



[プライマネージ]

PRIMANAGE

CSS Open Package Series for Enterprise Resource Planning

ソフトウェア部品配布システム

PRIMANAGE MEISTER を使用して開発を行う場合、複数のエンドユーザー向けのソフトウェア部品システム^{【注1】}を同時に開発する必要が生じることがあります。

【注1】 しかも、エンドユーザーの選択する DB は、Oracle であつたり、MS SQLServer であつたりします。

この場合、どのように開発を進めるたらよいかについて、弊社が最適であると考え、また推奨する開発システムの形態を説明いたします。

1. PRIMANAGE MEISTER を軸とするシステム構成

まず、開発システムの中心は、PRIMANAGE MEISTER です。

すべてのソフトウェア部品を、PRIMANAGE MEISTER の上で開発します。

PRIMANAGE MEISTER は、DB として Oracle を採用しています。従って全てのソフトウェア部品を、Oracle をターゲット DB として開発します。

開発は、PRIMANAGE MEISTER の上で、正しく動作することの確認を取るところまで(つまり、DB を Oracle として、Oracle に対しては、完全に正しく動作することを確認するまで)行います。

(1) エンドユーザー固有の機能

このとき、複数のエンドユーザー向けのソフトウェア部品システムを同時に開発していると、各社に固有の機能を実現する必要になることが多々あるはずです。

開発側では、このエンドユーザー固有の機能を、それぞれがユニークな部品 ID を持つ 新しい部品(あるいは部品群)を作成することによって実現します。

これらの新規部品(群)は、既存のソフトウェア部品群と同様の、一般的な部品という位置づけで、開発システム(=PRIMANAGE MEISTER)の中に、既存のソフトウェア部品群と混在している状態になります。

(2) エンドユーザー固有のデータ

一方、DB についてみると、エンドユーザー A 社の DB と別のエンドユーザー B 社の DB では、通常は、それぞれが独自のテーブルや項目を持つはずで、従って、DB の構成は同じではなくなるはずで、

そこで、エンドユーザー各社(A 社、B 社、...) には、それぞれ別々のサブシステム ID^[注 2] を割り当てて、サブシステムを分離し、そのサブシステム内でエンドユーザー各社のリクエストに対応した、独自のテーブルを作成するようにします。

この結果、DB アクセス関数と、DB サーバー(DxxxSvr.exe)も分離され、DB のテーブルを作成するスクリプトも分離されます。

^[注 2] この、エンドユーザー毎に新しいサブシステムを作る、という方式は、早晚サブシステム ID を使い尽くしてしまう可能性があります。弊社では、(エンドユーザー毎に)DB を分離するという機能を、“メタ・ファイル”と“メタ・メタ・ファイル”を使用する方式によって、DB 内でのエンドユーザーのテーブルの分離を実現します。

この方式は、既存のシステムとコンパチブルな方式ですので、この方式

が実現するまでの過渡状態では、サブシステムを分ける形で、DB の分離を行います。なお、この新しい方式については、仕様が確定した適当な時点で、お知らせさせていただきます。

つまり、(複数のエンドユーザー向けの、それぞれ別々の)ソフトウェア部品システムが開発されている状態の開発環境(=PRIMANAGE MEISTER)を見ると、

- PRIMANAGE MEISTER(開発環境)には、複数のエンドユーザー向けの部品群が、既存の部品群と混在しており、
- DB は 1 つだけあって(=Oracle)、その中に複数のエンドユーザーそれぞれのサブシステム単位での領域の確保が行われ、その領域の中にそれぞれのエンドユーザー向けの DB テーブルが構成されている、という形になります。(各エンドユーザーのテーブルに対応した DB アクセス関数も、既存の DB アクセス関数と混在しています。)

開発環境(=PRIMANAGE MEISTER)は、この状態を保っていますので、既存の部品はもちろん、どのエンドユーザー向けの部品も、同時に動かすことができますから、テストも同時に行うことができます。^[注3]

^[注3] テストを行うと、DB の共通のテーブル(例えば、担当者テーブル)に、ユーザーA 社向けのデータとユーザーB 社向けのデータが混在することはありますが、ユーザーA 社の動作がユーザーB 社あるいは既存のソフトウェア部品の動作に干渉することはありません。

開発環境がこのような状態であるときは、特定のエンドユーザー向けのソフトウェア部品システムを開発環境から分離して取り出すことになります。

品質管理の面から見ても、あるいは、出荷後のメンテナンスの面から見ても、出荷を行うソフトウェア部品システムだけで構成されたシステムを作り、その上で(出荷用の)ソフトウェア部品システムの動作確認を行って、必要な記録を残した後、そのシステムを出荷する体勢をとることが大切であると考えます。この、開発環境と(出荷システムが搭載された)ターゲット・システムの間に置く、動作確認用のシステムを“準備システム”と呼びます。

2. “準備システム” の設置

前項で開発が完了した時点で、エンドユーザー向けに必要な部品やソフトウェア・モジュール(例えば、DB アクセス関数や、DB テーブル作成のスクリプト)を分離します。

このとき、分離されたモジュールを受け取るシステムが必要になります。これを、“(出荷)準備システム”と呼びます。この準備システムの目的は、出荷する前の動作確認を行うことです。エンドユーザーのシステム(ターゲット・システム)にインストールするのと、全く同じ状態にして、動作確認を行います。

(必要なモジュールが落ちていないことを確認し、不要なモジュールが紛れ込んでいる場合はここで除去します。また、この段階では、DB に任意のテストデータを設定することができます。)

従って、この準備システムには、エンドユーザーで採用する DB と同じ DB がインストールされていることが必要になります。

エンドユーザーが、DB として、PRIMANAGE MEISTER と同じ(バージョンも含めて)DB を採用している場合は、この準備システムは、エンドユーザーのシステムに、ソフトウェア部品システムをインストールするための単なる中継のシステムとなります。この場合は、準備システムとしてエンドユーザーのマシン(ターゲット・マシン)そのものを使うこともできます。

ユーザーの採用している DB が Oracle ではない場合は、この準備システム上で、PRIMANAGE MEISTER に付属している変換(コンバージョン)ツールを用いて、ソース・ファイル(DB アクセス関数)の再コンパイルを行います。変換前のソース・ファイルは、既に、(Oracle に対しては、)正しく動作することがわかっていますので、基本的に^[注 4]、再コンパイルを行うだけでそのまま(エンドユーザーの採用した)DB をターゲットとして動作するはずですが、(ただしその前に、変換後の DB テーブルを作成するスクリプトによって、DB の領域とテーブルを作成することは必要です。)^[注 5]

^[注 4] 開発のときに、DB アクセス関数に独自のコーディングを行われた場合は、コンバージョン・ツールの修正が必要になる可能性がありますので、タイムテーブルを設定される場合は、ご注意ください。

^[注 5] この方式は、弊社で実際に行っている方式です。

コンパイルが必要なため、このケースの場合は、準備システム上に、開発環境(Cのコンパイル実行環境と、DBの埋め込み型SQL文をCのソースファイルに変換するプリプロセッサ)が必要になります。

3. 出荷

準備システムで、動作確認を終えたならば、ソフトウェア部品のシステムを、そのままエンドユーザーの(ターゲット)システムに移して、出荷することができます。

準備システムは、出荷するエンドユーザー向けのソフトウェア部品システムのバックアップをとり、ターゲットシステムに、(出荷すべき)ソフトウェア部品システムを移した時点で、(このエンドユーザー向けの)準備システムとしての役割を終えます。

ここより、この準備システム(というより、コンピュータ・システムは)、次の出荷準備システムとして使われることになります。^[注6]

^[注6] つまり、(出荷)準備システムは、次々と使いまわされることとなります。

4. PRIMANAGE MEISTER から準備システムへのモジュールの配布

開発が完了した時点で、開発環境から、エンドユーザー向けに必要な部品(クライアント側実行モジュール、サーバー側実行モジュール、DBアクセス関数の実行形式)や、ソフトウェア・モジュール(例えば、DBテーブル作成のスクリプト)を取り出す必要がありますが、これは、手作業で行うと落ちが生じる可能性があります。

弊社では、この取り出し作業を自動化するシステムを作成します。(名称: ソフトウェア部品配布システム)

各開発担当者の方が、このシステムによって、開発環境から、特定のエンドユーザー向けのソフトウェア・モジュールを、準備システムに取り出すお手伝いを致します。