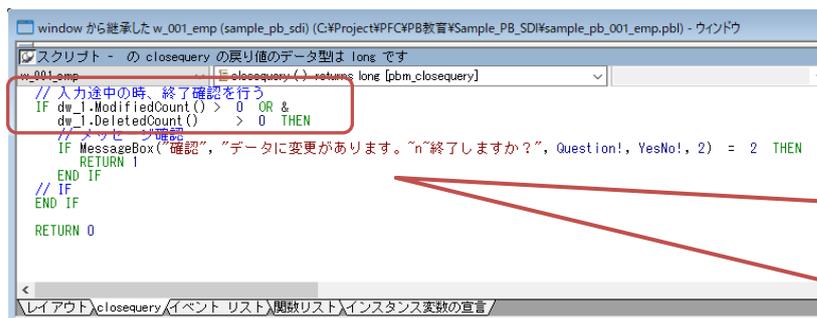
A) open イベント

データを検索する。



B) closequery イベント

画面を閉じる際に、画面データが編集中のときは、確認メッセージを表示する。



- データ行の状態を取得
- ・ dw_1.ModifiedCount()
- 修正済かつ未更新の変更データ行数を取得
- Primary と Filter バッファ内データが対象
- ・ dw_1.DeletedCount()
- 削除予定かつ未更新のデータ行数を取得
- Delete バッファ内データが対象

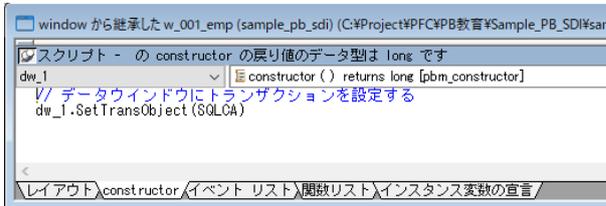
3. Appeon PowerBuilder 2017 開発手順 (11) 従業員メンテナンスの作成

⑤ dw_1 : データウインドウ

A) constructor イベント

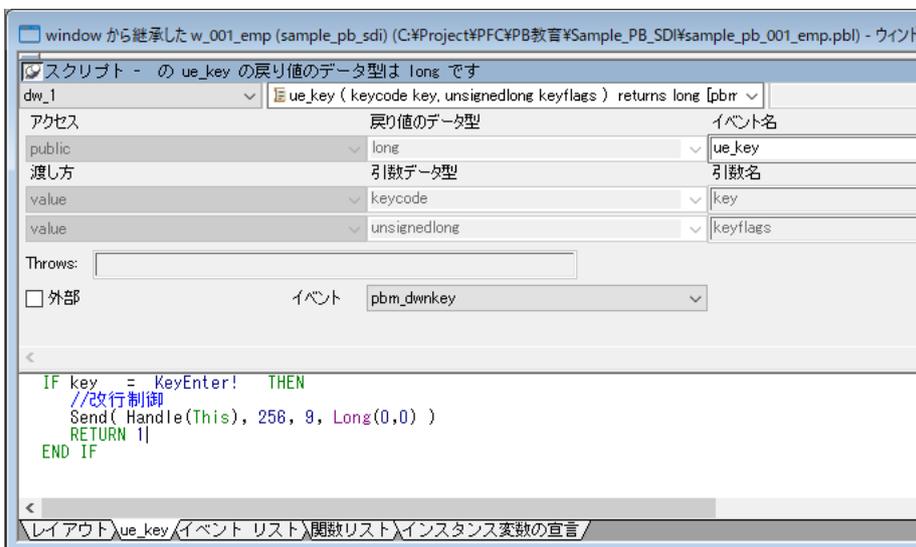
データウインドウをトランザクションで利用可能にする。
データウインドウで DB 操作をする際には必須である。

※当イベントは、ウインドウの open イベントの前に動作する。



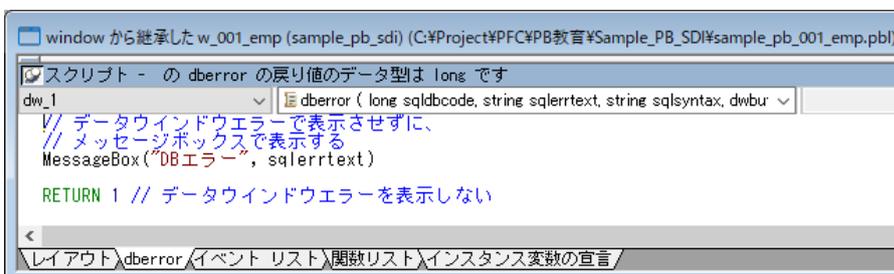
B) ユーザイベントの作成 ue_key

Enter キー押下でカラム移動する仕組みを入れる。



C) dberror イベント

データウインドウエラーの発生時、SQL ベンダーのエラー情報を表示する。



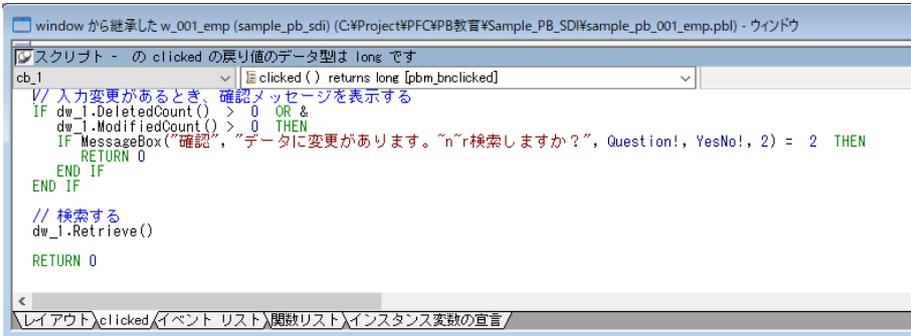
3. Appeon PowerBuilder 2017 開発手順 (11) 従業員メンテナンスの作成

⑥ cb_1 : 検索

A) clicked イベント

データを検索する。

データウインドウに変更があるとき、確認メッセージを表示する。

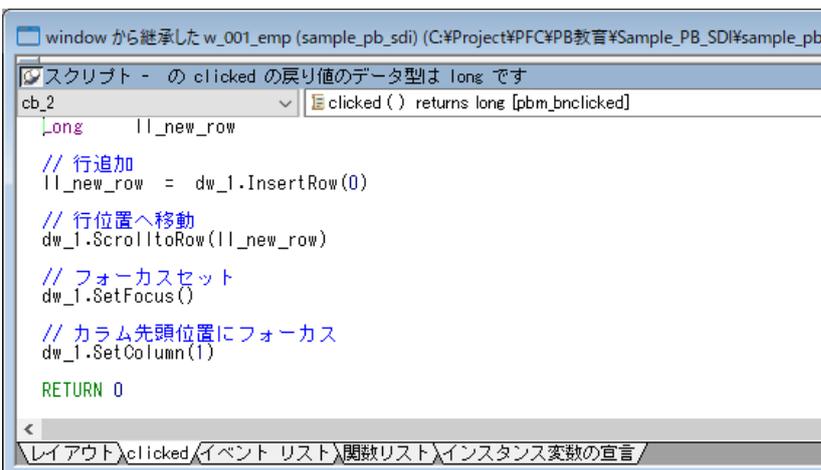


```
window から継承した w_001_emp (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_001_emp.pbl) - ウィンドウ
スクリプト - の clicked の戻り値のデータ型は long です
cb_1 clicked() returns long [pbm_bnclicked]
// 入力変更があるとき、確認メッセージを表示する
IF dw_1.DeletedCount() > 0 OR &
dw_1.ModifiedCount() > 0 THEN
IF MessageBox("確認", "データに変更があります。~n~r検索しますか?", Question!, YesNo!, 2) = 2 THEN
RETURN 0
END IF
END IF
// 検索する
dw_1.Retrieve()
RETURN 0
```

⑦ cb_2 : 行追加

A) clicked イベント

データウインドウに新規行を追加し、その行にフォーカスする。

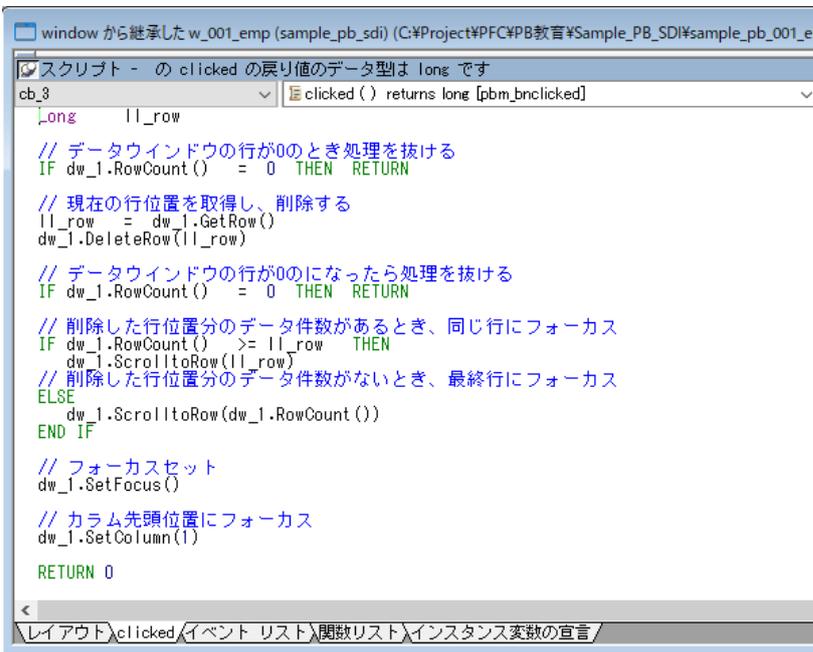


```
window から継承した w_001_emp (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_
スクリプト - の clicked の戻り値のデータ型は long です
cb_2 clicked() returns long [pbm_bnclicked]
long ll_new_row
// 行追加
ll_new_row = dw_1.InsertRow(0)
// 行位置へ移動
dw_1.ScrollToRow(ll_new_row)
// フォーカスセット
dw_1.SetFocus()
// カラム先頭位置にフォーカス
dw_1.SetColumn(1)
RETURN 0
```

⑧ cb_3 : 行削除

A) clicked イベント

データウインドウの既存行を削除し、フォーカスする。



```
long ll_row

// データウインドウの行が0のとき処理を抜ける
IF dw_1.RowCount() = 0 THEN RETURN

// 現在の行位置を取得し、削除する
ll_row = dw_1.GetRow()
dw_1.DeleteRow(ll_row)

// データウインドウの行が0になったら処理を抜ける
IF dw_1.RowCount() = 0 THEN RETURN

// 削除した行位置分のデータ件数があるとき、同じ行にフォーカス
IF dw_1.RowCount() >= ll_row THEN
    dw_1.ScrollToRow(ll_row)
// 削除した行位置分のデータ件数がないとき、最終行にフォーカス
ELSE
    dw_1.ScrollToRow(dw_1.RowCount())
END IF

// フォーカスセット
dw_1.SetFocus()

// カラム先頭位置にフォーカス
dw_1.SetColumn(1)

RETURN 0
```

3. Appeon PowerBuilder 2017 開発手順 (11) 従業員メンテナンスの作成

⑨ cb_4 : 更新

A) clicked イベント

データの入力チェックを行い、データを更新する。

```

window から継承した w_001_emp (sample_pb_sdi) (C:\Project\PF\PB教育\Sample_PB_SDI
スクリプト - の clicked の戻り値のデータ型は long です
cb_4
  Long ll_row
  Long ll_find
  String ls_emp_cd
  String ls_emp_name
  String ls_emp_pass

  // 入力を確定させる
  dw_1.AcceptText()

  // マスタメンテナンスのため、全行チェックする
  FOR ll_row = 1 TO dw_1.RowCount()
    // 主キーの値を取得
    ls_emp_cd = dw_1.Object.emp_cd[ll_row]
    IF ISNULL(ls_emp_cd) OR &
      Trim(ls_emp_cd) = "" THEN
      ELSE
        // 主キー重複チェック
        IF ll_row <> dw_1.RowCount() THEN
          // 既存行の下を検索
          ll_find = dw_1.Find("emp_cd='" + ls_emp_cd + "'", ll_row + 1, dw_1.RowCount())
          // 同一キーが存在する
          IF ll_find > 0 THEN
            MessageBox("キー重複", "[" + ls_emp_cd + "]" + &
              String(ll_row) + "行目と" + &
              String(ll_find) + "行目が重複しています。")
            dw_1.ScrollToRow(ll_row)
            dw_1.SetFocus()
            dw_1.SetColumn(1)
            RETURN 0
          END IF
        END IF
      END IF
      // 入力変更がない行はスキップする
      IF dw_1.GetItemStatus(ll_row, 0, Primary!) = NotModified! OR &
        dw_1.GetItemStatus(ll_row, 0, Primary!) = New! THEN
        CONTINUE
      END IF
      // 主キーが未入力
      IF Trim(ls_emp_cd) = "" OR &
        ISNULL(ls_emp_cd) THEN
        MessageBox("入力エラー", "必須入力です")
        dw_1.ScrollToRow(ll_row)
        dw_1.SetFocus()
        dw_1.SetColumn("emp_cd")
        RETURN 0
      END IF
      // 氏名が未入力
      ls_emp_name = dw_1.Object.emp_name[ll_row]
      IF Trim(ls_emp_name) = "" OR &
        ISNULL(ls_emp_name) THEN
        MessageBox("入力エラー", "必須入力です")
        dw_1.ScrollToRow(ll_row)
        dw_1.SetFocus()
        dw_1.SetColumn("emp_name")
        RETURN 0
      END IF
      // パスワードが未入力
      ls_emp_pass = dw_1.Object.emp_pass[ll_row]
      IF Trim(ls_emp_pass) = "" OR &
        ISNULL(ls_emp_pass) THEN
        MessageBox("入力エラー", "必須入力です")
        dw_1.ScrollToRow(ll_row)
        dw_1.SetFocus()
        dw_1.SetColumn("emp_pass")
        RETURN 0
      END IF
    END IF
  NEXT

  // 更新する
  IF dw_1.Update() = 1 THEN
    COMMIT;
  ELSE
    ROLLBACK;
    RETURN 0
  END IF

  dw_1.Retrieve()
  dw_1.SetFocus()

  RETURN 0

```

データウィンドウカラムに入力した値は、カラムからフォーカスが移らなければ入力が確定されない。
入力直後に更新ボタンを押しても入力が確定されないため、AcceptText 関数を発行する。

レイアウト\clicked\イベント リスト\関数リスト\インスタンス変数の宣言/

⑩ cb_5 : 閉じる

A) clicked イベント

ウインドウを閉じる。

```

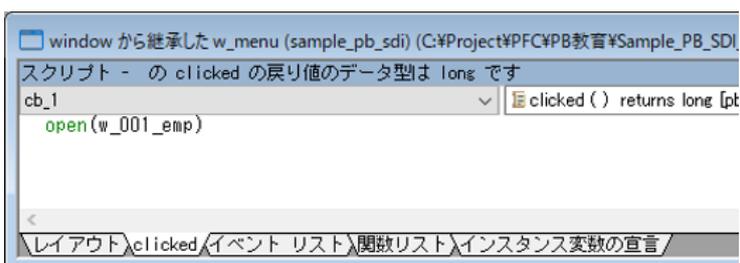
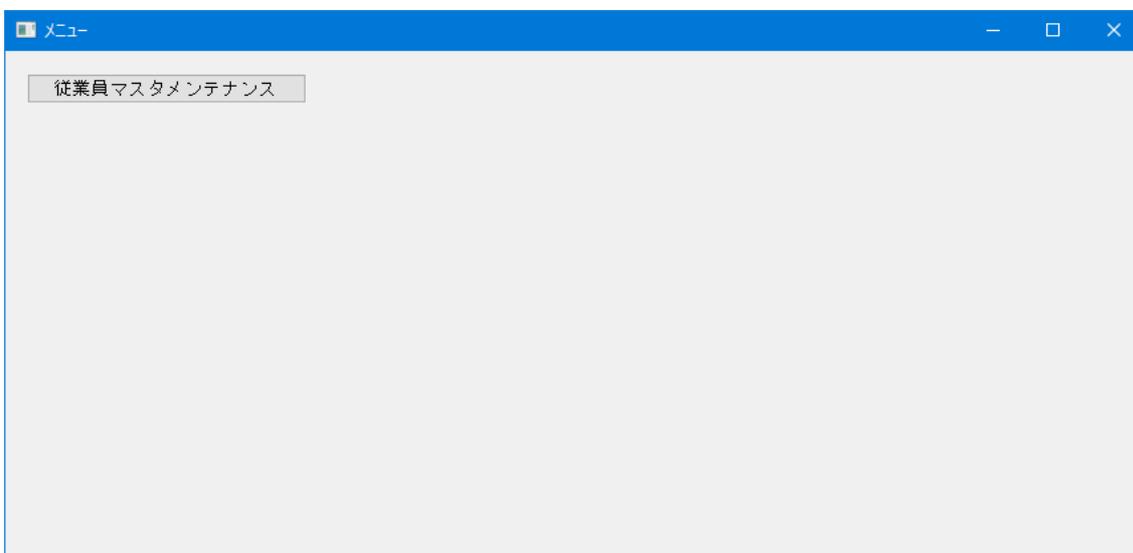
window から継承した w_001_emp (sample_pb_sdi) (C:\Project\PF\PB教育\Sample_PB_SDI
スクリプト - の clicked の戻り値のデータ型は long です
cb_5
  Close(Parent)

```

3. Appeon PowerBuilder 2017 開発手順
(12) 従業員マスタメンテナンスをメニューから起動

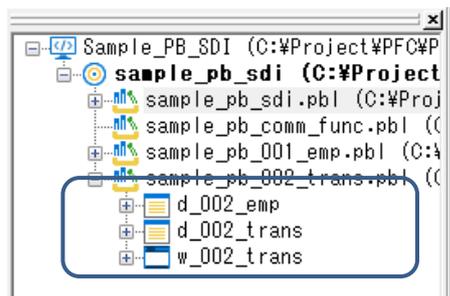
(12) 従業員マスタメンテナンスをメニューから起動

作成するウインドウのイメージ



3. Apeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

(13) 交通費精算画面の作成 オブジェクト保存の例

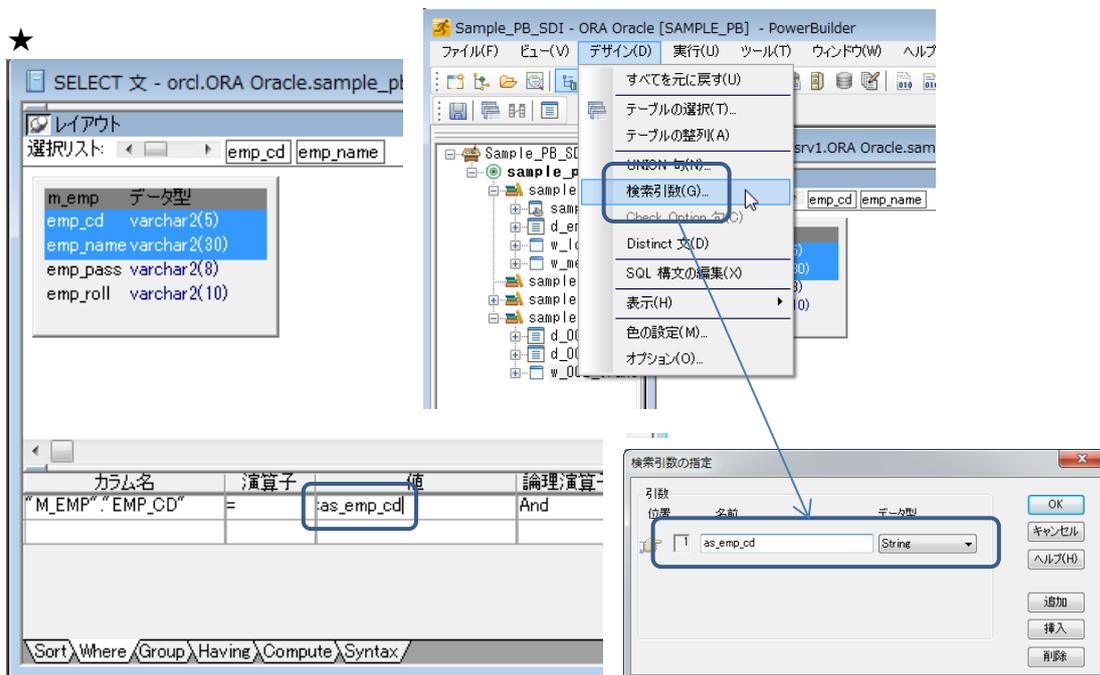


作成するウィンドウのイメージ

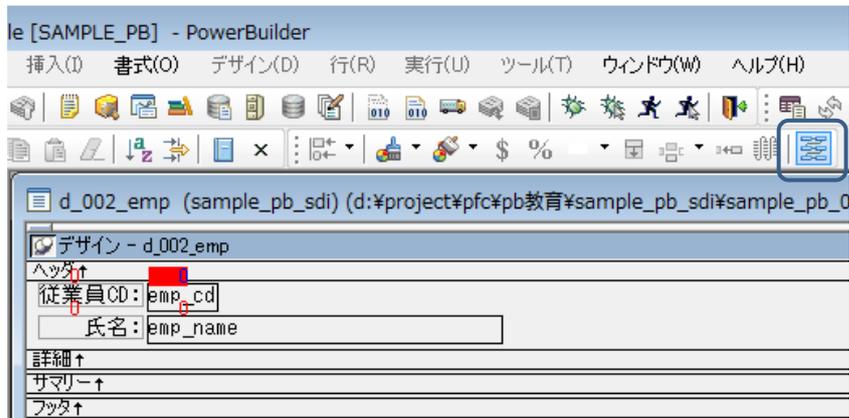


3. Apeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

- ① データウインドウ（画面ヘッダ部用）の作成（d_002_emp）
検索引数を持たせる。



デザイン画面で入力不可能にする。
(タブシーケンスを全て 0 にする) ★



3. Appeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

- ② データウインドウ（画面明細部用）の作成（d_002_trans）
（画面ヘッダ部用）と同様に検索引数を持たせる。

★

```

SELECT 文 - orcl.ORA Oracle.sample_pb1
SELECT
A.EMP_CD,
A.TRANS_YMD,
A.TRANS_SEQ,
A.PURPOSE,
A.VEHICLE,
A.TRANS_FROM,
A.TRANS_VIA,
A.TRANS_TO,
A.ROUND_TRIP,
A.CHARGE,
A.REQUEST,
A.SETTLE,
'0' AS new_flg
FROM T_TRANSPO A
WHERE A.EMP_CD = :as_emp_cd
    
```

プロパティ - trans_ymd

HTML 位置 ツールチップ 背景 編集

編集様式名

様式の種類

エディット マスク

書式

yyyy/mm/dd

アクセラレータ

IME

オフ固定 (3)

自動スキップ

フォーカス時の枠表示

種類

datetime

編集不可

必須フィールド

コード表

ドロップダウン カレンダー

スピン コントロール

スピンの増加分

0

スピンの最小値

0000-00-00 00:00:00.000000

スピンの最大値

0000-00-00 00:00:00.000000

プロパティ - round_trip

HTML 位置 ツールチップ 背景 編集

編集様式名

様式の種類

ドロップダウン リストボックス

大文字/小文字の区別

区別しない (0)

アクセラレータ

IME

なし (0)

編集可能

自動水平スクロール

ソート

空文字列を NULL とする

必須フィールド

リストを常時表示

矢印を常時表示

垂直スクロールバー

最大値

0

コード表

表示の値	データの値
1	片道
2	往復
3	

d_002_trans (sample_pb_sdi) (c:\project\pfc\pb教育\sample_pb_sdi\sample_pb_002_trans.pbl) - データウインドウ

デザイン - d_002_trans

乗車日	預番	業務・行先	乗物	発	経由	着	往復・片道	運賃	申請	精算
trans_ymd	trans_purpose	vehicle	trans_from	trans_via	trans_to	round_trip	charge	<input type="checkbox"/>	settle	

プロパティ - request

位置 ツールチップ 背景 編集 書式

編集様式名

様式の種類

チェックボックス

アクセラレータ

3D 表示

3 種類の状態

テキストを左に表示

サイズをフォントに合わせる

テキスト

オン状態のデータ値

1

オフ状態のデータ値

0

プロパティ - settle

HTML 位置 ツールチップ 背景 編集

編集様式名

様式の種類

ドロップダウン リストボックス

大文字/小文字の区別

区別しない (0)

アクセラレータ

IME

なし (0)

編集可能

自動水平スクロール

ソート

空文字列を NULL とする

必須フィールド

リストを常時表示

矢印を常時表示

垂直スクロールバー

最大値

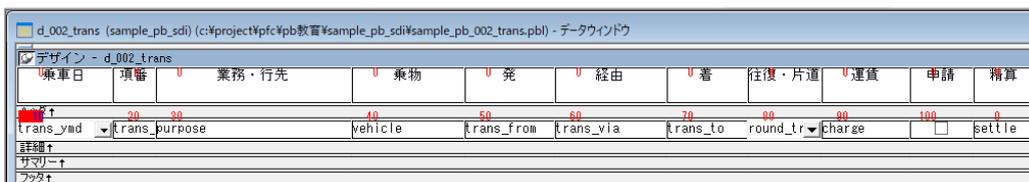
0

コード表

表示の値	データの値
1	未
2	済
3	

3. Appeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

タブオーダー



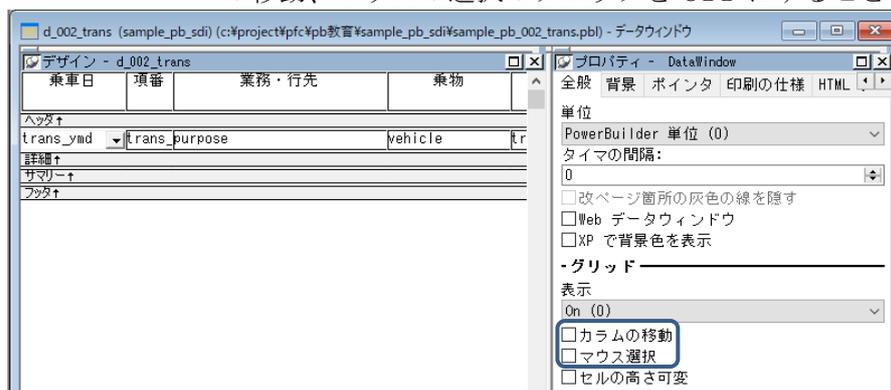
ソートカラムの指定



カラムの移動、マウスの選択のチェックを OFF

入力不可能な行であっても、表面上カラム選択ができて見えてしまい、その行を削除するつもりで画面上の「削除ボタン」を押下したとき、実際にフォーカスが当たっている（入力可能な行）が削除されることから、意図しない動作として錯覚してしまうケースが考えられる。

フォーカスが当たる、当たらない行が混在するケースの場合は、カラムの移動、マウスの選択のチェックを OFF にすることを推奨する。



3. Apeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

プロテクト、背景色 (乗車日～運賃)

The screenshot shows the 'round_trip.background.color' dialog box in PowerBuilder. The formula field contains the following code:

```
IF( settle = "1", RGB(192,192,192),
  IF( request = "1", RGB(192,192,255), RGB(255,255,255)))
```

The dialog also includes a list of functions (関数(E)) and columns (カラム(C)) available for use in the formula. The functions listed are: abs(x), acos(x), asc(s), ascA(s), asin(s), atan(s), avg(#x for all), and bitmap(s). The columns listed are: emp_cd, trans_ymd, trans_seq, purpose, vehicle, trans_from, trans_via, and trans_to.

プロテクト、背景色 (申請)

The screenshot shows the 'request.background.color' dialog box in PowerBuilder. The formula field contains the following code:

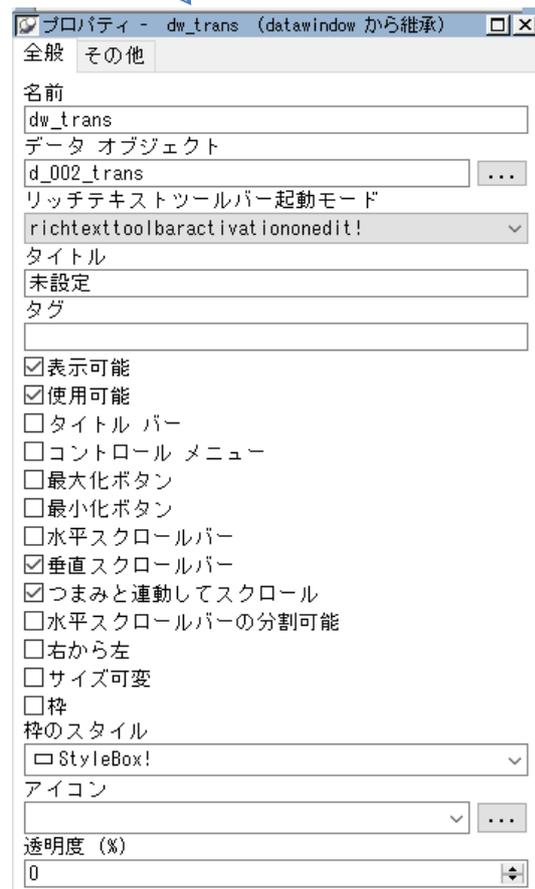
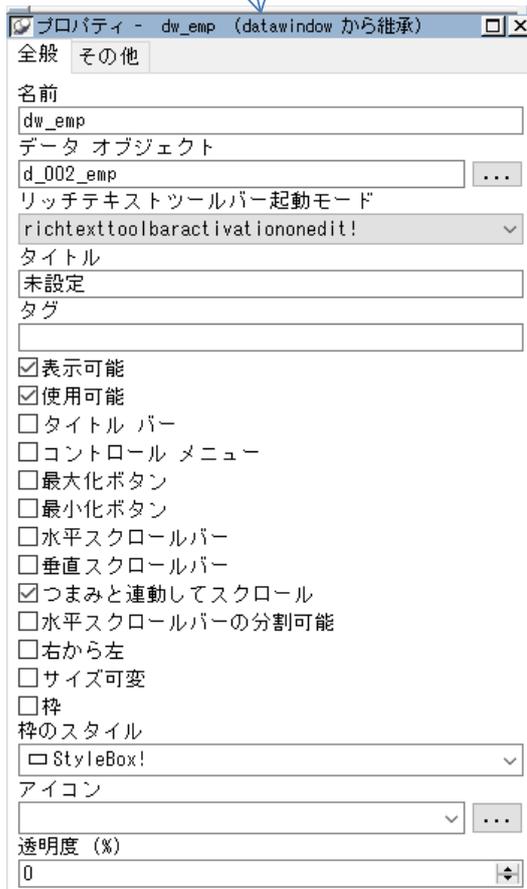
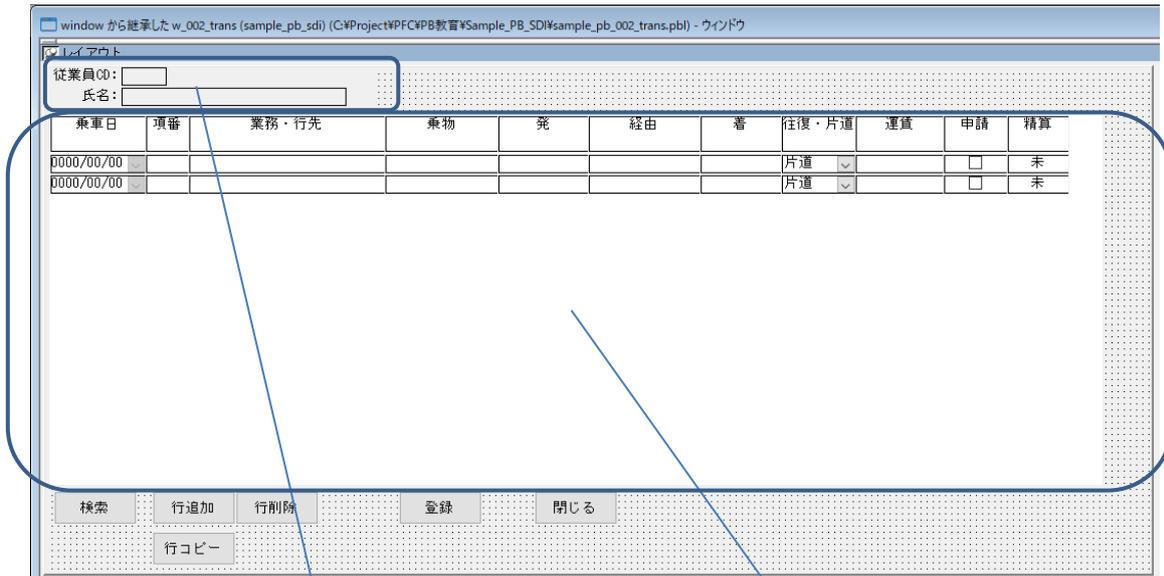
```
IF( settle = "1", RGB(192,192,192), RGB(255,255,255))
```

The dialog also includes a list of functions (関数(E)) and columns (カラム(C)) available for use in the formula. The functions listed are: abs(x), acos(x), asc(s), ascA(s), asin(s), atan(s), avg(#x for all), and bitmap(s). The columns listed are: emp_cd, trans_ymd, trans_seq, purpose, vehicle, trans_from, trans_via, and trans_to.

3. Apeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

③ ウィンドウの作成 (w_002_trans)

新規作成し、コントロールを配置する。



3. Apeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

④ スクリプトの記載箇所

- w_002_trans : ウィンドウのイベント
- dw_emp : 画面ヘッダ部データウィンドウのイベント
- dw_trans : 画面明細部データウィンドウのイベント
- cb_1 : 検索ボタンのイベント
- cb_2 : 行追加ボタンのイベント
- cb_3 : 行削除ボタンのイベント
- cb_4 : 行コピーボタンのイベント
- cb_5 : 登録ボタンのイベント
- cb_6 : 閉じるボタンのイベント

⑤ w_002_trans

A) open イベント

ログインユーザでデータ検索する。

```

// データを検索する
dw_emp.Retrieve(gs_emp_cd)
dw_trans.Retrieve(gs_emp_cd)

RETURN 0
    
```

データウィンドウ側で用意した
検索引数「: as_emp_cd」に
変数「gs_emp_cd」の値を条件にして
各データウィンドウを検索する。

上記の記載前に・・・

gs_emp_cd のグローバル変数を宣言

```

(Declare)
string gs_emp_cd // 従業員CD
    
```

ログインが成功したタイミングでグローバル変数に値をセット

```

// パスワード
ls_emp_pass = dw_1.Object.emp_pass[1]
IF IsNull(ls_emp_pass) OR ls_emp_cd = "" THEN
  MessageBox("Error", "パスワードを入力してください。")
  dw_1.SetColumn("emp_pass")
  dw_1.SetFocus()
  RETURN 0
END IF
IF ls_emp_pass <> ls_emp_pass_db THEN
  MessageBox("Error", "パスワードが違います。")
  dw_1.SetColumn("emp_pass")
  dw_1.SetFocus()
  RETURN 0
END IF

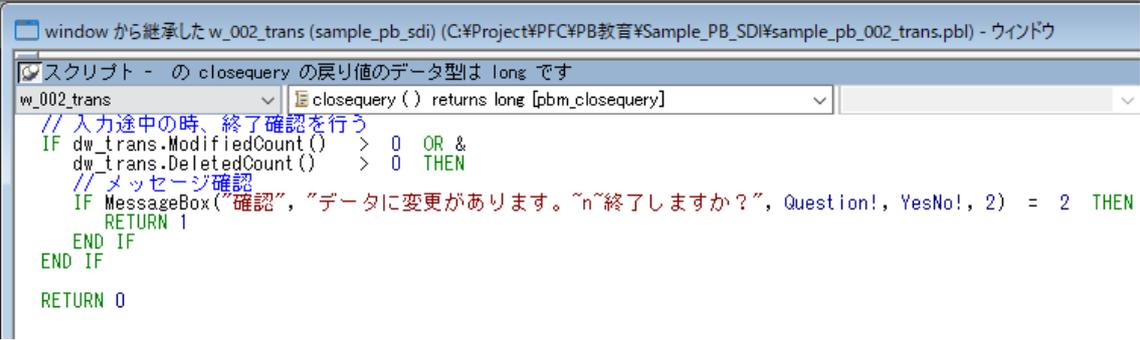
gs_emp_cd = ls_emp_cd
Open(w_menu)

RETURN 0
    
```

3. Appeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

B) closequery イベント

画面を閉じる際に、画面データが編集中的ときは、確認メッセージを表示する。

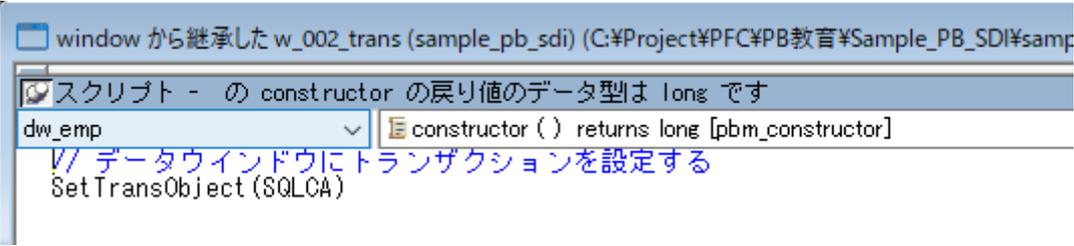


```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_002_trans.pbl) - ウィンドウ
スクリプト - の closequery の戻り値のデータ型は long です
w_002_trans | closequery ( ) returns long [pbm_closequery]
// 入力途中の時、終了確認を行う
IF dw_trans.ModifiedCount() > 0 OR &
dw_trans.DeletedCount() > 0 THEN
// メッセージ確認
IF MessageBox("確認", "データに変更があります。~n~終了しますか?", Question!, YesNo!, 2) = 2 THEN
RETURN 1
END IF
END IF
RETURN 0
```

⑥ dw_emp : 画面ヘッダ部データウインドウ

A) constructor イベント

データウインドウをトランザクションで利用可能にする。



```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_002_trans.pbl) - ウィンドウ
スクリプト - の constructor の戻り値のデータ型は long です
dw_emp | constructor ( ) returns long [pbm_constructor]
// データウインドウにトランザクションを設定する
SetTransObject(SQLCA)
```

3. Appeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

⑦ dw_trans : 画面明細部データウインドウ

A) constructor イベント

データウインドウをトランザクションで利用可能にする。

emp_cd はログイン ID のため、データウインドウの初期値として設定する。

```

window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample
スクリプト - の constructor の戻り値のデータ型は long です
dw_trans      constructor ( ) returns long [pbm_constructor]
// データウインドウにトランザクションを設定する
SetTransObject (SQLCA)
// 主キーの emp_cd は固定になるため、データウインドウの初期値に設定する。
Modify("emp_cd.initial=" + gs_emp_cd + ",")
RETURN 0

```

コードにおけるデータウインドウオブジェクトのプロパティ値のアクセス方法

例) ①新たに挿入された行の従業員カラムの初期値を設定する場合

②カラムの初期値を取得する場合

・メソッドによるアクセス

Describe メソッドと Modify メソッドは文字列を使用して、プロパティ名を指定できる

①Modify("カラム名.initial = " + ~)

②Describe("カラム名.initial")

・Object プロパティとドット (.) 表記

①dw_trans.Object.emp_cd.initial = gs_emp_cd

②ls_initial = dw_trans.Object.emp_cd.initial

B) ユーザイベントの作成 ue_key

Enter キー押下でカラム移動する仕組みを入れる。

```

window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PF\教育\Sample_PB_SDI\sample_pb_002_trans
スクリプト - の ue_key の戻り値のデータ型は long です
dw_trans      ue_key ( keycode key, unsignedlong keyflags ) returns long [pt
アクセス      戻り値のデータ型      イベント名
public        long                  ue_key
渡し方        引数データ型          引数名
value         keycode               key
value         unsignedlong          keyflags
Throws:
外部          イベント              obm dwmkey
IF key = KeyEnter! THEN
//改行制御
Send(Handle(This), 256, 9, Long(0,0))
RETURN 1
END IF

```

3. Appeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

C) dberror イベント

データウインドウエラーの発生時、SQL ベンダーのエラー情報を表示する。

```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PF\PB教育\Sample_PB_SDI\sample_pb_002_trans.pbl) - ウィンドウ
スクリプト - の dberror の戻り値のデータ型は long です
dw_trans
dberror ( long sqlcode, string sqlerrtext)
// データウインドウエラーで表示させずに、
// メッセージボックスで表示する
MessageBox("DBエラー", sqlerrtext)
RETURN 1 // データウインドウエラーを表示しない
```

⑧ cb_1 : 検索

A) clicked イベント

データを検索する。

データウインドウに変更があるとき、確認メッセージを表示する。

```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PF\PB教育\Sample_PB_SDI\sample_pb_002_trans.pbl) - ウィンドウ
スクリプト - の clicked の戻り値のデータ型は long です
cb_1
clicked () returns long [pbm_bnclicked]
// 入力変更があるとき、確認メッセージを表示する
IF dw_trans.DeletedCount() > 0 OR &
dw_trans.ModifiedCount() > 0 THEN
IF MessageBox("確認", "データに変更があります。~\r検索しますか?", Question!, YesNo!, 2) = 2 THEN
RETURN 0
END IF
END IF
// 検索する
dw_trans.Retrieve(gs_emp_cd)
RETURN 0
```

⑨ cb_2 : 行追加

A) clicked イベント

データウインドウに新規行を追加し、その行にフォーカスする。

```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PF\PB教育\Sample_PB_SDI\sample_pb_002_trans.pbl) - ウィンドウ
スクリプト - の clicked の戻り値のデータ型は long です
cb_2
clicked () returns long [pbm_bnclicked]
long ll_new_row
// 行追加
ll_new_row = dw_trans.InsertRow(0)
// 行位置へ移動
dw_trans.ScrollToRow(ll_new_row)
// フォーカスセット
dw_trans.SetFocus()
// カラム先頭位置にフォーカス
dw_trans.SetColumn(1)
RETURN 0
```

⑩ cb_3 : 行削除

A) clicked イベント

データウインドウの既存行を削除し、フォーカスする。
申請中、精算済のデータは削除できない。

```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PFPC\PB教育\Sample_PB_SDI\
スク립ト - の clicked の戻り値のデータ型は long です
cb_3 clicked () returns long [pbm_bnclicked]
long ll_row

// データウインドウの行が0のとき処理を抜ける
IF dw_trans.RowCount() = 0 THEN RETURN

// 現在の行位置を取得し、削除する
ll_row = dw_trans.GetRow()

// 申請中、精算済のデータは削除できない
IF dw_trans.Object.request[ll_row] = "1" OR &
   dw_trans.Object.settle[ll_row] = "1" THEN
RETURN 0
END IF

dw_trans.DeleteRow(ll_row)

// データウインドウの行が0になったら処理を抜ける
IF dw_trans.RowCount() = 0 THEN RETURN

// 削除した行位置分のデータ件数があるとき、同じ行にフォーカス
IF dw_trans.RowCount() >= ll_row THEN
dw_trans.ScrollToRow(ll_row)
// 削除した行位置分のデータ件数がないとき、最終行にフォーカス
ELSE
dw_trans.ScrollToRow(dw_trans.RowCount())
END IF

// フォーカスセット
dw_trans.SetFocus()

// カラム先頭位置にフォーカス
dw_trans.SetColumn(1)

RETURN 0
```

⑪ cb_4 : 行コピー

A) clicked イベント

選択された行をコピーし、新規データとして編集集中の状態にする。

```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PFPC\PB教育\Sample_PB_SDI\sample_pb_002_trans.pbl) - ウ
スク립ト - の clicked の戻り値のデータ型は long です
cb_4 clicked () returns long [pbm_bnclicked]
long ll_row
DateTime ldt_null
Dec ldc_null

SetNull(ldt_null)
SetNull(ldc_null)

// データが1件もないときは処理しない
IF dw_trans.RowCount() = 0 THEN RETURN 0

// 現在の行位置を取得
ll_row = dw_trans.GetRow()

// 最終行位置にコピー
dw_trans.RowsCopy(ll_row, ll_row, Primary!, dw_trans, dw_trans.RowCount() + 1, Primary!)
ll_row = dw_trans.RowCount()

// コピー先のデータを「新規」イメージにする
dw_trans.Object.trans_ymd[ll_row] = ldt_null
dw_trans.Object.trans_seq[ll_row] = ldc_null
dw_trans.Object.request[ll_row] = "0"
dw_trans.Object.settle[ll_row] = "0"

// 行位置へ移動
dw_trans.ScrollToRow(ll_row)

// フォーカスセット
dw_trans.SetFocus()

// カラム先頭位置にフォーカス
dw_trans.SetColumn(1)

RETURN 0
```

3. Apeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

⑫ cb_5 : 更新

A) clicked イベント

データの入力チェックを行い、データを更新する。

```
window から継承した w_002_trans (sample_pb_sdi) (C:\Project\PFCHPB教育\Sample_PB_SDI\sample_pb_002_trans.pbl) - ウィンドウ
スクリプト - の clicked の戻り値のデータ型は long です
cb_5
long ll_row
long ll_find
DateTime ldt_trans_ymd
long ll_trans_seq
string ls_string
dec ldc_decimal

// 入力を確認させる
dw_trans.AcceptText()

// 全行チェックする
FOR ll_row = 1 TO dw_trans.RowCount()
// 主キー重複チェック
ldt_trans_ymd = dw_trans.Object.trans_ymd[ll_row]
ll_trans_seq = dw_trans.Object.trans_seq[ll_row]
// 項番0はNull扱する
IF ll_trans_seq = 0 THEN
SetNull(ll_trans_seq)
END IF
IF IsNull(ldt_trans_ymd) AND IsNull(ll_trans_seq) THEN
ELSE
IF ll_row <> dw_trans.RowCount() THEN
// 既存行の存在検索
ll_find = dw_trans.Find("trans_ymd=DateTime(Date(" + String(ldt_trans_ymd, "yyyy/mm/dd") + " "), Time("00:00:00"))" &
AND trans_seq = " + String(ll_trans_seq, ll_row + 1, dw_trans.RowCount())
// 同一キーが存在する
IF ll_find > 0 THEN
MessageBox("キー重複", String(ldt_trans_ymd, "yyyy/mm/dd") + "[L] + String(ll_trans_seq) + "]" &
String(ll_row) + "行目と" &
String(ll_find) + "行目が重複しています。")
dw_trans.ScrollToRow(ll_row)
dw_trans.SetFocus()
dw_trans.SetColumn(1)
RETURN 0
END IF
END IF
END IF

// 入力変更がない行はスキップする
IF dw_trans.GetItemStatus(ll_row, 0, Primary!) = NotModified! OR &
dw_trans.GetItemStatus(ll_row, 0, Primary!) = New! THEN
CONTINUE
END IF

// 乗車日のチェック
ldt_trans_ymd = dw_trans.Object.trans_ymd[ll_row]
IF IsNull(ldt_trans_ymd) THEN
MessageBox("入力エラー", "必須入力です")
dw_trans.ScrollToRow(ll_row)
dw_trans.SetFocus()
dw_trans.SetColumn("trans_ymd")
RETURN 0
END IF

// 項番のチェック
ll_trans_seq = dw_trans.Object.trans_seq[ll_row]
IF IsNull(ll_trans_seq) OR &
ll_trans_seq <= 0 THEN
MessageBox("入力エラー", "必須入力です (1,2,3...の順で入力)")
dw_trans.ScrollToRow(ll_row)
dw_trans.SetFocus()
dw_trans.SetColumn("trans_seq")
RETURN 0
END IF

// 業務・先行が未入力
ls_string = dw_trans.Object.purpose[ll_row]
IF Trim(ls_string) = "" OR &
IsNull(ls_string) THEN
MessageBox("入力エラー", "必須入力です")
dw_trans.ScrollToRow(ll_row)
dw_trans.SetFocus()
dw_trans.SetColumn("purpose")
RETURN 0
END IF

// 発が未入力
ls_string = dw_trans.Object.trans_from[ll_row]
IF Trim(ls_string) = "" OR &
IsNull(ls_string) THEN
MessageBox("入力エラー", "必須入力です")
dw_trans.ScrollToRow(ll_row)
dw_trans.SetFocus()
dw_trans.SetColumn("trans_from")
RETURN 0
END IF

// 着が未入力
ls_string = dw_trans.Object.trans_to[ll_row]
IF Trim(ls_string) = "" OR &
IsNull(ls_string) THEN
MessageBox("入力エラー", "必須入力です")
dw_trans.ScrollToRow(ll_row)
dw_trans.SetFocus()
dw_trans.SetColumn("trans_to")
RETURN 0
END IF

// 運賃が未入力
ldc_decimal = dw_trans.Object.charge[ll_row]
IF ldc_decimal = 0 OR &
IsNull(ldc_decimal) THEN
MessageBox("入力エラー", "必須入力です")
dw_trans.ScrollToRow(ll_row)
dw_trans.SetFocus()
dw_trans.SetColumn("charge")
RETURN 0
END IF
NEXT

// 更新する
IF dw_trans.Update() = 1 THEN
COMMIT;
ELSE
ROLLBACK;
RETURN 0
END IF

dw_trans.Retrieve(gs_emp_cd)
dw_trans.SetFocus()
RETURN 0
```

DateTime の Find の例

3. Appeon PowerBuilder 2017 開発手順 (13) 交通費精算画面の作成

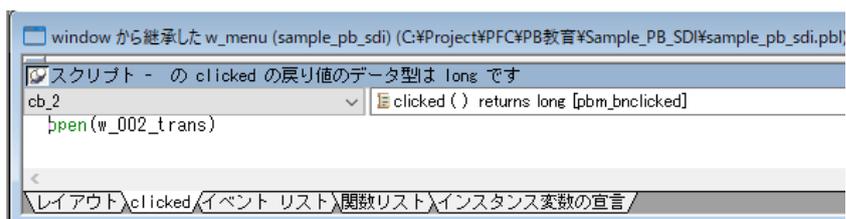
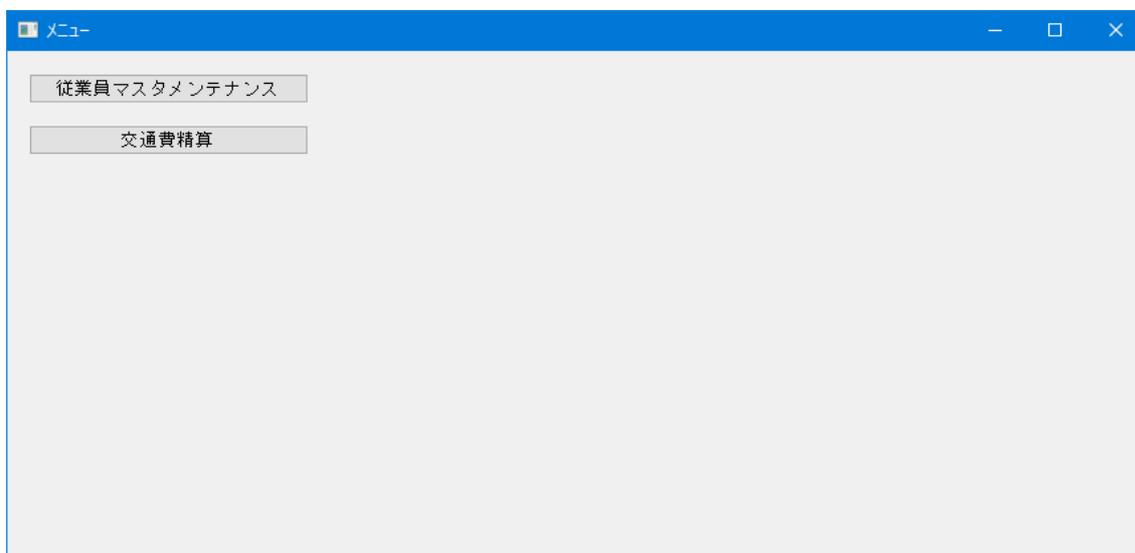
- ⑬ cb_6 : 閉じる
 - A) clicked イベント
ウインドウを閉じる。



3. Appeon PowerBuilder 2017 開発手順 (14)交通費精算画面をメニューから起動

(14) 交通費精算画面をメニューから起動

作成するウィンドウのイメージ



4. まとめ

本書にて、Appen PowerBuilder 2017 アプリケーションの作成について解説してきた。

Appen PowerBuilder 2017 を用いた初歩的な手法であるが、C/S システムの構築方法が理解できたと想定し、次回のコースでは開発効率を向上するための手法について説明する。

Appen PowerBuilder 2017 クラスを用いるメリットと、データウインドウの有効利用について解説する。

以上