Technical Note 05-32
サーバプロセス

By Larry Sharpe, Infoservices
Technical Note 05-32

（原題: Server Processes）

概要

セットアップと管理は 4D Client で制御され、実際には 4D Server のプロセスで実行される処理について解説します。別の 4D Client のプロセスで同じことを実行する方法についても取り上げます。

サンプルデータベースでは、サーバで起動したプロセスが一定の間隔で Import Needed フォルダの中を調べます。ファイルが見つかると、インポートコードが実行され、インポート完了の Email または警告メッセージを表示し、ファイルは Import Finished フォルダに移されます。今回の焦点はサーバプロセスの動作であり、インポート関連のコードについては詳述しません。サーバプロセスの原理は、インポートに限らず幅広い応用が可能ですが、サーバでインタフェースアイテムを使用することはできない点には留意が必要です。4D Server のプロセスで、警告、ダイアログ、入力フォームなどを表示すると、クライアントプロセスを含むすべてのプロセスが停止することになりかねません。

サンプルデータベース

Designer、Administrator、Test User というユーザがおり、ログインダイアログを表示するために designer というパスワードが Designer に設定されています。

[People]、[Companies]、[xPreferences]テーブルの構造は、以前の Tech Note から変わっていません。前回の Tech Note を執筆していた時点では、Server Preference のページ 1 を改造することを考えていましたが、それよりもサーバのプリファレンスをユーザプリファレンスから切り離し、新しいメソッド、フォーム、プロセスで管理したほうがずっと良いので、そのようにしました。

以前の Tech Note ではひとつだったプリファレンスのメソッドは、3 つに分かれています。編集メニューから呼び出される xPreferences_Users は、ユーザのプリファレンスを設定するためのもの、xPreferences はデータベースにあるすべてのプリファレンスを設定するためのもの、xPreferences_Processes は新しいフォーム[xPreferences]Prefs_Processes を開いてプロセスのプリファレンスを更新するためのものです。

サーバプロセスは、ユーザごとに実行されるわけではないので、ユーザプリファレンスのようにユーザ名を使用して管理するのではなく、Processes という名前で 1 件だけプリフ

ァレンスを保存します。

xFileNameHandler メソッドは、4D ドキュメントに掲載されているコードによく似ており、4D アプリケーションまたはストラクチャのファイル名やパスを返します。今回の一番のポイントは People_ImportProcess メソッドの中に凝縮されています。

[xPreferences]Pref_Processes フォーム

People Import プロセスに関係のあるオプションを編集することができるフォームです。スタートアップで起動するか、クライアント/サーバの場合はどこで実行するか、実行の間隔、通知 email の設定、完了後に警告ダイアログを表示するか、とった項目が用意されています。直ちにプロセスを実行するためのボタンもあります。プロセスが実行中の場合、すべての変数は入力不可になります。4D Server で実行されているプロセスはすべての 4D Client から停止できますが、4D Client で実行されているプロセスはその 4D Client だけが停止できます。プロセスが開始または停止すると、その情報はすべての 4D Client に知らされます。

[xPreferences]Pref_Processes フォームメソッド

On Load
表示する変数の値を取得し、People Import プロセスの状態に基づいて入力可あるいは不可にします。

On Outside Call
People_ImportProcess メソッドの UpdateWindow 部分が実行されると、プリファレンスの値を再度読み込まれ、表示が更新されます。このコードはクライアント/サーバでのみ有効です。

On Unload
People Import プロセスが実行中でなければ、変更を保存します。

People_ImportProcess プロジェクトメソッド

関連のある処理を別々のメソッドに分ける手法もありますが、すべてを大きな Case of ステートメントにまとめるほうが個人的には性に合っているのでそのようにしています。

¥ (Count parameters=0)

パラメータなしでコールされた場合、People_ImportProcess メソッド(自分)に引数として「RunProcess」を渡すことによって新しいプロセス(People Import)を起動します。連続してコールされた場合、前のプロセスが終了していなければ、新しいプロセスは起動しません。

新規プロセスは Execute on server 関数によってサーバで起動します。4th Dimension の場合、あるいは 4D Client で cbPrefPeopleImportRunOnServer が false の場合は、ローカルで新規プロセスが起動します。スタンドアロン環境では、Execute on server 関数と

New process 関数は同じ動作をしますが、クライアント/サーバ環境では、Execute on server がサーバのプロセス、New process がクライアントのプロセスを起動します。

```
¥ ($command="RunProcess")
```

自らをコールすることによって新規プロセスで起動するルーチンです。PeopleImport の値を読み込んだ後、ループを開始してインポートの確認活動を始めます。この部分では、警告やダイアログなど、4D Server で許されないコマンドを使用することはできません。随所に次のようなコードがあります。これにより、メソッドのウィンドウ更新部分が、サーバではなくクライアントで実行されます。

```
EXECUTE ON CLIENT(sPrefPeopleImportClientMsgName;"People_ImportProcess";"UpdateWindow")
```

While ループに先立って Repeat ループが置かれています。単純に DELAY PROCESS で待機した場合、途中で停止命令が入っても直ちには反映することができません。それで、実際には 1 秒置きに DELAY PROCESS を実行し、◊cbPrefPeopleImportOn を確認し、設定された時間だけ繰り返した後に、インポート作業の有無を調べるようにしています。

インポートフォルダの場所は、アプリケーションの種類によって異なります。PLATFORM PROPERTIES は、4D Client で実行された場合、ストラクチャの名前だけを返すので、その場合はアプリケーションの場所を使用してフォルダの場所を決定します。4D Server あるいは 4th Dimension であれば、ストラクチャの場所が返されるので、それに基づいてフォルダの場所を決定します。

```
¥ ($command="UpdateWindow")
```

接続中のすべてのクライアントで、CALL PROCESS が実行され、xPreferences_Processes メソッドが実行されます。ウィンドウが表示されていなければ何も起きませんが、表示されていれば On Outside Call フォームイベントによって画面が更新されます。

```
¥ ($command="ShowAlert")
```

警告ダイアログも、クライアントのプロセスで表示する必要があります。4th Dimension で実行した場合、このコードではプロセスがクリック待ちになります。その場合は別プロセスで実行したほうが無難です。

```
¥ ($command="UserInterface")
```

クライアントでのみ実行されるルーチンで、インタフェースのオプションを状況に合わせて設定します。例えば 4th Dimension では Run On Server チェックボックスが入力不可になります。

```
¥ ($command="OnStartup")
```

cbPrefPeopleImportRunAtStartup の値を調べて、スタートアップで People Import プロセスを実行すべきかを判断します。

4D コマンドに関する注記

New process

クライアントおよびスタンドアロンで同一の動作をし、実行中のマシンで新規プロセスを起動します。

Execute on server

クライアントで実行された場合は、サーバで新規プロセスを起動し、スタンドアロンでは New process と同じ動作をします。

EXECUTE ON CLIENT

プロセスではなく、メソッドをクライアントで実行します。クライアントは、環境設定の「接続時にクライアント登録」または REGISTER CLIENT で設定します。メソッドはあまり時間とメモリを消費しないようなものにするべきです。スタンドアロンで実行された場合、何も起こらず、エラーも発生しません。

REGISTER CLIENT、UNREGISTER CLIENT

環境設定による登録とは違い、各クライアントに名前を付けることができます。

GET PROCESS VARIABLE

渡されたサーバの変数を読み取ります。ローカル変数は読み取れません。

VARIABLE TO VARIABLE

渡されたサーバの変数に書き込みます。元の変数にローカル変数には書き込めません。

SET PROCESS VARIABLE

サーバの変数に書き込みますが、制限があります。配列、ポインタ、ローカル変数には書き込めません。

GET PROCESS VARIABLES の反対は VARIABLE TO VARIABLE です。SET PROCESS VARIABLE ではありません。

まとめ

この Tech Note は、05-25、05-28 の続編です。クライアント/サーバプロセス間でデータをやり取りし、変数の更新や画面の再描画をする方法、スタンドアロン・クライアント/サーバの両方で動作するコーディングの方法、不適当な変数を入力不可にするインタフェース、ユーザ別のプリファレンスに食わせてプロセス別のプリファレンスを管理する方法を取り上げました。