# Magic eDeveloper V10



Magic eDeveloper V10 コーディングサンプル Version 2 (コンポーネント編)



本書および添付サンプル(以下、本製品)の著作権は、マジックソフトウェアジャパン株式会社(MSJ)にあります。MSJ の書面による事前の許可なしでは、いかなる条件下でも、本製品 のいかなる部分も、電子的、機械的、撮影、録音、その他のいかなる手段によっても、コピー、検索システムへの記憶、電送を行うことはできません。

本製品の内容につきましては、万全を期して作成していますが、万一誤りや不正確な記述があったとしても、MSE (Magic Software Enterprises Ltd.) および MSJ はいかなる責任、債務も負いません。本製品を使用した結果、または使用不可能な結果生じた間接的、偶発的、副次的な損害(営利損失、業務中断、業務情報の損失などの損害も含む)に関し、事前に損害の可能性が勧告されていた場合であっても、MSE および MSJ、その管理者、役員、従業員、代理人は、いかなる場合にも一切責任を負いません。 MSE および MSJ は、本製品の商業価値や特定の用途に対する適合性の保証を含め、明示的あるいは黙示的な保証は一切していません。

本製品に記載の内容は、将来予告なしに変更することがあります。

サードパーティ各社商標の引用は、MSE および MSJ の製品に対する互換性に関しての情報提供のみを目的としてなされるものです。一般に、会社名、製品名は各社の商標または登録商標です。

本製品において、説明のためにサンプルとして引用されている会社名、製品名、住所、人物は、特に断り書きのないかぎり、すべて架空のものであり、実在のものについて言及するものではありません。

初版 2007年9月19日

マジックソフトウェア・ジャパン株式会社

# 目次

3コンポーネント	4
3.1 コンポーネントの構成	4
3.2 コンポーネントファイルの配置	6
3.2.1 ディレクトリ構造	6
3.2.2 プロジェクトソースファイル	6
3.2.3 コンポーネントキャビネットファイル	7
3.2.4 コンポーネントファイルの指定	7
3.3 共通コンポーネントの内容	10
3.3.1 ユーザ定義イベント	10
3.3.2 モデルリポジトリ	11
3.3.3 関連プログラム	14
3.3.4 ユーザ定義関数	14
uMsgBoxFull	1.5
uYesNoDlg	16
uTaskModeString	16
3.4 データコンポーネントの内容	17
3.4.1 モデルリポジトリ	17
3.4.2 データリポジトリ	17
3.4.3 プログラムリポジトリ	20
3.5 受注コンポーネントの内容	23
3.5.1 モデルリポジトリ	23
3.5.2 データリポジトリ	23
3.5.3 プログラムリポジトリ	23
3.6 ホストコンポーネントの内容	27
3.6.1 モデルリポジトリ	27
3.6.2 データリポジトリ	27
3.6.3 プログラムリポジトリ	28
3.6.4 アプリケーション特性	29
3.7 モジュール定義	31
3.7.1 モジュールの表示	31
3.7.2 モジュールのネスト	31
3.7.3 モジュールの用途	32
3.7.4 モジュールの追加	32

## 3 コンポーネント

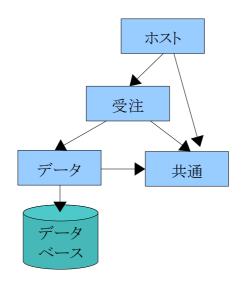
本書添付のサンプルアプリケーションでは、アプリケーションを機能的に分割してコンポーネント化しています。本章では各コンポーネントについて簡単に説明します。

## 3.1 コンポーネントの構成

サンプルアプリケーションは次のようなコンポーネントに分割されています。

コンポーネント	プロジェクト名	目的
共通コンポーネント	ED_COMMON	どのアプリケーションでも非常によく使われる便利なユーザイベント、モデル、ユーザ定義関数などを提供する。
データコンポーネント	ED_DATA	アプリケーションに固有なデータ定義(モデル、データソース)、およびそれに関連するプログラム(マスタメンテナンス、選択プログラム)を提供する。
受注コンポーネント	ED_JUCHU	受注業務に関連するプログラムを提供する。
ホストコンポーネント	ED_HOST	コンポーネント全体を統括し、ユーザメニューを提供する。

各コンポーネントの依存関係は下図の通りです。



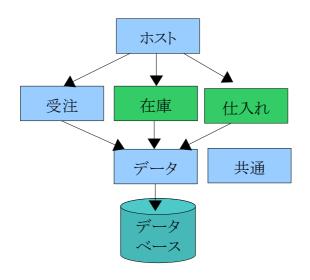
即ち、ホストコンポーネントは受注コンポーネントに依存し、受注コンポーネントはデータコンポーネントに依存しています。また、すべてのコンポーネントは共通コンポーネントに依存しています。

共通コンポーネントは、アプリケーションでよく使われるオブジェクトのうち、特定のアプリケーションに依存しないものを集めたものです。このコンポーネントはどのアプリケーションにも応用できることになり、開発者にとっての財産となります。

データコンポーネントは、データベースに依存したデータ定義を行うものです。このコンポーネントは、受注コンポーネントのほかに、仕入れ、在庫、その他具体的な業務を行うためのコンポーネント間で共有できるようになります。

ホストコンポーネントと受注コンポーネントとをあえて分けている理由は、拡張性を考慮してのことです。例えば、このアプリケーションに在庫および仕入れ業務のためのコンポーネントを追加するとなると、次の図のような構

造とすることができます。図中、緑色のコンポーネントが追加となったコンポーネントです。

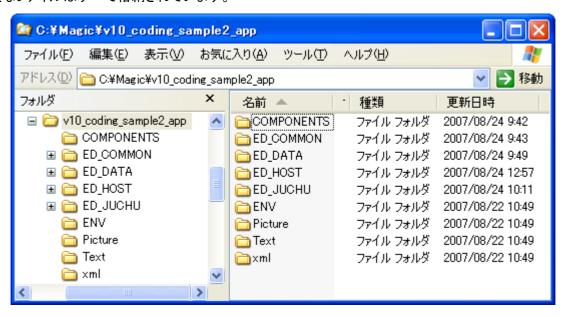


在庫および仕入れコンポーネントの追加方法 (共通コンポーネントの依存関係は省略してあります)

このように、受注コンポーネントの上にひとつホストコンポーネントをかぶせた形にすると、拡張性が高くなり、 パッケージでオプションコンポーネントとすることも容易になります。

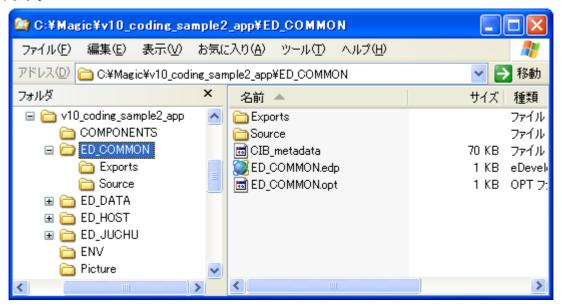
## 3.2.1 ディレクトリ構造

サンプルアプリケーションである ZIP ファイルを解凍すると、下図のようなディレクトリ構造が現れます。この中に必要なファイルはすべて格納されています。



## 3.2.2 プロジェクトソースファイル

各コンポーネントのプロジェクトは、それぞれのプロジェクト名と同じ名前のサブディレクトリに格納されています。 例えば、共通コンポーネント ED\_COMMON のプロジェクトソースファイルは、サブディレクトリ ED\_COMMON の下にあります。



#### 3.2.3 コンポーネントキャビネットファイル

各コンポーネントをキャビネットファイル化した ECF ファイルは、COMPONENTS ディレクトリにまとめて格納されています。



キャビネットファイルをプロジェクトファイルとは異なるディレクトリに格納してあるのは、単に管理上の容易さのためです。共通コンポーネントなど、独立に別のアプリケーションでも使われる可能性があるコンポーネントの場合に、ソースの位置とキャビネットファイルの位置とを分けたり、あるいは場合によっては、自社開発でないコンポーネントなどプロジェクトソースファイル自体持っていない場合もあることも考慮すると、キャビネットファイルおよびコンポーネントインターフェース定義ファイル(ECIファイル)を決まった共通のディレクトリに入れておくと便利です。

## 3.2.4 コンポーネントファイルの指定

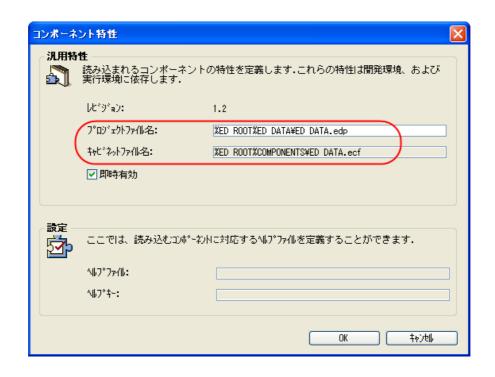
各プロジェクトが依存するコンポーネントはコンポーネントリポジトリに登録しますが、ここでコンポーネント特性 として、プロジェクトファイル名とキャビネットファイル名が設定されます。実行環境でアプリケーションをインストールする位置に自由度を与えるためには、ここには論理名を使って定義するのがベストです。

例えば、次の図は、ED\_JUCHU 中のコンポーネントリポジトリでの ED\_DATA コンポーネントのコンポーネント特性です。ここでは ED\_ROOT という論理名を使ってプロジェクトファイル名とキャビネットファイル名を定義しています。

このようにしておけば、論理名 ED\_ROOT (アプリケーションのルートディレクトリ名)が正しく設定されていれば、コンポーネントのキャビネットファイルもプロジェクトファイルも正しく読み込まれるようになります。

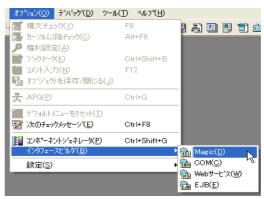


ここではプロジェクトファイル名とキャビネットファイル名だけを使っていますが、コンポーネントのヘルプファイルを使っている場合には、**ヘルプファイル** 欄も論理名 ED\_ROOT を使って定義しておくべきでしょう。



ところで、コンポーネント特性のプロジェクト特性のこれらのファイル名は、Magic コンポーネントインターフェースビルダを使ってコンポーネントインターフェイス定義ファイル(ECIファイル)を作成するときに、**コンポーネントとプロジェクトの設定**の画面で指定します(下図)。

Magic コンポーネントインターフェースビルダを開くには、プロジェクト ED\_DATA を開いている状態で、オプション(O)  $\rightarrow$  インターフェースビルダ(B)  $\rightarrow$  Magic (D) を選びます。



「ようこそ」画面が出るので、「次へ(N)」を押します。



「有効なコンポーネント」画面で、「新規(E)」ボタンを押します。

すでにコンポーネント定義を行っている場合には一覧に ED\_DATA があるはずですので、それを選んで「次へ(N)」を押します。



「コンポーネントとプロジェクトの設定」画面で、コンポーネント名を ED\_DATA (任意の名前で構いませんが、管理上の便利さのため、プロジェクト名と同じにしておきます) とし、プロジェクトファイル名、キャビネットファイル名を論理名 ED\_ROOT を使って指定します。



ここには、デフォルトではフルパス名でプロジェクトファイル名などが設定されていますが、論理名を使った名前に書き直してください。

ここで設定しておけば、その内容が ECI ファイルに反映され、それを読み込めばコンポーネント特性にも反映されるようになります。

## 3.3 共通コンポーネントの内容

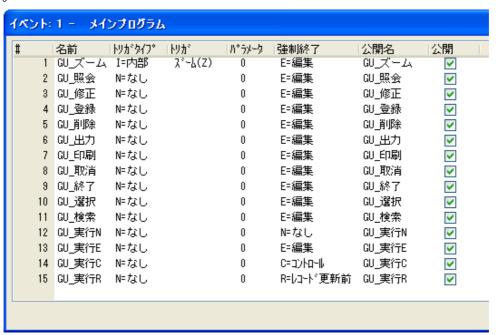
共通コンポーネントは、アプリケーションでよく使われるオブジェクトのうち、特定のアプリケーションに依存しないものを集めたものです。これには次にようなものがあります。

- ユーザ定義イベント
- 共通モデル
- 関連プログラム
- ユーザ定義関数

#### 3.3.1 ユーザ定義イベント

共通コンポーネントでは、どのアプリケーションでもよく使われるような、ユーザ定義イベントが定義されています。

メインプログラムを開いて、ユーザイベントテーブルを開いてください。下図のようなユーザ定義イベントが定義されています。



これらのユーザ定義イベントの利用目的は、だいたい次の表のような利用法を想定しています。

種類	イベント	目的
ズーム対応	GU_ズーム	ズームに対応したイベントハンドラを作成する時に、入力途中のデータを項目に反映させるため、強制終了が「E=編集」である必要があるので、定義している。
タスクモード 変更ボタン用	GU_照会、 GU_修正、 GU_登録	「照会」「修正」「登録」ボタンを押した場合に発生させるイベント。
一般機能ボタン用	GU_削除、 GU_出力、 GUI_印刷 等/	「削除」「出力」「印刷」等々、よく使うボタンを押した場合に発行させるイベント。

	GU_実行 N、 GU_実行 E、	汎用の「実行」ボタンを押した場合に発行させるイベント。4種類定義されているのは、「強制終了」の設定が異なるものを定義しているため。
	GU_実行 C、 GU 実行 R	

コンポーネントとして外部から利用してもらうために、それぞれのイベントには公開名が定義されており、公開フラグをオンにしてあります。



#### 注意:

- これらのユーザ定義イベントは、それ自体で上記のような機能を持っているものではありません。必ず、 これらのイベントに対応したイベントハンドラをプログラムで作成してやる必要があります。
- 強制終了パラメータについては、ここでは説明を省略します。

## 3.3.2 モデルリポジトリ

共通コンポーネントのモデルリポジトリには、Magic のアプリケーションならばよく使うような、汎用のモデルが定義されています。(下図)



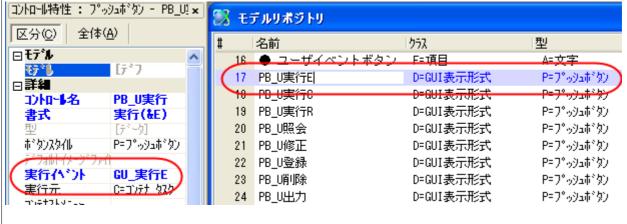
これには次のようなものがあります。

種類	Į	モデル例	目的
内部 ボタ	3イベント用 ン	PB_取消、 PB_終了、 PB_選択 等	プッシュボタンのモデルであり、内部イベント(「キャンセル(C)」、「クローズ(C)」などが <b>実行イベント</b> として設定されています。このモデルを利用すれば、プログラムロジックを書かずとも、フォーム上に貼り付けるだけで、取消、終了などの機能を持ったボタンを実現できます。
/Fil.	/BI DD 467 15 15 11		

#### 例: PB\_終了 ボタンモデル

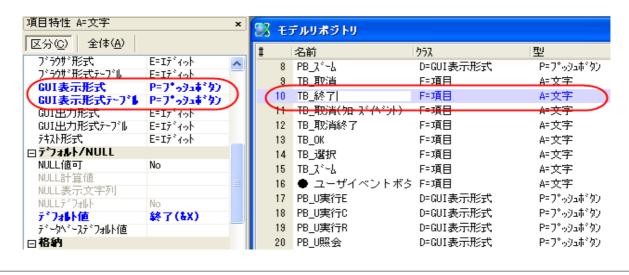


種類	モデル例	目的		
ユーザイベント用 ボタン	PB_U 照会、 PB_U 印刷、 PB_U 実行 E 等	ザ定義イベントが <b>実行</b>	イベント として設定	ーネントで定義されたユー されています。利用する場 るイベントハンドラを作成す
例: PB_U 実行 E ボ	タンモデル			
コントローは特性: プッシュ		モデルリポジトリ		
区分(0) 全体(A)	#	名前	クラス	型



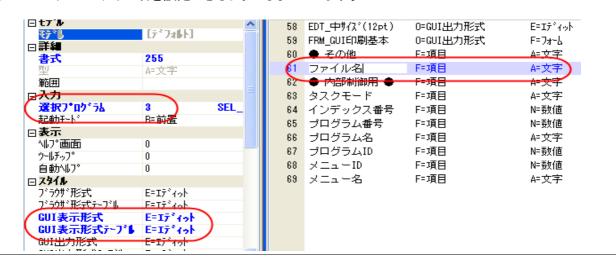
種類	モデル例	目的
ボタン項目	TB_終了、 TB_U 照会 等	上の内部イベント用ボタンあるいはユーザイベント用ボタンを GUI 表示形式のスタイルとして参照している、文字型の項目モデルです。 プログラムでは、データビューでこのモデルを参照して変数を定義し、その項目をフォームエディッタ上に貼り付けることにより、各種の機能を持ったボタンを配置することができます。変数項目として定義しているので、カーソルがパークします。

#### 例: TB\_終了 ボタン項目モデル



種類	モデル例	目的
	ファイル名、 プログラム番号	どんなアプリケーションにでもよく使われるようなデータ項目を定義します。

例: ファイル名 項目モデル。ファイル名のモデルでは、ズームしたらファイルダイアログが表示されるよう、選択プログラムが設定してあります。また、ドラッグアンドドロップも有効にしてあり、エクスプローラなどからドラッグアンドドロップでファイル名を設定できるようにもなっています。

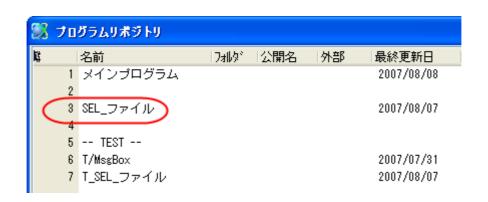


種類	モデル例	目的				
その他の GUI 部品	TBL_交互色、 FRM_基本、 LBL_ラベル、 EDT_表示専用	ム、ラ/ るコント た。開発 設定忘	ベル(テキ -ロール) 発時には れなどが	・フェースに一貫性を ・ストコントロール)、コ について、いくつかの はモデルを選択するた がなくなるし、保守時 ・一ザインターフェース	ニディットコントロ- パターンをモデル ごけでよいので設 こはモデルの変	ールなど多用され レとして用意しまし 定が容易になり 更だけでアプリケー
例: TB_交互色: 交コントロール特性: テーブル			ロールを	を定義します。		
	- IDL	x	□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	テルリポジトリ		
区分(C) 全体(A)			#	名前	クラス	型
□表示		^	42	TB_U検索	F=項目	A=文字
色 テープ・10色の指定	」 T=テーブルに依存		43	● GUIコントロール類	F=項目	A=文字
交互表示色	101		44	TBL_交互色	D=GUI表示形式	T=デーフ*ル
ツールチッフ。	0		45	TBL_交互色(位置100)	D=GUI表示形式	T=テープル
<b>አ</b> ቃረቡ	W=Windows		46	FRM_基本	D=GUI表示形式	F=7*~4
境界のスタイル	H太線		47	FRM_選択画面フォーム	D=GUI表示形式	F=7*~L
スクロールバドー	Yes		48	FRM_スクリーン入力フォーム	D=GUI表示形式	F=7*~L
区切り	No		49	FRM_MDI7	D=GUI表示形式	F=フォーム
ハイライト行うのスタイル	C=背景とコントロール		50	LBL_ <del>5</del> \*\\	D=GUI表示形式	S=スタティック
<b>ハイライト行の色</b> 下辺の間隔	<b>3</b> №なし		51	EDT_表示専用	D=GUI表示形式	E=エディット
1 XZV/IBJNG	14 -80					

## 3.3.3 関連プログラム

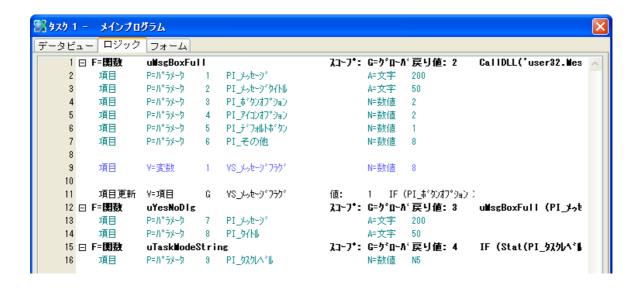
サンプルアプリケーションの共通コンポーネントでは、関連するプログラムとして、ファイル名モデルをズームした場合に、ファイルダイアログが表示されるよう、選択プログラムを用意しました。

その他のプログラムはテスト用であり、コンポーネントの外には公開されていません。



## 3.3.4 ユーザ定義関数

共通コンポーネントには次のユーザ定義関数が定義されています。



#### uMsgBoxFull

名前	uMsgBoxFull		
目的	メッセージダイアログボック	なる表示します。	
パラメータ	PI_メッセージ	(文字型) メッセージ本文	
	PI_メッセージタイトル	(文字型) タイトルバーに表示する文字列	
	PI_ボタンオプション	(数値)表示するボタンの下図やラベルを指示するフラグ	
	PI_アイコンオプション	(数値)表示するアイコンの種類を指示するフラグ	
	PI_デフォルトボタン	(数値) デフォルトボタンを指示するフラグ	
	PI_その他	(数値) その他のオプションを指示するフラグ	
戻り値	(数値) 押されたボタンに対応する番号		
補足	Windows の MessageBox APIを利用しています。オプションフラグは Windows API の値に 準拠して定義されています。		

ここで、各パラメータの値の意味は、以下の通りです。

パラメータ	値	意味
PI_ボタンオプション	0	ок
	1	OK/キャンセル
	2	中止/再試行/無視
	3	はい/いいえ/キャンセル
	4	はい/いいえ
	5	再試行/キャンセル
	その他	(0と同じ)
PI_アイコンオプション	0	アイコンなし

	1	中止 (× マーク)
	2	疑問 (? マーク)
	3	警告(! マーク)
	4	情報 (i マーク)
	その他	0 と同じ
PI_デフォルトボタン	0	左から1番目のボタン
	1	左から2番目のボタン
	2	左から3番目のボタン
戻り値	1	OK ボタンが押された
	2	キャンセルボタンが押された。あるいは、ウィンドウの[X] ボタンで閉じられた。
	3	中止 ボタンが押された。
	4	再試行 ボタンが押された
	5	無視 ボタンが押された
	6	はいボタンが押された
	7	いいえ ボタンが押された

## uYesNoDlg

名前	uYesNoDlg		
目的	はい/いいえのメッセージダイアログボックスを表示します。		
パラメータ	PI_メッセージ (文字型)メッセージ本文		
	PI_メッセージタイトル	(文字型) タイトルバーに表示する文字列	
戻り値	(論理型) はい が押された場合には True、その他の場合には False が返ります。		
補足	uMsgBoxFull の簡略版です。		

## uTaskModeString

名前	uTaskModeString
目的	現在のタスクのタスクモードを文字列で返します。
パラメータ	(なし)
戻り値	(文字型)タスクのモードに応じ、「登録」「照会」「修正」のいずれかが返ります。

## 3.4 データコンポーネントの内容

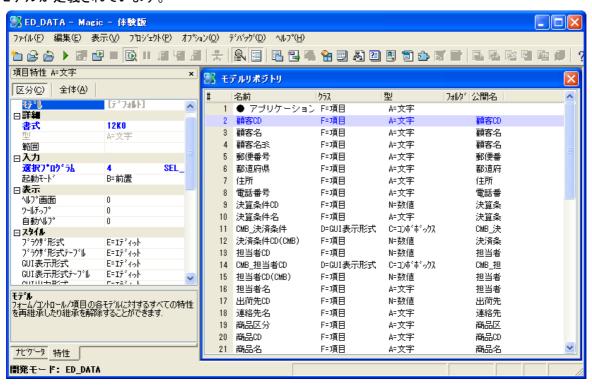
データコンポーネント ED\_DATA には、サンプルアプリケーションに固有なデータの定義と、それに関連するプログラムが定義されています。

## 3.4.1 モデルリポジトリ

モデルリポジトリには、アプリケーションに固有なデータ項目がモデルとして定義されています。これには、顧客コード、顧客名、商品コード等があります。

また、顧客コードや商品コードなどには、選択プログラムが設定されています。

さらに、選択肢が比較的少ないデータ項目(担当者コード、単位名など)には、データコントロールとしてコンボボックスモデルが定義されています。

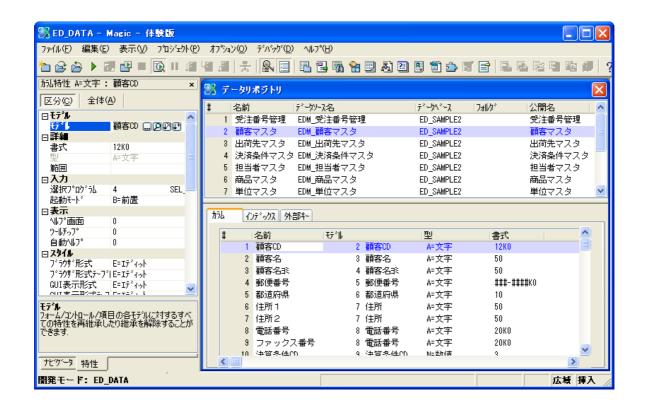


## 3.4.2 データリポジトリ

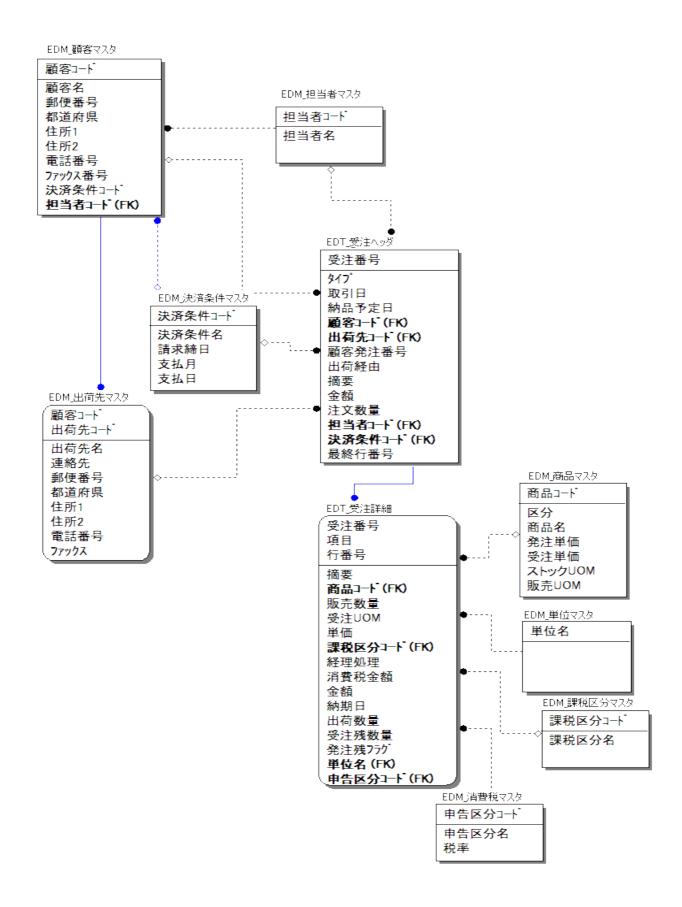
データリポジトリには、アプリケーションに固有なテーブルの定義がされています。これには、顧客マスタ、商品マスタ、受注ヘッダ、受注詳細ヘッダなどが含まれます。

また、一時テーブルとして、メモリデータベースに 受注ヘッダ TMP、受注明細 TMP が定義されています。

生産性と保守性の向上のために、それぞれのテーブルのカラムは、すべてモデルを参照して定義されています。



データリポジトリで定義されているテーブルの ER 図は次のページの図の通りです。



## 3.4.3 プログラムリポジトリ

下図は、データコンポーネントのプログラムリポジトリです。



データコンポーネントのプログラムリポジトリには、次のような種類のプログラムが定義されています。

- 選択プログラム
- ▼スタ照会・修正プログラム
- サンプルデータ入力

それぞれについて、次ページ以下に簡単に説明します。

種類	プログラム例	目的
選択プログラム		コードデータ(顧客コード、商品コードなど)でズームされたと きに表示される、一覧選択プログラム。

例: 顧客選択プログラム。この選択プログラムでは、顧客コード、顧客名、顧客名読みを入力してキー検索を 行うことができるようになっています。



種類	プログラム例	目的
マスタ照会・修正プログラム		マスタテーブル(顧客マスタ、商品マスタなど)を照会、修正、登録などするためのプログラム。

例: 顧客マスタ。下図では、顧客 0110 の顧客についてデータを修正する修正画面も表示されています。



種類	プログラム例	目的	
サンプルデータ入力	BT_サンプルデータ入力	テキストファイルのサンプルデータを、DBMS にインポートします。	
例:BT_サンプルデータ入力。このプログラムはバッチタスクなので、画面表示はありません。			

## 3.5 受注コンポーネントの内容

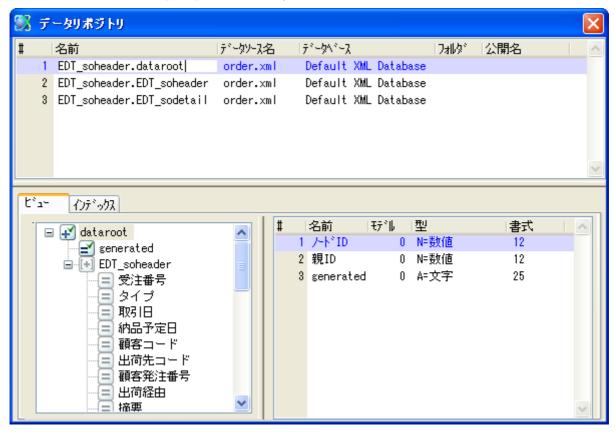
受注コンポーネント ED\_JUCHU には、受注業務に関連するプログラムが定義されています。

## 3.5.1 モデルリポジトリ

受注コンポーネントには、モデルが定義されていません。モデルはすべてデータコンポーネント、および共通コンポーネントのものを参照しています。

## 3.5.2 データリポジトリ

データリポジトリには、XML データベースとして定義されてデータソースが3つ定義されていますが、現時点のサンプルアプリケーションでは利用されていません。

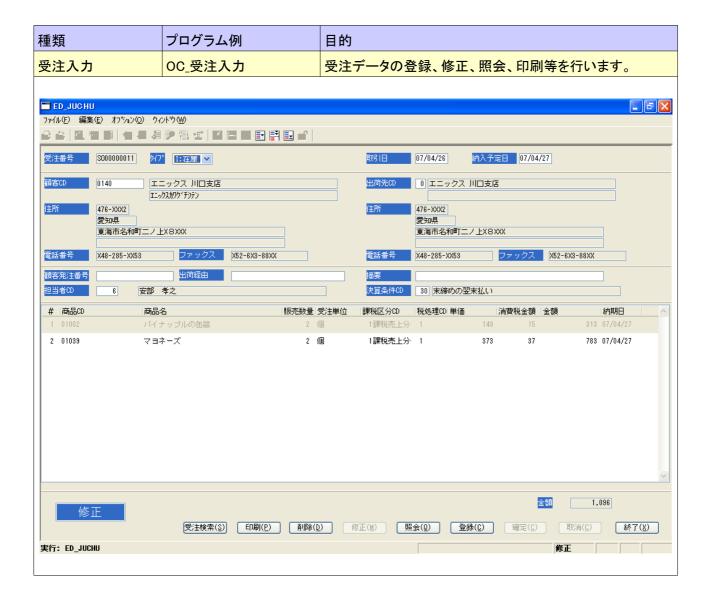


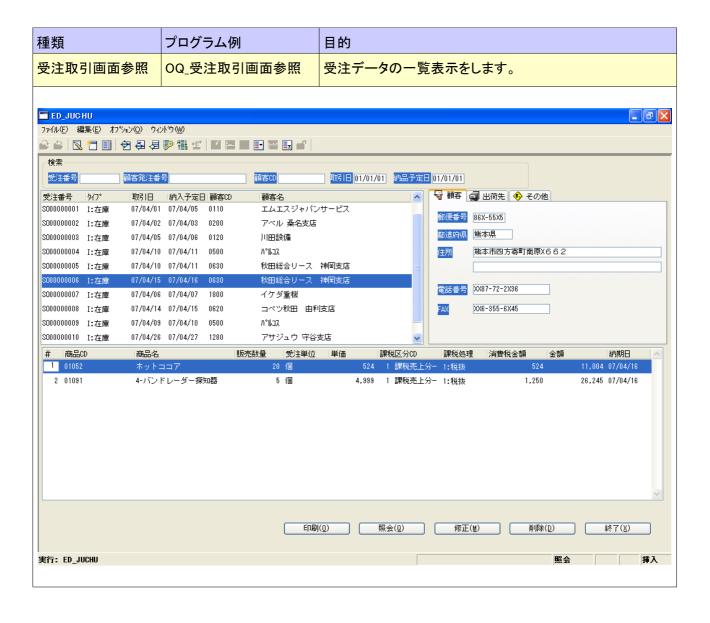
## 3.5.3 プログラムリポジトリ

受注コンポーネントのプログラムリポジトリには、次のようなプログラムが定義されています。

- 受注入力
- 受注取引画面参照
- 印刷
- 関連プログラム

以下にそれぞれについて簡単に見ていきます。

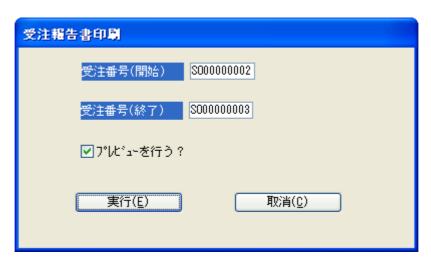




種類	プログラム例	目的
印刷	OP_受注報告書印刷	受注データを印刷する。

#### 例: OP\_受注報告書印刷。

パラメータ入力画面は次のとおりで、受注番号などで範囲付けできます。



#### 印刷結果は、次のようなものとなります。

		受注執	<b>设告書</b>				ページ: 1 日付: 2007/08/24 時刻: 16:26:04
顧客 0110 顧客発注番号LS00	)000001 取引日 07/04/01 納入予定日 07/04/05 エムエスジャパンサービス 1 出荷経由 横浜 !川 由番	摘要	0 エムエスジャパンサ 重要 ‡ 30 末締めの翌末払			取引夕	イプ 二在庫
NO 商品コート*	商品名	販売数量	課税区分	経理処理	単価	消費税金額	金額 納期日
1 01000	コーン	20	1課税売上分一般仕入	1:税抜	75	75	1,575 07/04/08
2 01003	レタス	20	1課税売上分一般仕入	1:税抜	188	188	3, 948 07/04/05
					合計	263	5, 523
顧客 0200 顧客発注番号AB	000002 取引日 07/04/02 納入予定日 07/04/03 アベル 桑名支店 AOO2 出荷経由 神戸 部 孝之	摘要	0 アベル 桑名支店 宅急便 ‡ 28 末締めの翌20払	,\		取引タ	イプ 二在庫
NO 商品コート*	商品名	販売数量	課税区分	経理処理	単価	消費税金額	金額 納期日
1 01026	チキン胸		1課税売上分一般仕入	1:税抜	431	431	9,051 07/04/03
2 01052	ホットココア	20	1課税売上分一般仕入	1:税抜	524	524	11,004 07/04/03
3 01065	コンパックオフィスツール	20	1課税売上分一般仕入	1:税抜	1, 249	1, 249	26, 229 07/04/03
					合計	2, 204	46, 284
顧客 0120 顧客発注番号	000003 取引日 07/04/05 納入予定日 07/04/06 川田設備 出荷経由 :部 孝之	摘要	1 川田設備一東京 ‡ 35 末締めの翌々5払	, L v		取引タ	イプ Ⅰ∶在庫
NO 商品コート <sup>*</sup>	商品名	販売数量	課税区分	経理処理	単価		金額 納期日
1 01065	コンパックオフィスツール	20	1課税売上分一般仕入	1:税抜	1, 249	1, 249	26, 229 07/04/06
2 01065	コンパックオフィスツール	20	1課税売上分一般仕入	1:税抜	1, 249	1, 249	26, 229 07/04/06
					合計	2, 498	52, 458
		摘要	0 パルコス 重要 ‡ 1 随時			取引タ	イプ I:在庫
NO 商品コート*	商品名	販売数量	課税区分	経理処理	単価	消費税金額	金額 納期日
1 01078	2 スペ"シャルテ"サ"インカ"スキャッフ"	20	1課税売上分一般仕入	1:税抜	1, 250	1, 250	26, 250 07/04/11
2 01082	8インチ 2ーウェイカー木°ックススヒ°ーカー	20	1課税売上分一般仕入	1:税抜	10, 545	10, 545	221, 445 07/04/11

## 3.6 ホストコンポーネントの内容

ホストコンポーネントの目的は、3.1 コンポーネントの構成で説明したように、業務ごとに分割したコンポーネントを一つにまとめることにあり、実質的な内容としては、メニューを提供する機能があるだけです。

#### 3.6.1 モデルリポジトリ

ホストコンポーネントのモデルリポジトリでは何も定義されていません。

## 3.6.2 データリポジトリ

ホストコンポーネントのデータリポジトリでは、メニュー定義用のデータを格納するテーブルが定義されています。 このテーブルはメモリデータベースに作成されるもので、データはアプリケーションが起動されるたびに、テキストファイルから読み込んできます。



メニュー定義のテキストファイルは、ENV ディレクトリの EDM\_メニュー.TXT です。

このファイルの内容は、階層的に定義されているメニューのデータで、1レコードが1メニュー項目に対応します。各レコードでは、ノードID、親ノードID、メニューラベル、および選択時に呼び出すプログラムの公開名が定義されています。

1	0 受注入力	
2	1 受注入力	00_受注入力
3	1 受注取引画面参照	00_受注取引画面参照
4	1 受注報告書印刷	OP_受注報告書印刷
10	0 マスタメンテナンス	
11	10 顧客マスタ照会	OM_顧客マスタ照会
12	10 商品マスタ照会	OM_商品マスタ照会
13	10 決済条件マスタ照会	OM_決済条件マスタ照会
14	10 担当者マスタ照会	OM_担当者マスタ照会
15	10 単位マスタ照会	OM_単位マスタ照会
16	10 課税区分マスタ照会	OM_課税区分マスタ照会
17	10 消費税マスタ照会	OM_消費税マスタ照会
20	0 印刷	
21	20 受注確認書印刷	0P_受注確認書印刷
22	20 受注報告書印刷	OP_受注報告書印刷
90	0 スペシャル	

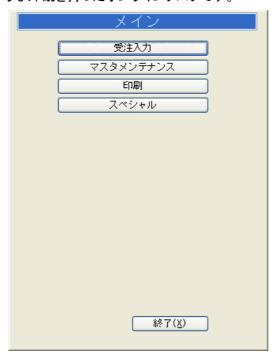
91	90 サンプルデータ入力	BT_サンプルデータ入力
92	90 メニュー照会	OM_メニュー照会

## 3.6.3 プログラムリポジトリ

ホストコンポーネントのプログラムリポジトリでは、メニューを実現するプログラム、およびメニューデータの保守のためのプログラムが定義されています。

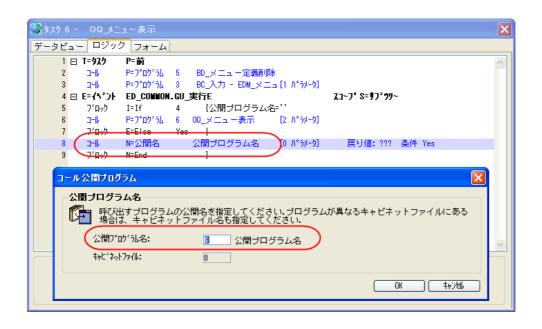


メニュープログラムは、下図のような外観を持ったオンラインタスクです。



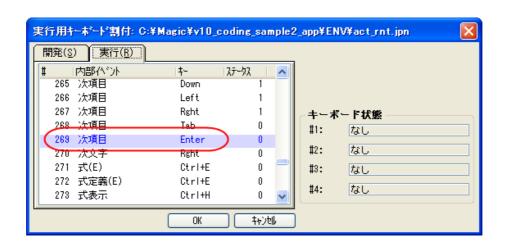
ここでのプッシュボタンは、実は境界もスクロールバー、タイトルもないテーブルコントロールに表示されるレコードデータで、表示にエディットコントロールの代わりにプッシュボタンコントロールを使っているものです。

メニューに対応するプログラムを呼び出すときには、コール 公開名 コマンドを使い、メニューレコードに定義されている公開プログラム名を参照して呼び出しを行います。

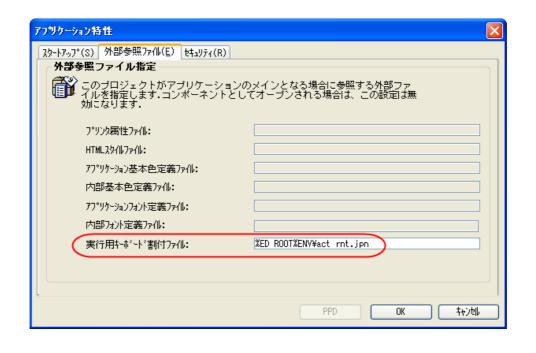


## 3.6.4 アプリケーション特性

ホストコンポーネントのアプリケーション特性では、一箇所だけ変更を行っています。それは、Enter キーを 次項目 イベントにマップするために、独自のキーボード定義ファイルを使っていることです。



このキーボード定義ファイルは、ENV ディレクトリの下の act\_rnt.jpn です。



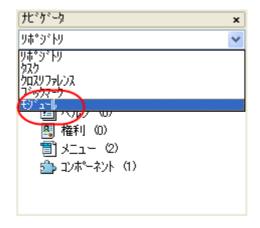
## 3.7 モジュール定義

モジュール定義というのは、V10で新しく導入されたもので、コンポーネントなどの関連するプロジェクトを モ ジュールとして参照の定義を記録しておく機能です。

#### 3.7.1 モジュールの表示

モジュール定義は、ナビゲータからモジュールペインを開くことによって、見ることができます。

例えば、データコンポーネント ED\_DATA プロジェクト に定義されているモジュールを見るには、ナビゲータ を表示した状態で、ナビゲータのコンボボックスから モジュールを選択します。

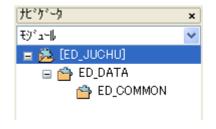


すると、モジュールの依存関係が表示されます。この 場合、ED DATA が、ED COMMON コンポーネントに 依存していることを表しています。



#### 3.7.2 モジュールのネスト

モジュールはネストして定義することも可能です。すなわち、ED DATA プ ロジェクトは ED COMMON プロジェクトに依存していますが、ED JUCHU プロジェクトが ED\_DATA コンポーネントに依存しているので、ED\_JUCHU → ED\_DATA → ED\_COMMON という依存関係があることがわかります。



このような場合に、ED\_JUCHU プロジェクトで、ED\_DATA プロジェクトをモ ジュールとして追加すると、モジュールペインには、ED\_DATA → ED COMMON という依存関係も自動的に取り込まれ、表示されるように なります。(右図)

また、一番トップの ED HOST プロジェクトは、ED JUCHU プロジェクトに 依存しているので、ED\_HOST プロジェクトを開いてモジュールペインを表 示させてみれば、



 $ED\_HOST \rightarrow ED\_JUCHU \rightarrow ED\_DATA \rightarrow ED\_COMMON$ 

という関係が Magic Studio で認識されていることがわかります。



#### 3.7.3 モジュールの用途

モジュールにより、プロジェクト間の依存関係を定義することはわかりましたが、実際的にはモジュールを定義することによりどういった利点があるのでしょうか?

まず、第一には、プロジェクトのオープンの手間が若干省略されます。モジュールペインを表示している状態で、依存しているモジュールのノードをマウスでダブルクリックすると、そのプロジェクトが開きます。モジュールの依存関係が定義されているならば、再帰的に全モジュールの依存関係が表示されるので、複数のコンポーネントからなるアプリケーションの場合に、いちばんトップのプロジェクト(サンプルでは ED\_HOST)だけを最初に開いておけば、あとはモジュールペインからコンポーネントのプロジェクトを開くことができます。

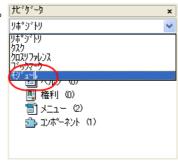
第二には、デバッグの際にソースコードが表示されることが、大きな利点として挙げられます。デバッグを行っている場合、コンポーネントの中のプログラムについては、通常はソースコードを表示することができません。従って、ステップ実行やブレークなどを入れて実行の流れを制御観察することが困難になります。しかし、モジュールとしてコンポーネントのプロジェクトが定義されていたならば、Magic Studio はコンポーネントの中にまで入って実行制御やブレイクポイントを行うことが可能になります。

## 3.7.4 モジュールの追加

サンプルプログラムでは予めモジュール定義がされていましたが、モジュール定義は自動的に登録されるものではなく、あくまで、開発者が自分でモジュールの追加を行う必要があります。

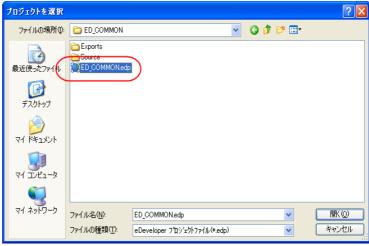
モジュールの追加は、次のように行います。これは、ED\_DATA プロジェクトモジュールが定義されていない状態で、ED\_COMMON プロジェクトをモジュールとして追加する場合の例です。

次に、ポップアップメニューから、モジュール追加 メニューを選択します。





プロジェクトを選択 ダイアログが開くので、モジュールとして追加したいプロジェクトファイルを指定し、OKを押します。



モジュールが追加されました。



一般的には、コンポーネントとして使っているプロジェクトをモジュールとして定義しますが、コンポーネントリポジトリにコンポーネントとして使うプロジェクトを登録しても、自動的にモジュールとして定義されるわけではありません。

また逆に、コンポーネントリポジトリに取り込まれていないプロジェクトであっても、モジュールとして追加することができます。これは、コンポーネントリポジトリには登録しないで、コール公開プログラムにより呼び出すコンポーネントを使う場合に便利です。

# Magic eDeveloper V10



Magic eDeveloper V10

コーディングサンプル Version 2 (コンポーネント編)

Copyright © 2007, Magic Software Japan K.K., All rights reserved.

第1版 2007年9月19日

発行 〒151-0053 東京都渋谷区代々木三丁目二十五番地三号

あいおい損保新宿ビル14階

マジック ソフトウェア・ジャパン (株)

 $\verb|http://www.magicsoftware.co.jp/|$