



Magic xpa レコードメインからイベント化移行

マジックソフトウェア・ジャパン株式会社



はじめに

- Magicは、V10からイベント指向プログラミングに移行し、レコードメインは旧バージョンからの互換レベルで利用できるようにしておりましたが、それからすでに15年が経過いたしました。
- 本セミナーでは、レコードメインによるプログラミングのデメリットと、イベント化することによるメリットをご案内すると共に、イベント指向のプログラミング方法を理解し、移行することができるよう、概念と機能およびプログラム方法について具体例をあげて説明します。
- レコードメインからイベント志向のプログラミングへの移行を考えているMagic 技術者には是非お聞きしていただきたい内容です。



目次

- V 8 / V 9 のおさらい：レコードメイン（RM）はどう動いていたか？
- Magic xpaタスクエディッタの構成
- 例① リンクの成功チェックをする
- 例② 条件により入力項目をスキップする
- 例③ 項目更新により再計算を行う
- 例④ プッシュボタンをハンドリングする
- 例⑤ ズーム処理を行う
- まとめ



V 8 / V 9 のおさらい レコードメイン (RM) はどう動いていたか？



V 8 のタスクエディッタの構成

レベルテーブル
レコード、タスク、
ブレイクレベルなど

処理テーブル
レコードメインでは

- データ定義
- オンライン画面上での操作を定義

タスク定義: 718.1 - SO入力 受注情報1 (597).SO受注明細

#	ブレイク	処理レベル	前処理	メイン	後処理	トリガアクション	エラー
1		レコード	0	31	8	L=オロック	A=アラート
2		タスク	0		7	Yes	A=アラート

処理テーブル: レコード メイン

順	処理コマンド	内容	範囲	位置付	加	条件					
1	抜外 V=変数 : 1	V.商品DLRC	代入	0 0 0	0 0	S C Yes					
2	抜外 V=変数 : 2	V.物流拠点省略DLRC	代入	0 0 0	0 0	S C Yes					
3	抜外 V=変数 : 3	V.物流拠点(開始)DLRC	代入	0 0 0	0 0	S C Yes					
4	抜外 V=変数 : 4	V.物流拠点(終了)DLRC	代入	0 0 0	0 0	S C Yes					
5	抜外 V=変数 : 5	V.別件DLRC	代入	0 0 0	0 0	S C Yes					
6											
7	抜外 R=実行データ : 1	受注#	代入	29 29	0 0	S C Yes					
8	アクション : 1	KBPUT ('次項目'ACT)			戻 ??	S F 7					
9	抜外 R=実行データ : 2	受注明細#	代入	24 0 0	28 0	S C Yes					
10	ブロック	[P.双対ト = 'C' OR P.				C C 7					
11	コール P=プログラム : 566	SO選択 商品一覧	ハラ	3 フォム	0	0	0	0	0	B C	Yes
12	抜外 R=実行データ : 3	商品#	代入	0 0 0	0 0	S C Yes					
13	抜外 R=実行データ : 4	商品Sub#	代入	0 0 0	0 0	S C No					
14	リンク Q=照会 : 16	商品	リンク	1 順	A=昇順	戻 LW					Yes
15	抜外 R=実行データ : 1	商品#	代入	0 0 0	39 39	S C No					
16	抜外 R=実行データ : 2	商品Sub#	代入	0 0 0	40 40	S C No					
17	抜外 R=実行データ : 9	規格_型番	代入	0 0 0	0 0	S C No					
18	抜外 R=実行データ : 7	商品名	代入	0 0 0	0 0	S C No					
19	抜外 R=実行データ : 11	サイズ	代入	0 0 0	0 0	S C No					
20	抜外 R=実行データ : 13	単価	代入	0 0 0	0 0	S C No					
21	抜外 R=実行データ : 16	原価_CDROM_輸入	代入	0 0 0	0 0	S C No					
22	抜外 R=実行データ : 22	原価	代入	0 0 0	0 0	S C No					
23	抜外 R=実行データ : 28	商品タイプ	代入	0 0 0	0 0	S C No					
24	抜外 R=実行データ : 29	商品タイプ2	代入	0 0 0	0 0	S C No					

OK 取消



R Mの基本的考え方は

- 「カーソルの動きに応じて、R Mに処理を記述する」

定価/販価/原価	仕切/数量	売上/原価/粗利	引列生成依頼	備考
-###,###,###	###.#####	-###,###,###	<input type="checkbox"/> 引列#生成	XXXXXXXXXXXXXXXXXX
-###,###,###	-#####	-###,###,###	<input type="checkbox"/> 引列#確認	

65	仕外	R=実行コマ	8	受注定価	代入	45
66	仕外	R=実行コマ	9	受注単価	代入	45
67	仕外	R=実行コマ	10	受注原価単価	代入	46
68	項目更新	:NU		受注原価単価	式	49
69	項目更新	:NV		仕切	式	62
70	仕外	R=実行コマ	11	仕切	代入	0
71	項目更新	:NT		受注単価	式	63
72	仕外	R=実行コマ	12	受注数量	代入	22
73	項目更新	:NY		受注金額	式	64
74	項目更新	:NZ		粗利金額	式	65
75	仕外	R=実行コマ	21	備考	代入	67

受注数量 から 備考 に移るときに、この二つの項目更新コマンドが実行される



RMフローパラメータによる実行の制御

位置付	加	条件
0	0	S C 14
止	Yes	C C 48
止	Yes	C C 27
0	0	S C Yes
止	Yes	C C 27

フローモード

方向

フローモード：

C=両用 → S=通常 + F=高速

B=前置、A=後置 → ズームで使う

方向： C=両方向 → F=前方 + B=後方

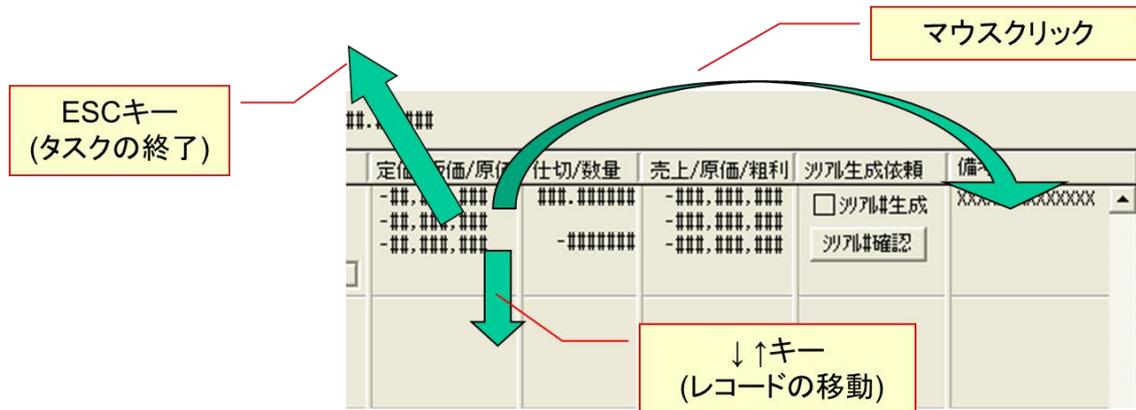


RMフローモード

- S=通常モード： Enter/Tab (次項目)で動く場合に実行



- F=高速モード： マウス/↑ ↓キー/タスク終了などでカーソルが動く場合に実行



RMフローパラメータによる実行の制御

位置付	加	条件
0	0	S C 14
止	Yes	C C 48
止	Yes	C C 27
0	0	S C Yes
止	Yes	C C 27

フローモード

方向

フローモード：

C=両用 → S=通常 + F=高速

B=前置、A=後置 → ズームで使う

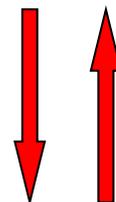
方向： C=両方向 → F=前方 + B=後方



R Mカーソル移動方向

- 順方向にカーソルが動く → 上から下にコマンド実行
- 逆方向にカーソルが動く → 下から上にコマンド実行
- 特定の方向のときだけ実行したい → 方向パラメータを F = 前方/B=後方に設定

72	例外	R=実行モード :	12	受注数量	代入	22
73	項目更新		:NY	受注金額	式	64
74	項目更新		:NZ	粗利金額	式	65
75	例外	R=実行モード :	21	備考	代入	67



逆方向

順方向

※ 実際にはもっと複雑な動きのルールあり。
R Mフローモードは結構難しいブラックボックス。



RMの書き方ABC

- A) どこで？

→ ある項目（セレクトコマンド）と別の項目（セレクトコマンド）の間で

- B) いつ？

→ カーソルの動くとき

- [次項目]アクション（Enter/Tabキ）で動く場合のみ実行 → S=通常

- ↓↑キー、ESC、マウス等で動く場合 → F=高速

- 常に実行する → C=両用 ※ 特定の移動方向を限定したければ、F=前方またはB=後方を設定

- C) 何を？

→ セレクトコマンドの間に、実行するコマンドを記述。



Magic xpa の タスクエディタの構成



Magic xpaタスクエディッタの構成

データビュー：
データの定義だけ
(RMのセレクト、
リンクコマンド)

行	タイプ	番号	名前	属性	値
1	M=メイン	79	受注明細		インデック1
2	V=変数	1	LW.V.商品DLRC	[35]	L=論理 5
3	V=変数	2	LX.V.物流拠点省略	[35]	L=論理 5
4	V=変数	3	LY.V.物流拠点(開始)	[35]	L=論理 5
5	V=変数	4	LZ.V.物流拠点(終了)	[35]	L=論理 5
6	V=変数	5	MA.V.シリアル#DLRC	[35]	L=論理 5
7	C=カラム	1	MB.受注#	[24]	N=数値 6P0 範囲:29 終:29 代入:29 受注#
8	C=カラム	2	MC.受注明細#	[21E]	N=数値 4 代入:24 最終明細#+5
9	C=カラム	3	MD.商品#	[12C]	N=数値 9P0
10	C=カラム	4	ME.商品Sub#	[121]	N=数値 6P0

データビューにはロジックを書きません

ロジック：
RM以外のレベル
(タスク前/後、
レコード前/後・・・)
RMの処理コマンド

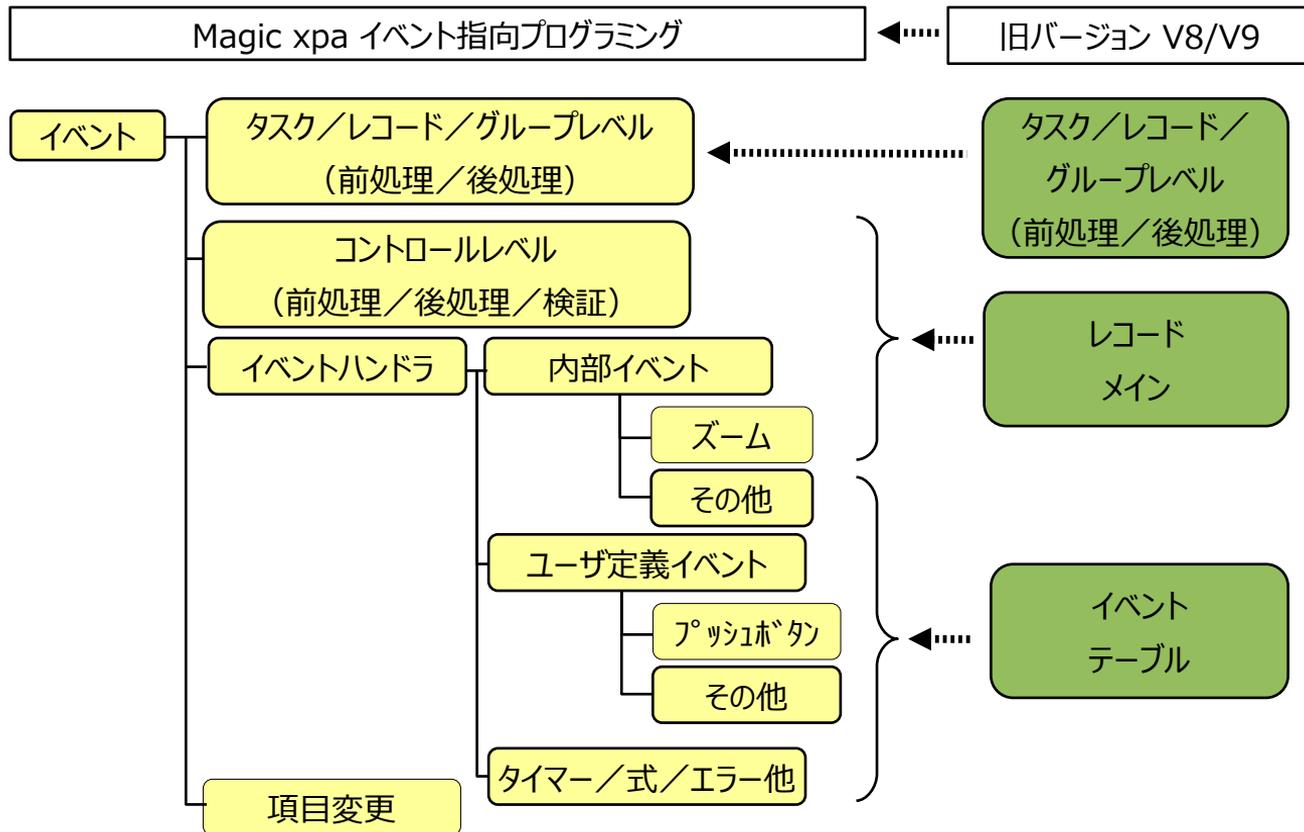
行	タイプ	名前	条件	値
2	ブロック	I=[f	7	[P.タスク= 'C' OR P.タスク= 'M'
3	コール	P=プログラム	594	SB集計 明細行数 [5 パラメータ]
4	コール	P=プログラム	590	SB変更 明細番号の引[2 パラメータ]
5	項目更新	V=項目	BV	最終明細# 値: 26 明細行数*5
6	コール	P=プログラム	561	SBカット MSE社に対する[3 パラメータ]
7	項目更新	V=項目	BN	V.明細番号保存 値: 35 受注明細#
8	ブロック	N=End]
9	R=レコード	P=前		
10	R=レコード	S=後		
11	ブロック	I=[f	7	[P.タスク= 'C' OR P.タスク= 'M'
12	項目更新	V=項目	LH	受注売上合計 値: 50 受注金額
13	項目更新	V=項目	LI	受注粗利合計 値: 51 粗利金額

RMに書いていたコマンド（コール、
項目更新、・・・）は、すべてハンドラ
として記述。



Magic xpaでのハンドラの種別とタイプ

すべて「ロジック」
タブにイベントハン
ドラとして記述し
ます



ハンドラの書き方

ロジック タブに記述

ヘッダ行：ハンドラごとに、種別とタイプ他を定義。
ハンドラレベル、ハンドラタイプ、コントロール名、その他
(ハンドラタイプによる)を指定

コントロール名：カーソルがここにあるときに
ハンドラが有効になる。

データビュー		ロジック	フォーム		
37	日	C=コントロール=検証	コント	受注数量	条件 Yes
38		項目更新 V=項目	NS	受注金額	値: 56 受注単価*受注数量
39		項目更新 V=項目	NT	粗利金額	値: 57 (受注単価-受注原

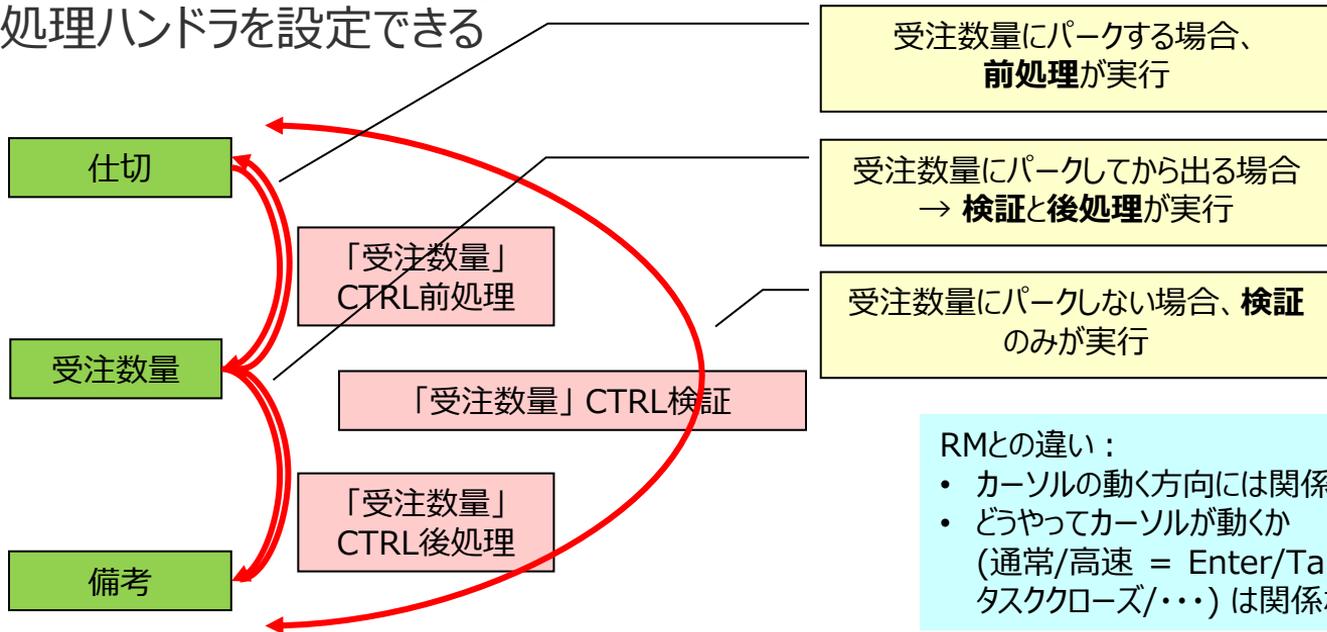
詳細行：処理内容を記述。
項目更新、コール、アクション、・・・
などのコマンド

条件：ハンドラが有効に
なる条件を式で指定



コントロールレベルのハンドラ

- オンライン画面のカーソル移動時に実行すべきコマンドは、コントロールレベルのハンドラに記述する。
- 各コントロール(表示項目)ごとに、コントロール前処理、コントロール検証、コントロール後処理ハンドラを設定できる



RMとの違い：

- カーソルの動く方向には関係ない
- どうやってカーソルが動くか (通常/高速 = Enter/Tab/マウス、↑ ↓キー/タスクローズ/...) は関係ない。



コントロールレベルハンドラの書き方 A B C

- A) どこで？ →ある表示項目 (例： 受注数量) において…
- B) いつ？
 - カーソルが項目にパークする場合にだけ実行するなら
 - 入るときの処理 → コントロール前処理に書く
 - 出るときの処理 → コントロール後処理に書く
 - カーソルが動くときに常に実行するなら
 - → コントロール検証に書く
- C) 何を？ → ハンドラの中にコマンドを記述



RM→コントロールハンドラのTIPS

- 機械的に変換しようとしても、却って不必要に複雑になってしまう。
- 「本来何をしたいのか？」の原点に立ち帰って、ハンドラ設定を考える。
- フローモード、方向により実行を制御したい場合、パラメータ設定もできる。(後述)

※ 注意：

- 以下に説明する内容は「考え方」を説明するために簡略化したものです。
- RMの複雑な詳細に依存した作りになっている場合には、全く同じ動きにすることが困難な場合もあります。



例① リンクの成功チェック



例① リンクの成功チェック (例：商品#) 昔のやりかた

(A) どこで？
→ 「商品Sub#」項目の後で

処理テーブル：レコード メイン

#	処理コマンド	内容	範囲	位置付	加-	条件
11	コール P=アプル私 :	566 SO選択 商品一覧	パラ 3	フォーム 0	バック No	B C Yes ▲
12	例外 R=実行データ :	3 商品#	代入 0	0 0	0 0	S C Yes
13	例外 R=実行データ :	4 商品Sub#	代入 0	0 0	0 0	S C No
14	リンク Q=照会 :	16 商品	インデックス 1	順 A=昇順	戻 LW	Yes
15	例外 R=実行データ :	1 商品#	代入 0	0 0	39 39	S C No
16	例外 R=実行データ :	2 商品Sub#	代入 0	0 0	40 40	S C No
17	例外 R=実行データ :	9 規格_型番	代入 0	0 0	0 0	S C No

この間、パークしない項目 (リンク項目)

39	例外 R=実行データ :	49 別列#自動生成対象	代入 0	0 0	0 0	S C No
40	例外 R=実行データ :	53 商品有効F	代入 0	0 0	0 0	S C No
41	リンク終了					
42	エラー :	4 '\$無効な商品#が選択さ	モード W=警告			C C 37

(C) 何を？ →
エラーコマンド

(B) いつ？ → 常に実行
→ C=両用、C=両方向
(リンクフラグで条件付け)



例① リンクの成功チェック : Magic xpaでは

(A) どこで?
→ 「商品#」項目

No.	商品#&Sub#	名称/型式/
	XXXXXXXXXX	XXXXXXXXXX
	XXXXXXXXXX	XXXXXXXXXX
	XXXXXXXXXX	XXXXXXXXXX

2:MCC

プロパティ

商品# エディット コントロール

検索

データ	MD
項目	商品#
コントロール名	商品#
書式	999
型	数値

(B) いつ? → 常に実行
→ コントロール検証

コントロール欄
に設定

※ V8 APだと、コントロール名がついていないことが多い
→ 必要に応じ設定。

データビュー ロジック フォーム

23	C=コントロール=検証	商品#
24	エラー W=警告 4	'無効な商品マスタが選択表示: B=ボックス

条件 Yes
条件 34 (P.タスクモード=

(C) 何を? (複数のコマンド指定可)
→ エラーコマンド

コマンドの実行の
条件付け



例① 微妙な違い

- RMの場合、前の項目に戻る場合にはエラーチェックを行わなかった。
- Magic xpa コントロール検証の場合には、前方向でも後方向でもエラーチェックを行う。

タブ	ロジック	フォーム	
24	C=コントロール	V=検証	コントロール商品
25	I=エラー	W=警告	4 '無効な商品'

特性: エラー 処理コマンド

区分(C) 全体(A)

詳細

モード W=警告

リスト 4

表示 B=チェック

加-モード C=両用

加-方向 F=前方

条件 E=前方
B=後方
C=両方向

フローモードや方向を指定したい場合には、コマンドごとに、特性シートで指定することができる。
Flow 関数で条件付けしても可。



例② 条件により、 入力項目を飛ばす



例② 受注番号の後で K B P U T (‘次項目’) を行う

受注#の直後で、
K B P U T(‘次項目’ACT)を実行

フローは
S=通常
F=前方

処理テーブル：レコード メイン

#	処理コマンド	内容	70-	条件
7	例外 R=実行データ :	1 受注#	S C	Yes ▲
8	アクション :	1 K B P U T (‘次項目’ACT)	S F	7
9	例外 R=実行データ :	2 受注明細#	S C	Yes

どういう動作になるか？

受注# から Enter / Tab で移動するとき、

条件 7 が成立したら、次の項目（受注明細#）をスキップする。

※ (S=通常モード のみ) マウスやレコード移動時は実行しない

※ (F=前方 のみ) 受注明細# から 受注# に戻る場合も実行しない



例② xpa コントロールレベルハンドラで書くと・・・

(A) どこで？
「受注#」で

(B) いつ？ → パークして出るときだけ
→ コントロール後処理

ID	名前	条件
25	C=コントロールS=後 コト 受注#	条件 Yes
26	イベント実行 次項目	ウェイト: No 条件: 7 P.奴奴ト≠'C'

(C) 何を？
「次項目」イベントを実行

※ KBPUTでなく、イベント実行 コマンドを使いましょう。

「条件」と「フロー方向」で条件付け

項目	値
条件	Yes 7
フロー方向	F=前方
イベント	次項目
ウェイト	No 0
パラメータ	0
出力先コンテキスト	0

※ ただ、この例について言えば、受注明細# 項目のパーク条件で制御した方がよい。
→ 別の方法による解も考えてみましょう。



例③ 項目更新による再計算



例③ 項目更新による再計算ロジック

処理テーブル：レコード メイン

#	処理コマンド	内容	
72	例外 R=実行 : 12	受注数量	代入 22
73	項目更新 :NY	受注金額	式 64
74	項目更新 :NZ	粗利金額	式 65
75	例外 R=実行 : 21	備考	代入 67

- 「受注数量」から「備考」に移る時、受注金額と粗利金額の再計算を行う。



例③ Magic xpa コントロール ハンドラで書くと...

(A) どこで?
「受注数量」

定価/販価/原価	仕切/数量	売上/原価/粗利	別7
-###,###,###	###.#####	-###,###,###	
-###,###,###	○-###.#####	-###,###,###	
-###,###,###	○-###.#####	-###,###,###	

プロパティ

受注数量 エディットコントロール

検索

詳細

データ	NY
項目	受注数量
コントロール名	受注数量
書式	N7
型	数値

(B) いつ? → 常に実行
→ コントロール検証

データビュー ログック フォーム

37	日	C=コントロールV=検証	コントロール	受注数量		条件	Yes
38		項目更新	V=項目	NS	受注金額	値: 56	受注単価*受注数量
39		項目更新	V=項目	NT	粗利金額	値: 57	(受注単価-受注原価)

(C) 何を? → 二つの
処理コマンドはそのまま



例③ 微妙な違い

- 項目更新コマンドが二つある (受注金額と粗利金額)
 - RMの時は、カーソルの移動方向により、コマンドの実行順序が異なる。
(逆方向の場合は、下から上へ)
 - Magic xpa コントロール検証の時は、カーソルの移動方向に関わらず、上から下に実行される。
-
- ✓ この例では、どちらでも結果は同じになるので変更する必要はないが、依存関係のある場合には実行結果が異なるので注意。
 - ✓ 必要なら方向パラメータなどで条件づけ。



項目変更ハンドラ（V10～新機能）

- 例③のように、項目（ex. 受注数量）の値の変更をきっかけとして実行すべきコマンドは、「項目変更」ハンドラにした方がわかりやすい。

項目の値に変更があった場合に
→ 「V=項目 C=変更」ハンドラ

どの項目について？
→ NY (受注数量)

ID	項目	条件	項目	値	計算式
37	V=項目	C=変更	NY 受注数量		条件
38	項目更新	V=項目	NS 受注金額	値: 56	受注単価*受注数量
39	項目更新	V=項目	NT 粗利金額	値: 57	(受注単価-受注原価)

これを実行する

- 意図がわかりやすい。
- 設定忘れがなくなる： いつ、どこで、どのようにして変更されたかには関係なく、変更があったら常に実行される。
(ex. ユーザ入力、項目更新コマンド、プログラムへのパラメータ、など。画面に表示されていなくともよい)
- 値に変更がない場合には実行されない → 無駄な再計算をなくす



例④ プッシュボタン



例④ プッシュボタン：V8のタスクイベントテーブル

売上/原価/粗利	シリアル生成依頼	備考
# -###,###,###	<input type="checkbox"/> シリアル生成	XXXXXXXXXXXXXXXX
# -###,###,###	<input type="checkbox"/> シリアル確認	

コントロール特性: プッシュボタン

詳細 入力 外観 位置とサイズ

パラメータ	値	式
ボタン スタイル	P=プッシュボタン	
データ	??	0
項目	??	
コントロール名		
ラベル/書式	シリアル確認	0
デフォルト イメージファイル名		
アクション	ユーザアクション1	

OK 取消

プッシュボタンの
コントロール特性で

アクションを設定
(ユーザアクション1)

タスクイベントテーブルにア
クションを指定

タスクイベント: 718.1 - SO入力 受注情報1 (507).SO受注明細

#	ホットキー	アクション	経過時間	式	スコープ	コール	Prog/Task	パラメータ	ロック
1		ユーザアクション1	00:00:00	0	T=タタ	P=プッシュ	588	2	No

処理内容は別タスク/プロ
グラムに記述



例④ プッシュボタン： Magic xpa ユーザ定義イベントを利用

「ユーザアクションn」の代わりに、**ユーザ定義イベント**を使うのが便利です

① ユーザイベントテーブルにわかりやすい名前で作成

#	名前	トガタイプ	トガ	パラメタ	強制終了
12	U_シリアル番号発行	N=なし		0	N=なし

売上/原価/粗利	シリアル生成依頼	備考
-###,###,###	<input type="checkbox"/> シリアル#生成	XXXXXXXXXXXXXXXX
-###,###,###	<input type="checkbox"/> シリアル#確認	
-###,###,###		

②フォーム デザイナ上でコントロール特性を開く

プロパティ PB_シリアル#確認 ボタンコントロール

検索

項目	値
コントロール名	PB_シリアル#確認
書式	シリアル#確認
型	文字
ボタンスタイル	P=プッシュボタン
デフォルトイメージファイル	ボタン
イベントタイプ	U=ユーザ
イベント	U_シリアル番号発行
パラメタ	0
実行元	C=コンテナタスク

③プッシュボタン特性に「イベント」を設定

ボタンを押したときに、このイベントが発行されます。
タスクイベントテーブルで定義していた処理は、Magic xpaでは**イベントハンドラ**で行います。



Magic xpa イベントハンドラの書き方 A B C

(A)どこで？

「シリアル# 確認」PushButtonで。
→ Pushボタンのコントロール名
「PB_シリアル# 確認」を指定

E=イベント のハンドラでは省略も可。
省略すると「どこでも有効」になる

ハンドラの種別は
「E=イベント」

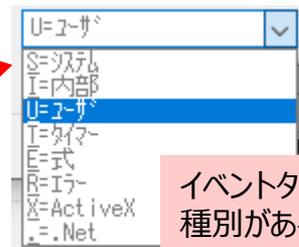
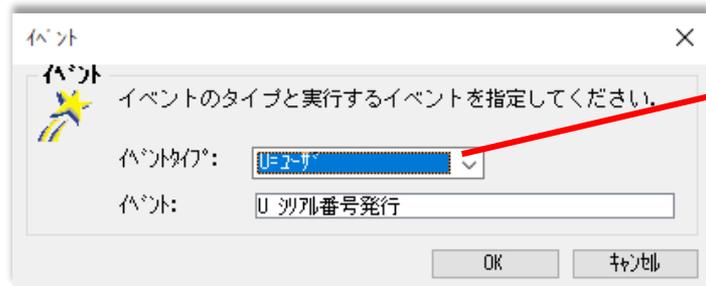


(C) 何を？

実行するコマンドを記述

(B)いつ？ → トリガとなる
ユーザ定義イベント「U_シリアル
番号発行」を指定

ズームして詳細設定



イベントタイプにはこれだけの
種別がある



ユーザ定義イベントを使うイベントハンドラ方式の利点

- わかりやすい。
 - V8では「ユーザアクションn」しかなく、意味がわからなかった。
 - Magic xpaでは意味のある名前 (ex. U_シリアル番号発行) で記述できる。
- イベント数に制限なし。
 - 「ユーザアクションn」は20個までだった。
 - それ以上の場合には、CTRLNAME()で条件付けしていた。
- 別途プログラムを作る必要なし。
 - V 8 では中身がコマンド 1 行でも、別途タスクを作る必要があった。



V8 vs. Magic xpa

タスクイベント: 718 - SO入力 受注情報1 (597)

#	ホットキー	アクション	経過時間	式	スコープ	コール	Prog/Task	アラート	出力
1		ユーザアクション1	00:00:00	0	T=奴	P=フ*ロカ*私	628	1	No
2		ユーザアクション2	00:00:00	0	T=奴	P=フ*ロカ*私	628	2	No
3		ユーザアクション3	00:00:00	0	T=奴	P=フ*ロカ*私	619	2	No
4		ユーザアクション4	00:00:00	0	T=奴	P=フ*ロカ*私	618	2	No
5		ユーザアクション5	00:00:00	0	T=奴	P=フ*ロカ*私	616	3	No
6		ユーザアクション6	00:00:00	0	T=奴	P=フ*ロカ*私	621	2	No
7		ユーザアクション7	00:00:00	0	T=奴	P=フ*ロカ*私	561	1	No

V8: どこで何が実行されるかわかりにくい。

xpa: 意図は明瞭

タスク 725 - SO入力 受注情報(元598)

データビュー ログ フォーム

#	イベント	U_受注書	コト PB_受注書	スコープ	T=奴	条件	Yes
51	E=イベント	U_受注書	コト PB_受注書	スコープ	T=奴	条件	Yes
52	コト	P=フ*ロカ*私 629	SB印刷 受注書 [1	アラート			
53	E=イベント	U_仕入先注文書		スコープ	T=奴		
54	コト	P=フ*ロカ*私 562	SB印刷 MSE社に対する票[1	アラート			
55	E=イベント	U_手配書	コト PB_手配書	スコープ	T=奴		
56	コト	P=フ*ロカ*私 624	SB印刷 手配書 [2	アラート			
57	E=イベント	U_宅配伝票	コト PB_宅配伝票	スコープ	T=奴		
58	コト	P=フ*ロカ*私 617	SB印刷 宅配伝票 [3	アラート			
59	E=イベント	U_請求書	コト PB_請求書	スコープ	T=奴		
60	コト	P=フ*ロカ*私 619	SB印刷 請求書 [2	アラート			
61	E=イベント	U_送り状	コト PB_送り状	スコープ	T=奴		
62	コト	P=フ*ロカ*私 621	SB印刷 送り状 [2	アラート			
63	E=イベント	U_納品書	コト PB_納品書	スコープ	T=奴		
64	コト	P=フ*ロカ*私 623	SB印刷 納品書 [2	アラート			
65	E=イベント	U_注文書	コト PB_注文書	スコープ	T=奴		



例⑤ ズーム処理



例⑤ ズーム処理（昔のやりかた）

- セレクトコマンドの直前にコマンド記述
- コマンドのフローモードをB=前置

11	コール	P=ブロック :	588	SO選択 商品一覧	ハウ	3	フォーム	0	ロック	No	B	C	Yes
12	例外	R=実行 :	3	商品#	代入	0	0	0	0	0	S	C	Yes

- 複数のコマンドのある場合は、ブロックコマンドで囲み、B=前置を設定

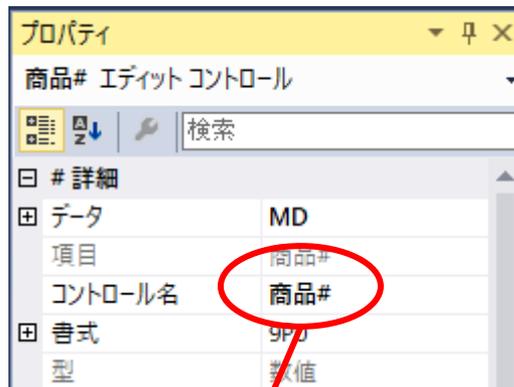
370	ブロック			[Yes							B	C	Yes
371	コール	P=ブロック :	587	SO入力 営業実績配分率	ハウ	2	フォーム	0	ロック	Yes	S	C	Yes
372	項目更新	:H		V.子奴終了F	式	3	計	N=代入	止	Yes	C	C	Yes
373	項目更新	:I		V.子奴画面消去F	式	4	計	N=代入	止	Yes	C	C	Yes
374	コール	T=奴 :	3	営業実績配分	ハウ	0	フォーム	0	ロック	Yes	S	C	Yes
375	ブロック終了]									
376	例外	V=変数 :	45	V.営業実績配分	代入	13	0	0	0	0	S	C	Yes



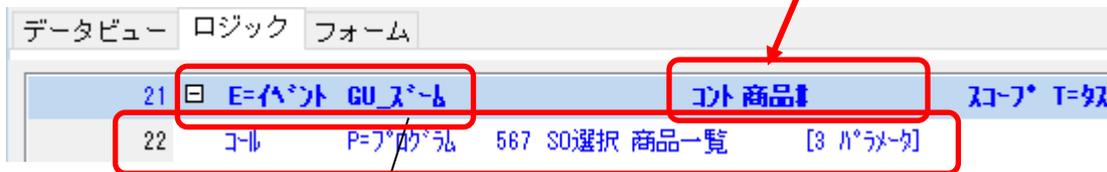
例⑤ ズーム処理 : Magic xpa イベントハンドラを使うと・・・

(A) どこで?
「商品#」の項目で

フォームデザイナー上で
ズーム項目を選択



「ズーム」を直接ハンドル
するのではなく、ユーザ
定義イベント「GU_ズーム」
をハンドルしている。
(理由 → 次へ)



(B) いつ?
ユーザ定義イベント
「GU_ズーム」が起きたら

(C) 何を?
プログラムをコール



ユーザ定義イベント「GU_ズーム」を使う理由

イベント: 1- メインプログラム

順	名前	トリガタイプ	トリガ	パラメータ	強制終了
1	GU_一覧	N=なし		0	E=編集
2	GU_実行N	N=なし		0	N=なし
3	GU_実行E	N=なし		0	E=編集
4	GU_ズーム	I=内部	ズーム(Z)	0	E=編集

メインプログラムのイベント
テーブルで定義。
1箇所の定義でAP全体に有効。

フォーム上のエディットコントロール
※ 入力途中では、まだ
項目に値が反映されて
いない。



F5キーを押す → ズーム アクションがトリガとなり、GU_ズーム イベントが発行される。

「強制終了」が E=編集 の場合には、まず項目に入力途中の値が反映される。

No.	商品#&Sub#	名称/型式/メディア/物流
5	482345	dbMAGIC V8 クライアント実行
	000007	JMG80-R-001
		CD-ROM
		2:MCC

入力途中のデータを項目に反映させるため、強制終了 E=編集 のユーザ定義イベントを介在させる必要がある。
(そうしないと、入力途中のデータが反映されないまま、以前の値がコールコマンドのパラメータに渡される)

データビュー ログック フォーム

21	日	E=イベント	GU_ズーム	コト	商品#	ズーム	T=タスク
22	コト	P=プログラム	567	SO選択	商品一覧	[3	パラメータ]

その後ハンドラを実行



まとめ



R Mの問題点

- キーボードのみで操作していた時代にはR Mでよかったが、G U I 全盛のいまどきでは制御が複雑である。

タスク定義: 718.1 - SO入力 受注情報1 (597) SO受注明細

#	処理コマンド	内容	範囲	位置付	加-	条件
1	録外 V=変数 :	1 V.商品DLRC	代入	0 0 0	0 0 S C	Yes
2	録外 V=変数 :	2 V.物流拠点省略値DLRC	代入	0 0 0	0 0 S C	Yes
3	録外 V=変数 :	3 V.物流拠点(開始)DLRC	代入	0 0 0	0 0 S C	Yes
4	録外 V=変数 :	4 V.物流拠点(終了)DLRC	代入	0 0 0	0 0 S C	Yes
5	録外 V=変数 :	5 V.別列DLRC	代入	0 0 0	0 0 S C	Yes
6						
7	録外 R=実行:	1 受注#	代入	29 29 29	0 0 S C	No
8	アクション :	1 KINPUT ('次項目'ACT)			戻 ?? S F	?
9	録外 R=実行:	2 受注月別割	代入	24 0 0	28 0 S C	Yes
10	アクション :	1 P.リスト = 'C' OR P.			C C C	?
11	録外 P=実行:	568 SO選択 商品一覧	0.5	8 24 0	0.0 No R C	Yes
12	録外 R=実行:	3 商品#				
13	録外 R=実行:	4 商品Sub#				
14	リスト O=照会 :	16 商品				
15	録外 R=実行:	1 商品#				
16	録外 R=実行:	2 商品Sub#				
17	録外 R=実行:	9 規格_型番				
18	録外 R=実行:	7 商品名				
19	録外 R=実行:	11 行/行				
20	録外 R=実行:	13 単価				
21	録外 R=実行:	16 原価_CDROM_輸入				
22	録外 R=実行:	22 原価				
23	録外 R=実行:	28 商品タイプ				
24	録外 R=実行:	29 商品タイプ2				

タスクイベント: 718 - SO入力 受注情報1 (597)

#	ホットキー	アクション	経過時間	式	スコープ	コール			
1		ユーザアクション1	00:00:00	0	T=タスク	P=アクション	626	1	No
2		ユーザアクション2	00:00:00	0	T=タスク	P=アクション	620	2	No
3		ユーザアクション3	00:00:00	0	T=タスク	P=アクション	619	2	No
4		ユーザアクション4	00:00:00	0	T=タスク	P=アクション	618	2	No
5		ユーザアクション5	00:00:00	0	T=タスク	P=アクション	616	3	No
6		ユーザアクション6	00:00:00	0	T=タスク	P=アクション	621	2	No
7		ユーザアクション7	00:00:00	0	T=タスク	P=アクション	561	1	No

レコードメイン

- どこでいつ何が起るのか、見通しが悪い。

タスクイベントテーブル

- プッシュボタンの動作はイベントテーブルに記述
- どのボタンで何が呼び出されるかわからない。





Thank You!

magicsoftware.com

