前坂 誠二

株式会社ミガロ.

システム事業部 システム2課

[Delphi/400] フレームを利用した開発手法

略歴
1989 年 3 月 21 日生まれ
2011 年 3 月 関西大学文学部卒業
2011 年 4 月 株式会社ミガロ. 入社
2011 年 4 月 システム事業部配属

現在の仕事内容
Delphi/400 を利用したシステム開発や保守作業を担当。Delphi、Delphi/400 の開発経験を積みながら、日々スキルを磨いている。

1.はじめに

プログラムを開発する上で、開発効率・保守性の向上は非常に重要である。開発効率とは、いかに少ない労力で開発が完了するかを表した度合いであり、保守性とは既に作成されたプログラムに対して、維持・管理の容易さを表したものである。

これらを向上させるためには、共通化という考え方が非常に有効である。理由としては、処理を共通化することで、同一の処理を複数回記述する手間が不要となり、プログラムを修正する際には、共通処理の修正のみで作業が完了するからである。

Delphi/400 のプログラムにおいて、処理を共通化する方法には、大きく以下の 4 つの方法がある。

①共通関数ユニットによる処理の共通化
②コンポーネントによる部品の共通化
③継承によるクラスの共通化
④フレームによる画面一部の共通化

本稿では、これらの方法から④のフレームによる共通化の方法を題材としている。

「2.」ではフレームを使用することによるメリット、「3.」ではフレームを作成する手順、「4.」ではフレームの利用方法、「5.」では応用として IBM i 連携を行ったフレームの作成方法について紹介する。なお、本稿でのプログラム例は、Delphi/400 XE7 を使用している。

2.フレームについて

2-1. フレームとは

フレームとは、複数のコンポーネントを含めた画面の一部を 1 つにまとめるクラスのことである。作成したフレームは 1 つのコンポーネントとして流用して貼り付けることができる。

Delphi 言語の特徴には、ツールパレットやオブジェクトインスペクタを利用したビジュアル開発を行える点がある。しかし、開発の際、【図 1】のように、複数画面で部分的に同じ画面設計を作成しなければならない場合がある。そういった時に、今回紹介するフレームが非常に有効である。

2-2. フレーム利用によるメリット

フレームを利用すると、処理が共通化され、「1. はじめに」でも紹介したように、開発効率・保守性が向上する。

フレームでは、他の処理を共通化する方法と異なり、コンポーネントの配置についても共通化できるため、各画面で個別にコンポーネントを配置する必要がなくなり、開発工数の短縮につなげることができる。

3.フレームの作成手順

3-1. C/S アプリケーションでの作成手順

C/S アプリケーションでは、以下の 4 ステップによりフレームを作成できる。

1. ツールバーより、「ファイル」→「新規作成」→「その他」の順で選択する。【図 2】

図1

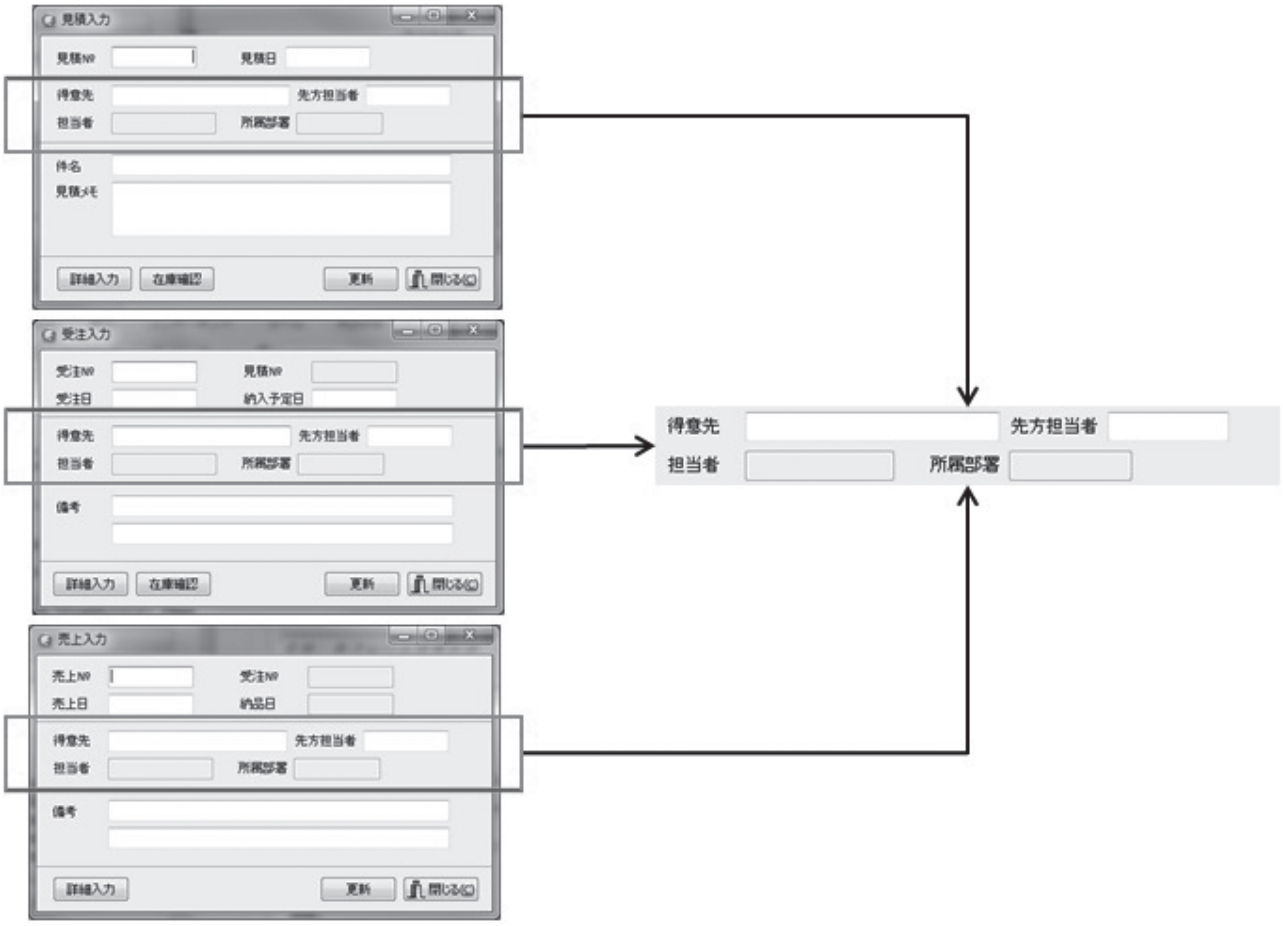
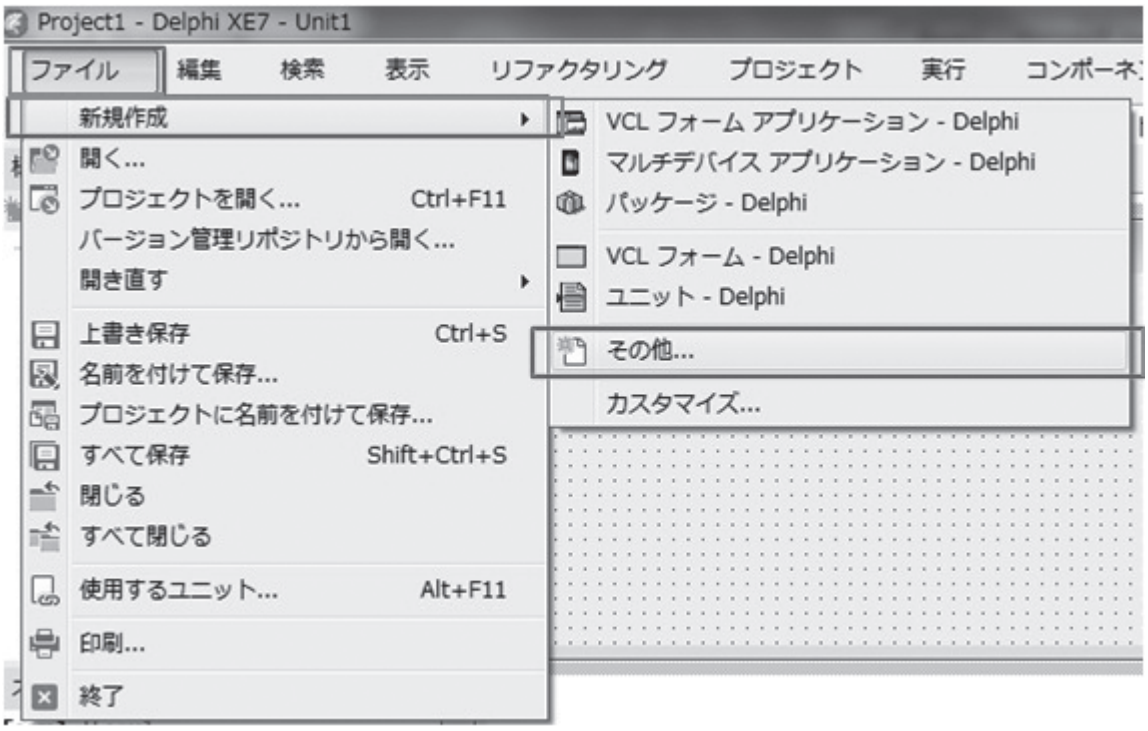

見積入力
見積№
見積日
得意先
先方担当者
担当者
所属部署
件名
見積メモ
詳細入力
在庫確認
更新
閉じる
受注入力
受注№
見積№
受注日
納入予定日
得意先
先方担当者
担当者
所属部署
備考
詳細入力
在庫確認
更新
閉じる
売上入力
売上№
受注№
売上日
納品日
得意先
先方担当者
担当者
所属部署
備考
詳細入力
更新
閉じる
得意先
先方担当者
担当者
所属部署

図2


Project1 - Delphi XE7 - Unit1
ファイル
編集
検索
表示
リファクタリング
プロジェクト
実行
新規作成
開く...
プロジェクトを開く... Ctrl+F11
バージョン管理リポジトリから開く...
開き直す
上書き保存 Ctrl+S
名前を付けて保存...
プロジェクトに名前を付けて保存...
すべて保存 Shift+Ctrl+S
閉じる
すべて閉じる
使用するユニット... Alt+F11
印刷...
終了
VCL フォーム アプリケーション - Delphi
マルチデバイス アプリケーション - Delphi
パッケージ - Delphi
VCL フォーム - Delphi
ユニット - Delphi
その他...
カスタマイズ...

2. 新規作成のダイアログが起動するので、「Delphi プロジェクト」→「Delphi ファイル」より、VCL フレームを選択する。【図 3】
3. 配置したいコンポーネントを貼り付ける。【図 4】
4. 任意のファイル名を入力し、ファイルを保存する。【図 5】

3-2. Web アプリケーションでの作成手順

Delphi/400 の Web アプリケーション構築フレームワークである「IntraWeb」（VCL for the Web）においても、フレーム作成は可能である。また、C/S アプリケーションでの作成手順と同様に、以下のステップで簡単に作成できる。

1. ツールバーより「ファイル」→「新規作成」→「その他」の順で選択する（C/S アプリケーションと同様）。【図 2】
2. 新規作成のダイアログが起動するので、「Delphi プロジェクト」→「IntraWeb」より、NewFrame を選択する。【図 6】
3. 配置したいコンポーネントを貼り付ける。【図 7】
4. 任意のファイル名を入力し、ファイルを保存する（C/S アプリケーションと同様）。【図 5】

3-3. フレームの継承

フレームにおいても、フォームと同様に、継承による作成が可能である。今回は商品リストのフレームを生成し（fraShohin01）、そのフレームを継承して、単価、在庫数、数量、金額の項目を追加したフレーム（fraShohin02）を作成する。【図 8】

まずは、「3-1.」で紹介した手順で、fraShohin01 のフレームを作成する。その後、以下の手順で fraShohin02 のフレームを継承する。

1. ツールバーより「ファイル」→「新規作成」→「その他」の順で選択する。【図 2】
2. 新規作成のダイアログが起動するので、「Delphi プロジェクト」→「継承可能項目」より、fraShohin01 のフレームを選択する。【図 9】
3. 継承されたフレームに、追加で配置したいコンポーネントを貼り付ける。【図 10】
4. 任意のファイル名を入力し、ファイルを保存する。【図 5】

フレームの継承では、継承元フレーム、継承先フレームそれぞれを各画面で利用することが可能である。【図 11】

3-4. フレーム内でのプログラミングポイント

＜フレーム生成時・破棄時のイベント記述＞

フレームは、フォームと同じようにツールパレットからコンポーネントを選択し、配置することで設計を行うことができる。しかし、フレームのオブジェクトインスペクタを確認してみると、フレームはフォームとは違い、生成時の OnCreate イベントや破棄時の OnDestroy イベントが存在しない。【図 12】

そのため、フレームで生成時や破棄時の処理を記述したい場合は、上位クラスより Create、Destroy を継承して内部的にロジックを記述する必要がある。

ロジックの記述方法については、public 宣言にて、Create の場合は constructor、Destroy の場合は destructor と定義する。この際、上位クラスより処理を継承するため、override と記述する点に注意する。

public 宣言にて Create および Destroy の定義を行った後、「Ctrl + Shift + C」キーを押下すると、処理の記述部が補完されるため、フレーム生成時の処理および破棄時の処理を記述する。【ソース 1】

なお、フレーム破棄時の処理を記述する場合は、inherited の前に処理を記述する点に注意する。理由としては、inherited 処理にて、フレーム自体のメモリが解放されるため、記述した処理が正しく動作しない可能性があるためである。

＜プロパティ定義の利用＞

フレームを作成する際、フレーム利用画面との値の受け渡しのために、プロパティ定義を利用することをお勧めする。これにより、フレーム利用画面では、フレームに配置しているコンポーネントを直接指定しなくても値の受け渡しが可能となる。コンポーネントを直接指定しない利点としては、フレーム側のコンポーネントの変更などがあった場合でも、フレーム利用画面側のロジックを修正する必要がなくなる。また、プロパティ定義の read や write に項目ごとのメソッドを記述することで、コンポーネントの Enable 制御や色の変更などを簡単に実装することが可能となる。【ソース 2】

4.フレームの利用方法

4-1. ツールパレットへの追加

フレームを、ツールパレットに追加するには、まずフォームデザイナで作成したフレームを開く。次に右クリックでポップアップメニューを開き、「パレットに追加」を選択する。【図 13】

「パレットに追加」を選択すると、【図 14】のようなダイアログが開く。

ツールパレットで表示させたいコンポーネント名、パレットページ名、設定したいアイコンを選択する。すると、【図 15】のようにツールパレットに作成したフレームが追加される。

4-2. アプリケーションへの追加

作成したフレームをアプリケーションに追加する方法は 2 つある。

①ツールパレットで追加したフレームを選択し、対象画面のフォームデザイナに貼り付ける。【図 16】
②ツールパレットより、「Standard/Frames」を選択し、プロジェクトに登録されているフレームの一覧から、対象のフレームを選択する。【図 17】

4-3. フレームの動的生成

フレームはフォームデザイナ上で貼り付けて使用するだけでなく、ロジックで動的に生成することも可能である。【図 18】

まず、ソースの Uses に生成させたいフレームを定義する（本稿の場合は、ShohinFra02 を定義）。そして、動的生成の方法は【ソース 3】のように、対象のフレームを Create した後、フレーム名や配置などのフレームのプロパティを設定する。フレーム名については、同一の名前で生成した場合、エラーとなるため注意が必要である。また、生成したフレームを破棄する場合は、FreeAndNil（対象フレーム）により破棄することが

図3

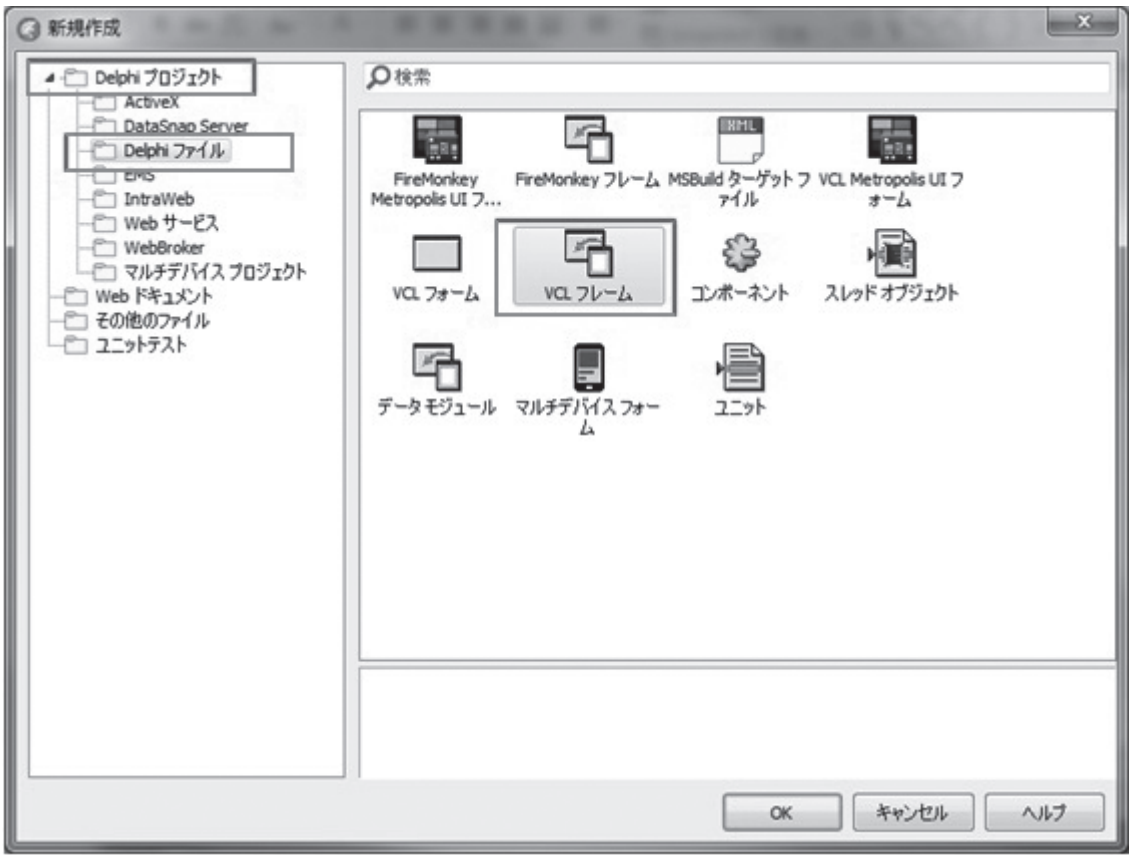
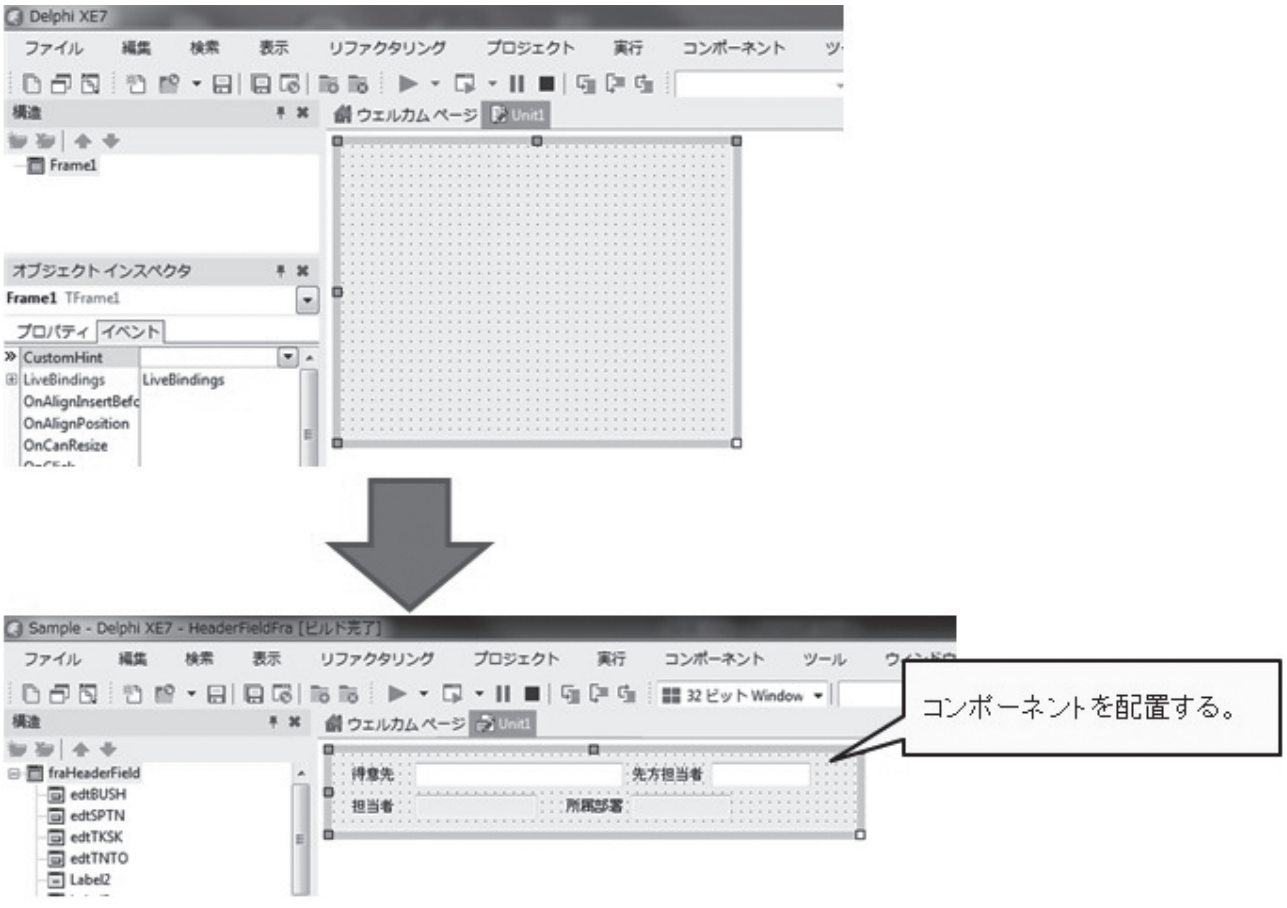

新規作成
Delphi プロジェクト
ActiveX
DataSnap Server
Delphi ファイル
IntraWeb
Web サービス
WebBroker
マルチデバイス プロジェクト
Web ドキュメント
その他のファイル
ユニットテスト
検索
FireMonkey Metropolis UI フ...
FireMonkey フレーム
MSBuild ターゲット ファイル
VCL Metropolis UI フォーム
VCL フォーム
VCL フレーム
コンポーネント
スレッド オブジェクト
データ モジュール
マルチデバイス フォーム
ユニット
OK
キャンセル
ヘルプ

図4


Delphi XE7
ファイル 編集 検索 表示 リファクタリング プロジェクト 実行 コンポーネント
構造
ウェルカム ページ
Unit1
Frame1
オブジェクト インスペクタ
Frame1 TFrame1
プロパティ イベント
CustomHint
LiveBindings
OnAlignInsertBefo
OnAlignPosition
OnCanResize
Sample - Delphi XE7 - HeaderFieldFra [ビルド完了]
32 ビット Window
fraHeaderField
edtBUSH
edtSPTN
edtTKSK
edtTNTO
Label2
得意先
先方担当者
担当者
所属部署
コンポーネントを配置する。

できる。

4-4. フレーム利用画面でのプログラミングポイント

＜フレーム項目値の取得について＞

フレーム利用画面で、フレーム項目の値を取得する2つの方法を紹介する。【ソース4】

①フレーム内コンポーネントを直接指定して値を取得する方法

フレーム名を指定し、そのあとにフレーム内で配置しているコンポーネント名を指定する。

②フレームのプロパティ定義名を指定して値を取得する方法

フレーム内で項目ごとにプロパティ定義を行い、プロパティのwriteメソッドで、画面値のセット処理を行っている場合は、フレーム名＋プロパティ定義名を指定する。

①②どちらの方法でも値の取得自体は可能であるのだが、3-4.で紹介した通り、作成後にコンポーネントの変更があった場合やプログラムのメンテナンスのしやすさを考慮すると、②の方法を使用した方がよい。

＜フレーム利用画面で可能なフレーム項目の変更＞

フレーム利用画面側で、フレームに配置している項目の削除は禁止されている。コンポーネントを選択し、Deleteボタンなどで削除しようとすると、【図19】のようにエラーが表示される。

ただし、以下の内容については、フレーム利用画面側で個別に設定・変更が可能である。

①フレームで配置しているコンポーネントのサイズ変更や位置の移動
②フレームで配置しているコンポーネントのプロパティ値の変更

フレーム利用画面側で変更した内容については、【図20】のようにフレーム自体には反映されないため、画面ごとに個別に設定を変更することも可能である。よって、もし特定の項目を使用したくない場合は、項目のVisibleプロパティをFalseに設定し、非表示にするとよい。

＜フレーム配置項目の処理追加＞

【ソース5】のように、フレーム内に既に処理を組み込んでいる項目に対して、フレーム利用画面側でさらに処理を追加したい場合、【ソース6】のように記述する。フレーム利用画面で同一項目のイベントをダブルクリックすると、既にフレーム側で処理が記述されている場合、「フレーム名.同一項目のイベント(Sender)」といった内容が自動で生成される。フレーム内のイベントより前に処理を追加したい場合、この記述の前に処理を記述するとよい。

5.フレーム内でのIBM i処理

5-1. 使用するコンポーネントとフレーム

本章では、フレームの応用例として、C/SアプリケーションでのIBM iへの接続を行ったフレームの作成および利用方法について紹介する。フレームは「3.」で作成したfraShohin01およびfraShohin02のフレームを使用する。

fraShohin01のフレームでは、TDBLookUpComboBoxを使用し、マスタの内容をリスト形式にて表示する。【図21】

今回の例では、商品マスタを参照する。【図22】

5-2. フレームからのIBMi接続

フレームからIBM i接続を行う方法は、フォーム画面からIBM i接続を行う場合と同様の手順で実装できる。

今回はdbExpress接続を使用し、IBM iとの接続処理を行う。dbExpressのコンポーネントはフレームの中でも定義できるが、フレームごとに新しい接続を定義するとアプリケーション全体での接続が複数になってしまうため、接続を処理するTSQLConnectionは共通のデータモジュールなどを参照するように設計した方がよい。

まず、データモジュールを作成し(dmMain)、TSQLConnectionのコンポーネントを配置する。

このTSQLConnectionのコンポーネントのConnectionNameプロパティには、IBM iに接続するためのCO400 Connectionを設定し、接続パラメータを指定しておく。

fraShohin01のフレームでは、TSQLQuery、TDataSetProvider、TClientDataSet、TDataSourceを配置し、Usesには先ほど作成したデータモジュール（dmMain）を追加する。各IBM iとの接続コンポーネントの設定については、【図23】に示す。

また、TDBLookUpComboBoxのプロパティ設定は【図24】の通りとする。

5-3. フレーム内/利用時のプログラミングポイント

＜フレーム内のプログラミング＞

fraShohin01のフレームでは、商品マスタを参照し、リスト形式で表示する。今回、リスト内容のセットは、SetListItemという名前のサブルーチンにて行う。

リスト内容のセット処理は、フレーム利用画面で、呼び出しが行えるように、public宣言にて記述する。本稿では、データの取得をSQLで行うため、まずはSQL文の記述を行い、その後にClientDataSetのOpen処理を実行して、リスト内容の取得を行う。また、その際にBlankAddのプロパティがTrueで渡された場合は、先頭行をブランク行とするように処理を記述する。【ソース7】【ソース8】

fraShohin01を継承したfraShohin02のフレームでは、商品のリストを選択した時に単価・在庫数をセットするロジックを記述する。【ソース9】

＜フレーム利用画面でのプログラミング＞

フレーム利用画面では、「4.」で紹介したフレームを動的生成する処理を利用する。【ソース3】のロジックに、フレームで定義しているSetListItemの呼出し処理を追加し、リスト選択が可能な詳細入力画面を起動する。【ソース10】

6.まとめ

本稿では、開発効率および保守性を向上させるための手法の一つであるフレームについて紹介した。フレームはフォームと同様にフォームデザイナにて、ツールパレットからコンポーネントを貼り付

図5

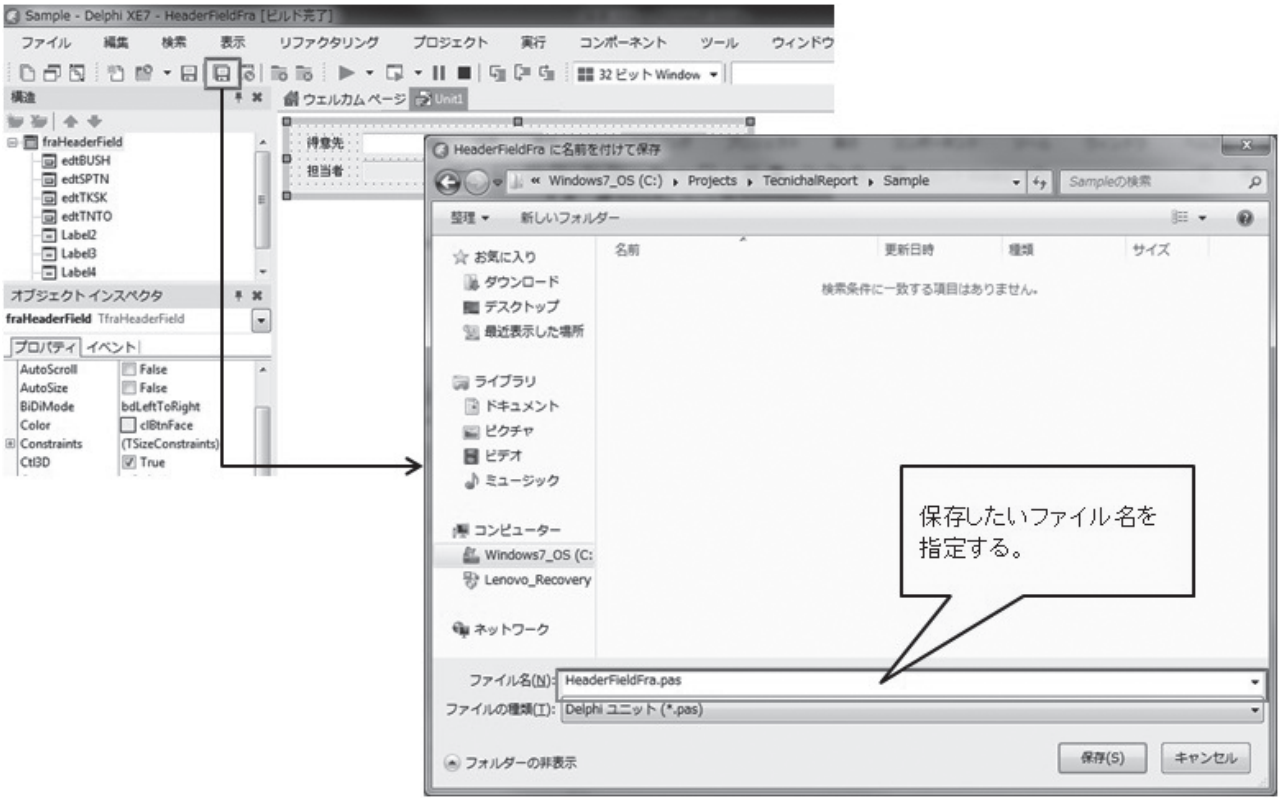
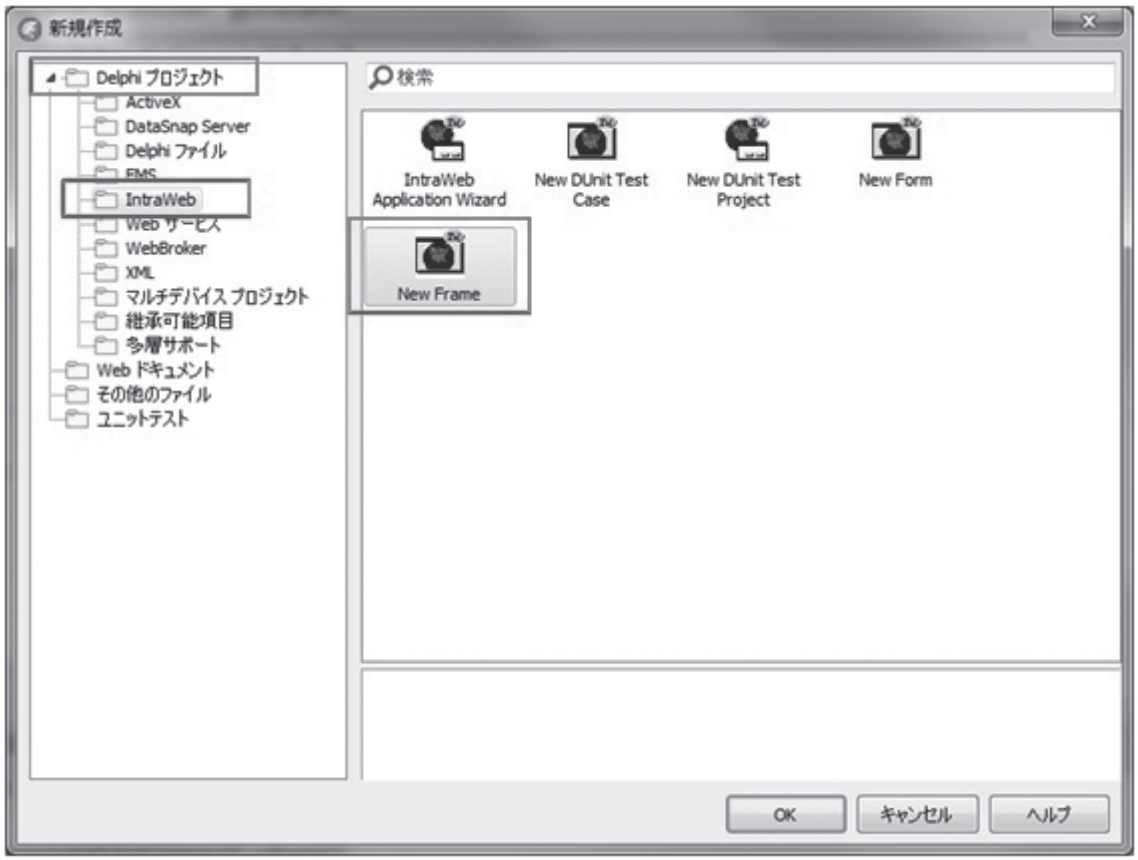

Sample - Delphi XE7 - HeaderFieldFra [ビルド完了]
ファイル 編集 検索 表示 リファクタリング プロジェクト 実行 コンポーネント ツール ウィンドウ
構造
fraHeaderField
edtBUSH
edtSPTN
edtTKSK
edtTNTO
Label2
Label3
Label4
オブジェクト インスペクタ
fraHeaderField TfraHeaderField
プロパティ イベント
AutoScroll False
AutoSize False
BiDiMode bdLeftToRight
Color clBtnFace
Constraints (TSizeConstraints)
Ctl3D True
ウェルカム ページ
Unit1
得意先
担当者
HeaderFieldFra に名前を付けて保存
Windows7_OS (C:) ▸ Projects ▸ TecnichalReport ▸ Sample
Sampleの検索
整理
新しいフォルダー
お気に入り
ダウンロード
デスクトップ
最近表示した場所
ライブラリ
ドキュメント
ピクチャ
ビデオ
ミュージック
コンピューター
Windows7_OS (C:
Lenovo_Recovery
ネットワーク
名前
更新日時
種類
サイズ
検索条件に一致する項目はありません。
保存したいファイル名を
指定する。
ファイル名(N): HeaderFieldFra.pas
ファイルの種類(T): Delphi ユニット (*.pas)
フォルダーの非表示
保存(S)
キャンセル

図6


新規作成
Delphi プロジェクト
ActiveX
DataSnap Server
Delphi ファイル
FMS
IntraWeb
Web サービス
WebBroker
XML
マルチデバイス プロジェクト
継承可能項目
多層サポート
Web ドキュメント
その他のファイル
ユニットテスト
検索
IntraWeb Application Wizard
New DUnit Test Case
New DUnit Test Project
New Form
New Frame
OK
キャンセル
ヘルプ

けるだけで設計が可能であるため、作成方法については通常のフォーム開発と同じである。

また、フレームを使用する際は、作成したフレームをフォームに貼り付けるだけでコンポーネント同様に使用できるため、誰でも容易にフレームを利用した共通化が行える。

「1.」で説明した通り、Delphi/400では処理を共通化する手法がいくつか用意されている。

コンポーネントや継承といった方法は、一般的によく使用されているが、今回紹介したフレームは意外と使われていない（知られていない）ことも多い。

しかしこのフレームを使った開発は、コンポーネント作成より簡単で、さらに画面の一部分だけを共通化できるため、画面単位の継承よりも流用性が高い。この機能を知った上で開発を行えば、必ず開発効率、保守性に役立つはずである。

M

図7

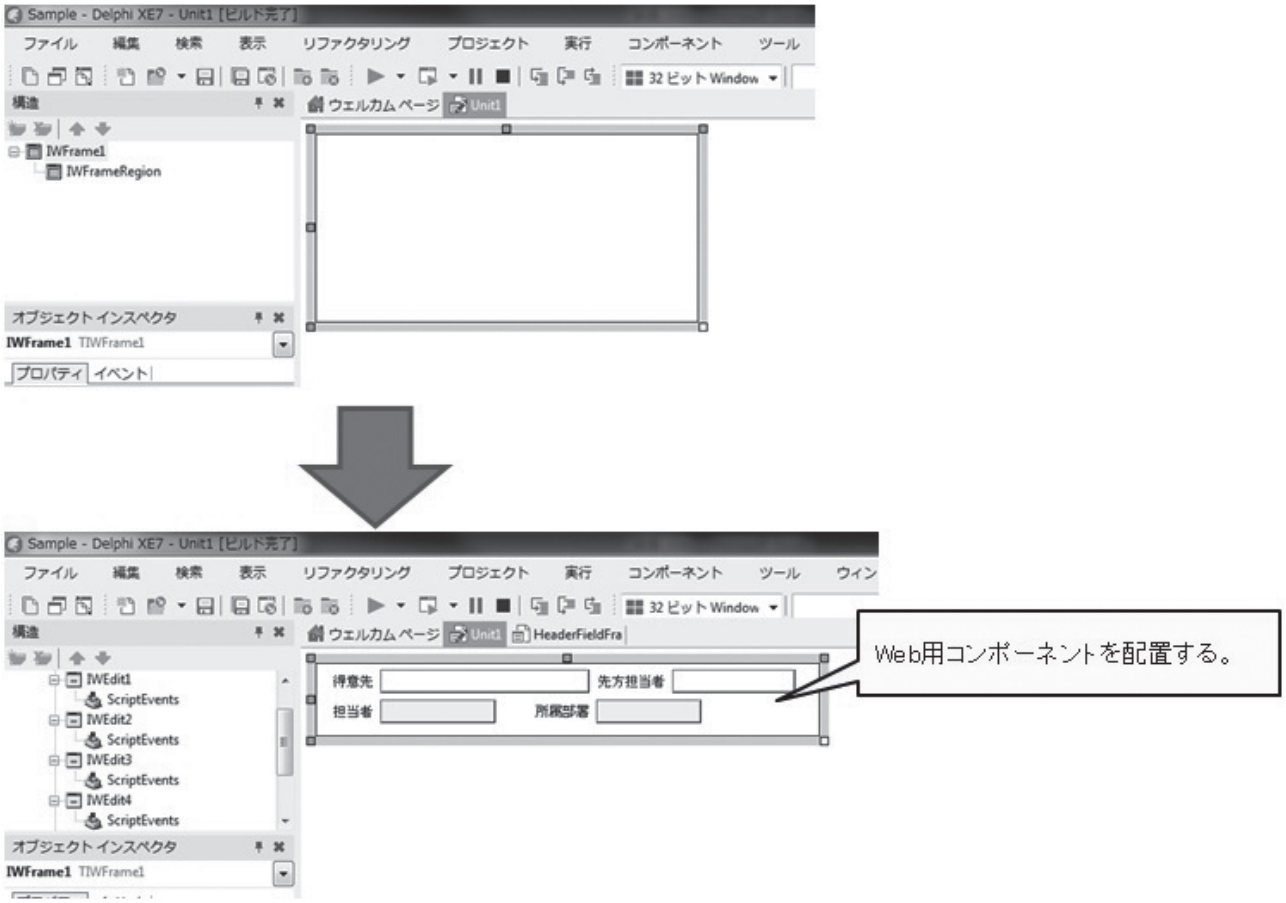

Sample - Delphi XE7 - Unit1 [ビルド完了]
ファイル 編集 検索 表示 リファクタリング プロジェクト 実行 コンポーネント ツール
32 ビット Window
構造
ウェルカムページ Unit1
IWFrame1
IWFrameRegion
オブジェクト インスペクタ
IWFrame1 TIWFrame1
プロパティ イベント
Sample - Delphi XE7 - Unit1 [ビルド完了]
ファイル 編集 検索 表示 リファクタリング プロジェクト 実行 コンポーネント ツール ウィン
32 ビット Window
構造
ウェルカムページ Unit1 HeaderFieldFra
IWEdit1
ScriptEvents
IWEdit2
ScriptEvents
IWEdit3
ScriptEvents
IWEdit4
ScriptEvents
得意先
先方担当者
担当者
所属部署
オブジェクト インスペクタ
IWFrame1 TIWFrame1
Web用コンポーネントを配置する。

図8

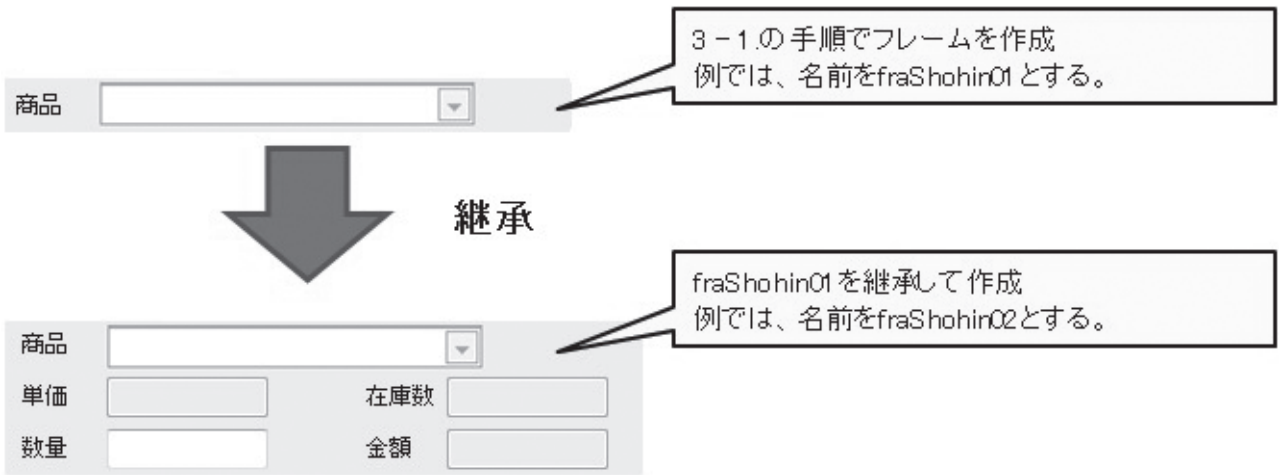

商品
3－1の手順でフレームを作成
例では、名前をfraShohin01とする。
継承
商品
単価
在庫数
数量
金額
fraShohin01を継承して作成
例では、名前をfraShohin02とする。

図9

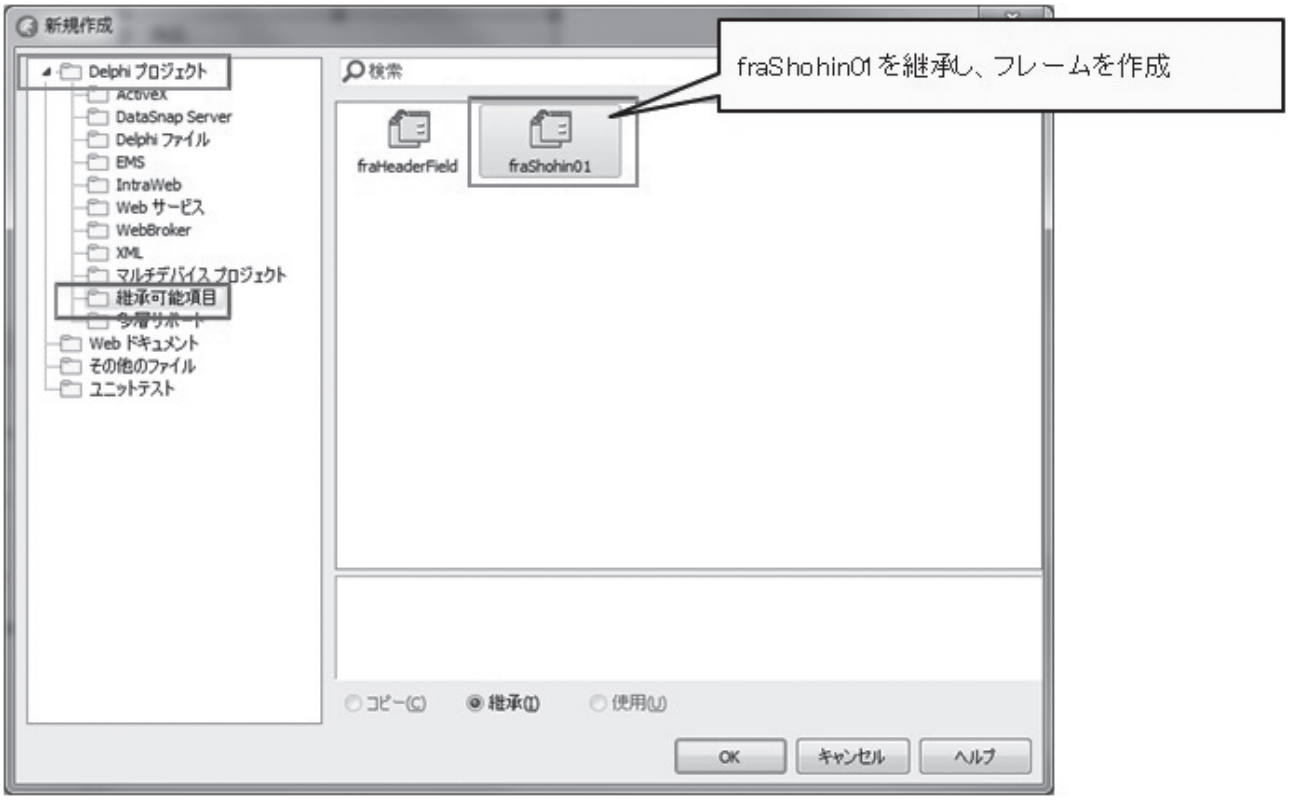

新規作成
Delphi プロジェクト
ActiveX
DataSnap Server
Delphi ファイル
EMS
IntraWeb
Web サービス
WebBroker
XML
マルチデバイス プロジェクト
継承可能項目
多層サポート
Web ドキュメント
その他のファイル
ユニットテスト
検索
fraHeaderField
fraShohin01
fraShohin01を継承し、フレームを作成
コピー(C)
継承(I)
使用(U)
OK
キャンセル
ヘルプ

図10

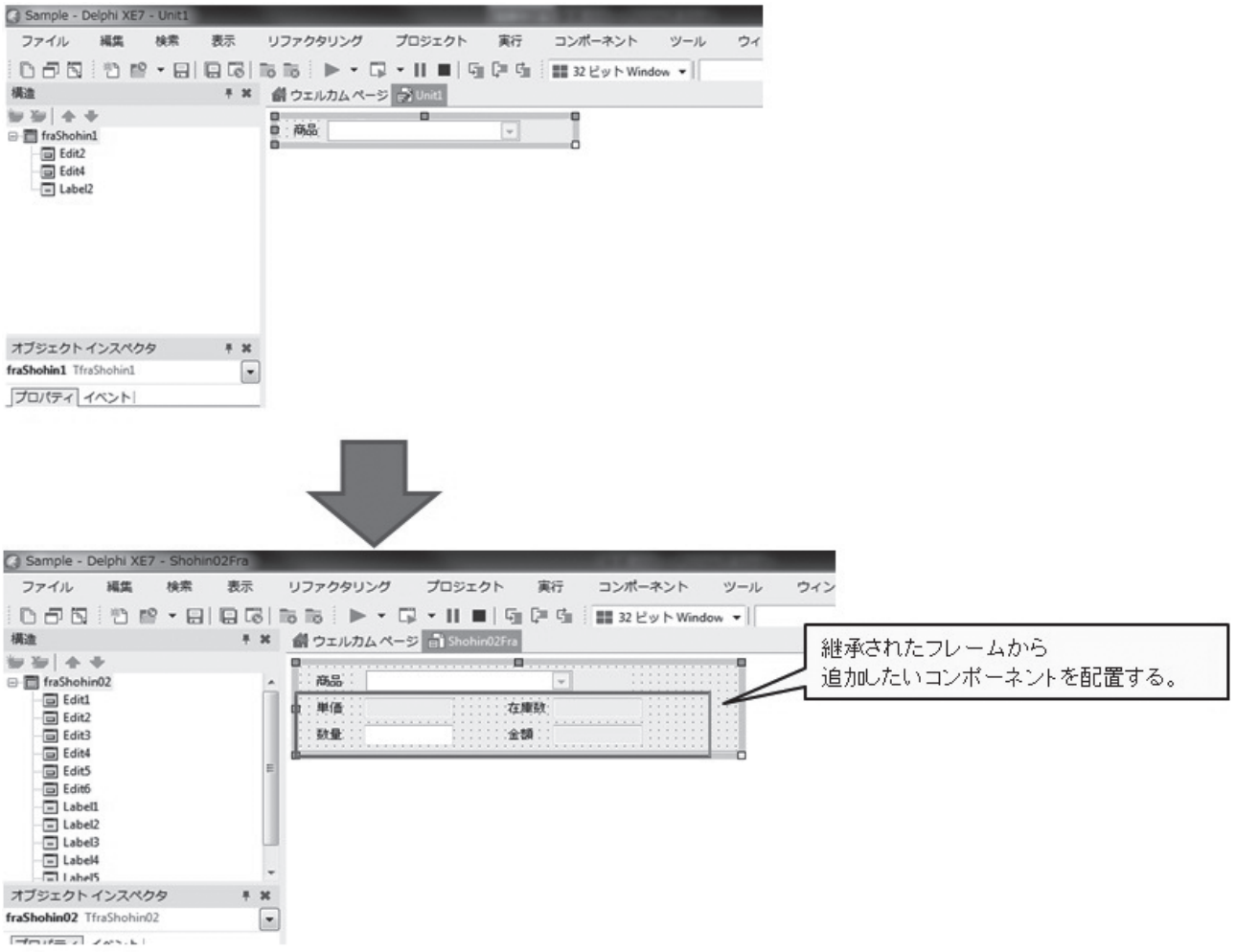

Sample - Delphi XE7 - Unit1
ファイル 編集 検索 表示 リファクタリング プロジェクト 実行 コンポーネント ツール
32 ビット Window
構造
ウェルカム ページ
Unit1
fraShohin1
Edit2
Edit4
Label2
商品
オブジェクト インスペクタ
fraShohin1 TfraShohin1
プロパティ イベント
Sample - Delphi XE7 - Shohin02Fra
ファイル 編集 検索 表示 リファクタリング プロジェクト 実行 コンポーネント ツール
32 ビット Window
構造
ウェルカム ページ
Shohin02Fra
fraShohin02
Edit1
Edit2
Edit3
Edit4
Edit5
Edit6
Label1
Label2
Label3
Label4
Label5
商品
単価
在庫数
数量
金額
オブジェクト インスペクタ
fraShohin02 TfraShohin02
継承されたフレームから
追加したいコンポーネントを配置する。

図11

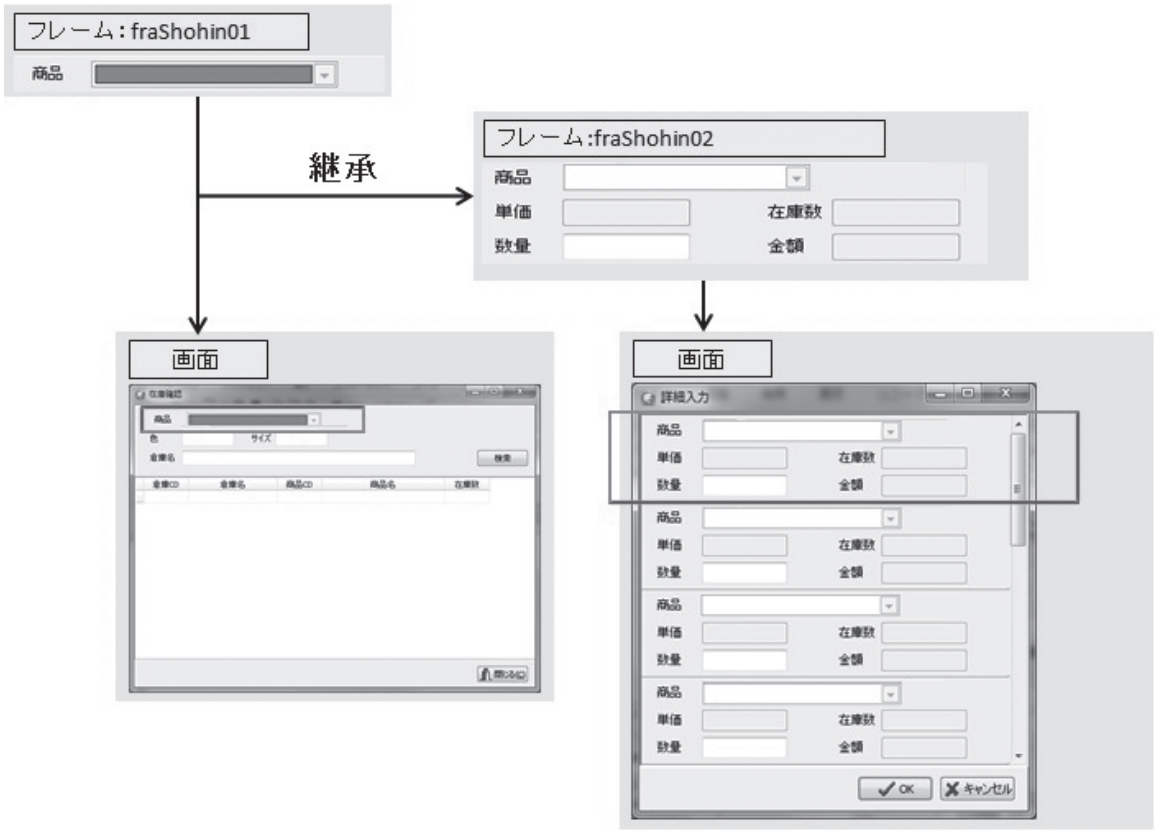
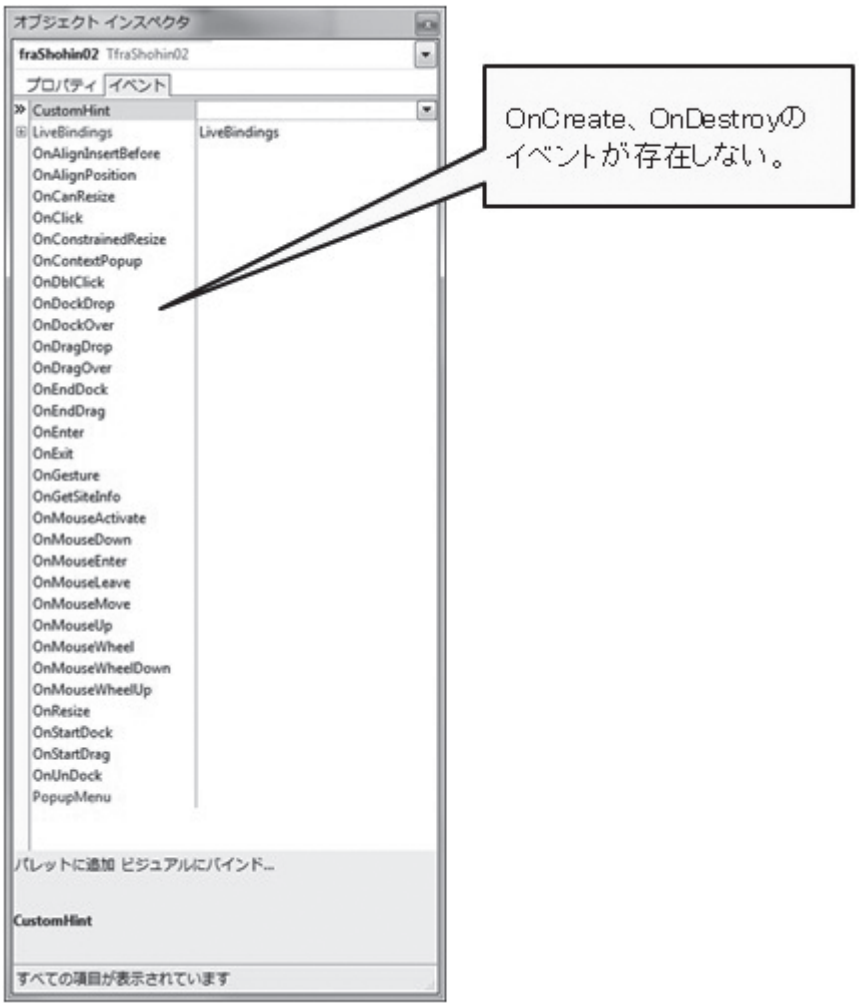

フレーム：fraShohin01
商品
継承
フレーム:fraShohin02
商品
単価
在庫数
数量
金額
画面
画面
詳細入力
商品
単価
在庫数
数量
金額
OK
キャンセル

図12


オブジェクト インスペクタ
fraShohin02 TfraShohin02
プロパティ
イベント
CustomHint
LiveBindings
OnAlignInsertBefore
OnAlignPosition
OnCanResize
OnClick
OnConstrainedResize
OnContextPopup
OnDblClick
OnDockDrop
OnDockOver
OnDragDrop
OnDragOver
OnEndDock
OnEndDrag
OnEnter
OnExit
OnGesture
OnGetSiteInfo
OnMouseActivate
OnMouseDown
OnMouseEnter
OnMouseLeave
OnMouseMove
OnMouseUp
OnMouseWheel
OnMouseWheelDown
OnMouseWheelUp
OnResize
OnStartDock
OnStartDrag
OnUnDock
PopupMenu
パレットに追加 ビジュアルにバインド...
CustomHint
すべての項目が表示されています
OnCreate、OnDestroyの
イベントが存在しない。

ソース1

```
  private
    { Private 宣言 }
  public
    { Public 宣言 }
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  end;

implementation

{$R *.dfm}

{ TfraShohin01 }

constructor TfraShohin01.Create(AOwner: TComponent);
begin
  inherited;

  // フレーム生成時の処理を記述

end;

destructor TfraShohin01.Destroy;
begin
  // フレーム解放時の処理を記述

  inherited;
end;

end.
```

publicにて、Create、Destroyを宣言

フレーム生成時の処理を記述

フレーム破棄時の処理を記述
inheritedにて、フレーム自体のメモリを解放

ソース2

```
  private
    { Private 宣言 }
    FTNKA: Currency;
    FKING: Currency;
    FZAIK: Currency;
    FSURY: Currency;
    procedure SetKING(const Value: Currency);
    procedure SetSURY(const Value: Currency);
    procedure SetTNKA(const Value: Currency);
    procedure SetZAIK(const Value: Currency);
    function GetKING: Currency;
    function GetSURY: Currency;
    function GetTNKA: Currency;
    function GetZAIK: Currency;
  public
    { Public 宣言 }
    property TNKA: Currency read GetTNKA write SetTNKA;     // 単価
    property ZAIK: Currency read GetZAIK write SetZAIK;     // 在庫数
    property SURY: Currency read GetSURY write SetSURY;     // 数量
    property KING: Currency read GetKING write SetKING;     // 金額

    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  end;
```

画面の配置項目をプロパティ定義する

プロパティへ値を渡す(例:金額)

```
{*********************************************************************************
 目的：金額取得処理
 引数：
 戻値：
*********************************************************************************}
function TfraShohin02.GetKING: Currency;
var
  sTemp: string;
begin
  sTemp  := StringReplace(edtKING.Text, ',', '', [rfReplaceAll]);
  Result := StrToCurrDef(sTemp, 0);
end;
```

KINGプロパティの値を取得するタイミングで処理が走る。

画面値のカンマ区切りを外しプロパティに値をセット

プロパティより値を受け取る(例:金額)

```
{*********************************************************************************
 目的：金額セット時処理
 引数：
 戻値：
*********************************************************************************}
procedure TfraShohin02.SetKING(const Value: Currency);
begin
  FKING := Value;

  edtKING.Text := FormatFloat('#,0', FKING);

  if FKING < 0 then
    edtKING.Font.Color := clRed;
end;
```

KINGプロパティに値がセットされたタイミングで処理が走る。

フォーマットを編集し、画面のEditにセット
値が0より小さい場合、文字を赤色に設定

図13


得意先
先方担当者
担当者
所属部署
パレットに追加(P)
編集(W)
コントロール(X)
ビジュアルにバインド(Y)...

図14


コンポーネント テンプレートの情報
コンポーネント名(C) TfraHeaderFieldTemplate
パレット ページ名(P) TechnicalReport
パレット アイコン：
変更(A)...
OK
キャンセル
ヘルプ
コンポーネント名・パレットページ名・
アイコンを任意で指定

図15


ツール パレット
検索
Tethering
IntraWeb Standard
IntraWeb Data
IntraWeb Control
IntraWeb Authentication
Cloud
AFControls
AFDBControls
AFFrame
SCD400IFS
Screen5250
SCD400 Data
SCD400 System
SCD400 DB
SCD400 Tools
TechnicalReport
TfraHeaderFieldTemplate
図14にて設定した情報で追加される

図16

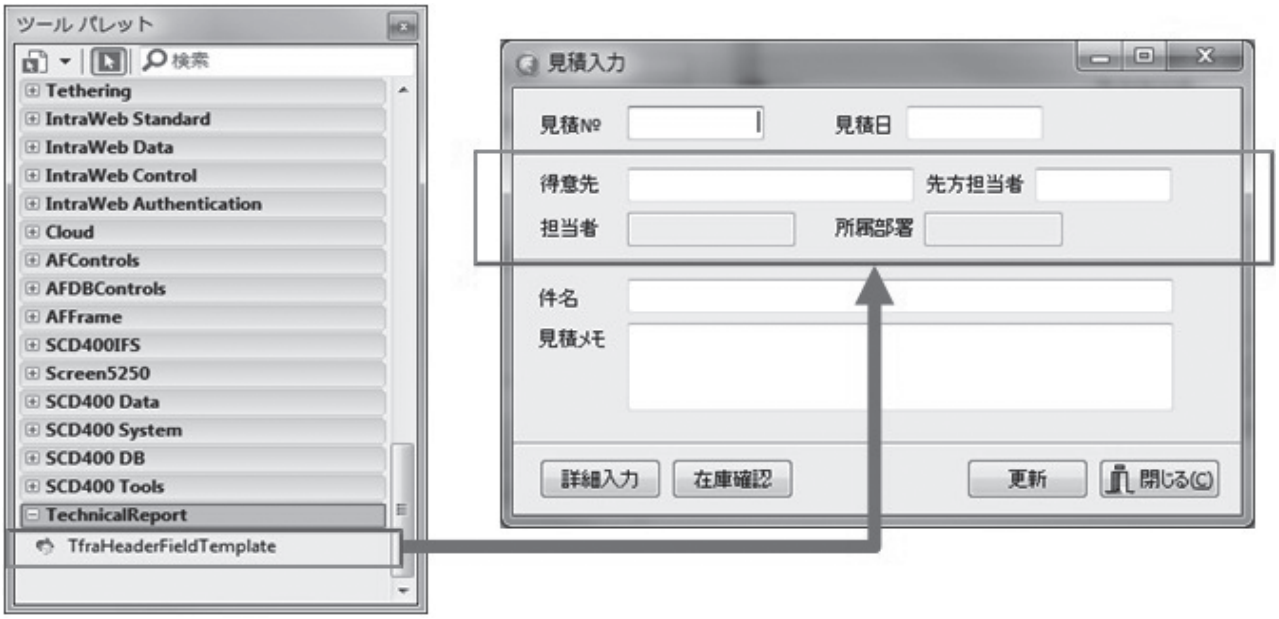

ツール パレット
検索
Tethering
IntraWeb Standard
IntraWeb Data
IntraWeb Control
IntraWeb Authentication
Cloud
AFControls
AFDBControls
AFFrame
SCD400IFS
Screen5250
SCD400 Data
SCD400 System
SCD400 DB
SCD400 Tools
TechnicalReport
TfraHeaderFieldTemplate
見積入力
見積№
見積日
得意先
先方担当者
担当者
所属部署
件名
見積メモ
詳細入力
在庫確認
更新
閉じる(C)

図17

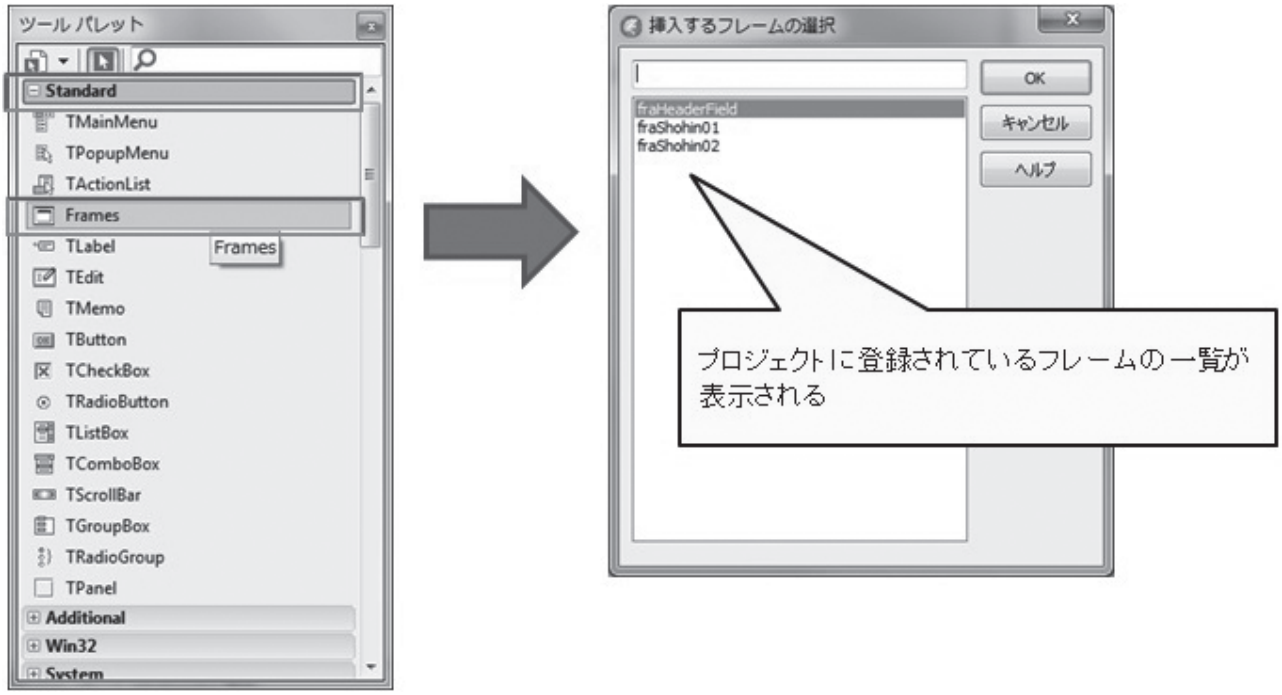

ツール パレット
Standard
TMainMenu
TPopupMenu
TActionList
Frames
TLabel
TEdit
TMemo
TButton
TCheckBox
TRadioButton
TListBox
TComboBox
TScrollBar
TGroupBox
TRadioGroup
TPanel
Additional
Win32
System
Frames
挿入するフレームの選択
fraHeaderField
fraShohin01
fraShohin02
OK
キャンセル
ヘルプ
プロジェクトに登録されているフレームの一覧が表示される

図18

ソース3

```
{*******************************************************************************
 目的：画面表示時処理
 引数：
 戻値：
*******************************************************************************}
procedure TfrmDetail.FormShow(Sender: TObject);
var
  i: Integer;
  fraShohin02: TfraShohin02;
  wCtl: TControl;
begin

  // フレーム初期化
  for i:= ScrollBox1.ControlCount-1 downto 0 do
  begin
    wCtl := ScrollBox1.Controls[i];

    if (wCtl is TfraShohin02) then
    begin
      FreeAndNil(wCtl);
    end;
  end;

  // フレーム生成
  for i := 0 to 10 do
  begin
    fraShohin02          := TfraShohin02.Create(Self);     // フレーム生成
    fraShohin02.Name     := 'fra' + FormatFloat('00', i);  // フレーム名
    fraShohin02.TabOrder := i - 1;                         // タブ順
    fraShohin02.Parent   := ScrollBox1;                    // 親コンポーネント
    fraShohin02.Top      := Height * (i);                  // フレームの位置設定
    fraShohin02.Align    := alTop;                         // フレームを上から整列

    fraShohin02.BlankAdd := True;                    // リストにブランク行追加
    fraShohin02.SetListItem;                         // リスト内容の設定

  end;

end;
```

ScrollBox上で生成されている
フレーム(TfraShohin02)の破棄

ScrollBox上にフレーム
(TfraShohin02)を生成

ソース4

更新ボタンの
OnClickイベント

①フレームに配置しているコンポーネントを
直接指定する場合

```
{*************************************************************************************
 目的：更新ボタン押下時処理
 引数：
 戻値：
*************************************************************************************}
procedure TfrmEstimates.bbtnUpdateClick(Sender: TObject);
begin

  SQLQuery1.ParamByName('TKSK').AsString := fraHeaderField1.edtTKSK.Text;   // 得意先
  SQLQuery1.ParamByName('SPTN').AsString := fraHeaderField1.edtSPTN.Text;   // 先方担当者
  SQLQuery1.ParamByName('TNTO').AsString := fraHeaderField1.edtTNTO.Text;   // 担当者
  SQLQuery1.ParamByName('BUSH').AsString := fraHeaderField1.edtBUSH.Text;   // 所属部署

  SQLQuery1.ExecSQL;

end;
```

フレーム名＋コンポーネント名

②フレームのプロパティ定義名を
指定する場合

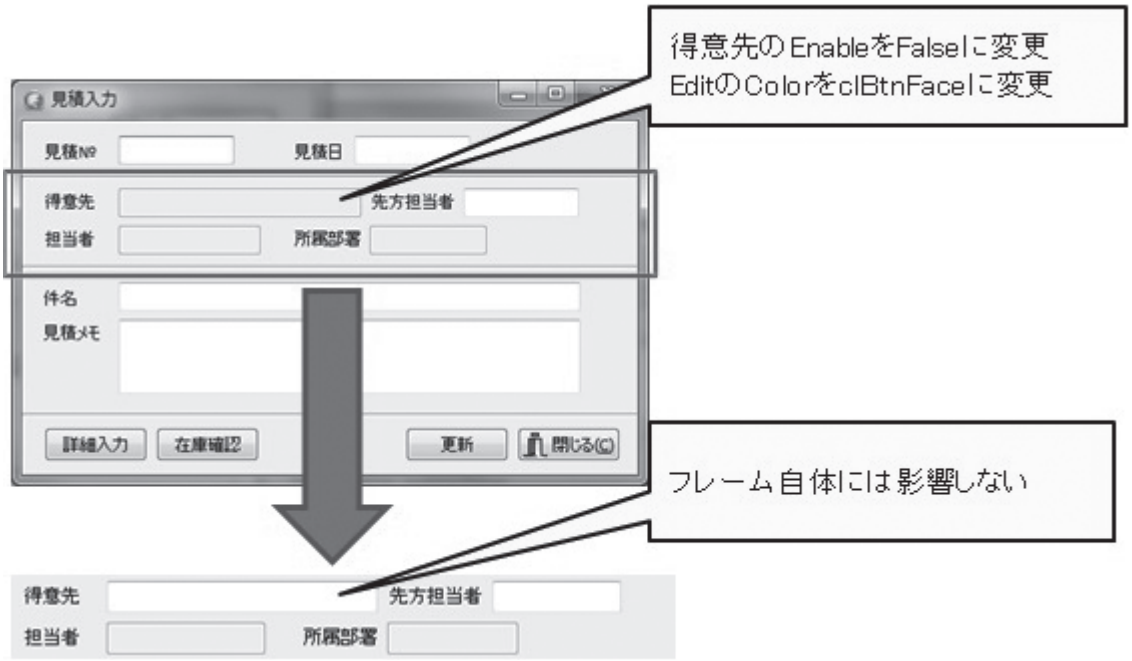
```
{*************************************************************************************
 目的：更新ボタン押下時処理
 引数：
 戻値：
*************************************************************************************}
procedure TfrmEstimates.bbtnUpdateClick(Sender: TObject);
begin

  SQLQuery1.ParamByName('TKSK').AsString := fraHeaderField1.TKSK;   // 得意先
  SQLQuery1.ParamByName('SPTN').AsString := fraHeaderField1.SPTN;   // 先方担当者
  SQLQuery1.ParamByName('TNTO').AsString := fraHeaderField1.TNTO;   // 担当者
  SQLQuery1.ParamByName('BUSH').AsString := fraHeaderField1.BUSH;   // 所属部署

  SQLQuery1.ExecSQL;

end;
```

フレーム名＋プロパティ定義名

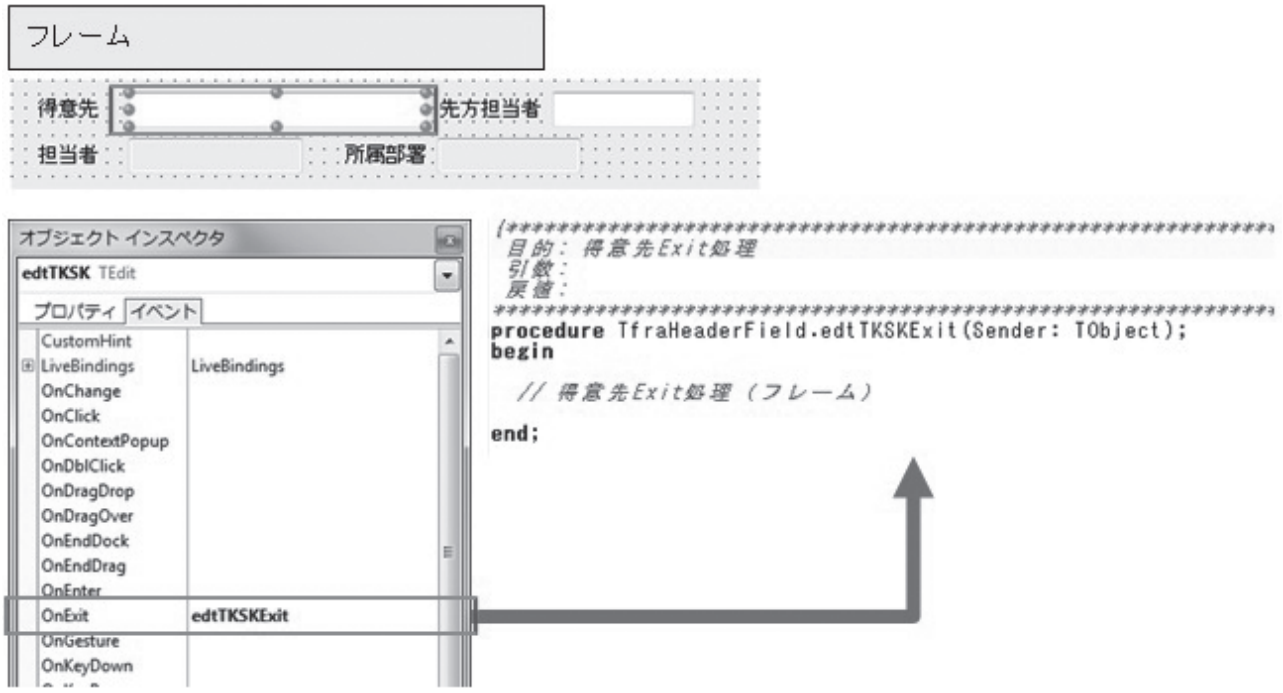
図19

削除しようとしたコンポーネント名

図20


得意先のEnableをFalseに変更
EditのColorをclBtnFaceに変更
見積入力
見積№
見積日
得意先
先方担当者
担当者
所属部署
件名
見積メモ
詳細入力
在庫確認
更新
閉じる(C)
フレーム自体には影響しない
得意先
先方担当者
担当者
所属部署

ソース5


フレーム
得意先
先方担当者
担当者
所属部署
オブジェクト インスペクタ
edtTKSK TEdit
プロパティ
イベント
CustomHint
LiveBindings
LiveBindings
OnChange
OnClick
OnContextPopup
OnDblClick
OnDragDrop
OnDragOver
OnEndDock
OnEndDrag
OnEnter
OnExit
edtTKSKExit
OnGesture
OnKeyDown
{***
目的：得意先Exit処理
引数：
戻値：
***}
procedure TfraHeaderField.edtTKSKExit(Sender: TObject);
begin
// 得意先Exit処理（フレーム）
end;

ソース6

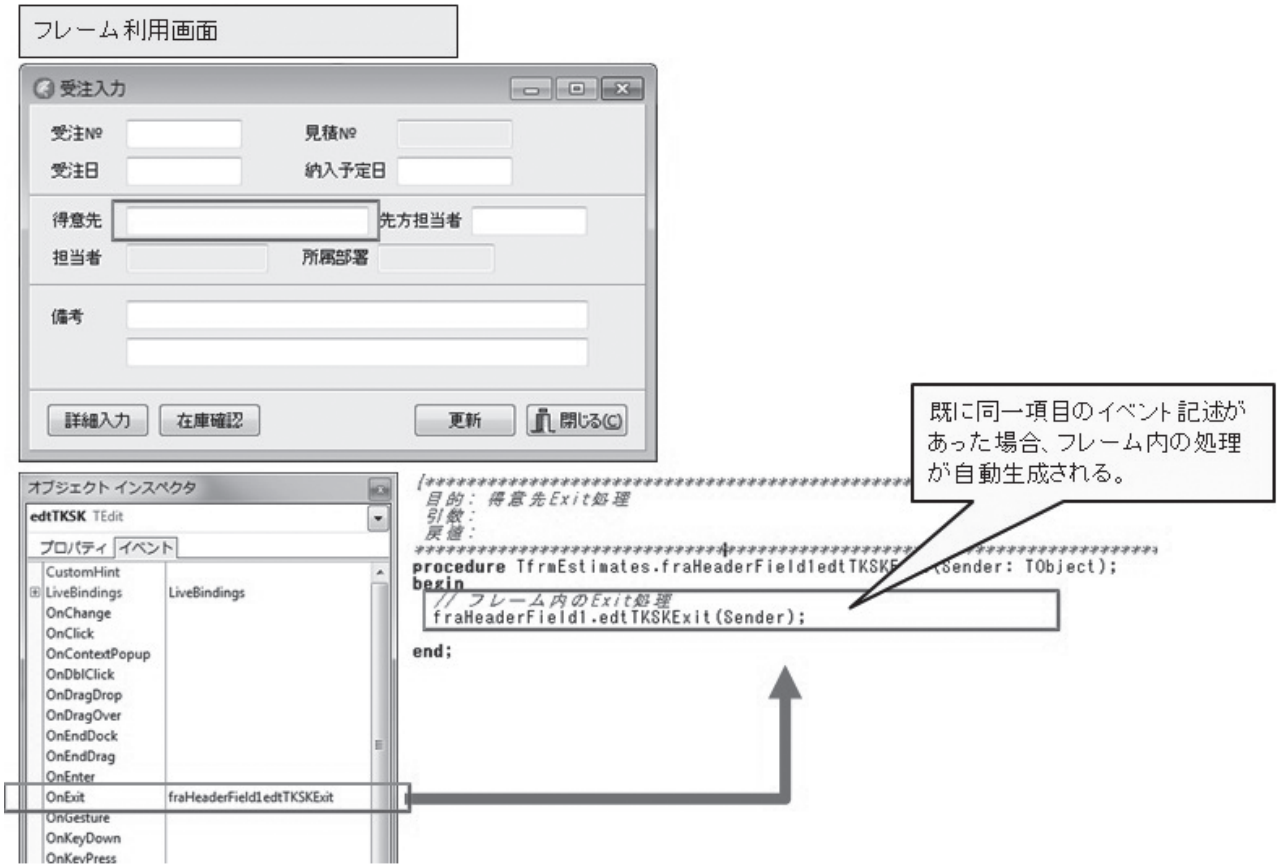

フレーム利用画面
受注入力
受注№
見積№
受注日
納入予定日
得意先
先方担当者
担当者
所属部署
備考
詳細入力
在庫確認
更新
閉じる(C)
オブジェクト インスペクタ
edtTKSK TEdit
プロパティ
イベント
CustomHint
LiveBindings
OnChange
OnClick
OnContextPopup
OnDblClick
OnDragDrop
OnDragOver
OnEndDock
OnEndDrag
OnEnter
OnExit
OnGesture
OnKeyDown
OnKeyPress
fraHeaderField1edtTKSKExit
目的：得意先Exit処理
引数：
戻値：
procedure TfrmEstimates.fraHeaderField1edtTKSK
(Sender: TObject);
begin
// フレーム内のExit処理
fraHeaderField1.edtTKSKExit(Sender);
end;
既に同一項目のイベント記述があった場合、フレーム内の処理が自動生成される。

図21

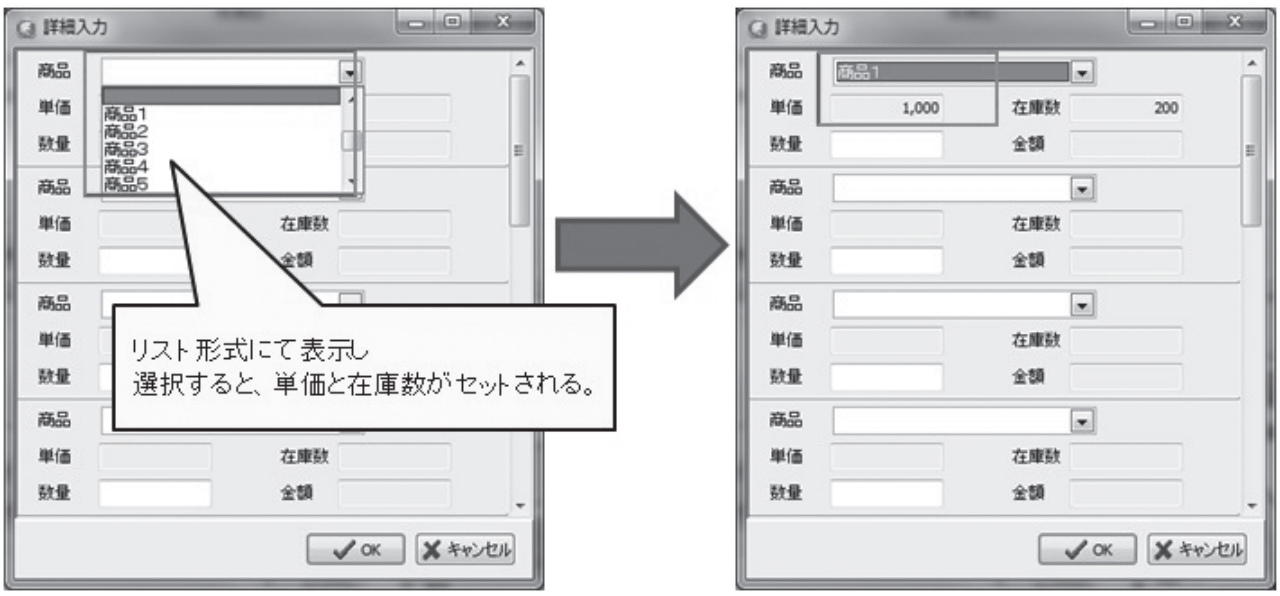

詳細入力
商品
単価
数量
在庫数
金額
商品1
商品2
商品3
商品4
商品5
OK
キャンセル
リスト形式にて表示し
選択すると、単価と在庫数がセットされる。
1,000
200

図22

商品マスタレイアウト

DSPFMT　　レコード設計書　　日付 15/08/17
時刻 18:57:46

物理ファイル	MAELIB/MGSHOHPF	様式名	MGSHOH	レコード長	6
様式記述	商品マスタサンプル				

5= 詳細

選択	項目名	桁数	属性	キー順	開始	終了	テキスト記述／欄見出し
_	MGSHCD	4	A		1	4	商品 CD
_	MGSHNM	48	O		5	52	商品名
_	MGTANK	6 0	S		53	58	単価
_	MGZAIK	6 0	S		59	64	在庫数

図23

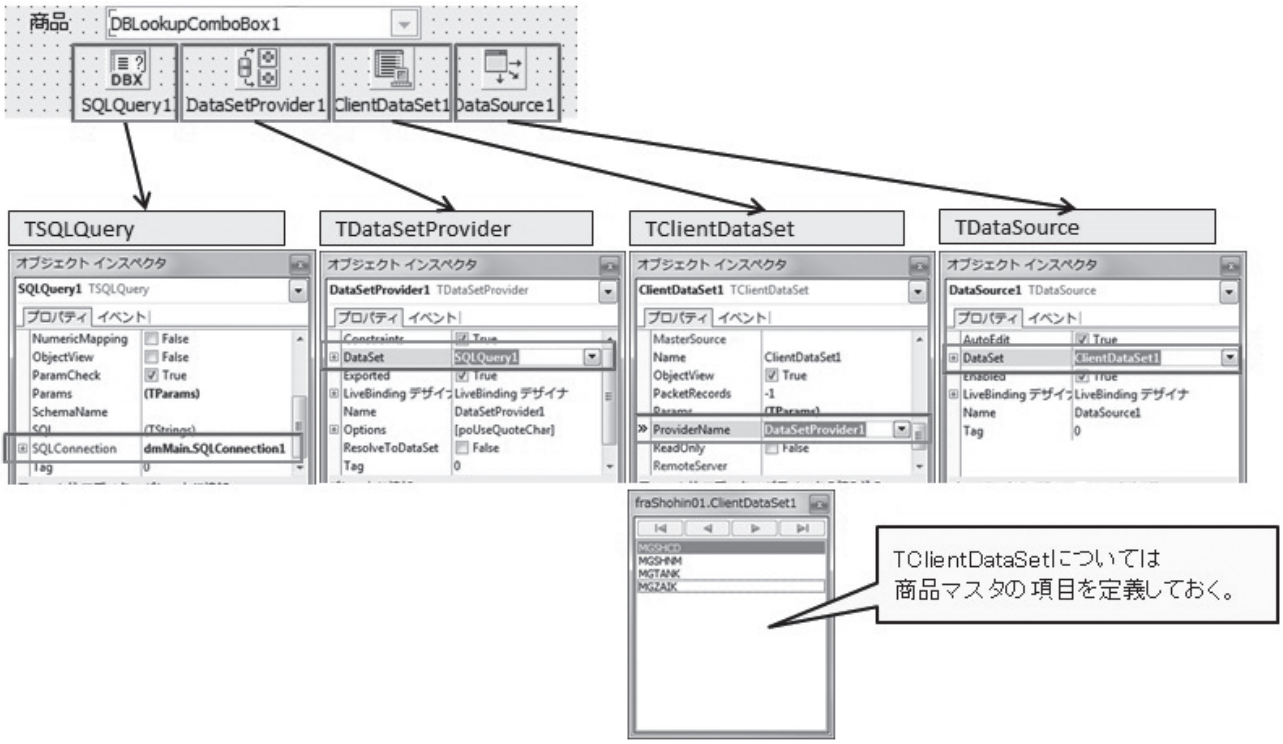


図24

TDBLookUpComboBoxのプロパティ設定

プロパティ名	設定値
ListSource	DataSource1
KeyField	MGSHCD
ListField	MGSHNM

ソース7

fraShohin01フレーム

```
private
  { Private 宣言 }
  FSHNM: String;
  FSHCD: String;
  FBlankAdd: Boolean;
  procedure SetSHCD(const Value: String);
  procedure SetSHNM(const Value: String);
  function GetSHCD: String;
  function GetSHNM: String;
public
  { Public 宣言 }
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;

  property BlankAdd: Boolean read FBlankAdd write FBlankAdd;  // ブランク行
  property SHCD: String read GetSHCD write SetSHCD;           // 商品コード
  property SHNM: String read GetSHNM write SetSHNM;           // 商品名

  procedure SetListItem;
end;
```

ソース8

fraShohin01フレーム

```
{*****************************************************************************
 目的：リスト内容設定処理
 引数：
 戻値：
*****************************************************************************}
procedure TfraShohin01.SetListItem;
begin

  // データ取得SQL設定
  SQLQuery1.SQL.Text := ' SELECT * FROM MGSHOHPF ';

  // データセットのClose
  ClientDataSet1.Close;

  // データセットのOpen
  ClientDataSet1.Open;

  // 対象データがない場合
  if ClientDataSet1.IsEmpty then
  begin
    // データセットのClose
    ClientDataSet1.Close;
    Exit;
  end;

  // BlankAddプロパティがTrueの場合
  if (FBlankAdd) then
  begin
    try
      // ブランク行を追加
      ClientDataSet1.First;
      ClientDataSet1.Insert;

      ClientDataSet1.FieldByName('MGSHCD').AsString  := '  ';   // 商品コード
      ClientDataSet1.FieldByName('MGSHNM').AsString  := '  ';   // 商品名
      ClientDataSet1.FieldByName('MGTANK').AsInteger := 0;      // 単価
      ClientDataSet1.FieldByName('MGZAIK').AsInteger := 0;      // 在庫

      ClientDataSet1.Post;
    except
      ClientDataSet1.Cancel;
    end;
  end;

  // 初期値の設定
  if FBlankAdd then
  begin
    // 先頭行
    DBLookupComboBox1.KeyValue :=
      ClientDataSet1.FieldByName('MGSHCD').AsString;
  end
  else
  begin
    DBLookupComboBox1.KeyValue := '';
  end;
```

ソース9

fraShohin02フレーム


```
{*************************************************************************
 目的：単価セット時処理
 引数：
 戻値：
*************************************************************************}
procedure TfraShohin02.SetTNKA(const Value: Currency);
begin
  FTNKA := Value;

  edtTNKA.Text := FormatFloat('#,0', FTNKA);
end;

{*************************************************************************
 目的：在庫数セット時処理
 引数：
 戻値：
*************************************************************************}
procedure TfraShohin02.SetZAIK(const Value: Currency);
begin
  FZAIK := Value;

  edtZAIK.Text := FormatFloat('#,0', FZAIK);
end;

{*************************************************************************
 目的：商品選択時処理
 引数：
 戻値：
*************************************************************************}
procedure TfraShohin02.DBLookupComboBox1Click(Sender: TO
begin
  inherited;

  TNKA := ClientDataSet1.FieldByName('MGTANK').AsInteger;  // 単価
  ZAIK := ClientDataSet1.FieldByName('MGZAIK').AsInteger;  // 在庫数

end;
```

単価、在庫数のプロパティ定義にデータセットの値をセット

ソース10

フレーム利用画面

```
{*************************************************************************
 目的：画面表示時処理
 引数：
 戻値：
*************************************************************************}
procedure TfrmDetail.FormShow(Sender: TObject);
var
  i: Integer;
  fraShohin02: TfraShohin02;
  wCtl: TControl;
begin

  // フレーム初期化
  for i:= ScrollBox1.ControlCount-1 downto 0 do
  begin
    wCtl := ScrollBox1.Controls[i];

    if (wCtl is TfraShohin02) then
    begin
      FreeAndNil(wCtl);
    end;
  end;

  // フレーム生成
  for i := 0 to 10 do
  begin
    fraShohin02           := TfraShohin
    fraShohin02.Name      := 'fra' + FormatFloa                    // フレーム名
    fraShohin02.TabOrder  := i - 1;                                // タブ順
    fraShohin02.Parent    := ScrollBox1;                           // 親コンポーネント
    fraShohin02.Top       := Height * (i);                         // フレーム位置
    fraShohin02.Align     := alTop;

    fraShohin02.BlankAdd := True;                                  // リストにブランク行追加
    fraShohin02.SetListItem;                                       // リスト内容の設定
  end;

end;
```

フレーム利用画面は
SetListItemを呼び出すだけでリスト内容を設定できる。
BlankAddプロパティをTrueに設定すると、ブランク行を追加。