Backup Settings for Distributed Applications

By Jean-Yves Fock-Hoon, QA Manager, 4D Inc.

TN 06-13

Overview

バージョン 2004 で開発したアプリケーションを配布する場合、一緒にバックアップ設定も配布するのが理想的です。

このテクニカルノートでは、特にアプリケーションを配布する場合を想定し、カスタマイズされた Backup.XML ファイル(バックアッププロジェクトファイルとも呼ばれます)を作成する方法を説明しています。

Building an Application

The Problem

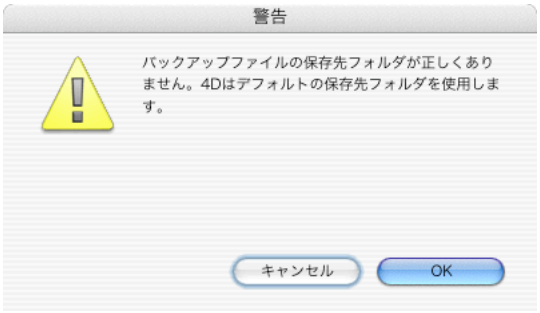
アプリケーションを開発している段階では、普通、主要な機能の動作確認に注意が集中しているものです。アプリケーションビルド(4D Runtime または 4D Server の組み込み)や、バックアップなどの基本メカニズムは、大抵後回しにされ、何かと見過ごされがちです。このテクニカルノートでは、あまり注意を受けない配布アプリケーションのバックアップ設定というテーマに焦点をあてています。

典型的なシナリオを考慮してみましょう：

データベースの開発が完了しました。デベロッパはアプリケーションの動作を確認し、バックアップの動作も確認しました。すべて正常に動いているようです。ちなみにバックアップ機能をテストした結果、Backup.XML ファイルが生成されました。

仕上げにデベロッパはビルドされたクライアント/サーバアプリケーションとしてデータベースを配布する準備にとりかかります。しかしながらビルドの結果、すべての主要ファイルが Server フォルダにコピーされるわけではありません。Preferences フォルダの中に残っているファイルもあります。そこでデベロッパは、アプリケーションビルドのプロジェクトファイルなど、配布先では不要な XML ファイルは除外しつつ、Preferences フォルダの中にあるファイルも Server フォルダにコピーします。

デベロッパのマシンでは、引き続き問題が起こらないので、次に配布先のマシンにアプリケーションをコピーして動作を確認します。ここで最初の問題に遭遇することになります。



バックアップの保存先が見つからないため、デフォルトの保存先を使用する(キャンセル)、または新しいパスを指定するように 4D が求めてきました。一体、何がいけないのでしょうか。

The 4D Backup Project File

この問題は、Backup.XML プロジェクトファイルに原因があります。このファイルには、バックアップファイルの保存先フォルダのパスが記録されているのです。データベースが新しいマシンに移動された結果、そのパスは無効なものになってしまいました。標準動作、あるいはコマンドによるバックアップは、起動直後に実行される可能性もあるため、4D は起動前に有効なバックアップ設定を求めます。

このダイアログの表示を避けるにはどうしたら良いのでしょうか。もっとも単純な解決策として、データベースを移動する際には Backup.XML ファイルをコピーしない、という方法が挙げられます。Backup.XML ファイルが見つかれなければ、4D はデフォルトの値で新規プロジェクトファイルを作成した上でバックアップを実行するからです。

そのような場合、デフォルトのバックアッププロジェクトファイルはストラクチャファイルと同じ階層の Preferences:Backup フォルダに作成される点に注目してください。これはビルドされたアプリケーション、ビルドされたサーバアプリケーションの場合でも同様です。このように Backup.XML ファイルがなくてもバックアップは実行されます。保存場所についても特に問題がないように思えるかもしれません。

しかし、デフォルトのバックアップ設定では都合が悪い場合もあります。デベロッパは保存するアーカイブの数、バックアップに含めるファイルやフォルダ、バックアップ時間を短縮するための特殊な設定、あるいは特別なスケジュールを念頭に置いているかもしれません。そのような場合、Backup.XML ファイルの使用は必須です。ここでジレンマが生じます。カスタム設定をすれば変なダイアログが表示され、ダイアログを回避すればカスタムバックアップ設定が失われてしまうのです。

A Different Approach

Building a Backup.XML File

より優れたソリューションは、最初の起動で動的にカスタム Backup.XML ファイルを作成し、それをバックアップ設定として使用するというアプローチです。4D には XML を解析し、構築するためのコマンド群が用意されているので、使用するタグさえ分かっていれば、Backup.XML ファイルは容易に編集することができます。実際、Backup.XML ファイルの構造は比較的単純です。

設定ファイルの作成にユーザを介入させない方針をとるのであれば、On Startup、On Server Startup などのデータベースメソッドでファイルを作成することができます。起動時にバックアップ設定ファイルの存在を確認し、ファイルがなければカスタム Backup.XML ファイルを作成するようにすれば、自動的にバックアップの設定をすることができます。

設定の一部をユーザインタフェースで実行できるようにするのであれば、設定を更新するためのダイアログを用意し、デベロッパが定めたハードコーディングの値とユーザが入力した値に基づき、カスタム Backup.XML ファイルを再生成することができます。

生成された Backup.XML ファイルに問題があってはなりません。パラメータの値が無効、または余計な空白が混入している項目については、デフォルトの値でバックアップが実行されます。

バックアッププロジェクトファイルを解析し、更新するサンプルデータベースは、すでに公開されていますが、タグや名前空間を挿入し、XML を構築する部分はハードコーディングされているため、後に XML の構造に変更を加える必要が生じた場合、メソッドを修正しなくてはならないというデメリットを抱えています。

Another Concept

別のアイデアとしては、テンプレートとなるバックアップ設定を外部ファイルとして管理するという方法が考えれます。そのようにしておけば、将来 XML の構造に変更が生じた場合、テンプレートを修正するだけで対応することができます。

どのようにすれば、テンプレートが利用できるようになるのでしょうか。ポイントは、変動する箇所をすべて 4D HTML タグで記述された Backup.XML ファイルを用意するという点にあります。使用する変数は、4D メソッドの中であらかじめ定義しておきます。PROCESS HTML TAGS コマンドを実行すれば、テンプレートが有効な XML ファイルに変換され、Preferences フォルダに書き出されます。XML コマンドはまったく必要ありません。

このテクニカルノートのサンプルデータベースには、テンプレート XML ファイルが付属しています。ファイル名は Backup.XML2 で、Preferences:Backup フォルダに保存されています。

使用した主なテクニックは次のとおりです：

- ビルドされたサーバアプリケーションの Server フォルダには Preferences:Backup フォルダがコピーされ、このとき Backup.XML ではなく、テンプレートである Backup.XML2 がコピーされるものと想定しています。
- データベースを起動すると、On Server Startup データベースメソッドで Backup.XML が探されます。
- Preferences フォルダの場所を割り出すためには、Get 4D folder で Extras フォルダの場所を調べ、そこから Preferences の場所を計算しています。Preferences フォルダの場所は、ストラクチャファイルの場所からも計算できますが、Extras フォルダを起点としたほうが簡単です。
- Preferences フォルダの場所が確定したならば、Test path name で Backup.XML の存在を調べます。
- Backup.XML が存在するならば、それはバックアップの設定が完了していることを意味するので、そのままアプリケーションを実行します。
- Backup.XML が存在しない場合、それはアプリケーションが初めて起動されたことを意味するので、Backup.XML ファイルを作成することになります。これには幾つかの方法が考えられます：
    - デベロッパは、この動作がユーザの目に触れないようにすることができます。Backup.XML ファイルは、既定の設定値に従って作成されます。
    - バックアップファイルの保存先だけはユーザが変更できるようにするのであれば、そのようなダイアログを用意することができます。
    - ある程度はユーザがバックアップの設定を変更できるようにすることもできます。サンプルデータベースには、そのようなバックアップの「詳細設定(Advanced)」ダイアログが付いています。

サンプルデータベースには、バックアップの保存先フォルダを指定するための簡単なダイアログが存在し、そのダイアログには詳細設定(Advanced)を変更するためのボタンが付いています。ボタンを使用すると、環境設定のバックアップテーマで設定することのできるほとんどの項目にアクセスすることができます。変更を確定すると、PROCESS HTML TAGS コマンドが実行され、Backup.XML2 テンプレートに基づいて生成されたファイルが Backup.XML の名前で Preferences フォルダに書き出され、以後、バックアッププロジェクトファイルとして使用されます。

変数名	タグ名
BKP_rb_AlwaysWaitBKP	<WaitForEndOfTransaction>
BKP_NbMinWait	<Timeout>
BKP_rb_RetryNextTime	<TryBackupAtTheNextScheduledDate>
BKP_at_timeretry	<TryToBackupAfter>
BKP_cbCancelRetryBKP	<AbortIfBackupFail>
BKP_NbTries	<RetryCountBeforeAbort>
BKP_cbRestoreLastBKP	<AutomaticRestore>
BKP_cbIntegrateLastLog	<AutomaticLogIntegration>
BKP_cbStartDbAfterRestore	<AutomaticRestart>
BKP_cbIfModified	<BackupIfDataChange>
BKP_cbKeepLastBKP	<Enable>
BKP_BackupSet	<CompressionRate>
BKP_at_CompressionRate	<Redundancy>
BKP_at_RedundancyRate	<Interlacing>
BKP_at_InterlacingRate	<DefaultSize>
BKP_at_SegmentSize	<EraseOldBackupBefore>
BKP_at_DelOldBKP	<IncludeStructureFile>
Bkp_cbStructureFile	<IncludeDataFile>
Bkp_cbDataFile	<IncludeAltStructFile>
Bkp_cbAltFile	<DestinationFolder>
BKP_BackupFileDest	<ItemsCount>
BKP_SA_Attatchments	<Item>
BKP_SA_Attatchments	<Frequency>
BKP_rbSched_NoBKP	<Hourly>
BKP_rbSched_Hours	
BKP_rbSched_Days	
BKP_rbSched_Weeks	
BKP_rbSched_Months	
BKP_atSchedStartHours	<Hourly>
BKP_atSchedStartDay	<Daily>
BKP_SchedEveryWeek	<Weekly>
BKP_cbSchedEveryMonday	<Monday>
BKP_atSchedStartTuesday	<Tuesday>
BKP_CBSCHEDEVERYWEDNESDAY	<Wednesday>
BKP_CBSCHEDEVERYTHURSDAY	<Thursday>
BKP_CBSCHEDEVERYFRIDAY	<Friday>
BKP_CBSCHEDEVERYSATURDAY	<Saturday>
BKP_CBSCHEDEVERYSUNDAY	<Sunday>
BKP_SchedEveryMonth	<Monthly>
BKP_atSchedStartMonth	<Hour>
BKP_SchedEveryMonthDay	<Day>

Summary

このテクニカルノートでは、バックアップの保存先パスが無効であることから生じるダイアログの表示を回避する方法を紹介しました。カスタムバックアッププロジェクトファイルを構築する方法についても取り上げました。