フレームワークによる開発

for .NET Framework

Lv3くまーbyやじゅ

はじめに

実はどのように開発するのが良いのかよく分からない

これは、外部の開発者と接する機会が少ないことと、
Google先生でも教えてくれない為、独自の開発方法
となり、いわば井の中の蛙状態となってしまっている。

ともかく今の開発方法を公開した上で、みなさんの
知識の情報を収集しようという作戦です。

※本番では印刷資料と説明内容や説明順序を変更する可能性があります。

自己紹介

やじゅ＠静岡の田舎　http://blogs.wankuma.com/yaju/

ＳＬ(大井川鉄道)が通っているところに住んでいます。

セッション

- 「ドラえもんの世界をオブジェクト指向で」
  2月23日　わんくま同盟東京勉強会＃17
- 「設計時の見落とし　Google先生も教えてはくれない」
  3月29日　わんくま同盟大阪勉強会＃17

説明手順

1. 開発仕様について
2. フレームワークについて

開発仕様

- 開発内容　（仮定）わんくま販売管理システム
- モジュール構成について
- アプリケーション・ドメインについて
- フォルダ構成について
- 開発方法について

  バージョンの付け方ってどうしてます？

開発内容

（仮定）わんくま販売管理システム

- **開発規模**

  プログラム本数　１００本

  ＤＢ Oracle10g　　**８０テーブル**

- **開発ツール**

  Microsoft Visual Studio 2005 VB.NET2005

  サードパーティ製

  入力コンポーネント、グリッドコンポーネント、

  帳票作成コンポーネント

モジュール構成

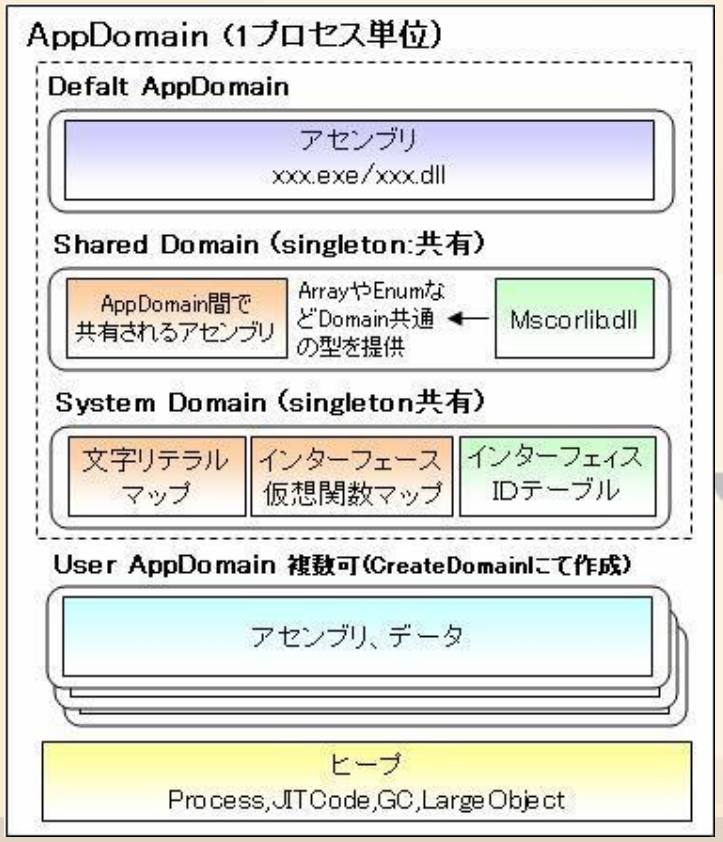
メニュー画面(ログイン入力画面を含む)のみ
ＥＸＥ形式、それ以外はＤＬＬ形式とする。

１画面(機能)で１プロジェクトとはせずに
サブシステム単位でプロジェクトを分割する。

※ここで言うプロジェクトは、ソリューションファイルを指す。

※タスクスケジューラで動作させる処理は、EXE形式　または　BATファイル＋SQLPlus(ストアド実行)　とする。
単独で動作させたい処理は、別途EXE形式で作成し、その中で該当するDLLを呼ぶ形式で作成する。

アプリケーション・ドメイン



．ＮＥＴアプリケーションでは１プロセスの中に型やセキュリティを管理する単位として、ＡｐｐＤｏｍａｉｎという器を作成する。

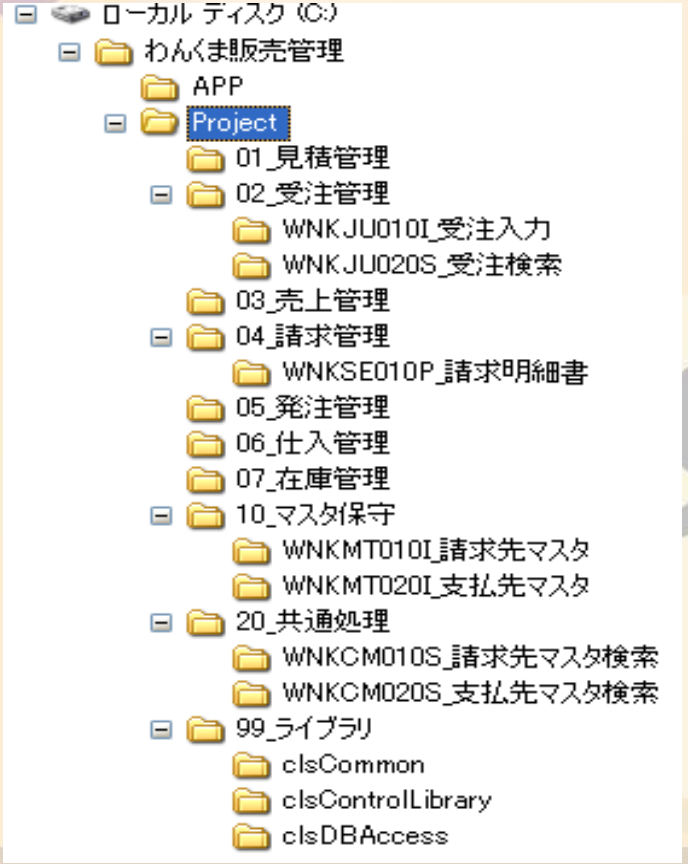
ＥＸＥ形式では、１プロセスとなるため、ＡｐｐＤｏｍａｉｎを作成することになり起動コストが高くなる。

ＤＬＬ形式では、ＡｐｐＤｏｍａｉｎ内にＤＬＬファイルが読み込まれ実行する。

注意点

ＤＬＬのアンロードはＡｐｐＤｏｍａｉｎ単位となるため、違う種類のＤＬＬを読み込めば、それだけメモリが増大する。

フォルダ構成

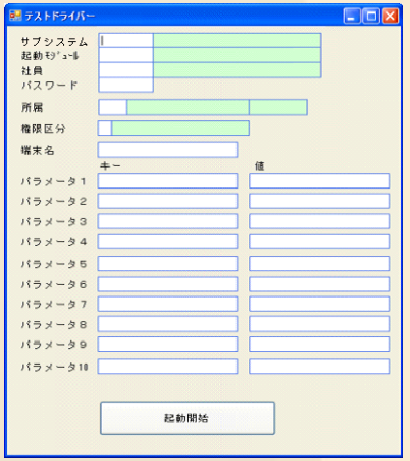


APPフォルダの中身

- wnkMenu.exe
- wnkMenu.exe.config
- wnkMitusmori.dll
- wnkJuchu.dll
- wnkUriage.dll
- wnkSeikyu.dll
- wnkShire.dll
- wnkZaiko.dll
- wnkMaster.dll
- wnkCommon.dll
- clsCommon.dll
- clsControlLibrary.dll
- clsDBAccess.dll
- 各サードパーティ製.dll x n本

開発方法

- 開発用にｄｌｌを起動するドライバを作成
  - 外部プログラムの設定に指定
  - コマンドラインに起動内容を設定



- バージョンの付け方について
  - <メジャー・バージョン>.<マイナー・バージョン>.<ビルド番号>.<リビジョン>
  - 開発時のアセンブリバージョン、ファイルバージョン

フレームワーク

- 開発目的
- 共通仕様を決める
- フォーム、入力コントロールの継承と拡張
- 抽象化を理解
- ポリモーフィズム
- 前処理・本処理・後処理
- 入力チェックの統一化

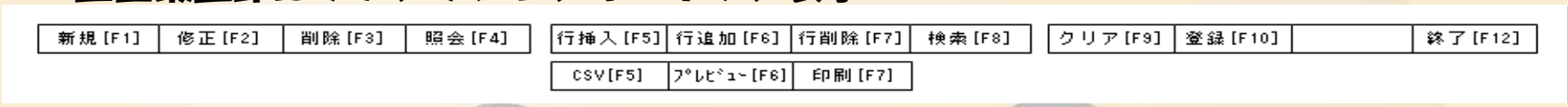
開発目的

・共通機能を内包することによりアプリケーション側のコードの絶対量が削減され、製造者の負担を減らす。
コードの統一化により、開発者のスキルの差が少なくなり、品質をある程度均一されることにより信頼性が向上する。

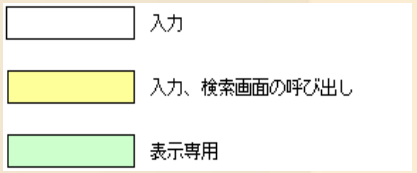
・型にはめる方が楽
自由すぎると思わぬ不具合を引き起こす可能性が高い
型にはめてしまう事で、中身の製造に専念させる。

共通仕様を決める

・画面最上部にファンクションキーボタン表示



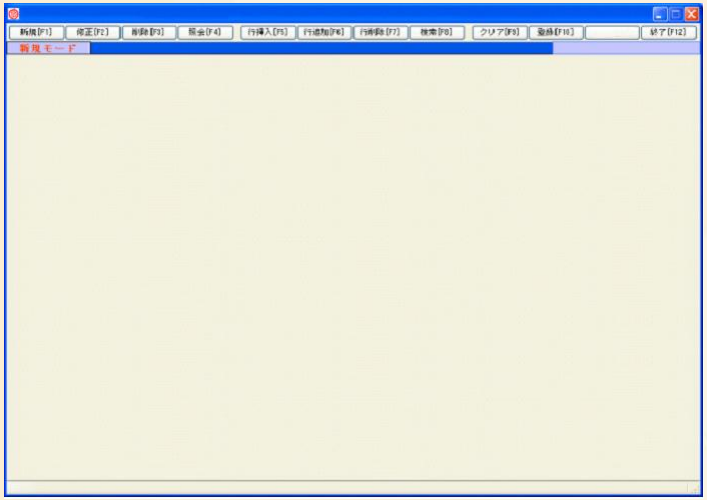
・Ｅｎｔｅｒキーで項目移動
・○○時(登録など)、再度、項目移動のチェック＋全体の整合性チェック
・項目移動で内容に変更なしなら、入力チェックはしない。
・背景黄色の項目は検索画面を呼び出し可能(Ｆ８キー連動)
・背景緑色の項目は表示専用



・入力項目を編集したなら、クリア時・終了時に確認メッセージを出力
・検索項目ならＦ８キーを有効化、それ以外は無効化
・グリッドコントロールならＦ５～Ｆ７キーを有効化、それ以外は無効化
・行挿入、行追加が最大行数に到達ならＦ５、Ｆ６キー無効化
・行削除が最低行数ならＦ７キーを無効化

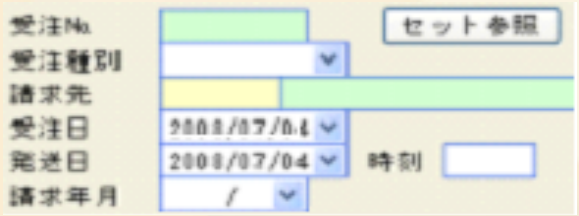
などなど

フォーム、入力コントロールの継承と拡張



・ファンクションキーを配置、モード表示ラベル、
タイトル表示ラベル、ステータスバー

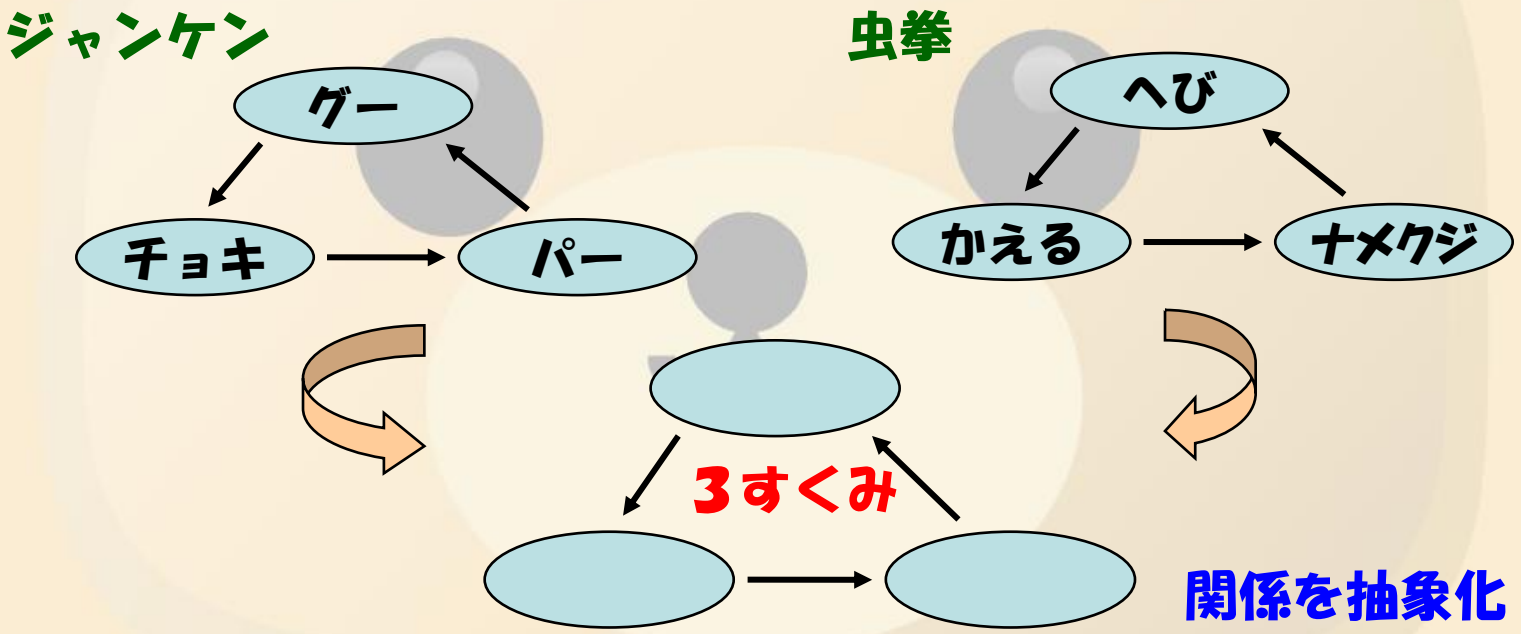
・フォームの拡張プロパティ
編集フラグ、処理モード、フォームタイプ
ファンクションキー表示、検索画面の結果受取
検索保持コントロール、グリッド保持コントロール
などなど



・入力コントロールの拡張プロパティ
拡張Ｅｎａｂｌｅｄ、検索呼出
値変更通知、全チェック、項目値保持
などなど

抽象化を理解

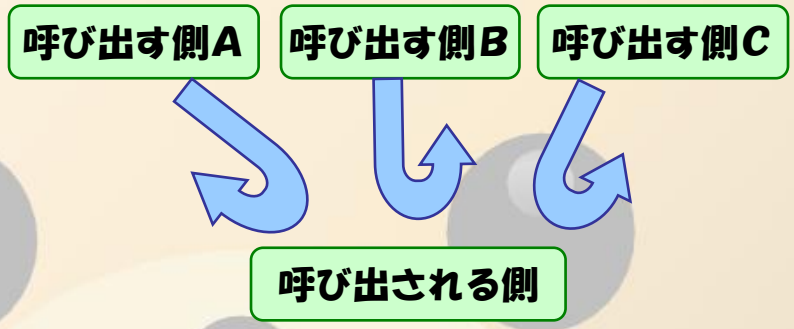
関係・機能を抽出し、システム化したもの



ポリモーフィズム

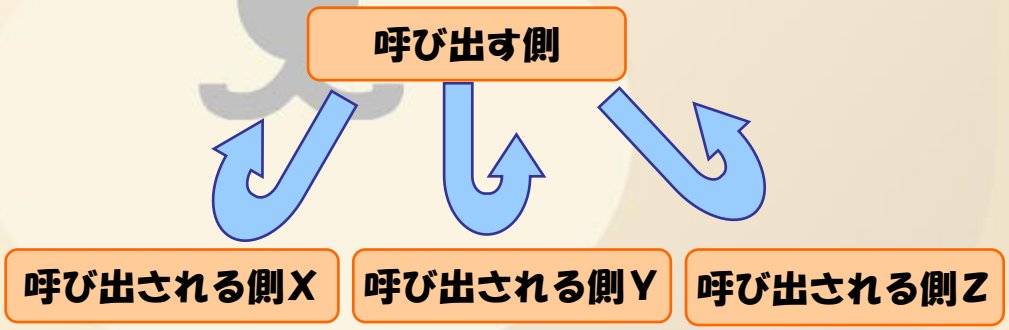
共通サブルーチン

呼び出す側が増えても
呼び出される側を修正
する必要がない



ポリモーフィズム

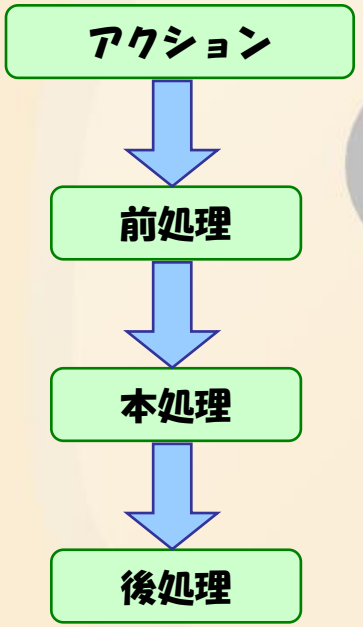
呼び出される側が増え
ても、呼び出す側を
修正する必要がない



オーバーライド(再定義)によって、中身を書き換える。

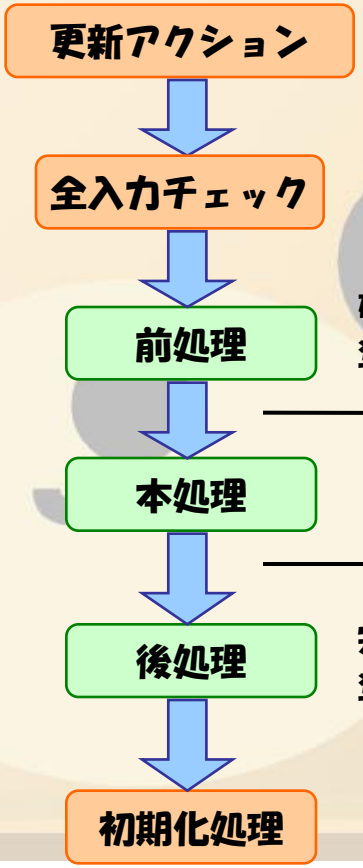
前処理・本処理・後処理

基本の考え方



継承元フォームにある程度記述し
必要なら、継承先にて書き換える。

例



各処理で異常なら次の処理へは
進まないで、処理を抜ける。

確認メッセージを出力
登録します。よろしいですか？

→ トランザクション開始

正常ならコミット
異常ならロールバック

完了メッセージを出力
登録が完了しました。

入力チェックの統一化

・各入力コントロールの入力チェック処理を、統一の入力チェックメソッド
ＩｎｐｕｔＣｈｅｃｋ　に統一する。
テキスト型、マスク型、数値専用型、日付型、コンボボックス型のＶａｌｉｄａｔｉｎｇ
チェックボックス型、ラジオボタン型のＣｈｅｃｋＣｈａｎｇｅｄ