

.NetFramework アプリケーション開発支援ツール
LLL/.net 概要ご紹介資料

LLL/.net は、こんな製品です



株式会社パーシモンシステム

使いこなしていただくために・・・

スマートクライアント対応.NetFramework アプリケーション開発支援ツールであるという LLL/.net とは、どのような仕組みで何をするものなのか、開発支援ツールと言うが、具体的に何をどう支援するのか、これを使用する場合としない場合では何が違うのか？

本書は、これから LLL/.net を使用する、また導入を検討しようとする方へ、LLL/.net の基本的な構造や考え方、特徴をご説明します。本書だけで LLL/.net の全てが理解でき、ただちに開発が行なえる訳ではありませんが、本書に目を通していただくことで、一通りの下知識をお持ちいただけ、マニュアルや関連資料類、プロダクトの理解力が深まります。

[はじめに - LLL/.net 導入成功の決め手]

マイクロソフト社から無料で入手できる.NetFrameworkSDK には、コモンランゲージランタイム (CLR)、クラスライブラリの他、ドキュメント、サンプル、ツール、コマンドライン コンパイラが、付属しています。すなわちこれだけで.Net アプリケーションは開発が可能であり、有償ツールである VisualStudio も必須の物ではありません。

であるにも関わらずどうして VisualStudio と、さら LLL/.net まで必要なのでしょうか。

開発ツールの目的はいくつかありますが、キーワードとして「開発作業効率の向上」、「品質の安定」、「標準化の推進」「メンテナンス容易性の確保」などが良く挙げられます。

VisualStudio の目的もそこにあります。さらにそれと組み合わせたいいただく LLL/.net は、その目的を一層強く実現するための手段です。それを徹底的に行ないたいのであれば LLL/.net が必要です。

いわゆる IT プロと言われる方のように、自分の範囲で計画を決定し、自らが実装も行なわれる場合、ツール導入もご自分が決断できれば良いだけであまり問題にならないのですが、チームで開発を行なうプロジェクトの場合、なかなか導入を決断できないことがあります。

どのようなツールでも、そのノウハウ習得に少なからず時間を要します。これは事実です。納期が決まり、しかもそれが差し迫ったプロジェクトにおいて、期日どおりの納品に不安を持ったリーダーの方が、急遽探し出したツールの採用を実装技術者に相談すると大抵の場合反発されます。ツールの良し悪しを議論する時間すら勿体無いからです。

それと、判断の責任を負わされるのは負担です。

多くの技術者の方は、型にはめられることをできれば避けたいと考えます。技術者のプライドもあるでしょう。できれば自由に好きなスタイルで開発を行ないたいものです。

このこと自体は、向上心のある技術者であるほど、自然な考えであり決して悪いとも言えません。しかし業務として請け負った開発の仕事においては、当然のことながら同時に採算性や品質の確保、そしてスピードがより重要になります。

標準化とは少なからず型に嵌められるものです。多少不自由な面が出ることは否めませんが、そのことによって作業効率の向上と品質の安定が図れます。

ツールの採用はプロジェクトの方針を決めるリーダー、マネージャーの方が、強い決断力と意志を持って行なうことが重要です。できるだけ早い時期に採用を決めていただき、準備を整えた上でツール利用の徹底を図っていただくことが、技術者の方にも物理的、精神的に与える負担が少なく、ツール採用の成功への決め手になると思われます。

[LLL/.net のプロフィール]

LLL/.net は、VisualStudio と連携して使用する .NetFramework アプリケーションの開発支援ツールです。

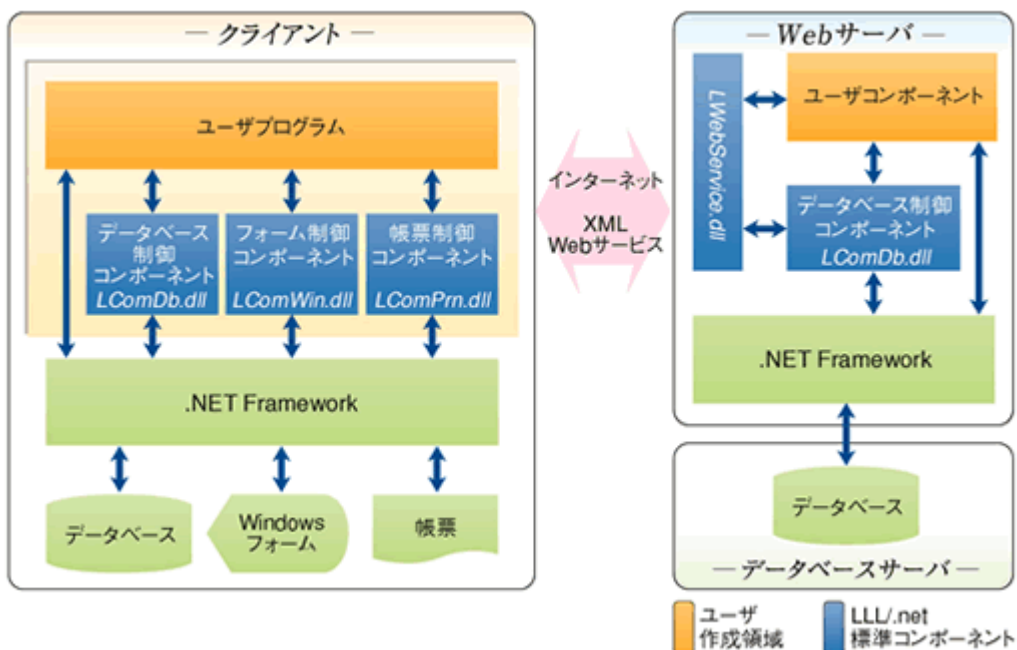
20 年一貫して、優れたユーザビリティの実現を特徴としてきた、パーシモンの業務アプリケーション開発ツール製品シリーズの最新版で、開発効率の向上と標準化の推進という相反するテーマを同時に解決します。最新技術を容易に取り込むことができ、短期間少人数での開発にも威力を発揮し、多くの IT プロの方々からも支持を得ています。新規開発はもちろん既存 Access システム等のマイグレーション目的にも多く利用されています。

[LLL/.net の基本構造]

LLL/.net は、業務用アプリケーションで必須となる画面制御、DB 制御、及びスマートクライアント構築に欠かせない Web サービスによる通信機能などを、それぞれの汎用機能集であるコンポーネント（クラスライブラリ）として提供します。

このコンポーネントが LLL/.net の製品の中核になります。そしてコンポーネントのクラスやメソッドを使用するプログラムが、LLL/.net アプリケーションであると言えます。

■ LLL/.net アプリケーション構造図



LLL/.net アプリケーションのコーディングの雛型がプログラムタイプ別に用意されており、それをテンプレートと呼びます。テンプレートは VB または、C#でコーディングされた VisualStudio の Windows アプリケーションプロジェクトファイルとして提供されます。そのテンプレートに対して、作成したい目的のプログラム固有のフォームデザインやアクセスする DB のテーブル情報、さらにはアプリケーションロジックの機能を追加していき、さらにそれを VisualStudio でデバッグしてプログラムとして完成させるというのが、LLL/.net アプリケーションの開発手順です。そしてこれらの作業を一連の流れとし、また対話型操作で実現するための専用の統合環境が「LLL/.net プログラム設計」として用意されています。一般的に LLL/.net というところの統合環境をイメージされるケースも多いですが、実際にはコンポーネント、テンプレート、統合環境の 3 つを基本構成とする製品です。

[LLL/.net アプリケーションの特長]

LLL/.net は、スマートクライアント型アプリケーションの構築に威力を発揮します。

できあがった LLL/.net アプリケーションは、操作性の良さ、高速応答性、ローカルリソース活用、プログラム配布容易性、インターネット越しの通信、DLL 非競合、そしてオフライン稼動などを特長として持ちます。これらはシンクライアントとファットクライアントから良いところ取りをしたと言われるスマートクライアントそのもの特長でもあります。ただし、スマートクライアントは、特長として羅列した前述のような機能を実装できる仕組みのことであり、そのように実装しなければそうはなりません。

LLL/.net は、そのような特長をスマートクライアントとして具体的に実装する仕組みです。

LLL/.net で構築するスマートクライアントは、Windows フォーム+Web サービスによって構成されます。もちろんクライアント/サーバー型システムも作成できます。

データベースアクセスには ADO.NET を使用し、セッションの維持を前提としない非接続型の DB アクセスで、整合性のとれたトランザクションを実現します。これは実際にインターネット越しのアクセスを使用するの否かによらず、たとえスタンドアロン運用のシステムの開発であっても全て自動的にそういう作りになります。

さらにスマートクライアントの実現に欠かせない、Web サービス経由の DB 利用機能、サーバーからクライアントへのプログラムの配信や、自動更新、及び配信状況のロギングといったデプロイメント機能も LLL/.net が専用の仕組みを既に実装済みです。

したがって、アプリケーション開発者の方は、インターネット、SOAP、XML、などを意識してプログラムを作成していただく必要はありません。アプリケーションの運用形態が、スマートクライアント型か、クライアント/サーバー型かは、統合環境「LLL/.net プログラム設計」での若干の設定が異なるだけで、プログラムの構造や作り方は変わりません。

[LLL/.net でのプログラム開発手順]

LLL/.net で実際にプログラムを開発していく手順を説明します。

1. まず最初に、LLL/.net の統合環境に、プログラムで使用するデータベースの接続情報を入力し、テーブル情報を読み込ませます。テーブルは DB 側で予めご用意ください。使用できる DB は SQL-Server、ORACLE、JET の 3 種類です。
2. 読み込んだテーブル情報に、日本語表示名称などを追加登録して辞書化します。

テーブルカラム名(F3)	英字カラム名	表示カラム名	データ型	サイズ	配列	表示フォ
コード						
BUMON_CD	BUMON_CD	部門CD	Int32			00
CHIKU_CD	CHIKU_CD	地区CD	Int32			00
HINSYU_CD	HINSYU_CD	品種CD	Int16			00
KAIKYU_CD	KAIKYU_CD	階級CD	Int16			00
SHOHIN_CD	SHOHIN_CD	商品CD	String	4		
SIR_CD	SIR_CD	仕入先CD	String	4		0000
TANTO_CD	TANTO_CD	担当CD	Int16			00
TOKUI_CD	TOKUI_CD	得意先CD	String	4		0000
TORISAKI_CD	TORISAKI_CD	取引先CD	String	4		0000
ZELCD	ZELCD	消費税CD	String	4		0000
名称						
BUMON_NM	BUMON_NM	部門名	String	20		

プロパティ(F4)	値
テーブルカラム名	HINSYU_CD
英字カラム名	HINSYU_CD
表示カラム名(日本語)	品種CD
データ型	Int16

3. 開発するプログラムのタイプにあわせ、フォームテンプレートを選択します。テンプレートが必須とする条件、例えば伝票形式の入力プログラムであれば、使用するヘッダーテーブルと明細テーブル等を指定します。

テンプレート	説明
PAA100	フリーフォーマット
PDA100	単票形式データ入力
PDA200	明細伝票形式データ入力
PDA300	明細伝票形式データ入力(頁制御)
PGA100	明細一覧表示
PGA200	明細一覧表示(ListView)

パラメータ	値
フォームタイトルの設定	売上伝票入力
データベースの選択	DbMain
メインテーブルの選択	DENHEAD
明細テーブルの選択	DENMEI
明細カラムのデータ選択	0~の連番

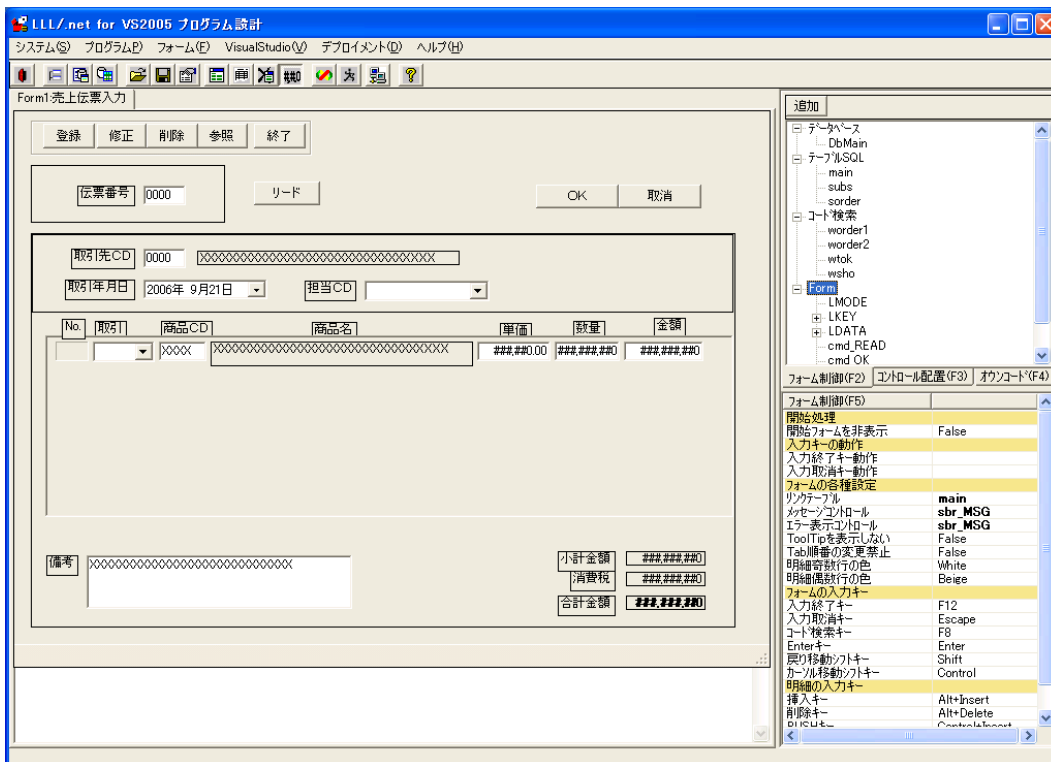
(注意)プロジェクトにクラスを追加します。取消しはフォーム削除して下さい。

レートの必須とする条件、例えば伝票形式の入力プログラムであれば、使用するヘッダーテーブルと明細テーブル等を指定します。



4. 統合環境上でテンプレートが必須とするテーブルをアクセスするためのSQL文(テーブルSQL)が自動的に初期生成されていますので、必要に応じてこれを修正します。

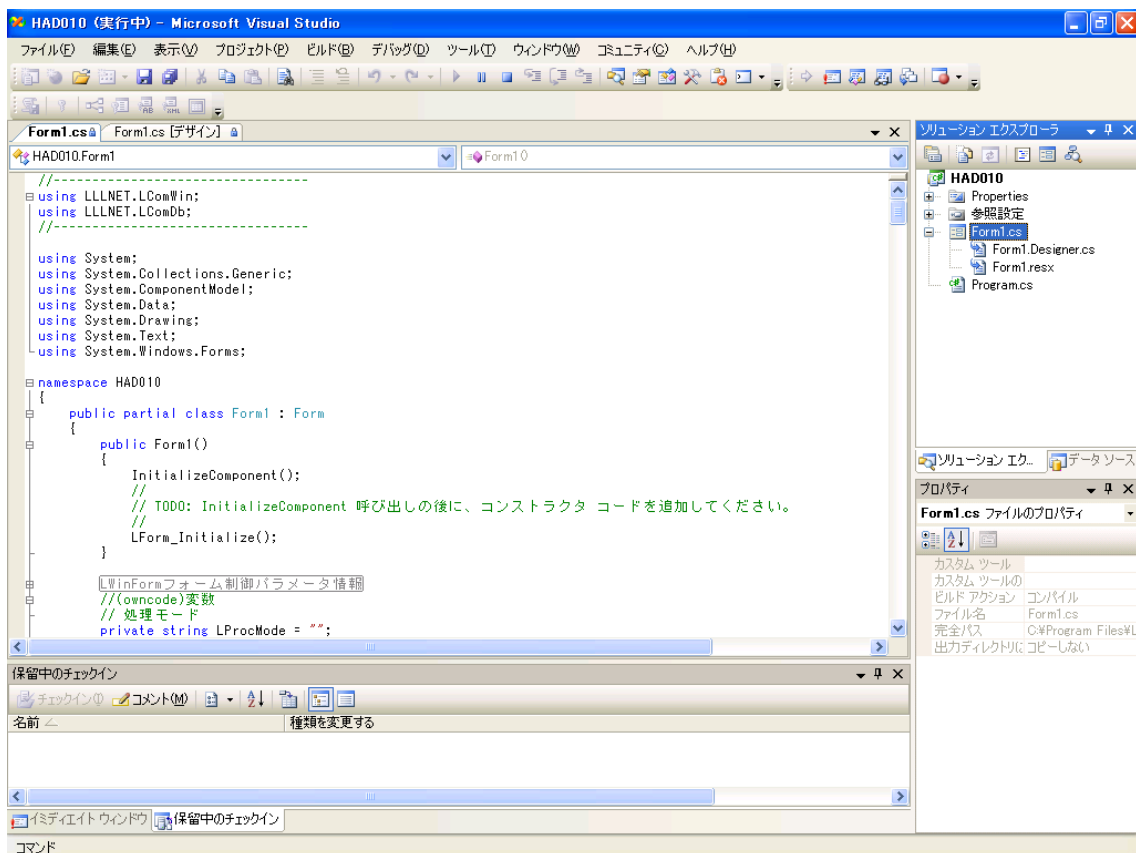
5. 表示されたテンプレートの雛形フォームにテーブル情報からカラムをドラッグ&ドロップで配置していきます。必要に応じて他のテーブルの参照設定や、リストコントロールの表示設定なども対話型で行います。



6. フォーカス制御や、挿入／削除、などの明細制御、その他フォーム制御の基本動作に使用するキーの割付などは、デフォルトで一通り設定されていますが、ここで好みの割付に設定変更することもできます。



7. LLL/.net の統合環境から VisualStudio を起動すると、ここまでに行った設定などの作業を反映させた新しい Windows アプリケーションプロジェクトが立ち上がります。



8. テンプレートから作成されたプロジェクトは、基本的な機能をコードとして実装済みです。そのままデバッグモードで実行すると、すでにプログラムは動作します。

No.	取引	商品CD	商品名	単価	数量	金額
1	売上	A001	アーモンドカレー 中辛	300.00	2	600
2	売上	A002	カップラーメン 味噌醤油	100.00	3	300
3	売上	A003	ソーセージ M3本束	200.00	2	400
4	売上	A004	えびシューマイ 20個入り	500.00	2	1,000
5	売上	B001	シャンプー ポンプ式	1,000.00	2	2,000
6	売上	B002	浴用石鹸 3個組み	250.00	4	1,000

備考 配送先: 東京都千代田区神田2

小計金額 15,600
消費税 780
合計金額 16,380

9. ここから先は、必要に応じて VisualStudio を使ってフォームや、コードの修正、追加を行ないプログラムを完成させます。

10. VisualStudio を保存して終了すると、LLL/.net の統合環境に戻ります。

VisualStudio 側で行なった修正は、改めて LLL/.net 統合環境から見ても反映されています。さらに必要であれば、また LLL/.net で修正を行なえます。必要であればこのように何度でも LLL/.net と VisualStudio の間で行き来して開発作業を行なえます。

11. クライアント/サーバー型の開発手順はこれで終わりです。スマートクライアント型の開発の場合は、配布元となる Web サーバーへのアクセス情報、Web サーバーからクライアントに配布すべきプログラムやファイルの一覧、起動時に最初に立ち上がるプログラムなどを指定したダウンロードリストの作成という作業が最後に加わります。この部分の機能は「スマートクライアント デプロイメント」として後述します。

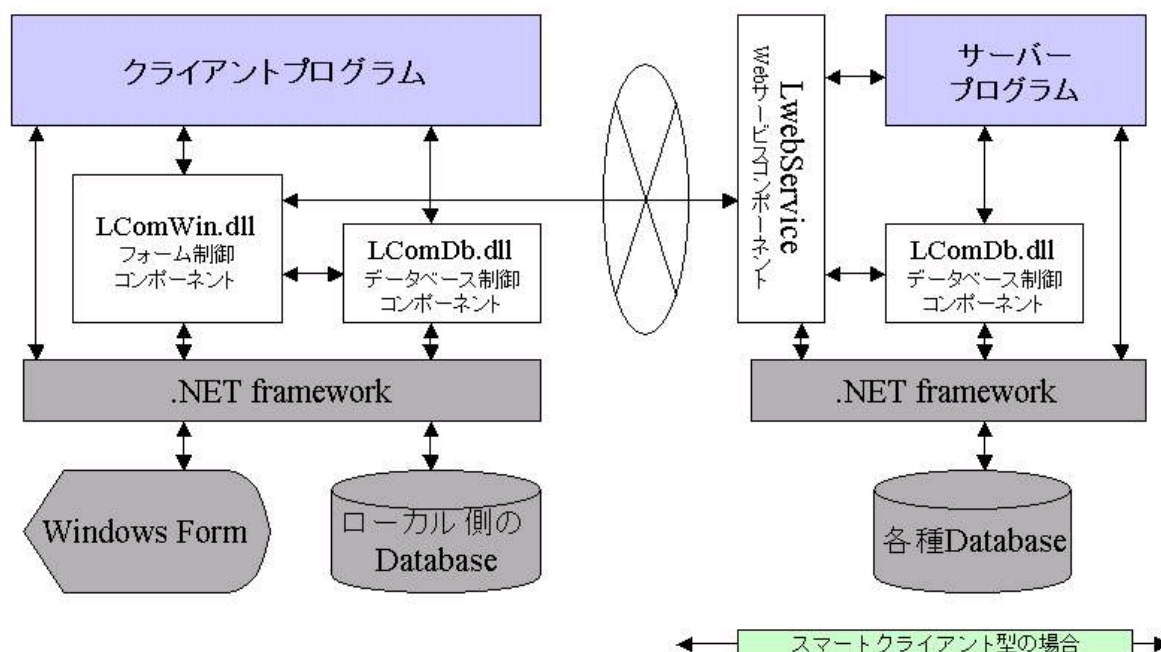
「LLL/.net アプリケーションコードの構造 (技術情報)」

前章では、開発作業の大まかな流れをご説明しましたが、ここではもう少し技術的な仕組み、LLL/.net アプリケーションがコンポーネントのクラスやメソッドを使用する方法、構造を説明します。

LLL/.net が標準提供するコンポーネントには、次の4種類があります。

1. フォーム制御コンポーネント LComWin.dll
2. データベース制御コンポーネント LComDb.dll
3. 帳票制御コンポーネント LComPrn.dll (※オプション)
4. Web サービスコンポーネント LWebService.dll

このうち、LComPrn.dll はオプションです。この紹介は別の機会にします。



残りの3つのコンポーネントの中で、アプリケーション開発者の方に特に意識していただく必要があるのは1. のフォーム制御コンポーネント (LComWin.dll) です。

コンポーネントのクラスのメソッドを使用するプログラムがLLL/.net アプリケーションであると前述しましたが、具体的にユーザープログラムのコーディングに現れるのは、ほとんどがフォーム制御コンポーネント (LComWin.dll) のメソッドです。

他のコンポーネントが不要なのではなく、それらはフォーム制御コンポーネント

(LComWin.dll) が内部から呼び出して使用しますので運用上は必要なものです。

※ 本章のこれ以降と次章「テンプレートの作成」は技術情報です。コーディングレベルでの構造を知りたい方以外は読み飛ばしてください。ソースリストがあれば併せてご覧ください。

テンプレートを用いて「新規作成」した Windows アプリケーションとフォーム制御コンポーネント (LComWin.dll) の連携は、LComWin.dll に定義されている フォーム制御オブジェクト LWinForm クラスを、アプリケーション内で自身(デフォルトでは class Form1) を引数としてインスタンス化することによって行ないます。また、ユーザープログラムからのフォーム制御オブジェクトのメソッドやプロパティの呼び出しは、LWinForm のインスタンスへの参照変数を介して行ないます。

では、実際のテンプレート記述されたコーディングの構造を説明しましょう。

各テンプレートの冒頭のコンストラクタの直後には、まずリージョン句が存在します。

```
#Region ~ #End Region <VB>、 #region ~ #endregion <C#>
```

この中身は、予めテンプレートにコーディング済みか、または統合環境「LLL/.net プログラム設計」が設計情報に基づいて自動的に用意 (生成) した部分であり、通常は開く必要はなく、手で修正などを行なってはいけない部分です。

実はここには、ユーザープログラムと、フォーム制御オブジェクトの連携に必要な準備が施されています。この記述が変更されると動作に支障を来す恐れがあります。

ここでは、LLL/.net アプリケーションの構造を理解していただくために、あえて通常は意識する必要のない部分ですが、意味を説明しておきます。

リージョン句の先頭は、LWinForm 型の変数 LForm の宣言です。

```
Public LForm As LWinForm <VB>、 public LWinForm LForm; <C#>
```

この後、

```
Private Sub LForm_Initialize() <VB>、 private void LForm_Initialize() <C#>
```

というプロシージャが続きます。

その中で、統合環境「LLL/.net プログラム設計」で定義したフォーム制御情報に従い、フォーム制御パラメータを自動的に生成して展開していきます。まず、最初は、

```
LForm = New LWinForm(Me) <VB>、 LForm = new LWinForm(this); <C#>
```

とあり、前述の通り VB では、Me、C#では This キーワード、すなわち自身（デフォルトでは class Form1）を表す代名詞を引数としてフォームオブジェクトの生成（インスタンス化）が行なわれています。これによりアプリケーションの Form クラスと、LWinForm のインスタンスがリンクされます。そして、先に定義した変数 LForm に LWinForm のインスタンスへの参照を代入することによって、前述の通りアプリケーション内での LWinForm オブジェクトの各メソッドの呼び出しは、

```
LForm. GetDataTable (DtMain) <VB>、 LForm. GetDataTable (DtMain); <C#>
```

のように、 LForm. メソッド名 という書式になります。

その後には、統合環境「LLL/.net プログラム設計」で定義、作成された情報の展開が続きます。LForm.SetDatabase()、LForm.SetTable()、LForm.SetWin()、LForm.SetForm()、LForm.SetField()、LForm.SetArea()、LForm.SetRow()、のように Lform.Set×××メソッドの羅列がそれです。これによってフォーム制御オブジェクトが、テーブルやフォームを制御できるようになります。ここの修正は「プログラム設計」で行ないます。

これら一連の定義により、Form 上で発生したイベントは、フォーム制御オブジェクト LwinForm のインスタンスがハンドリングできるようになります。

ハンドリングされたイベントは、フォーム制御オブジェクト内で、適切な処理が行われた後、必要に応じて加工され、フォーム制御オブジェクトイベント LWinEventArgs を発生させます。これをアプリケーション側で処理するために用意されたコードが下記です。

ここまでが折りたたまれたリージョン句です。この中は手作業で改変しないでください。

```
AddHandler LForm. LWinEvent, AddressOf LForm_Event <VB>  
LForm. LWinEvent += new LWinForm. LWinEventHandler (LForm_Event); <C#>
```

LWinEvent はデリゲートです。これに上記コードで LForm_Event を追加登録することで、フォーム制御オブジェクトでイベントが発生した時に、LForm_Event がコールバックされる仕組みができています。LForm_Event をLLL/.net では「イベントエントリ」と呼びます。フォーム制御オブジェクトイベント LwinEventArgs をディスパッチし、適切な処理を行うために、テンプレートには LForm_Event が必須となります。

これ以降に記述されているのが実際のプログラムの処理ロジックです。ここは必要に応じて改変可能です。

前述にとおり「イベントエントリ」LForm_Event では、発生した LwinEventArgs の種類によって処理を分岐し、それぞれのイベントに応じたハンドラに処理をディスパッチします。その他の処理ロジックの構成はテンプレートの目的によって当然異なる部分もありますが、

LLL/.net のフォーム制御オブジェクトを使用するという構造上、基本的な動作に関わる部分は共通であることが多くなっています。

テンプレートのプログラムを構成するプロシージャの内、「変数」「イベントエントリ」「フォーム開始処理」「フォーム終了処理」「入力データの正当性チェック処理」「コントロールクリック処理」「その他」は、フリーフォーマットを含む、全ての標準テンプレートに共通に存在し、予め用意されているプロシージャです。「その他」は空のプロシージャです。

テンプレートの代表格である、「PDA200 明細伝票形式データ入力テンプレート」には、前述のものに加え、「データエリア計算」「明細行エリア計算」「各処理モードの設定」「各入力モードの設定」「データリード処理」「登録更新処理」「修正更新処理」「削除更新処理」「データベース更新」といったプロシージャが追加されています。

その中に、入力状態によるエリア単位のフォーカスロック／アンロック制御、フォームとデータベース連携による登録、修正、削除、参照、またボタン操作により処理モードの切替を行う機能など、伝票入力プログラムとして動作するための必要最低限の処理コードは実装済みです。

さらにもし、この伝票が売上传票入力プログラムであれば、得意先コードによって得意先マスターから得意先名を参照して画面表示をしたり、明細行においては数量と単価を掛け合わせて結果を金額欄に表示させるなどの機能が求められますが、このテンプレートは売上传票のみを前提にしたテンプレートではなく、そのような機能は実装していません。

これらの機能は、前者については統合環境「LLL/.net プログラム設計」でのテーブル参照情報の追加設定で対応し、後者については「明細行エリア計算」プロシージャに対する追加コーディング（オウンコーディング）で対応します。

ちなみに、コントロール間のフォーカス移動や、行挿入、行削除のような明細行制御など、業務処理の流れとは直接関係無く発生する画面制御やデータベース接続、テーブルアクセス、及び Web サービスなどの基本機能はテンプレートのコーディングではなく、コンポーネントが内部で実装しています。テンプレートは、コンポーネントが持つ単独の機能を一連の流れの中で呼び出し、意図を持ったプログラムとして動作するようにした処理ロジックの雛形です。コンポーネントと、テンプレートはこのように役割を分担しています。

テンプレートの実体は、実装済みの VisualStaio のプロジェクトです。これらはテンプレートソリューションとしてまとめて保管されています。バージョン 2.00 ではプログラムパターン別に以下の 6 種類のテンプレートを標準提供します。

1. 「PAA100 フリーフォーマット」
2. 「PDA100 単票形式データ入力」

3. 「PDA200 明細伝票形式データ入力」
4. 「PDA300 明細伝票形式データ入力（頁制御）」
5. 「PGA100 明細一覧表示」
6. 「PGA200 明細一覧表示（ListView）」

それぞれのテンプレートの目的は、それぞれの名称が示すとおりです。

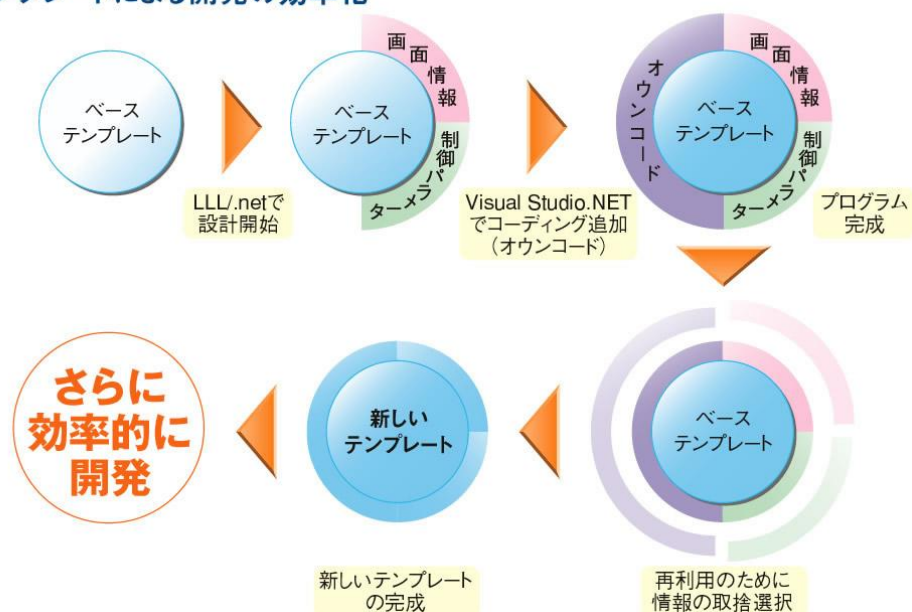
2. ～6. は、全てフォームと、データベースアクセス機能を伴うプログラムです。1. は、2. ～6. の元となった基本のテンプレートです。フォームは存在しますが、決まったデザインやロジックはありません。データベースのアクセスも必須ではありません。必要に応じて機能を追加コーディングして使用します。

ここでは、テンプレートの構造と考え方を説明しました。個々のテンプレートのロジックなど詳細については、マニュアルや「LLL/.net 開発環境評価手引書」をご参照ください。

「テンプレートの作成（技術情報）」

LLL/.net の大きな特長のひとつとして、テンプレートの改変や追加が行なえることがあります。単純な例としては既に存在するテンプレートに必要な改造を加え名前を変えたものを保存すると新しいテンプレートが追加できます。テンプレートは開発ツールの一部を構成するものですから、言い方を変えると LLL/.net は、開発ツールのユーザーカスタマイズが行え、使い込むほどに使い勝手の上がる仕組みとなっているということになります。

テンプレートによる開発の効率化



当然ですが、テンプレートは抽象化が高いほど汎用性も高く、業務処理などのコーディン

グが具体的であるほど汎用性は低くなりますが、適用できれば開発効率は上がります。

これまで説明したテンプレートは、正確にはフォームテンプレート（フォームを伴うクライアントプログラム用）というテンプレートについてでした。LLL/.net にはこれ以外に、コンポーネントテンプレート（フォームを伴わず、主に Web サーバー上での動作を想定したプログラム用 => ただし、クライアントで動作するコンポーネントプログラムの作成も可能）と、帳票テンプレート（帳票エンジンを制御するプログラム用。※オプション）という異なる2つの種類のテンプレートが存在します。

これらも、フォームテンプレート同様に改造や自作して追加を行なえます。

テンプレートは、LLL/.net のインストールディレクトリに原本が VisualStudio のソリューションとして存在しています。新たに開発ディレクトリを作成する都度、インストールディレクトリから開発ディレクトリにテンプレートソリューションがコピーされ、個々のプログラム作成時には開発ディレクトリ上のテンプレートを使用して、新たなプロジェクトが作成されます。すなわちテンプレートは、システム単位、または開発ディレクトリ単位で管理することができます。

「LLL/.net プログラム設計」で「新規プログラム」を選択すると、テンプレートの選択パネルが表示されます。

ここでは、テンプレートを選択すると同時に、そのテンプレートの使用にあたり必須となるパラメータの入力を選択パネル上で行ないます。例えば「PDA200 明細伝票形式データ入力」テンプレートでは、「フォームタイトルの設定」、「データベースの選択」、「メインテーブルの選択」、「明細テーブルの選択」、「明細カラムのデータ選択」がそれに相当し、これらを入力、または選択するためのテキストボックスやドロップダウンリストが選択パネルに表示されています。

「LLL/.net プログラム設計」は、そこで選択されたテンプレートプロジェクトをコピーして新しい VisualStudio のプロジェクトを作成すると同時に、入力されたテンプレートパラメータの値を、テンプレートのフォームコードの中にある {*パラメータ ID} の文字列を置換えます。たとえば、`Me.Text = "{*title}"` はフォームのタイトル文字列を編集します。

しかしテンプレートを自作、または改造できると言いましたが、このようなテンプレートパラメータも、テンプレートによって必要なものは異なるはずです。

このテンプレートパラメータも、改造できなければなりません。

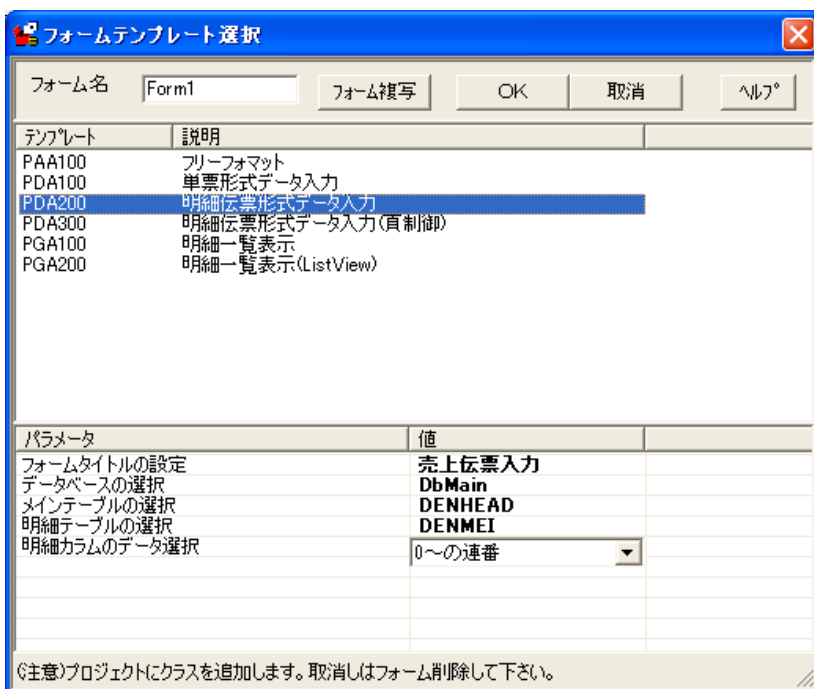
テンプレートパラメータは、標準テンプレートのフォームコードの先頭にコメントとして、以下のサンプルのように設定されています。（C#の場合、コメント記号は // ）

```

' [Template]
' Template#caption=明細伝票形式データ入力
' [Data]
' title#caption=フォームタイトルの設定|input=text
' main_db#caption=データベースの選択|input=combo|list={dbs}
' main_sql#caption=メインテーブルの選択|input=combo|list={tables}|sql=main
' subs_sql#caption=明細テーブルの選択|input=combo|list={tables}|sql=subs
' subs_row#caption=明細カラムのデータ選択|input=combo|list=0~の連番,1~の連番,明細フィールド

```

上記設定のテンプレートを選択すると、テンプレート選択パネル下段が、下のように表示されるようになります。



LLL/.net で作成したプログラム (プロジェクト) は、原本のテンプレートのプロジェクトをコピーしたものに、「LLL/.net プログラム設計」で定義したフォーム制御情報をフォーム制御パラメータとして、前述のリージョン句に展開する他、テンプレートパラメータで指定したパラメータもフォームコード中に展開します。ただし原本

テンプレートのコードに存在するこのコメント文は、作成した「新規プログラム」のコードにはコピーされません。また、テンプレートパラメータがユーザープログラムに展開されるのは、最初の「新規プログラム」作成時の1回のみです。「LLL/.net プログラム設計」と VisualStudio を何度行き来しても再度生成されることはありません。

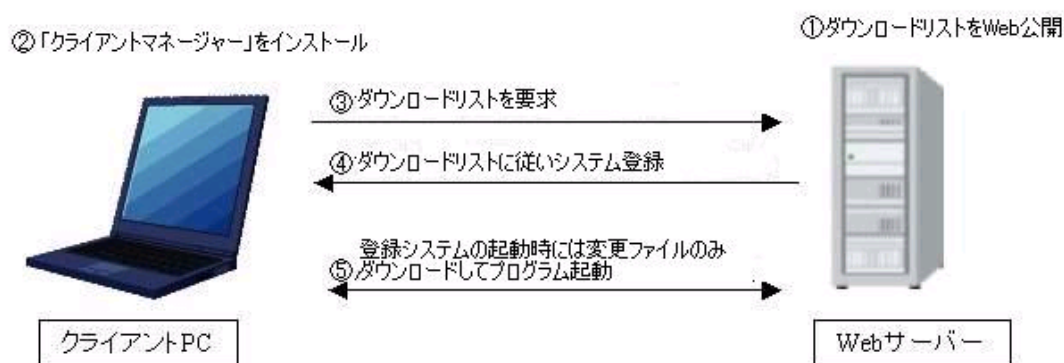
テンプレートの作成は、概ね下記のような流れになります。

- (1) フォームテンプレートの雛型となるプログラムを作成します。
- (2) テンプレートソリューションに、フォームを追加します。
- (3) 追加したフォームに、テンプレートパラメータを設定します

[スマートクライアント デプロイメント]

LLL/.net アプリケーションのデプロイメントは、スマートクライアントで良く知られたノータッチデプロイメントや、ClickOnce ではなく、専用の「LLL/.net デプロイメントツール」を使用して行います。アセンブリ単位の配信や更新で無いのが特徴です。

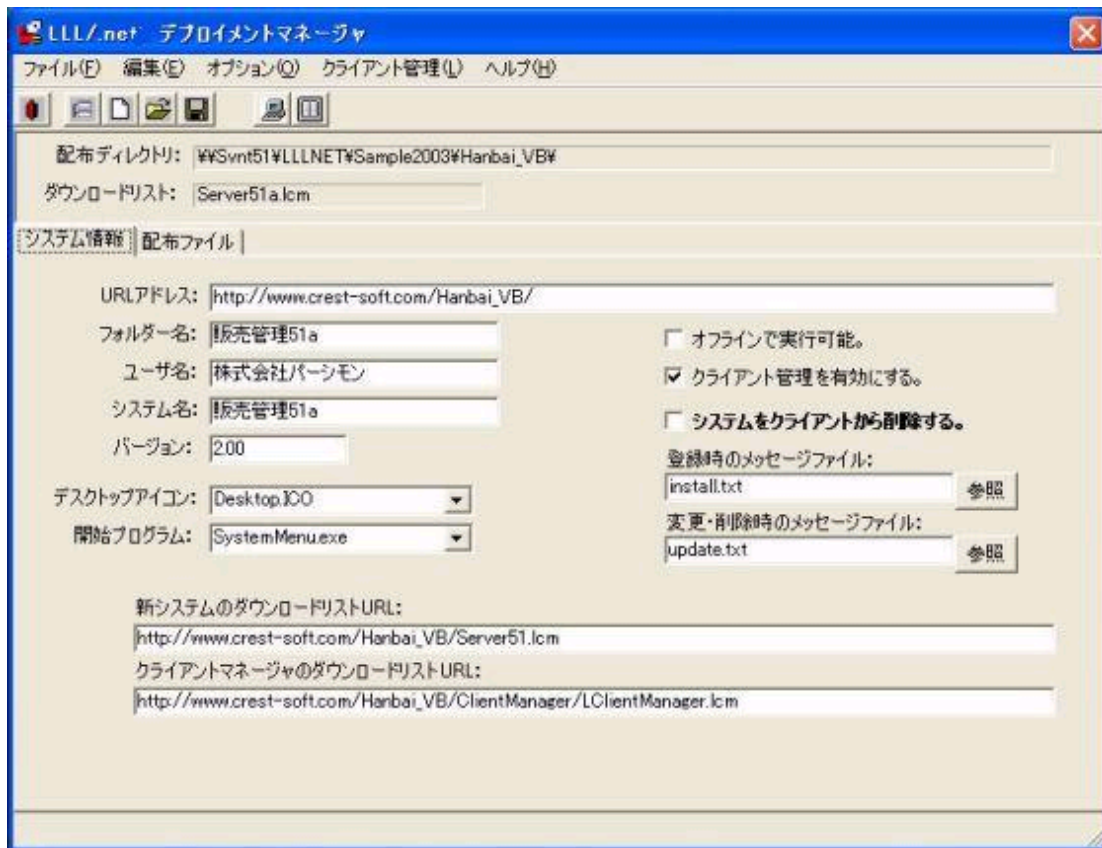
「LLL/.net プログラム設計」と連携する「デプロイメントマネージャー」で作成するダウンロードリストを元に、クライアントに予め配置された「LLL/.net クライアントマネージャ」と呼ぶ専用のダウンローダー兼ランチャーが、指定された Web サイトから必要なプログラムやデータをファイル単位でダウンロードしてローカルディスク上にコピー配置した上で、指定プログラムを起動するという方法を採用します。2 回目以降の起動時には自動的にサーバー上の公開ファイルと、自身の持つローカルファイルを比較し、差分を検知したファイルのみ再度ダウンロードし、常にサーバーと自身を同一環境にしてから実行します。これは、マイクロソフトが提唱する「副作用なしのインストール」「XCOPY によるデプロイメント」を実装した仕組みです。



前述の通り、配信の単位はアセンブリでなくファイルです。

配信対象は、必ずしもマネージドコードやプログラムでさえある必要はありません。例えば、画像など単なるデータファイルや、セットアップは別問題として考慮する必要がありますが、Win32 アプリなどアンマネージドコードもその対象に含めることができます。

ACCESS の MDB ファイル、EXCEL シート、ワード文章なども自動的に配信や更新を行います。



ダウンロードリストによる配布済みクライアントを一覧表示したり、選択したクライアントの「クライアント情報」を表示するクライアント管理機能もあります。

システム情報		値		
ホスト名	mercury			
IPアドレス	192.168.10.21			
OSバージョン	Microsoft Windows NT 5.1.2600 Service Pack 2			
.NET Framework	ver2.0.50727.42 (RTM.050727-4200)			
帳票ランタイム	CoReports Ver.8 (8, 5, 1, 8)			
クライアントマネージャ	ver2.00			
ダウンロードリストURL	http://www.crest-soft.com/Hanbai_VB/Server51.lcm			
インストールディレクトリ	C:\Program Files\LLLnet ClientManager\System%販売管理51%			
システム名	販売管理51			
バージョン	2.00			
最終ログインユーザー	incue			
最終ログイン日時	2005/11/26 10:21:58			

販売管理51	ファイル名	サイズ	更新日時	バージョン
	Client.lcm	2486	2005/11/26 10:21:58	
	Control.DEF	1405	2005/11/26 10:04:18	
	Desktop.ICO	1078	2000/11/21 10:34:31	
	HAD010.exe	73728	2005/04/19 19:10:45	1.0.1936.34477
	HAM020.exe	73728	2004/02/29 11:34:40	1.0.1520.20815
	HAM040.exe	73728	2004/02/29 11:35:25	1.0.1520.20859
	HAM050.exe	36864	2004/02/29 11:35:59	1.0.1520.20877
	HAM060.exe	65536	2004/02/29 11:36:28	1.0.1520.20892
	HAM100.exe	49152	2005/08/26 14:33:36	1.0.2064.26197
	HAT010.exe	25600	2005/04/20 11:44:04	1.0.1936.21081
	HAT100.exe	23040	2004/02/29 11:38:28	1.0.1520.20952
	HAT110.exe	18432	2005/04/20 11:46:00	1.0.1936.21144
	HAT120.exe	16896	2004/02/29 11:39:21	1.0.1520.20977

[捕捉情報、その他]

1. フォーム制御オブジェクトと Windows フォーム

LLL/.net アプリケーションでは、直接 Windows フォーム上のコントロールのプロパティを制御するのではなく、フォーム制御オブジェクトに SET されたフィールドを制御するコーディングで画面を操作します。

このことにより、ロジックと、フォームが切り離され画面デザイン変更によるコードへの影響を小さく抑えることが可能です。またコントロールがフォーム上に実在するフィールドと、そうではないワークフィールドを同列で操作できることから、コーディングがシンプルになり見やすくなるメリットもあります。

2. 自作及び 3rd パーティ製コントロールの使用

3rd パーティ製や自作のコントロールは、統合環境「LLL/.net プログラム設計」のフォームデザイナーによりドラッグ&ドロップ配置することはできません。しかし LLL/.net の成果物は VisualStudio のプロジェクトファイルです。したがって一旦 LLL/.net で作成したプロジェクトに対し、VisualStudio でフォームにコントロールを配置し、それに対するコーディングを通常通りに行なえば、どのようなコントロールでも組み合わせて使用できます。

「LLL/.net プログラム設計」のフォームデザイナーは、デフォルトでは保護のため VisualStudio で 3rd パーティ製コントロールを配置したフォームの編集を禁止します。ただし Program.DEF に登録すると 3rd パーティ製コントロールを含むフォームをデザイン表示できるようになります。また Control.DEF ファイルに、フィールド制御するコントロールタイプ名と制御パラメータを登録すると、明細エリアでは使用できない他、いくつかの制限はありますが、LLL/.net のフォーム制御オブジェクトの管理対象にできフォーカス制御対象に加えたり、フォーム制御オブジェクトのメソッド(LForm.XXXX)を使ったアクセスが可能になります。制約はありますので注意が必要です。

3. チーム開発とソース変更履歴管理

アプリケーションの信頼性向上のために、ソースコード保護や改変履歴の管理が重要とされています。LLL/.net は独自にこれらの機能の提供は行ないませんが市販のツールと組み合わせて使用できます。

特に使用できるツールに制限がある訳ではありませんが、一般的には VisualStudio との相性からも VisualSourceSafe の使用を考えられるケースが多いようです。

LLL/.net では開発資源を、独自の開発ディレクトリという単位で管理しています。

VisualSouceSafe を使用した開発においては、まず VisualSouceSafe にログインし、作業に必要な資源をチェックアウトし、作業終了時にチェックインするという手順を踏んでいただければ特に問題ありません。

運用との兼ね合いにもなりますが、LLL/.net では開発ディレクトリへのログイン時に標準コンポーネントを開発ディレクトリに自動的にコピーします。これは開発環境側でのバージョンアップなどに自動対応するためです。

しかし、コンポーネントを VisualSouceSafe の管理対象に加える場合、これをチェックアウトしなければコピーできませんが、チェックアウトしたままだと LLL/.net を利用する全プログラムが共通で使用するコンポーネントであるため他の開発者の作業に支障を与える可能性があります。こういう運用上の注意は発生します。

4. 使用できるデータベース

LLL/.net は、Oracle、SQL-Server、Jet の 3 種類のデータベースが使用できます。

使用するデータプロバイダーは、それぞれ Oracle = OracleCient (.NET Framework Data Provider for Oracle) または OleDb (MSDAORA)、SQL-Server = SqlClient (.NET Framework Data Provider for SQL Server)、Jet = OleDb (Microsoft.Jet.OLEDB.4.0) です。

LLL/.net 統合環境に、データベースへの接続情報をセットすることにより、接続は LLL/.net が自動的に行ないます。LLL/.net データベース制御コンポーネントが、データプロバイダーへのアクセスを隠蔽しますので、LLL/.net アプリケーションは、コーディング上使用するデータベースの種類を意識することはありません。

データベースの種類や接続方法に変更があった場合、LLL/.net 統合環境の「データベース接続情報」の登録変更が必要です。

データベース接続情報は、コードではなく、Runtime.Def という暗号化ファイルにデータとして保存されています。接続情報をコードから分離することによりプログラムとデータベースの独立性が保たれ、この内容に変更があっても必ずしもコードには影響を与えないというメリットがあります。

プログラム作成後にテーブル構造の変更があった場合はどうなるでしょうか。

テーブルカラムの桁数や他の属性に変化があっても、既にテーブルカラム情報をドラッグアンドドロップして作成されたフィールドとフォーム上のコントロールは自動的に属性変更されることはありません。プログラムへの影響度はケースによって異なりますが、完全に対応させるには、手作業で修正するか一旦削除して再配置する必要があります。

テーブルカラムが、削除されたり、追加された場合も同様です。

5. LLL/.net の Web サービス機能

LLL/.net は、インターネット越しの通信によるデータベースアクセスを実現する仕組みとして Web サービス (XML Web サービス) を使用しています。本来 Web サービスは、モジュール化したネットワーク上のアプリケーションの機能やデータをオンデマンドに組み合わせ、ダイナミックに結合したシステムを構築することも可能にするための標準技術です。これは、XML (Extensible Markup Language) 、SOAP (Simple Object Access Protocol) : XML Protocol、WSDL (Web Services Description Language)、UDDI (Universal Description, Discovery and Integration)といった Web の標準テクノロジーを基盤として実現されます。

しかし、LLL/.net ではあくまでも LLL/.net アプリケーション自身が Web 越しで DB アクセス機能などのサーバーコンポーネント機能を Web サービスとして実装しているのであって、広く世間に公開する Web サービスアプリケーションを作成する機能を提供している訳ではありません。

したがって LLL/.net アプリケーションの開発者が、前述の基盤テクノロジーを熟知する必要はまったくありません。LLL/.net アプリケーションが必要とする Web サービスを提供する機能は LWebService.dll が、またこれを利用するクライアント側の機能も LComWin.DLL や、LComDb.dll といったコンポーネントが既に実装済みだからです。

Webサービス接続の登録

登録 取消 ヘルプ

WebサービスのURLアドレス:
http://(URLアドレス)/(ディレクトリ)/

Webサービスの認証情報を設定する。

ユーザー名:
パスワード:
ドメイン:

接続テスト

Web サービスの接続認証には、IIS の基本認証と統合 Windows 認証が使えます。

6. プログラムトレース機能

プログラムをトレースモードに設定すると、データベースやWebサービスのアクセスログなどコンポーネント内部の動作を含むプログラムのトレースファイルに出力可能です。プログラムの動作確認やデバッグなどの解析に使用することができます。

[おわりに]

LLL/.net に関するさらなる詳細、コンポーネントのメソッドの説明やテンプレートの具体的なコーディングを含むプログラム作成手順をお知りになりたい場合は、「評価版 CD-ROM」と「LLL/.net を開発環境評価手引書」をご請求ください。無料でご提供いたします。

不明点や疑問点がございましたら、お気軽にお問い合わせください。

2010年12月6日 改訂

[お問い合わせ]



株式会社パーシモンシステム

<http://www.ascendia.jp/persimmon/>
persimmon@ascendia.jp

TEL 06-6204-0618

大阪市中央区今橋一丁目 6 番 19 号